```
/* oosgc_r1.prg                          Norm Swanson - 02/09/04          */
/* Consistent Generic ICM Prediction Test  with new boot                  */
/*                                                                        */
/* main simulation program for new recur paper with Valentina             */
/*                                                                        */
/* as oosgc2-4.prg, but revised to construct multiple tests                 */
/* including CS, CCS, DM, CM, and F                                       */

/* ORIGINAL DGPs from first version of the paper */

output file=c:\oosgc_r1.out reset; outwidth 255; output on;
format /MA1 /LD 6,2;

simtot=500;
ptot=1;
smpl=200;
Pvector=round(0.5*smpl); Pmax=maxc(Pvector);
ls1={2,4,5};
Bootnum=100;

gam=0.0; i=1;
do while i<=10;
  gam=gam|((i-1)/2)+0.5;
  i=i+1;
endo;

gamgam=zeros(1,2);
i=1;
do while i<=10;
  j=1;
  do while j<=10;
    gamgam=gamgam|((((j-1)/2)+0.5)~gam[i]);
    j=j+1;
  endo;
  i=i+1;
endo;

gam=gam[2:11,.];        /* 10x1 vector  */
gamgam=gamgam[2:101,.]; /* 100x2 matrix */

/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */
/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */
/*                                                                        */
/* PROCEDURES                                                             */
/*                                                                        */
/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */
/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */

/* data generation */

proc (2) = dgp_sim(smpl);
local si,e,ax1,ax2,ax3,a1,a2,a3,aauto,a21,a22,a23,a1z,a3z,a21z,a22z,
                        nl1,nl2,nl1a,nl2a,xd,yd,et,etlast;

e=zeros(smpl+100,2); e=rndn(smpl+100,2);

e[.,1]=sqrt(1)*e[.,1];
e[.,2]=sqrt(1)*e[.,2];

/* basic parameters for quadratic loss - data is yd */
ax1=1.0; ax2=1.0; ax3=1.0; a1=1.0; a2=1.0; a3=1.0; aauto=0.3; a21= 0.3; a22=0.6;
a23=0.9; nl1=1.00; nl2=2.00;
```

```
/* Quadratic Loss DGPs */
/* Basic 28 DGPs includes 14 basic - 2 size and 12 power: */
/* all x2 for different a21 a22 */


xd=zeros(smpl+100,3); yd=zeros(smpl+100,42);
xd[1,.]=xd[1,.]+0.5;
yd[1,.]=yd[1,.]+0.5;
si=2; etlast=0;
do while si <= smpl+100;

   et=e[si,2]+(aauto*e[si-1,2]);

   xd[si,1]=ax1+a21*xd[si-1,1]+e[si,1];
   yd[si,1]=a1+a21*yd[si-1,1]+e[si,2];
   yd[si,2]=a1+a21*yd[si-1,2]+et;
   yd[si,3]=a1+a21*yd[si-1,3]+e[si,2]+
                              nl1*exp(atan(xd[si-1,1]/2));
   yd[si,4]=a1+a21*yd[si-1,4]+e[si,2]+
                              nl2*exp(atan(xd[si-1,1]/2));

   yd[si,5]=a1+a21*yd[si-1,5]+e[si,2]+nl1*xd[si-1,1];
   yd[si,6]=a1+a21*yd[si-1,6]+e[si,2]+nl2*xd[si-1,1];

   if xd[si-1,1]>(ax1/(1-a21));
      yd[si,7]=a1+a21*yd[si-1,7]+e[si,2]+nl1*xd[si-1,1];
   else;
      yd[si,7]=a1+a21*yd[si-1,7]+e[si,2];
   endif;
   if xd[si-1,1]>(ax1/(1-a21));
      yd[si,8]=a1+a21*yd[si-1,8]+e[si,2]+(nl2*xd[si-1,1]);
   else;
      yd[si,8]=a1+a21*yd[si-1,8]+e[si,2];
   endif;
   yd[si,9]=a1+a21*yd[si-1,9]+et+
                              nl1*exp(atan(xd[si-1,1]/2));
   yd[si,10]=a1+a21*yd[si-1,10]+et+
                              nl2*exp(atan(xd[si-1,1]/2));

   yd[si,11]=a1+a21*yd[si-1,11]+et+nl1*xd[si-1,1];
   yd[si,12]=a1+a21*yd[si-1,12]+et+nl2*xd[si-1,1];

   if xd[si-1,1]>(ax1/(1-a21));
      yd[si,13]=a1+a21*yd[si-1,13]+et+nl1*xd[si-1,1];
   else;
      yd[si,13]=a1+a21*yd[si-1,13]+et;
   endif;
   if xd[si-1,1]>(ax1/(1-a21));
      yd[si,14]=a1+a21*yd[si-1,14]+et+(nl2*xd[si-1,1]);
   else;
      yd[si,14]=a1+a21*yd[si-1,14]+et;
   endif;

   /* change from using a1 to a2 and a21 to using a22 */


   xd[si,2]=ax2+a22*xd[si-1,2]+e[si,1];
   yd[si,15]=a2+a22*yd[si-1,15]+e[si,2];
   yd[si,16]=a2+a22*yd[si-1,16]+et;
   yd[si,17]=a2+a22*yd[si-1,17]+e[si,2]+
                              nl1*exp(atan(xd[si-1,2]/2));
```

```
yd[si,18]=a2+a22*yd[si-1,18]+e[si,2]+
                            nl2*exp(atan(xd[si-1,2]/2));


yd[si,19]=a2+a22*yd[si-1,19]+e[si,2]+nl1*xd[si-1,2];
yd[si,20]=a2+a22*yd[si-1,20]+e[si,2]+nl2*xd[si-1,2];

if xd[si-1,2]>(ax2/(1-a22));
   yd[si,21]=a2+a22*yd[si-1,21]+e[si,2]+nl1*xd[si-1,2];
else;
   yd[si,21]=a2+a22*yd[si-1,21]+e[si,2];
endif;
if xd[si-1,2]>(ax2/(1-a22));
   yd[si,22]=a2+a22*yd[si-1,22]+e[si,2]+(nl2*xd[si-1,2]);
else;
   yd[si,22]=a2+a22*yd[si-1,22]+e[si,2];
endif;
yd[si,23]=a2+a22*yd[si-1,23]+et+
                            nl1*exp(atan(xd[si-1,2]/2));
yd[si,24]=a2+a22*yd[si-1,24]+et+
                            nl2*exp(atan(xd[si-1,2]/2));

yd[si,25]=a2+a22*yd[si-1,25]+et+nl1*xd[si-1,2];
yd[si,26]=a2+a22*yd[si-1,26]+et+nl2*xd[si-1,2];

if xd[si-1,2]>(ax2/(1-a22));
   yd[si,27]=a2+a22*yd[si-1,27]+et+nl1*xd[si-1,2];
else;
   yd[si,27]=a2+a22*yd[si-1,27]+et;
endif;
if xd[si-1,2]>(ax2/(1-a22));
   yd[si,28]=a2+a22*yd[si-1,28]+et+(nl2*xd[si-1,2]);
else;
   yd[si,28]=a2+a22*yd[si-1,28]+et;
endif;

/* change from using a2 to a3 and a22 to using a23 */


xd[si,3]=ax3+a23*xd[si-1,3]+e[si,1];
yd[si,29]=a3+a23*yd[si-1,29]+e[si,2];
yd[si,30]=a3+a23*yd[si-1,30]+et;
yd[si,31]=a3+a23*yd[si-1,31]+e[si,2]+
                            nl1*exp(atan(xd[si-1,3]/2));


yd[si,32]=a3+a23*yd[si-1,32]+e[si,2]+
                            nl2*exp(atan(xd[si-1,3]/2));


yd[si,33]=a3+a23*yd[si-1,33]+e[si,2]+nl1*xd[si-1,3];
yd[si,34]=a3+a23*yd[si-1,34]+e[si,2]+nl2*xd[si-1,3];

if xd[si-1,3]>(ax3/(1-a22));
   yd[si,35]=a3+a23*yd[si-1,35]+e[si,2]+nl1*xd[si-1,3];
else;
   yd[si,35]=a3+a23*yd[si-1,35]+e[si,2];
endif;
if xd[si-1,3]>(ax3/(1-a23));
   yd[si,36]=a3+a23*yd[si-1,36]+e[si,2]+(nl2*xd[si-1,3]);
else;
   yd[si,36]=a3+a23*yd[si-1,36]+e[si,2];
endif;
```

```
   yd[si,37]=a3+a23*yd[si-1,37]+et+
                              nl1*exp(atan(xd[si-1,3]/2));
   yd[si,38]=a3+a23*yd[si-1,38]+et+
                              nl2*exp(atan(xd[si-1,3]/2));

   yd[si,39]=a3+a23*yd[si-1,39]+et+nl1*xd[si-1,3];
   yd[si,40]=a3+a23*yd[si-1,40]+et+nl2*xd[si-1,3];

   if xd[si-1,3]>(ax3/(1-a23));
      yd[si,41]=a3+a23*yd[si-1,41]+et+(nl1*xd[si-1,3]);
   else;
      yd[si,41]=a3+a23*yd[si-1,41]+et;
   endif;
   if xd[si-1,3]>(ax3/(1-a23));
      yd[si,42]=a3+a23*yd[si-1,42]+et+(nl2*xd[si-1,3]);
   else;
      yd[si,42]=a3+a23*yd[si-1,42]+et;
   endif;

   etlast=et;
   si=si+1;
endo;

retp(yd[101:smpl+100,.],xd[101:smpl+100,.]);
endp;

/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */
/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */

/* model prediction */
/* Pbig is the P for this loop */
/* yy is the data, xx is the extra variable */
/* gams is a 1x2 vector with a gamma pair, with the first element changing */
/* each time */

proc (8) = pred(yy,xx,Pbig,gamgam);
local ii,T,cnst,X,Y,N,Qu0,QmY,QmEXTRA,QsY,QsEXTRA,Qbet,adjA,adjAA,
Qf1,Qf1A,Qg,QgA,mp1,bstatad,bstatadA,fullerrA,fullerrB,
mp2,qf1B,d1,d1bar,d1sderr,dmstat,c1,c1bar,c1sderr,cmstat,Fstat,Qbetout;

Qf1A={};  Qf1B={};  QgA={};  adjAA={}; bstatadA={}; Qbetout={};

T=rows(yy);

/* Construct 1-step ahead predictions */

ii=1;
do while ii<=Pbig;

/* prediction and main statistic parameter estimation */

  N=T-Pbig-1+ii; cnst=ones(T-Pbig-1+ii-1,1);
  X=cnst~yy[1:N-1]; Y=yy[2:N];
  Qbet=inv(X'*X)*X'*Y;
  Qbetout=Qbetout~Qbet; /* output actual beta estimators, size= k x Pbig */

  /*Qbet=0.0|0.0;*/
  /*print "Qbet" Qbet'~ii;*/

  /* adjustment terms for boot parameter estimation */
  cnst=ones(T-1,1); X=cnst~yy[1:T-1]; Y=yy[2:T];
  adjA= (meanc(X.*(Y-X*Qbet)))';
  adjAA=adjAA|adjA; /* each row is a beta estimator recenting mean of k elements
```

```
long */
                    /* and there are Pbig rows in total for Pbig predictions */

  /* I. prediction errors and pieces for generically comprehensive CS statistic */

  QmY=meanc(yy[1:N]); QmEXTRA=meanc(xx[1:N]);
  QsY=stdc(yy[1:N]);  QsEXTRA=stdc(xx[1:N]);

  Qu0=yy[N+1]-(1~yy[N])*Qbet; /* 1x1 */ /* prediction error */
  Qf1=Qu0;                    /* 1x1 */ /* f', quad loss assumed for stat construct
*/
  Qf1A=Qf1A|Qf1;              /* Pbigx1 vector of prediction errors */
  Qg=exp( ((atan((yy[N]-QmY)/(2*QsY)))~(atan((xx[N]-QmEXTRA)/(2*QsEXTRA))))*gamgam'
);
                              /* 1x100, as gamgam is 100x2 */
  QgA=QgA|Qg;

  /* full sample residuals */

  if ii==Pbig;
      fullerrA=Y-X*Qbet;
  endif;

  /* adjusted statistic for use in boot statistic */
  QmY=meanc(yy[1:T-1]); QmEXTRA=meanc(xx[1:T-1]);
  QsY=stdc(yy[1:T-1]);  QsEXTRA=stdc(xx[1:T-1]);
  Qg=exp(
((atan((yy[1:T-1]-QmY)/(2*QsY)))~(atan((xx[1:T-1]-QmEXTRA)/(2*QsEXTRA))))*gamgam' );
                                /* 1x100, as gamgam is 100x2 */

  /* NOTE: There was the 2 missing in the next line in the old version of this
program!!!! */

  bstatad=(meanc( (Y-X*Qbet).*Qg ))';
  bstatadA=bstatadA|bstatad;

  /* II. construct prediction errors from the bigger alternative model for use in */
  /*     the DM, ENC-T and F-test statistics                                      */

  N=T-Pbig-1+ii; cnst=ones(T-Pbig-1+ii-1,1);
  X=cnst~yy[1:N-1]~xx[1:N-1]; Y=yy[2:N];
  Qbet=inv(X'*X)*X'*Y;
  Qu0=yy[N+1]-(1~yy[N]~xx[N])*Qbet; /* 1x1 */ /* prediction error */
  Qf1=Qu0;
  Qf1B=Qf1B|Qf1; /* Pbigx1 vector of prediction errors */

  /* full sample residuals */

  if ii==Pbig;
     fullerrB=Y-X*Qbet;
  endif;

  ii=ii+1;
endo;

/* main statistic - generically comprehensive CS statistics */

mp1=((1/sqrt(Pbig))*sumc((Qf1A*2).*QgA)); /* a vector of rows same as number of
gammas        */
                                        /* = ( mp(gam1) mp(gam2) ... )' =numgammasx1
vector */

/* main statistic - non generically comprehensive CCS statistics */
```

```
mp2=((1/sqrt(Pbig))*sumc((Qf1A).*xx[T-Pbig:T-1,1])); /* 1x1 final CCS statistic */

/* DM statistic -- h=1 so no NW estimator used in variance estiamate */

d1= Qf1A-Qf1B; d1bar= meanc(d1); d1sderr= stdc(d1);
dmstat=sqrt(Pbig)*d1bar/d1sderr;

/* ENC-T statistic -- h=1 so no NW estimator used in variance estiamate */

c1=Qf1A.*(Qf1A-Qf1B); c1bar=meanc(c1); c1sderr=stdc(c1);
cmstat=sqrt(Pbig)*c1bar/c1sderr;

/* F-test -- Chi square version - - 1 dof as there is only 1 extra variable */

Fstat=rows(fullerrB)*((sumc(fullerrA^2)-sumc(fullerrB^2))/sumc(fullerrB^2));

clear X,Y,cnst;
retp(mp1,mp2,dmstat,cmstat,Fstat,adjAA,bstatadA,Qbetout);
endp;

/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */
/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */

/* bootstrap statistics prediction loop used for construction of bootstrap CS tests
*/
/* use modified likelihood function in LS estimation to recenter boot statistic
automatically */

proc (5) = predb(yy,xx,Pbig,gamgam,recterm,betnoPEE);
local
ii,T,cnst,X,Y,N,Qu0,QmY,QmEXTRA,QsY,QsEXTRA,Qbet,Qf1,Qf1A,Qg,QgA,Qf1B,Qf1C,Bmp2;

Qf1A={}; Qf1B={}; Qf1C={};  QgA={};

T=rows(yy);

/* Construct 1-step ahead predictions */

ii=1;
do while ii<=Pbig;

/* quadratic */

  N=T-Pbig-1+ii; cnst=ones(T-Pbig-1+ii-1,1);
  X=cnst~yy[1:N-1]; Y=yy[2:N];
  Qbet=inv(X'*X)*((X'*Y)-((N-1)*recterm[ii,.]'));
  /*Qbet=0.0|0.0;*/

  /* I. prediction errors and pieces for generically comprehensive CS statistic */

  QmY=meanc(yy[1:N]); QmEXTRA=meanc(xx[1:N]);
  QsY=stdc(yy[1:N]);  QsEXTRA=stdc(xx[1:N]);

  Qu0=yy[N+1]-(1~yy[N])*Qbet; /* 1x1 */ /* prediction error */
  Qf1=Qu0;                    /* 1x1 */ /* f', quad loss assumed for stat construct
*/
  Qf1A=Qf1A|Qf1;              /* Pbigx1 */

  Qg=exp( ((atan((yy[N]-QmY)/(2*QsY)))~(atan((xx[N]-QmEXTRA)/(2*QsEXTRA))))*gamgam'
);
                                      /* 1x100, as gamgam is 100x2 */
  QgA=QgA|Qg;
```

```
  /* II. additional pieces: case where assume no PEE and use theta hat from original
stat */

  Qu0=yy[N+1]-(1~yy[N])*betnoPEE[.,ii]; /* 1x1 */ /* prediction error */
  Qf1=Qu0;                    /* 1x1 */ /* f', quad loss assumed for stat construct
*/
  Qf1B=Qf1B|Qf1;              /* Pbigx1 */

  /* III. additional pieces: case where use naiive no adjust boot statistic */

  Qbet=inv(X'*X)*X'*Y;
  Qu0=yy[N+1]-(1~yy[N])*Qbet; /* 1x1 */ /* prediction error */
  Qf1=Qu0;                    /* 1x1 */ /* f', quad loss assumed for stat construct
*/
  Qf1C=Qf1C|Qf1;              /* Pbigx1 */

  ii=ii+1;
endo;

/* main statistic - non generically comprehensive CCS statistics */

Bmp2=((1/sqrt(Pbig))*sumc((Qf1A).*xx[T-Pbig:T-1,1])); /* 1x1 final CCS statistic */

clear X,Y,cnst;
retp(Qf1A,Qf1B,Qf1c,QgA,Bmp2);
endp;

/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */
/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */

/* block bootstrap data sets from entire sample */

proc (1) = bootblok(dat1,lval);
local N,num_uns,undraw1,x1,ib;

N=rows(dat1);
num_uns=N/lval;

/* draw uniforms U[0,T-l+1] */

undraw1=round((N-lval)*rndu(num_uns,1));

x1={};
ib=1;
do while ib<=num_uns;
  x1=x1|dat1[undraw1[ib]+1:undraw1[ib]+lval,.];
  ib=ib+1;
endo;

retp(x1);
endp;

/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */
/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */

/* Killian VAR parametric method bootstrap */

proc (2) = bootkill(yy,xx);
local ii,T,cnst,X,Y,N,Qbet1,Qbet2,res1,res2,num_uns,undraw1,undraw2,yya,xxa;

T=rows(yy);
```

```
/* Generate boostrap data for use in bootstrap statistics construction */

/* 1. fit VAR model to the two series, null imposed */

  N=T-1; cnst=ones(T-2,1); /* assume 1 lag in first equation */
  X=cnst~yy[1:N-1]; Y=yy[2:N];
  Qbet1=inv(X'*X)*X'*Y;
  res1=Y-X*Qbet1;
  X=cnst~yy[1:N-1]~xx[1:N-1]; Y=xx[2:N];
  Qbet2=inv(X'*X)*X'*Y;
  res2=Y-X*Qbet2;

  num_uns=rows(res1);

/* 2. draw uniformly integer values in range (1,...,numres), and draw numres of
these */

undraw1=(round((num_uns-1)*rndu(T,1)))+1;

/* 3. do again, but one value for starting value of x,y in data construction */

undraw2=(round((rows(yy)-1)*rndu(1,1)))+1;

/* build up bootstrap datasets */

yya=zeros(T,1);
xxa=zeros(T,1);

yya[1]=(1~yy[undraw2])*Qbet1+res1[undraw1[1]];
xxa[1]=(1~yy[undraw2]~xx[undraw2])*Qbet2+res2[undraw1[1]];

ii=2;
do while ii<=rows(yy);
    yya[ii]=(1~yy[ii-1])*Qbet1+res1[undraw1[ii]];
    xxa[ii]=(1~yy[ii-1]~xx[ii-1])*Qbet2+res2[undraw1[ii]];
    ii=ii+1;
endo;

clear X,Y,cnst;
retp(yya,xxa);
endp;

/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */
/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */

/* CS statistics */
/* mp(gamma) in, final mp stats out, aggregated over the gamma */

proc (1) = mpstat(Qmpz);
local sout1,Qmpp1,Qmpp2,Qmpp3;

Qmpp1=meanc(Qmpz^2);
Qmpp2=maxc(abs(Qmpz));
Qmpp3=meanc(abs(Qmpz));
sout1=Qmpp1|Qmpp2|Qmpp3;

retp(sout1);
endp;


/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */
/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */
/* MAIN PROGRAM -                                                                */
```

```
/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */
/* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% */

/* MAIN MONTE CARLO EXPERIMENTS */

pp=1;
do while pp <=ptot;

  Pbig=Pvector[pp];

  Qtab5A =zeros(42,3*rows(ls1));
  Qtab10A=zeros(42,3*rows(ls1));

  Qtab5B =zeros(42,3*rows(ls1));
  Qtab10B=zeros(42,3*rows(ls1));

  Qtab5C =zeros(42,3*rows(ls1));
  Qtab10C=zeros(42,3*rows(ls1));

  Qtab5D =zeros(42,rows(ls1));
  Qtab10D=zeros(42,rows(ls1));

  Qtab5E =zeros(42,3);
  Qtab10E=zeros(42,3);

  Qtab5F =zeros(42,3);
  Qtab10F=zeros(42,3);

  Qtab5G =zeros(42,3);
  Qtab10G=zeros(42,3);

  Qtab5H =zeros(42,3);
  Qtab10H=zeros(42,3);

  st=1;
  do while st <= simtot;

    {ydat16,xdat2}=dgp_sim(smpl);

    dg=1;
    dgnum=cols(ydat16);
    do while dg<=dgnum;
      yy1=ydat16[.,dg];

      if dg<=14;
        xx1=xdat2[.,1];
      elseif dg<=28;
        xx1=xdat2[.,2];
      else;
        xx1=xdat2[.,3];
      endif;

      {mp,ccs,dm,cm,ftest,oadjA,bstatadj,Qbet1}=pred(yy1,xx1,Pbig,gamgam);

      /* mp is actual statistics for All Gammas */
      /* these are 100x1 of the mp(gamma), with 100 gammas */

      /* all other tests are 1x1 as they are not constructed for each gamma! */

      /* mp is the cs test; ccs is the Chao, Corradi and Swanson (2001) */
      /* simple version of the CS test; */
      /* dm is the Deibold-Mariano test, called the MSE-T or OOS-T by Clark */
      /* and McCracken (2001 - called MSE-T) and discussed in McCracken (2004 -
```

```
called OOS-T); */
      /* cm is the ENC-T from Clark and McCracken (2001); */
      /* ftest is the chisquared version of the F-test */

      /* oadjA is adj term for boot estimator construction in bpred */

      /* bstatadj is as mp, but uses all T and replaces mp in */
      /* construction of final bootstrap statistics Bmp */

      /* Calculate Actual Final sup, average, etc. mp statistics */
      /* 3x1 vectors with 3 rows, one for each stat */

      {mpstats}=mpstat(mp); /* this simply aggregates over the gamma */

      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */
      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */
      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */

      /* ===================================== */
      /* Now, bootstrap the critical values!!! */
      /* ===================================== */

      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */
      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */
      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */

      /* part A -- bootstrap CS and CCS and use tabulated critical values for the
rest */
      /* these values used in Part A Organization of Results */


      BcritA={}; BcritB={}; BcritC={}; BcritD={};

      booti=1;
      do while booti <= bootnum;

        allstatA={}; allstatB={}; allstatC={}; allstatD={};
        lli=1;
        do while lli<=rows(ls1);

          l_val=ls1[lli];

          {sdataa} = bootblok(yy1~xx1,l_val);


{BoQf1A,BoQf1B,BoQf1C,BoQgA,Bmp2a}=predb(sdataa[.,1],sdataa[.,2],Pbig,gamgam,oadjA,Q
bet1);

          /* Calculate Boot Statistics for All Gammas */
          /* these are 100x1 of the mp(gamma), with 100 gammas */

          /* with regard to BmpA-BmpD below, note the following: */
          /* the first statistic is our regular statistic, with both adjustment
terms (i.e. */
          /* the recentered parameter estimator + the bias adjustment term in the
actual final */
          /* boot statistic. */
          /* the second has no adjust at all and sets theta tilde star = theta hat
*/
          /* the third also has no adjust at all, but at least calculates a theta
star */
          /* the fourth statistic is the simple CCS test bootstrap statistic */
```

```
          BmpA=(1/sqrt(Pbig))*sumc(((BoQf1A*2).*BoQgA)-bstatadj);
          BmpB=((1/sqrt(Pbig))*sumc((BoQf1B*2).*BoQgA))-mp;
          BmpC=((1/sqrt(Pbig))*sumc((BoQf1C*2).*BoQgA))-mp;

          BmpD=Bmp2a-ccs;

          /* Calculate Actual Final sup, average, etc. mp statistics */
          /* 3x1 vectors with 3 rows, one for each stat */

          {BmpstatA}=mpstat(BmpA);
          {BmpstatB}=mpstat(BmpB);
          {BmpstatC}=mpstat(BmpC);

          /* build up a row with all 3 stats times # of lvals in the columns of the
row */
          /* do this for each alternative CV approach for CS test */
          allstatA=allstatA~BmpstatA';
          allstatB=allstatB~BmpstatB';
          allstatC=allstatC~BmpstatC';

          allstatD=allstatD~BmpD; /* a row with the single CCS boot stat times # of
lvals */

          lli=lli+1;
        endo;

        BcritA=BcritA|allstatA;
        BcritB=BcritB|allstatB;
        BcritC=BcritC|allstatC;

        BcritD=BcritD|allstatD;

        booti=booti+1;
      endo;

      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */
      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */
      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */

      /* part B -- Killian bootstrap used for CVs for all tests */
      /* these values used in Part B Organization of Results */

      CSboot={}; CCSboot={}; DMboot={}; CMboot={}; Fboot={};

      booti=1;
      do while booti <= 100;

        {yyb,xxb} = bootkill(yy1,xx1);

        {mpb,ccsb,dmb,cmb,ftestb,junkA,junkb,junkc}=pred(yyb,xxb,Pbig,gamgam);

        {mpstatsb}=mpstat(mpb); /* this simply aggregates over the gamma */

        CSboot=CSboot|mpstatsb';
        CCSboot=CCSboot|ccsb;
        DMboot=DMboot|dmb;
        CMboot=CMboot|cmb;
        Fboot=Fboot|ftestb;


        booti=booti+1;
      endo;
```

```
      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */
      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */
      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */

      /* ==================================== */
      /* END!! bootstrap the critical values!!! */
      /* ==================================== */

      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */
      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */
      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */

      /* Orgainzation of Results */

      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */
      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */
      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */

      /* Part A Organization of Results -- Block Bootstrap and Standard Distribution
Results */


      /* for the 3 CS tests, construct cv's for each of the 3 version of the
bootstrap statistic and tabulate results*/

      /* I. results based on BmpA */

      Bcrit=BcritA;

      cv10={}; cv5={};
      cvi=1;
      do while cvi<=cols(Bcrit);
        temp=sortc(Bcrit[.,cvi],1);
        cv5=cv5|temp[trunc(0.95*rows(Bcrit))];
        cv10=cv10|temp[trunc(0.90*rows(Bcrit))];
        cvi=cvi+1;
      endo;

      /* put together tabulated results */
      /* the table has each row as a DGP, and then the columns are 3 stats for the
first lval, */
      /* and then 3 stats for the second lval, etc. */

      tabi=1;
      do while tabi<=rows(ls1);
        Qmatlval=mpstats .gt cv5[(tabi-1)*3+1:(tabi-1)*3+3,.];

Qtab5A[dg,(tabi-1)*3+1:(tabi-1)*3+3]=Qtab5A[dg,(tabi-1)*3+1:(tabi-1)*3+3]+Qmatlval';
        Qmatlval=mpstats .gt cv10[(tabi-1)*3+1:(tabi-1)*3+3,.];

Qtab10A[dg,(tabi-1)*3+1:(tabi-1)*3+3]=Qtab10A[dg,(tabi-1)*3+1:(tabi-1)*3+3]+Qmatlval
';
        tabi=tabi+1;
      endo;

      /* II. results based on BmpB */

      Bcrit=BcritB;

      cv10={}; cv5={};
      cvi=1;
      do while cvi<=cols(Bcrit);
```

```
        temp=sortc(Bcrit[.,cvi],1);
        cv5=cv5|temp[trunc(0.95*rows(Bcrit))];
        cv10=cv10|temp[trunc(0.90*rows(Bcrit))];
        cvi=cvi+1;
      endo;

      /* put together tabulated results */
      /* the table has each row as a DGP, and then the columns are 3 stats for the
first lval, */
      /* and then 3 stats for the second lval, etc. */

      tabi=1;
      do while tabi<=rows(ls1);
        Qmatlval=mpstats .gt cv5[(tabi-1)*3+1:(tabi-1)*3+3,.];

Qtab5B[dg,(tabi-1)*3+1:(tabi-1)*3+3]=Qtab5B[dg,(tabi-1)*3+1:(tabi-1)*3+3]+Qmatlval';
        Qmatlval=mpstats .gt cv10[(tabi-1)*3+1:(tabi-1)*3+3,.];

Qtab10B[dg,(tabi-1)*3+1:(tabi-1)*3+3]=Qtab10B[dg,(tabi-1)*3+1:(tabi-1)*3+3]+Qmatlval
';
        tabi=tabi+1;
      endo;

      /* III. results based on BmpC */

      Bcrit=BcritC;

      cv10={}; cv5={};
      cvi=1;
      do while cvi<=cols(Bcrit);
        temp=sortc(Bcrit[.,cvi],1);
        cv5=cv5|temp[trunc(0.95*rows(Bcrit))];
        cv10=cv10|temp[trunc(0.90*rows(Bcrit))];
        cvi=cvi+1;
      endo;

      /* put together tabulated results */
      /* the table has each row as a DGP, and then the columns are 3 stats for the
first lval, */
      /* and then 3 stats for the second lval, etc. */

      tabi=1;
      do while tabi<=rows(ls1);
        Qmatlval=mpstats .gt cv5[(tabi-1)*3+1:(tabi-1)*3+3,.];

Qtab5C[dg,(tabi-1)*3+1:(tabi-1)*3+3]=Qtab5C[dg,(tabi-1)*3+1:(tabi-1)*3+3]+Qmatlval';
        Qmatlval=mpstats .gt cv10[(tabi-1)*3+1:(tabi-1)*3+3,.];

Qtab10C[dg,(tabi-1)*3+1:(tabi-1)*3+3]=Qtab10C[dg,(tabi-1)*3+1:(tabi-1)*3+3]+Qmatlval
';
        tabi=tabi+1;
      endo;

      /* IV. results based on CCS test */

      Bcrit=BcritD;

      cv10={}; cv5={};
      cvi=1;
      do while cvi<=cols(Bcrit);
        temp=sortc(Bcrit[.,cvi],1);
        cv5=cv5|temp[trunc(0.95*rows(Bcrit))];
        cv10=cv10|temp[trunc(0.90*rows(Bcrit))];
```

```
        cvi=cvi+1;
      endo;

      /* put together tabulated results */
      /* the table has each row as a DGP, and then the columns are 1 stats for the
first lval, */
      /* and then 1 stats for the second lval, etc. */

      tabi=1;
      do while tabi<=rows(ls1);
        Qmatlval=ccs .gt cv5[(tabi-1)*1+1:(tabi-1)*1+1,.];

Qtab5D[dg,(tabi-1)*1+1:(tabi-1)*1+1]=Qtab5D[dg,(tabi-1)*1+1:(tabi-1)*1+1]+Qmatlval';
        Qmatlval=ccs .gt cv10[(tabi-1)*1+1:(tabi-1)*1+1,.];

Qtab10D[dg,(tabi-1)*1+1:(tabi-1)*1+1]=Qtab10D[dg,(tabi-1)*1+1:(tabi-1)*1+1]+Qmatlval
';
        tabi=tabi+1;
      endo;

      /* V. results based on Ftest, DM, CM tests, assume pi=0 */

      thestats=dm|cm|ftest;

      cv5=1.645|1.645|3.84;
      cv10=1.280|1.280|2.71;

      /* put together tabulated results */
      /* the table has each row as a DGP, and then the columns are the 3 stats */
        Qmatlval=thestats .gt cv5;
        Qtab5E[dg,1:3]=Qtab5E[dg,1:3]+Qmatlval';
        Qmatlval=thestats .gt cv10;
        Qtab10E[dg,1:3]=Qtab10E[dg,1:3]+Qmatlval';

      /* VI. results based on Ftest, DM, CM tests, assume pi not equal 0 */

      thestats=dm|cm|ftest;

      cv5=0.771|1.331|3.84;   /* values for P/R = 1, i.e. P=0.5T */
      cv10=0.443|1.005|2.71;

      /* put together tabulated results */
      /* the table has each row as a DGP, and then the columns are the 3 stats */
        Qmatlval=thestats .gt cv5;
        Qtab5F[dg,1:3]=Qtab5F[dg,1:3]+Qmatlval';
        Qmatlval=thestats .gt cv10;
        Qtab10F[dg,1:3]=Qtab10F[dg,1:3]+Qmatlval';


      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */
      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */
      /* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */

      /* Part B Organization of Results -- Killian Bootstrap Results*/


      /* for the 3 CS tests, construct cv's for the Killian version of the bootstrap
statistic and tabulate results*/

      /* I. results based on CSboot */

      Bcrit=CSboot;
```

```
      cv10={}; cv5={};
      cvi=1;
      do while cvi<=cols(Bcrit);
        temp=sortc(Bcrit[.,cvi],1);
        cv5=cv5|temp[trunc(0.95*rows(Bcrit))];
        cv10=cv10|temp[trunc(0.90*rows(Bcrit))];
        cvi=cvi+1;
      endo;

      /* put together tabulated results */
      /* the table has each row as a DGP, and then the columns are 3 stats */

        Qmatlval=mpstats .gt cv5;
        Qtab5G[dg,1:3]=Qtab5G[dg,1:3]+Qmatlval';
        Qmatlval=mpstats .gt cv10;
        Qtab10G[dg,1:3]=Qtab10G[dg,1:3]+Qmatlval';
        tabi=tabi+1;

      /* for the rest of the tests construct cv's for the Killian version of the
bootstrap statistic and tabulate results*/

      /* II. results based on CSboot */

      Bcrit=CCSboot~DMboot~CMboot;

      thestats= ccs|dm|cm;

      cv10={}; cv5={};
      cvi=1;
      do while cvi<=cols(Bcrit);
        temp=sortc(Bcrit[.,cvi],1);
        cv5=cv5|temp[trunc(0.95*rows(Bcrit))];
        cv10=cv10|temp[trunc(0.90*rows(Bcrit))];
        cvi=cvi+1;
      endo;

      /* put together tabulated results */
      /* the table has each row as a DGP, and then the columns are 3 stats */

        Qmatlval=thestats .gt cv5;
        Qtab5H[dg,1:3]=Qtab5H[dg,1:3]+Qmatlval';
        Qmatlval=thestats .gt cv10;
        Qtab10H[dg,1:3]=Qtab10H[dg,1:3]+Qmatlval';
        tabi=tabi+1;

      if st%100 == 0; if dg == dgnum;

         print "/* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
*/";
         print "/* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
*/";
         print "/* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
*/";

         print
"*************************************************************************";
         print "Intermediate OUTPUT
*****************************************";
         print
"*************************************************************************";
         print "there are" st " sims completed";

         /* Part A Output of Results */
```

```
print "=======================================================";
print "=======================================================";
print "====Block Boot and standard distribution Results=======";
print "=======================================================";
print "=======================================================";
print "OUTPUT for Generic CS test, Type A -- full adjustment ";
print "Output for Bierens Type Test, Quadratic Loss";
print " ";
print "rows are DGPs: 1-2 are size, 3-14 are power";
print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
print " ";
print " Stats are mean(sq), max, mean(abs)";
print " ";
print "3 stats across columns for 1st lval, then again for 2nd lval, etc.";
print " ";
print "quadratic loss";
print " ";
print "5% ";
print Qtab5A./simtot;
print " ";
print "10% ";
print Qtab10A./simtot;
print "=======================================================";
print " ";

print "=======================================================";
print "OUTPUT for Generic CS test, Type B -- no PEE, no adj ";
print "Output for Bierens Type Test, Quadratic Loss";
print " ";
print "rows are DGPs: 1-2 are size, 3-14 are power";
print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
print " ";
print " Stats are mean(sq), max, mean(abs)";
print " ";
print "3 stats across columns for 1st lval, then again for 2nd lval, etc.";
print " ";
print "quadratic loss";
print " ";
print "5% ";
print Qtab5B./simtot;
print " ";
print "10% ";
print Qtab10B./simtot;
print "=======================================================";
print " ";

print "=======================================================";
print "OUTPUT for Generic CS test, Type C -- PEE assumed, but naiive boot
";
print "Output for Bierens Type Test, Quadratic Loss";
print " ";
print "rows are DGPs: 1-2 are size, 3-14 are power";
print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
print " ";
print " Stats are mean(sq), max, mean(abs)";
print " ";
print "3 stats across columns for 1st lval, then again for 2nd lval, etc.";
print " ";
print "quadratic loss";
print " ";
print "5% ";
print Qtab5C./simtot;
```

```
        print " ";
        print "10% ";
        print Qtab10C./simtot;
        print "=========================================================";
        print " ";

        print "=========================================================";
        print "OUTPUT for Simple CCS test full modified boot used as in CS type A
test ";
        print "Output for Bierens Type Test, Quadratic Loss";
        print " ";
        print "rows are DGPs: 1-2 are size, 3-14 are power";
        print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
        print " ";
        print "1 stats across columns for 1st lval, then again for 2nd lval, etc.";
        print " ";
        print "quadratic loss";
        print " ";
        print "5% ";
        print Qtab5D./simtot;
        print " ";
        print "10% ";
        print Qtab10D./simtot;
        print "=========================================================";
        print " ";

        print "=========================================================";
        print "OUTPUT for Simple (tabulated CVs) DM, CM, F tests -- PI=0 ";
        print " ";
        print "rows are DGPs: 1-2 are size, 3-14 are power";
        print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
        print " ";
        print "all 3 stats across columns";
        print " ";
        print "quadratic loss";
        print " ";
        print "5% ";
        print Qtab5E./simtot;
        print " ";
        print "10% ";
        print Qtab10E./simtot;
        print "=========================================================";
        print " ";

        print "=========================================================";
        print "OUTPUT for Simple (tabulated CVs) DM, CM, F tests -- PI not equal 0
";
        print " ";
        print "rows are DGPs: 1-2 are size, 3-14 are power";
        print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
        print " ";
        print "all 3 stats across columns";
        print " ";
        print "quadratic loss";
        print " ";
        print "5% ";
        print Qtab5F./simtot;
        print " ";
        print "10% ";
        print Qtab10F./simtot;
        print "=========================================================";
        print " ";
```

```
       /* Part B Output of Results */

       print "=======================================================";
       print "=======================================================";
       print "====Killiank Boot Results=======";
       print "=======================================================";
       print "=======================================================";
       print "OUTPUT for Generic CS test, Killian bootstrap ";
       print "Output for Bierens Type Test, Quadratic Loss";
       print " ";
       print "rows are DGPs: 1-2 are size, 3-14 are power";
       print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
       print " ";
       print " Stats are mean(sq), max, mean(abs)";
       print " ";
       print "3 stats across columns for 1st lval, then again for 2nd lval, etc.";
       print " ";
       print "quadratic loss";
       print " ";
       print "5% ";
       print Qtab5G./simtot;
       print " ";
       print "10% ";
       print Qtab10G./simtot;
       print "=======================================================";
       print " ";

       print "=======================================================";
       print "OUTPUT for Generic CCS, DM, CM test, Killian bootstrap ";
       print "Output for Bierens Type Test, Quadratic Loss";
       print " ";
       print "rows are DGPs: 1-2 are size, 3-14 are power";
       print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
       print " ";
       print "all 3 stats across columns";
       print " ";
       print "quadratic loss";
       print " ";
       print "5% ";
       print Qtab5H./simtot;
       print " ";
       print "10% ";
       print Qtab10H./simtot;
       print "=======================================================";
       print " ";

    endif; endif;

    dg=dg+1;
  endo;

  print "Simulation Number " st " is done!!";

  st=st+1;
endo;

print "/* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */";
print "/* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */";
print "/* NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN */";

print "********************************************************************";
print "FINAL OUTPUT *******************************************************";
print "********************************************************************";
```

```
/* Part A Output of Results */

print "=======================================================";
print "=======================================================";
print "====Block Boot and standard distribution Results=======";
print "=======================================================";
print "=======================================================";
print "OUTPUT for Generic CS test, Type A -- full adjustment ";
print "Output for Bierens Type Test, Quadratic Loss";
print " ";
print "rows are DGPs: 1-2 are size, 3-14 are power";
print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
print " ";
print " Stats are mean(sq), max, mean(abs)";
print " ";
print "3 stats across columns for 1st lval, then again for 2nd lval, etc.";
print " ";
print "quadratic loss";
print " ";
print "5% ";
print Qtab5A./simtot;
print " ";
print "10% ";
print Qtab10A./simtot;
print "=======================================================";
print " ";

print "=======================================================";
print "OUTPUT for Generic CS test, Type B -- no PEE, no adj ";
print "Output for Bierens Type Test, Quadratic Loss";
print " ";
print "rows are DGPs: 1-2 are size, 3-14 are power";
print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
print " ";
print " Stats are mean(sq), max, mean(abs)";
print " ";
print "3 stats across columns for 1st lval, then again for 2nd lval, etc.";
print " ";
print "quadratic loss";
print " ";
print "5% ";
print Qtab5B./simtot;
print " ";
print "10% ";
print Qtab10B./simtot;
print "=======================================================";
print " ";

print "=======================================================";
print "OUTPUT for Generic CS test, Type C -- PEE assumed, but naiive boot ";
print "Output for Bierens Type Test, Quadratic Loss";
print " ";
print "rows are DGPs: 1-2 are size, 3-14 are power";
print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
print " ";
print " Stats are mean(sq), max, mean(abs)";
print " ";
print "3 stats across columns for 1st lval, then again for 2nd lval, etc.";
print " ";
print "quadratic loss";
print " ";
print "5% ";
```

```
print Qtab5C./simtot;
print " ";
print "10% ";
print Qtab10C./simtot;
print "=======================================================";
print " ";

print "=======================================================";
print "OUTPUT for Simple CCS test full modified boot used as in CS type A test ";
print "Output for Bierens Type Test, Quadratic Loss";
print " ";
print "rows are DGPs: 1-2 are size, 3-14 are power";
print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
print " ";
print "1 stats across columns for 1st lval, then again for 2nd lval, etc.";
print " ";
print "quadratic loss";
print " ";
print "5% ";
print Qtab5D./simtot;
print " ";
print "10% ";
print Qtab10D./simtot;
print "=======================================================";
print " ";

print "=======================================================";
print "OUTPUT for Simple (tabulated CVs) DM, CM, F tests -- PI=0 ";
print " ";
print "rows are DGPs: 1-2 are size, 3-14 are power";
print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
print " ";
print "all 3 stats across columns";
print " ";
print "quadratic loss";
print " ";
print "5% ";
print Qtab5E./simtot;
print " ";
print "10% ";
print Qtab10E./simtot;
print "=======================================================";
print " ";

print "=======================================================";
print "OUTPUT for Simple (tabulated CVs) DM, CM, F tests -- PI not equal 0 ";
print " ";
print "rows are DGPs: 1-2 are size, 3-14 are power";
print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
print " ";
print "all 3 stats across columns";
print " ";
print "quadratic loss";
print " ";
print "5% ";
print Qtab5F./simtot;
print " ";
print "10% ";
print Qtab10F./simtot;
print "=======================================================";
print " ";

/* Part B Output of Results */
```

```
  print "=========================================================";
  print "=========================================================";
  print "====Killiank Boot Results=======";
  print "=========================================================";
  print "=========================================================";
  print "OUTPUT for Generic CS test, Killian bootstrap ";
  print "Output for Bierens Type Test, Quadratic Loss";
  print " ";
  print "rows are DGPs: 1-2 are size, 3-14 are power";
  print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
  print " ";
  print " Stats are mean(sq), max, mean(abs)";
  print " ";
  print "3 stats across columns for 1st lval, then again for 2nd lval, etc.";
  print " ";
  print "quadratic loss";
  print " ";
  print "5% ";
  print Qtab5G./simtot;
  print " ";
  print "10% ";
  print Qtab10G./simtot;
  print "=========================================================";
  print " ";

  print "=========================================================";
  print "OUTPUT for Generic CCS, DM, CM test, Killian bootstrap ";
  print "Output for Bierens Type Test, Quadratic Loss";
  print " ";
  print "rows are DGPs: 1-2 are size, 3-14 are power";
  print "same for 15-16 and 17-28, but different coeff on AR y_t-1";
  print " ";
  print "all 3 stats across columns";
  print " ";
  print "quadratic loss";
  print " ";
  print "5% ";
  print Qtab5H./simtot;
  print " ";
  print "10% ";
  print Qtab10H./simtot;
  print "=========================================================";
  print " ";



  pp=pp+1;
endo;

output off;
```