Introduction to R

Objective: This exercise is an *optional* pre-exercise. You can use the exercise to check that you have the sufficient coding skills, as well as ensuring that you have your integrated development environment (IDE) setup.

The objective is to setup your R IDE. Additionally, the objective is to get you acquainted with R, as well as the exercise format of the course. Upon completing this exercise it is expected that you:

- Have setup your R IDE.
- Can write and execute basic R code.
- Understand how data can be represented as vectors and matrices in R
- Understand basic operations and plotting in R.
- Understand the format of the exercises.

Material: For this exercise, you will need the 02450 toolbox. See the optional section in the end of the document for more material and tutorials on coding.

Piazza discussion forum: You can get help by asking questions on Piazza: https://piazza.com/dtu.dk/fall2020/02450

0.1 How to do the exercises

The exercises in the course are structured such that you go through an exercise document like this one.

This exercise (Exercise 0) is an optional exercise that you can use to make sure have the sufficient coding skills and have setup the your IDE—that is: the exercise is not introducing any course material, and is meant as a help in preparing for starting with the course and ensuring that you fulfill the course requirements. If you encounter any problems in doing this exercise—in setting up your IDE or installing the 02450 Toolbox—make sure to get in touch with a teaching assistant after the first lecture at the exercises. The first real course exercise (Exercise 1) will be done after the first lecture. Note that if you have difficulties following the scripts, catching up on your progamming skills are to be done as a self-study exercise (see optional material in the end of this document).

The exercises are centered around running and understanding a series of scripts provided in the 02450 Toolbox. The exercise descriptions guide you through the scripts and help you understand what is going on in them. You wont have time to code everything from scratch for the exercises, and so it is important that you get familar with this workflow centered around the existing scripts. The scripts that you go through in the exercises provide the basis for your work the project you will work on, where you will be able to re-use large parts of the code you have worked with in the exercises. However, for the reports you will have to code more yourself and tailor the scripts to your dataset and problem.

The exercises are structured as smaller numbered sections. When a certain section concerns a particular script, it will be stated and their number will match. For instance, the first script you will run (in a little while) is called `ex0_4_3.R` and corresponds to the section 0.4.3 in this document.

0.2 Getting started with R

R is an open-source software environment for statistical analyses which is extensively used in (classical) statistics and machine learning. The homepage of the project is `www.r-project.org`. The installation obtained from r-project.org is very basic, and we recommend downloading the `RStudio` distribution which contain many of the packages we will use as well as a graphical user interface. To download RStudio, go to `http://www.rstudio.com/products/rstudio/download/` and download the version appropriate for your operation system.

In the following we will assume RStudio is installed and used to run commands and edit files. In addition to RStudio several additional packages need to be installed. These can be installed from RStudio by going to `Tools -> Install packages..` and input the package name. The following packages need to be installed: `R.matlab, mltools, data.table, readxl, FNN, gplots, cvTools, neuralnet, randomForest, mclust, mixtools, sm, SnowballC, scatterplot3d, rgl, tm, sos, lsa, glmnet`.

If you are used to Emacs, you might want to work with R in Emacs. A plug-in called ESS (Emacs Speaks Statistics) is available for Emacs. This plugin allows you to execute R code from within Emacs. The homepage for the plug-in is `http://ess.r-project.org/`.

0.3 Installing the 02450 Toolbox

The course will make use of several specialized scripts and toolboxes not included with R. These are distributed as a toolbox which need to be installed.

0.3.1 Download and unzip the 02450 Toolbox for R, `02450Toolbox_R.zip` . It will be assumed the toolbox is unpacked to create the directories:

```
<base-dir>/02450Toolbox_R/Tools/     # Misc. tools and packages
<base-dir>/02450Toolbox_R/Data/      # Datasets directory
<base-dir>/02450Toolbox_R/Scripts/   # Scripts for exercises
```

For the exercises, you should work on the example scripts in `<base-dir>/02450Toolbox_R/Scripts/` (notice the scripts are labelled according to exercise number) and not try to write the scripts from the bottom up.

0.3.2 To finalize the installation you need to update your path. To do this run the file `<base-dir>/02450Toolbox_R/setup.R` and ensure you do not get any errors.

0.4 Basic operations in R

0.4.1 *Console* The IDE provides both a console and an editor. The console can be used to execute code fast and interactively, and is often used when debugging your code. You can define and display variables, plot data, and even define functions on-the-fly using interactive console. It is useful when you debug your code or experiment with small chunks of code, mathematical expressions, etc. In RStudio, find the Console (the console) and try typing: `a=9`. The variable `a` should then appear in your Workspace If you write `a` in the console, the value of `a` is displayed.

0.4.2 *Editor, scripts and functions* The editor is used to write longer scripts and functions. You can inspect the working directory in R by typing `getwd()`. You can specify the working directory to be in the directory with path given by `path` typing `setwd("path")` (note that R uses `/` as delimiter for directories.). Open a script and write the following lines

```
a = 3
b = 4
print(a*b)
```

and save the file as `myscript.R`. Run the script in R by typing `source("myscript.R")` at the command prompt.

Make a new file `myfunction.R` and write the following function

```
myfunction <- function(n){
    #myfunction will return an array of values ranging
    #from 1 to the input integer n.
    x = 1:n
  }
```

Note that the symbol `#` indicates comments and the rest of the line is not evaluated. Run the function in R by first running `source("myfunction.R")` to make R recognize the function in the workspace. Then run `y = myfunction(9)` and observe the variable `y`.

R help for a function is obtained by typing `?<function name>` in the R command prompt. In practice, the fastest and easiest way to get help in R is often to simply Google your problem. For instance: `"How to add legends to a plot in R"` or the content of an error message. In the later case, it is often helpful to find the *simplest* script or input to script which will raise the error.

0.4.3 *Making sequences* We often need to use a sequence of numbers. Observe the various results obtained when running `ex0_4_3.R`.

0.4.4 *Indexing* We often need to retrieve or assign a value to a certain part of a vector or matrix. Inspect `ex0_4_4.R` to see how to do indexing in R.

0.4.5 *Matrix operations* We will use various matrix operations extensively throughout the course. Inspect `ex0_4_5.R` to see how to do some common ones in R.

0.5 Basic plotting

It is important to visualize the results you obtain. In this part of the exercise, we will make two plots, a simple, and a more elaborate example. Going forwards, try to keep the elements of the figure made in 1.5.2 in mind as a model for minimum required considerations when making a figure.

0.5.1 Inspect `ex0_5_1.R` to see how to do basic plotting in R. Try modifying the script to display a cosine curve for values in the range $[0, \pi]$.

0.5.2 Now, let us consider a slightly more advanced plot. Inspect `ex0_5_2.R`. We simulate measurements from two sensors (sensor 1 and sensor 2) for a period of 10 seconds, and we say that the sensors output measurents in millivolt. Since the two measurements are done at the same time, we want to do plot them in the same figure. Furthermore, we want to make sure that the axis are labelled correctly both with a name and a designation of the unit of measurement. We also need to make sure that it is easy to see which curves comes from which sensor. Lastly, we ensure that the axis are readable (large enough), both when looking at them in the IDE, but also in an exported version that we might use in a report about the sensors.

0.6 OPTIONAL

The following website contains introductory tutorials to R:

- `http://www.cyclismo.org/tutorial/R/` (especially “Data Types”, “Basic Operations”, and “Plotting”)

References