BCBL-Pupillometry-Workshop

Marco A. Flores Coronado

2024-05-23

Hi!

Let's start our R Jedi path.

There is no emotion, there is peace.

There is no ignorance, there is documentation.

There is no passion, there is serenity.

There is no chaos, there is a bug.

There is no death, there is autosave

Figure 1: Breath in, R code

Anyway what are R and R-studio?

Long story short, R is a programming language developed with a special affinity for statistical purposes. R is open source and multi-platform (e.g., works for Linux, Mac-OS, and Windows).

You can do almost anything with it, thus it is specially know for **cool pretty plots** after doing **amazing stats**!

So, R is a high order programming language. That means its the language where we humans can tell the computer what to do, while using human-like syntax and words (i.e., *for*, *if*, *while*, . . .) instead of computer's binary language (i.e., 01000010 01000011 01000010 01001100)

- **Wait a moment! So we don't program in R-studio?**
- **Yes, but actually no**

R is a programming language that translates some plain files (e.g., .R, .txt, .csv) where we write with commands for it. RStudio, in turn, is a nice Graphical User Interface (GUI) that is way way more informative than looking at plain files because:

1- You can load several plain text files

2- You can browse your directory

3- You can auto complete variables, do spelling check and debugging

4- You can explore and visualize variables and their content

R as a calculator

Now that we are starting to use R. So, first you can know that R can be used like a really powerful calculator that in the future will help us from writing/solving by hand bothering ANOVAs or Linear models (i.e., Linear regression, multiple regression, Linear Mixed models)

Sum

```r
100 + 99
```

```
## [1] 199
```

Substraction

```r
100 - 99
```

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

Division

```r
2024 / 360
```

```
## [1] 5.622222
```

Multiplication

```
2024 * 360
```

```
## [1] 728640
```

Exponentiation

```
2 ** 3
```

```
## [1] 8
```

Scary things

Remember: PEMDAS (parenthesis, exponents, multiplication, division, addition, subtraction)

```
(2024 * 360) + (2024 / 4)
```

```
## [1] 729146
```

Wow! look at us, using our powerful machine to do some maths. This may be trivial, but while your coding experience, you are going to use operations like the aforementioned, thus with **variables**

Variables

A variable is a data item. Imagine you are an elf with n amount of magical points. It may be the case that you want to infuze with your MP some objects. For example, you infuze a sheet with the word *HEAL* with *healing* .

In this case the variable named *HEAL* contains the magic *healing*. You can name your sheets of papers however you can, thus it is always useful to have names that remember you what they contain. You only have one restriction, yo can not use the exactly variable word twice or it may be overwritten.

In R we can name our variables as we want, thus we can not start to name them with numbers, capitals or special characters. Special characters besides . or _ are not recommended.

```
# First of all, as you can see this line gives no instruction to R. However, R doesn't yell at us. Why?

#lets make our first variable!

my.number <- 10 #this means I have a variable called my.number that contains the number 10

my.letter <- "a" #this means I have a variable called my.letter that contains the string a (how we call
# as you can notice, strings are defined by being surrounded by quotes (smart!)

robotaren.agurra <- "HELLO WORLD" #Strings can be really long!

print(robotaren.agurra)# we are running our first command!
```

```
## [1] "HELLO WORLD"
```

You may not have noticed, but we run our first command! we asked the computer to print within the console the string contained in “robotaren.agurra”

In R it’s tradition (*it’s the law*) that when we have multiple words to name a variable we use a dot to separate them. Also, we use the symbol <- intead of =

We have already defined some variables with different data types. Do yo know which type? If you are like me and have the memory power of Dory from finding Nemo, you will explore which data type you are working with. Luckily R developers have us covered

```
class(my.number) # I'm asking the class of whats inside the variable my.number
```

```
## [1] "numeric"
```

```
class(my.letter)# I'm asking the class of whats inside the variable my.letter
```

```
## [1] "character"
```

```
#now I'm gonna create a new data type, a Factor *mind blown*

my.factor = as.factor("level")

class(my.factor)
```

```
## [1] "factor"
```

During your statistical Jedi trip, you will encounter this type of elements most of the time. numeric and characters are kinda self-explanatory, but

- **What is a factor?**
- Im glad that you asked! Long story short, **I do not know**

Factors are this really weird R class type that helps us to differentiate between element types. Imagine you have a scroll with the names of all the elves and orcs living as friends inside a village (*It’s my example, and I want them to be friends*).

OK, then in your scroll you have their names and their specie. In this example Specie would be a factor class. This is because its not a value that we can change (sorta). An elf is an elf, despite of it’s height, hair or being played by Orlando Bloom.

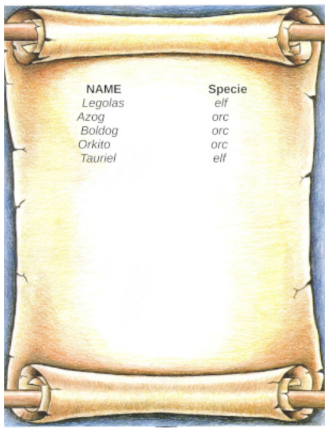
Data structures

My fellow padawan, I sense the R force is strong in you! You may be asking, “Ha, but what if I want many elements inside a variable?” We are going places here!

OK, this kind of **structures** that hold data are called *data structures.* We have one great distinction, we may want structures containing the same data class or structures with different ones.

Imagine now you want to store a lot of *healing* spells against fire, PhD burn-out, hangover and fantasy stuff like that (blink blink). We’ll then you would use a stone that can store tons of healing spells (*homogeneous* if you want to sound nerdy).

If you want, instead, to store different type of spells like *healing*, *explosion*, and *light*, because you want an elvish swiss-army-spell-rock . Well, then you are looking for a rock that can hold different spell types at the same time (*heterogeneous*).



NAME	Specie
Legolas	elf
Azog	orc
Boldog	orc
Orkito	orc
Tauriel	elf

Figure 2: Happy village scroll

Figure 3: What about a friend

Homogeneous data type

vector Vectors are 1 dimensional data structures (**1D**). Think of it like a row or things

```r
my.numbervector <- c(1,2,3,4,5)

my.stringvector <- c("It's", "a", "trap", "!")
```

```r
#this will convert the number to a string because a vector can't contain numbers
# and characters
my.vector=c(1,"2")
```

Matrix Matrix further than being a matrix, they can be seen as several vectors tied together. This structure is bi-dimensional (**2D**). Kinda like if you make a boat by gluing wood logs. Thus, **every vector has to have the same length**

```r
first.log <- c("a","b","c","d")
second.log <- c("1","2","3","4")
my.logmatrix <- rbind(first.log,second.log)
```

cool right! *There'r another multidimentional homogeneous data structure called table or array (**n-dimensional**), but we are not going to speak about these #sorry!

Heterogeneous data type

list List are 1 dimentional data structures (**1D**) like vectors, but we can have different data types within

```r
my.list <- list(1,"2","3",4)

my.spellrock.list <- list(heal= "woosh",lives=2, mp.level=10,explosion=c("boom","kaboom"))

print(my.spellrock.list[1])
```

```
## $heal
## [1] "woosh"
```

```r
print(my.spellrock.list$explosion)
```

```
## [1] "boom"   "kaboom"
```

We did a lot here! As you can see, we can store different kind of class types inside our list. How cool is that (well, not so much TBH, but its useful).

Table Matrix further than being a matrix, they can be seen as several vectors tied together. Kinda like if you make a boat by gluing wood logs. Thus, **every vector has to have the same length**

```r
first.log <- c("a","b","c","d")
second.log <- c("1","2","3","4")
my.logmatrix <- rbind(first.log,second.log)
```

Cool! Right? *There's another multi-dimensional homogeneous data.structure called table or array, but we are not going to speak about these #sorry!

Figure 4: Explosion!