Programming course

Lecture 5-6: Introduction to R

Diego de Sousa Rodrigues
diego.desousarodrigues@sciencespo.fr

Sciences Po Paris

Disclaimer

- Most of the material is drawn from the excellent course prepared by software carpentry
- In particular, most exercises are drawn from it (If you really want to learn something, don't look up the answers)
- Other source of inspiration is the very complete QuantEcon website

What and why ?

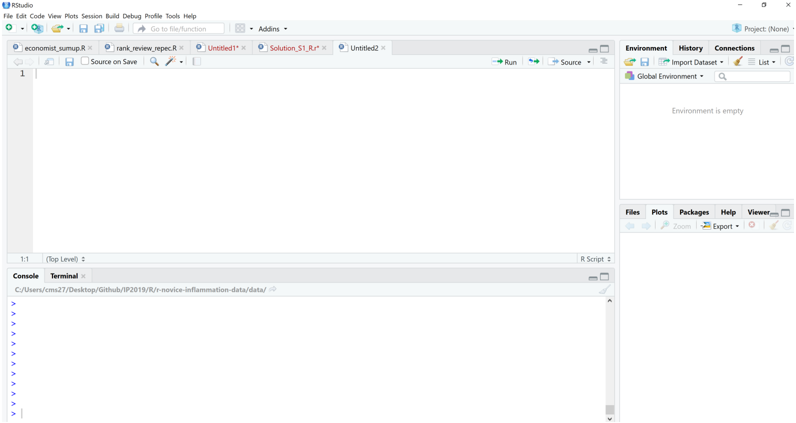
R : Let's start !

- Why are we using R ?
  - **Better than Stata** by ANY metric
  - Free
  - Extremely popular amongst scientists, in particular statisticians and economists
  - Exists a large library of external packages

What and why ?

R : Let's start !

- Let's start by opening RStudio !



Environment is empty

```
C:/Users/cms27/Desktop/Github/IP2019/R/r-novice-inflammation-data/data/
>
>
>
>
>
>
>
>
>
>
>
>
```

Variables

Create a variable in R

- A variable : a container with a name
- To create a variable called weight with value 55, just type :

  `weight <- 55` (or `weight = 55`)

Variables

Create a variable in R

- A variable : a container with a name
- To create a variable called `weight` with value 55, just type :

  `weight <- 55` (or `weight = 55`)
- Can treat the variable like a regular number. Try `weight + 1`
- Can change an variable's value by assigning it a new value. Just type :

  `weight <- 60`

Variables

Create a variable in R

- R only stores the value, not the calculation used to create a variable ($\neq$ Excel). Try this :

Variables

Create a variable in R

- R only stores the value, not the calculation used to create a variable (≠ Excel). Try this :

  ```
  weightlb <- 2.2 * weightkg
  c(weightkg, weightlb)
  weightkg <- 80
  c(weightkg, weightlb)
  ```

- c is also a function (probably the most used function in R), stands for combine

Variables

Create a variable in R

- Some conventions on the name of variables
  1. start with lower case letters
  2. separate words with underscores
  3. use only lowercase letters, underscores, and numbers

Motivating example

Analyzing data w. R

- **The data** : We are studying inflammation in patients who have been given a new treatment for arthritis, and we need to analyze the first dozen data sets. The data sets are stored in comma-separated values (CSV) format. Each row holds the observations for just one patient. Each column holds the inflammation measured in a day, so we have a set of values in successive days.

Motivating example

Analyzing data w. R

- **The data** : We are studying inflammation in patients who have been given a new treatment for arthritis, and we need to analyze the first dozen data sets. The data sets are stored in comma-separated values (CSV) format. Each row holds the observations for just one patient. Each column holds the inflammation measured in a day, so we have a set of values in successive days.

  1. Go to the clone of my Github Repo on your computer (or to my Github repo (github.com/Diego-de-Sousa-Rodrigues/IP2022)) and download : `r-novice-inflammation-data`
  2. Goal : load the data, calculate the average value of inflammation per day, plot the results

Motivating example

Analyzing data w. R

- Loading data :

  1. Set the directory where the data is stored with `setwd()` :

     ```
     setwd("C:/Users/YourName/.../data")
     ```

  2. Import data in `d` with :

     ```
     d=read.csv(file="inflammation-01.csv", header=FALSE)
     ```

- both `setwd()` and `read.csv()` are functions that takes some arguments

Motivating example

Analyzing data w. R

- Loading data :

  1. Set the directory where the data is stored with `setwd()` :

     ```
     setwd("C:/Users/YourName/.../data")
     ```

  2. Import data in `d` with :

     ```
     d=read.csv(file="inflammation-01.csv", header=FALSE)
     ```

- both `setwd()` and `read.csv()` are functions that takes some arguments

  1. the first argument of both functions is a String ⇒ put quotes
  2. the second argument of read.csv is what we call a Boolean value (either true or false). Header : whether the first line of the file contains names for the columns of data
  3. d = data frame. more on this later : but basically, like an excel sheet.

Motivating example

Analyzing data w. R

- Manipulating the data :

  1. Display the first lines of the data set with `head` :

     ```
     head(d, n = 3L)
     ```

  2. To take a subset of the data set, provide an index in square bracket : [# row, # column] :

     ```
     d[1,1]
     ```

     ```
     d[c(1, 3, 5), c(10, 20)]
     ```

     ```
     d[1, 1:5]
     ```

     ```
     d[, 1]
     ```

Motivating example

Analyzing data w. R

- Manipulating the data :

  1. Display the first lines of the data set with `head` :

     `head(d, n = 3L)`

  2. To take a subset of the data set, provide an index in square bracket : [# row, # column] :

     `d[1,1]` # first row, first column

     `d[c(1, 3, 5), c(10, 20)]` # rows (1, 3 and 5), columns (10 and 20)

     `d[1, 1:5]` # columns from (1 to 5) and row 1

     `d[, 1]` # all rows from column 1

Motivating example

Analyzing data w. R

- In our data set, each row is a patient, each column is a day, such that `d[1,1]` is the inflammation measured on patient 1 on day 1

- **Exercise 1** : given that `min(data)`, `max(data)`, `mean(data)` are functions returning the equivalent statistics on data, find :

  1. the minimum inflammation on day 1 across all patients

Motivating example

Analyzing data w. R

- In our data set, each row is a patient, each column is a day, such that `d[1,1]` is the inflammation measured on patient 1 on day 1

- **Exercise 1** : given that `min(data)`, `max(data)`, `mean(data)` are functions returning the equivalent statistics on data, find :

    1. the minimum inflammation on day 1 across all patients
    2. the maximum inflammation experienced by patient 5 (across all days)

Motivating example

Analyzing data w. R

- In our data set, each row is a patient, each column is a day, such that `d[1,1]` is the inflammation measured on patient 1 on day 1

- **Exercise 1** : given that `min(data)`, `max(data)`, `mean(data)` are functions returning the equivalent statistics on data, find :

    1. the minimum inflammation on day 1 across all patients
    2. the maximum inflammation experienced by patient 5 (across all days)
    3. the maximum inflammation on days 4, 8 and 12 across all patients

Motivating example

Analyzing data w. R

- In our data set, each row is a patient, each column is a day, such that `d[1,1]` is the inflammation measured on patient 1 on day 1
- **Exercise 1** : given that `min(data)`, `max(data)`, `mean(data)` are functions returning the equivalent statistics on data, find :

    1. the minimum inflammation on day 1 across all patients
    2. the maximum inflammation experienced by patient 5 (across all days)
    3. the maximum inflammation on days 4, 8 and 12 across all patients
    4. the minimum inflammation experienced by patients 3 and 6 from day 1 to 5

Motivating example

Analyzing data w. R

- In our data set, each row is a patient, each column is a day, such that `d[1,1]` is the inflammation measured on patient 1 on day 1

- **Exercise 1** : given that `min(data)`, `max(data)`, `mean(data)` are functions returning the equivalent statistics on data, find :

  1. the minimum inflammation on day 1 across all patients
  2. the maximum inflammation experienced by patient 5 (across all days)
  3. the maximum inflammation on days 4, 8 and 12 across all patients
  4. the minimum inflammation experienced by patients 3 and 6 from day 1 to 5
  5. the mean inflammation experienced by patients 2, 4 and 10 (across all days)

Motivating example

Analyzing data w. R

- Faster way to get some sufficient statistics (by columns) : `summary` (ex : `summary(d[, 1:5])`)
- What if we want some info, say the median, for each partient (= row) ? No such things as `rowMedian`
- `apply` : repeat a function on all of the rows (MARGIN = 1) or columns (MARGIN = 2) of a data frame (`apply(d, 1, median)`)
- **Exercise 2** : compute in two different ways the mean for the first 10 patients of our data

Motivating example

Analyzing data w. R

- R plot are very nice :
- Try `plot(apply(d, 2, max), xlab = "day", ylab = "maximum", main = "maximum inflammation by day")`
- and `boxplot(d, main = "Summary")`

Motivating example

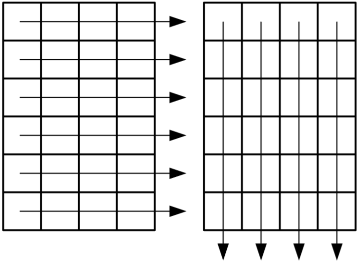
Analyzing data w. R

- **Exercise 2** : Load the data, use the function `apply` to create a variable containing the min value each day + `plot` the result

Motivating example

Analyzing data w. R

- **Exercise 2** : Load the data, use the function apply to create a variable containing the min value each day + plot the result



across rows (1) across columns (2)

What and why ?

Function in R

- Questions :
  - How do I make a function ?
  - How can I test my functions ?
  - How should I document my code ?

What and why ?

Function in R

- Questions :
  - How do I make a function ?
  - How can I test my functions ?
  - How should I document my code ?
- Objectives
  - Define a function that takes arguments.
  - Return a value from a function.
  - Test a function.
  - Explain why we should divide programs into small, single-purpose functions.

What and why ?

Function in R

- Function : why is this so useful ?

What and why ?

Function in R

- Function : why is this so useful ?
- If we only had one data set to analyse, it would probably be faster to load the file into a spreadsheet and use that to plot some simple statistics...
- ... but we have twelve files to check, and may have more in the futur !
- In this lesson, we'll learn how to write a function so that we can repeat several operations with a single command.

Your first function

Defining a function if R

- A function start with a name, some arguments and an output :

```
fname <- function(arg1, arg2, ...) {
output = f(arg1, arg2)
return(output)
}
```

Your first function

Defining a function if R

- A function start with a name, some arguments and an output :

  ```
  fname <- function(arg1, arg2, ...) {
  output = f(arg1, arg2)
  return(output)
  }
  ```

- **Exercise 3** : Imagine you want to convert temperatures from Fahrenheit to Kelvin. i) Create a function called `fk` that takes as argument a temperature in F and return a temperature in K (Hint : K = ((F - 32) x 5/9) + 273.15) and ii) test the function for value 32 and 212.

Your first function

Combining functions if `R`

- **Exercise 4** : Create a function called `kc` that takes as argument a temperature in K and return a temperature in C (celsius) (Hint : C = K - 273.15)

Your first function

Combining functions if R

- **Exercise 4** : Create a function called `kc` that takes as argument a temperature in K and return a temperature in C (celsius) (Hint : C = K - 273.15)
- **Exercise 4bis** : Combine `fk` and `kc` in order to create a new function `fc` that converts a temperature in F to C.

Your first function

Combining functions if R

- **Exercise 4** : Create a function called kc that takes as argument a temperature in K and return a temperature in C (celsius) (Hint : C = K - 273.15)

- **Exercise 4bis** : Combine fk and kc in order to create a new function fc that converts a temperature in F to C.

- Nesting functions : Try `kc(fk(32))`. What is the result ?

Your first function

Combining functions if R

- **Exercise 4** : Create a function called `kc` that takes as argument a temperature in K and return a temperature in C (celsius) (Hint : C = K - 273.15)
- **Exercise 4bis** : Combine `fk` and `kc` in order to create a new function `fc` that converts a temperature in F to C.
- Nesting functions : Try `kc(fk(32))`. What is the result ?

⇒ This is our first taste of how larger programs are built : we define basic operations, then combine them in ever-larger chunks to get the effect we want !

Your first function

With words !

- Imagine you have a vector of words `vc` and a ponctuation vector `vp` s.t :

  `vc <- c("Hello", "World")`

  and `vp <- c("***")`

⇒ Remeber the combine function `c` ?

Your first function

With words !

- Imagine you have a vector of words `vc` and a ponctuation vector `vp` s.t :

  `vc <- c("Hello", "World")`

  and `vp <- c("***")`

⇒ Remeber the combine function `c` ?

- **Exercise 5** : create a function `fence` that return `"***" "Hello" "World" "***"`
- **Exercise 5bis** :create a function `outside` that returns the first and the last element of a vector (here : `"***" "***"`). Hint : use the function `length(v)`

Your first function

Default value for arguments

```
mySum <- function(input1, input2 = 10) {
output <- input1 + input2
return(output)
}
```

Your first function

Default value for arguments

```
mySum <- function(input1, input2 = 10) {
output <- input1 + input2
return(output)
}
```

- Q1 : what is the result of `mySum(input1 = 1,3)` ?

Your first function

Default value for arguments

```
mySum <- function(input1, input2 = 10) {
output <- input1 + input2
return(output)
}
```

- Q1 : what is the result of `mySum(input1 = 1,3)` ?
- Q2 : what is the result of `mySum(3)` ?

Your first function

Default value for arguments

```
mySum <- function(input1, input2 = 10) {
output <- input1 + input2
return(output)
}
```

- Q1 : what is the result of `mySum(input1 = 1,3)` ?
- Q2 : what is the result of `mySum(3)` ?
- Q3 : what is the result of `mySum(input2 = 3)` ? Why ?

Your first function

Working with several files

- **Exercise 6 (Difficult !)**
- Write a function called `analyze` that :

  1. takes a filename as an argument
  2. displays the three graphs produced in the previous lesson (average, min and max inflammation over time).

- Hint : analyze(".../data/inflammation-01.csv") should produce the graphs already shown, while analyze(".../data/inflammation-02.csv") should produce corresponding graphs for the second data set. Be sure to document your function with comments.

Your first function

Working with several files

- **Exercise** 6^* **(Very Hard !)**
- Write a function called `analyze_all` that :

  1. takes a path as an argument
  2. displays all filenames in the path
  3. use the function `analyze_all` recursively to display the three graphs produced in the previous lesson (average, min and max inflammation over time).