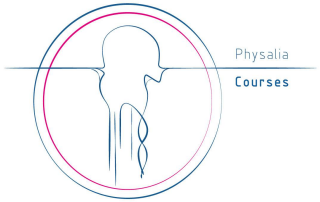
Getting started with **R**

Alexandre Courtiol

Leibniz Institute of Zoo and Wildlife Research

June 2018



Getting started with **R**

1. Before we start
2. What is **R**?
3. First steps in **R**

About this course

I will:

- give you all the slides (so write only what is not being displayed)
- explain you the main principles of the **R** language, so to give you good basics
- provide you with short examples on how to use some of the most useful functions in **R**
- provide you with suggestions on how to learn more about **R** on your own

About this course

I will:

- give you all the slides (so write only what is not being displayed)
- explain you the main principles of the **R** language, so to give you good basics
- provide you with short examples on how to use some of the most useful functions in **R**
- provide you with suggestions on how to learn more about **R** on your own

You will not:

- be fluent in the **R** language after one week
- be able to remember how to implement many of the things we will see

About this course

I will:

- give you all the slides (so write only what is not being displayed)
- explain you the main principles of the **R** language, so to give you good basics
- provide you with short examples on how to use some of the most useful functions in **R**
- provide you with suggestions on how to learn more about **R** on your own

You will not:

- be fluent in the **R** language after one week
- be able to remember how to implement many of the things we will see

You should:

- accept that, at the beginning, it will be a little abstract
- focus on the big picture (try to understand the logic) $\rightarrow$ consider the long term gains
- ask any silly question that pops up in your creative minds
- let me know immediately when you stop following

Who am I?

- evolutionary biologist / statistician
- studies in France (Montpellier), postdoc in the UK (Sheffield)
- senior researcher at Leibniz IZW / lecturer at Freie University (Berlin)

Who am I?

- evolutionary biologist / statistician
- studies in France (Montpellier), postdoc in the UK (Sheffield)
- senior researcher at Leibniz IZW / lecturer at Freie University (Berlin)
- experience with **R**:
  - since 2003: studying **R** (still ongoing)
  - since 2008: using **R** most days
  - since 2010: teaching **R**
  - since 2013: debugging **R** packages
  - since 2016: developing **R** packages
  - since 2018: debugging **R**

Getting started with **R**

1 Before we start

2 What is **R**?

3 First steps in **R**

Getting started with R

1. Before we start
2. What is R?
   - R in brief
   - the history and pre-history of R
   - why use R?
   - who uses R?
3. First steps in R
   - installing R
   - arithmetic
   - script
   - objects
   - functions
   - packages
   - housekeeping
   - learning R on your own

R in brief

R is a programming language and software environment for statistical computing & graphics.

Key points about **R**:

- free for all
- open source (explore: `https://github.com/wch/r-source`)

R in brief

R is a programming language and software environment for statistical computing & graphics.

Key points about **R**:

- free for all
- open source (explore: `https://github.com/wch/r-source`)
- polyvalent:
  - from laptop to most advanced supercomputers
  - local or remote
  - Windows, MacOS, linux or many other Unix-based systems

R in brief

R is a programming language and software environment for statistical computing & graphics.

Key points about **R**:

- free for all
- open source (explore: `https://github.com/wch/r-source`)
- polyvalent:
  - from laptop to most advanced supercomputers
  - local or remote
  - Windows, MacOS, linux or many other Unix-based systems
- rich (tons of **R** packages out there)
- cutting edge (check updates for today: `http://dirk.eddelbuettel.com/cranberries/cran/updated/`)

R in brief

R is a programming language and software environment for statistical computing & graphics.

Key points about **R**:

- free for all
- open source (explore: `https://github.com/wch/r-source`)
- polyvalent:
  - from laptop to most advanced supercomputers
  - local or remote
  - Windows, MacOS, linux or many other Unix-based systems
- rich (tons of **R** packages out there)
- cutting edge (check updates for today: `http://dirk.eddelbuettel.com/cranberries/cran/updated/`)
- used by millions
- **R** is the best software environment for statistical computing, but it is far from perfect!

Getting started with R

1. Before we start
2. What is R?
   - R in brief
   - the history and pre-history of R
   - why use R?
   - who uses R?
3. First steps in R
   - installing R
   - arithmetic
   - script
   - objects
   - functions
   - packages
   - housekeeping
   - learning R on your own

A short history of **S**/**R**

S (`http://ect.bell-labs.com/sl/S/`)

- 1976-1980: version 1: interactive statistical system, Fortran based (Becker, Chambers, & al. at Bell Labs)
- 1980-1988: version 2: portable version (thanks to Unix)
- 1988: version 3 (**S**3): "everything is an object" paradigm, C-based (very much like R)
- 1991: a large statistical modeling toolbox is added to **S**3
- 1993: **S**+ exclusive license (to StatSci, later MathSoft, later SolutionMetrics)
- 1998: version 4 (**S**4): advanced object-oriented features
- 2012: **S**+ becomes TIBCO Enterprise Runtime for **R** (TERR)

A short history of **S**/**R**

S (`http://ect.bell-labs.com/sl/S/`)

- 1976-1980: version 1: interactive statistical system, Fortran based (Becker, Chambers, & al. at Bell Labs)
- 1980-1988: version 2: portable version (thanks to Unix)
- 1988: version 3 (**S**3): "everything is an object" paradigm, C-based (very much like R)
- 1991: a large statistical modeling toolbox is added to **S**3
- 1993: **S**+ exclusive license (to StatSci, later MathSoft, later SolutionMetrics)
- 1998: version 4 (**S**4): advanced object-oriented features
- 2012: **S**+ becomes TIBCO Enterprise Runtime for **R** (TERR)

R (`https://www.r-project.org/about.html`)

- 1993: the replication of **S** as the **R** project starts (Ihaka & Gentleman at University of Auckland)
- 23/04/1997: first version of **R** archieved on The Comprehensive R Archive Network (CRAN)
- 05/12/1997: **R** version 0.6 is part of GNU project ("freedom to share, freedom to change")
- 29/02/2000: **R** version 1.0 (judged stable enough for production use by the R Development Core Team)

Getting started with R

1. Before we start
2. What is R?
   - R in brief
   - the history and pre-history of R
   - why use R?
   - who uses R?
3. First steps in R
   - installing R
   - arithmetic
   - script
   - objects
   - functions
   - packages
   - housekeeping
   - learning R on your own

Is **R** good for you?

Good for:

- data manipulation
- plots, including GIS
- analysing small, medium and big data
- programming around data

Is **R** good for you?

Good for:

- data manipulation
- plots, including GIS
- analysing small, medium and big data
- programming around data

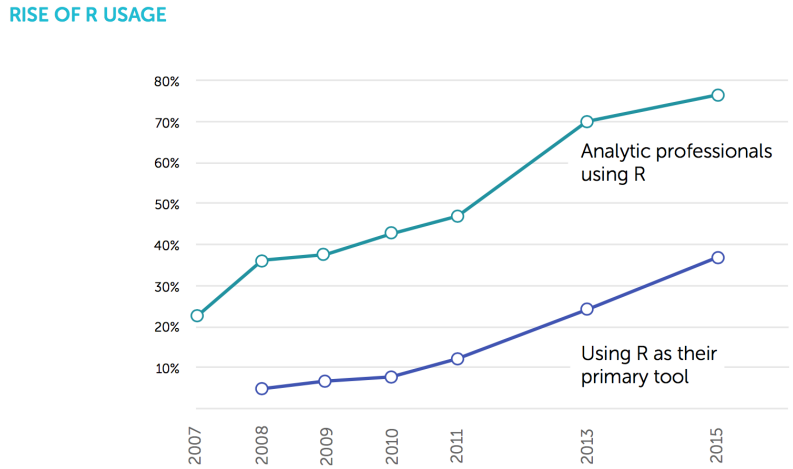
Not optimal for:

- beginners
- data entry
- formal algebra

Getting started with R

1. Before we start
2. What is R?
   - R in brief
   - the history and pre-history of R
   - why use R?
   - who uses R?
3. First steps in R
   - installing R
   - arithmetic
   - script
   - objects
   - functions
   - packages
   - housekeeping
   - learning R on your own

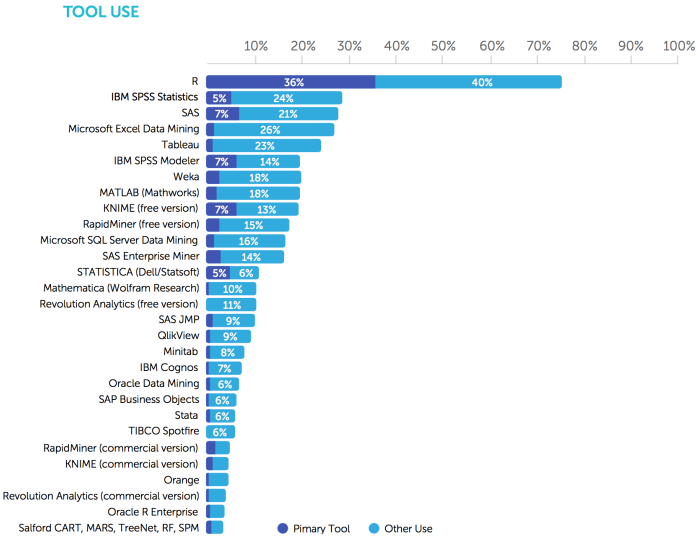
Who uses **R**?



Rexer Analytics

[1220 analytic professionals from 72 countries participated in this survey]

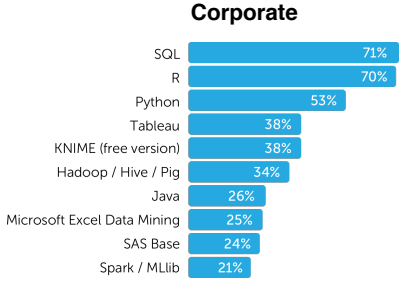
What else?

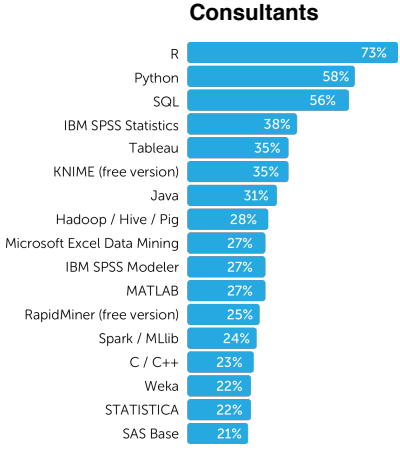


Rexer Analytics

Most Data Scientists use Multiple Tools

What data science / analytic tools, technologies, and languages did you use in the past year?



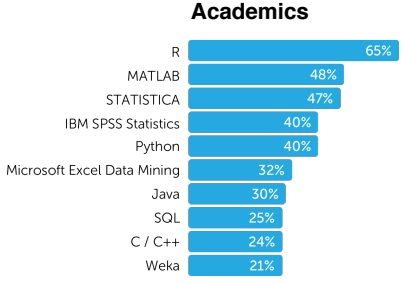


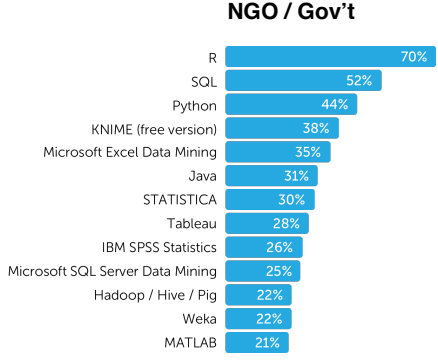
© 2017 Rexer Analytics

All tools used by more than 20% of a group are shown

Most Data Scientists use Multiple Tools

What data science / analytic tools, technologies, and languages did you use in the past year?





© 2017 Rexer Analytics

All tools used by more than 20% of a group are shown

Rich companies rely on **R** too!

Some examples:
(`http://blog.revolutionanalytics.com/2014/05/companies-using-r-in-2014.html`)

- Facebook (data analysis, big-data visualization, user behaviour analysis)
- Google (advertising effectiveness, economic forecasting, and big-data statistical modeling)
- Twitter (data visualization and semantic clustering)
- The City of Chicago (food poisoning monitoring)
- The New York Times (interactive features such as the Dialect Quiz and the Election Forecast)
- Microsoft (Xbox matchmaking + plus much more these days!!)
- The Human Rights Data Analysis Group (counts of casualties in war zones)
- ANZ Bank (credit risk analysis)
- The FDA (regulatory drug approvals process)
- Monsanto (statistical analysis in plant breeding, fertility mapping and yield forecasting)
- Lloyds of London (risk analysis and catastrophe modeling)
- RealClimate.org (climate change analysis)
- NOAA (flood warnings)

Getting started with **R**

1. Before we start
2. What is **R**?
3. First steps in **R**

Getting started with R

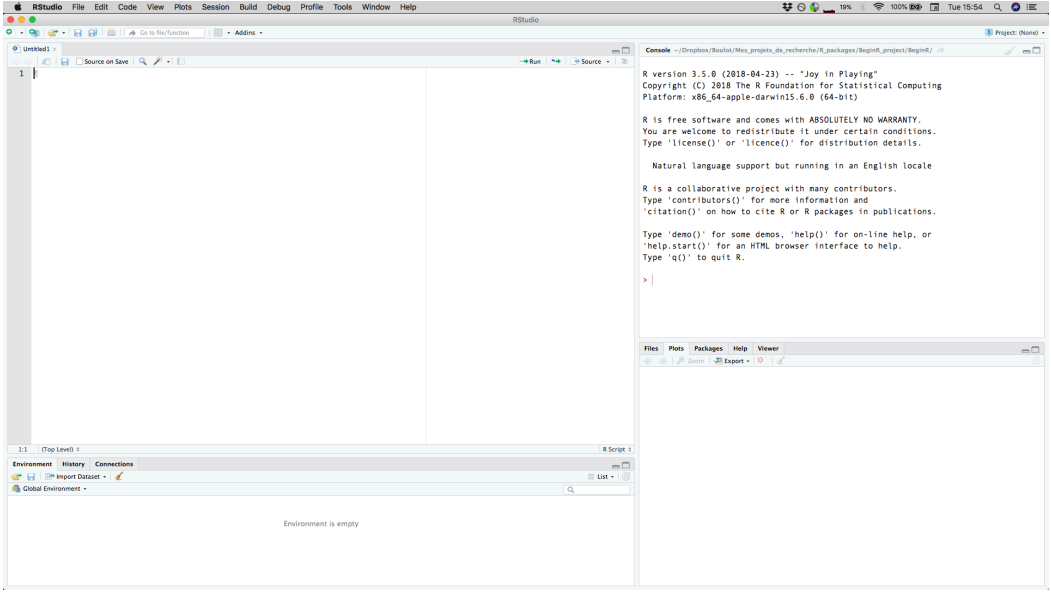
1. Before we start
2. What is R?
   - R in brief
   - the history and pre-history of R
   - why use R?
   - who uses R?
3. First steps in R
   - installing R
   - arithmetic
   - script
   - objects
   - functions
   - packages
   - housekeeping
   - learning R on your own

Installation steps

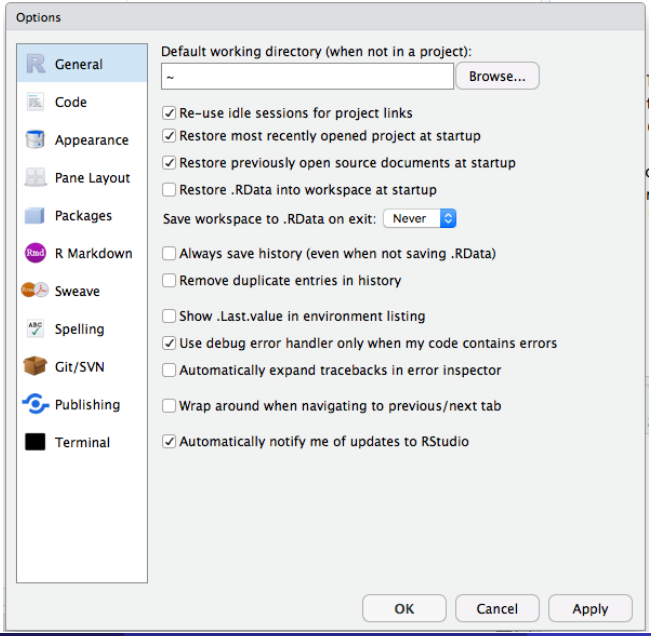
1. check that you do get internet access
2. install **R**: `https://cran.r-project.org/`
3. install the RStudio IDE: `https://www.rstudio.com/products/rstudio/download/`
4. open RStudio

Note: we will use RStudio but you don't have to (the RStudio IDE is free and open source).

RStudio


RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help
RStudio
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
Console ~/Dropbox/Boulot/Mes_projets_de_recherche/R_packages/BeginR_project/BeginR/
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Files Plots Packages Help Viewer
Export
1:1 (Top Level)
R Script
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty

Better default setting for RStudio



Getting started with R

1. Before we start
2. What is R?
   - R in brief
   - the history and pre-history of R
   - why use R?
   - who uses R?
3. First steps in R
   - installing R
   - arithmetic
   - script
   - objects
   - functions
   - packages
   - housekeeping
   - learning R on your own

Basic arithmetic

Try in the following in the "Console" pannel:

```
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!

Getting started with R

1. Before we start
2. What is R?
   - R in brief
   - the history and pre-history of R
   - why use R?
   - who uses R?
3. First steps in R
   - installing R
   - arithmetic
   - script
   - objects
   - functions
   - packages
   - housekeeping
   - learning R on your own

The concept of an **R** script

All instructions must be written as a computer script!

- it is just a text file (no need for **R** to read it, it never gets corrupted)
- the script must be saved at a known location
- all non-**R** instruction must be preceeded by the character # (called number sign, hash, or pound sign)

The concept of an **R** script

All instructions must be written as a computer script!

- it is just a text file (no need for **R** to read it, it never gets corrupted)
- the script must be saved at a known location
- all non-**R** instruction must be preceeded by the character # (called number sign, hash, or pound sign)

```
##############################
## this is my first R script ##
##############################

### simple arithmetic
1 + 1 ## compute 1 + 1
```

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

```
#1 + 2 ## commented lines of code won't run!

## Note: I personally use ## (or more) for explanation and # for preventing code to run because if you uncomment using the menu or shortcut,
## then explanation do not risk to be run (it would trigger errors).
```

The concept of an R script

All instructions must be written as a computer script!

- it is just a text file (no need for **R** to read it, it never gets corrupted)
- the script must be saved at a known location
- all non-**R** instruction must be preceeded by the character # (called number sign, hash, or pound sign)

```
##############################
## this is my first R script ##
##############################

### simple arithmetic
1 + 1 ## compute 1 + 1
```

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

```
#1 + 2 ## commented lines of code won't run!

## Note: I personally use ## (or more) for explanation and # for preventing code to run because if you uncomment using the menu or shortcut,
## then explanation do not risk to be run (it would trigger errors).
```

Why bother writing a script?

- transparent & reproducible
- easy to share & modify

Good practice

1. only use the "Console" pannel to mess around
2. write a script and comment it thoroughly
3. make sure your script always work by re-running the whole script often
4. name objects with useful names

Getting started with R

1. Before we start
2. What is R?
   - R in brief
   - the history and pre-history of R
   - why use R?
   - who uses R?
3. First steps in R
   - installing R
   - arithmetic
   - script
   - objects
   - functions
   - packages
   - housekeeping
   - 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
```

The concept of an **R** object

What is an object?

- everything in **R** is an object
- objects have names
- objects allow abstraction
- objects belongs to classes for which specific methods exist (and can be created)

Note: we will come back on that later (for the programming session).

Note for geeks who know other computer languages

R objects are (by default) not mutable (there is copy on demand):

```
a <- 1
b <- a
b <- b + 1
b
```

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

```
a ## although 'b' derives from 'a' changing 'b' has no impact on 'a' (because 'a' and 'b' share different physical memory addresses)
```

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

Note: if you don't know other computer languages, it just behaves as you would expect while most other programming languages don't.

Getting started with R

1. Before we start
2. What is R?
   - R in brief
   - the history and pre-history of R
   - why use R?
   - who uses R?
3. First steps in R
   - installing R
   - arithmetic
   - script
   - objects
   - functions
   - packages
   - housekeeping
   - learning R on your own

Functions

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

```
##
## To cite R in publications use:
##
##   R Core Team (2018). 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 = {2018},
##     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 (2018). 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 = {2018},
##     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 (2018). 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 = {2018},
##     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
```

Key principles of the **R** language

- Everything that exists in **R** is an object
- Everything that happens in **R** is a function call

John M. Chambers

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
```

Getting started with R

1. Before we start
2. What is R?
   - R in brief
   - the history and pre-history of R
   - why use R?
   - who uses R?
3. First steps in R
   - installing R
   - arithmetic
   - script
   - objects
   - functions
   - packages
   - housekeeping
   - 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 developpers; 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 developpers; 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:
  - 12598 packages are available on `cran.r-project.org`
  - $\sim$ 1500 packages aimed at bioinformatic 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:

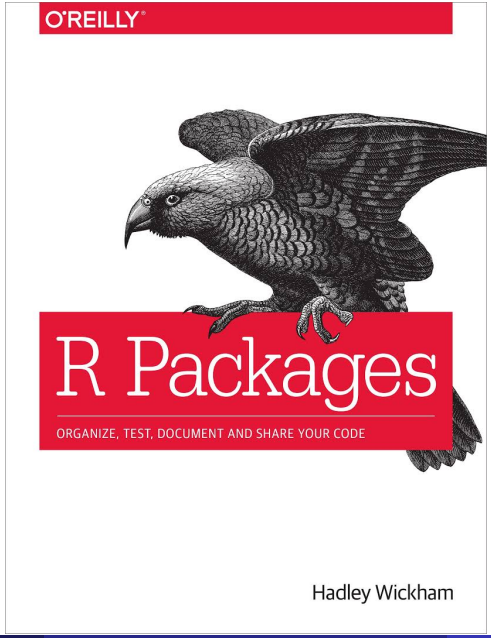
- for most users; e.g. `ggplot2`
- for specific users; e.g. `IsoriX`
- for developpers; eg. `Rcpp`

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:
  - 12598 packages are available on `cran.r-project.org`
  - $\sim$ 1500 packages aimed at bioinformatic 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)
```

```
##
##  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').
```

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').
```

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

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

Getting started with R

1. Before we start
2. What is R?
   - R in brief
   - the history and pre-history of R
   - why use R?
   - who uses R?
3. First steps in R
   - installing R
   - arithmetic
   - script
   - objects
   - functions
   - packages
   - housekeeping
   - learning R on your own

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 in general 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 in general more efficient and less buggy

What to do?

- check for **R** new versions on CRAN
- check for what has changed if you fancy (`https://cran.r-project.org/index.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 in general more efficient and less buggy

What to do?

- check for **R** new versions on CRAN
- check for what has changed if you fancy (`https://cran.r-project.org/index.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 started with R

1. Before we start
2. What is R?
   - R in brief
   - the history and pre-history of R
   - why use R?
   - who uses R?
3. First steps in R
   - installing R
   - arithmetic
   - script
   - objects
   - functions
   - packages
   - housekeeping
   - learning R on your own

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`)

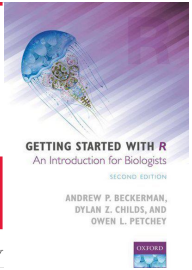
Useful resources

Official documentation:

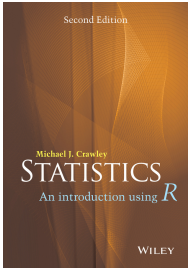
- the help files: every single (exported) function has a help file associated with it!
- official manuals (boring and difficult but thorough: `https://cran.r-project.org/manuals.html`)

Books (roughly sorted by amount of conceptual content):

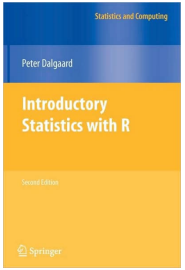
$\sim$ 30 €



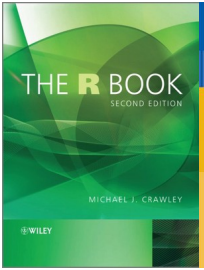
$\sim$ 30 €



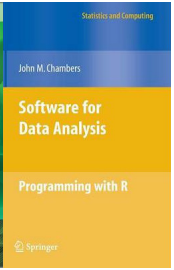
$\sim$ 35 €



$\sim$ 40 €



$\sim$ 60 €



$\sim$ 90 €

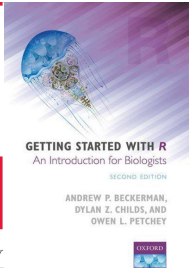
Useful resources

Official documentation:

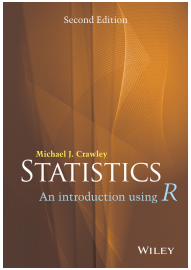
- the help files: every single (exported) function has a help file associated with it!
- official manuals (boring and difficult but thorough: `https://cran.r-project.org/manuals.html`)

Books (roughly sorted by amount of conceptual content):

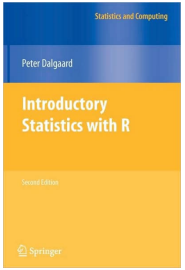
~ 30 €



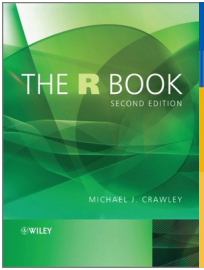
~ 30 €



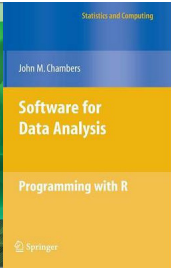
~ 35 €



~ 40 €



~ 60 €



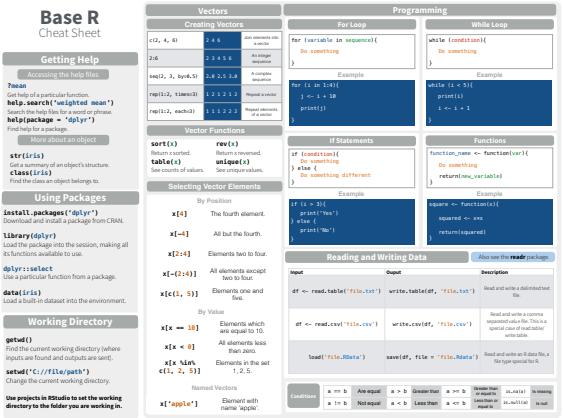
~ 90 €

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!!!