Problem Set 3 Solutions

Tyler Ransom

1 Main Problem Set Code

```
clear all;
clc;
options = ...
    optimset('Disp','iter-detailed','MaxFunEvals',1e12,'MaxIter',1e6,'TolX',1e-9,'TolFun',1
rand('seed',1234); randn('seed',1234);
% %% Problem 1(a)
% do the quadrature of the integral of f(x) = e^(-x^2))
tic;
int_hat_quad = quad(@(x) exp(-x.^2),-10,1);
quad_time=toc;
%% Problem 1(b)
% do the simulated integral over 7 different sample sizes
n = 2*[1e1 1e2 1e3 1e4 1e5 1e6 1e7]';
% initialize integral and time matrices
int_hat = zeros(size(n));
int_time = zeros(size(n));
% create a loop that does the simulation over each of the 7 different sizes
for i = 1:length(n)
    tic;
    area        = 11;
    grid        = [-11*rand(n(i),1)+1 rand(n(i),1)];
    below       = (grid(:,2)<=exp(-(grid(:,1)).^2))&(grid(:,2)>0);
    above       = (grid(:,2)>=exp(-(grid(:,1)).^2))&(grid(:,2)<0);
    int_hat(i)  = area*(sum(below)-sum(above))/length(grid);
    int_time(i) = toc;
end
%% Problem 2(a)
load nlsw88
B = 10000;
[bootstat,bootsam] = bootstrp(B,@mean,ttl_exp);
bootmean2a = mean(bootstat)
bootSE2a = sqrt((1/(B-1))*(bootstat-bootmean2a)'*(bootstat-bootmean2a))
popmean2a = mean(ttl_exp)
popSE2a = std(ttl_exp)/sqrt(length(ttl_exp))
%% Problem 2(b)
bootstat = zeros(B,1);
for b = 1:B
    bootstat(b) = median(log(randsample(wage,length(wage),true)));
end
```

```
bootmedian2b = mean(bootstat)
bootSE2b = sqrt((1/(B-1))*(bootstat-bootmedian2b)'*(bootstat-bootmedian2b))
popmedian2b = median(log(wage))
popSE2b = 1.253*std(log(wage))/sqrt(length(wage))
%% Problem 2(c)
X = [ones(size(wage)) age race==2 race==3 collgrad grade married south...
    c_city union ttl_exp tenure age.^2 hours never_married];
y = log(wage);
% create a vector that is 1 if all obs are there; 0 otherwise:
subset1 = ¬isnan(wage)&¬isnan(age)&¬isnan(race)&¬isnan(married)...
         &¬isnan(grade)&¬isnan(collgrad)&¬isnan(south)&¬isnan(c_city)...
         &¬isnan(union)&¬isnan(ttl_exp)&¬isnan(tenure)&¬isnan(hours)...
         &¬isnan(never_married);
y = log(wage(subset1)); %drop missing observations from y
X = X(subset1,:);       %drop missing observations from X
nb = size(X,2);         %initialize the number of regressors for later use
% Initialize the baseline closed-form OLS formulas (for later comparison)
[bpop2c,¬,¬,¬,stats]=regress(y,X);
sepop2c = sqrt(diag(stats(end)*((X'*X)\eye(size(X,2)))));

% do the bootstrap
b = regress(y,X);
yfit = X*b;
resid = y - yfit;

% compare results
bpop2c
sepop2c
bootb2c = b
bootSE2c = std(bootstrp(B, @(bootr) regress(yfit+bootr,X), resid))'
%% Problem 2(d)
X = [ones(size(wage)) age race==2 race==3 collgrad grade married south...
    c_city union ttl_exp tenure age.^2 hours never_married];
y = log(wage);
bootstat = zeros(size(X,2),B);
unioner = union;
IDprime = [1:1:length(idcode)]';
for b = 1:B
    IDtemp = sort(randsample(IDprime,length(idcode),true));
    yboot = log(wage(IDtemp));
    Xboot = X(IDtemp,:);
    bootstat(:,b) = regress(yboot,Xboot);
end
bootb2d = mean(bootstat,2)
bootSE2d = ...
   sqrt(diag((1/(B-1))*(bootstat-repmat(bootb2d,1,B))*(bootstat-repmat(bootb2d,1,B))'))
bpop2d = bpop2c
sepop2d = sepop2c
%% Problem 3(a)
load nlsy97
N = length(ID);
T = 5;
activityt = activity';
log_waget = log_wage';
```

```
hgct      = hgc';
expert    = exper';
Diplomat  = Diploma';
AAt       = AA';
BAt       = BA';
malet     = repmat(male,1,T)';
AFQTt     = repmat(AFQT,1,T)';
Mhgct     = repmat(Mhgc,1,T)';
X = [ones(N*T,1) malet(:) AFQTt(:) Mhgct(:) hgct(:) expert(:) Diplomat(:) ...
    AAt(:) BAt(:)];
y = activityt(:);
y = (y==2);
[b_no_het,¬,stats]=glmfit(X,y,'binomial','link','probit','constant','off');
SE_b_no_het = stats.se;
results_no_het = [b_no_het SE_b_no_het];
%% Problem 3(b)
d = (activity==2);
[N,T] = size(activity);
K = 9;
% create data matrix which is NxKxT (for ease of doing heterogeneity)
X = zeros(N,K,T);
for t=1:T
    X(:,:,t) = [ones(N,1) male AFQT Mhgc hgc(:,t) exper(:,t) Diploma(:,t) ...
        AA(:,t) BA(:,t)];
end
bstart = .5*b_no_het.*rand(size(b_no_het))-.25*b_no_het;
b_disc_RE                                = ...
    fminsearch('probit_het_disc',[bstart;.4*rand],options,X,d);
b_disc_RE                                = ...
    fminsearch('probit_het_disc',b_disc_RE,options,X,d);
b_disc_RE                                = ...
    fminsearch('probit_het_disc',b_disc_RE,options,X,d);
b_disc_RE                                = ...
    fminsearch('probit_het_disc',b_disc_RE,options,X,d);
[b_disc_RE,l_disc_RE,¬,¬,¬,h_disc_RE] = fminunc   ...
    ('probit_het_disc',b_disc_RE,options,X,d);
SE_b_disc_RE = sqrt(diag(inv(h_disc_RE)));
results_disc_RE = [b_disc_RE SE_b_disc_RE];
%% Problem 3(c)
[b_RE]                  = ...
    fminsearch('probit_het_grid',[bstart;.4*rand],options,X,d);
[b_RE,l_RE,¬,¬,¬,h_RE] = fminunc   ('probit_het_grid',b_RE,options,X,d);
SE_b_RE = sqrt(diag(inv(h_RE)));
results_RE = [b_RE SE_b_RE];
results_compare=[[b_no_het;NaN;NaN] [b_disc_RE;l_disc_RE] [b_RE;l_RE]];
save estimation_results int_hat_quad quad_time int_hat int_time n ...
    results_no_het results_disc_RE results_RE results_compare
% save estimation_results int_hat_quad quad_time int_hat int_time n ...
    results_no_het results_disc_RE
```

2 MLE Function Code

```
function like = probit_het_disc(b,X,y)
%PROBIT_HET_GRID Computes discrete random effects for a binary probit
% [like] = probit_het(b,X,y)
% Estimates parameters b of a random-effects probit given data X and y
% Note that the std dev of the random effect is the last element of b
% Heterogeneity has finite number of types
[N,K,T]=size(X);
beta  = b(1:end-1);
pi    = b(end);
% alpha = b(end-1);
alpha = 1;

% set up the multiplication X*beta
X2 = permute(X, [1 3 2]);
Xb  = reshape(X2,N*T,K)*beta;
Xb1 = reshape(Xb,N,T);


%%% test code
% test = beta'*squeeze(X(1,:,:));
% First row of Xb1 should be the same as test
%%%

like_1 = prod(normcdf(Xb1+alpha).^(y==1) .* (1-normcdf(Xb1+alpha)).^(y==0),2);
like_2 = prod(normcdf(Xb1+0    ).^(y==1) .* (1-normcdf(Xb1+0    )).^(y==0),2);

like = -sum(log(pi*like_1+(1-pi)*like_2));
end
```

```
function like = probit_het_grid(b,X,y)
%PROBIT_HET_GRID Computes random effects for a binary probit
% [like] = probit_het(b,X,y)
% Estimates parameters b of a random-effects probit given data X and y
% Note that the std dev of the random effect is the last element of b
% Integration method is composite Simpson's Rule (by hand)
[N,K,T]=size(X);

% set up the grid matrices for integration
% grid should be NxTxG
step = .03;
grid = [-3:step:3]'*ones(1,T);
G = size(grid,1);
grid = grid';
grid = reshape(grid,[1 T G]);
grid = repmat(grid,[N 1 1]);

% set up the multiplication X*beta
X2 = permute(X, [1 3 2]);
Xb  = reshape(X2,N*T,K)*b(1:end-1);
```

```
Xb1 = reshape(Xb,[N T 1]);
Xb1 = repmat(Xb1,[1 1 G]);
y   = reshape(y,[N T 1]);
y   = repmat(y,[1 1 G]);

% form matrix of normal pdf weights (i.e. f(x) in an integrand)
% these weights matrices should be of size NxG
normpdf_weights = normpdf(squeeze(grid(:,1,:)),0,b(end));
% form matrix of composite simpson's rule weights
simpson_weights = ones(size(squeeze(grid(:,1,:))));
for g=1:G
    if mod(g,2)==0;
        simpson_weights(:,g)=2;
    else
        simpson_weights(:,g)=4;
    end
end
simpson_weights(:,1)=ones(N,1);
simpson_weights(:,end)=ones(N,1);
simpson_weights = (step/3)*simpson_weights;

if sum(simpson_weights.*normpdf_weights,2)-1>1e-3
    error('normdpdf doesn`t integrate to 1')
end

like = -sum(log(sum(squeeze(prod(normcdf(Xb1+grid).^((y==1)) .* ...
   (1-normcdf(Xb1+grid)).^(y==0),2)).*normpdf_weights.*simpson_weights,2)),1);
end
```