Comment

Supplementary information to:

A road map to improve software sharing practices

A Comment published in *Nature* **646**, 284–286 (2025)
https://doi.org/10.1038/ d41586-025-03196-0

Roberto Di Cosmo, Sabrina Granger, Konrad Hinsen, Nicolas Jullien, Daniel Le Berre, Violaine Louvet, Camille Maumet, Clémentine Maurice, Raphaël Monat & Nicolas P. Rougier

This Supplementary information comprises:

1. Supplementary Table S1
2. Supplementary Table S2

Supplementary Information to:
A road map to improve software sharing practices

Authors

Roberto Di Cosmo[1,2], Sabrina Granger[6], Konrad Hinsen[3,4], Nicolas Jullien[5], Daniel Le Berre[7], Violaine Louvet[8], Camille Maumet[9], Clémentine Maurice[10], Raphaël Monat[10] & Nicolas P. Rougier[11,*]

* **Corresponding author:** nicolas.rougier@inria.fr

Affiliations

[1] Inria Paris, Paris, France
[2] Universite Paris Cité, Paris, France
[3] Centre de Biophysique Moléculaire (CNRS), Orléans, France
[4] Synchrotron SOLEIL, Saint Aubin, France
[5] IMT Atlantique, Brest, France
[6] Inria Lyon, Lyon, France
[7] CRIL, Université d'Artois, Lens, France
[8] Laboratoire Jean Kuntzmann, Grenoble, France
[9] Inria, Univ Rennes, CNRS, Inserm, Rennes, France
[10] Univ. Lille, CNRS, Inria, UMR 9189 CRIStAL, Lille, France
[11] Centre Inria de l'université de Bordeaux, Bordeaux, France

(Continues on next page)

Table S1: Create, curate, collaborate

Many researchers and engineers write code for their research projects yet have not received specialized training in software engineering. These steps should be taken, gradually, to ensure software is suitably shared and archived.

	Mandatory	**Recommended**	**Optional**
Make software open source: ensure that everyone can inspect and use the software	Publish code on a public platform Ensure that it is saved in a dedicated archive such as Software Heritage or Zenodo	Choose an open licence* Declare authors and rightholders	Put software under version control
Document: make the code intelligible to others		Use meaningful names for everything Explain how the code works Provide examples and tutorials	For larger projects, provide a reference documentation explaining how to use a given function, in what conditions, what are the arguments, their type and their meaning, etc.
Execute: enable others to run the software		Provide a list of software and hardware dependencies. For example, the operating system on which it can be run and any other software or libraries that need to be installed prior to usage.	Provide a ready-to-run computational environment**, a test suite, and real-life usage examples

Collaborate: interact with a community of users			Have a strategy to effectively deal with any future contributions from collaborators Describe the limits that the developers have decided concerning maintenance, feature addition and support Respond to questions Engage in active community building, such as explaining to fellow researchers and engineers how to contribute

*Without a licence, the default copyright laws apply and no one can reproduce, distribute or create derivative works. This is the reason why an open-source licence is strongly recommended even if there is no legal obligation to do so.

**This follows an initiative started by some computer science communities in 2011, who have put in place an artifact evaluation process aiming at ensuring computational reproducibility of software-based empirical evidence[1]. The Association for Computer Machinery (ACM) has put this practice into policy since 2016.

1. Krishnamurthi, S. Artifact evaluation for software conferences. ACM SIGPLAN Notices 48(4S), 17–21 (2013)

Table S2: Team up for good

Good practices need support and involvement from all actors of the scientific ecosystem.

Research institutions, funders, libraries and publishers have more substantial means at their disposal to support research software than research groups — and so they also have more responsibility.

	Mandatory	**Recommended**	**Optional**
Research institutions	Provide the infrastructure and human resources required for software development Recognize the work software development and maintenance involves; embed it in employee assessments	Consider software as a valuable research output, in the same way as journal articles are, for example in researchers' production/career evaluation	Host software developments on institutional platforms Provide training in software engineering

Funders	Provide durable financial support for software research, development and maintenance, beyond the cycle of research grants. The Chan Zuckerberg Foundation, for example, has made such grants available	Add reproducibility as a criterion for continued funding	Facilitate collaborations through specific funding schemes
Libraries	Organize**,** curate and maintain software metadata and archives Create reference tools addressing the specificities of software (evolution, authorship, ...)	Build catalogs of software to ensure findability, visibility and accessibility	
Publishers	Following the push for open science, mandate free and open-source software for all published research	Archive the version of software associated with publications	Review software