Getting started with R

Intro to Stats, Spring 2017

Prof. Gaston Sanchez

Learning Objectives

- Complete installation of R and RStudio
- Get started with R as a scientific calculator
- First steps using RStudio
- Getting help in R
- Installing packages
- Using R script files
- Using Rmd files
- Get to know markdown syntax

R and RStudio

- Install **R**
  - R for Mac: https://cran.r-project.org/bin/macosx/
  - R for windows: https://cran.r-project.org/bin/windows/base/
- Install **RStudio**
  - RStudio download (desktop version): https://www.rstudio.com/products/rstudio/download/

Difference between R-GUI and RStudio

The default installation of R comes with R-GUI which is a simple graphical user interface. In contrast, RStudio is an *Integrated Development Environment* (IDE). This means that RStudio is much more than a simple GUI, providing a nice working environment and development framework. In this course, you will use R mainly for doing computations and plots, not really for programming purposes. And you are going to interact with R via RStudio, using the so-called **Rmd** files.

R as a scientific calculator

Open RStudio and locate the *console* (or prompt) pane. Let's start typing basic things in the console, using R as a scientific calculator:

```
# addition
1 + 1
2 + 3

# subtraction
4 - 2
```

```
5 - 7

# multiplication
10 * 0
7 * 7

# division
9 / 3
1 / 2

# power
2 ^ 2
3 ^ 3
```

Functions

R has many functions. To use a function, type its name followed by parenthesis. Inside the parenthesis you pass an input. Most functions will produce some type of output:

```
# absolute value
abs(10)
abs(-4)

# square root
sqrt(9)

# natural logarithm
log(2)
```

Comments in R

All programming languages use a set of characters to indicate that a specific part or lines of code are **comments**, that is, things that are not to be executed. R uses the hash or pound symbol `#` to specify comments. Any code to the right of `#` will not be executed by R.

```
# this is a comment
# this is another comment
2 * 9

4 + 5  # you can place comments like this
```

Variables and Assignment

R is more powerful than a calculator, and you can do many more things than practically most scientific calculators. One of the things you will be doing a lot in R is creating variables or objects to store values.

For instance, you can create a variable `x` and give it the value of 1. This is done using what is known as the **assignment operator** `<-`, also known in R as the *arrow* operator:

```
x <- 1
x
```

This is a way to tell R: "create an object `x` and store in it the number 1". Alternatively, you can use the equals sign `=` as an assignment operator:

```
y = 2
y
```

With variables, you can operate the way you do algebraic operations (addition, subtraction, multiplication, division, power, etc):

```
x + y
x - y
x * y
x / y
x ^ y
```

Case Sensitive

R is case sensitive. This means that `abs()` is not the same as `Abs()` or `ABS()`. Only the function `abs()` is the valid one.

```
# case sensitive
x = 1
X = 2
x + x
x + X
X + X
```

Some Examples

Here are some examples that illustrate how to use R to define variables and perform basic calculations:

```
# convert Fahrenheit degrees to Celsius degrees
fahrenheit = 50
celsius = (fahrenheit - 32) * (5/9)
celsius


# compute the area of a rectangle
rec_length = 10
rec_height = 5
rec_area = rec_length * rec_height
rec_area


# degrees to radians
deg = 90
rad = (deg * pi) / 180
rad
```

More about RStudio

You will be working with RStudio a lot, and you will have time to learn many of the bells and whistles RStudio provides. Think about RStudio as your "workbench". Keep in mind that RStudio is NOT R. RStudio is an environment that makes it easier to work with R, while taking care of the little tasks that can be a hassle.

A quick tour of RStudio

- Understand the **pane layout** (i.e. windows) of RStudio
  - Source
  - Console
  - Environment, History, etc
  - Files, Plots, Packages, Help, Viewer
- Customize RStudio Appearance of source pane
- font
- size
- background

Using an R script file

Most of the time you won't be working directly on the console. Instead, you will be typing your commands in some *source* file. The basic type of source files are known as *R script files*. Open a new script file in the *source* pane, and rewrite the previous commands.

You can copy the commands in your source file and paste them in the console. But that's not very efficient. Find out how to run (execute) the commands (in your source file) and pass them to the console pane.

Getting help

Because we work with functions all the time, it's important to know certain details about how to use them, what input(s) is required, and what is the returned output.

There are several ways to get help.

If you know the name of a function you are interested in knowing more, you can use the function `help()` and pass it the name of the function you are looking for:

```
# documentation about the 'abs' function
help(abs)

# documentation about the 'mean' function
help(mean)
```

Alternatively, you can use a shortcut using the question mark `?` followed by the name of the function:

```
# documentation about the 'abs' function
?abs

# documentation about the 'mean' function
?mean
```

- How to read the manual documentation:
  - Title
  - Description
  - Usage of function
  - Arguments
  - Details
  - See Also
  - Examples!!!

`help()` only works if you know the name of the function your are looking for. Sometimes, however, you don't know the name but you may know some keywords. To look for related functions associated to a keyword, use `help.search()` or simply `??`

```
# search for 'absolute'
help.search("absolute")

# alternatively you can also search like this:
??absolute
```

Notice the use of quotes surrounding the input name inside `help.search()`

Installing Packages

R comes with a large set of functions and packages. A package is a collection of functions that have been designed for a specific purpose. One of the great advantages of R is that many analysts, scientists, programmers, and users can create their own pacakages and make them available for everybody to use them. R packages can be shared in different ways. The most common way to share a package is to submit it to what is known as **CRAN**, the *Comprehensive R Archive Network*.

You can install a package using the `install.packages()` function. Just give it the name of a package, surrounded by quotes, and R will look for it in CRAN, and if it finds it, R will download it to your computer.

```
# installing
install.packages("knitr")
```

You can also install a bunch of packages at once:

```
install.packages(c("readr", "ggplot2"))
```

The installation of a package needs to be done only once. After a package has been installed, you can start using its functions by *loading* the package with the function `library()`

```
library(knitr)
```

Your turn

- Install packages `"stringr"`, `"RColorBrewer"`
- Calculate: $3x^2 + 4x + 8$ when $x = 2$
- Look for the manual (i.e. help) documentation of the function `exp`
- Find out how to look for information about binary operators like `+` or `^`
- There are several tabs in the pane `Files, Plots, Packages, Help, Viewer`. Find out what does the tab **Files** is good for?

Introduction to Rmd files

Besides using R script files to write source code, you will be using other type of source files known as *R markdown* files, simply called `Rmd` files. These files use a special syntax called markdown.

Get to know the `Rmd` files

In the menu bar of RStudio, click on **File**, then **New File**, and choose **R Markdown**. Select the default option "Document" (HTML output), and click **Ok**.

Rmd files are a special type of file, referred to as a *dynamic document*, that allows you to combine narrative (text) with R code. It is extremeley important that you quickly become familiar with this resource. One reason is that you can use Rmd files to write your homework assignments and convert them to HTML, Word, or PDF files.

Locate the button **Knit** (the one with a knitting icon) and click on it so you can see how `Rmd` files are renderer and displayed as HTML documents.

Yet Another Syntax to Learn

R markdown (`Rmd`) files use markdown as the main syntax to write content. It is a very lightweight type of markup language, and it is relatively easy to learn.

Your turn

If you are new to Markdown, please take a look at the following tutorials:

- www.markdown-tutorial.com
- www.markdowntutorial.com/

Rmd basics

- YAML header:
  - title
  - author
  - date
  - output: `html_document`, `word_document`, `pdf_document`
- Code Chunks:
  - syntax
  - chunk options
  - graphics
- Math notation:
  - inline `$z^2 = x^2 + y^2$`
  - paragraph `$$z^2 = x^2 + y^2$$`

Example of inline equation: $z^2 = x^2 + y^2$

Example of equation in its own paragraph:

$$z^2 = x^2 + y^2$$

RStudio has a basic tutorial about R Markdown files: Rstudio markdown tutorial

Rmd files are able to render math symbols and expressions written using LaTeX notation. There are dozens of online resources to learn about math notation and equations in LaTeX. Here's some documentation from www.sharelatex.com/learn/

- Mathematical expressions
- Subscripts and superscripts
- Brackets and Parentheses
- Fractions and Binomials
- Integrals, sums and limits
- List of Greek letters and math symbols
- Operators