Introduction to R, Rstudio & Project Management

Berry Boessenkool, uni-potsdam.de, May 2017

berry-b@gmx.de
github.com/brry

swc-bb.github.io/2017-05-17-r-workshop

Presentation template generated with `berryFunctions::createPres`

Introduction to R, Rstudio & Project Management

📷 ENCOURAGED

Berry Boessenkool, uni-potsdam.de, May 2017

berry-b@gmx.de
github.com/brry

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, May 2017

berry-b@gmx.de
github.com/brry

swc-bb.github.io/2017-05-17-r-workshop

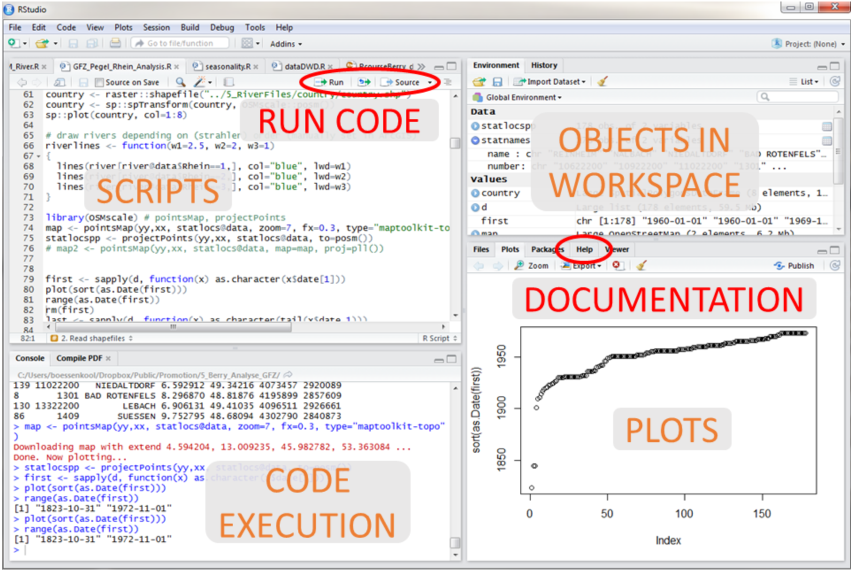
Presentation template generated with `berryFunctions::createPres`

Survey

knowledge survey to determine focus for this session

bit.ly/knowR

RStudio



```r
country <- raster::shapefile("../5_RiverFiles/country/country.shp")
country <- sp::spTransform(country, OSMscale::posm())
sp::plot(country, col=1:8)

# draw rivers depending on (strahler) order (manually set in ArcGIS)
riverlines <- function(w1=2.5, w2=2, w3=1)
{
  lines(river[river@data$Rhein==1,], col="blue", lwd=w1)
  lines(river[river@data$Rhein==2,], col="blue", lwd=w2)
  lines(river[river@data$Rhein==3,], col="blue", lwd=w3)
}

library(OSMscale) # pointsMap, projectPoints
map <- pointsMap(yy,xx, statlocs@data, zoom=7, fx=0.3, type="maptoolkit-to
statlocspp <- projectPoints(yy,xx, statlocs@data, to=posm())
# map2 <- pointsMap(yy,xx, statlocs@data, map=map, proj=pll())


first <- sapply(d, function(x) as.character(x$date[1]))
plot(sort(as.Date(first)))
range(as.Date(first))
rm(first)
last <- sapply(d, function(x) as.character(tail(x$date,1)))
```

```
Data
statlocspp     178 obs. of 2 variables
statnames      178 obs. of 2 variables
  name  : chr "REINHEIM" "NALBACH" "NIEDALTDORF" "BAD ROTENFELS"…
  number: chr "10622200" "10922200" "11022200" "1301" ...
Values
country        Large SpatialPolygonsDataFrame (8 elements, 1…
d              Large list (178 elements, 59.5 Mb)
first          chr [1:178] "1960-01-01" "1960-01-01" "1969-1…
map            Large OpenStreetMap (2 elements, 6.2 Mb)
```

```
C:/Users/boessenkool/Dropbox/Public/Promotion/5_Berry_Analyse_GFZ/
139 11022200   NIEDALTDORF 6.592912 49.34216 4073457 2920089
8       1301 BAD ROTENFELS 8.296870 48.81876 4195899 2857609
130 13322200        LEBACH 6.906131 49.41035 4096511 2926661
86      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=posm())
> 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"
> 
```

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 1: Basic R syntax

- 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.

Solutions to Exercise 1: Basic R syntax

```
# simple introductionary tasks
21+21   ;   7*6   ;   0.3/4*sqrt(313600)
0.5-0.2 == 0.3 # TRUE
0.4-0.1 == 0.3 # FALSE
print(0.4-0.1, digits=22) # Numerical accuracy limits
all.equal(0.4-0.1, 0.3) # TRUE

size <- c(1.75, 1.76, 1.83, 1.84, 1.77, 1.76, 1.77,
          1.66, 1.86, 1.76)
3:6 # A vector with consecutive integers
size[3:6] # Select the corresponding elements of a vector
size[-4] # Select all but the fourth value
mean(size); median(size); min(size); max(size)
range(size); quantile(size)
x <- rnorm(n=150, m=170, s=8)
ks.test(x, "pnorm", mean(x), sd(x) )
```

Exercise 2: Reading files

- Copy the file `treesize.txt` (from bit.ly/swc_tree)

Exercise 2: Reading files

- Copy the file `treesize.txt` (from bit.ly/swc_tree)
- Tell R where to look for it with: `setwd("C:/path/to/input")`

  ```
  # change back- to forwardslashes
  ```

Exercise 2: Reading files

- Copy the file `treesize.txt` (from 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`.

Exercise 2: Reading files

- Copy the file `treesize.txt` (from 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()`.

Exercise 2: Reading files

- Copy the file `treesize.txt` (from 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.

Exercise 2: Reading files

- Copy the file `treesize.txt` (from 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`.

Exercise 2: Reading files

- Copy the file `treesize.txt` (from 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`.
- BONUS: What arguments for `read.table` seem useful?

Exercise 2: Reading files

- Copy the file `treesize.txt` (from 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`.
- BONUS: What arguments for `read.table` seem useful?
- BONUS 2: What commands are useful to read csv files, excel sheets or dataset with fix column widths?

Solution to Exercise 2: Reading files

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

header = TRUE	read first line as column names

Solution to Exercise 2: Reading files

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

header = TRUE	read first line as column names
dec = ”,”	comma as decimal mark

Solution to Exercise 2: Reading files

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

header = TRUE	read first line as column names
dec = ”,”	comma as decimal mark
sep = ”_”	underscore as column separator (”\t” for tabstop)

Solution to Exercise 2: Reading files

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

header = TRUE	read first line as column names
dec = ”,”	comma as decimal mark
sep = ”_”	underscore as column separator (”\t” for tabstop)
fill = T	fill incomplete rows with NAs at the end

Solution to Exercise 2: Reading files

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

header = TRUE	read first line as column names
dec = ”,”	comma as decimal mark
sep = ”_”	underscore as column separator (”\t” for tabstop)
fill = T	fill incomplete rows with NAs at the end
skip = 12	ignore the first 12 lines (eg with meta data)

Solution to Exercise 2: Reading files

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

header = TRUE	read first line as column names
dec = ","	comma as decimal mark
sep = "_"	underscore as column separator ("\t" for tabstop)
fill = T	fill incomplete rows with NAs at the end
skip = 12	ignore the first 12 lines (eg with meta data)
comment.char = "%"	omit (rest of) lines that start with % (like R's #)

Solution to Exercise 2: Reading files

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

header = TRUE	read first line as column names
dec = ”,”	comma as decimal mark
sep = ”_”	underscore as column separator (”\t” for tabstop)
fill = T	fill incomplete rows with NAs at the end
skip = 12	ignore the first 12 lines (eg with meta data)
comment.char = ”%”	omit (rest of) lines that start with % (like R's #)
na.strings = c(-999, ”NN”)	identify NA entries (missing values)

Solution to Exercise 2: Reading files

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

header = TRUE	read first line as column names
dec = ","	comma as decimal mark
sep = "_"	underscore as column separator ("\t" for tabstop)
fill = T	fill incomplete rows with NAs at the end
skip = 12	ignore the first 12 lines (eg with meta data)
comment.char = "%"	omit (rest of) lines that start with % (like R's #)
na.strings = c(-999, "NN")	identify NA entries (missing values)
stringsAsFactors=FALSE	do not convert characters to factors

Solution to Exercise 2: Reading files

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

header = TRUE	read first line as column names
dec = ","	comma as decimal mark
sep = "_"	underscore as column separator ("\t" for tabstop)
fill = T	fill incomplete rows with NAs at the end
skip = 12	ignore the first 12 lines (eg with meta data)
comment.char = "%"	omit (rest of) lines that start with % (like R's #)
na.strings = c(-999, "NN")	identify NA entries (missing values)
stringsAsFactors=FALSE	do not convert characters to factors
as.is=TRUE	the same, but less typing, and potentially columnwise

Solution to Exercise 2: Reading files

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

header = TRUE	read first line as column names
dec = ","	comma as decimal mark
sep = "_"	underscore as column separator ("\t" for tabstop)
fill = T	fill incomplete rows with NAs at the end
skip = 12	ignore the first 12 lines (eg with meta data)
comment.char = "%"	omit (rest of) lines that start with % (like R's #)
na.strings = c(-999, "NN")	identify NA entries (missing values)
stringsAsFactors=FALSE	do not convert characters to factors
as.is=TRUE	the same, but less typing, and potentially columnwise

Alternatives to read.table:

Solution to Exercise 2: Reading files

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

header = TRUE	read first line as column names
dec = ”,”	comma as decimal mark
sep = ”_”	underscore as column separator (”\t” for tabstop)
fill = T	fill incomplete rows with NAs at the end
skip = 12	ignore the first 12 lines (eg with meta data)
comment.char = ”%”	omit (rest of) lines that start with % (like R’s #)
na.strings = c(-999, ”NN”)	identify NA entries (missing values)
stringsAsFactors=FALSE	do not convert characters to factors
as.is=TRUE	the same, but less typing, and potentially columnwise

Alternatives to read.table:

scan At the core of read.table - for complicated things

Solution to Exercise 2: Reading files

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

header = TRUE	read first line as column names
dec = ","	comma as decimal mark
sep = "_"	underscore as column separator ("\t" for tabstop)
fill = T	fill incomplete rows with NAs at the end
skip = 12	ignore the first 12 lines (eg with meta data)
comment.char = "%"	omit (rest of) lines that start with % (like R's #)
na.strings = c(-999, "NN")	identify NA entries (missing values)
stringsAsFactors=FALSE	do not convert characters to factors
as.is=TRUE	the same, but less typing, and potentially columnwise

Alternatives to read.table:

scan	At the core of read.table - for complicated things
read.csv	comma separated values (different defaults than read.table)

Solution to Exercise 2: Reading files

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

header = TRUE	read first line as column names
dec = ”,”	comma as decimal mark
sep = ”_”	underscore as column separator (”\t” for tabstop)
fill = T	fill incomplete rows with NAs at the end
skip = 12	ignore the first 12 lines (eg with meta data)
comment.char = ”%”	omit (rest of) lines that start with % (like R’s #)
na.strings = c(-999, ”NN”)	identify NA entries (missing values)
stringsAsFactors=FALSE	do not convert characters to factors
as.is=TRUE	the same, but less typing, and potentially columnwise

Alternatives to read.table:

scan	At the core of read.table - for complicated things
read.csv	comma separated values (different defaults than read.table)
read.fwf	fixed width formatted data

Solution to Exercise 2: Reading files

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

header = TRUE	read first line as column names
dec = ","	comma as decimal mark
sep = "_"	underscore as column separator ("\t" for tabstop)
fill = T	fill incomplete rows with NAs at the end
skip = 12	ignore the first 12 lines (eg with meta data)
comment.char = "%"	omit (rest of) lines that start with % (like R's #)
na.strings = c(-999, "NN")	identify NA entries (missing values)
stringsAsFactors=FALSE	do not convert characters to factors
as.is=TRUE	the same, but less typing, and potentially columnwise

Alternatives to read.table:

scan	At the core of read.table - for complicated things
read.csv	comma separated values (different defaults than read.table)
read.fwf	fixed width formatted data
read_excel	Excel files (install package, see github.com/hadley/readxl)

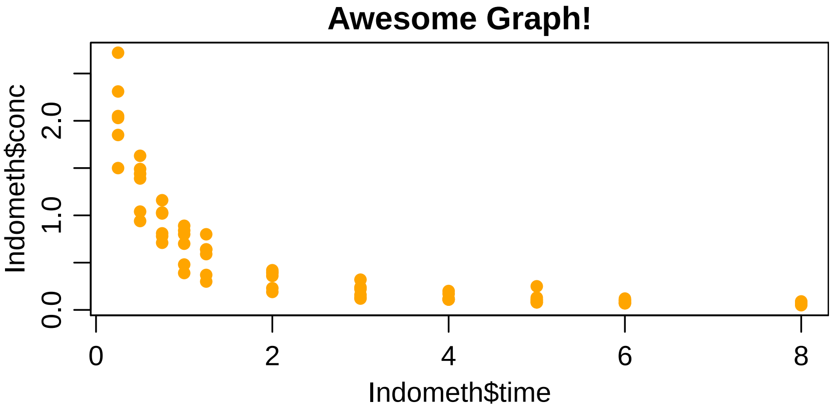
Plotting I

General code for scatterplots: `plot(x, y, ...)`

Plotting I

General code for scatterplots: `plot(x, y, ...)`

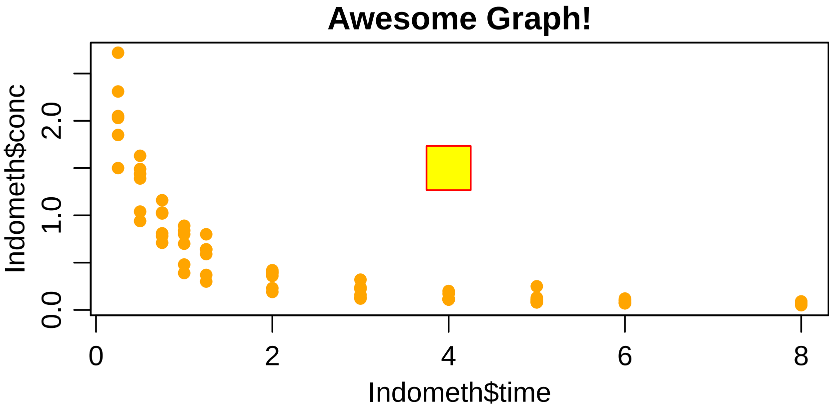
```
plot(x=Indometh$time, y=Indometh$conc,
     col="orange", pch=16, main="Awesome Graph!")
```



Plotting II

General code for scatterplots: `plot(x, y, ...)`

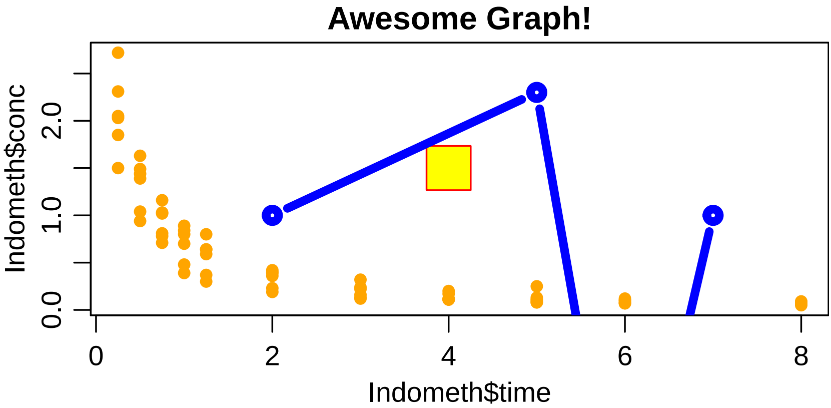
```
points(4, 1.5, pch=22, bg="yellow", cex=4, col="red")
# PointCHaracter, BackGround, Character EXpansion
```



Plotting III

General code for scatterplots: `plot(x, y, ...)`

```
lines(x=c(2,5,6,7), y=c(1,2.3,-3,1),
      col=4, type="b", lwd=5)
```



Exercise 3: Plotting the treesize dataset

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

Exercise 3: Plotting the treesize dataset

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

```
plot(x=xvalues, y=y_values, xlab="nice axis label",
     main="graph title", las=1)
```

Exercise 3: Plotting the treesize dataset

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

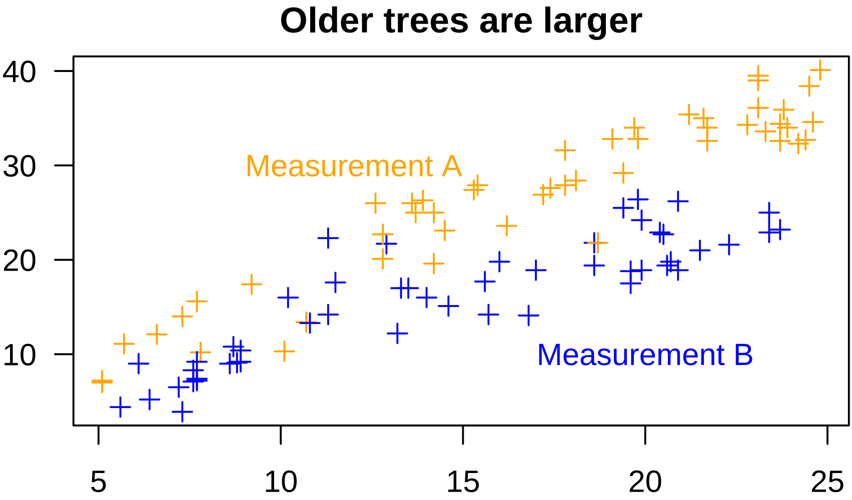
```
plot(x=xvalues, y=y_values, xlab="nice axis label",
     main="graph title", las=1)
```

- Plot tree height over age.
- Add labels to the plot.
- Change the point character (`pch`) and color (`col`).
- BONUS 1: Use a vector for colors, e.g. subset by tree measurement
- BONUS 2: Compare the histogram (`hist`) of the heights with the `boxplot` and `quantile(x, probs=c(0.1, 0.8))`.

Solution to Exercise 3: Plotting the treesize dataset I

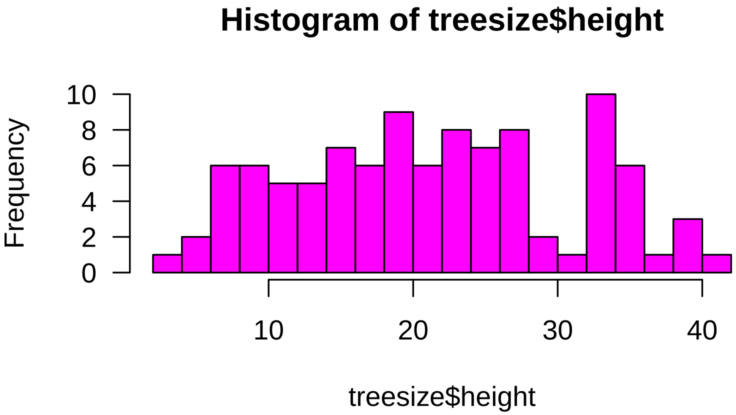
```
treesize <- read.table(file="treesize.txt", header=TRUE)

plot(treesize$age, treesize$height)
cols <- c("orange", "blue")
plot(treesize$age, treesize$height, las=1, ylab="Tree height [m]",
     xlab="Tree age [years]", col=cols[treesize$measurement],
     main="Older trees are larger", pch=3)
text(x=c(12,20), y=c(30,10),
     labels=paste("Measurement", levels(treesize$measurement)), col=cols)
quantile(treesize$height, probs=c(0.1, 0.8))

##   10%   80%
##  8.93 32.36
```

Solution to Exercise 3: Plotting the treesize dataset II



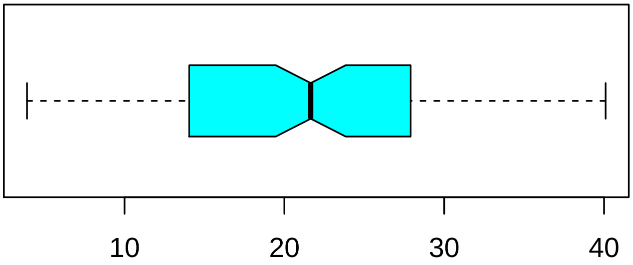
Solution to Exercise 3: Plotting the treesize dataset III

```
hist(treesize$height, col=6, breaks=20, las=1)
```



Solution to Exercise 3: Plotting the treesize dataset IV

```
boxplot(treesize$height, col=5, horizontal=TRUE, notch=TRUE)
```



Commonly needed plot arguments

```
plot(x, y, # point coordinates
col="lightblue", # point color
pch=0, # point character (symbol)
xlab="My label  [km]", ylab="", # axis labels
main="Graph title", # title
cex=1.8, # character expansion (symbol size)
type="l", # draw lines instead of points
lwd=3, # line width (thickness of lines)
las=1, # label axis type (axis numbers upright)
xaxt="n" # axis type (none to suppress axis)
)
```

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	**factor**
integer number	4:6	integer	integer
decimal	8.7	double	**numeric**
complex	5+3i	complex	complex
character string	"homer rocks"	character	character
time	Sys.time()	double	**POSIXct**
date	as.Date("2017-05-02")	double	**Date**
function	ncol	closure	**function**

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	**factor**
integer number	4:6	integer	integer
decimal	8.7	double	**numeric**
complex	5+3i	complex	complex
character string	"homer rocks"	character	character
time	Sys.time()	double	**POSIXct**
date	as.Date("2017-05-02")	double	**Date**
function	ncol	closure	**function**

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

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

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

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

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

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

Overview: Object types

Object	example	`typeof`	`class`
vector	*see data types*	...	...
matrix	matrix(9:15, ncol=2)	...	matrix
array	array(letters[1:24], dim=c(2,6,4))	...	array
data.frame	data.frame(C1=4:5, C2=c("a","b"))	list	data.frame
list	list(el1=7:15, el2="big")	list	list
function	function(x) 12+0.5*x	closure	function
...	lm(b ∼ a)	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	`typeof`	`class`
vector	*see data types*	...	...
matrix	matrix(9:15, ncol=2)	...	matrix
array	array(letters[1:24], dim=c(2,6,4))	...	array
data.frame	data.frame(C1=4:5, C2=c("a","b"))	list	data.frame
list	list(el1=7:15, el2="big")	list	list
function	function(x) 12+0.5*x	closure	function
...	lm(b ~ a)	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.frame`s can have multiple data types, but a column in itself also has only one type.

Overview: Object types

Object	example	`typeof`	`class`
vector	*see data types*	...	...
matrix	matrix(9:15, ncol=2)	...	matrix
array	array(letters[1:24], dim=c(2,6,4))	...	array
data.frame	data.frame(C1=4:5, C2=c("a","b"))	list	data.frame
list	list(el1=7:15, el2="big")	list	list
function	function(x) 12+0.5*x	closure	function
...	lm(b ~ a)	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.frame`s can have multiple data types, but a column in itself also has only one type.

`list`s can combine anything, even other lists.

Overview: Object types

Object	example	`typeof`	`class`
vector	*see data types*	...	...
matrix	matrix(9:15, ncol=2)	...	matrix
array	array(letters[1:24], dim=c(2,6,4))	...	array
data.frame	data.frame(C1=4:5, C2=c(”a”,”b”))	list	data.frame
list	list(el1=7:15, el2=”big”)	list	list
function	function(x) 12+0.5*x	closure	function
...	lm(b ∼ a)	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.frame`s can have multiple data types, but a column in itself also has only one type.
`list`s can combine anything, even other lists.
`is.atomic(Object)` returns TRUE (vector, matrix, array) or FALSE

Overview: Object types

Object	example	`typeof`	`class`
vector	*see data types*	...	...
matrix	matrix(9:15, ncol=2)	...	matrix
array	array(letters[1:24], dim=c(2,6,4))	...	array
data.frame	data.frame(C1=4:5, C2=c(”a”,”b”))	list	data.frame
list	list(el1=7:15, el2=”big”)	list	list
function	function(x) 12+0.5*x	closure	function
...	lm(b ∼ a)	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.frame`s can have multiple data types, but a column in itself also has only one type.

`list`s 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

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
- All $>$10'500 available packages: 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
- All >10'500 available packages: 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
- All >10'500 available packages: 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
- All >10'500 available packages: 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
- All >10'500 available packages: 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
- All >10'500 available packages: 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")`

Exercise 4: Linear regression

- Install and load the package `berryFunctions`

Exercise 4: Linear regression

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

Exercise 4: 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).

Exercise 4: 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?

Exercise 4: 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!).

Exercise 4: Linear regression

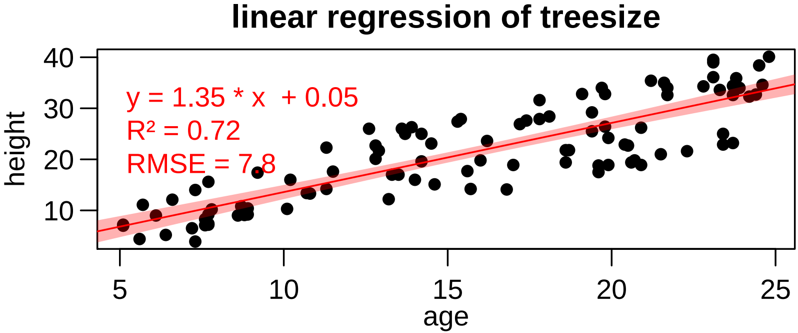
- Install and load the package `berryFunctions`
- How can we pass the treesize data to `?linReg` with a formula?
- Describe the resulting graph (height vs age).
- Look into the source code of `linReg`. What is actually the backbone for the calculation of the function?
- Feed the data into `lm`, assign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

Solution to Exercise 4: Linear regression

```
library("berryFunctions")
linReg(height~age, data=treesize)
linReg # nicer:  berryFunctions::funSource(linReg)
browseURL("https://github.com/brry/berryFunctions") #  R/linReg.R  -> lm
linear_model <- lm(height~age, data=treesize)
summary(linear_model)
```

blog.yhathq.com/posts/r-lm-summary.html

stats.stackexchange.com/questions/5135/interpretation-of-rs-lm-output



More things

- Connect Rstudio to github

More things

- Connect Rstudio to github
- Data.frames

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 `treesize` 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