Lab Exercise-2

This lab introduce the basic understating about reading and analyzing a given data set with the help of R. The lab exercise is divided into nine points (A-I), each has specific application. Go through these points and record it in your lab file

A. Working with R and R-Studio

You would like to save relevant commands for later use. Therefore you should write your commands in a file—a so called R script or R program. The main part of this exercise teaches you how to work with such files in R-Studio.

1. Start R-Studio. Go to the console (lower left window) and write

```
3+2
```

at the prompt. Then use R to calculate 3.4 and 8/2-3.4 (use * for multiplication). Commands written directly at the prompt are not saved for later use.

2. If there is not already an editor open in the upper left window, then go to the file menu and open a new script. Type one of the commands from before in the editor, hold the Control button and push the Enter button. Then the command is transferred to the console and the command is executed (just as before).

3. Make a few more commands in the editor, and run them. Also, mark several commands with the arrow buttons and use Control-Enter.

4. Save the file, close the file, and quit R-Studio (all via the File menu). You are asked if you want to save the workspace image. If you answer "Yes", then everything is saved—both relevant and irrelevant stuff—so unless you have made time-consuming computations, you should say "No".

5. Start R-Studio again, and open the file you just saved (again via the File menu), and redo some of the calculations.

6. Try the command

```
# 4+7
```

Nothing is computed! This is because the character # can be used for comments. Anything after # is discarded by R. It cannot be recommended enough that you make appropriate comments in your R programs. That makes it much easier to use the programs again after a while.

B. Built-in mathematical functions

All standard functions (and many non-standard, too) are programmed in R. Here come some examples.

1. Try the commands `sqrt(16)`, `16^0.5`. Compute 4^3.
2. Try the commands `log10(1000)`, `log(1000)`, `exp(log(1000))`. Then try the command `log2(64)`. Make sure you understand different logarithmic functions.
3. Try the command `?log`. Then a help page opens in the lower right window of RStudio. Read the first few lines; does the text match your observations from the previous question?
4. Try the commands `pi`, `round(pi)`, `round(pi, digits=4)`, and `trunc(pi)`.
5. The sine and cosine functions are implemented in `sin` and `cos`. Calculate $\sin(\pi)$, $\cos(\pi)$, $\sin(\pi/2)$, $\cos(\pi/2)$.

C. Vectors

1. Try the commands:

```
x <- c(3,6,8)
x
x/2
x^2
sqrt(x)

x[2]
x[c(1,3)]
x[-3]
y <- c(2,5,1)
y
x-y
x*y

x[y>1.5]
y[x==6]

4:10
seq(2,3,by=0.1)
rep(x,each=4)
```

An important point is that operations are carried out elementwise!

2. Assume that we have registered the height and weight for four people: Heights in cm are 180, 165, 160, 193; weights in kg are 87, 58, 65, 100. Make two vectors, `height` and `weight`, with the data. The bodymass index (BMI) is defined as

$$\frac{\text{weight in kg}}{(\text{height in m})^2}$$

Make a vector with the BMI values for the four people, and a vector with the natural logarithm to the BMI values. Finally make a vector with the weights for those people who have a BMI larger than 25.

D. Simple summary statistics

In an experiment the dry weight has been measured for 8 plants grown under certain conditions (the values are given below).

1. Try the following commands in order to make a vector with the values and compute various sample statistics:

```
dry <- c(77, 93, 92, 68, 88, 75, 100)
dry

sum(dry)
length(dry)
mean(dry)
sum(dry)/length(dry)                    ## Checking

sort(dry)
median(dry)

sd(dry)
var(dry)
sd(dry)^2
sum((dry-mean(dry))^2) / (length(dry)-1) ## Checking

min(dry)
max(dry)

summary(dry)
```

E. Vectors and sample statistics (more examples)?

1. Assume that we have the following three observations of temperature: 23°C, 27°C, 19°C. Make a vector with these values. Recall the relation between the Celcius and Fahrenheit temperature scale:

$$\text{degress in Fahrenheit} = \text{degrees in Celcisus} \cdot \frac{9}{5} + 32$$

   Make a new vector with the temperatures in Fahrenheit.

2. Assume that you are interested in cone-shaped structures, and have measured the height and radius of 6 cones. Make vectors with these values as follows:

```
R <- c(2.27, 1.98, 1.69, 1.88, 1.64, 2.14)
H <- c(8.28, 8.04, 9.06, 8.70, 7.58, 8.34)
```

   Recall that the volume of a cone with radius R and height H is given by $\frac{1}{3}\pi R^2 H$. Make a vector with the volumes of the 6 cones.

3. Compute the mean, median and standard deviation of the cone volumes. Compute also the mean of volume for the cones with a height less than 8.5.

F. Help pages

There are help pages for every function i R, but, admittedly, they are not always too easy to read for an inexperienced R user.

1. Try the command `?which.min`. This opens a help page in the lower right window of RStudio. What does the function do?

2. Try the commands

```
which.min(c(2,5,1,7,8))
which.max(c(2,5,1,1,8))
```

   Is the output as expected?

3. You must know the name of the function in order to open the help page as above. Sometimes (often, even) you do not know the name of the R functions; then google can often help you. Try, for example, to search the text `R minimum vector`.

G. Reading data from files and working with datasets

Suppose that your data is saved as a .csv file (csv stand for comma-separated values). We need to read the data into R. Consider an experiment where girth, height and volume has been measured for 31 cherry trees. The data are saved in the file cherry.csv.

1. Open the file `cherry.csv` in Excel (or a similar application), and make sure you understand the structure of the file.

2. Go to R, use the command

   ```
   cherry <- read.csv(file.choose(), dec='.', sep=';')
   ```

   Another window pops up, and you must click your way through to the relevant csv file.

   Once the dataset has been constructed, the name `cherry` appears in the upper right box in RStudio. Click the name, and you can see the content in the upper left box.

   Notice how you can change the characters for decimal comma and field separator according to what is used in the csv file, so the file is interpreted in the correct way (more about this below).

3. Whenever you have constructed a new dataset (or made changes in an existing one) it is extremely important that you check that it has been constructed correctly. The following commands may be useful for that purpose:

   ```
   head(cherry)
   plot(cherry)
   summary(cherry)
   ```

   Try the commands!

4. Did you get lists of minimum, maximum, mean, *etc.*, for all three variables?

   If not, it is because R has coded some of the variables as categorical variables rather than numerical variables (R reads a number with decimals as a text string). Luckily this can be changed with the dec option in `read.csv`. Try instead

   ```
   cherry1 <- read.csv(file.choose(), dec=',', sep=';')
   summary(cherry1)
   ```

   The question about coding is important, so let us say it a bit more specifically: A common problem is that one or more numerical variables have been coded as categorical variables because the decimal seperator has not been interpreted correctly. Then R reads a number with decimals as a text string. This causes all sorts of trouble, and the sooner you detect it, the better. The summary of numerical variable lists the minimum, maximum, mean, *etc.*, for the

variable. If, instead, you get information about how many times specific values occur in the dataset, then your variable is coded as a categorical value, and you have to change the `dec` option in your `read.csv` command.

5. (Optional) You can write the file name instead of `file.choose()`, such that you do not have to click your way through to the file. The path to the file must be given, starting from the current working directory. If the file is in the current working directory, then the relevant command is

   ```
   cherry2 <- read.csv('cherry.csv', dec='.', sep=';')
   ```

   You can change the working directory in the Session menu (Set working directory).

6. (Optional) Consider an experiment with tomatoes. Three different varieties and four different seed densities have been tested, and there are three replications for each of the 12 combinations. The yield has been registered for each of the 36 field plots. The data are saved in the file `tomatoes.csv`

   Create a dataset called `tomatoes`, say, in R, and inspect the data with `summary`. Which variables are quantitative and which variables are categorical?

H. Reading data from Excel files

You probably most often save your data in Excel (or something similar), and therefore need to "transfer" your data from Excel to R. There are several ways of doing this; here comes two if them:

(a) Read the Excel file directly into R with the function `read.xlsx`. This is the nicest solution; however problems now and then occur due to different versions of Excel and different computer systems. It also requires that you have the rights to install add-on packages on your computer (previously this was sometimes a problem on SCIENCE-PC's).

(b) Save your Excel sheet as a comma separated file (.csv), and use the function read.csv to read the file as described in Exercise G.1. Make sure that the decimal seperator and the field seperator in read.csv match those used to save the csv file.

Let us consider the same data as in Exercise G.1: Girth, height and volume has meen measured for 31 cherry trees. The data are saved in the file cherry.xlsx.

We first try method (a). Here is something that works on my computer:

1. First install the package xlsx via the Package menu (this requires connection to the internet). Second load the package via the Package menu. See Section 1 for more information about R packages.

2. Use the command

```
cherry3 <- read.xlsx(file.choose(), sheetIndex=1)
```

and choose the relevant file.

3. Inspect the dataset cherry3. In particular, make sure that all variable are coded correctly as numerical variables.

I. Working with datasets: Inspection, using variables, attaching

Consider the dataset cherry from Exercise G.1 or G.2.

1. *(Inspection of data)* Click the name `cherry` in the upper right box in RStudio (as you probably already did in Exercise 3.1). Then try the following commands (one at a time) and explain what you see:

```
head(cherry)
cherry
plot(cherry)
summary(cherry)
```

2. *(The $ syntax)* Try the following commands one at a time; notice that the first two give you an error message:

```
Girth
hist(Girth)
cherry$Girth
mean(cherry$Girth)
hist(cherry$Girth)
```

3. *(The `with` function)* Try the following commands:

```
with(cherry, hist(Height))
with(cherry, mean(Height))
```

4. *(Attaching a dataset)* Try the following commands one at a time:

```
attach(cherry)
Girth
mean(Girth)
hist(Girth)
plot(Height, Volume)
detach(cherry)
Girth
```

The point of questions 2–4 is that you need to tell R where to look for the variables. You do so either by using the $ syntax every time you need the variable (which quickly becomes tendersome), by using the `with` function "outside the command", or by attaching the dataset. If you attach datasets, you should be very careful not to have several datasets with the same variable names attached at the same time. This can be very dangerous because it is not always obvious which dataset is then used.

5. *(Structure of datasets)* Make sure you understand the structure of a dataset. More specifically, for example: What is the difference between a variable and a dataset? What do you see in the different rows of a dataset? What do you see in the different columns in a dataset?

----------End----------