Exercise – Understanding the method of block bootstrap

1. Aim of the exercise

The aim of this exercise is to understand how to correctly estimate the sampling distribution of a statistic (e.g., the sample mean) when data are serially correlated, such as in time-series settings. Standard bootstrap methods assume independent observations and can lead to biased inference. We use the block bootstrap method to account for dependence across observations and compare its performance to sampling directly from the population.

2. Theory

Let $X_1, \ldots, X_n$ be an ordered sample drawn from a stationary stochastic process, and suppose that $\mathrm{Cov}(X_i, X_{i+h}) \neq 0$ for some $h \neq 0$, indicating temporal dependence. Stationarity implies that the distribution of each individual observation X_t remains constant over time. That is, for all t, the probability that $X_t \leq x$ is given by the same cumulative distribution function (CDF), denoted $F(x) = P(X_t \leq x)$. This is referred to as the marginal distribution, as it describes the behavior of a single observation without reference to its dependence on other values in the sequence. Importantly, temporal dependence does not imply that dependence fades over time; rather, in a stationary process, the dependence structure is stable and time-invariant. Let $\mathcal{P}$ denote the joint distribution of the entire sequence $\{X_i\}$, which governs both the marginal distribution F and the temporal dependence across observations.

Because our goal is to perform inference for a stationary process exhibiting serial dependence, the standard bootstrap, which assumes independent observations, is not appropriate. A more suitable alternative is the block bootstrap. While several variants of the block bootstrap have been developed to accommodate dependence structures, we focus on the non-overlapping block bootstrap for its simplicity and historical prominence. This method preserves local dependence by resampling blocks of consecutive observations, making it well-suited for time series data.

To implement the non-overlapping block bootstrap, begin by choosing a block length $l \in \{1, \ldots, n\}$, where n is the total number of observations in the original sample. Then divide the sample into $k = \lfloor \frac{n}{l} \rfloor$ consecutive, non-overlapping blocks of length l:

$$Q_1 = (X_1, \ldots, X_l), \quad Q_2 = (X_{l+1}, \ldots, X_{2l}), \quad \ldots, \quad Q_k = (X_{(k-1)l+1}, \ldots, X_{kl}).$$

If n is not an exact multiple of l, the final $n - kl$ observations are discarded. The resulting effective sample size is denoted by $m = k \times l$.

In the block bootstrap, we first divide the data into blocks of consecutive observations. Instead of sampling individual data points, we treat each block as a single unit and randomly select blocks with replacement. This helps preserve the time-based dependence within each block. After resampling, we unpack the selected blocks back into individual observations to form a new pseudo-sample of size m. We then compute the statistic of interest, T_m^b, on this resampled data. Repeating this process B times gives an empirical approximation of the sampling distribution under serial dependence.

Let $G_{n,m}(\tau, \mathcal{P})$ denote the CDF of the statistic T_m, computed from a dependent sample of size m drawn from a stationary process governed by the joint distribution $\mathcal{P}$:

$$G_{n,m}(\tau, \mathcal{P}) = \mathbb{P}(T_m \leq \tau).$$

Since $\mathcal{P}$ is unknown, we approximate it using the empirical marginal distribution F_n, combined with block-based resampling to preserve temporal dependence. This yields an estimate of $G_{n,m}(\tau, F_n)$ via the empirical block bootstrap distribution:

$$\hat{G}_{n,m}(\tau, F_n) = \frac{1}{B} \sum_{b=1}^{B} \mathbb{I}\left(T_m^{(b)} \leq \tau\right),$$

where $\mathbb{I}(\cdot)$ denotes the indicator function, and $T_m^{(b)}$ is the statistic computed from the b-th block bootstrap replicate of length m.

The accuracy of the non-overlapping block bootstrap critically depends on the choice of block length l. If l is too large, the number of available blocks decreases, resulting in bootstrap replicates constructed from fewer distinct blocks and inflating the Monte Carlo variance. Conversely, if l is too small, correlated observations may be split across blocks, causing the resampled data to resemble independent draws and potentially leading to an underestimation of variance. Selecting l therefore requires balancing the bias introduced by neglecting long-range dependence against the variance introduced by excessive smoothing of the bootstrap distribution. A common theoretical approach to choosing l is to minimize the mean squared error (MSE) of the bootstrap estimator, which reflects the trade-off between bias and variance. However, this method is rarely applied directly in practice, as the optimal block length depends on unknown features of the data-generating process. As a practical alternative, Hall et al. (1995) propose setting $l = \lceil n^{1/k} \rceil$, where n is the effective sample size and k is a tuning constant determined by the inferential objective: $k = 3$ for bias or variance estimation, $k = 4$ for one-sided distribution function estimation, and $k = 5$ for two-sided distribution function estimation.

Under standard regularity conditions, such as stationarity, weak dependence (e.g., mixing), and appropriate block length growth, the block bootstrap is asymptotically consistent. This consistency is established in two steps. First, for a fixed sample size n and empirical distribution F_n, the empirical bootstrap distribution converges in probability to the true distribution under F_n as the number of bootstrap replications increases:

$$\hat{G}_{n,m}(\tau, F_n) \xrightarrow{p} G_{n,m}(\tau, F_n).$$

Second, the distribution under the empirical measure F_n converges in probability to the distribution under the true data-generating process $\mathcal{P}$ as the sample size n increases:

$$G_{n,m}(\tau, F_n) \xrightarrow{p} G_{n,m}(\tau, \mathcal{P}).$$

Provided the block length satisfies $l_n \to \infty$ and $l_n/n \to 0$, combining these two results yields the overall convergence:

$$\hat{G}_{n,m}(\tau, F_n) \xrightarrow{p} G_{n,m}(\tau, \mathcal{P}).$$

This justifies the use of the block bootstrap to approximate the sampling distribution of T_m in settings with serial dependence.

3. Set values for the parameters of the simulation

Clear all variables from memory and define the total number of simulations or samples to be generated.

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

% 3.1. Clear the memory
clear;

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

4. Generate population data

Set the population size. Our aim is to generate an autoregressive process of order one, AR(1), defined by $X_t = \phi X_{t-1} + \epsilon_t$, where ϵ_t represents white-noise errors. This construction yields a stationary time series with serial dependence. To simulate the process, we generate i.i.d. shocks ϵ_t from a standard normal distribution, which serve as the driving noise. The persistence parameter ϕ controls the strength of temporal dependence, with higher values indicating stronger autocorrelation between successive observations.

```
%% 4. Generate population data

% 4.1. Set the population size
N_obs_pop = 100000;

% 4.2. Set AR(1) persistence parameter
phi = 0.7;

% 4.3. Generate white noise innovations
varepsilon = random('Normal',0,1,[N_obs_pop,1]);

% 4.4. Simulate AR(1) time-series data
data_pop = filter(1,1-phi,varepsilon);
```

5. Plot AR(1) time series

Figure 1 presents the first 100 observations of the simulated AR(1) time series to illustrate the temporal structure and autocorrelation inherent in the population data. The series displays clear short-range dependence, characterized by local persistence and clusters of elevated or diminished values, an expected outcome given the autoregressive parameter $\phi = 0.7$. This pattern exemplifies the type of dependence for which the block bootstrap method is particularly well-suited.

```
%% 5. Plot AR(1) time series
figure
plot(1:100,data_pop(1:100));
xlabel('Time (t)');
ylabel('X_t');
title('Fig. 1. First 100 points of the AR(1) series');
legend('AR(1) process');
```

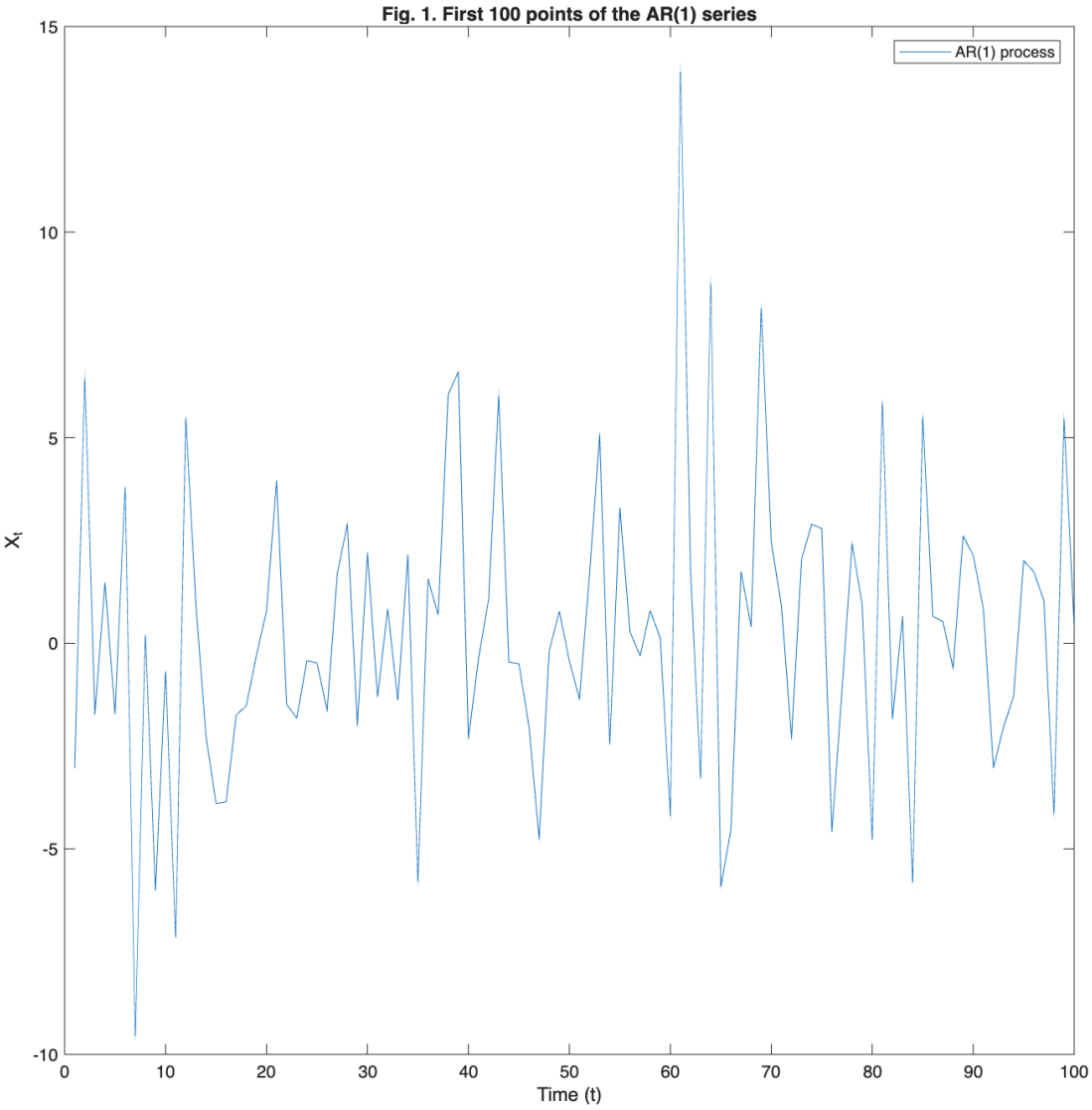


Fig. 1. First 100 points of the AR(1) series
AR(1) process
X_t
Time (t)
15
10
5
0
-5
-10
0
10
20
30
40
50
60
70
80
90
100

6. Draw samples from the population

In this bootstrap exercise, we focus on the sample mean as our statistic of interest and examine its sampling distribution under serial dependence. To evaluate how well the non-overlapping block bootstrap captures the uncertainty around the mean, we need a benchmark distribution for comparison. Since we have access to the full simulated AR(1) population, we can approximate the true sampling distribution by repeatedly drawing samples of size n directly from the population and computing their means.

Each sample is drawn as a contiguous block, a sequence of n consecutive observations, preserving the temporal structure and autocorrelation of the original time series. In each iteration, a random starting index is selected, and a block of n adjacent values is extracted. The sample mean of each block is stored in the vector `means_data_samples_pop`, forming an empirical reference distribution. This benchmark allows us to assess how closely the bootstrap-generated distribution replicates the true sampling variability of the mean.

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

% 6.1. Set the sample size
N_obs_sample = 5000;

% 6.2. Preallocate matrix to store samples
data_samples_pop = NaN(N_obs_sample,N_sim);

% 6.3. Preallocate vector to store sample means
means_data_samples_pop = NaN(N_sim,1);

% 6.4. Draw samples from the population and compute the sample mean each time
for i = 1:N_sim
    start_idx = randi(N_obs_pop-N_obs_sample+1);
    data_samples_pop(:,i) = data_pop(start_idx:start_idx+N_obs_sample-1);
    means_data_samples_pop(i) = mean(data_samples_pop(:,i));
end
```

7. Pick an "initial" sample

To initiate the bootstrap procedure, we begin by randomly selecting one of the previously drawn contiguous samples from the population. This sample, of length n, represents the kind of empirical data typically available in real-world applications. We then compute the optimal block length l using the rule-of-thumb $l = \lceil n^{1/5} \rceil$, as proposed by Hall et al. (1995) for estimating two-sided distribution functions. Based on this block length, we calculate the number of full blocks $k = \lfloor n/l \rfloor$ that can fit within the sample. To ensure clean partitioning, we truncate the sample to its first kl observations, discarding any leftover points that would prevent reshaping into equal-sized blocks. Finally, the trimmed sample is reshaped into k sequential, non-overlapping blocks of size l, which serve as the input units for the block bootstrap procedure.

```
%% 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);

% 7.3. Compute optimal block length
block_length = ceil(N_obs_sample^(1/5)); % Rule-of-thumb for block bootstrap

% 7.4. Compute how many full blocks can be formed
N_blocks = floor(N_obs_sample/block_length); % Maximum number of full blocks

% 7.5. Trim sample to fit an exact number of full blocks
trimmed_sample = data_sample(1:N_blocks*block_length); % Ensures reshape works c

% 7.6. Reshape trimmed sample into non-overlapping blocks
blocks = reshape(trimmed_sample,block_length,N_blocks)';
```

8. Draw (bootsrap) samples from the initial sample

We begin by preallocating a vector to store the sample means generated by the bootstrap procedure. For each of the `N_sim` replications, we draw k blocks with replacement from the original sample, which has already been partitioned into k non-overlapping blocks of size l. These sampled blocks are concatenated into a single bootstrap sample by flattening them into a one-dimensional vector. The sample mean of this bootstrap sample is then computed and stored. Repeating this process across all replications yields an empirical bootstrap distribution of the sample mean, which can be compared to the benchmark distribution derived from the population.

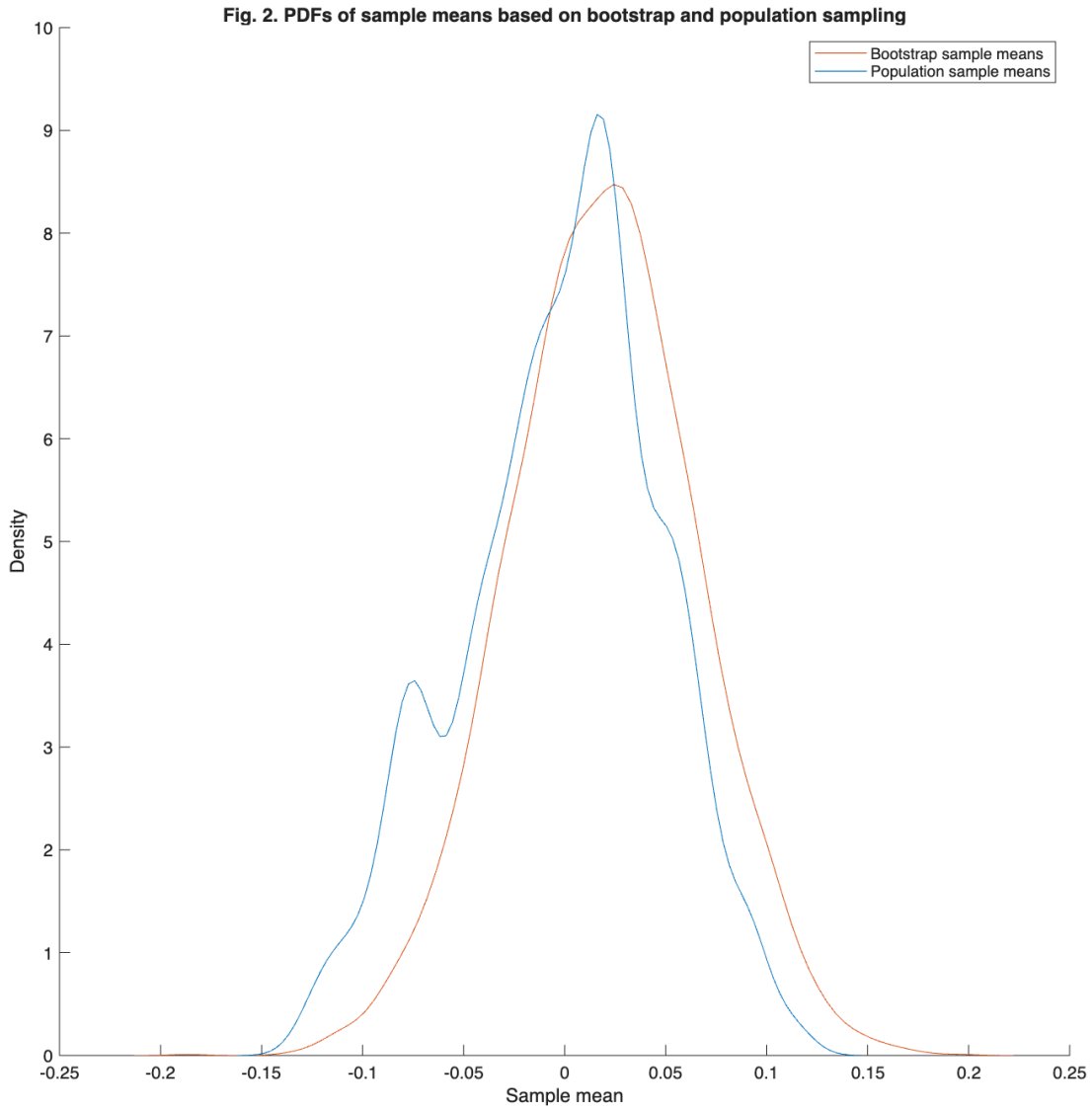
```
%% 8. Draw (bootsrap) samples from the initial sample

% 8.1. Preallocate vector to store (bootsrap) sample means
means_data_samples_boot = NaN(N_sim,1);

% 8.2. Draw samples of k blocks from the original sample and compute the sample
for i = 1:N_sim
    data_samples_boot = datasample(blocks,N_blocks,1,'Replace',true);
    data_samples_boot_flat = data_samples_boot(:);
    means_data_samples_boot(i) = mean(data_samples_boot_flat);
end
```

9. Plot estimated PDFs

We use `ksdensity` to estimate the probability density functions (PDFs) of the sample means obtained from both the block bootstrap and population-based sampling procedures. The bootstrap distribution is based on resampling blocks from a single sample originally drawn from the AR(1) population, while the benchmark distribution is derived from repeated sampling directly from the full simulated population. Figure 2 shows that the block bootstrap yields a PDF that closely resembles the benchmark distribution, successfully capturing the variance inflation induced by serial dependence. However, a noticeable difference in smoothness emerges between

the two curves. This is a natural consequence of the sampling methods: the block bootstrap resamples from a single contiguous realization of the AR(1) process, which limits variability and results in a smoother distribution of sample means. In contrast, population sampling draws from the entire simulated time series, encompassing a wider range of behaviors and producing a more dispersed and less smooth density estimate. This contrast highlights a key theoretical distinction, resampling from a fixed path versus sampling from the full data-generating process, and illustrates how the block bootstrap can slightly underestimate variability while still preserving the dependence structure inherent in the data.

```
%% 9. Plot estimated PDFs
figure
hold on
[f_boot,x_boot] = ksdensity(means_data_samples_boot,'function','pdf');
plot(x_boot,f_boot,'Color',[0.8500,0.3250,0.0980], ...
    'DisplayName','Bootstrap sample means');
[f_pop,x_pop] = ksdensity(means_data_samples_pop,'function','pdf');
plot(x_pop,f_pop,'Color',[0,0.4470,0.7410], ...
    'DisplayName','Population sample means');
title(['Fig. 2. PDFs of sample means based on bootstrap and ' ...
    'population sampling']);
ylabel('Density');
xlabel('Sample mean');
legend('show')
hold off
```

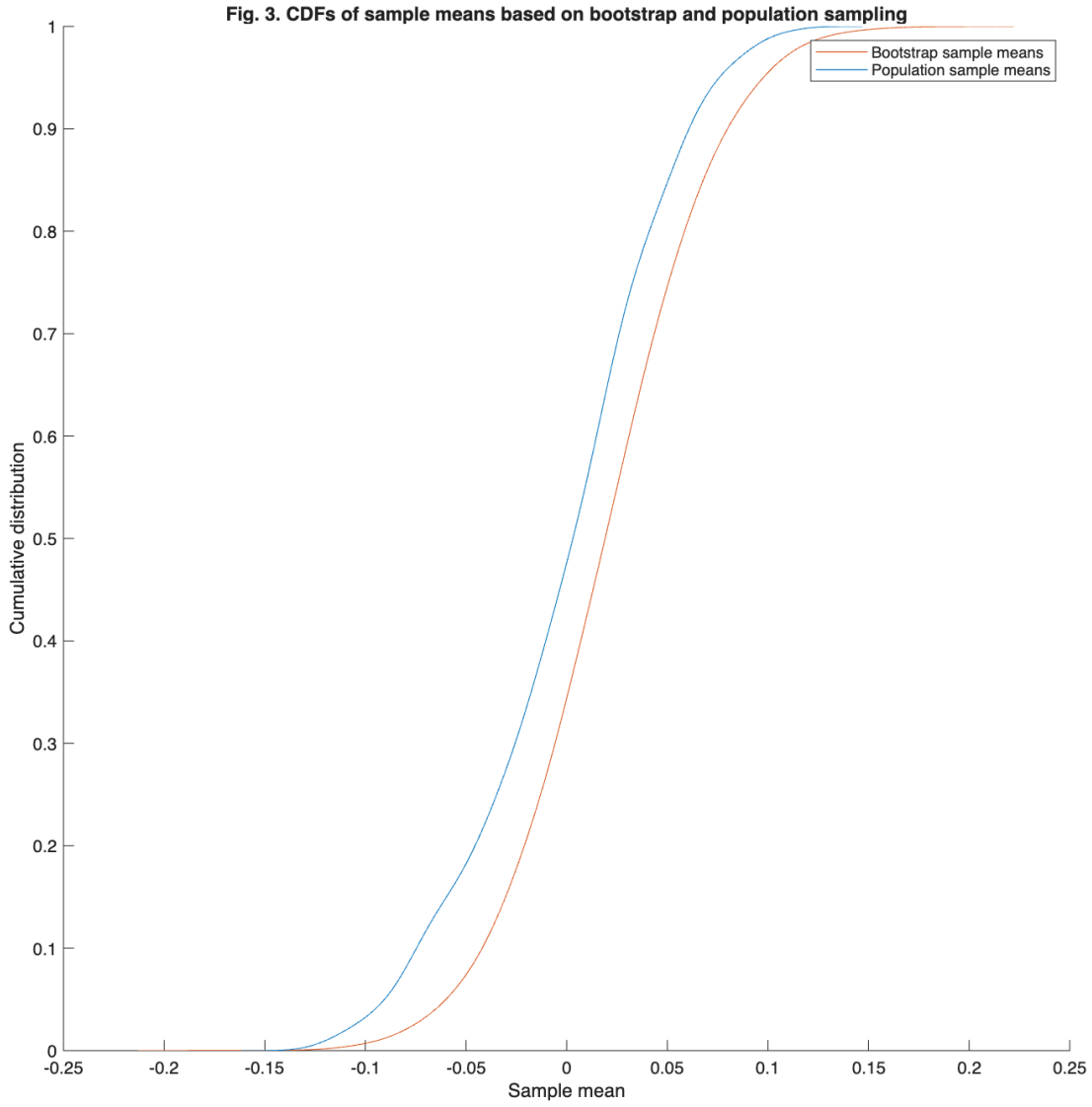


10. Plot estimated CDFs

Figure 3 compares the empirical CDFs of the sample means obtained via block bootstrap and direct population resampling. The close alignment between the two CDFs indicates that the bootstrap distribution $\hat{G}_{n,m}(\tau, F_n)$ provides a reliable approximation of the benchmark distribution $\hat{G}_{n,m}(\tau, \mathcal{P})$, thereby confirming the practical validity of the non-overlapping block bootstrap procedure under time dependence.

```
%% 10. Plot estimated CDFs
figure
hold on
[f_boot,x_boot] = ksdensity(means_data_samples_boot,'function','cdf');
plot(x_boot,f_boot,'Color',[0.8500,0.3250,0.0980], ...
    'DisplayName','Bootstrap sample means');
[f_pop,x_pop] = ksdensity(means_data_samples_pop,'function','cdf');
plot(x_pop,f_pop,'Color',[0,0.4470,0.7410], ...
    'DisplayName','Population sample means');
title(['Fig. 3. CDFs of sample means based on bootstrap and ' ...
```

```
    'population sampling']);
ylabel('Cumulative distribution');
xlabel('Sample mean');
legend('show');
hold off
```



6. Final notes

This file is prepared and copyrighted by Axel Zoons 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 four references. First is Radovanov and Marcikić, 2014. A comparison of four different block bootstrap methods. Croatian Operational Research Review. Second is Mignani and Rosa, 1995). The moving block bootstrap to assess the accuracy of statistical estimates in Ising model simulations. Computer physics communications. The third is Nordman and Lahiri, 2014. Convergence rates of empirical block length selectors for block bootstrap.

The fourth is Hall, P., Horowitz, J. L., Jing, B. Y., 1995. On blocking rules for the bootstrap with dependent data. Biometrika, 82(3), 561-574.