Laboratory 1: An Introduction to R

Felipe Bravo Oviedo
Universidad de Valladolid, Spain

In this Lab we will learn how to:

- Install R, RStudio and Note++
- Understand basic R codes
- Install and invoke R packages
- Establish a working directory and load a data set into R using RStudio
- View the raw data in R

Installing R, RStudio and Note++

You can download R from https://www.r-project.org/ and install in your computer or laptop. It is possible to conduct analysis on R directly but it is more user friendly if you run it via Rstudio that can be downloaded from https://www.rstudio.com/ Once both program are downloaded and installed in your computer you can open R by clicking on the start menu the RStudio icon. Then you should see a screen as the image below (fig 1). You can see from the top left clockwise: the scripting window, the objects (data sets,…) window, files and plots window and the command line console

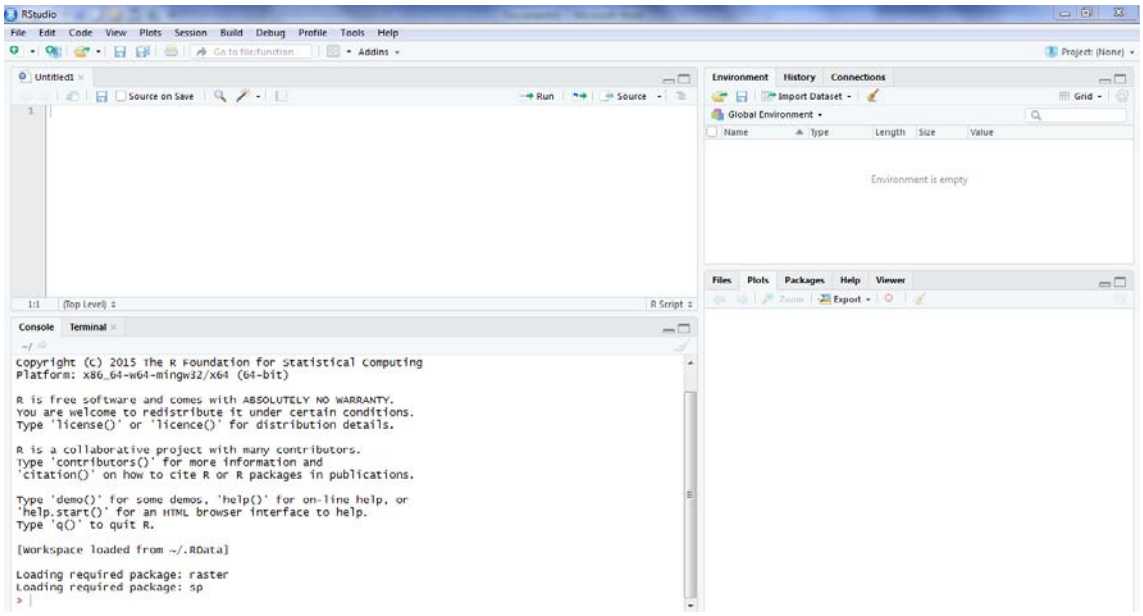

Fig. 1 Apparence of RStudio with the four windows you can see from the top left clockwise: the *scripting* window, the *objects* (data sets,…) window, *files and plots* window and the *command line* console

Besides R and Rstudio you can find useful to have in your computer Note++ that it as a free code editor that ca be downloaded from https://notepad-plus-plus.org/

Understanding basic R Code

R has a steep learning curve but once you master the basic of the R coding you will benefit with a powerful tool for your work. The best way to learn R is to take a sample code (there are a lot of on the web and this is one of the best things with R: the huge user community around the world) think about it and modify it by changing parameters to see how the results change, take a lot of notes and comment as much as you can the script for your benefits and the others that can learn from your work. If you are tired, relax and come back later with renewed energy. Remember that learning how to code is a long run and we are experts in our domain and use code as a tool not as an objective itself.

We will start with some comments about R:

- R is case sensitive so we should be aware about the use of capital letters because 'DATA' is different that 'data' or 'Data'.
- When a line starts with the hashtag symbol (#) your computer will skip it and it will jump to following line code so # means that this line is a comment. You should comment your code (script) as much as you can because your closer collaborators in one year will be you and, really you will not remember why you code something unless you have a clear comment. Also comment is useful to collaborate with other colleagues in your team.

```
# means that this line is a comment so R doesn't consider it a command
# it's useful to comment the program to understand what are you doing
# COMMENT AS MUCH AS YOU CAN
```

- If you need help about a command you can write ?*** that will connect yous to internet and search help about the command (***)
- Also it is important to remember when you are defining a path that you should write / instead of \ and write the path always within " "
- Other important thing is to know that in R <- means =
- R is based on different packages so for some actions you will need to install (the first time) and load (always) the adequate package

Install and invoke R packages

You can install a package directly from RStudio by clicking on Tools -> Install Packages and then follow the screen (fig 2) or by coding the following line:

```
install.packages('name_of_your_desired_package')
```

Always your package should be between ' ' and remember that this action is needed only the first time you want to use the package but when you install a new version of R is almost sure that you will need to reinstall your packages.

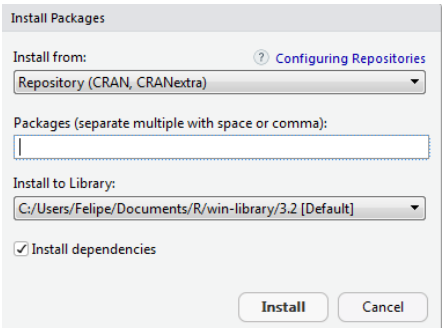

Fig. 2. Install Packages tool in RStudio

Once you have your package installed you need to invoke it in order to be able to use in your script. To invoke a package already installed you must code as follow:

```
require('name_of_your_desired_package')
```

or better

```
library('name_of_your_desired_package')
```

but maybe your prefer test if the package is already installed and if not install it and then load the package. You can do this by this short code:

```
if(!require('name_of_your_desired_package'))
        install.packages('name_of_your_desired_package')
library('name_of_your_desired_package')
```

You can check the packages already installed in your computer by clicking in the *packages* tab on the *files and plots* window (fig. 1)

Establishing a working directory and loading a data set into R

It is very useful that we define a working directory for our ongoing project. By defining our working directory we allow R to know where to find, open and save files for our project. To define your working directory you can go to the *files and plots* window and there click on the *files* tab (fig. 1) and then click on the *more* button and then *set as working directory*.

Also you can define it by typing the following code:

```
setwd('C:/your_desired_working_directoryR')
```

Remember always write / and not \

Now you can check wich is your working directory by typing:

```
getwd(' ')
```

Now we are ready to lead our dataset into R. One of the most interesting features of R is its flexibility to import different file types. We will start by importing data in csv (*comma separate values*) format. The general structure of the code to import csv files (with the variable names as header) is the following

```
data0<-read.csv('your_file.csv',  header=TRUE)
```

In some cases the comma is use as decimal separators and the the semicolon (;) is used as delimiters so then we should use the following code:

```
data0<-read.csv2('your_file.csv',  header=TRUE)
```

If you have other formats for your data (as plain text, .txt) you need to change slightly your code or in some cases (as excel files, .xls or .xlsx) you will need to install a package. See the following code lines as example:

```
# importing a txt dataset
data1<-read.table('your_file.txt',  sep='\t', dec=','  ,header=TRUE)

# importing a Excel dataset
if(!require('xlsx')) install.packages('xlsx') # test if the package is already
library('xlsx')
data3 <- read.xlsx('your_file.xlsx', sheetName = 'sheet1')
# You can import xls files and also change the sheetName according to your #computer,
i.e, in Spanish we use Hoja1 instead of sheet1
```

Now you can try the previous action with the files called dendro.csv, dendro.xlsx and dendro.txt

Also it is possible to read a file directly from the web by using this code as template:

```
#it is possible import data from a web page
data4 <-
read.table('http://sostenible.palencia.uva.es/sites/default/files/manuales/datos1.txt',
sep="\t", dec=".", header=TRUE)

# or better
web <- ('http://sostenible.palencia.uva.es/sites/default/files/manuales/datos1.txt')
data4 <- read.table(web, sep='\t', dec='.', header=TRUE)
```

Viewing the raw data

Once we have the data uploaded to R it is very important to check if the data are properly stored in R. To do that we have different options (1) see the whole dataset, (2) see the first observations or (3) see the final observations. Additionally, you should check if the column names are properly defined.

The first 1000 observations of the dataset can be observed by using this code:

```
# to view the top 1000 cases of a data frame
View (name_of_the_object)
# the name of the object should be, according to our previous actions: data0
```

But if you want to be the first (*head*) or the last (*tail*) 6 observations you must use the following code:

```
# to see the first 6 cases of a data frame
head (name_of_the_object)
```

```
# to see the last 6 cases of a data frame
tail (name_of_the_object)

# the name of the object should be, according to our previous actions: data0
```

By adding the number of observations you can see n cases instead of the 6 that appears by default:

```
# to see the first 10 cases of a data frame. You can replace 10 for the number of
# observations that you want to see
head (name_of_the_object, 10)
tail (name_of_the_object, 10)
# the name of the object should be, according to our previous actions: data0
```