

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

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

- RUN CODE**: Points to the 'Run' button in the top toolbar.
- SCRIPTS**: Points to the source editor containing R code.
- OBJECTS IN WORKSPACE**: Points to the 'Environment' pane on the right, which lists objects like 'statlocsp' and 'statnames'.
- DOCUMENTATION**: Points to the 'Help' button in the bottom toolbar.
- PLOTS**: Points to a scatter plot in the 'Plots' pane, showing 'sort(as.Date(first))' on the y-axis (ranging from 1850 to 1950) and 'Index' on the x-axis (ranging from 0 to 150).
- CODE EXECUTION**: Points to the 'Console' pane at the bottom, which shows the output of the R script, including map coordinates and date ranges.

The R code in the source editor includes:

```

country <- raster::shapefile("../RiverFiles/country", country.shp)
country <- sp::spTransform(country,
sp::plot(country, col="blue")

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

library(OSMscale) # pointsMap, projectPoints
map <- pointsMap(yy.xx, statlocs@data, zoom=7, fx=0.3, type="maptoolkit-to
statlocsp <- projectPoints(yy.xx, statlocs@data, to=pozm())
map2 <- pointsMap(yy.xx, statlocs@data, map=map, proj=pl1())

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

```

The console output shows:

```

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...
> statlocsp <- projectPoints(yy.xx, statlocs@data, to=pozm())
> 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)

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`
- ▶ 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

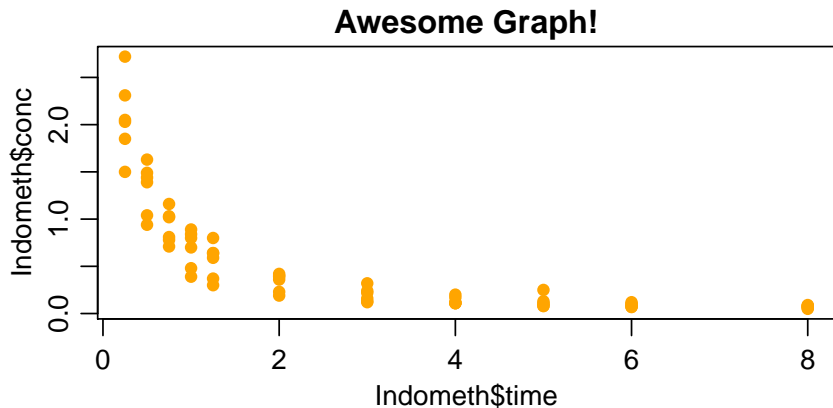
- ▶ Copy the file `treesize.txt` (from bit.ly/swc_tree)
- ▶ Tell R where to look for it with: `setwd("C:/path/to/input")`
`# change back- to forwardslashes`
- ▶ Read the file into R with the command `read.table`.
- ▶ If R tells you "no such file" exists, check the output of `dir()`.
- ▶ Use the documentation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

Plotting I

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

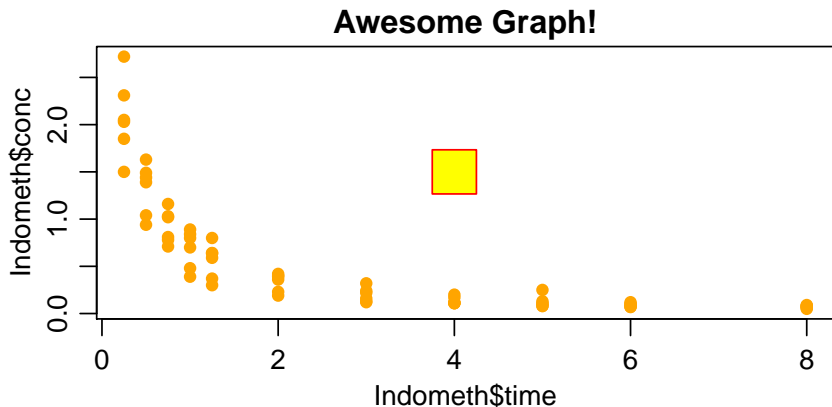
```
plot(x=Indometh$time, y=Indometh$conc,  
     col="orange", pch=16, main="Awesome Graph!")
```



Plotting I

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

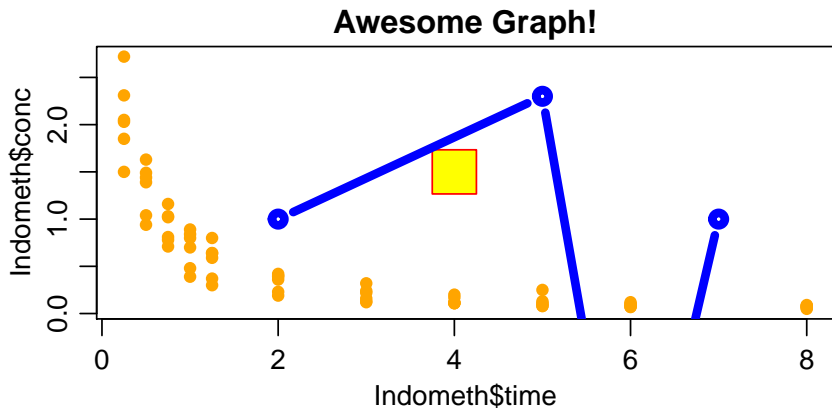
```
points(4, 1.5, pch=22, bg="yellow", cex=4, col="red")  
# PointCharacter, BackGround, Character EXpansion
```



Plotting I

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)
```



Plotting II: Our 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 for scatterplot exercise

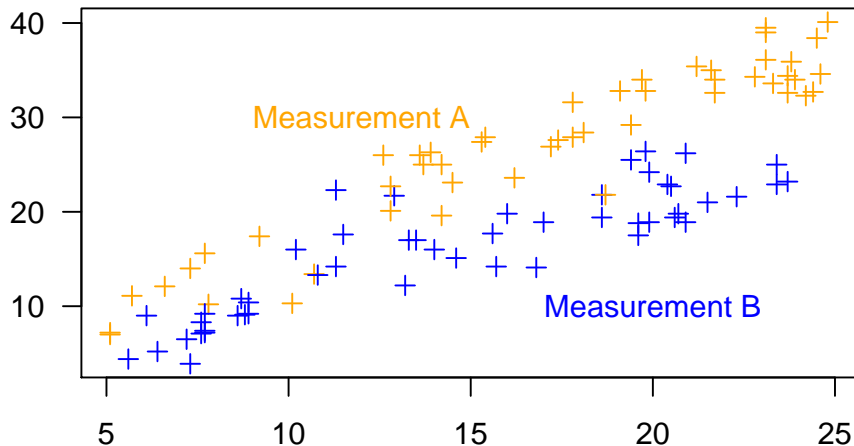
```
treesize <- read.table(file="treesize.txt", header=TRUE)

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 for scatterplot exercise

Older trees are larger



Solution for scatterplot exercise: Histogram

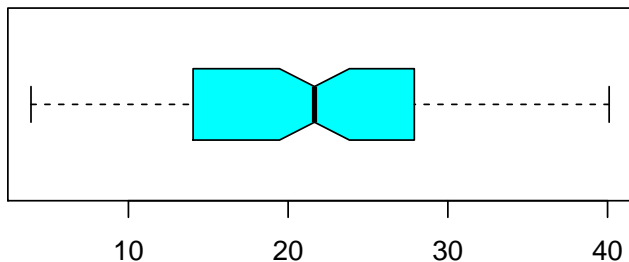
```
hist(treesize$height, col=6, breaks=20, las=1)
```

Histogram of treesize\$height



Solution for scatterplot exercise: Boxplot

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

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
- ▶ 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)]`

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