Installing R and RStudio

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 computer exercises of this course can be solved by using the basic R software, that 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/

Demo Problem 1: Introduction to R

a) Change your working directory. Try the commands `help(c)` and `help(matrix)`.

b) Calculate the affine transformation $\boldsymbol{y} = \boldsymbol{x}\boldsymbol{A}^{-1} + \boldsymbol{b}$, where

$$\boldsymbol{A} = \begin{pmatrix} 2 & 1 & 5 \\ -2 & 7 & 0 \\ 5 & -8 & -1 \end{pmatrix}, \qquad \boldsymbol{x}^T = \begin{pmatrix} 8 \\ -4 \\ 2 \end{pmatrix}, \qquad \boldsymbol{b}^T = \begin{pmatrix} 3 \\ 10 \\ -19 \end{pmatrix}.$$

c) Install the package `mvtnorm` and load the corresponding functions to your workspace. Set the seed to 123 using the command `set.seed(123)`. Generate 100 observations from a two dimensional normal distribution with expected value $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, where

$$\boldsymbol{\mu} = \begin{pmatrix} 3 \\ 1 \end{pmatrix} \quad \text{and} \quad \boldsymbol{\Sigma} = \begin{pmatrix} 4 & 1 \\ 1 & 2 \end{pmatrix}.$$

Visualize the observations.

d) Use the data from part c) and calculate the sample mean $\bar{\boldsymbol{x}}$ and the sample covariance matrix $\boldsymbol{S}_x$. Calculate the eigenvalues and eigenvectors from the matrix $\boldsymbol{S}_x$. Verify from the data, that the following equations hold,

$$\begin{aligned} &\operatorname{Tr}(\boldsymbol{S}_x) = \lambda_1 + \lambda_2 + \ldots + \lambda_p \text{ and} \\ &\operatorname{Det}(\boldsymbol{S}_x) = \lambda_1 \lambda_2 \ldots \lambda_p, \end{aligned}$$

where λ_i are the eigenvalues of $\boldsymbol{S}_x$.

e) Calculate the affine transformation $\boldsymbol{y}_i = \boldsymbol{A}\boldsymbol{x}_i + \boldsymbol{b}$, where

$$\boldsymbol{b} = \begin{pmatrix} 3 \\ 1 \end{pmatrix} \quad \text{and} \quad \boldsymbol{A} = \begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix},$$

verify that $\bar{\boldsymbol{y}} = \boldsymbol{A}\bar{\boldsymbol{x}} + \boldsymbol{b}$ and $\boldsymbol{S}_y = \boldsymbol{A}\boldsymbol{S}_x\boldsymbol{A}^T$. What does affine equivariance mean in practice?

f) Upload the data from the file `data.txt` into your workspace. Create a function, that centers your data (removes the mean) and pairwise scatterplots the variables. Calculate the sample covariance and correlation matrices and the corresponding eigenvalues- and vectors.

Solution

a) 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/multivariate/01week/markdown/")
getwd()
```

```
## [1] "/home/perej/teaching/multivariate/01week/markdown"
```

Manual pages for functions, classes etc. can be found with the command `help` or `?`. You can comment code by starting a line with `#`.

```
# How do I create vectors?
help(c)
?c

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

b) First, create matrix $\boldsymbol{A}$ and vectors $\boldsymbol{x}$ and $\boldsymbol{b}$. You can assign values to variables with either `=` or `<-`. Choose one and stick with it.

We create vectors $\boldsymbol{x}$ and $\boldsymbol{b}$ in two ways. First, with command `c` and then as a row vector with command `matrix`.

```
a <- matrix(c(2, 1, 5, -2, 7, 0, 5, -8, -1), nrow = 3, byrow = TRUE)
x1 <- c(8, -4, 2)
b1 <- c(3, 10, -19)

x2 <- matrix(x1, nrow = 1, byrow = TRUE)
b2 <- matrix(b1, nrow = 1, byrow = TRUE)
```

Matrix multiplication can be performed with the operator `%*%` and inverse matrix can be computed with the function `solve`. Both variables `y1` and `y2` give the right result. This is because according to the documentation of `%*%`, vectors are interpreted as row or column vectors such that arguments are conformable. I you want to be explicit about dimensions of vectors use the function `matrix`.

```
y1 <- x1 %*% solve(a) + b1
y2 <- x2 %*% solve(a) + b2

all(y1 == y2)
```

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

```
y1
```

```
##          [,1]     [,2]      [,3]
## [1,] 3.774775 11.45946 -17.12613
```

For example, the following code gives an error.

```
x3 <- matrix(x1, ncol = 1, byrow = FALSE)
b3 <- matrix(b1, ncol = 1, byrow = FALSE)

x3 %*% solve(a) + b3
```

c) If there is some functionality that is not implemented in base R there is most probably a package for it. You can install packages with the function `install.packages`. Note that the package name has to be given as a character string for the function `install.packages`. For example, the package `mvtnorm` that is required for this exercise session can be installed with the following line of code.

```
install.packages("mvtnorm")
```

Once the package is installed you can use functionality inside the package by specifying the correct namespace and using double colon `::` between the namespace and the function. Below we use the function `rmvnorm` from the package `mvtnorm`.

```
mvtnorm::rmvnorm(3, rep(0, 2))
```

Namespaces are a useful concept since there can be functions with the same name in different packages. For example function `lag` can be found at least in two different packages.

```
?stats::lag
?dplyr::lag
```

Instead of specifying the namespace one can attach the package with the function `library`. For an example, see following lines of code.

```
library(mvtnorm)
rmvnorm(3, rep(0, 2))
```

If you decide to attach packages it is a good practice to put `library` commands at the top of your script, instead of scattering them all around.

Next, we create a sample of size $n = 100$ from multivariate normal distribution with location vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. With function `head` you can see the first few observations.

```
mu <- c(3, 1)
sigma <- matrix(c(4, 1, 1, 2), byrow = TRUE, ncol = 2)
n <- 100

set.seed(123)
x <- rmvnorm(n, mu, sigma)
head(x)
```

```
##          [,1]       [,2]
## [1,] 1.823028  0.5149745
## [2,] 6.103696  1.5613434
## [3,] 3.766090  3.4096342
## [4,] 3.535096 -0.6118415
## [5,] 1.508960  0.1794479
## [6,] 5.527988  1.8617391
```

Figure 1 should look similar if you set seed to 123 before creating the sample.

```
plot(x, pch = 20, xlab = expression("X"[1]), ylab = expression("X"[2]),
     main = expression(paste("Sample from ", "N(", mu, ", ", Sigma, ")")))
```

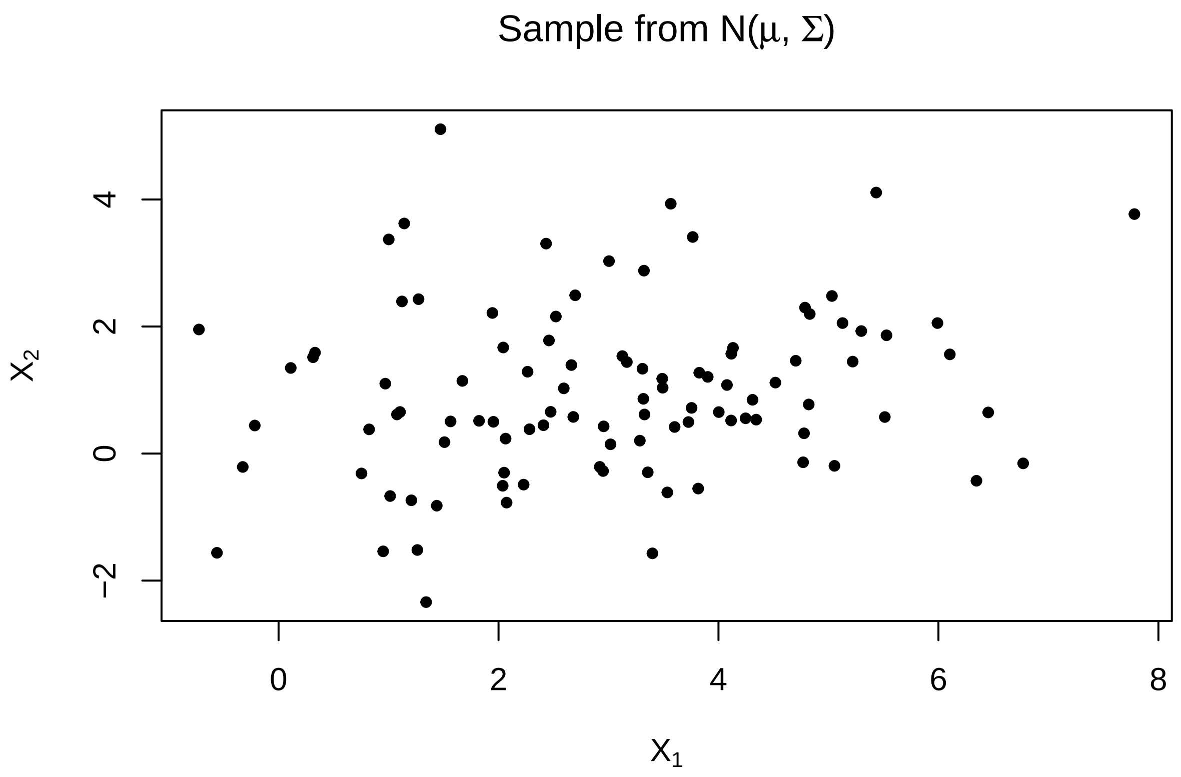

Figure 1: Scatter plot of a sample from a bivariate normal distribution.

d) There are a couple of ways to calculate the sample mean vector.

```
x_mean1 <- apply(x, 2, mean) # Apply function "mean" to every column
x_mean2 <- colMeans(x)
all(x_mean1 == x_mean2)
```

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

```
x_mean1
```

```
## [1] 3.004655 0.970002
```

One can use the function `cov` to calculate the sample covariance.

```
x_cov <- cov(x)
x_cov
```

```
##           [,1]      [,2]
## [1,] 3.0744739 0.4545397
## [2,] 0.4545397 1.8184535
```

Next, let us calculate eigenvalues and eigenvectors of sample covariance matrix $\boldsymbol{S}_x$. Columns of variable `eigvec` contain the eigenvectors.

```
eig <- eigen(x_cov)
eigval <- eig$values
eigvec <- eig$vectors
eigval
```

```
## [1] 3.221708 1.671220
```

```
eigvec
```

```
##            [,1]       [,2]
## [1,] -0.9513361  0.3081552
## [2,] -0.3081552 -0.9513361
```

Lastly, let us verify that

$$\mathrm{Tr}(\boldsymbol{S}_x) = \lambda_1 + \lambda_2$$

and that

$$\mathrm{Det}(\boldsymbol{S}_x) = \lambda_1 \lambda_2.$$

```
sum(diag(x_cov)) - sum(eigval)
```

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

```
det(x_cov) - prod(eigval)
```

```
## [1] -1.776357e-15
```

e) First, let us create matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$.

```
b <- c(3, 1)
a <- matrix(c(1, 2, 3, 1), byrow = TRUE, ncol = 2)
```

We can avoid having any for loops by doing matrix operations and using the function `sweep`. Function `sweep` is quite similar to `apply`. For example, below are two ways to sum vector $\mathbf{b}$ to each row of $\mathbf{A}$.

```
test1 <- sweep(a, 2, b, "+")
test2 <- t(apply(a, 1, "+", b))
all(test1 == test2)
```

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

```
test1
```

```
##      [,1] [,2]
## [1,]    4    3
## [2,]    6    2
```

Below is another example where we sum vector $\mathbf{b}$ to each column of $\mathbf{A}$.

```
test1 <- sweep(a, 1, b, "+")
test2 <- apply(a, 2, "+", b)
all(test1 == test2)
```

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

```
test1
```

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

Now that we have introduced function `sweep`, let us perform the affine transformations. Notice that

$$\mathbf{X}\mathbf{A}^T + \mathbf{1}_{100}\mathbf{b}^T = \begin{pmatrix} (\mathbf{A}\mathbf{x}_1 + \mathbf{b})^T \\ (\mathbf{A}\mathbf{x}_2 + \mathbf{b})^T \\ \vdots \\ (\mathbf{A}\mathbf{x}_{100} + \mathbf{b})^T \end{pmatrix}, \qquad (1)$$

where $\mathbf{1}_{100} \in \mathbb{R}^{100}$ is a column vector of ones.

```
# First way
y1 <- sweep(x %*% t(a), 2, b, "+")

# Second way
ones <- rep(1, n)
y2 <- x %*% t(a) + ones %*% t(b)

all(y1 == y2)
```

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

Lastly, let us check that $\bar{\boldsymbol{y}} = \boldsymbol{A}\bar{\boldsymbol{x}} + \boldsymbol{b}$ and $\boldsymbol{S}_y = \boldsymbol{A}\boldsymbol{S}_x\boldsymbol{A}^T$. We can use any matrix norm $\|\cdot\|$ to check that two matrices $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{n\times m}$ are equal. This works since

$$\boldsymbol{X} = \boldsymbol{Y} \iff \|\boldsymbol{X} - \boldsymbol{Y}\| = 0.$$

We can choose to use, e.g., Frobenius norm.

```
norm(colMeans(y1) - (a %*% colMeans(x) + b), type = "F")
```

```
## [1] 8.881784e-16
```

```
norm(cov(y1) - (a %*% cov(x) %*% t(a)), type = "F")
```

```
## [1] 3.552714e-15
```

f) Be sure that you know the path to the data with respect to the working directory. For example, in this case `data.txt` is located in a different directory than the R script.

```
getwd()
```

```
## [1] "/home/perej/teaching/multivariate/01week/markdown"
```

```
data <- read.table("../data/data.txt", sep = "\t", header = FALSE)
head(data)
```

```
##           V1         V2        V3        V4
## 1 0.04301325 0.39287882 0.3078980 0.8659341
## 2 0.07597216 0.16503982 0.9537341 0.6011355
## 3 0.82452548 0.60148149 0.7152086 0.9866722
## 4 0.56774748 0.81009764 0.7085845 0.8704875
## 5 0.57517488 0.12338534 0.3329567 0.9171741
## 6 0.81746640 0.08863711 0.6017773 0.2063420
```

Here we create the function that plots the pairwise scatter plots and centers the data. Additionally, Figure 2 shows the resulting plot.

```
center <- function(x) {
  pairs(x, pch = 19, col = "midnightblue", gap = 0, upper.panel = NULL,
        cex.labels = 1)
  sweep(x, 2, colMeans(x), "-")
}

data_center <- center(data)
```

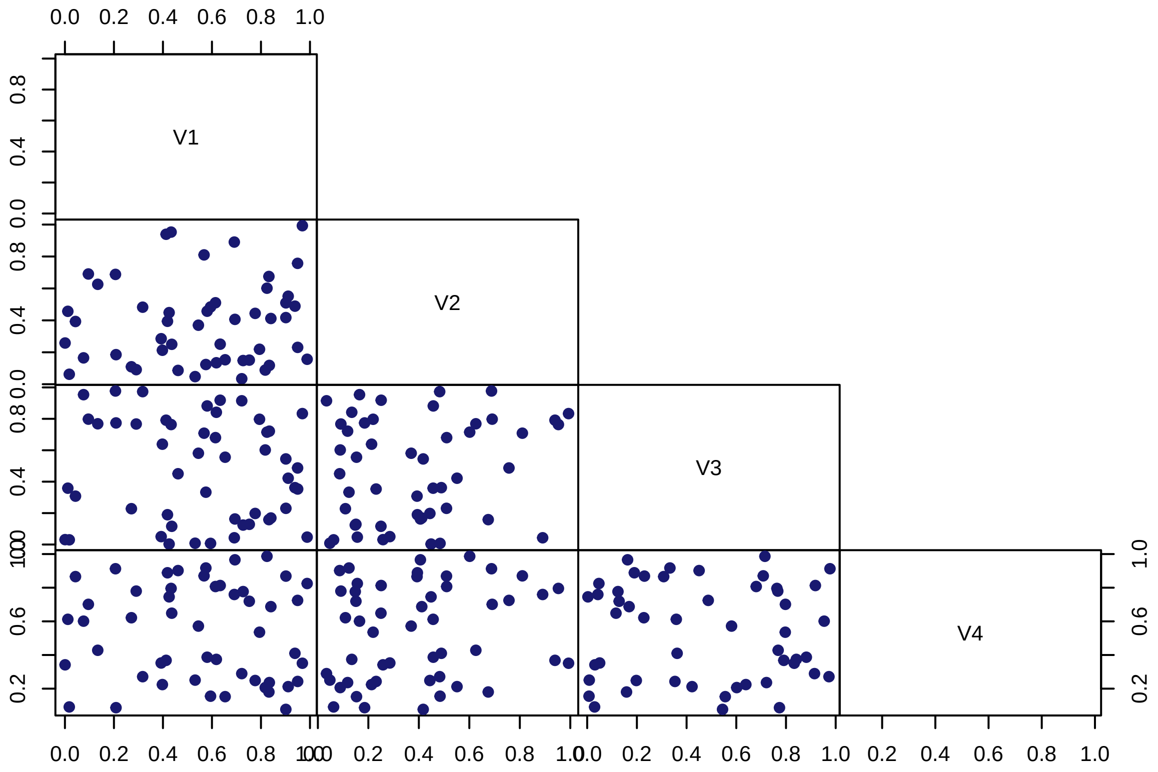

Figure 2: Scatter plot of variables.

```
colMeans(data_center)
```

```
##            V1            V2            V3            V4
## -1.887379e-17  1.221245e-17 -1.665335e-18 -2.109424e-17
```

Lastly, we calculate sample covariance, sample correlation and the corresponding eigenvalues and eigenvectors.

```
center_cov <- cov(data_center)
center_cor <- cor(data_center)
eigen(center_cov)$values
```

```
## [1] 0.11163469 0.08903608 0.08697829 0.05218629
```

```
eigen(center_cov)$vectors
```

```
##             [,1]        [,2]        [,3]       [,4]
## [1,] -0.24232756  0.51237112  0.76290091  0.3110231
## [2,]  0.30797566  0.55314104  0.04145492 -0.7729602
## [3,]  0.91739205 -0.09949153  0.23313028  0.3068282
## [4,]  0.06942745  0.64931676 -0.60159285  0.4600583
```

```
eigen(center_cor)$values
```

```
## [1] 1.2367328 1.0762947 1.0211537 0.6658188
```

```
eigen(center_cor)$vectors
```

```
##             [,1]       [,2]       [,3]       [,4]
## [1,] -0.02338207  0.7717167 -0.5016233  0.3902315
## [2,]  0.72821948  0.2244426 -0.1739016 -0.6237629
## [3,]  0.47737240 -0.5465891 -0.4787554  0.4941145
## [4,]  0.49118760  0.2352002  0.6992321  0.4631307
```

Demo Problem 2: The Eigenvalues of a Symmetric Matrix

Show that the eigenvalues of a real valued symmetric matrix are always real valued.

Solution

Let A be a symmetric real valued $p \times p$ matrix $\left(A = A^T\right)$. Note that, if the symmetry condition is dropped, A can have complex valued eigenvalues and -vectors. Let λ_i be the ith eigenvalue and v_i the corresponding eigenvector of A.

Definition 1 (Eigenvalues and eigenvectors)**.** A scalar λ_i is called an eigenvalue of $p \times p$ matrix A if there is a nontrivial solution v_i to

$$Av_i = \lambda_i v_i,$$

where v_i is called an eigenvector corresponding to the eigenvalue λ_i.

Here, trivial solutions are obtained if $v_i = 0$ (zero vector) since every scalar λ_i would then satisfy the equation above. First, we take the complex conjugate from both sides,

$$\begin{aligned} \overline{(Av_i)} &= \overline{(\lambda_i v_i)} \\ \Rightarrow A\bar{v}_i &= \bar{\lambda}_i \bar{v}_i, \end{aligned}$$

since A is real valued. Then, we multiply the above with v_i^T from the left side

$$\begin{aligned} v_i^\top A\bar{v}_i &= v_i^\top \bar{\lambda}_i \bar{v}_i \\ v_i^\top A^\top \bar{v}_i &= v_i^\top \bar{\lambda}_i \bar{v}_i \\ (Av_i)^\top \bar{v}_i &= \bar{\lambda}_i v_i^\top \bar{v}_i \\ \lambda_i v_i^\top \bar{v}_i &= \bar{\lambda}_i v_i^\top \bar{v}_i \\ \Rightarrow \left(\lambda_i - \bar{\lambda}_i\right) v_i^\top \bar{v}_i &= 0. \end{aligned}$$

Thus we must have that $\lambda_i - \bar{\lambda}_i = 0$ or $v_i^T \bar{v}_i = 0$. However, note that $v_i^T \bar{v}_i = \langle v_i, v_i \rangle$ is the canonical hermitian inner product. By properties of inner product we have

- $\langle v_i, v_i \rangle \geq 0$ and
- $\langle v_i, v_i \rangle = 0$ if and only if $v_i = 0$.

Remember that by definition of eigenvectors we assume that $v_i \neq 0$. Thus the option $v_i^T \bar{v}_i = 0$ is not possible. That is, we must have $\lambda_i = \bar{\lambda}_i$.

Remark. Real *nonsymmetric* matrices can have complex eigenvalues. For example, consider matrix

$$A = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}.$$

```r
a <- matrix(c(1, -1, 1, 1), ncol = 2, byrow = TRUE)
eigen(a)$values
```

```
## [1] 1+1i 1-1i
```

Thus the assumption of symmetricity cannot be dropped from the claim of the exercise.

Homework Problem 1: Functions

In this exercise do not use the built-in functions `cov`, `cor`, `cov2cor` or any additional R packages.

a) Create an R function that takes a data matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$, $n > p$, as an argument and returns the unbiased estimator of the covariance matrix.

b) Create an R function that takes a full-rank covariance matrix $\boldsymbol{A} \in \mathbb{R}^{p \times p}$ as an argument and returns the square root of the inverse matrix such that $\boldsymbol{A}^{-\frac{1}{2}}\boldsymbol{A}^{-\frac{1}{2}} = \boldsymbol{A}^{\boldsymbol{-1}}$.

c) Create an R function that takes a full-rank covariance matrix $\boldsymbol{A}$ as an argument and returns the corresponding correlation matrix.