# 6 Steps Towards Reproducible Research

Heidi Seibold

# Table of contents

# Overview

This booklet helps you implement impactful steps in **making your research reproducible** (and open).

What will it cover? The image below sums it up quite nicely:

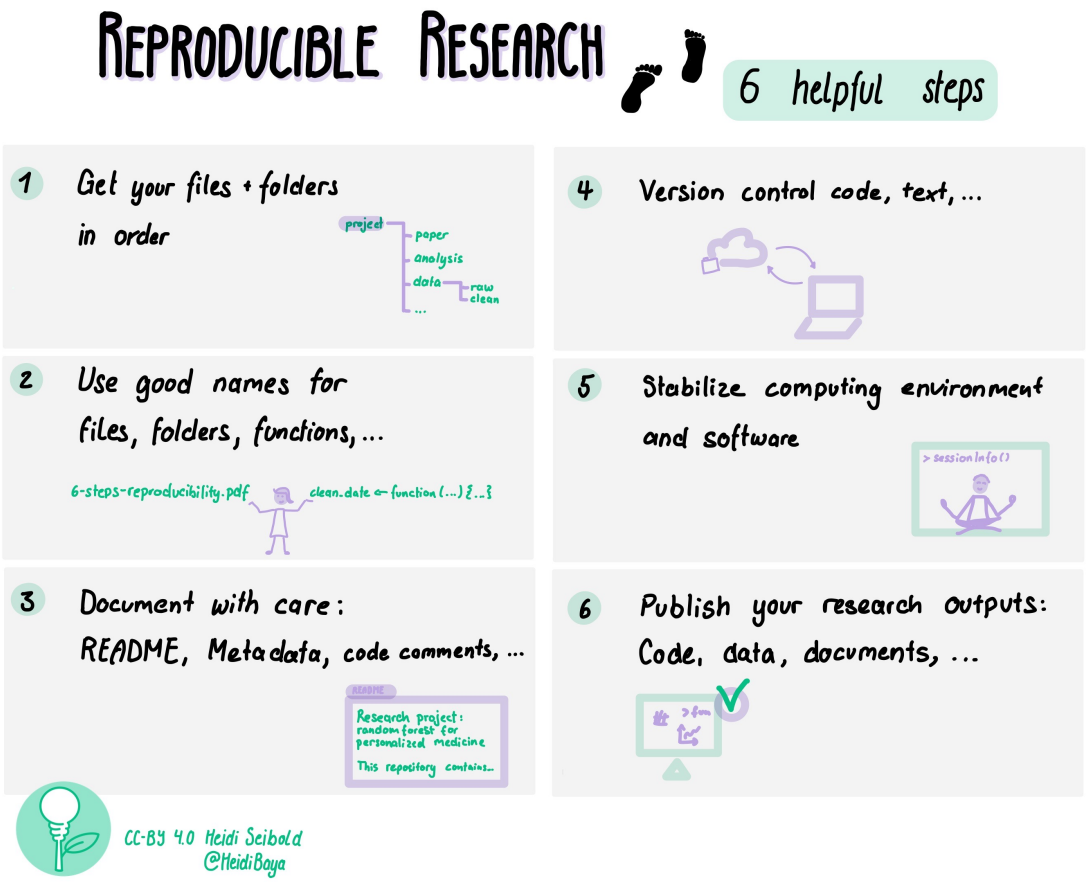



Figure 1: Reproducible Research: 6 helpful steps.

Making your work reproducible may seem daunting right now, but we'll take it **step by step** and you can choose what you want to implement now and what you want to keep for later.

This is a process and there is no need to take all the steps at once. Let's just try to move towards reproducible on the reproducibility scale.

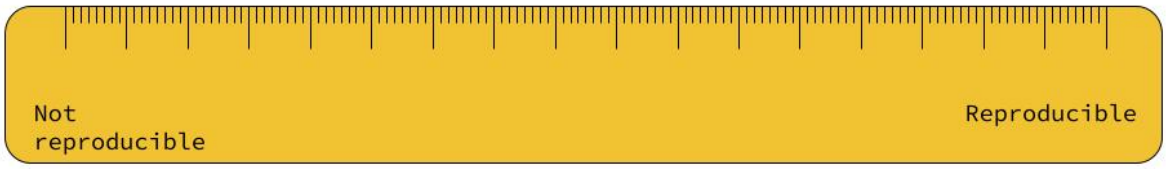



Figure 2: Reproducibility scale

## Your tasks

Go through the six steps and think about:

- What are you already doing well?
- What are you doing partly?
- What are you not doing yet?

Also, save the overview (or maybe event print it) so you can have it on your side throughout the next chapters when we discuss each step in detail:

Download overview (pdf).

In the **next chapter we will discuss step 1**: Get your files and folders in order.

# 1 Get your files and folders in order

**My first research project was a mess .** I had hundreds of files with dubious file names and sometimes several files with similar code written for computing on different infrastructures (my computer, the institute server, the cluster of the computing facility).

I felt like the worst researcher of all times. But I wasn't. **Many struggle with organizing their files and folders in increasingly complex research projects.**

## 1.1 How to organize files and folders well?

It basically comes down to structuring folders and files **systematically from the beginning**.

Think about what a good folder structure could be for your research project. A standard project of mine looks something like this:

```
.
  analysis            <- all things data analysis
     src              <- functions and other source files
  comm
     internal-comm    <- internal communication such as meeting notes
     journal-comm     <- communication with the journal, e.g. peer review
  data
     data_clean       <- clean version of the data
     data_raw         <- raw data (don't touch)
  dissemination
     manuscripts
     posters
     presentations
  documentation       <- documentation, e.g. data management plan
  misc                <- miscellaneous files that don't fit elsewhere
```

You can download this folder structure as a template from https://github.com/HeidiSeibold/research-project-template.

Not every project is the same and likely your project will be more complex than this. But if you think about good organization from the beginning, it will be easier in the long run.

What do you think about file or folder organization? Is your folder structure similar to mine?

## 1.2 Your tasks

Check one of your current research projects and get it organized :

- Does your folder structure help you and others find files fast?
- What would a better structure look like?
- If the project is a collaborative effort: suggest your new structure to your collaborators. Do they like it or do they have better ideas?
- Implement the new structure

## 1.3 Further reading

- Research Compendia, The Turing Way
- Towards a Standardized Research Folder Structure, GenR blog
- Folder structure of R packages, Making Packages in R, Software Carpentry
- Research Project Template, Heidi Seibold
- Data Analysis Project Template, a group of R users

# 2 Use good names

This chapter shows you how to pick good names.

**Good names for files, folders, functions** and other things can make a research project (or any project on your computer, really) more pleasant. Both for yourself and any people you work with.

Let's be kind to ourselves and the people around us and get into naming

## 2.1 Good naming

A few examples from Jenny Brian's slides of bad and good file names:

#### BAD

- Myabstract.docx
- Joe's Filenames Use Spaces and Punctuation.xlsx
- figure 1.png
- fig 2.png
- JW7d^(2sl@deletethisandyourcareerisoverWx2*.txt

#### GOOD

- 2014-06-08_abstract-for-sla.docx
- Joes-filenames-are-getting-better.xlsx
- Fig01_scatterplot-talk-length-vs-interest.png
- Fig02_histogram-talk-attendance.png
- 1986-01-28_raw-data-from-challenger-o-rings.txt

Names should be:

- **Machine readable**
- **Human readable**
- **Optional: Consistent** (decide how you use underscores _ and dashes -, if you want to use CamelCase or not, …)
- **Optional: Play well with default ordering** (e.g. start your file names with the creation date `YYYY-MM-DD`)

## 2.2 Your tasks

- Take a current project or code that you are writing on and check if you follow the naming rules discussed above.
- If you don't: go ahead and improve them

## 2.3 Further reading

- Naming files, folders and other things, The Turing Way
- Project structure slides, Danielle Navarro
- File naming slides, Jenny Brian
- ISO 8601, a standard for dates, Wikipedia

# 3 Document with care

In this chapter we discuss research documentation for reproducible research.

## 3.1 How can I document my research outputs?

There is actually no super-clear catch all answer to this question. It really depends on your needs, on your audience as well as on the types of research outputs you generate. In the following you find a few ideas to start from.

### README

One thing that I always do is to add a README-Text-File to each project. In the README I write the **most important info about the project**: What is it about? Who is involved? Where to find files? How to cite it? Where to find the paper? ...

### Code documentation

In my research projects code plays an important role. To make my code as understandable as possible for others, I use **literate programming** (mixing text and code to make it easier to read, e.g. Quarto) or add clear **code comments**. When writing functions in R I additionally use the standardized way to document R functions (via **Roxygen2**).

An example of code comments in R (`##`):

```
## Load package + data
library("model4you")
data("MathExam14W", package = "psychotools")

## scale points achieved to [0, 100] percent
MathExam14W$tests <- 100 * MathExam14W$tests/26
MathExam14W$pcorrect <- 100 * MathExam14W$nsolved/13

## select variables to be used
```

```
MathExam <- MathExam14W[ , c("pcorrect", "group", "tests", "study",
                             "attempt", "semester", "gender")]
```

### Metadata

Metadata is information about your data. It's information on the license of the data, who owns it, what information the data contain, ...

Many research fields have **standards for metadata**. If you can't find one for your field you can use a common standard (e.g. Dublin Core) or just ask a data manager or librarian at your institution. You can write metadata similar to a README (see e.g. this guide from Cornell University). If you upload your data to a data platform (e.g. Dryad) you won't have to think about it as the platform usually takes care of that (Dryad uses Dublin Core).

### Other

Whatever you work on, there might be parts of your research project that are difficult to understand. Say you work in a lab, then your documentation is a **lab notebook**. Or you do interviews, then your documentation may be your interview strategy. **Anything that might be useful for others is worth keeping and worth sharing**. *After all, we all want to build on the work of others in order to make the world a little better.*

## 3.2 Your tasks

- Check if your current research project already has a README. If not, create one
- Do you write code? Make a habit of writing code comments right when you create the code.
  - You will write code this upcoming week? Start doing it (if you don't already ).
  - You won't write code this week? Go to a recent script and check if you did a good job. If not, try code comments .
- Check out the literature linked below. Anything in particular you find interesting? Share your newly gained knowledge with your peers .

## 3.3 Further reading

Want to learn more? Check out:

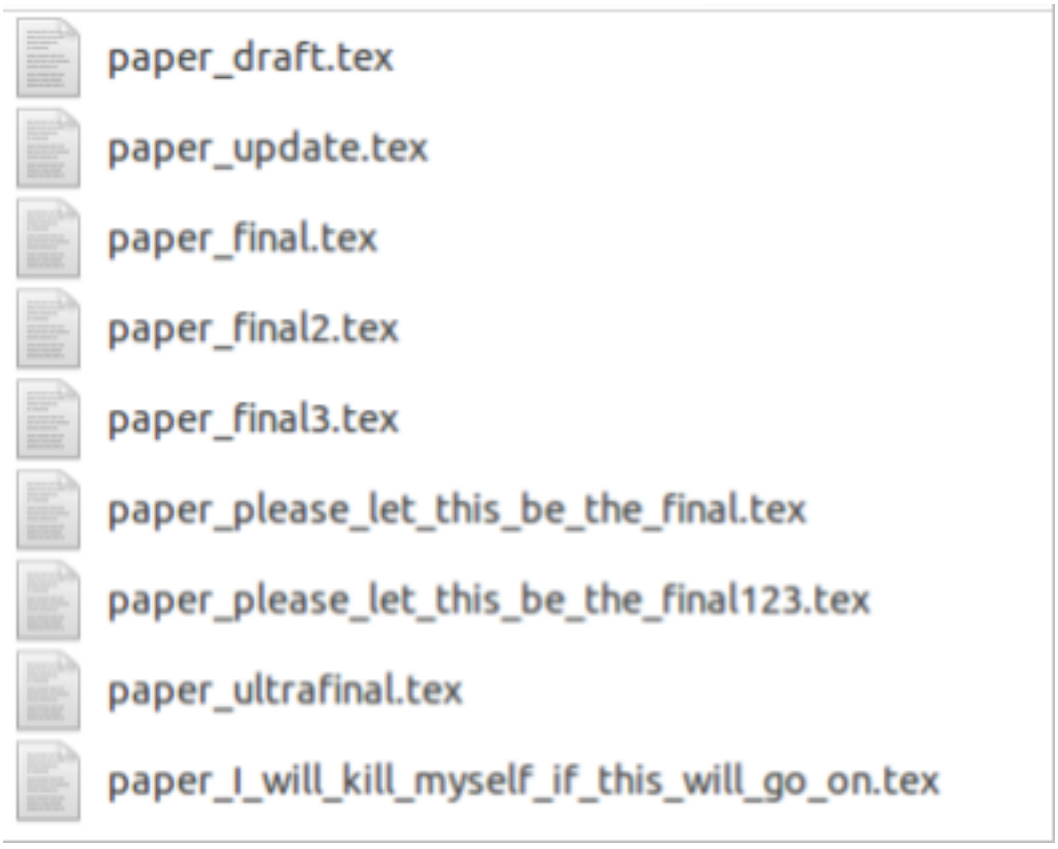
- Landing Page - README file, The Turing Way
- A beginner's guide to writing documentation, Write The Docs
- R Markdown: The Definitive Guide, Yihui Xie, J. J. Allaire, Garrett Grolemund
- knitr - Elegant, flexible, and fast dynamic report generation with R, Yihui Xie
- Guide to writing "readme" style metadata, research data management service group, Cornell University

# 4 Version control

Version control is a big topic for me. It completely changed the way I work. I am happy that we get to talk about version control as part of this 6 step process towards reproducibility.

## 4.1 What is version control?

Let's say you are writing a paper. You will edit your paper and might want to keep different versions of it. A common way to handle that is by using different file names for different versions.



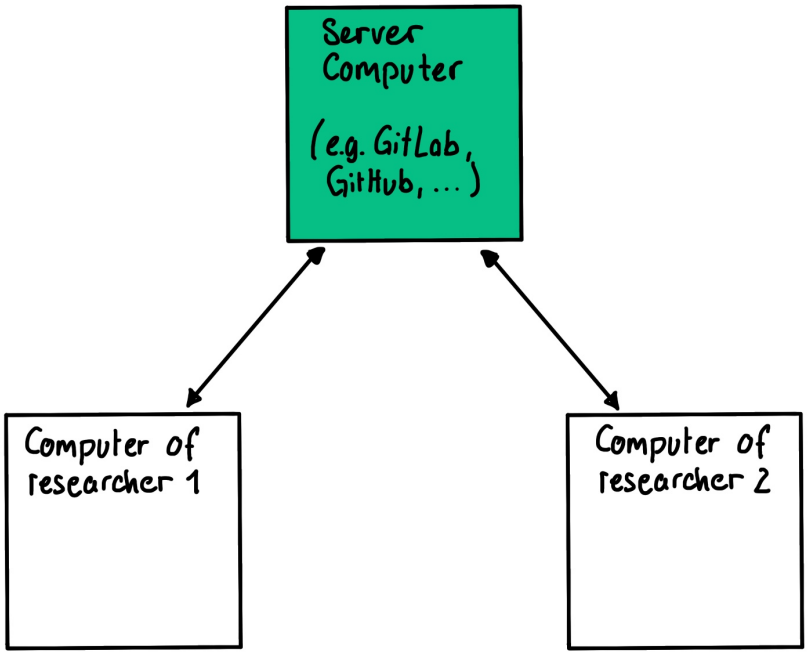
This way of "version control" is outdated and error-prone (hence the pixelated image ). The most common proper version control system today is Git, which I'd like to introduce to you now.

## 4.2 Git for version control

Git is free and open source .

With Git you can track different versions of your paper. For each version you can add a description ("commit message") and you even automatically track who made which change if you are working in a group. You can always go back to old versions.

The way you work with Git is that you have the version database both on our computers and on a server. To get the changes from and to the server you use commands (`pull` = download stuff from server, `push` = upload stuff to server).




Most researchers use GitLab or GitHub as platforms for working with Git and they also serve as a neat front end for the server. GitLab and GitHub give us some extra neat features for collaboration (e.g. issues, Wiki, ...).

Learning Git can be daunting . I recommend learning it with a group or in a class. I am always happy to teach version control. You can also check if there is a free Software Carpentry class in your area.

## 4.3 Other version control systems

There are many other ways of doing version control out there.

**Subversion:** Simpler systems like Subversion are used less these days as Git offers more flexibility.

**Google docs and friends:** Many online text editors (Google Docs, OneDrive, …) offer versioning now. It is not as advanced and versatile, but a nice way to work in a WYSIWYG (What You See Is What You Get) editor. Git really only works with real text files, so people usually use LaTeX or Markdown (not WYSIWYG) to write texts when using Git.

**Versioning data:** Version control of data is a difficult task. Let's leave that for another day. See here for more info for now.

## 4.4 Your tasks

For today's task it makes sense to do it together with a peer. Sometimes getting started with git is difficult, but with a friend and a cup of tea it is possible. Also I promise: it's worth it and gets easier over time .

- **Install git** on your computer, **create an account** on GitLab or GitHub, and start your **first repository**. For R users, I recommend following these instructions. Others, please check the "further reading" links below.

## 4.5 Further reading

- Version Control, The Turing Way
- Version Control with Git, Software Carpentry
- Version Control with Git (for R users), Anna Krystalli
- Set up Git with RStudio & GitLab, Heidi Seibold

# 5 Stabilize your computing environment and software

This topic may sound technical and boring at first, but please bare with me . **It will be useful!**

Have you ever had the problem that you ran an old code and it just did not work anymore? After hours of digging into the issue you find that it's because the software package you use has changed in the meantime

Or have you tried to reproduce someone else's code, which seems to run on their machine but not on yours and you just don't know why.

This chapter is all about avoiding such problems in the future by **stabilizing your computing environment and software**.

## 5.1 What is a computing environment?

Your computing environment is defined by your computer, the operating system and the software installed. If you update your operating system or your software, your computing environment changes. In R, for example, you can learn a lot about your computing environment by typing `sessionInfo()`.

```
sessionInfo()
```

```
R version 4.2.2 (2022-10-31)
Platform: aarch64-apple-darwin21.6.0 (64-bit)
Running under: macOS Ventura 13.2.1

Matrix products: default
BLAS:   /opt/homebrew/Cellar/openblas/0.3.21/lib/libopenblasp-r0.3.21.dylib
LAPACK: /opt/homebrew/Cellar/r/4.2.2_1/lib/R/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
```

```
attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

loaded via a namespace (and not attached):
 [1] compiler_4.2.2  magrittr_2.0.3  fastmap_1.1.0   cli_3.5.0
 [5] tools_4.2.2     htmltools_0.5.4 rstudioapi_0.14 stringi_1.7.8
 [9] rmarkdown_2.18  knitr_1.41      stringr_1.4.1   xfun_0.35
[13] digest_0.6.29   jsonlite_1.8.0  rlang_1.0.6     evaluate_0.16
```

It tells the R version, operating system, loaded R packages as well as their versions.

## 5.2 Options for stabilizing your computing environment

### 1) Record your computing environment

Document the software versions you used. For example if you use R, you could copy the output of `sessionInfo()` into your README or somewhere else where future you (and others) can find this information. This is not exactly "stabilizing" but it gives the possibility to install the same software versions again.

### 2) Use one virtual machine per research project

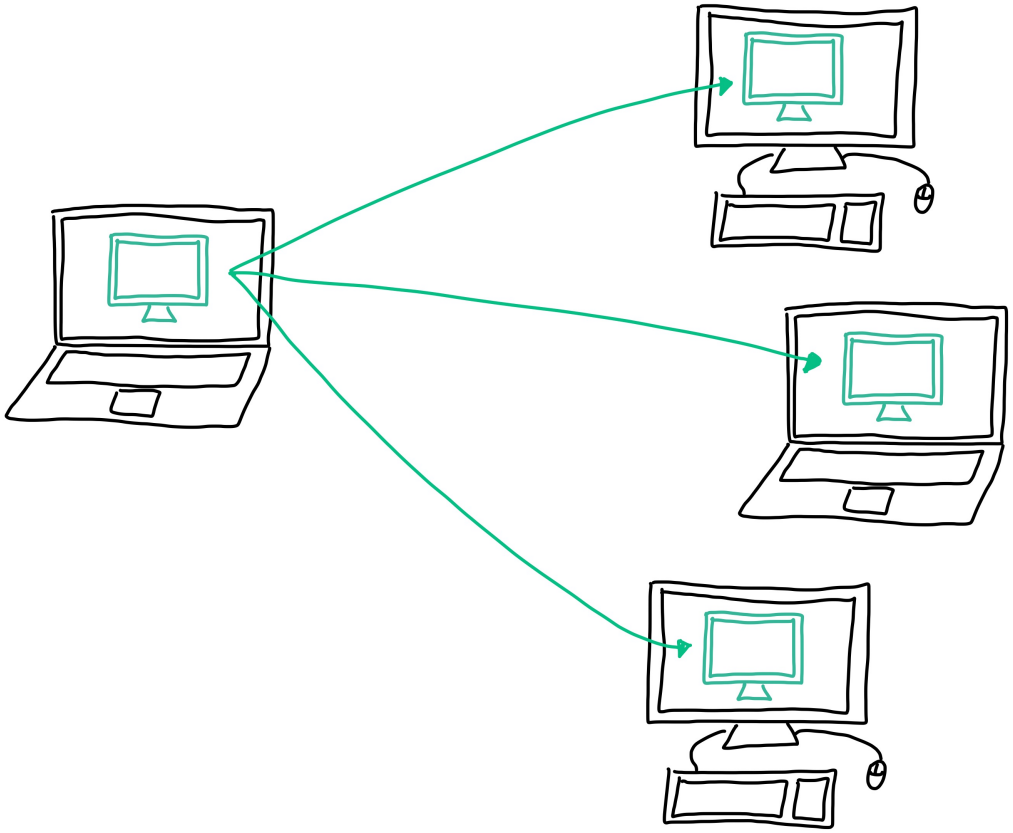
You don't need to know what a virtual machine is or how to set it up to be able to do this. I used to ask the wonderful IT person at my institute to set up a virtual machine for me and if your IT supporters know their job, they'll be able to help you here.

A virtual machine is essentially a virtual computer on another computer or server (To those nerds out there, I know I am probably explaining it incorrectly but for the purpose of what we want to achieve here, it's good enough). If you have one virtual machine for each project, you can keep the computing environment stable by not installing or updating software after you've finished the research project.

The downside of this strategy is that this is only for future you and your collaborators, but not for other researchers who want to work with the same computing environment.

### 3) Use one container per research project

Containers are similar to virtual machines (think little computer inside your computer). The big difference is that you can make them available for others. So you can send your container image (or the file describing it) to others.



Popular container tools are **Docker and Apptainer** (formerly Singularity). Learning to work with containers is not super easy, but it is worth the time and actually can be applied in so many other situations. So, a great skill to have even if you decide to quit research.

### 4) Other

There are many other options out there. I wrote down the three that are least dependent on the actual software you use. For R users, check out packages `logrx`, `rang`, `packrat`, `versions`, and `renv`.

## 5.3 Your tasks

- Take a moment to think about the 4 options . What would work best for you?
- Go ahead an implement one of the options in your current research project. Again, don't be shy to ask your IT colleagues. One tip on the side: being friendly with IT colleagues is a great advantage in research .

## 5.4 Further reading

- Reproducible Environments, The Turing Way
- Video: How can software containers help your research?, Paula Andrea Martinez + Australian Research Data Commons
- R Docker tutorial, maintained by Jemma Stachelek

That's all for this chapter. I hope it was helpful and not too technical. Happy researching!

# 6 Publish your research outputs

Wow friend, I can't believe we've made it this far! This is already the last chapter.

Last but not least I want to share with you some thoughts on **publishing research outputs**.

When people write the following:

> *The data/code will be made available upon request.*

This usually means:

> *Once the PhD student who wrote this paper leaves their position, the data/code will be lost in space.*

Am I right? But how can you do better? How can you make your research outputs available?

## 6.1 Publish in a repository

Publish your research outputs in a repository. You basically have three options here:

- A general purpose service (e.g. Zenodo or Open Science Framework),
- The service of your institution (e.g. Open Data LMU or ETH Zurich's Research Collection),
- A field or project specific service (e.g. a specific repository for high throughput sequencing data or CRAN for R-Packages)

Please make sure to use a trustworthy service. How to check if a service is trustworthy? My rule of thumb is that services that have investor backing (e.g. Figshare) are less trustworthy than services backed by the research community (e.g. Zenodo, which is developed by OpenAIRE and CERN). Why? Well, I think an Open Science service should not be driven primarily by commercial goals. At some point commercial services will take money from you, if that may be by selling your data, by locking your uploaded material behind a pay wall, or in another way.

## 6.2 Publish with the paper

Some journals offer to publish your research outputs with your paper. I will be honest, I have mixed feelings about this. Not all journals which offer this, really have the expertise to do so and they don't necessarily have the possibility to store data long term. For one of my papers we uploaded the material with the journal, but the link to the material keeps vanishing and I keep getting the confused emails of interested readers. So, make sure the journal you upload your material to, ensures long term storage and availability .

## 6.3 If your research outputs cannot be shared openly

What should you do if you cannot publish your research outputs openly?

If you have sensitive data (e.g. patient data) and no consent, do not publish the data! There are other options for you.

If for any reason you cannot share your research outputs, think of options how you can still ensure that others can trust in the reproducibility of your research.

- Can you maybe publish the metadata and the code?
- Can you publish a synthetic version of your data?
- Can you share the data with specific people (e.g. researchers in the same field)?

Brainstorm with your peers, librarians, or IT support. There are always solutions that are better than publishing nothing.

## 6.4 Your tasks

- Discuss with your research team: what are good places to publish your data, code and other research outputs in your field? Put it on the agenda of the next team meeting
- Do you know what you are allowed to do with your code and data? If not, discuss it with the people who might know
- Check out Open Science Framework (OSF) or Zenodo. Try uploading something simple (e.g. slides of your last presentation)

## 6.5 Further reading

- My favorite helpers for choosing licenses:
  - Code: Choose an open source license

  - Anything: Choose a Creative Commons license

- Zenodo Guide
- How create an OSF project

# Conclusion

Thanks for your interest in making your work more reproducible. Let's improve the way we do science!

I hope these 6 steps were able to help you.

## What are the next steps?

This booklet is only a short introduction and there is so much more out there to learn about reproducible research. Here are some ideas:

- Check out the resources listed in the chapters of this booklet.
- Book me as a consultant or trainer for your project/team/PhD program and let me help you (go to heidiseibold.com for more information).
- Join a group of like-minded people. Lot's of places have Open Science meetups these days. If there is none near you, why not found one?
- Implement and try out. Be playful with your research and try new things. There is no need to be perfect in your first research project, but if you implement a new good practice in each project you start, you'll soon be a research superhero.