SDA 2022 — Assignment 1

Solve the exercises below in RStudio (or some other IDE, but the assistants are most familiar with RStudio). 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 English**; the "particular R-manual" was recently translated for this course.

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 R and RStudio, you may skip these introductory exercises.
Start solving this (and other) exercise(s) **well before** the practical classes! If you do, you will be able to ask your teaching assistant *relevant* questions instead of *trivial* ones that can be answered by a brief look into the Syllabus, lectures, or an R-manual or by some small preparational work.

Hand in Exercises 1.4–1.6. Try to solve these as efficiently as possible.

Make a concise report of your answers in *one single PDF file*, with only *relevant* R code *in an appendix*, i.e. the code that is needed to reproduce your findings. 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 should be handed in.
Carefully read the file AssignmentFormat.pdf on canvas.vu.nl.

Exercise 1.1 (Class) At the beginning of the practical class your teaching assistant will give a short demonstration of how to use RStudio. Be prepared to follow the same steps on your own laptop (make sure to have R and RStudio installed in advance!). Afterwards, continue on your own with Exercises 1.2 and 1.3 and complete every part of them.
Warning: In later exercises or assignments, your teaching assistant will not answer your syntax-related question if it is trivially resolved by carefully following Exercises 1.2 or 1.3. He/she will point you to these exercises in that case.

Exercise 1.2 *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.
`> f <- function(x){x*x}`	defines a function `f(x)` that does the same as `x^2`
`> 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(sort(x), 1:50/50,`	plot empirical distribution function of `x`.
`+ type="s",ylim=c(0,1),`	
`+ xlab="x",ylab="the ecdf of x")`	
`> 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.

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

Exercise 1.3 *Vectors and matrices*

a. Use the function `c()` to create a vector `x` that consists of the numbers 23, 0.1, -5.15.

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`, `set.seed`, `rlnorm`, `rexp`, `list`, `quantile` and `apply` may be helpful. Use `help(xxxx)` for the manual of any function called `xxxx`.

Exercise 1.4 (.RData file hand-in) Write a function `lognorm(n,mu,sigma)` that

- sets the seed to 20220211 plus your group number,
  (Yes, for this it is utterly important that you have registered your group; if you don't have a group partner yet, you could still join one of the empty groups, without a partner for the time being!)
- draws a random sample of size `n` from the lognormal distribution with parameters `meanlog=mu, sdlog=sigma`,
- creates a list `mylist` that contains

`quants =`	a vector consisting of the 5%, 10%, 25%, 50%, 75%, 90%, and 95% quantiles of the drawn sample (use the function `quantile` with the default `type=7`),
`loc =`	the sample mean of the drawn sample,
`spread =`	the sample standard deviation of the drawn sample,
`stud_no =`	a vector that contains the student numbers of you (and your group partner);

- stores the list in an .RData file (use `save(mylist, file="[PATH]/myfile1_[GROUP_NO].RData")`, where `[PATH]` stands for the path on your computer where you wish to save the .RData file and `[GROUP_NO]` stands for the number of the assignment group you have chosen on Canvas).

Call your function `lognorm` with the parameters `n=100, mu=1, sigma=1`.
Hand in: Submit your .RData file created by a call of the above-described function to the (dummy) assignment called "RData A1" on Canvas.
In addition, the R-code of your function must be in the appendix of the regular report you will submit to "Assignment 1" on Canvas.
Note: The .RData file can be as small as e.g. 1KB.

Exercise 1.5

a. Write a function `bin(n,size,prob)` that

   - draws a sample `x` of size `n` from the binomial distribution $B(m, p)$ with parameters $m =$`size` and $p =$`prob`,
   - plots a scaled histogram of the sample (scaled to the probability level; with black colored lines and gray filling, i.e. the default).
   - plots in the same figure the density of the Poisson distribution with parameter `lambda`$= m \cdot p$ (as a red colored curve).
     Hint: Use the function `lines` *with the additional options* `type="s", col="red"` to generate a red step function.

   Use the functions `rbinom`, `hist` and `dpois` (see e.g. `help(dpois)` for back-up information). Make sure that the area under the histogram equals 1, like the area under the density (see `help(hist)`).
   Tip: Use the option `breaks=0:(max(x)+1)-0.001` *to make sure that the bins in the histogram have the width 1 and that your drawn sample will be completely and appropriately included in the plot.*
   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 could use the function `paste` for this), make sure that nothing is cropped.
   Note: Always make sure to use proper labels and captions for your plots!

b. Play around with the parameters by choosing 2 different values for each parameter `n`≥ 100 and `size`≥ 1000 (for some `prob`≤ 0.01) while keeping the product `size*prob` fixed. This way, you will get 4 plots. Indicate the influence of the parameters `n` and `size` on how well the histogram resembles the Poisson density.

Hand in: the (final) code of your function (in the appendix!), your answer to the last question, and the 4 graphical realisations of the function (as explained above) such that they support your answer.

Exercise 1.6 We wish to make data summaries of a recent status of the vaccinations against SARS-CoV2 in Asian countries (partly vaccinated people) and also investigate the relationship to the human development indices of these countries. To this end, we use the dataset `owid_covid_data_selection.csv`[1] (to be found on Canvas).

Proceed as follows: read the data into your R environment (e.g. with the help of `read.csv("owid_covid_data_selection.csv", header=TRUE)`) and extract the revelant data.

a. Make univariate numerical and graphical summaries of the vaccination rates across Asian countries.
   Note: for the graphical summaries, create at least a histogram and a boxplot.

b. Make bivariate numerical and graphical summaries of the relevant data.

Note: you should always describe your findings in words!

[1]source: https://ourworldindata.org/, accessed at Feb. 4, 2022