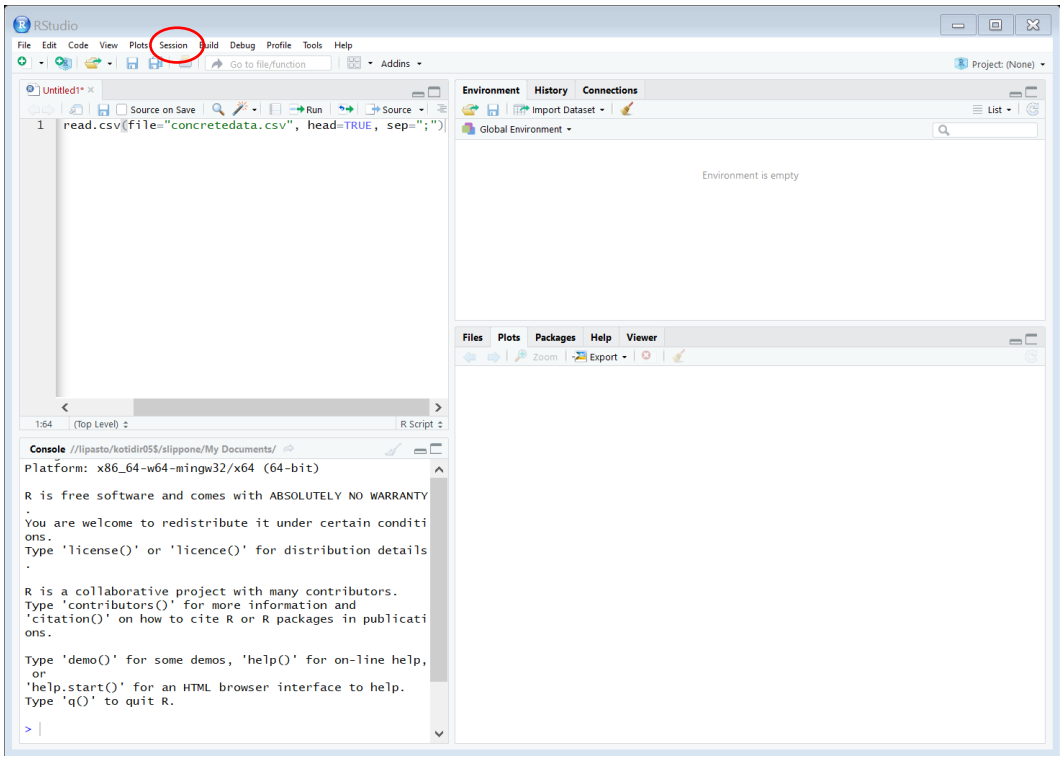
TOWARDS DATA MINING

EXERCISE 1. INTRODUCTION TO R

This exercise will introduce some basic functions in R. It is recommended to use RStudio for these tasks, because it is easier to use than plain R. If you don't have RStudio in your computer, you can download it freely from https://www.rstudio.com/ . But first, download R from https://cran.r-project.org/ .

When working with RStudio, you have to select first your working directory. It is easiest, if you have all necessary files in the same place. You can select the directory with Session-> Set working directory->Choose directory…, and after that the list of files in the directory should appear to lower right window. Note, that there you will find Files, Packages, Help and Viewer as well. The top left window is for script-writing (you can run the code either with Run (select an area with mouse, or run row by row) or Source, which will run everything in this script. Your data will appear to top right window. And the command window is on bottom left.



The material in this exercise will teach only the simple usage of R. Write a short report after the exercise and add there the tasks indicated with red font in this document.

1. GETTING YOUR DATA INTO R

To get started, you have to read your data from the file. First, some information about how to handle data files in R. These are only some examples and the task starts a bit later.

If you data is in .csv format, you can use the command:

```
mydata <- read.csv("filename.csv")
```

or for tabular separated .txt files:

```
mydata <- read.table("filename.txt", sep="\t")
```

With additional arguments, you can add more details. Do you have variable names in the first row? Look at your decimals, is there ',' instead of '.'? What if there are missing values? How they are indicated?

```
mydata <- read.csv("filename.csv", header=FALSE)
```

```
mydata <- read.table("filename.txt", sep="\t", header=TRUE)
```

```
mydata <- read.csv("filename.txt", sep=";", dec=",", header=TRUE, na.strings=
c("NA", " "))
```

Notice that '#' is used for comments in R (similarly as '%' in Matlab)!

You can see your data in the Environment window. If there are objects that you do not need, you can remove them with

```
rm(x)
```

When you end your work, you can save your entire workspace

```
save.image()
```

or selected objects to a file.

```
save(variablename, file="filename.rdata")
```

And when you are ready to work again, you can load them back simply by writing:

```
load("filename.rdata")
```

If you want to save a data matrix to .txt file, use the command

```
write.table(mydata, "testfile.txt", sep="\t")
```

In this exercise read the data by using

```
mydata <- read.csv(file="concretedata.csv", head=TRUE, sep=";")
```

If you enter wrong decimal type, you may end up having problems

```
wrongdata <- read.csv(file="concretedata.csv", sep=";", dec=",")
```

2. SIMPLE DATA ANALYSIS

Now look at your data more closely. First, check that your data is data.frame type:

```
is.data.frame(mydata)
```

You can find the dimensions of a data matrix

```
dim(mydata)
```

This command gives you first 6 rows of the data

```
head(mydata)
```

You may want to see more

```
head(mydata, n=10)
```
or
```
head(mydata, 10)
```

and similarly the last six (or more) lines

```
tail(mydata)
```

With this command you see the data types of each variable

```
str(mydata)
```

Let's look the second data the same way

```
str(wrongdata)
```

```
summary(mydata)
```

If your data set had the variable names at the first row, you can see them by writing

```
names(mydata)
```

You can also use commands

```
colnames(mydata)
```
or
```
rownames(mydata)
```

If you did not have the names, you can enter them manually. Let's make an data frame containing only zeros and utilize the names of the mydata:

```
mymatrix<-data.frame(matrix(0,5,12))
```

```
mymatrix
```

```
names(mymatrix) <- names(mydata)
```

```
mymatrix
```

When you data is data.frame type, you can refer to each variable with their name by writing

```
mydata$water
```

3. BASIC STATISTICS

You can calculate the correlations between variables in your data set with

```
cor(mydata[,7:10])
```

Note that the correlation is meaningful only for continuous variables.

If you have missing values in your data, you may need this argument

```
na.rm=TRUE
```

It removes the missing values while using the selected function. You can calculate the mean for one variable (vector) with

```
mean(mydata$water)
```

If you need to calculate means or medians for the data matrix, you use function `apply()` instead, and select the dimension 1 for row wise and 2 for column wise. For rows, you may not want to print out the whole data.

```
apply(mydata[,-4], 1, median)
```

```
apply(mydata[,-4], 2, median)
```

Here we left out the column “grade”, because median cannot be calculated to factors.

Do you want to learn more about the function, or maybe you need help with it?

```
?apply
```

4. SELECTING DATA

When you want to select (or view) only specific columns, you can refer to their dimensions (let’s select only ten first rows)

```
mydata[1:10,2:4]
```

Note that you need to specify a vector with c() if you do not want all the columns between 2 and 4

```
mydata[1:10,c(2,4)]
```

What happens, if you select all observations with value <160 for variable water?

```
mydata$water<160
```

Compare the result to this one:

```
mydata[mydata$water<160,]
```

You can continue by selecting both rows and columns

```
mydata[mydata$water<160,c(1,3)]
```

or refer to columns by their names

```
mydata[mydata$water<160,c("fly_ash","coarse_aggregate")]
```

Other way to do it:

```
water160 <- mydata$water < 160

cols <- c("fly_ash", "coarse_aggregate")

mydata[water160, cols]
```

If you need a subset of the original data, it can be selected with subset() function.

```
subset(mydata, water<160, c("fly_ash", "coarse_aggregate"))
```

Maybe you need only observations that have maximum value in class

```
subset(mydata, class==max(class))
```

If you wanted to select only variables, there are two ways to do it

```
newdata1<-subset(mydata, , c("fly_ash", "coarse_aggregate"))
```
or

```
newdata2<-subset(mydata, select=c("fly_ash", "coarse_aggregate"))
```

5. DATA VISUALIZATION

Next, you can produce tables or visualizations from the data. For categorical data, you may find the tables to be useful. For single variable

```
table(mydata$grade)
```

or two variables

```
table(mydata$grade, mydata$class)
```

For continuous variables, plot visualization is more useful

```
plot(mydata$age)
```

```
plot(mydata$cement,pch='.')
```
and add a line to graph

```
lines(mydata$cement)
```

Scatter plots may show interesting relationships

```
plot(mydata$water, mydata$fine_aggregate)
```

You may add more information to the figure

```
plot(mydata$water, mydata$fine_aggregate, xlab="Water content", ylab="Amount of
   fine aggregate", main="The relationship between water and fine aggregate")
```

Attach this figure to your exercise report. You can save it by selecting “Export” above the figure window, or simply Copy to Clipboard and add to your document.

There are lots of graphical parameters you can set in the figure

```
?par
```

Histograms are a good way to visualize data

```
hist(mydata$fine_aggregate, breaks = 15)
```

Another popular method is a boxplot figure

```
boxplot(mydata$fine_aggregate)
```

```
boxplot(mydata$fine_aggregate, mydata$water, mydata$cement)
```

You can extend the capabilities of R by adding new packages to your computer. You can call them with

```
library(psych)
```

and utilize functions that other people have created

```
describe(mydata)
```

If you don’t have this “psych” package yet, you need to install it first.

Lastly, tell about your previous experience with R or RStudio.