An intRoduction to R

Univerisity of Padova (IT)

Who aRe you?

- R is an open source software for statistical computing, graphics, and so much more
- RStudio is the perfect IDE for R $\rightarrow$ allows for a better, easier use of R
- R runs on Windows, MacOs, Unix

CalculatoR

```
3 + 2  # plus
3 - 2  # minus
3 * 2  # times
3/2  # divide
sqrt(4)  # square root
log(3)  # natural logarithm
exp(3)  # exponential
```

Use brackets as you would do in a normal equation:

```
(3 * 2)/sqrt(25 + 4)
```

R ignores everything after # (it's a comment)

Assign

The results of the operations can be "stored" into objects with specific names defined by the users.

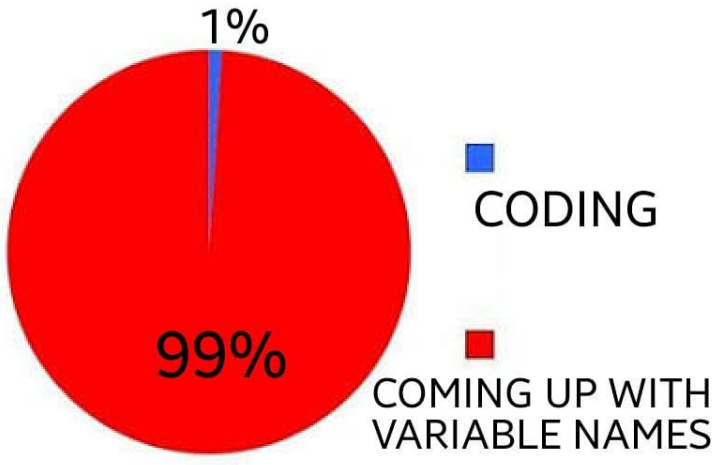
To assign a value to an object, there are two operators:

1. `= x = exp(2^2)`
2. `<- X <- log(2^2)`

The elements on the right are assigned to the object on the left

Careful! `R` is case sensitive: `x` and `X` are two different objects!!!

Variable names



Valid variable names are letters, numbers, dots, underscores (e. g., `variable_name`)

Variables names cannot start with numbers

Again, `R` is case sensitive

Get help

R is open source and it used world wide $\rightarrow$ there's a huge community ready to help you

Just copy & paste any error message or wanring in google or ask google "how to something in r"

Ask R to help you! Type ? in your console followed by the name of the function:

```
?mean()
```

Will show you the help page of the `mean()` function

Be tidy

Organize your files

R projects are the best way to organize your files (and your workflow)

It allows you to have all your files in a folder organized in sub folders

You don't have to worry about the wording directories because it's all there!

By creating a nw project, you can also initialize a shiny app

Create a new R project

File → New project and choose what is best for you (unless you have already initialized a directory for your project, select a new directory):

- R poject "basic"
- R package
- Shiny

and so much more

Take out the trash

The R environment should be always tidy

If it feels like you're losing it, just clean it up:

```
ls()  # list objects in the envrinoment
rm(A)  # remove object A from the environment
rm(list = ls())  # remove everything from the environment
```

Save the environment

It might be useful to save all the computations you have done:

```
save.image("my-computations.RData")
```

Then you can upload the enviroment back:

```
load("my-Computations.RData")
```

When to save the environment

The computations are slow and you need them to be always and easily accessible

The best practice wis to save the script and document it in an `RMarkdown` file

Working directories

If you choose not to use the R projects ({what a bad, bad, bad idea}), you need to know your directories:

```
getwd()  # the working directory in which you are right now

dir()  # list of what's inside the current working directory
```

Change your working directory:

```
setwd("C:/Users/huawei/OneDrive/Documenti/GitHub/RcouRse")
```

Structures in R

Functions and arguments (pt. I)

Almost everything in R is done with functions, consisting of:

- a name: `mean`
- a pair of brackets: `()`
- some arguments: `na.rm = TRUE`

```
mean(1:5, trim = 0, na.rm = TRUE)
```

```
[1] 3
```

Arguments may be set to default values; what they are is documented in `?mean`

Functions and arguments (pt. II)

Arguments can be passed

- without name (in the defined order)
- with name (in arbitrary order) $\rightarrow$ keyword matching

  mean(x, trim $=$ 0.3, na.rm $=$ TRUE)

No arguments? No problems, just brackets:

```
ls(), dir(), getwd()
```

Want to see the code of a function? Just type its name in the console without brackets:

```
mean
```

Vectors

Vectors are created by **c**ombining together different objects

Vectors are created by using the `c()` function.

All elements inside the `c()` function **must** be separated by a comma

Different types of objects $\rightarrow$ types of vectors:

- `int`: numeric integers
- `num`: numbers
- `logi`: logical
- `chr`: characters
- `factor`: factor with different levels

int and num

`int`: refers to integer -3, -2, -1, 0, 1, 2, 3

```
months = c(5, 6, 8, 10, 12, 16)
```

```
[1]  5  6  8 10 12 16
```

`num`: refers to all numbers from $-\infty$ to ∞ 1.0840991, 0.8431089, 0.494389, -0.7730161, 2.9038161, 0.9088839

```
weigth = seq(3, 11, by = 1.2)
```

logi

Logical values can be `TRUE` (`T`) or `FALSE` (`F`)

```
v_logi = c(TRUE, TRUE, FALSE, FALSE, TRUE)
```

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

logical vectors are often obtained from a comparison:

```
months > 30
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE
```

chr and factor

chr: characters a, b, c, D, E, F

```
v_chr = c(letters[1:3], LETTERS[4:6])
```

```
[1] "a" "b" "c" "D" "E" "F"
```

factor: use numbers or characters to identify the variable levels

```
ses = factor(c(rep(c("low", "medium", "high"), each = 3)))
```

```
[1] low    low    low    medium medium medium high   high   h
Levels: high low medium
```

Change order of the levels:

```
ses1 = factor(ses, levels = c("medium", "high", "low"))
```

```
[1] low    low    low    medium medium medium high   high   h
Levels: medium high low
```

Create vectors

Concatenate elements with `c()`:

```
vec = c(1, 2, 3, 4, 5)
```

Sequences:

```
-5:5  # vector of 11 numbers from -5 to 5
```

```
seq(-3, 3, by = 0.5)  # sequence in steps of 0.5 from -3 to 3
```

Repeating elements:

```
rep(1:3, 4)
```

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

```
rep(c("condA", "condB"), each = 3)
```

```
[1] "condA" "condA" "condA" "condB" "condB" "condB"
```

Don't mix them up unless you truly want to

`int` + `num` → `num`

`int/num` + `logi` → `int/num`

`int/num` + `factor` → `int/num`

`int/num` + `chr` → `chr`

`chr` + `logi` → `chr`

Vectors and operations

Vectors can be summed/subtracted/divided and multiplied with one another

```
a = c(1:8)
a
```

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

```
b = c(4:1)
b
```

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

```
a - b
```

```
[1] -3 -1  1  3  1  3  5  7
```

If the vectors do not have the same length, you get a warning

Vectors and operations PT. II

The function is applied to each value of the vector:

```
sqrt(a)
```

```
[1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.6
```

The same operation can be applied to each element of the vector:

```
(a - mean(a))^2  # squared deviation
```

```
[1] 12.25  6.25  2.25  0.25  0.25  2.25  6.25 12.25
```

Matrices and arrays

Create a 3×4 matrix:

```
A = matrix(1:12, nrow = 3, ncol = 4, byrow = TRUE)
```

Label and transpose:

```
rownames(A) = c(paste("a", 1:3))  # colnames()
t(A)  # transpose matrix
```

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

Matrices and arrays

Matrix can be created by concatening columns or rows:

```
cbind(a1 = 1:4, a2 = 5:8, a3 = 9:12)  # column bind
rbind(a1 = 1:4, a2 = 5.8, a3 = 9:12)  # row bind
```

Matrices and arrays

Arrays have more than two dimensions:

```
array(c(A, 2 * A), c(3, 4, 2))  # 3 x 4 x 2 array
```

```
, , 1

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

, , 2

     [,1] [,2] [,3] [,4]
[1,]    2    4    6    8
[2,]   10   12   14   16
[3,]   18   20   22   24
```

Lists

Can store different objects (e.g., vectors, data frames, other lists):

```
my_list = list(w = weigth, m = months, s = ses1, a = A)
```

The components of the list can be extracted with `$` or `[[]]` and the name (or position) of the component:

Extract `months`:

```
my_list[["m"]]  # my_list$a
```

```
[1]  5  6  8 10 12 16
```

Extract `weight`:

```
my_list[[1]]  # my_list$months or my_list[['a']]
```

```
[1]  3.0  4.2  5.4  6.6  7.8  9.0 10.2
```

The king of data structure: data frames

Data frames are lists that consist of vectors and factors of equal length. The rows in a data frame refer to one unit:

Execersizes

- Open a new R script
- Create one vector for each type (`int`, `num`, `chr`, `logi`, `factor`) and assign each of them to an object
- Compute the mean of the `int` and `num` vectors
- Standardize the values of the `int` and `num` vectors and store them in two new objects:

$$z = \frac{x_i - \bar{X}}{sd}$$

- Create a new vector by combining together the `logi` and `int` vectors
- Add the `logi` vector to the `num` vector