Polish Language(s) and Digital Humanities Using R

G. Moroz

2020

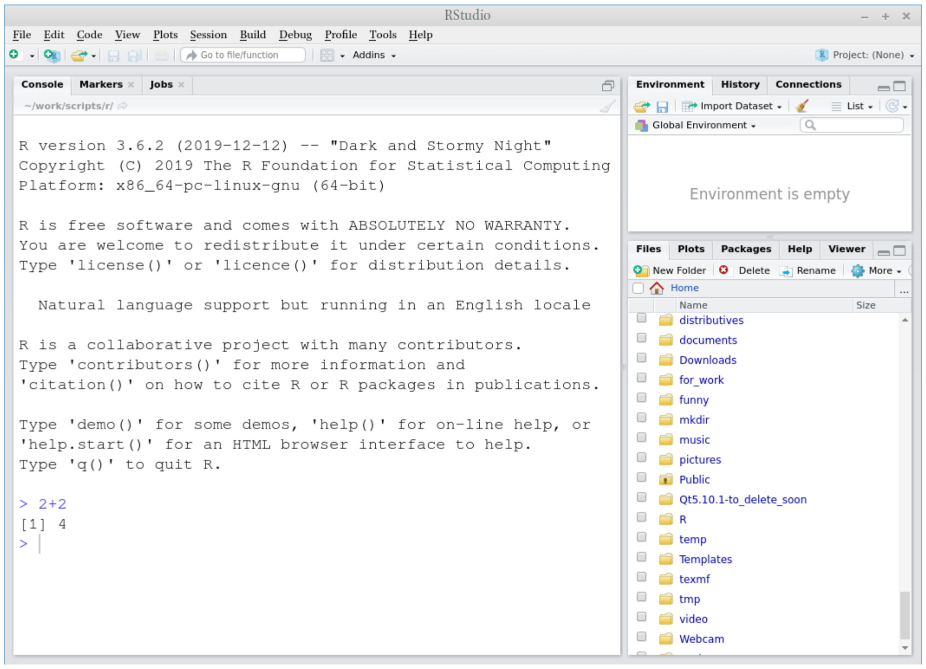
Contents

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
>
```

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 actively developing lately. This field merges computer science, math, 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 scientist need to be able:

- 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 the 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)
- …

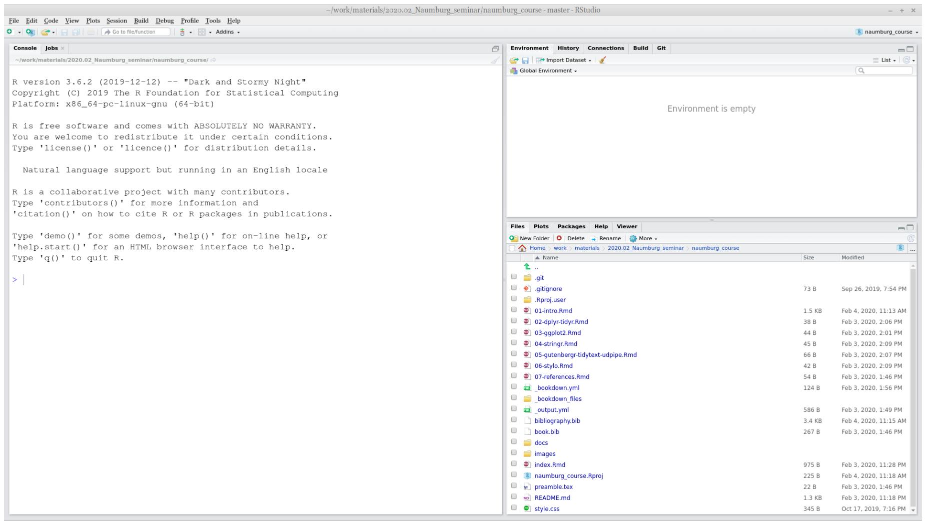
You can find some R answers here:

- 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.

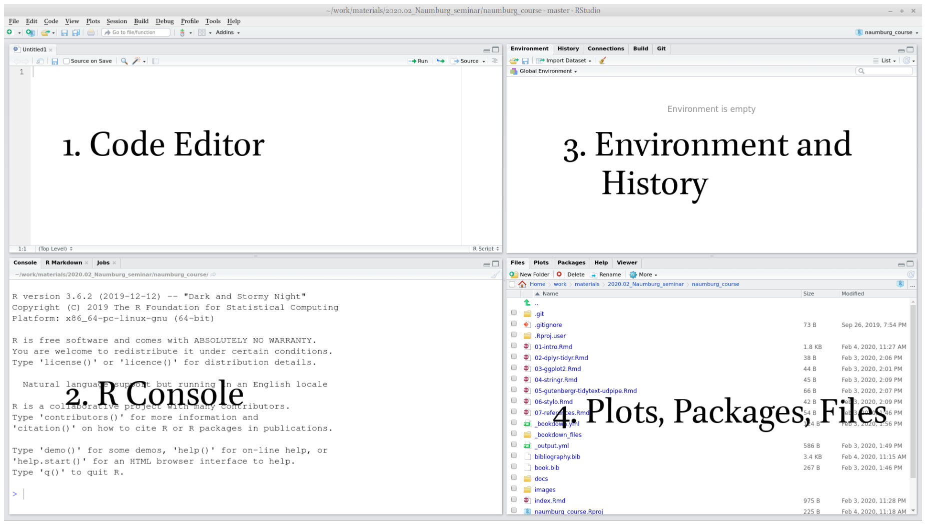
When you open RStudio for the first time you can see something like this:


~/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 the RStudio.



```
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.

>
```

2.3 R as a calculator

Lets first start with a calculator. Press in R console

```r
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 is no any brackets in the mathematical notation, you need to add them in R, otherwise the operation order will be wrong.

2.4 Comments

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

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

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

```
# 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:

```
sqrt(4)
```

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

```
abs(-5)
```

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

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

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

```
cos(pi)
```

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

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

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

```
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
```

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

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

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

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

```
## [1] "2020-02-06"
```

```r
Sys.Date # wrong
```

```
## function ()
## as.Date(as.POSIXlt(Sys.time()))
## <bytecode: 0x5b4841ea8320>
## <environment: namespace:base>
```

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

```r
?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.

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

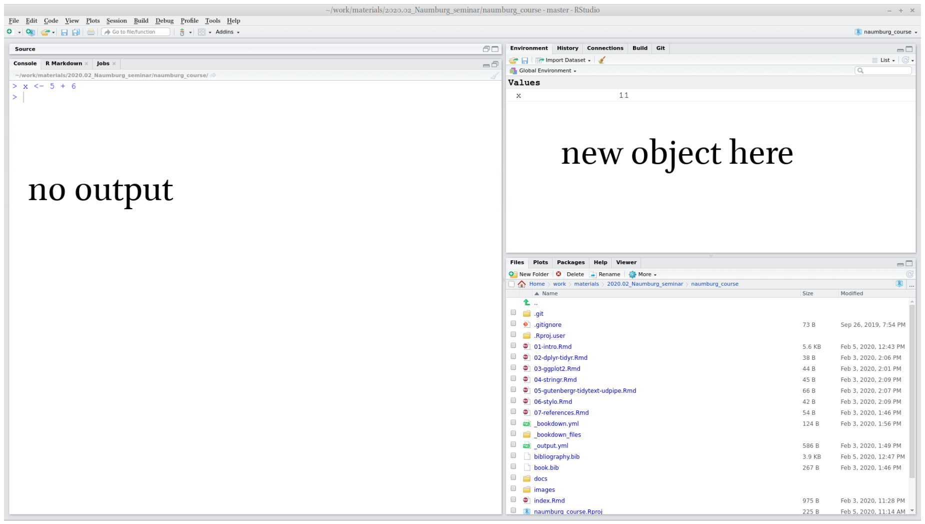
$$n = \log_{10}(1000)$$

What is this small 3 in the task means? 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/
> x <- 5 + 6
>
```

```
Values
  x                 11
```

Home › work › materials › 2020.02_Naumburg_seminar › naumburg_course

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 operation 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 use arrow `<-` for the assignment, and equal sign `=` for giving arguments' value inside the functions.

For removing vector you need to use 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
```

2.6.2 Variable types

There are several types of variable 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

R object that contain 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 you can understand that all 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 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 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 merge 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 hierarchy: strings > double > logical. It is not universal across different programming languages. It doesn't correspond to 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 operation 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 operation 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 apply this function to each ellement 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 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
```

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

2.7.4 NA

Some times there are some missing values in data, so it is represent 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

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

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

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

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

2.8 Packages

The mst important and useful part of R is hidden in its packages. Everything that we discussed so far is base R functionality invented back in 1979. Since then a lot of different things changed, so all new practicies 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 command. Let's install the `tidyverse` package:

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

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

```
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:

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

Not turning on R package is the most popular mistake of my students. So remmember:

- `install.packages("...")` is like you are buying a screwdriver set;
- `library("...")` is like you are start using 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)

2.9.1 Indexing dataframes

2.10 Data import

2.10.1 `.csv` files

Because of 2019–20 Wuhan coronavirus outbreak the city of Wuhan is evrywhere on media. In Russian for some reason Wuhan is masculine and sometimes 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?

2.10.2 `.xls` and `.xlsx` files

2.11 Rmarkdown

Chapter 3

Data manipulation: `dplyr`

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.