



21 AUGUST 2018

Herman Teirlinck,
01.69 - Paul Janssen

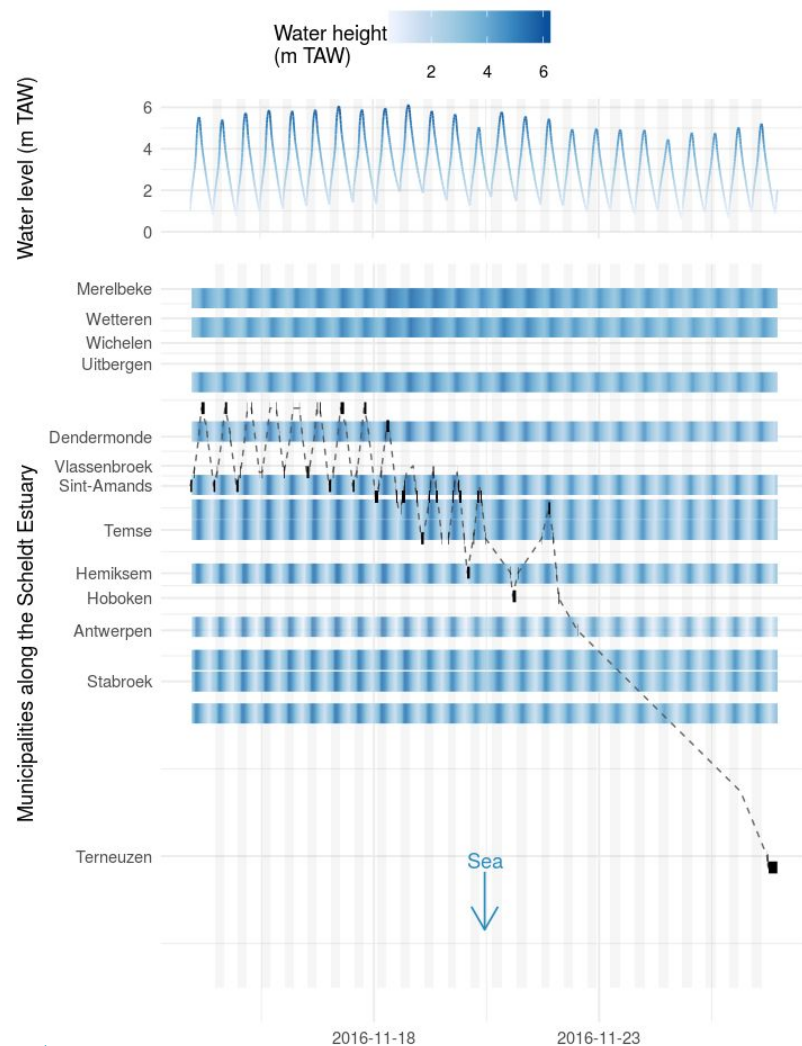
What have I done?!?

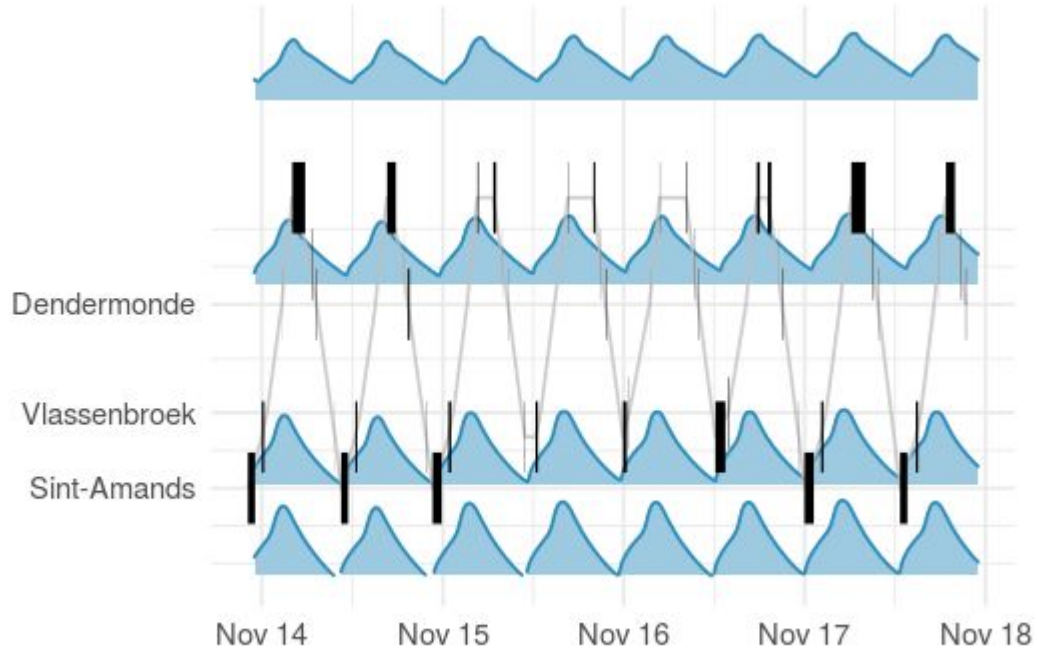
```
source("../files/plot_tidal_functions.R")

tidal_tracks <- plot_tidal_tracks(tidal_data_subset, eels_subset,
                                tide_periods_subset, date_breaks = "5 days")

tide_dendermonde <- plot_tide_with_background(
  tidal_data_subset %>% filter(station_name == "Dendermonde
  tij/Zeeschelde"), tide_periods_subset, date_breaks = "5
  days") +
  theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())

ggarrange(tide_dendermonde, tidal_tracks,
          ncol = 1, nrow = 2, common.legend = TRUE,
          align = "v", heights = c(1, 5))
```





tidyverse

lubridate

waterInfo

ggplot2

ggridges

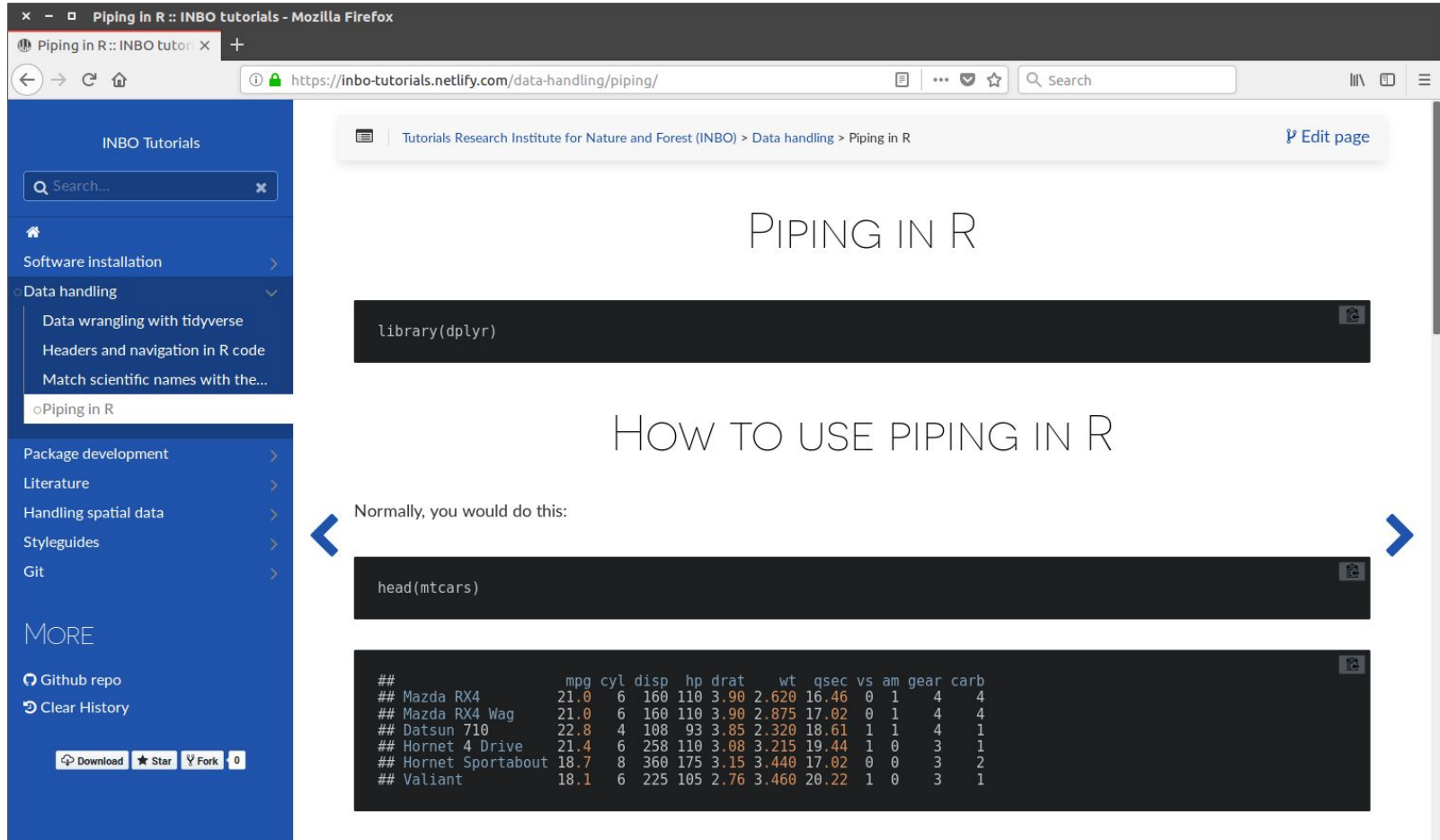
New tutorials available...

The screenshot shows a Mozilla Firefox browser window with the URL `https://inbo-tutorials.netlify.com/data-handling/`. The page title is "Data handling :: INBO tutorials". The left sidebar is blue and contains a search bar, a home icon, and a list of navigation items: "Software installation", "Data handling" (selected), "Data wrangling with tidyverse", "Headers and navigation in R code", "Match scientific names with the...", "Piping in R", "Package development", "Literature", "Handling spatial data", "Styleguides", and "Git". Below the sidebar is a "MORE" section with "Github repo" and "Clear History" buttons, and a GitHub repository footer with "Download", "Star", "Fork", and "0" buttons.

The main content area has a breadcrumb trail: "Tutorials Research Institute for Nature and Forest (INBO) > Data handling" and an "Edit page" link. The title "DATA HANDLING" is centered at the top. Below it is a list of four tutorial topics, each with a brief description and a code snippet.

- Data wrangling with tidyverse**
Real life datasources seldom provide data in exactly the format you need for the analysis. Hence most of the time you need to manipulate the data after reading it into R. There are several ways to do this, each with their pros and cons. We highly recommend the tidyverse collection of packages. The command `library(tidyverse)` will actually load the following packages: `ggplot2`, `dplyr`, `tidyr`, `readr`, `purrr`, `tibble`, `string` and `forcats`.
- Headers and navigation in R code**
Introduction R code can become elaborate and consequently unclear or difficult to navigate. Yet, it is possible to introduce headers and navigate through them. Creating sections manually To create a header of a section, different methods can be applied. Any comment line which includes at least four trailing dashes (-), equal signs (=), or hash tags (#) automatically creates a code section. # 1. Header 1 ##### # 2. Header 2 ---- # 3.
- Match scientific names with the GBIF backbone**
Introduction This tutorial will explain how you can match a list of 'scientific names' to the GBIF taxonomic backbone Important is that you have `rgbif` and `inborutils` installed and available: `library(tidyverse) # tidyverse library(rgbif) # To Match GBIF library(inborutils) # wrap GBIF api data library(knitr)` Read data file containing the scientific names Read file containing the scientific names you want to check against the GBIF taxonomic backbone: `species_list <- read_csv("https://raw.`
- Piping in R**
`library(dplyr)` How to use piping in R Normally, you would do this: `head(mtcars) ## mpg cyl disp hp drat wt qsec vs am gear carb ## Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4 ## Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4 ## Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1 ## Hornet 4 Drive 21.`

New tutorials available...



INBO Tutorials

Search...

Software installation >

Data handling >

- Data wrangling with tidyverse
- Headers and navigation in R code
- Match scientific names with the...

Piping in R

Package development >

Literature >

Handling spatial data >

Styleguides >

Git >

MORE

- Github repo
- Clear History

Download Star Fork 0

Tutorials Research Institute for Nature and Forest (INBO) > Data handling > Piping in R [Edit page](#)

PIPING IN R

```
library(dplyr)
```

HOW TO USE PIPING IN R

Normally, you would do this:

```
head(mtcars)
```

```
##           mpg  cyl  disp  hp drat   wt  qsec vs  am  gear carb
## Mazda RX4      21.0   6  160 110 3.90 2.620 16.46 0   1   4   4
## Mazda RX4 Wag  21.0   6  160 110 3.90 2.875 17.02 0   1   4   4
## Datsun 710      22.8   4  108  93 3.85 2.320 18.61 1   1   4   1
## Hornet 4 Drive  21.4   6  258 110 3.08 3.215 19.44 1   0   3   1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02 0   0   3   2
## Valiant        18.1   6  225 105 2.76 3.460 20.22 1   0   3   1
```



DEBUGGING

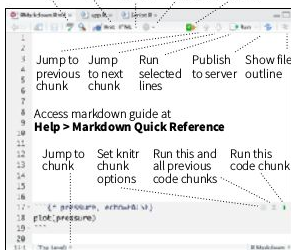
RStudio IDE :: CHEAT SHEET

Documents and Apps

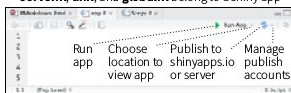


Open Shiny, R Markdown, knitr, Sweave, LaTeX, Rd files and more in Source Pane

Check spelling
Render output
Choose output format
Choose output location
Insert code chunk

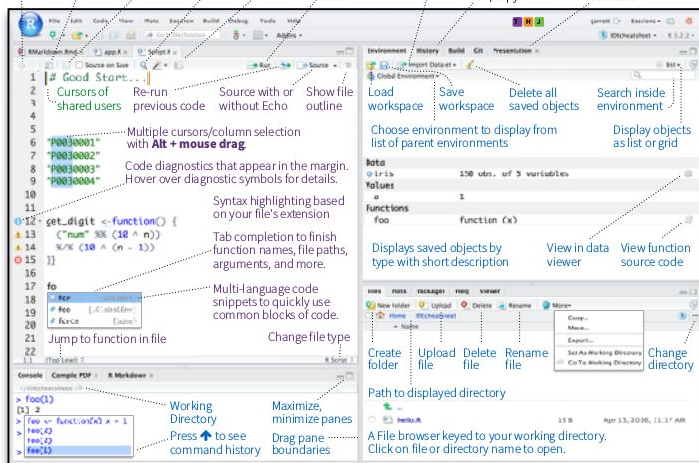


RStudio recognizes that files named **app.R**, **server.R**, **ui.R**, and **global.R** belong to a shiny app



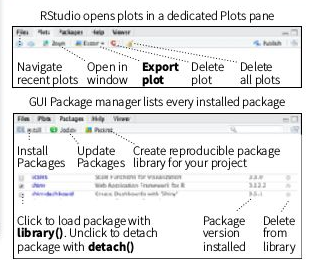
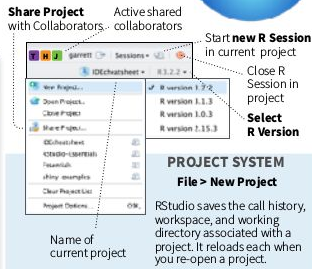
Write Code

Navigate tabs
Open in new window
Save
Find and replace
Compile as notebook
Run selected code



R Support

Pro Features

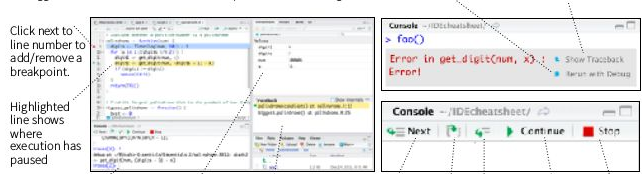


Debug Mode

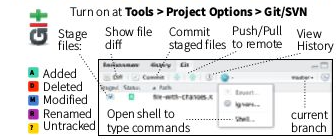
Open with **debug()**, **browser()**, or a breakpoint. RStudio will open the debugger mode when it encounters a breakpoint while executing code.

Launch debugger mode from origin of error

Open traceback to examine the functions that R called before the error occurred



Version Control with Git or SVN



Package Writing



RStudio IDE :: CHEAT SHEET



Documents and Apps

Open Shiny, R Markdown, knitr, Sweave, LaTeX, Rd files and more in Source Pane

Check spelling
 Render output
 Choose output format
 Choose output location
 Insert code chunk

Jump to previous chunk
 Jump to next chunk
 Run selected lines
 Publish to server
 Show file outline

Access markdown guide at **Help > Markdown Quick Reference**

Jump to chunk
 Set knitr options
 Run this code chunk
 Run all previous code chunks

RStudio recognizes that files named **app.R**, **server.R**, **ui.R**, and **global.R** belong to a shiny app

Run app
 Choose location to view app
 Publish to shinyapps.io or server
 Manage publish accounts

Write Code

Navigate tabs
 Open in new window
 Save
 Find and replace
 Compile as notebook
 Run selected code

Cursors of shared users
 Re-run previous code
 Source with or without Echo
 Show file outline

Multiple cursors/column selection with **Alt + mouse drag**
 Code diagnostics that appear in the margin. Hover over diagnostic symbols for details.
 Syntax highlighting based on your file's extension
 Tab completion to finish function names, file paths, arguments, and more.

Multi-Language code snippets to quickly use common blocks of code.
 Change file type

Working Directory
 Maximize, minimize panes
 Press **↑** to see command history
 Drag pane boundaries

R Support

Import data with wizard
 History of past commands to run/copy
 Display RPres slideshows
File > New File > R Presentation

Load workspace
 Save workspace
 Delete all saved objects
 Search inside environment

Choose environment to display from list of parent environments
 Display objects as list or grid

Displays saved objects by type with short description
 View in data viewer
 View function source code

Create folder
 Upload file
 Delete file
 Rename file
 Change directory

Path to displayed directory
 A file browser keyed to your working directory. Click on file or directory name to open.

Pro Features

Share Project with Collaborators
 Active shared collaborators

Start **new R Session** in current project
 Close R Session in project
Select R Version

PROJECT SYSTEM

File > New Project

RStudio saves the call history, workspace, and working directory associated with a project. It reloads each when you re-open a project.

Name of current project

RStudio opens plots in a dedicated Plots pane

Navigate recent plots
 Open in window
 Export plot
 Delete plot
 Delete all plots

GUI Package manager lists every installed package

Install Packages
 Update Packages
 Create reproducible package library for your project

Click to load package with **library()**. Unlick to detach package with **detach()**
 Package version installed
 Delete from library

Debug Mode

Open with **debug()**, **browser()**, or a breakpoint. RStudio will open the debugger mode when it encounters a breakpoint while executing code.

Click next to line number to add/remove a breakpoint.
 Highlighted line shows where execution has paused

Launch debugger mode from origin of error
 Open traceback to examine the functions that R called before the error occurred

Run commands in environment where execution has paused
 Examine variables in executing environment
 Select function in traceback to debug
 Step through code one line at a time
 Step into and out of functions to run
 Resume execution mode
 Quit debug

Version Control with Git or SVN

Turn on at **Tools > Project Options > Git/SVN**

Stage files
 Show file diff
 Commit staged files
 Push/Pull to remote
 View History

Added
 Deleted
 Modified
 Renamed
 Untracked

Open shell to type commands
 current branch

Package Writing

File > New Project > New Directory > R Package
 Turn project into package, Enable roxygen documentation with **Tools > Project Options > Build Tools**
 Roxygen guide at **Help > Roxygen Quick Reference**

RStudio opens documentation in a dedicated Help pane

Home page of helpful links
 Search within help file
 Search for help file

Viewer Pane displays HTML content, such as Shiny apps, RMarkdown reports, and interactive visualizations

Stop Shiny app
 Publish to shinyapps.io, rpubs, RSConnect, ...
 Refresh

View(<data>) opens spreadsheet like view of data set

Filter rows by value or value range
 Sort by values
 Search for value



Debug Mode

Open with **debug()**, **browser()**, or a breakpoint. RStudio will open the debugger mode when it encounters a breakpoint while executing code.

Click next to line number to add/remove a breakpoint.

Highlighted line shows where execution has paused

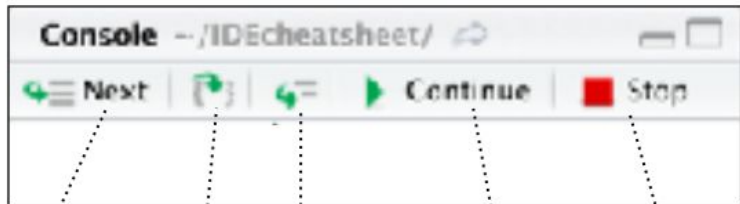
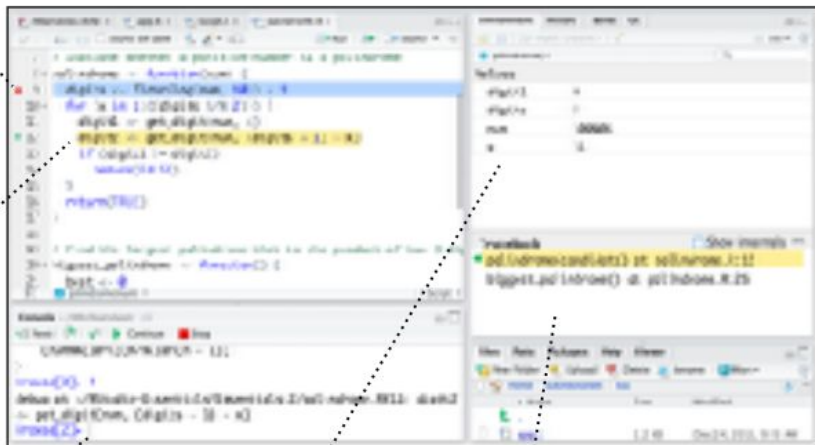
Run commands in environment where execution has paused

Examine variables in executing environment

Select function in traceback to debug

Launch debugger mode from origin of error

Open traceback to examine the functions that R called before the error occurred



Step through code one line at a time

Step into and out of functions to run

Resume execution mode

Quit debug


Share your snippets during the coding session!

Go to <https://hackmd.io/qn1X6GFATLiOQjvN96KENA> and post your code in between backticks:

For example:

```
```\n\nlibrary(lubridate)\n\nmy_data <- ... \n\n```
```

# The concept

We defined a number of challenges. If you were able to achieve a challenge, add a  to your laptop screen.

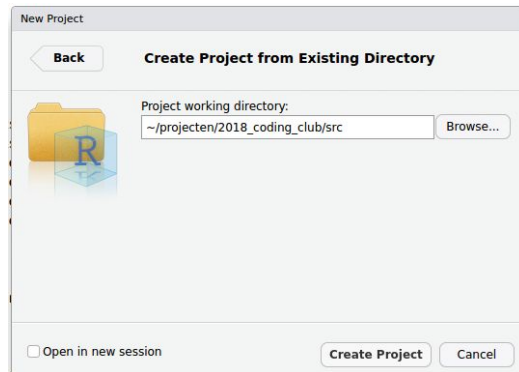
The objective is that **everyone** achieves !

- Someone has more  than you? **Ask for help!**
- Someone has less  than you? **Provide help!**

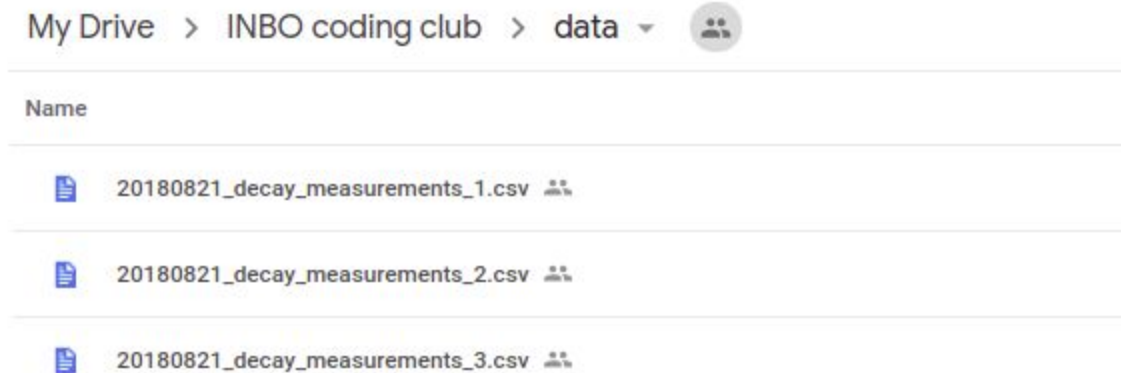
- Download coding club material and work locally, not in sync with the Google drive



- Create new Rstudio project in the **/src** folder



- Download coding club material and work locally, not in sync with the Google drive
- Create new Rstudio project in the **src** folder...
- Use relative paths to data files!



Introduction... some live coding!




# Remember...

WHY?

Adding `print` alone won't do the job...

Jump into `for/while` loops, `if/else` constructors and `functions`.

HOW?

- add **breakpoints**  in Rstudio and use '`source`' instead of '`Run`'
- in longer scripts (although short scripts are advised!) and Rmarkdown files:
  - Add `browser()` and use ``c`` (ontinue), ``n`` (ext line) and ``Q`` (uit)

- 
- Download the `20180821_decay_measurements_x.csv` (met x 1 tot 3) files from [data folder](#)
  - Download the file [20180821\\_challenge\\_1.R](#) and try to run the file

```
library(tidyverse)
```


```
files_in_dir <- list.files("../data", full.names = TRUE)
```

```
for (file in files_in_dir) {
 if (stringr::str_detect(file, "decay")) {
 # read the data
 concentrations <- readr::read_csv(file)

 # make and print a plot of the data
 plot_conc <- ggplot(concentrations) +
 geom_point(aes(time, conc_data), size = 2) +
 xlab("time (s)") + ylab("concentration (mg/l)") +
 ggtitle(basename(file))

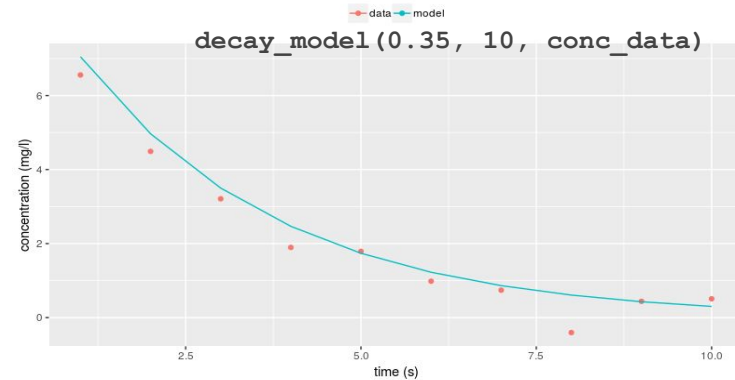
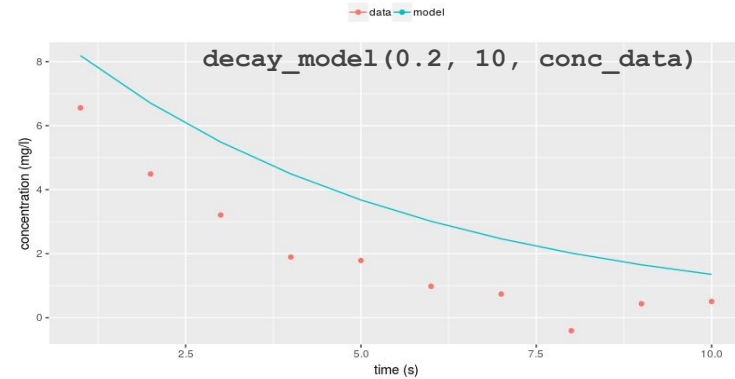
 print(plot_conc)

 }
}
```

- Try out the different ways of debugging (using  breakpoints, browser,...)
- What is the cause of the Error?
- Together **with your neighbour** :
  - Discuss how you would solve this issue
  - Explain/Add your suggestion/code to the [Hackmd](#)... \*

\* feel free to *tidyverse* and/or exclude-for-loop the functionality as well...

- Download the file [20180821\\_challenge\\_2.R](#) and run the file
- The function `decay_model` calculates the model of a pollutant decay by micro-organisms. Using the input data, the function calculates the *Sum of Squared Errors* (SSE) between the model and the data and creates a plot comparing the model and the data.
  - Run the model with the `20180821_decay_measurements_3.csv` data file
  - There is no error, but the SSE value is not calculated correctly.... **Debug** the functions and provide a solution in the appropriate function.
  - [Read the documentation](#) about the `debugonce()` command and try it out yourself.





Using my experimental values, I want to estimate the *decay rate* by my micro-organisms (cfr. efficiency). I can do this by optimizing my model (changing the decay rate parameter in order to minimize the distance between the model and the data, i.e. a minimal SSE).

```
Estimate the optimal parameter
optimized <- optim(0.3, decay_model, init = 10, data_conc = conc_data,
 method = "Brent", lower = 0.2, upper = 0.5)

Rerun with the optimal value...
decay_model(optimized$par, 10, conc_data)
```

This is actually running slow... In order to identify the bottleneck, we can use the profiler. [The profiler](#) is a tool for helping you to understand how R spends its time. It basically works as follows:

```
library(profvis)
profvis({ YOUR CODE })
```

**Identify the main reason the code is running so slowly!**





Zaal: Herman Teirlinck - 01.69 - Paul Janssen

Datum: 2018-09-20, van 10:00 tot 12:00

*(registration announced via [DG\\_useR@inbo.be](mailto:DG_useR@inbo.be))*