# Exercise – Understanding the method of wild bootstrap

## 1. Aim of the exercise

The aim of this exercise is to understand how to accurately estimate the standard errors of regression coefficients in the presence of heteroskedastic error variance. Traditional bootstrap methods, such as the paired bootstrap, may yield misleading inference under heteroskedasticity. To address this, we use the wild bootstrap method, which improves robustness by modifying residuals with random multipliers. We then compare its performance to that of direct sampling from the population.

## 2. Theory

Consider the linear regression model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where $\mathbf{y}$ is an $n \times 1$ vector of outcome, and $\mathbf{X}$ is an $n \times K$ matrix of regressors. The parameter vector $\boldsymbol{\beta}$ is $K \times 1$ and unknown. Assume the zero conditional mean assumption holds:

$$\mathrm{E}\left[\boldsymbol{\varepsilon} \mid \mathbf{X}\right] = 0,$$

but allow for heteroskedastic errors such that

$$\mathrm{E}\left[\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}' \mid \mathbf{X}\right] = \boldsymbol{\Omega},$$

where $\boldsymbol{\Omega} = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2)$. The ordinary least squares (OLS) estimator of $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}.$$

To account for heteroskedasticity, the variance-covariance matrix of $\hat{\boldsymbol{\beta}}$ can be estimated using the heteroskedasticity-consistent estimator:

$$\widehat{\mathrm{Var}}\left[\hat{\boldsymbol{\beta}}\right] = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\hat{\boldsymbol{\Omega}}\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1},$$

where $\hat{\boldsymbol{\Omega}} = \mathrm{diag}(\hat{u}_1^2, \hat{u}_2^2, \ldots, \hat{u}_n^2)$, and $\hat{u}_i$ are the OLS residuals.

Although the heteroskedasticity-consistent estimator is valid when the sample size is large, it may suffer from bias in small samples, particularly in the presence of high-leverage observations. These influential data points can disproportionately affect the estimated covariance matrix, leading to inaccurate standard errors and misleading inference. In such cases, the wild bootstrap provides a higher-order accurate approximation to the sampling distribution of the coefficient estimates. This means it not only approximates the mean and variance of the estimator, but also captures higher moments, such as skewness and kurtosis, that are important for constructing more reliable confidence intervals and hypothesis tests in small samples.

Let $(y_i,\ X_i)$ for $i = 1, \ldots, n$ be an i.i.d. sample from the described linear model. Estimate $\hat{\beta}$ by OLS and compute the residuals

$$\hat{u}_i = y_i - X_i\hat{\beta}.$$

Moreover, compute the leverage, $h_i$, for each observation as

$$h_i = X_i(X'X)^{-1}X_i'.$$

Using the residuals and the leverages, construct, in each bootstrap sampling,

$$y_i^b = X_i\hat{\beta} + \frac{\hat{u}_i}{\sqrt{1-h_i}}v_i^b,$$

where $v_i$ is a random variable with $\mathrm{E}[v_i] = 0$ and $\mathrm{Var}[v_i] = 1$.

The literature suggests many choices for the distribution of $v_i$, such as the Rademacher distribution where $v_i \in \{-1, +1\}$ with equal probability $\frac{1}{2}$. A refinement improving skewness correction is Mammen's two point law:

$$v_i = \begin{cases} \frac{-(\sqrt{5}-1)}{2} & \text{with probability } \frac{\sqrt{5}+1}{2\sqrt{5}} \\ \frac{\sqrt{5}+1}{2} & \text{with probability } \frac{\sqrt{5}-1}{2\sqrt{5}} \end{cases}$$

Another choice is simply the standard normal distribution which often offers good finite-sample performance and analytical convenience.

Once the pseudo outcomes $y_i^b$ are constructed for $b = 1, \ldots, B$, for each bootstrap sample, we regress the vector $y^b$ on $X$ to obtain a bootstrap coefficient estimate $\hat{\beta}^b$. Collecting these estimates yields an empirical approximation to the sampling distribution of $\hat{\beta}$.

By applying leverage-scaled reweighing of the residuals using $\frac{\hat{u}_i}{\sqrt{1-h_i}}$, the wild bootstrap preserves the original variance structure and mitigates distortion from influential data points (i.e., high-leverage observations). This adjustment improves the precision of bootstrap estimates in small samples, where traditional variance formulas can be inaccurate or misleading. If we were to simply scale residuals without randomization, each bootstrap sample would replicate the same residual pattern, yielding only one possible resampled version of the regression model. To introduce variability while maintaining heteroskedasticity, we incorporate random variables $v_i^b$ that modify the residuals. This enables the generation of multiple bootstrap replications, each simulating a different plausible outcome.

To justify the use of the wild bootstrap, we use the same notation as in the exercise on basic bootstrap. Note, however, that an observation here is the pair $(y_i,\ X_i)$, and the statistic of interest is $\hat{\beta}$. See the basic bootstrap exercise for more details.

Each bootstrap coefficient estimate $\hat{\beta}^b$ is an independent draw from the distribution $G_n(\cdot, F_n)$, which describes the behavior of the estimator given the observed data. As the number of bootstrap replications $B \to \infty$, the empirical CDF $\hat{G}_n(\tau, F_n)$, built from these draws, converges in probability to the true bootstrap distribution $G_n(\tau, F_n)$:

$$\hat{G}_n(\tau, F_n) \xrightarrow{p} G_n(\tau, F_n).$$

This is guaranteed by the law of large numbers applied conditionally on the data. Furthermore, the pseudo-errors $\frac{\hat{u}_i}{\sqrt{1-h_i}}v_i^b$ are constructed to be conditionally independent, have mean zero, and match the variance structure of the original residuals. As the sample size $n \to \infty$, the conditional central limit theorem (specifically, the Lindeberg-Feller version) ensures that the bootstrap distribution $G_n(\tau, F_n)$ converges to the true sampling distribution $G_n(\tau, F)$, the one we would obtain if we knew the full population:

$$G_n(\tau, F_n) \xrightarrow{p} G_n(\tau, F).$$

Together, these results show that the wild bootstrap not only approximates the variability of the estimator for a fixed dataset, but also faithfully recovers its true sampling behavior as the sample size grows.

3. Set values for the parameters of the simulation

Clear all variables from memory, and set the total number of simulation or bootstrap runs.

```
%% 3. Set values for the parameters of the simulation

% 3.1. Clear the memory
clear;

% 3.3. Set the number of simulations as the number of bootstrap samples
N_sim = 5000;
```

4. Generate population data

Set the population size. Generate random predictor values by independently sampling from a uniform distribution between -1 and 1. The error term is heteroskedastic, with its variance following an exponential pattern controlled by a parameter set to 1.5. This parameter determines the degree of heteroskedasticity: higher values intensify the variance pattern, allowing us to assess how well the paired bootstrap handles non-constant error variance. In this setup, the error variance scales multiplicatively with the predictor value. The exponential form ensures the variance remains positive, reflecting the log-linear variance structures commonly used in applied research.

Next, set the true value of the regression coefficient to 0.5. Using this value and the generated predictor variable, construct the outcome variable. Combine the outcome and predictor into an `N` × `2` matrix, where each row represents a single observation. That is, each pair $(y_i, X_i)$ forms one data point in the population. This is exactly what the paired bootstrap resamples in order to preserve the joint distribution of $(y_i, X_i)$ – see the exercise on paired bootstrap for further details.

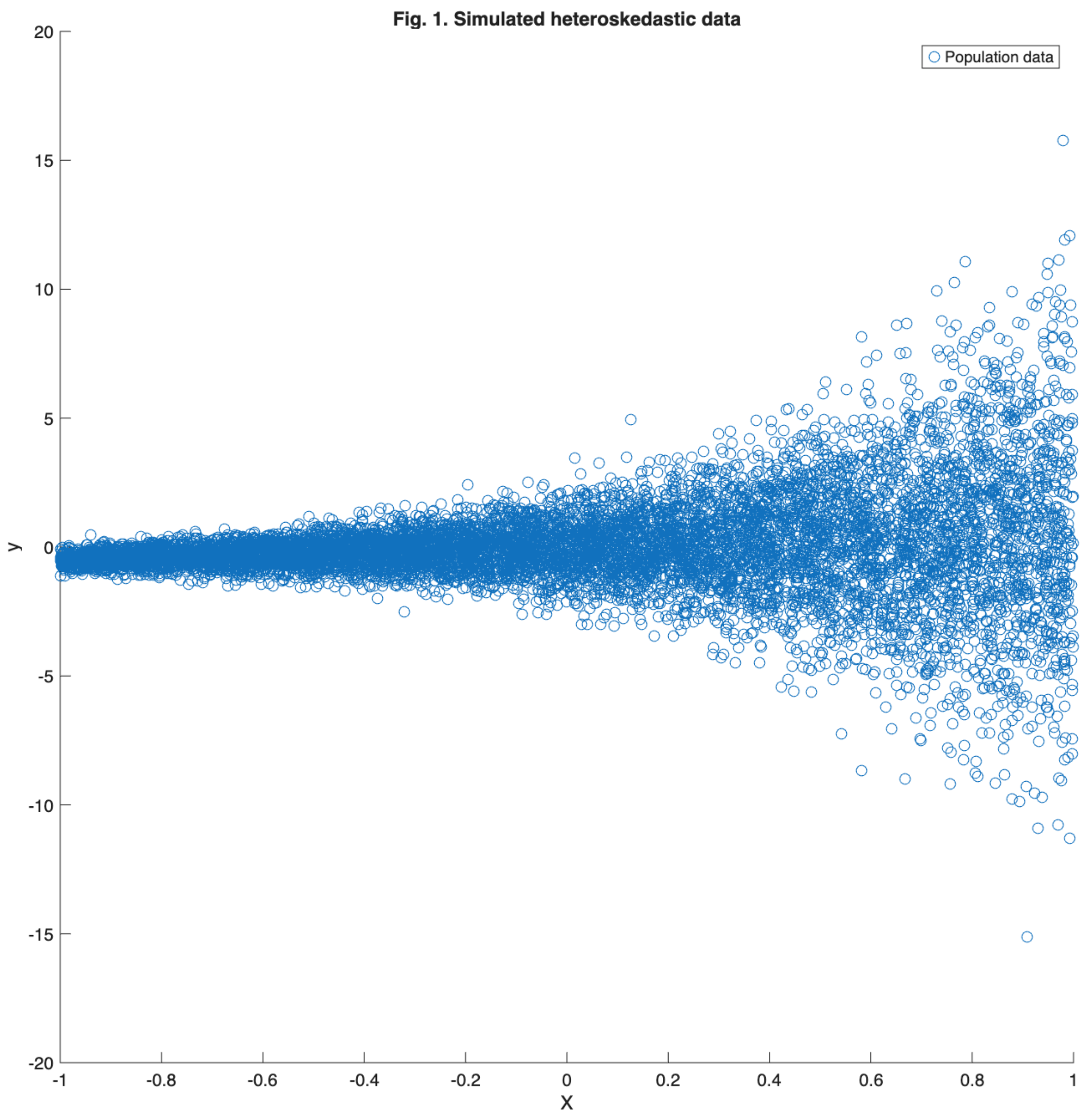
```
%% 4. Generate population data

% 4.1. Set the population size
N_obs_pop = 10000;

% 4.2. Generate data for the independent variable
X_pop = random('Uniform',-1,1,[N_obs_pop,1]);

% 4.3. Heteroskedasticity parameter
Gamma = 1.5;

% 4.4. Generate error
u_pop = random('Normal',0,exp(X_pop*Gamma),[N_obs_pop,1]);

% 4.5. Set true beta of the model
B_true = 0.5;

```

```
% 4.6. Generate y
y_pop = X_pop*B_true+u_pop;

% 4.7. Generate (paired) population data
data_pop = [y_pop X_pop];
```

5. Plot the scatter diagram and the OLS fitted line

Figure 1 plots the generated data. It illustrates the presence of heteroskedasticity: the dispersion of the outcome variable increases with the predictor. That is, the variance is not constant across the values of the predictor.

```
%% 5. Plot the heteroskedastic true data
figure
hold on
scatter(X_pop,y_pop);
title('Fig. 1. Simulated heteroskedastic data');
xlabel('X');
ylabel('y');
legend('Population data');
hold off
```



6. Draw samples from the population

We conduct a Monte Carlo experiment by repeatedly drawing i.i.d. random samples from the population dataset using MATLAB's `datasample` function, in order to examine the sampling distribution of the regression coefficient estimate. For each of `N_sim` samples, the code selects `N_obs_sample` paired observations $(y_i, X_i)$ without replacement and stores them in the three-dimensional matrix `data_samples_pop` of size $100 \times 2 \times$ `N_sim`. In this structure, rows hold individual observations, columns hold the two variables, and the third dimension indexes the simulation run, so each slice `(:,:,i)` contains one complete sample. The external OLS estimation function is then applied to each sample, and the resulting coefficient estimate from that run is saved in the vector `B_hats_data_samples_pop` for later analysis.

```
%% 6. Draw samples from the population

% 6.1. Set the sample size
N_obs_sample = 100;

```

```
% 6.2. Preallocate matrix to save (paired) samples
data_samples_pop = NaN(N_obs_sample,2,N_sim);

% 6.3. Preallocate vector to store coeficient estimatess
B_hats_data_samples_pop = NaN(N_sim,1);

% 6.4. Monte Carlo sampling
for i = 1:N_sim
    data_samples_pop(:,:,i) = datasample(data_pop,N_obs_sample, ...
        'Replace',false); % Save samples in 3rd dimension
    y = data_samples_pop(:,1,i);
    X = data_samples_pop(:,2,i);
    LSS = exercisefunctionlss(y,X);
    B_hats_data_samples_pop(i) = LSS.B_hat(1,1);
end
```

7. Pick an "initial" sample

Pick a sample among the samples already randomly drawn from the population and use it as the initial sample for bootstrap sampling. Note that the data comprise paired observations of the outcome and predictor, preserving their joint empirical distribution as in the exercise on paired bootstrap.

```
%% 7. Pick an "initial" sample

% 7.1. Randomly pick one sample index
sample_index = randi(N_sim);

% 7.2. Pick one sample from the previously drawn samples
data_sample = data_samples_pop(:,:,sample_index);
```

8. Draw (bootstrap) samples from the initial sample

Create vectors to hold the sampled observations for the outcome variable $y$ and the predictor variable $X$. Note that in the wild bootstrap, the design matrix X is treated as fixed and remains unchanged throughout the resampling process. Estimate the OLS coefficient $\hat{\beta}$ using the initial sample. Calculate the residuals $\hat{u}_i$ for each observation based on this estimate. Compute the leverage values $h_i$ for each observation, which will be used to adjust the residuals in the wild bootstrap procedure. Preallocate a matrix to store the bootstrap coefficient estimates. To approximate the sampling distribution of $\hat{\beta}$, we repeatedly resample from the initial sample (not the full population), using the same number of iterations `N_sim` as in the Monte Carlo simulation above. This ensures comparability between the two methods. For each bootstrap iteration, draw independent random multipliers $v_i$ from a standard normal distribution. Then, construct the pseudo-outcome variable $y_i^*$ using the formula:

$$y_i^* = X_i\hat{\beta} + \frac{\hat{u}_i}{\sqrt{1-h_i}} \cdot v_i.$$

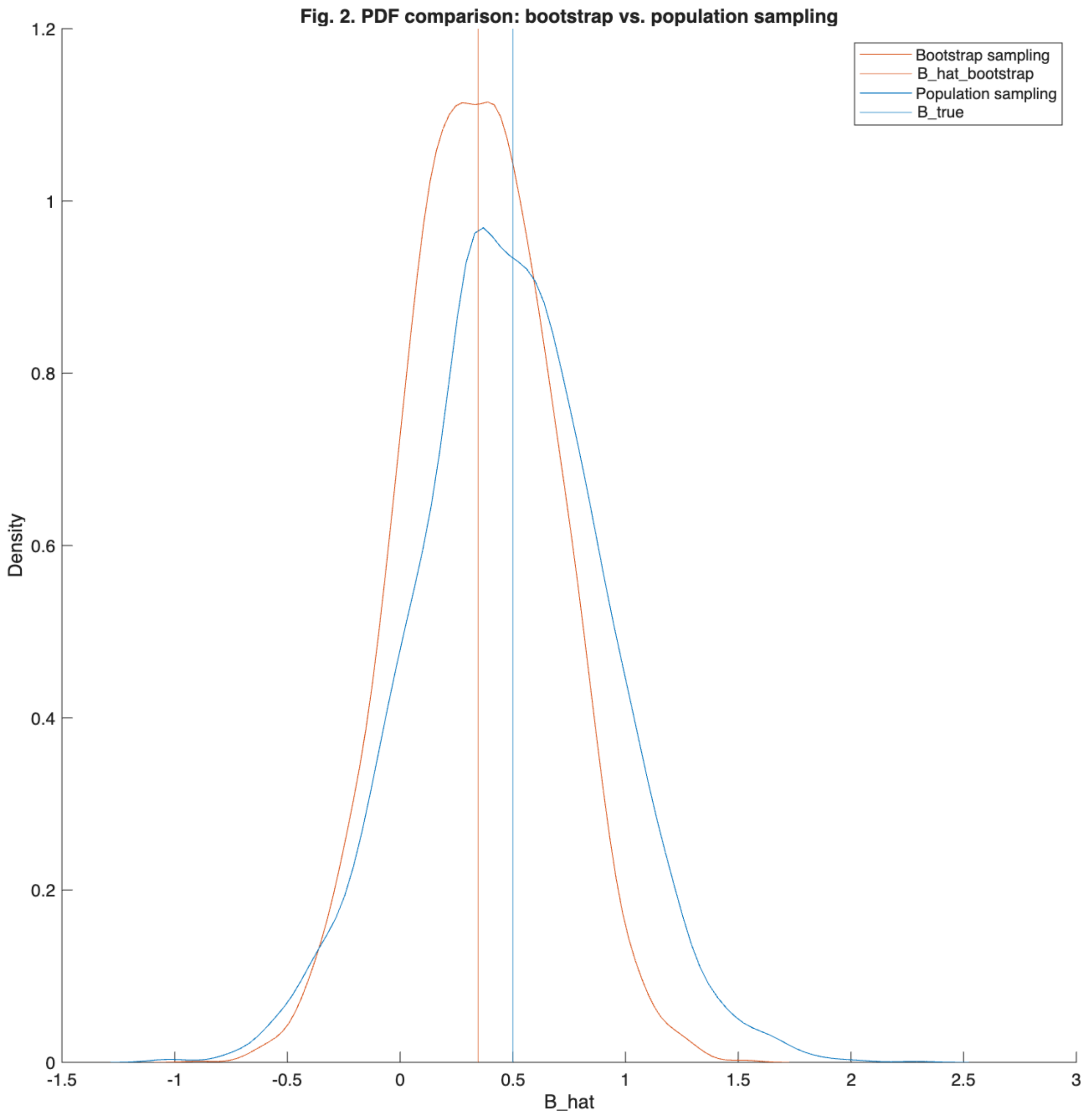
Using this pseudo-outcome, estimate the coefficient for each bootstrap sample.

```
%% 8. Draw (bootstrap) samples from the initial sample

% 8.1. Extract fixed design matrix and outcome variable
y = data_sample(:,1);
X = data_sample(:,2);

% 8.2. Estimate beta
LSS = exercisefunctionlss(y,X);
B_hat = LSS.B_hat(1,1);

% 8.3. Compute the residuals
u_hat = LSS.u_hat;

% 8.4. Compute leverage values used in HC2 correction
h = diag(X/(X'*X)*X');

% 8.5. Preallocate vector to store coefficient estimates
B_hats_data_samples_boot = NaN(N_sim,1);

% 8.6. Resample from initial sample and estimate the coefficient
for i = 1:N_sim
    v = random('Normal',0,1,[N_obs_sample,1]);
    % v = 2*(random('Uniform',-1,1,[N_obs_sample,1]) > 0.5)-1;
    y_boot = X*B_hat+(u_hat./sqrt(1-h)).*v;
    LSS = exercisefunctionlss(y_boot,X);
    B_hats_data_samples_boot(i) = LSS.B_hat(1,1);
end
```

9. Plot the PDFs

In Figure 2, we compare the sampling distribution of the coefficient estimate obtained using wild bootstrap sampling from the initial sample dataset with that obtained through direct sampling from the population. Each distribution is smoothed into a probability density function (PDF) using the `ksdensity` function. We also mark the true coefficient value and the mean of the bootstrap sampling distribution with vertical lines. The two PDFs exhibit very similar shapes, indicating that the wild bootstrap effectively replicates the form of the true sampling distribution. A horizontal offset between them arises because the bootstrap is conditioned on the finite initial sample: if the average coefficient estimate in that sample deviates from the population's true coefficient, the bootstrap distribution will inherit that bias. This illustrates that while the bootstrap accurately captures sampling variability, even under heteroskedasticity, it also preserves any bias present in the original sample.

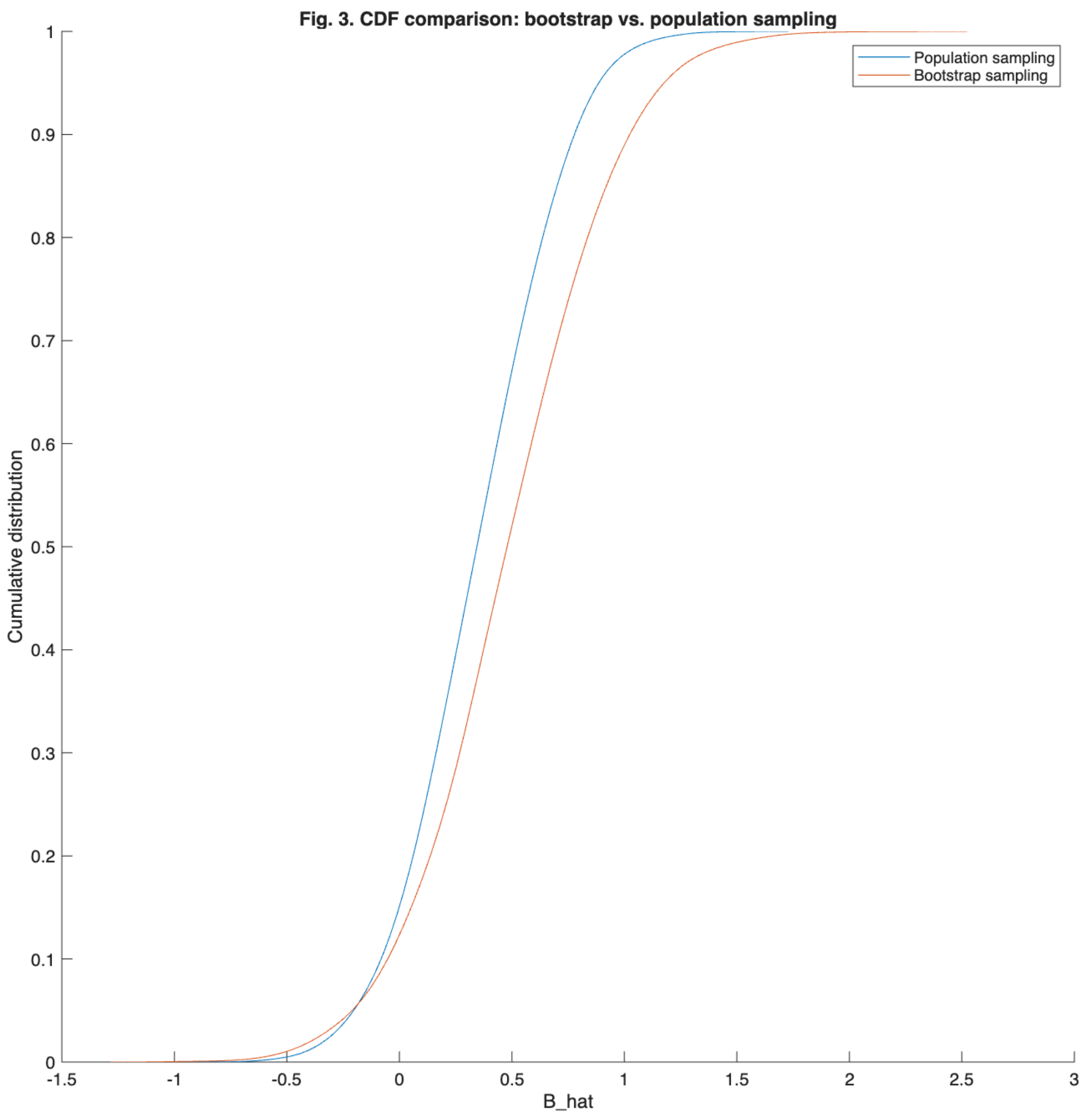
```
%% 9. Plot estimated PDFs
figure
hold on
[f_boot,x_boot] = ksdensity(B_hats_data_samples_boot,'function','pdf');
plot(x_boot,f_boot,'Color',[0.8500,0.3250,0.0980], ...
    'DisplayName','Bootstrap sampling');
xline(mean(B_hats_data_samples_boot),'Color',[0.8500,0.3250,0.0980], ...
    'DisplayName','B\_hat\_bootstrap');
[f_pop,x_pop] = ksdensity(B_hats_data_samples_pop,'function','pdf');
plot(x_pop,f_pop,'Color',[0,0.4470,0.7410], ...
    'DisplayName','Population sampling');
xline(B_true,'Color',[0,0.4470,0.7410], ...
    'DisplayName','B\_true');
ylabel('Density');
xlabel('B\_hat');
legend('show')
title(['Fig. 2. PDF comparison: bootstrap vs. ' ...
    'population sampling']);
hold off
```



10. Plot the CDFs

Here, we produce empirical CDFs for the bootstrap and population-based coefficient samples using the `ksdensity` function. The resulting CDF curves are plotted together to complement the PDF comparison.

```
%% 10. Plot estimated CDFs
figure
hold on
[f_boot,x_boot] = ksdensity(B_hats_data_samples_boot,'function','cdf');
plot(x_boot,f_boot,'Color',[0,0.4470,0.7410], ...
    'DisplayName','Population sampling');
[f_pop,x_pop] = ksdensity(B_hats_data_samples_pop,'function','cdf');
plot(x_pop,f_pop,'Color',[0.8500,0.3250,0.0980], ...
    'DisplayName','Bootstrap sampling');
ylabel('Cumulative distribution');
xlabel('B\_hat');
legend('show');
title(['Fig. 3. CDF comparison: bootstrap vs. ' ...
    'population sampling']);
hold off
```



## 11. Resampling strategies for heteroskedasticity: The case for fixed regressors

Another alternative to the heteroskedasticity-consistent estimator is the paired bootstrap estimator, which, like the wild bootstrap, produces more accurate distributional approximations by resampling the data. This resampling process allows it to reproduce heteroskedasticity more effectively than the heteroskedasticity-consistent estimator in small samples. However, the paired and wild bootstrap methods differ in how they treat the regressors. The paired bootstrap resamples entire $(y_i, X_i)$ pairs, implicitly assuming that both the outcome and the covariates are i.i.d. draws from a joint distribution. This approach is appropriate when $X$ is random, but problematic when $X$ is fixed by design, such as with calendar-based dummy variables (e.g., year or month indicators), experimental treatment assignments, or structurally influential regressors whose leverage arises from deliberate design choices rather than sampling variation. The wild bootstrap, by contrast, keeps $X$ fixed and re-weights the residuals, preserving the original design matrix and its leverage structure while still accommodating heteroskedasticity. Therefore, when the integrity of a fixed design is essential, the wild bootstrap is the preferred method.

## 12. Final notes

This file is prepared and copyrighted by Axel Zoons, Quinten Solomons, 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 two reference. First is Davidson, R., Flachaire, E., 2008. The Wild bootstrap, tamed at last. Journal of Econometrics, 146 (1), 162-169. The second is Davidson, R., MacKinnon, J. G., 2010. Wild bootstrap tests for IV regression. Journal of Business & Economic Statistics, 28 (1), 128-144.