REPRODUCIBILITY & PROJECT MANAGEMENT

AG FORSLUND MEETING || 9 FEB 2021

HARITHAA & MORGAN

AGENDA

- Introduction and survey results
- Spectrum of reproducibility
- Four focus areas (specific recommendations)
  - Organization
  - Documentation
  - Automation
  - Collaboration
- Further resources

WHY CARE ABOUT REPRODUCIBILITY?

- A matter of principle
  - It's a major component of the scientific method
  - Computational science/data analysis no different
  - Uncertainty about the level of reproducibility in scientific research – crisis?

- Practical reasons
  - Increasingly under scrutiny from funders, reviewers
  - Helps others to understand and trust your work
  - Helps you!

- Methods vs Results vs Inferential Reproducibility: https://cure.web.unc.edu/defining-reproducibility/

SURVEY RESULTS

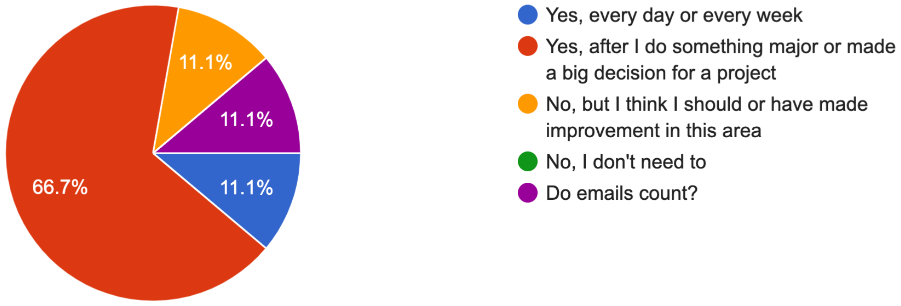
- Elaborate on your particular *challenges, development, and interests* with respect to reproducibility and (computational) project management :

- Improve my **coding skills** to make more understandable code
- I want to make the manuscript and **maintenance** phase of my analysis (i.e. the end) as painless as possible
- Proper **version controlling**, good **commenting**, sensible **script management** (which/how many script(s) for what)
- **Inconsistency in my attempts** at reproducibility, partly because it is **exhausting** and in the initial stages harder to keep it going
- Some form of **more elaborate explanation of my thought processes** would probably be beneficial in the future

SURVEY RESULTS

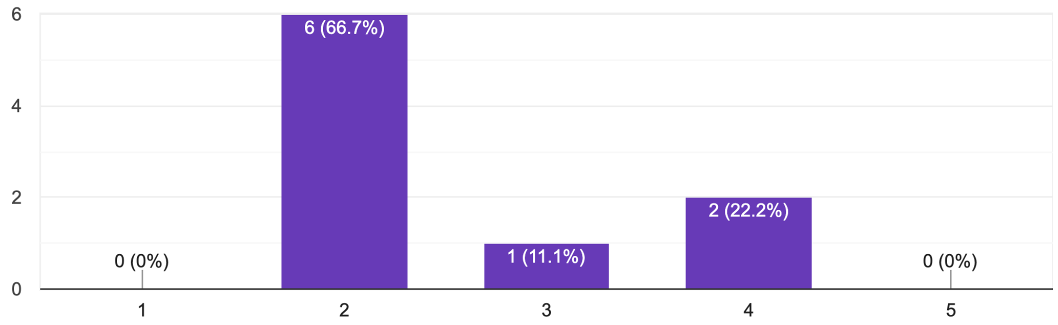
I keep a record of what I do for work

9 responses



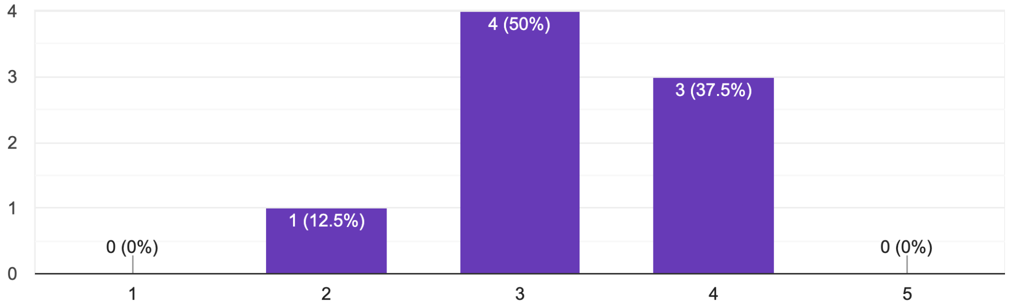
It would be difficult for me to reproduce my own analysis for a project (3 = neutral/no answer)

9 responses



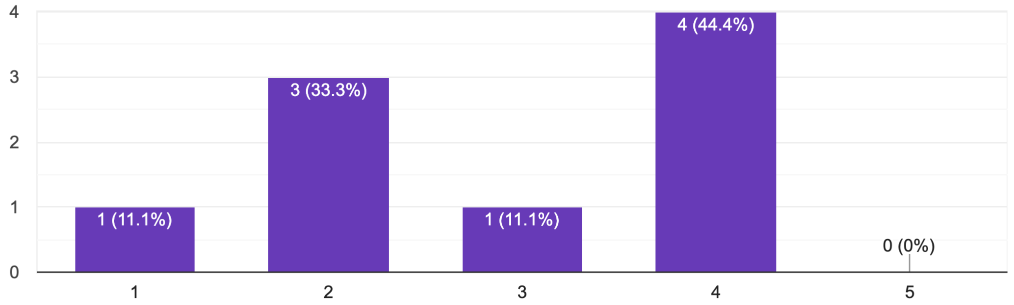
I would consider myself to be well-organized when it comes to my research projects

8 responses

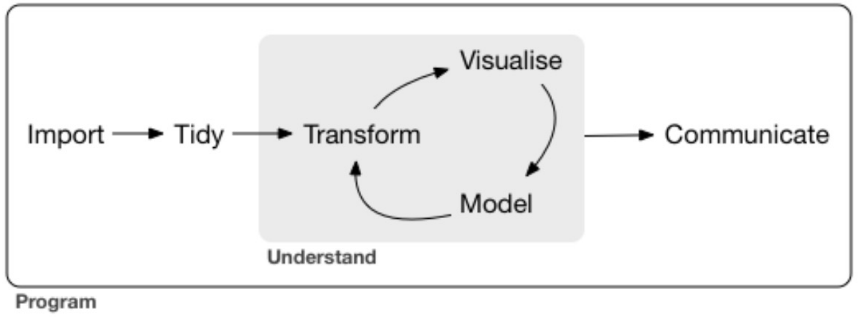


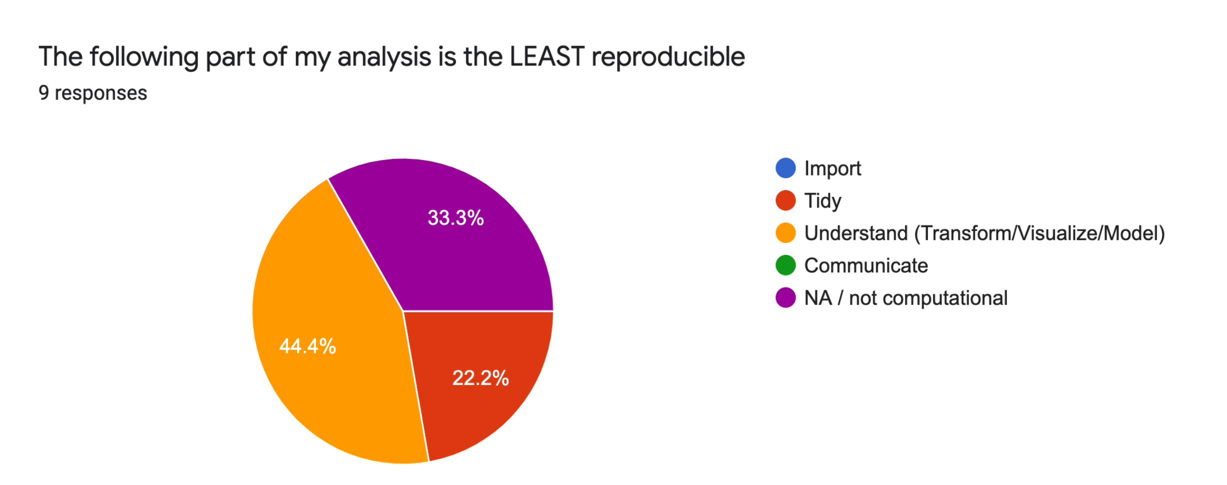
It would be difficult for a colleague or my supervisor to reproduce my analysis for a project (3 = neutral/no answer)

9 responses



PROBLEM AREAS

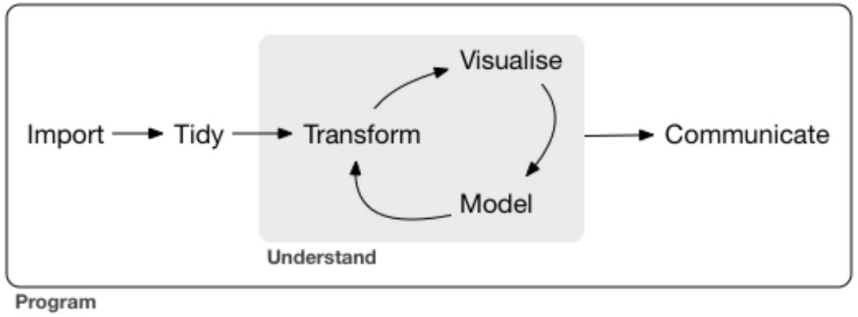


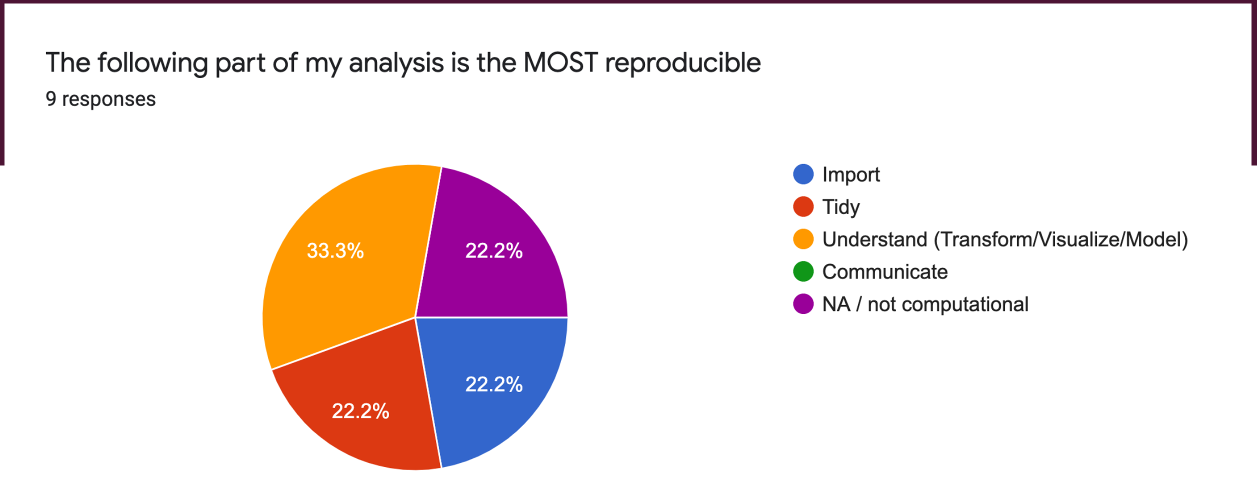

The following part of my analysis is the LEAST reproducible

9 responses

- I think **what do to when and where is not exactly clear** yet. I should disentangle my scripts and write a clear plan describing which script to use for what, why and in which order.
- Sometimes it is easier to fix **weird flaws in raw data** (especially excel files) manually (e.g in Excel). Since there is no code documenting these steps, they are hard to reproduce.
- I am pretty bad at **saving intermediate/exploratory results** in a way that could be recreated or accessed

STRENGTHS



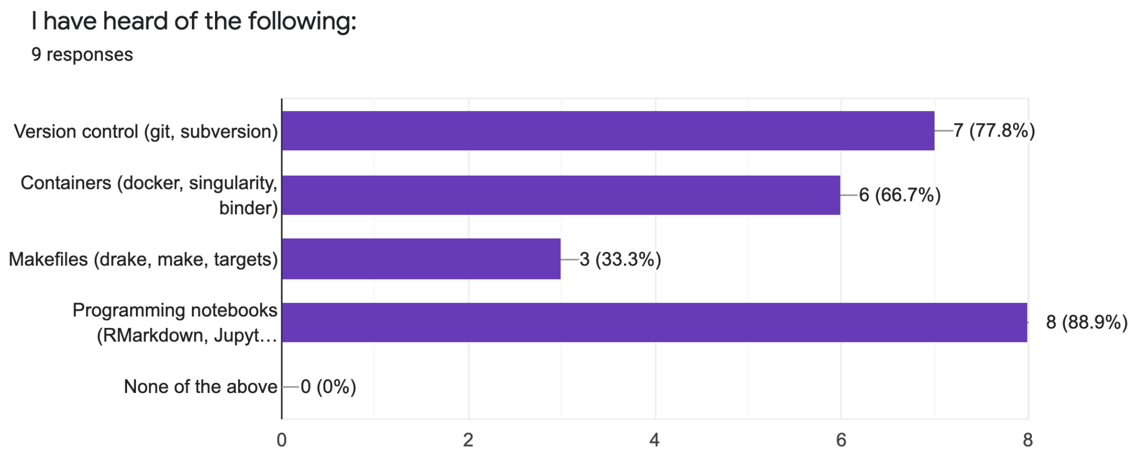

The following part of my analysis is the MOST reproducible

9 responses

- I do not think there could be many ways to import
  - File paths? Libraries? Format checks?
- All "data manipulation" is kept as **commented code** and so given all input data is present, analyses can be reproduced
- I describe every step and **why** this step is done in each script

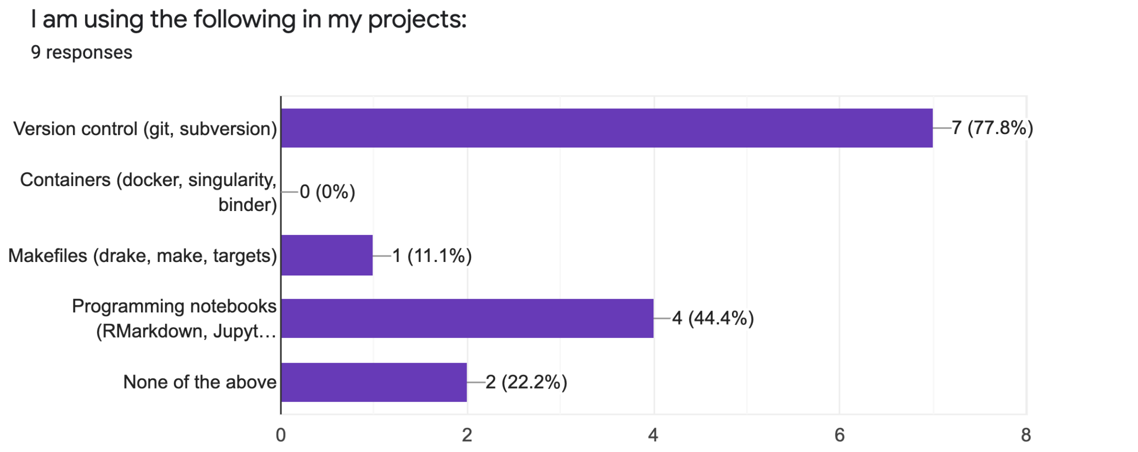
REPRODUCIBLE PRACTICES

- Gap between exposure and implementation
- Everyone's heard of git and programming notebooks like R Markdown and Jupyter
  - Only half using notebooks – reasons?
- Interest in containers? Expertise?
- Makefiles – TeX users?



I have heard of the following:

9 responses

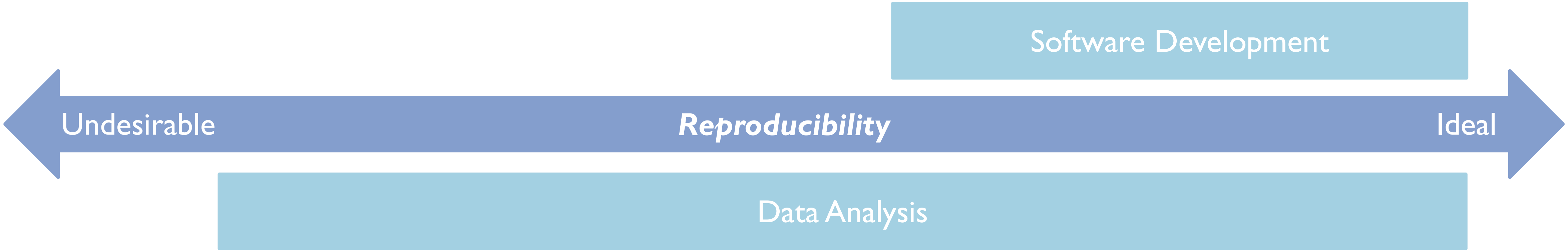


I am using the following in my projects:

9 responses

ONE SIZE DOES NOT FIT ALL

Intrinsic to the process,
somewhat required



We all start here

The **goal** is **incremental improvement**, not to be here

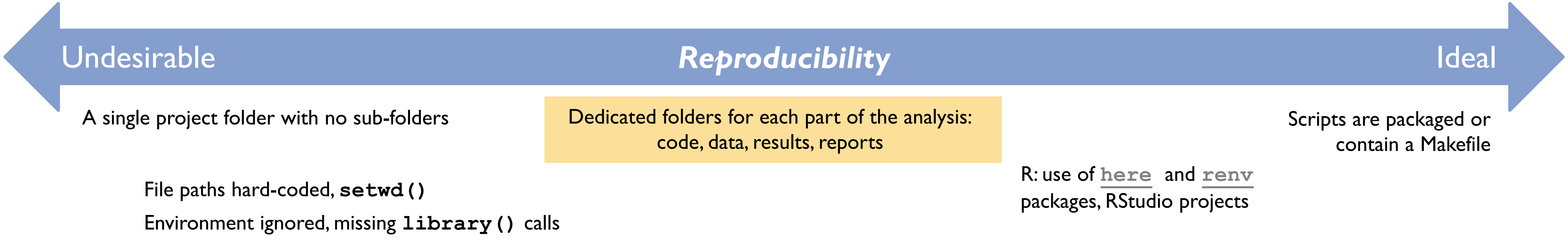
Using some of these strategies and tools is better than none

"Your closest collaborator is you six months ago, but you don’t reply to emails."

FOUR AREAS TO FOCUS ON

WHEN THINKING ABOUT REPRODUCIBILITY

ORGANIZATION



Undesirable

Reproducibility

Ideal

A single project folder with no sub-folders

File paths hard-coded, `setwd()`

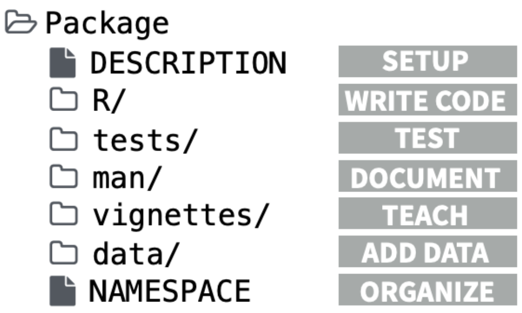
Environment ignored, missing `library()` calls

Dedicated folders for each part of the analysis: code, data, results, reports

R: use of `here` and `renv` packages, RStudio projects

Scripts are packaged or contain a Makefile

- How your projects and files are structured
- How adaptable that is to a naïve user or after time and updates
- Jenny Bryan's post on project-oriented workflows is a must-read
- Organization will enable or restrict documentation and automation options

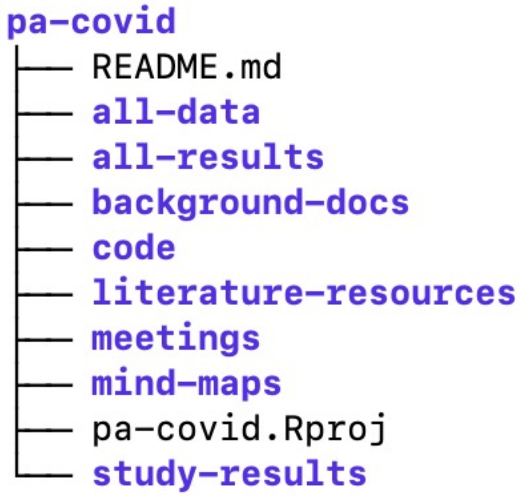


```
Package
  DESCRIPTION   SETUP
  R/            WRITE CODE
  tests/        TEST
  man/          DOCUMENT
  vignettes/    TEACH
  data/         ADD DATA
  NAMESPACE     ORGANIZE
```

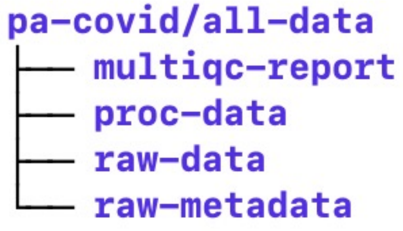
ORGANIZATION TAKEAWAY

- Steps to implement now:
  - Separate code from data
  - Separate raw from processed data
  - Use the **`here`** package
  - Store library calls in a script

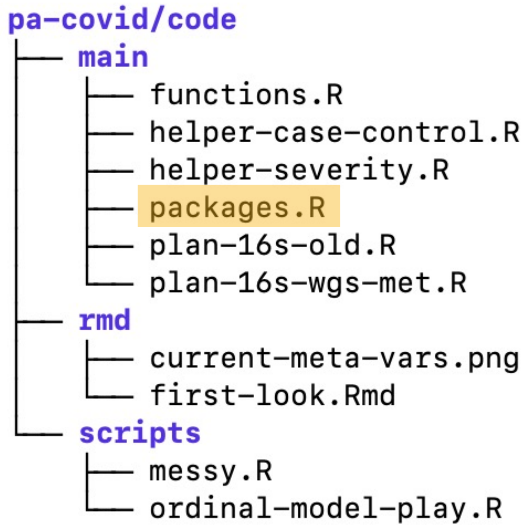
- Long term steps that would help:
  - Develop your own system
  - Be consistent and document it



```
pa-covid
├── README.md
├── all-data
├── all-results
├── background-docs
├── code
├── literature-resources
├── meetings
├── mind-maps
├── pa-covid.Rproj
└── study-results
```

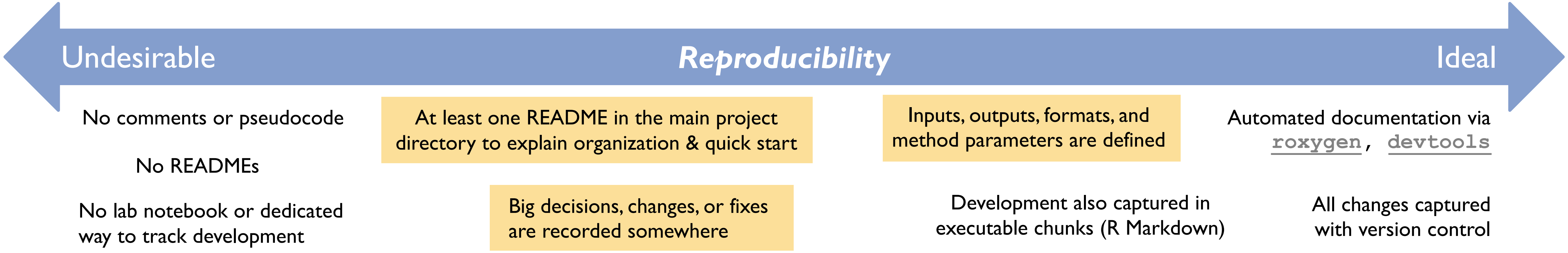


```
pa-covid/all-data
├── multiqc-report
├── proc-data
├── raw-data
└── raw-metadata
```



```
pa-covid/code
├── main
│   ├── functions.R
│   ├── helper-case-control.R
│   ├── helper-severity.R
│   ├── packages.R
│   ├── plan-16s-old.R
│   └── plan-16s-wgs-met.R
├── rmd
│   ├── current-meta-vars.png
│   └── first-look.Rmd
└── scripts
    ├── messy.R
    └── ordinal-model-play.R
```

DOCUMENTATION



Undesirable

Reproducibility

Ideal

No comments or pseudocode

No READMEs

No lab notebook or dedicated way to track development

At least one README in the main project directory to explain organization & quick start

Big decisions, changes, or fixes are recorded somewhere

Inputs, outputs, formats, and method parameters are defined

Development also captured in executable chunks (R Markdown)

Automated documentation via `roxygen`, `devtools`

All changes captured with version control

- How you capture the logic and development of your analysis and methods
- Determines how well a naïve user can follow (or improve/debug!) your work
- Ideally, your pseudocode and comments stay in – you just formalize and generalize them
- Programming notebooks (.Rmd) tie code to figures/reports
  - Great for intermediate results and documenting development of thought process

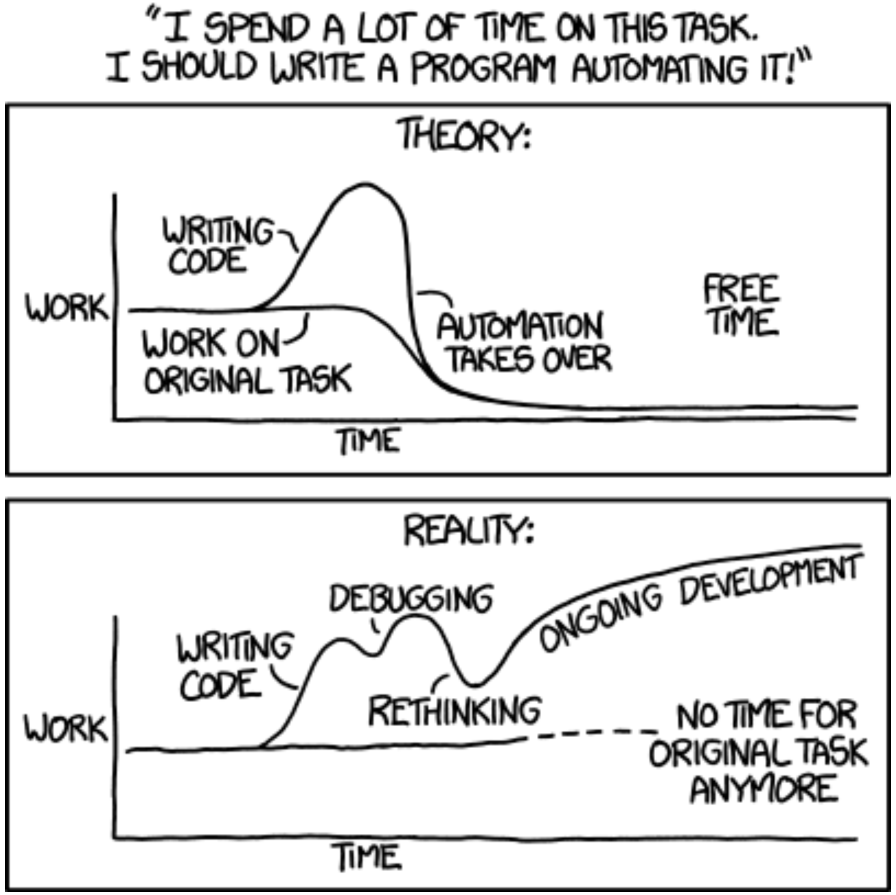
DOCUMENTATION TAKEAWAY

"Have sympathy for your future self"

- Steps to implement now:
  - Use READMEs to explain your organization
  - Plan to spend 20-25% of your coding time commenting
    - Note input, output, and purpose – not the mechanics

- Long term steps that would help:
  - Get better at git! Learn when and why to commit changes, there are tons of YouTube videos
  - Consider R Markdown in your workflows to save intermediate results and exploratory data analysis, especially

AUTOMATION

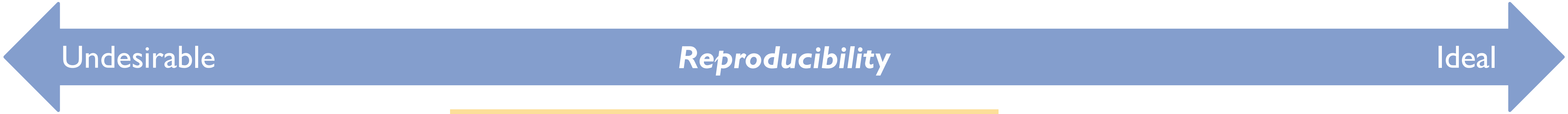

"I SPEND A LOT OF TIME ON THIS TASK.
I SHOULD WRITE A PROGRAM AUTOMATING IT!"
THEORY:
WRITING CODE
WORK
WORK ON ORIGINAL TASK
AUTOMATION TAKES OVER
FREE TIME
TIME
REALITY:
DEBUGGING
WRITING CODE
ONGOING DEVELOPMENT
WORK
RETHINKING
NO TIME FOR ORIGINAL TASK ANYMORE
TIME

AUTOMATION

Scale up the work you need.

Skip the work you don't.

See evidence of reproducibility.


Undesirable

Reproducibility

Ideal

Manual edits to input: data/metadata

Intermediate results overwritten during development process

Procedure is documented such that input → output is clear

`main/plan` script to calculate the main results, implementation abstracted away

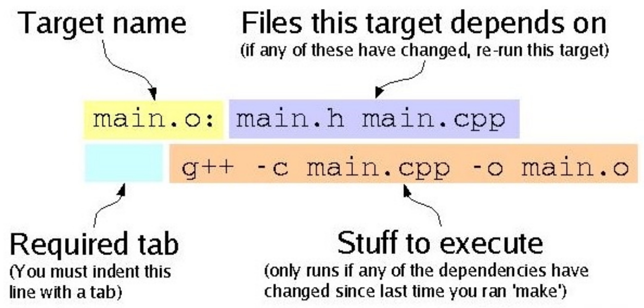
R Markdown

Makefiles, pipeline/workflow managers like `targets`

Containers like docker, singularity

Continuous Integration & Deployment (CI/CD)

- How much of your analysis depends on human intervention
- This depends a lot on your role (developer vs analyst) and needs, e.g. scaling up
- More automation improves chances a naïve user could reproduce your work
- Comes with organization/structure requirements for you to implement
- Can save time – especially if caching and dependency detection are included (as with targets)

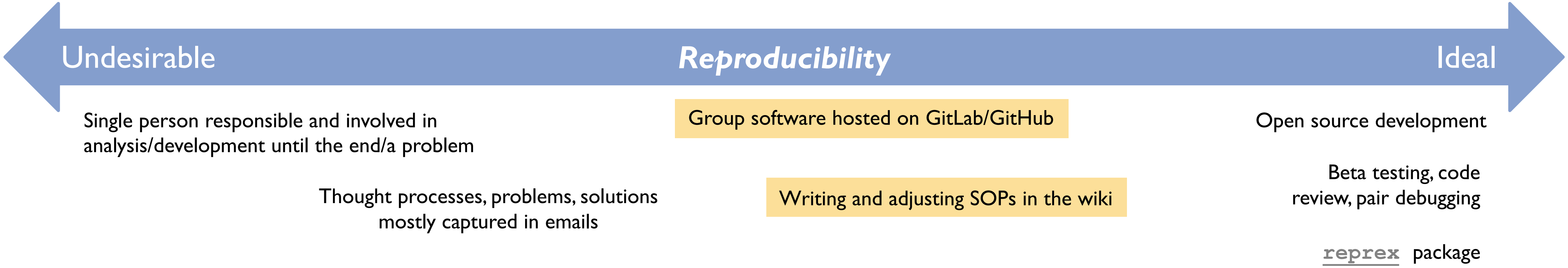

```
main.o: main.h main.cpp
	g++ -c main.cpp -o main.o
```
Target name
Files this target depends on
(if any of these have changed, re-run this target)
Required tab
(You must indent this line with a tab)
Stuff to execute
(only runs if any of the dependencies have changed since last time you ran 'make')

AUTOMATION TAKEAWAY

- Steps to implement now:
  - Data analysts: don’t worry too much, invest as you see fit
  - Developers: look for tools to help you scale and test

- Long term steps that would help:
  - Practice more scripting
  - Consider a workflow manager (GNU Make or targets for R)
  - Get better at formal programming so code breaks less and debugging is easier

COLLABORATION



Undesirable

Reproducibility

Ideal

Single person responsible and involved in analysis/development until the end/a problem

Thought processes, problems, solutions mostly captured in emails

Group software hosted on GitLab/GitHub

Writing and adjusting SOPs in the wiki

Open source development

Beta testing, code review, pair debugging

`reprex` package

- How easy it is for someone else to contribute to your efforts
- Within the scope of a project, within a team, as a member of the scientific community
- Open source development is inherently collaborative and **transparent**
- Requires a shared organization and documentation

COLLABORATION TAKEAWAY

- Steps to implement now:
  - Get used to git
    - Code for data analysis should be published along with manuscript
  - Capture your knowledge and problem solving efforts and share them in the wiki

- Long term steps that would help:
  - Take part in discussions on GitHub and StackOverflow
  - Share debugging practices? Swap scripts and document? Mini-hackathon for the wiki?

FURTHER RESOURCES AND CONTENT

- Slides from a graduate-level course on advanced data analysis (Karl Broman)
- Five minute 2020 useR flash talk on why use R projects and the **here** package
- Mattermost pin of key scientific publications discussing reproducibility in scientific research
- Jenny Bryan's 2020 useR keynote on nurturing your inner problem solver (+ reprex/debugging)
- Video lecture on lessons the scientific community should learn/adopt from open source software development
- Nature article on FAIR data principles intended to improve reuse of academic data