Basics of R: Tutorial

Below starts a tutorial that teaches you your first steps in R. Learn by doing! Type the lines of code below one by one in an empty R script, send them in the R console (using `[Ctrl] +[Enter]`), and study what happens. Do not hesitate to play around with the code! Make sure you completely understand what the commands do. Ask questions about every detail you do not understand.

1. Assignment and basics

- There are three ways to assign a value to an object:

  1. an equals sign =
  2. a "left arrow" `<-` (less than, hyphen)
  3. a "right arrow" `->` (hyphen, greater than).

  You can type the name of any object to retrieve the value of that object. Try it:

```
> n <- 15
> n
```

```
[1] 15
```

  The object (variable) `n` now stores the value 15.

  Now try the following:

```
> n<-15
```

```
[1] 15
```

  Note that we did not insert spaces around the "arrow". Yet, `n` is again assigned the value 15. The point is: you can use spacing to make your commands look "pretty" (and readable), but R usually does not care. It's (generally) up to you. Don't, however, be so silly as to put a space in the middle of the name of an object or operator.

  Here is one place where R insists on the correct spacing: The "arrow" assignment operator is actually two symbols, a less than sign and a hyphen or minus. There cannot be a space between them!

  We recommend that you leave spaces on each side of the arrow operator (and operators in general). That way you're less likely to make critical spacing errors. For example, suppose your fingers get all crossed up, and you type this:

```
> n < -15
```

  Try it and see what happens. Usually not a problem–just retype it correctly:

```
> n <- 15
```

```
[1] 15
```

  Instead of the arrow, you can also use an equality sign:

```
> a = 12
> a
```

```
[1] 12
```

You can even do this:

```
> 24 -> z # the arrow always points towards the variable name
> z
```

```
[1] 24
```

- As in any good programming language, you can insert comments into R code. Any line or partial line in R beginning with the hash symbol (#) is a comment, or note, and R will ignore it.

```
> # This is just a note.
> ### And so is this.
> n <- 15 # And so is this.
```

```
[1] 15
```

In the last line, R will assign the value 15 to the object `n` and ignore all the comments after the hash symbol.

- If you surround an assignment with parentheses, R prints the value to the screen. Compare the output of this command

```
> n <- 8
```

with the output of this command

```
> (n <- 8)
```

```
[1] 8
```

Note, by the way, that `n` previously had the value 15; we have now overwritten it with the value 8.

- Variable names must start with a letter, but may also contain numbers and periods:

```
> (var.s13 <- 4)
```

```
[1] 4
```

- R is case sensitive. That is, `n` and `N` are different variables.

```
> (N <- 26.42)
```

```
[1] 26.42
```

```
> n # The value of n is still 8.
```

```
[1] 8
```

- To see a list of your objects, use the function `ls()`. The parentheses `()` are required, even though there are no arguments.

```
> ls()
```

```
[1]  "a"  "n"  "N"  "var.s13"  "z"
```

- Use `rm()` to delete objects you no longer need.

```
> rm(n)
> ls()
```

```
[1] "a"   "N"   "var.s13"   "z"
```

- You may obtain online help about a function using either the `help()` command or a question mark.

```
> ?ls
> help(rm)
```

- Several commands are available to help find a command whose name you do not know.

```
> apropos("help")  # Find commands with "help" in their name
```

```
[1] ".helpForCall"   "help"    "help.search"    "help.start"
[5] "link.html.help"
```

2. Simple calculations

- R may be used for simple calculations, using the standard arithmetic symbols `+`, `-`, `*`, `/`, and `^` (exponentiation), as well as parentheses.

```
> a <- 12 + 14
> a
```

```
[1] 26
```

```
> 3*5
```

```
[1] 15
```

```
> (20 - 4)/2
```

```
[1] 8
```

```
> 7^2
```

```
[1] 49
```

- Standard mathematical functions and constants are available.

```
> exp(2)  # Exponential function
```

```
[1] 7.389056
```

```
> log(10)  # Natural log
```

```
[1] 2.302585
```

```
> log10(10)  # Log with base 10
```

```
[1] 1
```

```
> log2(64)  # Log with base 2
```

```
[1] 6
```

```
> pi   # The mathematical constant pi
```

```
[1] 3.141593
```

```
> cos(pi) # The cosine of pi
[1] -1
```

```
> sqrt(100) # The square root of 100
[1] 10
```