Getting started R and R-Studio

Erasmus Q-Intelligence B.V.

Data Science and Business Analytics
Programming

Content

Software requirements

References to Online Book

- Introduction
- Whole Game
- Chapter 2
- Chapter 6

Course documents and software

$\longrightarrow$ Canvas

Make sure to have the following software installed:

- R: `http://CRAN.R-project.org`
- RStudio: `http://RStudio.com`

About R

About R

- Open source environment for statistical computing
- Free as in "free speech" and "free beer"
- Cross platform: Linux/Unix, Mac OS X, Microsoft Windows
- Fully developed and easy-to-use programming language
- Extensible by community contributed packages
  $\longrightarrow$ Roughly 16000 packages on CRAN, Bioconductor and Github
- Support for businesses by Microsoft:
  `https://www.microsoft.com/en-us/cloud-platform/r-server`

$\longrightarrow$ More information on `http://www.R-project.org/`

R books

- Wickham and Grolemund (2017): R for Data Science
  (or read it online for free on `http://r4ds.had.co.nz/`)
- Modern Dive
  (read online on `https://moderndive.com`)
- Kabacoff (2015): R in Action
- Wickham (2009): `ggplot2`: Elegant Graphics for Data Analysis
- Specialized topics:
  - Springer's Use R! series
  - Chapman & Hall/CRC's The R Series

R resources

- Interactive online learning platform DataCamp: `http://www.DataCamp.com`
- Tutorial collection Quick-R: `http://www.statmethods.net`
- Aggregate news and tutorial site R-bloggers: `http://www.R-bloggers.com`
- Scientific articles on R software:
  - Journal of Statistical Software: `http://www.jstatsoft.org`
  - R Journal: `http://journal.R-project.org/`
- R for Matlab users (comparison of R and Matlab commands): `http://mathesaurus.sourceforge.net/octave-r.html`

Design principles

- Interactive data analysis via command line interface (CLI)
  $\longrightarrow$ Enter command and resulting output is printed
- Easy transition from user to developer
  $\longrightarrow$ Integrated development environment (IDE): e.g., RStudio
- Vectorized arithmetic
  $\longrightarrow$ Scalars are vectors of length 1
- Data frames for storing variables of different types
- Formula interface for model specification: `y ~ x1 + x2`

What is R used for?

1. Collecting data
   - Scrape from the web, import from databases, ...
2. Preparing, exploring and cleaning data
   - Data wrangling, exploratory data analysis, plotting
3. Modelling
   - Regression, segmentation, machine learning, custom methods, ...
4. Model evaluation
   - Assessing model quality
5. Reporting results
   - Writing (dynamic) reports, visualization, creating dashboards, ...

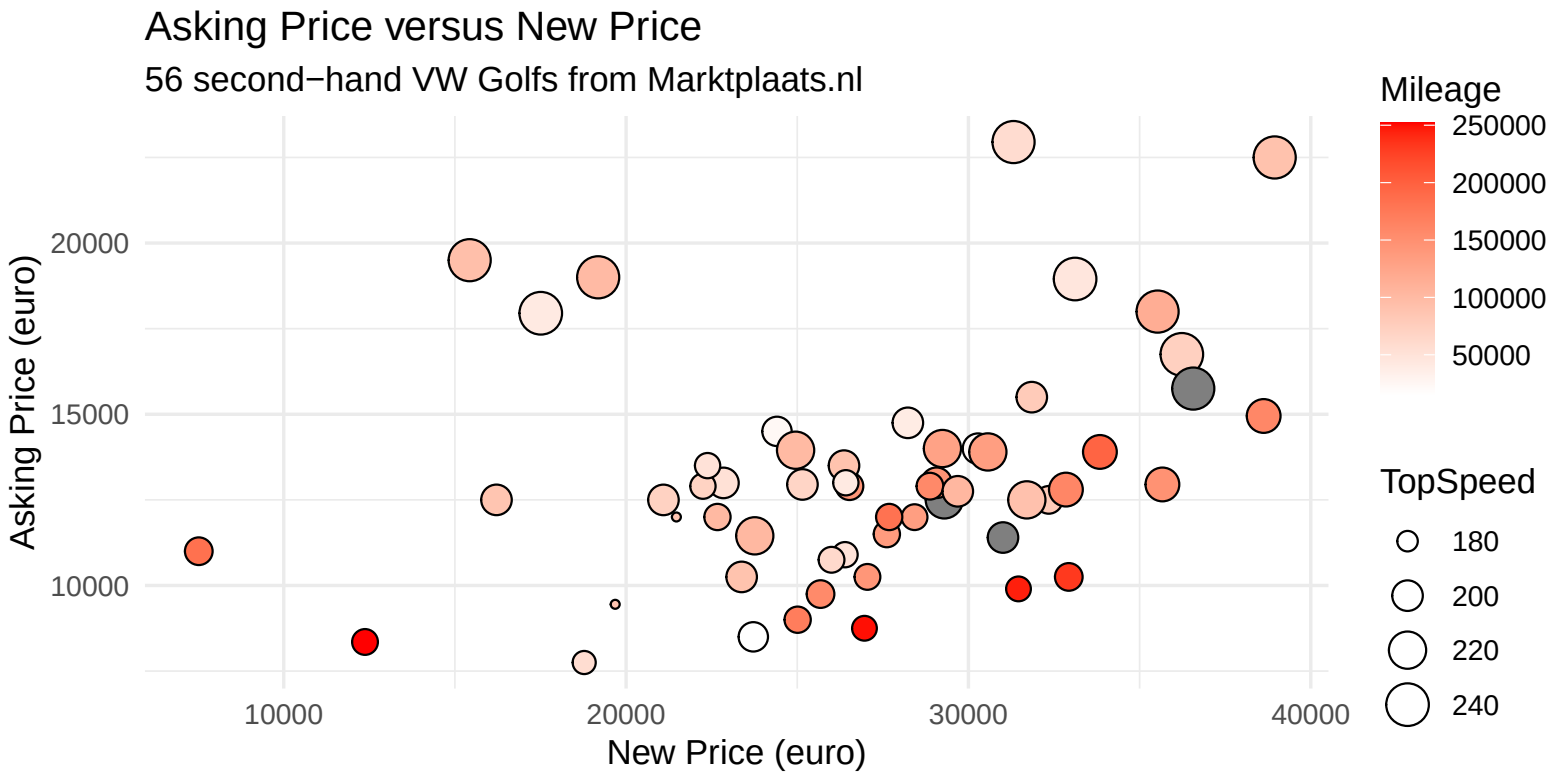
$\longrightarrow$ And various other tasks

Who Uses R?

- Google
- Twitter
- Facebook
- New York Times
- John Deere
- Deloitte
- Credit Suisse
- Novartis
- eBay
- Ford Motor Company
- Kickstarter
- Uber
- Airbnb
- Booking.com
- Bank of America
- McKinsey & Company
- FourSquare

$\longrightarrow$ You too?

Target

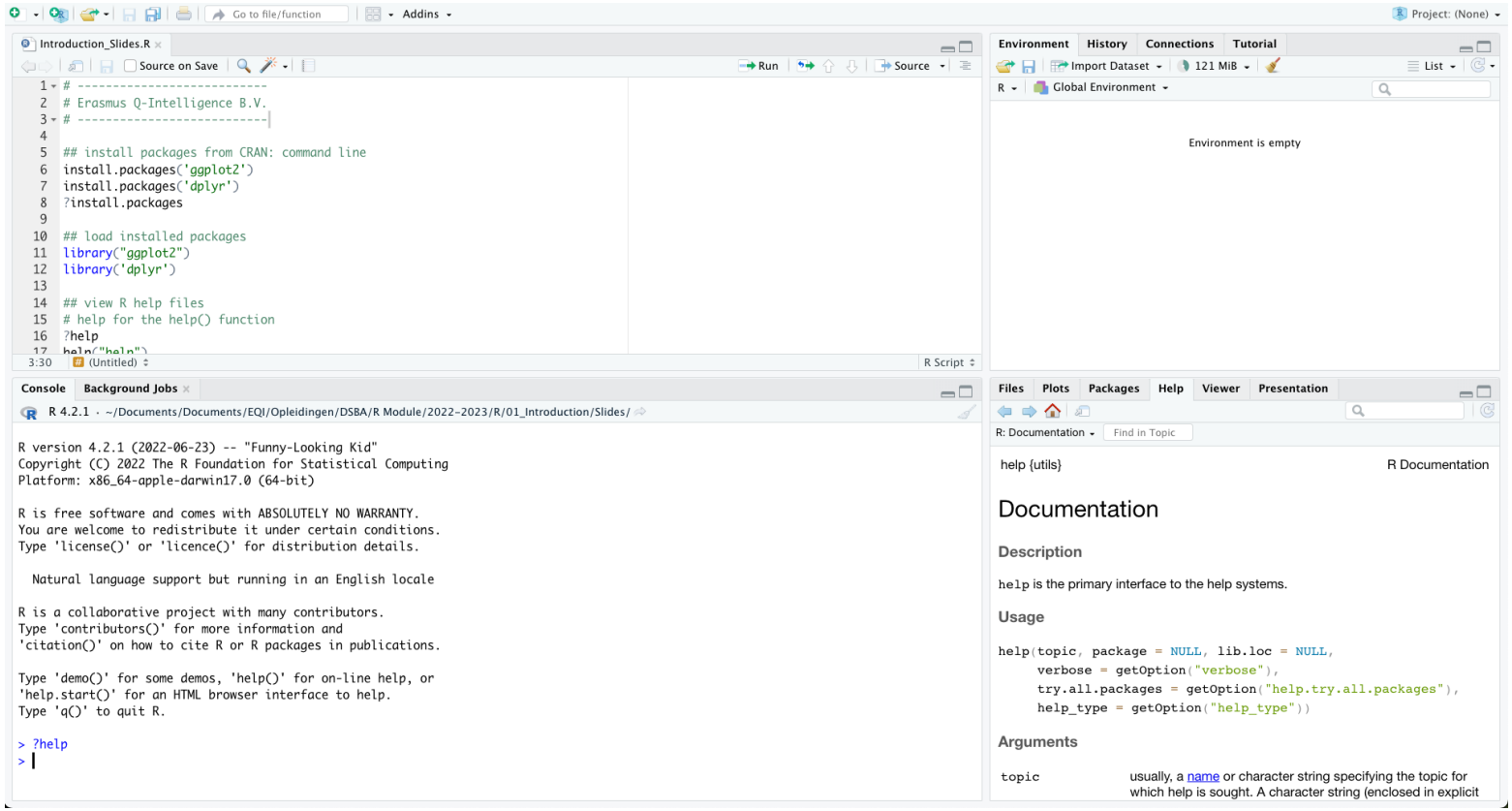


$\longrightarrow$ After first few topics

Getting started with R and RStudio

RStudio

→ Available from `https://posit.co/`



RStudio: four panels

Top left Script editor

Bottom left R console

Top right Two tabs

- Environment: list of objects used in the session
- History: allows to re-run previous commands

Bottom right Five tabs

- Files: browse through files on the computer
- Plots: graphics are displayed here
- Packages: list of installed packages
- Help: R help files are displayed here
- Viewer: local web content created in the session

How do I use R with RStudio?

Type commands into an R script and execute from there:

1. Create objects
   - Typically contain data, or statistics derived from data
   - For example, data loaded from a spreadsheet
   - You have to pick (informative) names
2. Manipulate these objects to create new ones
   - E.g., transform data, fit a model or create a plot
   - All actions are performed by functions
3. Interpret and report your results

Example session 1

Open RStudio, and open `my_first_script.R` in the script editor

$\longrightarrow$ Available on Canvas

$\longrightarrow$ Execute the line in which your cursor is with *ctrl+enter*

$\longrightarrow$ You can also type the command directly in the console, but it is better to store your commands in a script (reproducibility)

Using R

To use R (efficiently) you need to understand:

1. Different types of objects and how they behave
2. Various functions and what they do

$\longrightarrow$ This takes time, but it is worth it!

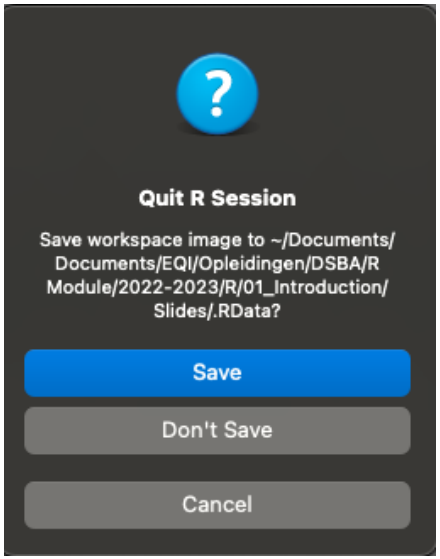
Functions & objects

- All actions are performed by functions:
  - We used `c()`, `mean()` and '<-' (an operator function)
  - Takes an object (or objects), and returns a transformed result
- Functions are pieces of code that perform specific actions
  - For example, `mean()` calculates the mean
  - Function usage is described in its help file (for example, `?mean`)
- Functions accept objects as arguments
  - Data to compute on and other options that control output
  - For example, age is a vector object containing ages

Workspace

Created objects are available in your workspace, also known as the Global Environment

You can save the workspace when you exit R:



$\longrightarrow$ Better not to: store the code that created the objects

Some details

- R is case sensitive
- The + prompt means R is waiting for you to complete the command
- Press Esc to cancel the command being evaluated
- Use the Tab key for code completion
- Remember to close your parentheses ()
- Press the up and down arrows to browse the command history in the console

Script editor

⟶ Idea: perform an analysis and save the commands in a script

⟶ Execute current line or selection on R console with keyboard shortcut *ctrl+enter*

Advantages and disadvantages:

+ Reproducibility
+ Easy to perform the same analysis with different data
  ⟶ Just change the input data
  ⟶ Particularly useful, e.g., for periodic reports
– Steeper learning curve than GUI

Script files

Create new script file: *File* → *New file* → *R Script*

Save script file:

- Keyboard shortcut: *ctrl+S* (Windows/Linux), *cmd+S* (Mac)
- *File* → *Save As...* and enter the file name in the dialog

⟶ Use file extension .R

Open existing script file:

- *File* → *Open File...* and select the script file in the dialog
- Click the *Files* tab in the lower right panel, navigate to the script file and click on it

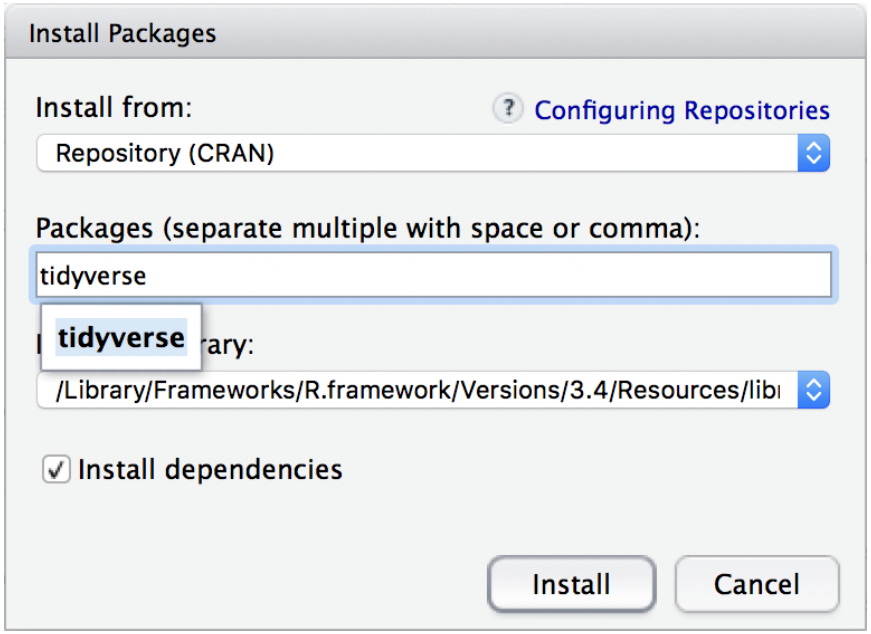
Packages, data and help

Install packages from CRAN

Tools → *Install Packages...*

In the dialog:

- In the *Install from:* box, select *Repository*
- Type the names of the packages into the text box (suggestions are shown as you type)
- Make sure that *Install dependencies* is checked
- Click *Install*



Load installed packages

In RStudio:

1. In the lower right panel, click the *Packages* tab
2. Check the box next to the packages to be loaded

On the command line:

```
R> library("tidyverse")
```

$\longrightarrow$ Install a package once on a computer

$\longrightarrow$ Load it in every new session

R help

- ⟶ R comes with built-in help facility to get more information on functionality
- ⟶ Help topic is usually the name of a function, data set or package

Contributed packages:

- ⟶ Help files are required for packages on CRAN
- ⟶ Package needs to be loaded to view its help files
- ⟶ Overview of all help files within a package is available

View R help files

In RStudio:

1. In the lower right panel, click the *Help* tab
2. Type the topic into the text box on the right (suggestions are shown as you type)

View R help files from the command line

Help for the `help()` function is available:

```
R> ?help
R> help("help")
```

List all help topics within a package:

```
R> help(package = "tidyverse")
```

Run examples from a help file:

```
R> example("mean")
```

Load data sets from packages

From a package that is already loaded:

```
R> data("storms")
R> ?storms
```

From a package that is installed, but not loaded:

```
R> data("storms", package = "dplyr")
```

List all data sets of an installed package:

```
R> data(package = "dplyr")
```

View loaded data set objects

In RStudio:

1. In the top right panel, click the *Environment* tab
2. Click on the data set object name

On the command line:

$\longrightarrow$ Type the name of the data set to print it

$\longrightarrow$ Use the `View()` function to open RStudio's viewer

View data sets: command line

⟶ Example: Atlantic hurricane database track data, 1975-2015

```
R> storms
# A tibble: 10,010 x 13
   name   year month   day  hour   lat  long status
   <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <chr>
 1 Amy    1975     6    27     0  27.5 -79   tropical depression |
 2 Amy    1975     6    27     6  28.5 -79   tropical depression |
 3 Amy    1975     6    27    12  29.5 -79   tropical depression |
------------------------------------ (output removed from slides)
 8 Amy    1975     6    28    18  34   -77   tropical depression |
 9 Amy    1975     6    29     0  34.4 -75.8 tropical storm      |
10 Amy    1975     6    29     6  34   -74.8 tropical storm      |
```

⟶ Too much output even for moderately sized data sets

View first rows of data: `head()`

⟶ Get overview of what the data look like

```
R> head(storms)
# A tibble: 6 x 13
  name   year month   day  hour   lat  long status  category  wind
  <chr> <dbl> <dbl> <int> <dbl> <dbl> <dbl> <fct>      <dbl> <int>
1 Amy    1975     6    27     0  27.5 -79   tropic~       NA    25
2 Amy    1975     6    27     6  28.5 -79   tropic~       NA    25
3 Amy    1975     6    27    12  29.5 -79   tropic~       NA    25
4 Amy    1975     6    27    18  30.5 -79   tropic~       NA    25
5 Amy    1975     6    28     0  31.5 -78.8 tropic~       NA    25
6 Amy    1975     6    28     6  32.4 -78.7 tropic~       NA    25
# i 3 more variables: pressure <int>,
#   tropicalstorm_force_diameter <int>,
#   hurricane_force_diameter <int>
```

View last rows of data: `tail()`

```
R> tail(storms)
# A tibble: 6 x 13
  name    year month   day  hour   lat  long status category  wind
  <chr>  <dbl> <dbl> <int> <dbl> <dbl> <dbl> <fct>     <dbl> <int>
1 Nicole  2022    11    10    18  29   -82.8 tropi~        NA    40
2 Nicole  2022    11    10    19  29.2 -83   tropi~        NA    40
3 Nicole  2022    11    11     0  30.1 -84   tropi~        NA    35
4 Nicole  2022    11    11     6  31.2 -84.6 tropi~        NA    30
5 Nicole  2022    11    11    12  33.2 -84.6 tropi~        NA    25
6 Nicole  2022    11    11    18  35.4 -83.8 other~        NA    25
# i 3 more variables: pressure <int>,
#   tropicalstorm_force_diameter <int>,
#   hurricane_force_diameter <int>
```

Summarize data: `summary()`

$\longrightarrow$ Get overview of the marginal distributions of the variables

```
R> summary(select(storms, name, year, lat, long, category))
     name                year            lat
 Length:19537        Min.   :1975   Min.   : 7.00
 Class :character    1st Qu.:1994   1st Qu.:18.30
 Mode  :character    Median :2004   Median :26.60
                     Mean   :2003   Mean   :27.01
                     3rd Qu.:2013   3rd Qu.:33.80
                     Max.   :2022   Max.   :70.70


      long             category
 Min.   :-136.90   Min.   :1.000
 1st Qu.: -78.80   1st Qu.:1.000
 Median : -62.30   Median :1.000
 Mean   : -61.56   Mean   :1.896
 3rd Qu.: -45.50   3rd Qu.:3.000
 Max.   :  13.50   Max.   :5.000
                   NA's   :14734
```

Loading .Rds files

⟶ RStudio: Click on the data file in the *Files* tab and give a name

⟶ Command line: Use

```
some_name <- readRDS("path/to/file")
```

Working directory

R follows a one directory per project philosophy

$\longrightarrow$ Location where R starts looking for files on the file system

In RStudio:

1. *Session* $\rightarrow$ *Set Working Directory* $\rightarrow$ *Choose Directory...*
2. In the dialog, select the desired working directory

You can automate this by using RStudio projects:

$\longrightarrow$ Opening the RStudio project restores the working directory

Working directory

On the command line:

```
R> setwd("../foo/bar")
```

$\longrightarrow$ Better to use relative path (with respect to working directory) than absolute path (e.g., `C:/Users/andreas/foo/bar`)

$\longrightarrow$ Go back to parent directory with `..`

$\longrightarrow$ Always use / as path separator (instead of \ on Windows)

Exercises

Download and open the *Introduction_Exercises.pdf* file from Canvas, and do Exercises 1.1 and 1.2

Conclusions

Conclusions

- R/RStudio are open source solutions for statistical analysis with a large community of users
- Command line interface has steeper learning curve, but offers greater flexibility

References

R.I. Kabacoff. **R in Action**. Manning, 2nd edition, 2015.

H. Wickham. `ggplot2`: **Elegant Graphics for Data Analysis**. Springer-Verlag, 2009.

H. Wickham and G. Grolemund. **R for Data Science**. O'Reilly, 2017.