# Programs for "Poisson models with dynamic random effects and nonnegative credibilities per # period", by Jean Pinquet. These programs are written in R, SAS or Mathematica. They are sorted by

# the order of appearance in the paper and appendices.

# This R program derives precision matrices related to ARFIMA(0,d,0) specifications, and the related

# correlation matrices. The generalized partial correlation coefficients are derived from the

# correlation matrices, and I check the positivity of the off-diagonal coefficients.

mesh=0.01

dmax=0.4999

maxind=floor(dmax/mesh)

abs=matrix(0,maxind,1)

ord=abs

vmingpac=1

indmin=0

dmin=0

for (indic in 1:maxind){

d=mesh\*indic

abs[indic,1]=d

dim=100

lrq=matrix(1,1,dim)

for (i in 2:dim){

j=i-1

k=i-2

lrq[i]=lrq[j]\*(k+d)/(j-d)}

mrq=matrix(1,dim,dim)

for (i in 1:dim){

for (j in 1:dim){

k=abs(i-j)+1

mrq[i,j]=lrq[k]}}

imrq=solve(mrq)

dimrq=diag(imrq)

invsigrq=1/sqrt(dimrq)

dinvsigrq=diag(invsigrq,dim,dim)

corrprecision=dinvsigrq%\*%imrq%\*%dinvsigrq

for (i in 1:dim){corrprecision[i,i]=-1}

mingpac=-max(corrprecision)

if (mingpac<vmingpac){

vmingpac=mingpac

indmin=which.max(corrprecision)

dmin=d}

ord[indic,1]=mingpac

}

d=abs

mingpac=ord

plot(d,mingpac)

c(vmingpac,indmin,dmin)

# This R program verifies the compatibility of the AR(1) specifications of type N1, N2 or S (positivity # types are equivalent in that case: the autocorrelation coefficients must be nonnegative) with

# stationary log-Gaussian sequences.

# Example with 10 periods: lines 57-87.

phi1=0.5

gam=2

length=10

lru[1]=phi1

for (h in 2:length){

lru[h]=lru[h-1]\*phi1}

lru

lrub=log(1+(gam\*lru))/log(1+gam)

# lrub: sequence for which the admissibility as an autocorrelation function is tested for (from h=1) lrub

# Levinson-Durbin recursion applied on lrub

pacub=lrub

sin2=1-(pacub[1]\*\*2)

vphi=matrix(pacub[1],1,1)

vrho=matrix(vphi,1,1)

rvphi=vphi

for (i in 2:length){

pscal=t(rvphi)%\*%vrho

pacub[i]=(lrub[i]-pscal)/sin2

pacs=pacub[i]

sin2=sin2\*(1-(pacs\*\*2))

vphib=vphi-(pacs\*rvphi)

vphi=rbind(vphib,pacs)

rvphi=apply(vphi,2,rev)

vrho=rbind(vrho,lrub[i])}

pacub

# pacub: is the line of partial autocorrelation coefficients

# AR(1): general case. Loop on the variance of the random effect and on phi1.

# Initialization

vmax=0.9999

phimesh=0.01

maxi1=floor(vmax/phimesh)

gammesh=0.01

gammax=5

indgammax=floor(gammax/gammesh)

length=10

lru=matrix(1,1,length)

nindmax=matrix(0,1,length)

maxpac=0

phi1opt=0

gamopt=1

lru=matrix(1,1,length)

for (i1 in 0:maxi1){

phi1=i1\*phimesh

lru[1]=phi1

for (h in 2:length){

lru[h]=lru[h-1]\*phi1}

for (ngam in 1: indgammax){

gam=ngam\*gammesh

lrub=log(1+(gam\*lru))/log(1+gam)

# lrub is the autocorrelation function of the autocovariance function transformed by log(1+)

pacub=lrub

sin2=1-(pacub[1]\*\*2)

vphi=matrix(pacub[1],1,1)

vrho=matrix(vphi,1,1)

rvphi=vphi

for (i in 2:length){

pscal=t(rvphi)%\*%vrho

pacub[i]=(lrub[i]-pscal)/sin2

pacs=pacub[i]

sin2=sin2\*(1-(pacs\*\*2))

vphib=vphi-(pacs\*rvphi)

vphi=rbind(vphib,pacs)

rvphi=apply(vphi,2,rev)

vrho=rbind(vrho,lrub[i])}

abspacub=abs(pacub)

maxpact=max(abspacub)

indice=which.max(abspacub)

nindmax[indice]=nindmax[indice]+1

if (maxpact>maxpac) {phi1opt=phi1}

if (maxpact>maxpac) {gamopt=gam}

if (maxpact>maxpac) {maxpac=maxpact}

}}

# results

maxpac

nindmax

c(maxpac,phi1opt,gamopt)

# maxpac is the maximum of the absolute value of the partial autocorrelation function

# The partial autocorrelation function has been derived with the Levinson-Durbin recursion.

# nindmax[h]: number of partial autocorrelation sequences where the maximum absolute value reaches the lag h.

# c (maxpac,phi1opt,gamopt): maximum absolute value and related parameters

/\* AR(2): comprehensive study \*/

# initialization

phimesh=0.01

gammesh=1

length=5

gammax=5

vmax=1.9999

maxi1=floor(vmax/phimesh)

phi1opt=0

phi2opt=0

gamopt=1

maxpac=0

indgammax=floor(gammax/gammesh)

lru=matrix(1,1,length)

nindmax=matrix(0,1,length)

# loop

for (i1 in 0:maxi1){

phi1=i1\*phimesh

maxi2=maxi1-i1-(1/phimesh)

minphi2=-0.25\*phi1\*phi1/phimesh

mini2=-floor(-minphi2)

if (maxi2>=1){

for (i2 in mini2:maxi2){

phi2=(i2\*phimesh)

matphi=matrix(0,2,2)

matphi[1,1]=1-phi2

matphi[2,1]=-phi1

matphi[2,2]=1

vphi=matrix(0,2,1)

vphi[1,]=phi1

vphi[2,]=phi2

vrho=solve(matphi,vphi)

lru[,1:2]=t(vrho)

for (i in 3:length){

j=i-1

k=i-2

lru[,i]=(phi1\*lru[j])+(phi2\*lru[k])}

for (ngam in 1:indgammax){

gam=ngam\*gammesh

lrub=log(1+(gam\*lru))/log(1+gam)

pacub=lrub

sin2=1-(pacub[1]\*\*2)

vphi=matrix(pacub[1],1,1)

vrho=matrix(vphi,1,1)

rvphi=vphi

for (i in 2:length){

pscal=t(rvphi)%\*%vrho

pacub[i]=(lrub[i]-pscal)/sin2

pacs=pacub[i]

sin2=sin2\*(1-(pacs\*\*2))

vphib=vphi-(pacs\*rvphi)

vphi=rbind(vphib,pacs)

rvphi=apply(vphi,2,rev)

vrho=rbind(vrho,lrub[i])}

abspacub=abs(pacub)

maxpact=max(abspacub)

indice=which.max(abspacub)

nindmax[indice]=nindmax[indice]+1

if (maxpact>maxpac) {phi1opt=phi1}

if (maxpact>maxpac) {phi2opt=phi2}

if (maxpact>maxpac) {gamopt=gam}

if (maxpact>maxpac) {maxpac=maxpact}

}}}}

# results

nindmax

c(maxpac,phi1opt,phi2opt,gamopt)

# nindmax[h]: number of partial autocorrelation sequences where the maximum absolute value reaches the lag h.

# c (maxpac,phi1opt,phi2opt,gamopt): maximum absolute value and related parameters (there are ties actually)

# This R program verifies the compatibility of the AR(3) specifications of type N2 (nonnegative linear # filtering, see Proposition 5) with log-normal distributions. Lines 97 to 159.

vmax=0.9999

mesh=0.01

maxi1=floor(vmax/mesh)

length=100

lru=matrix(1,1,length)

maxpac=0

npoints=0

totprem=0

for (i1 in 0:maxi1){

phi1=i1\*mesh

maxi2=maxi1-i1

if (maxi2>=0){

for (i2 in 0:maxi2){

phi2=(i2\*mesh)

maxi3=maxi1-(i1+i2)

if (maxi3>=1){

for (i3 in 0:maxi3){

phi3=(i3\*mesh)

npoints=npoints+1

matphi=matrix(0,3,3)

matphi[1,1]=1-phi2

matphi[1,2]=-phi3

matphi[2,1]=-(phi1+phi3)

matphi[2,2]=1

matphi[3,1]=-phi2

matphi[3,2]=-phi1

matphi[3,3]=1

vphi=matrix(0,3,1)

vphi[1,]=phi1

vphi[2,]=phi2

vphi[3,]=phi3

vrho=solve(matphi,vphi)

lru[,1:3]=t(vrho)

for (i in 4:length){

j=i-1

k=i-2

l=i-3

lru[,i]=(phi1\*lru[j])+(phi2\*lru[k])+(phi3\*lru[l])}

for (ngam in 1:10){

gam=ngam/5

lrub=log(1+(gam\*lru))/log(1+gam)

pacub=lrub

sin2=1-(pacub[1]\*\*2)

vphi=matrix(pacub[1],1,1)

pacm=lrub[1]

vrho=matrix(vphi,1,1)

rvphi=vphi

indprem=1

for (i in 2:length){

pscal=t(rvphi)%\*%vrho

pacub[i]=(lrub[i]-pscal)/sin2

pacs=pacub[i]

if (pacs>pacm) {indprem=0}

sin2=sin2\*(1-(pacs\*\*2))

vphib=vphi-(pacs\*rvphi)

vphi=rbind(vphib,pacs)

rvphi=apply(vphi,2,rev)

vrho=rbind(vrho,lrub[i])}

maxpact=max(abs(pacub))

if (maxpact>maxpac) {maxpac=maxpact}}

totprem=totprem+indprem

}}}}}

Maxpac

# Compatibility of ARFIMA(0,d,0) specifications with log-Gaussian sequences

length=100

dopt=0

gamopt=1

maxpac=0

gammesh=0.01

gammax=5

indgammax=floor(gammax/gammesh)

nindmax=matrix(0,1,length)

lru=matrix(1,1,length)

npositive=0

ndecreases=0

for (nd in 1:49){

d=nd/100

lru[1]=d/(1-d)

for (i in 2:length){

j=i-1

lru[i]=lru[i-1]\*(j+d)/(i-d)}

for (ngam in 1:indgammax){

gam=ngam\*gammesh

lrub=log(1+(gam\*lru))/log(1+gam)

pacub=lrub

sin2=1-(pacub[1]\*\*2)

vphi=matrix(pacub[1],1,1)

vrho=matrix(vphi,1,1)

rvphi=vphi

decrease=1

for (i in 2:length){

pscal=t(rvphi)%\*%vrho

pacub[i]=(lrub[i]-pscal)/sin2

j=i-1

if (pacub[i]>=pacub[j]){decrease=0}

pacs=pacub[i]

sin2=sin2\*(1-(pacs\*\*2))

vphib=vphi-(pacs\*rvphi)

vphi=rbind(vphib,pacs)

rvphi=apply(vphi,2,rev)

vrho=rbind(vrho,lrub[i])}

ndecreases=ndecreases+decrease

minpacub=min(pacub)

abspacub=abs(pacub)

minpacub=min(pacub)

if (minpacub>0) {npositive=npositive+1}

maxpact=max(abspacub)

indice=which.max(abspacub)

nindmax[indice]=nindmax[indice]+1

if (maxpact>maxpac) {dopt=d}

if (maxpact>maxpac) {gamopt=gam}

if (maxpact>maxpac) {maxpac=maxpact}

}}

ntrials=sum(nindmax)

nindmax

c(dopt,gamopt,maxpac)

c(ntrials,npositive,ndecreases)

# Last line: the 24500 trials are related to positive and strictly decreasing sequences of partial autocorrelation coefficients at the horizon of a century.

# The exponentiation entrywise of stationary Gaussian vectors that follow ARFIMA(0,d,0)

# specifications reach level S in the strict sense

mesh=0.01

dmax=0.4999

maxind=floor(dmax/mesh)

vmingpac=1

indmin=0

dmin=0

varmin=0

for (indic in 1:maxind){

vareff=0.01

# vareff: variance of the random effect

for (indvar in 1:10){

vareff=2\*vareff

d=mesh\*indic

dim=100

lrq=matrix(1,1,dim)

for (i in 2:dim){

j=i-1

k=i-2

lrq[i]=lrq[j]\*(k+d)/(j-d)}

lrq=(exp(vareff\*lrq)-1)/(exp(vareff)-1)

mrq=matrix(1,dim,dim)

for (i in 1:dim){

for (j in 1:dim){

k=abs(i-j)+1

mrq[i,j]=lrq[k]}}

imrq=solve(mrq)

dimrq=diag(imrq)

invsigrq=1/sqrt(dimrq)

dinvsigrq=diag(invsigrq,dim,dim)

corrprecision=dinvsigrq%\*%imrq%\*%dinvsigrq

for (i in 1:dim){corrprecision[i,i]=-1}

mingpac=-max(corrprecision)

if (mingpac<vmingpac){

vmingpac=mingpac

indmin=which.max(corrprecision)

dmin=d

varmin=vareff}

}}

c(vmingpac,indmin,dmin,varmin)

# The exponentiation entrywise of stationary Gaussian vectors that follow AR(p)

# specifications do not reach level S in the strict sense, but almost do so.

rho=0.5

vareff=1

vmingpac=1

indmin=0

lengthmin=0

maxlength=100

abs=matrix(0,maxlength,1)

ord=abs

for (length in 2:maxlength){

lrq=matrix(1,1,length)

abs[length,1]=length

for (i in 2:length){

j=i-1

lrq[i]=lrq[j]\*rho}

lrq=(exp(vareff\*lrq)-1)/(exp(vareff)-1)

mrq=matrix(1,length,length)

for (i in 1:length){

for (j in 1:length){

k=abs(i-j)+1

mrq[i,j]=lrq[k]}}

imrq=solve(mrq)

dimrq=diag(imrq)

invsigrq=1/sqrt(dimrq)

dinvsigrq=diag(invsigrq,length,length)

corrprecision=dinvsigrq%\*%imrq%\*%dinvsigrq

for (i in 1:length){corrprecision[i,i]=-1}

mingpac=-max(corrprecision)

if (mingpac<vmingpac){

vmingpac=mingpac

indmin=which.max(corrprecision)

lengthmin=length}

ord[length,1]=mingpac

}

length=abs

mingpac=ord

plot(length,mingpac)

c(vmingpac,indmin,lengthmin)

# This R program verifies the condition of Proposition 8 that extends the positivity property of the # dynamic component of the random effect if a time-invariant component is added. The verification # is achieved for the ARFIMA(0,d,0) specifications.

mesh=0.01

dmax=0.4999

indmax=floor(dmax/mesh)

length=100

dmin=0

valmin=100

intercept=matrix(1,length,1)

matres=matrix(0,indmax,3)

for (indic in 1:indmax){

d=mesh\*indic

matres[indic,1]=d

lrq=matrix(1,1,length)

mrq=matrix(0,length,length)

for (i in 2:length){

j=i-1

k=i-2

lrq[i]=lrq[j]\*(k+d)/(j-d)}

for (i in 1:length){

for (j in 1:length){

k=abs(i-j)+1

mrq[i,j]=lrq[k]}}

imrq=solve(mrq)

prod=imrq%\*%intercept

matres[indic,2]=min(prod)

matres[indic,3]=which.min(prod)}

d=matres[,1]

minprod=matres[,2]

plot(d,minprod)

d=0.49999

length=100

lrq=matrix(1,1,length)

mrq=matrix(0,length,length)

for (i in 2:length){

j=i-1

k=i-2

lrq[i]=lrq[j]\*(k+d)/(j-d)}

for (i in 1:length){

for (j in 1:length){

k=abs(i-j)+1

mrq[i,j]=lrq[k]}}

imrq=solve(mrq)

prod=imrq%\*%intercept

min(prod)

plot(prod)

# This R program verifies the condition of Proposition 8 that extends the positivity property of the # dynamic component of the random effect if a time-invariant component is added. The verification # is achieved for the ARFIMA(0,d,0) specifications exponentiated entrywise.

mesh=0.01

dmax=0.4999

indmax=floor(dmax/mesh)

length=100

dmin=0

valmin=100

varmin=0

prodmin=1

intercept=matrix(1,length,1)

matres=matrix(0,indmax,3)

for (indic in 1:indmax){

d=mesh\*indic

vareff=0.05

for (indvar in 1:6){

vareff=2\*vareff

lrq=matrix(1,1,length)

mrq=matrix(0,length,length)

for (i in 2:length){

j=i-1

k=i-2

lrq[i]=lrq[j]\*(k+d)/(j-d)}

lrq=(exp(vareff\*lrq)-1)/(exp(vareff)-1)

for (i in 1:length){

for (j in 1:length){

k=abs(i-j)+1

mrq[i,j]=lrq[k]}}

imrq=solve(mrq)

prod=imrq%\*%intercept

minprod=min(prod)

if (minprod<prodmin){

dmin=d

prodmin=minprod

varmin=vareff

locmin=which.min(prod)

}

}}

c(dmin,prodmin,varmin,locmin)

# This R program generates the table in Section 6 that verifies the limit credibility formula,

# with AR(1) specifications on the random effect

length=100

lambda=1/15

rho=0.8

gamq0=1

gamqnorml1=gamq0\*(1+rho)/(1-rho)

gamx0=(1/lambda)+gamq0

sx0=(1/lambda)+gamqnorml1

mult=(lambda\*gamq0)

multx=mult/(1+mult)

num=1+(lambda\*gamq0)

lrq=matrix(1,1,length)

lrq[1]=rho

for (i in (2:length)) {lrq[i]=lrq[i-1]\*rho}

lrx=multx\*lrq

sin2=1-(lrx[1]\*\*2)

vphi=matrix(lrx[1],1,1)

vrho=matrix(lrx[1],1,1)

tcred=lrx[1]

rvphi=vphi

# Levinson-Durbin

for (i in 2:length){

pscal=sum(rvphi\*vrho)

pacs=(lrx[i]-pscal)/sin2

sin2=sin2\*(1-(pacs\*\*2))

tcred=pacs+(tcred\*(1-pacs))

vphib=vphi-(pacs\*rvphi)

vphi=rbind(vphib,pacs)

rvphi=apply(vphi,2,rev)

vrho=rbind(vrho,lrx[i])}

gamres0=gamx0\*sin2

dist2=(1-tcred)\*(1-tcred)

sres0=sx0\*dist2

result=c(length,tcred,sin2,gamres0,sres0)

result

/ \* This SAS program provides unconstrained estimations of autocovariances such as those given in Table 1 in the case study (Section 7). The data are not disclosed. The three variables are described in the companion file. \*/

/\* semiparametric estimation of the variance of the random effect

at the disaggregated level (in Table 1, this is the value that

corresponds to h=0). It corresponds to a short term effect in

the prediction \*/

proc iml;

libname matrice 'C:\Donnees\Mapfre\ASTIN2019';

reset storage='matrice.matrbis';

load n prfreq period;

ntot=n[+];

res=n-prfreq;

varu=((res`\*res)-ntot)/(prfreq`\*prfreq);

print varu;

/\* The program that follows derives the estimated autocovariance

function of Table 1. The program derives two lines vectors "lnum"

and "lden", the entrywise ratio of which is the line vector "lcov"

of estimated autocovariances. \*/

tmax=max(period);

nobs=nrow(n);

lnum=repeat(0,1,tmax);

lden=lnum;

nperiod=lnum;

nper=0;

imax=nobs;

do i=1 to imax;

if i=1 then

do;

/\* Initialization \*/

nper=1; lres=res[i,]; lprim=prfreq[i,];

end;

if (period[i]=1 & i>1) then

do;

/\* Output \*/

dymo=(2\*nper)-1;

vres=lres`;

lresaug=repeat(0,1,dymo);

lresaug[,1:nper]=lres;

matresaug=repeat(0,dymo,tmax);

do j=1 to nper;

k=j+nper-1;

matresaug[j:k,j]=vres;

end;

lnum=lnum+(lresaug\*matresaug);

vprim=lprim`;

lprimaug=repeat(0,1,dymo);

lprimaug[,1:nper]=lprim;

matprimaug=repeat(0,dymo,tmax);

do j=1 to nper;

k=j+nper-1;

matprimaug[j:k,j]=vprim;

end;

lden=lden+(lprimaug\*matprimaug);

/\* Initialisation \*/

nper=1; lres=res[i,]; lprim=prfreq[i,];

end;

if period[i]>1 then

/\* Mise à jour \*/

do;

nper=nper+1; lres=lres||res[i,]; lprim=lprim||prfreq[i,];

end;

if i=imax then

do;

/\* output \*/

dymo=(2\*nper)-1;

vres=lres`;

lresaug=repeat(0,1,dymo);

lresaug[,1:nper]=lres;

matresaug=repeat(0,dymo,tmax);

do j=1 to nper;

k=j+nper-1;

matresaug[j:k,j]=vres;

end;

lnum=lnum+(lresaug\*matresaug);

vprim=lprim`;

lprimaug=repeat(0,1,dymo);

lprimaug[,1:nper]=lprim;

matprimaug=repeat(0,dymo,tmax);

do j=1 to nper;

k=j+nper-1;

matprimaug[j:k,j]=vprim;

end;

lden=lden+(lprimaug\*matprimaug);

end;

end;

lnum[1]=lnum[1]-ntot;

lvu=lnum/lden;

print lnum lden lvu;

# This R program derives the estimations of Table 2. Individual values are avoided, and

# the only input is the estimated autocovariance function.

lvu=cbind(1.269, 0.802, 0.615, 0.586, 0.553, 0.457, 0.442)

# mlvu: estimation of the variance for a time-invariant random effect

mlvu=mean(lvu)

mlvu

# estimation if Q is white noise (AR(0)): lvu needs to be defined

varp=mean(lvu[1,2:7])

varq=(lvu[1,1]-varp)/(1+varp)

c(varp,varq)

# estimation if Q is AR(1): lvu needs to be defined

phimax=0.9999

varpmax=1

varqmax=1

phimesh=0.01

varpmesh=0.01

varqmesh=0.01

maxi1=floor(phimax/phimesh)

maxi2=floor(varpmax/varpmesh)

maxi3=floor(varqmax/varqmesh)

lru=matrix(1,1,7)

lrq=lru

distopt=1

for (i1 in 0:maxi1){

for (i2 in 0:maxi2){

for (i3 in 0:maxi3){

phi1=i1\*phimesh

varp=i2\*varpmesh

varq=i3\*varqmesh

for (h in 2:7){

h1=h-1

lrq[h]=phi1\*lrq[h1]}

lvufit=varp+((1+varp)\*varq\*lrq)

dist2=sum((lvu-lvufit)^2)

if (dist2<distopt){

distopt=dist2

varpopt=varp

varqopt=varq

phi1opt=phi1

}

}}}

c(varpopt,varqopt,phi1opt,distopt)

# estimation if Q is ARFIMA(0,d,0): lvu needs to be defined

dmax=0.4999

varpmax=1

varqmax=2

dmesh=0.01

varpmesh=0.01

varqmesh=0.01

maxi1=floor(dmax/dmesh)

maxi2=floor(varpmax/varpmesh)

maxi3=floor(varqmax/varqmesh)

lru=matrix(1,1,7)

lrq=lru

distopt=1

for (i1 in 0:maxi1){

for (i2 in 0:maxi2){

for (i3 in 0:maxi3){

d=i1\*dmesh

varp=i2\*varpmesh

varq=i3\*varqmesh

for (h in 2:7){

h1=h-1

lrq[h]=(d+h-2)\*lrq[h1]/(h1-d)}

lvufit=varp+((1+varp)\*varq\*lrq)

dist2=sum((lvu-lvufit)^2)

if (dist2<distopt){

distopt=dist2

varpopt=varp

varqopt=varq

dopt=d

}

}}}

c(varpopt,varqopt,dopt,distopt)

# estimation if Q is ARFIMA(0,d,0) exponentiated entrywise: lvu needs to be defined

dmax=0.4999

varpmax=1

varqmax=2

dmesh=0.01

varpmesh=0.01

varqmesh=0.01

maxi1=floor(dmax/dmesh)

maxi2=floor(varpmax/varpmesh)

maxi3=floor(varqmax/varqmesh)

lru=matrix(1,1,7)

lrq=lru

distopt=1

for (i1 in 0:maxi1){

for (i2 in 0:maxi2){

for (i3 in 0:maxi3){

d=i1\*dmesh

varp=i2\*varpmesh

varq=i3\*varqmesh

for (h in 2:7){

h1=h-1

lrq[h]=(d+h-2)\*lrq[h1]/(h1-d)}

lvq=varq\*lrq

lvq=exp(lvq)-1

lvufit=varp+((1+varp)\*lvq)

dist2=sum((lvu-lvufit)^2)

if (dist2<distopt){

distopt=dist2

varpopt=varp

varqopt=varq

dopt=d

}

}}}

c(varpopt,varqopt,dopt,distopt)

# Estimation if Q is AR(2): lvu needs to be defined

opt1t=0.47

opt2t=0.54

opt3t=0.45

opt4t=0

valmin=10

mesh=0.01

for (iterat in 1:20){

opt1=opt1t

opt2=opt2t

opt3=opt3t

opt4=opt4t

for (ip in -1:1){

for (iq in -1:1){

for (i1 in -1:1){

for (i2 in -1:1){

varp=opt1+(ip\*mesh)

varq=opt2+(iq\*mesh)

phi1=opt3+(i1\*mesh)

phi2=opt4+(i2\*mesh)

matphi=matrix(0,2,2)

matphi[1,1]=1-phi2

matphi[2,1]=-phi1

matphi[2,2]=1

vphi=matrix(0,2,1)

vphi[1,]=phi1

vphi[2,]=phi2

vrho=solve(matphi,vphi)

lvust=matrix(0,1,7)

lvqst=lvust

lvqst[1]=varq

lrho=t(vrho)

lvqst[,2:3]=varq\*lrho

for (i in 4:7){

j=i-1

k=i-2

lvqst[,i]=(phi1\*lvqst[j])+(phi2\*lvqst[k])}

lvust=varp+((1+varp)\*lvqst)

diff2=lvu-lvust

dist2=sum(diff2\*diff2)

somphi=sum(vphi)

valid=1

if (somphi>=1) {valid=0}

if (phi1<0) {valid=0}

if (phi2<0) {valid=0}

if ((dist2<valmin) & (valid=1)){

valmin=dist2

opt1t=varp

opt2t=varq

opt3t=phi1

opt4t=phi2

results=c(varp, varq, phi1, phi2, valmin)

}}}}}}

results

# Estimation if Q is AR(3): lvu needs to be defined

opt1t=0.45

opt2t=0.56

opt3t=0.40

opt4t=0.07

opt5t=0

valmin=10

mesh=0.01

for (iterat in 1:10){

opt1=opt1t

opt2=opt2t

opt3=opt3t

opt4=opt4t

opt5=opt5t

for (ip in -4:4){

for (iq in -4:4){

for (i1 in -4:4){

for (i2 in -4:4){

for (i3 in -4:4){

varp=opt1+(ip\*mesh)

varq=opt2+(iq\*mesh)

phi1=opt3+(i1\*mesh)

phi2=opt4+(i2\*mesh)

phi3=opt5+(i3\*mesh)

matphi=matrix(0,3,3)

matphi[1,1]=1-phi2

matphi[1,2]=-phi3

matphi[2,1]=-(phi1+phi3)

matphi[2,2]=1

matphi[3,1]=-phi2

matphi[3,2]=-phi1

matphi[3,3]=1

vphi=matrix(0,3,1)

vphi[1,]=phi1

vphi[2,]=phi2

vphi[3,]=phi3

vrho=solve(matphi,vphi)

lvust=matrix(0,1,7)

lvqst=lvust

lvqst[1]=varq

lrho=t(vrho)

lvqst[,2:4]=varq\*lrho

for (i in 5:7){

j=i-1

k=i-2

l=i-3

lvqst[,i]=(phi1\*lvqst[j])+(phi2\*lvqst[k])+(phi3\*lvqst[l])}

lvust=varp+((1+varp)\*lvqst)

diff3=lvu-lvust

dist3=sum(diff3\*diff3)

valid=1

somphi=sum(vphi)

ctr1=phi1-((phi1\*phi2)+(phi2\*phi3))

ctr2=phi2-(phi1\*phi3)

if (somphi>=1) {valid=0}

if (phi1<0) {valid=0}

if (phi2<0) {valid=0}

if (phi3<0) {valid=0}

if (ctr1<0) {valid=0}

if (ctr2<0) {valid=0}

if ((dist3<valmin) & (valid==1)){

valmin=dist3

opt1t=varp

opt2t=varq

opt3t=phi1

opt4t=phi2

opt5t=phi3

results=c(varp, varq, phi1, phi2, phi3, valmin)

}}}}}}}

results

# This R program verifies that the limit credibility equals one for an ARFIMA(0,d,0) specification,

# with an equivalence of the difference between total and limit credibility.

length=1000

lambda=0.07

d=0.37

gamq0=1.28

lrq=matrix(1,1,length)

loglength=matrix(0,length,1)

logdist=loglength

lrq[1]=d/(1-d)

for (i in 2:length){

j=i-1

k=i-2

lrq[i]=lrq[j]\*(j+d)/(i-d)}

mult=(lambda\*gamq0)

multx=mult/(1+mult)

lrx=multx\*lrq

logdist[1,1]=log(1-lrx[1])

sin2=1-(lrx[1]\*\*2)

vphi=matrix(lrx[1],1,1)

vrho=matrix(lrx[1],1,1)

tcred=lrx[1]

rvphi=vphi

# Levinson-Durbin

for (i in 2:length){

loglength[i,1]=log(i)

pscal=sum(rvphi\*vrho)

pacs=(lrx[i]-pscal)/sin2

sin2=sin2\*(1-(pacs\*\*2))

tcred=pacs+(tcred\*(1-pacs))

logdist[i,1]=log(1-tcred)

vphib=vphi-(pacs\*rvphi)

vphi=rbind(vphib,pacs)

rvphi=apply(vphi,2,rev)

vrho=rbind(vrho,lrx[i])}

plot(loglength,logdist)

x1=loglength[100,1]

x2=loglength[1000,1]

y1=logdist[100,1]

y2=logdist[1000,1]

slope=(y2-y1)/(x2-x1)

intercept=y1-(slope\*x1)

c(exp(intercept),slope)

# This R program derives the total credibility as a function of the length, for the ARFIMA(0,d,0)

# specification in Figure 1

lambda=0.07

varp=0

varq=1.28

d=0.37

length=40

lrq=matrix(0,1,40)

lrq[,1]=d/(1-d)

for (i in 2:length){

j=i-1

lrq[,i]=lrq[,j]\*(j+d)/(i-d)}

lvq=varq\*lrq

lvu=varp+((1+varp)\*lvq)

lvu0=varp+((1+varp)\*varq)

lru=lvu/lvu0

mult=lambda\*lvu0/(1+(lambda\*lvu0))

lrx=mult\*lru

pacx=lrx

ltcred=pacx

tcred=pacx[1]

sin2=1-(pacx[1]\*\*2)

vphi=matrix(pacx[1],1,1)

vrho=vphi

rvphi=vphi

for (i in 2:40){

pscal=t(rvphi)%\*%vrho

pacx[i]=(lrx[i]-pscal)/sin2

pacs=pacx[i]

tcred=pacs+((1-pacs)\*tcred)

ltcred[i]=tcred

sin2=sin2\*(1-(pacs\*\*2))

vphib=vphi-(pacs\*rvphi)

vphi=rbind(vphib,pacs)

rvphi=apply(vphi,2,rev)

vrho=rbind(vrho,lrx[i])}

ltcred

/\* This SAS program compares two predictors based on dynamic random effects

to the usual predictor, with respect to second order stochastic dominance.

Estimation has been rerun from histories restricted to 6 periods.

The graph of integrated differences of quantiles is given in Picture 2. Then,

ROC curves and related Gini and Kuznets indices are derived. \*/

proc iml;

libname matrice 'C:\Donnees\Mapfre\ASTIN2019';

reset storage='matrice.matrbis';

load n prfreq period;

ntot=n[+];

res=n-prfreq;

varu=((res`\*res)-ntot)/(prfreq`\*prfreq);

print varu;

maxper=max(period);

res=n-prfreq ;

nobs=nrow(n);

/\* imax=nrow(n); print imax; \*/

ntot=n[+];

/\* Derivation of variance-covariance matrices of random effects.

The estimation of AR(1) and ARFIMA(0,d,0) has been rerun from

histories restricted to 6 periods \*/

varpa=0.70;

/\* a: Q=1 ; b: Q is AR(1) ; c: Q is ARFIMA(0,d,0) \*/

varp=0.47;

varq=0.54;

phi1=0.43;

lrq=repeat(1,1,7);

do i=2 to 7;

j=i-1;

lrq[1,i]=lrq[1,j]\*phi1;

end;

lvq=varq\*lrq;

lvub=varp+((1+varp)\*lvq);

matvub=repeat(0,6,6);

do i=1 to 6;

do j=1 to 6;

indice=1+abs(j-i);

matvub[i,j]=lvub[1,indice];

end;

end;

lvub=lvub[,2:7];

lphib=lvub\*inv(matvub);

varp=0;

varq=1.27;

d=0.37;

do i=2 to 7;

j=i-1; k=i-2;

lrq[1,i]=lrq[1,j]\*(k+d)/(j-d);

end;

lvq=varq\*lrq;

lvuc=varp+((1+varp)\*lvq);

matvuc=repeat(0,6,6);

do i=1 to 6;

do j=1 to 6;

indice=1+abs(j-i);

matvuc[i,j]=lvuc[1,indice];

end;

end;

lvuc=lvuc[,2:7];

lphic=lvuc\*inv(matvuc);

/\* Derivation of predictors, with three specifications:

a: Q=1; b: Q: AR(1) ; c: Q: ARFIMA(0,d,0) \*/

npol=0 ;

do i=1 to nobs;

ni=n[i,]; primfreq=prfreq[i,]; pepere=period[i,];

if pepere=1 then

do;

/\* initialization \*/

histn=ni;

exporisk=primfreq;

end; else

do;

/\* updating \*/

If pepere<7 then

do;

histn=histn||ni;

exporisk=exporisk||primfreq;

end;

end;

if pepere=7 then

do;

/\* output \*/

if npol=0 then

do;

npol=1;

line=repeat(0,1,5);

linpreda=(1+(varpa\*histn[,+]))/(1+(varpa\*exporisk[,+]));

lcredb=lvub\*inv(matvub+diag(1/exporisk));

fixeffects=histn/exporisk;

fixeffects=fixeffects[,6:1];

/\* The history must be reverted in the derivation of the predictor.

Otherwise, you can revert the updating formulas \*/

linpredb=1+(lcredb\*((fixeffects-1)`));

lcredc=lvuc\*inv(matvuc+diag(1/exporisk));

linpredc=1+(lcredc\*((fixeffects-1)`));

line[,1]=ni;

line[,2]=primfreq;

line[,3]=linpreda;

line[,4]=linpredb;

line[,5]=linpredc;

matout=line;

end; else

do;

npol=npol+1;

linpreda=(1+(varpa\*histn[,+]))/(1+(varpa\*exporisk[,+]));

lcredb=lvub\*inv(matvub+diag(1/exporisk));

fixeffects=histn/exporisk;

fixeffects=fixeffects[,6:1];

linpredb=1+(lcredb\*((fixeffects-1)`));

lcredc=lvuc\*inv(matvuc+diag(1/exporisk));

linpredc=1+(lcredc\*((fixeffects-1)`));

line[,1]=ni;

line[,2]=primfreq;

line[,3]=linpreda;

line[,4]=linpredb;

line[,5]=linpredc;

matout=matout//line;

end;

end;

/\* end of the output step if period=7 \*/

end;

noms={'nclaims' 'prior' 'lpreda' 'lpredb' 'lpredc'};

create one from matout[colname=noms];

append from matout;

close one;

libname perm 'C:\Jean\Articles\Mes\_papiers\_soumis\Ergodicity\ASTIN\_first';

%let nobs=80994;

data perm.comparison;

set one;

data two;

set one;

posta=prior\*lpreda;

postb=prior\*lpredb;

postc=prior\*lpredc;

proc means; var lpreda lpredb lpredc;

/\* Results are reported Table 4 \*/

%let nobs=80994;

data temp;

set one(keep=lpreda);

proc sort; by lpreda;

data roca;

set temp;

scale=100;

if mod(\_n\_,scale)=0 or \_n\_=1 or \_n\_=&nobs then

do;

freqa=(\_n\_/&nobs);

keep freqa lpreda;

output;

end;

data temp;

set one(keep=lpredb);

proc sort; by lpredb;

data rocb;

set temp;

scale=100;

if mod(\_n\_,scale)=0 or \_n\_=1 or \_n\_=&nobs then

do;

freqb=(\_n\_/&nobs);

keep freqb lpredb;

output;

end;

data temp;

set one(keep=lpredc);

proc sort; by lpredc;

data rocc;

set temp;

scale=100;

if mod(\_n\_,scale)=0 or \_n\_=1 or \_n\_=&nobs then

do;

freqc=(\_n\_/&nobs);

keep freqc lpredc;

output;

end;

data jointure;

merge roca rocb rocc;

diffba=lpredb-lpreda;

diffca=lpredc-lpreda;

keep freqa diffba diffca lpreda lpredb lpredc;

data intquant;

set jointure;

retain intqba intqca 0;

lfreqa=lag(freqa);

if \_n\_ gt 1 then

do;

intqba=intqba+((freqa-lfreqa)\*diffba);

intqca=intqca+((freqa-lfreqa)\*diffca);

keep freqa intqba intqca lpreda lpredb lpredc;

output;

end;

data \_null\_;

set intquant;

file 'C:\Jean\Articles\Mes\_papiers\_soumis\Ergodicity\ASTIN\_first\DS2.txt';

put freqa f7.4 @9 intqba f7.4 @17 intqca f7.4;

data three;

set one(keep=nclaims prior lpreda);

posta=prior\*lpreda;

proc sort; by descending posta;

data four vier;

set three;

retain abs 0;

retain ord 0;

retain Gini 0;

/\* Pseudo Lorenz curve: ROC curve \*/

abs=abs+(1/80994);

ord=ord+(nclaims/6042);

diff=ord-abs;

Gini=Gini+(2\*diff/80994);

if \_n\_=80994 then output vier;

output four;

proc means data=four; var diff;

/\* The maximum value of diff is the Kuznets index \*/

proc print data=vier;

/\* Gini index \*/

data five;

set one(keep=nclaims prior lpredb);

postb=prior\*lpredb;

proc sort; by descending postb;

data six sechs;

set five;

retain abs 0;

retain ord 0;

retain Gini 0;

/\* Pseudo Lorenz curve: ROC curve \*/

abs=abs+(1/80994);

ord=ord+(nclaims/6042);

diff=ord-abs;

Gini=Gini+(2\*diff/80994);

if \_n\_=80994 then output sechs;

output six;

proc means data=six; var diff;

/\* The maximum value of diff is the Kuznets index \*/

proc print data=sechs;

/\* Gini index \*/

data seven;

set one(keep=nclaims prior lpredc);

postc=prior\*lpredc;

proc sort; by descending postc;

data eight acht;

set seven;

retain abs 0;

retain ord 0;

retain Gini 0;

/\* Pseudo Lorenz curve: ROC curve \*/

abs=abs+(1/80994);

ord=ord+(nclaims/6042);

diff=ord-abs;

Gini=Gini+(2\*diff/80994);

if \_n\_=80994 then output acht;

output eight;

proc means data=eight; var diff;

/\* The maximum values of diff is the Kuznets index \*/

proc print data=acht;

/\* Gini index \*/

run;

**data** uno;

set one;

posta=prior\*lpreda;

postb=prior\*lpredb;

postc=prior\*lpredc;

lva=(nclaims\*log(posta))-posta;

lvb=(nclaims\*log(postb))-postb;

lvc=(nclaims\*log(postc))-postc;

**proc** **means** sum; var lva lvb lvc;

run;

/\* This SAS program derives efficient GMM estimators from given frequency premiums

derived at the individual level (individual: pair (policyholder, period)).

Estimations are performed for time-invariant and dynamic random effects with an AR(1) or

ARFIMA(0,d,0) ergodic component. The mathematics are given in Section A.3, and comments

on the program are given in Section C.10. \*/

proc iml;

libname matrice 'C:\Donnees\Mapfre\ASTIN2019';

reset storage='matrice.matrbis';

load n prfreq period;

tmax=max(period);

totobs=nrow(n);

ntotest=prfreq[+];

/\* Time-invariant random effect: estimation with efficient GMM and given frequency premiums \*/

varp=0.67;

lvu=varp\*repeat(1,1,tmax);

/\* Derivation of the matrix F(alpha) \*/

Falpha=repeat(0,totobs,tmax);

imax=totobs;

res=n-prfreq;

do i=1 to imax;

if period[i]=1 then

do;

/\* Output \*/

cres=res[i,]; cprim=prfreq[i,]; nsin=n[i,];

Falpha[i,1]=(cres\*\*2)-(nsin+(lvu[1]\*(cprim\*\*2)));

/\* Initialization \*/

nper=1; lres=cres; lprim=cprim;

end;

if period[i]>1 then

do;

/\* Output \*/

cres=res[i,]; cprim=prfreq[i,]; nsin=n[i,];

Falpha[i,1]=(cres\*\*2)-(nsin+(lvu[1]\*(cprim\*\*2)));

nper=nper+1;

Falpha[i,2:nper]=(cres\*lres)-(cprim\*(lprim#lvu[,2:nper]));

/\* Updating \*/

lres=cres||lres; lprim=cprim||lprim;

end;

end;

/\* Derivation of the objective function g(alpha) \*/

F=Falpha;

Fmean=F[:,];

W=(F`\*F)/totobs;

iW=inv(W);

distopt=Fmean\*iW\*Fmean`;

print distopt;

do varp=0.65 to 0.8 by 0.01;

lvu=varp\*repeat(1,1,tmax);

Falpha=repeat(0,totobs,tmax);

imax=totobs;

res=n-prfreq;

do i=1 to imax;

if period[i]=1 then

do;

/\* Output \*/

cres=res[i,]; cprim=prfreq[i,]; nsin=n[i,];

Falpha[i,1]=(cres\*\*2)-(nsin+(lvu[1]\*(cprim\*\*2)));

/\* Initialization \*/

nper=1; lres=cres; lprim=cprim;

end;

if period[i]>1 then

do;

/\* Output \*/

cres=res[i,]; cprim=prfreq[i,]; nsin=n[i,];

Falpha[i,1]=(cres\*\*2)-(nsin+(lvu[1]\*(cprim\*\*2)));

nper=nper+1;

Falpha[i,2:nper]=(cres\*lres)-(cprim\*(lprim#lvu[,2:nper]));

/\* Updating \*/

lres=cres||lres; lprim=cprim||lprim;

end;

end;

F=Falpha;

Fmean=F[:,];

W=(F`\*F)/totobs;

iW=inv(W);

dist=Fmean\*iW\*Fmean`;

if dist<distopt then

do;

varpopt=varp;

distopt=dist;

metric=iW;

end;

end;

print metric ;

print varpopt distopt;

/\* Q: AR(1) : estimation with efficient GMM and given frequency premiums \*/

/\* First pass: initial values are derived from the rough metric \*/

varp=0.47;

varq=0.54;

phi1=0.45;

lrq=repeat(1,1,tmax);

do i=2 to tmax;

j=i-1;

lrq[1,i]=lrq[1,j]\*phi1;

end;

lvq=varq\*lrq;

lvu=varp+((1+varp)\*lvq);

Falpha=repeat(0,totobs,tmax);

imax=totobs;

res=n-prfreq;

do i=1 to imax;

if period[i]=1 then

do;

/\* Output \*/

cres=res[i,]; cprim=prfreq[i,]; nsin=n[i,];

Falpha[i,1]=(cres\*\*2)-(nsin+(lvu[1]\*(cprim\*\*2)));

/\* Initialization \*/

nper=1; lres=cres; lprim=cprim;

end;

if period[i]>1 then

do;

/\* Output \*/

cres=res[i,]; cprim=prfreq[i,]; nsin=n[i,];

Falpha[i,1]=(cres\*\*2)-(nsin+(lvu[1]\*(cprim\*\*2)));

nper=nper+1;

Falpha[i,2:nper]=(cres\*lres)-(cprim\*(lprim#lvu[,2:nper]));

/\* Updating \*/

lres=cres||lres; lprim=cprim||lprim;

end;

end;

F=Falpha;

Fmean=F[:,];

W=(F`\*F)/totobs;

iW=inv(W);

distopt=Fmean\*iW\*Fmean`;

print distopt;

/\* Loop \*/

opt1t=0.47;

opt2t=0.54;

opt3t=0.45;

mesh=0.01;

niterat=10;

do iterat=1 to niterat;

opt1=opt1t;

opt2=opt2t;

opt3=opt3t;

do ip=-1 to 1;

do iq=-1 to 1;

do i1=-1 to 1;

varp=opt1+(ip\*mesh);

varq=opt2+(iq\*mesh);

phi1=opt3+(i1\*mesh);

lrq=repeat(1,1,tmax);

do i=2 to tmax;

j=i-1;

lrq[1,i]=lrq[1,j]\*phi1;

end;

lvq=varq\*lrq;

lvu=varp+((1+varp)\*lvq);

Falpha=repeat(0,totobs,tmax);

imax=totobs;

res=n-prfreq;

do i=1 to imax;

if period[i]=1 then

do;

cres=res[i,]; cprim=prfreq[i,]; nsin=n[i,];

Falpha[i,1]=(cres\*\*2)-(nsin+(lvu[1]\*(cprim\*\*2)));

nper=1; lres=cres; lprim=cprim;

end;

if period[i]>1 then

do;

cres=res[i,]; cprim=prfreq[i,]; nsin=n[i,];

Falpha[i,1]=(cres\*\*2)-(nsin+(lvu[1]\*(cprim\*\*2)));

nper=nper+1;

Falpha[i,2:nper]=(cres\*lres)-(cprim\*(lprim#lvu[,2:nper]));

lres=cres||lres; lprim=cprim||lprim;

end;

end;

F=Falpha;

Fmean=F[:,];

W=(F`\*F)/totobs;

iW=inv(W);

dist=Fmean\*iW\*Fmean`;

if dist<distopt then

do;

distopt=dist;

opt1t=varp;

opt2t=varq;

opt3t=phi1;

end;

end;

end;

end;

end;

varp=opt1t;

varq=opt2t;

phi1=opt3t;

print varp varq phi1 distopt;

/\* Q: ARFIMA(0,d,0) : estimation with efficient GMM and given frequency premiums \*/

/\* First pass: initial values are derived from the rough GMM \*/

varp=0;

varq=1.28;

d=0.37;

lrq=repeat(1,1,tmax);

do i=2 to tmax;

j=i-1; k=i-2;

lrq[1,i]=lrq[1,j]\*(k+d)/(j-d);

end;

lvq=varq\*lrq;

lvu=varp+((1+varp)\*lvq);

Falpha=repeat(0,totobs,tmax);

imax=totobs;

res=n-prfreq;

do i=1 to imax;

if period[i]=1 then

do;

/\* Output \*/

cres=res[i,]; cprim=prfreq[i,]; nsin=n[i,];

Falpha[i,1]=(cres\*\*2)-(nsin+(lvu[1]\*(cprim\*\*2)));

/\* Initialization \*/

nper=1; lres=cres; lprim=cprim;

end;

if period[i]>1 then

do;

/\* Output \*/

cres=res[i,]; cprim=prfreq[i,]; nsin=n[i,];

Falpha[i,1]=(cres\*\*2)-(nsin+(lvu[1]\*(cprim\*\*2)));

nper=nper+1;

Falpha[i,2:nper]=(cres\*lres)-(cprim\*(lprim#lvu[,2:nper]));

/\* Updating \*/

lres=cres||lres; lprim=cprim||lprim;

end;

end;

F=Falpha;

Fmean=F[:,];

W=(F`\*F)/totobs;

iW=inv(W);

distopt=Fmean\*iW\*Fmean`;

print distopt;

/\* Loop \*/

opt1t=0;

opt2t=1.28;

opt3t=0.37;

mesh=0.01;

niterat=10;

do iterat=1 to niterat;

opt1=opt1t;

opt2=opt2t;

opt3=opt3t;

do ip=-1 to 1;

do iq=-1 to 1;

do i1=-1 to 1;

varp=opt1+(ip\*mesh);

varq=opt2+(iq\*mesh);

d=opt3+(i1\*mesh);

lrq=repeat(1,1,tmax);

do i=2 to tmax;

j=i-1; k=i-2;

lrq[1,i]=lrq[1,j]\*(k+d)/(j-d);

end;

lvq=varq\*lrq;

lvu=varp+((1+varp)\*lvq);

Falpha=repeat(0,totobs,tmax);

imax=totobs;

res=n-prfreq;

do i=1 to imax;

if period[i]=1 then

do;

cres=res[i,]; cprim=prfreq[i,]; nsin=n[i,];

Falpha[i,1]=(cres\*\*2)-(nsin+(lvu[1]\*(cprim\*\*2)));

nper=1; lres=cres; lprim=cprim;

end;

if period[i]>1 then

do;

cres=res[i,]; cprim=prfreq[i,]; nsin=n[i,];

Falpha[i,1]=(cres\*\*2)-(nsin+(lvu[1]\*(cprim\*\*2)));

nper=nper+1;

Falpha[i,2:nper]=(cres\*lres)-(cprim\*(lprim#lvu[,2:nper]));

lres=cres||lres; lprim=cprim||lprim;

end;

end;

F=Falpha;

Fmean=F[:,];

W=(F`\*F)/totobs;

iW=inv(W);

dist=Fmean\*iW\*Fmean`;

if dist<distopt then

do;

distopt=dist;

opt1t=varp;

opt2t=varq;

opt3t=d;

end;

end;

end;

end;

end;

varp=opt1t;

varq=opt2t;

d=opt3t;

print varp varq d distopt;

# This R program derives the matrices related to the projective approach in the affine plane that

# contains the simplex

rho=0.5

mro=matrix(1,3,3)

for (i in 1:3){

for (j in 1:3){

diff=abs(i-j)

mro[i,j]=rho^diff

}}

vro=matrix(1,3,1)

for (i in 1:3){vro[i]=rho^i}

pvro=vro/sum(vro)

# "pvro": barycentric coordinates of vgamma\_3 in the canonical basis

rowscol=rowSums(mro)

idscol=diag(1/rowscol)

mata=mro%\*%idscol

imata=solve(mata)

precision=solve(mro)

mult=imata%\*%mro

# the columns of "mata" are the barycentric coordinates of a\_1, a\_2, a\_3 in the canonical basis.

# the columns of "imata" are the barycentric coordinates of the canonical basis in a\_1, a\_2, a\_3.

# mult: is a diagonal matrix, that expresses the lines of "imata" as multiples of the lines of

# "precision", the precision matrix.

# This Mathematica program generates the 3D plot of the stationarity domain for AR(3) time series.

Plot3D[{1-Abs[phi1+phi3],-(1+(phi1\*phi3)-(phi3\*phi3))}, {phi1,-3,3}, {phi3,-1,1}, RegionFunction->Function[{phi1,phi3,phi2},Abs[phi3-phi1]<2],

PlotStyle->{Red,Green}, ImageSize->800,

AxesLabel->{Style["phi1",FontSize->24,Bold],Style["phi3",FontSize->24,Bold],Style["phi2",FontSize->24,Bold]},

AxesOrigin->{0,0,0}]

# This Mathematica program generates the 3D plot of the set associated to stationary times series of

# the AR(3) type that reach level S

Plot3D[Min[1-(phi1+phi2),phi1\*(1-phi2)/phi2,phi2/phi1], {phi1,0,1}, {phi2,0,1}, RegionFunction->Function[{phi1,phi2,phi3},phi1+phi2<1],

PlotStyle->{Yellow}, ImageSize->800,

AxesLabel->{Style["phi1",FontSize->24,Bold],Style["phi2",FontSize->24,Bold],Style["phi3",FontSize->24,Bold]},

AxesOrigin->{0,0,0}]

# This R program provides a between-within analysis of the random effects (Table 3).

# See Appendix B.15 for the formulas.

# Q: ARFIMA(0,d,0)

varp=0

varq=1.28

d=0.37

Tmax=1000

lvq=matrix(0,1,Tmax)

lvq[,1]=varq

for (i in 2:Tmax){

j=i-1

lvq[,i]=lvq[,j]\*(j-1+d)/(j-d)}

lvu=varp+((1+varp)\*lvq)

T=5

mvarq=matrix(0,T,T)

for (i in 1:T){

for (j in 1:T){

k=1+abs(j-i)

mvarq[i,j]=lvq[k]}}

within=(1+varp)\*(varq-(sum(mvarq)/(T\*T)))

between=varp+((1+varp)\*sum(mvarq)/(T\*T))

c(T,between,within)

# Q: AR(1)

varp=0.47

varq=0.54

phi1=0.45

Tmax=1000

lvq=matrix(0,1,Tmax)

lvq[,1]=varq

for (i in 2:Tmax){

j=i-1

lvq[,i]=lvq[,j]\*phi1}

lvu=varp+((1+varp)\*lvq)

T=5

mvarq=matrix(0,T,T)

for (i in 1:T){

for (j in 1:T){

k=1+abs(j-i)

mvarq[i,j]=lvq[k]}}

within=(1+varp)\*(varq-(sum(mvarq)/(T\*T)))

between=varp+((1+varp)\*sum(mvarq)/(T\*T))

c(T,between,within)