Polish Language(s) and Digital Humanities Using R

G. Moroz

2020

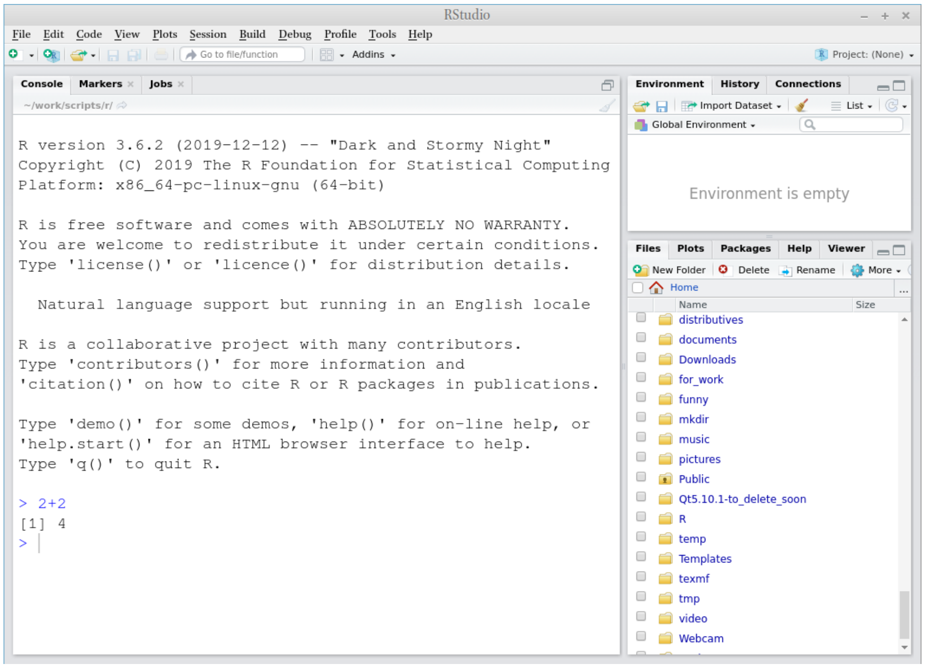
Contents

Chapter 1

Prerequisites

Before the classes I would like to ask you to follow the instructions mentioned below to prepare your device for the class work:

- install **R** from the following link: https://cloud.r-project.org/
- install **RStudio** from the following link: https://rstudio.com/products/rstudio/download/#download (FREE version, no need to pay!)
- after the installation run the RStudio program, type `2+2`, and press `Enter`.



```
R version 3.6.2 (2019-12-12) -- "Dark and Stormy Night"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 2+2
[1] 4
>
```

Environment is empty

If you see something like this, then you are well prepared for classes.

- Go to the https://rstudio.cloud/ website and sign up there. This is optional, but it will be a backup version, if something will not work on your computer.

Chapter 2

Introduction to R and RStudio

2.1 Introduction

2.1.1 Why data science?

Data science is a new field that is actively developing lately. This field merges computer science, mathematics, statistics, and it is hard to say how much science in data science. In many scientific fields a new data science paradigm arises and even forms a new sub-field:

- Bioinformatics
- Crime data analysis
- Digital humanities
- Data journalism
- Data driven medicine
- …

There are a lot of new books "Data Science for …":

- psychologists (Hansjörg, 2019)
- immunologists (Thomas and Pallett, 2019)
- business (Provost and Fawcett, 2013)
- public policy (Brooks and Cooper, 2013)
- fraud detection (Baesens et al., 2015)
- …

Data scientists need to be able to:

- gather data
- transform data

- visualize data
- create a statistical model based on data
- share and represent the results of this work
- organize the whole workflow in a reproducible way

2.1.2 Why R?

R (R Core Team, 2019) is a programming language with a big infrastructure of packages that helps to work in different fields of science and computer technology.

There are several alternatives:

- Python (VanderPlas, 2016; Grus, 2019)
- Julia (Bezanson et al., 2017)
- bash (Janssens, 2014)
- java (Brzustowicz, 2017)
- ...

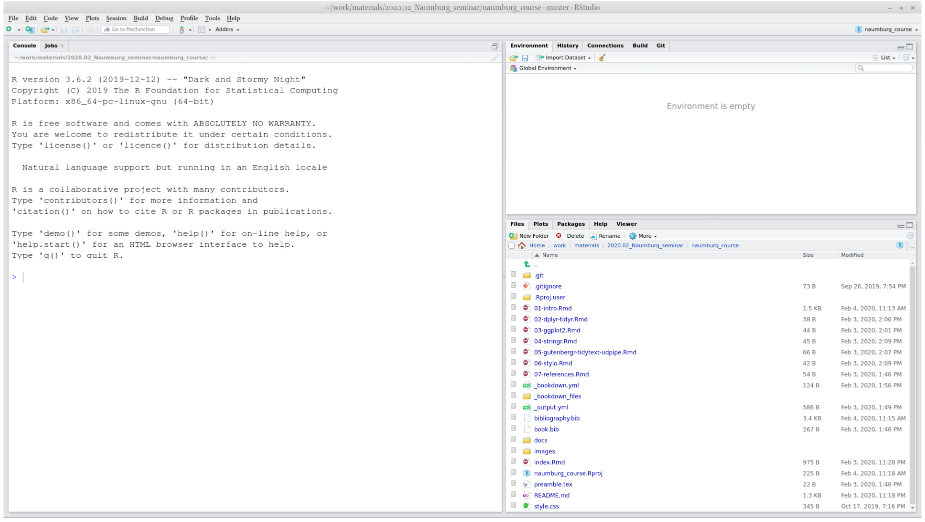
You can find some R answers here:

- R for data science (Wickham and Grolemund, 2016), it is online
- R community
- stackoverflow
- any search engine you use
- ...

2.2 Introduction to RStudio

R is the programming language. RStudio is the most popular IDE (Integrated Development Environment) for R language.

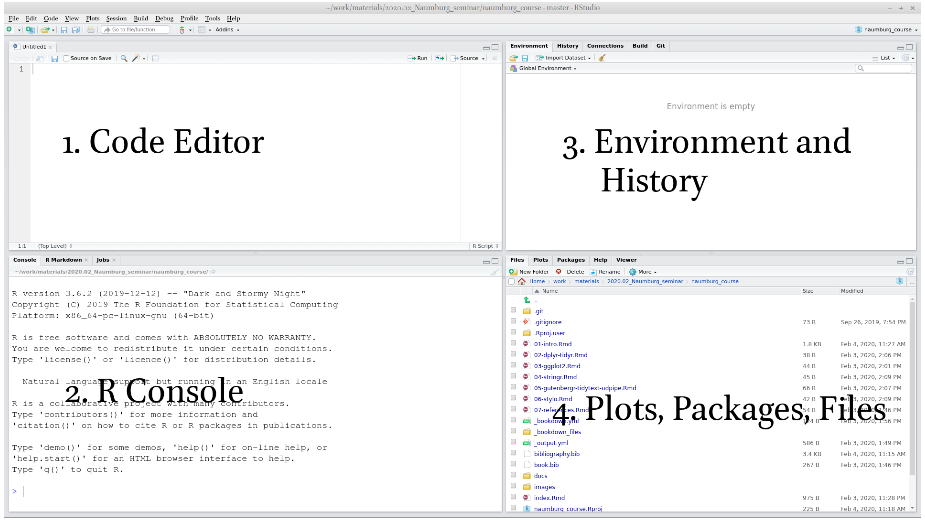
When you open RStudio for the first time you can see something like this:


~/work/materials/2020.02_Naumburg_seminar/naumburg_course - master - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
naumburg_course
Console
Jobs
~/work/materials/2020.02_Naumburg_seminar/naumburg_course/
Environment
History
Connections
Build
Git
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home › work › materials › 2020.02_Naumburg_seminar › naumburg_course

```
R version 3.6.2 (2019-12-12) -- "Dark and Stormy Night"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

Name	Size	Modified
..		
.git		
.gitignore	73 B	Sep 26, 2019, 7:54 PM
.Rproj.user		
01-intro.Rmd	1.5 KB	Feb 4, 2020, 11:13 AM
02-dplyr-tidyr.Rmd	38 B	Feb 3, 2020, 2:06 PM
03-ggplot2.Rmd	44 B	Feb 3, 2020, 2:01 PM
04-stringr.Rmd	45 B	Feb 3, 2020, 2:09 PM
05-gutenbergr-tidytext-udpipe.Rmd	66 B	Feb 3, 2020, 2:07 PM
06-stylo.Rmd	42 B	Feb 3, 2020, 2:09 PM
07-references.Rmd	54 B	Feb 3, 2020, 1:46 PM
_bookdown.yml	124 B	Feb 3, 2020, 1:56 PM
_bookdown_files		
_output.yml	586 B	Feb 3, 2020, 1:49 PM
bibliography.bib	3.4 KB	Feb 4, 2020, 11:15 AM
book.bib	267 B	Feb 3, 2020, 1:46 PM
docs		
images		
index.Rmd	975 B	Feb 3, 2020, 11:28 PM
naumburg_course.Rproj	225 B	Feb 4, 2020, 11:18 AM
preamble.tex	22 B	Feb 3, 2020, 1:46 PM
README.md	1.3 KB	Feb 3, 2020, 11:18 PM
style.css	345 B	Oct 17, 2019, 7:16 PM

When you press ⧉ button at the top of the left window you will be able to see all four panels of RStudio.


~/work/materials/2020.02_Naumburg_seminar/naumburg_course - master - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
naumburg_course
Untitled1
Source on Save
Run
Source
1. Code Editor
1:1
(Top Level)
R Script
Environment
History
Connections
Build
Git
Import Dataset
List
Global Environment
Environment is empty
3. Environment and History
Console
R Markdown
Jobs
~/work/materials/2020.02_Naumburg_seminar/naumburg_course/
2. R Console
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Home › work › materials › 2020.02_Naumburg_seminar › naumburg_course
4. Plots, Packages, Files

```
R version 3.6.2 (2019-12-12) -- "Dark and Stormy Night"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

Name	Size	Modified
..		
.git		
.gitignore	73 B	Sep 26, 2019, 7:54 PM
.Rproj.user		
01-intro.Rmd	1.8 KB	Feb 4, 2020, 11:27 AM
02-dplyr-tidyr.Rmd	38 B	Feb 3, 2020, 2:06 PM
03-ggplot2.Rmd	44 B	Feb 3, 2020, 2:01 PM
04-stringr.Rmd	45 B	Feb 3, 2020, 2:09 PM
05-gutenbergr-tidytext-udpipe.Rmd	66 B	Feb 3, 2020, 2:07 PM
06-stylo.Rmd	42 B	Feb 3, 2020, 2:09 PM
07-references.Rmd	54 B	Feb 3, 2020, 1:46 PM
_bookdown.yml	124 B	Feb 3, 2020, 1:56 PM
_bookdown_files		
_output.yml	586 B	Feb 3, 2020, 1:49 PM
bibliography.bib	3.4 KB	Feb 4, 2020, 11:15 AM
book.bib	267 B	Feb 3, 2020, 1:46 PM
docs		
images		
index.Rmd	975 B	Feb 3, 2020, 11:28 PM
naumburg_course.Rproj	225 B	Feb 4, 2020, 11:18 AM

2.3 R as a calculator

Lets first start with the calculator. Press in R console

```
2+9
```

```
## [1] 11
```

```
50*(9-20)
```

```
## [1] -550
```

```
3^3
```

```
## [1] 27
```

```
9^0.5
```

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

```
9+0.5
```

```
## [1] 9.5
```

```
9+.5
```

```
## [1] 9.5
```

```
pi
```

```
## [1] 3.141593
```

Reminder after division

```
10 %% 3
```

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

So you are ready to solve some really hard equations (round it four decimal places):

$$\frac{\pi + 2}{2^{3-\pi}}$$

list of hints

Are you sure that you rounded the result? I expect the answer to be rounded to four decimal places: `0.87654321` becomes `0.8765`.

Are you sure you didn't get into the brackets trap? Even though there isn't any brackets in the mathematical notation, you need to add them in R, otherwise the operation order will be wrong.

2.4 Comments

Any text after a hash # within the same line is considered a comment.

```r
2+2 # it is four
```

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

```r
# you can put any comments here
3+3
```

```
## [1] 6
```

2.5 Functions

The most important part of R is functions: here are some of them:

```r
sqrt(4)
```

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

```r
abs(-5)
```

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

```r
sin(pi/2)
```

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

```r
cos(pi)
```

```
## [1] -1
```

```r
sum(2, 3, 9)
```

```
## [1] 14
```

```r
prod(5, 3, 9)
```

```
## [1] 135
```

```
sin(cos(pi))
```

```
## [1] -0.841471
```

Each function has a name and zero or more arguments. All arguments of the function should be listed in parenthesis and separated by comma:

```
pi
```

```
## [1] 3.141593
```

```
round(pi, 2)
```

```
## [1] 3.14
```

Each function's argument has its own name and serial number. If you use names of the function's arguments, you can put them in any order. If you do not use names of the function's arguments, you should put them according the serial number.

```
round(x = pi, digits = 2)
```

```
## [1] 3.14
```

```
round(digits = 2, x = pi)
```

```
## [1] 3.14
```

```
round(x = pi, d = 2)
```

```
## [1] 3.14
```

```
round(d = 2, x = pi)
```

```
## [1] 3.14
```

```
round(pi, 2)
```

```
## [1] 3.14
```

```
round(2, pi) # this is not the same as all previous!
```

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

There are some functions without any arguments, but you still should use parenthesis:

```
Sys.Date() # correct
```

```
## [1] "2020-02-07"
```

```
Sys.Date # wrong
```

```
## function ()
## as.Date(as.POSIXlt(Sys.time()))
## <bytecode: 0x598a6edc3308>
## <environment: namespace:base>
```

Each function in R is documented. You can read its documentation typing a question mark before the function name:

```
?Sys.Date
```

Explore the function `log()` and calculate the following logarithm:

$$\log_3(3486784401)$$

list of hints

A-a-a! I don't remember anything about logarithms... The logarithm is the inverse function to exponentiation. That means the logarithm of a given number x is the exponent to which another fixed number, the base b, must be raised, to produce that number x.

$$10^n = 1000,\ \text{what is n?}$$

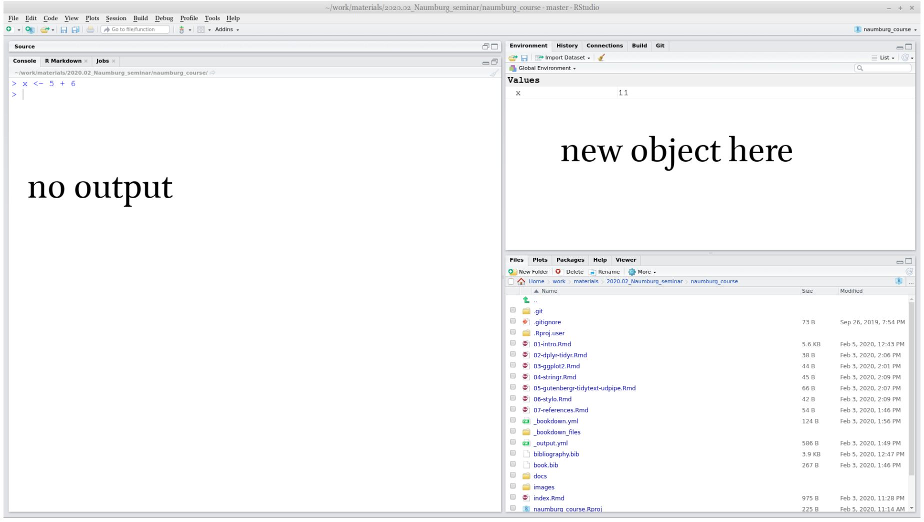
$$n = \log_{10}(1000)$$

What does this small 3 in the task mean? This is the base of the logarithm. So the task is: what is the exponent to which another fixed number, the base *3*, must be raised, to produce that number *3486784401*.

2.6 Variables

Everything in R can be stored in a variable:

```
x <- 5 + 6
```


~/work/materials/2020.02_Naumburg_seminar/naumburg_course - master - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
naumburg_course
Source
Console
R Markdown
Jobs
~/work/materials/2020.02_Naumburg_seminar/naumburg_course/
Environment History Connections Build Git
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home › work › materials › 2020.02_Naumburg_seminar › naumburg_course

```
> x <- 5 + 6
> |
```

no output

```
Values
  x                 11
```

new object here

Name	Size	Modified
..		
.git		
.gitignore	73 B	Sep 26, 2019, 7:54 PM
.Rproj.user		
01-intro.Rmd	5.6 KB	Feb 5, 2020, 12:43 PM
02-dplyr-tidyr.Rmd	38 B	Feb 3, 2020, 2:06 PM
03-ggplot2.Rmd	44 B	Feb 3, 2020, 2:01 PM
04-stringr.Rmd	45 B	Feb 3, 2020, 2:09 PM
05-gutenbergr-tidytext-udpipe.Rmd	66 B	Feb 3, 2020, 2:07 PM
06-stylo.Rmd	42 B	Feb 3, 2020, 2:09 PM
07-references.Rmd	54 B	Feb 3, 2020, 1:46 PM
_bookdown.yml	124 B	Feb 3, 2020, 1:56 PM
_bookdown_files		
_output.yml	586 B	Feb 3, 2020, 1:49 PM
bibliography.bib	3.9 KB	Feb 5, 2020, 12:47 PM
book.bib	267 B	Feb 3, 2020, 1:46 PM
docs		
images		
index.Rmd	975 B	Feb 3, 2020, 11:28 PM
naumburg_course.Rproj	225 B	Feb 5, 2020, 11:14 AM

As a result, no output in the Console, and a new variable x appear in the Environment window. From now on I can use this new variable:

```
x + x
```

```
## [1] 22
```

```
sum(x, x, 7)
```

```
## [1] 29
```

All those operations don't change the variable value. In order to change the variable value you need to make a new assignment:

```
x <- 5 + 6 + 7
```

The fast way for creating `<-` in RStudio is to press `Alt -` on your keyboard.

It is possible to use equal sign `=` for assignment operation, but the recommendations are to use arrow `<-` for the assignment, and equal sign `=` for giving arguments' value inside the functions.

For removing vector you need to use the function `rm()`:

```
rm(x)
x
```

```
## Error in eval(expr, envir, enclos): object 'x' not found
```

2.6.1 Variable comparison

It is possible to compare different variables

```
x <- 18
x > 18
```

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

```
x >= 18
```

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

```
x < 100
```

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

```
x <= 18
```

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

```
x == 18
```

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

```
x != 18
```

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

Operator `!` can work by itself changing logical values into reverse:

```
!TRUE
```

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

```
!FALSE
```

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

2.6.2 Variable types

There are several types of variables in R. In this course the only important types will be `double` (all numbers), `character` (or strings), and `logical`:

```
x <- 2+3
typeof(x)
```

```
## [1] "double"
```

```
y <- "Cześć"
typeof(y)
```

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

```
z <- TRUE
typeof(z)
```

```
## [1] "logical"
```

2.7 Vector

An R object that contains multiple values of the same type is called **vector**. It could be created with the command `c()`:

```
c(3, 0, pi, 23.4, -53)
```

```
## [1]   3.000000   0.000000   3.141593  23.400000 -53.000000
```

```
c("Kraków", "Warszawa", "Cieszyn")
```

```
## [1] "Kraków"   "Warszawa" "Cieszyn"
```

```
c(FALSE, FALSE, TRUE)
```

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

```
a <- c(2, 3, 4)
b <- c(5, 6, 7)
c(a, b)
```

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

For the number sequences there is an easy way:

```
1:10
```

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

```
3:-5
```

```
## [1]  3  2  1  0 -1 -2 -3 -4 -5
```

From now on you can understand that everything we have seen before is a vector of length one. That is why there is `[1]` in all outputs: it is just an index of elements in a vector. Have a look here:

```
1:60
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## [26] 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
## [51] 51 52 53 54 55 56 57 58 59 60
```

```
60:1
```

```
##  [1] 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36
## [26] 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11
## [51] 10  9  8  7  6  5  4  3  2  1
```

There is also a function `sec()` for creation of arithmetic progressions:

```
1:20
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
seq(from = 1, to = 20, by = 1)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
seq(from = 2, to = 100, by = 13)
```

```
## [1]  2 15 28 41 54 67 80 93
```

Use the argument `length.out` of function `seq()` and create an arithmetic sequence from π to 2π of length 50.

There are also some built-in vectors:

```
letters
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"
```

```
LETTERS
```

```
##  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
## [20] "T" "U" "V" "W" "X" "Y" "Z"
```

```
month.name
```

```
##  [1] "January"   "February"  "March"     "April"     "May"       "June"
##  [7] "July"      "August"    "September" "October"   "November"  "December"
```

```
month.abb
```

```
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
```

2.7.1 Vector coercion

Vectors are R objects that contain multiple values of **the same type**. But what if we merged together different types?

```
c(1, "34")
```

```
## [1] "1"  "34"
```

```
c(1, TRUE)
```

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

```
c(TRUE, "34")
```

```
## [1] "TRUE" "34"
```

It is clear that there is a hierarchy: strings > double > logical. It is not universal across different programming languages. It doesn't correspond to the amount of values of particular type:

```
c(1, 2, 3, "34")
```

```
## [1] "1"  "2"  "3"  "34"
```

```
c(1, TRUE, FALSE, FALSE)
```

```
## [1] 1 1 0 0
```

The same story could happen during other operations:

```
5+TRUE
```

```
## [1] 6
```

2.7.2 Vector operations

All operations, that we discussed earlier, could be done with vectors of the same length:

```
1:5 + 6:10
```

```
## [1]  7  9 11 13 15
```

```
1:5 - 6:10
```

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

```
1:5 * 6:10
```

```
## [1]  6 14 24 36 50
```

There are operations where the vector of any length and vector of length one is involved:

```
1:5 + 7
```

```
## [1]  8  9 10 11 12
```

```
1:5 - 7
```

```
## [1] -6 -5 -4 -3 -2
```

```
1:5 / 7
```

```
## [1] 0.1428571 0.2857143 0.4285714 0.5714286 0.7142857
```

There are a lot of functions in R that are **vectorised**. That means that applying this function to a vector is the same as applying this function to each element of the vector:

```
sin(1:5)
```

```
## [1]  0.8414710  0.9092974  0.1411200 -0.7568025 -0.9589243
```

```
sqrt(1:5)
```

```
## [1] 1.000000 1.414214 1.732051 2.000000 2.236068
```

```
abs(-5:3)
```

```
## [1] 5 4 3 2 1 0 1 2 3
```

2.7.3 Indexing vectors

How to get some value or banch of values from a vector? You need to index them:

```
x <- c(3, 0, pi, 23.4, -53)
y <- c("Kraków", "Warszawa", "Cieszyn")

x[4]
```

```
## [1] 23.4
```

```
y[2]
```

```
## [1] "Warszawa"
```

It is possible to have a vector as index:

```
x[1:2]
```

```
## [1] 3 0
```

```
y[c(1, 3)]
```

```
## [1] "Kraków"  "Cieszyn"
```

It is possible to index something that you **do not** want to see in the result:

```
y[-2]
```

```
## [1] "Kraków"  "Cieszyn"
```

```
x[-c(1, 4)]
```

```
## [1]   0.000000   3.141593 -53.000000
```

It is possible to have other variables as an index

```
z <- c(3, 2)
x[z]
```

```
## [1] 3.141593 0.000000
```

```
y[z]
```

```
## [1] "Cieszyn"  "Warszawa"
```

It is possible to index with a logical vector:

```
x[c(TRUE, FALSE, TRUE, TRUE, FALSE)]
```

```
## [1]  3.000000  3.141593 23.400000
```

That means that we could use `TRUE/FALSE`-vector produced by comparison:

```
x[x > 2]
```

```
## [1]  3.000000  3.141593 23.400000
```

It works because `x > 2` is a vector of logical values:

```
x > 2
```

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

It is possible to use `!` operator here changing all `TRUE` values to `FALSE` and vice versa.

```
x[!(x > 2)]
```

```
## [1]   0 -53
```

How many elements in the vector `g` if expression `g[pi < 1000]` does not return an error?

2.7.4 NA

Sometimes there are some missing values in the data, so it is represented with `NA`

```
NA
```

```
## [1] NA
```

```
c(1, NA, 9)
```

```
## [1]  1 NA  9
```

```
c("Kraków", NA, "Cieszyn")
```

```
## [1] "Kraków"  NA        "Cieszyn"
```

```
c(TRUE, FALSE, NA)
```

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

It is possible to check, whether there are missing values or not

```
x <- c("Kraków", NA, "Cieszyn")
y <- c("Kraków", "Warszawa", "Cieszyn")
is.na(x)
```

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

```
is.na(y)
```

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

Some functions doesn't work with vecotors that contain missed values, so you need to add argument `na.rm = TRUE`:

```
x <- c(1, NA, 9, 5)
mean(x)
```

```
## [1] NA
```

```
mean(x, na.rm = TRUE)
```

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

```
min(x, na.rm = TRUE)
```

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

```
max(x, na.rm = TRUE)
```

```
## [1] 9
```

```
median(x, na.rm = TRUE)
```

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

```
range(x, na.rm = TRUE)
```

```
## [1] 1 9
```

2.8 Packages

The most important and useful part of R is hidden in its packages. Everything that we discussed so far is basic R functionality invented back in 1979. Since then a lot of different things changed, so all new practices for data analysis, visualisation and manipulation are packed in packages. During our class we will learn the most popular *"dialect"* of R called `tidyverse`.

In order to install packages you need to use a command. Let's install the `tidyverse` package:

```
install.packages("tidyverse")
```

For today we also will need the `readxl` package:

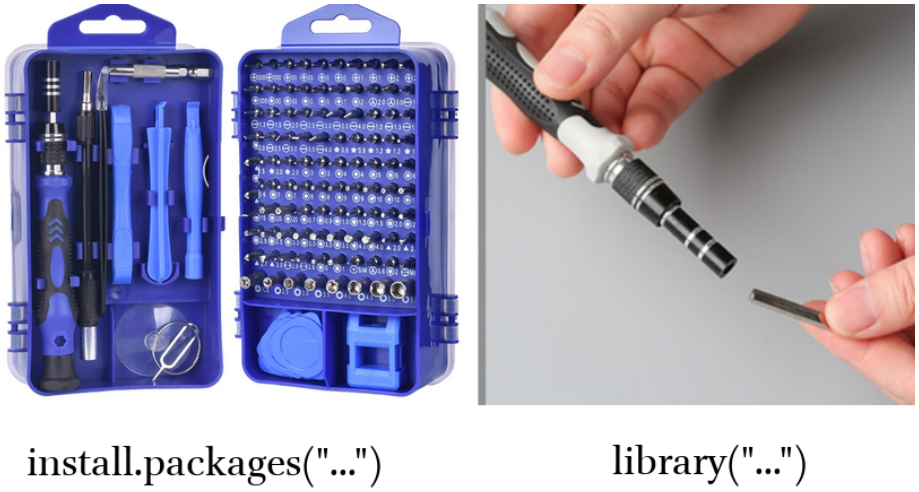
```r
install.packages("tidyverse")
```

After you have downloaded packages nothing will change. You can not use any fucntionality from packages unless you load the package with the `library()` function:

```r
library("tidyverse")
```

Not turninloadingpackage is the most popular mistake of my students. So remember:

- `install.packages("...")` is like you are buying a screwdriver set;
- `library("...")` is like you are stusing art your screwdriver.



For the further lectures we will need `tidyverse` package.

Please install `tidyverse` package and load it.

2.8.1 `tidyverse`

The `tidyverse` is a set of packages:

- `tibble`, for tibbles, a modern re-imagining of data frames — analugue of tables in R
- `readr`, for data import
- `dplyr`, for data manipulation
- `tidyr`, for data tidying (we will discuss it later today)

- ggplot2, for data visualisation
- purrr, for functional programming

2.9 Dataframe (tibble)

A data frame is a collection of variables of the same number of rows with unique row names. Here is an example dataframe with the Tomm Moore filmography:

```
moore_filmography <- tibble(title = c("The Secret of Kells",
                                      "Song of the Sea",
                                      "Kahlil Gibran's The Prophet",
                                      "The Breadwinner",
                                      "Wolfwalkers"),
                            year = c(2009, 2014, 2014, 2017, 2020),
                            director = c(TRUE, TRUE, TRUE, FALSE, TRUE))
moore_filmography
```

```
## # A tibble: 5 x 3
##   title                        year director
##   <chr>                       <dbl> <lgl>
## 1 The Secret of Kells          2009 TRUE
## 2 Song of the Sea              2014 TRUE
## 3 Kahlil Gibran's The Prophet  2014 TRUE
## 4 The Breadwinner              2017 FALSE
## 5 Wolfwalkers                  2020 TRUE
```

There are a lot of built-in dataframes:

```
mtcars
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

```
iris
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1            5.1         3.5          1.4         0.2     setosa
## 2            4.9         3.0          1.4         0.2     setosa
## 3            4.7         3.2          1.3         0.2     setosa
## 4            4.6         3.1          1.5         0.2     setosa
## 5            5.0         3.6          1.4         0.2     setosa
## 6            5.4         3.9          1.7         0.4     setosa
## 7            4.6         3.4          1.4         0.3     setosa
## 8            5.0         3.4          1.5         0.2     setosa
## 9            4.4         2.9          1.4         0.2     setosa
## 10           4.9         3.1          1.5         0.1     setosa
## 11           5.4         3.7          1.5         0.2     setosa
## 12           4.8         3.4          1.6         0.2     setosa
## 13           4.8         3.0          1.4         0.1     setosa
## 14           4.3         3.0          1.1         0.1     setosa
## 15           5.8         4.0          1.2         0.2     setosa
## 16           5.7         4.4          1.5         0.4     setosa
## 17           5.4         3.9          1.3         0.4     setosa
## 18           5.1         3.5          1.4         0.3     setosa
## 19           5.7         3.8          1.7         0.3     setosa
## 20           5.1         3.8          1.5         0.3     setosa
## 21           5.4         3.4          1.7         0.2     setosa
## 22           5.1         3.7          1.5         0.4     setosa
## 23           4.6         3.6          1.0         0.2     setosa
## 24           5.1         3.3          1.7         0.5     setosa
## 25           4.8         3.4          1.9         0.2     setosa
## 26           5.0         3.0          1.6         0.2     setosa
## 27           5.0         3.4          1.6         0.4     setosa
## 28           5.2         3.5          1.5         0.2     setosa
## 29           5.2         3.4          1.4         0.2     setosa
## 30           4.7         3.2          1.6         0.2     setosa
## 31           4.8         3.1          1.6         0.2     setosa
## 32           5.4         3.4          1.5         0.4     setosa
## 33           5.2         4.1          1.5         0.1     setosa
## 34           5.5         4.2          1.4         0.2     setosa
## 35           4.9         3.1          1.5         0.2     setosa
## 36           5.0         3.2          1.2         0.2     setosa
## 37           5.5         3.5          1.3         0.2     setosa
## 38           4.9         3.6          1.4         0.1     setosa
## 39           4.4         3.0          1.3         0.2     setosa
## 40           5.1         3.4          1.5         0.2     setosa
## 41           5.0         3.5          1.3         0.3     setosa
## 42           4.5         2.3          1.3         0.3     setosa
## 43           4.4         3.2          1.3         0.2     setosa
## 44           5.0         3.5          1.6         0.6     setosa
## 45           5.1         3.8          1.9         0.4     setosa
## 46           4.8         3.0          1.4         0.3     setosa
## 47           5.1         3.8          1.6         0.2     setosa
## 48           4.6         3.2          1.4         0.2     setosa
## 49           5.3         3.7          1.5         0.2     setosa
## 50           5.0         3.3          1.4         0.2     setosa
## 51           7.0         3.2          4.7         1.4 versicolor
## 52           6.4         3.2          4.5         1.5 versicolor
## 53           6.9         3.1          4.9         1.5 versicolor
## 54           5.5         2.3          4.0         1.3 versicolor
## 55           6.5         2.8          4.6         1.5 versicolor
## 56           5.7         2.8          4.5         1.3 versicolor
## 57           6.3         3.3          4.7         1.6 versicolor
## 58           4.9         2.4          3.3         1.0 versicolor
## 59           6.6         2.9          4.6         1.3 versicolor
## 60           5.2         2.7          3.9         1.4 versicolor
## 61           5.0         2.0          3.5         1.0 versicolor
## 62           5.9         3.0          4.2         1.5 versicolor
## 63           6.0         2.2          4.0         1.0 versicolor
## 64           6.1         2.9          4.7         1.4 versicolor
## 65           5.6         2.9          3.6         1.3 versicolor
## 66           6.7         3.1          4.4         1.4 versicolor
## 67           5.6         3.0          4.5         1.5 versicolor
## 68           5.8         2.7          4.1         1.0 versicolor
## 69           6.2         2.2          4.5         1.5 versicolor
## 70           5.6         2.5          3.9         1.1 versicolor
## 71           5.9         3.2          4.8         1.8 versicolor
## 72           6.1         2.8          4.0         1.3 versicolor
## 73           6.3         2.5          4.9         1.5 versicolor
## 74           6.1         2.8          4.7         1.2 versicolor
## 75           6.4         2.9          4.3         1.3 versicolor
## 76           6.6         3.0          4.4         1.4 versicolor
## 77           6.8         2.8          4.8         1.4 versicolor
## 78           6.7         3.0          5.0         1.7 versicolor
## 79           6.0         2.9          4.5         1.5 versicolor
## 80           5.7         2.6          3.5         1.0 versicolor
## 81           5.5         2.4          3.8         1.1 versicolor
## 82           5.5         2.4          3.7         1.0 versicolor
## 83           5.8         2.7          3.9         1.2 versicolor
## 84           6.0         2.7          5.1         1.6 versicolor
## 85           5.4         3.0          4.5         1.5 versicolor
## 86           6.0         3.4          4.5         1.6 versicolor
## 87           6.7         3.1          4.7         1.5 versicolor
## 88           6.3         2.3          4.4         1.3 versicolor
## 89           5.6         3.0          4.1         1.3 versicolor
## 90           5.5         2.5          4.0         1.3 versicolor
## 91           5.5         2.6          4.4         1.2 versicolor
## 92           6.1         3.0          4.6         1.4 versicolor
## 93           5.8         2.6          4.0         1.2 versicolor
## 94           5.0         2.3          3.3         1.0 versicolor
## 95           5.6         2.7          4.2         1.3 versicolor
## 96           5.7         3.0          4.2         1.2 versicolor
## 97           5.7         2.9          4.2         1.3 versicolor
## 98           6.2         2.9          4.3         1.3 versicolor
## 99           5.1         2.5          3.0         1.1 versicolor
## 100          5.7         2.8          4.1         1.3 versicolor
## 101          6.3         3.3          6.0         2.5  virginica
## 102          5.8         2.7          5.1         1.9  virginica
## 103          7.1         3.0          5.9         2.1  virginica
## 104          6.3         2.9          5.6         1.8  virginica
## 105          6.5         3.0          5.8         2.2  virginica
## 106          7.6         3.0          6.6         2.1  virginica
## 107          4.9         2.5          4.5         1.7  virginica
## 108          7.3         2.9          6.3         1.8  virginica
## 109          6.7         2.5          5.8         1.8  virginica
## 110          7.2         3.6          6.1         2.5  virginica
## 111          6.5         3.2          5.1         2.0  virginica
## 112          6.4         2.7          5.3         1.9  virginica
## 113          6.8         3.0          5.5         2.1  virginica
## 114          5.7         2.5          5.0         2.0  virginica
## 115          5.8         2.8          5.1         2.4  virginica
## 116          6.4         3.2          5.3         2.3  virginica
## 117          6.5         3.0          5.5         1.8  virginica
## 118          7.7         3.8          6.7         2.2  virginica
## 119          7.7         2.6          6.9         2.3  virginica
## 120          6.0         2.2          5.0         1.5  virginica
## 121          6.9         3.2          5.7         2.3  virginica
## 122          5.6         2.8          4.9         2.0  virginica
## 123          7.7         2.8          6.7         2.0  virginica
## 124          6.3         2.7          4.9         1.8  virginica
## 125          6.7         3.3          5.7         2.1  virginica
## 126          7.2         3.2          6.0         1.8  virginica
## 127          6.2         2.8          4.8         1.8  virginica
## 128          6.1         3.0          4.9         1.8  virginica
## 129          6.4         2.8          5.6         2.1  virginica
## 130          7.2         3.0          5.8         1.6  virginica
## 131          7.4         2.8          6.1         1.9  virginica
## 132          7.9         3.8          6.4         2.0  virginica
## 133          6.4         2.8          5.6         2.2  virginica
## 134          6.3         2.8          5.1         1.5  virginica
## 135          6.1         2.6          5.6         1.4  virginica
## 136          7.7         3.0          6.1         2.3  virginica
## 137          6.3         3.4          5.6         2.4  virginica
## 138          6.4         3.1          5.5         1.8  virginica
## 139          6.0         3.0          4.8         1.8  virginica
## 140          6.9         3.1          5.4         2.1  virginica
## 141          6.7         3.1          5.6         2.4  virginica
## 142          6.9         3.1          5.1         2.3  virginica
## 143          5.8         2.7          5.1         1.9  virginica
## 144          6.8         3.2          5.9         2.3  virginica
## 145          6.7         3.3          5.7         2.5  virginica
## 146          6.7         3.0          5.2         2.3  virginica
## 147          6.3         2.5          5.0         1.9  virginica
## 148          6.5         3.0          5.2         2.0  virginica
## 149          6.2         3.4          5.4         2.3  virginica
## 150          5.9         3.0          5.1         1.8  virginica
```

You can find information about them:

```
?mtcars
?iris
```

Dataframe consists of vectors that could be called using `$` sign:

```
moore_filmography$year
```

```
## [1] 2009 2014 2014 2017 2020
```

```
moore_filmography$title
```

```
## [1] "The Secret of Kells"         "Song of the Sea"
## [3] "Kahlil Gibran's The Prophet" "The Breadwinner"
## [5] "Wolfwalkers"
```

It is possible to add a vector to an existing dataframe:

```
moore_filmography$producer <- c(TRUE, TRUE, FALSE, TRUE, TRUE)
moore_filmography
```

```
## # A tibble: 5 x 4
##   title                        year director producer
##   <chr>                       <dbl> <lgl>    <lgl>
## 1 The Secret of Kells          2009 TRUE     TRUE
## 2 Song of the Sea              2014 TRUE     TRUE
## 3 Kahlil Gibran's The Prophet  2014 TRUE     FALSE
## 4 The Breadwinner              2017 FALSE    TRUE
## 5 Wolfwalkers                  2020 TRUE     TRUE
```

There are some useful functions that tell you somethig about a dataframe:

```
nrow(moore_filmography)
```

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

```
ncol(moore_filmography)
```

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

```
summary(moore_filmography)
```

```
##     title                year       director        producer
##  Length:5           Min.   :2009   Mode :logical   Mode :logical
##  Class :character   1st Qu.:2014   FALSE:1         FALSE:1
##  Mode  :character   Median :2014   TRUE :4         TRUE :4
##                     Mean   :2015
##                     3rd Qu.:2017
##                     Max.   :2020
```

```
str(moore_filmography)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    5 obs. of  4 variables:
##  $ title   : chr  "The Secret of Kells" "Song of the Sea" "Kahlil Gibran's The Prophet" "The E
##  $ year    : num  2009 2014 2014 2017 2020
##  $ director: logi  TRUE TRUE TRUE FALSE TRUE
##  $ producer: logi  TRUE TRUE FALSE TRUE TRUE
```

We will work exclusively with dataframes. But it is not the only data structure in R.

How many rows are in the `iris` dataframe?

How many columns are in the `mtcars` dataframe?

2.9.1 Indexing dataframes

Since dataframes are two-dimensional objects it is possible to index its rows and columns. Rows are the first index, columns are the second index:

```
moore_filmography[3, 2]
```

```
## # A tibble: 1 x 1
##    year
##   <dbl>
## 1  2014
```

```
moore_filmography[3,]
```

```
## # A tibble: 1 x 4
##   title                        year director producer
##   <chr>                       <dbl> <lgl>    <lgl>
## 1 Kahlil Gibran's The Prophet  2014 TRUE     FALSE
```

```
moore_filmography[,2]
```

```
## # A tibble: 5 x 1
##    year
##   <dbl>
## 1  2009
## 2  2014
## 3  2014
## 4  2017
## 5  2020
```

```
moore_filmography[,1:2]
```

```
## # A tibble: 5 x 2
##   title                        year
##   <chr>                       <dbl>
## 1 The Secret of Kells          2009
## 2 Song of the Sea              2014
## 3 Kahlil Gibran's The Prophet  2014
## 4 The Breadwinner              2017
## 5 Wolfwalkers                  2020
```

```
moore_filmography[,-3]
```

```
## # A tibble: 5 x 3
##   title                        year producer
##   <chr>                       <dbl> <lgl>
## 1 The Secret of Kells          2009 TRUE
## 2 Song of the Sea              2014 TRUE
## 3 Kahlil Gibran's The Prophet  2014 FALSE
## 4 The Breadwinner              2017 TRUE
## 5 Wolfwalkers                  2020 TRUE
```

```
moore_filmography[,-c(1:2)]
```

```
## # A tibble: 5 x 2
##   director producer
##   <lgl>    <lgl>
## 1 TRUE     TRUE
## 2 TRUE     TRUE
## 3 TRUE     FALSE
## 4 FALSE    TRUE
## 5 TRUE     TRUE
```

```
moore_filmography[,"year"]
```

```
## # A tibble: 5 x 1
##    year
##   <dbl>
## 1  2009
## 2  2014
## 3  2014
## 4  2017
## 5  2020
```

```
moore_filmography[,c("title", "year")]
```

```
## # A tibble: 5 x 2
##   title                        year
##   <chr>                       <dbl>
## 1 The Secret of Kells          2009
## 2 Song of the Sea              2014
## 3 Kahlil Gibran's The Prophet  2014
## 4 The Breadwinner              2017
## 5 Wolfwalkers                  2020
```

```
moore_filmography[moore_filmography$year > 2014,]
```

```
## # A tibble: 2 x 4
##   title            year director producer
##   <chr>           <dbl> <lgl>    <lgl>
## 1 The Breadwinner  2017 FALSE    TRUE
## 2 Wolfwalkers      2020 TRUE     TRUE
```

2.10 Data import

2.10.1 .csv files

A .csv files (comma-separated values) is a delimited text file that uses a comma (or other delemeters such as tabulation or semicolon) to separate values. It is broadly used bacause it is possible to parse such a file using computers and people can edit it in the Office programs (Microsoft Excel, LibreOffice Calc, Numbers on Mac). Here is our `moore_filmography` dataset in the `.csv` format:

```
title,year,director,producer
The Secret of Kells,2009,TRUE,TRUE
Song of the Sea,2014,TRUE,TRUE
Kahlil Gibran's The Prophet,2014,TRUE,FALSE
The Breadwinner,2017,FALSE,TRUE
Wolfwalkers,2020,TRUE,TRUE
```

Let’s create a variable with this file:

```
our_csv <- "title,year,director,producer
The Secret of Kells,2009,TRUE,TRUE
Song of the Sea,2014,TRUE,TRUE
Kahlil Gibran's The Prophet,2014,TRUE,FALSE
The Breadwinner,2017,FALSE,TRUE
Wolfwalkers,2020,TRUE,TRUE"
```

Now we are ready to use `read_csv()` function:

```
read_csv(our_csv)
```

```
## # A tibble: 5 x 4
##   title                        year director producer
##   <chr>                       <dbl> <lgl>    <lgl>
## 1 The Secret of Kells          2009 TRUE     TRUE
## 2 Song of the Sea              2014 TRUE     TRUE
## 3 Kahlil Gibran's The Prophet  2014 TRUE     FALSE
## 4 The Breadwinner              2017 FALSE    TRUE
## 5 Wolfwalkers                  2020 TRUE     TRUE
```

It is also possible to read files from your computer. Download this file on your computer (press `Ctrl S` or `Cmd S`) and read into R:

```
read_csv("C:/path/to/your/file/moore_filmography.csv")
```

```
## # A tibble: 5 x 4
##   title                        year director producer
##   <chr>                       <dbl> <lgl>    <lgl>
## 1 The Secret of Kells          2009 TRUE     TRUE
## 2 Song of the Sea              2014 TRUE     TRUE
## 3 Kahlil Gibran's The Prophet  2014 TRUE     FALSE
## 4 The Breadwinner              2017 FALSE    TRUE
## 5 Wolfwalkers                  2020 TRUE     TRUE
```

It is also possible to read files from the Internet:

```
read_csv("https://raw.githubusercontent.com/agricolamz/2020.02_Naumburg_R/master/data/
```

```
## Parsed with column specification:
## cols(
##   title = col_character(),
##   year = col_double(),
##   director = col_logical(),
##   producer = col_logical()
## )
```

```
## # A tibble: 5 x 4
##   title                            year director producer
##   <chr>                           <dbl> <lgl>    <lgl>
## 1 The Secret of Kells              2009 TRUE     TRUE
## 2 Song of the Sea                  2014 TRUE     TRUE
## 3 Kahlil Gibran's The Prophet      2014 TRUE     FALSE
## 4 The Breadwinner                  2017 FALSE    TRUE
## 5 Wolfwalkers                      2020 TRUE     TRUE
```

Because of the 2019–20 Wuhan coronavirus outbreak the city of Wuhan is on media everywhere. In Russian for some reason Wuhan is sometimes masculine and sometimes it is feminin. I looked into other Slavic languages and recorded obtained data into the `.csv` file. Download this files to R. What variables does it have?

All file manipulations in R are somehow connected with space on your computer via working directory. You can get information about your current working directory using `getwd()` function. You can change your working directory using `setwd()` function. If a file you want to read is in the working directory you don't need to write the whole path to file:

```
read_csv("moore_filmography.csv")
```

The same simple function will create your `.csv` file:

```
write_csv(moore_filmography, "moore_filmography_v2.csv")
```

Sometimes reading `.csv` files into Microsoft Excel is complicated, please follow the following instructions.

2.10.2 `.xls` and `.xlsx` files

There is a package `readxl` that allows to open and save `.xsl` and `.xslx` files. Install and load the package:

```
library(readxl)
```

Here is a test file. Download it to your computer and put it to your working directory:

```
read_xlsx("moore_filmography.xlsx")
```

```
## # A tibble: 5 x 4
##   title                           year director producer
##   <chr>                          <dbl> <chr>    <chr>
## 1 The Secret of Kells             2009 TRUE     TRUE
## 2 Song of the Sea                 2014 TRUE     TRUE
## 3 Kahlil Gibran's The Prophet     2014 TRUE     FALSE
## 4 The Breadwinner                 2017 FALSE    TRUE
## 5 Wolfwalkers                     2020 TRUE     TRUE
```

`.xls` and `.xlsx` files could have multiple tables on different sheets:

```
read_xlsx("moore_filmography.xlsx", sheet = "iris")
```

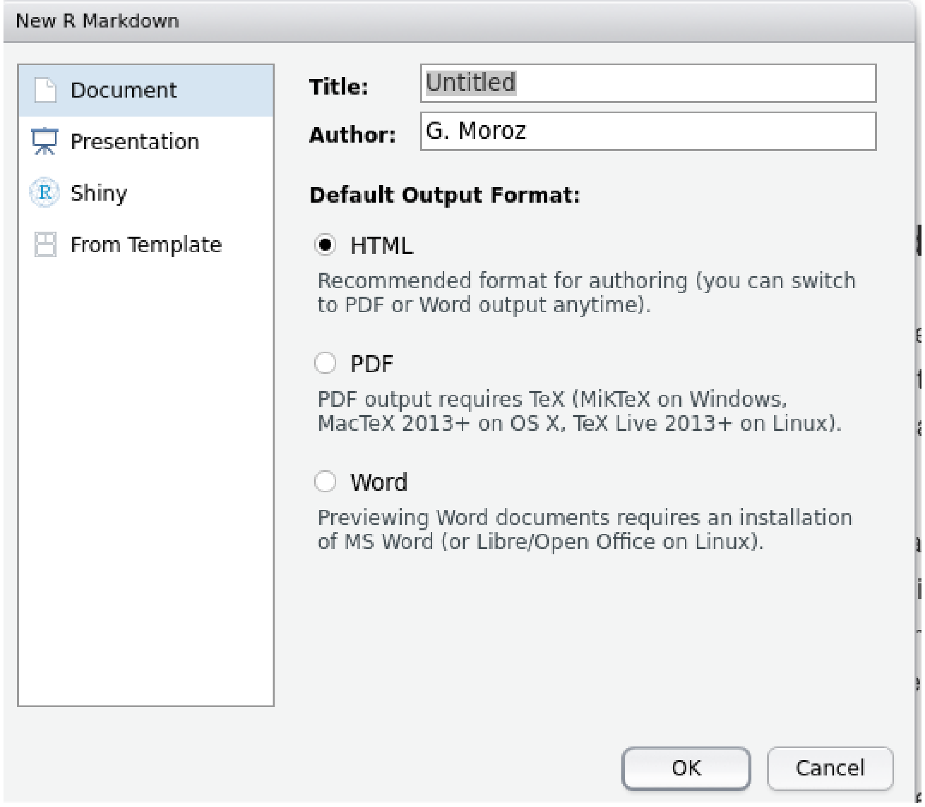
```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <chr>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # … with 140 more rows
```

2.11 Rmarkdown

If you press `Ctrl S` or `Cmd S` then you will save your script. There is also another useful type of coding in R: `rmarkown`. First install this package:

```
install.packages("rmarkdown")
```

Then it will be possible to create a new file: `File > New File > R Markdown...`.



New R Markdown

Title: Untitled

Author: G. Moroz

Default Output Format:

● HTML
Recommended format for authoring (you can switch to PDF or Word output anytime).

○ PDF
PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

○ Word
Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

Press `OK` in the following menu and you will get the template of your R Markdown file. You can modify it, then press Knit and the result file will be created in your working directory. `rmarkdown` package is a really popular and well developed package that creates output into:

- markdown
- html
- docx
- pdf
- beamer presentation
- pptx presentation
- epub
- …
- multiple templates for different scientific journals (package `rticsles` and `papaja`)
- …

Chapter 3

Data manipulation: `dplyr`

First, load the library:

```
library(tidyverse)
```

3.1 Data

In this chapter we will use the following datasets.

3.1.1 Misspelling dataset

This dataset I gathered after some manipulations with data from The Gyllenhaal Experiment By Russell Goldenberg and Matt Daniels for pudding. They analysed mistakes in spellings of celebrities during the search.

```
misspellings <- read_csv("https://raw.githubusercontent.com/agricolamz/DS_for_DH/master/data/miss
```

```
## Parsed with column specification:
## cols(
##   correct = col_character(),
##   spelling = col_character(),
##   count = col_double()
## )
```

```
misspellings
```

```
## # A tibble: 15,477 x 3
##    correct   spelling    count
##    <chr>     <chr>       <dbl>
##  1 deschanel deschanel   18338
##  2 deschanel dechanel     1550
##  3 deschanel deschannel    934
##  4 deschanel deschenel     404
##  5 deschanel deshanel      364
##  6 deschanel dechannel     359
##  7 deschanel deschanelle   316
##  8 deschanel dechanelle    192
##  9 deschanel deschanell    174
## 10 deschanel deschenal     165
## # … with 15,467 more rows
```

There are the following variables in this dataset:

- `correct` — correct spelling
- `spelling` — user's spelling
- `count` — number of cases of user's spelling

3.1.2 `diamonds`

`diamonds` — is the dataset built-in `tidyverse` package.

```
diamonds
```

```
## # A tibble: 53,940 x 10
##    carat cut       color clarity depth table price     x     y     z
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
##  2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
##  3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
##  4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
##  5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
##  6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
##  7 0.24  Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47
##  8 0.26  Very Good H     SI1      61.9    55   337  4.07  4.11  2.53
##  9 0.22  Fair      E     VS2      65.1    61   337  3.87  3.78  2.49
## 10 0.23  Very Good H     VS1      59.4    61   338  4     4.05  2.39
## # … with 53,930 more rows
```

```
?diamonds
```

Chapter 4

Data visualisation: `ggplot2`

Chapter 5

Strings manipulation: `stringr`

Chapter 6

Text manipulation: gutenbergr, tidytext, udpipe

Chapter 7

Stylometric analysis: `stylo`

Bibliography

Baesens, B., Van Vlasselaer, V., and Verbeke, W. (2015). *Fraud analytics using descriptive, predictive, and social network techniques: a guide to data science for fraud detection.* John Wiley & Sons.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98.

Brooks, H. and Cooper, C. L. (2013). *Science for public policy.* Elsevier.

Brzustowicz, M. R. (2017). *Data Science with Java: Practical Methods for Scientists and Engineers.* O'Reilly Media, Inc.

Grus, J. (2019). *Data science from scratch: first principles with python.* O'Reilly Media, Inc.

Hansjörg, N. (2019). *Data Science for Psychologists.* self published.

Janssens, J. (2014). *Data Science at the Command Line: Facing the Future with Time-tested Tools.* O'Reilly Media, Inc.

Provost, F. and Fawcett, T. (2013). *Data Science for Business: What you need to know about data mining and data-analytic thinking.* O'Reilly Media, Inc.

R Core Team (2019). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria.

Thomas, N. and Pallett, L. (2019). *Data Science for Immunologists.* CreateSpace Independent Publishing Platform.

VanderPlas, J. (2016). *Python data science handbook: Essential tools for working with data.* O'Reilly Media, Inc.

Wickham, H. and Grolemund, G. (2016). *R for data science: import, tidy, transform, visualize, and model data.* O'Reilly Media, Inc.