R-Bootcamp: Series 1

Dr. Matteo Tanadini

February 2022

Preparations

In this first part of this series we are going to move our first steps in **R**. Open **R** and notepad such that you can create a script. Please do not open Rstudio or any other editor yet. Editors will be introduced and used very soon.

As the audience of this course is mixed, we are providing exercises of different level of complexity. Those that are not novices to **R** and feel comfortable can try to solve the "Going further" exercises as well.

1 Exercise: Simple Arithmetics

1.1 Question:

*Use **R** to find the results of the following calculations.*

a) $2 * 10 + 4/(3 + 1)$

b) $2 * (37 - 3^3)$

c) $\sqrt{16} + (10 + 3)$

d) $2^4 + \frac{1*3}{2+3}$

Going further (*)

e) $3 + (\sqrt[3]{27} - 8)$

f) *Guess what the results of* $1/(2^3 - 8)$ *is. Then run this computation with* ***R***

g) $radius = 10 \qquad Area = radius^2 * \pi$

1.2 Question:

Use the results of the computations done in a) b) and c) to continue your analysis. To do that, you must store these results into objects. Keep in mind the naming rules previously seen.

```
## example for a)
results.a <- 2 * 10 + 4 / (3 + 1)
```

Once you created the 3 objects, run the following computations:

```
results.a * results.b / (results.c + results.a^2)
```

1.3 Question:

Which ones of the following are valid object names?

1. results.a
2. results_1

3. 1.results.a
4. results.&_a
5. results.computation.done.in.a
6. A.res
7. A_.res
8. A

---

2 Exercise: Vectors

From here onwards do make use of Rstudio (or any other editor you may find convenient).

2.1 Question:

Try to create the following vectors using the `c()` function and the ":" colon operator. When several options are possible, prefer the shortest one (short in terms of code length).

```
[1]  2 10 13
```

```
[1] 10 11 12 13 14 15
```

```
 [1]  10  20  30  40  50  60  70  80  90 100 110 120 130 140 150 160 170 180
```

2.2 Question:

*Another option to create sequence in **R** is to use the `seq()` function to create the following vectors. Note that the function `seq()` has the following arguments: `from`, `to`, `by` and `length.out`*

```
 [1]  0  1  2  3  4  5  6  7  8  9 10
```

```
[1]  0  2  4  6  8 10
```

```
 [1]  0.00  0.83  1.67  2.50  3.33  4.17  5.00  5.83  6.67  7.50  8.33  9.17
[13] 10.00
```

2.3 Question:

Look at the statements below and try to predict the outcome without running them. You may want to note the predicted results on a sheet of paper.

```r
c(10,20,10) * c(0,1,2)

1:10 * 3

c(1,2,100) / c(1,4,0)

c(1,2,100) + c(1:6)

c(1,2,100) + c(1:7)

c("A", "B", "C") * 2

c("1", "2", "3") * 2
```

Going further (*)

Look at the statements below and try to predict the outcome without running them. You may want to note the predicted results on a sheet of paper.

```r
letters[1:12]

0 / 0

c(TRUE, true, FALSE, FALSE)

c(TRUE, FALSE, TRUE, TRUE) + 1

c(T, FALSE, TRUE)
```

2.4 Question:

Given the vector `seq.1`

```r
seq.1 <- c(1,2,3, 1,2,3, 1,2,3)
```

Access the following elements

1. the very first element
2. the last element
3. the 3 central elements
4. all those elements that are equal or greater than 2

---

3 Exercise: Matrices

3.1 Question:

try to create the following matrices by using the `matrix()`, `rbind()` *or* `cbind()` *function.*

1.

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
```

2.

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
```

3.

```
     [,1] [,2] [,3]
[1,]    3    1    1
[2,]    3    2    2
[3,]    3    3    3
[4,]    3    1    4
[5,]    3    2    5
[6,]    3    3    6
```

4.

```
     [,1] [,2]
[1,] "a"  "c"
[2,] "b"  "d"
```

5.

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,] "a"  "a"  "a"  "a"  "a"  "a"
[2,] "b"  "b"  "b"  "b"  "b"  "b"
[3,] "c"  "c"  "c"  "c"  "c"  "c"
[4,] "d"  "d"  "d"  "d"  "d"  "d"
```

Going further (*)

Create the following matrices with whatever function you find more convenient. Obviously, when there are several viable options, prefer the shortest solution.

6.

```
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1
```

7.

```
     [,1] [,2] [,3] [,4] [,5]
[1,]  999    0    0    0    0
[2,]    0  999    0    0    0
[3,]    0    0  999    0    0
[4,]    0    0    0  999    0
[5,]    0    0    0    0  999
```

8.

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
```

---

4 Exercise: Data frames

4.1 Question:

Try to create the following data frame by using the `data.frame()` function.

```
d.jobs
```

```
             job  salary satisfaction field.exists.since field.old
1 Data scientist      $$      depends                 15     FALSE
2         banker     $$$        money               1000      TRUE
3         artist      $?         high                Inf      TRUE
4           else unknown      unknown                 NA        NA
5       gardener       $       medium                500      TRUE
```

Going further (*)

4.1.1 Can two columns of a data frame have the same name? In theory, this is difficult to achieve, but in reality this is indeed feasible. Show how and explain why this is not a "smart" thing to do.

4.1.2 Can the name of a column contain empty spaces or parentheses?

4.2 Question:

Assume you created the following three vectors:

```
t.num <- 1:10
t.alph <- LETTERS[1:10]
t.logical <- sample(x = c(TRUE, FALSE), size = 10, replace = TRUE)
```

How can you create the following data frame just by using the ***`data.frame()`*** *function and the four vectors created above.*

```
        Numbers Alphabet   Log
case 1        1        A  TRUE
case 2        2        B FALSE
case 3        3        C FALSE
case 4        4        D  TRUE
case 5        5        E  TRUE
case 6        6        F  TRUE
case 7        7        G FALSE
case 8        8        H  TRUE
case 9        9        I  TRUE
case 10      10        J FALSE
```

4.3 Question:

Given the ***`d.jobs`*** *data frame, perform the following selections*

1. get the very first row
2. get the last column (via the "$" symbol)
3. get the second and third column (via the square brackets and column numbers)
4. get the second and third column (via the square brackets and column names)
5. get all rows for which the value of `field.old` is `TRUE`
6. get all rows, but the first one
7. get all rows, but those for which `field.old` is `TRUE`

Going further (*)

1. get the last row in a programmatic way. In other words, your code must return the last row regardless of the number of rows of the data frame
2. get those rows for which the the field exists since 500 or more years
3. create a list with 3 elements
   - the first element must be a logical vector of length 10
   - the second element must be the `d.jobs` dataset
   - the last element must be a named empty list
4. access the following of the list you just created
   - the logical vector (i.e. you must access the fist "slot")
   - the `salary` column present in the second slot
   - the fist and second slot
   - the named list. NB: you must access the list via its name, not its index

---

5 Exercise: Functions

5.1 Question:

The function `rep()` *allows the user to create vector in which a value or some values are replicates. Use it arguments "x", "times", "each" and "length.out" to create the following vectors.*

1.
```
[1] 1 1 1 1 1
```

2.
```
 [1] 3 2 1 3 2 1 3 2 1 3 2 1 3 2 1
```

3.
```
 [1] 3 3 3 3 2 2 2 2 1 1 1 1
```

4.
```
 [1] 1 2 3 1 2 3 1 2 3 1 2
```

5.
```
[1] "A"  "Z"  "ZU" "A"  "Z"  "ZU"
```

Going further (*)

6.
```
 [1] "A" "B" "B" "C" "C" "C" "Z" "Z" "Z" "Z"
```

7.
```
 [1] 0 2 4 6 0 2 4 6 0 2 4 6
```

5.2 Question:

The `rnorm()` *function allows the user to simulate normally distributed values. Look at its help page and answer the following questions:*

1. does the help page only refers to `rnorm()`?
2. what are the arguments of `rnorm()`?
3. which one(s) are compulsory?
4. are there any arguments with default values?

Going further (*)

1. is it possible to provide a vector instead of a value (scalar) to the argument "mean". For example: `rnorm(n = 10, mean = 1:10))`? What would happen in this case?
2. without looking at the help page, how can I find out what the arguments of the `pnorm()` function are?
3. in which package is the `rnorm()` function available?

5.3 Question:

Given the following ***R*** *command*

```r
## orginal command:
plot(x = 1:10, y = 101:110, main = "First graph", type = "b")
```

Decide whether the commands below are fully equivalent to the "original command". Make use of the help page instead of running the commands.

1.

```
plot(1:10, 101:110, main = "First graph", type = "b")
```

2.

```
plot(1:10, 101:110, m = "First graph", ty = "b")
```

3.

```
plot(main = "First graph", type = "b", x = 1:10, y = 101:110)
```

4.

```
plot(x = 1:10, y = 101:110,
     main = "First graph",
     type = "b")
```

5.

```
plot(1:10, 101:110, "b", main = "First graph")
```

---

6 Exercise: Vectors, matrices, data frame and functions

6.1 Question:

Write a short exercise that about one or more arguments seen up to this point. Solving the exercise should take at most a few minutes. Once you have the exercise ready, swap it with your neighbour. Finally discuss the solutions together and ask to the lecturers if needed

Go quickly over all arguments seen so far. Together with your neighbour prepare a question about a topic that is not yet 100% clear to you. Upload the question in the forum of this course (on ILIAS).

---

7 Exercise: Importing data

7.1 Question:

*Working directory: Assume you have two folders on your "Desktop". One is named "folder_A" and contains an **R**-script (i.e "AnalysisFebruary.R") and a data file (i.e. "DatasetCheese.csv"). The other folder, named "folder_B", contains a data file (i.e. "DatasetHospital.xlsx").*

1. *If you start Rstudio by opening the **R**-script file, where is you working directory?*

2. *Assume you started Rstudio by opening the **R**-script file in "folder_A" and you want to import the data file "DatasetCheese.csv". What path do you need to provide to a function to import the data?*

3. *Assume you started Rstudio by opening the **R**-script file in "folder_A" and you want to import the data file "DatasetHospital.xlsx". What path do you need to provide to a function to import the data?*

3. *Assume you started Rstudio by launching the application and you want to import the data file "DatasetHospital.csv". What path do you need to provide to a function to import the data?*

4. *Assume you started Rstudio by opening the **R**-script file in "folder_A". Would the following command work?*

```
d.Cheese <- read.csv("DatasetCheese.csv", header = TRUE)
```

5. *Assume you started Rstudio by launching the application and your current directory happens to be the Desktop. Would the following command work?*

```r
d.Cheese <- read.csv("folder_A/Datasetcheese.csv", header = TRUE)
```

7.2 Question:

Try to import the following datasets without using the Rstudio interface. NB: all datasets can be found in the "DataSets" folder. The focus of this exercise in not the path but rather the functions to import data and their arguments. Therefore, you can locate of data files where you prefer.

1. *Import the dataset "Framingham.dat"*
2. *Import the dataset "Blaueeier.txt"*
3. *Import the dataset "birthrates.csv" (hint: use the function* `read.csv()` *or* `read.csv2()`*)*
4. *Import the dataset "birthrates_WithDescription.csv". Before reading the data, open the file and decide on how to import the data (hint: you may want to ignore the first few rows).*

7.3 Question:

Try to import the following datasets using the Rstudio interface.

1. *Import the dataset "Framingham.dat"*
2. *Import the dataset "Blaueeier.txt"*
3. *Import the dataset "BlaueEier.xlsx"*

7.4 Question:

Going further (*)

1. *How can you read only the first 10 rows of the "BlaueEier.txt" file.*
2. *How can you import a very large dataset in a time-efficient way? Use your friend Google to find a solution*
3. *Try to import a your own dataset that is located on you computer. Discuss the results/difficulties you may encounter with the course instructors.*
4. *Get the dataset Credibility.dat at "http://stat.ethz.ch/Teaching/Datasets"*
