Introduction to R

Miguel-Angel Canela, IESE Business School

October 27, 2017

Contents

General introduction

`R` is a language which comes with an environment for computing and graphics, available for Windows, Linux and Macintosh. It includes an extensive variety of techniques for statistical testing, predictive modelling and data visualization. `R` can be extended by hundreds of additional packages available at the **Comprehensive R Archive Network** (CRAN), which cover virtually every aspect of data management and analysis. As a rule, this course does not use one of these packages unless it makes a real difference.

`R` has been gaining acceptance in the business intelligence industry in the last years: today we can use `R` resources within data management systems like Oracle, or resource planning systems like SAP. They are also available in cloud environments like Amazon Web Services (AWS), Microsoft Azure or IBM Data Scientist Workbench (so far completely free). Although facing strong competition from Python, `R` is still the default analytics tool in many well known companies (see Bion *et al*, 2017, for an example).

It will be very easy for you to download `R` from `www.r-project.com` and install it in your computer. This provides you with a graphical user interface (GUI), called the **console**, in which we can type or paste our code. The console works a bit different in Macintosh and Windows. In the Windows console, what we type is in red and the `R` response is in blue. In Macintosh, we type in blue and the response comes in black. When `R` is ready for our code, the console shows a **prompt** which is the symbol "greater than" (>). With the `Return` key, we finish a line of code, which is usually interpreted as a request for execution. But `R` can detect that your input is not finished, and then it waits for more input, showing a different prompt, the symbol "plus" (+).

In an `rmarkdown` document like the one that you are reading, this is usually seen as follows.

```
2 + 2
```

```
## [1] 4
```

If we type `2 + 2`, we get the result of this calculation. But, when we want to store this result, we give it a name, as follows.

```
a <- 2 + 2
```

Note that the value of `2 + 2` is not printed now. If we want it to be printed to the screen, we have to ask for that explicitly.

```
a
```

```
## [1] 4
```

We can replace all the "arrows" (<-) by equal signs, and nothing will change. Nevertheless, it is recommended, to the beginner, to use the arrow system, to avoid mistakes. `x <- 2 + 2` is read `assign("x", 2+2)`. We can also write `2 + 2 -> x`, but `2 + 2 <- x` does not make sense.

Interacting with R

The `R` console is not user-friendly, so you will probably prefer to work in an interactive developer environment (IDE). **RStudio** is the leading choice and, nowadays, most R coders prefer RStudio to the console. In RStudio, you have the console plus other windows that may help you to organize your task.

Besides working directly in the console or in an IDE like RStudio, a third approach is based on the **Notebook** concept. In a notebook, you enter your code as the **input**, getting the results as the output. You can include in the notebook graphcs that, in the console or in RStudio, would appear in a separate **Graphics Window**. In the notebook field, the two main competitors are **Jupyter** (`jupyter.org`) and **Apache Zeppelin** (`zeppelin.apache.org`). These two are multilingual, that is, can be used for other languages, besides `R`.

Learning about R

There are many books for learning about `R`, but some of them are too statistically-oriented, so they may not be for you. Kabacoff (2010) is a popular choice. If you want a thick book, Crawley (2012) will do. Also, Wickham & Grolemund (2016) is excellent, but strongly based on a set of packages developed by the authors (both at RStudio). These packages are used by many data scientists, but this course skips them to make it shorter.

There is also plenty of learning materials in Internet, including MOOC's. For instance, there is a big supply of (free) Data Science courses in the **Big Data University** (`bigdatauniversity.com`), including an *R 101 course*. Those willing to carry out deeper study can find in **Coursera** a pack of courses called *Data Science Specialization* (`www.coursera.org/specializations/jhu-data-science`), based on `R`. **DataCamp** offers a wide range of free Data Science courses, not all of the same quality.

Objects in R

`R` is **object-oriented**, with many classes of objects. I briefly comment in this lecture three examples. First, we have **vectors**. A vector is an ordered collection of elements which are all of the same type. Vectors can be **numeric**, **character** (string), **logical** (TRUE/FALSE) or of other types not discussed in this course. The following three examples are numeric (`x`), character (`y`) and logical (`z`), respectively.

```
x <- 1:10
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
y <- c("Messi", "Neymar", "Cristiano")
y
```

```
## [1] "Messi"     "Neymar"    "Cristiano"
```

```
z <- x > 5
z
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Some comments on these examples:

- The expression `c(a,b)` packs the elements `a` and `b` as the terms of a vector. This is only possible if they are of the same type.
- The quote marks indicate character type.
- An expression like `x > 5` is translated as a logical vector with one term for each term of `x`.

The first term of the vector `x` can be extracted as `x[1]`, the second term as `x[2]`, etc. A **factor** is a numeric vector in which the values have labels, called **levels**. Factors look very natural to statisticians (stat packages, like SPSS or Stata use similar systems), but weird to computer scientists. We can transform any character vector into a factor with the function `factor`.

```
y_factor <- factor(y)
y_factor
```

```
## [1] Messi     Neymar    Cristiano
## Levels: Cristiano Messi Neymar
```

A **list** is like a vector, but it can contain objects of different type. Elements of a list are identified as `L[[1]]` (double brackets) or as `L$L1`, with the name of the list followed by the name of the element, separated by the dollar sign.

```
L <- list(L1=1:10, L2=c("Messi", "Neymar", "Cristiano"))
L[[1]]
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
L$L2
```

```
## [1] "Messi"     "Neymar"    "Cristiano"
```

Subsetting

Extracting parts of `R` objects is called is called **subsetting**. Vectors, and also the data frames discussed in the next lecture, can be subsetted in an various ways. Some examples follow.

```
x[1:3]
```

```
## [1] 1 2 3
```

```
x[x>=5]
```

```
## [1]  5  6  7  8  9 10
```

Functions

`R` is a fully functional language. The real power of `R` comes from defining the operations that we wish to perform as **functions**, so they can be applied many times. For instance, import and export are usually managed by read/write functions. A simple example of the definition of a function follows. More complex examples will appear later in this course.

```
f <- function(x) 1/(1-x^2)
```

When we define a function, `R` just takes note of the definition, accepting it when it is syntactically correct. The function can apply it later to different arguments.

```
f(0.5)
```

```
## [1] 1.333333
```

But, if we apply the function to an argument for which it does not make sense, `R` will complain. The complaint can come in various ways. For instance, even if the argument supplied is not right, we get a response in the following example, but, trying `f("Mary")`, we would get an error message "Error in x^2 : non-numeric argument to binary operator".

```
f(1)
```

```
## [1] Inf
```

References

1. R Bion, R Chang & J Goodman (2017), *How R Helps Airbnb Make the Mostof Its Data*, `https://peerj.com/preprints/3182`.
2. MJ Crawley (2012), *The R Book*, Wiley. Free access at 'ftp://ftp.tuebingen.mpg.de/pub/kyb/bresciani/Crawley%20-%20The%20R%20Book.pdf
3. RI Kabacoff (2010), *R in Action*, Manning. Free access at `http://kek.ksu.ru/eos/DataMining/1379968983.pdf`.
4. H Wickham & G Grolemund (2016), *R for Data Science*, O'Reilly. Free access at `http://r4ds.had.co.nz`.