Introduction to Simulation with Variance Estimation

Exercise 4

Bosse Behrens, st.id 12347333

2024W

Task 1

part 1

We will draw samples of size 12 from the original sample of size 12. Since we consider two samples with the elements but a different ordering different samples overall, we have n^n possible samples.

```
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)
n <- length(data)
combinations <- n^n
names(combinations) <- c("possible combinations")
print(combinations)
```

```
## possible combinations
##            8.9161e+12
```

part 2

Calculating mean and median of our provided data.

```
cat("mean", mean(data), "median", median(data))
```

```
## mean 4.840833 median 4.935
```

part 3.1-3.3

We create 2000 bootstrap samples by sampling from the original data 2000 times and calculate the mean of each sample as well the mean based on the first 20, 200, 2000 sample means.

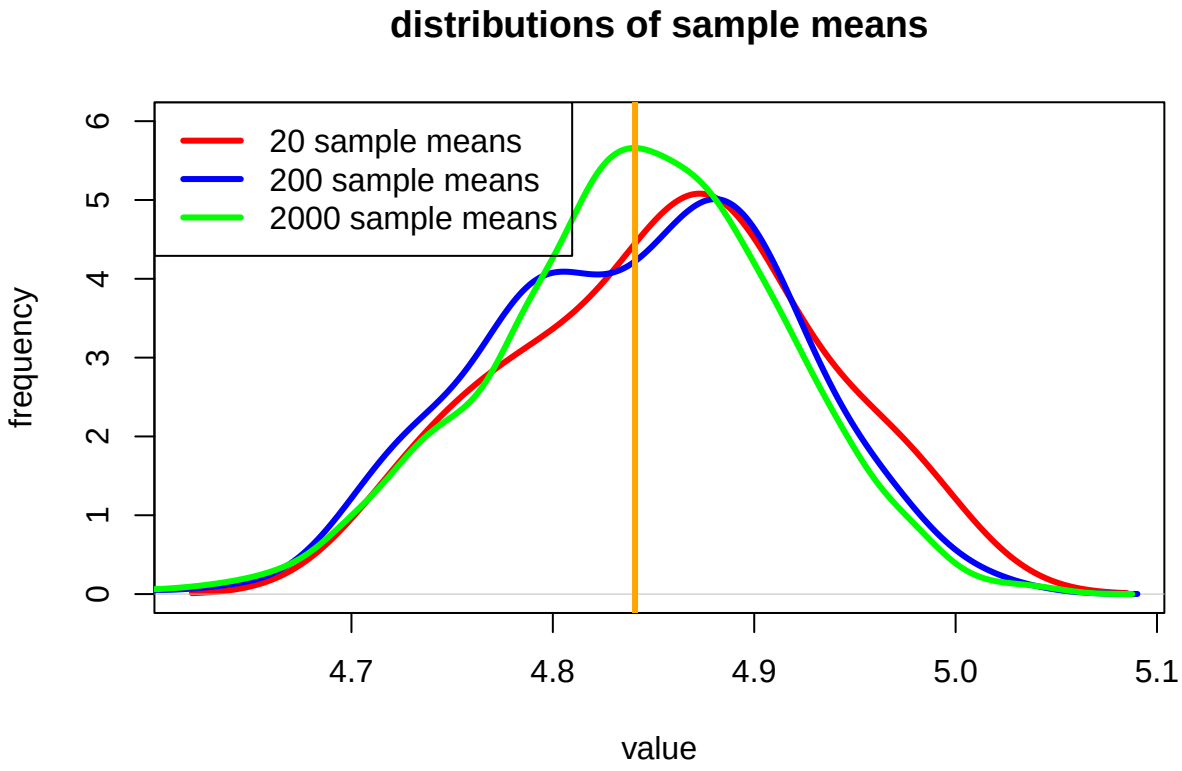
```
set.seed(12347333)
boot_samples <- list()
boot_samples <- lapply(1:2000, function(i) sample(data, 12, replace = TRUE))
boot_means <- sapply(boot_samples, mean)
means_20 <- mean(boot_means[1:20])
means_200 <- mean(boot_means[1:200])
means_2000 <- mean(boot_means[1:2000])
means <- c(means_20, means_200, means_2000)
names <- c("mean of 20 sample means", "mean of 200 sample means", "mean of 2000 sample means")
results <- data.frame(names, means)
print(results)
```

```
##                        names    means
## 1   mean of 20 sample means 4.856292
## 2  mean of 200 sample means 4.841138
```

```
## 3 mean of 2000 sample means 4.840721
```

part 3.4

```
dens_20 <- density(boot_means[1:20])
dens_200 <- density(boot_means[1:200])
dens_2000 <- density(boot_means[1:2000])
plot(dens_20, type = "l", col = "red", ylim = range(c(0, 6)), lwd = 3, ylab = "frequency", xlab ="value
lines(dens_200, col = "blue", lwd = 3)
lines(dens_2000, col = "green", lwd = 3)
abline(v = mean(data), col="orange", lwd = 3) #original sample mean
legend("topleft", legend = c("20 sample means", "200 sample means", "2000 sample means"),
       col = c("red", "blue", "green"), lty = 1, lwd = 3)
```



We can observe that for the first 20 sample means the mode is clearly right of the original sample mean. The distribution for the first 200 sample means looks slightly more centered towards the originals ample mean but still similar to 20. The distribution of the 2000 sample means seems very centered around the original sample mean and the visualization of its distribution is very similar to a normal distribution with the original sample mean as the mean. Therefore it can be concluded that the central limit theorem kicks in since for a larger number of bootstrap samples the distribution of the means follows more a normal distribution.

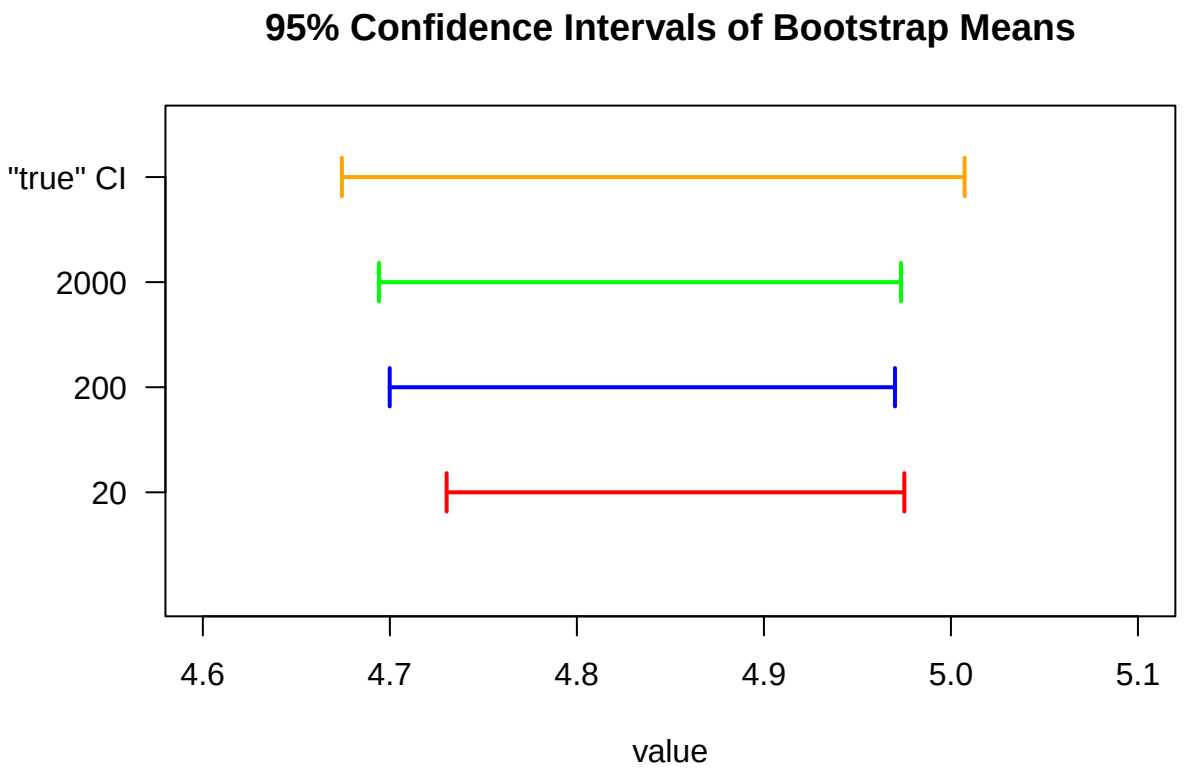
part 3.5

```
q_20 <- quantile(boot_means[1:20], c(0.025,0.975))
q_200 <- quantile(boot_means[1:200],c(0.025,0.975))
q_2000 <- quantile(boot_means[1:2000], c(0.025,0.975))
q_sample <- t.test(data, conf.level = 0.95)
```

```r
q_names <- c("20 samples", "200 samples", "2000 samples", "\"true\" CI")
q_low <- c(q_20[1], q_200[1], q_2000[1], q_sample$conf.int[1])
q_up <- c(q_20[2], q_200[2], q_2000[2], q_sample$conf.int[2])

results_q <- data.frame(q_names, q_low, q_up)
print(results_q) # results in table
```

```
##        q_names    q_low     q_up
## 1   20 samples 4.730354 4.975083
## 2  200 samples 4.699917 4.970083
## 3 2000 samples 4.694146 4.973333
## 4    "true" CI 4.674344 5.007323
```

```r
plot(1:4, type = "n", xlim = range(4.6, 5.1), #plotting results
     ylim = c(0, 4.5), yaxt = "n", xaxt = "n", ylab = "", xlab = "value",
     main = "95% Confidence Intervals of Bootstrap Means")
arrows(q_20[1], 1, q_20[2], 1, angle = 90, code = 3, length = 0.1, col = "red", lwd = 2)
arrows(q_200[1], 2, q_200[2], 2, angle = 90, code = 3, length = 0.1, col = "blue", lwd = 2)
arrows(q_2000[1], 3, q_2000[2], 3, angle = 90, code = 3, length = 0.1, col = "green", lwd = 2)
arrows(q_sample$conf.int[1], 4, q_sample$conf.int[2], 4, angle = 90, code = 3, length = 0.1, col = "ora

axis(2, at = 1:4, labels = c("20", "200", "2000", "\"true\" CI"), las = 2)
axis(1)
```



We can observe that with fewer samples the confidence interval is also smaller. With an increasing number of bootstrap samples, the confidence intervals for the mean are also getting wider. The “true” confidence interval has the largest range and the with an increasing number of bootstrap samples the confidence intervals

are nearing it more and more.

part 4.1-4.3

We calculate the mean of the medians of the first 20/200/2000 samples similar to the means in part 3.

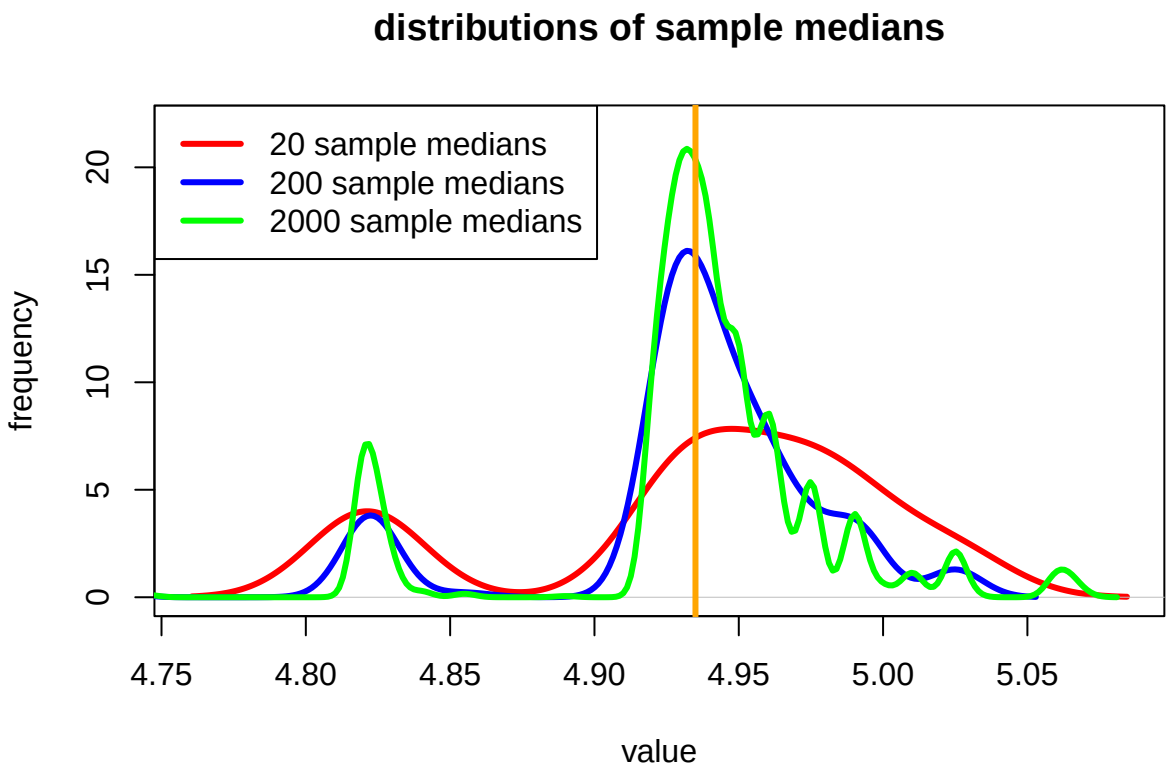
```
boot_medians <- sapply(boot_samples, median)
medians_20 <- mean(boot_medians[1:20])
medians_200 <- mean(boot_medians[1:200])
medians_2000 <- mean(boot_medians[1:2000])
medians <- c(medians_20, medians_200, medians_2000)
names_med <- c("mean of 20 sample medians", "mean of 200 sample medians", "mean of 2000 sample medians"
results <- data.frame(names, medians)
print(results) # results in table
```

```
##                       names medians
## 1   mean of 20 sample means 4.93650
## 2  mean of 200 sample means 4.91260
## 3 mean of 2000 sample means 4.91225
```

part 4.4

Plotting the distribution of the medians.

```
dens_20m <- density(boot_medians[1:20])
dens_200m <- density(boot_medians[1:200])
dens_2000m <- density(boot_medians[1:2000])
plot(dens_20m, type = "l", col = "red", ylim = range(c(0, 22)), lwd = 3, ylab = "frequency", xlab ="valu
lines(dens_200m, col = "blue", lwd = 3)
lines(dens_2000m, col = "green", lwd = 3)
abline(v = median(data), col="orange", lwd = 3)
legend("topleft", legend = c("20 sample medians", "200 sample medians", "2000 sample medians"),
       col = c("red", "blue", "green"), lty = 1, lwd = 3)
```



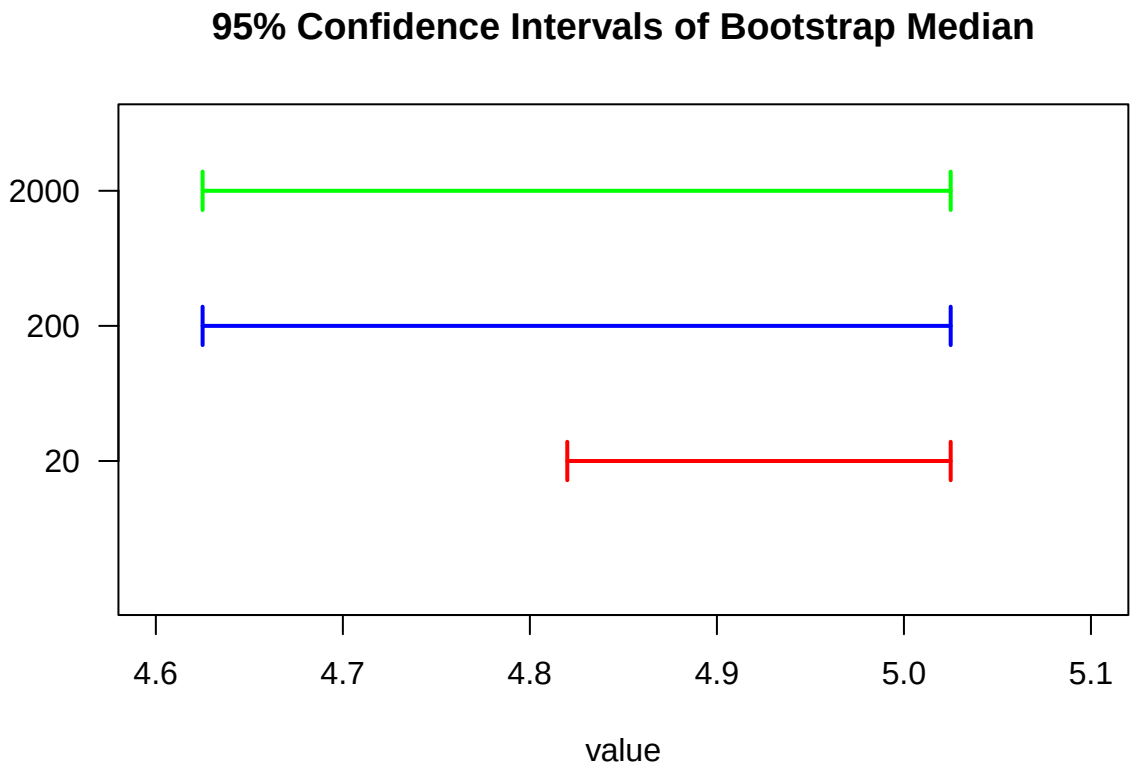
As we can observe, we can not recognize any similarity to a normal distribution for the distributions of the sample mean. There seem to be multiple local maxima and overall not much likeness to a normal distribution. Therefore we cannot observe the central limit theorem. It might work with an even larger sample size but that is only an assumption. The central limit theorem specifically applies to the sample mean so for other values like the median which also has a discrete range things get more complicated. Due to its sensitivity to outliers the mean also smoothes out way more quickly than a robust indicator like the median.

part 4.5

```
q_20m <- quantile(boot_medians[1:20], c(0.025,0.975))
q_200m <- quantile(boot_medians[1:200],c(0.025,0.975))
q_2000m <- quantile(boot_medians[1:2000], c(0.025,0.975))

q_names <- c("20 samples", "200 samples", "2000 samples")
q_low <- c(q_20m[1], q_200m[1], q_2000m[1])
q_up <- c(q_20m[2], q_200m[2], q_2000m[2])

results_q <- data.frame(q_names, q_low, q_up)
print(results_q)
```

```
##        q_names q_low  q_up
## 1   20 samples 4.820 5.025
## 2  200 samples 4.625 5.025
## 3 2000 samples 4.625 5.025
```

```r
plot(1:3, type = "n", xlim = range(4.6, 5.1),
     ylim = c(0, 3.5), yaxt = "n", xaxt = "n", ylab = "", xlab = "value",
     main = "95% Confidence Intervals of Bootstrap Median")
arrows(q_20m[1], 1, q_20m[2], 1, angle = 90, code = 3, length = 0.1, col = "red", lwd = 2)
arrows(q_200m[1], 2, q_200m[2], 2, angle = 90, code = 3, length = 0.1, col = "blue", lwd = 2)
arrows(q_2000m[1], 3, q_2000m[2], 3, angle = 90, code = 3, length = 0.1, col = "green", lwd = 2)

axis(2, at = 1:3, labels = c("20", "200", "2000"), las = 2)
axis(1)
```



For the first 20 samples the confidence interval seems quite small while the other two are the same and also about double that range. Something like this is to be expected since the median has a range of discrete values.

Task 2

part 1

We set the specified seed and create the data x, x.clean and x.cont.

```r
set.seed(1234)
x.clean <- rnorm(1960)
x.cont <- runif(40, 4, 5)
x <- c(x.clean, x.cont)
set.seed(12347333)
```

part 4.2

We calculate the estimates for median, mean and the trimmed mean for both x and x.clean.

```
cat("x median", median(x), "x mean", mean(x), "x trimmed mean", mean(x, trim = 0.05))
```

```
## x median 0.0113797 x mean 0.08395508 x trimmed mean 0.03683294
```

```
cat("x.clean median", median(x.clean), "x.clean mean", mean(x.clean), "x.clean trimmed mean", mean(x.cl
```

```
## x.clean median -0.0172536 x.clean mean -0.005968976 x.clean trimmed mean -0.001462623
```

We implement nonparametric bootstrap by creating 1000 new samples from the original data and applying median, mean and trimmed mean. First we write the function that does just that for some input data. It returns a list with the three vectors containing the medians, means and trimmed means respectively.

```
boot_func <- function(y){
  n <- length(y)
  boot_median <- replicate(1000, median(sample(y, n, replace = TRUE)))
  boot_mean <- replicate(1000, mean(sample(y, n, replace = TRUE)))
  boot_mean_trim <- replicate(1000, mean(sample(y, n, replace = TRUE), trim = 0.05))
  return(list(boot_median, boot_mean, boot_mean_trim))
}
```

Now we execute the function on x and x.clean and calculate the standard error and 95% confidence interval using the result vectors. Then we put the results in a data frame and show the table.

```
sd_x <- sapply(boot_func(x), sd)
ci_x_clean <- sapply(boot_func(x), function(s){quantile(s, c(0.025,0.975))})
c1 <- ci_x_clean[1,]
c2 <- ci_x_clean[2,]
estimates <- c(median(x), mean(x), mean(x, trim = 0.05))
results_npb <- data.frame(estimates, sd_x, c1, c2)#
rownames(results_npb) <- c("median", "mean", "trimmed mean")
colnames(results_npb) <- c("estimate", "standard_error", "lower ci", "upper ci")
print("Results for x:")
```

```
## [1] "Results for x:"
```

```
print(results_npb) # showing results in a table
```

```
##                estimate standard_error     lower ci   upper ci
## median       0.01137970     0.02806653 -0.040206937 0.05872684
## mean         0.08395508     0.02719658  0.031844411 0.13264887
## trimmed mean 0.03683294     0.02409846 -0.007910807 0.08144425
```

```
sd_x_clean <- sapply(boot_func(x.clean), sd)
ci_x_clean <- sapply(boot_func(x.clean), function(s){quantile(s, c(0.025,0.975))})
c1_clean <- ci_x_clean[1,]
c2_clean <- ci_x_clean[2,]
estimates_clean <- c(median(x.clean), mean(x.clean), mean(x.clean, trim = 0.05))
results_npb_clean <- data.frame(estimates_clean, sd_x_clean, c1_clean, c2_clean)
rownames(results_npb_clean) <- c("median", "mean", "trimmed mean")
colnames(results_npb_clean) <- c("estimate", "standard_error", "lower ci", "upper ci")
print("Results for x.clean:")
```

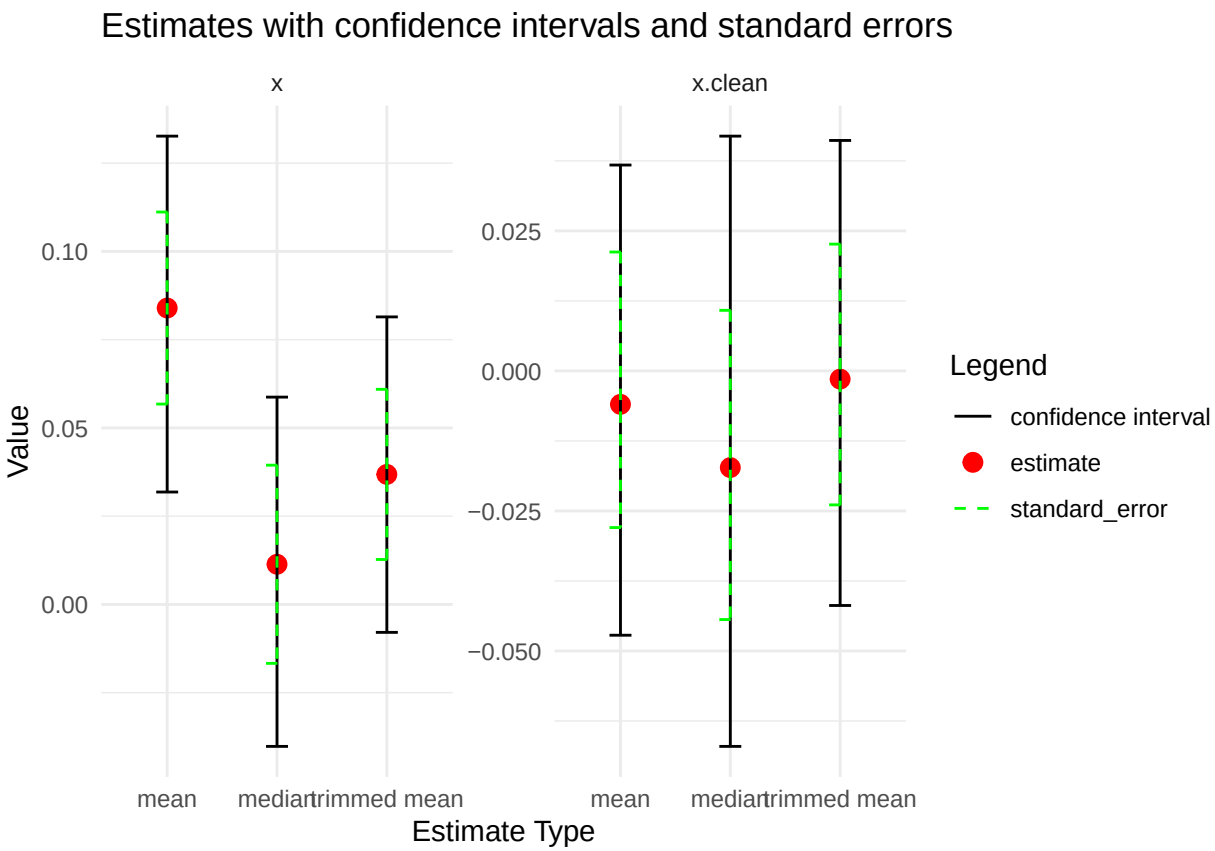
```
## [1] "Results for x.clean:"
```

```
print(results_npb_clean) # sholwing results in a table
```

```
##                  estimate standard_error    lower ci   upper ci
## median       -0.017253605     0.02713706 -0.06703859 0.04191743
## mean         -0.005968976     0.02199991 -0.04719300 0.03675698
## trimmed mean -0.001462623     0.02245399 -0.04188423 0.04114877
```

Now we want to visualize the data. To do so we use ggplot2 to plot the confidence intervals, estimates and the estimates +/− the standard error.

```
library(ggplot2)
results_npb$estimate_type <- rownames(results_npb) # add column for correct mapping
results_npb_clean$estimate_type <- rownames(results_npb_clean)
results_npb$data_source <- "x"     # add column  to distinguish data
results_npb_clean$data_source <- "x.clean"
results_combined <- rbind(results_npb, results_npb_clean) # combine dataframe

ggplot(results_combined, aes(x = estimate_type)) +
  geom_point(aes(y = estimate, color = "estimate"), size = 3) +
  geom_errorbar(aes(ymin = `lower ci`, ymax = `upper ci`, color = "confidence interval"),
                width = 0.2) +
  geom_errorbar(aes(ymin = estimate - standard_error, ymax = estimate + sd_x, color = "standard_error")
                width = 0.2, linetype = "dashed") +
  scale_color_manual(values = c("estimate" = "red", "confidence interval" = "black", "standard_error" =
  labs(title = "Estimates with confidence intervals and standard errors",
       x = "Estimate Type",
       y = "Value",
       color = "Legend") +
  theme_minimal() +
  facet_wrap(~ data_source, scales = "free")
```



part 4

We now want to use parametric bootstrap to calculate bias, standard error, the 95 percentile CI and the bias corrected estimate for the mean and the trimmed mean based on x and x.clean. In parametric bootstrapping we have to make an assumption about the underlying distribution of the data. First we make the assumption of standard normal distributed data (as x.clean actually is). This means we use the mean of the data as the location parameter and the standard deviation of the data as the scale parameter for drawing samples. Therefore we write a funciton that draws 1000 samples of the same length as the input data from that distribution. Based on this the aforementioned values for both the mean and the trimmed mean are calculated. The results are returned as a data frame.

```
boot_param <- function(y){
  n <- length(y)
  sde <- sd(y) # setting assumed parameters
  location <- mean(y)

  means <- numeric(1000)
  trimmed_means <- numeric(1000)
  medians <- numeric(1000)
  for (i in 1:1000) {
    samples <- rnorm(n, location, sde)
    means[i] <- mean(samples)
    trimmed_means[i] <- mean(samples, trim = 0.05)
    medians[i] <- median(samples)
  }
```

```
  est <- c(mean(y), mean(y, trim = 0.05), median(y))
  bias <- c(mean(means)-mean(y),
            mean(trimmed_means)-mean(y, trim = 0.05),
            median(medians)-median(y))
  se <- c(sd(means), sd(trimmed_means), sd(medians))
  ci_mean <- quantile(means, c(0.025, 0.975))
  ci_trim <- quantile(trimmed_means, c(0.025, 0.975))
  ci_median <- quantile(medians, c(0.025, 0.975))
  ci_low <- c(ci_mean[1],ci_trim[1], ci_median[1])
  ci_up <- c(ci_mean[2],ci_trim[2], ci_median[2])
  bias_cor <- c(mean(y)-bias[1],
                mean(y, trim = 0.05)-bias[2],
                median(y)-bias[3])

  results <- data.frame(estimate = est,
                        bias = bias,
                        se = se,
                        ci_lower = ci_low,
                        ci_upper = ci_up,
                        bias_corr_estimate = bias_cor)
  rownames(results) <- c("mean", "tr_mean", "median")
  return(results)

}
```

Showing the resulting table.

```
results_pb <- boot_param(x)
results_pb_clean <- boot_param(x.clean)
print("results for x")
```

```
## [1] "results for x"
```

```
print(results_pb)
```

```
##           estimate        bias         se   ci_lower  ci_upper
## mean    0.08395508 0.001255616 0.02614410 0.03550537 0.1387855
## tr_mean 0.03683294 0.048344060 0.02630019 0.03552177 0.1383636
## median  0.01137970 0.074598792 0.03242702 0.02333204 0.1494567
##         bias_corr_estimate
## mean            0.08269946
## tr_mean        -0.01151112
## median         -0.06321909
```

```
print("results for x.clean")
```

```
## [1] "results for x.clean"
```

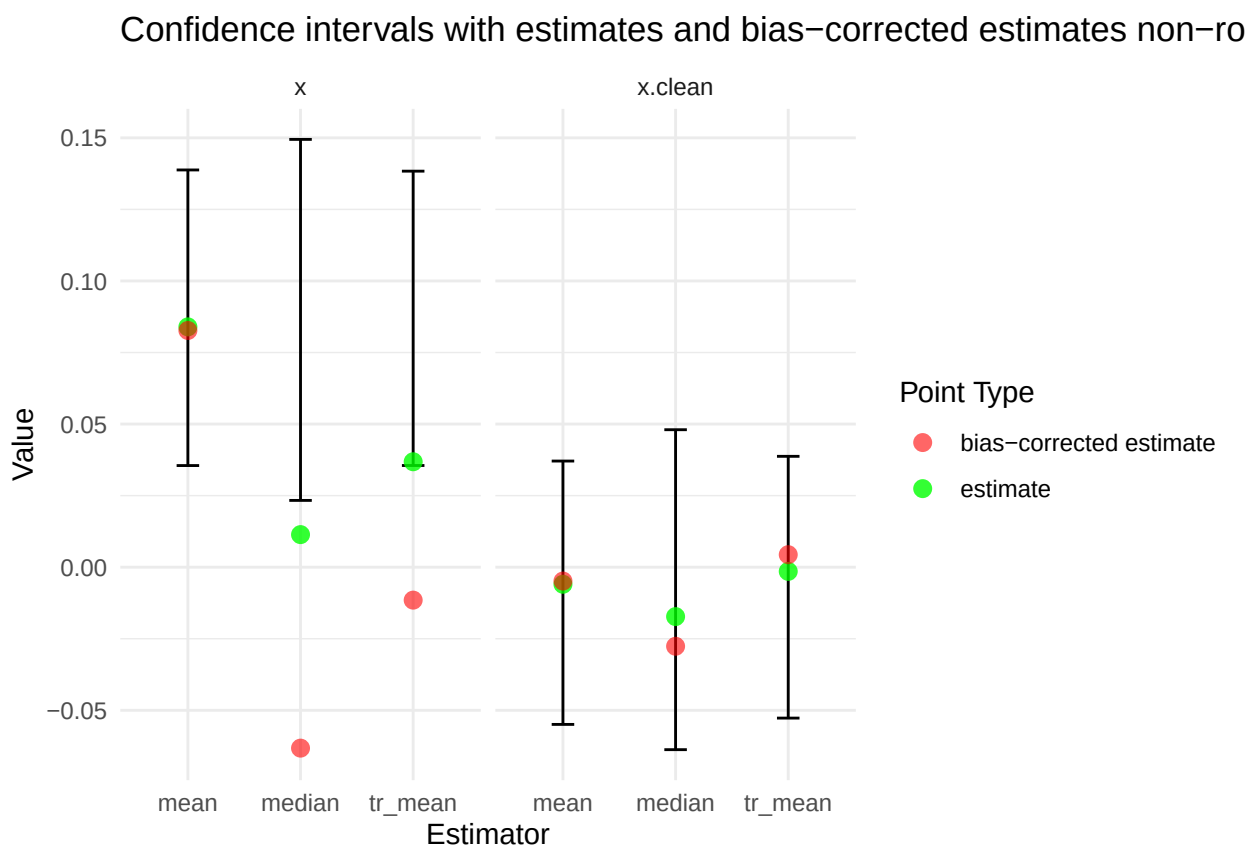
```
print(results_pb_clean)
```

```
##             estimate         bias         se    ci_lower   ci_upper
## mean    -0.005968976 -0.001129922 0.02280776 -0.05492990 0.03708279
## tr_mean -0.001462623 -0.005833317 0.02306910 -0.05271906 0.03872808
## median  -0.017253605  0.010370828 0.02791014 -0.06375058 0.04803064
##         bias_corr_estimate
## mean          -0.004839055
## tr_mean        0.004370694
```

```
## median        -0.027624433
```

The data is visualized by plotting the confidence intervals as well as the estimates and bias corrected estimates.

```
library(ggplot2)
results_pb$Data <- "x" # creating new column to distinguish data
results_pb_clean$Data <- "x.clean"
results_pb$Estimator <- rownames(results_pb) # new column for mapping correctly
results_pb_clean$Estimator <- rownames(results_pb_clean)

combined_results <- rbind(results_pb, results_pb_clean)

ggplot(combined_results, aes(x = Estimator)) +
  geom_errorbar(aes(ymin = ci_lower, ymax = ci_upper), width = 0.2) +
  geom_point(aes(y = estimate, color = "estimate"), size = 3, shape = 16, alpha = 0.8) +
  geom_point(aes(y = bias_corr_estimate, color = "bias-corrected estimate"), size = 3, shape = 16, alpha
  scale_color_manual(
    name = "Point Type",
    values = c("estimate" = "green", "bias-corrected estimate" = "red"))+
  labs(title = "Confidence intervals with estimates and bias-corrected estimates non-robust case",
       y = "Value",
       x = "Estimator") +
  facet_wrap(~ Data) +
  theme_minimal()
```



As a second implementation we do the same as before but as the assumption of the underlying distribution we use the more robust approach of using the median as the location parameter and as specified the median absolute deviation (MAD) as the scale parameter. The resukts for the mean and the trimmed mean are

again returned as a data frame.

```
boot_param_robust <- function(y){
  n <- length(y)
  sde <- mad(y)
  location <- median(y)

  means <- numeric(1000)
  trimmed_means <- numeric(1000)
  medians <- numeric(1000)
  for (i in 1:1000) {
    samples <- rnorm(n, location, sde)
    means[i] <- mean(samples)
    trimmed_means[i] <- mean(samples, trim = 0.05)
    medians[i] <- median(samples)
  }
  est <- c(mean(y), mean(y, trim = 0.05), median(y))
  bias <- c(mean(means)-mean(y),
            mean(trimmed_means)-mean(y, trim = 0.05),
            median(medians)-median(y))
  se <- c(sd(means), sd(trimmed_means), sd(medians))
  ci_mean <- quantile(means, c(0.025, 0.975))
  ci_trim <- quantile(trimmed_means, c(0.025, 0.975))
  ci_median <- quantile(medians, c(0.025, 0.975))
  ci_low <- c(ci_mean[1],ci_trim[1], ci_median[1])
  ci_up <- c(ci_mean[2],ci_trim[2], ci_median[2])
  bias_cor <- c(mean(y)-bias[1],
                mean(y, trim = 0.05)-bias[2],
                median(y)-bias[3])

  results <- data.frame(estimate = est,
                        bias = bias,
                        se = se,
                        ci_lower = ci_low,
                        ci_upper = ci_up,
                        bias_corr_estimate = bias_cor)
  rownames(results) <- c("mean", "tr_mean", "median")

  return(results)
}
```

Showing the resulting table.

```
results_pb_robust <- boot_param_robust(x)
results_pb_robust_clean <- boot_param_robust(x.clean)
print(results_pb_robust)
```

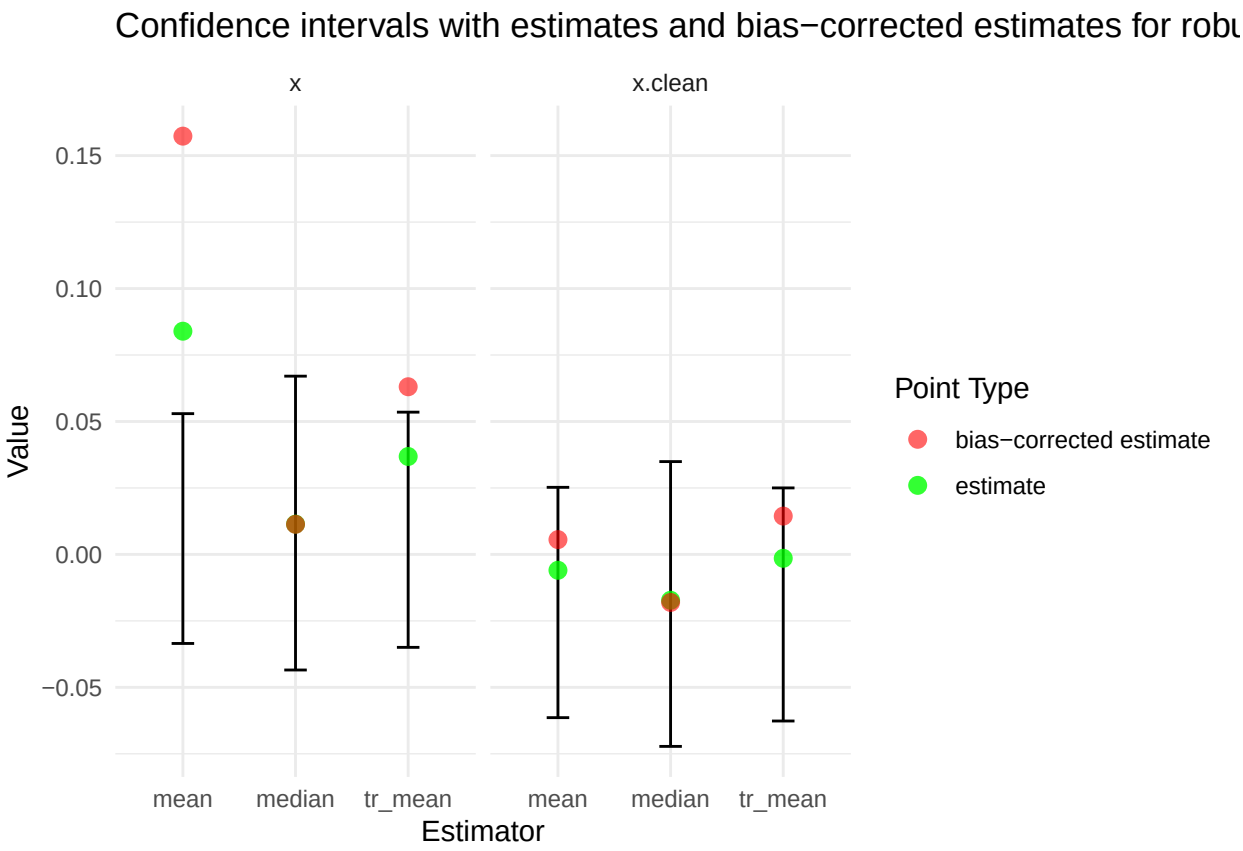
```
##           estimate          bias         se    ci_lower   ci_upper
## mean    0.08395508 -0.0733768850 0.02277133 -0.03349527 0.05294980
## tr_mean 0.03683294 -0.0262279383 0.02308336 -0.03496622 0.05350826
## median  0.01137970  0.0000606986 0.02781481 -0.04347866 0.06704980
##         bias_corr_estimate
## mean            0.15733196
## tr_mean         0.06306088
## median          0.01131900
```

```r
print(results_pb_robust_clean)
```

```
##             estimate          bias         se    ci_lower   ci_upper
## mean    -0.005968976 -0.0115315704 0.02196798 -0.06140786 0.02523500
## tr_mean -0.001462623 -0.0158759376 0.02237249 -0.06266577 0.02500932
## median  -0.017253605  0.0008119238 0.02803414 -0.07221515 0.03491513
##         bias_corr_estimate
## mean           0.005562594
## tr_mean        0.014413314
## median        -0.018065529
```

The data for the more robust paramteric bootstrappinbg is visualized in the same way as before.

```r
library(ggplot2)
results_pb_robust$Data <- "x"
results_pb_robust_clean$Data <- "x.clean"
results_pb_robust$Estimator <- rownames(results_pb_robust)
results_pb_robust_clean$Estimator <- rownames(results_pb_robust_clean)

combined_results_robust <- rbind(results_pb_robust, results_pb_robust_clean)

ggplot(combined_results_robust, aes(x = Estimator)) +
  geom_errorbar(aes(ymin = ci_lower, ymax = ci_upper), width = 0.2) +
  geom_point(aes(y = estimate, color = "estimate"), size = 3, shape = 16, alpha = 0.8) +
  geom_point(aes(y = bias_corr_estimate, color = "bias-corrected estimate"), size = 3, shape = 16, alpha
  scale_color_manual(
    name = "Point Type",
    values = c("estimate" = "green", "bias-corrected estimate" = "red"))+
  labs(title = "Confidence intervals with estimates and bias-corrected estimates for robust case",
       y = "Value",
       x = "Estimator") +
  facet_wrap(~ Data) +
  theme_minimal()
```



part 5

In the first part we used nonparametric bootstrapping to get estimates for the mean, trimmed mean and median. To do so we resampled the original data multiple times for new samples of the same length and then calculated the three values for each sample. The resulting values were then used to get the standard error and 95 percentile CI. For x we therefore resample the already contaminated data. The uncontaminated data x.clean was normally standrad distributed (with mean 0) and the contmainated part uniform (4, 5). Therefore it is not unsurprising that especially then mean is to a significant degree higher than 0 since it is very sensitive to outliers. The median is a very robust estimator and therefore still very close to zero, while the trimmed mean is more robust than the normal mean but still a bit more sensitive than the median (depending on the trim percentage) and therefore between the mean and median. Since we simply resampled the original data and calculated the three values based on that, unsurprisingly the estimators are also each centered in its respective confidence interval. For the clean data x.clean all three estimates are very close to zero since there can only be actual random outliers and not the contaminated part in the used data (and CIs cenered around the estimators again).
Now we take a look at the results of the parametric bootstrap. In parametric bootstrap an assumption is made about the underlying distribution of the data and samples then taken from that distribution. We made two versions of the bootstrap sampling. In the first we simply assumed a standard normal distribution with location parameter mean of the data used and scale parameter its standard deviation. For the x.clean data the results are simply that all three values mean, trimmed mean and median have estimates and bias-corrected estimates close to zero and are mostly centered in their confidence intervals. This can easily be explained since the data x.clean is actually standard normally distributed and therefore the underlying assumption used in the bootstrapping correct. This means the samples follow the same distribution as the original data x.clean. In the case of x the results are different. For the mean confidence interval, estimate and corrected estimate are significantly higher than 0, but the estimate and corrected estimate are still centered in the CI. Since the comntaminated data was used, this just represents the actual variance and underlying distribution

in the contmainated data that is not normally distributed anymore. The mean is sensitive to outliers and therefore catches this variance that the drawn samples also have. This is different for the median. The median is more robust and therefore its estimator is still close to zero since it is not sensitive to the outleirs in the contaminated data. On the other hand it now does not catch the variance in the contaminated data and the samples that were drawn from it. Therefore its bias is way higher but does not catch the variance properly. All this results in both estimate and bias-corrected estimate being outside the CI as well as the corrected estimate being far off. The trimmed mean behaves similar to the median but slightly less off since it is a bit less robust than the median regarding outliers.
In the second version we used more robust paramtric bootstrap sampling by using the median as the location paramter and the MAD as the scale parameter which are both not as sensitive to outliers as mean and standard deviation. Using the clean data x.clean again gives similar results since the data is not contmainated and both parameters good estimates and therefore the bootstrap samples follow a similar distribution as x.clean. This means the estimates/corrected estimates and CIs are all roughly centered around zero. For the data x it is different in this case. Since the drawn samples are now less sensitive to the outliers and do not catch the variance of the contaminated data, for the mean the bias is high and catches the variance wrongly which results in both estimate and corrected estimate being outside the CI as well as the corrected estimate being even more off. The median estimate and corrected estimate are both close to zero with a low bias, also being inside the CI even if on the low end. They now more resemble the underlying parameters of the uncontaminated data since the sampling was more robust and put less weight on outliers. The trim is again a mix between both. The CIs are now centered around 0 since the sampling was robust.

Task 3

Bootstrapping means resampling based on your original data to gain confidence intervals and bias for certain estimators. A large amount of samples are drawn and then a certain statistic is calculated for each new sample (median, mean, etc.). Based on the values for these statistic for all these samples their distribution is generated and therefore the Confidence interval can be calculated. There is non-parametric and parametric bootstrapping. In the non-paramteric version samples are simply redrawn many times from the original data which simulates drawing from a population based on the observed data. This is especially useful when the underlying distribution of the data is not known or the sample is small. In parametric bootstrapping an assumption about the underlying distribution is being made. Estimators for the parameters of this distribution are derived from the original data and then from this distribution samples are drawn. In our example we had normal distributed data that was contaminated with some few outliers from a uniform distribution. We can then make the choice if we want to incldue such outliers or try tofocus on the original distribution by using robust estimators for the parameters such as the median and MAD or sensitive ones like the mean and standard error. Parametric bootstrapping is preferable when having osme informaiton about the underlying distribution but needing the data to fit the assumed distribution reasonably well. In bootstrap testing hypothesis tests could be made using in parametric bootstrapping by using some derived estimators for the parameters as the null-hypothesis and then using the distribution of the test statistic for the test. In nonparametric bootstrap testing the observed difference of some statistic can be compared to the difference of that statistic using nonparametric bootstrapping and samples redrawn from the original data.