Exercise – Understanding the method of clustered bootstrap

1. Aim of the exercise

We are interested in conducting regression analysis to explain unemployment rates by macroeconomic factors including GDP, unemployment, capital mobility and trade volume. The dataset is part of the Zelig software package developed by the Institute for Quantitative Social Science at Harvard University and is available through Zelig's GitHub repository. It contains annual data from 1966 to 1990 for 14 countries, providing 34 years of observations per country. We expect regression errors within each country, which we treat as a cluster, to be correlated due to country-specific factors such as policies, geography, or institutions. This violates the independence assumption of OLS and leads to biased standard errors, often downward, resulting in overconfident inference. The clustered bootstrap addresses this by resampling entire clusters, preserving within-group dependence and yielding more reliable standard error estimates. While it reduces bias, some distortion may still remain. We also compare the clustered bootstrap estimator to two alternatives: the conventional estimator that assumes independent observations, and the analytical cluster-robust estimator based on the sandwich formula. This comparison highlights how well the clustered bootstrap accounts for within-cluster dependence and improves inference.

2. Theory

In regression analysis, reliable inference depends not only on accurate coefficient estimates but also on the validity of assumptions about the error structure. A central assumption in ordinary least squares (OLS) estimation is that the error terms are independently and i.i.d., meaning they are uncorrelated and exhibit constant variance across observations. This is known as the spherical error variance assumption, formally expressed as:

$$\mathrm{E}\left[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}'\right] = \sigma^2\mathbf{I}.$$

Under this assumption, the standard error estimator for the OLS coefficient estimates is derived by taking the square roots of the diagonal elements of the following variance-covariance matrix:

$$\widehat{\mathrm{Var}}\left[\hat{\boldsymbol{\beta}}\right] = \frac{1}{n-K}(\mathbf{X}'\mathbf{X})^{-1}\hat{\boldsymbol{\varepsilon}}'\hat{\boldsymbol{\varepsilon}}.$$

When the data exhibit clustering, serial correlation, or heteroskedasticity, the spherical error variance assumption is violated. Applying OLS under the incorrect assumption of independent errors leads to biased standard error estimates, typically downward, which results in overstated statistical significance and unreliable hypothesis tests. In the presence of clustering, the assumption of independent and homoskedastic errors no longer holds. Instead, the error covariance matrix must allow for correlation within clusters:

$$\mathrm{E}\left[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}'\right] = \sigma^2\boldsymbol{\Omega}$$

where $\boldsymbol{\Omega}$ is a block-diagonal matrix, with each diagonal block corresponding to a cluster and taking the form:

$$\begin{bmatrix} 1 & \rho & \dots & \rho \\ \rho & 1 & \dots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \dots & 1 \end{bmatrix}$$

where ρ represents the intra-cluster correlation coefficient, summarizing the average correlation between error terms within each cluster, and is assumed constant across clusters.

Greenwald (1983) analyzes how misspecification of the error structure, particularly due to intra-cluster dependence, biases the estimated variance of OLS coefficients. The bias arises from the deviation between the true error covariance matrix and the spherical assumption, captured by

$$\mathbf{I} - \mathbf{\Omega}.$$

This structural difference drives distortions in standard error estimates when intra-cluster correlation is ignored.

To illustrate the practical consequences of this deviation, Greenwald provides an approximate expression for the bias in the estimated variance of coefficient i:

$$b_{ii} \cong -\frac{\rho\sigma^2}{n^2} \sum_{c=1}^{C} \sum_{s=1}^{n_c} \sum_{\substack{t=1 \\ t \neq s}}^{n_c} \mathrm{Cov}(x_{si}, x_{ti}).$$

Here, n_c denotes the number of observations within cluster c, such that summing over all clusters yields the total sample size of n. $\mathrm{Cov}(x_{si}, x_{ti})$ represents the covariance between the values of regressor i for observations s and t in cluster c. The influence of error correlation enters through the scalar ρ, which scales the entire expression. This formula shows that when the values of a given regressor are correlated across observations within the same cluster and errors are positively correlated, the conventional standard error estimator, based on the assumption of independent errors, understates the true variance of the coefficient estimate. The bias becomes more pronounced when clusters are large or when intra-cluster correlation is strong. As a result, hypothesis tests relying on conventional standard errors may be misleading, overstating statistical significance and increasing the risk of false positives.

Beyond this scalar approximation, Greenwald also derives the exact matrix expression for the bias in the estimated variance-covariance matrix of the OLS coefficients:

$$\mathbf{b} = \left(\mathrm{E}\left[\hat{\sigma}^2\right] - \sigma^2\right)(\mathbf{X}'\mathbf{X})^{-1} + \sigma^2(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'(\mathbf{I} - \mathbf{\Omega})\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}.$$

This formula decomposes the bias into two components. The first component reflects the bias introduced by estimating the residual variance $\hat{\sigma}^2$ under the incorrect assumption that errors are i.i.d.. In clustered data, the assumption of independent errors fails because observations within the same cluster share correlated disturbances. When residuals are computed under the false assumption of independence, they ignore this intra-cluster correlation. As a result, the estimated residual variance $\hat{\sigma}^2$ captures less variability than actually exists in the data, leading to an understatement of the true variance in the coefficient estimates. This bias arises even before accounting for the misspecification of the error covariance structure $\mathbf{\Omega}$, compounding the distortion in standard error estimation.

The second component captures the bias resulting from misspecifying the error covariance structure, specifically, assuming a spherical form $\mathbf{I}$ when the true covariance matrix is $\mathbf{\Omega}$, which allows for intra-cluster correlation. Together, they quantify how the conventional variance estimator deviates from its true value when intra-cluster dependence is ignored. Unlike the scalar approximation b_{ii}, this matrix formulation reveals how bias affects the entire variance-covariance structure, including off-diagonal elements, making it essential for understanding and correcting inference in clustered data settings.

These theoretical insights motivate the use of robust alternatives to conventional standard error estimation. One widely adopted solution is the cluster-robust variance estimator, which adjusts for intra-cluster dependence by aggregating residual variation within each cluster. This estimator is computed from the square root of the diagonal entries of the following matrix:

$$\widehat{\mathrm{Var}}\left[\hat{\boldsymbol{\beta}}\right] = \left(\sum_{c=1}^{C} \mathbf{X}_c'\mathbf{X}_c\right)^{-1} \left(\sum_{c=1}^{C} \mathbf{X}_c'\hat{\boldsymbol{\varepsilon}}_c\hat{\boldsymbol{\varepsilon}}_c'\mathbf{X}_c\right) \left(\sum_{c=1}^{C} \mathbf{X}_c'\mathbf{X}_c\right)^{-1}$$

where C denotes the total number of clusters, $\mathbf{X}_c$ is the design matrix restricted to observations within cluster c, and $\hat{\boldsymbol{\varepsilon}}_c$ is the corresponding vector of residuals from the full-sample regression, also limited to cluster c. This formulation ensures that the estimator captures within-cluster dependence by incorporating each cluster's contribution to the overall variability. To improve accuracy in finite samples, particularly when the number of clusters is small, it is recommended to apply a degrees-of-freedom correction:

$$\frac{n-1}{n-K} \times \frac{C}{C-1}.$$

This correction accounts for the loss of degrees of freedom due to estimating K parameters and helps mitigate bias introduced by limited cluster variation.

An alternative approach to estimating cluster-robust standard errors is the cluster bootstrap estimator, which derives standard errors from the variability of coefficient estimates obtained through repeated resampling of entire clusters from the original sample.

3. Load data

Clear the memory. Import the content of the CSV file into a table array named `data`.

```matlab
%% 3. Load data

% 3.1. Clear the memory
clear;

% 3.2. Import CSV file
data = readtable('data.csv');
```

4. Obtain OLS statistics for conventional and sandwich estimators

Create the design matrix and obtain OLS statistics to be used to compute the conventional and sandwich estimators.

```matlab
%% 4. Obtain OLS statistics for conventional and sandwich estimators

% 4.1. Define response variable
y = table2array(data(:,4));

% 4.2. Create intercept term
x_0 = ones(size(data,1),1);

% 4.3. Create the design matrix
```

```
X = [x_0,table2array(data(:,[3,5,6]))];

% 4.4. Obtain OLS statistics
LSS = exercisefunctionlss(y,X);
```

5. Estimate standard errors using the conventional estimator

Estimate the standard errors of the coefficients using the conventional standard error formula, without accounting for the clustered structure of the data.

```
%% 5. Estimate standard errors using the conventional estimator

% Estimate standard errors using the conventional estimator
SEE_conventional = LSS.B_hat_SEE;
```

6. Set parameters for sandwich and bootstrap estimators

Specify the number of regression coefficients. Next, identify the structure of the data in terms of clusters. First, extract the grouping variable to determine which observations belong to which cluster. Next, identify the distinct groups to establish how many unique clusters are present in the dataset. Finally, count the total number of clusters, which is essential for applying our cluster-aware method, clustered bootstrap and cluster-robust inference.

```
%% 6. Set parameters for sandwich and bootstrap estimators

% 6.1. Define the number of coefficients
N_k = size(X,2);

% 6.2. Obtain cluster identifiers
cluster_identifiers = data.country;

% 6.3. Identify unique clusters
unique_clusters = unique(cluster_identifiers);

% 6.4. Count the number of clusters
N_clusters = numel(unique_clusters);
```

7. Preallocate matrices for sandwich estimator

Preallocate matrices needed to compute the analytical cluster-robust standard error estimator. It initializes empty matrices for the cluster-level components, specifically, the meat matrix and the design matrix cross-product, as well as their respective sums across all clusters. Each matrix is filled with zeros to prepare for accumulation during the loop over clusters.

```
%% 7. Preallocate matrices for sandwich estimator

% 7.1. Initialize cluster-level meat matrix (X_c'*u_c*u_c'*X_c)
M_c = zeros(N_k,N_k);

```

```
% 7.2. Initialize sum of meat matrices across clusters
M_c_sum = zeros(N_k,N_k);

% 7.3. Initialize cluster-level design matrix cross-product (X_c'*X_c)
X_cTX_c = zeros(N_k,N_k);

% 7.4. Initialize sum of design matrix cross-products across clusters
X_cTX_c_sum = zeros(N_k,N_k);
```

8. Create cluster-level design matrix for sandwich estimator

Here we loop over each cluster to compute the components needed for the analytical cluster-robust standard error estimator. For each cluster, it identifies the relevant observations, extracts the residuals and corresponding rows of the design matrix, and then calculates the cluster-level meat matrix and design matrix cross-product. These matrices are accumulated across clusters to form the full sandwich estimator, which accounts for within-cluster dependence in the variance calculation.

```
%% 8. Create cluster-level design matrix for sandwich estimator

% 8.1. Loop over each cluster to compute meat and bread components
for i = 1:N_clusters
    % Identify current cluster
    cluster_identifier = unique_clusters(i);

    % Find rows corresponding to the current cluster
    cluster_rows = strcmp(data.country,cluster_identifier);

    % Extract residuals for the current cluster
    e_hat_c = LSS.u_hat(cluster_rows);

    % Extract design matrix rows for the current cluster
    X_c = X(cluster_rows,:);

    % Compute cluster-level meat matrix
    M_c = X_c'*(e_hat_c*e_hat_c')*X_c;

    % Accumulate meat matrices
    M_c_sum = M_c_sum+M_c;

    % Compute cluster-level design matrix cross-product
    X_cTX_c = X_c'*X_c;

    % Accumulate design matrix cross-products
    X_cTX_c_sum = X_cTX_c_sum+X_cTX_c;
end
```

9. Estimate standard errors using the sandwich estimator

This section computes the analytical cluster-robust standard errors using the sandwich estimator. It begins by defining the total sample size and applying a finite-sample degrees-of-freedom correction to adjust for bias when the number of clusters is small. Then, it calculates the standard errors by taking the square root of the diagonal entries of the sandwich variance matrix, which combines the accumulated meat and bread components across all clusters. The result is a vector of cluster-robust standard errors for each regression coefficient.

```
%% 9. Estimate standard errors using the sandwich estimator

% 9.1. Define the sample size
N_obs = size(data,1);

% 9.2. Apply finite sample correction
df_correction = (N_obs-1)/(N_obs-N_k)*N_clusters/(N_clusters-1);

% 9.3. Compute cluster-robust standard errors using the sandwich formula
SEE_sandwich = df_correction*sqrt( ...
    diag(inv(X_cTX_c_sum)*M_c_sum*inv(X_cTX_c_sum)));
```

10. Draw bootstrap samples from the initial sample

Specify the number of bootstrap replications to be performed. Preallocate a vector to store the bootstrap coefficient estimates.

Next, we set up a loop for bootstrap resampling and coefficient estimation. In each iteration, we randomly draw a new set of cluster identifiers, such as country names, from the original pool. Some clusters may appear more than once, while others may be excluded. All observations linked to each selected cluster are included in the resampled dataset. This approach preserves the internal correlation structure within clusters, which is essential for valid inference when observations are related within groups. Unlike the basic bootstrap that resamples individual observations (see the exercise on the method of basic bootstrap), the clustered bootstrap resamples entire groups to maintain this dependence.

Define `resampled_data_by_cluster`, a cell array where each cell corresponds to one of the sampled clusters. Each cell will be populated with the full set of observations associated with that cluster. This structure preserves the grouping of data by cluster, which is essential for maintaining within-cluster dependencies during resampling and subsequent analysis.

The inner loop iterates over each sampled cluster identifier, that is country name, and extracts the corresponding subset of observations from the full dataset. For each iteration, it uses a logical comparison to identify all rows in the dataset where the cluster label matches the current sampled identifier. These matched rows, which include all variables for that cluster's observations, are then stored in a cell within the `resampled_data_by_cluster` array. As a result, the cell array holds a complete set of data for each sampled cluster, preserving the grouped structure necessary for valid bootstrap resampling in the presence of within-cluster dependence.

Next, consolidate the data from all sampled clusters into a single dataset. After storing the observations for each sampled cluster in separate cells of the `resampled_data_by_cluster`, the `vertcat` function is used to vertically concatenate these individual tables into one unified table called `resampled_data`. This combined dataset represents a full bootstrap replicate, preserving the grouped structure of the original data while allowing for clusters to appear multiple times or be omitted entirely, consistent with the logic of cluster-level resampling. The resulting table is then ready for regression analysis or other statistical procedures within the bootstrap loop.

In the remaining code, extract the response variable and predictors from the resampled data, construct the design matrix with an intercept, run an OLS regression, and store the estimated coefficients for the current bootstrap iteration.

```
%% 10. Draw bootstrap samples from the initial sample

% 10.1. Set the number of bootstrap samples
N_sim = 1000;

% 10.2. Preallocate vector to store coefficient estimates
B_hats_data_samples_boot = NaN(N_k,N_sim);

% 10.3 Sample clusters with replacement and estimate coefficients
for i = 1:N_sim
    % Randomly sample clusters with replacement
    sampled_cluster_identifiers = datasample(unique_clusters, ...
        N_clusters,'Replace',true);

    % Preallocate cell array for selected cluster data
    resampled_data_by_cluster = cell(numel( ...
        sampled_cluster_identifiers),1);

    % Extract rows corresponding to the sampled cluster
    for j = 1:numel(sampled_cluster_identifiers)
        resampled_data_by_cluster{j} = data(strcmp( ...
            data.country,sampled_cluster_identifiers(j)),:);
    end

    % Combine data from all selected clusters into one table
    resampled_data = vertcat(resampled_data_by_cluster{:});

    % Define response variable
    y = table2array(resampled_data(:,4));

    % Create intercept term
    x_0 = ones(size(resampled_data,1),1);

    % Create the design matrix
    X = [x_0,table2array(resampled_data(:,[3,5,6]))];

    % Compute OLS statisitics
    LSS = exercisefunctionlss(y,X);

    % Extract OLS coefficient estimates
    B_hats_data_samples_boot(:,i) = LSS.B_hat;
end
```

11. Estimate cluster-robust standard errors using bootstrap resampling

This line calculates the standard deviation of each regression coefficient across all bootstrap replications. The matrix `B_hats_data_samples_boot` contains estimated coefficients from every bootstrap iteration, with each row representing a specific coefficient and each column representing one bootstrap sample. The function `std(...,0,2)` computes the standard deviation along each row-meaning, for each coefficient, it measures how much that estimate varies across the bootstrap samples. The resulting vector contains the bootstrap estimator of the cluster-robust standard errors: one value per coefficient, reflecting its sampling variability under cluster resampling.

```
%% 11. Estimate standard errors using bootstrap resampling

% Std. dev. of the bootstrap sampling distribution: Bootstrap estimates
SEE_bootstrap = std(B_hats_data_samples_boot ,0 ,2);
```

12. Evaluating standard error estimators

As expected, the conventional standard error estimator substantially underestimates the standard errors. In contrast, the cluster-robust and bootstrap estimators are closely aligned, indicating that the bootstrap approach offers a reliable alternative for capturing cluster-level dependence in standard error estimation.

13. Final notes

This file is prepared and copyrighted by Quinten Salomons, Jesper Eizenga and Tunga Kantarcı. This file and the accompanying MATLAB file are available on GitHub and can be accessed using this link. This file has made use of Greenwald, B., 1983. A general analysis of bias in the estimated standard errors of least squares coefficients. Journal of Econometrics 22 (3), 323-338.