MBA
USP
ESALQ

Introduction to the R Programming Language

Rafael de Freitas Souza

Why Learn a Programming language?

- Learning to program is very important when we want to understand data! Plus: it is important to ensure that we can interpret the data so we can transform them into information!

- If you work or want to work with data, programming is an extremely relevant skill

A fair question would be :

"I already work or wish to work with data, and I already have access to them via Windows, OSx, Android, iOS, Chrome, Mozilla, Edge, Safari...

Still, do I need to learn to program?"

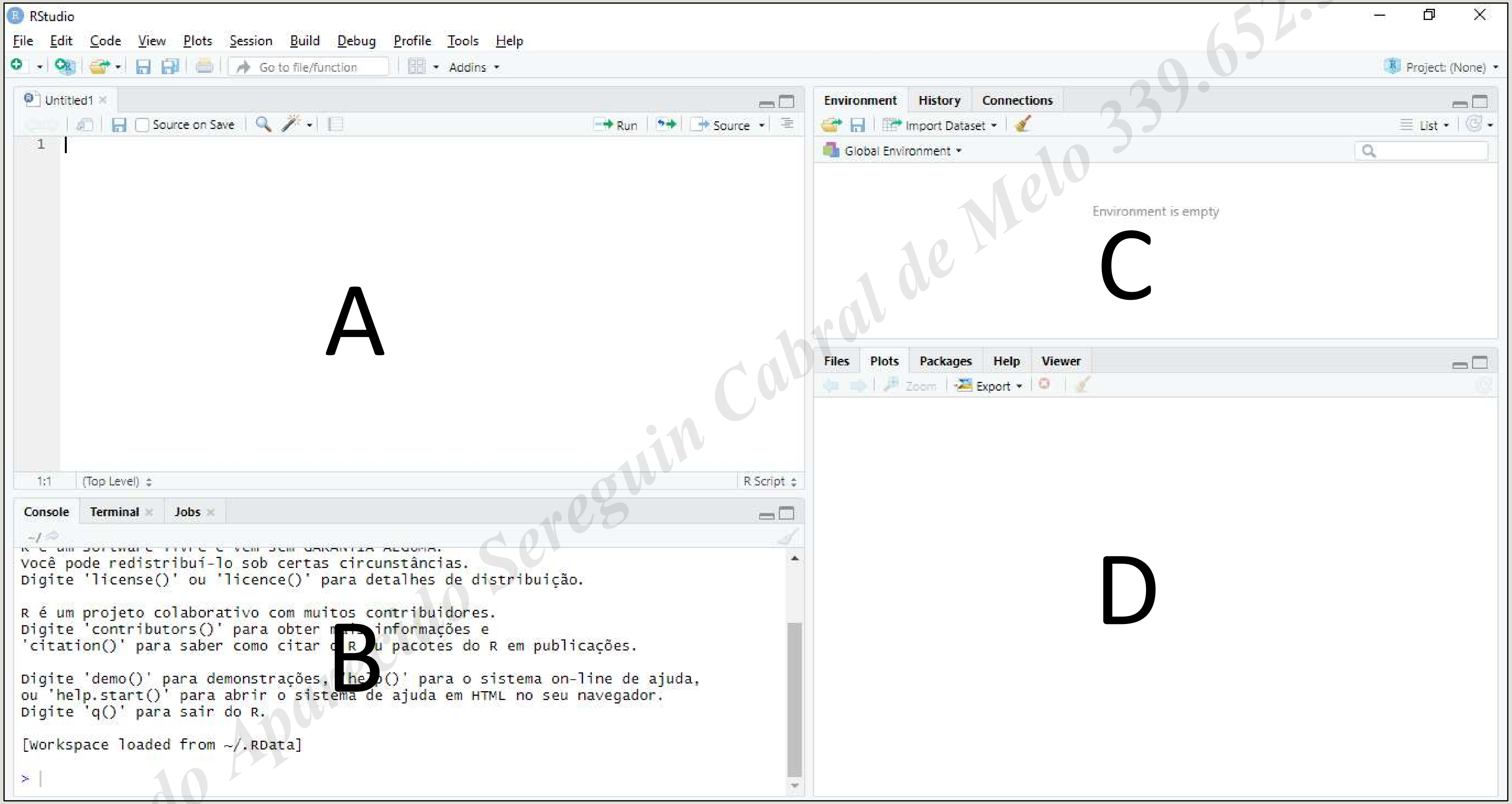
The Limitations of a *Graphic User Interface* (GUI)

- Reproducibility
- Automation
- Communication

Why R?

About R

- The R language emerged in 1995, derived from the S language, and is object-oriented.
- It has several packages (over 17 thousand) with advantage for the application of the Advanced Statistics and a vast support community, in addition to strong capacities focused on Data Science.
- Comprehensive R Archive Network (CRAN) is the R language repository in which each user can contribute to new packages (collections of functions in R with a compiled code). These packages can be easily installed with a command line.
- Recommended readings for those who are not yet familiar with the R language and wish to delve deeper:
  - Hands-On Programming with R (Grolemund, 2014);
  - R for Data Science (Wickham & Grolemund, 2016);
  - Ciência de Dados com R (Oliveira, Guerra & McDonnell, 2018).



```
R é um software livre e vem sem GARANTIA ALGUMA.
Você pode redistribuí-lo sob certas circunstâncias.
Digite 'license()' ou 'licence()' para detalhes de distribuição.

R é um projeto colaborativo com muitos contribuidores.
Digite 'contributors()' para obter mais informações e
'citation()' para saber como citar o R ou pacotes do R em publicações.

Digite 'demo()' para demonstrações, 'help()' para o sistema on-line de ajuda,
ou 'help.start()' para abrir o sistema de ajuda em HTML no seu navegador.
Digite 'q()' para sair do R.

[workspace loaded from ~/.RData]

> 
```

Objects, Functions and Arguments

There are authors who define R objects as being a variable. For the course, we will understand that variables correspond to characteristics of a sample or population.

- Objects are simple ways of accessing something that was saved in the machine memory. It can be a value, a word, one or more variables, an URL, a sample or population database, a list of different things containing information and different sizes, a graph, a map, an image, a new command, etc. **In R EVERYTHING is an object!** Each of these objects has a class!
- Functions correspond to actions, to orders directed to the machine;
- Arguments correspond to refinement or a better direction of the actions or orders proposed by the functions.

Creating an Object in R

- There are two ways of creating an object in the R language:
  - Using the symbol =; or
  - Using the symbol <− called the assignment operator (prefer this form, reserving the symbol = for the value assignments of function arguments or for mathematical operations). The assignment sign can be quickly declared by pressing the ‘Alt’ and ‘-’ keys together.

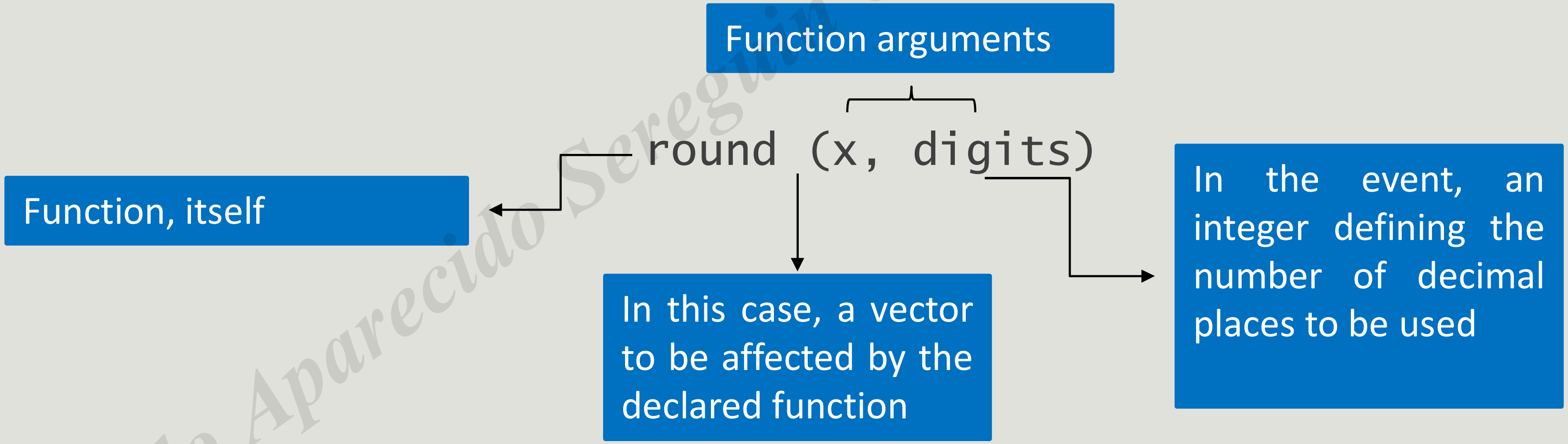
- The names of objects established in R must follow certain rules:
  1. They cannot start with numbers and neither with points;
  2. It is not desirable for them to contain spaces, although the situation is possible with the use of symbols ``;
  3. Their nomenclatures also do not accept special characters, such as !, ~, $, @, +, --, /*.
  4. It is recommended to avoid appointing the objects with the same name of already established functions;
  5. It is recommended not to use accents and, whenever possible, to avoid capital letters, as the language is *case sensitive*

Main Introductive Functions of the Course

Function	Serves to:
args()	Verify the arguments of a given function in R
round()	Round numbers
sample()	Create samples
class()	Verify the classes of R objects
View()	View objects in spreadsheet form
head()	View the first observations of a database
tail()	View the latest observations of a database
str()	Observe the structure of a database
length()	Observe the length of a vector or of a data list
dim()	Discover the dimensions of an object
nrow()	Count the number of lines in a database
ncol()	Count the number of columns in a database
rm()	Remove an object from the work environment
install.packages()	Install packages
library()	Load packages

Using functions in R

To use a function in R, we must know its functional form, that is, we must, as a rule, declare the arguments inherent to it. Example of using the `round()` function:



Function arguments

```
round (x, digits)
```

Function, itself

In this case, a vector to be affected by the declared function

In the event, an integer defining the number of decimal places to be used

Packages in R Language

- The R language has thousands of packages directed to the most diverse areas of knowledge, and most is not installed in our computers. To install a package, we must command:

```
install.packages("package name here")
```

- The installation of a package is not enough for its use. Thus, in each new open section of RStudio, we must call them in the following way:

```
library(package name here)
```

Creating and Excluding Variables in a *Dataset*

- Creating a variable:

```
basededados$nova_variável <- NA
or
mtcars[“nova_variável"] <- NA
```

- Excluding a variable:

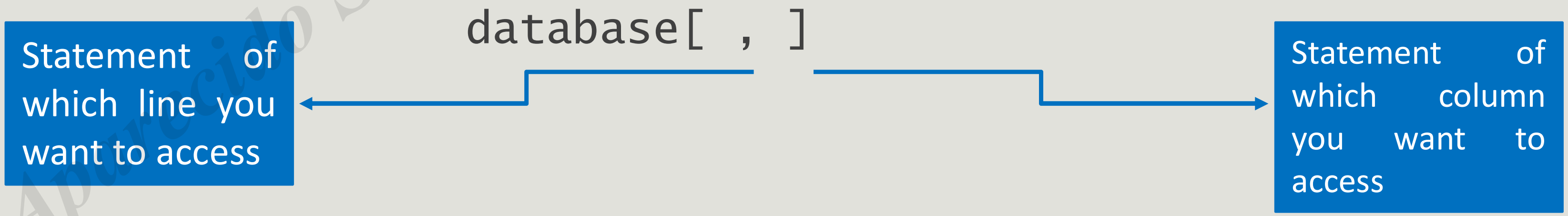
```
basededados$nova_variável <- NULL
```

Extracting Values from a *Dataset*

- To extract a column from a *dataset*, use the operator $:

```
basededados$nova_variável
```

More precisely, values can also be extracted from the *dataset* with the operator [,]:



```
database[ , ]
```

Statement of which line you want to access

Statement of which column you want to access

Functions if, else and ifelse:

```
if(teste lógico)${
if the answer of the logical test is TRUE, do this

} else {
if the answer of the logical test is FALSE, do this other thing
}
```

```
ifelse(teste lógico,
yes =  if the answer of the logical test is TRUE, do this,
no =  if the answer of the logical test is FALSE, do this other thing)
```

Functions **for**, **while** and **repeat**

```
y <- 10

for (i in 1:5){
        print(y + i)
}
```

For each i (could be any symbol or word), present in the sequence of 1 to 5, print the value of the sum between y and i

Functions for, while and repeat

```
z <- 0

while(z < 10){
      print(z)
      z <- z + 1
}
```

While z is smaller than 10, print z and then update the value of z by adding its value in one unit

Functions for, while and repeat

```
w <- 3


repeat{
        print(w)
        w <- w + 2
         if(w > 18) break()


}
```

Repeat the steps below:

- Print the value of w;
- Update the value of w, adding it in two units;
- If w becomes greater than 18, stop everything.

Visualization of Data with ggplot2

- The most basic syntax of the ggplot2 package, for the creation of a chart from a *data frame*, is the following:

```
ggplot(data = database here) +
geom_geometry chosen here(aes(main elements of chart here))
```

Rafael de Freitas Souza
Linkedin