MATLAB codes

Listing 1.1: Dummy File For Reading Data

```
data = readmatrix('/Volumes/External/AdvMicroEconMetrics/PS_2/
   data_ps2.csv');

onesVector = ones(size(data, 1), 1);

D = data(:, 5);
T = data(:, 1:4);
W = [onesVector, data(:, 7:20)];
X = [onesVector, data(:, [7, 19, 20])];
Y = data(:, 6);
Z = [onesVector, data(:, [7, 19:22])];
```

Listing 1.2: OLS Regression

```
function [beta, sigma] = OLS_Est(X, Y)
%OLS regression
beta = inv(X' * X) * X' * Y;
eps = Y - X*beta;
sigma = sqrt((eps' * eps) / (size(Y, 1) - size(X, 2)));
end
```

Listing 1.3: MLE

```
function output = loglikelihood_ps2(init)


% init -> initial guess
global D T W X Y Z

colW = size(W, 2);
colX = size(X, 2);
colZ = size(Z, 2);

% Identification auxiliary
aux = [colX, ...
    2 * colX, ...
    2 * colX + colZ, ...
```

```
    2 * colX + colZ + colW, ...
    2 * colX + colZ + 2 * colW, ...
    2 * colX + colZ + 3 * colW, ...
    2 * colX + colZ + 4 * colW, ...
    2 * colX + colZ + 4 * colW + 1, ...
    2 * colX + colZ + 4 * colW + 2, ...
    2 * colX + colZ + 4 * colW + 3, ...
    2 * colX + colZ + 4 * colW + 4, ...
    2 * colX + colZ + 4 * colW + 5, ...
    2 * colX + colZ + 4 * colW + 6, ...
    2 * colX + colZ + 4 * colW + 7, ...
    2 * colX + colZ + 4 * colW + 8, ...
    2 * colX + colZ + 4 * colW + 9, ...
    2 * colX + colZ + 4 * colW + 10, ...
    2 * colX + colZ + 4 * colW + 11, ...
    2 * colX + colZ + 4 * colW + 12, ...
    2 * colX + colZ + 4 * colW + 13];

% Parameters
beta_0 = init(1:aux(1));
beta_1 = init(aux(1) + 1:aux(2));
beta_D = init(aux(2) + 1:aux(3));
omega_T1 = init(aux(3) + 1:aux(4));
omega_T2 = init(aux(4) + 1:aux(5));
omega_T3 = init(aux(5) + 1:aux(6));
omega_T4 = init(aux(6) + 1:aux(7));
sigma_0 = init(aux(7) + 1:aux(8));
sigma_1 = init(aux(8) + 1:aux(9));
sigma_T1 = init(aux(9) + 1:aux(10));
sigma_T2 = init(aux(10) + 1:aux(11));
sigma_T3 = init(aux(11) + 1:aux(12));
sigma_T4 = init(aux(12) + 1:aux(13));
alpha_0 = init(aux(13) + 1:aux(14));
alpha_1 = init(aux(14) + 1:aux(15));
alpha_I = init(aux(15) + 1:aux(16));
alpha_T2 = init(aux(16) + 1:aux(17));
alpha_T3 = init(aux(17) + 1:aux(18));
alpha_T4 = init(aux(18) + 1:aux(19));
sigma_theta = init(aux(19) + 1:aux(20));

MLfunc = @(theta) (normpdf(Y - X * beta_0 - theta * alpha_0, 0,
   exp(sigma_0)) ...
    .* normpdf(Y - X * beta_1 - theta * alpha_1, 0, exp(sigma_1))
       ...
    .* normpdf(T(:, 1) - W * omega_T1 - theta, 0, exp(sigma_T1))
       ...
    .* normpdf(T(:, 2) - W * omega_T2 - theta * alpha_T2, 0, exp(
       sigma_T2)) ...
```

```
    .* normpdf(T(:, 3) - W * omega_T3 - theta * alpha_T3, 0, exp(
       sigma_T3)) ...
    .* normpdf(T(:, 4) - W * omega_T4 - theta * alpha_T4, 0, exp(
       sigma_T4)) ...
    .* (1 - normcdf(Z * beta_D + theta * alpha_I, 0, 1)) .^ (1 - D
       ) ...
    .* normcdf(Z * beta_D + theta * alpha_I, 0, 1) .^ D ...
    .* normpdf(theta, 0, exp(sigma_theta)));


% Can use Gauss- Hermit or Montecarlo
% However there is a tradeoff between computing time and
   programming G-H
% integration.
sol = integral(MLfunc, -Inf, Inf, 'ArrayValued', true);

output = -sum(log(sol));

end
```

Listing 1.4: Estimation of Parameters

```
clear
% Set Seet: LGMV
randn('seed', 180618)

% Set global
global D T W X Y Z

%check dummy file read.m
read;

[beta_T1, sigma_T1] = OLS_Est(W, T(:, 1));
[beta_T2, sigma_T2] = OLS_Est(W, T(:, 2));
[beta_T3, sigma_T3] = OLS_Est(W, T(:, 3));
[beta_T4, sigma_T4] = OLS_Est(W, T(:, 4));
[beta_D, sigma_D] = OLS_Est(Z, D); % Take sigma_D = 1, to simplify
[beta_1, sigma_1] = OLS_Est(X, Y);
[beta_0, sigma_0] = OLS_Est(X, Y);

[alpha_T1, alpha_T2, alpha_T3, alpha_T4, alpha_I, alpha_1, alpha_0
   , sigma_theta] = deal(1);

% Initial guesses for
guesses = [beta_0; beta_1; beta_D; beta_T1; beta_T2; beta_T3;
   beta_T4; ...
    sigma_0; sigma_1; sigma_T1; sigma_T2; sigma_T3; sigma_T4; ...
    alpha_0; alpha_1; alpha_I; alpha_T2; alpha_T3; alpha_T4; ...
```

```
    sigma_theta];

tic
options = optimoptions(@fminunc, 'Algorithm', 'quasi-newton', '
   Display', 'iter',...
    'GradObj', 'off', 'HessUpdate', 'bfgs', 'UseParallel', false,
       ...
    'TolFun', 1e-6, 'TolX', 1e-6, 'MaxIter', 10000, 'MaxFunEvals',
        10000);
[est_beta, est_F, exitflag, output, grad, hessian] = fminunc('
   loglikelihood_ps2', guesses, options);

runtime = toc;

se_Hess=sqrt(diag(inv(hessian)));
[est_beta se_Hess]
```