

# Introduction to R, Rstudio & Project Management

Berry Boessenkool, [uni-potsdam.de](http://uni-potsdam.de), May 2017

`berry-b@gmx.de`  
[github.com/brry](https://github.com/brry)

[swc-bb.github.io/2017-05-17-r-workshop](http://swc-bb.github.io/2017-05-17-r-workshop)

*Presentation template generated with `berryFunctions::createPres`*

# Introduction to R, Rstudio & Project Management



Berry Boessenkool, [uni-potsdam.de](http://uni-potsdam.de), May 2017

[berry-b@gmx.de](mailto:berry-b@gmx.de)  
[github.com/brry](https://github.com/brry)

[swc-bb.github.io/2017-05-17-r-workshop](http://swc-bb.github.io/2017-05-17-r-workshop)

*Presentation template generated with `berryFunctions::createPres`*

# Introduction to R, Rstudio & Project Management



Berry Boessenkool, [uni-potsdam.de](http://uni-potsdam.de), May 2017

[berry-b@gmx.de](mailto:berry-b@gmx.de)  
[github.com/brry](https://github.com/brry)

[swc-bb.github.io/2017-05-17-r-workshop](http://swc-bb.github.io/2017-05-17-r-workshop)

*Presentation template generated with `berryFunctions::createPres`*

# Survey

knowledge survey to determine focus for this session

[bit.ly/knowR](https://bit.ly/knowR)

## RStudio

The screenshot shows the RStudio environment with several annotations in red and orange text boxes:

- RUN CODE**: A red circle highlights the 'Run' button in the top toolbar.
- SCRIPTS**: A red circle highlights the 'Source' button in the top toolbar.
- OBJECTS IN WORKSPACE**: A red circle highlights the 'Environment' tab in the top right pane.
- DOCUMENTATION**: A red circle highlights the 'Help' button in the top right pane.
- PLOTS**: A red circle highlights the 'Plots' tab in the bottom right pane.
- CODE EXECUTION**: A red circle highlights the 'Console' tab in the bottom left pane.

The main editor window displays R code for a project named 'GFZ\_Pegel\_Rhein\_Analysis.R'. The code includes loading shapefiles, plotting rivers, and creating a map. The console shows the execution output, including a warning about downloading a map and the resulting plot of sorted dates versus index.

```

1 country <- raster::shapefile("../R_SilverFiles/country_y/country.shp")
2 country <- sp::spTransform(country,
3 sp::plot(country, col="blue"))
4
5 # draw rivers depending on (strahler
6 riverlines <- function(w1=2.5, w2=2, w3=1)
7 {
8   lines(river[river@data@rhein=1,], col="blue", lwd=w1)
9   lines
10   lines
11   lines
12 }
13
14 library(leaflet) # pointsMap, projectPoints
15 map <- pointsMap(yy.xx, statlocs@data, zoom=7, fx=0.3, type="maptoolkit-to
16 statlocspp <- projectPoints(yy.xx, statlocs@data, to=proj4())
17 # map2 <- pointsMap(yy.xx, statlocs@data, map=map, proj=proj4())
18
19 first <- sapply(d, function(x) as.character(x$date[1]))
20 plot(sort(as.Date(first)))
21 range(as.Date(first))
22 rm(first)
23 last <- sapply(d, function(x) as.character(x$date[1]))
24
25 2. Read shapefiles :

```

Console output:

```

139 11022200 NIEDALTDORF 6.592912 49.34216 4073457 2920089
140 1301 BAD ROTENFELS 8.296870 48.81876 4195899 2857609
141 13322200 LEBACH 6.906131 49.41035 4096511 2926661
142 1409 SUESSEN 9.752795 48.68094 4302790 2840873
> map <- pointsMap(yy.xx, statlocs@data, zoom=7, fx=0.3, type="maptoolkit-topo"
)
Downloading map with extend 4.594204, 13.009235, 45.982782, 53.363084 ...
Done. Now plotting...
> statlocspp <- projectPoints(yy.xx, statlocs@data, to=proj4())
> first <- sapply(d, function(x) as.character(x$date[1]))
> plot(sort(as.Date(first)))
> range(as.Date(first))
[1] "1823-10-31" "1972-11-01"
> plot(sort(as.Date(first)))
> range(as.Date(first))
[1] "1823-10-31" "1972-11-01"
>

```

The plot in the bottom right pane shows 'sort(as.Date(first))' on the y-axis (ranging from 1850 to 1950) and 'Index' on the x-axis (ranging from 0 to 150). The plot displays a series of points forming a curve that rises from approximately 1850 to 1950 over the index range.

# RStudio configuration

keyboard shortcuts (ALT+SHIFT+K)

## RStudio configuration

### keyboard shortcuts (ALT+SHIFT+K)

Recommended settings for reproducible research under

#### Tools - Global Options - General

**ON**: Restore previously open source documents at startup

**OFF**: Restore .Rdata into workspace at startup

Save workspace to .RData on exit: **NEVER**

## RStudio configuration

### keyboard shortcuts (ALT+SHIFT+K)

Recommended settings for reproducible research under

#### Tools - Global Options - General

**ON**: Restore previously open source documents at startup

**OFF**: Restore .Rdata into workspace at startup

Save workspace to .RData on exit: **NEVER**

*Instead use `save(object, file="object.Rdata")` after long computations. You can load them later with `load("object.Rdata")`.*



## RStudio configuration

### keyboard shortcuts (ALT+SHIFT+K)

Recommended settings for reproducible research under

#### Tools - Global Options - General

**ON:** Restore previously open source documents at startup

**OFF:** Restore .Rdata into workspace at startup

Save workspace to .RData on exit: **NEVER**

*Instead use `save(object, file="object.Rdata")` after long computations. You can load them later with `load("object.Rdata")`.*

#### Tools - Global Options - Code - Display

**ON:** Show margin (Margin column:80) *People hate horizontal scrolling!*

#### Tools - Global Options - Code - Saving

Line ending conversion: **Windows (CR/LF)**

## Assignments

- objects: assignment with `<-`  
`nstudents <- 15`  
`nstudents`  
`nstudents > 12`

## Assignments

- ▶ objects: assignment with `<-`  
`nstudents <- 15`  
`nstudents`  
`nstudents > 12`
- ▶ Rstudio Keyboard shortcut: `ALT + -`

## Assignments

- ▶ objects: assignment with `<-`  
`nstudents <- 15`  
`nstudents`  
`nstudents > 12`
- ▶ Rstudio Keyboard shortcut: `ALT + -`
- ▶ What's a good object name?

## Assignments

- ▶ objects: assignment with `<-`  
`nstudents <- 15`  
`nstudents`  
`nstudents > 12`
- ▶ Rstudio Keyboard shortcut: `ALT + -`
- ▶ What's a good object name? → short, but explanatory,

## Assignments

- ▶ objects: assignment with `<-`  
`nstudents <- 15`  
`nstudents`  
`nstudents > 12`
- ▶ Rstudio Keyboard shortcut: `ALT + -`
- ▶ What's a good object name? → short, but explanatory,  
lowerCamelStandard.or.dot\_or\_underscore are good naming conventions

## Assignments

- ▶ objects: assignment with `<-`  
`nstudents <- 15`  
`nstudents`  
`nstudents > 12`
- ▶ Rstudio Keyboard shortcut: `ALT + -`
- ▶ What's a good object name? → short, but explanatory,  
lowerCamelStandard.or.dot\_or\_underscore are good naming conventions
- ▶ comments: `# everything after a hashtag is not executed.`

## Exercise

- ▶ Open Rstudio, start new script. Write comments about what you do, save the file in a useful place.
- ▶ Calculate  $21+21$  ,  $7*6$  and  $\frac{0,3}{4} * \sqrt{313600}$
- ▶ Is  $0.5 - 0.2$  equal to  $0.3$ ? Is  $0.4 - 0.1$  equal to  $0.3$ ?
- ▶ With the `c` command, create a vector with body sizes of people around you. You can also use the values 1.75, 1.76, 1.83, 1.84, 1.77, 1.76, 1.77, 1.66, 1.86, 1.76
- ▶ What does `3:6` create? What does `YourObject[3:6]` do?
- ▶ What does `YourObject[-4]` do?
- ▶ BONUS (for fast people): Analyze the descriptive statistics: `mean(YourObject)`, `median`, `min`, `max`, `range`, `quantile`
- ▶ BONUS 2: Generate 150 random numbers from a normal distribution with  $\mu = 170cm$  and  $\sigma = 8cm$ . Perform a Kolmogorov-Smirnov test for normality of that sample.



## Reading files

- ▶ Copy the file `treesize.txt` (from [bit.ly/swc\\_tree](https://bit.ly/swc_tree))

## Reading files

- ▶ Copy the file `treesize.txt` (from [bit.ly/swc\\_tree](https://bit.ly/swc_tree))
- ▶ Tell R where to look for it with: `setwd("C:/path/to/input")`  
# change back- to forwardslashes

## Reading files

- ▶ Copy the file `treesize.txt` (from [bit.ly/swc\\_tree](https://bit.ly/swc_tree))
- ▶ Tell R where to look for it with: `setwd("C:/path/to/input")`  
# change back- to forwardslashes
- ▶ Read the file into R with the command `read.table`.

## Reading files

- ▶ Copy the file `treesize.txt` (from [bit.ly/swc\\_tree](https://bit.ly/swc_tree))
- ▶ Tell R where to look for it with: `setwd("C:/path/to/input")`  
# change back- to forwardslashes
- ▶ Read the file into R with the command `read.table`.
- ▶ If R tells you "no such file" exists, check the output of `dir()`.

## Reading files

- ▶ Copy the file `treesize.txt` (from [bit.ly/swc\\_tree](https://bit.ly/swc_tree))
- ▶ Tell R where to look for it with: `setwd("C:/path/to/input")`  
# change back- to forwardslashes
- ▶ Read the file into R with the command `read.table`.
- ▶ If R tells you "no such file" exists, check the output of `dir()`.
- ▶ Use the documentation to find out the correct settings of the arguments: `help(read.table)`, `?read.table`, or press F1.

## Reading files

- ▶ Copy the file `treesize.txt` (from [bit.ly/swc\\_tree](https://bit.ly/swc_tree))
- ▶ Tell R where to look for it with: `setwd("C:/path/to/input")`  
`# change back- to forwardslashes`
- ▶ Read the file into R with the command `read.table`.
- ▶ If R tells you "no such file" exists, check the output of `dir()`.
- ▶ Use the documentation to find out the correct settings of the arguments: `help(read.table)`, `?read.table`, or press F1.
- ▶ `str(YourObject)` must yield the column data types: `num`, `num`, `factor`.

## Reading files

- ▶ Copy the file `treesize.txt` (from [bit.ly/swc\\_tree](https://bit.ly/swc_tree))
- ▶ Tell R where to look for it with: `setwd("C:/path/to/input")`  
# change back- to forwardslashes
- ▶ Read the file into R with the command `read.table`.
- ▶ If R tells you "no such file" exists, check the output of `dir()`.
- ▶ Use the documentation to find out the correct settings of the arguments: `help(read.table)`, `?read.table`, or press F1.
- ▶ `str(YourObject)` must yield the column data types: num, num, factor.
- ▶ You need to set the argument header.

## Reading files

- ▶ Copy the file `treesize.txt` (from [bit.ly/swc\\_tree](https://bit.ly/swc_tree))
- ▶ Tell R where to look for it with: `setwd("C:/path/to/input")`  
# change back- to forwardslashes
- ▶ Read the file into R with the command `read.table`.
- ▶ If R tells you "no such file" exists, check the output of `dir()`.
- ▶ Use the documentation to find out the correct settings of the arguments: `help(read.table)`, `?read.table`, or press F1.
- ▶ `str(YourObject)` must yield the column data types: num, num, factor.
- ▶ You need to set the argument header.

```
treesize <- read.table(file="treesize.txt", header=TRUE)
```



# Objects

- ▶ Check the objects in your workspace with `ls()`.

# Objects

- ▶ Check the objects in your workspace with `ls()`.
- ▶ Remove objects with `rm(YourObject, AnotherOne)`

# Objects

- ▶ Check the objects in your workspace with `ls()`.
- ▶ Remove objects with `rm(YourObject, AnotherOne)`
- ▶ Remove all objects with `rm(list=ls() )`

## Objects

- ▶ Check the objects in your workspace with `ls()`.
- ▶ Remove objects with `rm(YourObject, AnotherOne)`
- ▶ Remove all objects with `rm(list=ls() )`
- ▶ Or just the Rstudio button

## Objects

- ▶ Check the objects in your workspace with `ls()`.
- ▶ Remove objects with `rm(YourObject, AnotherOne)`
- ▶ Remove all objects with `rm(list=ls() )`
- ▶ Or just the Rstudio button
- ▶ To make sure your script is reproducible (you may rename objects, for example, and miss one occurrence):  
restart R (**CTRL** + **SHIFT** + **F10**) every once in a while (Make sure Rstudio settings are reproducible as shown on slide 4).

## Overview: data types

In order of coercion (if mixed, TRUE is converted to 1, 3.14 to "3.14" etc)

Description	example	typeof	class
empty set	NULL	NULL	NULL
not available	NA	logical	logical
logical	c(T, F, FALSE, TRUE)	logical	logical
category	factor("left")	integer	<b>factor</b>
integer number	4:6	integer	integer
decimal	8.7	double	<b>numeric</b>
complex	5+3i	complex	complex
character string	"homer rocks"	character	character
time	Sys.time()	double	<b>POSIXct</b>
date	as.Date("2017-05-02")	double	<b>Date</b>
function	ncol	closure	<b>function</b>

## Overview: data types

In order of coercion (if mixed, TRUE is converted to 1, 3.14 to "3.14" etc)

Description	example	<code>typeof</code>	<code>class</code>
empty set	NULL	NULL	NULL
not available	NA	logical	logical
logical	<code>c(T, F, FALSE, TRUE)</code>	logical	logical
category	<code>factor("left")</code>	integer	<b>factor</b>
integer number	4:6	integer	integer
decimal	8.7	double	<b>numeric</b>
complex	5+3i	complex	complex
character string	"homer rocks"	character	character
time	<code>Sys.time()</code>	double	<b>POSIXct</b>
date	<code>as.Date("2017-05-02")</code>	double	<b>Date</b>
function	<code>ncol</code>	closure	<b>function</b>

[adv-r.had.co.nz/Data-structures](http://adv-r.had.co.nz/Data-structures).

## Overview: data types

In order of coercion (if mixed, TRUE is converted to 1, 3.14 to "3.14" etc)

Description	example	<code>typeof</code>	<code>class</code>
empty set	NULL	NULL	NULL
not available	NA	logical	logical
logical	<code>c(T, F, FALSE, TRUE)</code>	logical	logical
category	<code>factor("left")</code>	integer	<b>factor</b>
integer number	4:6	integer	integer
decimal	8.7	double	<b>numeric</b>
complex	5+3i	complex	complex
character string	"homer rocks"	character	character
time	<code>Sys.time()</code>	double	<b>POSIXct</b>
date	<code>as.Date("2017-05-02")</code>	double	<b>Date</b>
function	<code>ncol</code>	closure	<b>function</b>

[adv-r.had.co.nz/Data-structures](http://adv-r.had.co.nz/Data-structures). `as.character(3.14)` converts a data type;



## Overview: data types

In order of coercion (if mixed, TRUE is converted to 1, 3.14 to "3.14" etc)

Description	example	<code>typeof</code>	<code>class</code>
empty set	NULL	NULL	NULL
not available	NA	logical	logical
logical	<code>c(T, F, FALSE, TRUE)</code>	logical	logical
category	<code>factor("left")</code>	integer	<b>factor</b>
integer number	4:6	integer	integer
decimal	8.7	double	<b>numeric</b>
complex	5+3i	complex	complex
character string	"homer rocks"	character	character
time	<code>Sys.time()</code>	double	<b>POSIXct</b>
date	<code>as.Date("2017-05-02")</code>	double	<b>Date</b>
function	<code>ncol</code>	closure	<b>function</b>

[adv-r.had.co.nz/Data-structures](http://adv-r.had.co.nz/Data-structures). `as.character(3.14)` converts a data type; `is.integer(4:6)` checks.

## Overview: data types

In order of coercion (if mixed, TRUE is converted to 1, 3.14 to "3.14" etc)

Description	example	<code>typeof</code>	<code>class</code>
empty set	NULL	NULL	NULL
not available	NA	logical	logical
logical	<code>c(T, F, FALSE, TRUE)</code>	logical	logical
category	<code>factor("left")</code>	integer	<b>factor</b>
integer number	4:6	integer	integer
decimal	8.7	double	<b>numeric</b>
complex	5+3i	complex	complex
character string	"homer rocks"	character	character
time	<code>Sys.time()</code>	double	<b>POSIXct</b>
date	<code>as.Date("2017-05-02")</code>	double	<b>Date</b>
function	<code>ncol</code>	closure	<b>function</b>

[adv-r.had.co.nz/Data-structures](http://adv-r.had.co.nz/Data-structures). `as.character(3.14)` converts a data type; `is.integer(4:6)` checks. `str` shows an abbreviation of `class`.

## Overview: data types

In order of coercion (if mixed, TRUE is converted to 1, 3.14 to "3.14" etc)

Description	example	<code>typeof</code>	<code>class</code>
empty set	NULL	NULL	NULL
not available	NA	logical	logical
logical	<code>c(T, F, FALSE, TRUE)</code>	logical	logical
category	<code>factor("left")</code>	integer	<b>factor</b>
integer number	4:6	integer	integer
decimal	8.7	double	<b>numeric</b>
complex	5+3i	complex	complex
character string	"homer rocks"	character	character
time	<code>Sys.time()</code>	double	<b>POSIXct</b>
date	<code>as.Date("2017-05-02")</code>	double	<b>Date</b>
function	<code>ncol</code>	closure	<b>function</b>

[adv-r.had.co.nz/Data-structures](http://adv-r.had.co.nz/Data-structures). `as.character(3.14)` converts a data type; `is.integer(4:6)` checks. `str` shows an abbreviation of `class`. `mode` (for users) is like `typeof` (R internal), but combines integer and double to numeric (& closure, special and builtin to function).

## Overview: data types

In order of coercion (if mixed, TRUE is converted to 1, 3.14 to "3.14" etc)

Description	example	<code>typeof</code>	<code>class</code>
empty set	NULL	NULL	NULL
not available	NA	logical	logical
logical	<code>c(T, F, FALSE, TRUE)</code>	logical	logical
category	<code>factor("left")</code>	integer	<b>factor</b>
integer number	4:6	integer	integer
decimal	8.7	double	<b>numeric</b>
complex	5+3i	complex	complex
character string	"homer rocks"	character	character
time	<code>Sys.time()</code>	double	<b>POSIXct</b>
date	<code>as.Date("2017-05-02")</code>	double	<b>Date</b>
function	<code>ncol</code>	closure	<b>function</b>

[adv-r.had.co.nz/Data-structures](http://adv-r.had.co.nz/Data-structures). `as.character(3.14)` converts a data type; `is.integer(4:6)` checks. `str` shows an abbreviation of `class`. `mode` (for users) is like `typeof` (R internal), but combines integer and double to numeric (& closure, special and builtin to function). When mixing date/time with others, the order of appearance determines the output class.

## Overview: Object types

Object	example	<code>typeof</code>	<code>class</code>
vector	<i>see data types</i>	...	...
matrix	<code>matrix(9:15, ncol=2)</code>	...	matrix
array	<code>array(letters[1:24], dim=c(2,6,4))</code>	...	array
data.frame	<code>data.frame(C1=4:5, C2=c("a","b"))</code>	list	data.frame
list	<code>list(el1=7:15, el2="big")</code>	list	list
function	<code>function(x) 12+0.5*x</code>	closure	function
...	<code>lm(b ~ a)</code>	list	lm

## Overview: Object types

Object	example	<code>typeof</code>	<code>class</code>
vector	<i>see data types</i>	...	...
matrix	<code>matrix(9:15, ncol=2)</code>	...	matrix
array	<code>array(letters[1:24], dim=c(2,6,4))</code>	...	array
data.frame	<code>data.frame(C1=4:5, C2=c("a","b"))</code>	list	data.frame
list	<code>list(el1=7:15, el2="big")</code>	list	list
function	<code>function(x) 12+0.5*x</code>	closure	function
...	<code>lm(b ~ a)</code>	list	lm

A `matrix` consists of only one data type. If you accidentally change one element to a character, all are converted and calculations are not possible any more (See coercion order in previous slide).

## Overview: Object types

Object	example	<code>typeof</code>	<code>class</code>
vector	<i>see data types</i>	...	...
matrix	<code>matrix(9:15, ncol=2)</code>	...	matrix
array	<code>array(letters[1:24], dim=c(2,6,4))</code>	...	array
data.frame	<code>data.frame(C1=4:5, C2=c("a","b"))</code>	list	data.frame
list	<code>list(el1=7:15, el2="big")</code>	list	list
function	<code>function(x) 12+0.5*x</code>	closure	function
...	<code>lm(b ~ a)</code>	list	lm

A **matrix** consists of only one data type. If you accidentally change one element to a character, all are converted and calculations are not possible any more (See coercion order in previous slide).

**data.frames** can have multiple data types, but a column in itself also has only one type.

## Overview: Object types

Object	example	typeof	class
vector	<i>see data types</i>	...	...
matrix	<code>matrix(9:15, ncol=2)</code>	...	matrix
array	<code>array(letters[1:24], dim=c(2,6,4))</code>	...	array
data.frame	<code>data.frame(C1=4:5, C2=c("a","b"))</code>	list	data.frame
list	<code>list(el1=7:15, el2="big")</code>	list	list
function	<code>function(x) 12+0.5*x</code>	closure	function
...	<code>lm(b ~ a)</code>	list	lm

A **matrix** consists of only one data type. If you accidentally change one element to a character, all are converted and calculations are not possible any more (See coercion order in previous slide).

**data.frames** can have multiple data types, but a column in itself also has only one type.

**lists** can combine anything, even other lists.



## Overview: Object types

Object	example	typeof	class
vector	<i>see data types</i>	...	...
matrix	<code>matrix(9:15, ncol=2)</code>	...	matrix
array	<code>array(letters[1:24], dim=c(2,6,4))</code>	...	array
data.frame	<code>data.frame(C1=4:5, C2=c("a","b"))</code>	list	data.frame
list	<code>list(el1=7:15, el2="big")</code>	list	list
function	<code>function(x) 12+0.5*x</code>	closure	function
...	<code>lm(b ~ a)</code>	list	lm

A **matrix** consists of only one data type. If you accidentally change one element to a character, all are converted and calculations are not possible any more (See coercion order in previous slide).

**data.frames** can have multiple data types, but a column in itself also has only one type.

**lists** can combine anything, even other lists.

`is.atomic(object)` returns TRUE (vector, matrix, array) or FALSE

## Overview: Object types

Object	example	typeof	class
vector	<i>see data types</i>	...	...
matrix	<code>matrix(9:15, ncol=2)</code>	...	matrix
array	<code>array(letters[1:24], dim=c(2,6,4))</code>	...	array
data.frame	<code>data.frame(C1=4:5, C2=c("a","b"))</code>	list	data.frame
list	<code>list(el1=7:15, el2="big")</code>	list	list
function	<code>function(x) 12+0.5*x</code>	closure	function
...	<code>lm(b ~ a)</code>	list	lm

A **matrix** consists of only one data type. If you accidentally change one element to a character, all are converted and calculations are not possible any more (See coercion order in previous slide).

**data.frames** can have multiple data types, but a column in itself also has only one type.

**lists** can combine anything, even other lists.

`is.atomic(Object)` returns TRUE (vector, matrix, array) or FALSE

`as.matrix(Object)` converts the class of an object by force.

## R Packages

- ▶ Many people write code for specific tasks and publish it on CRAN, the Comprehensive R Archive Network

## R Packages

- ▶ Many people write code for specific tasks and publish it on CRAN, the Comprehensive R Archive Network
- ▶ Packages for a range of topics: [cran.r-project.org/web/views](https://cran.r-project.org/web/views)

## R Packages

- ▶ Many people write code for specific tasks and publish it on CRAN, the Comprehensive R Archive Network
- ▶ Packages for a range of topics: [cran.r-project.org/web/views](https://cran.r-project.org/web/views)
- ▶ All >10'500 available packages: [cran.r-project.org/web/packages](https://cran.r-project.org/web/packages)

## R Packages

- ▶ Many people write code for specific tasks and publish it on CRAN, the Comprehensive R Archive Network
- ▶ Packages for a range of topics: [cran.r-project.org/web/views](https://cran.r-project.org/web/views)
- ▶ All >10'500 available packages: [cran.r-project.org/web/packages](https://cran.r-project.org/web/packages)
- ▶ `install.packages("ggplot2")` to download and install.  
(only needs to be executed once, works on user level, no admin rights required)  
You can do this in Rstudio

## R Packages

- ▶ Many people write code for specific tasks and publish it on CRAN, the Comprehensive R Archive Network
- ▶ Packages for a range of topics: [cran.r-project.org/web/views](https://cran.r-project.org/web/views)
- ▶ All >10'500 available packages: [cran.r-project.org/web/packages](https://cran.r-project.org/web/packages)
- ▶ `install.packages("ggplot2")` to download and install.  
(only needs to be executed once, works on user level, no admin rights required)  
You can do this in Rstudio
- ▶ `library("ggplot2")` to load it  
(needed in every new R session) Put this in the script for reproducibility

## R Packages

- ▶ Many people write code for specific tasks and publish it on CRAN, the Comprehensive R Archive Network
- ▶ Packages for a range of topics: [cran.r-project.org/web/views](https://cran.r-project.org/web/views)
- ▶ All >10'500 available packages: [cran.r-project.org/web/packages](https://cran.r-project.org/web/packages)
- ▶ `install.packages("ggplot2")` to download and install.  
(only needs to be executed once, works on user level, no admin rights required)  
You can do this in Rstudio
- ▶ `library("ggplot2")` to load it  
(needed in every new R session) Put this in the script for reproducibility
- ▶ Better to use the `package::function` syntax



## R Packages

- ▶ Many people write code for specific tasks and publish it on CRAN, the Comprehensive R Archive Network
- ▶ Packages for a range of topics: [cran.r-project.org/web/views](https://cran.r-project.org/web/views)
- ▶ All >10'500 available packages: [cran.r-project.org/web/packages](https://cran.r-project.org/web/packages)
- ▶ `install.packages("ggplot2")` to download and install.  
(only needs to be executed once, works on user level, no admin rights required)  
You can do this in Rstudio
- ▶ `library("ggplot2")` to load it  
(needed in every new R session) Put this in the script for reproducibility
- ▶ Better to use the `package::function` syntax
- ▶ Regularly run `update.packages()` or use the Rstudio button

## R Packages

- ▶ Many people write code for specific tasks and publish it on CRAN, the Comprehensive R Archive Network
- ▶ Packages for a range of topics: [cran.r-project.org/web/views](https://cran.r-project.org/web/views)
- ▶ All >10'500 available packages: [cran.r-project.org/web/packages](https://cran.r-project.org/web/packages)
- ▶ `install.packages("ggplot2")` to download and install.  
(only needs to be executed once, works on user level, no admin rights required)  
You can do this in Rstudio
- ▶ `library("ggplot2")` to load it  
(needed in every new R session) Put this in the script for reproducibility
- ▶ Better to use the `package::function` syntax
- ▶ Regularly run `update.packages()` or use the Rstudio button
- ▶ Rarely needed: `remove.packages("packagename")`

# Linear Regression

- ▶ Install and load the package `berryFunctions`

## Linear Regression

- ▶ Install and load the package `berryFunctions`
- ▶ How can we pass the `treesize` data to `?linReg` with a formula?

## Linear Regression

- ▶ Install and load the package `berryFunctions`
- ▶ How can we pass the `treesize` data to `?linReg` with a formula?
- ▶ Describe the resulting graph (height vs age).

## Linear Regression

- ▶ Install and load the package `berryFunctions`
- ▶ How can we pass the treesize data to `?linReg` with a formula?
- ▶ Describe the resulting graph (height vs age).
- ▶ Look into the source code of `linReg`. What is actually the backbone for the calculation of the function?

## Linear Regression

- ▶ Install and load the package `berryFunctions`
- ▶ How can we pass the `treesize` data to `?linReg` with a formula?
- ▶ Describe the resulting graph (height vs age).
- ▶ Look into the source code of `linReg`. What is actually the backbone for the calculation of the function?
- ▶ Feed the data into `lm`, assign the output to an object (useful name!).

## Linear Regression

- ▶ Install and load the package `berryFunctions`
- ▶ How can we pass the `treesize` data to `?linReg` with a formula?
- ▶ Describe the resulting graph (height vs age).
- ▶ Look into the source code of `linReg`. What is actually the backbone for the calculation of the function?
- ▶ Feed the data into `lm`, assign the output to an object (useful name!).
- ▶ Briefly explain the `summary` of the linear model.



## More things

- ▶ Connect Rstudio to github

## More things

- ▶ Connect Rstudio to github
- ▶ Plotting

## Objects: data.frames

- ▶ For tables with different data types (numbers, characters, categories, integers), R has the object type `data.frame`:

```
data.frame(count=c(2,6,5), type=c("a","k","k"))
```

- ▶ `read.table` also returns a `data.frame`
- ▶ If we have the object `df`, we can subset with `df[rows,columns]`
- ▶ `df[1,2:4]; df[2, ]; df[, "name"]; df$name`
- ▶ Logical values: `vect[c(TRUE,TRUE,FALSE,FALSE,TRUE,FALSE)]`

## Objects: data.frames

- ▶ For tables with different data types (numbers, characters, categories, integers), R has the object type `data.frame`:  
`data.frame(count=c(2,6,5), type=c("a","k","k"))`
- ▶ `read.table` also returns a `data.frame`
- ▶ If we have the object `df`, we can subset with `df[rows,columns]`
- ▶ `df[1,2:4]; df[2, ]; df[, "name"]; df$name`
- ▶ Logical values: `vect[c(TRUE,TRUE,FALSE,FALSE,TRUE,FALSE)]`

From the dataset `treecsize` from the previous exercise, obtain:

- ▶ The first 5 values in column 2
- ▶ The maximum "Height" (the maximum of the values in that column)
- ▶ For each entry: is the measurement equal to (`==`) A?
- ▶ BONUS 1: The height entries for trees older than 23.5 years
- ▶ BONUS 2: All rows, excluding rows 3, 7,8,9,...,20