Ex4

Yelyzaveta Klysa

2023-10-13

Contents

Task 1

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.

```
set.seed(12208877)
```

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

Bootstrapp does not account for the order, it also allows repetition, so we can use formula of an Unordered Sampling with Replacement: n + k -1 over k:

```
sample_orig <- 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)
choose(2*length(sample_orig)-1, length(sample_orig))
```

```
## [1] 1352078
```

Overall we get 1352078 possible samples.

1.2 Compute the mean and the median of the original sample.

```
cat("mean: ", mean(sample_orig), "\n")
```

```
## mean:  4.840833
```

```
cat("median: ", median(sample_orig))
```

```
## median:  4.935
```

1.3 Create 2000 bootstrap samples and compute their means.

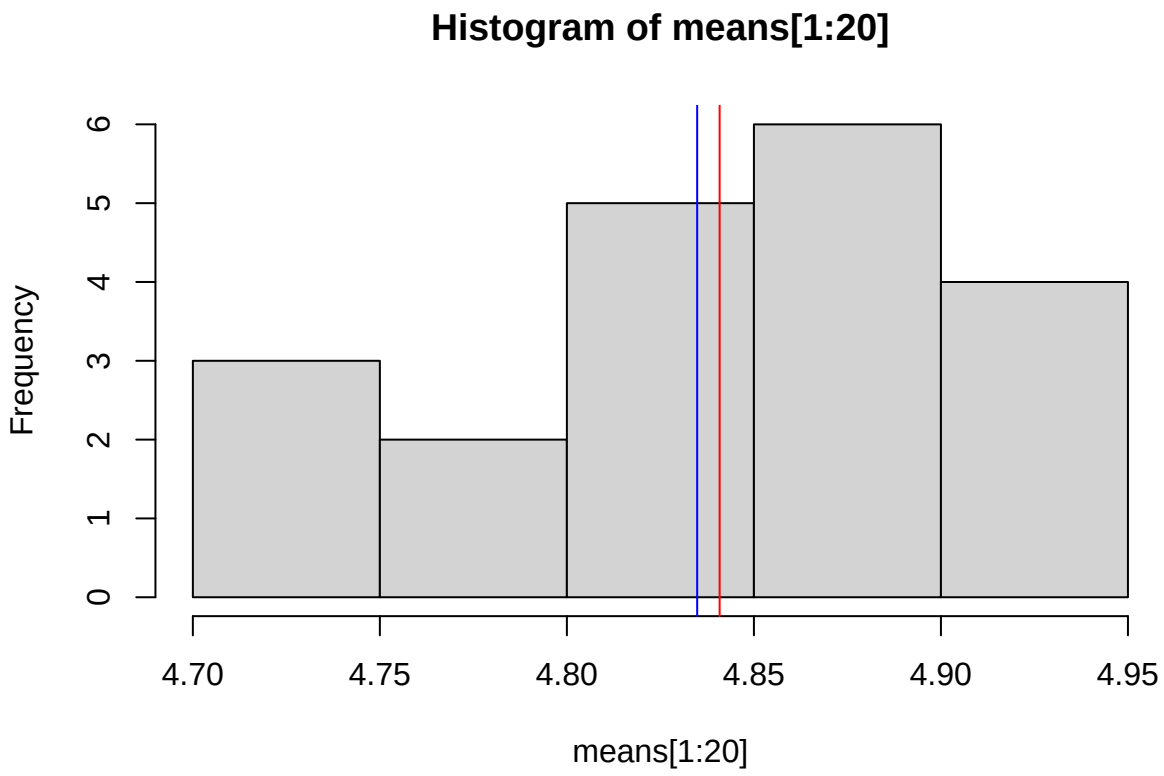
```
set.seed(12208877)
samples_2000 <- replicate(2000, sample(sample_orig, size=length(sample_orig), replace=TRUE))
means <- c()
for (i in 1:2000){
  means[i] <- mean(samples_2000[,i])
}
```

1.3.1 Compute the mean on the first 20 bootstrap means.

```
mean(means[1:20])
```

```
## [1] 4.834833
```

```
hist(means[1:20])
abline(v=mean(sample_orig), col='red')
abline(v=mean(means[1:20]), col='blue')
```

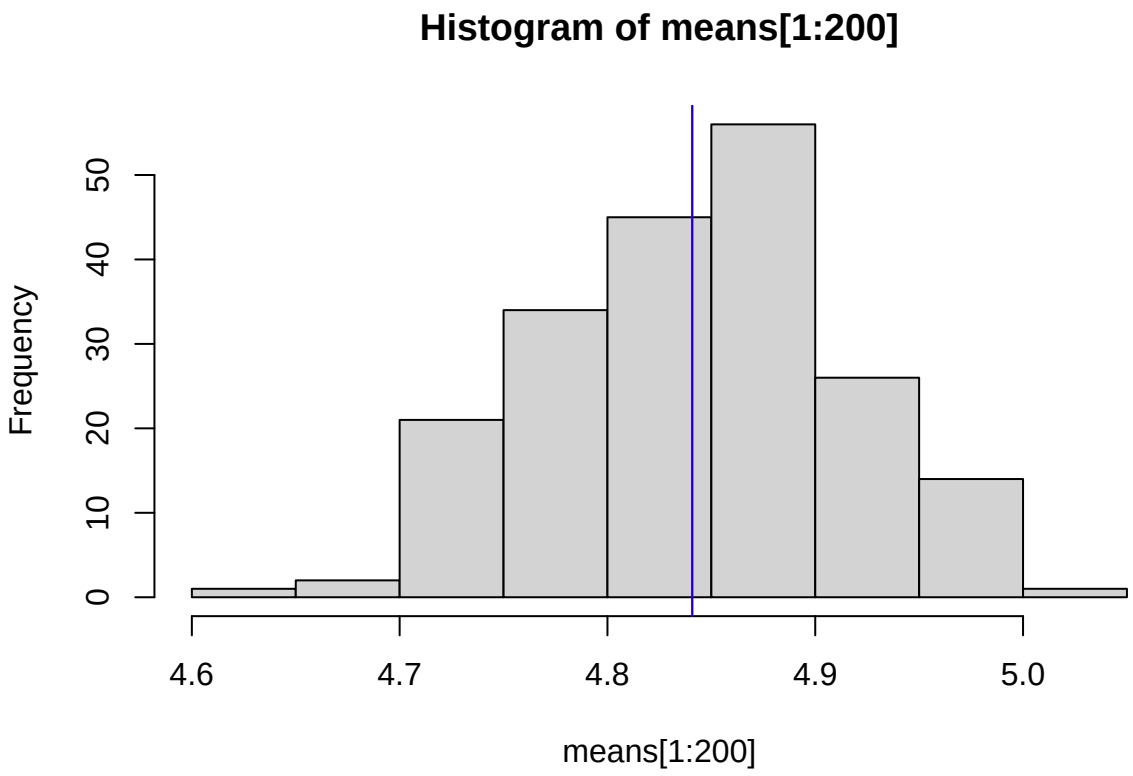


1.3.2 Compute the mean on the first 200 bootstrap means.

```
mean(means[1:200])
```

```
## [1] 4.840792
```

```
hist(means[1:200])
abline(v=mean(sample_orig), col='red')
abline(v=mean(means[1:200]), col='blue')
```

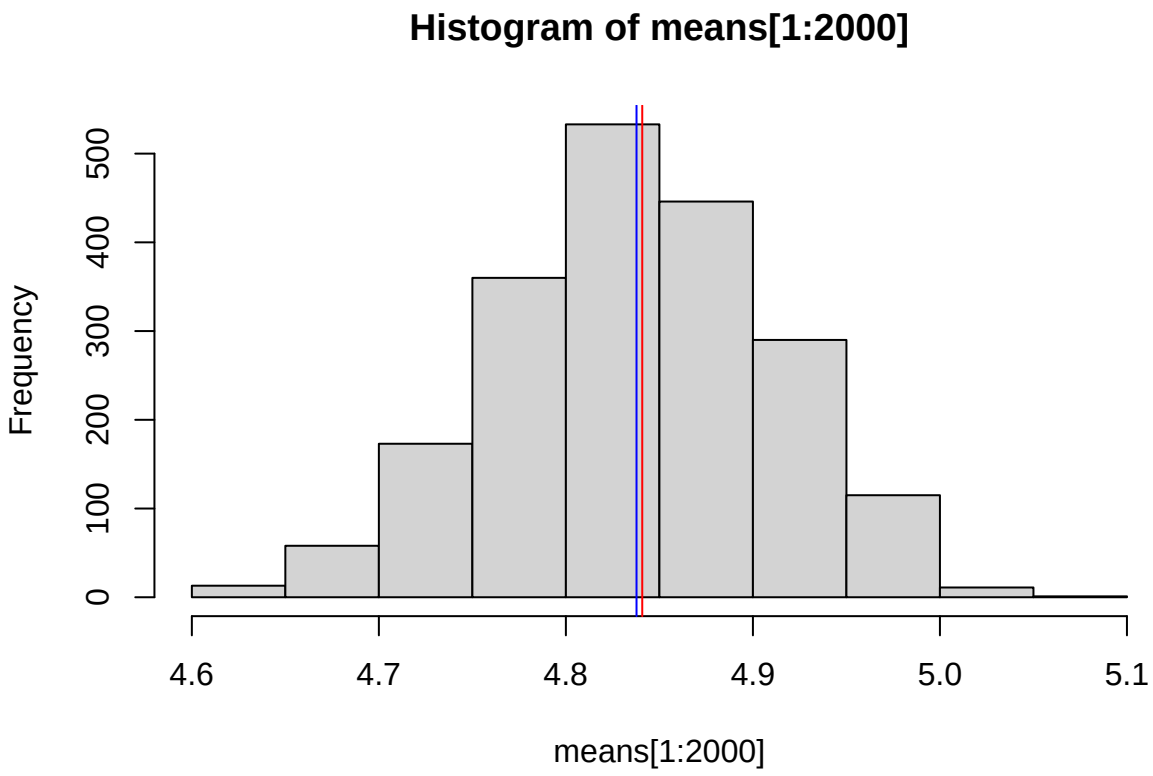


1.3.3 Compute the mean based on all 2000 bootstrap means.

```
mean(means[1:2000])
```

```
## [1] 4.837759
```

```
hist(means[1:2000])
abline(v=mean(sample_orig), col='red')
abline(v=mean(means[1:2000]), col='blue')
```



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

Visualization is shown above. For the first 20 samples the Central Limit Theorem did not yet fully worked, but with the further number of used samples, we can see it working since the means look more like a normal distribution.

1.3.5 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.)

```
q <- 0.025
qs1 <- c()
qs3 <- c()
for (i in 1:3){
  qs1[i] <- quantile(means[1:2*10^i], q)
  qs3[i] <- quantile(means[1:2*10^i], 1-q)
}
qsi = data.frame(
  count=c(20, 200, 2000),
  CILw=qs1,
  CIUp=qs3,
  interval=qs3-qs1
)
```

```
qsi
```

```
##   count     CILw     CIUp   interval
## 1    20 4.864729 4.917771 0.05304167
## 2   200 4.857917 4.905417 0.04750000
## 3  2000 4.875708 4.934292 0.05858333
```

```
true_ci <- t.test(sample_orig, conf.level = 0.95)$conf.int
true_ci
```

```
## [1] 4.674344 5.007323
## attr(,"conf.level")
## [1] 0.95
```

Comparing the resulting intervals we can see that the ones from bootstrap are quite smaller than the original one.

1.4 Create 2000 bootstrap samples and compute their medians.

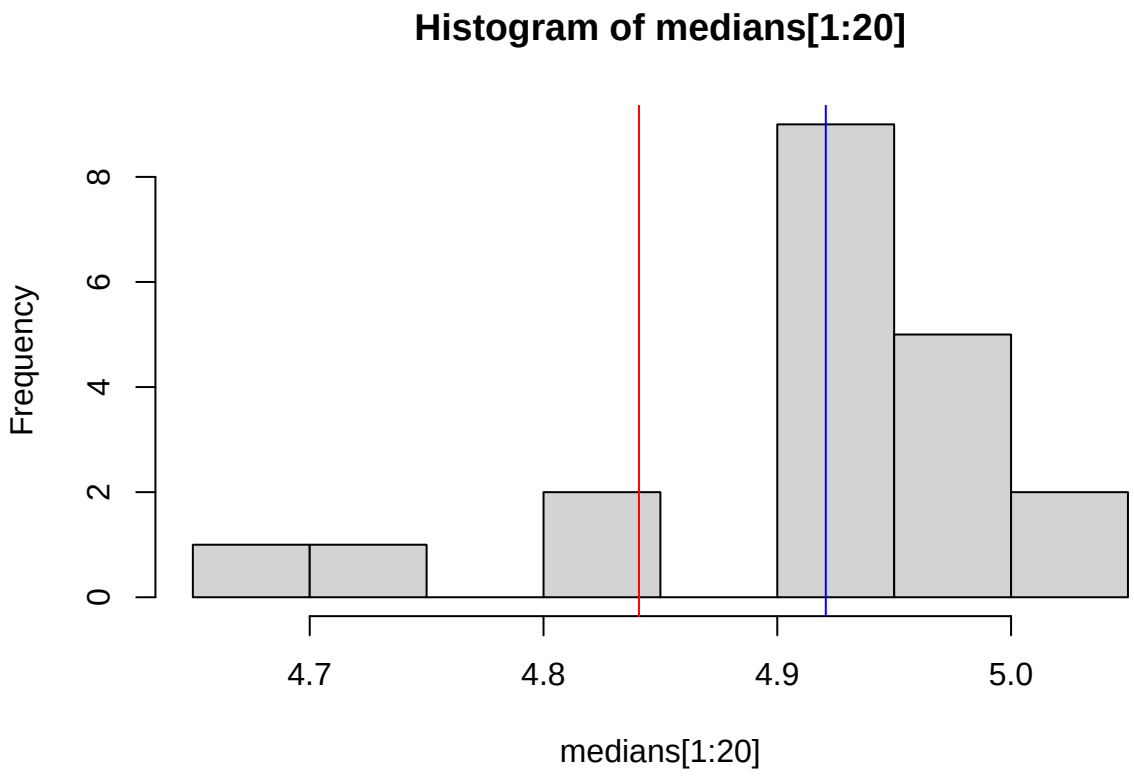
```
medians <- c()
for (i in 1:2000){
  medians[i] <- median(samples_2000[,i])
}
```

1.4.1 Compute the mean on the first 20 bootstrap medians.

```
mean(medians[1:20])
```

```
## [1] 4.92075
```

```
hist(medians[1:20])
abline(v=mean(sample_orig), col='red')
abline(v=mean(medians[1:20]), col='blue')
```

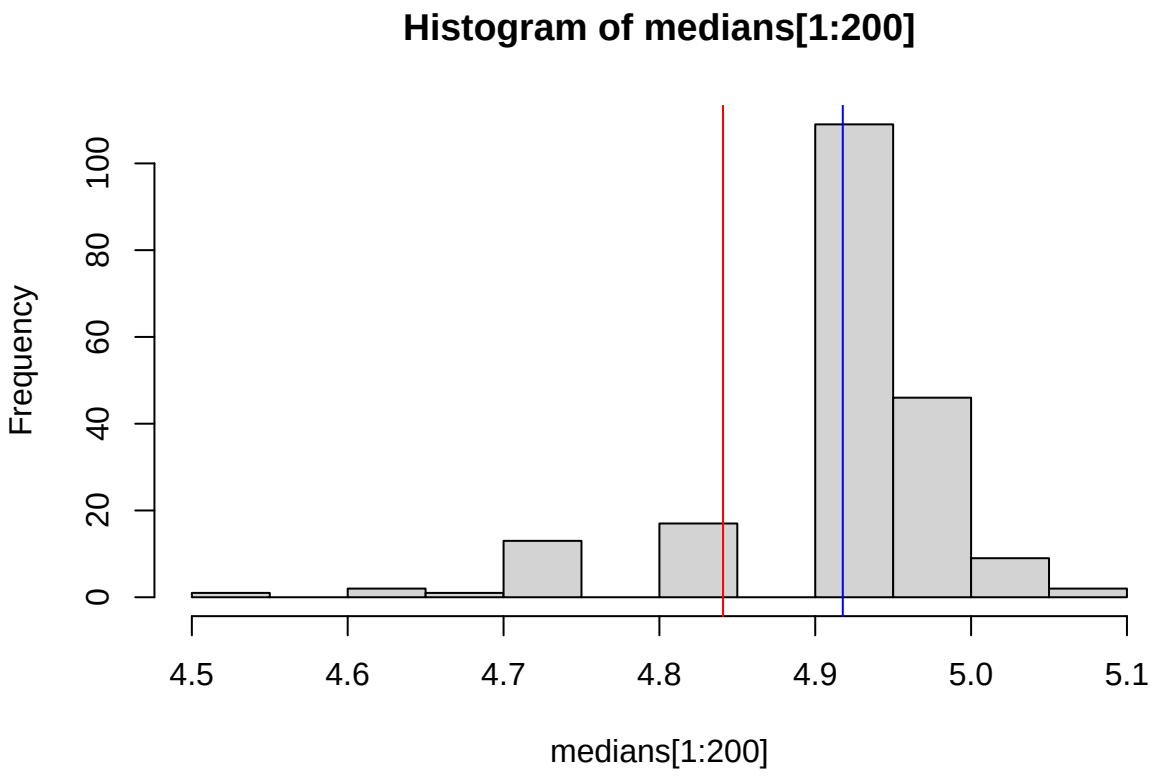


1.4.2 Compute the mean of the first 200 bootstrap medians.

```
mean(medians[1:200])
```

```
## [1] 4.917675
```

```
hist(medians[1:200])
abline(v=mean(sample_orig), col='red')
abline(v=mean(medians[1:200]), col='blue')
```

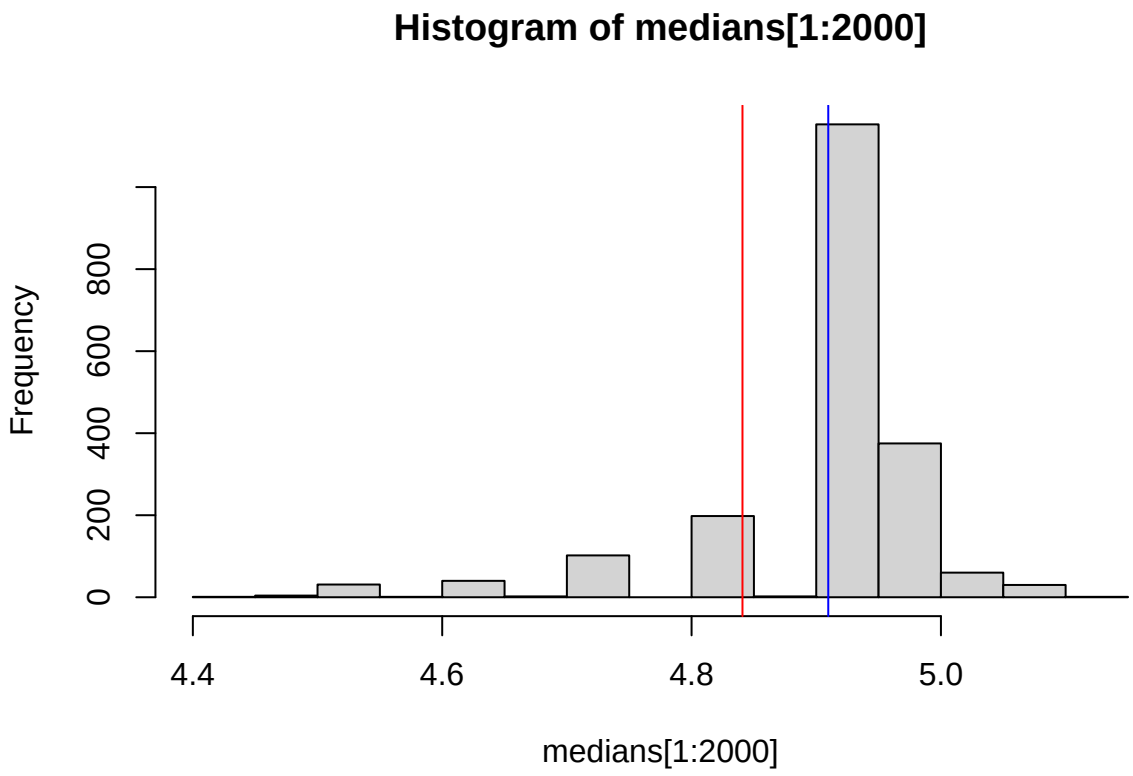


1.4.3 Compute the mean based on all 2000 bootstrap medians.

```
mean(medians[1:2000])
```

```
## [1] 4.909635
```

```
hist(medians[1:2000])
abline(v=mean(sample_orig), col='red')
abline(v=mean(medians[1:2000]), col='blue')
```



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

Visualized above.

1.4.5 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.

```
q <- 0.025
qs1 <- c()
qs3 <- c()
for (i in 1:3){
  qs1[i] <- quantile(medians[1:2*10^i], q)
  qs3[i] <- quantile(medians[1:2*10^i], 1-q)
}
qsi = data.frame(
  count=c(20, 200, 2000),
  CILw=qs1,
  CIUp=qs3,
  interval=qs3-qs1
)
qsi
```

```
##   count     CILw     CIUp interval
```

```
## 1    20 4.965875 4.999125  0.03325
## 2   200 4.945375 4.959625  0.01425
## 3  2000 4.941750 5.008250  0.06650
```

The interesting observation is the fact that the interval for medians of 200 samples is more narrow than for 20 and 2000.

Task 2

2.1 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.

```
set.seed(1234)
x.clean <- rnorm(1960)
x.cont <- runif(40, 4, 5)
x <- c(x.clean,x.cont)
set.seed(12208877)
```

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

```
mean_x <- c(mean(x), mean(x.clean))
median_x <- c(median(x), median(x.clean))
mean_t_x <- c(mean(x, trim=0.05), mean(x.clean, trim=0.05))

median_x
```

```
## [1]  0.0113797 -0.0172536
```

```
mean_x
```

```
## [1]  0.083955076 -0.005968976
```

```
mean_t_x
```

```
## [1]  0.036832939 -0.001462623
```

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

Standard errors:

```
n <- 2000
bs_x <- replicate(n, sample(x, replace=TRUE))
bs_x_clean <- replicate(n, sample(x.clean, replace=TRUE))

bs_x_means <- numeric(n)
for (i in 1:n) {bs_x_means[i] <- mean(bs_x[,i])}
bs_x_se <- sqrt(1 / (n - 1) * sum((bs_x_means - mean(bs_x_means))^2))

bs_x_clean_means <- numeric(n)
for (i in 1:n) {bs_x_clean_means[i] <- mean(bs_x_clean[,i])}
bs_x_clean_se <- sqrt(1 / (n - 1) * sum((bs_x_clean_means - mean(bs_x_clean_means))^2))
```

```
cat("standard error for x = ", bs_x_se, "\nstandard error for x.clean = ", bs_x_clean_se)
```

```
## standard error for x =  0.02557421
## standard error for x.clean =  0.02251347
```

95 percentile CI of all 3 estimators:

```
q <- 0.975
q_1 <- 1 - q
ci_mean_up = c(quantile(bs_x_means, q), quantile(bs_x_clean_means, q))
ci_mean_lw = c(quantile(bs_x_means, q_1), quantile(bs_x_clean_means, q_1))

bs_x_medians <- numeric(n)
bs_x_clean_medians <- numeric(n)
for (i in 1:n) {bs_x_medians[i] <- median(bs_x[,i])}
for (i in 1:n) {bs_x_clean_medians[i] <- median(bs_x_clean[,i])}
ci_med_up = c(quantile(bs_x_medians, q), quantile(bs_x_clean_medians, q))
ci_med_lw = c(quantile(bs_x_medians, q_1), quantile(bs_x_clean_medians, q_1))

bs_t_x_means <- numeric(n)
bs_t_x_clean_means <- numeric(n)
for (i in 1:n) {bs_t_x_means[i] <- mean(bs_x[,i], trim=0.05)}
for (i in 1:n) {bs_t_x_clean_means[i] <- mean(bs_x_clean[,i], trim=0.05)}
ci_mean_t_up = c(quantile(bs_t_x_means, q), quantile(bs_t_x_clean_means, q))
ci_mean_t_lw = c(quantile(bs_t_x_means, q_1), quantile(bs_t_x_clean_means, q_1))

table_ci <- data.frame(
  data=c('x', 'x.clean'),
  CIMeanUp=ci_mean_up,
  CIMeanLw=ci_mean_lw,
  CIMedUp=ci_med_up,
  CIMedLw=ci_med_lw,
  CIMeanTUp=ci_mean_t_up,
  CIMeanTLw=ci_mean_t_lw
)
table_ci
```

```
##      data   CIMeanUp    CIMeanLw    CIMedUp     CIMedLw  CIMeanTUp    CIMeanTLw
## 1       x 0.13312156  0.03258194 0.05939342 -0.04507868 0.08015885 -0.009237934
## 2 x.clean 0.03710364 -0.05071325 0.03899314 -0.06659420 0.04168319 -0.047072145
```

2.4 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 trimmed mean. When estimating the scale of the of the data in the "robust" case use the mad.

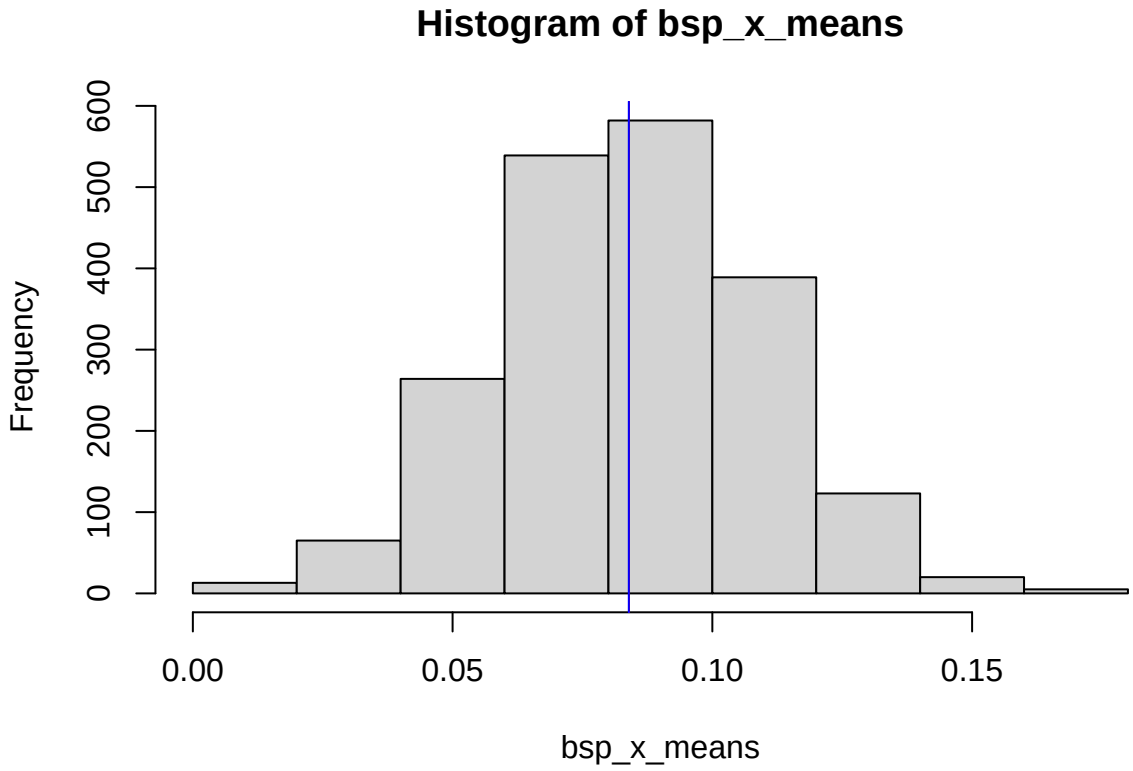
```
sd_x <- c(sd(x), sd(x.clean))
set.seed(12208877)
mad_x <- c(mad(x), mad(x.clean))

bs_param_x <- replicate(n, rnorm(length(x), mean=mean_x[1], sd=sd_x[1]))
bsp_x_rob <-  replicate(n, rnorm(length(x), median_x[1], mad_x[1]))
bs_param_x_clean <- replicate(n, rnorm(length(x.clean), mean=mean_x[2], sd=sd_x[2]))
bsp_x_clean_rob <-  replicate(n, rnorm(length(x.clean), median_x[2], mad_x[2]))
```

```r
bsp_x_means <- numeric(n)
for (i in 1:n) {bsp_x_means[i] <- mean(bs_param_x[,i])}
bsp_x_se <- sqrt(1 / (n - 1) * sum((bsp_x_means - mean(bsp_x_means))^2))

bsp_x_clean_means <- numeric(n)
for (i in 1:n) {bsp_x_clean_means[i] <- mean(bs_param_x_clean[,i])}
bsp_x_clean_se <- sqrt(1 / (n - 1) * sum((bsp_x_clean_means - mean(bsp_x_clean_means))^2))

bias_x <- mean(bsp_x_means) - mean_x[1]
bias_x_clean <- mean(bsp_x_clean_means) - mean_x[2]

bspt_x_means <- numeric(n)
for (i in 1:n) {bspt_x_means[i] <- mean(bsp_x_rob[,i], trim=0.05)}
bspt_x_se <- sqrt(1 / (n - 1) * sum((bspt_x_means - mean(bspt_x_means, trim=0.05))^2))

bspt_x_clean_means <- numeric(n)
for (i in 1:n) {bspt_x_clean_means[i] <- mean(bsp_x_clean_rob[,i], trim=0.05)}
bspt_x_clean_se <- sqrt(1 / (n - 1) * sum((bspt_x_clean_means - mean(bspt_x_clean_means, trim=0.05))^2))

biast_x <- mean(bspt_x_means) - mean_t_x[1]
biast_x_clean <- mean(bspt_x_clean_means) - mean_t_x[2]



q <- 0.975
q_1 <- 1 - q
cip_mean_up = c(quantile(bsp_x_means, q), quantile(bsp_x_clean_means, q))
cip_mean_lw = c(quantile(bsp_x_means, q_1), quantile(bsp_x_clean_means, q_1))

bsp_t_x_means <- numeric(n)
bsp_t_x_clean_means <- numeric(n)
for (i in 1:n) {bsp_t_x_means[i] <- mean(bsp_x_rob[,i], trim=0.05)}
for (i in 1:n) {bsp_t_x_clean_means[i] <- mean(bsp_x_clean_rob[,i], trim=0.05)}
cip_mean_t_up = c(quantile(bsp_t_x_means, q), quantile(bsp_t_x_clean_means, q))
cip_mean_t_lw = c(quantile(bsp_t_x_means, q_1), quantile(bsp_t_x_clean_means, q_1))


table <- data.frame(
  data=c('x', 'x.clean'),
  bias_mean=c(bias_x, bias_x_clean),
  bias_trimmed_mean=c(biast_x, biast_x_clean),
  SE_mean=c(bsp_x_se, bsp_x_clean_se),
  SE_trimmed_mean=c(bspt_x_se, bspt_x_clean_se),
  CIMeansUp=cip_mean_up,
  CIMeansLw=cip_mean_lw,
  CITrimMeansUp=cip_mean_t_up,
  CITrimMeansLw=cip_mean_t_lw,
  bias_corr_mean=c(mean(bsp_x_means) - bias_x, mean(bsp_x_clean_means) - bias_x_clean),
  bias_corr_trimmed_mean=c(mean(bspt_x_means) - biast_x, mean(bspt_x_clean_means) - biast_x_clean)
)
table
```

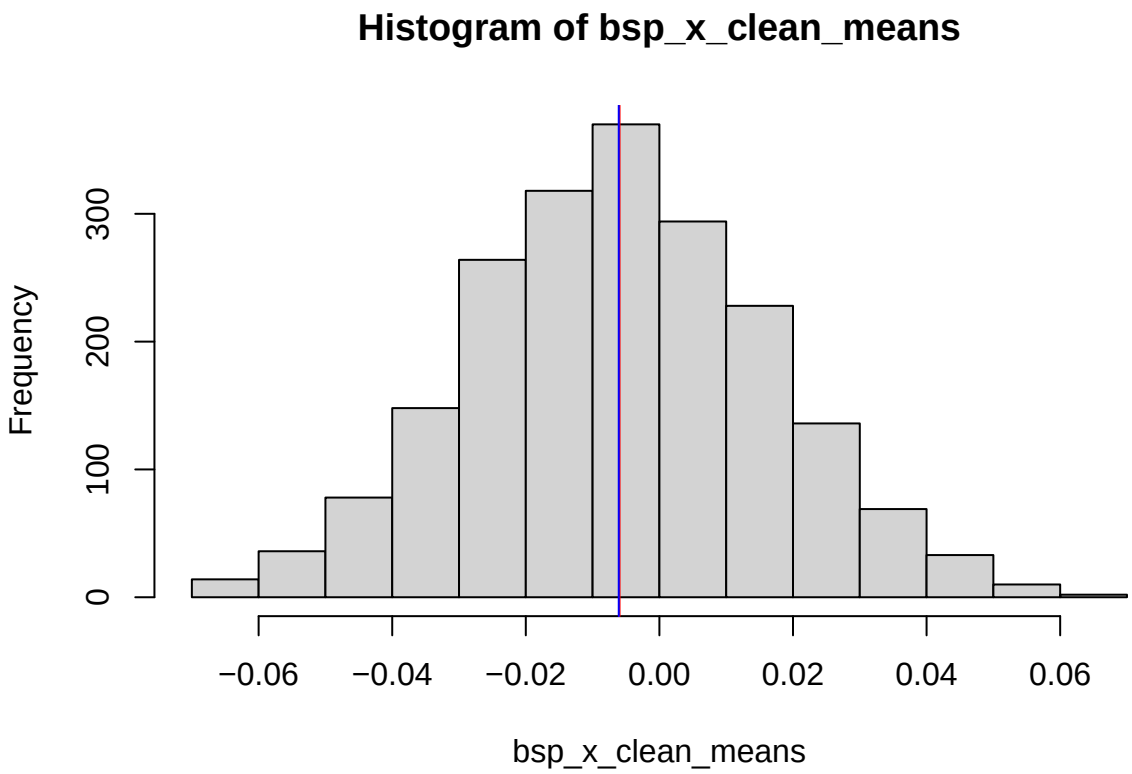
```
##      data     bias_mean bias_trimmed_mean    SE_mean SE_trimmed_mean  CIMeansUp
## 1       x -3.204124e-05       -0.02483537 0.02516422      0.02163054 0.13364663
## 2 x.clean -1.171351e-04       -0.01538537 0.02232124      0.02150093 0.03681794
##     CIMeansLw CITrimMeansUp CITrimMeansLw bias_corr_mean bias_corr_trimmed_mean
## 1  0.03489823    0.05366618   -0.03071093    0.083955076            0.036832939
## 2 -0.04995125    0.02370884   -0.05881464   -0.005968976           -0.001462623
```

We can observe that confidence intervals are quite smaller for the parametric case comparing to non-parametric one. Additionally, difference for the trimmed means comparing between `x` and `x.clean`, is smaller than for the normal mean.

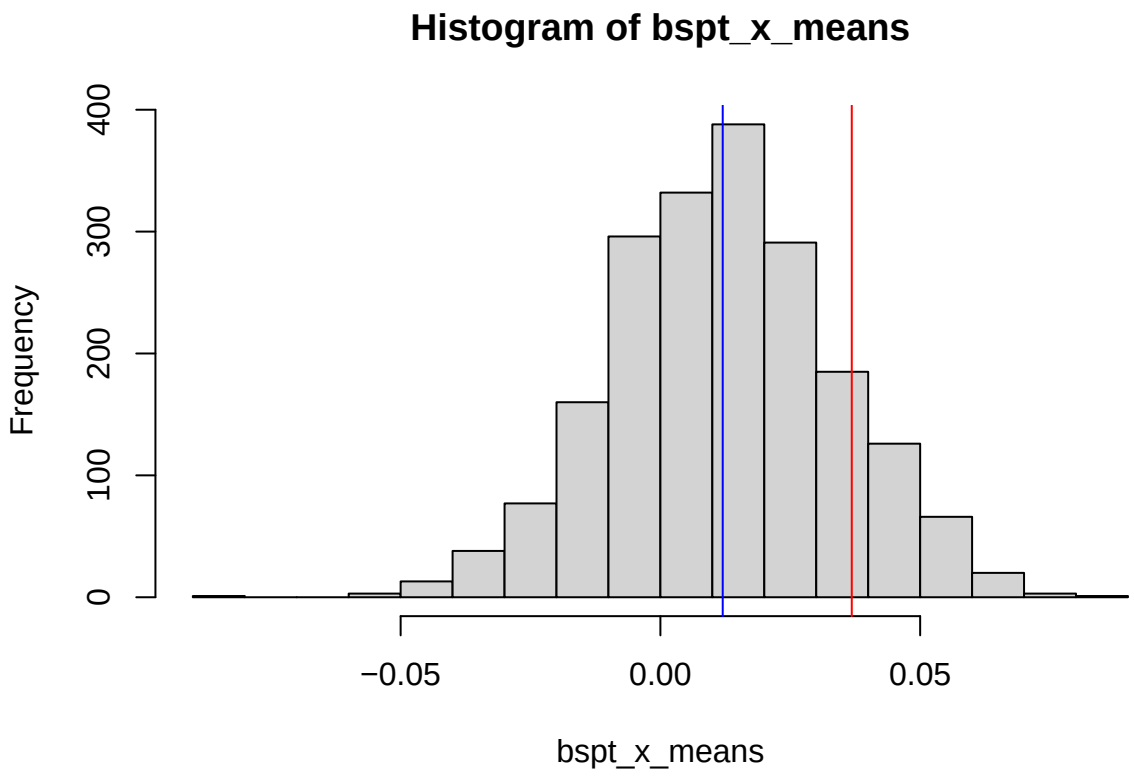
```
hist(bsp_x_means)
abline(v=mean_x[1], col='red')
abline(v=mean(bsp_x_means), col='blue')
```



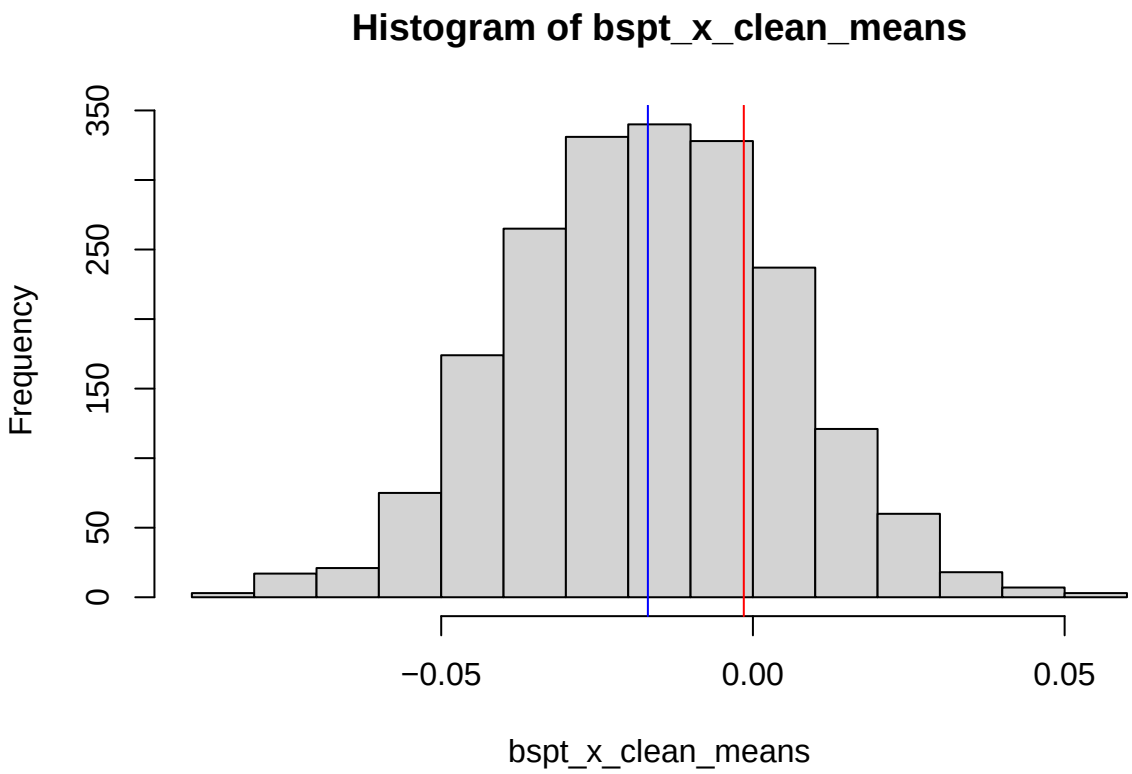

```
hist(bsp_x_clean_means)
abline(v=mean_x[2], col='red')
abline(v=mean(bsp_x_clean_means), col='blue')
```




```
hist(bspt_x_means)
abline(v=mean_t_x[1], col='red')
abline(v=mean(bspt_x_means), col='blue')
```




```
hist(bspt_x_clean_means)
abline(v=mean_t_x[2], col='red')
abline(v=mean(bspt_x_clean_means), col='blue')
```



The visualization of sampled mean with the lines for original mean and obtained mean can be seen above.

Task 3. Based on the above tasks and your lecture materials, explain the methodology of bootstrapping for the construction of conifdence intervals and parametric or non-parametric tests.

Non-parametric bootstrapping is a method for creating a new sample of data from the original one using replacement. It follows a discrete distribution, and because all the elements of original data set are equally weighted, in case there are any outliers present, they will strongly affect the resulting samples and their further analysis. In contrast to the non-parametric one, parametric bootstrapping accounts for the specific features of the distribution of the given population. The parameters are estimated based on the given distribution, hence they are dependent on the empirical sample. For the construction of confidence intervals, we can either construct them based on the quantiles of the normal distribution or based on the estimates from bootstrap samples. Both options of bootstrapping allows to estimate the parameters of distribution, namely mean, median, trimmed mean that we did in the current exercise. For this, a big number of samples can be created to give the closest to the reality results and then, its statistics can be calculated, by taking the mean of samples medians for example. Subsequently, the confidence intervals can be calculated based on the obtained results.