TutoRial - Part 1

Marine Ecosystem Dynamics - 2024

R syntax

R is a programming language that use a simplified syntax. In this section, we will explore how to write a script and execute it.

But first some syntax information:

- Everything after # is considered as a comment and will not be executed. It is very important to write what we are doing, so we do not get lost next time we open our scripts.

```
# 2 + 2 will not work because of the #
2 + 2 # We should then annotate our script like this
#> [1] 4
```

- Several lines of code can be written in one line but must be separated by a semicolon

```
2 + 2
#> [1] 4
3 * 2
#> [1] 6

# This can also be written as follow:
2 + 2 ; 3 * 2
#> [1] 4
#> [1] 6
```

- In **R** we can name any object using =, `<-`, `->` or `assign`

```
c(1, 2, 3, 4) -> my_first_vector
my_vector <- c(1, 2, 3, 4)
my_function = function(x){x + 2}
assign("x", c(2, 3, 4, 5))
```

- == is a logical function that can be translated as *is equal to*, contrarily *is not equal to* is written `!=`

```
2 + 2 == 4
#> [1] TRUE
3 * 2 == 4
#> [1] FALSE
3 * 2 != 4
#> [1] TRUE
```

Exercises

Using a new **R** script, do these calculations:

- 2^7
- $cos(\pi)$
- The sum of all number from 1 to 100

Create a parameter `x1` that equals to 5 and a parameter `x2` that equals to 10

- Is `2* x1` equal to `x2`?

Functions

As seen during the lecture, **R** works with functions that can:

- Already be implemented in base **R**
- Comming from another package
- Created by the user

We will see these three examples in this section, but first it is important to remember that the typical structure of a function is `function(argument1, ...)`.

Fortunately **R** helps us to remember what are the needed `arguments`:

- Using `help()` or `?`
- Using `example`

For the functions that comes from external packages, we first need to install the new packages. The most common way to do so is by executing `install.packages("Package_Name")`. Then when we want to load the functions, we start the script by executing `library(Package_Name)`.

Finally, if we really do not find a suitable function in a package, we can create your functions following this general structure, but this will not be covered in this tutorial:

```
my_function <- function(<argument1>, <argument2>, ...){
  <here comes the definition of my function>
  return(<output of the definition>)
}
```

Exercises

- What is the function `log()` doing and from were does this function come from (base **R**, other packages)?

- What are the mandatory arguments for the function `plot()`
- Is there help associated with the functions from a loaded package?