Assignment 5

xw-zeng

2022-12-14

Problem sets.

- `9.1`
- `9.4 (a)`
- `9.6 (a)(b)(c)(d)`

List of functions.

- `Bootstrap_Res`: Bootstrap regression with the method of bootstrapping the residuals.
- `Bootstrap_Case`: Bootstrap regression with the method of bootstrapping paired cases.
- `Subsample_Bootstrap`: Use Subsampling + Bootstrapping method to determine the optimal block size.
- `Moving_Block`: Moving block Bootstrap method.
- `Model_Based_np`: Assume an AR(1) model and bootstrap this model the method of bootstrapping the residuals.
- `Model_Based_p`: Assume an AR(1) model s.t. errors following normal distribution and bootstrap this model the method of bootstrapping the errors.
- `Blocks_of_Blocks`: Moving block bootstrap method with Blocks-of-Blocks strategy.

Load the R packages.

```
library(moments)
library(ggplot2)
```

9.1

$$X_1, X_2, ..., X_n \sim \text{i.i.d. Bernoulli}(\theta)$$
$$R(\mathcal{X}, F) = \bar{X} - \theta, \quad R^* = R(\mathcal{X}^*, \widehat{F}) = \bar{X}^* - \bar{X}$$
$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i, \quad \bar{X}^* = \frac{1}{n^*}\sum_{i=1}^{n^*} X_i^*$$
$$X_1^*, X_2^*, ..., X_n^* \sim \text{i.i.d. Bernoulli}(\bar{X}) \sim \widehat{F}$$
$$Let\ Z^* = \sum_{i=1}^{n^*} X_i^* \sim \text{i.i.d. Binomial}(n^*, \bar{X})$$
$$E^*(R^*) = E(\frac{1}{n^*}Z^* - \bar{X}) = \frac{1}{n^*}n^*\bar{X} - \bar{X} = 0$$
$$Var^*(R^*) = Var(\frac{1}{n^*}Z^*) = \frac{1}{n^{*2}}n^*\bar{X}(1-\bar{X}) = \frac{1}{n^*}\bar{X}(1-\bar{X})$$

9.4

(a)

The classic Beverton–Holt model for the relationship between spawners and recruits is:

$$R = \frac{1}{\beta_1 + \beta_2/S}$$
$$\frac{1}{R} = \beta_1 + \beta_2\frac{1}{S}, \quad \beta_1 \geq 0 \text{ and } \beta_2 \geq 0$$

where R and S are the numbers of recruits and spawners, respectively.

Let $T = R + S$. T is the total population abundance, which will only stabilize if $R = S$.

$$\frac{2}{T} = \beta_1 + \beta_2\frac{2}{T}$$
$$T = \frac{2(1-\beta_2)}{\beta_1} \Rightarrow R^* = S^* = T^*/2 = \frac{1-\beta_2^*}{\beta_1^*}$$

Load the data, and transform R, S to new variables $\frac{1}{R}$, $\frac{1}{S}$.

```
data <- read.table('salmon.dat', header = TRUE)
data$R1 <- 1 / data$recruits; data$S1 <- 1 / data$spawners
```

First obtain the observed data estimate of $\hat{R}$.

```
fit <- lm(R1 ~ S1, data)
R_hat <- (1 - fit$coefficients[2]) / fit$coefficients[1]; as.numeric(R_hat)
```

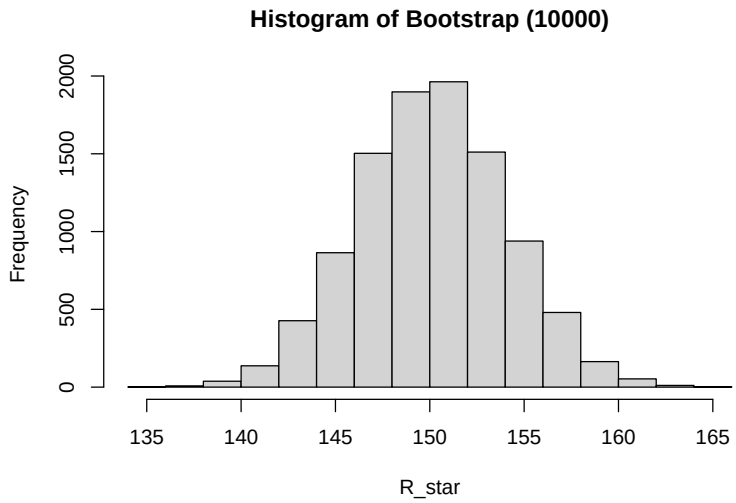
```
## [1] 150.0976
```

1. Bootstrapping the residuals.

```
Bootstrap_Res <- function(B){

  ###INITIAL VALUES###
  fit <- lm(R1 ~ S1, data)
  res <- residuals(fit)
  R_star <- rep(NA, B)

  ###MAIN###
  for (i in 1:B){
    res_bootstrap <- sample(res, length(res), replace = T)
    data$R2 <- fit$fitted.values + res_bootstrap
    fit2 <- lm(R2 ~ S1, data)
    beta_star <- fit2$coefficients
    R_star[i] <- (1 - beta_star[2]) / beta_star[1]
  }

  ###OUTPUT###
  hist(R_star, main = paste0('Histogram of Bootstrap (', B, ')'))
  return (R_star)
}
```

Histogram the bootstrap distribution.

```
set.seed(5201314)
result1 <- Bootstrap_Res(10000)
```



Obtain the corresponding 95% confidence interval.

```
quantile(result1, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 142.5282 157.8563
```

Obtain the mean and standard error for R^*.

```
print(paste('Mean:', mean(result1)))
print(paste('Standard Error:', sd(result1)))
```

```
## [1] "Mean: 150.132639243075"
## [1] "Standard Error: 3.96375803229591"
```

Obtain the skewness and kurtosis for R^*.

```
print(paste('Skewness:', skewness(result1)))
print(paste('Kurtosis:', kurtosis(result1)))
```

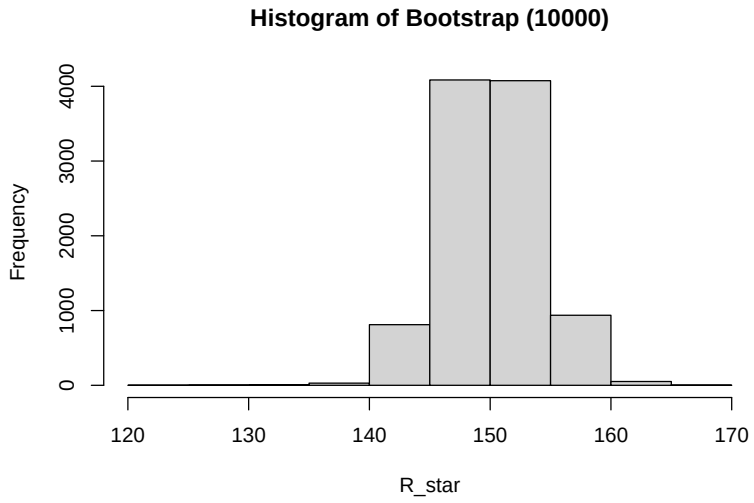
```
## [1] "Skewness: 0.0369427135780219"
## [1] "Kurtosis: 2.98009830362271"
```

2. Bootstrapping the cases.

```
Bootstrap_Case <- function(B){

  ###INITIAL VALUES###
  R1 <- data$R1; S1 <- data$S1; R_star <- rep(NA, B)

  ###MAIN###
  for (i in 1:B){
    idx <- sample(1:length(R1), length(R1), replace = T)
    R2 <- R1[idx]; S2 <- S1[idx]
    fit2 <- lm(R2 ~ S2)
    beta_star <- fit2$coefficients
    R_star[i] <- (1 - beta_star[2]) / beta_star[1]
  }

  ###OUTPUT###
  hist(R_star, main = paste0('Histogram of Bootstrap (', B, ')'))
  return (R_star)}
```

Histogram the bootstrap distribution.

```
set.seed(5201314)
result2 <- Bootstrap_Case(10000)
```



Obtain the corresponding 95% confidence interval.

```
quantile(result2, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 142.8971 157.6158
```

Obtain the mean and standard error for R^*.

```
print(paste('Mean:', mean(result2)))
print(paste('Standard Error:', sd(result2)))
```

```
## [1] "Mean: 150.107094724613"
## [1] "Standard Error: 3.80472782620562"
```

Obtain the skewness and kurtosis for R^*.

```
print(paste('Skewness:', skewness(result2)))
print(paste('Kurtosis:', kurtosis(result2)))
```

```
## [1] "Skewness: -0.0595999835862295"
## [1] "Kurtosis: 3.78175695413902"
```

3. Compare the results of two methods.

`BR` is referred to the method of Bootstrapping the residuals, and similarly `BC` is referred to the method of Bootstrapping the cases.

1. Bootstrap distribution:

- The bootstrap distributions of both methods are nearly symmetric, but there exist some outliers on the left side of the distribution of `BC`.
- Kurtosis: `BR` $<$ `BC`

2. Mean, standard error and confidence interval:

- Mean: `BR` $>$ `BC`
- Standard Error: `BR` $>$ `BC`
- Length of 95% Confidence Interval: `BR` $>$ `BC`

9.6

Load the data and difference the data.

```
data <- read.table('earthquake.dat', header = TRUE)
data <- diff(data$quakes, lag = 1)
```

(a)

Use `Subsampling + Bootstrapping` method to find the best block size.

Step 1 Find subsets of smaller size $m < n$, say $\mathcal{X}_i^{(m)} = (X_i, ..., X_{i+m-1})$.

Step 2 Bootstrap counterparts applied in $\mathcal{X}_i^{(m)}$, for a fixed l', $\hat{\phi}_{i,m}(l') = B^{-1} \sum_{b=1}^{B} H\left(\bar{X}_{i,b}^*\right) - H\left(\bar{X}_i\right)$.

Step 3 Estimate $\hat{\phi}_m(l')$ based on $\widehat{\mathrm{MSE}}\left\{\hat{\phi}_m(l')\right\} = \frac{1}{n-m+1} \sum_{i=1}^{n-m+1} \left(\hat{\phi}_{i,m}(l') - \hat{\phi}_0(l_0)\right)^2$.

Step 4 Find $l_{\mathrm{opt}}^{(m)'} = \arg\min \widehat{\mathrm{MSE}}\left\{\hat{\phi}_m(l')\right\}$, then $l_{\mathrm{opt}}^{(m)'} \approx \left(\frac{2c_2^2}{c_1}\right)^{1/3} m^{1/3}$.

Step 5 Rescale and find $l_{\mathrm{opt}} \approx l_{\mathrm{opt}}^{(m)'} (n/m)^{1/3}$.

In this problem, we don't exactly know what parameter we need to estimate and in general the parameter to be estimated is usually a function of μ with μ as the mean, so to find l_{opt} we can simply suppose that $\theta = \mu$.

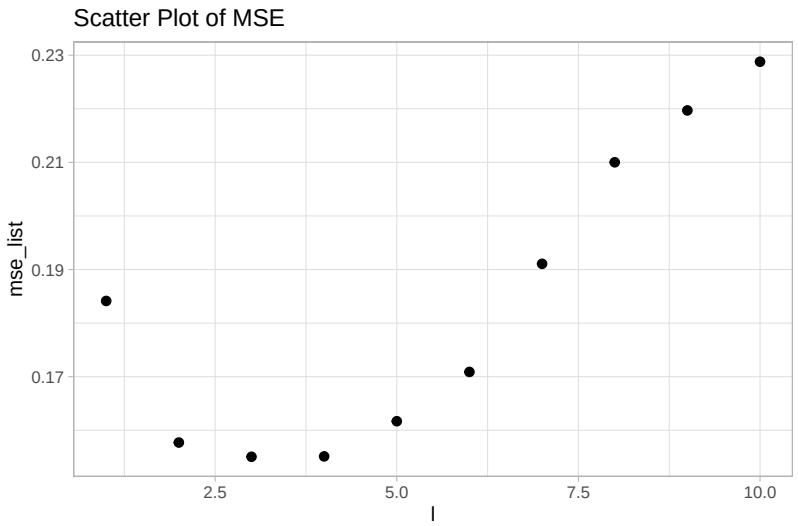
Define the function of `Subsampling + Bootstrapping` method.

```
Subsample_Bootstrap <- function(df, m, l0, B, l){

  ###INITIAL VALUES###
  n <- length(df)
  subsets <- matrix(NA, nrow = n - m + 1, ncol = m)
  for (i in 1:(n - m + 1)){subsets[i, ] <- df[i:(i + m - 1)]}

  ###FUNCTIONS###
  generate_phi_i <- function(df_sub, l, B){
    ##generate blocks
    b <- length(df_sub) - l + 1
    blocks <- matrix(NA, b, l)
    for (i in 1:b){blocks[i, ] <- df_sub[i:(i + l - 1)]}
    ##generate bootstrap counterparts
    bs <- round(length(df_sub) %/% l)
    X_mean <- rep(NA, B)
    for (i in 1:B){
      idx <- sample(b, bs, replace = T)
      tmp <- c()
      for (j in idx){tmp <- c(tmp, blocks[j, ])}
      X_mean[i] <- mean(tmp)}
    ##generate phi_i
    phi_i <- mean(X_mean) - mean(df)
    ##output
    return (phi_i)
  }

  generate_mse <- function(l, B, m){
    phi_i_list <- rep(NA, n - m + 1)
    for (i in 1:(n - m + 1)){phi_i_list[i] <- generate_phi_i(subsets[i, ], l, B)}
    phi_l0 <- generate_phi_i(df, l0, B)
    mse <- mean((phi_i_list - phi_l0) ^ 2)
    return (mse)
  }

  ###MAIN###
  mse_list <- rep(NA, length(l))
  for (i in l){mse_list[i] <- generate_mse(i, B, m)}
```

```
  l_opt_m <- which.min(mse_list)
  l_opt <- l_opt_m * (n / m) ^ (1 / 3)

  ###OUTPUT###
  p <- ggplot() + geom_point(aes(x = l, y = mse_list), size = 2) +
    labs(title = 'Scatter Plot of MSE')+ theme_light()
  print(p)
  return (round(l_opt))
}
```

Choose $m = 0.25n \approx 25, l_0 = 0.05n \approx 5$.

```
set.seed(5201314)
l_opt <- Subsample_Bootstrap(data, 25, 5, 2000, 1:10)
```



View l_{opt}.

```
print(paste('Optimal l =', l_opt))
```

```
## [1] "Optimal l = 5"
```

(b)

Obtain the observed data estimate for the 90th percentile of the annual change.

```
x_90 <- quantile(data, 0.9)
as.numeric(x_90)
```

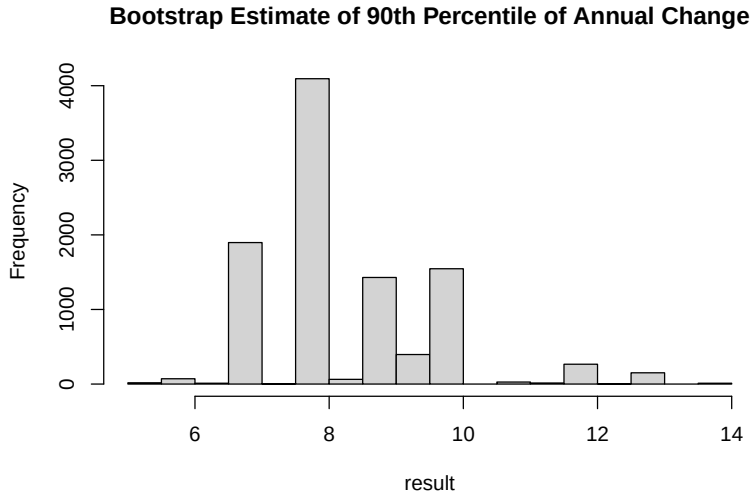
```
## [1] 8
```

Define the function of `Moving Block Bootstrap` method.

```
Moving_Block <- function(df, l, B){

  ###INITAL VALUES###
  b <- length(df) - l + 1
  ##generate blocks
  blocks <- matrix(NA, b, l)
  for (i in 1:b){blocks[i, ] <- df[i:(i + l - 1)]}
  bs <- round(length(df) %/% l)
  X_90 <- rep(NA, B)

  ###MAIN###
  for (i in 1:B){
    idx <- sample(b, bs, replace = T)
    tmp <- c()
    for (j in idx){tmp <- c(tmp, blocks[j, ])}
    X_90[i] <- quantile(tmp, 0.9)}

  ###OUTPUT###
  return (X_90)
}
```

Bootstrap for 10000 times and plot the distribution of bootstrap estimates.

```
set.seed(5201314)
result <- Moving_Block(data, 5, 10000)
hist(result, main = 'Bootstrap Estimate of 90th Percentile of Annual Change')
```



Estimate the standard error of this estimate using the moving block bootstrap.

```
print(paste('Standard Error:', sd(result)))
```

```
## [1] "Standard Error: 1.28131416795402"
```

(c)

1. Nonparametric Bootstrap

Bootstrap the residuals without any assumption on ϵ_t.

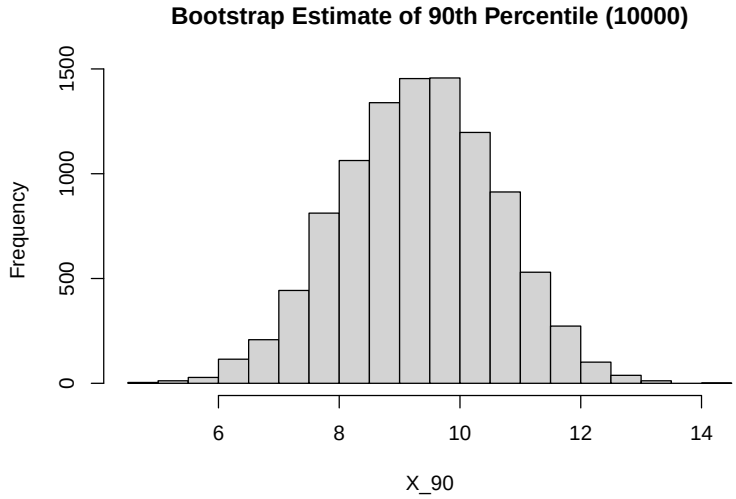
$$X_t = \alpha X_{t-1} + \epsilon_t$$

Define the function of nonparametric `Model-Based` approach.

```
Model_Based_np <- function(y, x, B){

  ###INITIAL VALUES###
  fit <- lm(y ~ x - 1)
  coef <- fit$coefficients
  res <- residuals(fit)
  X_90 <- rep(NA, B)

  ###MAIN###
  for (i in 1:B){
    res_bootstrap <- sample(res, length(res) + 2, replace = T)
    tmp <- res_bootstrap[1]
    for (j in 1:(length(res) + 1)){
      tmp <- c(tmp, coef %*% tmp[j] + res_bootstrap[j + 1])
    }
    X_90[i] <- quantile(tmp[-1], 0.9)
  }

  ###OUTPUT###
  hist(X_90, main = paste0('Bootstrap Estimate of 90th Percentile (', B, ')'))
  return (X_90)
}
```

Histogram the bootstrap distribution.

```
set.seed(5201314)
result <- Model_Based_np(data[2:98], data[1:97], 10000)
```



Estimate the standard error for the estimate from part (b).

```
print(paste('Standard Error:', sd(result)))
```

```
## [1] "Standard Error: 1.29256563482098"
```

2. Parametric Bootstrap

Bootstrap the errors with an assumption of $\epsilon_t \sim N(0, \sigma^2)$.

$$X_t = \alpha X_{t-1} + \epsilon_t, \epsilon_t \sim N(0, \sigma^2)$$

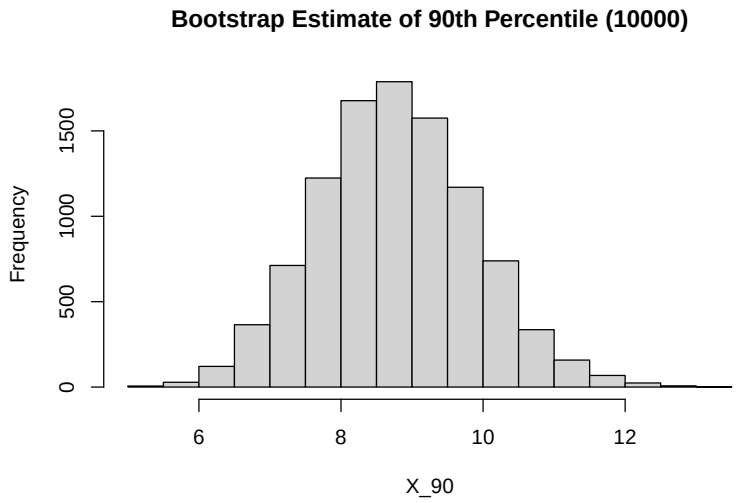
Define the function of parametric `Model-Based` approach.

```
Model_Based_p <- function(y, x, B){

  ###INITIAL VALUES###
  fit <- lm(y ~ x - 1)
  coef <- fit$coefficients
  res <- residuals(fit)
  X_90 <- rep(NA, B)

  ###MAIN###
  for (i in 1:B){
```

```
    res_bootstrap <- rnorm(length(res) + 2, 0, sd(res))
    tmp <- res_bootstrap[1]
    for (j in 1:(length(res) + 1)){
      tmp <- c(tmp, coef %*% tmp[j] + res_bootstrap[j + 1])
    }
    X_90[i] <- quantile(tmp[-1], 0.9)
  }

  ###OUTPUT###
  hist(X_90, main = paste0('Bootstrap Estimate of 90th Percentile (', B, ')'))
  return (X_90)
}
```

Histogram the bootstrap distribution.

```
set.seed(5201314)
result <- Model_Based_p(data[2:98], data[1:97], 10000)
```



Estimate the standard error for the estimate from part (b).

```
print(paste('Standard Error:', sd(result)))
```

```
## [1] "Standard Error: 1.10551685321516"
```

So we may conclude:

- Standard Error: Nonparametric Model-based Bootstrap > Moving Block Bootstrap > Parametric Model-based Bootstrap.

(d)

Obtain the observed data estimate for the lag-1 autocorrelation of the annual change.

$$\hat{r} = \sum_{t=1}^{n-1} (X_t - M)(X_{t+1} - M) / \sum_{t=1}^{n} (X_t - M)^2$$

Where n is the length of the series and M is the mean of $X_1, ..., X_n$.

```
m <- mean(data)
rho_hat <- sum((data[1:97] - m) * (data[2:98] - m)) / sum((data - m) ^ 2); rho_hat
```

```
## [1] -0.3661926
```

$\mathcal{X}$ yields 97 couples $Y_t = (X_t, X_{t+1})$, and the vectorized series is $\mathcal{Y} = (Y_1, Y_2, ..., Y_{97})$. From these 97 blocks, we may resample blocks of blocks. Let each of these blocks of blocks be comprised of 5 of the small blocks. The lag 1 autocorrelation can be estimated as:

$$\hat{r}^* = \sum_{t=1}^{n^*} (Y^*_{t,1} - M)(Y^*_{t,2} - M) / \sum_{t=1}^{n} (X_t - M)^2$$

Where $Y^*_{t,j}$ is the j^{th} element in Y^*_t and n^* is the length of Y^*_t.

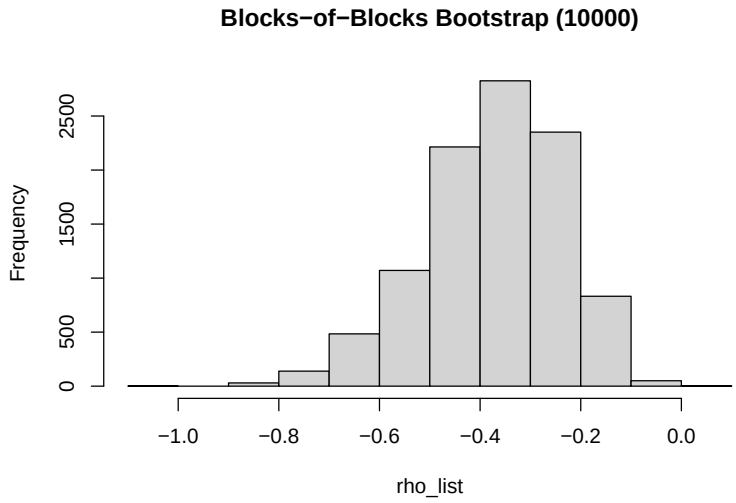
Define the function of the moving block bootstrap with `Blocks-of-Blocks` strategy.

```
Blocks_of_Blocks <- function(df, l, B, method = NA){

  ###INITIAL VALUES###
  rho_list <- rep(NA, B)
  M <- mean(df)
  b <- length(df) - 1
  ##generate blocks of blocks
  smallblocks <- rbind(df[1:b], df[2:(b + 1)])
  bigblocks <- array(NA, c(2, l, b - l + 1))
  for (i in 1:(b - l + 1)){bigblocks[, , i] <- smallblocks[, i:(i + l - 1)]}
  bs <- round(length(df) %/% l)

  ###FUNCTIONS###
  calculate_rho <- function(y){
    if (is.na(method)){sum((y[1, ] - M) * (y[2, ] - M)) / sum((df - M) ^ 2)}
    else {cor(y[1, ], y[2, ])}
  }

  ###MAIN###
```

```
  for (i in 1:B){
    idx <- sample(1:(b - l + 1), bs, replace = T)
    tmp <- c(); for (j in idx){tmp <- cbind(tmp, bigblocks[, , j])}
    rho_list[i] <- calculate_rho(tmp)
  }

  ###OUTPUT###
  hist(rho_list, main = paste0('Blocks-of-Blocks Bootstrap (', B, ')'))
  return(rho_list)
}
```

Histogram the bootstrap distribution of lag-1 autocorrelation of the annual change.

```
set.seed(5201314)
result <- Blocks_of_Blocks(data, 5, 10000)
```



Obtain the standard error of the estimate.

```
print(paste('Standard Error:', sd(result)))
```

```
## [1] "Standard Error: 0.13597669184054"
```

Obtain the bootstrap bias.

$$\hat{\Delta} = \frac{1}{B}\sum_{i=1}^{B}(\hat{r}^* - \hat{r}) = \frac{1}{B}\sum_{i=1}^{B}\hat{r}^* - \hat{r}$$

```
mean(result) - rho_hat
```

```
## [1] -0.008038157
```

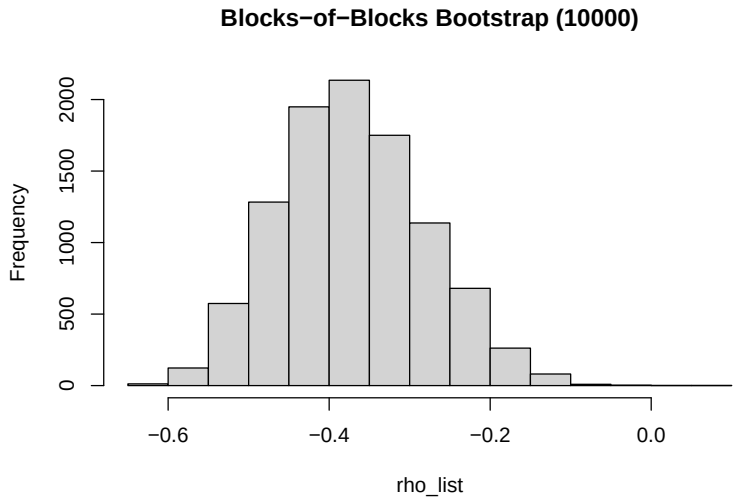
We can also obtain the observed data estimate with `cor` function.

```
rho_hat <- cor(data[1:97], data[2:98]); rho_hat
```

```
## [1] -0.3668661
```

Using this method to estimate lag-1 autocorrelation and bootstrap the distribution.

```
set.seed(5201314)
result <- Blocks_of_Blocks(data, 5, 10000, 'cor')
```



Obtain the standard error of the estimate.

```
print(paste('Standard Error:', sd(result)))
```

```
## [1] "Standard Error: 0.0906526357242568"
```

Obtain the bootstrap bias.

```
mean(result) - rho_hat
```

```
## [1] -0.003928003
```

So we can conclude that the first estimator is inferior to the second estimator because the former is more biased (due to approximation) and less efficient.