Open Code and Software: a Primer from UKRN

Authors (Z-A): Andy Turner, Marta Topor, Andrew Stewart, Nicholas Owen, Adam R Kenny, Alex Jones, David Ellis.

What is open code/software?

Open Source Software (OSS) is software for which the source code is **openly licensed** and **available for scrutiny, adaption and reuse**. This resource aims to help in the process of choosing, adopting, using, developing and sharing OSS for reproducible research. It also considers open code that is not software in itself but is code that is again openly licensed and available for scrutiny, adaption and reuse and that helps others reproduce research results, for instance, workflows which can be scripts that enact various software to process data in a particular order, experiments described using recipes, or scripts run within a particular software environment to automate some data processing in that environment. This document aims to encourage researchers to use and develop OSS and open code and provides some guidance in doing this.

Why should I make my code/software open?

Making code open and using open code **encourages collaboration** between code users and developers, helping them form a **community** enabling others to understand how and where they can contribute in a more **open development process**. The collaboration may result in **enhanced functionality** of software and **more efficient** code that can be used in more **diverse contexts** for larger and more detailed studies.

Making code openly available **helps others follow** the steps taken to achieve a result and makes it **easier** for them to try it themselves. It also **facilitates collaboration** and interaction between researchers who may want to adapt, extend and enhance the code. In addition, developing and using open code enhances the **sustainability** of research workflows on a larger scale and supports the **reproducibility** of results.

Computational workflows are a set of steps that process inputs into outputs. Each step might involve some configuration and involve different software. The original workflow is likely to have evolved via some experimentation with different parameters and software settings, but ultimately these configuration details can be stored to make it easy to process input data through the entire workflow to achieve the same output data.

Software that is not OSS can be problematic from both a scientific and reproducibility perspective for a number of reasons:

1. Without the source code, it is **hard to fully test** and be sure what the software is and isn't doing. This can be problematic for security reasons as well as for appreciating exactly how the use of pseudo-random sequences, different data types and rounding in calculations may affect results.
2. It may be **harder to overcome any problems**/errors encountered when using the software. You may be dependent on those developing the software to resolve issues and they may not have the resources or inclination to help. In some circumstances it might be that the software source code can be obtained and you can help develop a fix, but arranging this is unlikely to be straightforward.

3. Proprietary software licences can be **prohibitively expensive** and reduce the ability of others to engage in research and reproduce results.
4. Proprietary software licences and licence managers can make it **difficult to install and reuse** software in the future.

How do I make my code/software open?

Supposing you have developed some code that automates or makes it easier for others to recreate a result then do the following and make this code open:

1. Choose a name for the code.
2. Choose an open source licence for it.
3. Decide how to version it.
4. Decide where, when and how to release it.
5. Decide if you are going to support others to use it – and if so – how you want to communicate with them.
6. Produce some descriptive metadata about it and put this in a README file.
7. Push a versioned release.

For software, if you plan to support a community of users then it is also a good idea to develop and share a plan for developing the code over time. Such a **roadmap** typically details how the software development will be managed and what the current plans and timescales are. Opening things up a step further involves making the software development process more open and detailing how others can contribute and get involved in sustaining the development effort.

Licensing code is important and there can be much to consider in choosing or adapting a licence. Essentially, a licence details: permitted uses of the code; and, how others may redistribute it. The licence is a legal document which typically also limits liabilities in case something goes wrong when someone else uses your code.

Some licences are more open and liberal than others. The following may help you choose a licence: https://choosealicense.com/.

Knowing what **version** of code was used to create a result can be important for reproducibility. Code can be versioned with a single number (e.g., Version 1) and possibly a date. For software, see: https://semver.org/ for details of how to do versioning.

Typically a **README** file contains a short description of what the code is, perhaps provides details of who has written/contributed to it, and perhaps indicates who has resourced the development of it. It may also link to further information about the code containing some of these details. Some good examples of README files can be found via the following website: https://www.makeareadme.com/.

Table 1 (below) provides details of a number of contemporary popular systems for developing open code, many of them geared towards developing OSS and creating a community of users. Here are some additional general resources that may also help:

- The Carpentries: a global community supporting those interested in data carpentry, software carpentry, and library carpentry providing training tools and workshops to help individuals develop data and computational skills.
- The Turing Way: a handbook to support students, their supervisors, funders and journal editors in ensuring that reproducible data science is 'too easy not to do'.

- [Eliademy Open Science MOOC Module 5: Open Research Software and Open Source](#).
- [The Software Sustainability Institute Website](#).
- [Stack Exchange](#): a network of question-and-answer websites on topics in diverse fields, each site covering a specific topic, where questions, answers, and users are subject to a reputation award process.
- [Stack Overflow](#): a question-and-answer site for professional and enthusiast programmers.
- [Mozilla Science Lab Working Open Workshop Resources](#): this may help with understanding and transitioning to more open ways of working.

Table 1. Systems for developing, sharing and using open code.

System	Details	Guides for getting started
GitHub	These systems are used to upload documentation, source code and software in any language or format. Essentially, they host online repositories that can be synced to your own computer. They provide a search tool for exploration of other projects as well as helpful discussion forums.	https://guides.github.com/activities/hello-world/
GitLab		https://docs.gitlab.com/ee/gitlab-basics/
SourceForge		https://www.software.ac.uk/resources/guides/how-create-and-manage-sourceforge-projects
Bitbucket		https://www.atlassian.com/git/tutorials/learn-git-with-bitbucket-cloud
Open Science Framework (OSF)	The OSF facilitates code sharing along with other project components that can be manually uploaded and updated.	https://cos.io/our-services/training-services/cos-training-tutorials/
Binder	Binder allows you to reproduce results stored in repositories such as GitHub, GitLab, and Zenodo.	https://mybinder.readthedocs.io/en/latest/