8

Research Data Support: Quick Guide 8

# ‘FAIR’ Software


In 2016, the ‘FAIR Guiding Principles for scientific data management and stewardship’ were published in Scientific Data. These principles sought to improve the **Findability, Accessibility, Interoperability, and Reusability** of research data assets. The distinction between data and code is not always clear-cut, so it was only a matter of time before the FAIR principles were applied to research software too. Code is a vital link in the Open Research chain, sitting between raw datasets and the published conclusions that are drawn from them. Without being able to find, access, understand and re-run software, much research becomes irreproducible and the conclusions become weaker. This guide aims to serve as a step-by-step resource for researchers at the University who wish to make their research software as discoverable and reusable as possible.

## Use a publicly accessible repository with version control

### Why this is important

#### Why public?

Developing scientific software in publicly accessible repositories enables the early involvement of users, helps build collaborations, contributes to the reproducibility of results generated by the software, facilitates software reusability, and contributes to improving software quality. Taken together, this ensures that your software has the best chance of being used by as many people as possible while at the same time promoting transparency in research.

#### Why version control?

Using a version control system allows you easily to track changes in your software: your own changes as well as those made by collaborators. There are many flavours of version control systems, ranging from older systems such as CVS and Subversion to more modern ones such as Git, Mercurial, and Bazaar. By using a Git version control system (such as GitHub, GitLab or Bitbucket) you’ll have backups of every version of the software you or your contributors ever made. Additionally, these platforms offer collaboration tools such as issue trackers and project management functions, and you’ll be able to use third-party services such as code quality checkers, correctness checkers, and many more.

### Version control at Edinburgh

GitLab is a Git repository manager/hosting service. When files stored in a GitLab repository are updated, older versions are retained, making it possible to roll-back to them if required. The University provides the GitLab service at no charge, but if you require a repository of >10GB then a charge may be made.

Find out more and access GitLab via:

**www.ed.ac.uk/information-services/research-support/research-data-service/during/versioning**

## Add a licence

### Why this is important

Any creative work (including software) is automatically protected by copyright. Even when the software is available via code sharing platforms such as GitHub, no one can use it unless they are explicitly granted permission. This is done by adding a software licence, which defines the set of rules and conditions for people who want to use the software. Finally, be aware that you, as the developer of a given piece of software, may not be a copyright owner of the code you write. Usually the copyright holder of a work is the employer (or hiring party) and not the author of the work.

### Help me choose

We recommend you stick to one of the more popular licences. Because these are typically written by lawyers, the licence text is precise in expressing its terms. While that sometimes makes them difficult for the lay person to understand, widespread use of the more popular licences means that there is a larger number of people who understand how the letter of the law should be interpreted.

Some of our favourite licences are the Apache-2.0 and MIT licences. These permissive licences have very few restrictions, allowing others to easily reuse your work.
We recommend you use **choosealicense.com** to find out which licence is best for your purposes. If you just want to check what is and what is not allowed under a given licence, visit **tldrlegal.com**

**How to add a licence to your GitHub repository**

## Register your code in a community registry

### Why this is important

For others to make use of your work, they need to be able to find it first. Community registries are like the yellow pages for software – registering your software makes it easier for others to find it, particularly through the use of search engines such as Google. Community registries typically employ metadata to describe each software package. With metadata, search engines are able to get some idea of what the software is about, what problem it addresses, and what domain it is suited for. In turn, this helps improve the ranking of the software in the search results – better metadata means better ranking.

### Help me choose

Community registries come in many flavours, and choosing the one that is best suited for your needs can be tricky. Here are some things to think about:

- How much traffic does the community registry get?
- Does the community registry target the audience you are trying to reach?
- What metadata does the community registry offer? This is sometimes described in the documentation of the registry, but you can also see for yourself by installing a tool like the **OpenLink Structured Data Sniffer**

Finally, why not ask colleagues which registries they would use if they were looking for software like yours?

**Browse the Netherlands eScience Center's list of research software registries on GitHub**

## Enable citation of the software

### Why this is important

Citation helps software developers get recognition for their work. Citation is an integral part of scientific accountability and reproducibility, but citing software accurately is inherently more difficult than citing a paper. To the outsider especially, even seemingly trivial things such as identifying who should be recognized as an author can be difficult. It is therefore convenient when software developers themselves provide the information necessary to enable citation, in addition to managing version control.

### Help me choose

The **CodeMeta** standard and **Citation File Format** are specifically designed to enable citation of software. Each requires you to write a plain text file with citation metadata, which is then distributed along with your software.

**Initialise your CITATION.cff files**

When archiving copies of your software, look for services that store their own copy of a snapshot of your software, so that whatever **persistent identifier** you get (DOI, URN, ARK, etc) points to a specific version of the software, and will continue to resolve to that exact version for the foreseeable future. Ideally, storing snapshots of your code should be as easy as possible: either at the push of a button, or automatically, for example **each time you make a new release of your software**.

Some archiving services that meet these requirements are: **Zenodo**; **FigShare**; and the **Software Heritage Archive**.

# Use a software quality checklist

## Why this is important

Checklists help you write good quality software. What exactly constitutes 'good quality' depends on the specific application of the software, but typically covers things like documenting the source code, using continuous testing, and following standardised code patterns.

## Help me choose

There are many checklists available. The most useful checklists are those that:

1. Allow for a granular evaluation of a software package, as opposed to just pass or fail
2. Explain the rationale behind each item in the checklist
3. Explain how to get started with implementing each item in the checklist

We recommend that you include the checklist as part of the README file, for example as a badge or by including the checklist as a **Markdown** table. The point is not necessarily to demonstrate perfect compliance, but rather to be transparent about the state of the code whilst providing the necessary guidance on any aspects that could be improved.

One such checklist that meets these criteria is the **Interactive Badge Program developed by the Core Infrastructure Initiative**, but there are many others to choose from.

Here is a list of some candidates:

- Deutsches Zentrum für Luft- und Raumfahrt: **Class 1**, **Class 2**, **Class 3** (MarkDown)
- The Software Sustainability Institute's software evaluation checklist (**Google form**)
- CLARIAH checklist (**PDF pages 38-42**)
- EURISE (**MarkDown**)

## Contacts and resources

- The FAIR Principles: **www.go-fair.org/fair-principles/**
- FAIR-Software project: **https://fair-software.eu/**
- Research Data Service website: **www.ed.ac.uk/is/research-data-service**
- Snapshots of code can be submitted to Edinburgh DataShare, and receive DOIs and Pure records: **https://datashare.is.ed.ac.uk/**
- Contact Research Data Support by email: **data-support@ed.ac.uk** or contact the IS Helpline: **is.helpline@ed.ac.uk**

**If you require this document in an alternative format, such as large print or a coloured background, please contact [insert: name and contact details]**