SHARING RESEARCH SOFTWARE

Research software includes source code, algorithms, scripts, computational workflows, and executable files developed in the course of research or specifically for a research purpose. It can range from small scripts to complex applications and is considered a research output in its own right. Yet, research software often remains inaccessible and even undocumented.

This guide is designed for open science trainers to help them provide practical guidance on why and how to share research software, along with actionable steps to support best practices in software sharing.

WHY SHARE RESEARCH SOFTWARE

- Publicly available code lets others inspect methods, re-run analyses, and validate claims presented in research publications.
- Sharing software code prevents duplication of effort, facilitates collaboration across disciplines and accelerates progress.
- Software developers can benefit from early feedback on issues, bugs, pull requests and potential improvements.
- Various communities can continue developing software, making it more sustainable and integrating it into new contexts.
- Sharing aligns with funder mandates, institutional policies, and community expectations for openness.
- Software deposited in a suitable repository can be cited.
- Sharing software can help developers build their reputation and advance their career, as software contributions are increasingly valued in hiring, tenure, and funding decisions.

HOW TO SHARE RESEARCH SOFTWARE

01 Make the code available in a public repository with version control (e.g. GitHub, GitLab, BitBucket, Comprehensive R Archive Network - CRAN, etc.), so that each software update can be released and clearly labeled as a new version.

02 Add a README file including the information about the software development project (software name, authors, contact details), clear instructions for installing the software, running the code and associated tests, and links to related resources.

03 Include tests and example data or notebooks to demonstrate how the software works. This helps others verify that the code runs as intended, understand its functionality, and learn how to apply it in their own research.

04 Provide the licence information in a plain text file to make the terms of use clear, enable reuse, and provide legal protection for contributors and users.

Licensing - The Turing Way

05 Create a CITATION.cff file using the cffinit web application or manually in a code editor. As widely used software repositories like GitHub do not use persistent identifiers, it is convenient to archive software on Zenodo to get a DOI.

Software Citation with CITATION.cff - The Turing Way

GitHub and Software | Zenodo

06 Provide guidance on how others can contribute to code/software development (e.g. by reporting bugs or suggesting new features) in a CONTRIBUTING.md file.

How to Build a CONTRIBUTING.md - Best Practices

07 Set behaviour rules for contributors and users by providing a CODE_OF_CONDUCT.md file. The Contributor Covenant can be used as a ready-made, standardized solution for setting the Code of Conduct but developers can write their own if needed.

README, CITATION.cff, CONTRIBUTING.md and CODE_OF_CONDUCT.md files should be added to the software project's root folder (top-level directory).

TAKE ACTION!

- Advocate for the adoption of institutional policies that require and reward research software sharing.
- Help researchers choose suitable repositories for research software sharing and provide guidance on best practices.
- Facilitate training on software sharing, licensing, and reproducibility.

SELECTED RESOURCES

- Reusable Code - The Turing Way
- ADORE.software Toolkit
- Tierney, Aoife, Carlos Tighe, Clare Dillon, et al. 2024. Framework for Managing University Open Source Software. https://zenodo.org/records/14392733
- Open Code and Software: a Primer from UKRN