# Thesis With R Markdown

Mehdi Jaiem

7/7/2020

## 1 Introduction

This Document generated with an *R Markdown* will include a small recapitulation concerning different aspects and tools that we based ourselves on during the first 4 weeks. We will not cover the theoretical but the practical and most relevant points that we encountered.

## 2 RSTUDIO and R

As we may actually know, data sets are covered from different data sets providers and these data sets can be accessed through what we call **packages** such as the on found provided by *Cran*, *BioConductor* and *Github*. These repositories offer a large library of data sets har cover different topics. Of course we should use and benefit only from the packages that are suited for our analysis. As a search engine for packages, we also have RDocumentation which is a search engine for packages and functions from CRAN, BioConductor, and GitHub.

### 2.1 Packages Installation and manipulation in RSTUDIO

- **Package Installation** As stated, before accessing a package we have to install it after checking if it is suitable for our search. To do so the following method is used: install.packages("desired package name") Installing multiple packages at once is done with the following command: install.packages(c("package1", "package 2", "package 3", …)). Notice that the graphical interface of RStudio can also be used to accomplish the above task.

Sometimes, the standard package installation way is not enough to access what we want. In fact, repositories such as Bioconductor or GitHub have their own way to do so.

- **Install packages from Bioconductor**

For **BioConductor** it consists in getting the necessary functions to install from BioConductor. Then call the main install function to execute the installation of the desired packages. It's like downloading a program from the internet then clicking on the executable folder when the download is done.

```
1. source("https://bioconductor.org/bioclite.R")
2. bioclLite("desired package name")
```

- **Installing packages from Github**

Now coming to **Github**. The procedure consists in finding the package name and the author of the package. Then we simply follow the steps under:

```
1. install.packages("package name")
2. library(package name)
3. install_github("author/package")
```

- **Which packages are installed**

Sometimes and after a long use of R, we may forget which packages have been installed and check after a specific package and see if it's available or if it has to be installed. The commands responsible for that are:

```
installed.packages()
or
library()
```

- **UPDATE PACKAGES**

Since we are working with data sets and as data analysts we want to exploit various sets of data of data to reach deductions and concrete and useful conclusions, it is wise to update the packages and have access to the latest versions. The commands targeting this are listed in what follows

**old.packages()** : List all the packages that can be updated.

**update.packages()**: Updates all packages that need an Update

**install.packages("desired package name")**: updates a desired package

- **REMOVE PACKAGES**

To remove packages the *remove* command in RStudio can be used. It simply deletes one, several or all the packages depending on the instruction provided.

**remove.packages()**: removes all installed packages

**remove.packages("desired package name")**: removes a desired package

- **Check R Version**

Check R Version and all installed packages with: version or sessioInfo()

- **HELP**

Provides the different applicable instruction while using a specific package. the *help* command can also be used to ask RStudio about detailed clarifications concerning tools, functionalities and other program features.

```
helps(package= "desired package name")
```

## 2.2 R Projects

Generally speaking, A project is a folder containing all the necessary files we worked with during a specific project.It creates a folder for you and now you have a place to store all of your input data, your code, and the output of your code. This is helpful when having different separate projects.

- **Creating a project** There are three ways to make a Project: 1- From scratch - this will create a new directory for all your files to go in 2- From an existing folder - this will link an existing directory with RStudio 3- From version control - this will "clone" an existing project onto your computer (Don't worry too much about this one, you'll get more familiar with it in the next few lessons)

## 3 Github

Git is a free and open source version control system. It was developed in 2005 and has since become the most used version control system around. It is useful for tracking the evolution of projects, written codes and any type of private or public task, since it allows you to commit and save every change, enabling the possibility to recover previous versions of any data type or project. Moreover, it strengthens the bonds between every community member interested in the same topics and subjects as the owner of the *Git repository.* As we may have seen for example, an author provides a written package to the community so that it can be useful for it and why not build upon it future concepts and developed ideas that will be also shared and so on. The most commonly used commands to use Github are: * **commit** : To commit is to save your edits and the changes made. When you commit a file, typically you accompany that file change with a little note about what you changed and why. Commits are mainly used to track the different saved versions on Github, which remarkably allows the user to recover a previous versions if some file is corrupt, full with errors or presenting some anomalies. To do so the following commit instruction can be used:

```
commit -m "desired commit message"
```

- **Push**: Stores the committed files in the corresponding Github repository so now everybody has access to your edits.
- **Pull**: Since one of the goals of Github is to share your experience with the community or with your group of work, it involves then possible changes in online repository. This involves the fact that you local repository on you computer may be outdated. To activate the update we use the *pull* command

## 4 R Markdown

With the **R markdown** we are going to discover how to write and generate a pdf script including different features of R Markdown. To display some of these features you can go to *help* and select *Markdown Quick Reference*

### 4.1 Cars CODE

#### 4.1.1 displaying scripts

This section presents an easy way to manipulate written scripts or data sets and display the results or the intended content. To call a code (or script) we do as shown.

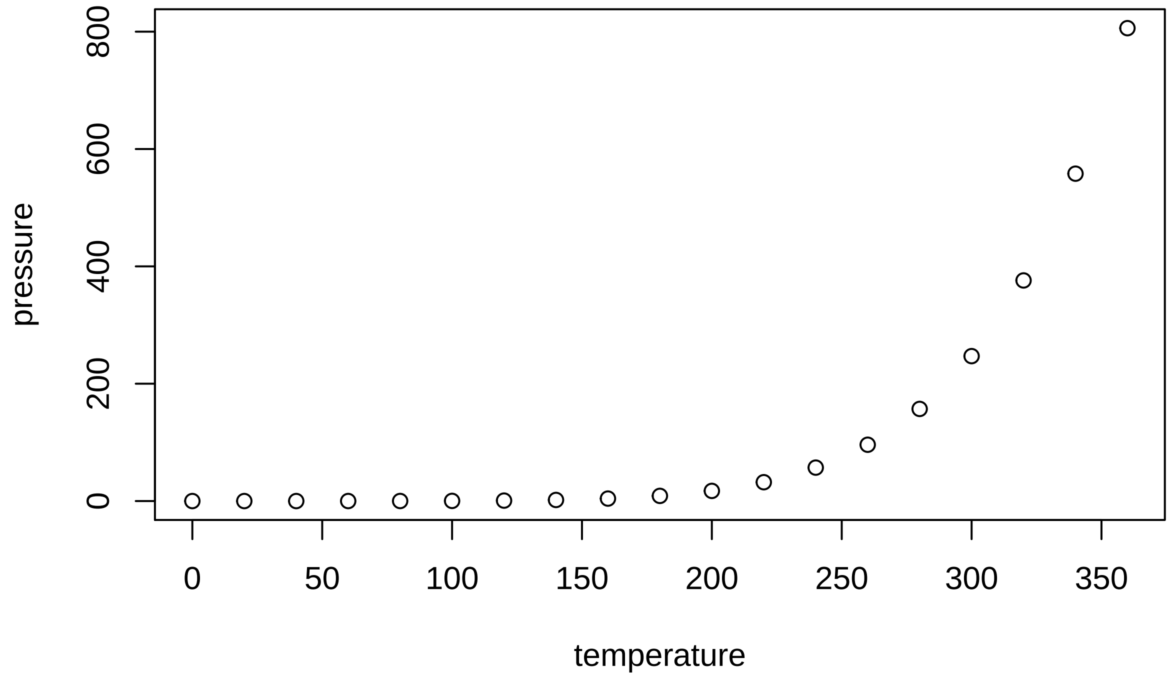
```r
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00  
```

*Summary* displays a table of different aspects in our set with different values like: Min, Max, Mean, etc..

#### 4.1.2 Including Plots

We can also embed plots, for example:



- Note that the *echo = FALSE* parameter was added to the code chunk to prevent printing of the R code that generated the plot.

### 4.2 Linking or inserting a website

This is a link to the rstudio website: RStudio

### 4.3 Image Insertion

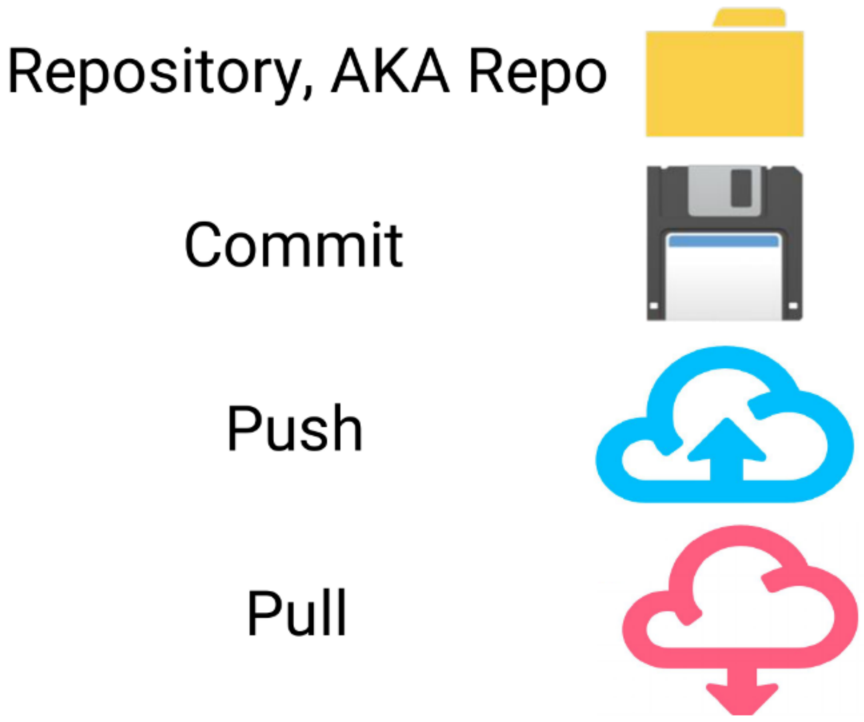


Figure 1: GitBash main commands