good practices in programming

Help-desk workshop for statistical analyses in R, 2021

Main Sources of Content

Two-day workshop "Best practices in Programming"

ETH zürich

Scientific IT Services

SIB Swiss Institute of Bioinformatics

BES guide :
https://www.britishecologicalsociety.org/wp-content/uploads/2019/06/BES-Guide-Reproducible-Code-2019.pdf?utm_source=web&utm_medium=web&utm_campaign=better_science

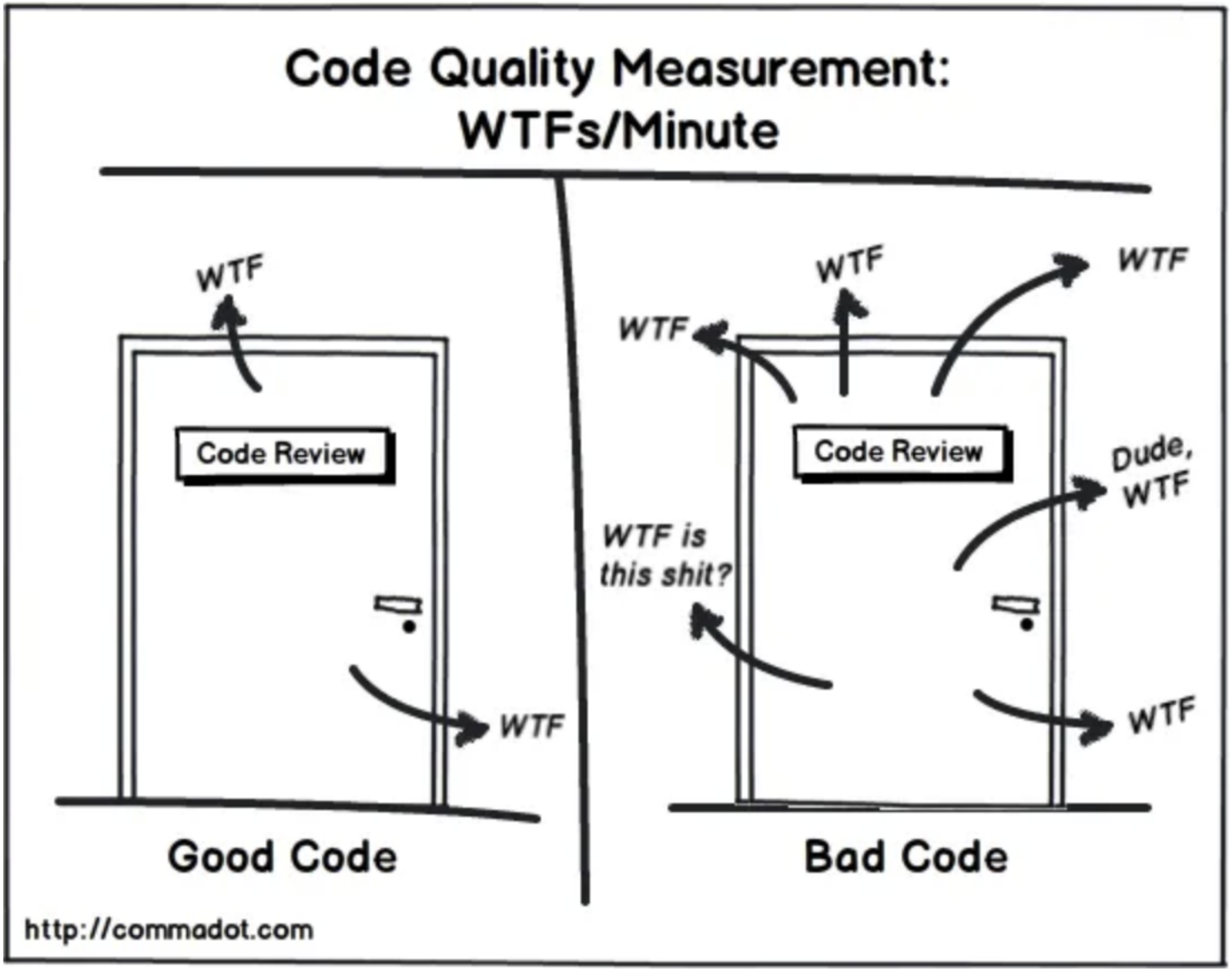
Course : https://www.sib.swiss/training/course/20210707_BPP

Content

- why care?
- Good programming practices
- Take home/ Reflection

why care?

why care? individual level



why care? individual level

- happier programming experience
- easier to …
  - understand for collaborators/ reviewers
  - reproduce
  - find errors
  - build on the code
- enables publishing / sharing

"Code is read more often than it is written"

why care? societal level

SORTEE

Society for Open, Reliable, and Transparent Ecology and Evolutionary Biology

NEWS | PLANTS & ANIMALS

Psychology's replication crisis inspires ecologists to push for more reliable research

New society aims for transparency and culture change in ecology

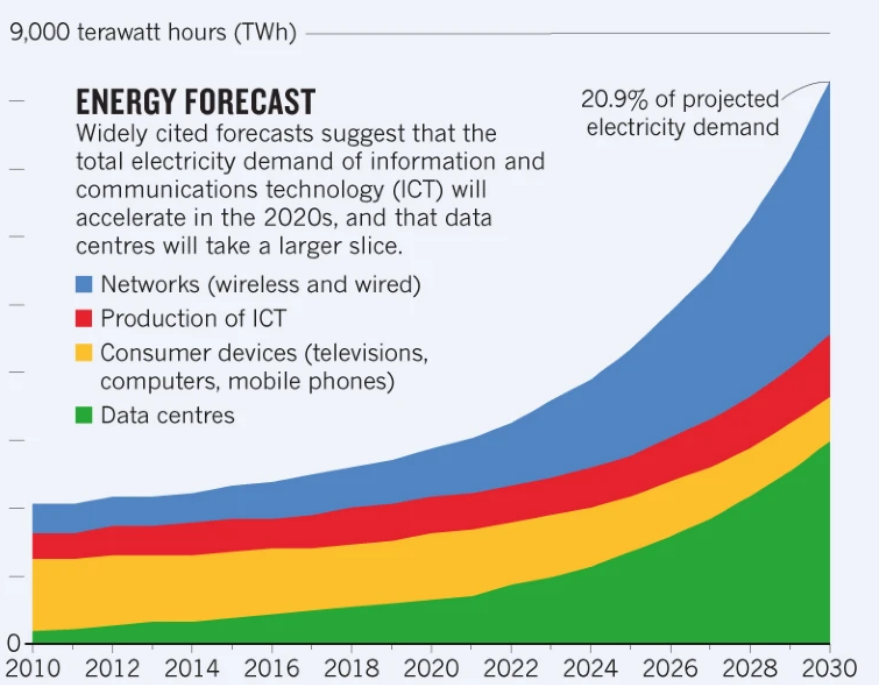
9 DEC 2020 · BY CATHLEEN O'GRADY

why care? societal level

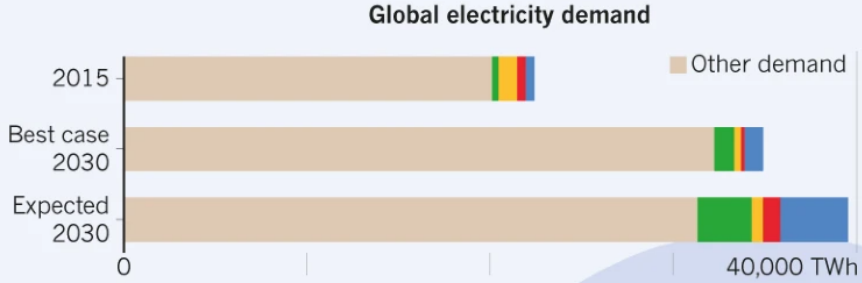
- computation is energy-expensive
- inefficient code, cloud-computing and remote working environemtns
- Ecology :

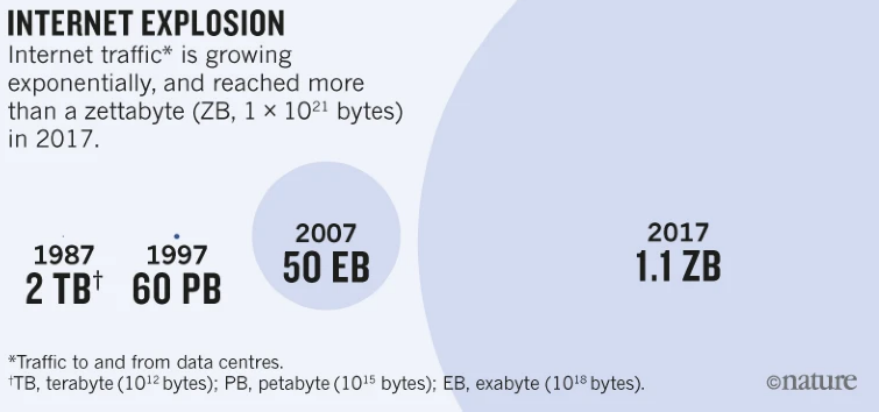
  doing the work twice is energy-expensive
- clean coding is sustainable

Source : “How to stop data centers from gobbling up the world’s electricity” – Nature News feature Article https://www.nature.com/articles/d41586-018-06610-y

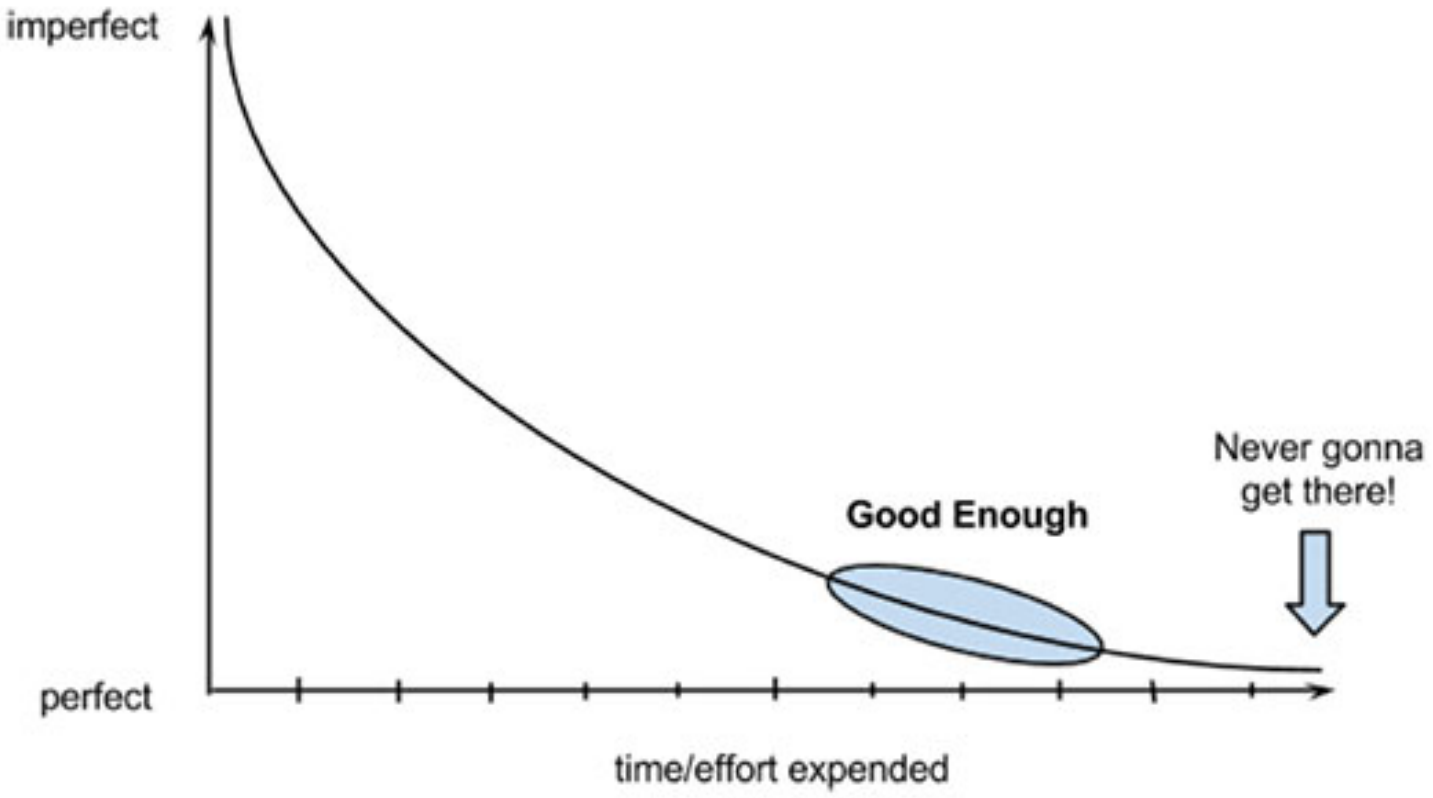


The chart above is an ‘expected case’ projection from Anders Andrae, a specialist in sustainable ICT. In his ‘best case’ scenario, ICT grows to only 8% of total electricity demand by 2030, rather than to 21%.





about perfection ...

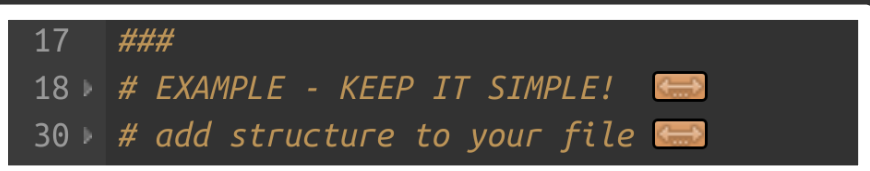


Good programming practices

keep it simple!

```r
###
# EXAMPLE - KEEP IT SIMPLE!  ####
#
# bad example : all in one line
info_data <- data.table::fread(paste(pathtodata, "/data_assembly/helper_data/info_data.csv", sep = ""), header=T)
#
# good example : use many lines
info_data <- data.table::fread(
  paste(pathtodata,
        "/data_assembly/helper_data/info_data.csv",
        sep = ""),
  header=T)

# add structure to your file ####
# if you add four "#" in the end of the title, RStudio will allow
#   you to fold in and out the section
```



- use the lines
- use indenting
- add structure to your file

https://github.com/biodiversity-exploratories-synthesis/useful_functions/blob/main/2021_synthesis_Rhelpdesk_good_programming_practices.R

use a code style

- Good coding style is like correct punctuation: you can manage without it, **butitsuremakesthingseasiertoread**.
- e.g. the tidyverse style guide

```r
# Good
average <- mean(feet / 12 + inches, na.rm = TRUE)
# Bad
average<-mean(feet/12+inches,na.rm=TRUE)

# Good
do_something_very_complicated(
  something = "that",
  requires = many,
  arguments = "some of which may be long"
)
# Bad
do_something_very_complicated("that", requires, many, arguments,
                              "some of which may be long"
)
```

tidyverse style guide : https://style.tidyverse.org/index.html
short version : http://adv-r.had.co.nz/Style.html
R packages to help with code style : https://styler.r-lib.org/ and https://github.com/jimhester/lintr

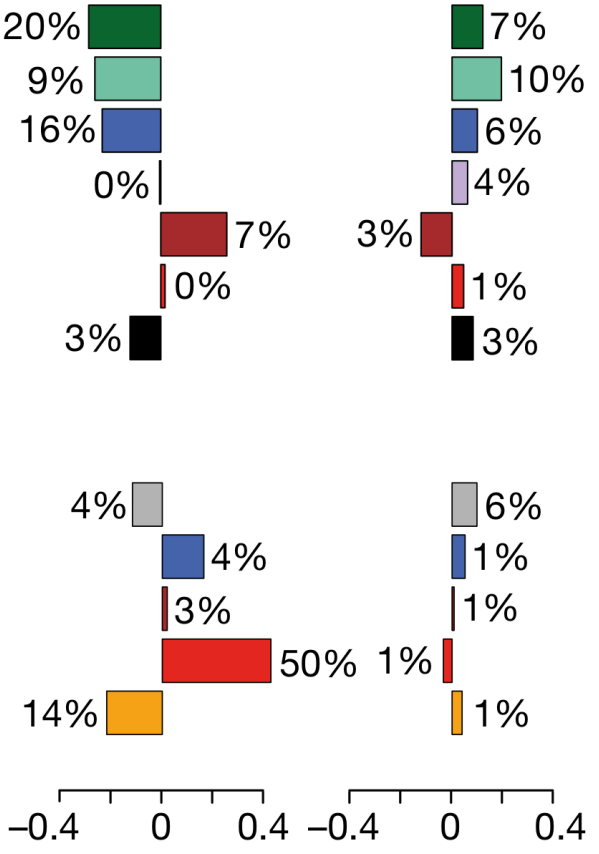
use a code style

- Variable and function names with lowercase letters, separated by underscore "_"
  - Good : `day_one, day_1`
  - Bad : `DayOne, dayone`
- variable names should be nouns and function names should be verbs
  - `day_one`
  - `calculate_time_span`
- avoid re-using names of common functions and variables. This will cause confusion for the readers of your code
  - `Bad : T ← FALSE`
  - `Bad : mean ← function(x) sum(x)`
- more recommendation in the tidyverse styleguide

tidyverse style guide : https://style.tidyverse.org/index.html
short version : http://adv-r.had.co.nz/Style.html
R packages to help with code style : https://styler.r-lib.org/ and https://github.com/jimhester/lintr

comment your code

- code without comments can be as useless as a plot without axis labels
- comment the unexpected
  - everything not obvious from your code
  - e.g. if you use a solution from the internet, provide the link in a comment
  - explain alternative values for parameters, or things you had to look up when you wrote the code
- better too much than too few comments




```
//When I wrote this, only God and I understood what I was doing
//Now, God only knows
```

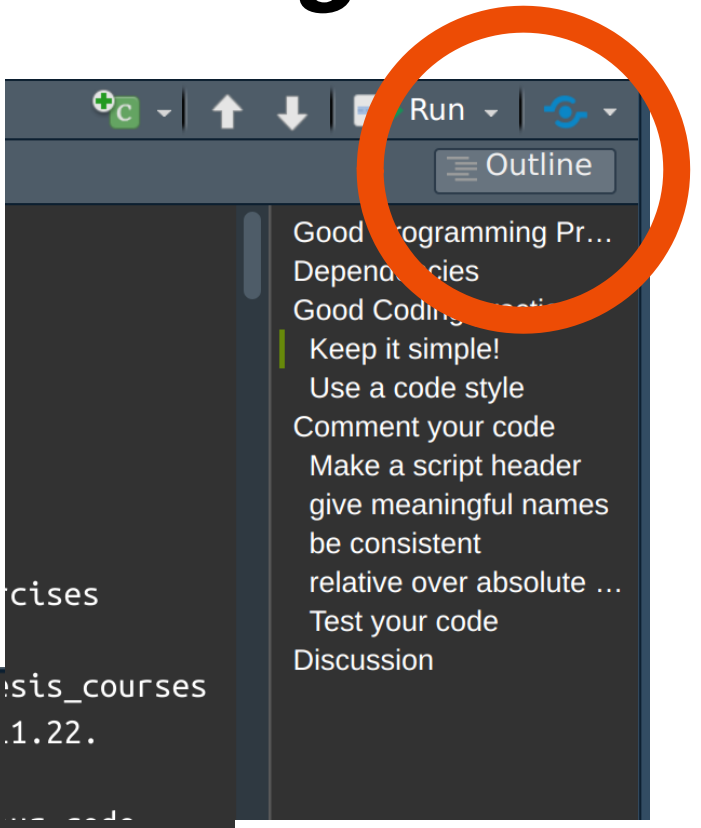
comment your code

```
26  ###
27  # EXAMPLE 2 - COMMENT YOUR CODE
28  #
29  # Bad example : comment the obvious
30  # read table example.csv
31  info_data <- data.table::fread("example.csv", header = T)
32  #
33  # Good example : comment the unexpected
34  # use package data.table to read in very large table "example.csv"
35  info_data <- data.table::fread("example.csv", header = T)
```

https://github.com/biodiversity-exploratories-synthesis/useful_functions/blob/main/2021_synthesis_Rhelpdesk_good_programming_practices.R

script header and subheadings

- minimal header :
  - aim of the script
  - date and creator
  - requirements / dependencies
  - output




```
1  ####################################
2  #                                  #
3  # GOOD PROGRAMMING PRACTICES       #
4  #                                  #
5  ####################################
6  # Biodiversity Exploratories Synthesis R helpdesk
7  # Noëlle Schenk 05.02.2021
8  #
9  # this file contains examples of practices to improve the
10 # readability of your code.
```

https://github.com/biodiversity-exploratories-synthesis/useful_functions/blob/main/2021_synthesis_Rhelpdesk_good_programming_practices.R

script header and subheadings

- libraries go first

```
17  library(data.table)
18  library(reshape2)
19  library(tidyr)
```

- hard-coded variables go second

```
21  path_to_data <- "example_data_synthesis_helpdesk/"
22  path_to_output <- "R_outputs_and_plots/"
```

  - relative paths over absolute paths

```
43  absolute_path <- "C:/user/yourname/Documents/Project1/Rscripts/data/info_data.csv"
44  relative_path <- "/data/info_data.csv"
```

- set a seed

```
24  set.seed(12)
```

https://towardsdatascience.com/best-practices-for-r-programming-ec0754010b5a

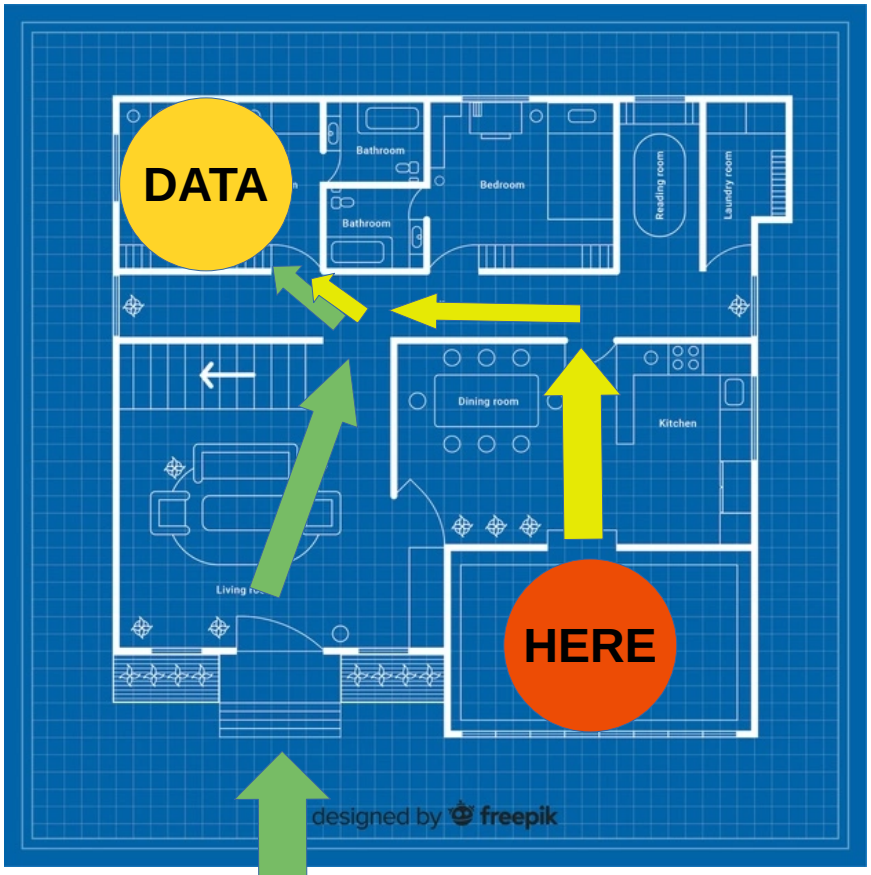
prefer relative file paths

- relative file paths

describe path from your working directory on

- absolute file paths

describe a path from your home directory on



```
43  absolute_path <- "C:/user/yourname/Documents/Project1/Rscripts/data/info_data.csv"
44  relative_path <- "/data/info_data.csv"
```

naming things

- file names :
  - meaningful
  - end in .R
  - no special characters but numbers, letters, - and _
- if files should be run in particular order, prefix them with numbers (and pad with zero).

```
# Good
fit_models.R
utility_functions.R

# Bad
fit models.R
foo.r
stuff.r
```

```
00_download.R
01_explore.R
...
09_model.R
10_visualize.R
```

naming things

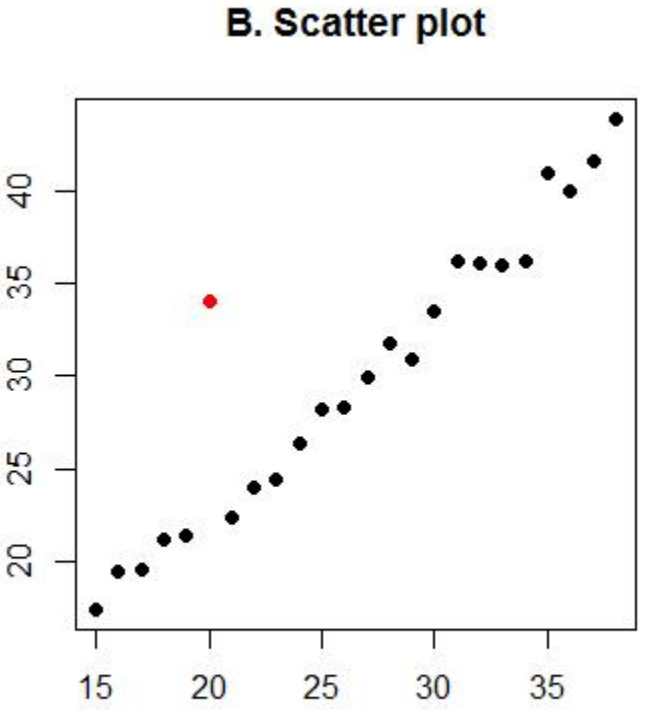
- names reveal intention
  - “area_rectangle” instead of “size”
  - reduce comments

```
89   60 <- time_tolerance # seconds
90   60 <- time_tolerance_in_seconds
```

- constants in capital : BYTES_PER_GB = 1073741824
- tab completion (RStudio) → easy to use long names
- understandable
- inambigous (e.g. “cat” = category or catalogue?)

T E N C Y - C O N S I S T E N C Y - C O N S I S T E N

Test your code



- Does the code do the job? Really?
  - simple summary stats : # NA, min, max
  - visual checking : “View()”, plot, print column
  - check format : #lines, data type [factor, character, numeric]
- keep checks and expected output in script
- use test packages in R
- unexpected input!

```
> if(nrow(na.omit(example)) != nrow(example)){
+   stop("The function requires complete data, but the dataset contains NA values.")
+ }
Error: The function requires complete data, but the dataset contains NA values.
```

Test your code

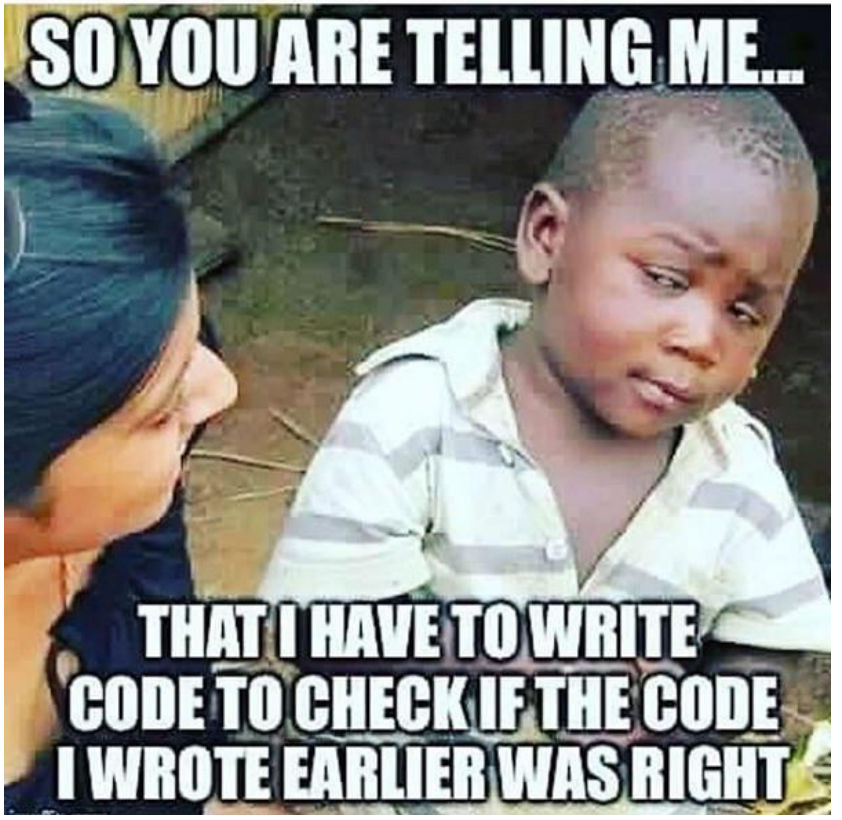
```
84  ###
85  # EXAMPLE - Test your code
86  #
87  # create example data.frame
88  example <- data.frame(species = c("Erithacus rubecula", "Arion vulgaris",
89                                    "Plantago Lanceolata"),
90                        abundance = c(2, NA, 10))
91  # action : recode NA value to 0
92  #    because Arion vulgaris was measured, but no individuals were found
93  example[which(example$species == "Arion vulgaris"), 2] <- 0
94  #
95  # test : did I do what I wanted?
96  sum(is.na(example$abundance)) # no NA values any more
97  sum(is.na(example$species))
98  min(example$abundance)        # min is 0 now, as expected
99  max(example$abundance)
100 View(example)
101 plot(example)
102 example$abundance
```

```
> if(nrow(na.omit(example)) != nrow(example)){
+   stop("The function requires complete data, but the dataset contains NA values.")
+ }
Error: The function requires complete data, but the dataset contains NA values.
```

`unit testing` (advanced)

- entangled software **breaks upon tiny changes** (e.g. new R version, package updates, bug fixes, ...)
- write integrated test functions which check if your code works right
- unit testing in R :

  `testthat`



save RStudio environment

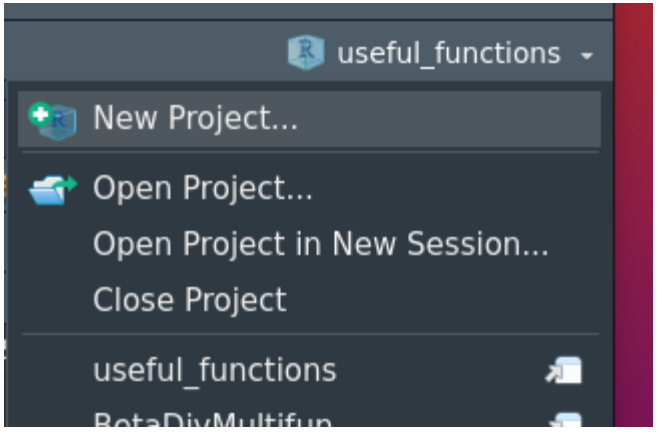
- save packages and their versions
- automatic reload by the next user
- package : **renv**

https://rstudio.github.io/renv/articles/renv.html

```
- naniar          [* -> 0.6.1]
- nlme            [* -> 3.1-162]
- nloptr          [* -> 2.0.3]
- nnet            [* -> 7.3-18]
- norm            [* -> 1.0-10.0]
- numDeriv        [* -> 2016.8-1.1]
- openssl         [* -> 2.0.5]
- parameters      [* -> 0.20.0]
- pbkrtest        [* -> 0.5.1]
- performance     [* -> 0.10.1]
- permute         [* -> 0.9-7]
- pillar          [* -> 1.8.1]
- pkgconfig       [* -> 2.0.3]
- pkgload         [* -> 1.3.2]
- plyr            [* -> 1.8.8]
- polynom         [* -> 1.4-1]
- praise          [* -> 1.0.0]
- prettyunits     [* -> 1.1.1]
- processx        [* -> 3.8.0]
- progress        [* -> 1.2.2]
- ps              [* -> 1.7.2]
- purrr           [* -> 0.3.5]
- quantreg        [* -> 5.94]
- randomForest    [* -> 4.7-1.1]
- rappdirs        [* -> 0.3.3]
- readr           [* -> 2.1.3]
```

use RStudio projects

- divide your work into contexts
  - each with it's own working directory
  - workspace (opened documents)
  - history
- Create new project:
  - in a brand new directory
  - in an existing directory
  - by cloning a version control (e.g. git, see later)

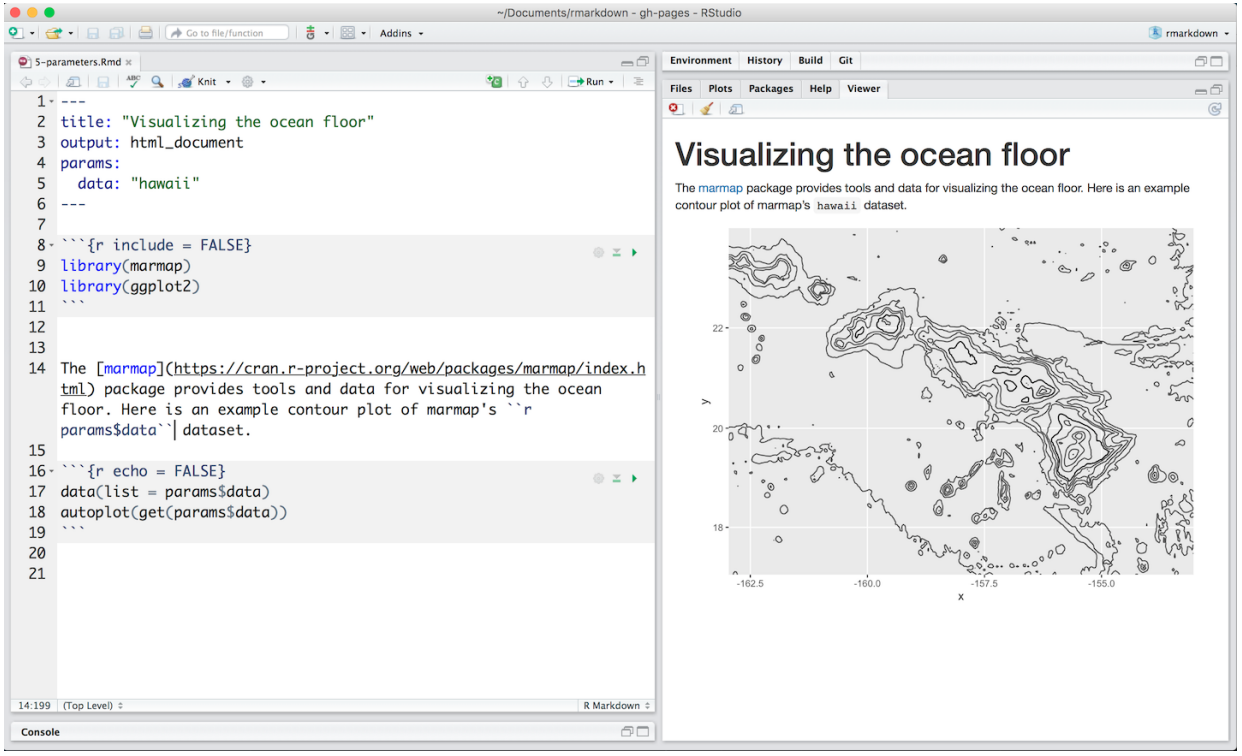


Tutorial creating RStudio project : https://support.rstudio.com/hc/en-us/articles/200526207-Using-Projects

use Rmarkdown

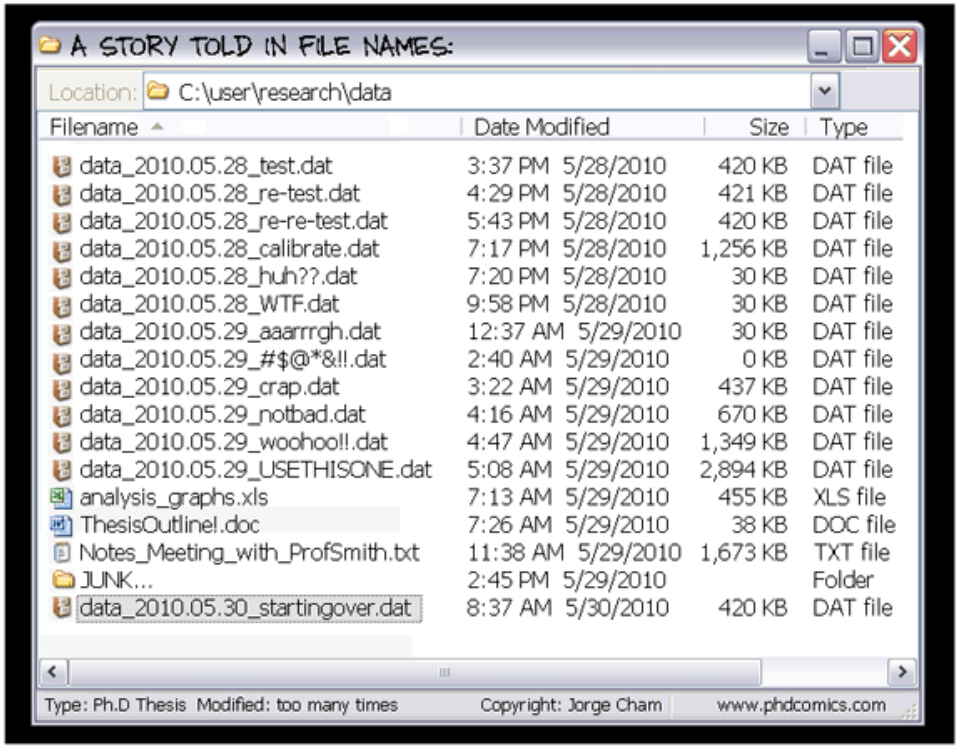
- combining R script with documentation
- Suitable to document workflow
  - combine analysis and documentation
    https://rmarkdown.rstudio.com/lesson-1.html

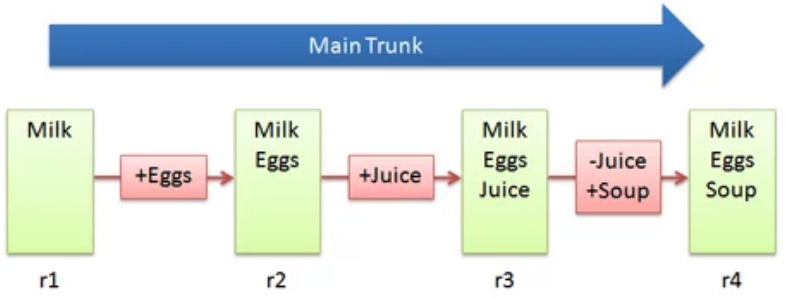
- Distill package for scientific appearance : https://rstudio.github.io/distill/

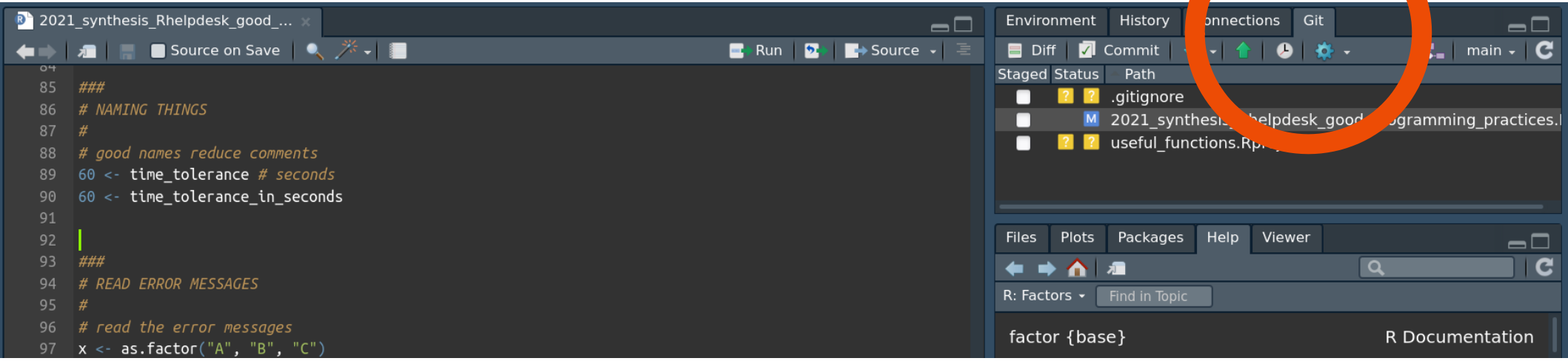


outlook

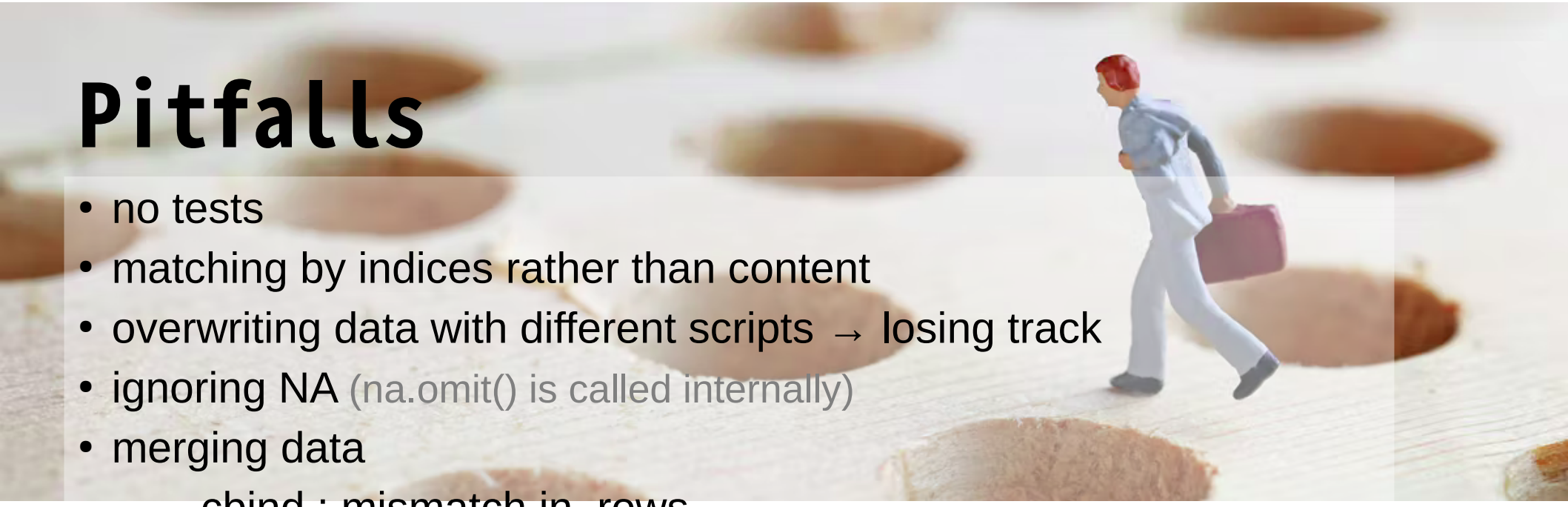
use version control







Pitfalls



- no tests
- matching by indices rather than content
- overwriting data with different scripts → losing track
- ignoring NA (na.omit() is called internally)
- merging data
  - → cbind : mismatch in rows
  - → duplicated or deleted rows
- missspelled variable names (continue coding with wrong variable)
- mixing up variables : similar or same name
- function with same name occurs in 2 packages
  - cowplot::plot_grid() vs. sjplot::plot_grid
  - reshape2::merge() vs. data.table::merge()
- overseeing special cases

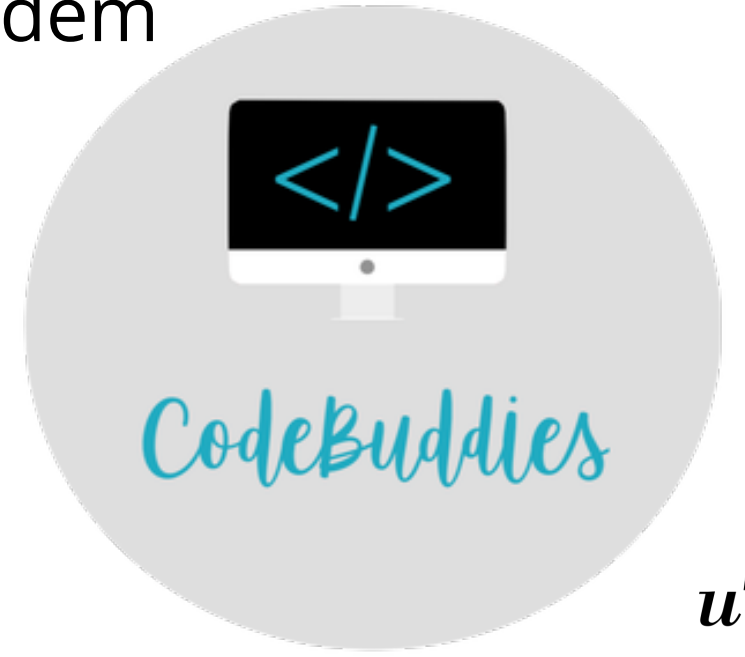
- … what is your experience?

how to improve my coding skills?

- be brave to try things out!
- take the time to learn & practise
- step by step – keep next steps in mind (e.g. functions)
- read other people's code : tandem
- exchange about coding

  https://www.tidytuesday.com/

- enjoy coding!

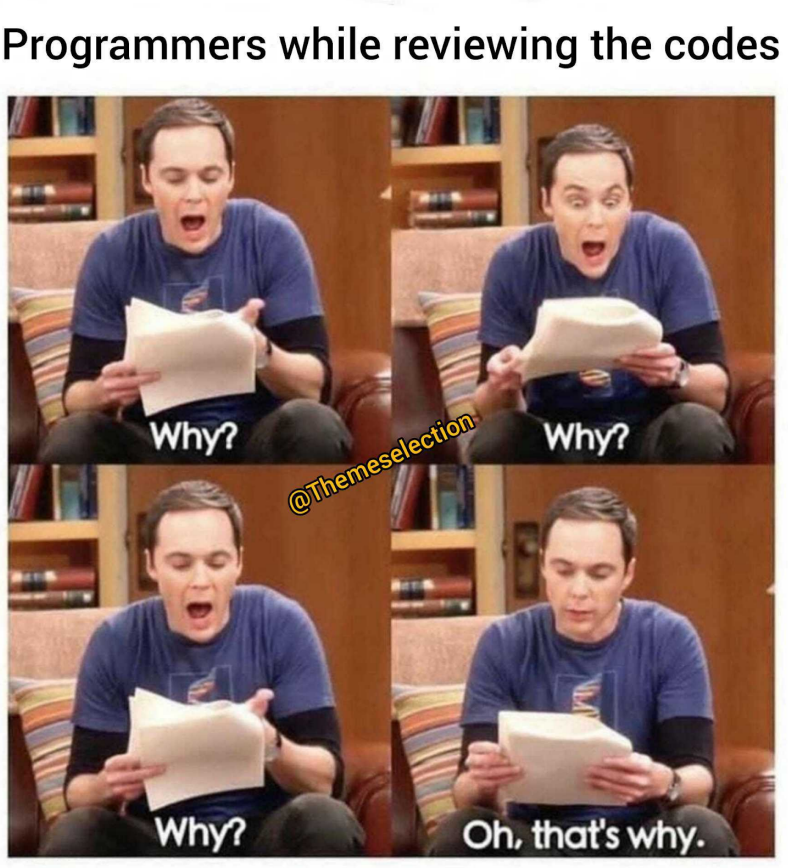


enjoy coding

Take home

TAKE HOME

- Good programming practices increase reproducibility
- Speficied practices to apply within & around the script
  - script header
  - comments
  - indenting
  - code style
  - Rmarkdown
  - testing



Reflections

- Which of the practices will you apply?
- How much of the practices did you already know, how much were new?
- Which practices, you think, will help you the most in the future?
- Which practices seem less useful to you?
- Where to go next?

- Do you think the amount of information was too much, or too less, or just right?

references and further reading

references and resources

- the links provided in this presentation
- ETH Zürich Science IT course : “Best practices in Programming” https://sis.id.ethz.ch/services/consultingtraining/programming_best_practices.html
- Exploratories Synthesis page : https://github.com/biodiversity-exploratories-synthesis
- BES Guide for Reproducible Code : https://www.britishecologicalsociety.org/wp-content/uploads/2019/06/BES-Guide-Reproducible-Code-2019.pdf
- paper Wilson 2017 “Good enough practices in scientific computing” : https://doi.org/10.1371/journal.pcbi.1005510
- Florian Schneider resources , e.g. : basic R intro : http://fdschneider.de/r_basics/
- books of Hadley Wickham
  - R for Data Science : https://r4ds.had.co.nz/
  - Advanced R : https://adv-r.hadley.nz/
  - tidyverse : https://www.tidyverse.org/
- Introduction Course to Data Science by Brian Enquist : https://www.researchgate.net/publication/310798573_How_to_think_About_Your_Data_Introduction_to_Data_Science_Management_what_they_don%27t_-_but_should_-_teach_you_about_the_scientific_method