

# Getting ready to work with R

Alexandre Courtiol

Leibniz Institute of Zoo and Wildlife Research

June 2019



**Leibniz Institute for Zoo  
and Wildlife Research**

IN THE FORSCHUNGSVERBUND BERLIN E.V.



# Getting ready to work with R

1 Installation

2 The RStudio IDE

3 R markdown

4 Basics of the R language

5 R packages

6 Housekeeping

7 Learning R on your own

# Practice

- ① check that you do get internet access
- ② install R: <https://cran.r-project.org/>
- ③ install the RStudio IDE: <https://www.rstudio.com/products/rstudio/download/>
- ④ open the RStudio IDE

# Getting ready to work with R

1 Installation

2 The RStudio IDE

3 R markdown

4 Basics of the R language

5 R packages

6 Housekeeping

7 Learning R on your own

# Getting ready to work with R

## 1 Installation

## 2 The RStudio IDE

- introduction
- top menu
- the panes

## 3 R markdown

- introduction
- YAML header
- markdown text
- code chunks

## 4 Basics of the R language

- arithmetic
- objects
- functions

## 5 R packages

## 6 Housekeeping

## 7 Learning R on your own

# The RStudio Integrated Desktop Environment

The screenshot shows the RStudio IDE interface. The top menu bar includes RStudio, File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Window, and Help. The title bar indicates the current project is "My\_first\_project" located at "/Desktop/My\_first\_project - RStudio".

The main workspace consists of two panes:

- Code Editor:** Displays an R Markdown script named "Untitled1". The code includes R code chunks and Markdown text. Key parts of the code include:
  - Setting the title, author, date, and output type.
  - An R code chunk with `r setup` and `include=FALSE`.
  - A `knitr::opts\_chunk\$set(echo = TRUE)` command.
  - Markdown text explaining R Markdown.
  - A note about clicking the Knit button to generate a document.
- Console:** Shows the R environment information and the R license notice.

At the bottom, there are tabs for Environment, History, Connections, and Global Environment. The Global Environment tab shows "Environment is empty".

The right side of the interface features a file browser pane with tabs for Files, Plots, Packages, Help, and Viewer. It displays the contents of the "My\_first\_project" directory, which contains a file named "My\_first\_project.Rproj".

# Why using the RStudio IDE?

You can use **R** without RStudio, but RStudio is:

- provided with more functionalities than the official **R** IDE
- integrated with several key **R** packages developed by RStudio (e.g. `rmarkdown`)
- suitable for both desktop and remote computers (web server)
- free and open source

# Overall structure

- Top menu
- Toolbar (small subset from the top menu)
- 4 visible panes

# Getting ready to work with R

## 1 Installation

## 2 The RStudio IDE

- introduction
- top menu
- the panes

## 3 R markdown

- introduction
- YAML header
- markdown text
- code chunks

## 4 Basics of the R language

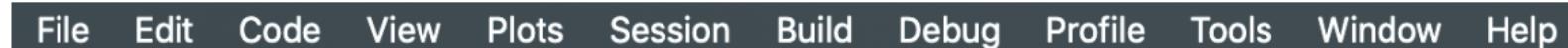
- arithmetic
- objects
- functions

## 5 R packages

## 6 Housekeeping

## 7 Learning R on your own

## The top menu



10 most useful items:

- File → New File
- File → New Project
- File → Import Dataset
- File → Save
- Edit → Undo
- Edit → Clear console
- Code → Comment/Uncomment Lines
- Tools → Global Options...
- Help → Check for Updates (for updating RStudio, not R, nor R packages)
- Help → Cheatsheets

# Practice

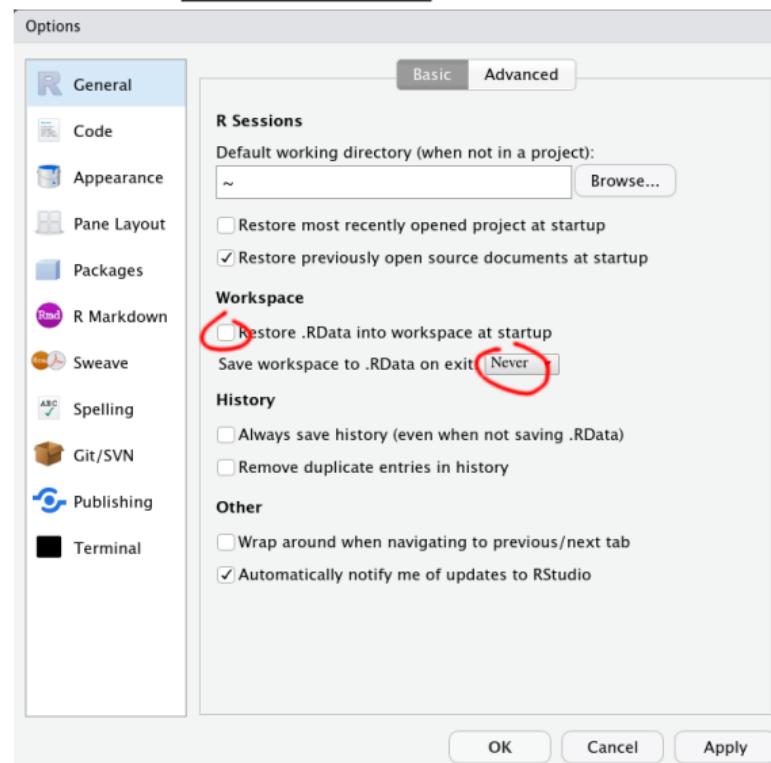
- ① create a new project
- ② create a new **R** markdown file
- ③ save the created **R** markdown file into the project folder directory
- ④ have a look at the RStudio cheatsheet

## Notes:

- a project is defined by a simple (text) file. Its main benefit is to open RStudio with the correct working directory set (that is, the one where the project file is)
- an **R** markdown file is a (text) file where we can write text together with code (more later)

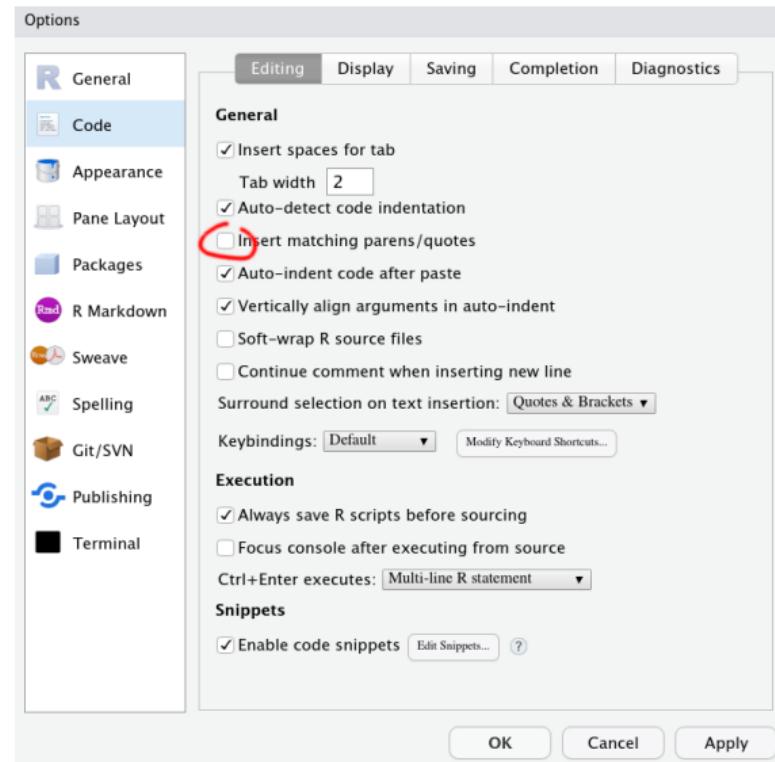
## My tips about the global options

### 1. I never save or restore the workspace and so should you (default settings are a receipe for disasters)



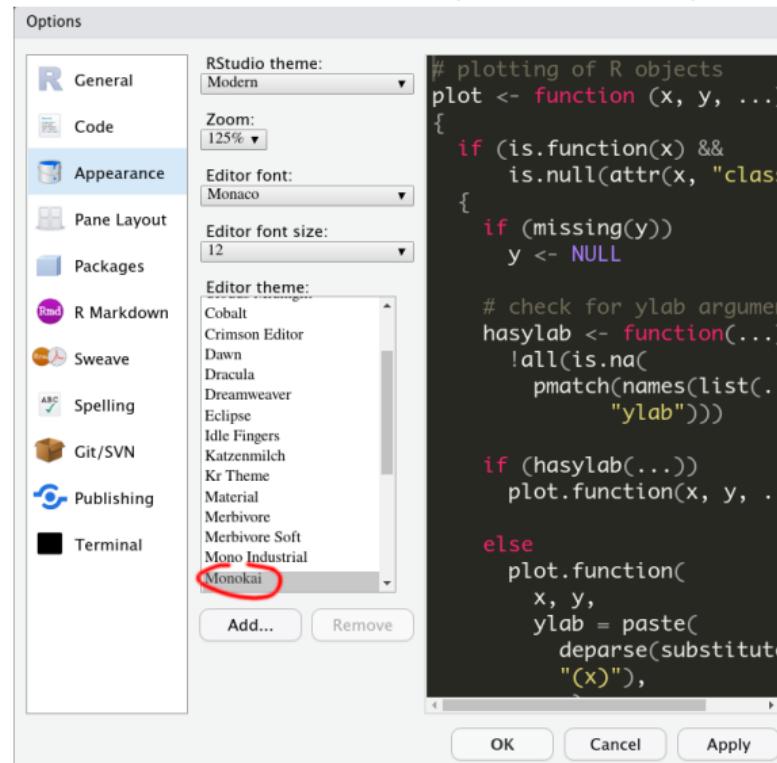
# My tips about the global options

## 2. I don't like parentheses and quotes auto-completion, but you may



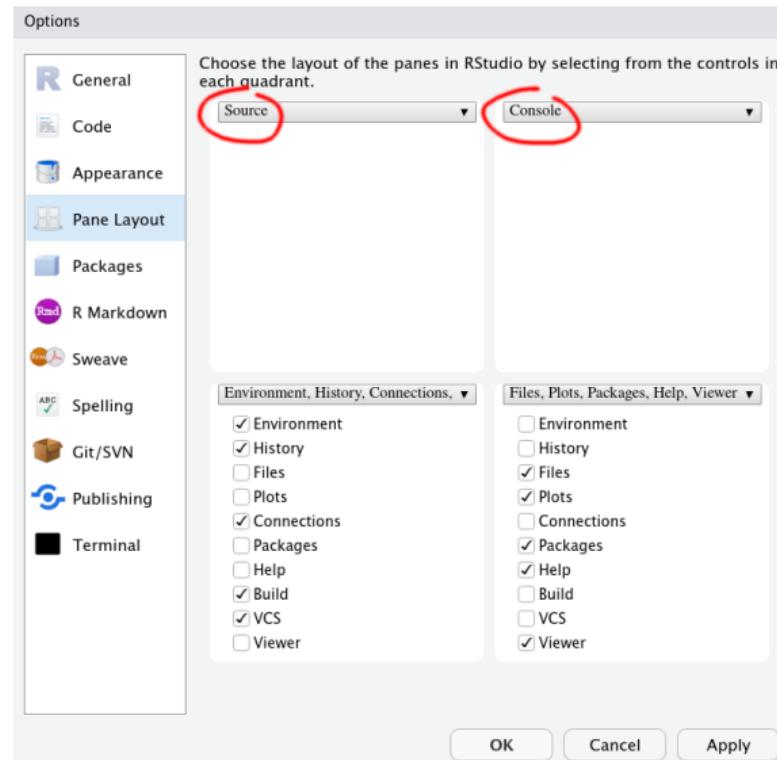
## My tips about the global options

3. I use a black theme except when using a video projector (do as you please)



## My tips about the global options

4. I like to put the source and the console panes side-by-side to give them more vertical space on the screen



# Getting ready to work with R

## 1 Installation

## 2 The RStudio IDE

- introduction
- top menu
- the panes

## 3 R markdown

- introduction
- YAML header
- markdown text
- code chunks

## 4 Basics of the R language

- arithmetic
- objects
- functions

## 5 R packages

## 6 Housekeeping

## 7 Learning R on your own

# The Source pane

This is where you type the code you want to keep

The screenshot shows the RStudio IDE interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Window, Help, and a status bar indicating R version 3.6.0 (2019-04-26) -- "Planting of a Tree". The main window has several panes:

- Source pane (highlighted with a red border):** Contains R code for a Markdown document. The code includes metadata (title, author, date, output type), R Markdown setup, and two code chunks. The second chunk contains explanatory text about R Markdown and a link to the R Markdown website.
- Console pane:** Displays the R startup message, including the R version, copyright information, and a note about the lack of warranty.
- Terminal pane:** Shows the command line interface for R, including help messages for 'demo()', 'help()', and 'help.start()'.
- Environment pane:** Shows the global environment, which is currently empty.
- Files pane:** Shows the project structure under 'My\_first\_project' with a single file 'My\_first\_project.Rproj'.

Notes: tabs give you access to different files (if several files are open)

# The Console pane

This is where you can run R code directly

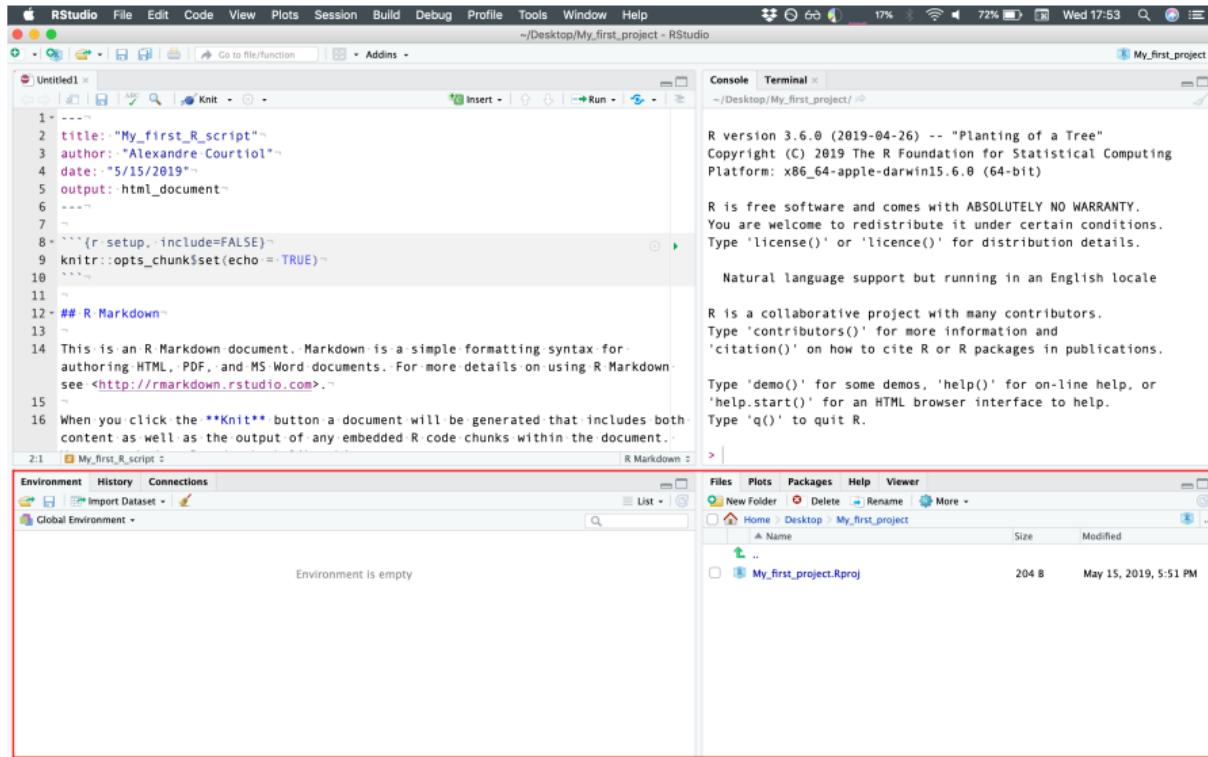
The screenshot shows the RStudio IDE interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Window, Help, and a status bar showing the current project path (~/Desktop/My\_first\_project - RStudio) and system information (17%, 72%, Wed 17:53). The main window has several panes:

- Code Editor:** Displays an R Markdown script named "Untitled1.Rmd". The code includes R code chunks and a Knit button.
- Console:** Displays the R startup message, version information (R 3.6.0), and a welcome message about the R license and natural language support.
- Environment:** Shows the Global Environment tab with an empty environment.
- Files:** Shows the project structure with a file named "My\_first\_project.Rproj".

Notes: tabs give you access to other panes depending on the context. They are all used to run code!

# The other panes

This is where everything else is



Notes: tabs give you access to various panes

# The other panes

You should have at least access to:

- **Environment**: the list of objects known to the console
- **History**: all the code that has been run in the console
- **Connections**: connections to databases
- **Files**: the integrated file manager
- **Plots**: display plots
- **Packages**: tools for installing, loading and updating **R** packages
- **Help**: display help about **R** functions and packages
- **Viewer**: a html/pdf viewer

# The other panes

You should have at least access to:

- Environment: the list of objects known to the console
- History: all the code that has been run in the console
- Connections: connections to databases
- Files: the integrated file manager
- Plots: display plots
- Packages: tools for installing, loading and updating **R** packages
- Help: display help about **R** functions and packages
- Viewer: a html/pdf viewer

But other panes will appear depending on the context

# Getting ready to work with R

1 Installation

2 The RStudio IDE

3 R markdown

4 Basics of the R language

5 R packages

6 Housekeeping

7 Learning R on your own

# Getting ready to work with R

## 1 Installation

## 2 The RStudio IDE

- introduction
- top menu
- the panes

## 3 R markdown

- introduction
- YAML header
- markdown text
- code chunks

## 4 Basics of the R language

- arithmetic
- objects
- functions

## 5 R packages

## 6 Housekeeping

## 7 Learning R on your own

# What is R markdown?

It is a framework to write reproducible reports, which combines . . .

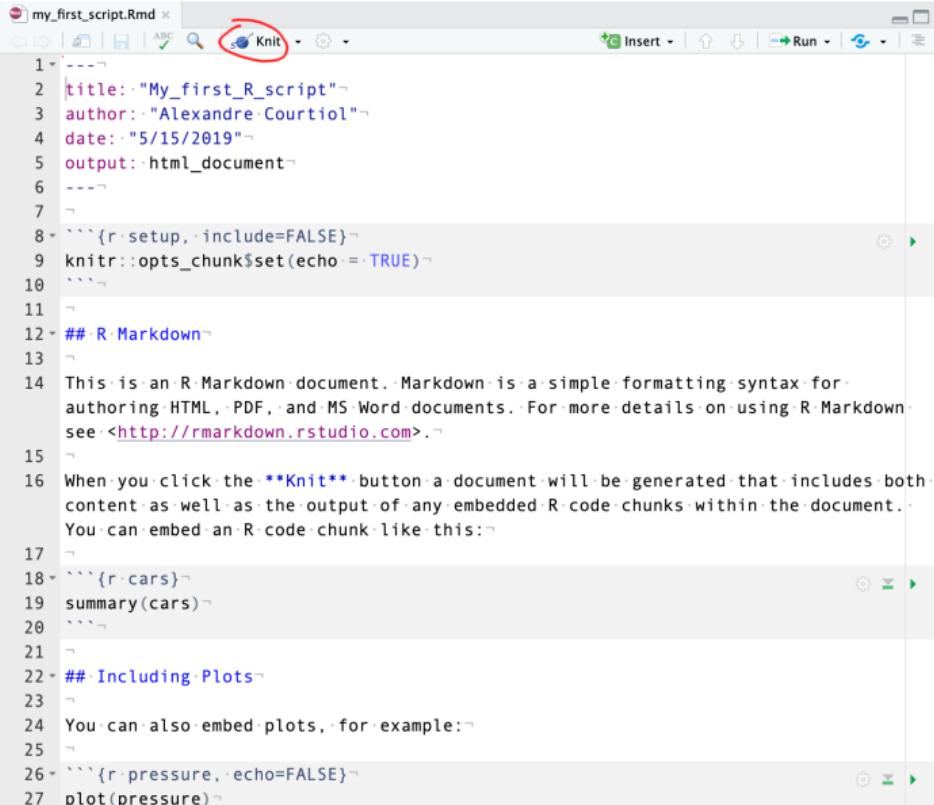
- text
- R code
- R outputs (e.g. plots, tables)

. . . to form a document (e.g. html page, PDF, Microsoft Word, Libre Office, website . . . )

Alternative frameworks are available (e.g. these slides are created using a combination of L<sup>A</sup>T<sub>E</sub>X and R, but R markdown remains simple)

# Practice

Render the R markdown file using knitr and observe the outcome

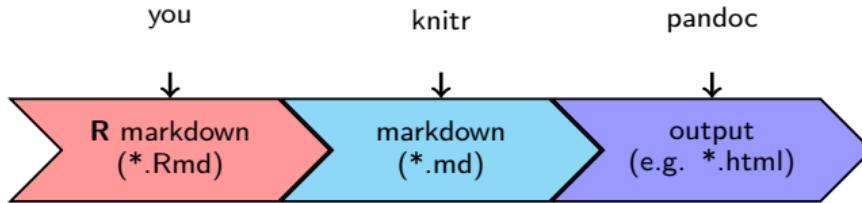


```
my_first_script.Rmd x
[...] Knit [...]
1 ----
2 title: "My_first_R_script"
3 author: "Alexandre.Courtiol"
4 date: "5/15/2019"
5 output: html_document
6 ----
7
8 ```{r, setup, include=FALSE}
9 knitr:::opts_chunk$set(echo = TRUE)
10 ``
11
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting syntax for
authoring HTML, PDF, and MS Word documents. For more details on using R Markdown
see <http://rmarkdown.rstudio.com>.
15
16 When you click the Knit button a document will be generated that includes both
content as well as the output of any embedded R code chunks within the document.
You can embed an R code chunk like this:
17
18 ```{r cars}
19 summary(cars)
20 ``
21
22 ## Including Plots
23
24 You can also embed plots, for example:
25
26 ```{r pressure, echo=FALSE}
27 plot(pressure)
```

# Why using R markdown?

- clear: text, R code, and output are not dispatched in different files
- structured: the document can be divided in sections and the R code into chunks
- transparent: all the R code must be in the R markdown file(s)
- reproducible: enforce that the R code must run (unlike notebooks and pure R script)
- adaptable: you can change one thing and everything else will adjust
- dynamic: you can create so-called dynamic html documents (with buttons, menus...)
- easy to communicate: outputs look quite nice out-of-the-box

# How does it work?



## Notes:

- knitr is an **R** package that turns the **R** code and its outputs into markdown syntax
- pandoc is an external program that turns markdown documents into many possible outputs
- the calls to knitr and pandoc are orchestrated by the **R** package rmarkdown

# General structure of a R markdown file

3 parts:

```

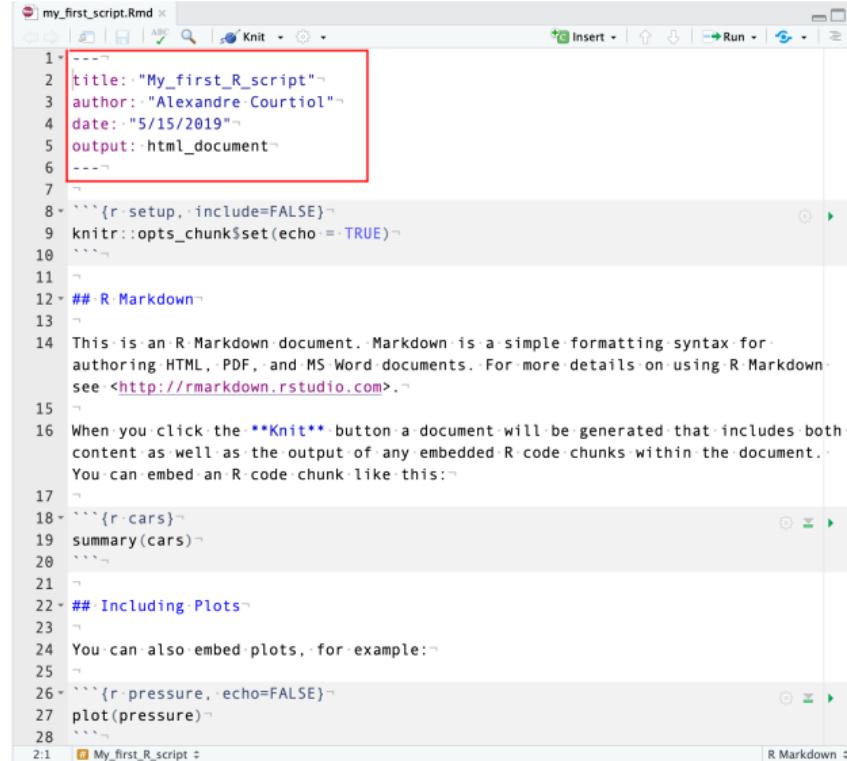
1 ---  
2 title: "My_first_R_script"  
3 author: "Alexandre Courtiol"  
4 date: "5/15/2019"  
5 output: html_document  
6 ---  
7  
8 ```{r setup, include=FALSE}  
9 knitr::opts_chunk$set(echo = TRUE)  
10 ```  
11  
12 ## R Markdown  
13  
14 This is an R Markdown document. Markdown is a simple formatting syntax for  
authoring HTML, PDF, and MS Word documents. For more details on using R Markdown  
see <http://rmarkdown.rstudio.com>.  
15  
16 When you click the Knit button a document will be generated that includes both  
content as well as the output of any embedded R code chunks within the document.  
You can embed an R code chunk like this:  
17  
18 ```{r cars}  
19 summary(cars)  
20 ```  
21  
22 ## Including Plots  
23  
24 You can also embed plots, for example:  
25  
26 ```{r pressure, echo=FALSE}  
27 plot(pressure)  
28 ```

```

# General structure of a R markdown file

3 parts:

- the header (written in YAML)



The screenshot shows the RStudio interface with a code editor window titled "my\_first\_script.Rmd". The code is written in R Markdown. A red box highlights the first five lines, which define the document's metadata in YAML format:

```
1 ---  
2 title: "My_first_R_script"  
3 author: "Alexandre Courtiol"  
4 date: "5/15/2019"  
5 output: html_document  
6 ---
```

Below the header, the document content begins with a section header and a paragraph:

```
7 

## ## R Markdown

  
8 This is an R Markdown document. Markdown is a simple formatting syntax for  
authoring HTML, PDF, and MS Word documents. For more details on using R Markdown  
see <http://rmarkdown.rstudio.com>.
```

Further down, there is an R code chunk followed by a section header:

```
16 When you click the Knit button a document will be generated that includes both  
content as well as the output of any embedded R code chunks within the document.  
You can embed an R code chunk like this:  
17  
18 

```
{r cars}  
19 summary(cars)  
20 }
```


```
22 

## ## Including Plots

  
23  
24 You can also embed plots, for example:  
25  
26 

```
{r pressure, echo=FALSE}  
27 plot(pressure)
```


```


```

# General structure of a R markdown file

3 parts:

- the header (written in YAML)
- the text (written in markdown)

```

1 ---  

2 title: "My_first_R_script"  

3 author: "Alexandre Courtiol"  

4 date: "5/15/2019"  

5 output: html_document  

6 ---  

7  

8 ```{r setup, include=FALSE}  

9 knitr::opts_chunk$set(echo = TRUE)  

10 ````  

11 ## R Markdown  

12  

13 This is an R Markdown document. Markdown is a simple formatting syntax for  

authoring HTML, PDF, and MS Word documents. For more details on using R Markdown  

see <http://rmarkdown.rstudio.com>.  

15  

16 When you click the **Knit** button a document will be generated that includes both  

content as well as the output of any embedded R code chunks within the document.  

You can embed an R code chunk like this:  

17  

18 ```{r cars}  

19 summary(cars)  

20 ````  

21  

22 ## Including Plots  

23  

24 You can also embed plots, for example:  

25  

26 ```{r pressure, echo=FALSE}  

27 plot(pressure)  

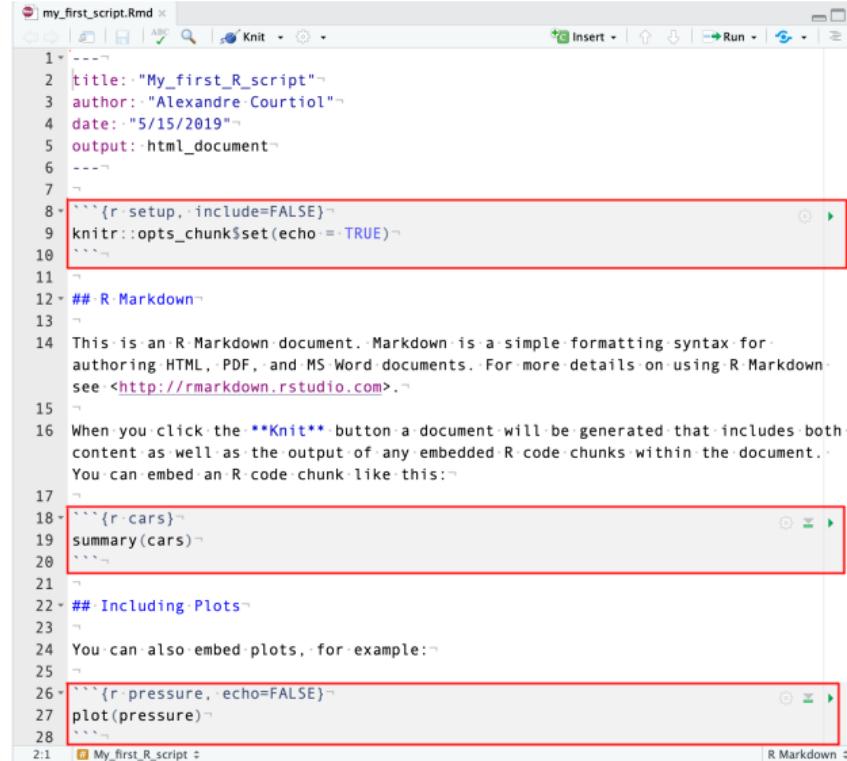
28 ````  

2:1 My_first_R_script
  
```

# General structure of a R markdown file

3 parts:

- the header (written in YAML)
- the text (written in markdown)
- the code chunks (written in R or alternatives)



```

1 ---  

2 title: "My_first_R_script"  

3 author: "Alexandre Courtiol"  

4 date: "5/15/2019"  

5 output: html_document  

6 ---  

7  

8 ```{r setup, include=FALSE}  

9 knitr::opts_chunk$set(echo = TRUE)  

10 ```  

11  

12 ## R Markdown  

13  

14 This is an R Markdown document. Markdown is a simple formatting syntax for  

authoring HTML, PDF, and MS Word documents. For more details on using R Markdown  

see <http://rmarkdown.rstudio.com>.  

15  

16 When you click the Knit button a document will be generated that includes both  

content as well as the output of any embedded R code chunks within the document.  

You can embed an R code chunk like this:  

17  

18 ```{r cars}  

19 summary(cars)  

20 ```  

21  

22 ## Including Plots  

23  

24 You can also embed plots, for example:  

25  

26 ```{r pressure, echo=FALSE}  

27 plot(pressure)  

28 ```

```

# Getting ready to work with R

## 1 Installation

## 2 The RStudio IDE

- introduction
- top menu
- the panes

## 3 R markdown

- introduction
- YAML header
- markdown text
- code chunks

## 4 Basics of the R language

- arithmetic
- objects
- functions

## 5 R packages

## 6 Housekeeping

## 7 Learning R on your own

## The YAML header

The header sets the configuration for pandoc, so it is used to influence the format and style of the output file  
(It is also be used to set some options for rmarkdown)

A simple YAML header:

```
---
```

```
title: "My_first_R_script"
author: "Alexandre Courtiol"
date: "5/15/2019"
output: html_document
---
```

## The YAML header

The header sets the configuration for pandoc, so it is used to influence the format and style of the output file  
(It is also be used to set some options for rmarkdown)

A simple YAML header:

```
---
```

```
title: "My_first_R_script"
author: "Alexandre Courtiol"
date: "5/15/2019"
output: html_document
---
```

A slightly more complex YAML header:

```
---
```

```
title: "My_first_R_script"
author: "Alexandre Courtiol"
date: "5/15/2019"
output:
  html_document:
    toc: true
    toc_float: true
    theme: journal
    keep_md: yes
---
```

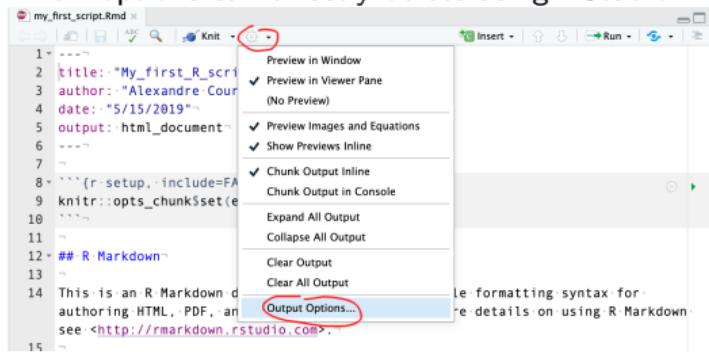
Notes:

- indentation matters in YAML!!! (beware while copy & pasting with RStudio autoindentation)
- not all settings work for all types of documents
- YAML = YAML Ain't Markup Language

# Practice

Modify the YAML options in the header and observe how the output changes accordingly

A few options can directly be set using RStudio:



For info on what you can change, check:

- The **R Markdown Cheat Sheet**
- <https://rmarkdown.rstudio.com>
- <https://bookdown.org/yihui/rmarkdown>

# Getting ready to work with R

## 1 Installation

## 2 The RStudio IDE

- introduction
- top menu
- the panes

## 3 R markdown

- introduction
- YAML header
- markdown text
- code chunks

## 4 Basics of the R language

- arithmetic
- objects
- functions

## 5 R packages

## 6 Housekeeping

## 7 Learning R on your own

# Markdown

Markdown is a very simple plain text formatting syntax that can be learned in minutes  
Check Help/Markdown Quick Reference for (almost) all there is to know!

The screenshot shows the RStudio interface with the 'Viewer' tab selected. The title bar includes 'Files', 'Plots', 'Packages', 'Help', and 'Viewer'. Below the title bar are standard browser-like navigation buttons (back, forward, home, search) and a 'Find in Topic' input field. The main content area displays the 'Markdown Quick Reference' page. It features a heading 'Markdown Quick Reference' and a paragraph stating 'R Markdown is an easy-to-write plain text format for creating dynamic documents and reports. See [Using R Markdown](#) to learn more.' Under the heading 'Emphasis', there is a code block showing examples for italic and bold text: `*italic* **bold**  
_italic_ __bold__`. Under the heading 'Headers', there is another code block showing examples for different header levels: `# Header 1  
## Header 2  
### Header 3`.

You can directly embed raw html code for more functionalities

For example, this is useful for commenting text that you don't want to appear in the output:

This text will show in the output <!-- but this one will not -->

# Practice

- try to experiment a bit with the syntax
- edit the markdown text of your **R** markdown file, so to prepare the file to take notes about the course (but don't trash the chunk of **R** code yet)

# Getting ready to work with R

## 1 Installation

## 2 The RStudio IDE

- introduction
- top menu
- the panes

## 3 R markdown

- introduction
- YAML header
- markdown text
- code chuncks

## 4 Basics of the R language

- arithmetic
- objects
- functions

## 5 R packages

## 6 Housekeeping

## 7 Learning R on your own

## Anatomy of a code chunk

```
```{language_name chunk name, option1=argument1, option2=argument2, ...}
some code ## a comment
```
```

Examples:

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

```{r cars}
summary(cars)
```

```{r pressure, echo=FALSE}
plot(pressure)
```
```

# Anatomy of a code chunk

```
```{language_name chunk name, option1=argument1, option2=argument2, ...}
some code ## a comment
```

```

Examples:

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
```
```{r cars}
summary(cars)
```
```
```{r pressure, echo=FALSE}
plot(pressure)
```

```

The screenshot shows a code editor window in RStudio. The code in the editor is:

```
25
26  ``{r pressure, echo=
27  p My_first_R_script
28  ↴
29  ↪ Chunk 1: setup
30  ↪ R Markdown
31  ↪ N Chunk 2: cars
32  ↪ g Including Plots
33  ↪ ↪
34  ↪ ↪ Chunk 3: pressure
35
36  # My_first_R_script
```

A context menu is open at line 27, with "Chunk 1: setup" highlighted. The menu also lists "R Markdown", "Chunk 2: cars", and "Including Plots". The status bar at the bottom shows "5:22" and "# My\_first\_R\_script".

# Examples of code chunk options

Some options good to know:

| Option | what for?        | default |
|--------|------------------|---------|
| echo   | display the code | TRUE    |
| error  | stop on error    | TRUE    |
| eval   | run the chunk    | TRUE    |

# Examples of code chunk options

Some options good to know:

| Option | what for?        | default |
|--------|------------------|---------|
| echo   | display the code | TRUE    |
| error  | stop on error    | TRUE    |
| eval   | run the chunk    | TRUE    |

You can set some of them directly using RStudio:

The screenshot shows a code editor window with R code and a 'Chunk options' dialog box from RStudio.

Code in the editor:

```
18 ~~~{r·cars}~  
19 summary(cars)~  
20 ~~~~  
21 ~  
22 ## Including Plots~  
23 ~  
24 You can also embed plots, for example:~  
25 ~  
26 ~~~{r·pressure,·echo=FALSE}~  
27 plot(pressure)~  
28 ~~~~
```

Chunk options dialog (highlighted with a red circle):

- Name: cars
- Output: (Use document default)
- Show warnings
- Show messages
- Use paged tables
- Use custom figure size
- Chunk options
- Revert
- Apply

# Examples of code chunk options

Some options good to know:

| Option | what for?        | default |
|--------|------------------|---------|
| echo   | display the code | TRUE    |
| error  | stop on error    | TRUE    |
| eval   | run the chunk    | TRUE    |

You can set some of them directly using RStudio:

The screenshot shows a code editor with R code and a 'Chunk options' dialog box. The code editor contains:

```
18 ~~~{r cars}~  
19 summary(cars)~  
20 ~~~~  
21 ~  
22 ## Including Plots~  
23 ~  
24 You can also embed plots, for example:~  
25 ~  
26 ~~~{r pressure, echo=FALSE}~  
27 plot(pressure)~  
28 ~~~~
```

The 'Chunk options' dialog box is open, with the 'Name' field set to 'cars'. A red circle highlights the 'Output' dropdown menu, which is currently set to '(Use document default)'. Other options shown include:

- Show warnings
- Show messages
- Use paged tables
- Use custom figure size

Buttons at the bottom of the dialog are 'Revert' and 'Apply'.

For more options, check:

- The **R Markdown Cheat Sheet** (for a few more)
- <https://yihui.name/knitr/options> (for all of them)

## Examples of code chunk options

Some options good to know:

| Option | what for?        | default |
|--------|------------------|---------|
| echo   | display the code | TRUE    |
| error  | stop on error    | TRUE    |
| eval   | run the chunk    | TRUE    |

You can set some of them directly using RStudio:



The screenshot shows a code editor with R code and a 'Chunk options' dialog box. The code includes a summary of the 'cars' dataset and a note about including plots. The 'Chunk options' dialog box is open, with the 'Name' field set to 'cars' and the 'Output' dropdown set to '(Use document default)'. Other options like 'Show warnings', 'Show messages', 'Use paged tables', and 'Use custom figure size' are listed but not selected. A red circle highlights the 'Output' dropdown.

```
18 ~~~{r cars}~  
19 summary(cars)~  
20 ~~~~  
21 ~  
22 ## Including Plots~  
23 ~  
24 You can also embed plots, for example:~  
25 ~  
26 ~~~{r pressure, echo=FALSE}~  
27 plot(pressure)~  
28 ~~~~
```

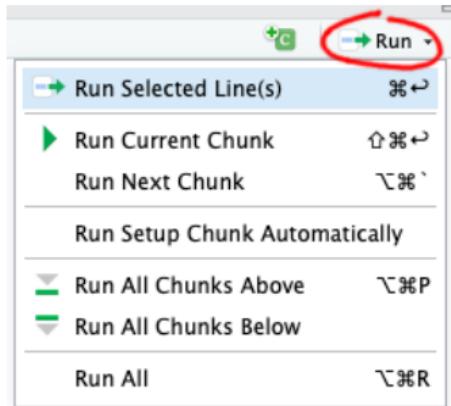
For more options, check:

- The **R Markdown Cheat Sheet** (for a few more)
- <https://yihui.name/knitr/options> (for all of them)

Note: if you want the option to be set in all chunks, put it in `knitr:::opts_chunk$set(...)` in the first chunk.

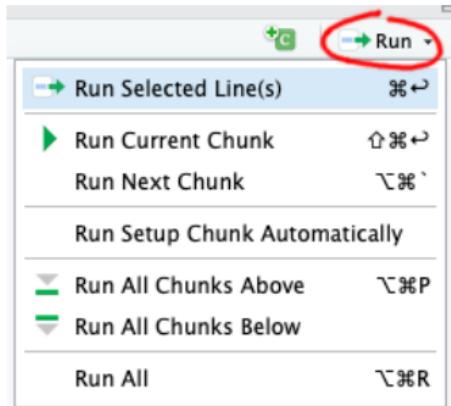
# Running a code chunk without knitting

You can run a chunk using the mouse or the keyboard:

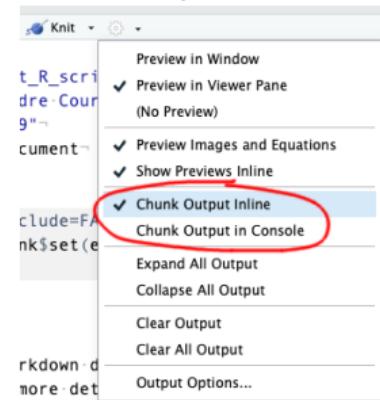


## Running a code chunk without knitting

You can run a chunk using the mouse or the keyboard:



All **R** chunks are run in the **Console** pane (unless when knitting), but you can decide if you want to copy the output within the **Source** pane too!



# Practice

Experiment a little by knitting existing chunks using different options!

# Getting ready to work with R

1 Installation

2 The RStudio IDE

3 R markdown

4 Basics of the R language

5 R packages

6 Housekeeping

7 Learning R on your own

# Getting ready to work with R

## 1 Installation

## 2 The RStudio IDE

- introduction
- top menu
- the panes

## 3 R markdown

- introduction
- YAML header
- markdown text
- code chunks

## 4 Basics of the R language

- arithmetic
- objects
- functions

## 5 R packages

## 6 Housekeeping

## 7 Learning R on your own

# Arithmetic

R can perform basic arithmetic:

```
1 + 1
## [1] 2
1 - 1
## [1] 0
2 * pi
## [1] 6.283185
3 / 2
## [1] 1.5
10 %% 3
## [1] 1
5^2
## [1] 25
5^2 + 1
## [1] 26
5^(2 + 1)
## [1] 125
```

Conclusion: you may never need a hand calculator anymore!

# Practice

Delete all existing chunks and create a new chunk with some arithmetic

Note: to create a new chunk, you can use `Code → Insert chunk` (or just type)

# Tips

- ❶ only use the **Console** pane to mess around
- ❷ write proper **R** code inside a chunk in your Rmd document
- ❸ knit chunks one by one before knitting the whole Rmd document
- ❹ name objects with useful names

# Getting ready to work with R

## 1 Installation

## 2 The RStudio IDE

- introduction
- top menu
- the panes

## 3 R markdown

- introduction
- YAML header
- markdown text
- code chunks

## 4 Basics of the R language

- arithmetic
- **objects**
- functions

## 5 R packages

## 6 Housekeeping

## 7 Learning R on your own

# Creating objects

Objects are being assigned using the “arrow” operator:

```
one.plus.one <- 1 + 1 ## storing the result
```

## Creating objects

Objects are being assigned using the “arrow” operator:

```
one.plus.one <- 1 + 1 ## storing the result
```

Objects are being used through their name (that is the whole point):

```
one.plus.one ## displaying the result
## [1] 2
one.plus.one.plus.one <- one.plus.one + 1
one.plus.one.plus.one
## [1] 3
```

# Creating objects

Objects are being assigned using the “arrow” operator:

```
one.plus.one <- 1 + 1 ## storing the result
```

Objects are being used through their name (that is the whole point):

```
one.plus.one ## displaying the result
## [1] 2
one.plus.one.plus.one <- one.plus.one + 1
one.plus.one.plus.one
## [1] 3
```

Tip:

```
(one.times.two <- 1 * 2) ## storing and displaying the result at once
## [1] 2
```

# Creating objects

Objects are being assigned using the “arrow” operator:

```
one.plus.one <- 1 + 1 ## storing the result
```

Objects are being used through their name (that is the whole point):

```
one.plus.one ## displaying the result
## [1] 2
one.plus.one.plus.one <- one.plus.one + 1
one.plus.one.plus.one
## [1] 3
```

Tip:

```
(one.times.two <- 1 * 2) ## storing and displaying the result at once
## [1] 2
```

Note 1: avoid spaces & weird characters in object names to avoid troubles (but “\_” and “.” are OK).

Note 2: names are case sensitive.

## Common mistakes

The huge majority of beginner's problems are typos:

```
one.plus.one  
## [1] 2
```

```
one.plus.One  
## Error in eval(expr, envir, enclos): object 'one.plus.One' not found
```

## Common mistakes

The huge majority of beginner's problems are typos:

```
one.plus.one  
## [1] 2
```

```
one.plus.One  
## Error in eval(expr, envir, enclos): object 'one.plus.One' not found
```

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

## Common mistakes

The huge majority of beginner's problems are typos:

```
one.plus.one  
## [1] 2
```

```
one.plus.One  
## Error in eval(expr, envir, enclos): object 'one.plus.One' not found
```

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

```
1 +  
one.plus.one <- 1 + 1  
## Error in 1 + one.plus.one <- 1 + 1: target of assignment expands to non-language object
```

# Getting ready to work with R

## 1 Installation

## 2 The RStudio IDE

- introduction
- top menu
- the panes

## 3 R markdown

- introduction
- YAML header
- markdown text
- code chunks

## 4 Basics of the R language

- arithmetic
- objects
- **functions**

## 5 R packages

## 6 Housekeeping

## 7 Learning R on your own

# Functions

```
citation() ## function showing how to cite R

##
## To cite R in publications use:
##
##   R Core Team (2019). R: A language and environment for statistical computing. R Foundation for Statistical
##   Computing, Vienna, Austria. URL https://www.R-project.org/.
##
## A BibTeX entry for LaTeX users is
##
## @Manual{,
##   title = {R: A Language and Environment for Statistical Computing},
##   author = {{R Core Team}},
##   organization = {R Foundation for Statistical Computing},
##   address = {Vienna, Austria},
##   year = {2019},
##   url = {https://www.R-project.org/},
## }
##
## We have invested a lot of time and effort in creating R, please cite it when using it for data analysis.
## See also 'citation("pkgname")' for citing R packages.
```

# Functions

```
citation() ## function showing how to cite R

##
## To cite R in publications use:
##
##   R Core Team (2019). R: A language and environment for statistical computing. R Foundation for Statistical
##   Computing, Vienna, Austria. URL https://www.R-project.org/.
##
## A BibTeX entry for LaTeX users is
##
## @Manual{,
##   title = {R: A Language and Environment for Statistical Computing},
##   author = {{R Core Team}},
##   organization = {R Foundation for Statistical Computing},
##   address = {Vienna, Austria},
##   year = {2019},
##   url = {https://www.R-project.org/},
## }
##
## We have invested a lot of time and effort in creating R, please cite it when using it for data analysis.
## See also 'citation("pkgname")' for citing R packages.
```

```
help(citation) ## getting help for this function
```

```
?citation() ## same but shorter (syntactic sugar)
```

# Functions

```
citation() ## function showing how to cite R

##
## To cite R in publications use:
##
##   R Core Team (2019). R: A language and environment for statistical computing. R Foundation for Statistical
##   Computing, Vienna, Austria. URL https://www.R-project.org/.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {R: A Language and Environment for Statistical Computing},
##     author = {{R Core Team}},
##     organization = {R Foundation for Statistical Computing},
##     address = {Vienna, Austria},
##     year = {2019},
##     url = {https://www.R-project.org/},
##   }
##
## We have invested a lot of time and effort in creating R, please cite it when using it for data analysis.
## See also 'citation("pkgname")' for citing R packages.
```

```
help(citation) ## getting help for this function
```

```
?citation() ## same but shorter (syntactic sugar)
```

Note: always look at the help before using a function new to you!

# Functions

```
mean()
```

```
?mean()
```

Usage:

```
mean(x, ...)  
## Default S3 method:  
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments:

x: An R object. Currently there are methods for numeric/logical vectors and date, date-time and time interval objects, and for data frames all of whose columns have a method. Complex vectors are allowed for 'trim = 0', only.

trim: the fraction (0 to 0.5) of observations to be trimmed from each end of 'x' before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.

na.rm: a logical value indicating whether 'NA' values should be stripped before the computation proceeds.

[...]

# Syntax for functions

## Basic syntax:

```
sign(x = -5)
## [1] -1
sign(-5) ## dangerous: try to avoid!
## [1] -1
sign(y = -5)
## Error in sign(y = -5): supplied argument name 'y' does not match 'x'
```

# Syntax for functions

Calling a function without its parentheses reveals its definition:

```
sign
## function (x) .Primitive("sign")
```

# Syntax for functions

Calling a function without its parentheses reveals its definition:

```
sign
## function (x) .Primitive("sign")
```

All functions need parentheses to work and exceptions correspond to short-cuts (called “syntactic sugar”):

```
1 + 1
## [1] 2
`+` (1, 1)
## [1] 2
a <- 1
a
## [1] 1
`<-` (a, 1)
a
## [1] 1
```

# Finding functions

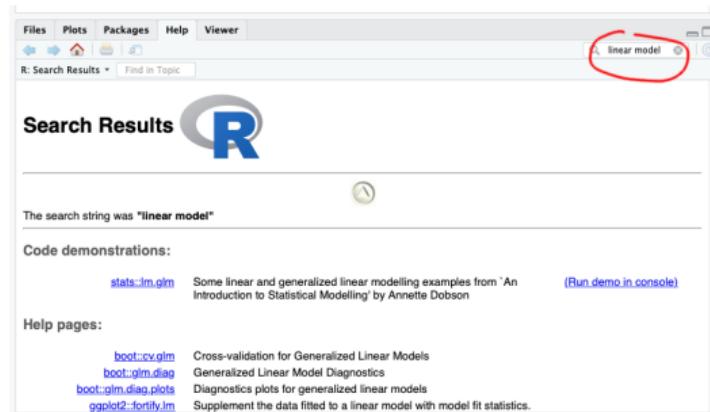
To find the name of the function you are looking for, you may try:

```
??"linear model"
```

or

```
help.search(pattern = "linear model", package = "stats") ## if you know where to look for
```

or



# Getting ready to work with R

1 Installation

2 The RStudio IDE

3 R markdown

4 Basics of the R language

5 R packages

6 Housekeeping

7 Learning R on your own

# The concept of an **R** package

Packages extend **R** functionalities:

- for most users; e.g. `ggplot2`
- for specific users; e.g. `IsoriX`
- for developers; e.g. `Rcpp`

# The concept of an R package

Packages extend R functionalities:

- for most users; e.g. ggplot2
- for specific users; e.g. IsoreX
- for developers; e.g. Rcpp

Key facts about packages:

- a package is just a folder (often compressed) containing R functions, data & documentation
- a library is the installed version of the package (also a folder)
- there are tons of packages out there:
  - 14244 packages are available on [cran.r-project.org](http://cran.r-project.org)
  - ~ 1500 packages aimed at bioinformatics on [bioconductor.org](http://bioconductor.org)
  - many more on [github.com](http://github.com)
  - many more shared between users in other ways

# The concept of an R package

Packages extend R functionalities:

- for most users; e.g. ggplot2
- for specific users; e.g. IsoreX
- for developers; e.g. Rcpp

Key facts about packages:

- a package is just a folder (often compressed) containing R functions, data & documentation
- a library is the installed version of the package (also a folder)
- there are tons of packages out there:
  - 14244 packages are available on [cran.r-project.org](http://cran.r-project.org)
  - ~ 1500 packages aimed at bioinformatics on [bioconductor.org](http://bioconductor.org)
  - many more on [github.com](http://github.com)
  - many more shared between users in other ways

Note: packages can be used to create research compendia!

# Installing a package

Simple situation: the package is available as a binary file prepared for your system on CRAN

You can use RStudio:

The screenshot shows the RStudio interface for managing packages. On the left, a modal window titled 'Install Packages' is open. It has the following fields:

- 'Install from:' dropdown set to 'Repository (CRAN)'. A tooltip 'Configuring Repositories' is visible.
- 'Packages (separate multiple with space or comma):' input field containing 'drat', which is circled in red.
- 'Install to Library:' dropdown set to '/Library/Frameworks/R.framework/Versions/3.6/Resources/library'.
- A checked checkbox 'Install dependencies'.
- 'Install' and 'Cancel' buttons at the bottom.

On the right, the main RStudio window shows the following:

- A code editor with the following text:

```
help.start() for an HTML brows  
Type 'q()' to quit R.  
> |
```
- A navigation bar with tabs: Files, Plots, **Packages**, Help, Viewer. The 'Packages' tab is circled in red.
- An 'Install' button under the navigation bar, also circled in red.
- A table titled 'System Library' listing packages:

| Name       | Description                         |
|------------|-------------------------------------|
| adRes      | Assessing Whether Climate are Adapt |
| askpass    | Safe Password Ent                   |
| assertthat | Easy Pre and Post                   |
| backports  | Reimplementation                    |
| base64enc  | Tools for base64 e                  |

# Installing a package

Simple situation: the package is available as a binary file prepared for your system on CRAN

You can use RStudio:

The screenshot shows the RStudio interface. On the left, a modal dialog titled 'Install Packages' is open. It has the following fields:

- Install from:** Repository (CRAN)
- Packages (separate multiple with space or comma):** drat (with a red circle around it)
- Install to Library:** /Library/Frameworks/R.framework/Versions/3.6/Resources/library (with a red circle around it)
- Install dependencies:**

At the bottom are 'Install' and 'Cancel' buttons.

On the right, the RStudio sidebar is visible, featuring tabs for Files, Plots, Packages (which is highlighted with a red circle), Help, and Viewer. Below the tabs, there's a section for 'System Library' listing several packages with their descriptions. The 'Install' button in the sidebar is also circled in red.

| Name       | Description                         |
|------------|-------------------------------------|
| adRes      | Assessing Whether Climate are Adapt |
| askpass    | Safe Password Ent                   |
| assertthat | Easy Pre and Post                   |
| backports  | Reimplementation                    |
| base64enc  | Tools for base64 e                  |

or you can type in the Console pane:

```
install.packages("drat") ## install drat
```

## Installing a package

In general, the installation procedure depends on:

- where the package is being hosted (local, CRAN, bioconductor, GitHub, other)
- if the package contains sources in another language that have been compiled or not

## Installing a package

In general, the installation procedure depends on:

- where the package is being hosted (local, CRAN, bioconductor, GitHub, other)
- if the package contains sources in another language that have been compiled or not

In order to be able to install packages that require compilation (and thus have access to more or newer version of packages), you need to install:

- Rtools if you use Windows (<https://cran.r-project.org/bin/windows/Rtools>)
- clang and gfortran (<https://cran.r-project.org/bin/macosx/tools>)  
or the large Xcode (<https://developer.apple.com/xcode>) if you use macOS
- nothing if you use Linux or other Unix-based system

## Installing the package for this course

The package is not on CRAN as I want to be able to update it instantaneously and have potentially large files.

I host the package here: <https://github.com/courtiol/BeginR>

You should install it using drat as follows:

```
install.packages("drat")      ## install drat from CRAN; only run once per R lifetime
library(drat)                  ## load the package drat
addRepo("courtiol")            ## use drat to declare my GitHub account
install.packages("BeginR")     ## install the package
```

## Installing the package for this course

The package is not on CRAN as I want to be able to update it instantaneously and have potentially large files.

I host the package here: <https://github.com/courtio1/BeginR>

You should install it using drat as follows:

```
install.packages("drat")      ## install drat from CRAN; only run once per R lifetime
library(drat)                  ## load the package drat
addRepo("courtio1")            ## use drat to declare my GitHub account
install.packages("BeginR")     ## install the package
```

Note: every morning of the course you may have to rerun the last 3 lines of code to get the lastest version of this course.

# Practice

- ① create a separate **R** markdown file that you will knit every morning to update BeginR
- ② knit the **R** markdown file to install the package for the first time

# Loading a package

Loading a package makes new functions and data available to the user:

Example:

```
library(BeginR)
##
##  The package for the course 'Getting Started with R'
##  by @alexcourtio (version 0.0.0.9000), is now loaded!
##  To access the slides, just type browseVignettes(package = 'BeginR'),
##  [or get_vignettes() if you also need to see the sources of the vignettes].
##
##  All sources for this package are available at https://github.com/courtio/BeginR
##  where you can find more information on how to use this package
##  and where you can also leave comments (under 'Issues').
```

# Loading a package

Loading a package makes new functions and data available to the user:

Example:

```
library(BeginR)
##
##  The package for the course 'Getting Started with R'
##  by @alexcourtiol (version 0.0.0.9000), is now loaded!
##  To access the slides, just type browseVignettes(package = 'BeginR'),
##  [or get_vignettes() if you also need to see the sources of the vignettes].
##
##  All sources for this package are available at https://github.com/courtiol/BeginR
##  where you can find more information on how to use this package
##  and where you can also leave comments (under 'Issues').
```

Notes:

- you can also use RStudio to load a package but that would not stay in your R makrdown file, so I do not recommend doing that

# Loading a package

Loading a package makes new functions and data available to the user:

Example:

```
library(BeginR)
##
##  The package for the course 'Getting Started with R'
##  by @alexcourtio (version 0.0.0.9000), is now loaded!
##  To access the slides, just type browseVignettes(package = 'BeginR'),
##  [or get_vignettes() if you also need to see the sources of the vignettes].
##
##  All sources for this package are available at https://github.com/courtio/BeginR
##  where you can find more information on how to use this package
##  and where you can also leave comments (under 'Issues').
```

Notes:

- you can also use RStudio to load a package but that would not stay in your R makrdown file, so I do not recommend doing that
- you can check the (exported) content of a package:

```
library(help = "BeginR")
```

# Getting ready to work with R

1 Installation

2 The RStudio IDE

3 R markdown

4 Basics of the R language

5 R packages

6 Housekeeping

7 Learning R on your own

# Updating R packages

Some things to know:

- R packages evolve quickly
- young R packages can be very buggy
- packages are not reviewed  
(CRAN tests that they can install and that the examples run without generating error or warning messages)

# Updating R packages

Some things to know:

- R packages evolve quickly
- young R packages can be very buggy
- packages are not reviewed  
(CRAN tests that they can install and that the examples run without generating error or warning messages)

Good practice:

- update your R packages frequently (I do it daily)

```
update.packages(ask = FALSE) ## or use RStudio menus
```

- check what is being changed if you heavily rely on a recent package  
(see file called NEWS easily shown if you use RStudio to update)
- contact the maintainer when you spot bugs  
(but write minimal reproductive examples otherwise they will most likely not be able to help you)

# Updating R itself

Some things to know:

- R has many bugs (like all other software)
- R bugs are reported, discussed and solved in the open (unlike most other software):  
<https://bugs.r-project.org/bugzilla3>
- each new version of R is more efficient and less buggy

# Updating R itself

Some things to know:

- R has many bugs (like all other software)
- R bugs are reported, discussed and solved in the open (unlike most other software):  
<https://bugs.r-project.org/bugzilla3>
- each new version of R is more efficient and less buggy

What to do?

- check for R new versions on CRAN
- check for what has changed if you fancy  
(<http://cran.r-project.org/doc/manuals/r-release/NEWS.html>)
- install the new version of R (unless it is not a minor update that you don't need)
- re-install all your packages

# Updating R itself

Some things to know:

- R has many bugs (like all other software)
- R bugs are reported, discussed and solved in the open (unlike most other software):  
<https://bugs.r-project.org/bugzilla3>
- each new version of R is more efficient and less buggy

What to do?

- check for R new versions on CRAN
- check for what has changed if you fancy  
(<http://cran.r-project.org/doc/manuals/r-release/NEWS.html>)
- install the new version of R (unless it is not a minor update that you don't need)
- re-install all your packages

Notes:

- some packages can help to do this: InstallR on Windows and UpdateR on macOS
- also update RStudio for full compatibility with R (Help → Check for Updates)

# Getting ready to work with R

1 Installation

2 The RStudio IDE

3 R markdown

4 Basics of the R language

5 R packages

6 Housekeeping

7 Learning R on your own

# Useful resources

RStudio cheatsheets (<https://www.rstudio.com/resources/cheatsheets/>):

**Base R Cheat Sheet**

## Getting Help

- Accessing the help files**
- mean**: Get help for a particular function.
- help.search("weighted mean")**: Search the help files for a word or phrase.
- help(package = "dplyr")**: Find help for a package.
- More about an object**
- str(iris)**: Get a summary of an object's structure.
- class(iris)**: Find the class an object belongs to.

## Using Packages

- install.packages("dplyr")**: Download and install a package from CRAN.
- library(dplyr)**: Load the package into the session, making all its functions available to use.
- dplyr::select**: Use a particular function from a package.
- data(iris)**: Load a built-in dataset into the environment.

## Working Directory

- getwd()**: Find the current working directory (where inputs are found and outputs are sent).
- setwd("C:/file/path")**: Change the current working directory.
- Use projects in RStudio to set the working directory to the folder you are working in.

## Vectors

### Creating Vectors

|                                |                          |                                   |
|--------------------------------|--------------------------|-----------------------------------|
| <code>c(2, 4, 6)</code>        | <code>2 4 6</code>       | All elements enclosed in a vector |
| <code>2:6</code>               | <code>2 3 4 5 6</code>   | An integer sequence               |
| <code>seq(2, 3, by=0.5)</code> | <code>2.0 2.5 3.0</code> | A complex sequence                |
| <code>rep(1:2, times=3)</code> | <code>1 2 1 2 1 2</code> | Repeat a vector                   |
| <code>rep(1:2, each=3)</code>  | <code>1 1 1 2 2 2</code> | Repeat elements of a vector       |

## Programming

### For Loop

```
for (variable in sequence){
  Do something
}
```

### While Loop

```
while (condition){
  Do something
}
```

### Example

```
for (i in 1:k){
  i <- i + 2k
  print(i)
  i <- i + 1
}
```

### If Statements

```
if (condition){
  Do something
} else {
  Do something different
}
```

### Example

```
if (a > 3) {
  print("Yes")
} else {
  print("No")
}
```

## Vector Functions

|                 |                  |  |
|-----------------|------------------|--|
| <b>sort(x)</b>  | <b>rev(x)</b>    | Return x sorted. Return x reversed.      |
| <b>table(x)</b> | <b>unique(x)</b> | See counts of values. See unique values. |

## Selecting Vector Elements

### By Position

|                         |                                  |
|-------------------------|----------------------------------|
| <code>x[4]</code>       | The fourth element.              |
| <code>x[-4]</code>      | All but the fourth.              |
| <code>x[-2:4]</code>    | Elements two to four.            |
| <code>x[-(2:4)]</code>  | All elements except two to four. |
| <code>x[c(1, 5)]</code> | Elements one and five.           |

### By Value

|                                   |                                 |
|-----------------------------------|---------------------------------|
| <code>x[x == 10]</code>           | Elements which are equal to 10. |
| <code>x[x &lt; 0]</code>          | All elements less than zero.    |
| <code>x[x %in% c(1, 2, 5)]</code> | Elements in the set 1, 2, 5.    |

### Named Vectors

|                         |                            |
|-------------------------|----------------------------|
| <code>x["apple"]</code> | Element with name "apple". |
|-------------------------|----------------------------|

## Reading and Writing Data

Also see the [readr package](#)

| Input  | Output                                     | Description  |
|--|--|--|
| <code>df &lt;- read.table("file.txt")</code> | <code>write.table(df, "file.txt")</code>   | Read and write a delimited text file.  |
| <code>df &lt;- read.csv("file.csv")</code>   | <code>write.csv(df, "file.csv")</code>     | Read and write a comma-separated value file. This is a special case of read.table/write.table. |
| <code>load("file.RData")</code>              | <code>save(df, file = "file.Rdata")</code> | Read and write an R data file, a file type special for R.                                      |

## Conditions

|                     |                        |                       |                           |                       |                        |                       |                       |
|---------------------|------------------------|-----------------------|---------------------------|-----------------------|------------------------|-----------------------|-----------------------|
| <code>a == b</code> | <code>!a == b</code>   | <code>a &gt; b</code> | <code>a &gt;= b</code>    | <code>a &lt; b</code> | <code>a &lt;= b</code> | <code>is.na(a)</code> | <code>is.na(b)</code> |
| <code>a != b</code> | <code>Not equal</code> | <code>a &gt; b</code> | <code>Greater than</code> | <code>a &lt; b</code> | <code>Less than</code> | <code>is.na(a)</code> | <code>is.na(b)</code> |

Note: there are many cheatsheets covering many aspects of R and several packages developed by RStudio!

# Useful resources

Official documentation:

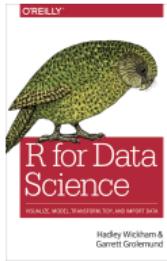
- the help files: every single (exported) function has a help file associated with it!
- official manuals (boring and difficult but thorough: <https://cran.r-project.org/manuals.html>)

# Useful resources

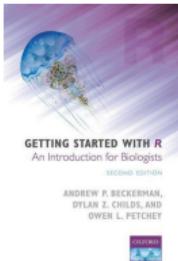
## Official documentation:

- the help files: every single (exported) function has a help file associated with it!
- official manuals (boring and difficult but thorough: <https://cran.r-project.org/manuals.html>)

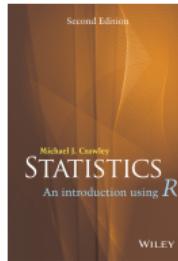
## Books (roughly sorted by amount of conceptual content):



~ 30 € (or 0€)



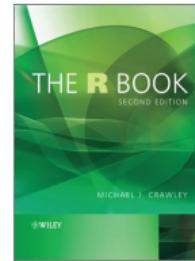
~ 30 €



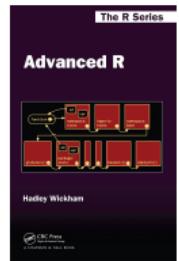
~ 35 €



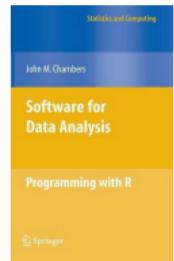
~ 40 €



~ 60 € (or 0€)



~ 60 € (or 0€)



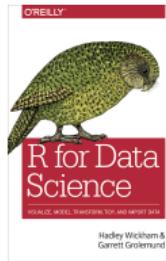
~ 90 €

# Useful resources

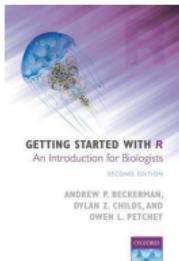
## Official documentation:

- the help files: every single (exported) function has a help file associated with it!
- official manuals (boring and difficult but thorough: <https://cran.r-project.org/manuals.html>)

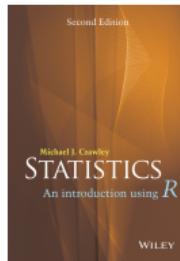
## Books (roughly sorted by amount of conceptual content):



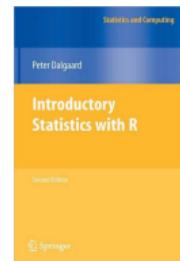
~ 30 € (or 0€)



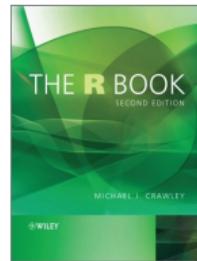
~ 30 €



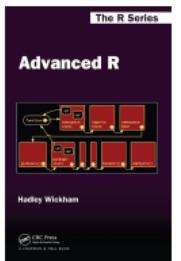
~ 35 €



~ 40 €



~ 60 €



~ 60 € (or 0€)



~ 90 €

## Journals:

- Journal of Statistical Software (<https://www.jstatsoft.org/index>)
- the R Journal (<https://journal.r-project.org>)
- many others (e.g. Journal of Open Source Software, Methods in Ecology & Evolution)

# Useful resources

Forum:

- <https://stackoverflow.com/questions/tagged/r>
- <https://stats.stackexchange.com/questions/tagged/r>
- <https://community.rstudio.com>

# Useful resources

## Forum:

- <https://stackoverflow.com/questions/tagged/r>
- <https://stats.stackexchange.com/questions/tagged/r>
- <https://community.rstudio.com>

## Blogs:

- <http://www.r-bloggers.com>
- <https://rweekly.org>
- <http://blog.revolutionanalytics.com>

# Useful resources

## Forum:

- <https://stackoverflow.com/questions/tagged/r>
- <https://stats.stackexchange.com/questions/tagged/r>
- <https://community.rstudio.com>

## Blogs:

- <http://www.r-bloggers.com>
- <https://rweekly.org>
- <http://blog.revolutionanalytics.com>

## Mailing lists:

- <https://www.r-project.org/mail.html>

# Useful resources

## Forum:

- <https://stackoverflow.com/questions/tagged/r>
- <https://stats.stackexchange.com/questions/tagged/r>
- <https://community.rstudio.com>

## Blogs:

- <http://www.r-bloggers.com>
- <https://rweekly.org>
- <http://blog.revolutionanalytics.com>

## Mailing lists:

- <https://www.r-project.org/mail.html>

## Twitter:

- #rstats

# Useful resources

Meetup groups:

- <https://www.meetup.com/Berlin-R-Users-Group/>
- <https://www.meetup.com/r-ladies-berlin/>

# Useful resources

## Meetup groups:

- <https://www.meetup.com/Berlin-R-Users-Group/>
- <https://www.meetup.com/r-ladies-berlin/>

## Courses & Workshop:

- Physalia (<https://www.physalia-courses.org>)
- DataCamp (online: <https://www.datacamp.com/courses/tech:r>)

# Useful resources

## Meetup groups:

- <https://www.meetup.com/Berlin-R-Users-Group/>
- <https://www.meetup.com/r-ladies-berlin/>

## Courses & Workshop:

- Physalia (<https://www.physalia-courses.org>)
- DataCamp (online: <https://www.datacamp.com/courses/tech:r>)

## Conferences:

- useR (<https://user2019.r-project.org>)
- European R User meetings (<https://erum.io>)

The best person who can teach you **R** is YOU!

After having learned some basics, just open RStudio and test your understanding by performing experiments!

Do not copy and paste stuff from internet without trying to understand!!!