Day 1
Lecture 2:
Getting started with R

Short course on modelling infectious disease dynamics in R

Ankara, Türkiye, September 2025

Dr Juan F Vesga

Aims of the session

- Understand the R engine and syntax
- Introduce initial important notions of R programming
- Learn the basic syntax of R language
- Understand data objects
- Understand user defined functions

How does R store information?

- no files, all in the RAM (i.e. temporary memory)
- data, results, functions, etc. are all R *objects*
- one *object* can be saved / loaded using *saveRDS/readRDS* (output: *.rds* files)
- *several objects* can be saved / loaded using *save/load (*output: *.RData files)*
- an entire session can be saved using save.image

How to create objects?

- General syntax: `object_name <- content`:

Assignment

Object

script

```
toto <- 1:8
toto # check content: 1, 2, 3, ...
```

console

```
## [1] 1 2 3 4 5 6 7 8
```

Result if you check in the console

Note: In R, the syntax "<-" is used to reflect assignment, if you use "=" it will work but can be confusing as is not the convention among R users

How to create objects?

- General syntax: `object_name <- content`:

script

```
toto <- “some text”
toto # reassigned a different value
```

console

```
## [1] "some text"
```

Note: The object “toto” is just an envelope. You can put inside any value you want and change it as you want.

Round numbers: `integer`

```
a <- 1:10
a
```

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

```
class(a)
```

```
## [1] "integer"
```

Decimal numbers: `numeric`

```
b <- c(-0.1, 10.123, pi)
b
```

```
## [1] -0.100000 10.123000 3.141593
```

```
class(b)
```

```
## [1] “numeric"
```

Text: character

```
a <- c("hello world", "hello Turkey!")
a
```

```
## [1] "hello world“     "hello Turkey!"
```

```
class(a)
```

```
## [1] "character"
```

Categorical variables: `factor`

```
a <- factor(c(“red", “blue", “green", “red") )
a
```

```
## [1] red blue green red
## Levels: blue green red
```

```
class(a)
```

```
## [1] “factor"
```

```
levels(a)
```

```
## [1] "blue" "green" “red"
```

Booleans: `logical`

The `logical` type can be `TRUE` or `FALSE`:

```
a <- c(TRUE, FALSE, TRUE,TRUE)
a
```

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

```
class(a)
```

```
## [1] “logical"
```

Booleans can also be interpreted as 0’s (FALSE) and 1’s (TRUE), hence:

```
a + 1
```

```
## [1] 2 1 2 2
```

Vectors

A **vector** stores several values of the **same type** as a one-dimensional array:

```
a <- c(1, 2, -2 , 1.123)
a
```

```
## [1] 1.000 2.000 10.000 -1.000 1.123
```

```
length(a)
```

```
## [1] 5
```

Matrices

A matrix stores several values of the same type as a table:

```
a <- matrix(sample(1:12),ncol = 4)
a
```

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

```
class(a)
```

```
## [1] “matrix”
```

```
dim(a)
```

```
## [1] 3 4
```

Data Frames

A data.frame is a table where variables (columns) can have different types (equivalent to a spreadsheet):

```
a <- data.frame(age = c(10, 54, 3), sex = c("m", "f", "m"))
a
```

```
##   age sex
## 1  10   m
## 2  54   f
## 3   3   m
```

```
class(a)
```

```
## [1] “data.frame”
```

```
dim(a)
```

```
## [1] 3 2
```

Lists

A list is a collection of objects of any types and sizes, stored as different slots. It is a powerful structure to save information

```
age <- c(10, 54, 3)
sex <- factor(c("m", "f", "m"))
swab  <- matrix(
  sample(c("+", "-"), replace = TRUE, 10), nrow = 2,
  dimnames = list(NULL, paste("t", 1:5, sep = "")))
x <- list(age = age, gender = sex, swab_results = swab)
```

Lists (continued)

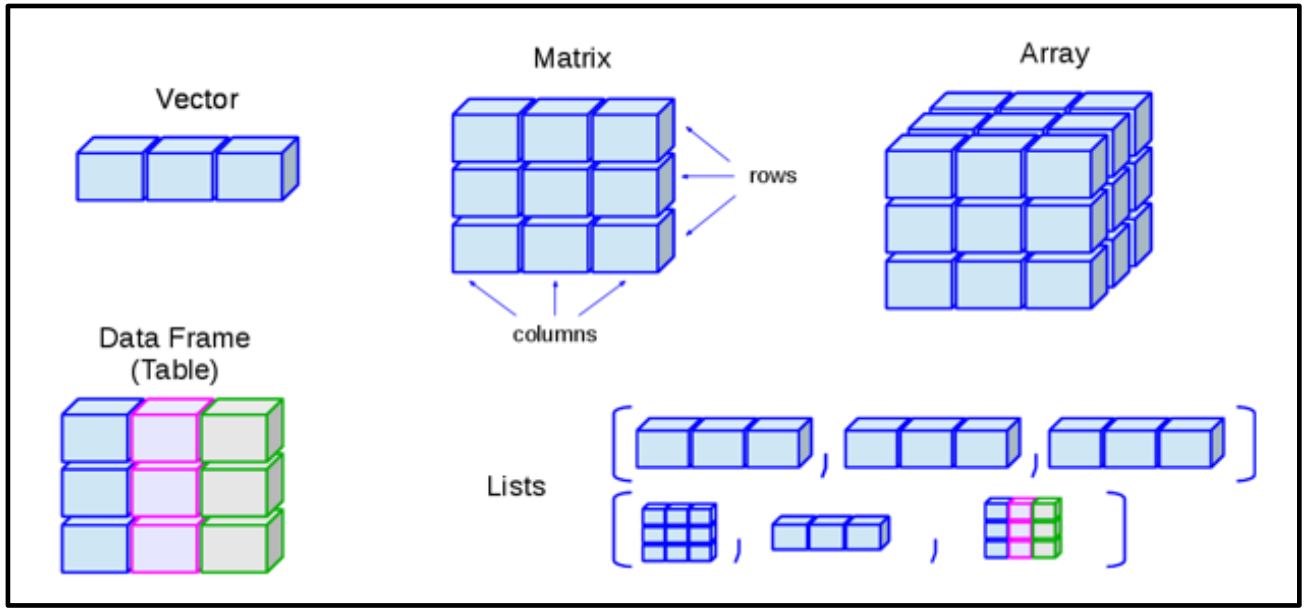
A list is a collection of objects of any types and sizes, stored as different slots. It is a powerful structure to save information

```
x
```

```
## $age
## [1] 10 54  3
##
## $gender
## [1] m f m
## Levels: f m
##
## $swab_results
##      t1  t2  t3  t4  t5
## [1,] "+" "+" "-" "+" "+"
## [2,] "+" "+" "+" "-" "+"
```

Data structures summary

- Vectors
- Matrices and arrays
- Data frames
- Lists
- R has some datasets already loaded



G. Tiwari, 2019. https://medium.com/@tiwarigaurav2512/r-data-types-847fffb01d5b

Accessing contents in an object

The objects we reviewed can be accessed or subsetted by index, name, or logical condition. Using:

- `object_name[]` for a vector
- `object_name[rows, columns]` for a matrix / data.frame
- `object_name[[]]` for a list

- An index is an integer or set of integers that points to the position of an element in an array: `my_vector[2]`

- Lists can be accessed using the name of the slot : `my_list[["age"]]`

- Note that square brackets imply access to an object, while parenthesis contain the arguments of a function

Using functions

- Functions are short-cuts for doing complicated things, e.g.
  - sort(c(2, 1, 3))=c(1, 2, 3)
  - abs(-4)=4

- Functions can be written by:
  - R (i.e. already installed)
  - the user (i.e. you)
  - someone else (in a package)

What is a function?

A set of operations made on a given output

Syntax: function_name(argument1, argument2, …)

Example:

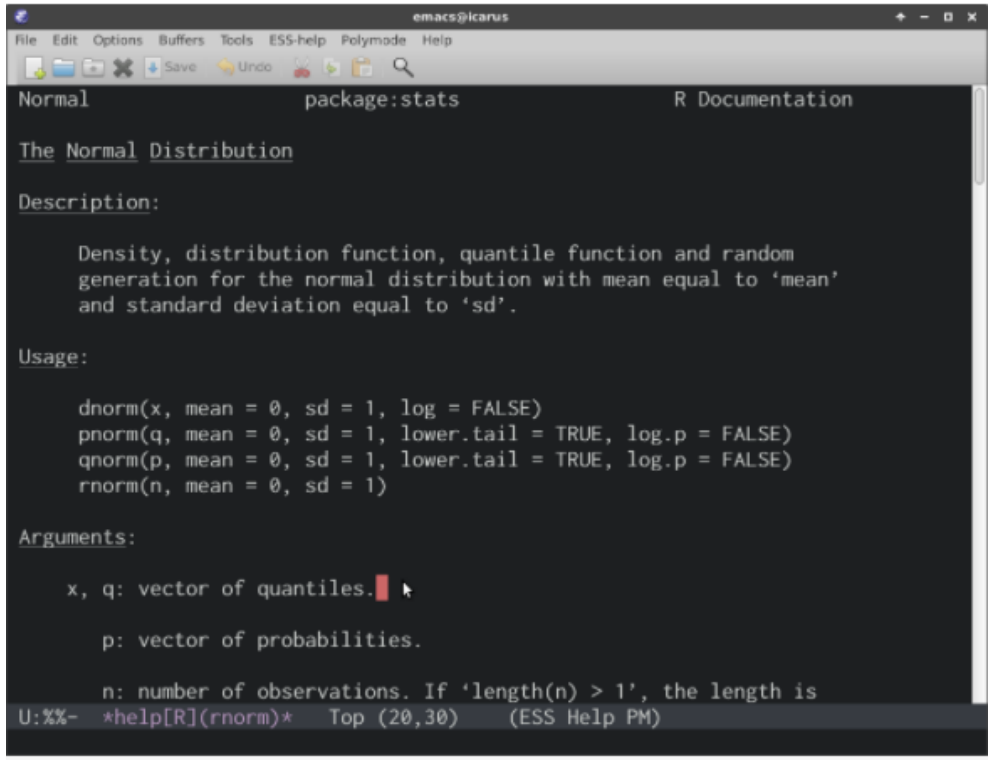
```
rnorm(8, mean=5, sd=3)
```

```
## [1]  2.713883  7.373878  9.364017  1.173302  6.436040  5.624428  7.107626
## [8] -1.727200
```

How to use a function?

When a function is created either by R or the user or a package, its definition can be retrieved using “?”

```
?rnorm
```



User defined functions

Apart from base R functions and functions in packages, you can create your own functions.

Syntax:
```
function_name <- function(argument1, argument2,…)
                    {
                     # perform an operation using your arguments
                     x <-  argument1 + argument2

                     return(x)
                    }
```

- Naming functions is an important part of good coding: use meaningful names! : Sum_arguments (Good); func1 (Bad)

What we should know by now

- How the R engine works
- How to store information using R objects
- Accessing your information
- What are functions
- How functions can be defined by the user

Good R resources to explore

- R for Data Science - https://r4ds.had.co.nz/
- Advanced R http://adv-r.had.co.nz/
- R packages http://r-pkgs.had.co.nz/
- RECONLearn https://www.reconlearn.org/