Exercise 5

Bootstrap II

Olesia Galynskaia 12321492

2025

Task 9

1. Fixing the seed and sampling

I create a clean normal sample and then add 40 large outliers.

Making a sample

```r
set.seed(1234)

n.clean <- 1960
x.clean <- rnorm(n.clean)
x.cont  <- runif(40, min = 4, max = 5)
x       <- c(x.clean, x.cont)

length(x); length(x.clean)
```

```
## [1] 2000
```

```
## [1] 1960
```

2. Mean and trimmed mean

I calculate the ordinary mean and the alpha = 0.05 trimmed mean for both datasets. This shows how much outliers change the estimates.

```r
alpha <- 0.05
trim_fun <- function(z) mean(z, trim = alpha)

mean_x       <- mean(x)
mean_x_clean <- mean(x.clean)

tmean_x       <- trim_fun(x)
tmean_x_clean <- trim_fun(x.clean)

mean_x; mean_x_clean
```

```
## [1] 0.08395508
## [1] -0.005968976
```

```
tmean_x; tmean_x_clean
```

```
## [1] 0.03683294
```

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

For the clean data, both the mean and the trimmed mean are very close to zero.

For the contaminated data, the ordinary mean jumps to about 0.084, while the trimmed mean is only around 0.037.
This shows that the trimmed mean reduces the impact of outliers.

3. Bootstrap

I define helper functions for nonparametric and parametric bootstrap.
They resample the data and compute the statistic repeatedly so I can estimate bias, standard error, and confidence intervals.

```
boot_np <- function(data, stat_fun, B = 2000) {
  n <- length(data)
  replicate(B, stat_fun(sample(data, size = n, replace = TRUE)))
}

boot_par <- function(data, stat = c("mean", "trim"), B = 2000, alpha = 0.05) {
  stat <- match.arg(stat)
  n <- length(data)

  if (stat == "mean") {
    mu_hat <- mean(data)
    sigma_hat <- sd(data)
    stat_fun <- mean
  } else {
    mu_hat <- mean(data, trim = alpha)
    sigma_hat <- mad(data)
    stat_fun <- function(z) mean(z, trim = alpha)
  }

  replicate(B, {
    z <- rnorm(n, mean = mu_hat, sd = sigma_hat)
    stat_fun(z)
  })
}

summarise_boot <- function(theta_hat, boot_vals, conf_level = 0.95) {
  alpha <- 1 - conf_level
  boot_mean <- mean(boot_vals)
  bias_hat  <- boot_mean - theta_hat
  se_hat    <- sd(boot_vals)
  ci_perc   <- quantile(boot_vals, probs = c(alpha/2, 1 - alpha/2), type = 6)
  theta_bc  <- theta_hat - bias_hat
```

```r
  list(
    theta_hat = theta_hat,
    bias      = bias_hat,
    se        = se_hat,
    ci_lower  = ci_perc[1],
    ci_upper  = ci_perc[2],
    theta_bc  = theta_bc
  )
}
```

Here I run both bootstrap methods (nonparametric and parametric) for four situations:
clean mean, clean trimmed mean, contaminated mean, contaminated trimmed mean.

```r
B <- 2000

# Nonparametric
np_mean_clean  <- boot_np(x.clean, mean, B)
np_tmean_clean <- boot_np(x.clean, trim_fun, B)

np_mean_x  <- boot_np(x, mean, B)
np_tmean_x <- boot_np(x, trim_fun, B)

# Parametric
par_mean_clean  <- boot_par(x.clean, stat = "mean", B = B, alpha = alpha)
par_tmean_clean <- boot_par(x.clean, stat = "trim", B = B, alpha = alpha)

par_mean_x  <- boot_par(x, stat = "mean", B = B, alpha = alpha)
par_tmean_x <- boot_par(x, stat = "trim", B = B, alpha = alpha)
```

Clean data:

Both bootstrap methods give nearly identical bias and standard errors.
All bias values are extremely small, and SE is about 0.022 for all cases. Everything behaves normally.

Contaminated data:

The mean has higher SE (around 0.025-0.026), and its estimate is clearly shifted.
The trimmed mean stays close to its clean-data version, and its SE barely increases.
The parametric bootstrap reacts more strongly for the mean, because the sample standard deviation is inflated by outliers.
For the trimmed mean, both bootstrap types stay stable.

4. Summary

Here is summarizing of the results

```r
res <- list(
  "clean_mean_np"    = summarise_boot(mean_x_clean, np_mean_clean),
  "clean_mean_par"   = summarise_boot(mean_x_clean, par_mean_clean),
  "clean_tmean_np"   = summarise_boot(tmean_x_clean, np_tmean_clean),
  "clean_tmean_par"  = summarise_boot(tmean_x_clean, par_tmean_clean),

  "cont_mean_np"     = summarise_boot(mean_x, np_mean_x),
  "cont_mean_par"   = summarise_boot(mean_x, par_mean_x),
  "cont_tmean_np"   = summarise_boot(tmean_x, np_tmean_x),
  "cont_tmean_par"  = summarise_boot(tmean_x, par_tmean_x)
)

res_df <- do.call(rbind, lapply(names(res), function(nm) {
  z <- unlist(res[[nm]])
  data.frame(
    case     = nm,
    theta_hat = z["theta_hat"],
    bias      = z["bias"],
    se        = z["se"],
    ci_lower  = z["ci_lower"],
    ci_upper  = z["ci_upper"],
    theta_bc  = z["theta_bc"],
    row.names = NULL
  )
}))

res_df
```

```
##              case    theta_hat          bias         se ci_lower ci_upper
## 1   clean_mean_np -0.005968976  0.0008105311 0.02279841       NA       NA
## 2  clean_mean_par -0.005968976  0.0005176975 0.02262726       NA       NA
## 3  clean_tmean_np -0.001462623 -0.0002018596 0.02244772       NA       NA
## 4 clean_tmean_par -0.001462623 -0.0005119722 0.02127208       NA       NA
## 5    cont_mean_np  0.083955076  0.0003422021 0.02544500       NA       NA
## 6   cont_mean_par  0.083955076 -0.0011603273 0.02611290       NA       NA
## 7   cont_tmean_np  0.036832939  0.0005067088 0.02357230       NA       NA
## 8  cont_tmean_par  0.036832939 -0.0002487888 0.02214398       NA       NA
##        theta_bc
## 1 -0.0067795073
## 2 -0.0064866738
## 3 -0.0012607637
## 4 -0.0009506511
## 5  0.0836128735
## 6  0.0851154029
## 7  0.0363262301
## 8  0.0370817276
```

The results show that the ordinary mean is very sensitive to outliers: its estimate shifts, and its uncertainty grows.
The 5% trimmed mean remains much closer to the value from the clean data, and its bootstrap SE stays almost unchanged.
Both bootstrap methods confirm the same pattern.
The trimmed mean is much more robust, while the standard mean can be misleading when outliers are present.

Task 10

1. Generate sample

I fix the random seed and draw 100 observations with replacement from the integers -15 to 15.
Then I compute the sample median which I want to study.

```
set.seed(1234)

x <- sample((-15):15, size = 100, replace = TRUE)
n <- length(x)

theta_full <- median(x)
theta_full
```

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

2. Jackknife standard error of the median

I apply the usual jackknife by deleting one observation at a time.
For each reduced sample I compute the median and then use the jackknife formula to estimate the standard error.

```
theta_j <- numeric(n)

for (i in 1:n) {
  theta_j[i] <- median(x[-i])  # median after deleting observation i
}

theta_bar_j <- mean(theta_j)

se_jack <- sqrt((n - 1) / n * sum((theta_j - theta_bar_j)^2))
se_jack
```

```
## [1] 0
```

The delete-1 jackknife standard error is 0.
This happens because removing any single observation does not change the median.
All jackknife medians are equal to 1, so the formula produces SE = 0.
This shows that the median is very stable under single-point deletions for this dataset.

3. Delete-2 jackknife standard error of the median

I apply the delete-2 jackknife.
I remove all possible pairs of observations, compute the median for each subsample, and then use the delete-d jackknife formula with d = 2 to estimate the standard error.

```
d <- 2
comb_idx <- combn(n, d)    # all pairs of indices to delete
b <- ncol(comb_idx)        # number of subsamples
```

```r
theta_d2 <- numeric(b)

for (k in 1:b) {
  idx_drop <- comb_idx[, k]
  theta_d2[k] <- median(x[-idx_drop])
}

theta_bar_d2 <- mean(theta_d2)

se_jack_d2 <- sqrt((n - d) / (d * b) * sum((theta_d2 - theta_bar_d2)^2))
se_jack_d2
```

```
## [1] 1.530547
```

The delete-2 jackknife SE is about 1.53.
Removing two observations at a time sometimes shifts the median, so the jackknife values are no longer identical.
The formula detects this variation and produces a positive SE. Delete-2 jackknife is more sensitive here and gives a more realistic idea of uncertainty.

Summary

Delete-1 jackknife underestimates the uncertainty because the median does not change when only one point is removed.
Delete-2 jackknife shows that the median can shift when slightly larger parts of the sample are removed, so the SE becomes non-zero.
Together these results show that the usual jackknife can fail when the statistic is too stable, while delete-d jackknife captures variability more accurately.