UPPSALA UNIVERSITET
Matematiska Institutionen
Statistics for biology. Spring semester 2015.
Jordi-Lluís Figueras (figueras@math.uu.se)

An introduction to R.

R is a powerful environment for statistical computing. In this lesson we will start working with it. Through a set of exercises we will learn its basics. We will use RStudio, an integrated R framework with lot of features that will help us to work more easily.

1. Open RStudio. As you would see, RStudio shows different windows:
   - A console (where we can input commands).
   - A raw file (where we can write scripts).
   - A window with all the saved data.
   - A window with different features like: Files, Plots, Packages, Help, Viewer.
2. Type

   ```
   > q()
   ```

   on the console. This should close RStudio.
3. Open again RStudio. Type `1+2`. What do you obtain? And if you type `cos(1)`? As you see, RStudio can be used as a normal calculator. You can perform standard arithmetic operations like `+, -, *, %`. Also, you can use standard mathematical functions like `sqrt, log, exp`.
4. Compute $\sin(3)^2 + \cos(3)^2$. What do you obtain? Do you know why?
5. Type

   ```
   > (-1)^2
   > -1^2
   ```

   Do you see any difference?
6. Can you tell which of the following expressions gives a bigger result? $\sqrt{3^2 + 2}$ and $\sqrt{3^2} + \sqrt{2}$.
7. In R there are also some constants. Type

   ```
   > pi
   ```

   Do you recognize it?
8. Now, type

```
> Pi
```

Do you observe any difference? That's because R is *case sensitive*: Expressions like `Aa, aa` are different.

9. One of the most important operators in R is the assignment operator: `<-`. Type

```
> x <- 1+1
```

Now `x` stores the numeric value `2`. Can you see it in the top left window? Now, if you type

```
> x
```

you should obtain the value stored in `x`.

10. Instead of using the assignment operator `<-` you can use `=`. Test it with

```
> x = 1+1
```

11. In R we can also use logical operators. These are `<=, <, ==, > >=` and $!=$. Test them. Do you see the difference between the operators `=` and `==`? Keep it in mind!

12. Type again

```
> x <- 1
> z <- 2
```

Although the top left window lists all the objects defined during the session, we can list them by typing

```
> ls()
```

and we can delete any of them by using the command `rm()`. As an exercise, delete `x`.

1 Vectors and matrices

1. In R there is a useful function: `c`. This function (concatenate) can be used to create vectors. Type

```
> x <- c(1,4,6,3)
> x
```

What do you observe?

2. Experiment with the following:

```
> x <- c(1,4,6,3)
> log(x)
> cos(x)
> x+x
> y <- c(2,3,5,6)
> x+y
> x*y
```

What do you observe?

3. With R, matrices can be created. To do it, we will use `matrix`. For example,

```
> x <- c(1,4,6,3)
> M <- matrix(x, 2, 2)
```

What do you observe?

Remark: Now that some commands have been introduced, there is one quite important that is missing: the command `help` or `?`. Experiment with them and try to understand their outputs.

4. Type the following:

```
> x <- c(1,3)
> y <- c(5,3)
> M <- matrix(c(1,1,1,1), 2, 2)
> x*y
> x%*%y
> M*x
> M%*%x
> M <- matrix(c(1,2,3,4), 2, 2)
> M[2]
> M[1,]
> M[2,]
> M[,1]
> M[,2]
```

What do you observe?

2 Lists and data frames

1. A *list* is a generalized vector. Vectors can only have elements of one type (numbers, characters, strings...) but a list can have them mixed.

To create a list, we just type

```
> l <- list(name="jordi", age="42")
```

2. Now, if we type

```
> l
```

we can see its components. Type now

```
> l$name
> l$age
> l[1]
> l[2]
```

What do you observe?

3. *Data frames* are used for storing data tables. In fact, it is a list of vectors of equal length. For example, type

```
> x <- c(10,20,30)
> y <- c("y", "n", "n")
> m <- data.frame(x,y)
> m
```

What do you observe?

4. Now, try yourself. Create a table, using the commands `data.frame` and `list`, with 2 columns and 4 rows. This table should contain both numbers and strings. Find a good example.

5. Have a look at the following example:

```
> x <- list(name="first variable", values = c(10,20,30))
> y <- list(name="second variable", values = c(1,2,3))
> str(x)
> x
> y
> x$name+y$name
> y$values+y$values
```

What do you observe?

3 Reading files

1. A lot of time we will use data coming from files. That's because we want to work with data of moderate size, and introducing it via command-line could be a lengthy task.

2. Files with input can have different formats. The format that we will learn today is called `csv2`. This format assumes that data is separated with semicolons (`;`) and that decimal numbers are expressed with commas. Also, this format assumes that the first line contains the headers of the columns.

   An example of how data are written down inside a file is

```
A;B
1;2
1,2;2
1,4;3
1,5;5
```

3. Given a file in `csv2` format, we will read it and stored in a data frame using the command `read.table("filename",sep=";", dec=",", header=T)`

4. Create a file with two columns and four rows in format `csv2`. (Remember that you should create it in the same working directory you are). Now, read this file using the function `read.table`.

4 Scripting

1. A lot of times the computations and manipulations are several lines. When we have long tasks it is proper to write down *scripts*. These are just files with all the commands we want to execute in order and separated by semicolons.

2. To execute the script, we just type `source("`*filename*`")` in the command-line.

3. As an exercise, script the first few commands that appeared during this lesson.