Open Science and Research Software Engineering

Workshop
Center for Advanced Internet Studies (CAIS)

Quirin Würschinger

September 21, 2023

Introduction

> `whoami`

Quirin Würschinger
q.wuerschinger@lmu.de
Wissenschaftlicher Mitarbeiter and PostDoc in (computational) linguistics
LMU Munich

Current work

- **research**
  - lexical innovation on the web and in social networks
  - variation and change in language use and social polarization in social networks

  - using Large Language Models (LLMs) like ChatGPT for research in linguistics and social science.

- **teaching**: corpus linguistics and research methodology

Promoting Open Science in (computational) linguistics at LMU

- teaching and applying reproducible corpuslinguistic methods
- creating and sharing corpora among researchers and students

---

Workshop materials

GitHub repository https://github.com/wuqui/opensciws
slides https://wuqui.github.io/opensciws/opensciws_slides.html
website version https://wuqui.github.io/opensciws/opensciws_website.html

---

`Open` Open Science workshop

Focus on …

- ask questions
- discuss
- apply and practice
- collaborate

Time table

Topic	Start	End
intro	09:00	
Open Science principles		10:30
~		
version control	10:50	
project structure		12:00
~		

Topic	Start	End
code	13:15	
data and methods		14:40
~		
publishing	15:00	
authoring		15:25
recap, open issues, feedback	16:00	16:30

Addressing different backgrounds and goals

Backgrounds and interests

CAIS: Forschung zu Digitalisierung und Digitale Gesellschaft

research fiels

- education and pedagogy
- political science
- sociology
- communications studies
- …

data and methods

- qualitative interviews
- text analysis
- quantitative surveys
- experimental designs
- social media studies
- …

Survey: main interests

- reproducible workflows
  - managing files and folders
  - programming with Python and R
  - plain text authoring
- data and methods
  - text analysis
  - social media analysis
  - questionnaires
- publishing
  - sharing data and code
  - authoring papers

Who are you?

Please briefly introduce yourself …

1. name
2. place and position
3. your research interest in about 3 sentences for someone outside your field

---

Open Science principles

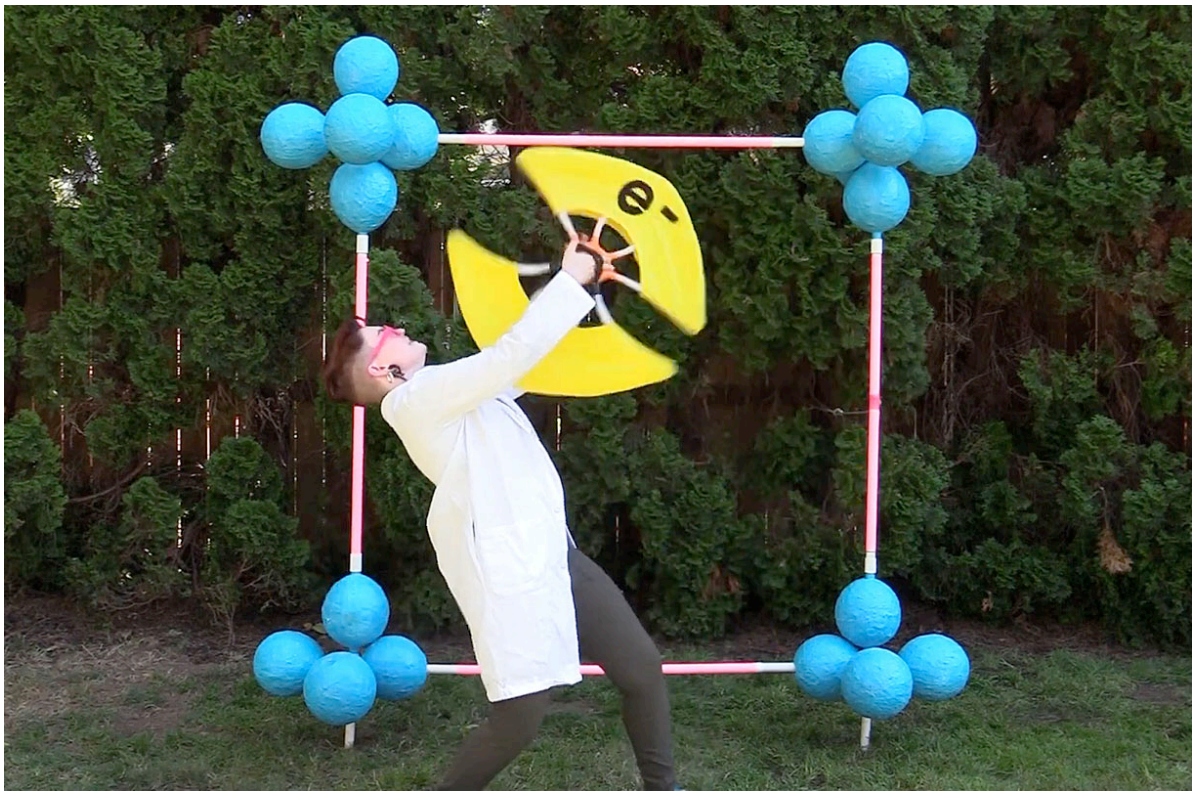
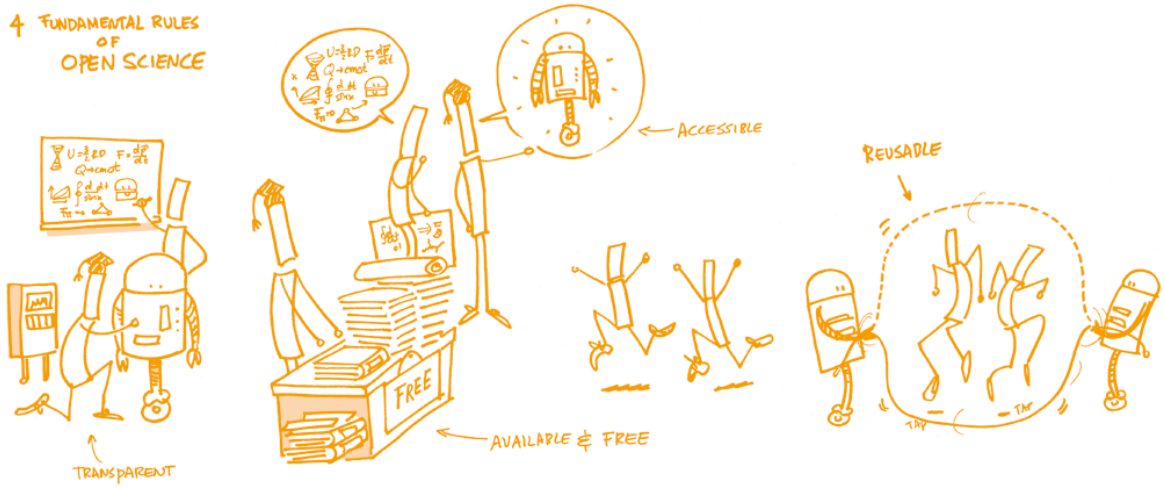


Figure 1: Dance your PhD

What is Open Science?



Why should we do Open Science?

- dataset/sample size
- effect sizes
- selection/number of relationships
- flexibility in design
- financial interests
- hype around topic/field

What are the reasons why science can go wrong?

Figure 2: source

Principles of Open Science

09:20

Open Science lifecycle

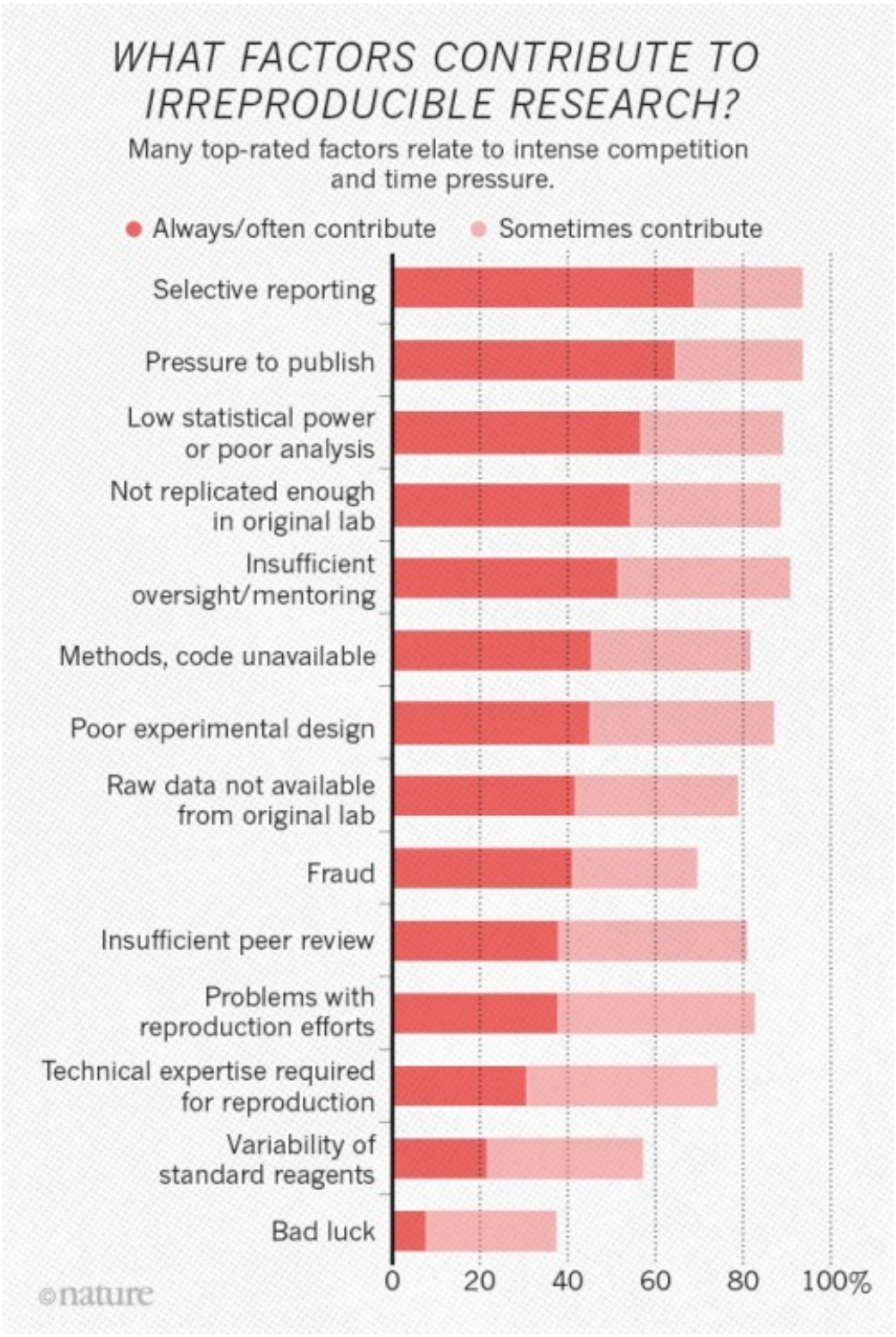


Figure 3: source

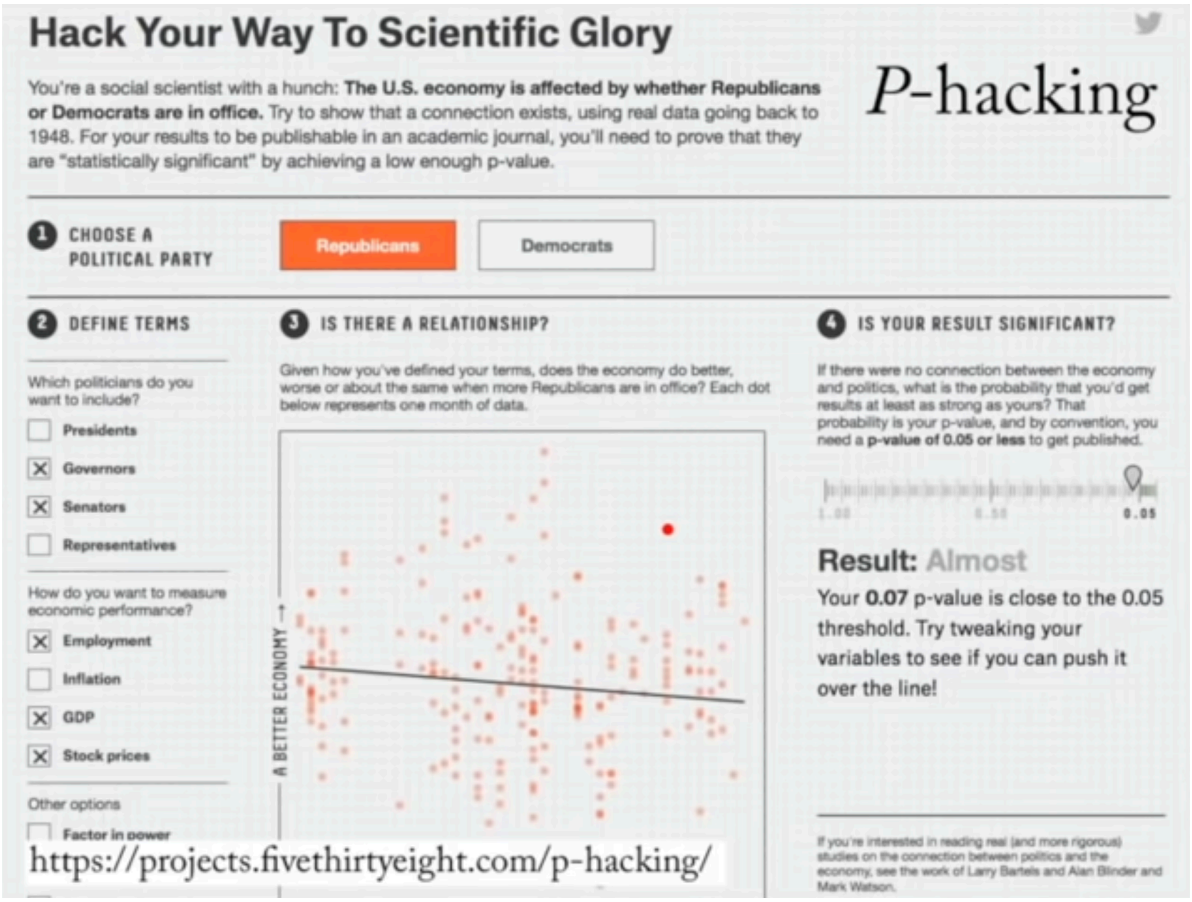


Figure 4: source

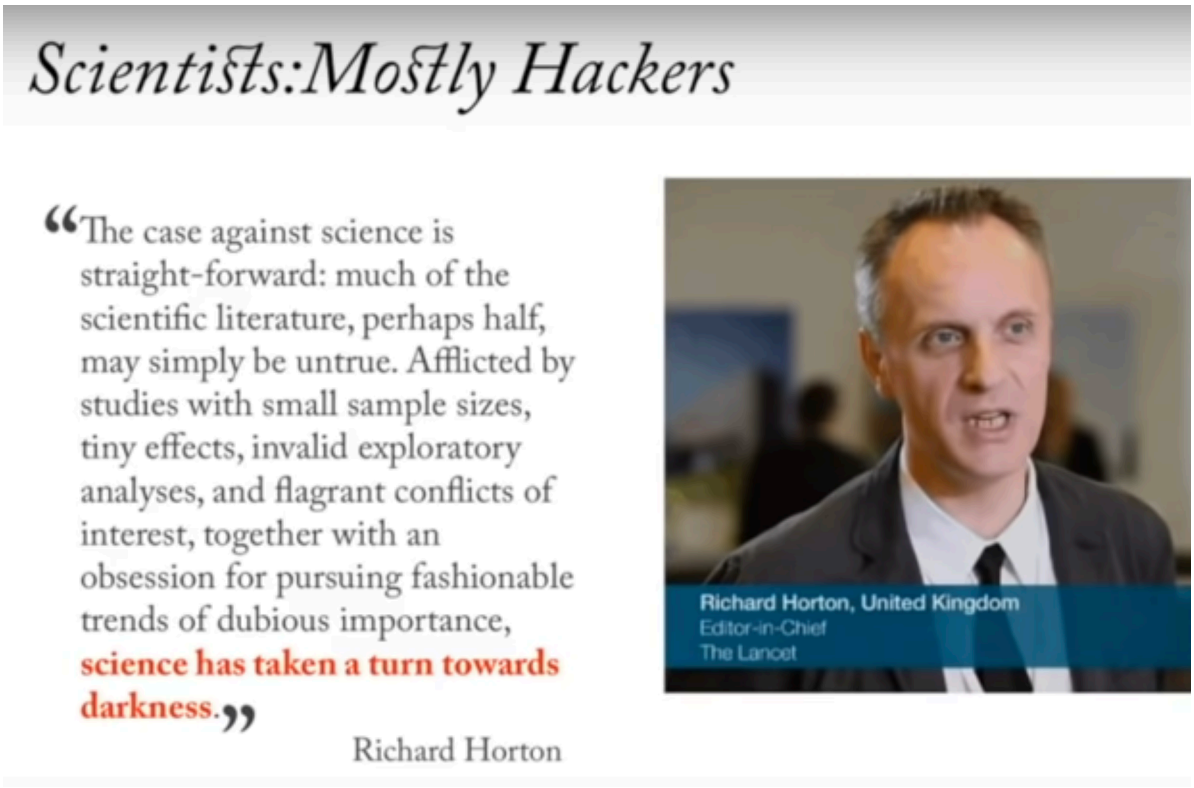


Figure 5: Richard McElreath: *Science as Amateur Software Development*

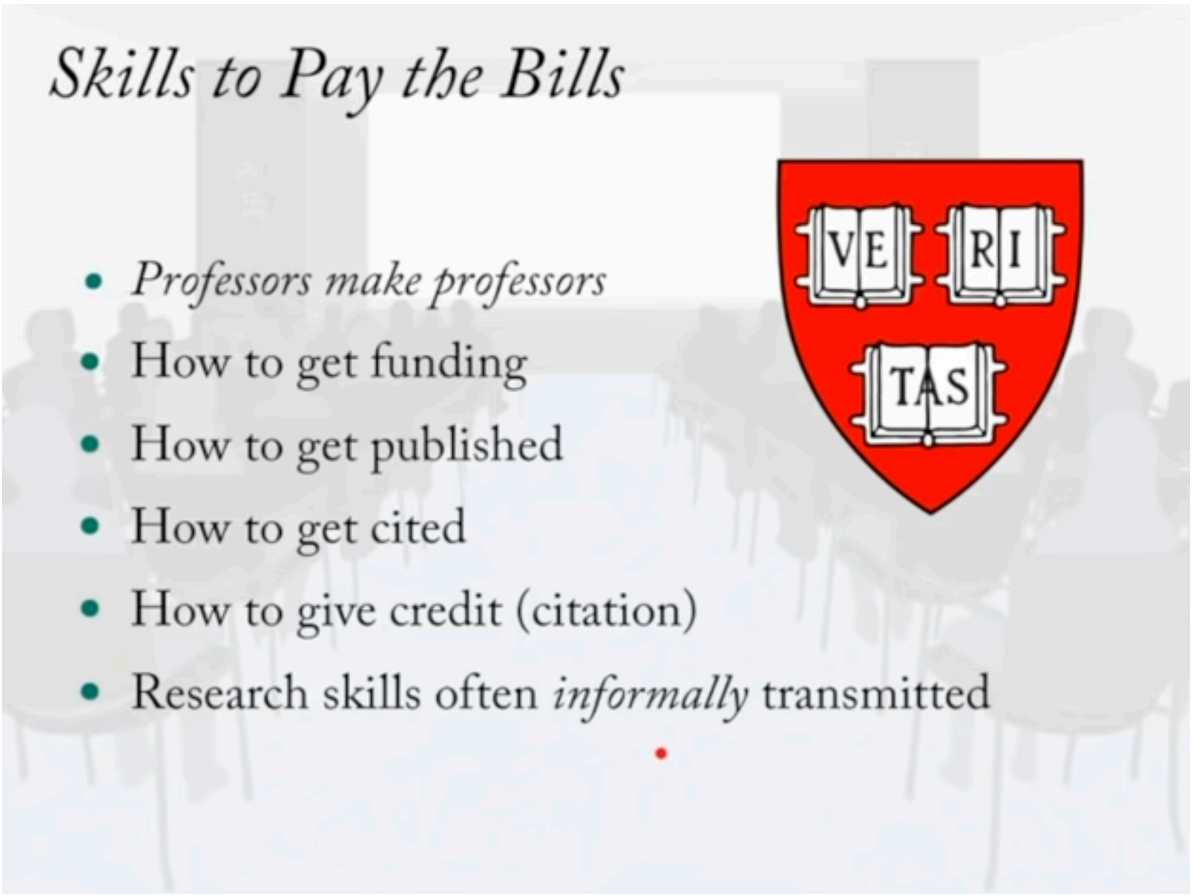
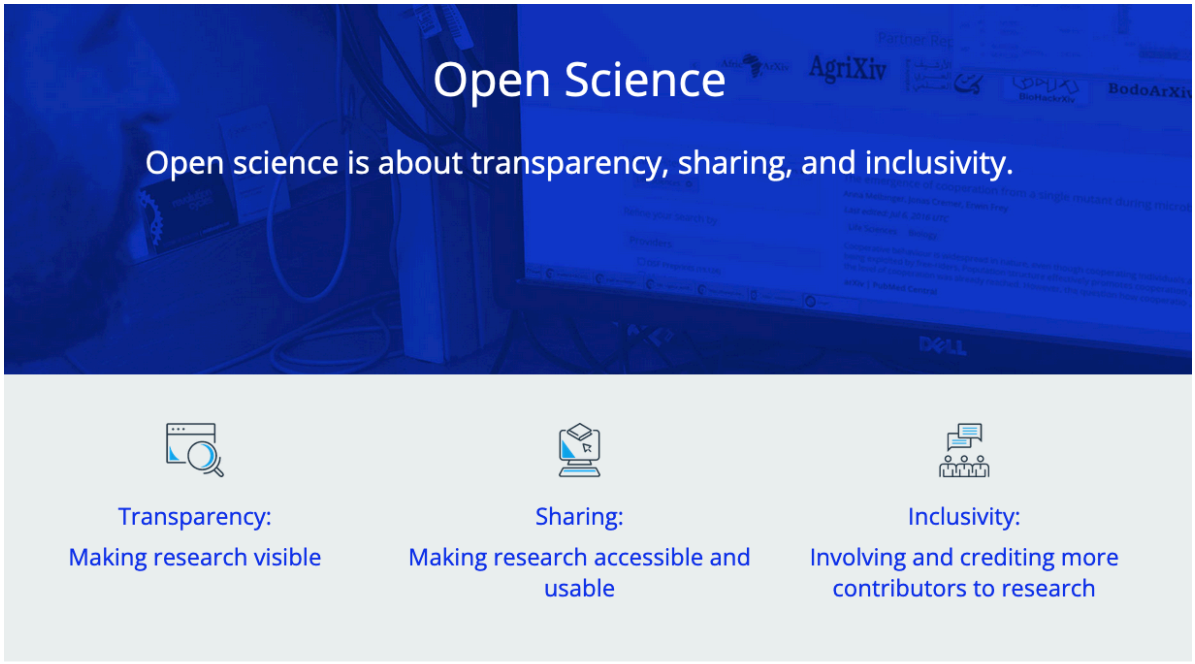


Figure 6: source



These principles aim to democratize access to research, promote equitable resource distribution, foster accountability and trustworthiness, accelerate self-correction, and improve rigor and reproducibility.

Figure 7: Center for Open Science

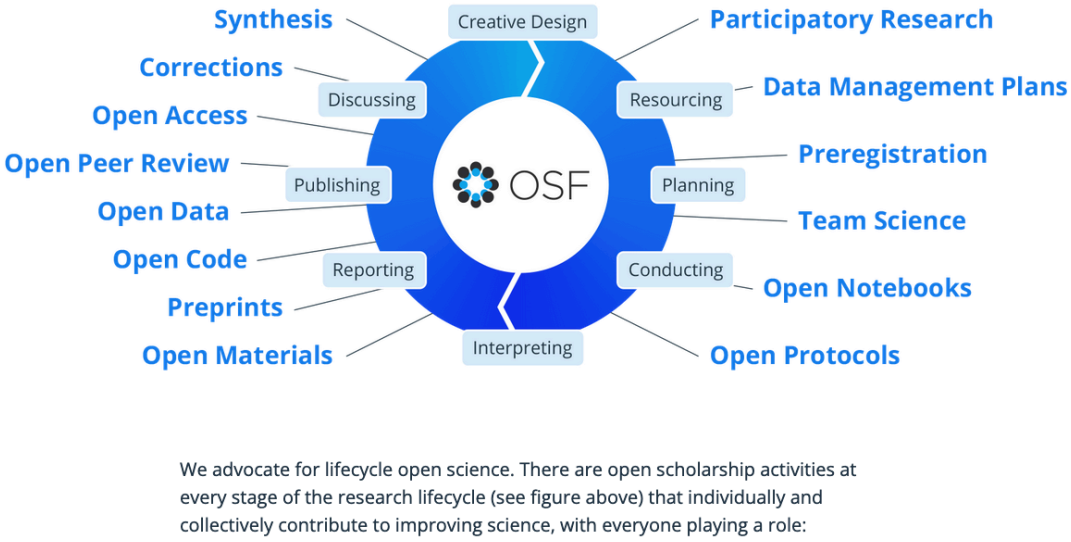


Figure 8: Center for Open Science

Roles in Open Science

Funders make open science part of the selection process, and conditions for grantees conducting research.

Publishers make open science part of the review process, and conditions for articles published in their journals.

Institutions make open science part of academic training, and part of the selection process for research positions and evaluation for advancement and promotion.

Societies make open science part of their awards, events, and scholarly norms.

Researchers enact open science in their work and advocate for broader adoption in their communities.

(Center for Open Science)

Who profits from Open Science?

Figure 9: source

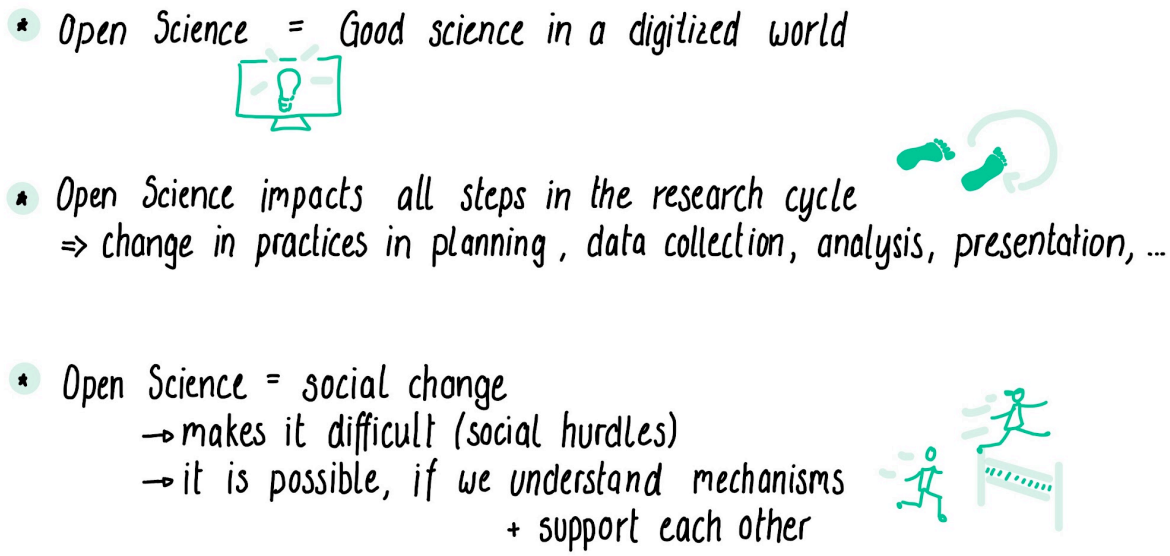
What is Open Science to you?

What do you find interesting, important, or attractive about Open Science?

https://tinyurl.com/opnsci

09:50

Learning outcomes



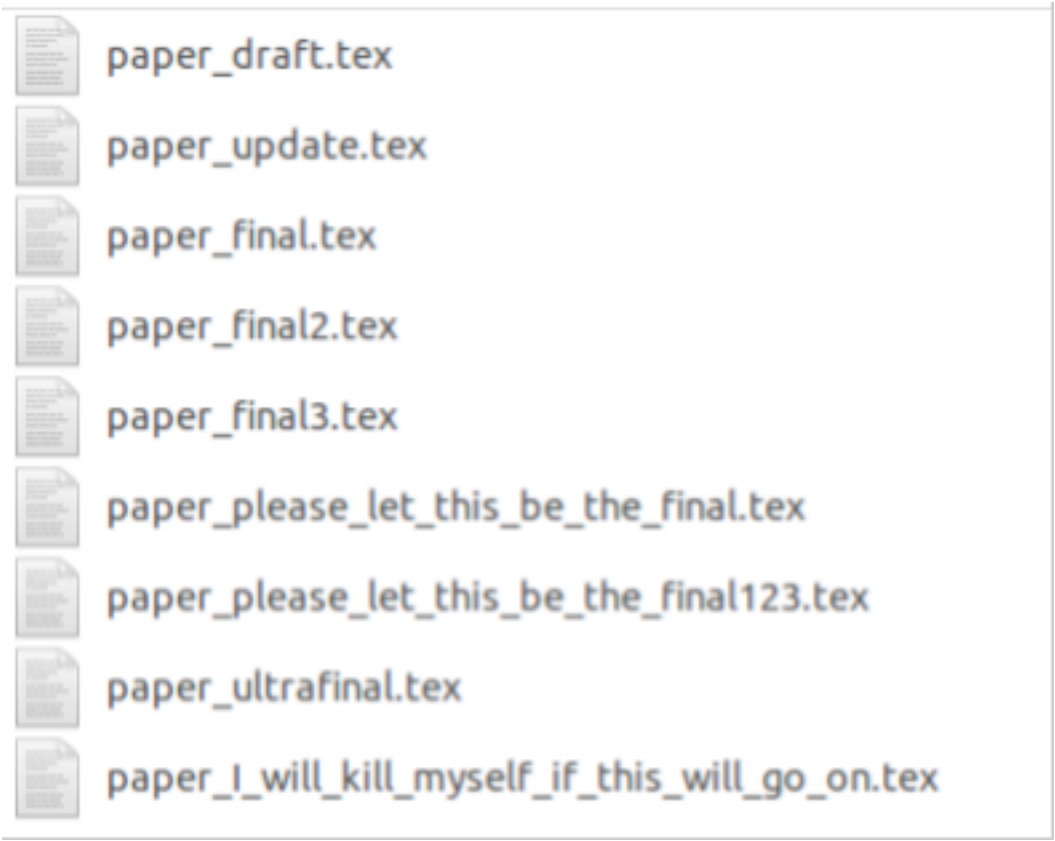
Implementing an open and reproducible workflow

1. version control
2. project structure
3. code

4. data and methods
5. authoring
6. publishing

Version control

Why use version control?



Dear colleagues,

attached you find the first public version of the [redacted] protocol.
Please have a look and do comment. We can also meet to aggregate our reviews.

1 attachment: StudyProposa[redacted]_Validation_V1_250918docx.docx

source

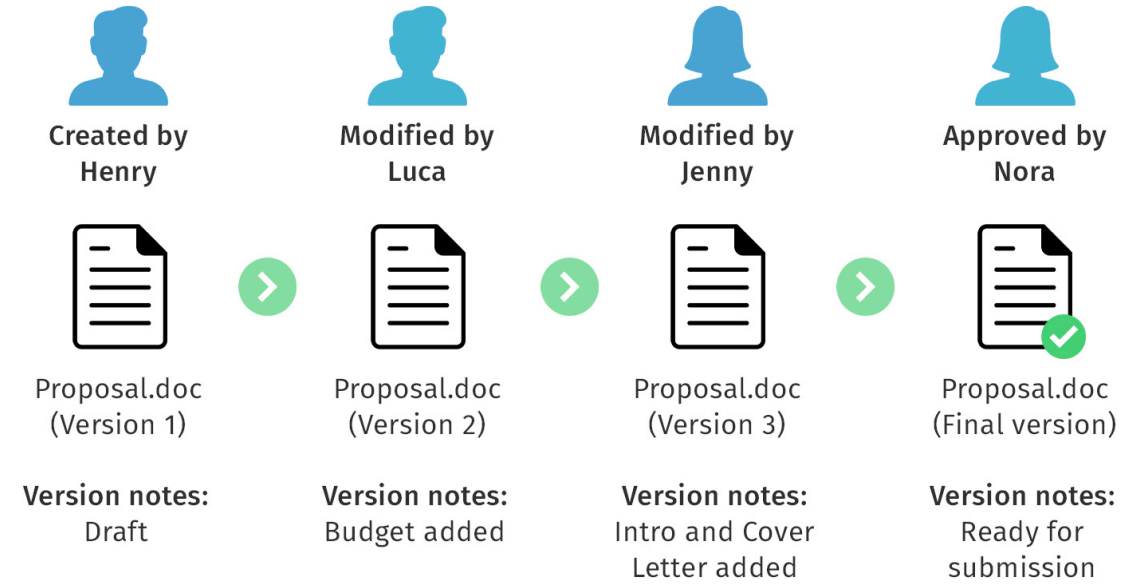


Figure 10: source

---

git and GitHub/GitLab

git software on your machine

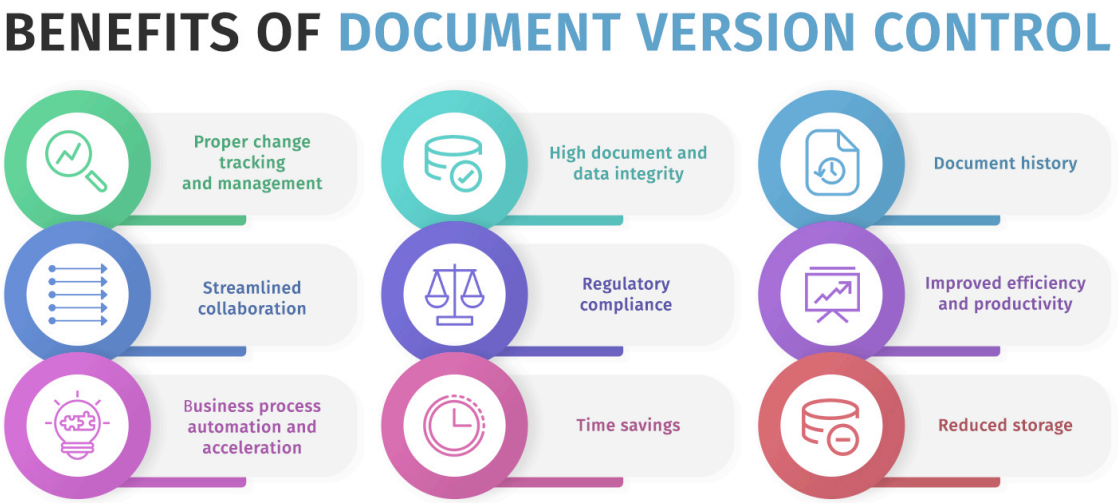


Figure 11: source

```
git add src/tests.py
git commit -m 'add tests'
git push
```

GitHub and GitLab services on a remote server

Collaborating using GitHub

git commands

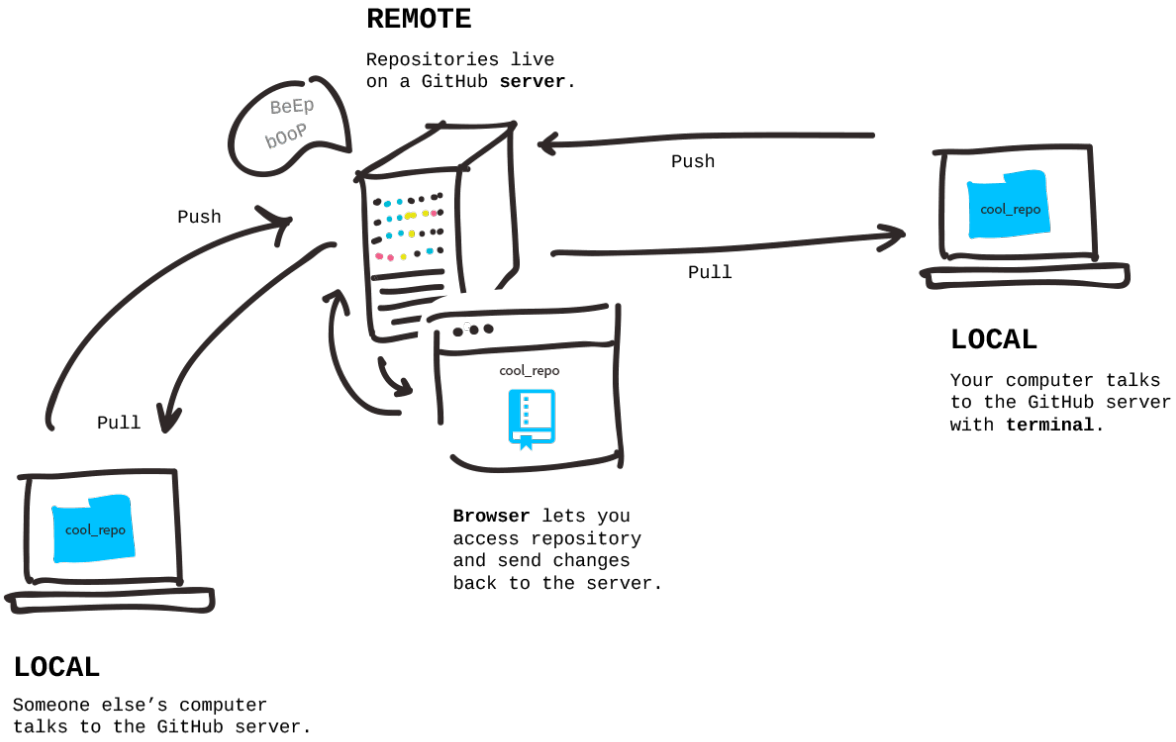


Figure 12: (source)

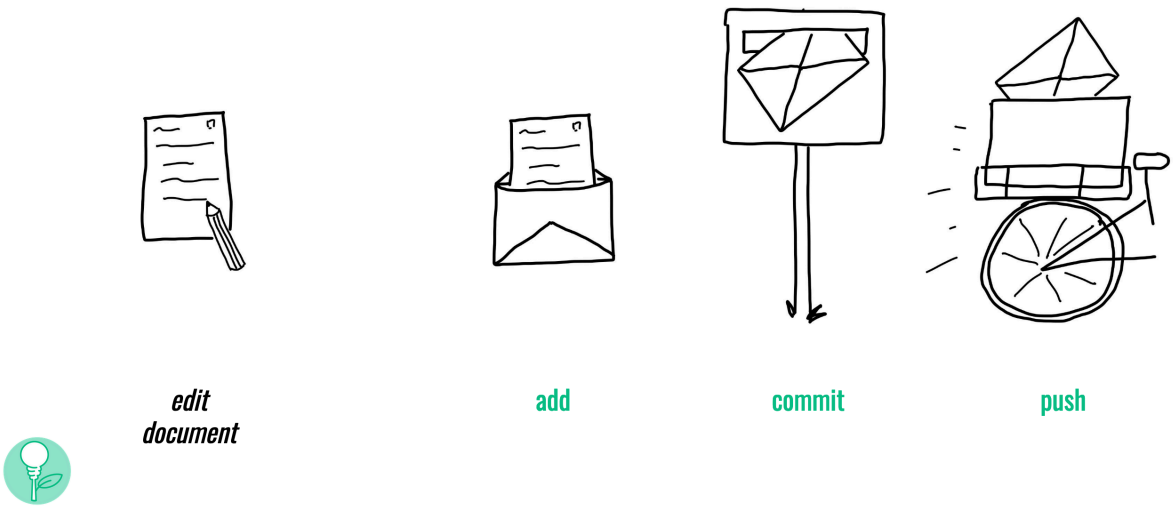


Figure 13: (source)

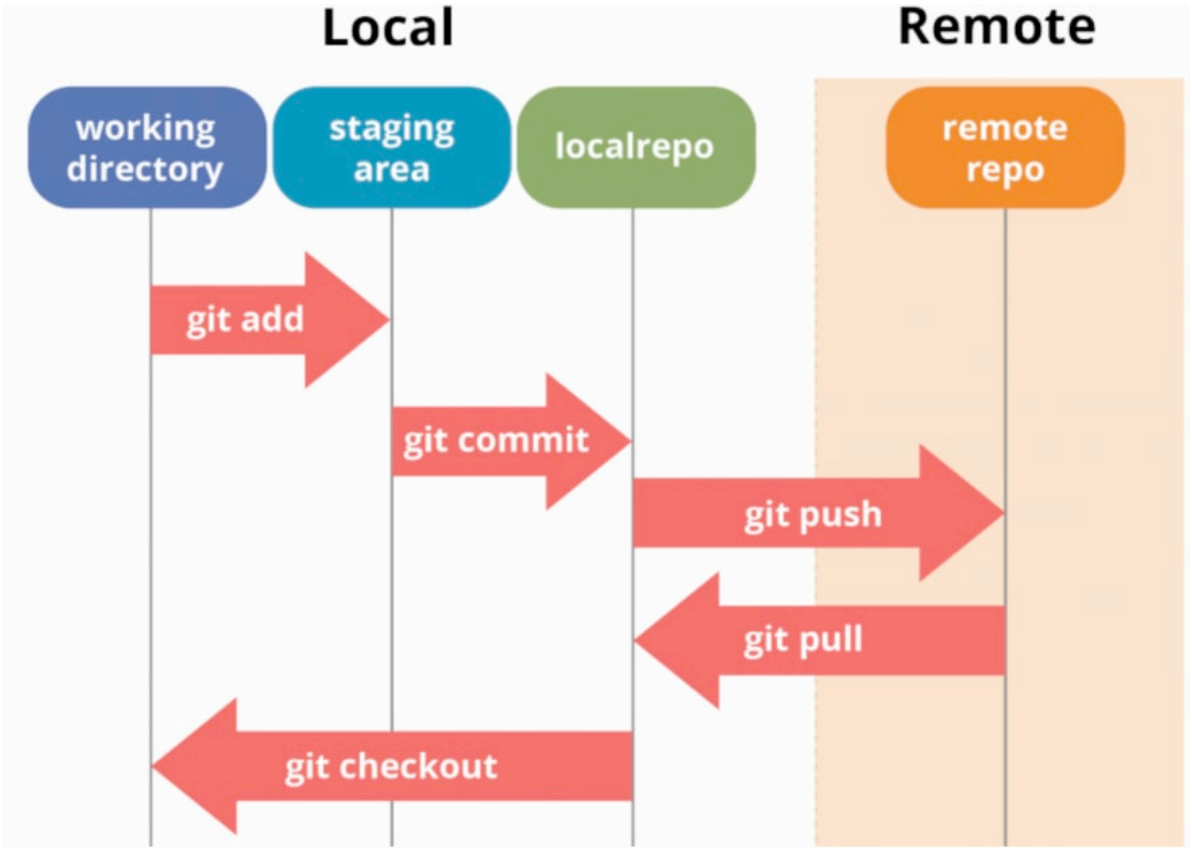


Figure 14: (source)

GitHub workflow

Example

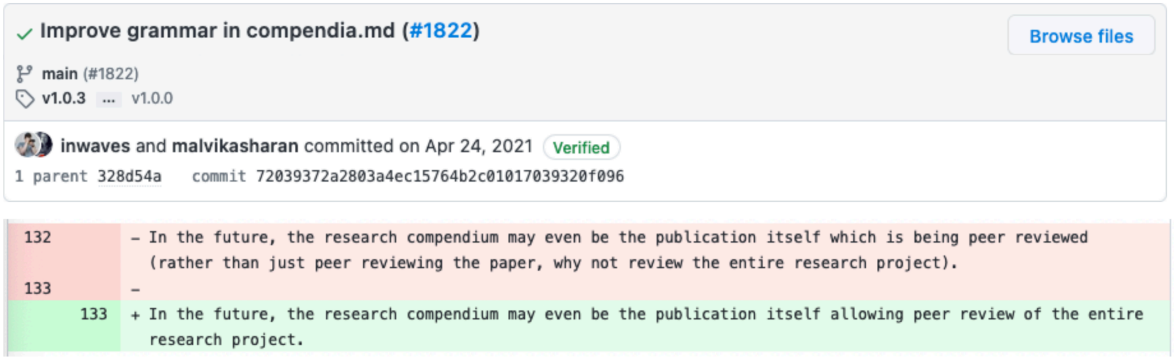


Figure 15: source

How to set up a GitHub repository

---

set up `git`

Installing `git`: see tutorial

Using `git`:

- from the command line
- using a standalone GUI[1] tool; e.g.:
  - GitKraken
  - GitHub Desktop
- from within your editor/IDE[2]; e.g.:

[1] Graphical User Interface
[2] Integrated Development Environment

- RStudio
- VSCode

---

set up `GitHub`

tutorial

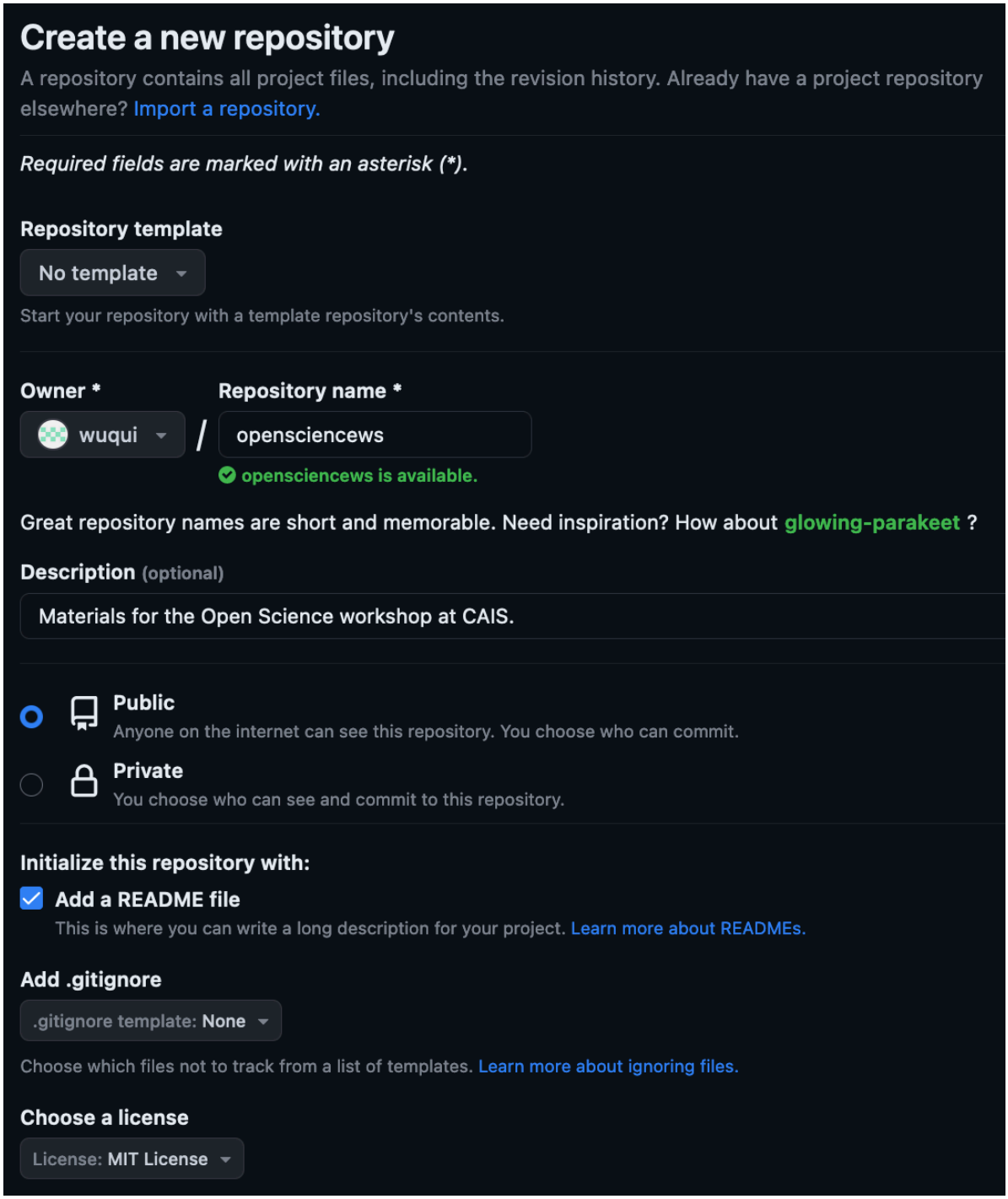
- setting up git user information (name, passwort)
- setting up GitHub authentication
- setting and storing authentication ('token')

---

create a repository on GitHub

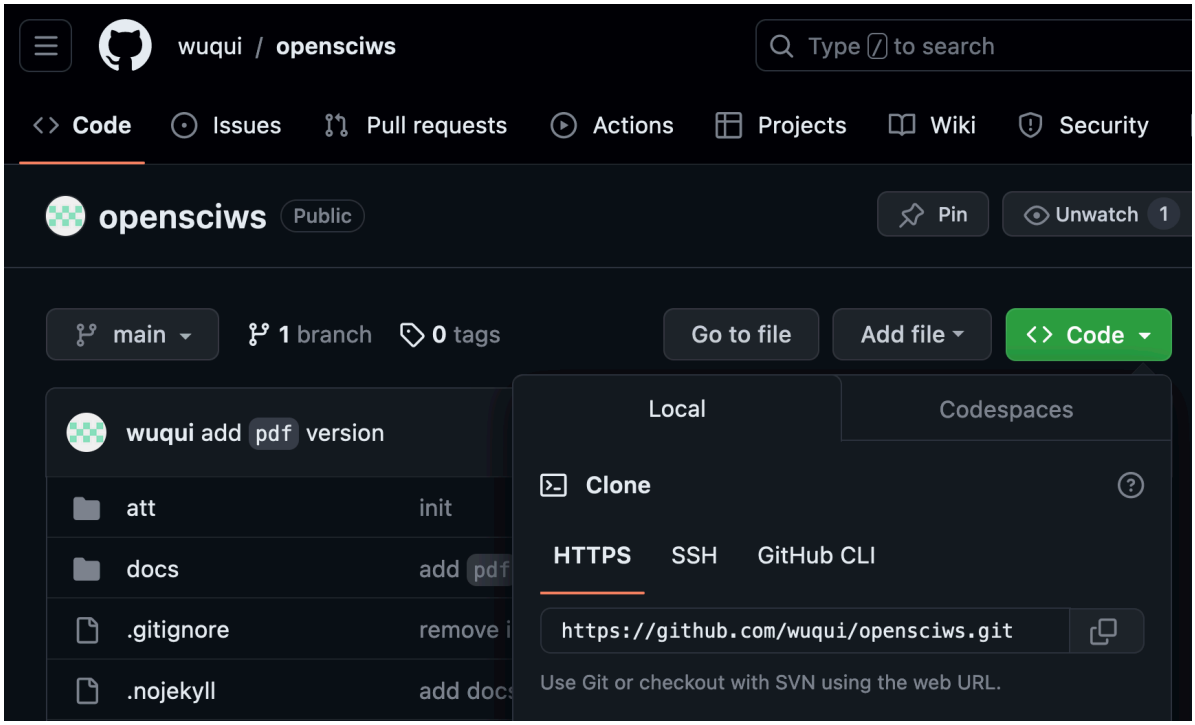
1. (create GitHub account)
2. click on `New` (https://github.com/new)
3. specify repo name [3]
4. specify description
5. specify visibility: private or public
6. select `Add a README file`
7. specify licence [4]

[3] safe: lowercase alphabet characters
[4] good choice for many purposes: MIT license


Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.
Required fields are marked with an asterisk (*).
Repository template
No template
Start your repository with a template repository's contents.
Owner *
Repository name *
wuqui
/
opensciencews
opensciencews is available.
Great repository names are short and memorable. Need inspiration? How about glowing-parakeet ?
Description (optional)
Materials for the Open Science workshop at CAIS.
Public
Anyone on the internet can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Initialize this repository with:
Add a README file
This is where you can write a long description for your project. Learn more about READMEs.
Add .gitignore
.gitignore template: None
Choose which files not to track from a list of templates. Learn more about ignoring files.
Choose a license
License: MIT License

`clone` repositories

go to the folder where you want your project to live




```
git clone https://github.com/wuqui/opensciws.git
```

---

adding, `commit`ing, and `push`ing changes

```
git add src/tests.py
git commit -m 'add tests'
git push
```

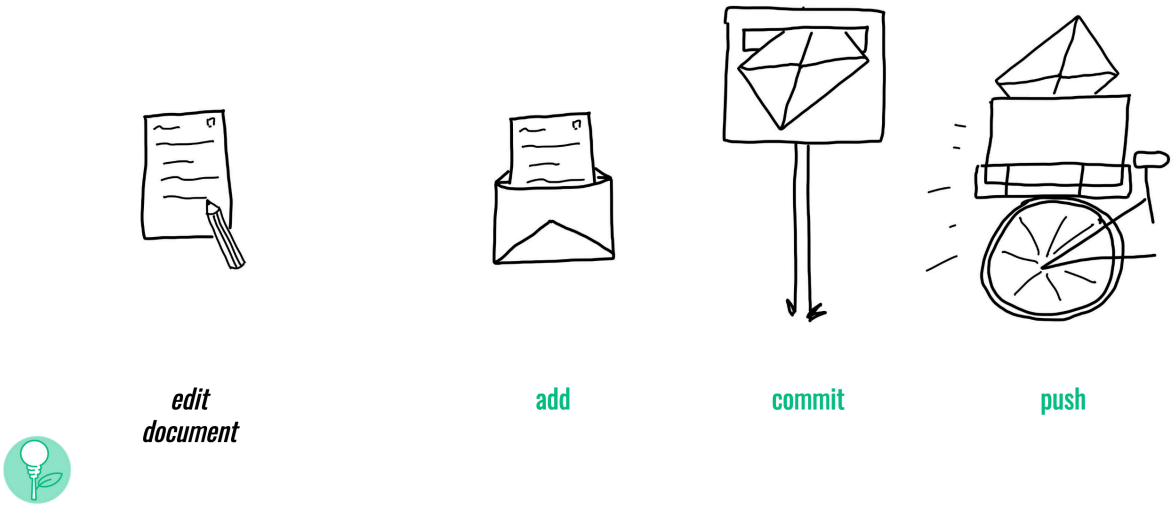
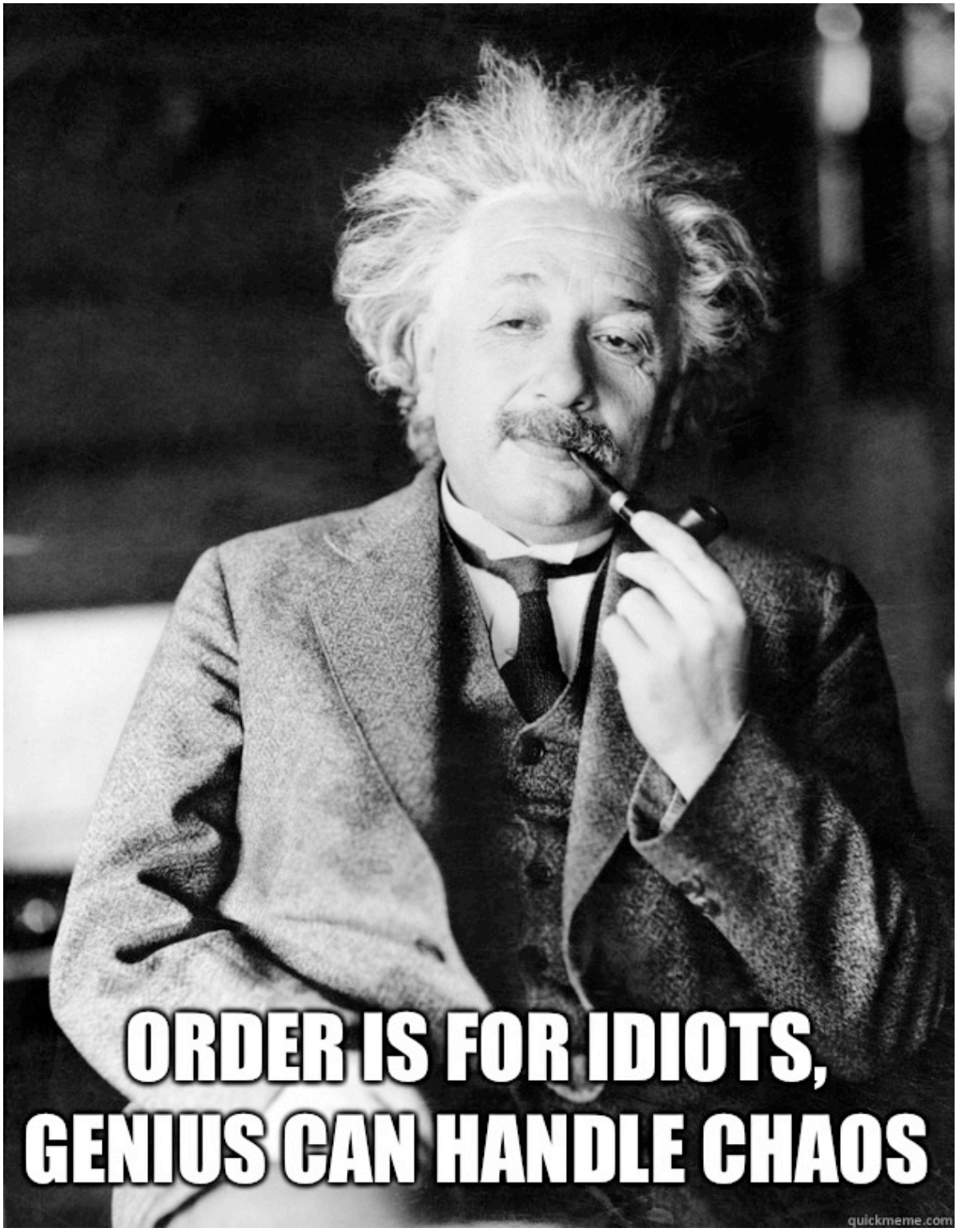


Figure 16: (source)

Project structure


ORDER IS FOR IDIOTS,
GENIUS CAN HANDLE CHAOS
quickmeme.com

Let's not pretend we're all geniuses ...

File names

NO

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

YES

- `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`

File names should be:

- machine-readable
- human-readable
- consistent
- optional: play well with default ordering (e.g. include timestamps)

File structure

```
.
  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
```

Practice: project management

You have until 11:50 h to work on either …

1. developing a project structure for your needs from scratch
2. refactoring/cleaning an existing project[5]

Optionally: set up version control via git/GitHub for this project.

Code

Reproducibility

Reproducibility (crisis)

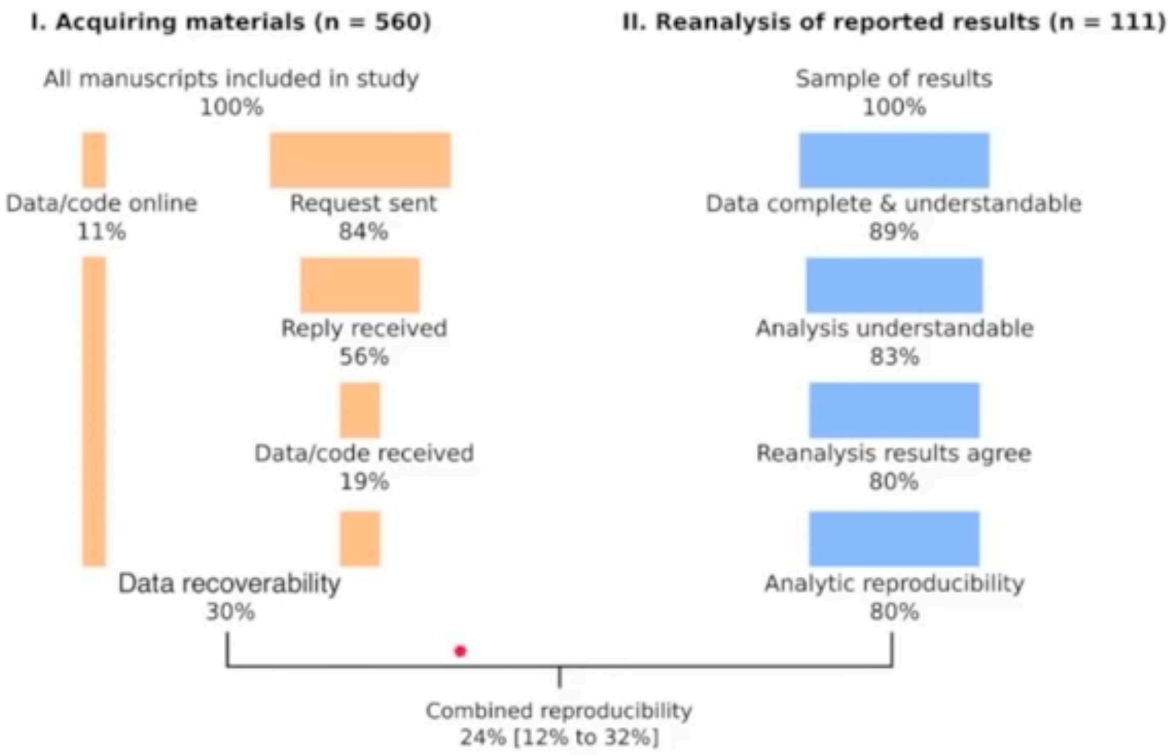


Figure 17: source

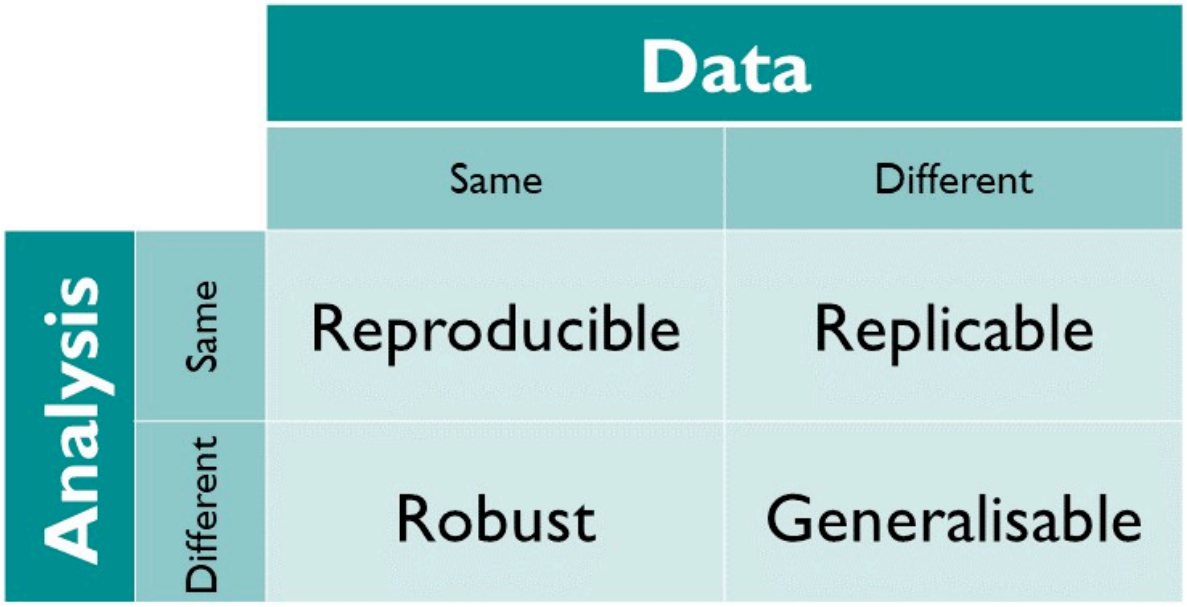
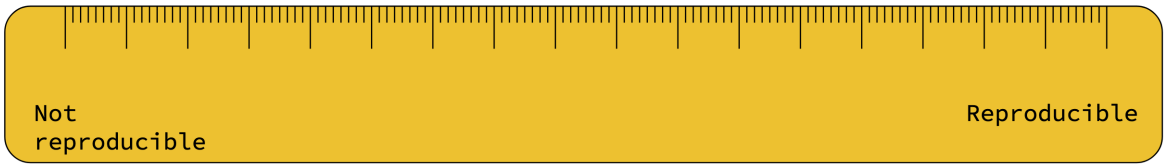


Figure 18: The Turing Way

Reproducibility et al.



interesting for student projects

The quality of tools

SCIENCE / TECH / MICROSOFT

Scientists rename human genes to stop Microsoft Excel from misreading them as dates

/ Sometimes it's easier to rewrite genetics than update Excel

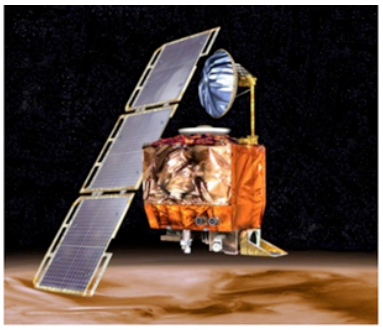
By James Vincent, a senior reporter who has covered AI, robotics, and more for eight years at The Verge.

Aug 6, 2020, 2:44 PM GMT+2 |

Figure 19: source

LISA GROSSMAN 11.10.10 7:00 AM

NOV. 10, 1999: METRIC MATH MISTAKE MUFFED MARS METEOROLOGY MISSION



The **$125 million satellite** was supposed to be the first weather observer on another world. But as it approached the red planet to slip into a stable orbit Sept. 23, the orbiter vanished. Scientists realized quickly it was gone for good. "It was pretty clear that morning, within half-an-hour, that the spacecraft had more or less **hit the top of the atmosphere and burned up**," recalled NASA engineer Richard Cook, who was project manager for Mars exploration projects at the time.

A NASA review board found that the problem was in the software controlling the orbiter's thrusters. **The software calculated the force the thrusters needed to exert in *pounds* of force. A separate piece of software took in the data assuming it was in the metric unit: *newtons*.**

https://www.wired.com/2010/11/1110mars-climate-observer-report/

Figure 20: source

Testing code

Why we should test code

---

not all our projects have that high stakes

Professional testing

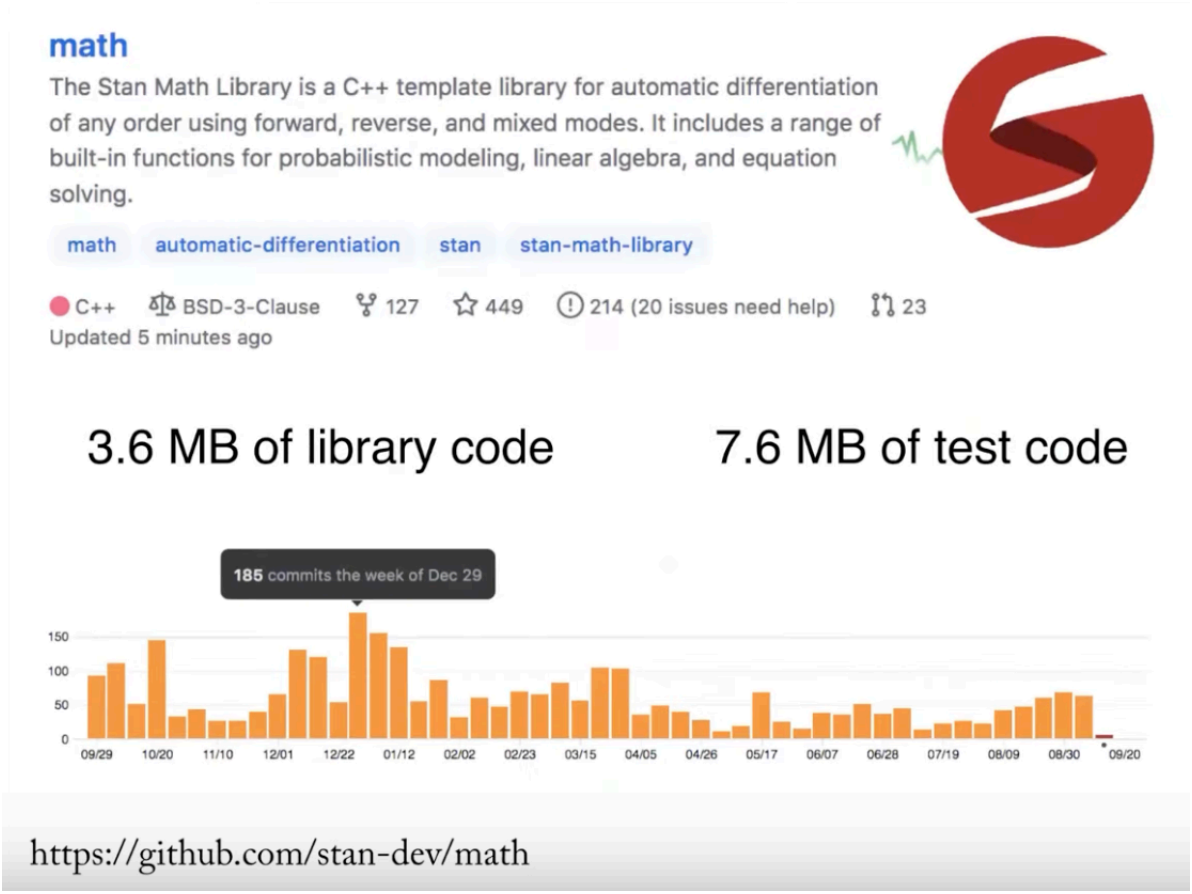


Figure 21: source

same is true of industry

---

[5]make a backup first

Types of tests

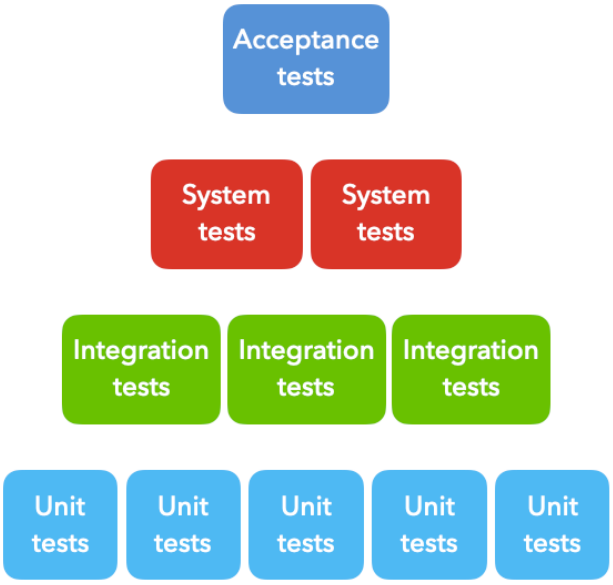


Figure 22: source

Testing frameworks

- Python: `pytest`
- R: `testthat`

Analogy

- during the process of manufacturing a ballpoint pen, the cap, the body, the tail, the ink cartridge and the ballpoint are produced separately and **unit tested** separately.
- When two or more units are ready, they are assembled and **integration testing** is performed, for example a test to check the cap fits on the body.
- When the complete pen is integrated, **system testing** is performed to check it can be used to write like any pen should.
- **Acceptance testing** could be a check to ensure the pen is the colour the customer ordered.

source

Testing example

using `pytest` for Python

Python

```python
# test_assert_examples.py

def test_uppercase():
    assert "loud noises".upper() == "LOUD NOISES"

def test_reversed():
    assert list(reversed([1, 2, 3, 4])) == [4, 3, 2, 1]

def test_some_primes():
    assert 37 in {
        num
        for num in range(2, 50)
        if not any(num % div == 0 for div in range(2, num))
    }
```

Documenting code

Literate programming

- 'Literate programming is a methodology that combines a programming language with a **documentation language**,
- thereby making programs **more robust**, **more portable**, **more easily maintained**,
- and arguably **more fun** to write than programs that are written only in a high-level language.
- The main idea is to treat a program as a piece of literature, **addressed to human beings** rather than to a computer.
- The program is also viewed as a **hypertext document**, rather like the World Wide Web. (Indeed, I used the word WEB for this purpose long before CERN grabbed it!)'

Donald Knuth

psychological benefit: conversation - helps reasoning - more fun (human) - → ChatBots

---

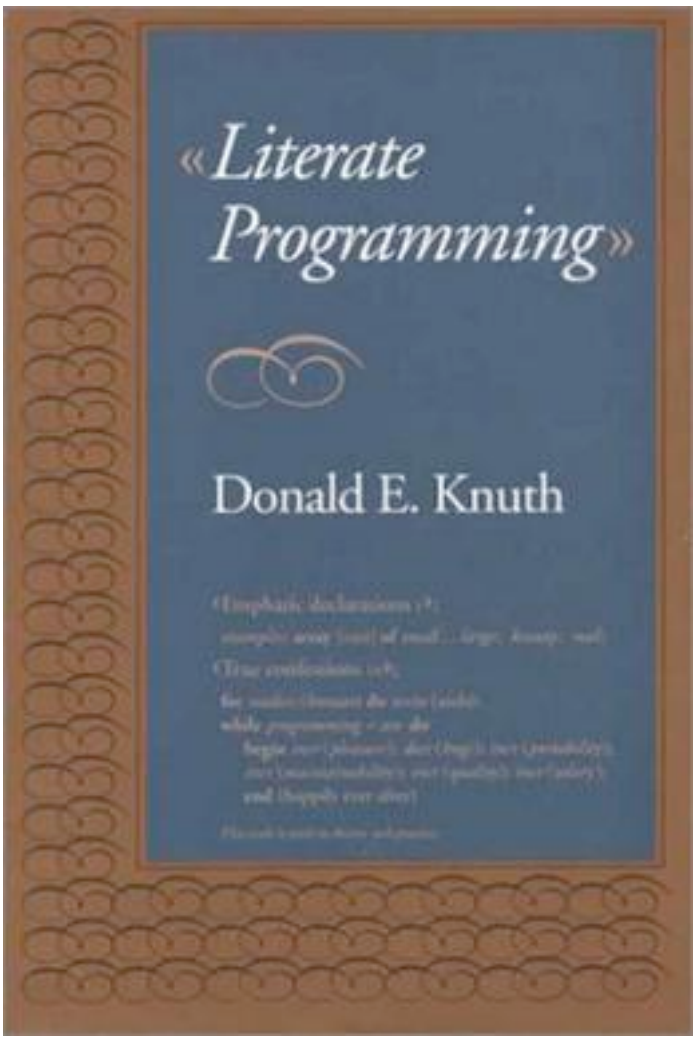


Figure 23: source

Notebooks

Python

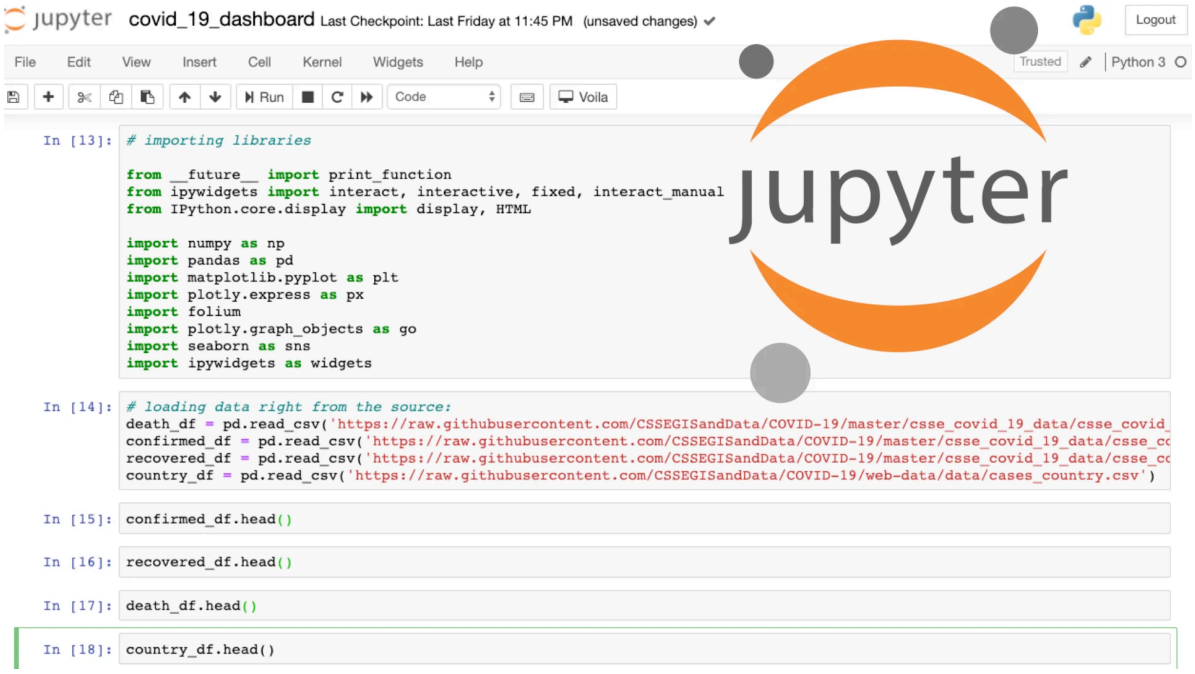


Figure 24: source

R

→ both work with Quarto

- who uses notebooks?
- which ones?
- good for novices & experts

Example using `nbdev` for Python

Programming a deck of cards: https://github.com/fastai/nbdev_cards/

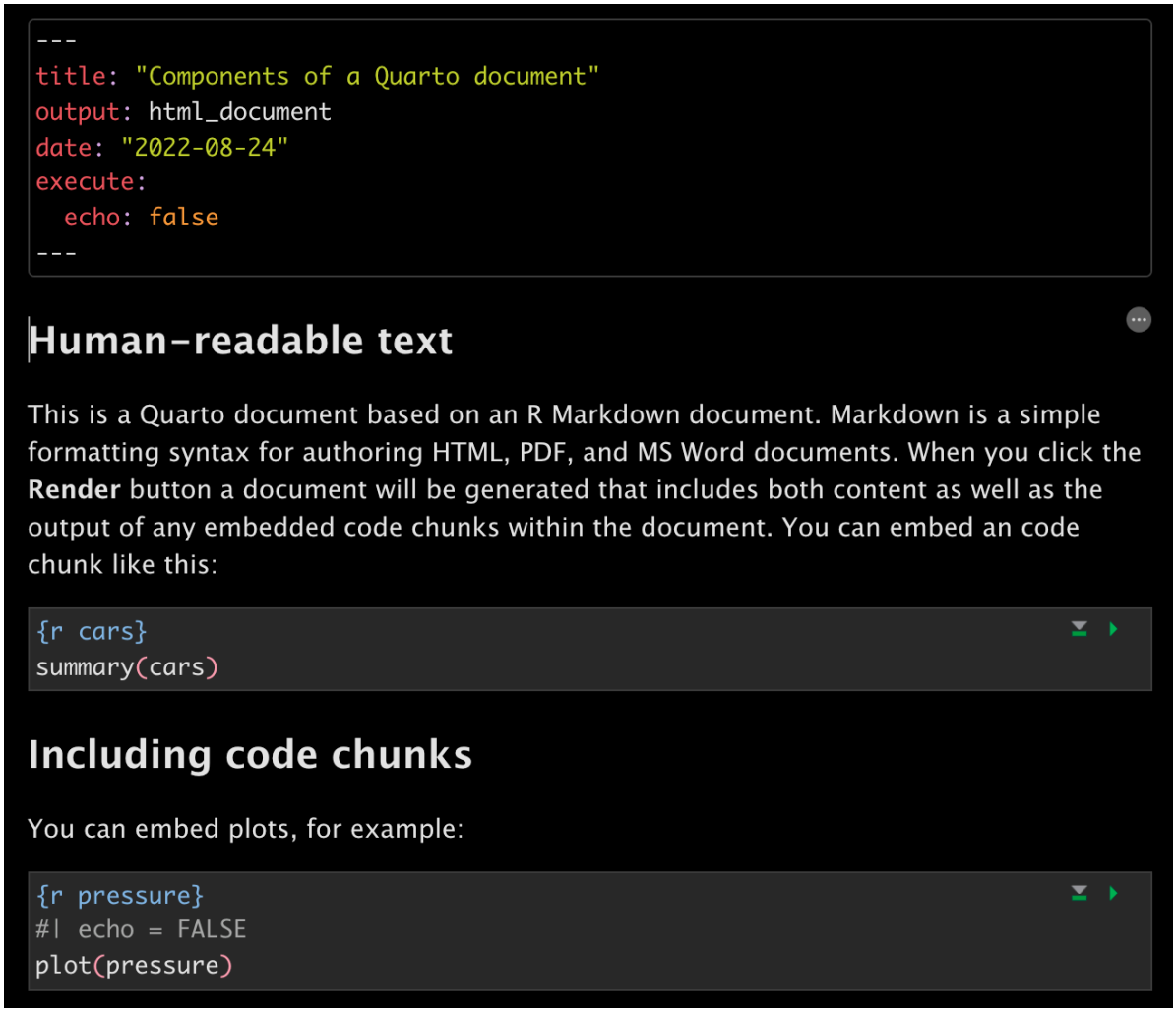


Figure 25: source

In [ ]:
```
#| export
suits = ["♣","♦","❤","♠"]
ranks = [None, "A"] + [str(x) for x in range(2,11)]  + ["J", "Q", "K"]
```

We will be using numbers to represent playing card clubs and ranks. These are the suits:

In [ ]:
```
suits
```

Out[ ]:
```
['♣', '♦', '❤', '♠']
```

For instance the suit at index `0`:

In [ ]:
```
suits[0]
```

Out[ ]:
```
'♣'
```

These are the ranks:

In [ ]:
```
ranks
```

Out[ ]:
```
[None, 'A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K']
```

For instance the rank at index `1` (note that there isn't a playing card at position `0`, since we want the ranks to match the indicies where possible):

Literate testing with `nbdev`

For instance, here's a test of equality...

In [ ]:
```
test_eq(Card(suit=1, rank=3), Card(suit=1, rank=3))
```

In [ ]:
```
#| hide
test_ne(Card(suit=2, rank=3), Card(suit=1, rank=3))
test_ne(Card(suit=1, rank=2), Card(suit=1, rank=3))
```

...and a test of `<` ...

In [ ]:
```
assert Card(suit=1, rank=3)
```

...and finally of `>` :

In [ ]:
```
assert Card(suit=3, rank=3)>Card(suit=2, rank=3)
assert not Card(suit=1, rank=3)>Card(suit=2, rank=3)
```

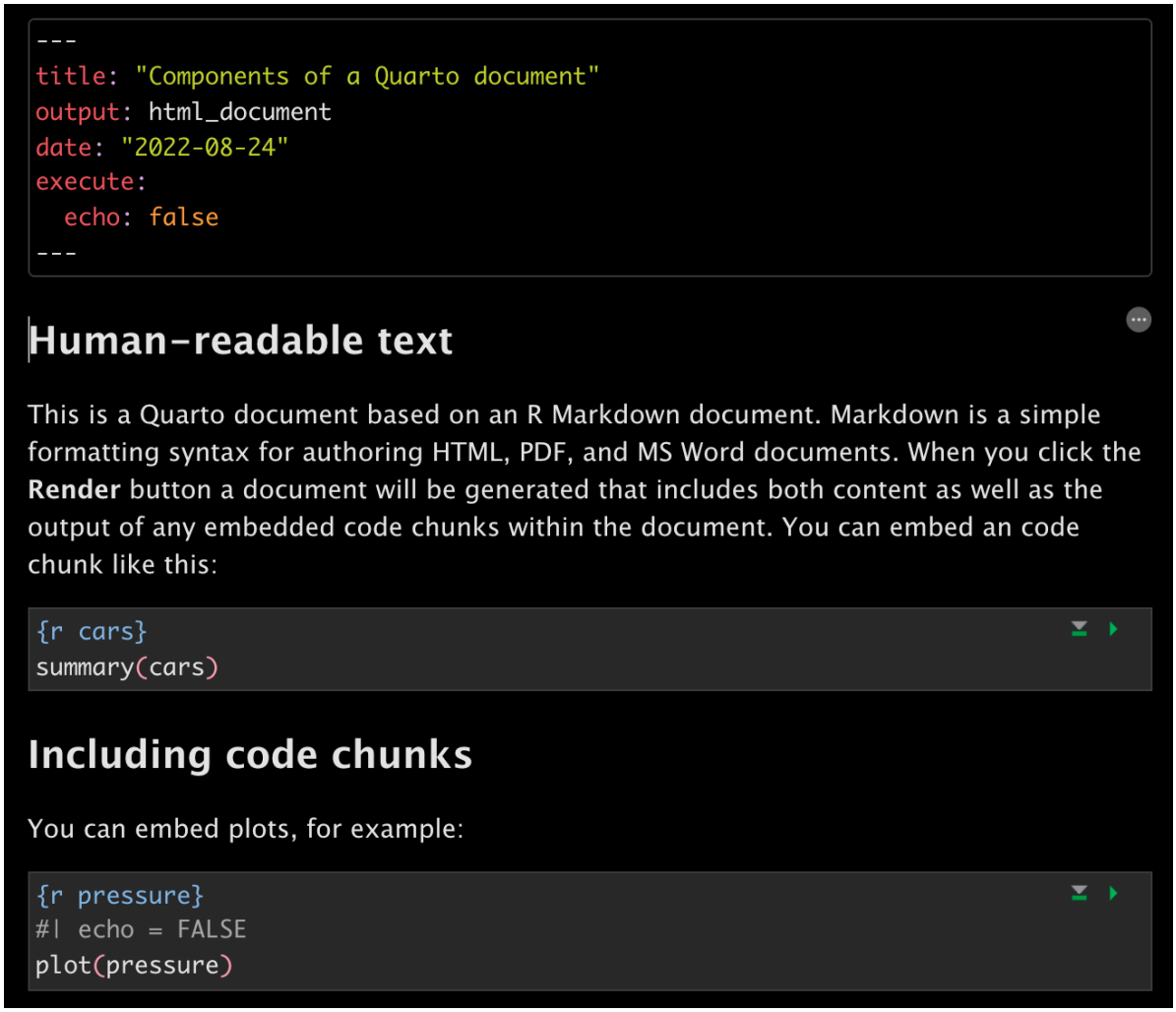
compare with `pytest`: much easier/natural

Additional benefits of `nbdev`

- simple, integrated testing
- continuous integration
- dependency management
- publishing code for PyPI and conda
- publishing documentation via Quarto

- good for novices & experts
- covers about 80% of programming setup for free
- more about Quarto later

R: Quarto and RMarkdown



benefit: visual editor

Licensing

Code

often appropriate: MIT license

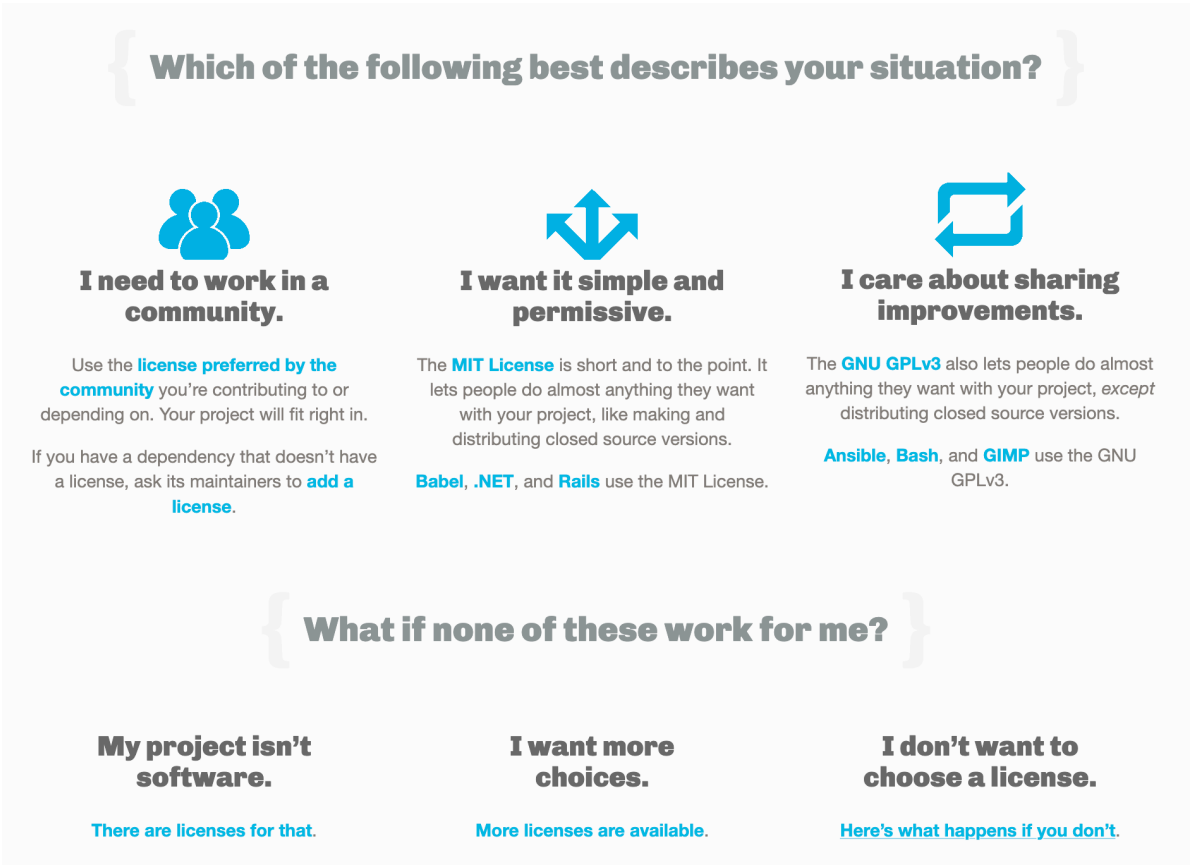


Figure 26: https://choosealicense.com

Other materials

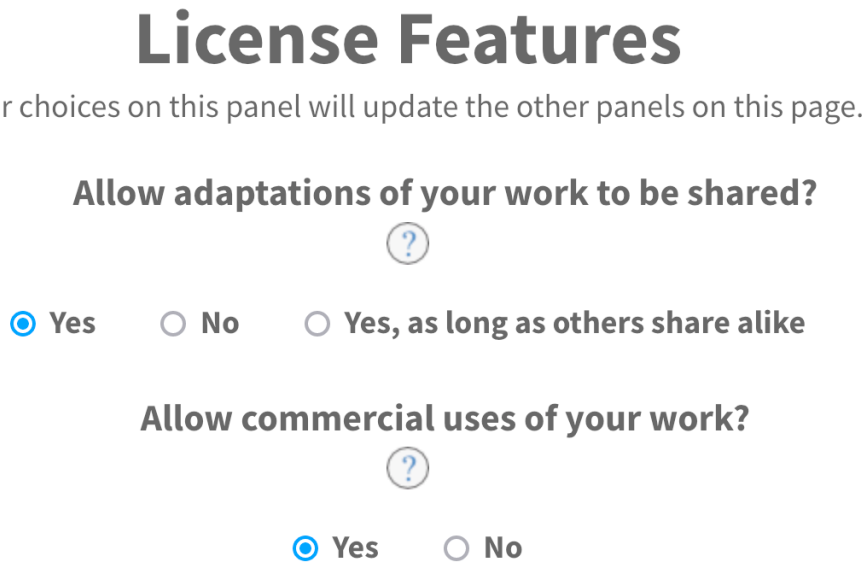


Figure 27: https://creativecommons.org/choose/

Data and methods

Diversity in data and methods

CAIS: Forschung zu Digitalisierung und Digitale Gesellschaft

research fiels

- education and pedagogy
- political science
- sociology
- communications studies
- ...

data and methods

- qualitative interviews
- text analysis
- quantitative surveys
- experimental designs
- social media studies
- ...

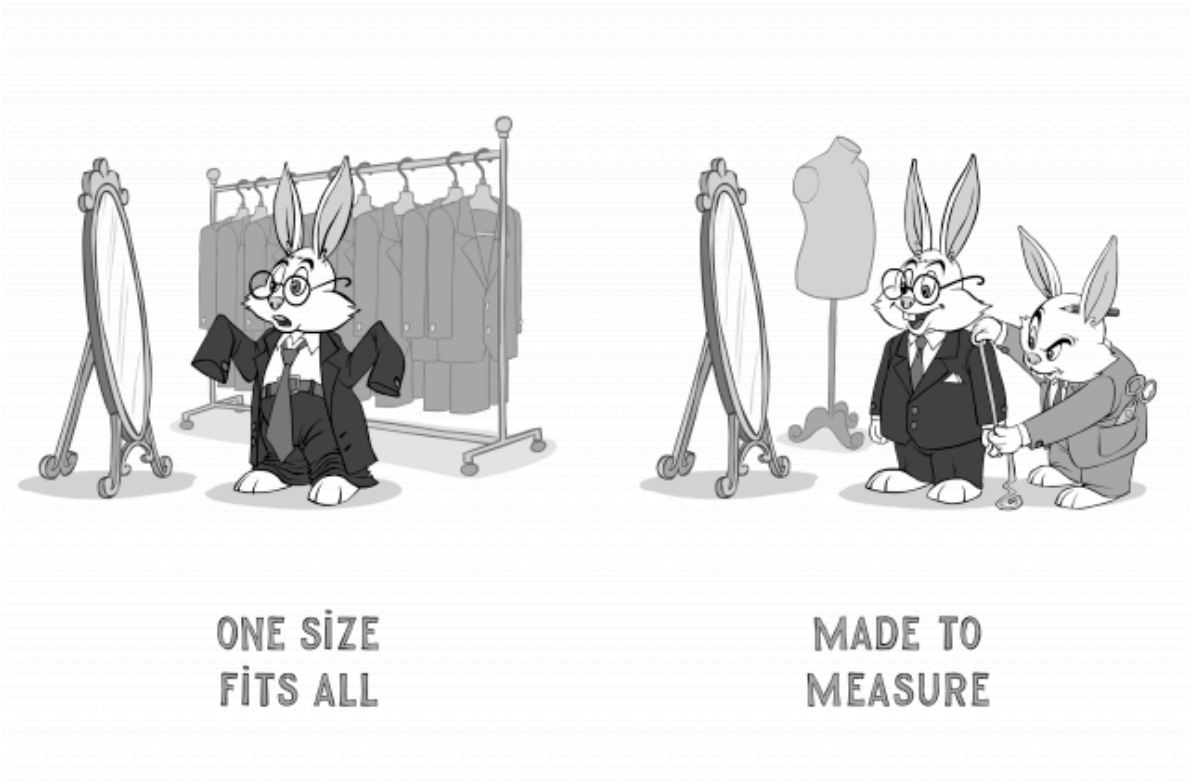


Figure 28: source

Reasons to share your data

- To allow the possibility to fully **reproduce** a scientific study.

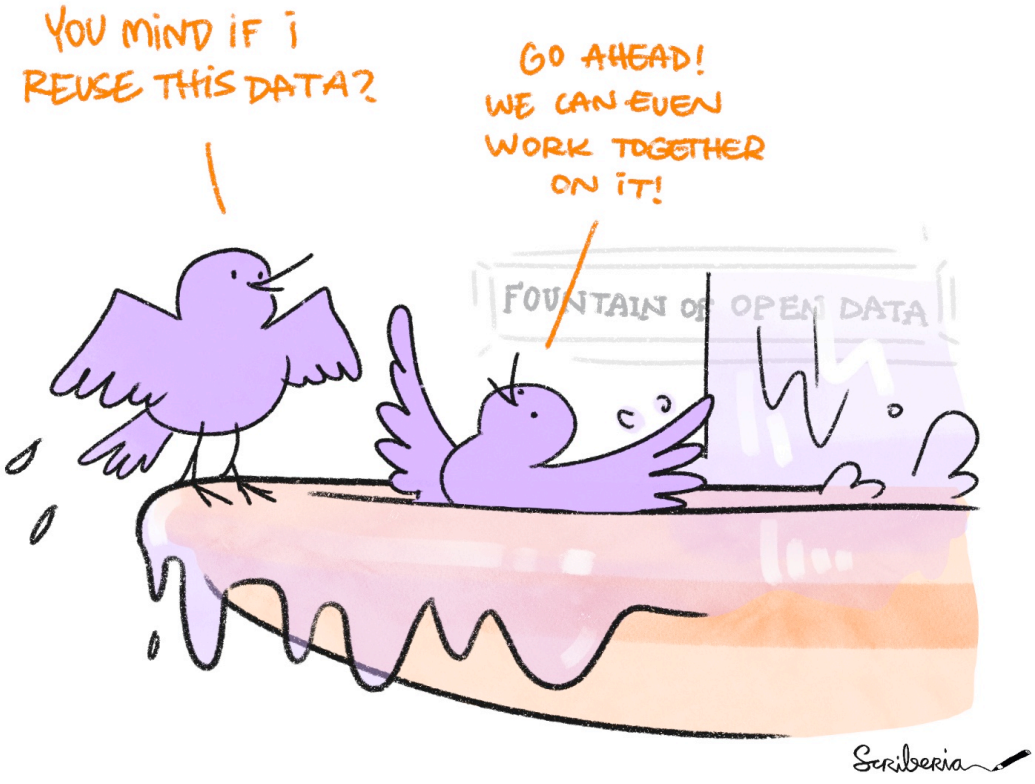


Figure 29: source

- To prevent duplicate efforts and **speed up scientific progress**. Large amounts of research funds and careers of researchers can be wasted by only sharing a small part of research in the form of publications.
- To facilitate **collaboration** and increase the impact and quality of scientific research.
- To make results of research openly available as a **public good**, since research is often publicly funded.

FAIR data

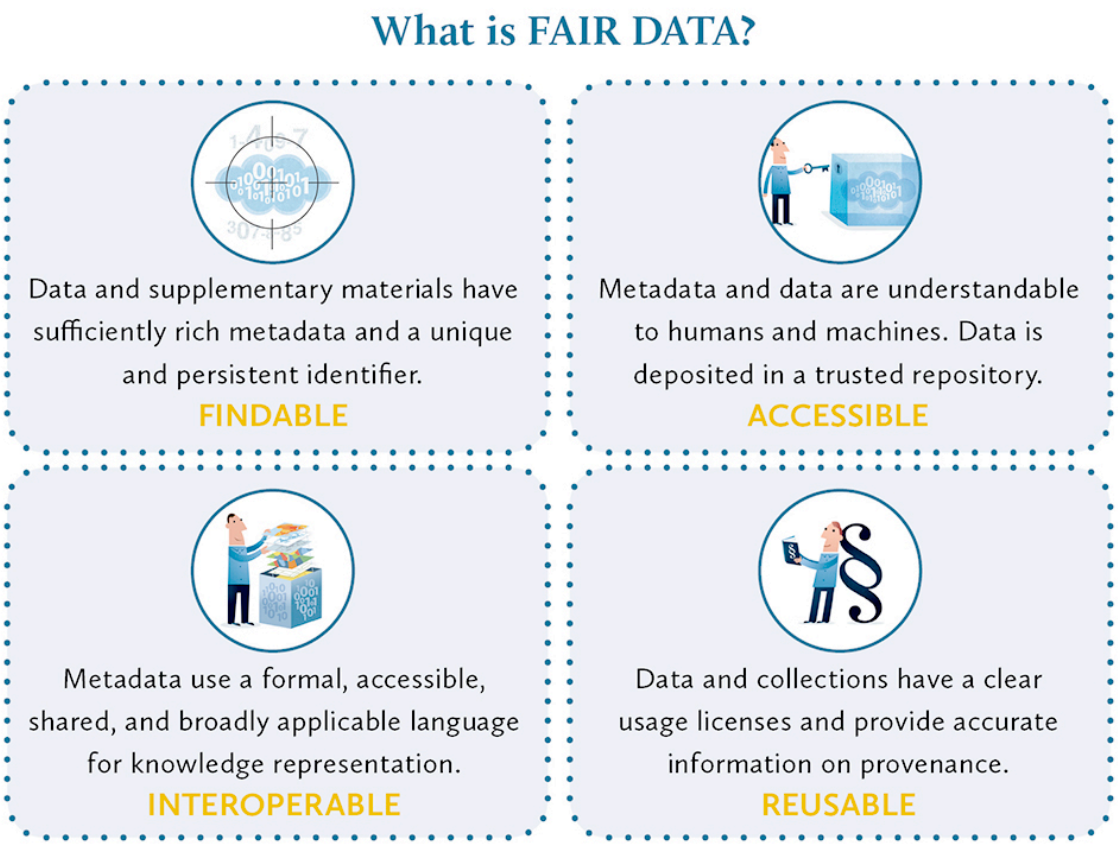


Figure 30: source

How to share your data

Turing Way tutorial

- Step 1: Select **what data** you want to share; eg.:
  - ethical concerns
  - commercial concerns
- Step 2: Choose a **data repository** or other sharing platform
  - overview: re3data, NIH, FAIRsharing
  - examples: Zenodo, Dryad
- Step 3: Choose a **licence** and link to your paper and code; e.g.:
  - Creative Commons
  - Open Data Commons
- Step 4: **Upload** your data and documentation
  - good file organisation
  - appropriate file formats (e.g. `csv` > `xlsx`)
- Excel issues
  - formatting
  - thousands separators: `,` vs `.`
  - date conversion
  - formulas

Sharing social media data

Obstacles to data-sharing

- Reason 1: Preparing data for sharing is **resource-intensive**
- Reason 2: Not enough **credit** for data sharing
- Reason 3: Lack of **confidence** and **knowledge**
- Reason 4: Data protection **laws**
- Reason 5: Platform **terms of service**
- Reason 6: **Copyright**
- Reason 7: Informed **consent**

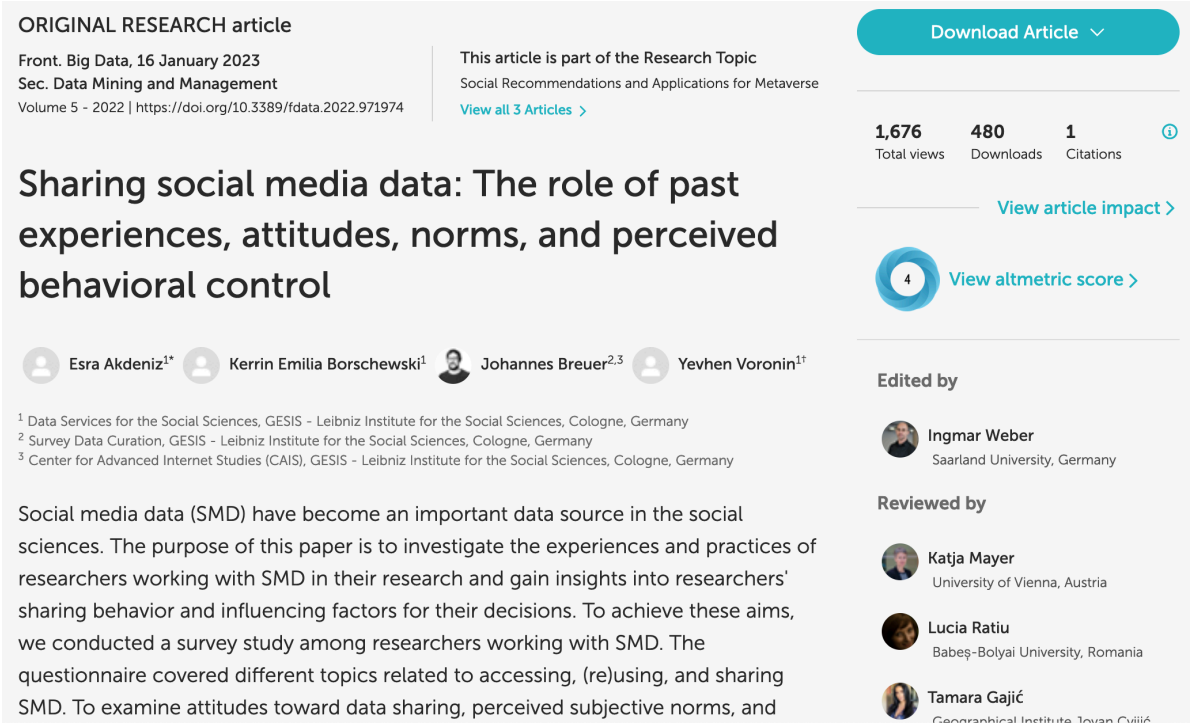


Figure 31: source

- Reason 8: **Ethical** challenges
- Reason 9: Lack of common **standards**

source

- additional reason: sharing not **considered** (necessary)
- **ethical** and **privacy**: paper on incest on Twitter

The case of Twitter

- stage 1: access costly & legal grey area for scraping
- stage 2: Research API
- stage 3: Elon Musik → X → ...

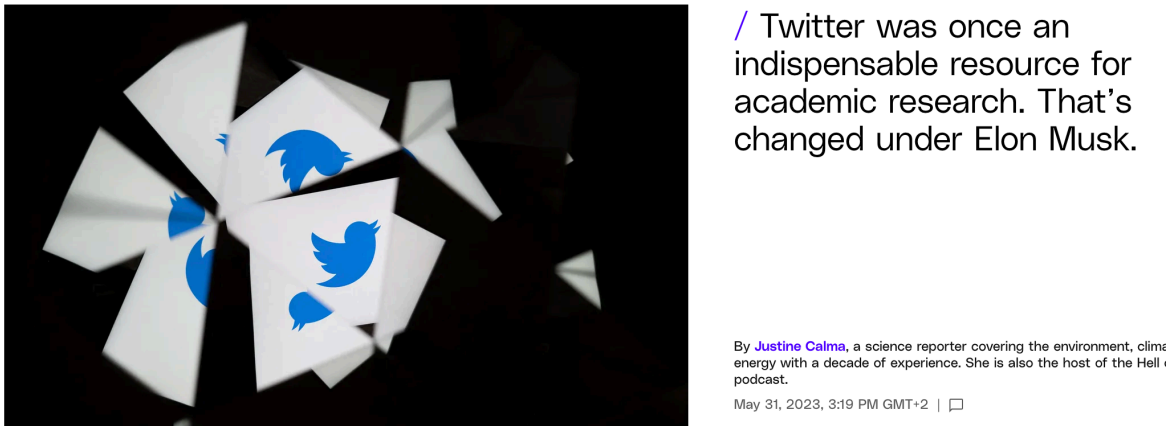


Figure 32: source

Data and methods

get active: see https://tinyurl.com/opnsci

Publishing

Open access

The Turing Way tutorial on open access

Routes to open access publishing

Preregistration

What is preregistration?

When you preregister your research, you're simply specifying your **research plan in advance** of your study and submitting it to a registry.

Preregistration separates **hypothesis-generating** (exploratory) from **hypothesis-testing** (confirmatory) research.

- Both are important.
- But the same data cannot be used to generate and test a hypothesis, which can happen unintentionally and reduce the credibility of your results.
- Addressing this problem through planning improves the quality and transparency of your research.
- This helps you clearly report your study and helps others who may wish to build on it.

Open Science Center tutorial

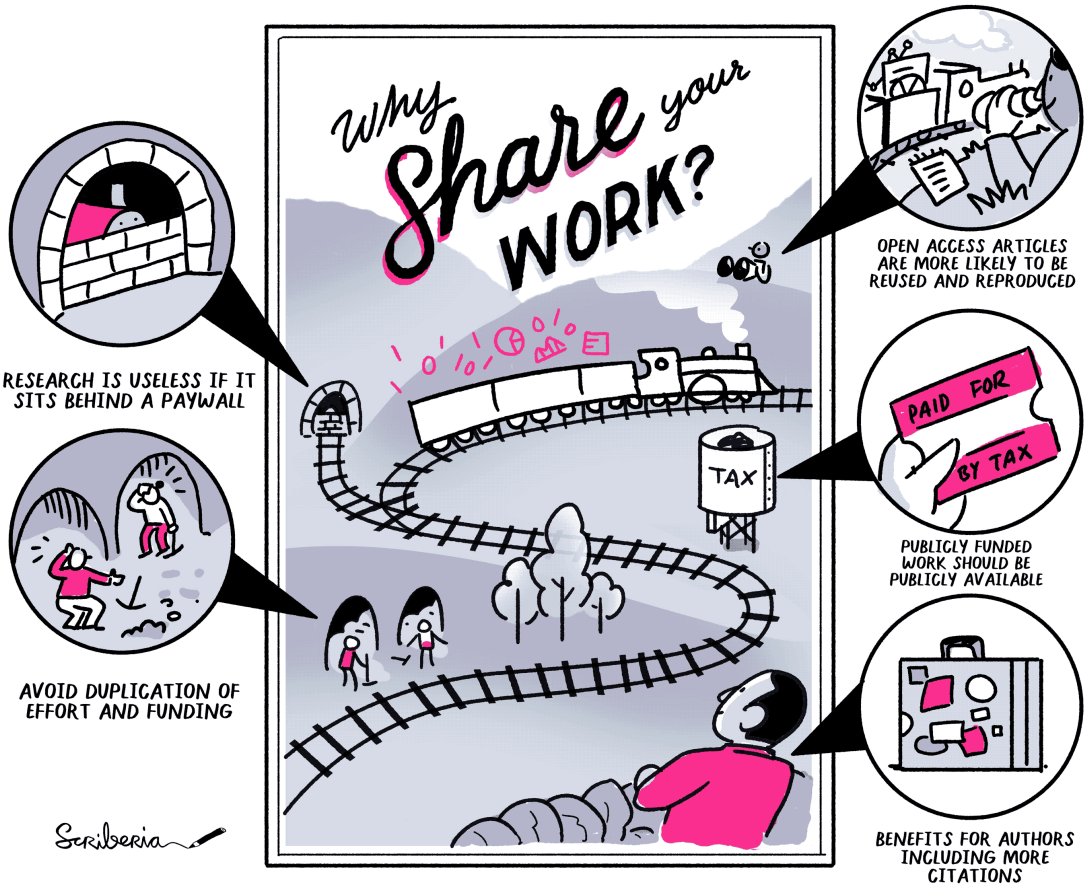


Figure 33: source

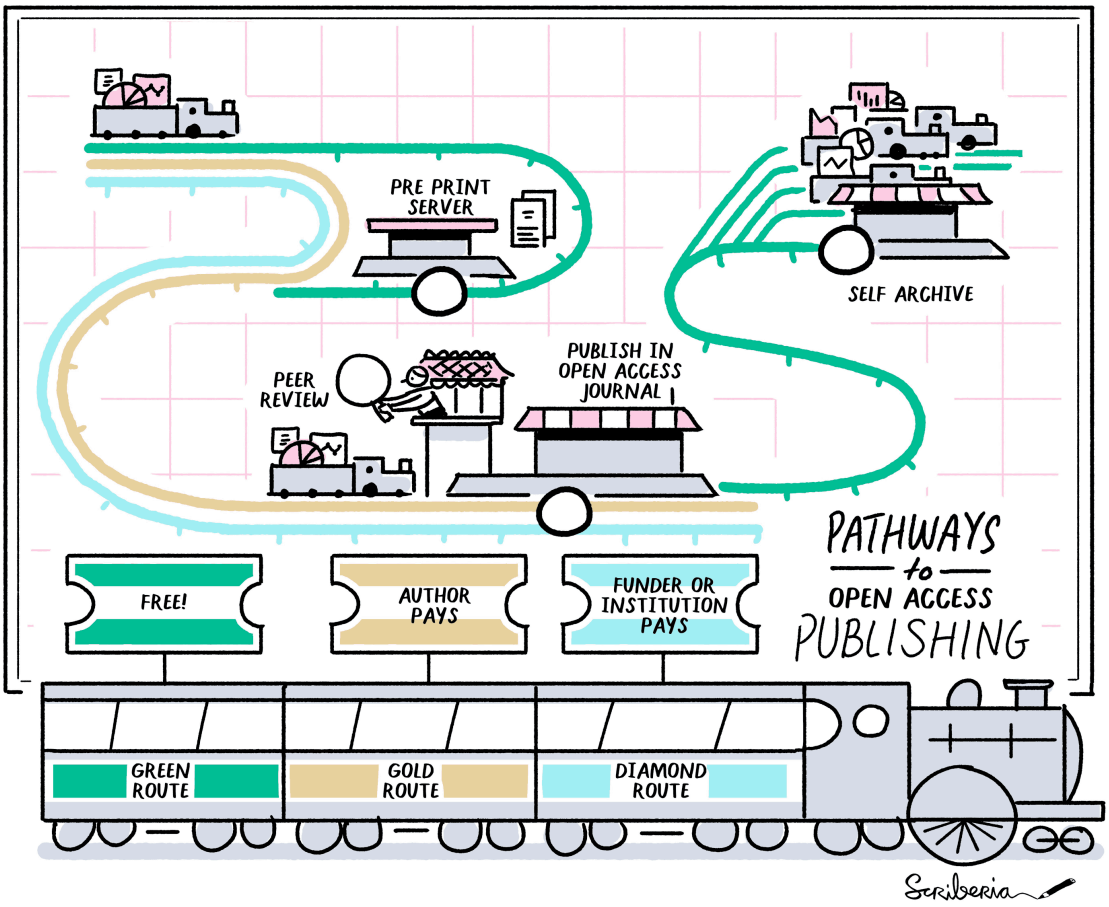


Figure 34: source

The preregistration process

Figure 35: source

Avoiding pitfalls through preregistration

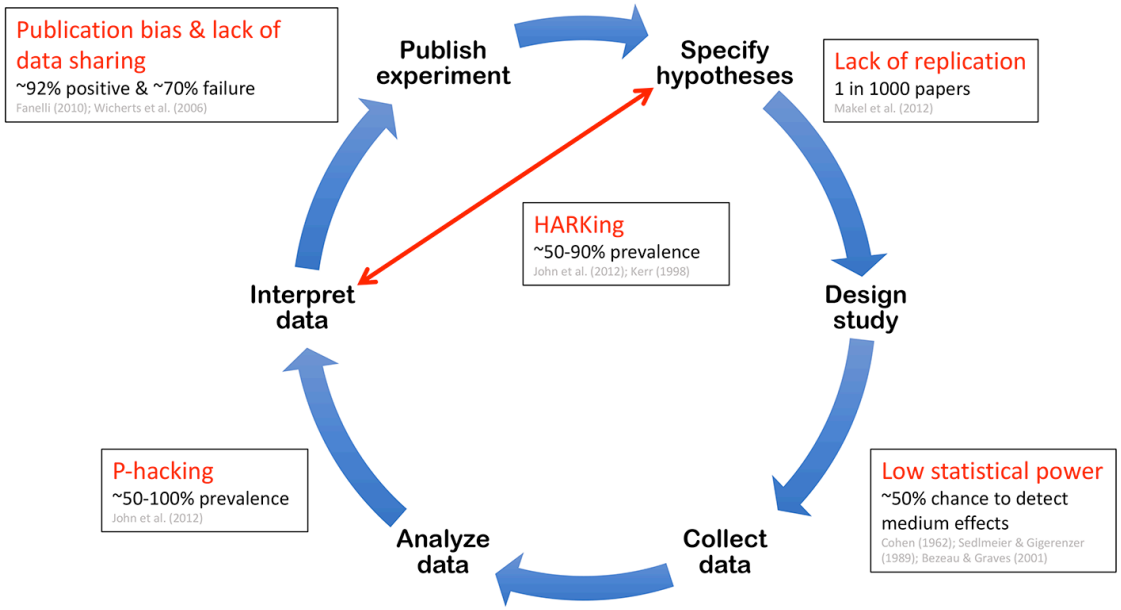


Figure 36: source

HARKing: hypothesizing after results are known

Outlets

ArXiV preprints
Zenodo all kinds including data, code, preprints, etc.
GitHub and GitLab code, software
Open Science Framework all kinds including data, code, preprints, preregistration, etc.
Software Heritage archival of code (long-term)
Papers with Code code and data for and with papers, mostly Machine Learning
...

Authoring

Plain text authoring

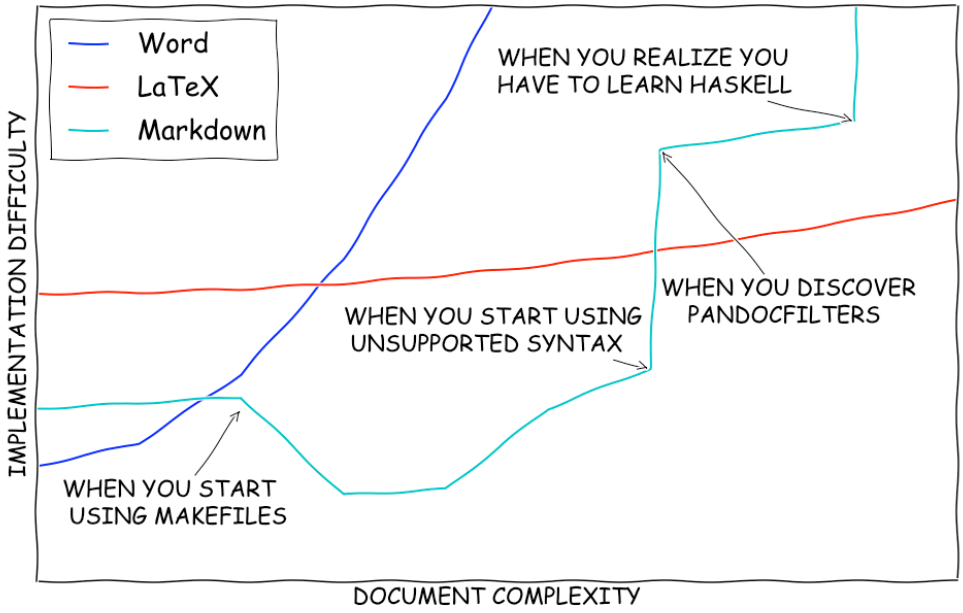


Figure 37: source

Markdown syntax

Text Formatting

Markdown Syntax	Output
`*italics*, **bold**, ***bold italics***`	*italics*, **bold**, ***bold italics***
`superscript^2^ / subscript~2~`	superscript^2 / subscript$_2$
`~~strikethrough~~`	~~strikethrough~~
`` `verbatim code` ``	`verbatim code`

Quarto: Markdown basics

Benefits of Markdown

- Markdown provides **semantic meaning** for content in a relatively simple way
- You can write rich formatted content extremely **quickly** (compared to writing directly in HTML tags)
- You can **read** Markdown easily in plain text before rendered by HTML
- It doesn't interrupt your **workflow** with the need to click buttons
- It's **platform-agnostic** so your content is not tied to the format of your editor

---

Quarto

https://quarto.org/

Welcome to Quarto®

An open-source scientific and technical publishing system

- Author using Jupyter notebooks or with plain text markdown in your favorite editor.
- Create dynamic content with Python, R, Julia, and Observable.
- Publish reproducible, production quality articles, presentations, websites, blogs, and books in HTML, PDF, MS Word, ePub, and more.
- Share knowledge and insights organization-wide by publishing to Posit Connect, Confluence, or other publishing systems.
- Write using Pandoc markdown, including equations, citations, crossrefs, figure panels, callouts, advanced layout, and more.

Multi-purpose publishing

Dynamic Documents

Generate dynamic output using Python, R, Julia, and Observable. Create reproducible documents that can be regenerated when underlying assumptions or data change.

Learn more »

Beautiful Publications

Publish high-quality articles, reports, presentations, websites, and books in HTML, PDF, MS Word, ePub, and more. Use a single source document to target multiple formats.

Learn more »

Scientific Markdown

Pandoc markdown has excellent support for LaTeX equations and citations. Quarto adds extensions for cross-references, figure panels, callouts, advanced page layout, and more.

Learn more »

Authoring Tools

Use your favorite tools including VS Code, RStudio, Jupyter Lab, or any text editor. Use the Quarto visual markdown editor for long-form documents.

Learn more »

Interactivity

Engage readers by adding interactive data exploration to your documents using Jupyter Widgets, htmlwidgets for R, Observable JS, and Shiny.

Learn more »

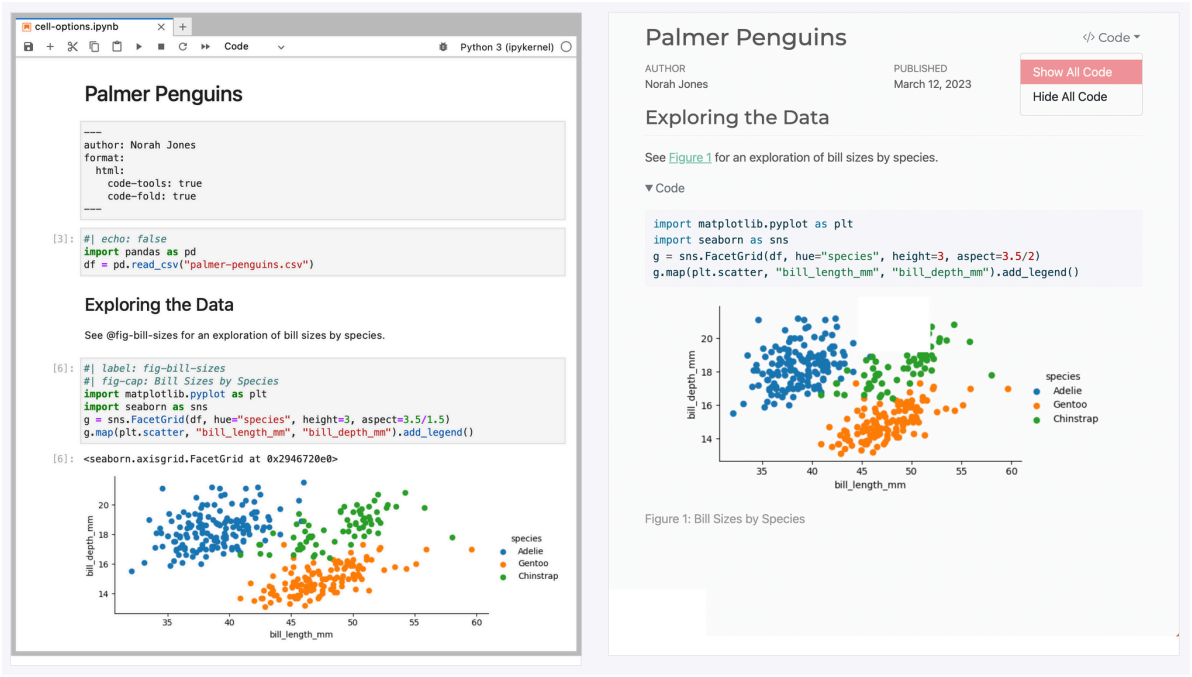
Websites and Books

Publish collections of documents as a blog or full website. Create books and manuscripts in both print formats (PDF and MS Word) and online formats (HTML and ePub).

Learn more »

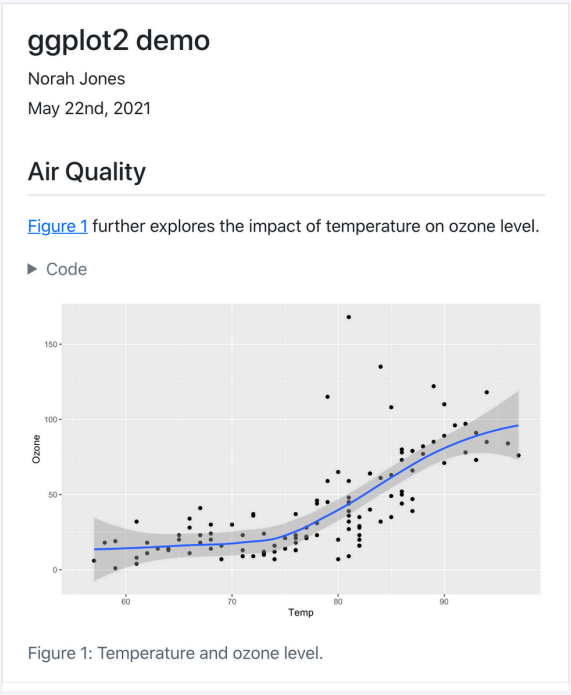
Code

Python



R

````
---
title: "ggplot2 demo"
author: "Norah Jones"
date: "5/22/2021"
format:
  html:
    fig-width: 8
    fig-height: 4
    code-fold: true
---

## Air Quality

@fig-airquality further explores the impact of
temperature on ozone level.

```{r}
#| label: fig-airquality
#| fig-cap: "Temperature and ozone level."
#| warning: false

library(ggplot2)

ggplot(airquality, aes(Temp, Ozone)) +
  geom_point() +
  geom_smooth(method = "loess"
)
```
````



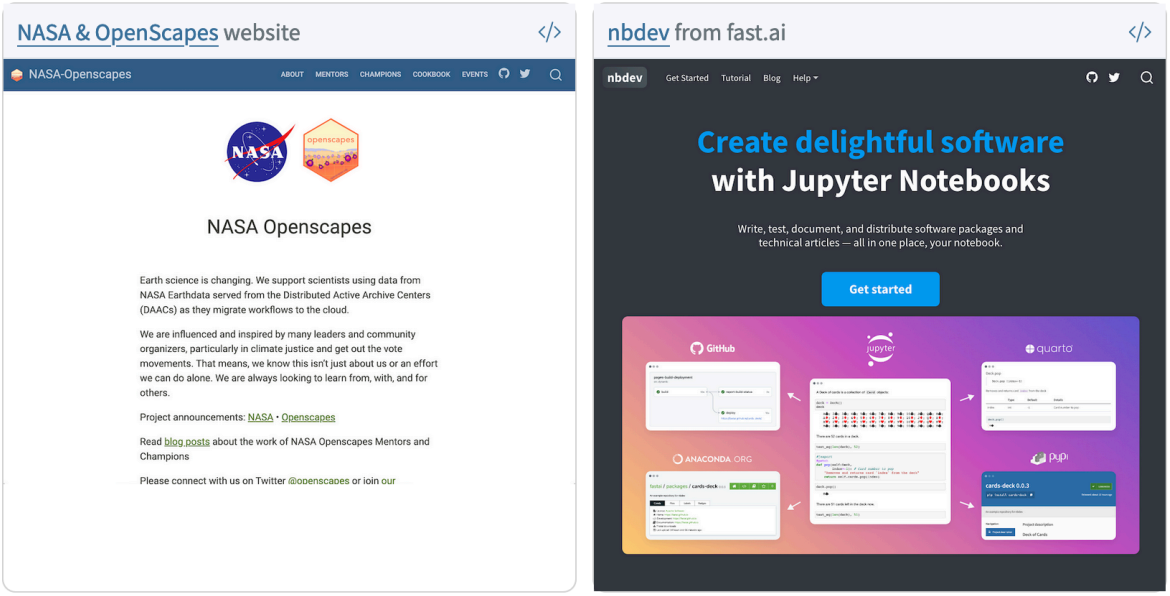
Multi-format publishing

see the Quarto gallery

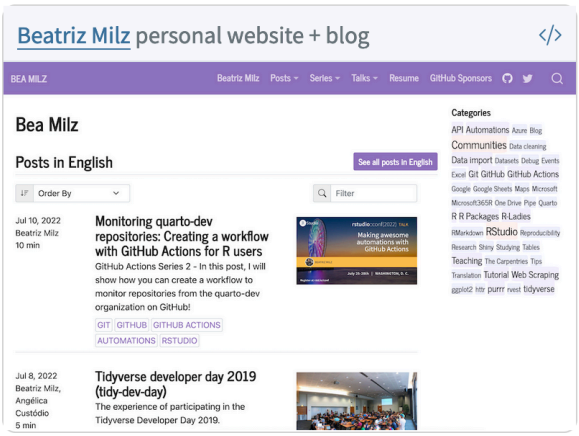
Articles

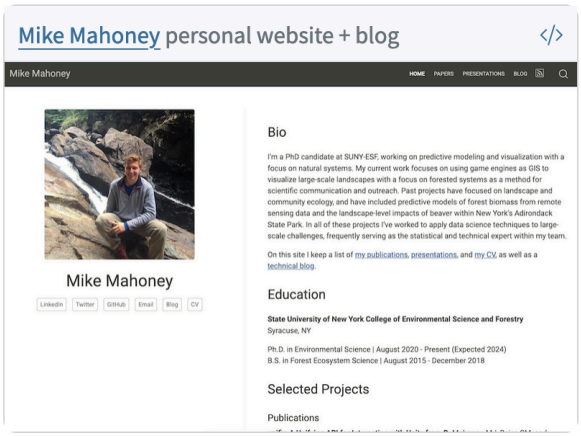
Presentations

Websites



Personal websites and blogs





Demo

Demo repository: https://github.com/wuqui/opensciwsdemo

CAIS workshop demo

AUTHOR
qw

On this page
- Heading 1
  - Code
  - Lists
- Heading 2

Heading 1

Code

```
for i in range(23):
    print('servus')
```

Lists

bullets

- one
- two
- three

Authoring a Quarto document

You have until 16:00 h to,

based on Quarto's excellent getting started guide,

- install Quarto
- make an example text document using Quarto Markdown syntax (optional: clone my demo repo)
- `render` the document to multiple output formats (`pdf`, `html`, `docx`, etc.)

further steps

- publishing your document(s) as a website on GitHub pages (see tutorial)
- refactoring existing document (e.g. paper, website) to Quarto Markdown

---

Recap, open issues, feedback

1. version control
2. project structure
3. code
4. data and methods
5. authoring
6. publishing