ECONOMETRICS II, Fall 2024

Bent E. Sørensen

Midterm 1—September 25, 2024.

Each sub-question in the following carries equal weight except if otherwise noted.

1. (40%)

a) (5%) Write down the formula for the White robust standard error estimator.

b) (10%) Assume you are estimating the model

$$Y_i = aX_i + u_i \ ,$$

by OLS. Here a is a scalar and we assume for simplicity that there is no intercept and that in the true underlying model (not censured or truncated) the error term has mean 0.
Assume that you only have 2 observations: $X' = (1, 2)$, $Y' = (2, 5)$.
Find the OLS estimate $\hat{a}$ and the residuals.

c) (5%) Calculate the White robust standard error.

d) (20%) Now assume that the two observations above form a group and we have a second group where (for computational simplicity) we also assume $X' = (1, 2)$, $Y' = (2, 5)$. So your data are now $X' = (1, 2, 1, 2)$, $Y' = (2, 5, 2, 5)$.
Estimate a again [(or you can use the estimate you found in part a))] and calculate the residuals. Then calculate the Robust standard error if you cluster on the two groups. (The question is about the standard error estimation.)

2. (20%) Assume a population has a continuous outcome determined by the model

$$y_i = \beta x_i + u_i \, ,$$

where u_i is a standard normal distribution. You can assume there is only one regressor. However, a researcher only observe y_i with probability p. The researcher does observe N

individuals even only a sub-sample will have y_i observed (selected). For the observed sub-sample (that is, conditional on being selected), the distribution of u_i is Normal with variance one but mean $(1-p)\beta\overline{X}$. For the non-selected individuals the mean of u_i is $-p\beta\overline{X}$. For simplicity, assume that we have a large sample so that mean $\overline{X}$ can be considered the same in the full and in the selected sample.

i) (5%) Starting from the conditional distributions, verify from the that u_i has mean 0.
ii) (15%) Write down the full log-likelihood for the probability of being selected and outcome y_i (for observed x_i).

3. (10%) Write down the log-likelihood function for a sample $i = 1, ..., N$ where

$$y_i = X_i\beta + u_i$$

where u_i is $N(0, \sigma^2)$ and the data are censored such that if the true $y_i < K$, we only observe $y_i = K$.

4. (9%)

a) Write down the latent-variable model used to derive the Probit model and write down the log-likelihood function for the probit model for a sample of N observations.

5. (21%) Go through the Matlab code below.

a) Explain in detail what model(s) the program estimate(s).
b) What does the variable PPPP do?
c) Near the end of the program there is a line where it says "WHAT GOES HERE." What goes there? .

```
************************************************
Question %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Econometrics 2
```

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This code simulates and estimates the parameters of ...something.....
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc
clear

global x T PPPP

% Set up the true parameters and placeholders for the results.

T = 50;                                                                 % Number of
PPPP = 1;                                                                 % xxx orde
sigma = 2;                                                              % Standard d
beta1 = 0.5;                                                            % xxxcoeffic

if PPPP == 1                                                              % For xxx.
    sim = 10;                                                           % Number of
    results_mat = zeros(sim,3);                                         % Results ma
    se_mat = zeros(sim,3);                                              % Standard e
    t_mat = zeros(sim,3);                                               % t-stats.

elseif PPPP == 2
    beta2 = 0.4;                                                        % xx coeffic
    sim = 5;                                                            % Number of
    results_mat = zeros(sim,4);                                         % Results ma
    se_mat = zeros(sim,4);                                              % Standard e
```

```
    t_mat = zeros(sim,4);                                                % t-stats.

end

% Begin Simulation.

for s = 1:sim

    % Generate the data using the model.
    %           xxx
    %          xxxx

    u = normrnd(0,sigma,T,1);
    x = zeros(T,1);

    if PPPP== 1                                                           % For proce

        x(1) = u(1) + beta1*normrnd(0,sigma,1,1);

        for j = 2:T

            x(j) = u(j) + beta1*u(j-1);

        end

        init = [1 1 0.1];                                                % Initial va

    elseif PPPP == 2

        c = normrnd(0,sigma,1,1);
```

```
        x(1) = u(1) + beta1*c + beta2*normrnd(0,sigma,1,1);
        x(2) = u(2) + beta1*u(1) + beta2*c;

        for j = 3:T

            x(j) = u(j) + beta1*u(j-1) + beta2*u(j-2);

        end

        init = [1 1 0.1 0.1];                                          % Initial va

    end

    options = optimset('Display','off');                               % Turn off t
    [b_mle,~,~,~,~,hess] = fminunc('logl_MA',init,options);            % Minimizati

    results_mat(s,:) = b_mle';                                         % Store esti
    se_mat(s,:) = sqrt(diag(inv(hess)))';                              % Store stan
    t_mat(s,:) = results_mat(s,:)./se_mat(s,:);                        % Store t-st

end

% Display the results of each simulation.

for k = 1: size(results_mat,1)
    fprintf('Simulation %d \n',k)
    fprintf('          b            SE             t \n')
    fprintf('mu      %0.4f       %0.4f         %0.4f    \n', results_mat(k,1), se_mat(k,
    fprintf('sigma   %0.4f       %0.4f         %0.4f    \n', results_mat(k,2), se_mat(k,
    fprintf('beta1   %0.4f       %0.4f         %0.4f    \n', results_mat(k,3), se_mat(k,
```

```
    if PPPP== 2
        fprintf('beta2   %0.4f       %0.4f         %0.4f    \n', results_mat(k,4), se_ma
    end
    fprintf('\n')
end

fprintf('-----------------------------------------------------------------\n')
fprintf('\n')

% Display the empirical means and standard deviations.

fprintf('Empirical Results \n')
fprintf('         Mean       Std.Dev   \n')
fprintf('mu      %0.4f       %0.4f     \n', mean(results_mat(:,1)), std(results_mat(:,1)
fprintf('sigma   %0.4f       %0.4f     \n', mean(results_mat(:,2)), std(results_mat(:,2)
fprintf('beta1   %0.4f       %0.4f     \n', mean(results_mat(:,3)), std(results_mat(:,3)
if MMMM == 2
fprintf('beta2   %0.4f       %0.4f     \n', mean(results_mat(:,4)), std(results_mat(:,4)
end
fprintf('\n')


function [ L ] = logl_name( b0 )
% Loglikelihood for ..something....

global x T PPPP

omega = zeros(T,T);                                                    % Placeholde
mean = b0(1);                                                          % Mean.
stddev = b0(2);                                                        % Standard d
```

```
theta1 = b0(3);                                                    % xxx coeffi
mu = ones(T,1)*mean;                                               % Mean Vecto

    if PPPP == 1                                                     % For zz p

        omega = omega + eye(T).*(stddev^2).*(1+theta1^2);          % Fill in th

        omega(2,1) = (stddev^2)*theta1;                            % Fill in th
        omega(T-1,T) = omega(2,1);

        for i = 2:T-1
            omega(i-1,i) = (stddev^2)*theta1;
            omega(i+1,i) = omega(i-1,i);
        end

    elseif PPPP == 2                                                 % For zzz

        theta2 = b0(4);                                            % xx coeffic

        omega = omega + eye(T).*(stddev^2).*(1+(theta1^2)+(theta2^2));   % Fill in th

        omega(2,1) = (stddev^2).*(theta1 + (theta1*theta2));       % Fill in th
        omega(T-1,T) = omega(2,1);

        for i = 2:T-1
            omega(i-1,i) = (stddev^2).*(theta1 + (theta1*theta2));
            omega(i+1,i) = omega(i-1,i);
        end

        omega(3,1) = (stddev^2)*theta2;                            % Fill in .
```

```
        omega(4,2) = omega(3,1);
        omega(T-2,T) = omega(3,1);
        omega(T-3,T-1) = omega(3,1);

        for i = 3:T-2
        WHAT GOES HERE? (Two lines)
        end

    end


L = -0.5*T*log(2*pi) - 0.5*log(abs(det(omega))) ...                    % Loglikelih
                                        - 0.5*(x-mu)'*inv(omega)*(x-mu);

L = -L;                                                                % Negative o

end
```