A minimalist introduction to R

Timothée Bonnet & al.

January 25, 2018

This document recycles tutorials written by Koen van Benthem and Tina Cornioley

Contents

There are many ways to achieve the same goal in `R`, and we do not claim to teach you the most efficient way to use `R`. If you at some point during the computer practicals encounter a code that you could make more efficient or elegant, please do let us know!

Do try to understand exactly what the code and the functions we use do. The best way to learn how functions work is by either using the `R`-manual (type `?functionname` or use the RStudio Help tab by clicking on it or pressing F1) or by creating dummy data (just make up a small amount of data yourself, using `R` if possible!) and analyse what the function does to this data.

How this document works

This is a `knitr` document, which knits `R` code and output within a LaTeXdocument. R code and output is generally contained within boxes with a gray background. Comments within the R code start with a # symbol; lines with R-outputs start with ##.

All the files necessary to go through the workshop are (or should be!) in the folder of a github repository. We recommend you copy these files, or fork the repository if you are a git user.

Now, let's the fun begin.

1 Trash your calculator

1.1 Operators

`R` can be used as a calculator, and a far more powerful one that any physical calculator. If you use your calculator to enter numbers in `R`, you are being inefficient.

Below we demonstrate the use of some basic mathematical operators:

```
1+3 #addition

## [1] 4

5-2 #substraction

## [1] 3

6*4 #multiplication

## [1] 24

14/2 #division

## [1] 7

2^3 #exponent

## [1] 8

2**3 #or equivalently

## [1] 8
```

There are many mathematical functions already present in R:

```r
exp(3) #exponential
```

```
## [1] 20.08554
```

```r
log(2.71) #logarithm
```

```
## [1] 0.9969486
```

```r
sqrt(9) #square root, which of course you can also write as:
```

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

```r
9 ^ (1/2)
```

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

```r
sin(pi/2); cos(1); tan(pi/3) #trigonometric functions
```

```
## [1] 1
## [1] 0.5403023
## [1] 1.732051
```

Small exercise
Use R to compute

$$y = \frac{1}{2\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{3-\pi}{2})^2}$$

Logical operators are very important for programming and scripting. You can test whether two things are equal with double = signs:

```r
3 == 6/2 #is 3 equal to 6/2? TRUE!
```

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

```r
3 == pi  # FALSE!
```

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

You can also test if they are NOT equal with the operator `!=`:

```r
2 != 3
```

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

```r
2 != 2
```

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

The AND operator is `&`

```
2 == 2 & 3==3
```

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

```
2 ==2 & 3==2
```

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

The OR operator is |

```
2 == 2 | 3==2
```

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

```
2 == 4 | 3==2
```

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

Small exercise
Try and guess the result of these logical tests before running them:

```
! 1==2
(1!=2 | 3==4) & (2==4/2)
"abc" != "bc"
```

1.2 Assignment

Values can be assigned to objects to store them and make your code flexible. You assign a value to an object using the operator `<-` (or `=` , but be careful not to confuse this with the `==` used in tests).

```
a <- 12
a + 2
```

```
## [1] 14
```

```
b <- a
a*b
```

```
## [1] 144
```

```r
a <- "c"
b <- "c"
a == b

## [1] TRUE

a == "b"

## [1] FALSE
```

2 Containers

A container is some kind of object that contain several values. The simplest container is a vector. You can create a vector by *concatenating* several values with the syntax `c(...)`.

3 loops and if statements

3.1 for loops

3.2 while loops

3.3 if statements