output file = poissongme.out reset;   
  
format /M1 /RD 6,3; aa = 3331711;  
  
"";  
" Y is Poisson ";  
""; "Seed= " aa;  
  
  
simun = 500;  
nboot = 5; /\* btno is number of bootstrap say for standard error \*/  
k = 2;  
etaout = 0; /\* indicator of reporting nuisance estimates \*/  
gammat = 0.2|0.8;  
sigmaeps = 1;  
naivediv = 0; rcdiv = 0; buzasbio1div = 0; buzasbio2div = 0;  
npcdiv = 0;  
  
paran = 1; do while paran .<4;  
if paran == 1; n = 1000; xmod = 1; umod = 1;  
beta0 = -ln(2); beta1 = ln(2); endif;  
if paran == 2; n = 1000; xmod = 2; umod = 1;   
beta0 = -ln(2); beta1 = ln(2); endif;  
if paran == 3; n = 1000; xmod = 3; umod = 1;

beta0 = -ln(2); beta1 = ln(2); endif;  
if paran == 4; n = 1000; xmod = 4; umod = 1;   
beta0 = -ln(2); beta1 = ln(2); endif;  
  
seed = aa;  
mux = 0; sigmax = 1; sigmau = 1;  
  
betatrue = beta0|beta1|mux|sigmax|sigmau;   
  
/\* betatrue: beta0,beta1,mux,sigmax,sigmae,sigmau \*/  
  
naiveest = ones(simun,15); rcest = ones(simun,15);   
buzasbio1est = ones(simun,15); buzasbio2est = ones(simun,15);  
npcest = ones(simun,15);  
  
preva = 0;  
  
count = 1; do while count .<= simun;   
lab1:  
if xmod == 1;  
xx = (sigmax .\* rndns(n,1,seed)) + mux;   
endif;   
if xmod == 2;  
xx = (sigmax .\*(sqrt(12) .\* (rndus(n,1,seed) - .5))) + mux;   
endif;  
if xmod == 3;  
temp = (rndus(n,1,seed) .< .5);   
xx = ((1-temp) .\* ((2 .\* rndns(n,1,seed))+ 1)) +  
(temp .\* (rndns(n,1,seed)-1)); xx = xx ./ sqrt(3.5);   
xx = xx + mux;  
endif;  
  
if xmod == 4;  
xx = rndns(n,1,seed)^2; xx = xx -1;  
xx = xx ./ sqrt(2);  
xx = xx + mux;  
endif;  
  
yy = zeros(n,1);  
count1 = 1; do while count1 .<= n;  
temp = rndus(1,1,seed);  
lambda = exp(betatrue[1]+(betatrue[2] .\* xx[count1]));  
yprob = 0;  
count2 = 0; do while count2 .<= 500;  
yy[count1] = count2;  
pryest = (exp(-lambda) .\* (lambda.^count2)) ./ (count2!);  
yprob = yprob + pryest;  
if yprob .> 1; "bad data prob(y|x) " yprob "simu" count; goto lab1; endif;   
if temp < yprob; count2 = 999; endif;  
count2 = count2 + 1; endo;  
count1 = count1 + 1; endo;  
  
if umod == 1;  
uu = sigmau .\* rndns(n,k,seed);   
endif;   
  
if umod == 2;  
uu = sigmau .\*(sqrt(12) .\* (rndus(n,k,seed) - .5));   
endif;  
if umod == 3;  
temp = (rndus(n,k,seed) .< .5);   
uu = ((1-temp) .\* ((2 .\* rndns(n,k,seed))+ 1)) +  
(temp .\* (rndns(n,k,seed)-1)); uu = uu ./ sqrt(3.5);   
endif;  
  
if umod == 4;  
uu = rndns(n,k,seed)^2; uu = uu -1;  
uu = uu ./ sqrt(2);  
endif;  
  
ww = (xx \* ones(1,k)) + uu;   
wbar = meanc(ww');  
  
/\*screen off;  
output file = try.dat reset; print yy~ww; output file = try.dat off;  
screen on;\*/  
  
if count == 1; "First set meanyy " meanc(yy); endif;  
  
xxstar = (ones(n,1)~xx) \* gammat;  
eps = rndns(n,2,seed);  
  
qq = xxstar + (sigmaeps .\* eps);  
  
/\* eta is the calibration indicator; \*/  
pitrue = .40 .\* ones(n,1);  
eta = (rndus(n,1,seed) .< pitrue);  
nv = sumc(eta); np = n - nv;  
  
mat = xx~ww~qq~eta~yy;  
mat = sortc(mat,6);  
xx = mat[.,1];  
ww = mat[.,2:3]; qq = mat[.,4:5];  
matp = mat[1:np,.]; matv = mat[(np+1):n,.];  
wwv = matv[.,2:3]; wwp = matp[.,2:3];  
qqv = matv[.,4:5]; qqp = matp[.,4:5];  
wwvbar = meanc(wwv'); qqvbar = meanc(qqv'); k1 = 2; k2 = 2;  
qqpbar = meanc(qqp');  
yy = mat[.,7]; yyp = yy[1:np,.]; yyv = yy[(np+1):n,.];  
eta = mat[.,6];  
  
betastar = 0|0;  
{betahat,betase,errcode} = poisson(yyv,wwvbar,betastar);  
  
if errcode .> .5; "Naive not converge on data set " count "errcode =" errcode ;  
goto lab1; endif;  
  
muxhat = meanc(wwvbar);   
varuhat = 0; varwbhat = 0;  
count1 = 1; do while count1 .< (nv+1);  
varuhat = varuhat + sumc(((wwv[count1,.]-wwvbar[count1])^2)')./(k1-1);  
varwbhat = varwbhat + (wwvbar[count1]-muxhat)^2;  
count1 = count1 + 1; endo;  
varuhat = varuhat ./ nv;   
varwbhat = varwbhat ./ nv;  
varubhat = varuhat / k1;   
varxhat = varwbhat - varubhat;   
varxhat = maxc(0.001|varxhat);  
  
temp = varxhat+varubhat;  
temp = varxhat \* inv(temp);  
muxgwv = muxhat + ((wwvbar-muxhat) \* temp');  
  
betahat = betahat|muxhat|sqrt(varxhat)|sqrt(varuhat);  
betase = betase|1|1|1;  
  
naiveest[count,1:5] = betahat';  
naiveest[count,6:10] = betase';  
naiveest[count,11:15] = ((betahat - betatrue) ./ betase)';  
  
{betahat,betase,errcode} = rcpoisson(yyv,wwv,betahat);  
  
rcest[count,1:5] = betahat';  
rcest[count,6:10] = betase';  
rcest[count,11:15] = ((betahat - betatrue) ./ betase)';  
if errcode .> .5; "RC not converge on data set " count "errcode =" errcode ;  
endif;  
  
  
{betahat,betase,errbuzasbio1} = buzasrep(yyv,wwv,betahat[1:2]);  
  
betahat = betahat|muxhat|sqrt(varxhat)|sqrt(varuhat);  
betase = betase|1|1|1;  
  
buzasbio1est[count,1:5] = betahat';  
buzasbio1est[count,6:10] = betase';  
buzasbio1est[count,11:15] = ((betahat - betatrue) ./ betase)';  
  
if errcode .> .5; "Buzas-1 not converge on set " count "errcode =" errcode ;  
buzasbio1div = buzasbio1div + 1; endif;  
  
/\* Now Buzas using both ww and qq \*/  
  
{muxgw,muxgwq,muxgq,gammahat} = calibnewsub(wwv,qqv);  
  
{betahat,betase,errbuzasbio2} = buzasinstr(yyv,wwv,qqv,betahat[1:2]);  
  
if errbuzasbio2 .> .5;   
"Buzas-2 not converge on set " count "errcode =" errcode ;  
buzasbio2div = buzasbio2div + 1; endif;  
  
betahat = betahat|muxhat|sqrt(varxhat)|sqrt(varuhat);  
betase = betase|1|1|1;  
  
buzasbio2est[count,1:5] = betahat';  
buzasbio2est[count,6:10] = betase';  
buzasbio2est[count,11:15] = ((betahat - betatrue) ./ betase)';  
  
/\* Now all data \*/  
  
{betahat,betase,errnpc} = npcall(yy,ww,qq,eta,gammahat,betahat[1:2]);  
/\*{betahat,betase,errnpc} = test(yy,ww,qq,eta,gammahat,betahat[1:2]);\*/  
  
if errnpc .> .5; "NPC not converge on set " count "errnpc =" errnpc ;  
npcdiv = npcdiv + 1; endif;  
  
betahat = betahat|muxhat|sqrt(varxhat)|sqrt(varuhat);  
betase = betase|1|1|1;  
  
npcest[count,1:5] = betahat';  
npcest[count,6:10] = betase';  
npcest[count,11:15] = ((betahat - betatrue) ./ betase)';  
  
errcode = maxc(errbuzasbio1|errbuzasbio2|errnpc);  
if errcode .> 0.5; goto lab1; endif;  
  
/\*if count == 200 or count == simun;\*/  
  
  
  
  
if count == simun;  
naivebias = meanc(naiveest[1:count,1:5]) - betatrue;  
naivesd = stdc(naiveest[1:count,1:5]);  
naiveseave = meanc(naiveest[1:count,6:10]);  
naivecov = meanc((abs(naiveest[1:count,11:15]) .< 1.96));  
  
rcbias = meanc(rcest[1:count,1:5]) - betatrue;  
rcsd = stdc(rcest[1:count,1:5]);  
rcseave = meanc(rcest[1:count,6:10]);  
rccov = meanc((abs(rcest[1:count,11:15]) .< 1.96));  
  
buzasbio1bias = meanc(buzasbio1est[1:count,1:5]) - betatrue;  
buzasbio1sd = stdc(buzasbio1est[1:count,1:5]);  
buzasbio1seave = meanc(buzasbio1est[1:count,6:10]);  
buzasbio1cov = meanc((abs(buzasbio1est[1:count,11:15]) .< 1.96));  
  
buzasbio2bias = meanc(buzasbio2est[1:count,1:5]) - betatrue;  
buzasbio2sd = stdc(buzasbio2est[1:count,1:5]);  
buzasbio2seave = meanc(buzasbio2est[1:count,6:10]);  
buzasbio2cov = meanc((abs(buzasbio2est[1:count,11:15]) .< 1.96));  
  
npcbias = meanc(npcest[1:count,1:5]) - betatrue;  
npcsd = stdc(npcest[1:count,1:5]);  
npcseave = meanc(npcest[1:count,6:10]);  
npccov = meanc((abs(npcest[1:count,11:15]) .< 1.96));  
  
"----------------------------------------";  
"Current simulation number is " count;  
if xmod == 1; " X is normal "; endif;  
if xmod == 2; " X is uniform "; endif;   
if xmod == 3; " X is mixure of normals "; endif;  
if xmod == 4; " X is chi-square(1) "; endif;  
if umod == 1; " U is normal "; endif;  
if umod == 2; " U is uniform "; endif;   
if umod == 3; " U is mixure of normals "; endif;  
if umod == 4; " U is chi-square(1) "; endif;  
"";  
"Sample size = " n;  
"True parameters = " betatrue';   
"Total number of Buzasbio-1 divergence " buzasbio1div;   
"Total number of Buzasbio-2 divergence " buzasbio2div;   
"Total number of NPC divergence " npcdiv;   
"";  
"";  
"------------------- beta\_0-------------" betatrue[1];  
  
" Naive RC BUZASBIO-1 BUZASBIO-2 NPC ";  
"bias " naivebias[1]~rcbias[1]~buzasbio1bias[1]~buzasbio2bias[1]~npcbias[1];  
"sd " naivesd[1]~rcsd[1]~buzasbio1sd[1]~buzasbio2sd[1]~npcsd[1];  
"Ave se" naiveseave[1]~rcseave[1]~buzasbio1seave[1]~buzasbio2seave[1]~npcseave[1];  
"95%cov" naivecov[1]~rccov[1]~buzasbio1cov[1]~buzasbio2cov[1]~npccov[1];  
"";  
"------------------- beta\_1-------------" betatrue[2];  
  
" Naive RC BUZASBIO-1 BUZASBIO-2 ";  
"bias " naivebias[2]~rcbias[2]~buzasbio1bias[2]~buzasbio2bias[2]~npcbias[2];  
"sd " naivesd[2]~rcsd[2]~buzasbio1sd[2]~buzasbio2sd[2]~npcsd[2];  
"Ave se" naiveseave[2]~rcseave[2]~buzasbio1seave[2]~buzasbio2seave[2]~npcseave[2];  
"95%cov" naivecov[2]~rccov[2]~buzasbio1cov[2]~buzasbio2cov[2]~npccov[2];  
"";  
if etaout == 1;  
"";  
"------------------- mu\_x-------------" betatrue[3];  
  
" Naive RC ";  
"bias " naivebias[3]~rcbias[3];  
"sd " naivesd[3]~rcsd[3];  
"Ave se" naiveseave[3]~rcseave[3];  
"95%cov" naivecov[3]~rccov[3];  
"";  
"";  
"------------------- sigma\_x-------------" betatrue[4];  
  
" Naive RC ";  
"bias " naivebias[4]~rcbias[4];  
"sd " naivesd[4]~rcsd[4];  
"Ave se" naiveseave[4]~rcseave[4];  
"";  
"------------------- sigma\_u -------------" betatrue[5];  
  
" Naive RC ";  
"bias " naivebias[5]~rcbias[5];  
"sd " naivesd[5]~rcsd[5];  
"Ave se" naiveseave[5]~rcseave[5];  
"95%cov" naivecov[5]~rccov[5];  
"";  
endif;  
endif;  
count = count + 1; endo;  
  
paran = paran + 1; endo; /\* endo for paran \*/  
  
end;  
  
#include poissongmesub.g;  
#include lsfitsub.g;  
  
  
/\* ########################################################### \*/  
proc(4) = lsfit(yy,xx);  
local betahat,errcode,k,mm,n,se,sigma,x,yhat,resid;  
  
n = rows(yy);   
x = ones(n,1)~xx; k = cols(x); betahat = zeros(k,1); se = ones(k,1);  
sigma = 0;  
errcode = 1;  
mm = x'x;  
if minc(eigh(mm)) .> 0.0001;   
errcode = 0;  
betahat = (inv(x'x)) \* (x' \* yy);  
yhat = x \* betahat;  
resid = yhat - yy;  
sigma = sumc(resid^2) ./ (n-k); sigma = sqrt(sigma);  
se = sqrt(diag((inv(x'x) .\* sigma^2)));  
endif;  
  
retp(betahat,se,sigma,errcode);  
  
endp;  
  
/\* ################################################################# \*/  
/\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
this program computes the poisson regression  
coefficients  
  
the method of computation is standard scoring. i have added the  
twist that if the estimates get out of hand, i.e., too large, then i  
just restart with a randomly chosen starting value.  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*/  
proc (3) = poisson(yy,xx,betastar);  
local m,n,x,ii,betaold,betahat,aaaa,ss3,s1,s2,s3;  
local errcode,pred,fmf,stderr,st,xstar,ystar,eps;  
  
errcode = 0;  
m = cols(xx) + 1; n = rows(xx); betahat = betastar;  
x = ones(n,1)~xx;  
ii = 0 ; do until ii .>= 50 ; ii = ii + 1 ; betaold = betahat ;  
aaaa = zeros(m,m);   
ss3 = x\*betaold;  
pred = exp(ss3);  
xstar = x .\* ones(1,m);  
ystar = xstar' \* (yy - pred);  
aaaa = xstar' \* (xstar .\* (pred \* ones(1,m)));   
betahat = betaold + inv(aaaa) \* ystar;  
  
eps = ones(m,1)' \* abs(betahat - betaold) ;  
  
/\* "niter = " ii "eps" eps; \*/  
  
if eps .<= 0.0001 ; ii = 20000000 ; endif ; betaold = betahat ;  
/\*if (abs(betahat)' \* ones(m,1)) .> 50;  
betaold = 3 .\* (rndu(m,1) - (.5 .\* ones(m,1)));  
betahat = betaold; endif; \*/  
endo ;  
if ii .< 9999;  
aaaa = eye(rows(betahat)); errcode = 1;  
endif;  
stderr = sqrt(diag(inv(aaaa)));  
retp(betahat,stderr,errcode);  
endp;  
  
  
/\* ####################################################### \*/  
/\* subroutine for RC for Poisson regression \*/  
  
proc(3) = rcpoisson(yy,ww,betastar);  
local betanew,betaold,cov,eps,errcode,niter,nouse,ss,stderr;  
local diff,gg,gg1,gg2,gg3,gg4,gg5;  
local hh,hh11,hh12,hh13,hh14,hh15,hh21,hh22,hh23,hh24,hh25;  
local hh31,hh32,hh33,hh34,hh35,hh41,hh42,hh43,hh44,hh45;  
local hh51,hh52,hh53,hh54,hh55;  
local betaold1,betaold2,betaold3,betaold4,betaold5;  
local aa,bb,corwstar,count,count1,k,n,temp;  
  
errcode =0; n = rows(yy); k = cols(ww);  
stderr = zeros(rows(betastar),1);  
betaold = betastar;  
  
diff = 0.01;  
niter = 0;  
do until niter .> 20;  
niter = niter + 1;  
  
{gg,ss} = gradrcpoisson(yy,ww,betaold);  
  
betaold1 = betaold; betaold1[1] = betaold[1] + diff;  
{gg1,nouse} = gradrcpoisson(yy,ww,betaold1);  
  
betaold2 = betaold; betaold2[2] = betaold[2] + diff;  
{gg2,nouse} = gradrcpoisson(yy,ww,betaold2);  
  
betaold3 = betaold; betaold3[3] = betaold[3] + diff;  
{gg3,nouse} = gradrcpoisson(yy,ww,betaold3);  
  
betaold4 = betaold; betaold4[4] = betaold[4] + diff;  
  
{gg4,nouse} = gradrcpoisson(yy,ww,betaold4);  
  
betaold5 = betaold; betaold5[5] = betaold[5] + diff;  
{gg5,nouse} = gradrcpoisson(yy,ww,betaold5);  
  
hh11 = (gg1[1] - gg[1]) ./ diff; hh12 = (gg2[1] - gg[1]) ./ diff;  
hh13 = (gg3[1] - gg[1]) ./ diff; hh14 = (gg4[1] - gg[1]) ./ diff;  
hh15 = (gg5[1] - gg[1]) ./ diff;  
  
hh21 = (gg1[2] - gg[2]) ./ diff; hh22 = (gg2[2] - gg[2]) ./ diff;  
hh23 = (gg3[2] - gg[2]) ./ diff; hh24 = (gg4[2] - gg[2]) ./ diff;  
hh25 = (gg5[2] - gg[2]) ./ diff;  
  
hh31 = (gg1[3] - gg[3]) ./ diff; hh32 = (gg2[3] - gg[3]) ./ diff;  
hh33 = (gg3[3] - gg[3]) ./ diff; hh34 = (gg4[3] - gg[3]) ./ diff;  
hh35 = (gg5[3] - gg[3]) ./ diff;  
  
hh41 = (gg1[4] - gg[4]) ./ diff; hh42 = (gg2[4] - gg[4]) ./ diff;  
hh43 = (gg3[4] - gg[4]) ./ diff; hh44 = (gg4[4] - gg[4]) ./ diff;  
hh45 = (gg5[4] - gg[4]) ./ diff;  
  
hh51 = (gg1[5] - gg[5]) ./ diff; hh52 = (gg2[5] - gg[5]) ./ diff;  
hh53 = (gg3[5] - gg[5]) ./ diff; hh54 = (gg4[5] - gg[5]) ./ diff;  
hh55 = (gg5[5] - gg[5]) ./ diff;  
  
hh = (hh11~hh12~hh13~hh14~hh15)|(hh21~hh22~hh23~hh24~hh25)|  
(hh31~hh32~hh33~hh34~hh35)|(hh41~hh42~hh43~hh44~hh45)|  
(hh51~hh52~hh53~hh54~hh55);  
  
temp = det(-hh);  
  
if temp .< .0000001; niter = 99999; betanew = betaold;  
"singularity problem" det(-hh); errcode = 1; goto lab20; endif;  
  
if temp .> .0000001;  
betanew = betaold - (inv(hh) \* gg);  
if sumc(abs(betanew-betastar)) .> 50; niter = 99999; errcode = 2;  
endif;  
  
eps = sumc(abs(betanew - betaold));  
  
if niter .> 5; "niter=" niter "eps=" eps; endif;  
  
if eps .< .001;  
niter = 999990;  
cov = (inv(hh) \* ss) \* ( (inv(hh))');  
stderr = diag(sqrt(cov));  
errcode = 0;  
endif;  
  
betaold = betanew;  
endif;  
endo;  
  
lab20:  
if niter .< 5000; errcode = 3; endif;  
  
retp(betanew,stderr,errcode);  
endp;  
  
/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/  
proc(2) = gradrcpoisson(yy,ww,beta);  
  
local count,count1,den,dygx,fkx,fky,gg,n,num,pdf,rr,wgx,xgmu,ygx,zgy;  
local muxgw,muxgwy,pred,varxgwy,wbar,zbar;  
local mux,varx,vare,varegwy,varegw,vareps,varepsb,varu,varub;  
local errgw,errgwy,pp1,pp2,pp3,pp4,pp5,pp6,pp7,score,ss,temp1,temp2,temp3;  
local aa,bb,incmtx,incmty,inter,k,maxx,maxy,minx,miny,sumsqre,temp;  
  
n = rows(yy); k = cols(ww);  
gg = zeros(5,1);  
mux = beta[3]; varx = beta[4]^2; varu = beta[5]^2;  
wbar = meanc(ww');  
varub = varu ./ k;  
temp = varx ./ (varx + varub);  
muxgw = mux + ((wbar - mux) \* temp');  
  
  
pred = exp(beta[1] + (beta[2] .\* muxgw));  
pp1 = yy - pred; gg[1] = sumc(pp1);  
  
pp2 = muxgw .\* pp1; gg[2] = sumc(pp2);  
  
pp3 = wbar-mux; gg[3] = sumc(pp3);  
  
pp4 = (wbar-mux)^2 - (varub + varx); gg[4] = sumc(pp4);  
  
pp5 = zeros(n,1);  
count = 1; do while count .< (n+1);  
pp5[count] = sumc(((ww[count,.]-wbar[count])^2)') -  
((k-1).\* varu);  
count = count + 1; endo;  
gg[5] = sumc(pp5);  
  
score = pp1~pp2~pp3~pp4~pp5;  
ss = score' \* score;  
  
retp(gg,ss);  
endp;  
  
/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/  
  
/\* Nonparametric Correction for Poisson regression \*/  
  
proc(3) = npc(yy,ww,betastar);  
local betanew,betaold,cov,eps,errcode,niter,nouse,ss,stderr;  
local gg,gg1,gg2;  
local hh,hh11,hh12,hh21,hh22;  
local betaold1,betaold2;  
local count,count1,diff,n;  
  
errcode =0; n = rows(yy);  
stderr = zeros(rows(betastar),1);  
betaold = betastar;  
diff = 0.01;  
  
niter = 0;  
do until niter .> 30;  
niter = niter + 1;  
  
{gg,ss} = gradnpc(yy,ww,betaold);  
  
betaold1 = betaold; betaold1[1] = betaold[1] + diff;  
{gg1,nouse} = gradnpc(yy,ww,betaold1);  
  
betaold2 = betaold; betaold2[2] = betaold[2] + diff;  
{gg2,nouse} = gradnpc(yy,ww,betaold2);  
  
hh11 = (gg1[1] - gg[1]) ./ diff; hh12 = (gg2[1] - gg[1]) ./ diff;  
  
hh21 = (gg1[2] - gg[2]) ./ diff; hh22 = (gg2[2] - gg[2]) ./ diff;  
  
hh = (hh11~hh12)|(hh21~hh22);  
  
if abs(det(-hh)) .< .00000001; niter = 99999; betanew = betaold;  
"singularity problem" det(-hh); errcode = 1; goto lab20; endif;   
  
  
betanew = betaold - (inv(hh) \* gg);  
if sumc(abs(betanew)) .> 50; niter = 99999; errcode = 2;   
endif;   
  
eps = sumc(abs(betanew - betaold));  
  
if eps .< .00001;  
niter = 999990;  
cov = (inv(hh) \* ss) \* ( (inv(hh))');  
stderr = diag(sqrt(cov));  
errcode = 0;  
endif;  
  
betaold = betanew;  
  
endo;  
  
lab20:  
if niter .< 5000; errcode = 3; endif;  
  
retp(betanew,stderr,errcode);  
endp;  
  
  
/\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
This program computes the poisson regression gradient  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*/  
proc (2) = gradnpc(yy,ww,beta);  
local m,n,gg,pp1,pp2,pred1,pred2,score,ss,temp1,temp2,ww1,ww2;  
  
m = cols(xx) + 1; n = rows(ww); gg = zeros(2,1);  
ww1 = ww[.,1]; ww2 = ww[.,2];  
pred1 = exp((ones(n,1)~ww1)\*beta);  
pred2 = exp((ones(n,1)~ww2)\*beta);  
  
temp1 = yy - pred1; temp2 = yy - pred2;  
pp1 = temp1 + temp2; gg[1] = sumc(pp1);  
  
pp2 = (temp1 .\* ww2) + (temp2 .\* ww1); gg[2] = sumc(pp2);   
  
score = pp1~pp2;  
ss = score' \* score;  
  
retp(gg,ss);  
endp;  
  
  
/\* Method for Poisson regression for whole cocort.  
In the subsample, 2 "unbiased surrogate" ww, and 2 "instrumental   
variables" tt, in the non-calibration sample, 2 IV \*/  
  
proc(3) = npcall(yy,ww,qq,eta,gammahat,betastar);  
local betanew,betaold,cov,eps,errcode,errcode1,niter,nouse,ss,stderr;  
local gg,gg1,gg2;  
local hh,hh11,hh12,hh21,hh22;  
local betaold1,betaold2;  
local count,count1,diff,n;  
  
errcode =0; n = rows(yy);  
stderr = zeros(rows(betastar),1);  
betaold = betastar;  
diff = 0.01;  
  
niter = 0;  
do until niter .> 30;  
niter = niter + 1;  
  
{gg,ss,errcode1} = gradnpcall(yy,ww,qq,eta,gammahat,betaold);  
if errcode1 == 1; errcode = 1; endif;  
  
betaold1 = betaold; betaold1[1] = betaold[1] + diff;  
{gg1,nouse,errcode1} = gradnpcall(yy,ww,qq,eta,gammahat,betaold1);  
  
betaold2 = betaold; betaold2[2] = betaold[2] + diff;  
{gg2,nouse,errcode1} = gradnpcall(yy,ww,qq,eta,gammahat,betaold2);  
  
hh11 = (gg1[1] - gg[1]) ./ diff; hh12 = (gg2[1] - gg[1]) ./ diff;  
  
hh21 = (gg1[2] - gg[2]) ./ diff; hh22 = (gg2[2] - gg[2]) ./ diff;  
  
hh = (hh11~hh12)|(hh21~hh22);  
  
if abs(det(-hh)) .< .00000001; niter = 99999; betanew = betaold;  
"singularity problem" det(-hh); errcode = 1; goto lab20; endif;   
  
  
betanew = betaold - (inv(hh) \* gg);  
if sumc(abs(betanew)) .> 50; niter = 99999; errcode = 2;   
endif;   
  
eps = sumc(abs(betanew - betaold));  
  
if eps .< .00001;  
niter = 999990;

cov = (inv(hh) \* ss) \* ( (inv(hh))');  
stderr = diag(sqrt(cov));  
errcode = 0;  
endif;  
  
betaold = betanew;  
  
endo;  
  
lab20:  
if niter .< 5000; errcode = 3; endif;  
  
retp(betanew,stderr,errcode);  
endp;  
  
/\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
This program computes Buzas method using all data -- including  
calibration and non-calibration data sets. eta is the indicator  
for calibration sample. 2 ww and 2 qq available in the calibration  
sample, while 2 qq available in the non-calibration sample.   
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*/  
proc (3) = gradnpcall(yy,ww,qq,eta,gammahat,beta);  
local m,n,gg,pp1,pp2,score,ss,ww1,ww2,qq1,qq2;  
local temp1,temp2,temp3,temp4,temp5,temp6;  
local temp7,temp8,temp9,temp10,temp11,temp12;  
local alphase,temp,weight1,weight2,weight3,weight4,weight5,weight6;  
local weight7,weight8,weight9,weight10,weight11,weight12;  
local muxgw,muxgwq,muxgq,qq1new,qq2new,sigma;  
local alpha1,alpha2,alpha3,alpha4,alpha5,alpha6,wwbar,qqbar;  
local alpha7,alpha8,alpha9,alpha10,alpha11,alpha12;  
local errcode,errcode1,errcode2,errcode3,errcode4,errcode5,errcode6;  
local errcode7,errcode8,errcode9,errcode10,errcode11,errcode12;  
local ew1gw2,ew1gq1,ew1gq2,ew2gw1,ew2gq1,ew2gq2;  
local eq1gw1,eq1gw2,eq1gq2,eq2gw1,eq2gw2,eq2gq1;  
local pred1,pred2,pred3,pred4;  
  
m = cols(xx) + 1; n = rows(ww); gg = zeros(2,1); ss = zeros(2,2);  
ww1 = ww[.,1]; ww2 = ww[.,2]; qq1 = qq[.,1]; qq2 = qq[.,2];  
wwbar = meanc(ww'); qqbar = meanc(qq');  
qq1new = qq1 ./ gammahat[2] - (gammahat[1] ./ gammahat[2]);  
qq2new = qq2 ./ gammahat[2] - (gammahat[1] ./ gammahat[2]);  
  
{alpha1,alphase,sigma,errcode1} = wlsfit(ww1,ww2,eta);   
{alpha2,alphase,sigma,errcode2} = wlsfit(ww1,qq1new,eta);   
{alpha3,alphase,sigma,errcode3} = wlsfit(ww1,qq2new,eta);  
{alpha4,alphase,sigma,errcode4} = wlsfit(ww2,ww1,eta);  
{alpha5,alphase,sigma,errcode5} = wlsfit(ww2,qq1new,eta);  
{alpha6,alphase,sigma,errcode6} = wlsfit(ww2,qq2new,eta);  
{alpha7,alphase,sigma,errcode7} = wlsfit(qq1new,ww1,eta);   
{alpha8,alphase,sigma,errcode8} = wlsfit(qq1new,ww2,eta);   
{alpha9,alphase,sigma,errcode9} = wlsfit(qq1new,qq2new,eta);  
{alpha10,alphase,sigma,errcode10} = wlsfit(qq2new,ww1,eta);  
{alpha11,alphase,sigma,errcode11} = wlsfit(qq2new,ww2,eta);  
{alpha12,alphase,sigma,errcode12} = wlsfit(qq2new,qq1new,eta);  
  
ew1gw2 = alpha1[1] + alpha1[2] .\* ww2;  
ew1gq1 = alpha2[1] + alpha2[2] .\* qq1new;  
ew1gq2 = alpha3[1] + alpha3[2] .\* qq2new;  
ew2gw1 = alpha4[1] + alpha4[2] .\* ww1;  
ew2gq1 = alpha5[1] + alpha5[2] .\* qq1new;  
ew2gq2 = alpha6[1] + alpha6[2] .\* qq2new;  
eq1gw1 = alpha7[1] + alpha7[2] .\* ww1;  
eq1gw2 = alpha8[1] + alpha8[2] .\* ww2;  
eq1gq2 = alpha9[1] + alpha9[2] .\* qq2new;  
eq2gw1 = alpha10[1] + alpha10[2] .\* ww1;  
eq2gw2 = alpha11[1] + alpha11[2] .\* ww2;  
eq2gq1 = alpha12[1] + alpha12[2] .\* qq1new;  
  
errcode = maxc(errcode1|errcode2|errcode3|errcode4|errcode5|errcode6);  
errcode = maxc(errcode|errcode7|errcode8|errcode9|errcode10);  
errcode = maxc(errcode|errcode11|errcode12);  
  
if errcode .< 1;   
temp = (.5 .\* beta[2]) .\* ((alpha1[2] .\* ww2) - ww1);   
weight1 = exp(temp);  
temp = (.5 .\* beta[2]) .\* ((alpha2[2] .\* qq1new) - ww1);   
weight2 = exp(temp);  
temp = (.5 .\* beta[2]) .\* ((alpha3[2] .\* qq2new) - ww1);   
weight3 = exp(temp);  
temp = (.5 .\* beta[2]) .\* ((alpha4[2] .\* ww1) - ww2);   
weight4 = exp(temp);  
temp = (.5 .\* beta[2]) .\* ((alpha5[2] .\* qq1new) - ww2);   
weight5 = exp(temp);  
temp = (.5 .\* beta[2]) .\* ((alpha6[2] .\* qq2new) - ww2);   
weight6 = exp(temp);  
temp = (.5 .\* beta[2]) .\* ((alpha7[2] .\* ww1) - qq1new);   
weight7 = exp(temp);  
temp = (.5 .\* beta[2]) .\* ((alpha8[2] .\* ww2) - qq1new);   
weight8 = exp(temp);  
temp = (.5 .\* beta[2]) .\* ((alpha9[2] .\* qq2new) - qq1new);  
weight9 = exp(temp);  
temp = (.5 .\* beta[2]) .\* ((alpha10[2] .\* ww1) - qq2new);   
weight10 = exp(temp);  
temp = (.5 .\* beta[2]) .\* ((alpha11[2] .\* ww2) - qq2new);   
weight11 = exp(temp);  
temp = (.5 .\* beta[2]) .\* ((alpha12[2] .\* qq1new) - qq2new);   
weight12 = exp(temp);  
  
pred1 = exp((ones(n,1)~ww1)\*beta);  
pred2 = exp((ones(n,1)~ww2)\*beta);  
pred3 = exp((ones(n,1)~qq1new)\*beta);  
pred4 = exp((ones(n,1)~qq2new)\*beta);  
  
temp1 = (eta .\* weight1) .\* (yy - pred1);   
temp2 = (eta .\* weight2) .\* (yy - pred1);  
temp3 = (eta .\* weight3) .\* (yy - pred1);   
temp4 = (eta .\* weight4) .\* (yy - pred2);  
temp5 = (eta .\* weight5) .\* (yy - pred2);   
temp6 = (eta .\* weight6) .\* (yy - pred2);  
temp7 = (eta .\* weight7) .\* (yy - pred3);  
temp8 = (eta .\* weight8) .\* (yy - pred3);  
temp9 = (ones(n,1) .\* weight9) .\* (yy - pred3);   
temp10 = (eta .\* weight10) .\* (yy - pred4);  
temp11 = (eta .\* weight11) .\* (yy - pred4);   
temp12 = (ones(n,1) .\* weight12) .\* (yy - pred4);  
pp1 = ((temp1 + temp2) + (temp3 + temp4)) + (temp5 + temp6);  
pp1 = (pp1 + (temp7 + temp8)) + ((temp9 + temp10) + (temp11 + temp12));  
gg[1] = sumc(pp1);  
pp2 = (((temp1 .\* ew1gw2) + (temp2 .\* ew1gq1)) +   
((temp3 .\* ew1gq2) + (temp4 .\* ew2gw1))) +   
((temp5 .\* ew2gq1) + (temp6 .\* ew2gq2));  
pp2 = (pp2 + ((temp7 .\* eq1gw1) + (temp8 .\* eq1gw2))) +   
(((temp9 .\* eq1gq2) + (temp10 .\* eq2gw1)) +   
((temp11 .\* eq2gw2) + (temp12 .\* eq2gq1)));  
gg[2] = sumc(pp2);   
  
score = pp1~pp2;  
ss = score' \* score;  
endif;  
  
retp(gg,ss,errcode);  
endp;