ECONOMETRICS II, Fall, 2017
Bent E. Sørensen

Final Exam - December 5, 2017

Each sub-question in the following carries equal weight.

1. (24%) Consider the (SURE) system of equations

$$y_i = X_i\alpha + u_i$$

and

$$w_i = Z_i\gamma + e_i\,,$$

where y_i and w_i are dependent variables, X_i and Z_i are row-vectors of fixed regressors, and u_i and e_i mean zero errors terms with $E\{u_i^2\} = \sigma_u^2$, $E\{e_i^2\} = \sigma_{e^2}$, $E\{u_i e_i\} = \sigma_{ue}$ and $E\{u_i u_j\} = E\{e_i e_j\} = 0$, $i \neq j$.

a) Write down the Feasible (two-stage) GLS estimator for the two equations.
b) Prove that if $X = Z$ the "combined" GLS estimator is identical to estimating the equations one by one using OLS.
c) Write down a Wald test for $\alpha_1 = \gamma_1$.

2. (30%) Consider the Matlab programs below and fill in the missing part/answering the questions.

3. (16%) Consider the model

$$y = X\alpha + u\,.$$

Assume the matrix of regressors is

$$X \;=\; \begin{pmatrix} x_1 \\ x_1 \\ x_1 \\ x_2 \\ x_2 \\ x_2 \end{pmatrix},$$

and the error variance matrix is

$$V \;=\; \sigma^2 \begin{pmatrix} 1 & \rho & \rho & 0 & 0 & 0 \\ \rho & 1 & \rho & 0 & 0 & 0 \\ \rho & \rho & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \rho & \rho \\ 0 & 0 & 0 & \rho & 1 & \rho \\ 0 & 0 & 0 & \rho & \rho & 1 \end{pmatrix}.$$

Find the variance of the OLS estimated ($\hat{\alpha}$) as a function of $X'X$, ρ, etc.

4. (18%) a) Write down the latent-variable model used to derive the ordered Probit model with 3 ordered outcomes and derive the probability of each outcome.

b) Write down the log-likelihood function for the *Logit* model for a sample of N observations.

5. (12%) Explain how the Newton algorithm for optimizing a concave function works. I want you to use symbols and do the math of a given "Newton step," but you can assume the function is univariate.

```
QUESTION 1. Clustering:
function ret = clusterreg(y, X, g, varargin)
% This routine performs OLS estimation with either one- or
% two-way cluster-robust standard errors.
%
% SYNTAX: ret = clusterreg(y, X, g, varargin)
%
% where y is the dependent variable, X is the matrix of regressors,
% g is the first dimension of clustering and h is the (optional)
% second dimension of clustering. Note: y, g, and h should be vectors of
% equal length and X should have the same number of observations as y.
%
% ret = [b, se, t], the estimated coefficients (b), the estimated standard
% errors (se), and t-statistics (t).

  if nargin < 3 || nargin > 4
    error('Either 1 or 2 cluster variables are required');
  end

  [N, k] = size(X);

  % Calculate (X'*X)^(-1)
  if N < 10000
      [q r] = qr(X,0);
      xpxi = (r'*r)\eye(k);
    else % use Cholesky for very large problems
      xpxi = (X'*X)\eye(k);
  end;

  % calculate Bhat.
  pinvX = pinv(X);
  b = pinvX*y;

  % calculate residuals
  e = y - X*b;

  % Calculate variance robust to clustering on the first variable supplied.
  varBhat = singlecluster(xpxi, X, e, g);

  % If two cluster variables are supplied, adjust for the second cluster
```

```
% variable.
if nargin == 4
    h = varargin{1};
    gh = [g h];
    % Add the component attributable to the second cluster
    temp =  varBhat + singlecluster(xpxi, X, e, h);
    % Subtract the variance attributable to the intersection cluster. See
    % p.8 of "Robust Inference with Multi-way Clustering"
    % by A. Colin Cameron, Jonah B. Gelbach, and Douglas L. Miller for more
    % detail.
    varBhat = temp -  singlecluster(xpxi, X, e, gh);
end

% Calculate standard errors and t-stats
se = sqrt(diag(varBhat));
t = b ./ se;

% Return the calculated values
ret = [b se t];

function varB = singlecluster(xpxi, X, e, g)
% Nested function that handles a single cluster.

% Identify unique clusters
  G = unique(g, 'rows');
  M = size(G,1);

  % Now calculate the "meat" of the sandwich variance estimator.
  % See "Robust Inference with Multi-way Clustering"
  % by A. Colin Cameron, Jonah B. Gelbach, and Douglas L. Miller for more
  % detail.
  mid = 0;

  % This code handles clusters defined by more than one variable. It
  % treats observations having the same value for ALL cluster
  % variables as members of the same cluster.
  for i=1:size(G,1)
    test=[1:size(g,1)]';
    for j=1:size(G,2)
      test2 = find(g(:,j)==G(i,j));
```

```
        test=intersect(test,test2);
      end
      X_g = X(test,:);
      e_g = e(test,:);
      mid = WHAT GOES HERE?;
    end;

    % Calculate cluster-robust variance matrix estimate
    q_c = (N-1)/(N-k)*M/(M-1);
    varB = q_c*xpxi * mid * xpxi;

  end

end
**************************************************
Question 2%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Econometrics 2
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This code simulates and estimates the parameters of an MA(1) and an
% MA(2) process.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc
clear

global x T MA

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

T = 50;                                                                      % Number of periods
MA = 1;                                                                      % MA order.
sigma = 2;                                                                   % Standard deviatio
beta1 = 0.5;                                                                 % MA coefficient.

if MA == 1                                                                   % For MA(1) process
    sim = 10;                                                                % Number of simula
```

```
    results_mat = zeros(sim,3);                                    % Results matrix.
    se_mat = zeros(sim,3);                                         % Standard errors.
    t_mat = zeros(sim,3);                                          % t-stats.

elseif MA == 2                                                     % For MA(2) process
    beta2 = 0.4;                                                   % MA coefficient.
    sim = 5;                                                       % Number of simulat
    results_mat = zeros(sim,4);                                    % Results matrix.
    se_mat = zeros(sim,4);                                         % Standard errors.
    t_mat = zeros(sim,4);                                          % t-stats.

end

% Begin Simulation.

for s = 1:sim

    % Generate the data using the model.
    %               MA(1):   x(t) = u(t) + beta1*u(t-1)
    %               MA(2):   x(t) = u(t) + beta1*u(t-1) + beta2*u(t-2)

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

    if MA == 1                                                     % For MA(1) process

        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 values.

    elseif MA == 2                                                 % For MA(2) process

        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 values.

    end

    options = optimset('Display','off');                               % Turn off the disp
    [b_mle,~,~,~,~,hess] = fminunc('logl_MA',init,options);            % Minimization.

    results_mat(s,:) = b_mle';                                         % Store estimates.
    se_mat(s,:) = sqrt(diag(inv(hess)))';                              % Store standard er
    t_mat(s,:) = results_mat(s,:)./se_mat(s,:);                        % Store t-stats.

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,1), t_m
    fprintf('sigma   %0.4f       %0.4f         %0.4f    \n', results_mat(k,2), se_mat(k,2), t_m
    fprintf('beta1   %0.4f       %0.4f         %0.4f    \n', results_mat(k,3), se_mat(k,3), t_m
    if MA == 2
        fprintf('beta2   %0.4f       %0.4f         %0.4f    \n', results_mat(k,4), se_mat(k,4)
    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 MA == 2
fprintf('beta2   %0.4f       %0.4f     \n', mean(results_mat(:,4)), std(results_mat(:,4)))
end
fprintf('\n')
```

```
function [ L ] = logl_MA( b0 )
% Loglikelihood for MA(1) and MA(2).

global x T MA

omega = zeros(T,T);                                                    % Placeholder for V
mean = b0(1);                                                          % Mean.
stddev = b0(2);                                                        % Standard deviatio
theta1 = b0(3);                                                        % MA coefficient.
mu = ones(T,1)*mean;                                                   % Mean Vector.

    if MA == 1                                                         % For MA(1) process

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

        omega(2,1) = (stddev^2)*theta1;                                % Fill in the first
        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 MA == 2                                                     % For MA(2) process
```

```
        theta2 = b0(4);                                                    % MA coefficient.

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

        omega(2,1) = (stddev^2).*(theta1 + (theta1*theta2));               % Fill in the first
        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 the secon
        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))) ...                        % Loglikelihood fu
                                        - 0.5*(x-mu)'*inv(omega)*(x-mu);

L = -L;                                                                    % Negative of logl

end
```