Exercice n°4

Sample distribution and Central Limit Theorem

Grégoire de Lambertye

2022-11-14

I Task n°1 :

We consider the 12 sample data points: 4.94 5.06 4.53 5.07 4.99 5.16 4.38 4.43 4.93 4.72 4.92 4.96

I.1 How many possible bootstrap samples are there, if each bootstrap sample has the same size as the original?

Two bootstrap sample are considered equals if they are composed by the same elements without considering the order. With 12 points we can have up to 5200300 different bootstrap

$$\text{Number of boostrap samples} = \binom{2n+1}{n} = \binom{25}{12} = 5200300$$

I.2 Compute the mean and the median of the original sample.

```r
data <- c(4.94, 5.06, 4.53, 5.07, 4.99, 5.16, 4.38, 4.43, 4.93, 4.72, 4.92, 4.96)

original_mean <- mean(data)
original_median <- median(data)
```

The mean of the original sample is 4.8408333 and the median is 4.935

I.3 Create 2000 bootstrap samples and compute their means.

We create 2000 samples with the same length than the original sample.

```r
set.seed(12202211)
m <- 2000
samples <- replicate(m, sample(data, size=12, replace=TRUE))
means <- numeric(m)
for(i in 0:m){
  means[i] <- mean(samples[,i])
}
```

Compute the mean on the first 20.0.0 bootstrap means.

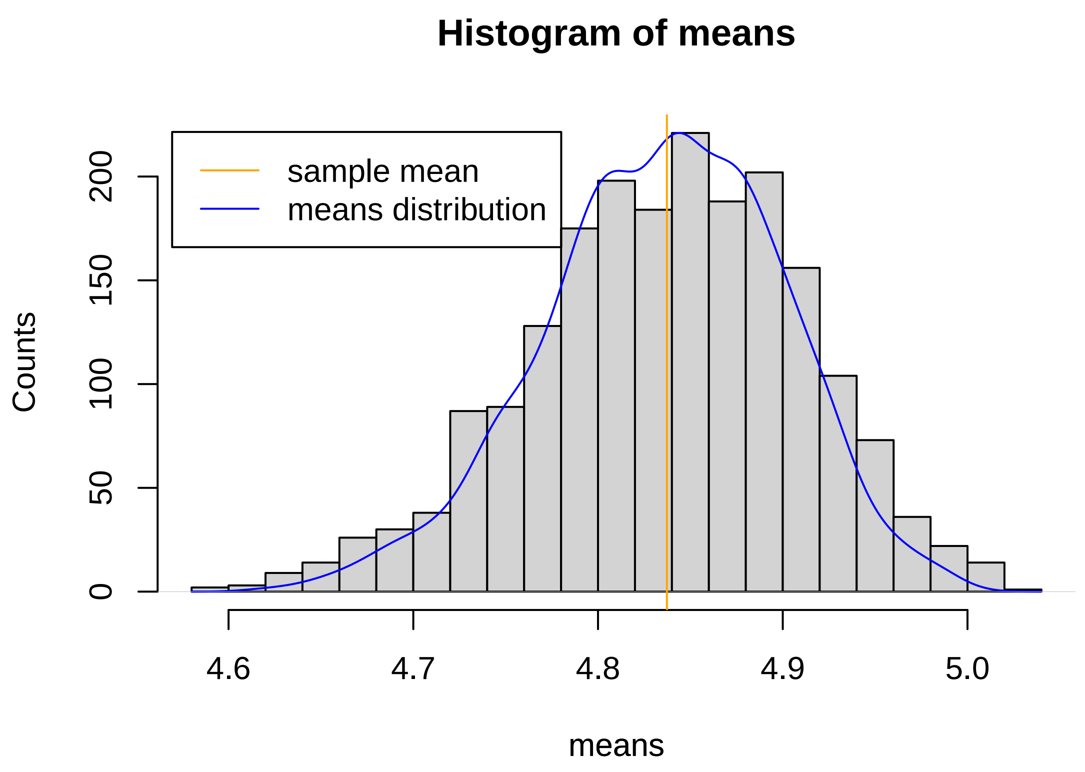
```r
mean_of_mean <- function(n){
  sum <- 0
  for(i in 1:n){
    sum <- sum + mean(samples[,i])
  }
  return(sum/n)
}
mean_20b <- mean_of_mean(20)
mean_200b <- mean_of_mean(200)
mean_2000b <- mean_of_mean(2000)
means_first_sample <- c(mean_20b, mean_200b, mean_2000b)
```

We have the following result :

number of sample	20	200	2000
mean	4.8549167	4.8411458	4.8378087

Visualise the distribution of all the different bootstrap means to the sample mean. Does the Central Limit Theorem kick in?

```r
par(mar = c(5, 4, 4, 4) + 0.3)
hist(means, ylab = "Counts", breaks = 25)
par(new = TRUE)
plot(density(means), axes = FALSE, xlab = "means", ylab = "", col='blue', main="")
abline(v=original_mean, col='orange')
legend(x=4.53, y=5.2, legend=c("sample mean", "means distribution"),col=c("orange", "blue"), lty=c(1,1)
```



The means distribution seems normaly distributed around the sample mean. So the Central Limit Theorem seems to kick in.

Based on the three different bootstrap sample lengths in 3. compute the corresponding 0.025 and 0.975 quantiles. Compare the three resulting intervals against each other and the "true" confidence interval of the mean under the assumption of normality. (Use for example the function t.test to obtain the 95% percent CI based on asympotic considerations for the mean.)

```
library("ggplot2")
a <- .025
b <- 1-a

print(paste('CI. for means (20): [', quantile(means[1:20], a),',', quantile(means[1:20], b),']'))
```

```
## [1] "CI. for means (20): [ 4.70489583333333 , 5.01627083333333 ]"
```

```
print(paste('CI. for means (200): [', quantile(means[1:200], a),',', quantile(means[1:200], b),']'))
```

```
## [1] "CI. for means (200): [ 4.69164583333333 , 4.990125 ]"
```

```
print(paste('CI. for means (200): [', quantile(means[1:2000], a),',', quantile(means[1:2000], b),']'))
```

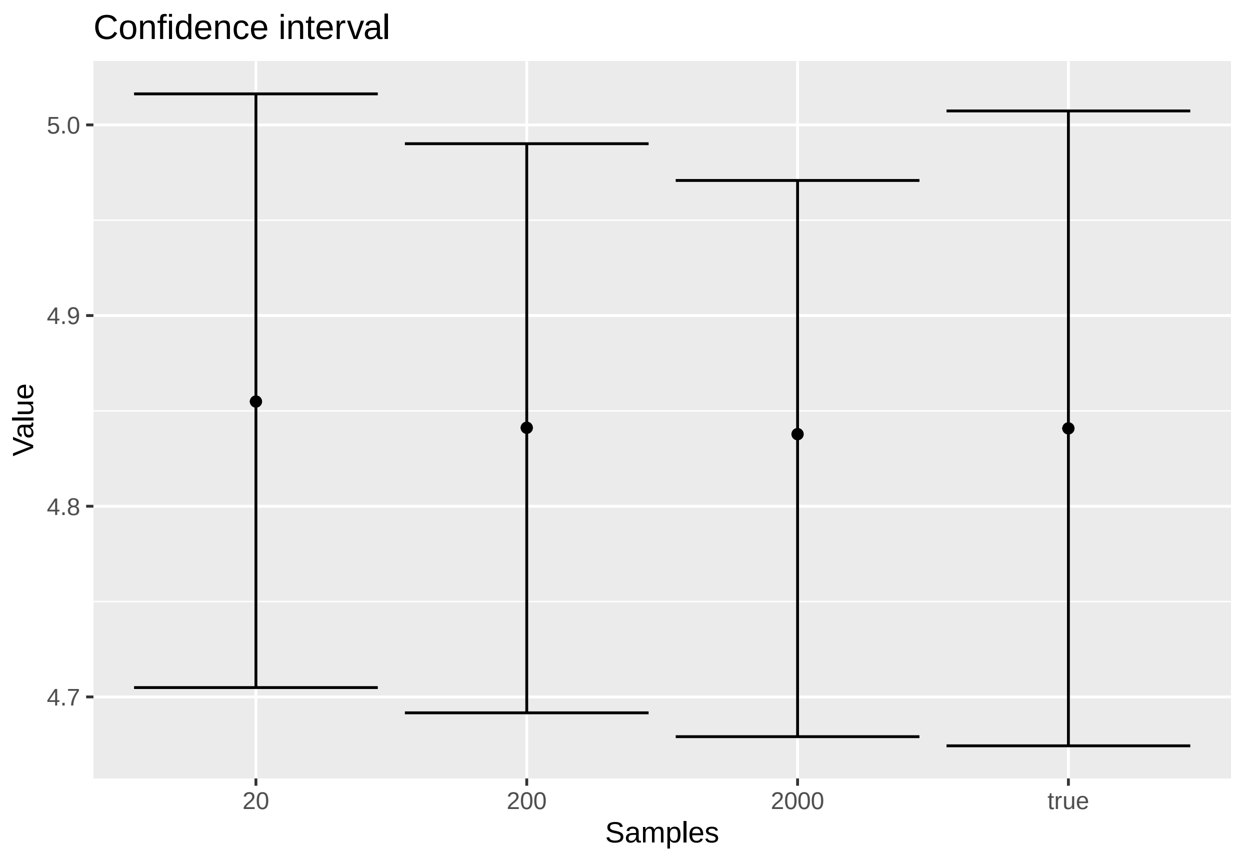
```
## [1] "CI. for means (200): [ 4.67914583333333 , 4.97085416666667 ]"
```

```r
true_confidence_interval <- t.test(data, conf.level = 0.95)$conf.int
print(paste('CI. for \'true\' interval: [',true_confidence_interval[1],',',true_confidence_interval[2],
```

```
## [1] "CI. for 'true' interval: [ 4.6743435855868 , 5.00732308107987 ]"
```

```r
x = c('20','200','2000','true')
y = c(mean_20b, mean_200b, mean_2000b, original_mean)
lower = c(quantile(means[1:20], a), quantile(means[1:200], a), quantile(means[1:2000], a),true_confidenc
upper = c(quantile(means[1:20], b), quantile(means[1:200], b),  quantile(means[1:2000], b),true_confider
df <- data.frame(x, y, lower, upper)

ggplot(df, aes(df[,1], df[,2])) +
  geom_point() +
  geom_errorbar(aes(ymin = df[,3], ymax = df[,4]))+
  ggtitle("Confidence interval") +
  xlab("Samples") +
  ylab("Value")
```



When computing CI with larger vectors, the interval goes lower. They also are smaller than the "true one" comming from the t-test.

I.4 Create 2000 bootstrap samples and compute their medians.

We keep the 2000 created samples from the last part.

```
medians <- numeric(m)
for(i in 0:m){
  medians[i] <- median(samples[,i])
}
```

Compute the mean on the first 20.0.0 bootstrap medians.

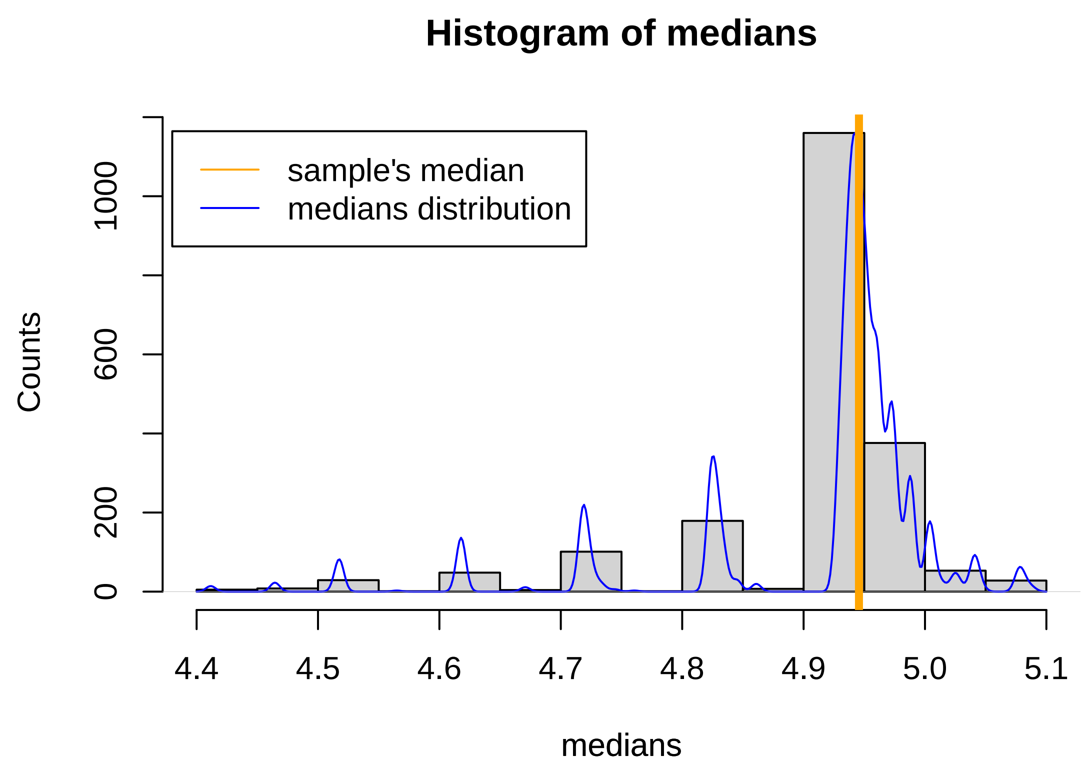
```
mean_of_medians <- function(n){
  sum <- 0
  for(i in 1:n){
    sum <- sum + medians[i]
  }
  return(sum/n)
}
median_20b <- mean_of_medians(20)
median_200b <- mean_of_medians(200)
median_2000b <- mean_of_medians(2000)
means_first_sample <- c(mean_20b, mean_200b, mean_2000b)
```

We have the following result :

number of sample	20	200	2000
medians	4.92725	4.91515	4.9075375

Visualise the distribution all the different bootstrap medians to the sample median.

```
par(mar = c(5, 4, 4, 4) + 0.3)
hist(medians, ylab = "Counts")
par(new = TRUE)
plot(density(medians), axes = FALSE, xlab = "medians", ylab = "", col='blue', main="")
abline(v=original_median, col='orange', lwd = 4)
legend(x=4.4, y=22, legend=c("sample's median", "medians distribution"),col=c("orange", "blue"), lty=c(
```



Based on the three different bootstrap sample lengths in 3. compute the corresponding 0.025 and 0.975 quantiles. Compare the three resulting intervals against each other.

```
library("ggplot2")
a <- .025
b <- 1-a

print(paste('CI. for median (20): [', quantile(medians[1:20], a),',', quantile(medians[1:20], b),']'))
```

```
## [1] "CI. for median (20): [ 4.6725 , 5.043375 ]"
```

```
print(paste('CI. for median (200): [', quantile(medians[1:200], a),',', quantile(medians[1:200], b),']'
```

```
## [1] "CI. for median (200): [ 4.67375 , 5.06 ]"
```

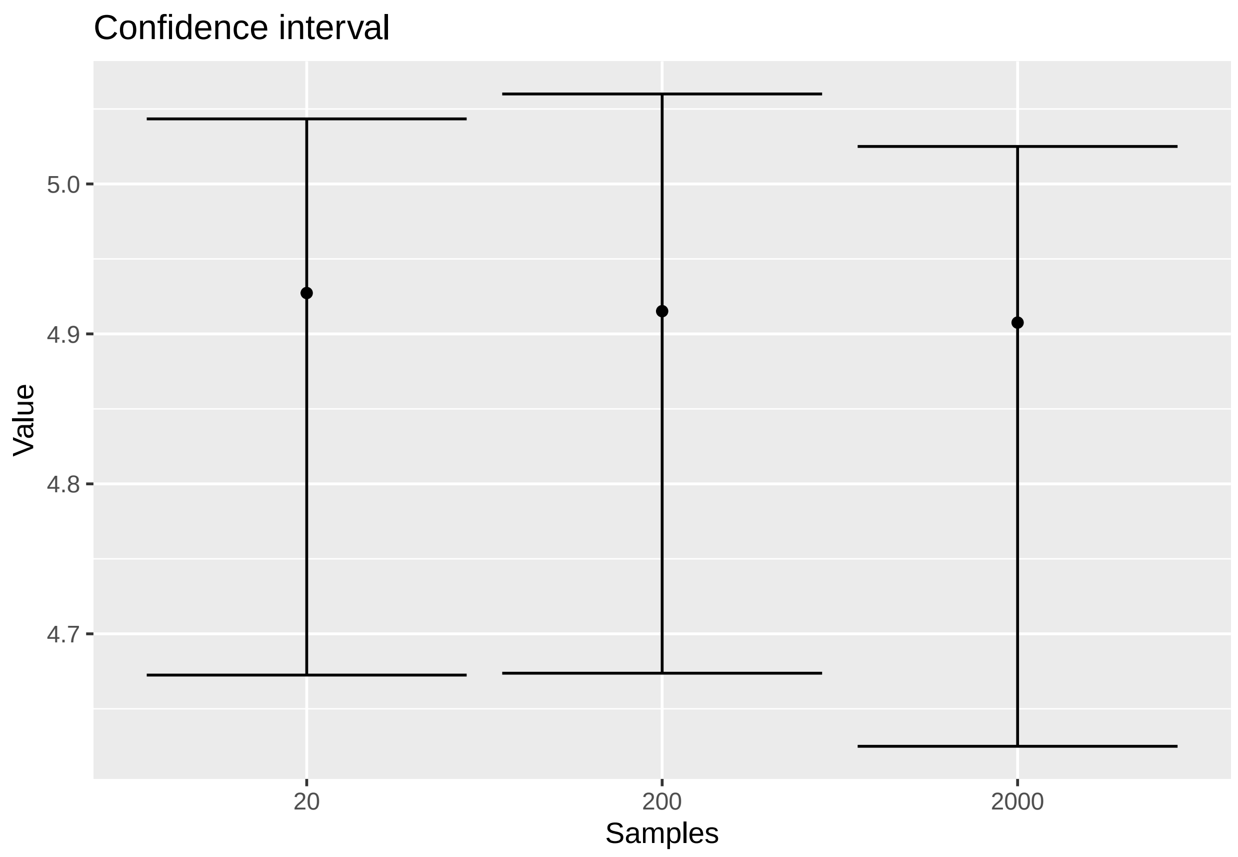
```
print(paste('CI. for median (200): [', quantile(medians[1:2000], a),',', quantile(medians[1:2000], b),'
```

```
## [1] "CI. for median (200): [ 4.625 , 5.025 ]"
```

```
x = c('20','200','2000')
y = c(median_20b, median_200b, median_2000b)
```

```r
lower = c(quantile(medians[1:20], a), quantile(medians[1:200], a), quantile(medians[1:2000], a))
upper = c(quantile(medians[1:20], b), quantile(medians[1:200], b),  quantile(medians[1:2000], b))
df <- data.frame(x, y, lower, upper)

ggplot(df, aes(df[,1], df[,2])) +
  geom_point() +
  geom_errorbar(aes(ymin = df[,3], ymax = df[,4]))+
  ggtitle("Confidence interval") +
  xlab("Samples") +
  ylab("Value")
```



The confidence interval size on the median is pretty stable when the sample size change.

II Task n°2 :

We wish to explore the effect of outliers on the outcomes of Bootstrap Sampling. ## II.1 Set your seed to 1234. And then sample 1960 points from a standard normal distribution to create the vector x.clean then sample 40 observations from uniform(4,5) and denote them as x.cont. The total data is x <- c(x.clean,x.cont). After creating the sample set your seed to your immatriculation number.

```r
set.seed(1234)
x.clean <- rnorm(1960)
x.cont <- runif(40,4,5)
x <- c(x.clean,x.cont)
set.seed(12202211)
```

II.2 Estimate the median, the mean and the trimmed mean with alpha = 0.05 for x and x.clean.

```
x_mean <- mean(x)
x_median <- median(x)
x_trimed_mean <- mean(x, trim=0.05)

xclean_mean <- mean(x.clean)
xclean_median <- median(x.clean)
xclean_trimed_mean <- mean(x.clean, trim=0.05)
```

sample	median	mean	trimed mean
x	0.0113797	0.0839551	0.0368329
x.clean	-0.0172536	-0.005969	-0.0014626

II.3 Use nonparametric bootstrap (for x and x.clean) to calculate the standard error and the 95 percentile CI of all 3 estimators.

To estimate the standard error of (a univariate) theta using nonparametric bootstrap the standard deviation of the bootstrap estimates is used:

$$\hat{se}(\hat{\theta}) = sqrt(\frac{1}{m-1}\sum(\hat{\theta}^i - moy(\hat{\theta}))^2)$$

```
m <- 2000
set.seed(12202211)

#Means
x_smpl <- replicate(m, sample(x, size=2000, replace=TRUE))
x_mean <- numeric(m)
for(i in 1:m){
  x_mean[i] <- mean(x_smpl[,i])
}

x_clean_smpl <- replicate(m, sample(x.clean, size=2000, replace=TRUE))
x_clean_mean <- numeric(m)
for(i in 1:m){
  x_clean_mean[i] <- mean(x_clean_smpl[,i])
}

#Standard error
sex <- sqrt(1 / (m - 1) * sum((x_mean - (mean(x_mean)))^2))
sex_clean <- sqrt(1 / (m - 1) * sum((x_clean_mean - mean(x_clean_mean))^2))

#CI
a <- 0.05
b <- 1 - a

ci_mean = c(
  quantile(x_mean, a), quantile(x_mean, b),
  quantile(x_clean_mean, a), quantile(x_clean_mean, b)
)

#Medians
medians_x <- numeric(m)
for (i in 1:m) {x_mean[i] <- median(x_smpl[,i])}
medians_x_clean <- numeric(m)
for (i in 1:m) {x_clean_smpl[i] <- median(x_clean_smpl[,i])}

#trimmed mean
trimmean_x <- numeric(m)
for (i in 1:m) {trimmean_x[i] <- mean(x_smpl[,i], trim=0.05)}
trimmean_x_clean <- numeric(m)
for (i in 1:m) {trimmean_x_clean[i] <- mean(x_clean_smpl[,i], trim=0.05)}

#CI
ci_median = c(
quantile(medians_x, a), quantile(medians_x, b),
quantile(medians_x_clean, a), quantile(medians_x_clean, b))

ci_trimmean = c(
  quantile(trimmean_x, a), quantile(trimmean_x, b),
  quantile(trimmean_x_clean, a), quantile(trimmean_x_clean, b))

df <- data.frame(
    sample=c('X', 'X.clean'),
    StandardError=c(sex, sex_clean),
    CIMeanLeft=c(ci_mean[1], ci_mean[3]),
    CIMeanRight=c(ci_mean[2], ci_mean[4]),
    CIMedianLeft=c(ci_median[1], ci_median[3]),
    CIMedianRight=c(ci_median[2], ci_median[4]),
    CITrimMeanLeft=c(ci_trimmean[1], ci_trimmean[3]),
    CITrimMeanRight=c(ci_trimmean[2], ci_trimmean[4])
)
knitr::kable(df, caption = "Standard error and CI for 3 estimators")
```

Table 4: Standard error and CI for 3 estimators

sample	StandardError	CIMeanLeft	CIMeanRight	CIMedianLeft	CIMedianRight	CITrimMeanLeft	CITrimMeanRight
X	0.0263069	0.0424261	0.1295130	0	0	-0.0000460	0.0777442
X.clean	0.0221006	-0.0414229	0.0315602	0	0	-0.0364382	0.0364772

II.4 Use parametric bootstrap (based on x and x.clean) to calculate: - bias - standard error - 95 percentile CI - bias corrected estimate for the mean and the trimed mean

In a parametric bootstrap, we assume our data follow a normal distribution and we can create samples from $N(\bar{x}, s_n^2)$

```r
#Original values
mean_estimate_x <- mean(x)
mean_estimate_x_clean <- mean(x.clean)
sd_estimate_x <- sd(x)
sd_estimate_x_clean <- sd(x.clean)
trimmean_estimate_x <- mean(x, trim=0.05)
trimmean_estimate_x_clean <- mean(x.clean, trim=0.05)

#Mean and sample creation
samples_x <- replicate(m, rnorm(length(x), mean=mean_estimate_x, sd=sd_estimate_x))
means_x <- numeric(m)
for (i in 1:m) {means_x[i] <- mean(samples_x[,i])}

samples_x_clean <- replicate(m, rnorm(2000, mean=mean_estimate_x_clean, sd=sd_estimate_x_clean))
means_x_clean <- numeric(length(samples_x_clean))
for (i in 1:m) {means_x_clean[i] <- mean(samples_x_clean[,i])}

#Standard error
standard_error_x <- sqrt(1 / (m - 1) * sum((means_x - mean(means_x))^2))
standard_error_x_clean <- sqrt(1 / (m - 1) * sum((means_x_clean - mean(means_x_clean))^2))

#CI
a <- 0.05
b <- 1 - a
ci_mean = c(
  quantile(means_x, a), quantile(means_x, b),
  quantile(means_x_clean, a), quantile(means_x_clean, b)
)

medians_x <- numeric(m)
for (i in 1:m) {means_x[i] <- median(samples_x[,i])}
medians_x_clean <- numeric(m)

for (i in 1:m) {means_x_clean[i] <- median(samples_x_clean[,i])}
trimmean_x <- numeric(m)

for (i in 1:m) {trimmean_x[i] <- mean(samples_x[,i], trim=0.05)}
trimmean_x_clean <- numeric(m)

for (i in 1:m) {trimmean_x_clean[i] <- mean(samples_x_clean[,i], trim=0.05)}
ci_median = c(
  quantile(medians_x, a), quantile(medians_x, b),
  quantile(medians_x_clean, a), quantile(medians_x_clean, b))

ci_trimmean = c(
  quantile(trimmean_x, a), quantile(trimmean_x, b),
  quantile(trimmean_x_clean, a), quantile(trimmean_x_clean, b))

bias_mean_x <- mean(means_x) - mean_estimate_x
bias_trim_mean_x <- mean(trimmean_x) - trimmean_estimate_x
bias_mean_x_clean <- mean(means_x_clean) - mean_estimate_x_clean
bias_trim_mean_x_clean <- mean(trimmean_x_clean) - trimmean_estimate_x_clean
```

II.5 Compare and summarize your findings with tables and graphically

```r
table <- data.frame(
  sample=c('X', 'X.clean'),
  Bias_Mean=c(bias_mean_x, bias_mean_x_clean),
  Bias_TrimMean=c(bias_trim_mean_x, bias_trim_mean_x_clean),
  StandardError=c(standard_error_x, standard_error_x_clean)
)
knitr::kable(table, caption = "Standard Error and Bias for the estimates")
```

Table 5: Standard Error and Bias for the estimates

sample	Bias_Mean	Bias_TrimMean	StandardError
X	0.0004865	0.0473983	0.0261849
X.clean	0.0059660	-0.0039562	0.0226672

```r
table_2 <- data.frame(
  sample=c('X', 'X.clean'),
  CIMeanLeft=c(ci_mean[1], ci_mean[3]),
  CIMeanRight=c(ci_mean[2], ci_mean[4]),
  CIMedianLeft=c(ci_median[1], ci_median[3]),
  CIMedianRight=c(ci_median[2], ci_median[4]),
  CITrimMeanLeft=c(ci_trimmean[1], ci_trimmean[3]),
  CITrimMeanRight=c(ci_trimmean[2], ci_trimmean[4])
)
knitr::kable(table_2, caption = "CI for the estimates")
```

Table 6: CI for the estimates

sample	CIMeanLeft	CIMeanRight	CIMedianLeft	CIMedianRight	CITrimMeanLeft	CITrimMeanRight
X	0.0400675	0.1277601	0	0	0.0400210	0.1285614
X.clean	0.0000000	0.0000000	0	0	-0.0420732	0.0307988

```r
table <- data.frame(
  sample=c('X', 'X.clean'),
  MeanEstimator=c(mean_estimate_x, mean_estimate_x_clean),
  TrimMeanEstimator=c(trimmean_estimate_x, trimmean_estimate_x_clean),
  bias_corrected_mean=c(mean_estimate_x - bias_mean_x, mean_estimate_x_clean - bias_mean_x_clean),
  bias_corrected_trimmean=c(trimmean_estimate_x - bias_trim_mean_x, trimmean_estimate_x_clean - bias_tri
)
names(table) <- c(
  'Sample', 'Mean', 'Trimmed Mean', 'Unbiased Mean', 'Unbiased Trimmed Mean'
)
knitr::kable(table, caption = "Estimators and Bias Corrected Estimators")
```

Table 7: Estimators and Bias Corrected Estimators

Sample	Mean	Trimmed Mean	Unbiased Mean	Unbiased Trimmed Mean
X	0.0839551	0.0368329	0.0834685	-0.0105654
X.clean	-0.0059690	-0.0014626	-0.0119350	0.0024936

III Task n°3 :

Based on the above tasks and your lecture materials, explain the methodology of bootstrapping for the construction of confidence intervals and parametric or non-parametric tests.

The main idea of bootstrapping is to create samples from the original dataset and use them to estimate and challenge estimators. In non-parametric bootsraping, the bootstrap are generated by picking element in the original sample (with replacement). In parametric, we assume the data follow a normal distribution and we can generate sample with a 'rnorn()' by using the sample's mean and standard deviation.

Bootstraping easily allows you to calculate standard errors, construct confidence intervals, and perform hypothesis testing. To build a confidence interval around a θ estimator we have to follow those steps : 1. Resample m times from the original sample (method differs for parametric and non-parametric bootstrap) and compute the estimator for each sample, 2. The confidence interval is given by the percentile of theses sample, take the 0.025 and 0.975 quantile to obtain a 95% interval for example.

Boostraping also allows to compute pramaetric and non parametric test: To build a t-test with bootstrap we have to follow those steps : 1. Resample m times from the original sample (method differs for parametric and non-parametric bootstrap) and compute the t-test for each bootstrap. 2. The p-value of the t-test is obtained with

$$p = \frac{\sum_{i=1}^{m} I(t_i^* \geq t)}{m}$$

with m, the number of bootstrap, $I(x) = 1$ when x is true and $I(x) = 1$ when x is false, t the p-value of the original t-test and t_i^* the bootstrap's t-test p-value.