Open and Reproducible Science in R and Rstudio

Niklas Hohmann

N.H.Hohmann@uu.nl

Schedule

- Structure (1h)
- Code and computation (1h)
- Licenses and publication (1h)
- Peer-feedback and advanced topics (1h)

Short lecture (10 mins) and plenty of time to work on code (30 mins)

Why Open Science?

- The right thing to do: methodological rigor and reproducibility
- Ease of review
- Useful skill
- Will become more important in the future
- Good learning experience
- Increased adaptation of your work

Part I: Structure

Keep your computational environment

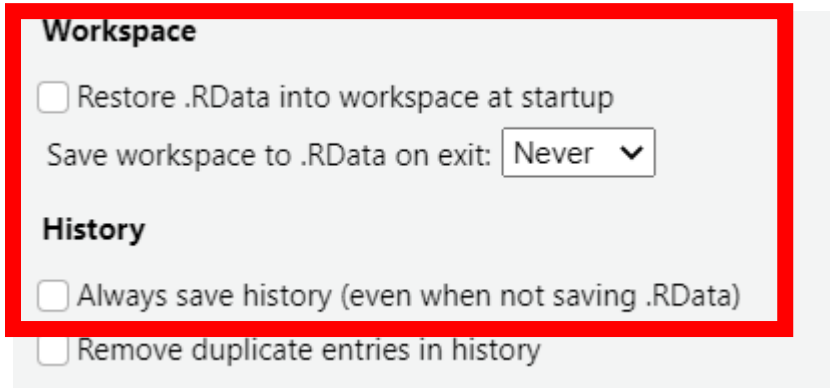
- Clean
- Ordered
- Labeled
- Documented
- Re-usable

Jean-Pierre from Cosne-Cours-sur-Loire (Nièvre, Burgundy region), France, CC BY-SA 2.0

Start from a clean slate

Do not save your workspace!

Tools > global options > general



If your code analysis runs long

- Save intermediate results using `save`
- Load results using `load`
- Make a conscious choice which data is in your workspace

https://datacarpentry.github.io/semester-biology/materials/basic-reproducibility-R/

Absolute and Relative Paths

```
setwd("C:/users/JohnDoe/science/project/analysis")
# or
x = read.csv("C:/users/JohnDoe/science/project/analysis/data.csv")
# or
## go to session > set working directory and set the
## directory to the place where the data is stored
```

Works only on one machine!

Absolute paths work only on your computer!
No manual change should be required by the user

Solution: Rprojects

- Associated with Rstudio
- Sets all paths relative to the Rproject file
- Very useful for advanced applications (e.g. reproducible environments, packages etc.)

https://support.posit.co/hc/en-us/articles/200526207-Using-RStudio-Projects

Working with Rprojects

Create new project

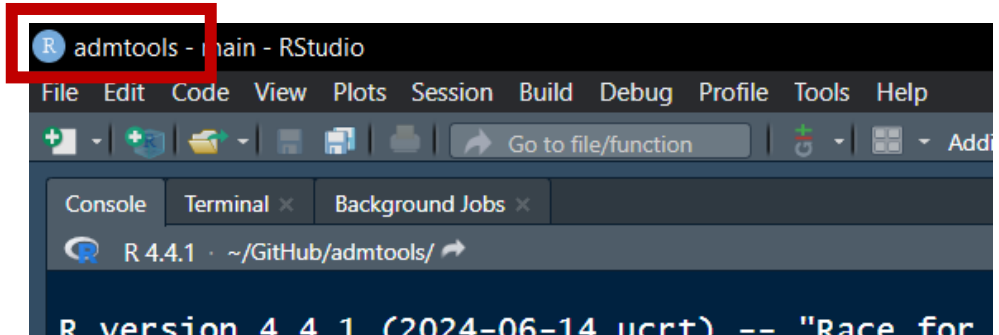
Rstudio > File > new project > existing directory > *select folder where your code is located*

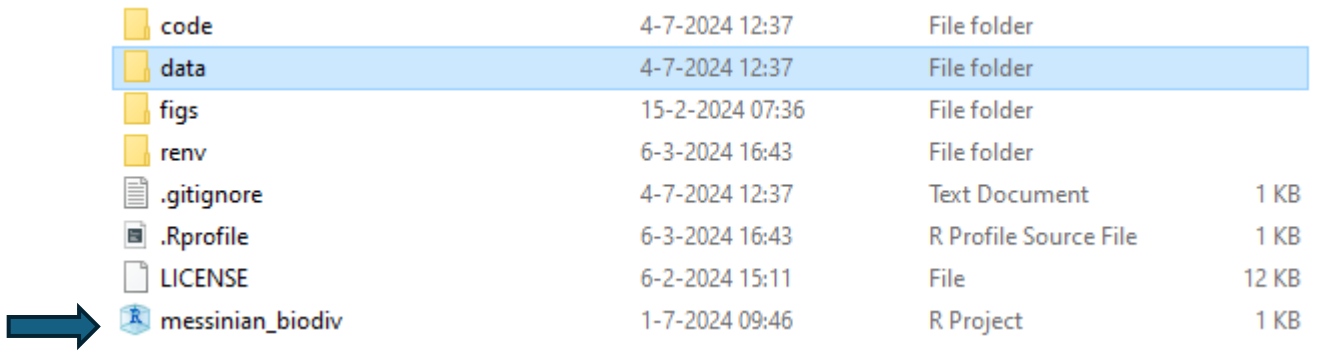
Working within an existing project

Rstudio > File > open project > *select the project*

In a project, all paths are relative to the .Rproj file

Project name





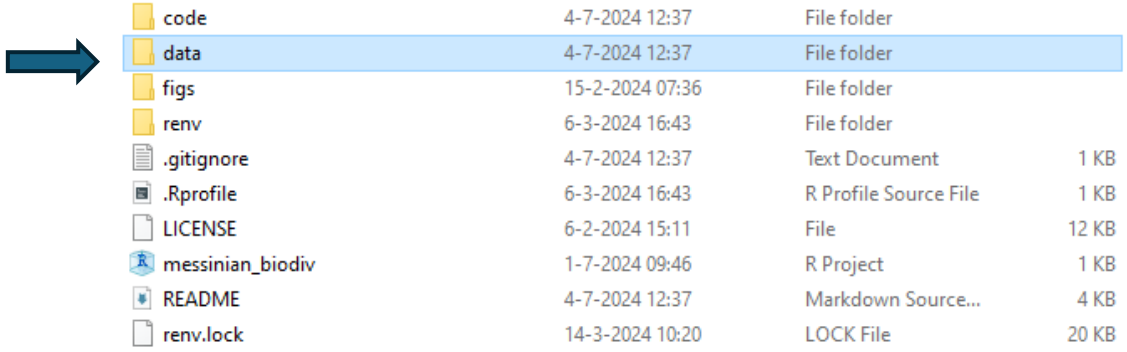
https://r4ds.hadley.nz/workflow-scripts.html

Folder Structure

Store parts of your analysis in folders with meaningful names

Commonly:

- `code`
- `figs`
- `data`



Potentially:

`data/raw` for **read only** data

Hohmann, N., & Agiadi, K. (2024). Supplementary code for "Late Miocene transformation of Mediterranean Sea biodiversity" (v1.2.0). Zenodo. https://doi.org/10.5281/zenodo.12654068

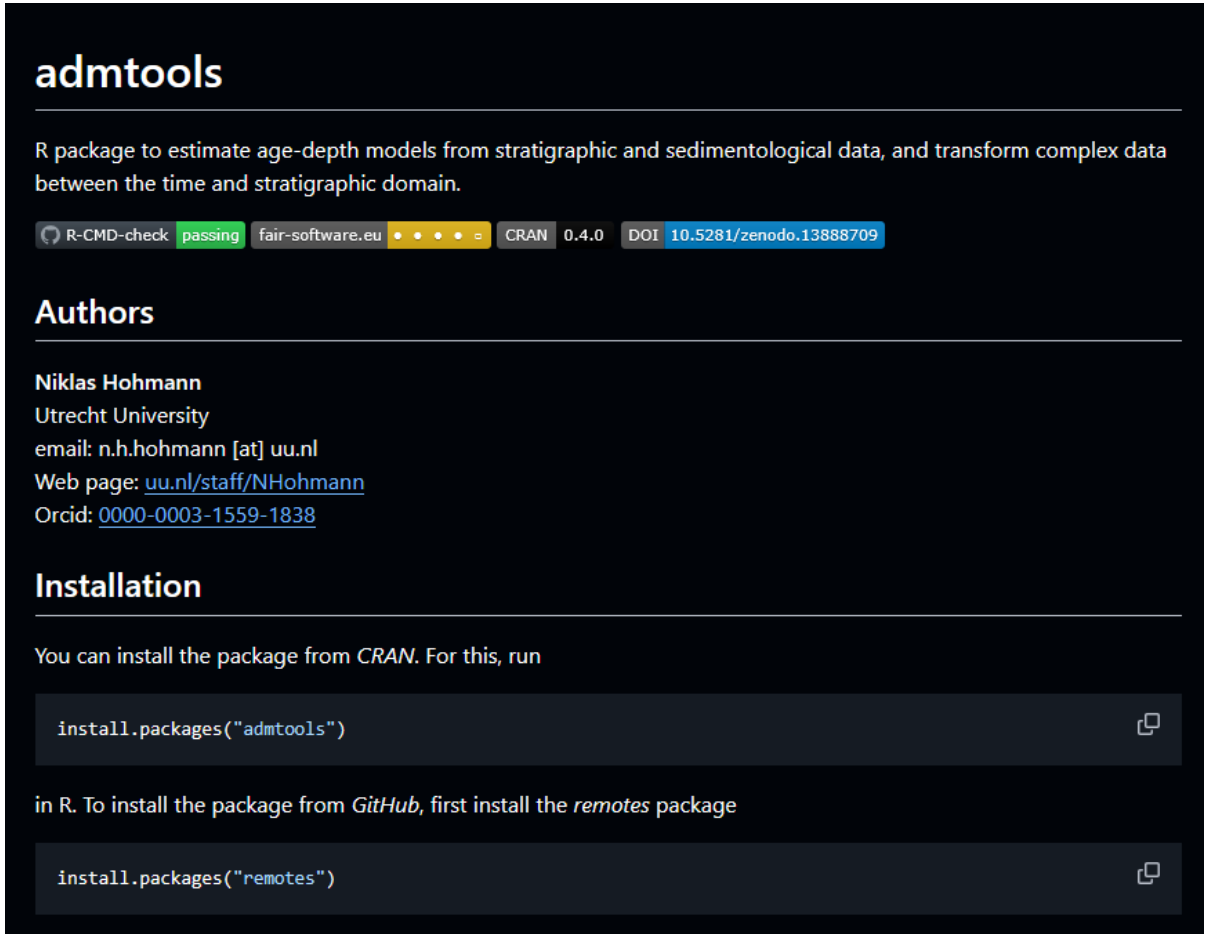
README

The README is the landing page of your project

It should contain:

- Aim of the code
- Authors (incl. contact details and identifiers)
- Software requirements
- Running instructions
- Funding sources (if applicable)
- Folder structure
- (License and citation -> more later)

The README is your lab notebook!
Keep it up to date & informative.



admtools

R package to estimate age-depth models from stratigraphic and sedimentological data, and transform complex data between the time and stratigraphic domain.

R-CMD-check passing | fair-software.eu | CRAN 0.4.0 | DOI 10.5281/zenodo.13888709

Authors

Niklas Hohmann
Utrecht University
email: n.h.hohmann [at] uu.nl
Web page: uu.nl/staff/NHohmann
Orcid: 0000-0003-1559-1838

Installation

You can install the package from *CRAN*. For this, run

```
install.packages("admtools")
```

in R. To install the package from *GitHub*, first install the *remotes* package

```
install.packages("remotes")
```

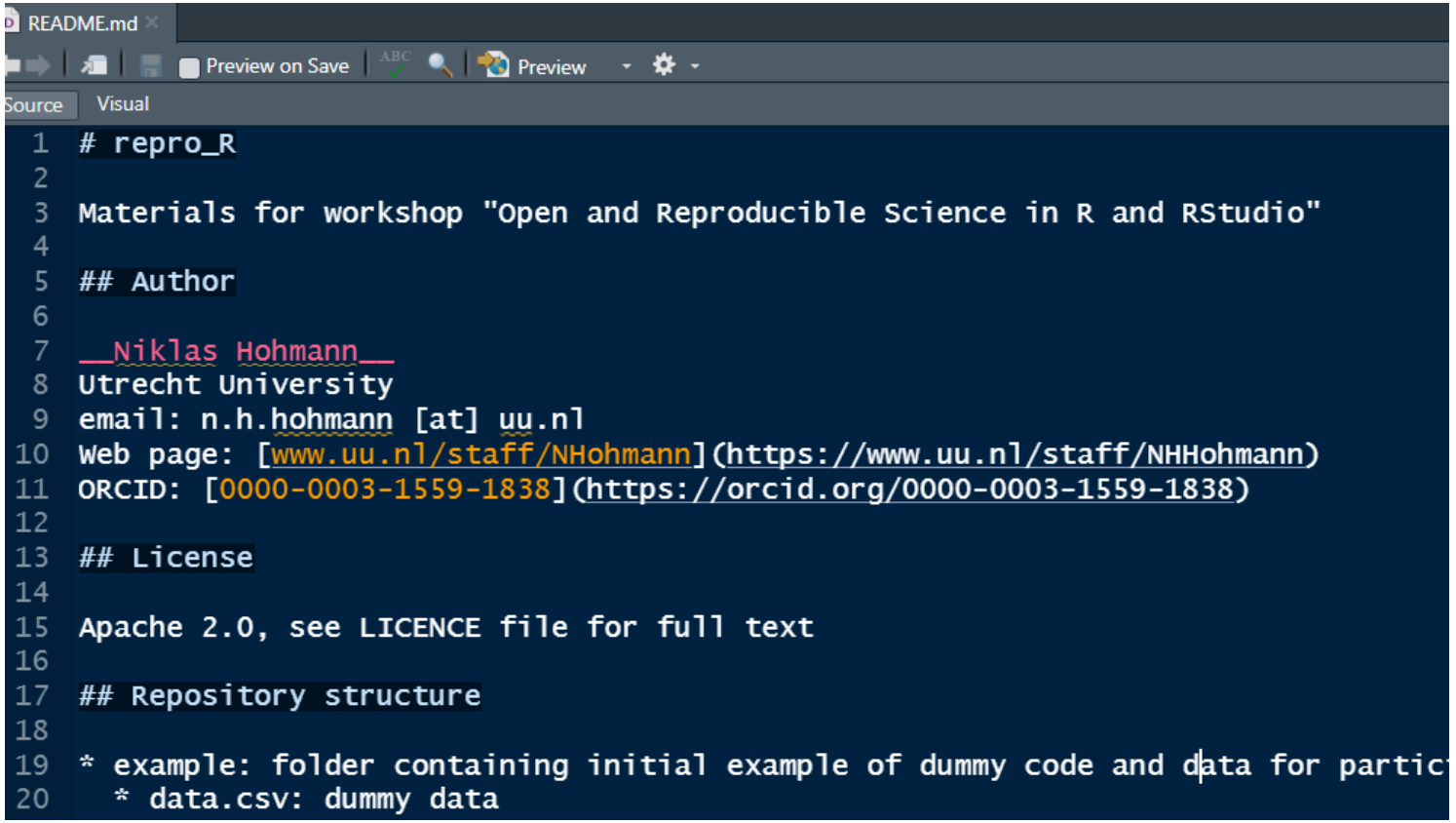
Writing a README

.txt or .md file

File > new file > markdown file

Formatting guidelines:

https://www.markdownguide.org/basic-syntax/



Task (30 mins)

- Make sure no workspace is saved
- Create an Rproject
- Replace all absolute with relative paths
- Create reasonable folder structure
- Save figures in figs folder, store intermediate data if necessary
- Write a README
- Check if everything runs smoothly

Coffee break!

Part II: Code

- Coding styles
- Variable names
- Functions
- Randomness
- Magic numbers
- Reproducible environments

Coding style

styler 1.0.0

Photo by Heng Films

2017/12/20

Lorenz Walthert, Kirill Müller

We're pleased to announce the release of styler 1.0.0. styler is a source code formatter - a package to format R code according to a style guide. It defaults to our implementation of the tidyverse style guide, but there is plenty of flexibility for a user to specify their own style. A coherent style is important for consistency and legibility. Just as it is important to putspcesbetweenwords. You can install styler from CRAN:

```
install.packages("styler")
```

https://www.tidyverse.org/blog/2017/12/styler-1.0.0/

- A consistent coding style increases readability
- Install `styler` package
- Restart R session
- Addins > style

Variable names

Use meaningful variable names with a consistent naming convention (e.g., snake_case, camelCase)

```
2  # no idea what this code does
3  for (i in x){
4    f(i)
5  }
6
```

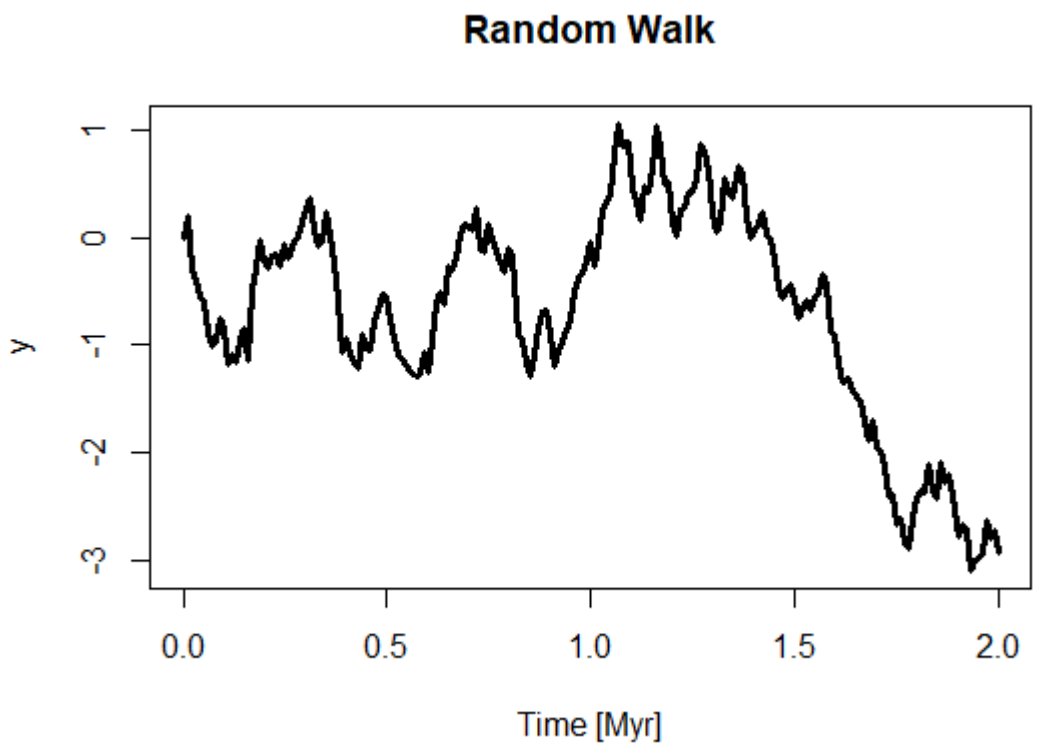
Bad – no context provided. This code could do anything

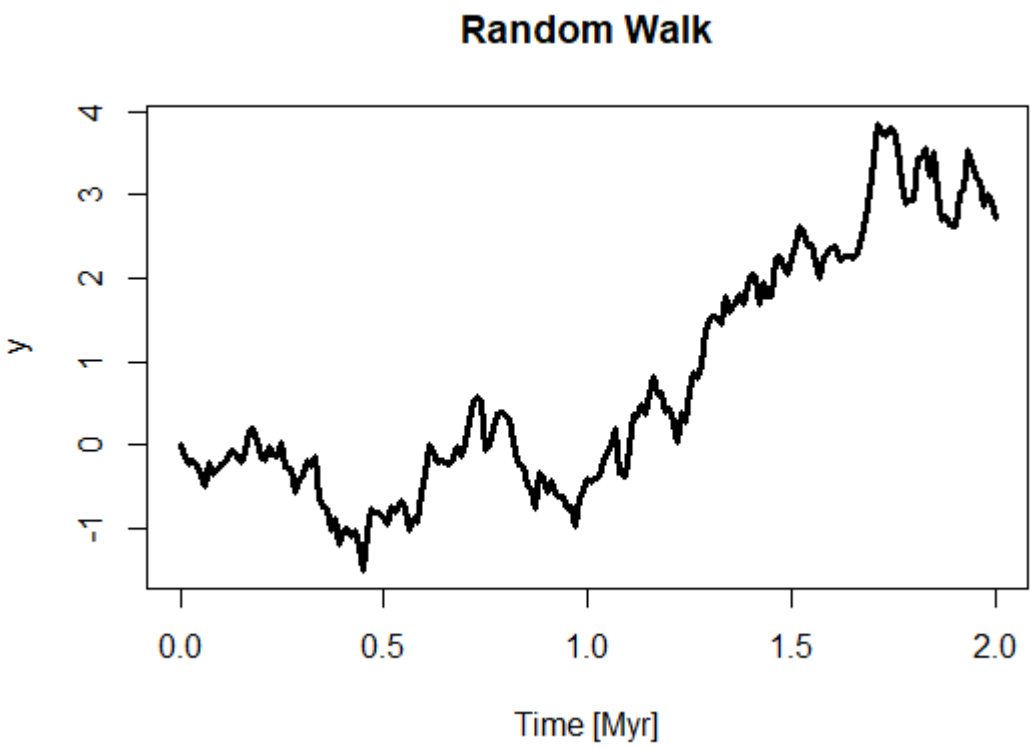
```
7  # much better
8  # determines niches of taxa
9  for (taxon in taxa_list){
10   get_niche(taxon)
11 }
```

Good – context clear:
Determines the niches of a list of taxa

https://curc.readthedocs.io/en/latest/programming/coding-best-practices.html

Random numbers





Identical code, different results! -> plot relies on the random number generator

Random numbers

If your code relies on random numbers, set a random seed!

```
13  # choose some seed
14  my_seed <- 42
15  set.seed(my_seed)
```

Dependency on the RNG might not be obvious!

https://blog.djnavarro.net/posts/2023-12-27_seedcatcher/

Magic numbers

Magic numbers = numeric values with no context

```
17 for (i in 1:5000){
18   sample(x, replace = TRUE)
19 }
20 # 5000 is magic number
21
```

Bad!
Why 5000?
What is the meaning?

```
bootstrap_replicates <- 5000
for (i in 1:bootstrap_replicates){
  sample(x, replace = TRUE)
}
# 5000 is magic number
```

Good!
Turn magic numbers into named variables/constants
-> Easy & global adjustment of analyses

https://en.wikipedia.org/wiki/Magic_number_(programming)

Functions

Functions perform well-defined computational tasks

Use functions if you

- Regularly copy & paste code
- Write the same code more than three times

Functions modularize code, making it more robust and readable!

Syntax

```
28 add_numbers = function(x1, x2){
29   y = x1 + x2
30   return(y)
31 }
```

`add_numbers`: function name
`x1, x2`: arguments
`y`: return value

Use descriptive function names!

https://r4ds.had.co.nz/functions.html

Documenting functions


```r
add_numbers = function(x1, x2){
  #' @description
    #' add two numbers
  #' @param x1 first number
  #' @param x2 second number
  #' @returns a number, sum of `x1` and `x2`
  y = x1 + x2
  return(y)
}
```

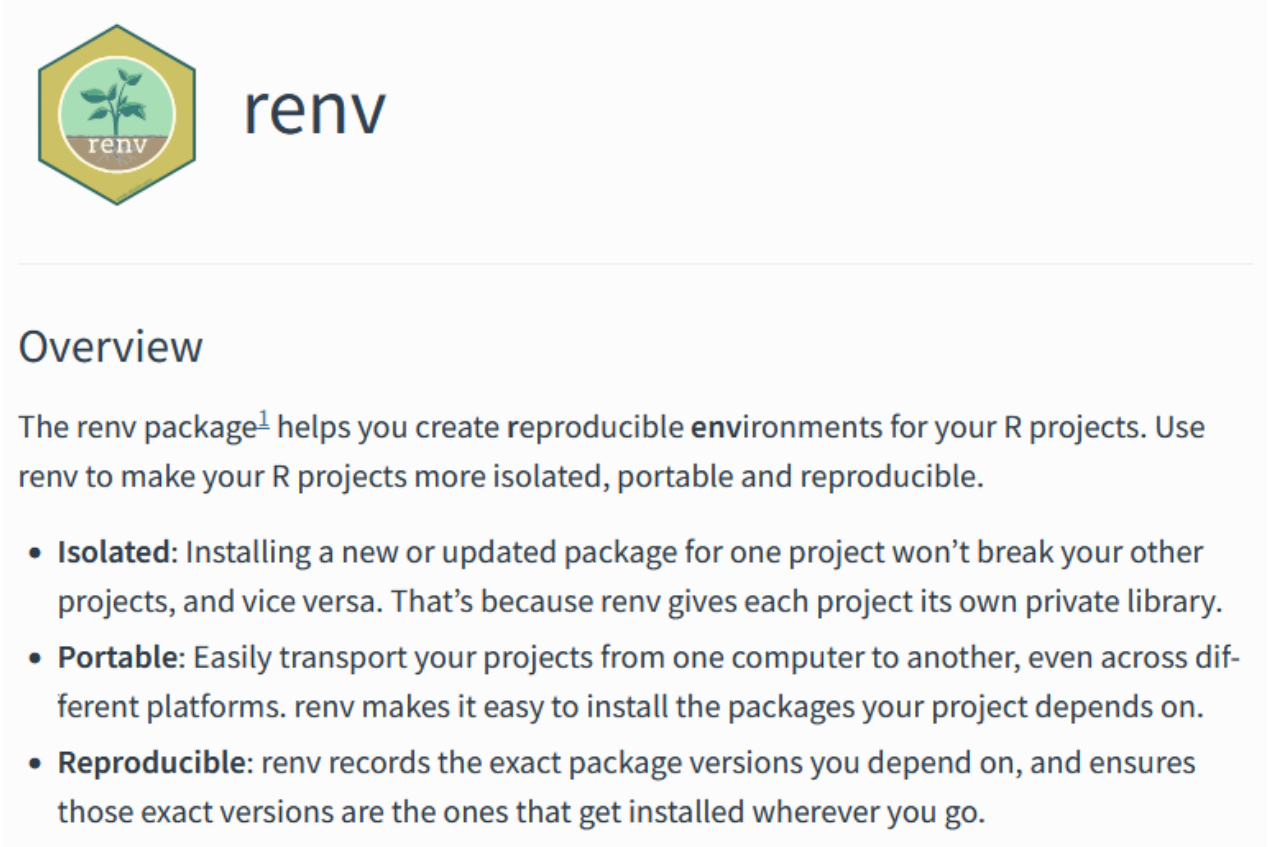
Documenting functions using docstrings
Commonly used for packaging

https://roxygen2.r-lib.org/

https://en.wikipedia.org/wiki/Docstring

Reproducible environments

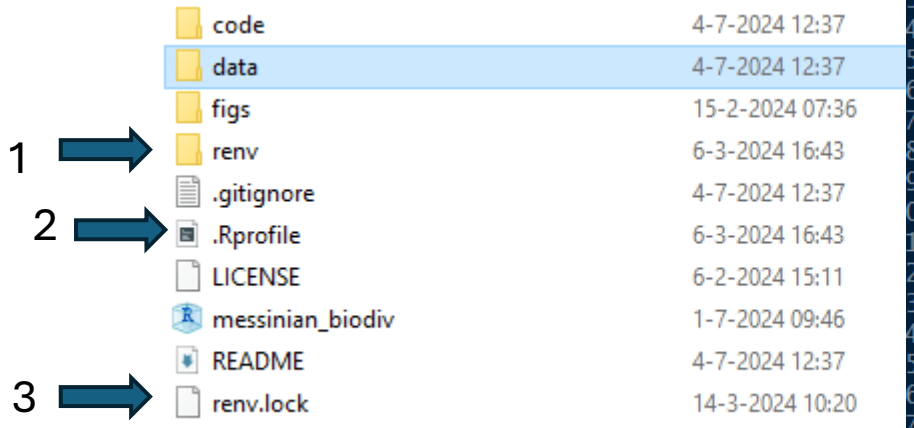
Problem: Different R versions and package versions on different systems

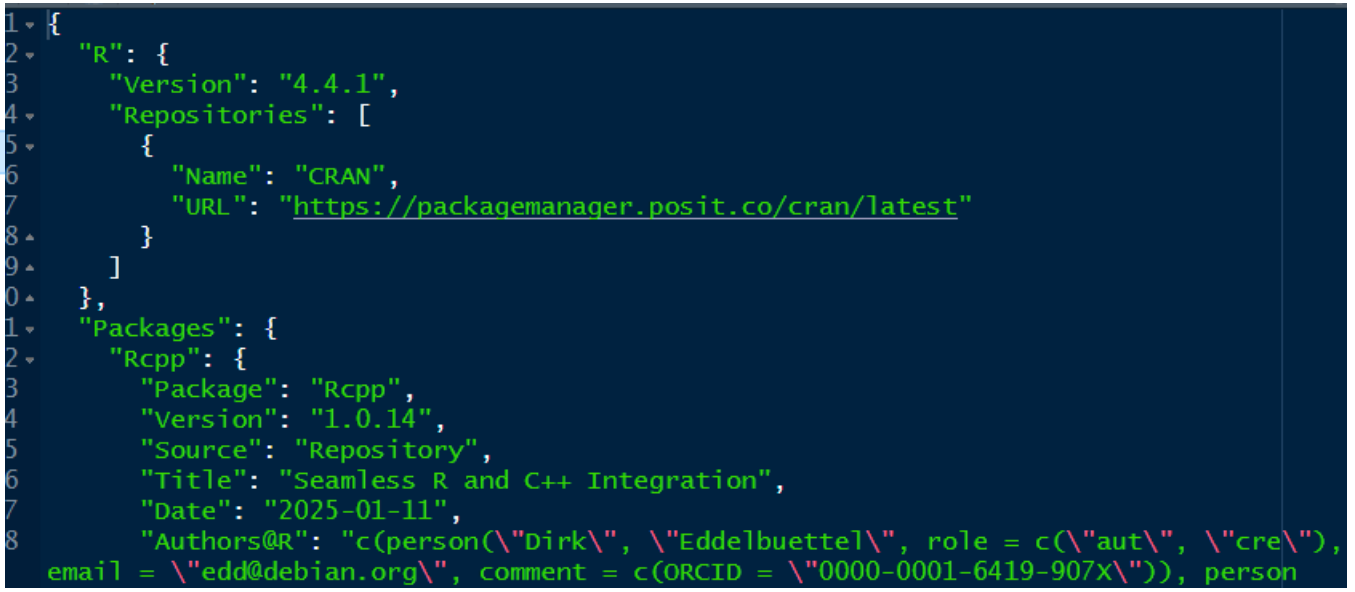


https://rstudio.github.io/renv

Working with `renv`: Getting started

```
install.packages("renv")
renv::init()
```



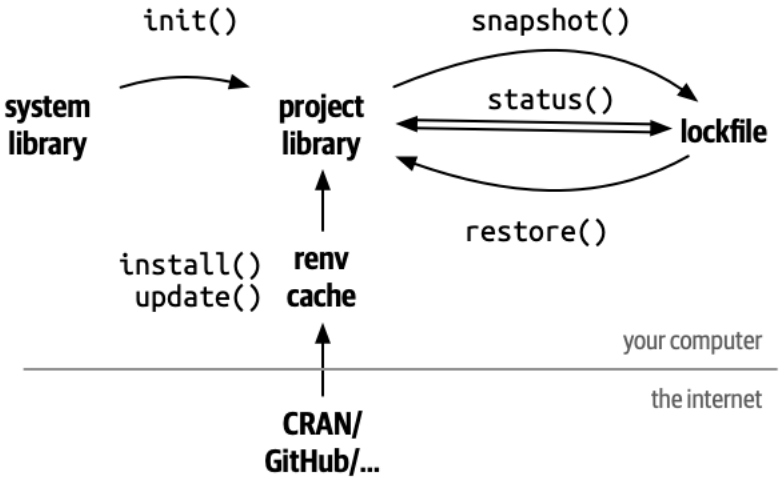


renv.lock

1: Folder with settings and code
2: R session settings – installs `renv` if necessary
3: Lock file – documents package versions

Working with `renv`

1. Install packages as usual
2. `renv::snapshot()` to record packages
3. `renv::status()` to check status



Using projects with renv:

`renv::restore()` to install all packages recorded in lockfile

-> No more need for `install.packages("packagename")` in code!

https://rstudio.github.io/renv

Tasks (30 mins)

- Use a consistent code style
- Introduce meaningful variable names
- Set a random seed at the beginning of your script
- Remove all magic numbers
- Replace duplicate code blocks by functions
- Start reproducible environment using `renv`
- Update your README
- Make sure everything works as intended

Coffee break!

Part III: Publishing

- Citing software
- Licenses
- Archiving and publishing

Citing software

Please cite all used software packages in your publications – a lot of work goes into them.

Write a code availability statement for your papers, using both a reference and doi:

“All code used in the analysis is available under <doi> (Author, year)”

might vary by journal

Creative Commons licenses

The Licenses

Attribution
CC BY

This license lets others distribute, remix, tweak, and build upon your work, even commercially, as long as they credit you for the original creation. This is the most accommodating of licenses offered. Recommended for maximum dissemination and use of licensed materials.

View License Deed | View Legal Code

Attribution-ShareAlike
CC BY-SA

This license lets others remix, tweak, and build upon your work even for commercial purposes, as long as they credit you and license their new creations under the identical terms. This license is often compared to "copyleft" free and open source software licenses. All new works based on yours will carry the same license, so any derivatives will also allow commercial use. This is the license used by Wikipedia, and is recommended for materials that would benefit from incorporating content from Wikipedia and similarly licensed projects.

View License Deed | View Legal Code

Attribution-NoDerivs
CC BY-ND

This license allows for redistribution, commercial and non-commercial, as long as it is passed along unchanged and in whole, with credit to you.

View License Deed | View Legal Code

Attribution-NonCommercial
CC BY-NC

This license lets others remix, tweak, and build upon your work non-commercially, and although their new works must also acknowledge you and be non-commercial, they don't have to license their derivative works on the same terms.

View License Deed | View Legal Code

Attribution-NonCommercial-ShareAlike
CC BY-NC-SA

This license lets others remix, tweak, and build upon your work non-commercially, as long as they credit you and license their new creations under the identical terms.

View License Deed | View Legal Code

Attribution-NonCommercial-NoDerivs
CC BY-NC-ND

This license is the most restrictive of our six main licenses, only allowing others to download your works and share them with others as long as they credit you, but they can't change them in any way or use them commercially.

View License Deed | View Legal Code

https://creativecommons.org/

Common for data, text, and visuals

Software licenses

Specialized for source code

Examples:

- GPL3.0
- Apache 2.0
- MIT

https://choosealicense.com/

Check with your funding agency/industry partner/data manager

No license, no reuse!

Permissive choice:
Apache 2.0 for code,
CC-BY 4.0 for data, text, and figures

Publishing and archiving your code

Many options, here: Zenodo – GitHub integration

Zenodo

- Domain agnostic data hoster
- Mints **doi**
- Rich metadata support
- Hosted at CERN
- Easy GitHub integration

GitHub

- Microsoft owned platform
- Largest code sharing platform in the world
- Easy publication of code

Metadata

Add all available metadata!

- Funders (doi)
- Authors & affiliations (Orcid)
- Keywords
- Programming language
- Links to publications
- The more the better

Example:
https://zenodo.org/records/13358742

Tasks (30 mins)

- Publish your code on Zenodo using the Fairly FAIR code instructions: https://github.com/PGijsbers/fairly-fair
- Add the citation info and doi to your README

Congratulations! Your code is now FAIR, open, archived, and citable!

Barker et al. (2022) Introducing the FAIR principles for research software. https://doi.org/10.1038/s41597-022-01710-x

Coffee break!

Part IV: Peer-feedback (30 mins)

- In the breakout rooms, exchange your code projects
- Provide feedback on the reproducibility – what worked, what did not work?
- Does the code run and produce the desired results?
- Is the structure logical and the README clear?

Example solution

Live demonstration

Advanced Topics: Version Control

The problem:

Files like

`My_codev3_3_NH_Final_submission.R`

The solution:

- Version control systems, e.g. git

1. Track changes
2. Restore previous versions
3. Work with alternate versions

https://git-scm.com/

https://git-scm.com/book/ms/v2

Advanced Topics: Collaboration on Code

GitHub provides functionality for

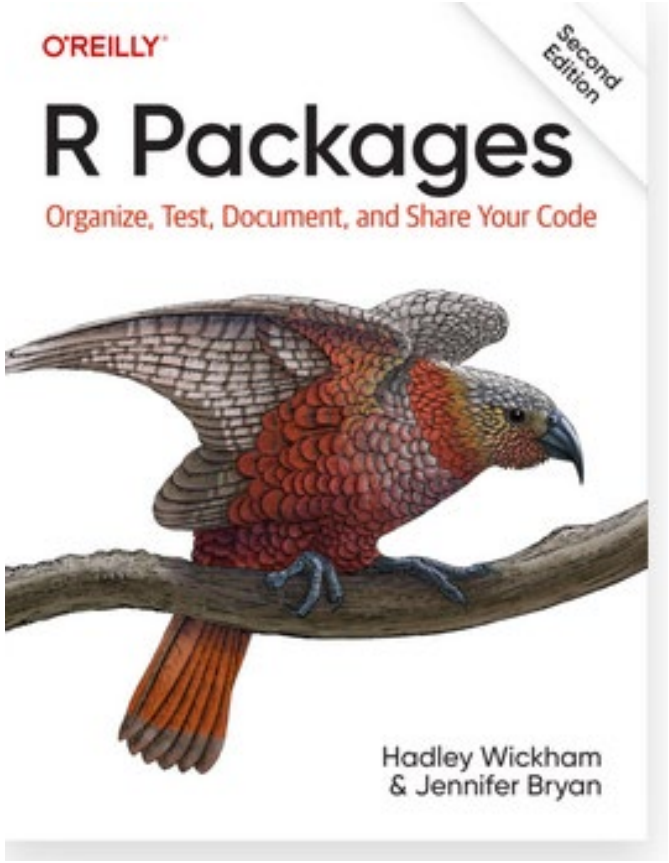
- Project management
- Bug reports
- Code sharing
- Discussion
- Publishing
- Etc.

Focus on collaboration!

https://github.com

Braga et al. (2023) https://doi.org/10.1111/2041-210X.14108

Advanced Topics: Packaging



Book available for free
https://r-pkgs.org/

Packages are portable sets of code with well-defined functionality

Package if

- your code might be useful to the community
- You reuse code across multiple research projects

My recommendation: package early, package often

Packages are research outputs/publications!

Wrap up

- Don’t be afraid to share your code & interact with other people's code
- Open science is not binary - do the best you can
- Keep an open mind
- There are plenty of resources available online (some listed on the last slide)

Questions/comments?

N.H.Hohmann@uu.nl

Free books

- R for data science: https://r4ds.hadley.nz/
- Advanced R: https://adv-r.hadley.nz/
- R packages: https://r-pkgs.org/
- The Turing Way: https://book.the-turing-way.org/