R introduction

start your programming journey in 1 hour

ENCOURAGED

Berry Boessenkool, June 2018

berry-b@gmx.de

github.com/brry/hour

Presentation template generated with `berryFunctions::createPres`

`print("Hello world!")`

- Berry Boessenkool → berry-b@gmx.de

`print("Hello world!")`

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University

`print("Hello world!")`

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fan

`print("Hello world!")`

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fanatic

print("Hello world!")

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fanatic since 2010

`print("Hello world!")`

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fanatic since 2010
- Teaching, programming, consulting, community BERLIN R USER GROUP

`print("Hello world!")`

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fanatic since 2010
- Teaching, programming, consulting, community BERLIN R USER GROUP
-

`print("Hello world!")`

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fanatic since 2010
- Teaching, programming, consulting, community BERLIN R USER GROUP
- R is free,

print("Hello world!")

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fanatic since 2010
- Teaching, programming, consulting, community
- R is free, open source,

`print("Hello world!")`

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fanatic since 2010
- Teaching, programming, consulting, community
- is free, open source, has a large user community,

print("Hello world!")

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fanatic since 2010
- Teaching, programming, consulting, community
- is free, open source, has a large user community, will make your work efficient and productive

`print("Hello world!")`

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fanatic since 2010
- Teaching, programming, consulting, community
- R is free, open source, has a large user community, will make your work efficient and productive and is the standard for data analysis in many universities and industries

print("Hello world!")

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fanatic since 2010
- Teaching, programming, consulting, community BERLIN R USER GROUP
- R is free, open source, has a large user community, will make your work efficient and productive and is the standard for data analysis in many universities and industries
- R installation instructions: github.com/brry/course#install

print("Hello world!")

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fanatic since 2010
- Teaching, programming, consulting, community
- R is free, open source, has a large user community, will make your work efficient and productive and is the standard for data analysis in many universities and industries
- R installation instructions: github.com/brry/course#install
- Today:

`print("Hello world!")`

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fanatic since 2010
- Teaching, programming, consulting, community
- is free, open source, has a large user community, will make your work efficient and productive and is the standard for data analysis in many universities and industries
- R installation instructions: github.com/brry/course#install
- Today: wide but shallow introduction, no deep understanding

print("Hello world!")

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fanatic since 2010
- Teaching, programming, consulting, community BERLIN R USER GROUP
- R is free, open source, has a large user community, will make your work efficient and productive and is the standard for data analysis in many universities and industries
- R installation instructions: github.com/brry/course#install
- Today: wide but shallow introduction, no deep understanding
- Brief inputs followed by short exercises (for max learning)

`print("Hello world!")`

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fanatic since 2010
- Teaching, programming, consulting, community
- R is free, open source, has a large user community, will make your work efficient and productive and is the standard for data analysis in many universities and industries
- R installation instructions: github.com/brry/course#install

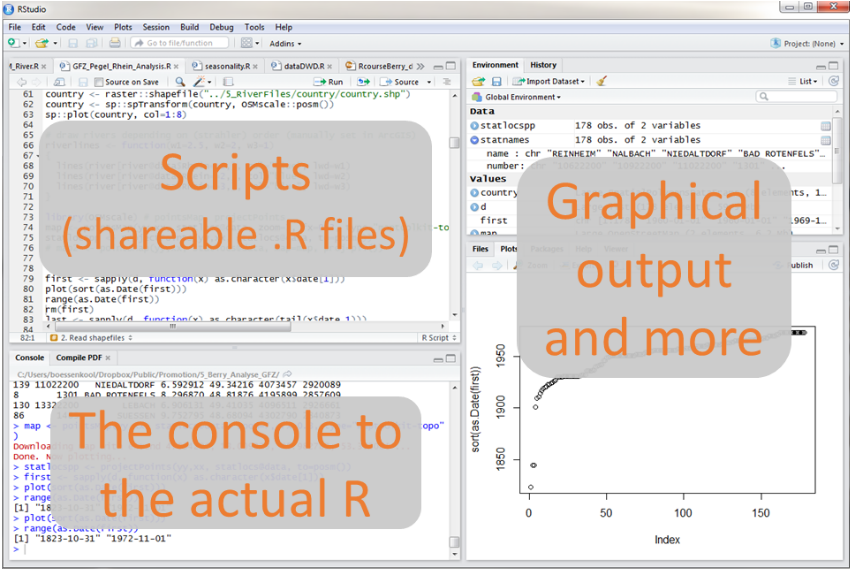
- Today: wide but shallow introduction, no deep understanding
- Brief inputs followed by short exercises (for max learning)
- Don't hesitate to ask the helpers

print("Hello world!")

- Berry Boessenkool → berry-b@gmx.de
- Geoecology @ Potsdam University
- R Fanatic since 2010
- Teaching, programming, consulting, community BERLIN R USER GROUP
- R is free, open source, has a large user community, will make your work efficient and productive and is the standard for data analysis in many universities and industries
- R installation instructions: github.com/brry/course#install

- Today: wide but shallow introduction, no deep understanding
- Brief inputs followed by short exercises (for max learning)
- Don't hesitate to ask the helpers
- If we're proceeding too fast, please interrupt!

Integrated Development Environment (IDE): RStudio



Get started in R

Get started in R

Exercise 1: R is an awesome calculator

In the console, calculate 21+21 , 7*6 and $\frac{0,3}{4} * \sqrt{313600}$

If you don't know how to compute a square root in R, you can google it!

Get started in R

Exercise 1: R is an awesome calculator

In the console, calculate $21+21$, $7*6$ and $\frac{0,3}{4} * \sqrt{313600}$

If you don't know how to compute a square root in R, you can google it!

```
21+21 ; 7*6 ; 0.3/4*sqrt(313600)
```

Get started in R

Exercise 1: R is an awesome calculator

In the console, calculate 21+21 , 7*6 and $\frac{0{,}3}{4} * \sqrt{313600}$

If you don't know how to compute a square root in R, you can google it!

```
21+21 ; 7*6 ; 0.3/4*sqrt(313600)
```

- objects: assign with $<-$ Rstudio Keyboard shortcut: `ALT` + `-`

```
nstudents <- 15
nstudents
nstudents > 12
```

Get started in R

Exercise 1: R is an awesome calculator

In the console, calculate 21+21 , 7*6 and $\frac{0{,}3}{4} * \sqrt{313600}$

If you don't know how to compute a square root in R, you can google it!

```
21+21 ; 7*6 ; 0.3/4*sqrt(313600)
```

- objects: assign with $<-$ Rstudio Keyboard shortcut: `ALT` + `-`

  ```
  nstudents <- 15
  nstudents
  nstudents > 12
  ```

- What's a good object name?

Get started in R

Exercise 1: R is an awesome calculator

In the console, calculate 21+21 , 7*6 and $\frac{0{,}3}{4} * \sqrt{313600}$

If you don't know how to compute a square root in R, you can google it!

```
21+21 ; 7*6 ; 0.3/4*sqrt(313600)
```

- objects: assign with $<-$ Rstudio Keyboard shortcut: `ALT` + `-`

  ```
  nstudents <- 15
  nstudents
  nstudents > 12
  ```

- What's a good object name? → short, but explanatory,

Get started in R

Exercise 1: R is an awesome calculator

In the console, calculate 21+21 , 7*6 and $\frac{0,3}{4} * \sqrt{313600}$

If you don't know how to compute a square root in R, you can google it!

```
21+21 ; 7*6 ; 0.3/4*sqrt(313600)
```

- objects: assign with $<-$ Rstudio Keyboard shortcut: `ALT` + `-`

```
nstudents <- 15
nstudents
nstudents > 12
```

- What's a good object name? → short, but explanatory, lowerCamelStandard_or_underscore are good naming conventions

Get started in R

Exercise 1: R is an awesome calculator

In the console, calculate 21+21 , 7*6 and $\frac{0{,}3}{4} * \sqrt{313600}$

If you don't know how to compute a square root in R, you can google it!

```
21+21 ; 7*6 ; 0.3/4*sqrt(313600)
```

- objects: assign with $<-$ Rstudio Keyboard shortcut: `ALT` + `-`

```
nstudents <- 15
nstudents
nstudents > 12
```

- What's a good object name? → short, but explanatory, lowerCamelStandard_or_underscore are good naming conventions
- comments: `# everything after a hashtag is not executed.`

Get started in R

Exercise 1: R is an awesome calculator

In the console, calculate 21+21 , 7*6 and $\frac{0,3}{4} * \sqrt{313600}$

If you don't know how to compute a square root in R, you can google it!

```
21+21 ; 7*6 ; 0.3/4*sqrt(313600)
```

- objects: assign with $<-$ Rstudio Keyboard shortcut: `ALT` + `-`
  ```
  nstudents <- 15
  nstudents
  nstudents > 12
  ```
- What's a good object name? → short, but explanatory, lowerCamelStandard_or_underscore are good naming conventions
- comments: `# everything after a hashtag is not executed.`
- scripts: Rstudio

Preparation to reading files

Exercise 2: Set up folder and script

1. Create a folder for this course
2. With `rightclick on Raw + save as`, download the weather dataset and metadata into that folder
3. Create an R script in the same folder (Rstudio: `CTRL` + `SHIFT` + `N`, then `CTRL` + `S`)
4. Tell R where to look for data through Rstudio: Session - Set Working Directory - To Source File Location
5. Copy the command thus sent to the console into the beginning of the script. This makes it reproducible later on.

```
setwd("C:/path/to/input") # change back- to forwardslashes
```

Reading files into R, opening the help documentation

- Read the climate data file into R with the command `read.table`

Reading files into R, opening the help documentation

- Read the climate data file into R with the command `read.table`
- Assign it to an object with a good name

Reading files into R, opening the help documentation

- Read the climate data file into R with the command `read.table`
- Assign it to an object with a good name

```
clim <- read.table(file="clim.txt")
```

Reading files into R, opening the help documentation

- Read the climate data file into R with the command `read.table`
- Assign it to an object with a good name

```
clim <- read.table(file="clim.txt")
```

- If R tells you "no such file" exists, check the output of `dir()`.

Reading files into R, opening the help documentation

- Read the climate data file into R with the command `read.table`
- Assign it to an object with a good name

```
clim <- read.table(file="clim.txt")
```

- If R tells you "no such file" exists, check the output of `dir()`.
- Check the object with `head(YourObject)`.

Reading files into R, opening the help documentation

- Read the climate data file into R with the command `read.table`
- Assign it to an object with a good name

```
clim <- read.table(file="clim.txt")
```

- If R tells you "no such file" exists, check the output of `dir()`.
- Check the object with `head(YourObject)`.
- `str(YourObject)` must yield 19 columns with data types: `int`, `factor`, `num`.

Reading files into R, opening the help documentation

- Read the climate data file into R with the command `read.table`
- Assign it to an object with a good name

```
clim <- read.table(file="clim.txt")
```

- If R tells you "no such file" exists, check the output of `dir()`.
- Check the object with `head(YourObject)`.
- `str(YourObject)` must yield 19 columns with data types: `int`, `factor`, `num`.
- Use the documentation to find out settings for the arguments:

```
help(read.table)   ;   ?read.table   #   or press F1.
```

Reading files into R, opening the help documentation

- Read the climate data file into R with the command `read.table`
- Assign it to an object with a good name

```
clim <- read.table(file="clim.txt")
```

- If R tells you "no such file" exists, check the output of `dir()`.
- Check the object with `head(YourObject)`.
- `str(YourObject)` must yield 19 columns with data types: `int`, `factor`, `num`.
- Use the documentation to find out settings for the arguments:

```
help(read.table)   ;   ?read.table   #   or press F1.
```

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

Reading files into R

Exercise 3: Reading files

1. Read the metadata file into R with the command `read.table`, again assigning it to an object with a good name.
2. You need to set the arguments `file`, `sep`, `header`, `stringsAsFactors`.
3. `str(YourObject)` must yield 5 chr (character) columns
4. BONUS: what does `tail(clim)` return?

Solution to exercise 3: Reading files

```
clim <- read.table(file="clim.txt", header=TRUE)
meta <- read.table(file="meta.txt", header=TRUE,
                   sep=";", stringsAsFactors=FALSE)
```

```
tail(Clim) # last rows of an object

##     STATIONS_ID MESS_DATUM QN_3   FX  FM QN_4 RSK
## 545        3987 2018-06-20    1  7.6 2.5    1 0.0
## 546        3987 2018-06-21    1 19.3 5.9    1 1.0
## 547        3987 2018-06-22    1 17.2 5.8    1 3.4
## 548        3987 2018-06-23    1 15.6 7.0    1 2.1
## 549        3987 2018-06-24    1 10.3 4.8    1 2.9
## 550        3987 2018-06-25    1 10.0 3.6    1 0.0
```

Objects: data.frames

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

Objects: data.frames

- For tables with different data types (numbers, characters, categories, integers), R has the object type data.frame
- `read.table` also returns a data.frame

Objects: data.frames

- For tables with different data types (numbers, characters, categories, integers), R has the object type data.frame
- `read.table` also returns a data.frame
- If we have the object `df`, we can subset with `df[rows,columns]`

Objects: data.frames

- For tables with different data types (numbers, characters, categories, integers), R has the object type data.frame
- `read.table` also returns a data.frame
- If we have the object `df`, we can subset with `df[rows,columns]`
- `df[1,2:4];`

Objects: data.frames

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

Objects: data.frames

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

Objects: data.frames

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

Objects: data.frames

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

Exercise 4: Data.frame indexing

From the climate dataset, obtain:

1. The first 5 values in column 4
2. The column UPM (relative humidity)
3. BONUS 1: The maximum sunshine duration (SDK)
4. BONUS 2: What command do you need to get the number of rows?
5. BONUS 3: What is better and worse in `df[ ,"name"]` vs `df[ ,3]`?

Solution to exercise 4: subsetting data.frames

```
clim[1:5, 4]

## [1] 10.0 13.9 17.1 22.2 22.6

Clim$UPM

##  [1] 96.13 87.92 93.29 79.63 76.00 96.17 93.13 93.96 83.38 82.92 94.33 93.04
## [13] 87.13 77.33 83.08 79.96 94.46 95.54 81.13 82.71

max(clim$SDK)  ;   max(clim[ ,"SDK"])

## [1] 15.933
## [1] 15.933

nrow(clim)

## [1] 550
```

`df[ ,"name"]` is better understandable for humans and still returns the same if the order of the columns is changed. But it takes more typing.

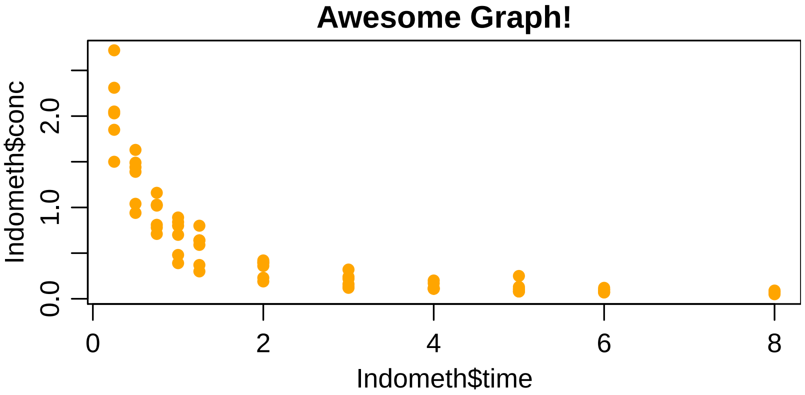
Plotting

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

Plotting

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

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



Plotting

Please convert the date column with

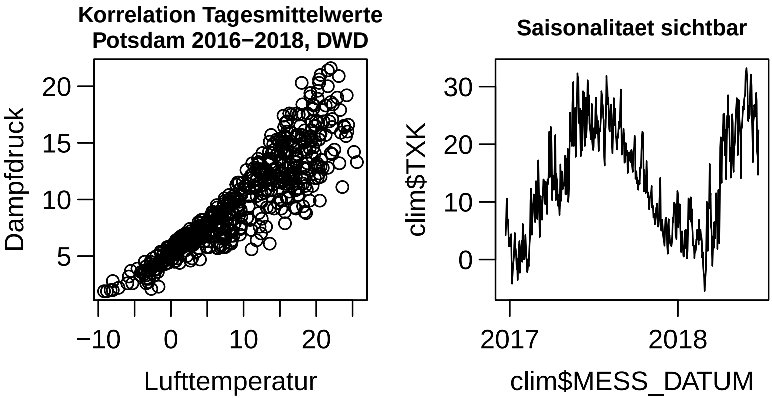
```
clim$MESS_DATUM <- as.Date(clim$MESS_DATUM, format="%Y-%m-%d")
```

Exercise 5: Scatterplots, line plots

1. Generate a figure with `plot(clim$VPM, clim$TMK)`
2. Improve the axis labels. Use the metadata to figure out the column name meanings.
3. BONUS: Add an informative graph title
4. Plot a time series of the daily temperature maximum. What value do you need to give to the argument type? Again, use the documentation of `?plot()` to find out.

Solution to exercise 5: Scatterplots, line plots

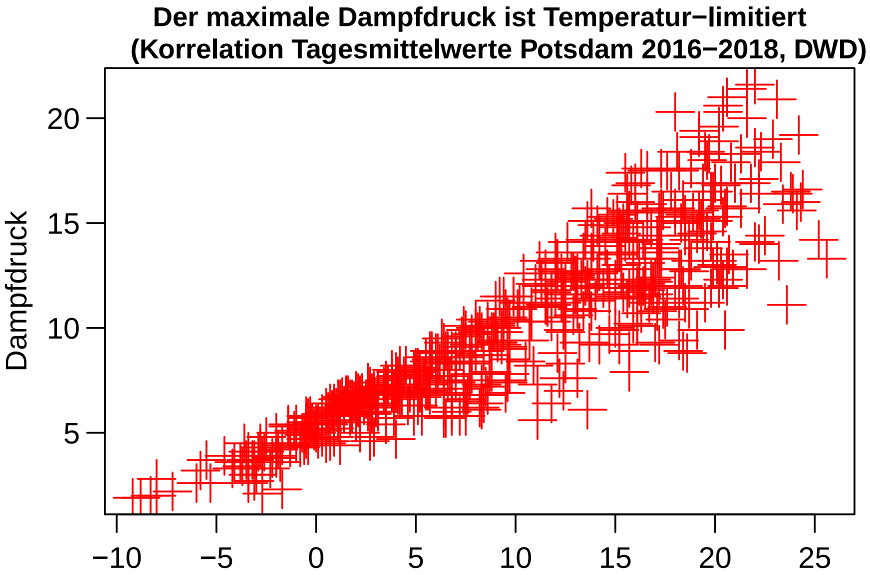
```
plot(clim$TMK, clim$VPM, xlab="Lufttemperatur", ylab="Dampfdruck",
     main="Korrelation Tagesmittelwerte\nPotsdam 2016-2018, DWD")
plot(clim$MESS_DATUM, clim$TXK, type="l", main="Saisonalitaet sichtbar")
```



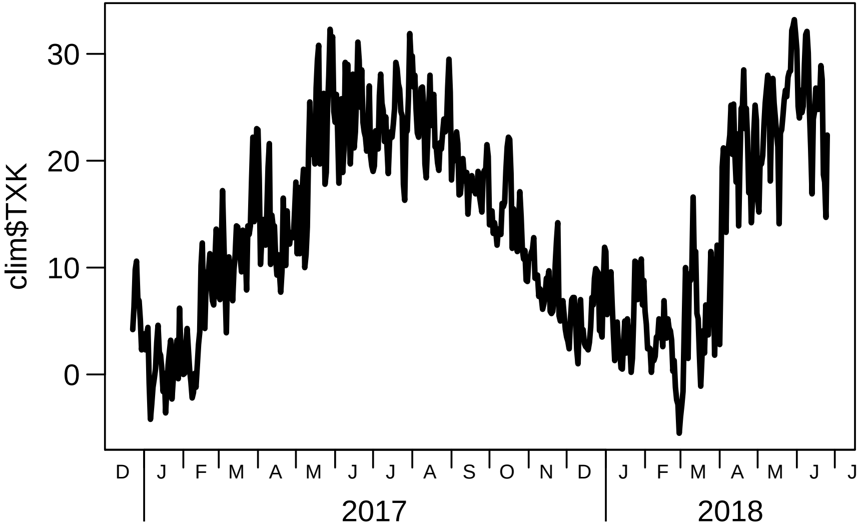
Plotting live demo I

```
par(mar=c(2,3,2,1), mgp=c(2,0.7,0))
plot(clim$TMK, clim$VPM, xlab="Lufttemperatur", ylab="Dampfdruck",
     main="Der maximale Dampfdruck ist Temperatur-limitiert
     (Korrelation Tagesmittelwerte Potsdam 2016-2018, DWD)",
     las=1, pch=3, cex=2, col="red", cex.main=0.9)
#
#
# install.packages("berryFunctions")
plot(clim$MESS_DATUM, clim$TXK, type="l", xlab="", xaxt="n", las=1, lwd=3)
berryFunctions::monthAxis()
```

Plotting live demo II



Plotting live demo III



commonly needed `plot` arguments

```r
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 style (axis numbers upright)
xaxt="n" # axis type (none to suppress axis)
)
```

Preparing for for loops

Please give your metadata rownames as follows:

```
rownames(meta) <- meta$Par
```

Exercise 6: using objects to subset

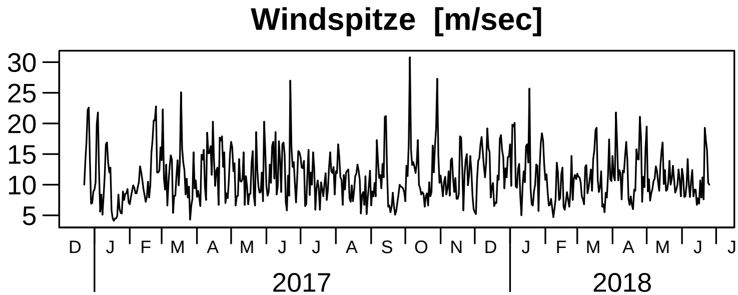
1. Create an object `var` with the character string `"FX"`.
2. Get the meta value for row `var` and column `"Label"`.
3. Plot the `var` column of `clim` as a time series.
4. Use the output from task 2 to give the plot a title.
5. BONUS: make the x-axis nicer by suppressing the default axis (`xaxt="n"`) and adding a monthAxis from the `berryFunctions` package.

Solution to exercise 6: using objects to subset

```
var <- "FX"
meta[var, "Label"]

## [1] "Windspitze  [m/sec]"

plot(clim$MESS_DATUM, clim[,var], type="l", xaxt="n",
     main=meta[var, "Label"], ylab="", xlab="")
berryFunctions::monthAxis()
```



For loops

A for loop creates a variable, sets it to a value, runs some code,

For loops

A for loop creates a variable, sets it to a value, runs some code, then sets the variable to the next value, runs the code with that, and so on.

For loops

A for loop creates a variable, sets it to a value, runs some code, then sets the variable to the next value, runs the code with that, and so on.

```
for(var in meta$Par)
{
  plot(clim$MESS_DATUM, clim[,var], type="l", xaxt="n",
       main=meta[var, "Label"], ylab="", xlab="")
  berryFunctions::monthAxis()
}
```

Selected resources

Books:

Selected resources

Books:

- Grolemund & Wickham (2017) R for Data Science

Selected resources

Books:

- Grolemund & Wickham (2017) R for Data Science
- J. Adler (2010): R in a Nutshell

Selected resources

Books:

- Grolemund & Wickham (2017) R for Data Science
- J. Adler (2010): R in a Nutshell
- U. Ligges (2008): Programmieren mit R (German)

Selected resources

Books:

- Grolemund & Wickham (2017) R for Data Science
- J. Adler (2010): R in a Nutshell
- U. Ligges (2008): Programmieren mit R (German)

Tutorials:

Selected resources

Books:

- Grolemund & Wickham (2017) R for Data Science
- J. Adler (2010): R in a Nutshell
- U. Ligges (2008): Programmieren mit R (German)

Tutorials:

- Microsoft + Datacamp (best course I know of, with login, but free)

Selected resources

Books:

- Grolemund & Wickham (2017) R for Data Science
- J. Adler (2010): R in a Nutshell
- U. Ligges (2008): Programmieren mit R (German)

Tutorials:

- Microsoft + Datacamp (best course I know of, with login, but free)
- STAT 545

Selected resources

Books:

- Grolemund & Wickham (2017) R for Data Science
- J. Adler (2010): R in a Nutshell
- U. Ligges (2008): Programmieren mit R (German)

Tutorials:

- Microsoft + Datacamp (best course I know of, with login, but free)
- STAT 545
- github.com/brry/course

Selected resources

Books:

- Grolemund & Wickham (2017) R for Data Science
- J. Adler (2010): R in a Nutshell
- U. Ligges (2008): Programmieren mit R (German)

Tutorials:

- Microsoft + Datacamp (best course I know of, with login, but free)
- STAT 545
- github.com/brry/course

Reference cards:

Selected resources

Books:

- Grolemund & Wickham (2017) R for Data Science
- J. Adler (2010): R in a Nutshell
- U. Ligges (2008): Programmieren mit R (German)

Tutorials:

- Microsoft + Datacamp (best course I know of, with login, but free)
- STAT 545
- github.com/brry/course

Reference cards:

- base and advanced cheatsheets from Rstudio

Selected resources

Books:

- Grolemund & Wickham (2017) R for Data Science
- J. Adler (2010): R in a Nutshell
- U. Ligges (2008): Programmieren mit R (German)

Tutorials:

- Microsoft + Datacamp (best course I know of, with login, but free)
- STAT 545
- github.com/brry/course

Reference cards:

- base and advanced cheatsheets from Rstudio
- RefCard by Tom Short & Jonas Stein

Selected resources

Books:

- Grolemund & Wickham (2017) R for Data Science
- J. Adler (2010): R in a Nutshell
- U. Ligges (2008): Programmieren mit R (German)

Tutorials:

- Microsoft + Datacamp (best course I know of, with login, but free)
- STAT 545
- github.com/brry/course

Reference cards:

- base and advanced cheatsheets from Rstudio
- RefCard by Tom Short & Jonas Stein

The internet:

Selected resources

Books:

- Grolemund & Wickham (2017) R for Data Science
- J. Adler (2010): R in a Nutshell
- U. Ligges (2008): Programmieren mit R (German)

Tutorials:

- Microsoft + Datacamp (best course I know of, with login, but free)
- STAT 545
- github.com/brry/course

Reference cards:

- base and advanced cheatsheets from Rstudio
- RefCard by Tom Short & Jonas Stein

The internet:

- StackOverflow for programming questions <– main resource