SDA 2019 — Assignment 1

Solve the exercises below in RStudio, which is available in the computer rooms. RStudio is a graphical shell over R. Both programs are freeware, and can be installed on your own computer. See the SDA Canvas page for useful links, including manuals in Dutch and English.

This assignment consists of 4 exercises. The aim of Exercises 1.1 and 1.2 is to get introduced to RStudio. If you have experience with RStudio, you may skip these introductory exercises. **Hand in Exercises 1.3 and 1.4**. Solve the exercises as efficiently as possible using R.

Make a concise report of your answers in *one single PDF file*, with only *relevant* R code *in an appendix*. It is important to make clear in your answers how you have solved the questions. Graphs should look neat (label the axes, give titles, use correct dimensions etc.). Multiple graphs can be put into one figure using the command `par(mfrow=c(k,l))`, see `help(par)`. Sometimes there might be additional information on what exactly has to be handed in. **Read the file AssignmentFormat.pdf on canvas.vu.nl carefully**.

Exercise 1.1 *Introduction to RStudio*

Start up RStudio, and choose `File` → `New` → `R Script`. Now you should have 4 windows.
top left script window — typing, editing, saving, executing R-commands
bottom left console window — for direct typing and executing commands (**no saving!**)
top right workspace window — overview of known variables, import datasets
bottom right plot window — for graphics (view, save, etc) and help-function

The sign '>' at the beginning of a line in the console window (bottom left) is the R prompt. It is followed by the commands that you type. In case a command is not yet finished, the next line will show a + sign, and you can continue your command after this sign.

Below you find a set of introductory R commands, that shows some of its possibilities. The right column contains some explanation of the corresponding command in the left column. Type these commands directly in the console window and see what you get.

`> x = 1:20` (or `x <- 1:20`)	make a vector `x` with values 1, 2, 3, . . . , 20.
`> x`	print value of `x` on the screen.
`> m = matrix(x,4,5,byrow=T)`	create a matrix `m` with 4 rows and 5 columns with the values of x ordered row-wise.
`> m`	print matrix `m` on the screen.
`> m[2,3]`	print element (2,3) of `m` on the screen.
`> m[2,]`	print all elements of 2^{nd} row of `m`.
`> m[,3]`	print all elements of 3^{rd} column of `m`.
`> y = sample(1:100,20)`	generate random sample of size 20 from the numbers $1, 2, 3, \ldots, 100$.
`> z = x+y`	compute the sum of `x` and `y` coordinate-wise.
`> y = x+ 2*y`	transform `y` coordinate-wise.
`> cbind(x,y)`	form a matrix with columns `x` and `y` and print the result.
`> z <- c(NA, 1/0, 0/0)`	creates a vector of `NA` (not available), `Inf` (infinite), `NaN` (not a number)
`> is.na(z)`	check for missing values and print the result.
`> plot(x,y)`	plot `y` against `x`.
`> abline(100,2)`	add the line `y=100+2*x` to the last plot.

The drawback of typing directly in the console window is the lack of saving the typed commands. The script window (top left) is very useful if you want to type, edit (e.g. correct your typo's) and save code. You can execute lines in the script window by pressing `Ctrl+Enter` (on Mac: `Cmd+Enter`). Execute the remainder of this introductory R commands from the script window. Try the `File` → `Save` option, you will need it later on!

`> x = rnorm(50,0,sqrt(2))`	generate random sample of size 50 from normal distribution with mean 0 and variance 2.
`> y = rnorm(50)`	idem from standard normal distribution.
`> mean(x)`	compute mean of the values in `x`.
`> sd(x)`	compute standard deviation of the values in `x`.
`> var(x)`	compute variance of the values in `x`.
`> cor(x,y)`	compute correlation between `x` and `y`.
`> x[x<0]`	select negative elements in `x`.
`> sum(x<0)`	count number of negative elements in `x`.
`> hist(x,prob=T)`	plot a scaled histogram.
`> help(hist)`	give documentation about the function `hist`.
`> ?hist`	the same.
`> u = seq(-5,5,0.1)`	form sequence of points between -5 and 5 with step size 0.1.
`> v = dnorm(u,0,sqrt(2))`	compute density of normal distribution with mean 0 and variance 2.
`> lines(u,v)`	add plot of computed normal density.
`> {hist(x,xlim=c(-6,6),prob=T)`	repeat plot of scaled histogram on larger interval, but do not yet execute the command.
`+ lines(u,v)}`	plus additional command, and execute both.

`> plot(c(-5,sort(x),5),`
`+ c(0,1:50/50,1),`
`+ type="s",ylim=c(0,1),`
`+ xlab="x",ylab="the ecdf of x")`
	plot empirical distribution function of `x` on (-5,5).[1]

`> lines(u,pnorm(u,0,sqrt(2)))`	add true distribution function.
`> w = seq(-pi,pi,length=100)`	form regular grid of 100 points.
`> plot(cos(w),sin(w),type="l")`	draw a circle.

[1] *Note: first and last value of vector to be plotted on x-axis need to be smaller than* `min(x)` *and larger than* `max(x)`*, respectively. You may need to replace the values -5 and/or 5 to account for this.*

Navigate through your plots, using the arrow buttons in the top line of the plot window (bottom right). Use the `Export` button to save pictures as separate picture files.

Exercise 1.2 *Vectors and matrices*

a. Create a vector `x` consisting of the numbers 23, 0.149, -5.15, and 36 using the function `c`.

b. Add to this vector the number -28, using again `c` or `append`.

c. Order the elements of `x` from small to large using `sort` and call the vector of ordered numbers `y`.

d. Multiply each element of `x` by 4.

e. Round each element of `x` to one decimal place using `round`.

f. Select all positive elements of `x`.

g. Change the third element of `x` into 7.

h. Create a vector `z` consisting of the sequence of numbers between 2 and 4 with step size 0.1. Create a matrix `m` with seven rows and three columns in which the 21 numbers in `z` are ordered columnwise.

i. Extract the element on the second row and in the first column of `m`.

j. Extract the third column of `m`.

k. Compute the mean value of the numbers in `m`.

l. Compute the mean value of each column of `m` by using the R function `apply`.

For the following exercises the R functions `hist`, `stem`, `boxplot`, `plot`, `summary`, and `apply` may be helpful. Use `help(xxxx)` for the manual of any function called `xxxx`.

Exercise 1.3

a. Write a function `t(n,df)` that

- draws a sample of size `n` from the t-distribution $t_{\texttt{df}}$ where $\texttt{df} > 0$ are the degrees of freedom,
  (*Remark:* The t-distribution has mean 0 if $\texttt{df} > 1$, otherwise undefined;
  variance $\frac{\texttt{df}}{\texttt{df}-2}$ if $\texttt{df} > 2$, ∞ if $\texttt{df} \in (1, 2]$, otherwise undefined;
  skewness 0 if $\texttt{df} > 3$, otherwise undefined;
  kurtosis $3 + \frac{6}{\texttt{df}-4}$ if $\texttt{df} > 4$, ∞ if $\texttt{df} \in (2, 4]$, otherwise undefined.)
- plots a scaled histogram of the sample (in black color).
- plots in the same figure the density of the $t_{\texttt{df}}$ distribution (in red color).

  Use the functions `rt`, `hist` and `dt` (see `help(rt)` for the correct parametrisation). Make sure that the area under the histogram equals 1, like the area under the density (see `help(hist)`). Use the function `lines` to combine two graphs in one figure. Furthermore, the graph should look appropriate: use a proper title and axis labels (you can use the function `paste` for this), make sure that nothing is cropped.

b. Play around with different values of `n`≥ 10 and `df`≥ 3. Indicate the influence of each parameter on how well the histogram resembles the true density.

Hand in: the (final) code of your function, your answer to the last question, and 3 or 4 graphical realisations of the function with the corresponding arguments `(n,df)` as illustration to your answer.

Exercise 1.4 is on the next page!

Exercise 1.4 Suppose you are a passionate fisher (perhaps you even are) and participating in a fisher's tournament which you will win if you catch the longest fish among all participants (length from the nose to the end of the tail). Before that, you have to think about which kind of bait you should buy because it will influence the species of fish that you are going to catch. The file `fishcatch.dat.txt` on Canvas contains information on 159 fishes of 7 species. Here is the variable description, i.e. a description of the values in the columns:

1 Observation number (ranges from 1 to 159).
2 Species (Numeric).

(The Latin names of the fish species are 1: Abramis brama 2: Leusiscus idus 3: Leuciscus rutilus 4: Abramis bjrkna 5: Osmerus eperlanus 6: Esox lucius 7: Perca fluviatilis.)

3 Weight of the fish (in grams).
4 Length from the nose to the beginning of the tail (in cm).
5 Length from the nose to the notch of the tail (in cm).
6 Length from the nose to the end of the tail (in cm).
7 Maximal height as % of the length in variable 6.
8 Maximal width as % of length in variable 6.
9 Sex: 1 = male, 0 = female.

Create an informative summary of the length distributions (column 6) of the species *Abramis brama*, *Leuciscus rutilus*, *Esox lucius*, and *Perca fluviatilis*, both graphically and numerically. Put relevant R code again in the appendix.

Note: the data can be loaded using the button `Import dataset` *in the workspace window (top right). Alternatively, you could load the data by using the function* `read.table`*. In both cases you need to apply the command* `as.matrix` *or* `matrix` *to* `fishcatch.dat` *in order to convert it to a* `matrix` *type object.*