



```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%      Problem Set #2 - Econ 673 - Spring 2008
%      Question 2
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

clear;clc;
diary Prob3q2.out;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%      The first step in this program is to generate the data
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

%      specify parameter values
beta0 = -1.0;
betag = 1.2;
betap = -.004;
betac = 1.0;
betae = 1.0;
beta  = [beta0; betag; betap; betac; betae];

```

```

%      generate explanatory variables
nobs  = 1000;
a     = 2.5;
b     = 3.5;
mu    = 2.9;
sig   = 0.5;
ua    = normcdf((a - mu)/sig);
ub    = normcdf((b - mu)/sig);
u     = ua + (ub - ua)*rand(nobs,1);
g     = norminv(u,mu,sig);           %gasoline prices

p     = 500 + 1500*rand(nobs,1);     %hybrid price premium

Dc    = (rand(nobs,1)>0.75);         %college graduate

De    = (rand(nobs,1)>0.65);         %membership in env. group

```

```

X     = [ones(nobs,1) g p Dc De];
k     = size(X,2);
ystar = X*beta - randn(nobs,1);
y     = (ystar>0);
q     = 2*y - 1;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
@
@      Problem Set #2 - Econ 673 - Spring 2008
@      Question 2
@
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

@      Note: MAXLIK is a special routine and you have to tell GAUSS that your
@      will be using it
@
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

library maxlik;
#include maxlik.ext;
maxset;

```

```

output file = Prob3q2.out reset;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
@
@      The first step in this program is to generate the data
@
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

/*      specify parameter values */
beta0 = -1.0;
betag = 1.2;
betap = -.004;
betac = 1.0;
betae = 1.0;
beta  = beta0|betag|betap|betac|betae;

```

```

/*      generate explanatory variables */
nobs  = 1000;
a     = 2.5;
b     = 3.5;
mu    = 2.9;
sig   = 0.5;
ua    = cdfn((a - mu)/sig);
ub    = cdfn((b - mu)/sig);
u     = ua + (ub - ua)*rndu(nobs,1);
g     = mu + sig*cdfni(u);           /*gasoline prices*/

p     = 500 + 1500*rndu(nobs,1);     /*hybrid price premium*/

Dc    = (rndu(nobs,1).>0.75);         /*college graduate*/

De    = (rndu(nobs,1).>0.65);         /*membership in env. group*/

```

```

X     = ones(nobs,1)~g~p~Dc~De;
k     = cols(X);
ystar = X*beta - rndn(nobs,1);
y     = (ystar.>0);
q     = 2*y - 1;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%      Generate Summary Statistics
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
z      = [y X];
meanz  = mean(z);
stdz   = std(z);
maxz   = max(z);
minz   = min(z);
fprintf('\n');
fprintf('summary statistics \n');
fprintf('***** \n');
fprintf('      mean      std      min      max \n');

for i = 1:size(meanz,2);
    fprintf(' %10.4f %10.4f %10.4f %10.4f\n', ...
           meanz(1,i), stdz(1,i), minz(1,i), maxz(1,i));
end;
fprintf('\n');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%      Do-loop for Newton Raphson routine
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
toler  = 0.000001;
convg  = 0;
startv = zeros(k,1);
iter   = 1;
imax   = 20;
betaold = startv;
while ((iter<=imax) & (convg == 0));
    index = X * betaold;
    %
    %      Compute gradient
    %
    %
    lambda = q .* normpdf(index) ./ normcdf(q .* index);
    grad   = (lambda'*X)';
    %
    %      Compute Hessian
    %
    %
    hess   = zeros(k,k);
    for i = 1:nobs;
        hess = hess - lambda(i,1)*(lambda(i,1) + index(i,1))...
                *X(i,:)'*X(i,:);
    end;
end;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
@
@      Generate Summary Statistics
@
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
z      = y~X;
meanz  = meanc(z)';
stdz   = stdc(z)';
maxz   = maxc(z)';
minz   = minc(z)';
print  ;
print  "summary statistics";
print  "*****";
print  "      mean      std      min      max ";
format /m1 /rd 10,4;
length = cols(meanz);
for i (1,length,1);
    print meanz[1,i] stdz[1,i] minz[1,i] maxz[1,i];
endfor;
print  ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
@
@      Do-loop for Newton Raphson routine
@
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
toler  = 0.000001;
convg  = 0;
startv = zeros(k,1);
iter   = 1;
imax   = 20;
betaold = startv;
do while ((iter<=imax).and(convg == 0));
    index = X * betaold;
    %
    %      Compute gradient
    %
    %
    lambda = q .* pdfn(index) ./ cdfn(q .* index);
    grad   = (lambda'*X)';
    %
    %      Compute Hessian
    %
    %
    hess1  = zeros(k,k);
    for i (1,nobs,1);
        hess1 = hess1 - lambda[i,1]*(lambda[i,1] + index[i,1])
                *X[i,:]'*X[i,:];
    endfor;
endfor;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%      Compute updated parameter vector and check for convergence
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
delta = inv(hess)*grad;
betanew = betaold - delta;
reldel = delta ./ betaold;
info = [betaold betanew delta reldel];
if abs(max(reldel)) < toler;
    convg = 1;
end;
betaold = betanew;
fprintf(...
'iteration %2.0f: coefficients are %4.4f %4.4f %4.4f %4.4f %4.4f \n',...
iter, betanew');
iter = iter + 1;
end;
if convg == 1;
    fprintf('convergence achieved after %2.0f: iterations \n',iter);
else;
    fprintf('convergence achieved after %2.0f: iterations \n',iter);
end;
beta = betanew;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%      Generate Asymptotic Standard Errors
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lambda1 = normpdf(index) ./ normcdf(index);
lambda0 = -normpdf(index) ./ normcdf(-index);
Ehess = zeros(k,k);
for i = 1:nobs;
    Ehess = Ehess - lambda1(i,1)*lambda0(i,1)*X(i,:)'*X(i,:);
end;
varcov = inv(Ehess);
stderr = sqrt(diag(varcov));
fprintf('\n');
fprintf('results of estimation \n');
fprintf('***** \n');
fprintf(' estimate      std. err. \n');
for i = 1:size(beta,1);
    fprintf(' %8.4f      %8.4f \n', beta(i,1), stderr(i,1));
end;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%      Compute fitted probabilities and marginal effects
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
riter = 1000;
XA = [1 2.5 1500 0 0];
XB = [1 3.5 500 1 1];
XC = [1 3.0 1000 1 0];

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
@
@      Compute updated parameter vector and check for convergence
@
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
delta = inv(hess1)*grad;
betanew = betaold - delta;
reldel = delta ./ betaold;
info = betaold~betanew~delta~reldel;
if abs(max(reldel)) < toler;
    convg = 1;
endif;
betaold = betanew;
print "iteration: " iter "coefficients are" betanew;

iter = iter + 1;
end;
if convg == 1;
    print "convergence achieved after " iter "iterations";
else;
    print "convergence not achieved after " iter "iterations";
endif;
beta = betanew;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
@
@      Generate Asymptotic Standard Errors
@
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lambda1 = pdfn(index) ./ cdfn(index);
lambda0 = -pdfn(index) ./ cdfn(-index);
Ehess = zeros(k,k);
for i (1,nobs,1);
    Ehess = Ehess - lambda1[i,1]*lambda0[i,1]*X[i,.]'*X[i,.];
endfor;
varcov = inv(Ehess);
stderr = sqrt(diag(varcov));
print
;
print "results of estimation";
print "*****";
print " estimate      std. err.";
print beta~stderr;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
@
@      Compute fitted probabilities and marginal effects
@
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
riter = 1000;
XA = {1 2.5 1500 0 0};
XB = {1 3.5 500 1 1};
XC = {1 3.0 1000 1 0};

```

```

XA1      = [1 2.5 1500 0 1];
XB1      = [1 3.5  500 1 1];
XC1      = [1 3.0 1000 1 1];
X1       = X - [zeros(nobs,2) 500*ones(nobs,1) zeros(nobs,2)];
pfitA    = zeros(riter,1);
pfitB    = zeros(riter,1);
pfitC    = zeros(riter,1);
mfitA    = zeros(riter,1);
mfitB    = zeros(riter,1);
mfitC    = zeros(riter,1);
dfitA    = zeros(riter,1);
dfitB    = zeros(riter,1);
dfitC    = zeros(riter,1);
rfit     = zeros(riter,1);
betar    = repmat(beta',riter,1) + randn(riter,5) * chol(varcov);
for i = 1:riter;
    pfitA(i,1) = normcdf(XA*betar(i,:));
    pfitB(i,1) = normcdf(XB*betar(i,:));
    pfitC(i,1) = normcdf(XC*betar(i,:));
    mfitA(i,1) = normpdf(XA*betar(i,:))*betar(riter,2);
    mfitB(i,1) = normpdf(XB*betar(i,:))*betar(riter,2);
    mfitC(i,1) = normpdf(XC*betar(i,:))*betar(riter,2);
    dfitA(i,1) = normcdf(XA1*betar(i,:)) - pfitA(i,1);
    dfitB(i,1) = normcdf(XB1*betar(i,:)) - pfitB(i,1);
    dfitC(i,1) = normcdf(XC1*betar(i,:)) - pfitC(i,1);
    rfit(i,1) = mean(normcdf(X1*betar(i,:))) - mean(normcdf(X*betar(i,:)));
end;
fprintf('\n');
fprintf('Fitted probabilities  \n');
fprintf('***** \n');
fprintf(' type      estimate      std. err. \n');
fprintf('  I         %8.4f      %8.4f \n', mean(pfitA), std(pfitA));
fprintf('  II        %8.4f      %8.4f \n', mean(pfitB), std(pfitB));
fprintf('  III       %8.4f      %8.4f \n', mean(pfitC), std(pfitC));

fprintf('\n');
fprintf('Marginal Impact of Gas Prices  \n');
fprintf('***** \n');
fprintf(' type      estimate      std. err. \n');
fprintf('  I         %8.4f      %8.4f \n', mean(mfitA), std(mfitA));
fprintf('  II        %8.4f      %8.4f \n', mean(mfitB), std(mfitB));
fprintf('  III       %8.4f      %8.4f \n', mean(mfitC), std(mfitC));

fprintf('\n');
fprintf('Change in Pr(y=1) from joining Env. Group  \n');
fprintf('***** \n');
fprintf(' type      estimate      std. err. \n');
fprintf('  I         %8.4f      %8.4f \n', mean(dfитA), std(dfитA));
fprintf('  II        %8.4f      %8.4f \n', mean(dfитB), std(dfитB));
fprintf('  III       %8.4f      %8.4f \n', mean(dfитC), std(dfитC));

conf     = 0.95;
fprintf('\n');
fprintf('Change in Pr(y=1) from $500 rebate  \n');

```

```

XA1      = {1 2.5 1500 0 1};
XB1      = {1 3.5  500 1 1};
XC1      = {1 3.0 1000 1 1};
X1       = X - (zeros(nobs,2)~(500*ones(nobs,1))~zeros(nobs,2));
pfitA    = zeros(riter,1);
pfitB    = zeros(riter,1);
pfitC    = zeros(riter,1);
mfitA    = zeros(riter,1);
mfitB    = zeros(riter,1);
mfitC    = zeros(riter,1);
dfitA    = zeros(riter,1);
dfitB    = zeros(riter,1);
dfitC    = zeros(riter,1);
rfit     = zeros(riter,1);
betar    = beta' + randn(riter,5) * chol(varcov);
for i (1,riter,1);
    pfitA[i,1] = cdfn(XA*betar[i,.]);
    pfitB[i,1] = cdfn(XB*betar[i,.]);
    pfitC[i,1] = cdfn(XC*betar[i,.]);
    mfitA[i,1] = pdfn(XA*betar[i,.])*betar[riter,2];
    mfitB[i,1] = pdfn(XB*betar[i,.])*betar[riter,2];
    mfitC[i,1] = pdfn(XC*betar[i,.])*betar[riter,2];
    dfitA[i,1] = cdfn(XA1*betar[i,.]) - pfitA[i,1];
    dfitB[i,1] = cdfn(XB1*betar[i,.]) - pfitB[i,1];
    dfitC[i,1] = cdfn(XC1*betar[i,.]) - pfitC[i,1];
    rfit[i,1] = meanc(cdfn(X1*betar[i,.])) - meanc(cdfn(X*betar[i,.]));
endfor;
print
;
print "Fitted probabilities";
print "*****";
print " type      estimate      std. err. ";
print "  I         " meanc(pfitA) stdc(pfitA);
print "  II        " meanc(pfitB) stdc(pfitB);
print "  III       " meanc(pfitC) stdc(pfitC);

print
;
print "Marginal Impact of Gas Prices ";
print "*****";
print " type      estimate      std. err. ";
print "  I         " meanc(mfitA) stdc(mfitA);
print "  II        " meanc(mfitB) stdc(mfitB);
print "  III       " meanc(mfitC) stdc(mfitC);

print
;
print "Change in Pr(y=1) from joining Env. Group";
print "*****";
print " type      estimate      std. err. ";
print "  I         " meanc(dfитA) stdc(dfитA);
print "  II        " meanc(dfитB) stdc(dfитB);
print "  III       " meanc(dfитC) stdc(dfитC);

conf     = 0.95;
print
;
print "Change in Pr(y=1) from $500 rebate";

```

```

fprintf('***** \n');
fprintf('          %2.0f percent confidence interval \n',100*conf);
fprintf('estimate  std. err.  lower bound  upper bound\n');
srebate = sort(rfit);
lb      = srebate(floor((1 - conf)*riter/2),1);
ub      = srebate(ceil((1 + conf)*riter/2),1);
fprintf('%8.4f  %8.4f  %8.4f  %8.4f \n',...
        mean(rebate), std(rebate), lb, ub);
diary off;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%      Estimate the model using fminunc
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
b0 = beta;
b0 = [1;0;0;0;0];

options = optimset('fminunc');
options = optimset('GradObj','off',...
                  'Hessian','off',...
                  'LargeScale','on',...
                  'Display','iter',...
                  'MaxIter',10000,...
                  'TolX', 1.0000e-008,...
                  'TolFun', 1.0000e-08,...
                  'DerivativeCheck','off',...
                  'Diagnostics','on',...
                  'MaxFunEvals', 50000);
[b_ml,fval,exitflag,output,grad,hessian] = ...
    fminunc(@(d) logit(d,y,X),b0,options);
diary on;
index = X * b_ml;
lambda1 = normpdf(index) ./ normcdf(index);
lambda0 = -normpdf(index) ./ normcdf(-index);
Ehess = zeros(k,k);
for i = 1:nobs;
    Ehess = Ehess - lambda1(i,1)*lambda0(i,1)*X(i,:)'*X(i,:);
end;
varcov = inv(Ehess);
stderr = sqrt(diag(varcov));
fprintf('\n');
fprintf('fminunc results of estimation \n');
fprintf('***** \n');
fprintf(' estimate  std. err. \n');
for i = 1:size(beta,1);
    fprintf(' %8.4f  %8.4f \n', b_ml(i,1), stderr(i,1));
end;

```

```

print "*****";
print "          " 100*conf "percent confidence interval";
print "estimate  std. err.  lower bound  upper bound";
srebate = sortc(rfit,1);
lb      = srebate[floor((1 - conf)*riter/2),1];
ub      = srebate[ceil((1 + conf)*riter/2),1];
print meanc(rfit) stdc(rfit) lb ub;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
@
@      Estimate the model using fminunc
@
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
b0 = beta;
b0 = {1,0,0,0,0};
dta = y~X;

_max_GradProc = &gradprbt;

{b_ml,logl,g,varcov,ret}=maxprt(maxlik(dta,0,&prbt,b0));

stderr = sqrt(diag(varcov));
print ;
print "maxlik results of estimation";
print "*****";
print " estimate  std. err.";
print b_ml~stderr;
end;

```