RICHMOND ESSIEKU PS 3 & PS4

PART 3

```
Rich_Data_ = New_Concatenated_Data1
matrixData = table2array(New_Concatenated_Data1);
matrixData = repmat(matrixData, 15, 1);
% Extract submatrices from A
Export = matrixData(:, 1:15);           % First 15 columns
size(Export)
distance = matrixData(:, 16:30);          % Next 15 columns
size(distance)
GDP = matrixData(:, 31);              % 31st column
size(GDP)
contiquity = matrixData(:, end-14:end);     % Last 15 columns
size(contiquity)
```

```
% % Take logs
log_(export) = log(X);
size(log_(export))
log_(distance) = log(distance);
size(log_(distance))
log_(contiquity) = log(contiquity);
size(log_(contiquity))
log_(GDP) = log(GDP);
size(log_(GDP))
```

```
Rich_Data_1_
```

```
Rich_Data_1_ = 3375×6 table
```

```
Rich_Data.Properties
```

```
ans = 
  TableProperties with properties:

             Description: ''
                UserData: []
          DimensionNames: {'Row'  'Variables'}
           VariableNames: {'log_export_'  'log_distance_'  'log_contiguity_'  'log_Pi_'  'log_Pj_'  'log_GDP_'}
    VariableDescriptions: {}
           VariableUnits: {}
      VariableContinuity: []
                RowNames: {}
        CustomProperties: No custom properties are set.
        Use addprop and rmprop to modify CustomProperties.
```

```
summary(Rich_Data_1_)
```

```
Variables:

    log_export_: 3375×1 double

        Values:

            Min          2.7391
            Median       9.6268
            Max          12.889

    log_distance_: 3375×1 double

        Values:

            Min          1.8353
            Median       3.7811
            Max          11.655

    log_contiguity_: 3375×1 double

        Values:

            Min            0
            Median         0
            Max            1

    log_Pi_: 3375×1 double

        Values:

            Min            0
            Median         0
            Max            1

    log_Pj_: 3375×1 double

        Values:

            Min            0
            Median         0
            Max            1

    log_GDP_: 3375×1 double

        Values:

            Min         -4.5491
            Median       1.2927
            Max          13.287
```

```
% Define the gravity model formula
% Define the gravity model formula
gravity_model_formula = 'log_export_~ log_GDP_ + log_distance_ + 
log_contiguity_ + log_Pi_ + log_Pj_';
```

```
 %gravity_model_formula = 'Rich_Data.log_export_~ Rich_Data.log_GDP_ -
Rich_Data.log_distance_ + Rich_Data.log_contiguity_ + Rich_Data.log_Pi_ +
Rich_Data.log_Pj_';



 % Estimate gravity model
 gravity_model = fitlm(Rich_Data, gravity_model_formula);



 % Display the regression results
 disp(gravity_model);
```

```
Linear regression model:
    log_export_ ~ 1 + log_distance_ + log_contiguity_ + log_Pi_ + log_Pj_ + log_GDP_

Estimated Coefficients:
                       Estimate        SE        tStat        pValue
                       _________    ________    _______    __________

    (Intercept)          12.973     0.055888     232.12             0
    log_distance_      -0.90555     0.016625    -54.468             0
    log_contiguity_    0.017231     0.039263    0.43886       0.66079
    log_Pi_             0.16729     0.048556     3.4453    0.00057736
    log_Pj_             0.16785     0.048767     3.4418    0.00058482
    log_GDP_          -0.049556     0.008778    -5.6455    1.7835e-08


Number of observations: 3375, Error degrees of freedom: 3369
Root Mean Squared Error: 0.697
R-squared: 0.839,  Adjusted R-Squared: 0.839
F-statistic vs. constant model: 3.51e+03, p-value = 0
```

```
 % Step 3: Report Point Estimates and Standard Errors
 disp('Point Estimates (Gravity Model):');
```

```
Point Estimates (Gravity Model):
```

```
 disp(gravity_model.Coefficients.Estimate);
```

```
   12.9726
   -0.9055
    0.0172
    0.1673
    0.1678
   -0.0496
```

```
 disp('Standard Errors (Gravity Model):');
```

```
Standard Errors (Gravity Model):
```

```
 disp(gravity_model.Coefficients.SE);
```

```
    0.0559
    0.0166
    0.0393
    0.0486
    0.0488
    0.0088
```

```
% Extract structural parameters
structural_parameters = table2array(gravity_model.Coefficients(2:end, 1));

% Display structural parameters
disp('Structural Parameters:');
```

```
Structural Parameters:
```

```
disp(structural_parameters);
```

```
   -0.9055
    0.0172
    0.1673
    0.1678
   -0.0496
```

```
Where b = 0.1673, p (rho) = 0.0172 and sigma = 0.822
```

```


% Step 4: Recover Standard Errors using Delta Method
% Assuming you have a function g(theta) representing the gravity model
% This is a simplified example; you'll need to adapt it to your specific model

% Define the Jacobian matrix of g(theta) with respect to the parameters
%Jacobian = [diff(g(theta), theta(1)), diff(g(theta), theta(2)),diff(g(theta), 
theta(3)), diff(g(theta), theta(4)),diff(g(theta), theta(5))];
% Calculate the standard errors using the Delta Method formula

%cov_matrix = gravity_model.CoefficientCovariance; % Covariance matrix from 
fitlm

%se_delta_method = sqrt(diag(Jacobian * cov_matrix * Jacobian'));

%disp('Standard Errors (Delta Method):');
%disp(se_delta_method);
num_obs = 3375;

% Step 4: Recover Point Estimates and Standard Errors using Delta Method
% Define the Jacobian matrix of the gravity model equation with respect to 
parameters

Jacobian = [ones(num_obs, 1), Rich_Data.log_GDP_, Rich_Data.log_distance_, 
Rich_Data.log_contiguity_, Rich_Data.log_Pi_,Rich_Data.log_Pj_];
```

```
% Calculate the point estimates using the Delta Method formula
point_estimates_delta_method = gravity_model.Coefficients.Estimate;


% Calculate the standard errors using the Delta Method formula
cov_matrix = gravity_model.CoefficientCovariance;


se_delta_method = sqrt(diag(Jacobian * cov_matrix * Jacobian'));
disp('Point Estimates (Delta Method):');
```

```
Point Estimates (Delta Method):
```

```
disp(point_estimates_delta_method);
```

```
   12.9726
   -0.9055
    0.0172
    0.1673
    0.1678
   -0.0496
```

```
disp('Standard Errors (Delta Method):');
```

```
Standard Errors (Delta Method):
```

```
disp(se_delta_method);
```

```
    0.4644
    0.3692
    0.3531
    0.3741
    0.3978
    0.3731
    0.3917
    0.3507
    0.3837
    0.3863
    0.4075
    0.3813
    0.3481
    0.3636
    0.3971
    0.4644
    0.3692
    0.3531
    0.3741
    0.3978
    0.3731
    ……………..
```

PART 4

```
% Import the data
IV_Data = readtable("/Users/richmondessieku/Downloads/IV_Data.xlsx", opts2,
"UseExcel", false);

% Clear temporary variables
clear opts2

% Display results
IV_Data
```

```
IV_Data = 65535×7 table
```

```
size(IV_Data)
```

```
ans = 1×2
       65535           7
```

```
% PS 4
Part 1
In light of the question, the export is identified as the dependent variable,
while the independent variable, protectionist policy, is notably missing from
the model. In this scenario, I have opted to employ "origin population" as an
instrumental variable in lieu of the missing protectionist policy, and I offer
the following rationale for this choice according to the requirement of the
question.
The choice of the IV must satisfy the orthogonality and rank tests to be
considered a valid instrument.
Orthogonality Test
The orthogonality test requires that the instrumental variable (origin
population in this case) is uncorrelated with the error term of the regression
model and is only correlated with the omitted variable (protectionist policy)
through its impact on the export.
Relevance
To start, we need to justify that "origin population" is relevant to the model.
I can argue that larger populations in the origin country are likely to lead to
increased exports because a larger domestic market can support more production
and export activities. This relationship is theoretically sound and widely
accepted in international trade theory.
```

```
 Exogeneity
 To satisfy the orthogonality condition, I assume that "origin population" is
not correlated with unobserved factors affecting exports other than the omitted
variable (protectionist policy). While this assumption may not always hold
perfectly in practice, I can argue that population size is a relatively stable
and exogenous variable over the short to medium term. It's unlikely that
countries would significantly alter their population sizes in response to
specific changes in protectionist policies. Therefore, "origin population" is a
plausible exogenous variable.
 Lack of Direct Effect
 It's essential to argue that "origin population" does not directly affect
exports between the two countries independently of protectionist policies. For
example, it should not be the case that a larger population automatically leads
to more exports without considering the influence of protectionist policies. If
such a direct effect exists, it could violate the orthogonality assumption.

 Rank Test
 The rank test checks whether the instrument has a higher correlation with the
endogenous variable (exports) than with the omitted variable (protectionist
policy). In this case, the rank test assesses whether "origin population" is a
better predictor of exports than it is of protectionist policies.
 Strength of Instrument
 To satisfy the rank test, I argue that "origin population" is a stronger
predictor of exports than protectionist policies. This argument can be made by
demonstrating that changes in "origin population" have a more substantial and
statistically significant effect on exports, as compared to their effect on
protectionist policies when included in a model.

% First stage regression
model_formula1 = 'log_distance_ ~ Log_pop_pwt_o_';
first_stage_model = fitlm(IV_Data, model_formula1);
IV_Data.predicted_DIST = predict(first_stage_model, IV_Data);
IV_Data
%n_obs = 65535;
%IV_Data.predicted_DIST = IV_Data.predicted_DIST + randn(n_obs, 1);
%IV_Data
% Display results
disp('First Stage Regression:');
```

```
First Stage Regression:
```

```
disp(first_stage_model);
```

```
Linear regression model:
    log_distance_ ~ 1 + Log_pop_pwt_o_

Estimated Coefficients:
                          Estimate        SE         tStat       pValue
```

```
                     ___________   ___________   _________   ________

    (Intercept)           3.9149     0.0049124      796.94          0
    Log_pop_pwt_o_   5.6769e-05    0.00060231    0.094253    0.92491


Number of observations: 65535, Error degrees of freedom: 65533
Root Mean Squared Error: 0.364
R-squared: 1.36e-07,  Adjusted R-Squared: -1.51e-05
F-statistic vs. constant model: 0.00888, p-value = 0.925
```

```
% Second stage regression (2SLS)
model_formula2 = 'log_export_ ~ - predicted_DIST + log_GDP_ + log_contiguity_ +
log_Pi_ + log_Pj_';
second_stage_model = fitlm(IV_Data, model_formula2);

% Display results
disp('Second Stage Regression (2SLS):');
```

```
Second Stage Regression (2SLS):
```

```
disp(second_stage_model);
```

```
Linear regression model:
    log_export_ ~ 1 + log_contiguity_ + log_Pi_ + log_Pj_ + log_GDP_

Estimated Coefficients:
                       Estimate        SE         tStat       pValue
                       _________   _________   _________   __________

    (Intercept)           10.026   0.0022485      4458.9            0
    log_contiguity_     -0.21084    0.012139     -17.369   2.0029e-67
    log_Pi_              -0.0409     0.01452     -2.8168    0.0048512
    log_Pj_            -0.085548    0.014558     -5.8763    4.216e-09
    log_GDP_            -0.47624   0.0012295     -387.34            0


Number of observations: 65535, Error degrees of freedom: 65530
Root Mean Squared Error: 0.217
R-squared: 0.697,  Adjusted R-Squared: 0.697
F-statistic vs. constant model: 3.78e+04, p-value = 0
```

```
% The Minimum Chi-Square Estimator (MCSE)

% Define the structural equation model
model_formula3 = 'log_export_~ log_GDP_ + log_distance_ + log_contiguity_ +
log_Pi_ + log_Pj_';
% Fit the model using the Minimum Chi-Square Estimator
options = statset('MaxIter', 100); % Increase max iterations if needed
fitResult = fit(IV_Data, model_formula3,  'mcest', 'Options', options);
```

```
Error using fit>iFit
X must be a matrix with one or two columns.

Error in fit (line 116)
[fitobj, goodness, output, convmsg] = iFit( xdatain, ydatain, fittypeobj, ...
% Display results
%disp(fitResult);
```