

Sharing code

Introduction to Data Management Practices course

NBIS DM Team

data-management@scilifelab.se

<https://doi.org/10.17044/scilifelab.c.6820587>



Learning objectives

- Be able to describe some good code management practices, especially related to sharing code
- Be able to start implementing some of these good practices in your organisation (or in your own work)

You should be able to follow and learn from this talk even if you have never written a line of code!

software – **set of computer programs** and associated documentation and data. This is in contrast to hardware, from which the system is built and which actually performs the work (from Wikipedia).

computer program – sequence or **set of instructions** in a programming language for a **computer to execute** (from Wikipedia).

source code (or **code**) – **set of instructions, or a system of rules, written in a particular programming language** (from Wikipedia).

sharing code / publishing code – making code publicly available.

scientific computational workflow – a multi-step process to coordinate multiple tasks and their data dependencies (from [Goble et al., *Data Intelligence* 2020; 2: 108–121](#))

- **Transparency** – “Look, this is what I did!”
- **Reproducibility** – “Try it yourself and see if you’ll get the same results!”
- **Reusability** – “Use it for your own needs!”
- For **describing research methods** in more detail – “See the software’s manual for a more detailed explanation of ...”
- To get more **citations**
- To get **feedback**
- Other?

Which reason is most important when it comes to code that is:

- used in science in general?
- produced within your organisation?
- produced by you?

Common excuses for not sharing code:

- *“It is not common practice.”*
- *“People will pick holes and demand support and bug fixes.”*
- *“The code is valuable intellectual property that belongs to my institution.”*
- *“It is too much work to polish the code.”*

WORLD VIEW

A personal take on events



Publish your computer code: it is good enough

Freely provided working code — whatever its quality — improves programming and enables others to engage with your research, says Nick Barnes.

I am a professional software engineer and I want to share a trade secret with scientists: most professional computer software isn't very good. The code inside your laptop, television, phone or car is often badly documented, inconsistent and poorly tested.

Why does this matter to science? Because to turn raw data into published research papers often requires a little programming, which

them and now intends to replace its original software with ours.

So, openness improved both the code used by the scientists and the ability of the public to engage with their work. This is to be expected. Other scientific methods improve through peer review. The open-source movement has led to rapid improvements within the software industry. But science source code, not exposed to scrutiny, cannot

Nature 467, 753 (2010)

<https://doi.org/10.1038/467753a>

Five recommendations for FAIR software (<https://fair-software.nl>)

1. Use a public **repository** with version control (e.g. GitHub or BitBucket)
2. Add a **license** (see e.g. <https://choosealicense.com/>)
3. Register your code in a community **registry**
4. Enable **citation** of the software (e.g. via Zenodo or FigShare)
5. Use a software quality **checklist**

Read more about each recommendation: why it is important and where to get links to additional resources: <https://fair-software.nl>

Four simple recommendations to encourage best practices in research software (<https://doi.org/10.12688/f1000research.11407.1>)

1. Make source code **publicly accessible** from day one
2. Make software easy to discover by providing software metadata via a popular **community registry**
3. Adopt a **licence** and comply with the licence of third-party dependencies
4. Define clear and **transparent contribution, governance and communication processes**

Some more good practices:

- Create a README file (see e.g. <https://www.makeareadme.com>)
- Identify all contributors and acknowledge them (e.g. in README)
- Decide on a version naming scheme (see e.g. <https://semver.org>)
- Create a release (using the chosen naming scheme)
- Create a changelog for describing changes (see e.g. <https://keepachangelog.com>)
- Create a [software management plan](#)



A software management plan should **minimally** include:

- What is expected to be produced (incl. documentation)?
- Who is responsible for releasing the software?
- What revision control process to be used?
- What license(s) will be used?



Adapted from <https://www.software.ac.uk/resources/guides/software-management-plans>



A software management plan **could also**:

- identify the software development model to be used
- identify the external software that will be brought into the project, and the associated licences
- what method will be used to accept each component being produced
- dependencies between developed components and with external dependencies
- major risks that might impact the delivery

Adapted from <https://www.software.ac.uk/resources/guides/software-management-plans>

- Scientific computational workflows (written in e.g. Snakemake or NextFlow) may be shared just like any kind of code. However, it is often better to follow guidelines that are specifically for sharing workflows.
- The organisations' behind the workflow management systems typically maintain their own documentation for how to share/publish workflows.
- You may publish your workflow in generic repositories like [Zenodo](#) or Figshare (e.g. [SciLifeLab Data Repository](#)) but WorkflowHub (<https://workflowhub.eu>) is probably a better place.



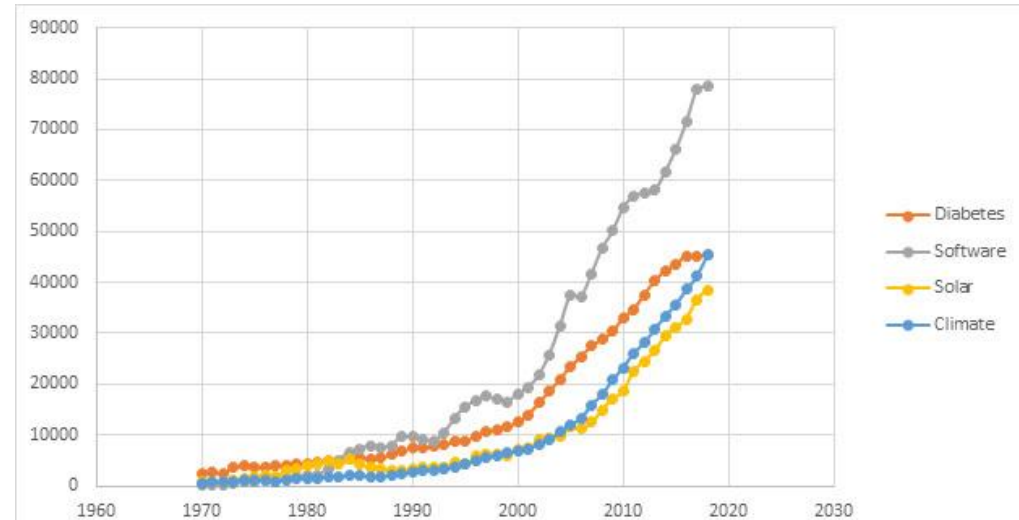
You can also write a software article to make your software more visible.

Software publications are becoming more and more common across disciplines.

The diagram compares the number of published software articles with number of published articles in three scientific fields (source Elsevier/Scopus).

From:

<https://www.elsevier.com/connect/4-reasons-to-publish-software-articles-even-if-youre-not-a-computer-scientist>



- Identify code and software within the organization
- Evaluate the sustainability of the organisation's software (e.g. using Software Sustainability Institute's [online evaluation tool](#))
- Create software management plans where needed
- Create a software and code sustainability plan for the organization
- Create organizational policies and best practices
 - Where to store and maintain code
 - How is code documented?
 - How should releases be named?
 - What code should be published?
 - Where should code be published?