Computer exercises 1

A few words about R programming

We solve the computer exercises of this course using the statistical software R. R is widely used and freely distributed programming language that is particularly suitable for statistical analysis. You should be able to find R from every computer located in the Undergraduate Centre (Otakaari 1) or Maarintalo. The exercises of this course can be solved by using the basic R software, which you can download free of charge to your personal computer. There exists many different integrated development environments (IDE) for the R programming language. We recommend that you use the one called RStudio. RStudio is also free of charge. Note that, in order to use RStudio, you need the basic R software installed. Below are links for installing R and RStudio, respectively.

- Install R – https://cran.r-project.org/
- Install Rstudio – https://www.rstudio.com/products/rstudio/download/

Next, let us review some basic commands that may help you get started.

Assigning values to variables

You can assign values to variables with either `=` or `<-`. Choose one and stick with it. You can comment code with `#`.

```r
# Below lines of code are equivalent.
x <- 5
x = 5
```

You can create vectors with the function `c`.

```r
y <- c(1, 2, 3)
```

A matrix is created by giving its elements as an vector.

```r
z <- matrix(c(1, 1, 1, 2, 2, 2, 3, 3, 3, 4), nrow = 5, ncol = 2, byrow = FALSE)
z
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    1    3
## [3,]    1    3
## [4,]    2    3
## [5,]    2    4
```

Arguments `nrow` and `ncol` define the number of columns and rows in the matrix. Argument `byrow` tells how the matrix is filled, in this case one column at a time from left to right. While assigning values to arguments of a function, always use `=`.

All objects from the workspace can be removed with the following command.

```r
rm(list = ls())
```

Alternatively, one can click the brush logo in the upper right-hand corner of RStudio interface.

Searching for documentation and using help

Manual pages for functions, classes etc. can be found with the command `help` or `?`.

```
# How do I create matrices?
help(matrix)
?matrix

# How does matrix multiplication work?
help("%*%")
?"%*%"
```

Command `help.search` is useful for searching documentation.

```
help.search("transpose")
```

Setting the working directory

Whenever you refer to a file with a relative path, you have to type the path relative to your working directory.

Path of the current working directory can be seen with the function `getwd`. The working directory can be set with the function `setwd`. In RStudio this can be done also by choosing from the upper panel: `Session` →`Set Working Directory` →`Choose Directory` (`Ctrl + Shift + H`).

```
setwd("~/teaching/time-series/1week/computer")
getwd()
```

```
## [1] "/home/perej/teaching/time-series/1week/computer"
```

Scripts

It is recommended to write your solutions as R scripts (or R markdown files, more about this later). A new script is created from the upper panel: `File` →`New File` →`R Script` (`Ctrl + Shift + N`). You can run a script with the following command.

```
source("nameofthescript.R")
```

Alternatively, script can be executed with the key combination `Ctrl + Shift + Enter`. Single lines of a script can be executed by pressing `Ctrl + Enter`.

Importing data

The first step of statistical analysis is importing your data into the software. Suppose that you have a file `data.txt` in your working directory. You can import data to R with the command `read.table`. Below is an example.

```
read.table("data.txt", header = TRUE, sep = "\t")
```

The argument `header = TRUE` indicates the first line of the file is a header. If the first line contains data, you can write `header = FALSE` instead. The argument `sep = "\t"` indicates that the observations are separated with a tabulator.

Further resources for learning R

- A Little Book of R for Time Series – https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/
- R for Data Science – https://r4ds.had.co.nz/index.html
- Advanced R – https://adv-r.hadley.nz/

R Markdown

With R Markdown one can combine R code and text into reports. In fact, this document is produced with R Markdown! Note that the use of R Markdown is not compulsory, however, it can provide a neat way for returning your homework. Below are a couple sources for learning R Markdown:

- R Markdown: The Definitive Guide – https://bookdown.org/yihui/rmarkdown/
- R Markdown Cookbook – https://bookdown.org/yihui/rmarkdown-cookbook/

Demo exercises

1.1

The file `emissions.txt` contains a study, where the nitrous emissions (`NOx`) of a diesel engine were measured under different values of humidity (`Humidity`), temperature (`Temp`) and pressure (`Pressure`).

a) Get familiar with the data by creating histograms for the different variables. Do the variables look normally distributed?

b) Estimate a linear model, where the amount of nitrous emissions is explained with variables humidity, temperature and pressure.

c) What is the coefficient of determination of the estimated model?

d) Use the permutation test to determine which regression coefficients are statistically significant at a significance level of 5%. Use 2000 as the number of permutations.

e) Remove explanatory variables that are not significant at a significance level of 5%. Estimate a new linear model, without the omitted variables. Solve the following parts without the omitted variables.

f) Compute standard deviations for the regression coefficient estimators by using formulas from the lecture slides.

g) Assume that the normality assumption holds. Compute 95% confidence intervals for the regression coefficients with the function `confint`.

h) Assume that the normality assumption holds. Repeat the step g) by using formulas from the lecture slides.

i) Compute 95% confidence intervals for the regression coefficients by bootstrapping and compare the results to parts g) ja h). Use a loop of length 2000 in your bootstrap code. Plot histograms of the bootstrap estimates.

Solution

a)

First, we import the file `emissions.txt`. In addition, set seed to 123 in order to get identical results to the models (parts d) and i)). With function `str` we can see the number of columns/rows, names of the columns and column data types of the data frame `emis`.

```
set.seed(123)
emis <- read.table("data/emissions.txt", header = TRUE, sep = "\t",
                   row.names = 1)
class(emis)
```

```
## [1] "data.frame"
```

```
str(emis)
```

```
## 'data.frame':    20 obs. of  4 variables:
##  $ NOx     : num  0.72 0.7 0.95 0.85 0.79 0.71 0.76 0.79 0.69 0.82 ...
##  $ Humidity: num  96.5 108.7 61.4 91.3 96.8 ...
##  $ Temp    : num  78.1 67.9 88.3 73.6 71 ...
##  $ Pressure: num  29.1 30 29.3 29 29.1 ...
```

Difference between matrices and data frames is that columns of data frame can be of different class.

The data contains 4 variables: `NOx`, `Humidity`, `Temp` and `Pressure`. The function `summary` provides some important statistics.

```
summary(emis)
```

```
##       NOx             Humidity           Temp          Pressure
##  Min.   :0.6900   Min.   : 33.85   Min.   :65.44   Min.   :28.87
##  1st Qu.:0.7175   1st Qu.: 77.94   1st Qu.:73.50   1st Qu.:29.03
##  Median :0.7600   Median : 96.22   Median :77.82   Median :29.07
##  Mean   :0.7910   Mean   : 93.98   Mean   :78.57   Mean   :29.15
##  3rd Qu.:0.8275   3rd Qu.:111.55   3rd Qu.:86.03   3rd Qu.:29.16
##  Max.   :1.0100   Max.   :139.47   Max.   :89.28   Max.   :29.98
```

Histograms can be plotted with the function `hist`. You can plot histograms separately in the following fashion. Histogram for variable `NOx` is shown in Figure 1.

```
hist(emis$NOx)
```

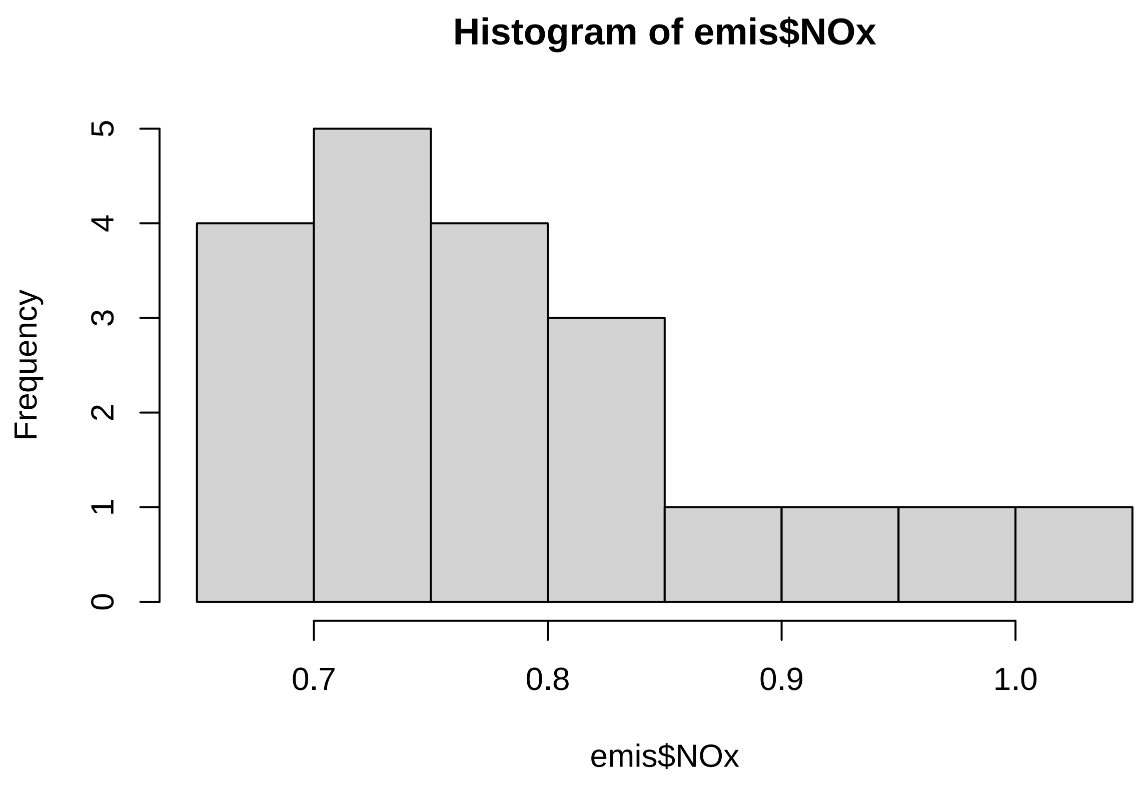

Figure 1: Histogram of NOx emissions.

Alternative one can have each histogram in the same figure by setting correct graphical parameters with the function `par`. Result is shown in Figure 2.

```
par(mfrow = c(2, 2))
hist(emis$NOx)
hist(emis$Humidity)
hist(emis$Temp)
hist(emis$Pressure)
```

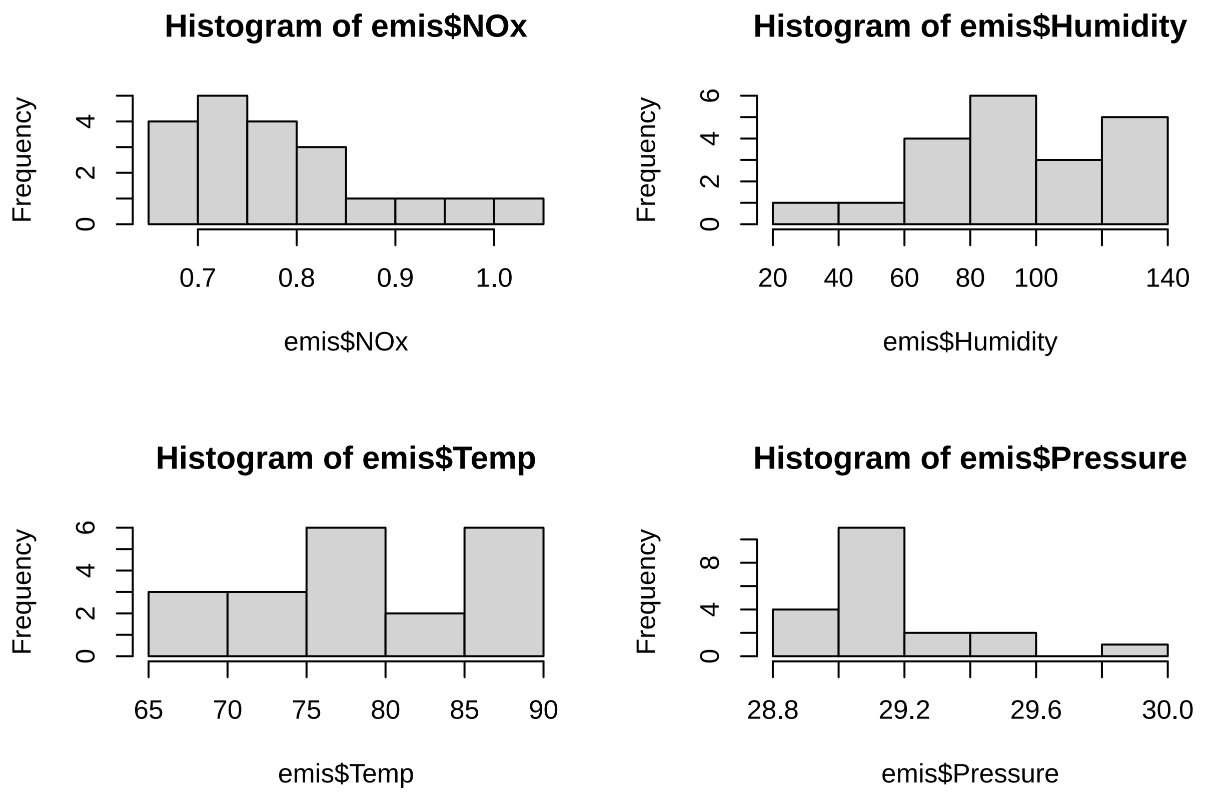

Figure 2: Histograms for each variable.

The correlation matrix is given by

```
cor(emis)
```

```
##                 NOx   Humidity        Temp    Pressure
## NOx       1.0000000 -0.8729408  0.65591970  0.27825058
## Humidity -0.8729408  1.0000000 -0.47282672 -0.27050695
## Temp      0.6559197 -0.4728267  1.00000000  0.02677886
## Pressure  0.2782506 -0.2705070  0.02677886  1.00000000
```

The variable `NOx` correlates negatively with the variable `Humidity` and positively with the variables `Temp` ja `Pressure`. According to Figure 2 it seems that the variables are not normally distributed. However, as the sample size is very small, one should avoid making strong conclusions.

b)

A linear regression model can be estimated with the function `lm`. In the function, we have a "∼"-sign between the response variable and the explanatory variables, and "+"-sign separating the explanatory variables. There are various shorthand notations that one can use with the function `lm`. For example, "`NOx` ∼ `.`" means that we use all the other variables in data frame `emis` other than `NOx` as explanatory variables. Thus two lines of code below are equivalent.

```
fit1 <- lm(NOx ~ Humidity + Temp + Pressure, data = emis)
fit1 <- lm(NOx ~ ., data = emis)
```

The produced object `fit1` is of class `lm`. It is essentially a list containing information about the estimated linear regression model. With summary we can reveal useful information about the model. Notice that the function `summary` works differently with objects of different class. For example, compare summaries of data frame and linear model.

```
summary(fit1)
```

```
##
## Call:
## lm(formula = NOx ~ ., data = emis)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.061616 -0.034795  0.003699  0.029233  0.077782
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.2707790  1.2538066  -0.216   0.8317
## Humidity    -0.0025280  0.0004242  -5.959    2e-05 ***
## Temp         0.0043960  0.0015320   2.869   0.0111 *
## Pressure     0.0327257  0.0418425   0.782   0.4456
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04249 on 16 degrees of freedom
## Multiple R-squared:  0.8441, Adjusted R-squared:  0.8149
## F-statistic: 28.89 on 3 and 16 DF,  p-value: 1.079e-06
```

c)

The coefficient of determination of the model is 84.4%, which corresponds to "Multiple R-square" in the output. Alternatively, one can extract the coefficient of determination from the summary.

```
names(summary(fit1))
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

```
summary(fit1)$r.squared
```

```
## [1] 0.8441422
```

d)

In order to test the significance of the explanatory variables, we use the permutation test. When using the permutation test, we do not need any distributional assumptions about the residuals. In general, strong distributional assumptions should be avoided when possible, especially, when the sample size is small.

The null hypothesis and alternative hypothesis for testing significance of one single regression parameter β_j are the following:

$$H_0 : \beta_j = 0,$$
$$H_1 : \beta_j \neq 0.$$

Next let us perform the permutation test according to lecture slides.

```r
# Number of permutated samples and significance level
k <- 2000
alpha <- 0.05

# Number of variables and number of observations.
m <- ncol(emis)
n <- nrow(emis)

# Create matrix for collecting values of R^2 of permutated samples.
rmatrix <- matrix(NA, nrow = k, ncol = m - 1)

# Conduct significance test for every j = 0,1,2,3.
for (i in 1:k) {
  for (j in 2:m) {
    # Permutate explatonary variable corresponding to B_(j-1).
    tmp <- emis
    tmp[, j] <- sample(tmp[, j], n, replace = FALSE)

    # Calculate R^2.
    fit_tmp <- lm(NOx ~ ., data = tmp)
    rmatrix[i, (j - 1)] <- summary(fit_tmp)$r.squared
  }
}
```

Notice that R coerses data types, whenever it is possible. That is, if you give wrong data types in a function, R tries to modify data types in such a way that the function would still work. For this reason we can sum boolean values since they are coersed into integers, `FALSE` to `0` and `TRUE` to `1`.

```r
FALSE + TRUE + TRUE
```

```
## [1] 2
```

By taking advantage of coersion, we can calculate p-values in the following fashion.

```r
rsq <- summary(fit1)$r.squared
hum <- sum(rmatrix[, 1] >= rsq) / k  # H0 rejected
temp <- sum(rmatrix[, 2] >= rsq) / k  # H0 rejected
pre <- sum(rmatrix[, 3] >= rsq) / k  # No evidence against H0

c(hum, temp, pre)
```

```
## [1] 0.000 0.012 0.459
```

Alternatively, one could follow the lecture slides and order the coefficient of determination values from the smallest to the largest and calculate the empirical 95th percentile from the sample and then compare the calculated value with the original coefficient of determination. If the original coefficient of determination is larger than the calculated percentile, the null hypothesis is rejected.

```
rsq > quantile(rmatrix[, 1], 1 - alpha)  # H0 rejected
```

```
##  95%
## TRUE
```

```
rsq > quantile(rmatrix[, 2], 1 - alpha)  # H0 rejected
```

```
##  95%
## TRUE
```

```
rsq > quantile(rmatrix[, 3], 1 - alpha) # No evidence against H0
```

```
##   95%
## FALSE
```

By the permutation test, there is no evidence against null hypothesis $\beta_3 = 0$, where β_3 is the coefficient corresponding to the variable `Pressure`. Hence variable `Pressure` is removed from the model. The null hypotheses regarding other explanatory variables are rejected. Model selection will be a relevant topic in the second exercise session.

e)

Now let us estimate a new linear model without the variable `Pressure`.

```
# Below two lines of code are equivalent
fit2 <- lm(NOx ~ Humidity + Temp, data = emis)
fit2 <- lm(NOx ~ . - Pressure, data = emis)
summary(fit2)
```

```
##
## Call:
## lm(formula = NOx ~ . - Pressure, data = emis)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.065394 -0.018456 -0.001075  0.032302  0.072869
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.703544   0.140272   5.016 0.000106 ***
## Humidity    -0.002625   0.000401  -6.547 4.98e-06 ***
## Temp         0.004253   0.001504   2.829 0.011586 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04201 on 17 degrees of freedom
## Multiple R-squared:  0.8382, Adjusted R-squared:  0.8191
## F-statistic: 44.03 on 2 and 17 DF,  p-value: 1.891e-07
```

f)

With the notation of the lecture slides we wish to calculate square roots of the diagonal values of the matrix

$$\widehat{D}^2(\mathbf{b}) = s^2(\mathbf{X}^\top\mathbf{X})^{-1},$$

where

$$s^2 = \frac{1}{n-k-1}\sum_{i=1}^{n} e_i^2.$$

```
# Make matrix X of above formula.
colnames(emis)
```

```
## [1] "NOx"      "Humidity" "Temp"     "Pressure"
```

```
tmp <- as.matrix(emis[, c(2, 3)])
xmatrix <- cbind(rep(1, n), tmp)
head(xmatrix)
```

```
##     Humidity  Temp
## 1 1    96.50 78.10
## 2 1   108.72 67.93
## 3 1    61.37 88.27
## 4 1    91.26 73.63
## 5 1    96.83 71.02
## 6 1    95.94 76.11
```

```
# Calculate s^2
l <- ncol(xmatrix)
s2 <- sum(fit2$res^2) / (n - l)

# Calculate sample covariance matrix of b.
covb <- s2 * solve(t(xmatrix) %*% xmatrix)

# Square roots of the diagonal values
stdev <- sqrt(diag(covb))
names(stdev)[1] <- "Intercept"
stdev
```

```
##    Intercept     Humidity         Temp
## 0.1402719539 0.0004009899 0.0015036171
```

Above values are identical with the ones on column `Std. Error` of `summary(fit2)`. Notice that above we did data conversion from data frame to matrix. This is because matrix multiplication "`% * %`" cannot have data frames an input.

g)

```
confint(fit2, level = 0.95)
```

```
##                    2.5 %       97.5 %
## (Intercept)  0.407595872  0.999491779
## Humidity    -0.003471148 -0.001779119
## Temp         0.001080766  0.007425475
```

h)

The same results can be obtained with manual calculation.

```
coef <- fit2$coef
up <- coef + stdev * qt(0.975, n - l)
down <- coef - stdev * qt(0.975, n - l)

down
```

```
##  (Intercept)     Humidity         Temp
##  0.407595872 -0.003471148  0.001080766
```

```
up
```

```
##  (Intercept)     Humidity         Temp 
##  0.999491779 -0.001779119  0.007425475
```

Here qt gives a quantile of t-distribution with $n - l$ degrees of freedom.

i)

Next let us calculate confidence intervals by bootstrapping.

```
# i) Number of bootstrap samples and significance level
k <- 2000
alpha <- 0.05

# Number of variables and number of observations.
m <- ncol(xmatrix)
n <- nrow(xmatrix)

# Create matrix for collecting values of b.
bmatrix <- matrix(NA, nrow = k, ncol = m)
y <- emis$NOx

for (i in 1:(k - 1)) {
  # take random sample of size n (n size of the original sample).
  ind <- sample(1:n, n, replace = TRUE)
  xtmp <- xmatrix[ind, ]
  ytmp <- y[ind]

  # Calculate bootstrap estimates
  btmp <- solve(t(xtmp) %*% xtmp) %*% t(xtmp) %*% ytmp
  bmatrix[i, ] <- t(btmp)
}

# Include original estimate
b <- solve(t(xmatrix) %*% xmatrix) %*% t(xmatrix) %*% y
bmatrix[k, ] <- t(b)

# Calculate confidence interval
qconst <- quantile(bmatrix[, 1], probs = c(0.025, 0.975))
qhum <- quantile(bmatrix[, 2], probs = c(0.025, 0.975))
qtemp <- quantile(bmatrix[, 3], probs = c(0.025, 0.975))

qconst
```

```
##      2.5%     97.5% 
## 0.4865997 0.9605247
```

```
qhum
```

```
##         2.5%        97.5% 
## -0.003368240 -0.001830377
```

```
qtemp
```

```
##        2.5%       97.5% 
## 0.001564712 0.006582653
```

Figure 3 show histograms of bootstrap estimates. Additionally, dashed lines represent the bootstrap confidence intervals.

```r
# Plot histograms
par(mfrow = c(2, 2))
hist(bmatrix[, 1], main = "B_0", col = "dodgerblue")
abline(v = qconst, lwd = 2, lty = "dashed", main = "B_0")

hist(bmatrix[, 2], main = "B_1", col = "dodgerblue")
abline(v = qhum, lwd = 2, lty = "dashed", main = "B_0")

hist(bmatrix[, 3], main = "B_2", col = "dodgerblue")
abline(v = qtemp, lwd = 2, lty = "dashed", main = "B_0")
par(mfrow = c(1, 1))
```

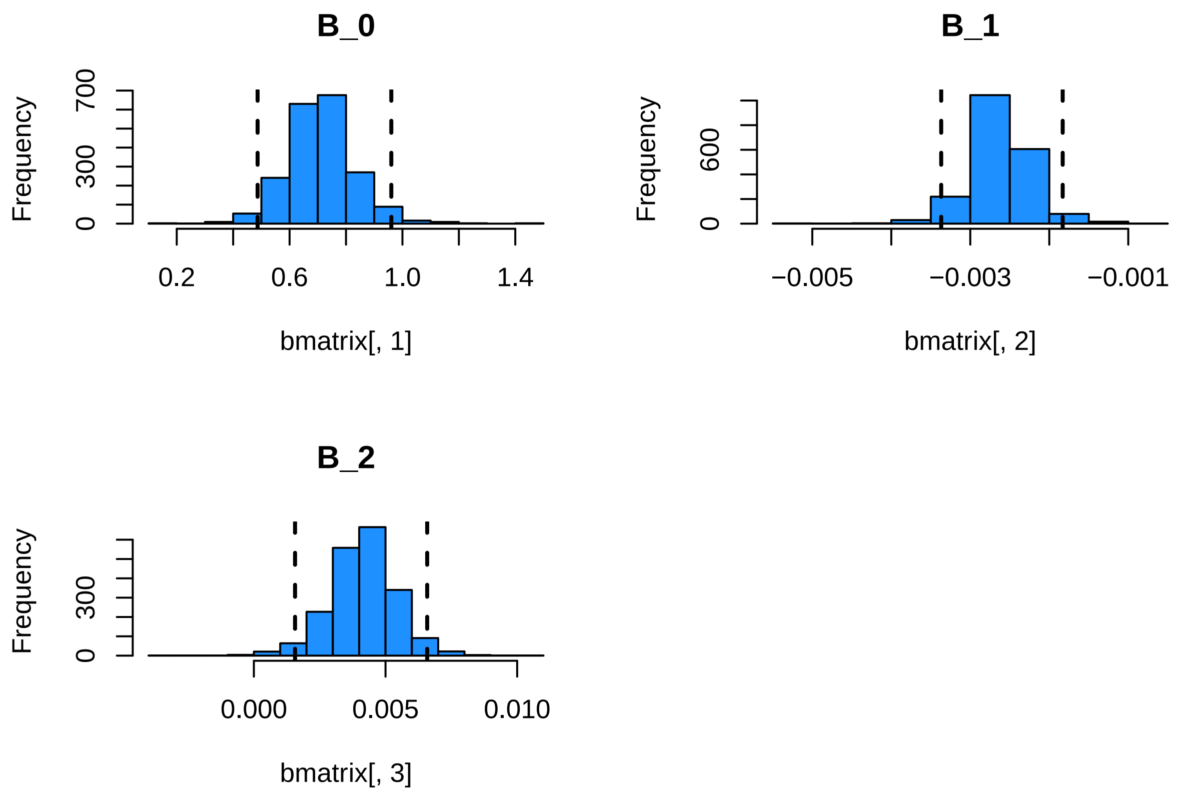

Figure 3: Histograms of bootsrap estimates.

Homework

1.2

The file `tobacco.txt` contains the following information from 11 different countries:

- `CONSUMPTION` = consumption of cigarettes per capita in 1930.
- `ILL` = Lung cancer cases per 100 000 individuals in 1950.

The numbering of the countries is the following:

```
1 = Iceland      7  = USA
2 = Norway       8  = Holland
3 = Sweden       9  = Switzerland
4 = Canada       10 = Finland
5 = Denmark      11 = England
6 = Austria
```

Note that the file contains additional information that is not relevant from the viewpoint of this exercise.

a) Formulate a linear regression model, where the variable `ILL` is explained with the variable `CONSUMPTION`. Include a constant term in your model.

b) Estimate the regression coefficients of the model by using the least squares method and give interpretations for the estimated regression coefficients.

c) What is the coefficient of determination of the model?

d) Is the model statistically significant according to the F-test? Use 1% as the level of significance.

e) Is the variable `CONSUMPTION` statistically significant according to the t-test? Compare the p-value with the p-value obtained in part d) and explain the connection between them.

f) Plot the estimated regression line together with the data.

g) Explain the concept of confidence interval.

h) Suppose that the normality assumptions holds. Form a 95% confidence interval for the slope of the regression line. Give also the 99% confidence interval. Note that the confidence interval is notably wide for the constant term. Can you give some explanation for this?

i) Compute 95% confidence intervals for the regression coefficients with bootstrapping (2000 repetitions). Compare to part h).

j) What advantages bootstrapping has compared to the conventional way of determining confidence intervals?