Assignment 12

Kornel Kovacs

2019 12 06

General setup for the whole project

```
df <- read_csv("./hotels-vienna.csv")
```

Exercise 1

I check the mean of the price in the data.

```
mean(df$price)
```

```
## [1] 131.3668
```

I am simulating 1 000 samples with replacement and saving their means in a df.

```
bootstrap_df <- data_frame(num = 1:1000) %>%
    group_by(num) %>%
    mutate(means = mean(sample(df$price, replace = TRUE)))
```

Therefore, the bootstrap standard deviation of this mean (aka 'the standard error') can be calculated with ease.

```
sd(bootstrap_df$means)
```

```
## [1] 4.367094
```

Exercise 2

I designed the function as the exercise wished.

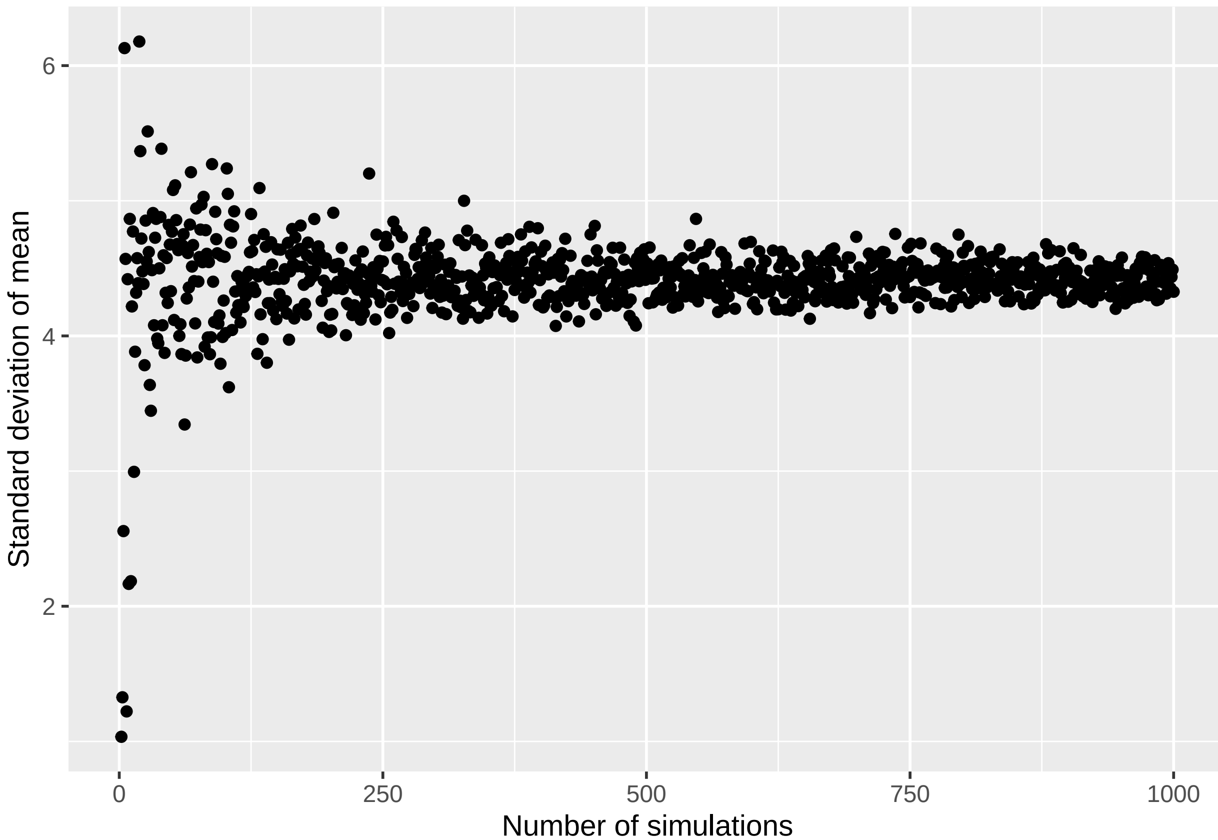
```
bootstrap_mean <- function(B,v) {
  bootstrap_df <- data_frame(num = 1:B) %>%
    group_by(num) %>%
    mutate(means = mean(sample(v, replace = TRUE)))
  return(sd(bootstrap_df$means))
}
```

I apply the function on a bunch of values for the plots.

```
simulated_deviations <- unlist(pblapply(2:1000, bootstrap_mean, v=df$price))
df_to_plot <- data.frame(number_of_simulations = 2:1000,
                         standard_dev = simulated_deviations)
```

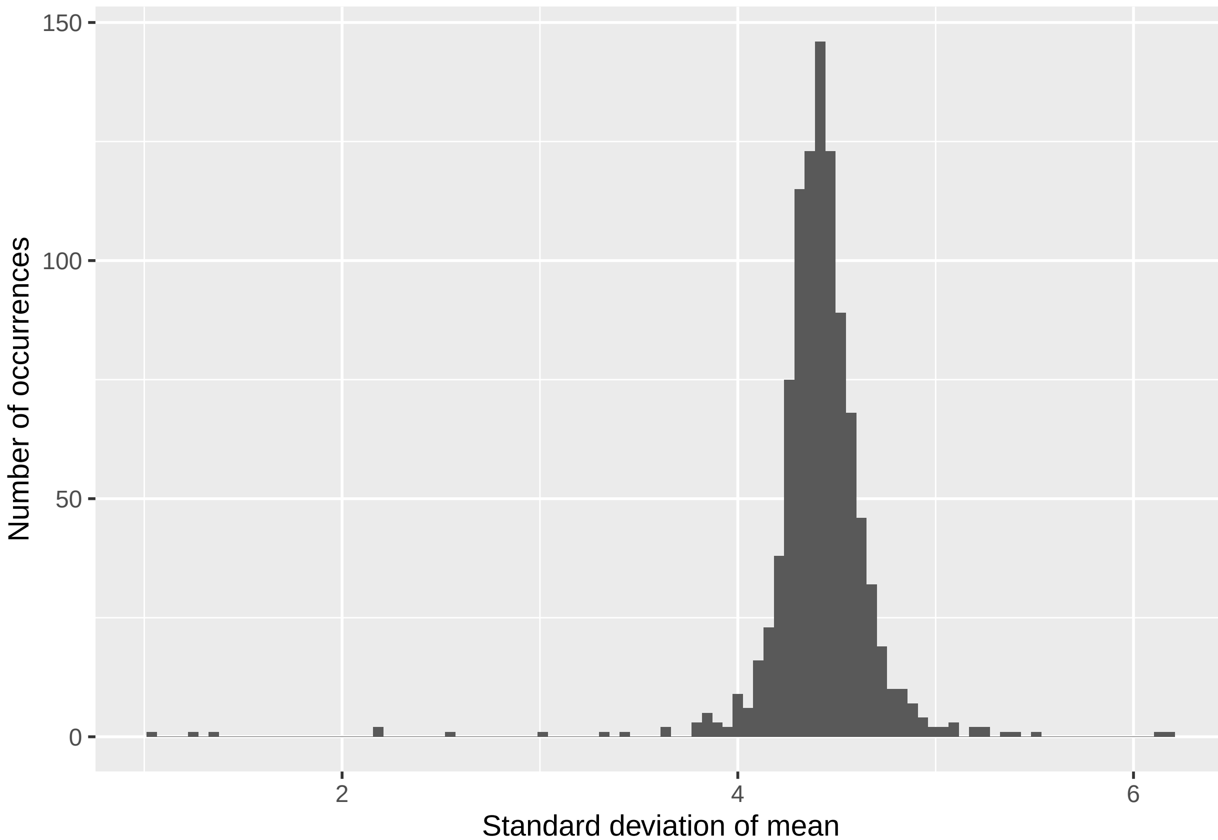
This is the first way of visualisation. It shows standard deviations as a function of how many values we simulate. We can see a nice funnel-like pattern.

```
ggplot(df_to_plot, aes(x = number_of_simulations, y = standard_dev)) +
  geom_point() +
  xlab("Number of simulations") +
  ylab("Standard deviation of mean")
```



A histogram of the standard deviations gives you a hint about the distribution of values. It seems to be normally distributed.

```
ggplot(df_to_plot, aes(x=standard_dev)) +
  geom_histogram(bins = 100) +
  ylab("Number of occurrences") +
  xlab("Standard deviation of mean")
```



Exercise 3

I found a nice package called `plotrix` which has a function that calculates the standard error for mean `price`.

```
std.error(df$price)
```

```
## [1] 4.426713
```

If you are conservative and do not like to use new packages, you may also compute the same result manually.

```
(conservative_std_error <- sd(df$price)/sqrt(length(df$price)))
```

```
## [1] 4.426713
```

Exercise 4

I define the function to calculate standard deviation for median values.

```
bootstrap_median <- function(B,v) {
  bootstrap_df <- data_frame(num = 1:B) %>%
    group_by(num) %>%
    mutate(medians = median(sample(v, replace = TRUE)))
  return(sd(bootstrap_df$medians))
}
```

I also try it out.

```r
bootstrap_median(1000, df$price)
```

```
## [1] 3.164271
```

Exercise 5

I defined my Swiss army knife, a multifunctional method. I had to differentiate between DFs and vectors because their sampling functions are named slightly different.

```r
bootstrap_func <- function(B,v, method) {
  if (is.data.frame(v)) {
    bootstrap_df <- data_frame(num = 1:B) %>%
    group_by(num) %>%
    mutate(metric_value = method(sample_n(v, nrow(v), replace = TRUE)))
    return(sd(bootstrap_df$metric_value))
  } else {
    bootstrap_df <- data_frame(num = 1:B) %>%
    group_by(num) %>%
    mutate(metric_value = method(sample(v, replace = TRUE)))
    return(sd(bootstrap_df$metric_value))
  }
}
```

Example calls:

```r
bootstrap_func(1000, df$price, sd)
```

```
## [1] 12.15995
```

```r
bootstrap_func(1000, df$price, max)
```

```
## [1] 156.2687
```

```r
bootstrap_func(1000, df$price, mean)
```

```
## [1] 4.394451
```

```r
bootstrap_func(1000, df$price, median)
```

```
## [1] 3.029068
```

As we do not have a basic function for the top quartile, some hacking is needed for this case.

```r
bootstrap_func(1000, df$price,function(metric) quantile(metric)[4])
```

```
## [1] 5.189489
```

Exercise 6

I define the function that returns the adequat value of the regression.

```r
get_coeff <- function(data){
  mod <- lm(price ~ distance_alter, data = data)
  return(summary(mod)$coefficients[2, 1])
}
```

I use the above as an input for the general function. This should return the standard error.

```r
bootstrap_func(1000, df, get_coeff)
```

```
## [1] 1.835898
```

Exercise 7

I have the model summary here. I do not see a great difference in the standard error of the coefficient compared to the bootstrap standard error. However, I have no intention to deny that there is. It may vary by simply random sampling, sample sizes could also matter. If you knit my Rmd file, then it is very likely that you will have different results compared to the ones in the submitted HTML and PDF files.

```r
mod <- lm(price ~ distance_alter, data = df)
summary(mod)
```

```
##
## Call:
## lm(formula = price ~ distance_alter, data = df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -95.99 -46.56 -22.31  14.77 881.36
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     164.444     10.903  15.083   <2e-16 ***
## distance_alter   -8.895      2.686  -3.312    0.001 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 90.53 on 426 degrees of freedom
## Multiple R-squared:  0.02511,    Adjusted R-squared:  0.02282
## F-statistic: 10.97 on 1 and 426 DF,  p-value: 0.001004
```