R Basics

Rüçhan Eker, Jean Monlong, Margot Zahm

Why R?

Why R?

Simple

- Interpretative language (no compilation needed)
- No manual memory management
- Vectorized

Free

- Widely used, vast community of R users
- Good life expectancy

Why R?

Flexible

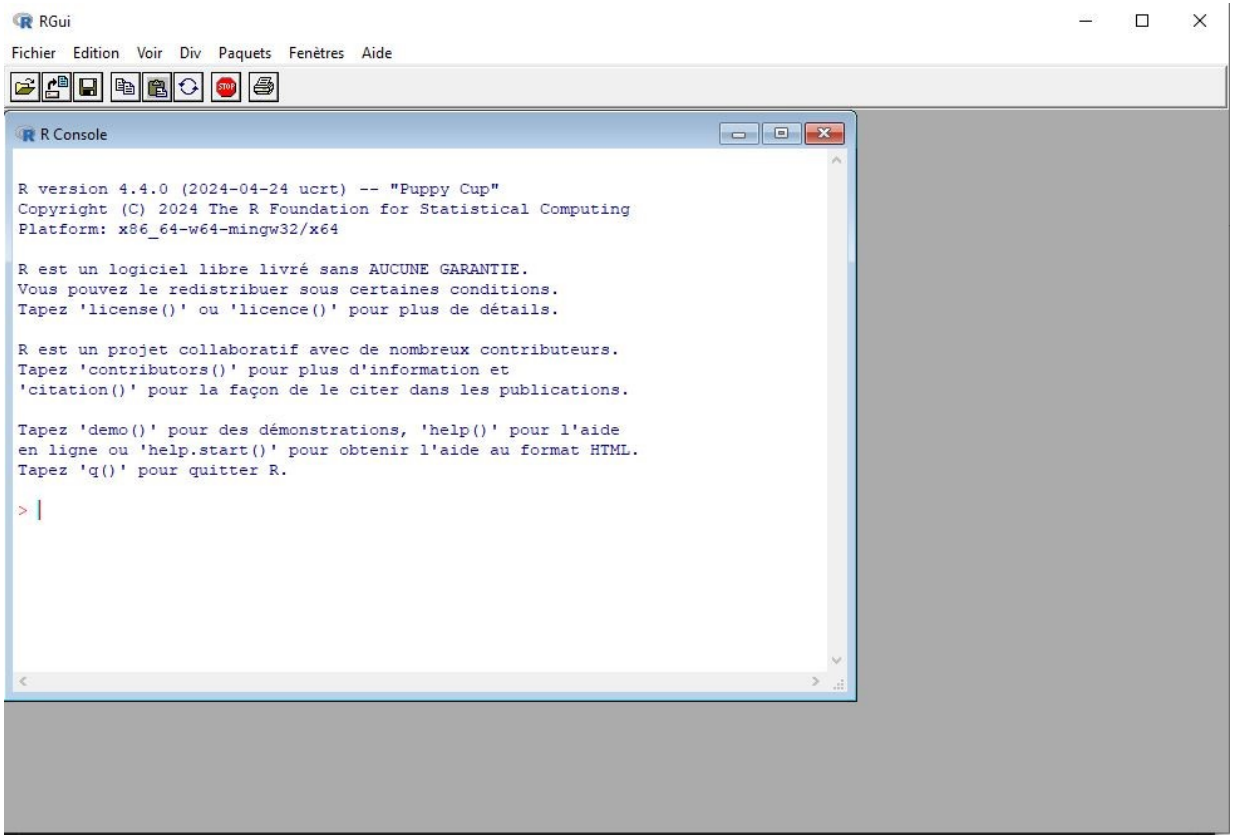
- Open-source: anyone can see/create/modify
- Multiplatform: Windows, Mac, Unix… It works everywhere

Trendy

- More and more packages
- More and more popular among data scientists and (now) biologists

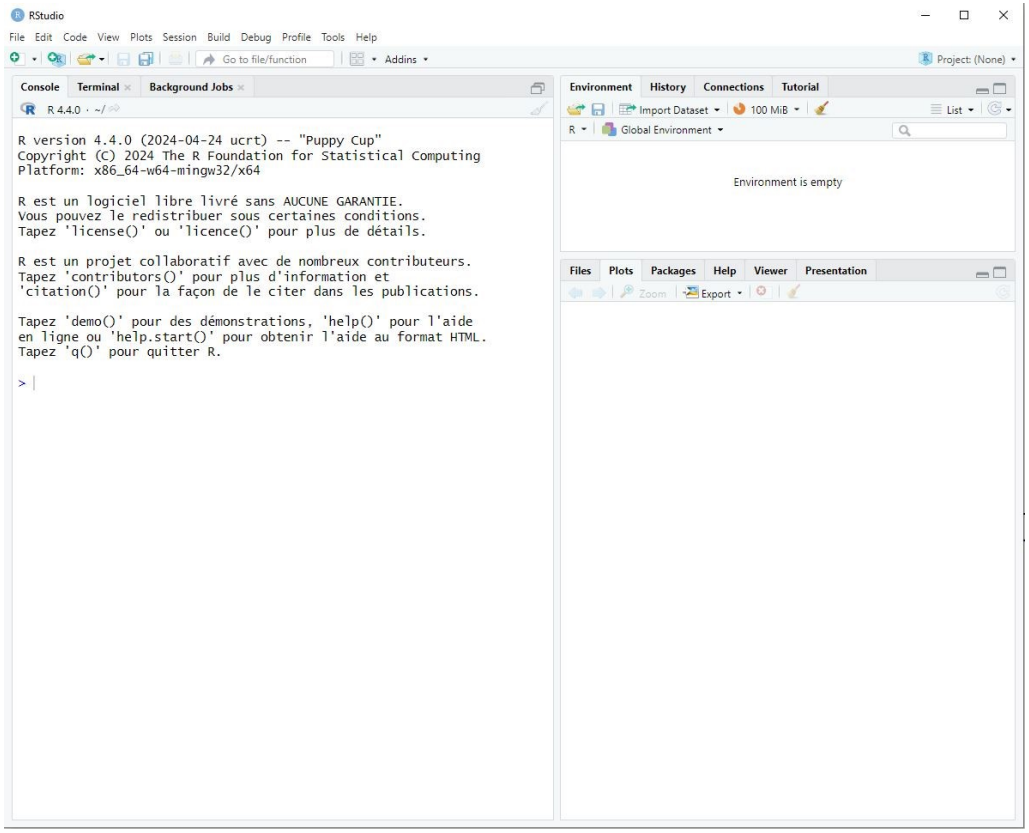
Workshop Setup

Open 



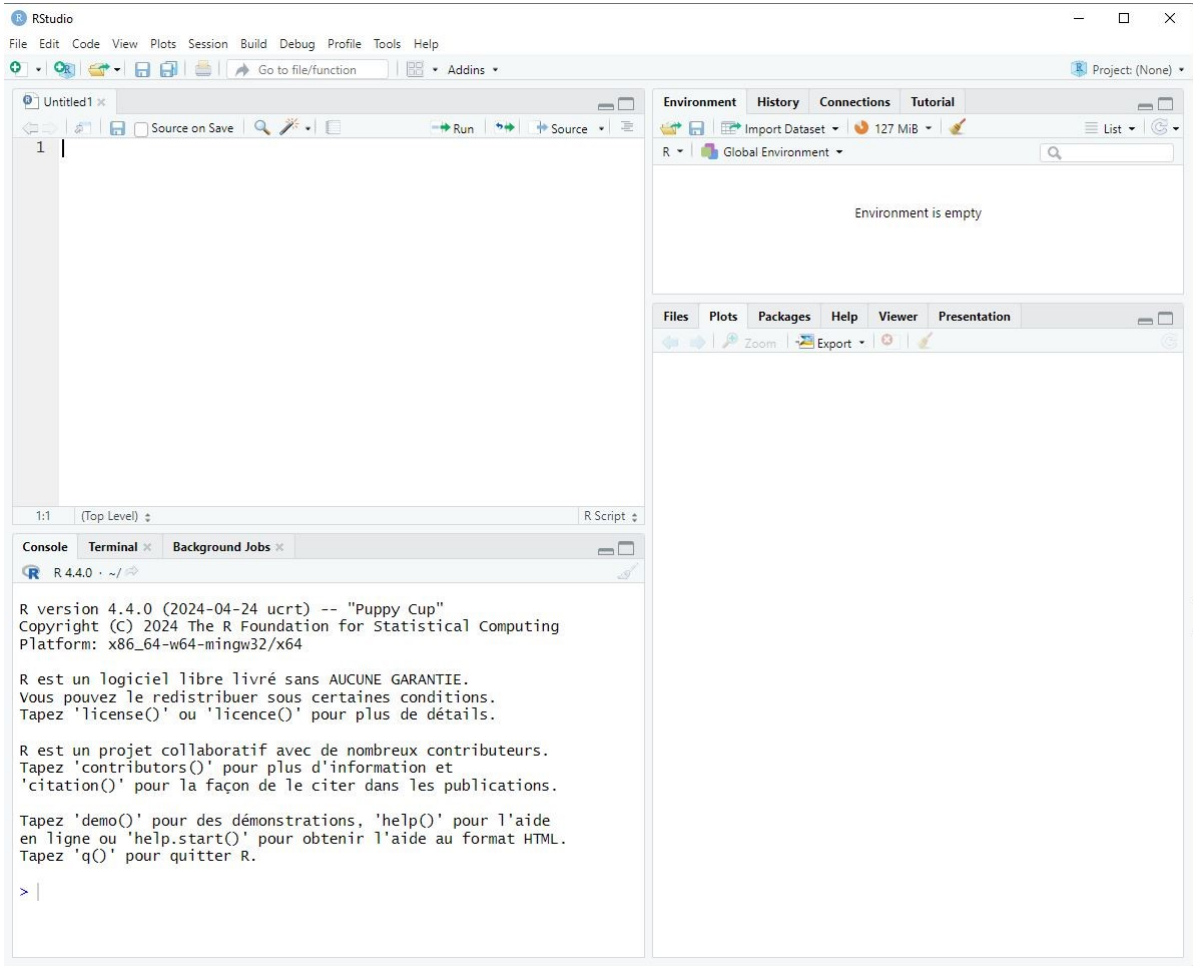
Workshop Setup

Open RStudio®

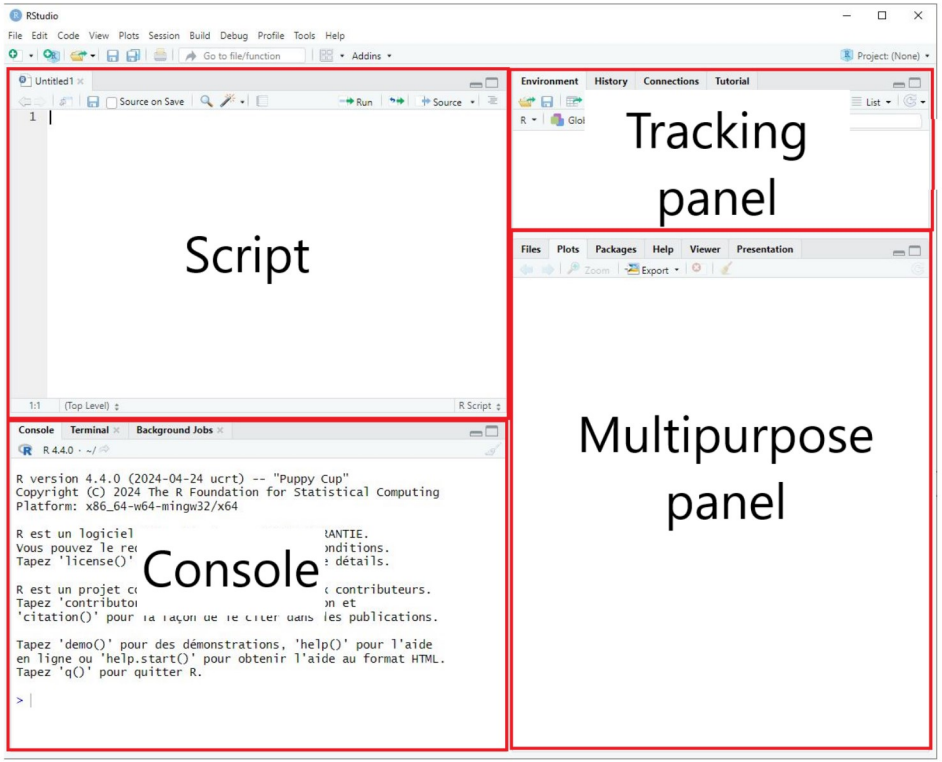


Workshop Setup

Open a new R script file (File > New File > R Script)



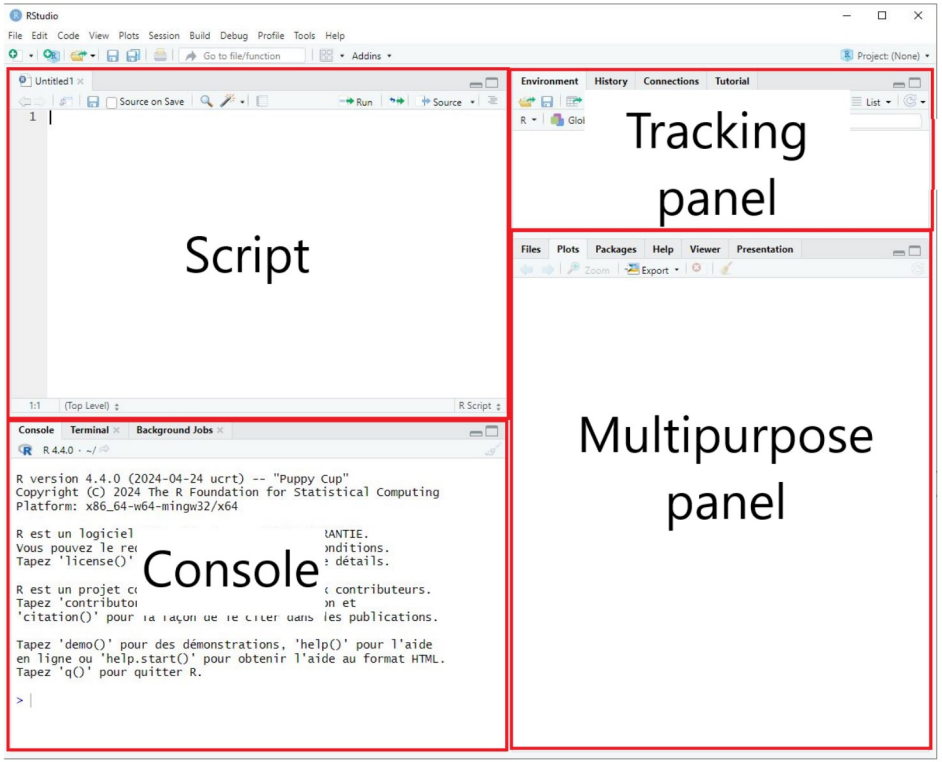
Workshop Setup



Console

- Where R is running
- You can write and run the commands directly here
- Your command executes when you press Enter

Workshop Setup

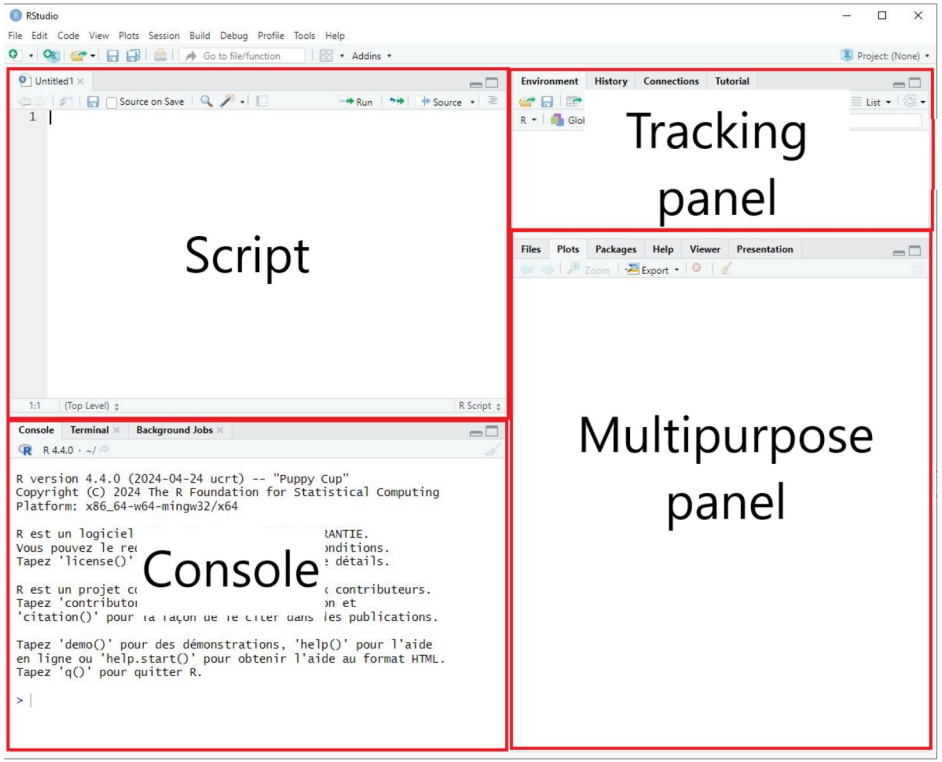


Console

Script

- A text file with commands. *Extension .R*
- To keep a trace of your analysis
- Highly recommended
- Run commands from a script to the console with Run button

Workshop Setup



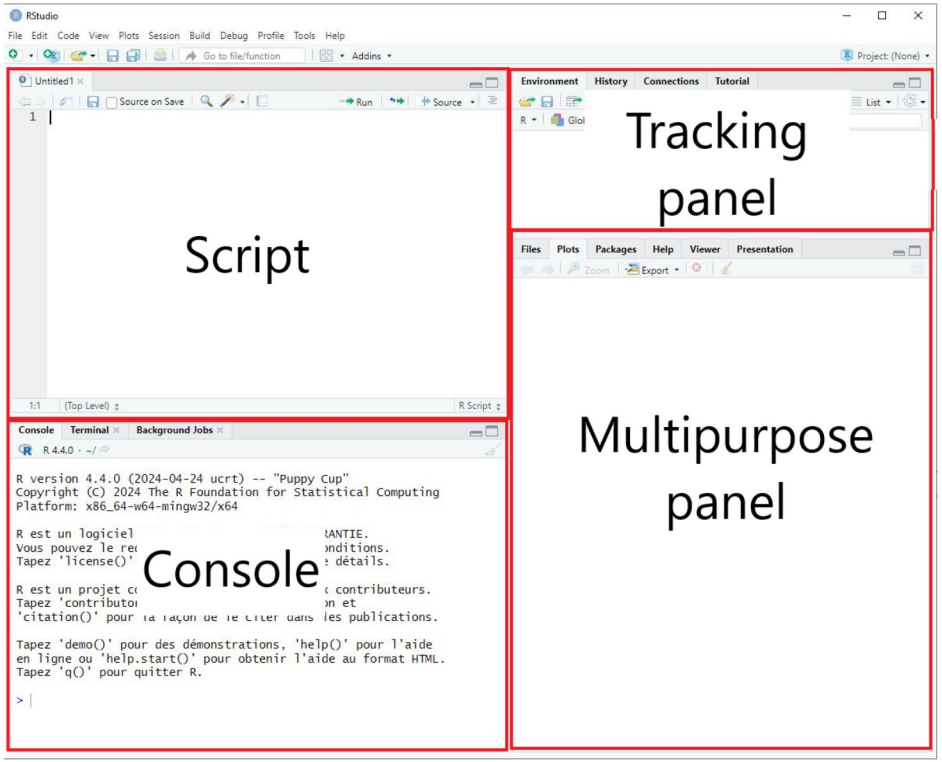
Console

Script

Tracking panel

- List all **variables** you generated
- An **history** of the commands you ran

Workshop Setup



Console

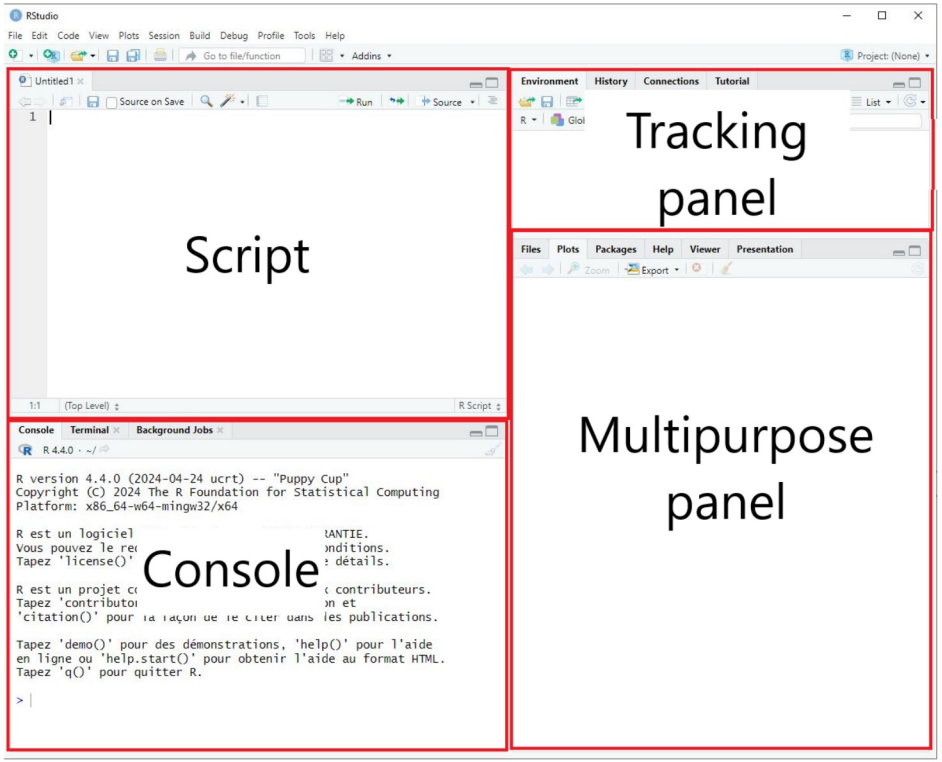
Script

Tracking panel

Multipurpose panel

Check **files** in your computer, see **plots**, manage **packages**, read **help** section of a function.

Workshop Setup



Console

Script

Tracking panel

Multipurpose panel

Caution

Write everything you do in scripts to avoid loosing your work.

When you get an error

1. Read the command, look for typos
2. Read the error message
3. 1. and 2. again
4. Raise your hand, someone will assist you

Tip

Solving errors is an important skill to learn.

Objects

Objects - Overview

Unit type

- **numeric** e.g. numbers

```
[1] 0.1
[1] 42
[1] -1e+07
```

- **logical Binary** two possible values

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

- **character** e.g. words between "

```
[1] "male"
[1] "ENSG007"
[1] "Allez les bleus"
```

- **comment**: line starting by #

```
1  # This is a comment line
2  # I can write everythin I want
```

Tip

Comment your script to help you remember what you have done.

Objects - Overview

Complex type

- **vector**: Ordered collection of elements of the same type

```
[1] 1 3 5 7 9
```

- **list**: Flexible container, mixed type possible. Recursive

```
$name
[1] "John Doe"

$age
[1] "40"

$skills
[1] "sing"  "dance" "run"

$glasses
[1] FALSE
```

Objects - Overview

Complex type

- **matrix**: Table of elements of the same type

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

```
     [,1]    [,2] [,3]
[1,] "one"   "1"  "4"
[2,] "two"   "2"  "5"
[3,] "three" "3"  "6"
```

- **data.frame**: Table of mixed type elements

```
  Name Age IsStudent
1 John  25      TRUE
2 Jane  30     FALSE
```

Note

These are the basic complex types. It exists a lot of different complex objects which mix all these basic objects.

Objects - Naming conventions

- Use letters, numbers, dot or underline characters
- Start with letter or the dot not followed by a number
- Some names are forbidden (ex. `if`, `else`, `TRUE`, `FALSE`)
- Correct: valid.name, valid_name, valid2name3
- Incorrect: valid name, valid-name, 1valid2name3

Tip

Avoid random names such as `var1`, `var2`. Use significant names: `gene_list`, `nb_elements`

Objects - Assign a value

> The name of the object followed by the assignment symbol and the value.

```
valid.name_123 = 2
valid.name_123
```

```
[1] 2
```

```
valid.name_123 <- 2
valid.name_123
```

```
[1] 2
```

```
valid.name_123 = 4
valid.name_123
```

```
[1] 4
```

Objects - Arithmetic operators

You can use operators on objects to modify them. Depending on the object format, operators have different behaviors and some are forbidden.

- addition: +
- substraction: -
- multiplication: *
- division: /
- exponent: ^ or **
- integer division: %/%
- modulo: %%

```
1  2+3
```

```
[1] 5
```

```
1  5-2
```

```
[1] 3
```

```
1  4*3
```

```
[1] 12
```

```
1  10/2
```

```
[1] 5
```

```
1  2^3
```

```
[1] 8
```

```
1  10%/%3
```

```
[1] 3
```

```
1  10%%3
```

```
[1] 1
```

Objects - Arithmetic operators

Exercise

1. Create a numeric object
2. Multiply it by 6
3. Add 21
4. Divide it by 3
5. Substract 1
6. Halve it
7. Substract its original value

Objects - Arithmetic operators

> **Correction**

```
my_number = 42
my_new_number = my_number * 6
my_new_number = my_new_number + 21
my_new_number = my_new_number / 3
my_new_number = my_new_number - 1
my_new_number = my_new_number / 2
my_new_number = my_new_number - my_number
my_new_number
```

```
[1] 3
```

Objects - Arithmetic operators

> **Exercise**
>
> Try to raise errors using operators.

```
1 #"Six" * "Three"
2 5/0
```

```
[1] Inf
```

```
1 0/0
```

```
[1] NaN
```

Objects - Function

- A function is a tool to create or modify an object
- Format: `function_name(object, parameter1 = ..., parameter2 = ...)`
- Read the help manual to know more about a function (`help`, `?` or `F1`)

```
1 sqrt(9)
```

```
[1] 3
```

```
1 sqrt.valid.name_123 = sqrt(valid.name_123)
2 sqrt.valid.name_123
```

```
[1] 2
```

```
1 help(sqrt)
2 ?sqrt
```

Note

Some functions are in the default installation of R. Other functions come from packages. You can also create your own functions.

Vectors

Vectors

> **vector construction**
>
> - `c()` Concatenate function
> - `1:10` Vector with numbers from 1 to 10

```
 1  luckyNumbers = c(4,8,15,16,23,42)
 2  luckyNumbers
```

```
[1]  4  8 15 16 23 42
```

```
 1  oneToTen = 1:10
 2  oneToTen
```

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

```
 1  tenOnes = rep(1,10)
 2  tenOnes
```

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

```
 1  samples = c("sampA", "sampB")
 2  samples
```

```
[1] "sampA" "sampB"
```

Vectors

vector construction

- `c()` Concatenate function
- `1:10` Vector with numbers from 1 to 10

Extra

- `seq` Create a sequence of numbers
- `rep` Repeat elements several times
- `runif` Simulate random numbers from Uniform distribution. Same for `rnorm`, `rpois`...

Exercise - Create some vectors

Instructions

- Create a `vector` withe 7 *numeric* values
- Create a `vector` with 7 *character* values

Vectors

Manipulation

Using index/position between []

Characterization

- `length()` Number of elements in the vector
- `names()` Get or set the names of the vector's value

```
 1  luckyNumbers[3]
```

```
[1] 15
```

```
 1  luckyNumbers[2:4]
```

```
[1]  8 15 16
```

```
 1  luckyNumbers[2:4] = c(14,3,9)
 2  luckyNumbers
```

```
[1]  4 14  3  9 23 42
```

```
 1 length(luckyNumbers)
```

```
[1] 6
```

```
 1 names(luckyNumbers)
```

```
NULL
```

```
 1 names(luckyNumbers) = c("frank", "henry", "philip", "steve", "tom", "franci
 2 names(luckyNumbers)
```

```
[1] "frank"   "henry"   "philip"  "steve"   "tom"     "francis"
```

```
 1 luckyNumbers["philip"]
```

```
philip
     3
```

Vectors

Manipulation

- `sort()` Sort a vector
- `sample()` Shuffle a vector
- `rev()` Reverse a vector

```
1 luckyNumbers
```

```
 frank  henry philip  steve    tom francis
     4     14      3      9     23      42
```

```
1 sort(luckyNumbers)
```

```
philip  frank  steve  henry    tom francis
     3      4      9     14     23      42
```

```
1 sort(c(luckyNumbers, 1:10))
```

```
                philip           frank
     1      2      3      3      4      4      5      6      7
8
```

```
  steve                   henry      tom francis
      9       9      10      14       23      42
```

```
1 rev(luckyNumbers)
```

```
francis      tom    steve   philip    henry    frank
     42       23        9        3       14        4
```

```
1 sample(1:10)
```

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

Extra

- `sort()/sample()` Explore extra parameters
- `order()` Get the index of the sorted elements

Vectors

Exploration

- `head()/tail()` Print the first/last values
- `summary()` Summary statistics
- `min()/max()/mean()/median()/var()` Minimum, maximum, average, median, variance
- `sum` Sum of the vector's values

```
1 head(samples)
```

```
[1] "sampA" "sampB"
```

```
1 summary(luckyNumbers)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   3.00    5.25   11.50   15.83   20.75   42.00 
```

```
1 mean(luckyNumbers)
```

```
[1] 15.83333
```

```
1 min(luckyNumbers)
```

```
[1] 3
```

Extra

- `log/log2/log10` Logarithm functions
- `sqrt` Square-root function

Vectors

Arithmetic operators

- Simple arithmetic operations over all the values of the vector
- Or values by values when using vectors of same length
- Arithmetic operations: +, -, *, /
- Other exist but let's forget about them for now

```
1 luckyNumbers + 2
```

```
frank   henry  philip   steve     tom francis
    6      16       5      11      25      44
```

```
1 luckyNumbers * 4
```

```
frank   henry  philip   steve     tom francis
   16      56      12      36      92     168
```

```
1 luckyNumbers - luckyNumbers
```

```
frank   henry  philip   steve     tom francis
```

```
     0        0        0        0        0        0
```

```
1 luckyNumbers / 1:length(luckyNumbers)
```

```
 frank   henry  philip   steve     tom francis
  4.00    7.00    1.00    2.25    4.60    7.00
```

Exercise - Guess my favorite number

Instructions

1. Create a vector with 5 numeric values
2. Multiply it by 6
3. Add 21
4. Divide it by 3
5. Substract 1
6. Halve it
7. Substract its original values

```
my_numbers = rnorm(5)
my_favorite_number = my_numbers*6
my_favorite_number = my_favorite_number + 21
my_favorite_number = my_favorite_number / 3
my_favorite_number = my_favorite_number - 1
```

```r
my_favorite_number = my_favorite_number * 2
my_favorite_number = my_favorite_number / 2
my_favorite_number - my_numbers
```

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