MONASH University

MONASH BUSINESS SCHOOL

Submission to Recognise Open Source Software as a Research Output

The Faculty treats high quality open source software as a research output and awards points accordingly. This is consistent with international trends to recognize and support research software development (SFDORA 2012; Prlić & Procter 2012; Wilson et al 2014; Doerr et al. 2019; Barba et al. 2019; Druskat 2019; Anzt et al. 2020; Research Software Alliance 2023). It is also consistent with recommendations from OECD and UNESCO.

This submission form is to be used when seeking points to be awarded for recognition of high quality open source software as a research output for the purposes of Quantitative Research Performance Standards.

Evaluation Committee

A peer evaluation committee will consist of a rotating group of discipline experts from within and external to the Monash Business School. The Panel Chair will be delegated by the Deputy Dean Research. The committee will obtain at least one review from an anonymous and external reviewer, or at least two reviews if the submission includes no previous peer review.

In its evaluation, the committee will consider the following issues:

- Where a software output implements a method proposed in a paper by the same author, the software should be assessed as to what it contributes beyond the methodology in the paper.
- Software that is maintained and developed over many years and many versions may be granted additional points over time.
- The comments of the external reviewer(s).

The committee may award 0–3 points for each submission, taking account of comparable quality standards for journal articles.

If a committee member has a conflict of interest (e.g., s/he has co-authored the software submission), the remaining committee members will consider the relevant submission.

The committee will provide brief reasons for its decision.

Submissions

Submissions to the committee should include the following:

Your Name	
Department/Centre	

Is there evidence of previous peer-review, such as acceptance on CRAN or via ROpenSci? If so, provide details here or attach with your submission.

Provide a link to the full source code in a version controlled repository such as github
The scope of the developer's contribution. For pre-existing software, include information about the state of the project before the contribution, and the state afterwards, and include links to the relevant code diff/pull request(s).
For pre-existing software, provide evidence of new features (e.g., a new method to do something the software could not do before) OR evidence of improvement of general features (e.g., improved memory management or linear algebra code used throughout the project).
Provide evidence of user-facing and/or developer-facing documentation.
Provide/attach testimonials from users (for added features) OR developers (for low-level contributions/improvements).

Email the completed form and any attachments to buseco-research@monash.edu

Guidelines

Method documentation
If the software implements a previously published method/model/algorithm, please provide citations to the relevant literature. If the software implements previously unpublished methods, they need to be described in the software documentation to the same level that would be required for a journal publication.

Generalizability
The Australian Research Data Commons Ltd (2022) provides a 3-level classification of research software.

1. "Analysis code" is typically the code associated with a research paper that allows the results to be reproduced.
2. "Prototype tools" provide a general implementation of a method/model/algorithm that can be used for a wide range of applications.
3. "Research software infrastructure" provides a general platform that allows others to build prototype tools.

In general, analysis code is unlikely to be awarded research points. We expect that research software submissions that are awarded points are likely to be either prototype tools or contributions to research software infrastructure.

Code documentation
As a minimum, we would expect each user-facing function to be fully documented with arguments explained, and examples provided. Even better would be an additional vignette describing the full workflow for an analysis using the software. Having the code documentation available online is desirable; this can be done using providers such as GitHub pages, readthedocs or Mintlify.

User impact
We are looking for evidence that the software is useful to other researchers, not just to the authors and their collaborators. Such evidence may be in the form of user testimonials, download statistics, or forks/stars on GitHub, for example.

Releases and versioning
It is important that the software is version controlled, and that users of the software can cite a specific version, released on a specific date. One way to do this is to use the GitHub release system. See Making software citable for some further suggestions.

Bug reports and contributions
Provide a mechanism for handling bug reports or software contributions. If using GitHub, this can be done via GitHub issues and GitHub pull requests respectively. Other version control platforms have similar facilities.

Long-term development
Describe any plans to continue to develop the software with additional features.

Governance
Describe how the software is to be managed. Who are the maintainers? For larger projects, the "Minimum Viable Governance" tool from GitHub may be appropriate.

License
Describe how the software is licensed. See https://choosealicense.com/

References

Anzt, H, F Bach, S Druskat, F Löffler, A Loewe, B Y Renard, G Seemann, et al. (2020). "An Environment for Sustainable Research Software in Germany and Beyond: Current State, Open Challenges, and Call for Action." CoRR abs/2005.01469. https://arxiv.org/abs/2005.01469.

Australian Research Data Commons Ltd. (2022). “A National Agenda for Research Software.” https://doi.org/10.5281/zenodo.6378082.

Barba, L. A., J Bazán, J Brown, R V Guimera, M Gymrek, A Hanna, L Heagy, KD Huff, DS Katz, CR Madan, KM Moerman, KE Niemeyer, JL Poulson, P Prins, K Ram, A Rokem, AM Smith, GK Thiruvathukal, KM Thyng, L Uieda, BE Wilson, Y Yehudi (2019). Giving software its due through community-driven review and publication. https://doi.org/10.31219/osf.io/f4vx6

Doerr, A, N Rusk, N Vogt, R Strack, L Tang, and S Arunima. (2019). “Giving Software Its Due.” Nature Methods 16 (207).

Druskat, S. (2019). “How to Make Software Fit in Research Citation Graphs.” In RSEConUK 2019 - 4th Conference of Research Software Engineering.

Prlić, A., & Procter, J. B. (2012). Ten simple rules for the open development of scientific software. PLoS computational biology, 8(12), e1002802.

Research Software Alliance. (2023). Amsterdam Declaration on Funding Research Software Sustainability (1.0). Zenodo. https://doi.org/10.5281/zenodo.8325436

SFDORA. 2012. San Francisco Declaration on Research Assessment. http://sfdora.org.

Wilson, G, D A Aruliah, C T Brown, N P Chue Hong, M Davis, R T Guy, S H D Haddock, K D Huff, I M Mitchell, M D Plumbley, B Waugh, E P White, P Wilson (2014). Best practices for scientific computing. PLoS biology, 12(1), e1001745.