Getting ready to work with **R**

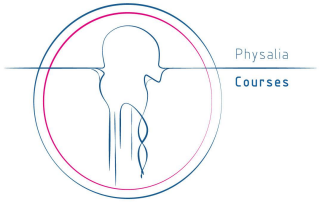
Alexandre Courtiol

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**

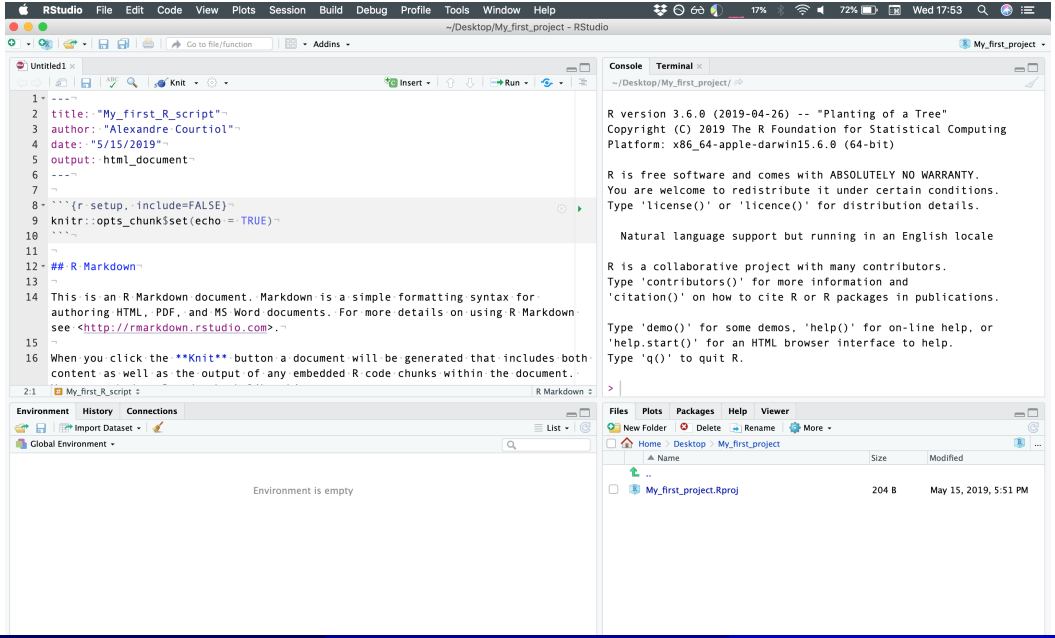
Practice

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**

Getting ready to work with **R**

The RStudio Integrated Desktop Environment



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**

The top menu

File Edit Code View Plots Session Build Debug Profile Tools Window Help

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

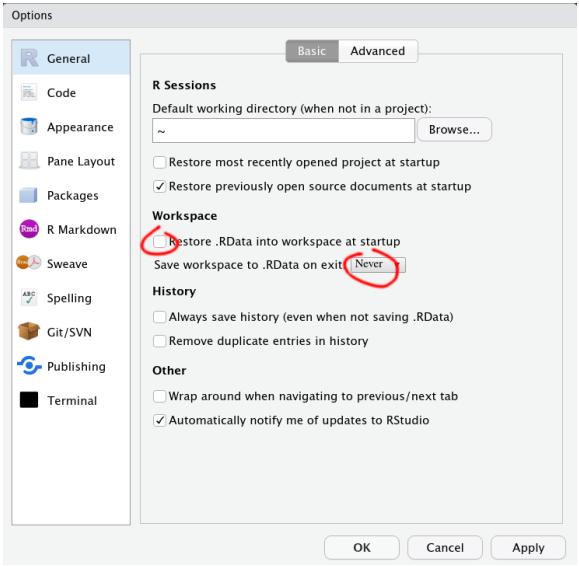
1. create a new project
2. create a new **R** markdown file
3. save the created **R** markdown file into the project folder directory
4. have a look at the RStudio cheatsheet

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)

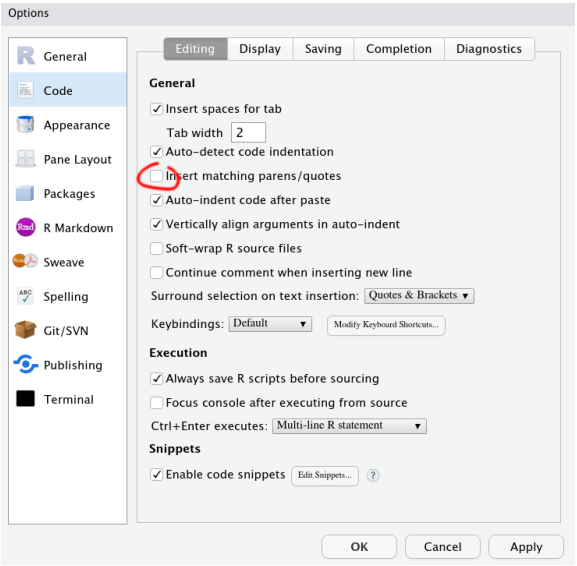
My tips about the global options

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



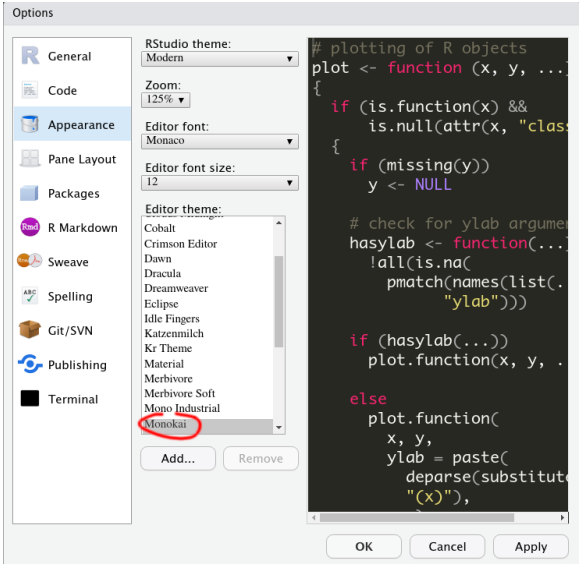
My tips about the global options

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



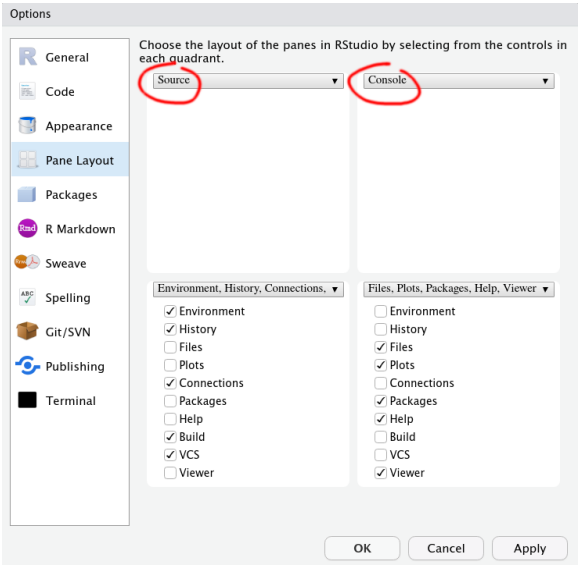
My tips about the global options

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



My tips about the global options

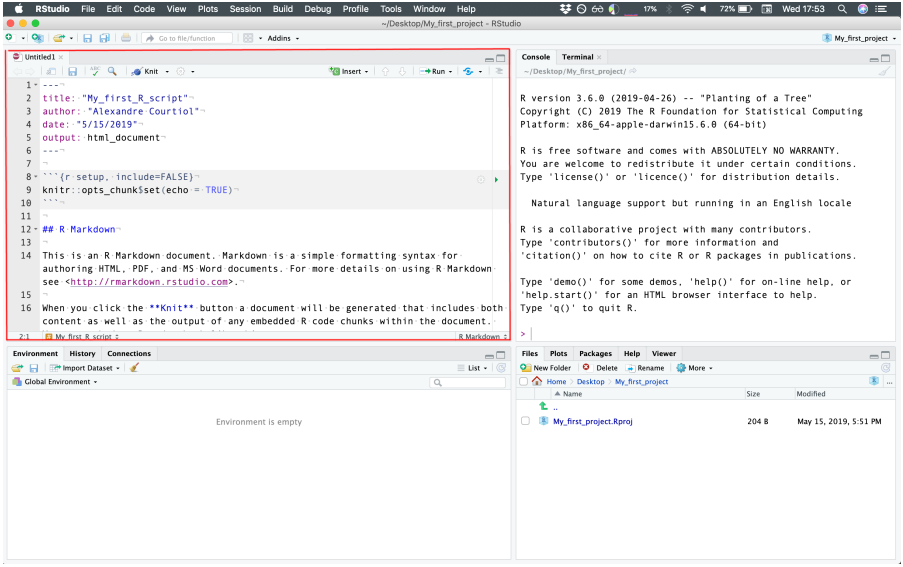
4. I like to put the `source` and the `console` panes side-by-side to give them more vertical space on the screen



Getting ready to work with **R**

The Source pane

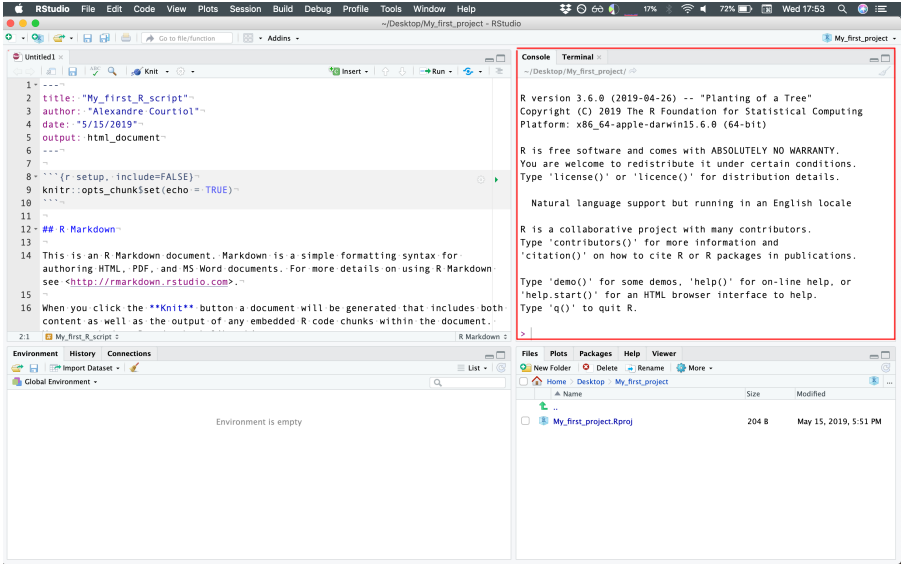
This is where you type the code you want to keep



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

The Console pane

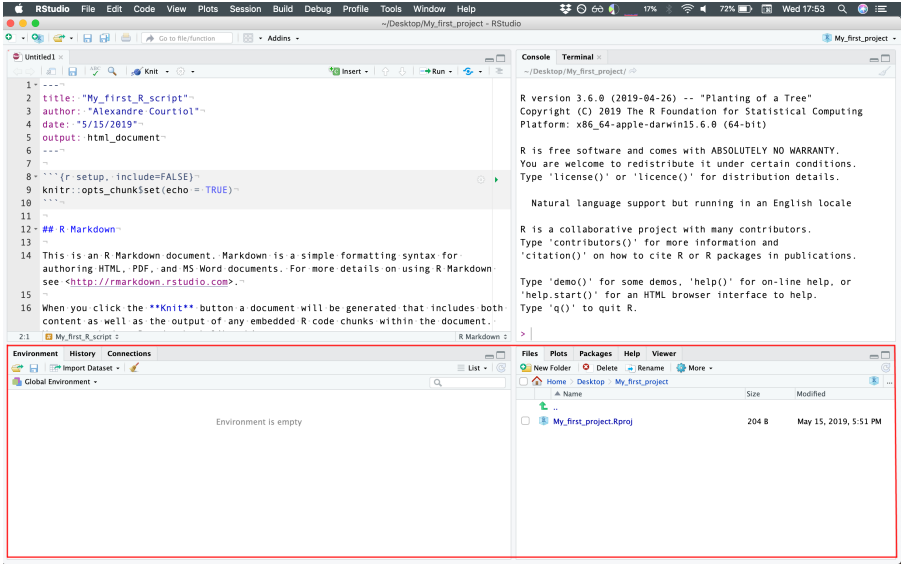
This is where you can run **R** code directly



Notes: tabs give you access to a 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 object 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 object 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**

Getting ready to work with **R**

What is **R** markdown?

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

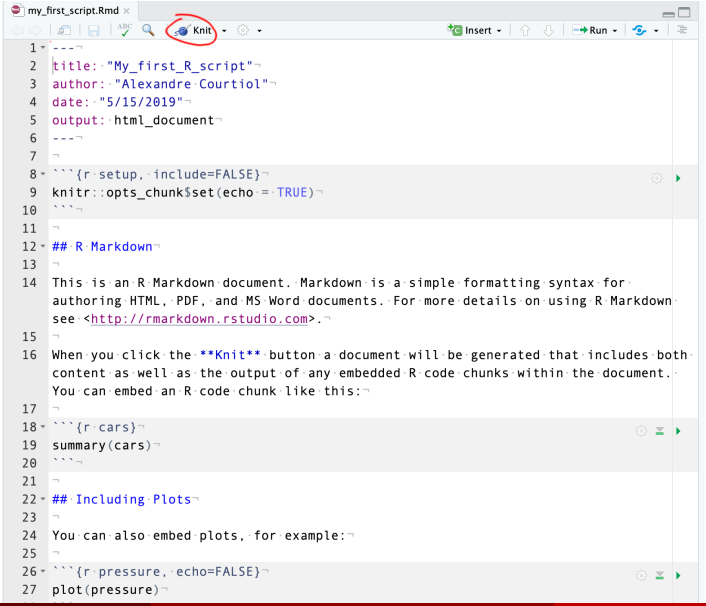
- text
- **R** code
- **R** outputs (e.g. plots, tables)

...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 LaTeX and **R**, but **R** markdown remains simple)

Practice

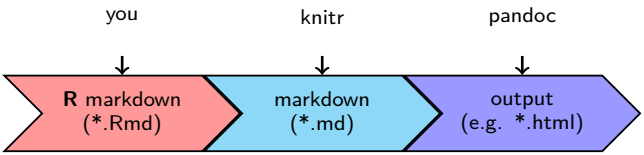
Render the **R** markdown file using `knitr` and observe the outcome



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?

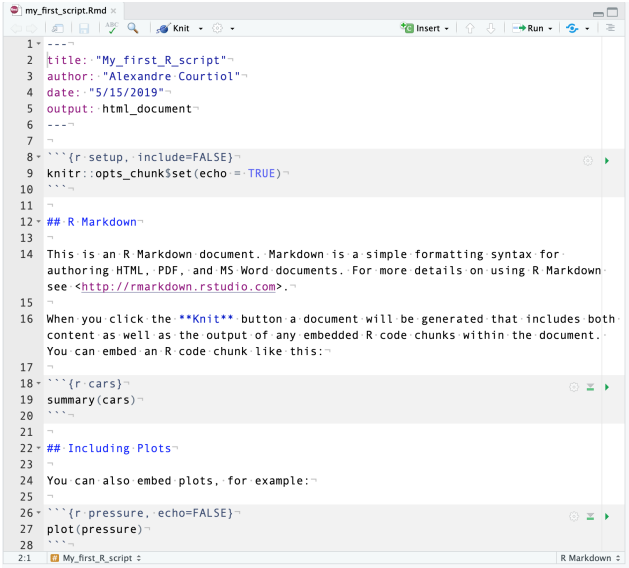


Notes:

- `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 `pandox` are orchestrated by the **R** package `rmarkdown`

General structure of a **R** markdown file

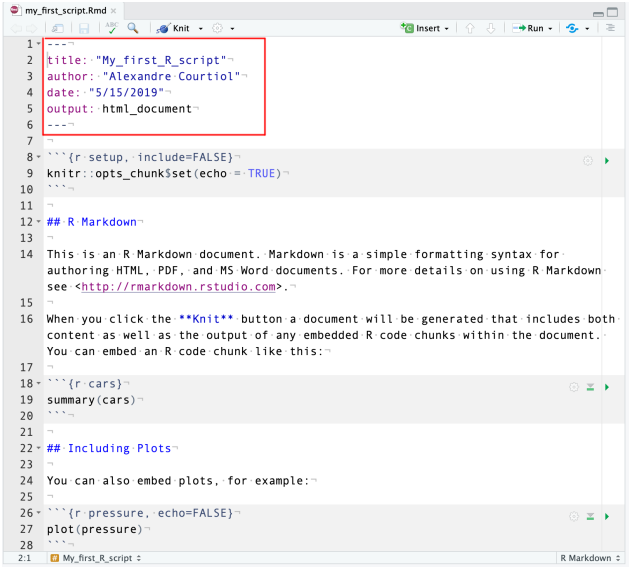
3 parts:



General structure of a **R** markdown file

3 parts:

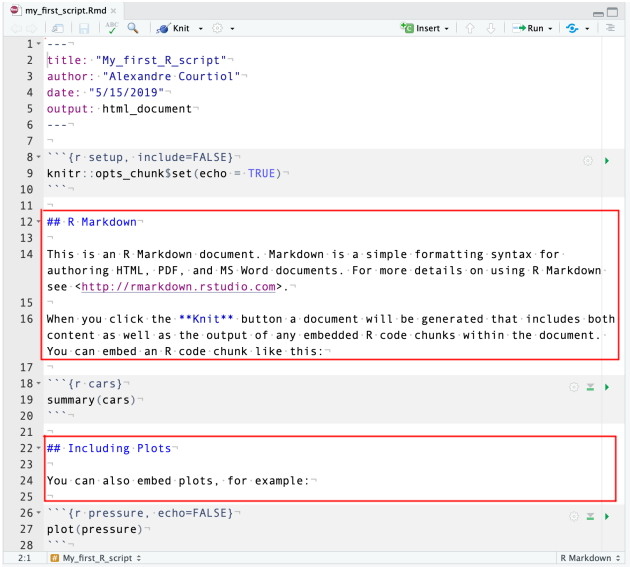
- the header (writen in `YAML`)



General structure of a **R** markdown file

3 parts:

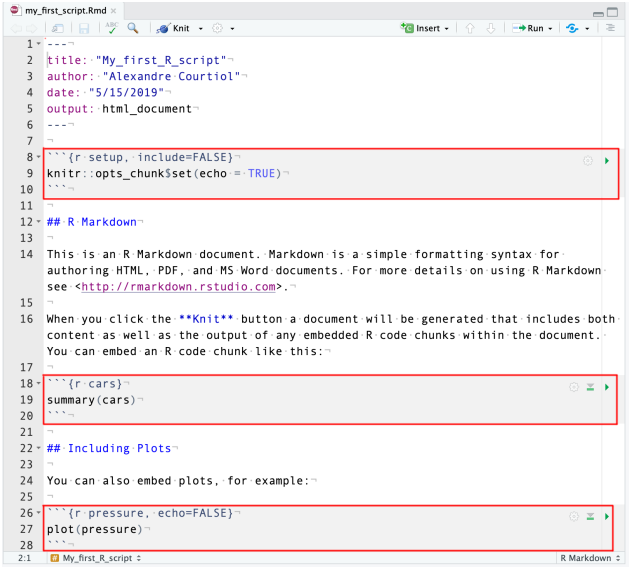
- the header (writen in `YAML`)
- the text (writen in `markdown`)



General structure of a **R** markdown file

3 parts:

- the header (writen in `YAML`)
- the text (writen in `markdown`)
- the code chunks (writen in **R** or alternatives)



Getting ready to work with **R**

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
---
```

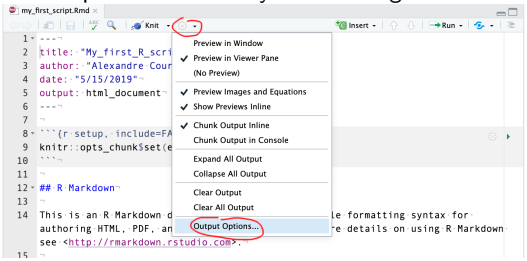
Notes:

- 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:



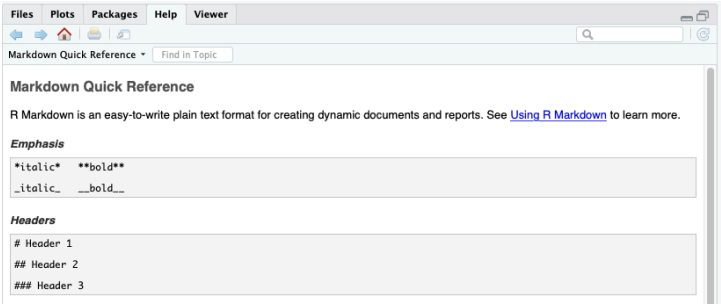
For info on what you can change, check:

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

Getting ready to work with **R**

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!



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

- notice that the structure of the document appears at the bottom of the `Source` pane
- 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**

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

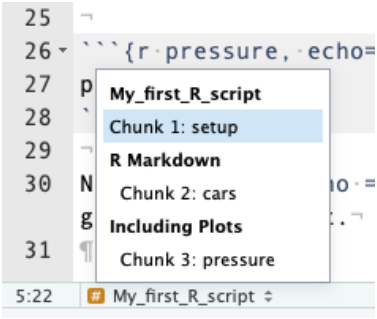
````
```{language_name chunk name, option1=argument1, option2=argument2, ...}
some code ## a comment
```
````

Examples:

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

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

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



Examples of code chunk options

Some options good to know:

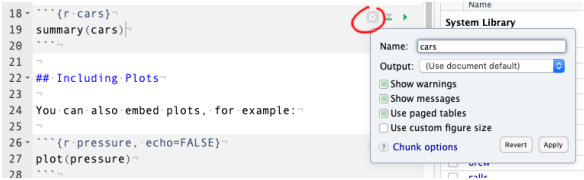
Option	what for?	default
echo	display the code	TRUE
error	stop on error	TRUE
eval	run the chunk	TRUE

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:



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:

For more options, check:

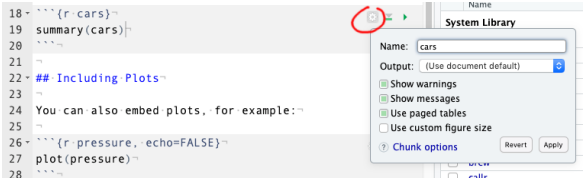
- The **R** `Markdown Cheat Sheet` (for a few more)
- `https://yihui.name/knitr/options` (for all of them)

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:



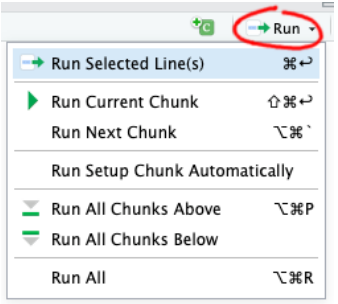
For more options, check:

- 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.

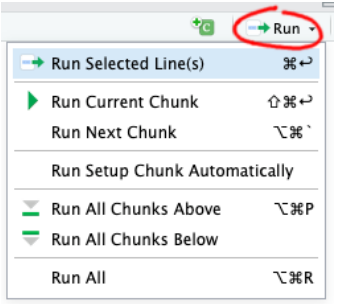
Running a code chunk without knitting

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

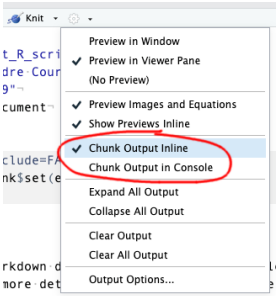


Running a code chunk without knitting

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**

Getting ready to work with **R**

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

1. only use the `Console` pane to mess around
2. write proper **R** code inside a chunk in your Rmd document
3. knit chunks one by one before knitting the whole Rmd document
4. name objects with useful names

Getting ready to work with **R**

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**

Functions

```r
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

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'
```

Note: equal signs and arrows are only equivalent in the so-called "global environment":

```
y = 2
y
## [1] 2
y <- 3
y
## [1] 3
```

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'
```

Note: equal signs and arrows are only equivalent in the so-called "global environment":

```
y = 2
y
## [1] 2
y <- 3
y
## [1] 3
```

But not inside functions:

```
sign(y <- -5) ## dangerous: avoid!
## [1] -1
y
## [1] -5
sign(x = y <- -5) ## same as above
## [1] -1
```

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'
```

Note: equal signs and arrows are only equivalent in the so-called "global environment":

```
y = 2
y
## [1] 2
y <- 3
y
## [1] 3
```

But not inside functions:

```
sign(y <- -5) ## dangerous: avoid!
## [1] -1
y
## [1] -5
sign(x = y <- -5) ## same as above
## [1] -1
```

So better to stick to arrows for creating objects and to equal signs for defining arguments!

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

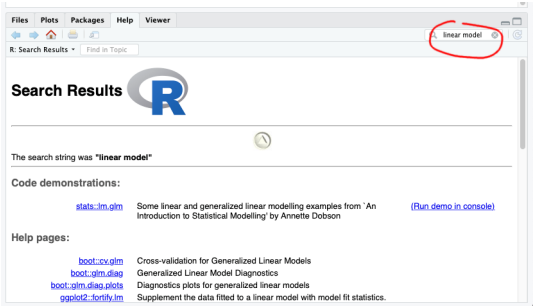
To find the name of the function you are look for, you may try:

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

or

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

or



Getting ready to work with **R**

The concept of an **R** package

Packages extend **R** functionalities:

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

The concept of an **R** package

Packages extend **R** functionalities:

- for most users; e.g. `ggplot2`
- for specific users; e.g. `IsoriX`
- for developers; eg. `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:
  - 14233 packages are available on `cran.r-project.org`
  - $\sim$ 1500 packages aimed at bioinformatics on `bioconductor.org`
  - many more on `github.com`
  - many more shared between users in other ways

The concept of an **R** package

Packages extend **R** functionalities:

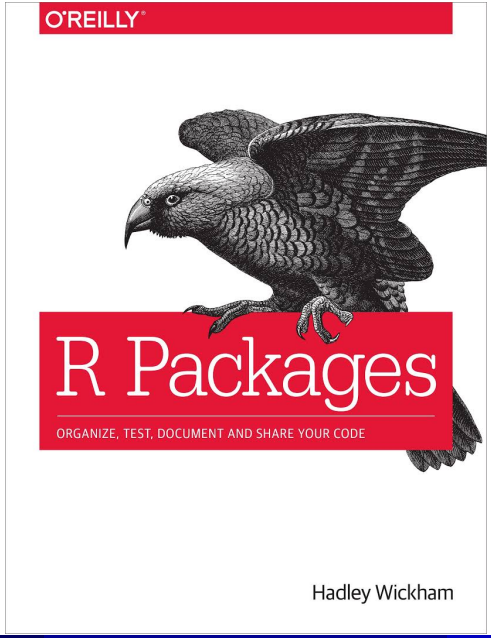
- for most users; e.g. `ggplot2`
- for specific users; e.g. `IsoriX`
- for developers; eg. `Rcpp`

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:
  - 14233 packages are available on `cran.r-project.org`
  - $\sim$ 1500 packages aimed at bioinformatics on `bioconductor.org`
  - many more on `github.com`
  - many more shared between users in other ways

Note: packages can be used to create research compendia!

Creating your own package is actually quite easy once you know **R**



Installing a package

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

```
install.packages("dpylr")  ## install dplyr
```

Installing a package

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

```
install.packages("dpylr")  ## install dplyr
```

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

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

```
install.packages("dpylr")  ## install dplyr
```

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/`)
- Xcode if you use macOS (`https://developer.apple.com/xcode/`)
- 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/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
```

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.

Loading a package

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

Example:

```
library(BeginR)
## Error in library(BeginR): there is no package called 'BeginR'
```

Loading a package

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

Example:

```
library(BeginR)
## Error in library(BeginR): there is no package called 'BeginR'
```

You can check the (exported) content of a package:

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

Getting ready to work with **R**

Updating **R** packages

Some things to know:

- **R** packages evovle 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 evovle 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 being 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

Note 1: some packages can help to do this: `InstallR` on Windows and `UpdateR` on macOS.

Note 2: also update RStudio for full compatibility with **R**.

Getting ready to work with **R**

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`)

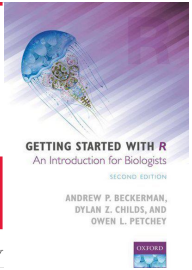
Useful resources

Official documentation:

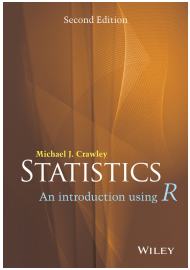
- the help files: every single (exported) function has a help file associated with it!
- official manuals (boring and difficult but thorough: `https://cran.r-project.org/manuals.html`)

Books (roughly sorted by amount of conceptual content):

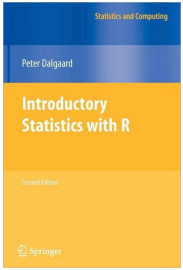
$\sim$ 30 €



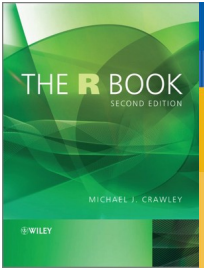
$\sim$ 30 €



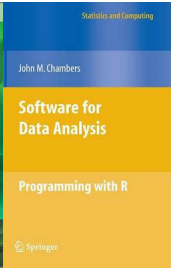
$\sim$ 35 €



$\sim$ 40 €



$\sim$ 60 €



$\sim$ 90 €

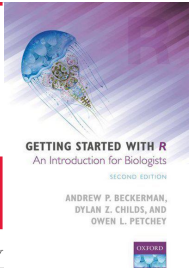
Useful resources

Official documentation:

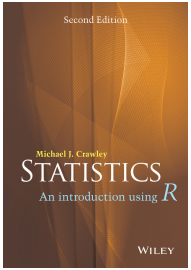
- the help files: every single (exported) function has a help file associated with it!
- official manuals (boring and difficult but thorough: `https://cran.r-project.org/manuals.html`)

Books (roughly sorted by amount of conceptual content):

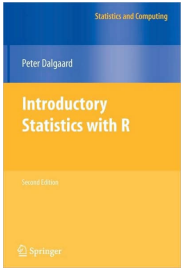
~ 30 €



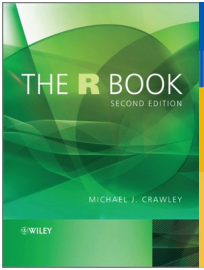
~ 30 €



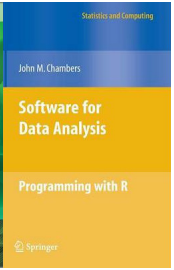
~ 35 €



~ 40 €



~ 60 €



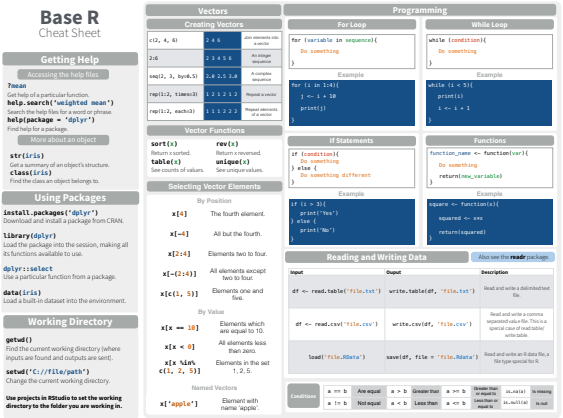
~ 90 €

Journals:

- Journal of Statistical Software (`https://www.jstatsoft.org/index`)
- The **R** Journal (`https://journal.r-project.org`)

Useful resources

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



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

Useful resources

Blogs:

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

Useful resources

Blogs:

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

Forum:

- `https://stackoverflow.com/questions/tagged/r`
- `https://stats.stackexchange.com/questions/tagged/r`

Useful resources

Blogs:

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

Forum:

- `https://stackoverflow.com/questions/tagged/r`
- `https://stats.stackexchange.com/questions/tagged/r`

Mailing lists:

- `https://www.r-project.org/mail.html`

Useful resources

Blogs:

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

Forum:

- `https://stackoverflow.com/questions/tagged/r`
- `https://stats.stackexchange.com/questions/tagged/r`

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/rladies-berlin/`

Useful resources

Meetup groups:

- `https://www.meetup.com/Berlin-R-Users-Group/`
- `https://www.meetup.com/rladies-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/rladies-berlin/`

Courses & Workshop:

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

Conferences:

- useR (`https://user2018.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 the console and test your understanding by performing experiments!

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