r o k r 2 . P R G

/* R o k k o                      -                               */
/* rokr2.prg -               jul 2003                         */
/* revision example - triple comparison                       */
/* as rok2.prg                                                */
/* related to multi2a.prg from Andrews paper                  */
/* this version considers bivariate joint distributions      */
/* empirical example                                         */
/*                                                     */

format /MA1/LD 7,4;
outwidth 255;
output file = c:\papers\v_rokko\revision\rokr2.out reset;
output on;

/* Parameters to Set Before Carrying out analysis */
clear bet, delt, n, gamm, thet, a2, A11, A22, ve1, ve2, cove1e2, pc_K1, pc_H1, pc_Y1, ld_pc_Y;

bet= 0.95 ; /* beta */
delt= 0.025 ; /* delta */
n= 1.012 ; /* n */
gamm= 0.0028 ; /* gamma */
thet= 0.50 ; /* theta */
a2= 0.0044 ; /* a_2 */
A11= 0.95 ; /* A_11 */
A22= -0.10 ; /* A_22 */
ve1= 0.039 ; /* var(e1) */
ve2= 0.077 ; /* var(e2) */
cove1e2= -0.030 ; /* cov(e1,e2) */

ls=(2,5,7); /* density of U grid */
unum=5; /* number of boot replications for CV calc */
simtot=1;

/* read in data for empirical and simulation exercises */
/* data assumed to be in log differences immediately at this stage */
/* for empirical need Y and K */

/* data are from 1929-2002 (74 obs) */
/* vars are K, GDP, and N */
/* last K obs for 2002 is a dummy value, but as lagged K used, poses no */
/* problem */

load datin[74,4]=c:\papers\v_rokko\revision\annUSrev.dat;

Yin=datin[.,2]*1000000000;
Yin=datin[.,2];
Kin=datin[.,1];
Nin=datin[.,3]*1000;
Nin=datin[.,3];
Hin=datin[.,4];

pc_Y=Yin./Nin;
pc_K=Kin./Nin;
pc_H=Hin;

pc_Y=pc_Y[5:rows(pc_Y),.];
pk_K=pk_K[5:rows(pk_K),.];
pk_H=pk_H[5:rows(pk_H),.];

print "Annual Data with obs = " rows(Yin);
print Yin~Kin~Nin~Hin;
**rokr2.PRG**

```
TT=74; /* define length of actual data sample */

/* log difference levels series */
ld_Y=ln(Yin[2:rows(Yin),.])-ln(Yin[1:rows(Yin)-1,.]);
ld_K=ln(Kin[2:rows(Kin),.])-ln(Kin[1:rows(Kin)-1,.]);
ld_N=ln(Nin[2:rows(Nin),.])-ln(Nin[1:rows(Nin)-1,.]);
ld_H=ln(Hin[2:rows(Hin),.])-ln(Hin[1:rows(Hin)-1,.]);

/* log difference per capita GDP */
ld_pc_Ya=ld_Y.ld_N;

/* log difference per capita hours */
ld_pc_Ha=ld_H;

/* make this sample 70 long */
ld_Y=ld_Y[4:rows(ld_Y),.];
ld_K=ld_K[4:rows(ld_K),.];
ld_N=ld_N[4:rows(ld_N),.];
ld_H=ld_H[4:rows(ld_H),.];
ld_pc_Y=ld_pc_Ya[4:rows(ld_pc_Ya),.]; /* pare down to 70 obs */
ld_pc_H=ld_pc_Ha[4:rows(ld_pc_Ha),.]; /* pare down to 70 obs */
ld_pc_Hl=ld_pc_Ha[3:rows(ld_pc_Ha)-1,.];

TT=rows(ld_pc_Y); /* define length of actual data sample */

/* in addition to using ld_pc_Y, may use quarterly y data for the actual GDP */
/* in the final comparison */
load datin2[224,2]=c:\papers\v_rokko\qrtUS.dat; /* 1947:1 - 2002:4 */
ld_Y1=ln(datin2[2:rows(datin2),1])-ln(datin2[1:rows(datin2)-1,1]);
ld_N1=ln(datin2[2:rows(datin2),2])-ln(datin2[1:rows(datin2)-1,2]);
ld_pc_Y1=ld_pc_Y-ld_N;
ld_pc_Q=ld_pc_Y1[4:rows(ld_pc_Y1),.]; /* pare down to 220 obs */
ld_pc_Q1=ld_pc_Y1[3:rows(ld_pc_Y1)-1,.];
```

/* procedure - simulate data */
```
proc (4) = sim2(smpl);
```
/* loop for different parameters, etc., to make 2 or three versions of y, k, h, c */

logk={};
logh={};
logy={};
logc={};
dati=1;
do while dati <= 3;

/* put in looping parameter values here */
if dati==1; bet=0.988;
elseif dati==2; bet=0.90;
else; bet=0.95; endif;

/* generate all of the parameters to use in final equations */
/* below, h is really htil */

w = (eye(2)-((A11~0)|(0~A22)))*(0|a2);
phi_ =
((n^thet)*[1-bet*(1-delt)]*exp(-w[2,1]/n)*exp(((2*thet)-1)*w[2,1]))/(bet*thet)^((-
1)/(1-thet));
h_ = (exp(w[1,1])/gamm)*(1-thet)*
((n^(-thet))*exp(-2*thet*w[2,1]))/((n^(-thet))*exp(-2*thet*w[2,1])*phi_^thet) -
(\phi_*exp(-w[2,1])*(1-((1-delt)*exp(-w[2,1])/n)) )

ktil_ = phi_*h_;
ytil_ = h_*(n^(-thet))*exp(-2*thet*w[2,1])*(\phi_^thet) -
(\phi_*exp(-w[2,1])*(1-((1-delt)*exp(-w[2,1])/n)));
ctil_ = (ln(ktil_)|w|w);
dstar_ = (ln(ktil_)|w|w);
sstar_ = ((ln(ktil_))|w|w);
B_ = zeros(5,2); B_[1,1]=1;

r_1 = -((exp(w[2,1]))*exp(-2*w[2,1]))/ctil_;
r_7 = -((exp(w[2,1]))/ctil_)*((thet*ytil_)+(((1-delt)/n)*exp(-2*w[2,1])*ktil_ ));
r_9 = (exp(w[1,1])/ctil_)*(-((1-delt)/n)*exp(-w[2,1])]*ktil_ * (1 -
((1-delt)/n)*exp(-w[2,1]) - exp(-w[2,1])/n )

r_8 = (exp(w[1,1])/ctil_)*ln(ctil_);
r_5=0;
r_6=0;
r_2 = -((exp(w[1,1]))/ctil_)*exp(-w[2,1]);

r_11 = -((exp(w[1,1]))*exp(-2*w[2,1]))/(ctil_*bet^2)) -
((thet*1-thet)*exp(w[1,1])*ytil_/ctil_); r_18 = r_1;
r_81 = r_18;
r_19 = -(r_1*r_9*exp(-w[1,1])) - (exp(-w[1,1])/(ctil_))*(thet^2)*ytil_;

r_91 = r_19;

r_16 = 0;

r_61 = r_16;

r_17 = -(r_1*r_7*exp(-w[1,1])) - (exp(-w[1,1])/(ctil_))*(thet^2)*ytil_;

r_71 = r_17;

r_98 = r_98;

r_88 = r_8;

r_68 = 0;

r_66 = r_68;

r_67 = r_7;

r_78 = 0;

r_87 = 0;

r_69 = 0;

r_96 = r_69;

r_99 = -(r_9^2)*exp(-w[1,1]) + (exp(-w[1,1])/(ctil_))*(thet^2)*ytil_;

r_79 = -(r_9*r_7*exp(-w[1,1])) - (exp(-w[1,1])/(ctil_))*(thet^2)*ytil_;
rokr2.PRG

VQlast = eye(5) * -0.00;

vi = 1;
do while vi <= 1000000;
    VQ = R_ + (bet*phi1_ * VQlast * phi1_ ) - ( (bet*phi1_ * VQlast * B_ ) + F_ ) * inv(Q_ +
bet*B_ * VQlast * B_ ) * ( (bet*B_ * VQlast * phi1_ ) + F_ );
    if sumc(sumc(T VQ - VQlast)^2) .le 0.00000001;
        goto vend1;
    else;
        VQlast = VQ_; 
        vi = vi + 1;
    endif;
endo;

vend1:

/*print "for VQ, use x sims, x = * vi; */
if vi == 1000000; print "VQ didn't converge, and VQ is * VQ_; endif;

Qbar_ = Q_ + bet*B_ * VQ_ * B_;
Fbar_ = F_ + bet*phi1_ * VQ_ * B_;
K1_ = -Qbar_ * Fbar_;`;
/*
print "K1"
print K1_; 
*/

VSlast = ones(5,1) * -0.0;

vi = 1;
do while vi <= 1000000;
    VS = inv(eye(5) - (bet*(K1_ * B_ + phi1_ )) * ( (K1_ *(c1_ +
2*bet*B_ * VQ_ * phi0_ ) ) + c2 + (2*bet*phi1_ * VQ_ * phi0_ ) ));
    if sumc(sumc((VS - VSlast)^2) .le 0.00000001;
        goto vend2;
    else;
        VSlast = VS_; 
        vi = vi + 1;
    endif;
endo;

vend2:

/*print "for VS, use x sims, x = * vi; */
if vi == 1000000; print "VS didn't converge, and VS is * VS_; endif;
c1bar_ = ( c1_ + (bet*VS_ * B_ ) + (2*bet*phi0_ * VQ_ * B_ ) )';

K0_ = -0.5*inv(Qbar_)*c1bar_;`;
/*
print "K0"
print K0_; 
*/
/*K0_[1,1]=0.0;
K0_[2,1]=0.0*/

P10_ = ( (K0_[1] + K1_[1,3] * a2) | (K0_[2] + K1_[2,3] * a2) );
P11_ = K1_[1,1] K1_[2,1];
P12_ = ( (K1_[1,2] * A11 + K1_[1,4] ) | (K1_[2,2] * A11 + K1_[2,4] ) ) ~ ( (K1_[1,3] * A22 + K1_[1,5] ) | (K1_[2,3] * A22 + K1_[2,5] ) );
P13_ = ( (K1_[1,2] | K1_[2,2]) ~ (K1_[1,3]| K1_[2,3] ) );
/*
print "VS" VS_;
print "VQ" VQ_; 
*/
/* estimate ve1, ve2, and cove1e2 */

/* generate (u x) */

ci1=1;
ci2=1;
boneb=1000000000000;
ve1=0.00005; ve2=0.0015; cove1e2=0.0;
do while ci1<=14;
  ve1=ve1*2;
do while ci2<=14;
  ve2=ve2+0.0005;
  ux=zeros(smpl+100,2);
  z=zeros(smpl+100,1);
  z[1]=1;
  ee = rndn(smpl+100,2); /* normal RVs */
  cova=(ve1~cove1e2)|(cove1e2~ve2);
  ee=(chol(cova)*ee')';
  dgi=2;
do while dgi<=smpl+100;
    ux[dgi,] = [ (0|a2)+((A11~0)|(0~A22))*ux[dgi-1,]'+ee[dgi,]' ];
    z[dgi,] = z[dgi-1,]*exp(ux[dgi,2]);
    dgi = dgi+1;
endo;

/* generate logk, logh, logy, logc */

logka=zeros(smpl+100,1);
logha=zeros(smpl+100,1);
logya=zeros(smpl+100,1);
logca=zeros(smpl+100,1);
dgi=3;
do while dgi<=smpl+100;
  logka[dgi,1]=ln(z[dgi,1]) + PI0_[1,1] + PI1_[1,1]*(logka[dgi-1,1]-ln(z[dgi-2,1])) + PI2_[1,1]*ux[dgi-1,]'+PI3_[1,1]*ee[dgi,]';
  logha[dgi,1]=PI0_[2,1] + PI1_[2,1]*(logka[dgi-1,1]-ln(z[dgi-2,1])) + PI2_[2,1]*ux[dgi-1,]'+PI3_[2,1]*ee[dgi,]';
  dgi = dgi+1;
endo;
dgi=3;
do while dgi<=smpl+100;
  logya[dgi,1]=(1-thet)*ln(z[dgi,1]) - thet*ln(n) + (1-thet)*logha[dgi,1] + (1-thet)*exp(logka[dgi-1,1]) +
  ((1-delt)/n)*exp(logka[dgi-1,1])
  doc: dgi = dgi+1;
endo;

varY=vcx(logya[101:smpl+100,] - logya[100:smpl+99,]);
bone=abs(varY-vcx(id_pc_Y));
if bone .le boneb;
  ve1b=ve1;
  ve2b=ve2;
endif;

ci2=ci2+1;
endo;
ci1=ci1+1;
endo;
ve1=ve1b;
ve2=ve2b;
print "Best sigeps:" ((ve1~ve2));

/* old sigma epsilon calcs */
lz = (ln(pc_Y1[2:rows(pc_Y1),.]) + thet*ln(n) - (1-thet)*ln(pc_H1[2:rows(pc_H1),.]) - 
    thet*ln(pc_K1[1:rows(pc_K1)-1,.]))/(1-thet);
xz=lx[2:rows(lx),.]-lx[1:rows(lx)-1,];
e2t = xt[2:rows(xt),.-a2-A22*xt[1:rows(xt)-1,];

/* old sigma epsilon calcs */
e2t=e2t - meanc(e2t);
print "e2t";
print e2t;
print "mean" meanc(e2t);
*/
e1t=zeros(rows(lx)-2,1); /* = rows(e2t) = rows(pc_Y) - 3 */
ei=1;
do while ei <= rows(e1t);
if ei == 1;
    ssume1=0;
else;
    ssume1=sume1*A11 + e1t[ei-1,1];
endif;
e1t[ei] = (1/K1_[1,2])*ln(pc_K1[ei+3,.]) - lz[ei+1,.] - (K0_[1] +K1_[1,3]*a2) - 
    (K1_[1,1]*ln(pc_K1[ei+2,.]) -lz[ei,.])) - ((K1_[1,5]+K1_[1,3]*A22)*xt[ei,.]) - 
    ((K1_[1,4]+K1_[1,2]*A11)*sume1) - (K1_[1,3]*e2t[ei,.]);
ei = ei +1;
endo;
e1t=e1t - meanc(e1t);
/*
print "e1t";
print e1t;
print "mean" meanc(e1t);
*/
covvv=vcx(e1t~e2t);
ve1=covvv[1,1];
ve2=covvv[2,2];
cove1e2=covvv[1,2];

print "Epsilon cals are" (ve1~ve2~cove1e2);
ve1=0.001;
ve2=0.001;
cove1e2=-0.0;
*/
/* end cov calcs */
/* generate all of the data!!! */
/* generate (u x) */

ux=zeros(smpl+100,2);
z=zeros(smpl+100,1);
z[1]=1;
eee=rndn(smpl+100,2); /* normal RVs */
cova=(ve1~cove1e2)|(cove1e2~ve2);
eee=chol(cova)*ee';
dgi=2;
do while dgi<=smpl+100;
  ux[dgi,.] = ( [0|a2]+((A11~0)|(0~A22))*ux[dgi-1,.]'*+ee[dgi,.]')';
  z[dgi,.] = z[dgi-1,.]'*exp(ux[dgi,2]);
  dgi=dgi+1;
endo;
/* generate logk, logh, logy, logc */

logka=zeros(smpl+100,1);
logha=zeros(smpl+100,1);
logya=zeros(smpl+100,1);
logca=zeros(smpl+100,1);
dgi=3;
do while dgi<=smpl+100;
  logka[dgi,1]=ln(z[dgi-1,.]') + PI0_[1,1] + PI1_[1,1]*ln(z[dgi-1,1]) + PI2_[1,1]*ux[dgi-1,1]' + PI3_[1,1]*ee[dgi,1]';
  logha[dgi,1]= PI0_[2,1] + PI1_[2,1]*ln(z[dgi-1,1]) + PI2_[2,1]*ux[dgi-1,1]' + PI3_[2,1]*ee[dgi,1]';
  dgi=dgi+1;
endo;
dgi=3;
do while dgi<=smpl+100;
  logya[dgi,1]=(1-thet)*ln(z[dgi,1]) - thet*ln(n) + (1-thet)*logha[dgi,1] + thet*logka[dgi,1] + ((1-del)/n)*exp(logka[dgi,1]) + exp(logka[dgi,1])';
  logc=logc(dgi,1); 
  dgi=dgi+1;
endo;
/* add to final data */

logk=logk~logka;
logh=logh~logha;
logy=logy~logya;
logc=logc~logca;
dati=dati+1;
endo;
/* return two or three different growth rate of Y series, one for each of two approximation methods */
/* with everything for a given alpha, beta, tau */
/* also, return the growth rates and secondly their lags */
```latex
rokr2.PRG

retp( (logy[101:smpl+100,.]-logy[100:smpl+99,.]),
      (logh[101:smpl+100,.]-logh[100:smpl+99,.]),
      (logy[100:smpl+99,.]-logy[99:smpl+98,.]),
      (logh[100:smpl+99,.]-logh[99:smpl+98,.]) );
endp;

/* */
/* */
/* calculate the main statistic for a given actual data series and simulated data
series */
/* one statistic for each benchmark-alternative combination */
/* i.e. calculate Z_{j,T,S} statistics */
/* */
/* UNIVARIATE or MARGINAL DISTRIBUTION CASE */
/* */

proc (1) zee1(Yd,Yds,uuone);
local zi,zn,simFben,simFalt,pieceben,outu;

/* for a given data series (Yd), there are NS columns of simulated series, */
/* the first for the benchmark model and the rest for the alternative models */
/* */

outu={};

zn=sqrt(rows(Yd));
simFben=(1/rows(Yds[.,1]))*sumc(Yds[.,1].le uuone);
pieceben=(1/zn)*sumc( ((Yd[.,.].le uuone)-simFben)^2 );
zi=1;
do while zi<=cols(Yds)-1;
    simFalt=(1/rows(Yds[.,zi+1]))*sumc(Yds[.,zi+1].le uuone);
    outu=outu~(pieceben- ((1/zn)*sumc( ((Yd[.,.].le uuone)-simFalt)^2 )));
    zi=zi+1;
endo;

/* outu is a row of Z_{j,T,S} statistics, one for each bench versus alternative
comparison */
/* Thus, Yds, the simulated data, must have at least two columns to allow for the */
/* construction of one bench and at least one alternative */
/* this output is for a single u value */

retp(outu);
endp;

/* */
/* */
/* calculate the main statistic for a given actual data series and simulated data
series */
/* one statistic for each benchmark-alternative combination */
/* i.e. calculate Z_{j,T,S} statistics */
/* */
```

Page 9
PROC (1) = zee2(Yd, Yh, Ydl, Yhl, Yds, Ydsh, Ydshl, uuone, uutwo, uuthree, uufour);
local zi, zN, simFben, simFalt, pieceben, outu;
/* for a given data series (Yd), there are NS columns of simulated series, */
/* the fist for the benchmark model and the rest for the alternative models */
/* */
outu={};
zN=sqrt(rows(Yd));
simFben=(1/rows(Yd[.,1]))*sumc((Yd[.,1] .le uuone) .and (Ydsh[.,1] .le uutwo) .and (Ydsl[.,1] .le uuthree) .and (Ydshl[.,1] .le uufour));
pieceben=(1/zN)*sumc( ( ( (Yd[.,1] .le uuone) .and (Yh[.,1] .le uutwo) .and (Ydl[.,1] .le uuthree) .and (Yhl[.,1] .le uufour) ) -simFben)^2 );
/* print "simFben " simFben;
print "pieceben " pieceben;/*
zi=1;
do while zi<=cols(Yds)-1;
simFalt=(1/rows(Yds[.,zi]))*sumc((Yds[.,zi] .le uuone) .and (Ydsh[.,zi] .le uutwo) .and (Ydsl[.,zi] .le uuthree) .and (Ydshl[.,zi] .le uufour));
/* print "simFalt " simFalt;/*
outu=outu~(pieceben- ((1/zN)*sumc( ( ( (Yd[.,1] .le uuone) .and (Yh[.,1] .le uutwo) .and (Ydl[.,1] .le uuthree) .and (Yhl[.,1] .le uufour) ) -simFalt)^2 )))
zi=zi+1;
endo;
/* outu is a row of Z_{j,T,S} statistics, one for each bench versus alternative comparison */
/* Thus, Yds, the simulated data, must have at least two columns to allow for the */
/* construction of one bench and at least one alternative */
/* this output is for a single u value */
/* print "outu is " outu;*/
retp(outu);
endp;
/* */
/* */
/* actual max stats */
/* */
/* */
proc (1) = stats(ouAv);
local ZT;
ZT={};
ZT=ZT|maxc(ouAv[1:rows(ouAv)]);
rokr2.PRG

/* so one statistic outputed here */
/* if doing simulations, may do max across various subsets of those individual
elements of ouAv, thus */
/* generating size and power statistics, etc. */
ret p(ZT);
end p;
/* */
/* */
/* */
/* bootstrap data sets */
/* */
/* */
/* */
proc (1) = bootk(dat1,lval);
local N,num_uns,undraw1,x1,ib;
N=rows(dat1);
um_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;
/* ---------------------------------------------------------- */
/* ---------------------------------------------------------- */
/* ^^^^^^^^^^^^^^^^^^ MAIN PROGRAM ^^^^^^^^^^^^^^^^^^^^^^^ */
/* ---------------------------------------------------------- */
/* ---------------------------------------------------------- */
/* for use in simulation */
/* {Udat}=sim1(1000);
Umin=(minc(Udat[,1:8]));
Umax=(maxc(Udat[,1:8]));
Uinc=(Umax-Umin)/Unum;
uone=meanc(udat);*/
/* for empirical application */
Umin=(minc(ld_pc_Y~ld_pc_H));
Umax=(maxc(ld_pc_Y~ld_pc_H));
Uinc=(Umax-Umin)/Unum;
outzaa={};
/* raw data assumed already to be in log differences */
/* define length of the simulated samples, given TT, where TT is actual data sample used */
outza=zeros(12,23);

tsi=1;
do while tsi <= 4;
   if tsi == 1; SS=TT;  elseif tsi==2; SS=2*TT;  elseif tsi==3; SS=5*TT; else;
   SS=10*TT; endif;
   li=1;
do while li <= 3;
      l_val=l{li}; /* l_val for actual data */
      /* now, make l_vals for simulated data */
      if tsi==1; l_valS=l{li}; elseif tsi==2; l_valS=2*l{li}; else;
      l_valS=5*l{li}; endif;
      /* assume that the "l_val" values are for the actual data -- for the simulated, multiply them */
      /* by the factor 1, 2, or 5, as SS is made by the same factors multiplying TT */
      si=1;
do while si<=simtot;
         /* simulate data according to some different models, where TT is the length of the simulated sample */
         /* set parameter estimation data equal to raw data */
         pc_Y1=pc_Y;
         pc_K1=pc_K;
         pc_H1=pc_H;
         {YdatS,HdatS,YdatSl,HdatSl}=sim2(SS); /* this is the simulated data and its lag, */
         /* given the actual data and any calculations done */
         /* to estimate parameters, form the actual data, etc. */
         /* now, calculate all stats over U range */
         outttu={};
         ui=1; uuui=Umin-Uinc;
do while ui <=Unum+1;
            uuui=uuui+uinc;
            uj=1; uuuj=Umin-Uinc;
do while uj <=Unum+1;
               uuuj=uuuj+uinc;
               uk=1; uuuk=Umin-Uinc;
do while uk <=Unum+1;
                  uuuk=uuuk+uinc;
                  ul=1; uuul=Umin-Uinc;
do while ul <=Unum+1;
                     uuul=uuul+uinc;

{outttu}=zee2(ld_pc_Y,ld_pc_Yl,ld_pc_H,ld_pc_Hl,YdatS,YdatSl,HdatS,HdatSl,uuui,uuuj,uuuk,uuul);

   outttu=outttu|outttu;
endo;
   uk=uk+1;
endo;
uj = uj + 1;
endo;
ui = ui + 1;
endo;
/* average up across the u */
obody = meance(outttu);
/* print "obody"; print oboy; */
/* calculate statistics */
{ZZ} = stats(obody);
/* print "stats ";
print ZZ;
*/
/* Now, bootstrap the critical values!!! */
ZZZb1 = {}; ZZZb2 = {};
booti = 1;
do while booti <= bootnum;
/* print "Boot sim " booti; */
{bdatAct} = bootk((pc_Y\~ pc_K\~ pc_H\~ ld_pc_Y\~ ld_pc_H\~ ld_pc_Yl\~ ld_pc_Hl), l_val);
/* change actual data used in parameter estimation in simulation routine */
pc_Y1 = bdatAct[., 1];
pc_K1 = bdatAct[., 2];
pc_H1 = bdatAct[., 3];
{YdatS_b, HdatS_b, YdatS_bl, HdatS_bl} = sim2(SS);
{bdatSim} = bootk((YdatS_b\~ HdatS_b\~ YdatS_bl\~ HdatS_bl), l_valS);
/* I carry out the usual boot approach */
outttu = {};
ui = 1; uuui = Umin - Uinc;
do while ui <= Unum + 1;
    uuui = uuui + uinc;
    uj = 1; uuuj = Umin - Uinc;
do while uj <= Unum + 1;
    uuuj = uuuj + uinc;
    uk = 1; uuuk = Umin - Uinc;
do while uk <= Unum + 1;
    uuuk = uuuk + uinc;
    ul = 1; uuul = Umin - Uinc;
do while ul <= Unum + 1;
    uuul = uuul + uinc;
{outttu} = zee2(bdatAct[., 4], bdatAct[., 5], bdatAct[., 6], bdatAct[., 7],
bdatSim[., 1: cols(YdatS_b)], bdatSim[., cols(YdatS_b) + 1: 2*cols(YdatS_b)],
bdatSim[., 2*cols(YdatS_b) + 1: 3*cols(YdatS_b)], bdatSim[., 3*cols(YdatS_b) + 1: 4*cols(YdatS_b)], uuui, uuuj, uuuk, uuul);
rokr2.PRG

outttu = outttu | outtu;
ul = ul + 1;
endo;
uk = uk + 1;
endo;
uj = uj + 1;
endo;
ui = ui + 1;
endo;

boboy1 = mean(outttu) - oboy; /* these are the usual boot stats with the
booted minus the actual */

/* I'll carry out the alternative boot for S growing faster than T */
outttu = {};
ui = 1; uuui = Umin - Uinc;
do while ui <= Unum + 1;
uuui = uuui + uinc;
ui = 1; uuuj = Umin - Uinc;
do while uj <= Unum + 1;
uuuj = uuuj + uinc;
uk = 1; uuuk = Umin - Uinc;
do while uk <= Unum + 1;
uuuk = uuuk + uinc;
ul = 1; uulu = Umin - Uinc;
do while ul <= Unum + 1;
uulu = uulu + uinc;

{outttu} = zee2(bdatAct[., 4], bdatAct[., 5], bdatAct[., 6], bdatAct[., 7], YdatS_b, HdatS_b,
               YdatS_bl, HdatS_bl, uuui, uuuj, uuuk, uuul);

outttu = outttu | outtu;
ui = ui + 1;
endo;
uk = uk + 1;
endo;
uj = uj + 1;
endo;
ui = ui + 1;
endo;

boboy2 = mean(outttu) - oboy; /* these are the alt boot stats with the
booted minus the actual, ut where */ /* only the actual data are resampled, and not
the booted, i.e. S grow faster T */

/* calculate statistics */
{ZZb1} = stats(boboy1);
ZZB1 = ZZB1 | ZZb1';
{ZZb2} = stats(boboy2);
ZZB2 = ZZB2 | ZZb2';

booti = booti + 1;
endo;
/* print "Boot Stats";
print ZZB1 ~ ZZB2;
*/
/* construct critical values */

cv1_10 = {};
cv1_5 = {};
rokr2.PRG

cvi = 1;
do while cvi <= cols(ZZZb1);
    temp = sortc(ZZZb1[., cvi], 1);
    cv1_10 = cv1_10 | temp[trunc(0.90 * rows(ZZZb1))];
    cv1_5 = cv1_5 | temp[trunc(0.95 * rows(ZZZb1))];
    cvi = cvi + 1;
endo;

cv2_10 = {};
cv2_5 = {};
cvi = 1;
do while cvi <= cols(ZZZb2);
    temp = sortc(ZZZb2[., cvi], 1);
    cv2_10 = cv2_10 | temp[trunc(0.90 * rows(ZZZb2))];
    cv2_5 = cv2_5 | temp[trunc(0.95 * rows(ZZZb2))];
    cvi = cvi + 1;
endo;

print "___________________________ ";
print "sim " si " done - {li,tsi} = " (li~tsi);
print "stats CV 5%,10% ";
print ZZ~cv1_5~cv1_10~cv2_5~cv2_10;

/* empirical collecting of results */

outza[li+(tsi-1)*3,1:1] = ZZ;
outza[li+(tsi-1)*3,2:5] = cv1_5-cv1_10-cv2_5-cv2_10;
outza[li+(tsi-1)*3,6:8] = {sumc(YdatS[,lt 0])/rows(YdatS)};
outza[li+(tsi-1)*3,9:11] = {minc(YdatS)};
outza[li+(tsi-1)*3,12:14] = {maxc(YdatS)};
outza[li+(tsi-1)*3,15:17] = {mean2(YdatS)};
outza[li+(tsi-1)*3,18:20] = {stdc(YdatS)};
corr1 = corrX(Ydat[,1]-Hdat[,1]);
corr2 = corrX(Ydat[,2]-Hdat[,2]);
corr3 = corrX(Ydat[,3]-Hdat[,3]);
corr4 = corrX(Ydat[,4]-Hdat[,4]);
corr5 = corrX(Ydat[,5]-Hdat[,5]);
corr6 = corrX(Ydat[,6]-Hdat[,6]);
corr7 = corrX(Ydat[,7]-Hdat[,7]);
corr8 = corrX(Ydat[,8]-Hdat[,8]);
corr9 = corrX(Ydat[,9]-Hdat[,9]);
corr10 = corrX(Ydat[,10]-Hdat[,10]);
corr11 = corrX(Ydat[,11]-Hdat[,11]);
corr12 = corrX(Ydat[,12]-Hdat[,12]);
corr13 = corrX(Ydat[,13]-Hdat[,13]);
corr14 = corrX(Ydat[,14]-Hdat[,14]);
corr15 = corrX(Ydat[,15]-Hdat[,15]);
corr16 = corrX(Ydat[,16]-Hdat[,16]);
corr17 = corrX(Ydat[,17]-Hdat[,17]);
corr18 = corrX(Ydat[,18]-Hdat[,18]);
corr19 = corrX(Ydat[,19]-Hdat[,19]);
corr20 = corrX(Ydat[,20]-Hdat[,20]);
corr21 = corrX(Ydat[,21]-Hdat[,21]);
corr22 = corrX(Ydat[,22]-Hdat[,22]);
corr23 = corrX(Ydat[,23]-Hdat[,23]);

/* simulation collecting of results */

/*outza[tsi,1] = ZZ;*/
/*outza[tsi,(li-1)*4+2:(li-1)*4+5] = outza[tsi,(li-1)*4+2:(li-1)*4+5] + (ZZ .ge (cv1_5-cv1_10-cv2_5-cv2_10));*/

iss = is + 1;
print "------------------------------------------";
print " ";
print "Total Output for li and tsi = " (li~tsi);
print " ";
print "5%,10% nominal rejection rates ";
print outza[.,.]/simtot;
print " ";
print "Actual and Simulated Growth Rate (per Capita) ";

iss = 1; YdatSRT = {};
do while iss <= cols(YdatS);
    temp = sortc(YdatS, iss);
    YdatSRT = YdatSRT | temp[., iss];
    iss = iss + 1;
endo;

Page 15
iss = iss + 1;
end;

if rows(YdatS) > rows(ld_pc_Y);
  ext = rows(YdatS) - rows(ld_pc_Y);
  extr = *ones(ext, cols(ld_pc_Y));
  print (sortc([ld_pc_Y extr], 1)) ~ YdatSRT;
else;
  print (sortc([ld_pc_Y], 1)) ~ YdatSRT;
endif;
print "totals * rows(sortc(ld_pc_Y, 1)) ~ rows(YdatSRT);
print "------------------------------------------";
li = li + 1;
end;

tsii = tsi + 1;
end;

print "------------- FINAL OUTPUT ----------------";
print "Rokko Paper ----------------";
print "Column";
print "Zstat boot1 5% 10% boot2 5% 10% %<0 (each y vbl) min max mean stdc corr(y1, h1)";
print "Row";
print "ti=1 ls=1";
print "ti=1 ls=2";
print "ti=1 ls=3";
print "ti=2 ls=1 ...";
print "Actuals for %<0, min, max, mean, stderr, corr(y1, h1)";
corr1 = corr(x(ld_pc_Y ~ ld_pc_H));
corr1a = corr1[2, 1];
print ((sumc(ld_pc_Y .lt 0)) / rows(ld_pc_Y)) ~ minc(ld_pc_Y) ~ maxc(ld_pc_Y) ~ meanc(ld_pc_Y) ~ stdc(ld_pc_Y) ~ corr1a;
output off;