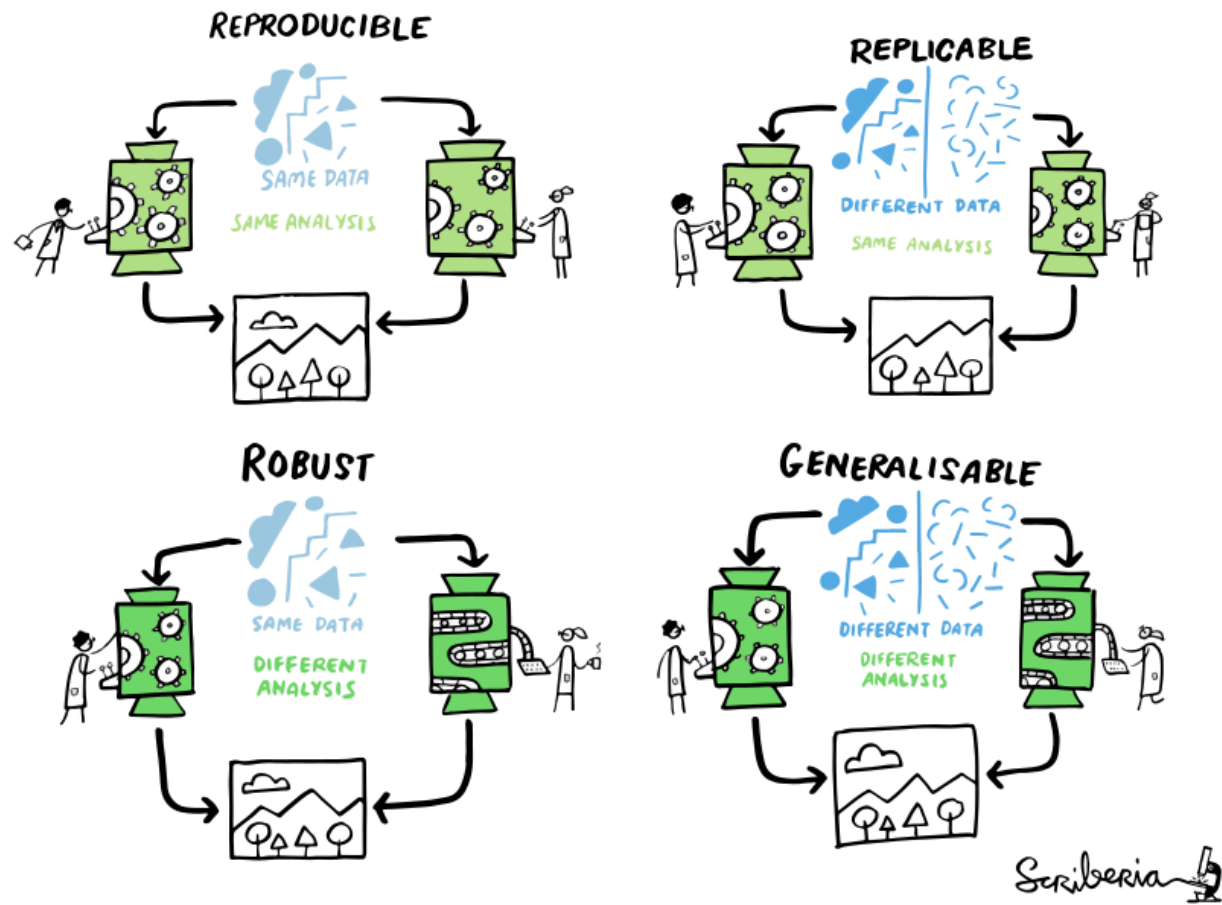


Seven Steps to Reproducible (Computational) Research

Liz Ing-Simmons

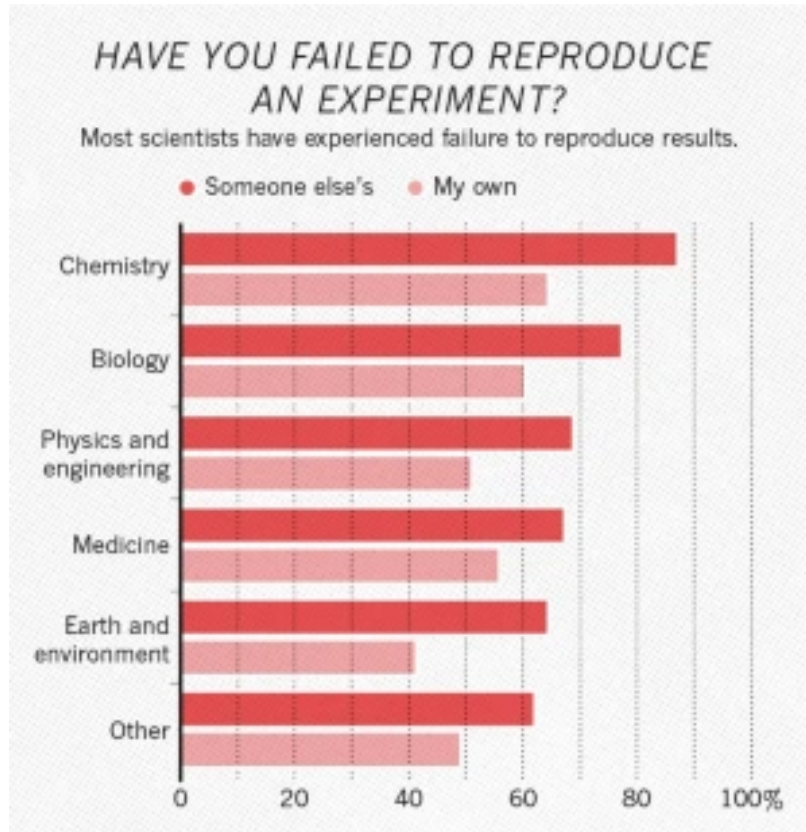
What is reproducibility?



The Turing Way project illustration by Scriberia

[The Turing Way Community & Scriberia \(2024\)](#), CC-BY 4.0.

The reproducibility crisis



Most scientists have experienced failure to reproduce results.

[Nature, 2016](#)

- Studies that attempt replication have low success rates
- Barriers include:
 - lack of data or experimental reagents
 - incomplete methods
 - unresponsive original authors

Why does reproducibility matter?

A lack of reproducibility leads to:

- Time, effort, and funding wasted on follow-up projects
- Loss of trust in research
- Potential for real harm

Why does reproducibility matter?

Personal benefits to working reproducibly:

- Avoid disaster
- Make it easier to write your paper/thesis
- Help people understand your work
- Enable continuity of your research
- Build your reputation

Five selfish reasons to work reproducibly, Markowetz, 2015

Seven steps

1. Planning to be organised
2. Keeping data tidy
3. Methodology and protocols
4. Testing and controls
5. Automation
6. Documentation
7. Publishing

Source: carpentries-incubator.github.io/ReproducibleResearch

1. Planning to be organised

Before you start a project:

- What data will you collect?
- What code will you write?

1. Planning to be organised

How will you organise your data / code?

- Directory structure
- File naming conventions
 - file names that are both human-readable and computer-readable
 - date formats - **YYYY-MM-DD** is best!
 - project-specific abbreviations - be consistent

1. Planning to be organised: tools



- Data Management Plans
 - outline how data will be looked after during and after the project
 - required by most major funders
- Project templates
 - many examples available - general and domain-specific
 - typically separate documentation, code, raw data, results

```
thesis-project/  
├── docs  
│   └── data_management_plan.md  
├── figures  
│   ├── Fig1.png  
│   └── Fig2.png  
├── notebooks  
│   └── data_exploration.ipynb  
├── raw-data  
│   ├── big_dataset1.csv  
│   ├── big_dataset2.csv  
│   └── big_dataset3.csv  
├── README  
├── results  
│   └── clean_data.csv  
└── src  
    ├── my_function.py  
    └── process_data.py
```

Example project template

2. Keeping data tidy

“each variable is a column, each observation is a row, and each type of observational unit is a table”

[Wickham, 2014](#)

(See also the [third normal form](#) for databases.)

- Tidy data facilitates analysis
- Makes data cleaning easier
- Works well with statistical tools

2. Keeping data tidy

“tidy datasets are all alike but every messy dataset is messy in its own way”

[Wickham, 2014](#)

	name	quiz1	quiz2	exam1
1	Billy	<NA>	D	C
2	Suzy	F	<NA>	<NA>
3	Lionel	B	C	B
4	Jenny	A	A	B

	assessment	Billy	Suzy	Lionel	Jenny
1	quiz1	<NA>	F	B	A
2	quiz2	D	<NA>	C	A
3	exam1	C	<NA>	B	B

2. Keeping data tidy

	name	assessment	grade
1	Billy	exam1	C
2	Billy	quiz1	<NA>
3	Billy	quiz2	D
4	Jenny	exam1	B
5	Jenny	quiz1	A
6	Jenny	quiz2	A
7	Lionel	exam1	B
8	Lionel	quiz1	B
9	Lionel	quiz2	C
10	Suzy	exam1	<NA>
11	Suzy	quiz1	F
12	Suzy	quiz2	<NA>

- 3 variables, 12 observations, 36 values
- Each combination of name and assessment is a single measured observation
- Definition of a variable depends on the task
- E.g. what if we wanted to compare quiz and exam scores?

2. Keeping data tidy: tools

- Python: `pandas`, `polars`
- R: `tidyr`, `dplyr`, `polars` (*not yet stable)

!! but cleaning and tidying your data is no substitute for planning good data organisation in the first place!

3. Methodology and protocols



Key questions:

- What did you do?
- Why did you do it that way?

3. Methodology and protocols



What did you do?

- Scripts - document the actual code you ran
- Computational environment
 - random seeds
 - package versions
 - hardware
 - operating system

3. Methodology and protocols: tools



Virtual environments

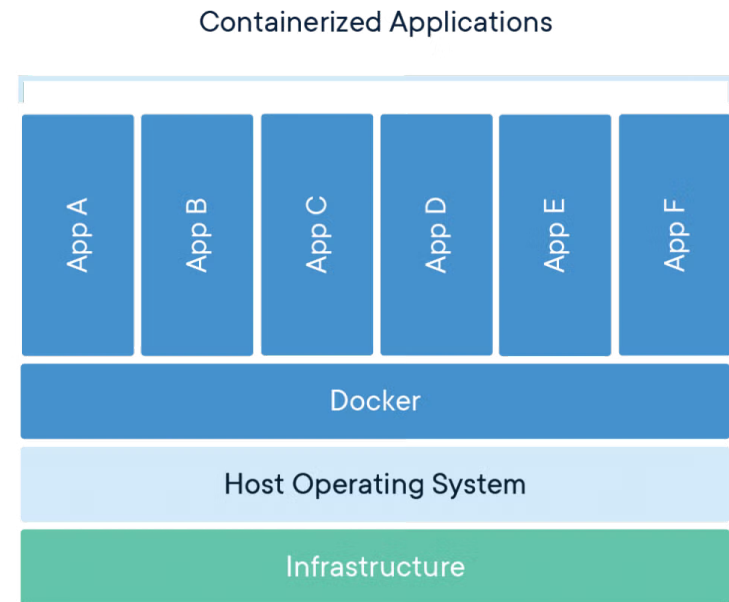
- Python: `python -m venv`
 - use `pip freeze` to record versions
- R: `renv`
 - use `renv::snapshot()` to record versions
- Conda
 - use `conda env export` to record versions

3. Methodology and protocols: tools



Containers

- Docker
- Singularity / Apptainer



Multiple Docker containers running on the same machine

3. Methodology and protocols



Why did you do it that way?

- Version control e.g. Git
- Literate programming - combine code with text and outputs
 - Jupyter notebooks, RMarkdown, Quarto...
 - Code documentation (docstrings)
- Code reviews
- Architectural decision records

4. Testing and controls

Is the data what we expect? Does the code do what it should?

- Data validation
 - checksums
 - size and shape, data types, missing data
- Software testing
 - unit tests - does this function do what it should do?
 - snapshot testing - does this code still give the *same* output?
 - integration testing - does everything still work together?

5. Automation

✨ Automate the boring things

- Scripts for data cleaning / processing
- Combine scripts into workflows: run a series of data analysis steps in order
- Continuous integration - automatically run your tests

5. Automation: tools

- Workflow managers
 - [Nextflow](#): lots of public workflows for bioinformatics
 - [Snakemake](#): Python-based, easy to create your own workflows
- Continuous integration
 - [GitHub Actions](#): lots of templates

6. Documentation



“Documentation is a love letter to your future self”

Damian Conway, *Perl Best Practices*, 2005

- Document how all these steps fit together
- Think about what you’d need to tell a new project contributor
- Or what you need to know when someone leaves

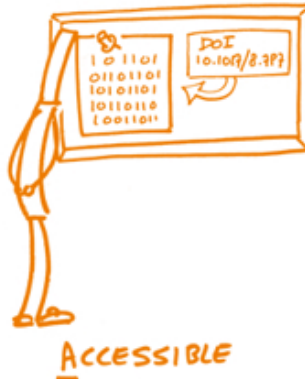
6. Documentation: tools

- README
- Electronic lab notebook
 - LabArchives, Benchling, Notion, OneNote...
- Project wiki
- Project website with code documentation, tutorials, etc
 - e.g. Read the Docs

7. Publishing



FAIR DATA PRINCIPLES



FAIR principles

[Open Science Training Handbook](#), CC0

7. Publishing: tools

- Persistent identifiers (e.g. DOI)
 - Data repositories can give a persistent identifier
 - Code can also be submitted to e.g. Zenodo
 - DOIs can include a version number
- Licensing
 - check funder requirements / recommendations
 - choosealicense.com can help you decide

7. Publishing: find out more!

Software Discovery, Publication and Citation

Alexander Struck & Stephan Druskat

!! Save the date - 19th June at 1pm

Summary

1. Planning to be organised
2. Keeping data tidy
3. Methodology and protocols
4. Testing and controls
5. Automation
6. Documentation
7. Publishing

Source: carpentries-incubator.github.io/ReproducibleResearch

Summary

1. Planning to be organised
2. Keeping data tidy
3. Methodology and protocols
4. Testing and controls
5. Automation
6. Documentation
7. Publishing

Questions?