Electronic Communications of the EASST
Volume 86 (1) Year 2026

deRSE26 - Selected Contributions of the 5th Conference for Research Software Engineering in Germany (Part 1)

Edited by Bernd Flemisch, René Caspart, Bernadette Fritzsch, Florian Goth, Jean-Noël Grad, Florian Thiery

The Journal of Open Source Software (JOSS): An open community and platform providing credit to software developers in the time of GenAI

Warrick Ball, Patrick Diehl, Daniel S Katz, Richard Littauer, Kyle E. Niemeyer, Arfon Smith, Charlotte Soneson

DOI: 10.14279/eceasst.v86.1.2818

License: This article is licensed under a CC-BY 4.0 License.

Electronic Communications of the EASST (https://eceasst.org).
Published by **Berlin Universities Publishing**
(https://www.berlin-universities-publishing.de/)

The Journal of Open Source Software (JOSS): An open community and platform providing credit to software developers in the time of GenAI

Warrick H. Ball [1], **Patrick Diehl** [2], **Daniel S. Katz** [3], **Richard Littauer** [4], **Kyle E. Niemeyer** [5], **Arfon M. Smith** [6], **and Charlotte Soneson** [7]

University of Birmingham[1], Los Alamos National Laboratory[2], University of Illinois Urbana-Champaign[3], Te Herenga Waka Victoria University of Wellington and CURIOSS[4], Oregon State University[5], Schmidt Sciences[6], and Friedrich Miescher Institute for Biomedical Research and SIB Swiss Institute of Bioinformatics[7]

Abstract: The Journal of Open Source Software (JOSS) is a community-driven publishing platform for research software, having published over 3,500 articles since its inception in 2016. Built on an open infrastructure hosted on GitHub, JOSS integrates elements of scholarly communication with open-source development practices. Its review process is transparent and iterative, evaluating both the submitted manuscript and the associated software, thereby promoting reproducibility and high-quality software engineering practices.

JOSS provides an important mechanism for recognizing the contributions of Research Software Engineers (RSEs) within the formal scholarly record. By offering a lightweight publication model tailored to research software, it enables authors to receive credit for software development while also fostering community participation through roles such as authors, reviewers, and editors.

As the research software landscape evolves, generative AI (GenAI) introduces both opportunities and challenges for platforms like JOSS. Advances in AI-assisted code generation significantly reduce the time required to develop software, complicating traditional notions of "substantial scholarly effort," a criterion introduced by JOSS to ensure meaningful contributions. While these tools may broaden participation and lower barriers to entry, they also raise critical questions about how to assess originality, effort, and scientific value in submitted work.

This paper presents an overview of JOSS as an open-source publishing platform, examines its role in supporting RSE recognition, and discusses emerging considerations for evaluating research software in the era of generative AI. We outline current thinking within the JOSS community and highlight the need for ongoing dialogue to adapt publication criteria to a rapidly changing technological landscape.

Keywords: generative AI, JOSS, open source software, research software

1 Introduction

The Journal of Open Source Software (JOSS) is a community-driven publishing platform for research software, having published more than 3,500 articles since its inception in 2016. The

primary purpose of a JOSS paper is to enable citation credit to be given to authors of research software, and JOSS also plays an important role in defining and implementing community best practices for research software. Software is increasingly being seen as an output of scientific work; however, citing software is difficult [KBH+19] and software citations do not receive the same prestige as other academic outputs. JOSS exists to facilitate software citations, following the principles of software citation [Dat14, SKN16]. An accepted software paper in JOSS becomes equivalent to an accepted paper in any other journal, in that it can be cited and listed as part of an author's scholarly contributions on their CV, websites, and on databases such as ORCID and Google Scholar. JOSS papers are peer reviewed—both the article and the software—and JOSS issues Crossref DOIs for published articles, like other journals.

In this paper, we discuss *(1)* the JOSS publishing process as a whole, *(2)* JOSS reviews, *(3)* the impact of generative AI, and *(4)* citing JOSS articles.

2 The JOSS Publishing process

JOSS[1] [SNK+18, DSK+25] makes it as easy as possible for authors to write software papers, in particular by using a publishing and review pipeline using GitHub and automated tooling. JOSS papers are intentionally short, typically under 1,800 words, and for well-documented software the writing itself is straightforward, since the software is the primary contribution rather than the manuscript. The system is aimed to be developer-friendly for authors, reviewers, and editors. To be included, software must be open source and use an OSI-approved[2] license. GitHub issues are used to invite reviewers and process their reviews, in an open collaborative discussion. The large editor community makes high-level decisions collectively, including determining if submissions are suitable for JOSS, and how JOSS should change its policies and processes. Bot-driven commands within reviews are usable by authors, reviewers, editors, track editors.

JOSS is unlike most hosted journals in the academic infrastructure ecosystem by nature of its GitHub-based publication model. The incentives for the founders was against using strongly-integrated systems, but rather, supporting standard APIs that allow competition and change at each layer. Reviews are on GitHub, but could potentially be done by email or a chat server. The website is currently hosted on Heroku, but could also be moved to another provider such as GitHub (natively) or AWS. The bot uses Heroku and GitHub Actions. The source code is available on GitHub[3].

Software archiving is left up to the author; most authors choose Zenodo, although figshare and institutional repositories are also used. The only requirement is that an archive stores the open source software and assigns a Digital Object Identifier (DOI). Paper compilation uses pandoc [Mac26]. Metadata for the paper is sent to Crossref [Cro26], which assigns the final DOI. The papers and reviews are also archived by Portico [Por26].

Due to the extensive use of existing systems, the JOSS costs are very low, less than US $5 per paper. This figure was calculated and documented in 2019, and is still broadly accurate [KBNS19]. JOSS has no full-time staff, and is entirely volunteer run, although some past grants

[1] https://joss.theoj.org
[2] https://opensource.org/licenses
[3] http://github.com/openjournals/buffy/

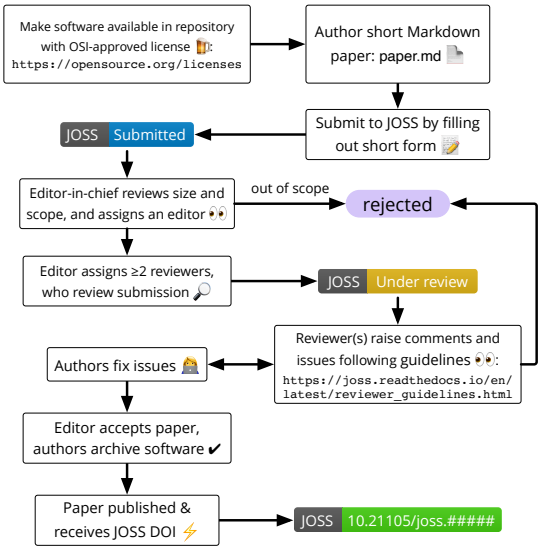


Figure 1: The JOSS Publishing Flowchart [Nie17]

have helped fund development work [DSK+25]. JOSS is fiscally sponsored by NumFOCUS.[4]

3 JOSS reviews

In JOSS, submissions are first handled by the track associate editor-in-chief (AEiC), with the track selected based on content provided by the submitting authors. JOSS's tracks are:

- Astronomy, Astrophysics, and Space Sciences
- Biomedical Engineering, Biosciences, Chemistry, and Materials
- Physics and Engineering
- Social, Behavioral, and Cognitive Sciences
- Data Science, Artificial Intelligence, and Machine Learning
- Earth Sciences and Ecology
- Computer science, Information Science, and Mathematics
- Miscellaneous

The AEiC checks the submission to ensure that it is suited to JOSS and follows its requirements, then assigns an editor, from those who preselected themselves as being able to manage

[4] https://numfocus.org/

submissions in that track. This occurs in public in a pre-review issue that the AEiC opened. The assigned editor's job is to manage the paper through submission; initially, this means finding reviewers. Authors may suggest reviewers from a list of academics who have volunteered to review for JOSS or from external sources. The list of potential reviewers is open for review (and for more reviewers to join if they are interested in reviewing for JOSS), as are the recommendations made by the author. Editors are expected to have a knowledge of their field, and to know reviewers who may be satisfactory, if no reviewer from the volunteer list is suitable. They can search Google, Google Scholar, other journals, previous authors in JOSS, and other resources to find reviewers.

Once reviewers are selected, the pre-review issue is closed and a new review issue is opened. At that point, the editor's job is to help reviewers and authors come to an agreement about if the submission is ready to be accepted, and if not, what else needs to be done by the authors. Most often, this means nudging reviewers or authors for timeliness, or to provide guidance on the review process and criteria for admittance, much of which is documented in the JOSS documentation. Authors and reviewers must agree to JOSS's Code of Conduct and also report any conflicts of interest. We expect reviewers to complete their reviews in 4-6 weeks, and authors should generally respond to reviewer feedback within 2 weeks and complete requested changes within 4-6 weeks, unless otherwise negotiated with the editor.

The review process for JOSS is open; all reviewer comments and interactions are made on the public GitHub review issue dedicated to that paper. JOSS reviews are checklist-driven, and largely based around the open source repository that the paper discusses. General checks include: a repository URL; a valid, OSI-approved license; contribution and authorship information; and documentation outlining the scope and significance of the work in the paper. This can touch on the development history and openness, which includes information on the development timeline, the state of openness for the development process, any record of collaborative effort, and best practices followed. The code must be functional, and include information on installation, functionality, performance, and testing. The documentation must be clear, and must include a statement of need, installation instructions, example usage, functionality documentation, automated tests, and community guidelines.

The paper itself includes these sections: a summary, a statement of need, the state of the field, software design, research impact statement, an AI disclosure (see below), and references. The quality of writing is also checked by the reviewers and editor, and finally by the AEiC.

For JOSS review criteria, some criteria have guidance. For instance, for the Installation section, the JOSS documentation says:

> Software dependencies should be clearly documented and their installation handled by an automated procedure. Where possible, software installation should be managed with a package manager. However, this criterion depends upon the maturity and availability of software packaging and distribution in the programming language being used. For example, Python packages should be `pip install`able, and have adopted packaging conventions, while Fortran submissions with a Makefile may be sufficient.
>
> **Good**: The software is simple to install, and follows established distribution and dependency management approaches for the language being used.

OK: A list of dependencies to install, together with some kind of script to handle their installation (*e.g.*, a Makefile).
Bad (not acceptable): Dependencies are unclear, and/or installation process lacks automation.

While the checklist has toggleable boxes, the intent is to create conversation between the reviewer and the author, not to allow performative or perfunctory sign-off.

3.1 JOSS as a community

JOSS practices have influenced reviewers and developers in terms of what is good and what is minimally acceptable, similar to rOpenSci's influence in the R community [BCHR15, RBC+19]. Cultures change based on rules and incentives: JOSS attempts to provide rules, and at a high-level, tries to nudge incentives. Over time, the software community changes, and JOSS levels also change to become stricter. Currently, the largest such change is with the rise of GenAI (see below). Other changes have also occurred.

Additionally, if software was cited directly, JOSS papers wouldn't be needed; but JOSS has enabled many authors to begin citing software, which has significantly influenced the field and related projects like `Citation.cff` [DSC+21]. We believe that JOSS reviews and JOSS community would still be important in shaping values even if all software were cited.

3.2 Observations on working openly

JOSS chose an open process because its founders believed the positive aspects of doing so far outweighed the negative ones.

Positive aspects include that many integrity problems do not occur because they would be publicly seen (declarations of potential conflicts are public, review content and editorial decisions are documented in the open, and there are clear expectations of authors), that reviewers often volunteer into a JOSS reviewers pool, and that JOSS generally needs a relatively small number of invites to identify reviewers, with the average being two to three invites per reviewer. Openness also reduces the complexity of the infrastructure, and participants can link to their reviews.

Negative aspects include that some editors receive semi-regular emails from people who are annoyed they haven't been asked to review yet, and likewise, there can be a vanity software package "pile-on" effect, where for high-profile open-source projects, there are often too many reviewers volunteering. In addition, good reviewers can become well known quickly, potentially leading to overuse and reviewer burnout. This may be partly because there is little privileged information in an open system. Finally, authors sometimes chase reviewers, editors, etc., which can lead to potential cultural barriers to entry for some and negative dynamics for junior staff. Several JOSS editors are also editors for other journals, and we note that the scale of these issues are unique to JOSS.

3.3 Scaling JOSS

Most of our challenges, before GenAI, have been about scaling people processes: AEiC/managing editor rotations, finding more editors, and adding in term limits for editors (to avoid burnout).

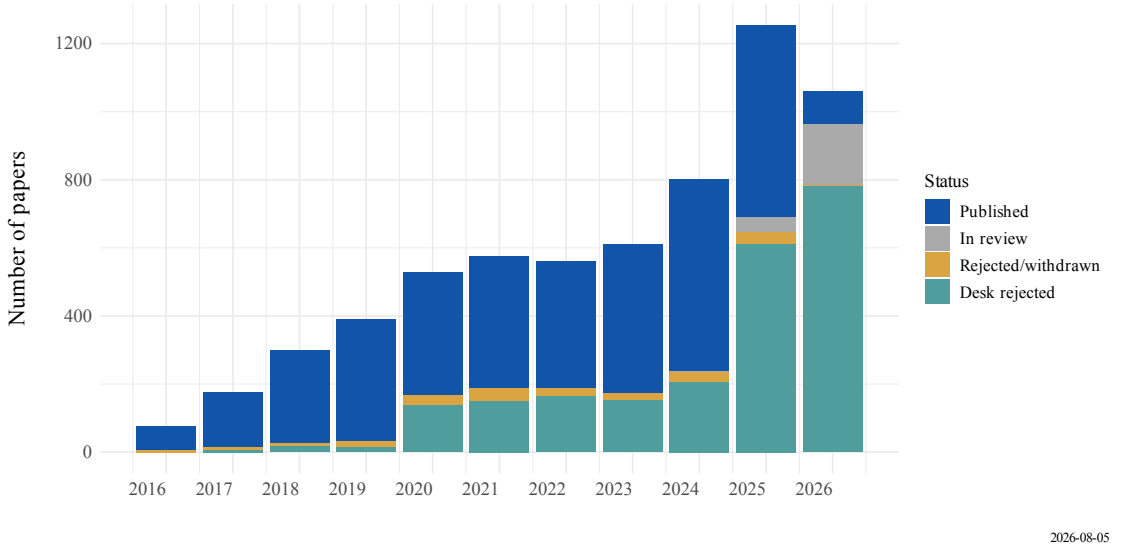


Figure 2: Number of papers desk rejected, reviewed and published per year. Papers in review had their issues opened in the year to which they were assigned.

Similarly, there have been technology improvements: Smarter reviewer assignments (for instance, by using the editorial bot to suggest similar reviewers and papers from our internal reviewer management system), better monitoring tools for editors, and better tools to help authors prepare their submissions. However, the greatest challenge for JOSS currently is undoubtedly the influence of AI.

4 The Impact of Generative AI

In 2020, with the introduction of GPT-3, large language models (LLMs) became known beyond the research community. The OpenAI Codex was subsequently trained on both natural language and source code from repositories, enabling models to generate and understand code. In 2021, GitHub Copilot, based on OpenAI's Codex, was released and provided real-time code suggestions directly within development environments.

Beginning in 2022, numerous LLM-based tools for code generation were introduced. Today, agentic AI systems are increasingly used to automate software development tasks, including code generation, debugging, testing, and refactoring. With AI agents now capable of producing entire codebases from natural language prompts, the marginal cost of code *generation* is rapidly approaching zero. In theory, this enables people of all backgrounds (and in some cases without directly interacting with the code at all, so called *vibe coding*) to create new software tools. However, LLM-generated code does not always exhibit high quality, correctness, or maintainability, and the generated code and artifacts can be difficult for humans to understand and debug.

JOSS was created to reward developers of research software with publications, since papers are a well-defined currency in the academic system. That the software must be useful for research

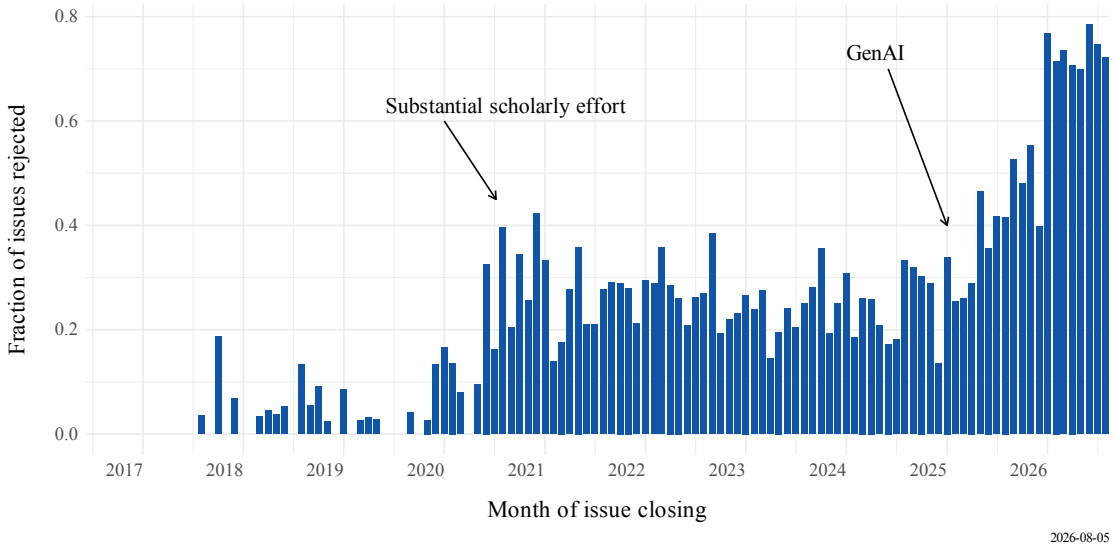


Figure 3: Desk rejection rate before review, from mid 2016 to late 2025, with y-axis lines at 20% intervals. Shows approximately when 2020 "substantial scholarly effort" and 2026 GenAI-motivated scope reviews were introduced.

has been a criterion from the start. In 2020, a second criterion—*substantial scholarly effort*[5]—was introduced and rigorously enforced in order to balance peer-review and credit for authors with academic trust in JOSS papers being equal to peer-reviewed journal papers.

The benchmark at the time offered a practical, human-centered proxy for effort: developers should have invested more than three months of work, and editors could judge this, in part, by the size of the codebase, where over a thousand lines of code (LOC) was seen as a positive sign. Papers that were judged to have had too little human effort or that were judged to not be research software were considered out of scope and were desk rejected. The more rigorous enforcement increased the rejection rate to about 20%. Figure 3 shows the rejection rate before review. Note that, in addition, about 5% of papers are rejected during the review, and this rate has been fairly constant over the 10 years of JOSS. Additionally, submissions can be withdrawn by the authors, if they do not have the time to address the reviewer's comment or do not want to address them.
The benchmark at the time offered a practical, human-centered proxy for effort: developers should have invested more than three months of work, and editors could judge this, in part, by the size of the codebase, where over a thousand lines of code (LOC) was seen as a positive sign. Papers that were judged to have had too little human effort or that were judged to not be research software were considered out of scope and were desk rejected. The more rigorous enforcement increased the rejection rate to about 20%. Figure 3 shows the rejection rate before review. Note that, in addition, about 5% of papers are rejected during the review, and this rate has been fairly constant over the 10 years of JOSS. Additionally, submissions can be withdrawn by the authors, if they do not have the time to address the reviewer's comment or do not want to address them.

[5] https://blog.joss.theoj.org/2020/07/minimum-publishable-unit

For that era, the three-month, ∼1000 LOC benchmark served JOSS well. Today, however, the landscape is fundamentally different: where generating a thousand lines of code once took three months, it can now be accomplished in three days, or even less than three hours, with generative AI tools. In 2025, the submission of JOSS papers containing AI-generated content increased markedly. In the last nine months of 2025, JOSS rejected around 40–50% of the submissions. We know of no other external reason for this increase in papers; there has been no corresponding rise in JOSS awareness or research software citation practices that we are aware of. Due to limitations for bot detection using Google Analytics, we are also not able to report on a rise in bot traffic specifically over this time period. Nevertheless, the rise in submissions and rejections indicated that the previous proxies of three months and roughly a thousand lines of code are no longer a sound threshold for a rigorous scholarly contribution criterion in the era of generative AI.

We do not attempt identification or keep an index of generative AI usage in papers. However, there are signals that indicate AI usage; *e.g.*, when large amounts of code is developed or committed using Git in a very short time with no clear inputs, along with the fact that much AI-generated code includes AI agents as the committer. Also, we see patterns that AEiCs recognize in such submissions that also appear in other submissions, which imply that they are AI-generated.

To address this issue, JOSS implemented a new *GenAI* and scope policy[6] at the start of 2026. This, along with the continued submissions of articles similar to those in 2025, has led to an increased desk rejection rate of 70–80%, as shown in Figure 3. However, there has been no change in the number of rejections post-review, as can be seen in Figure 4.

The shift is one of emphasis: rather than evaluating software through proxies for effort such as lines of code, JOSS now looks for clear evidence of research impact, intellectual contribution, scholarly significance, and sound open-source software practice. As AI becomes capable of handling increasing fractions of the actual implementation work, the irreplaceable human contributions to software become clearer: understanding context, collaboration with others, making design trade-offs, creating abstractions that capture domain expertise, and building the conceptual foundations and sustainable practices that enable future discovery. Regardless of whether authors leveraged AI assistance, submissions must show the human work. All papers not already under review after the release of the new policy must now have sections for:

- **Research impact**: Compelling evidence of realized impact or near-term significance for the scientific community.
- **Software design**: How was the software package designed and why this architecture was chosen.
- **AI usage disclosure**: Documentation of any usage of generative AI in the code or documentation or paper and how the tools were used. If generative AI tools were used, how was the quality and correctness of the generated content assessed.

As before, JOSS checks for best open-source practices, such as comprehensive testing, clear documentation, statements describing support and governance, and clear pathways for community contribution. In an era of rapid code generation, the work of creating maintainable, well-

[6] https://blog.joss.theoj.org/2026/01/preparing-joss-for-a-generative-ai-future

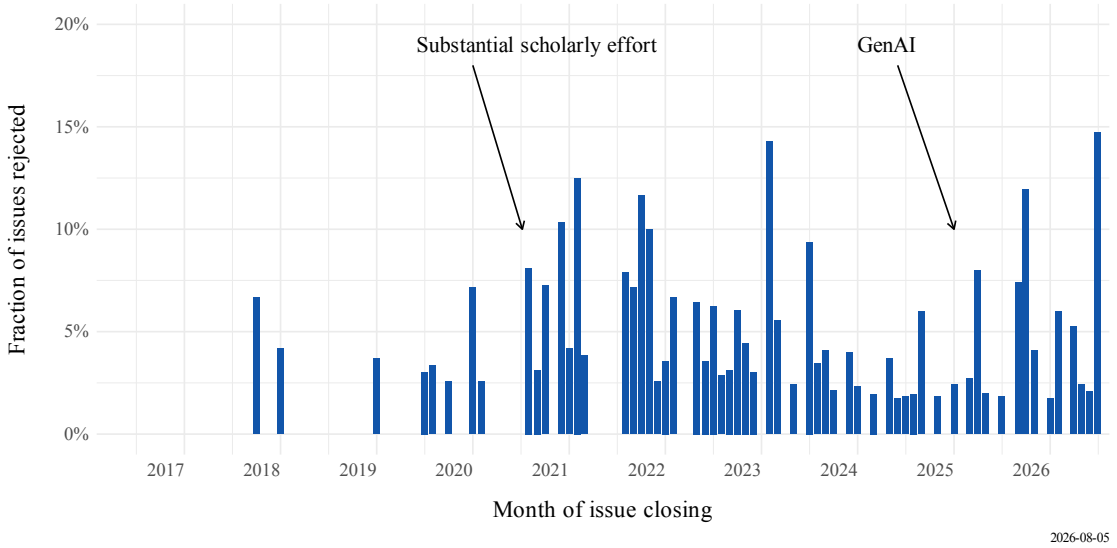


Figure 4: Rejection rates post-review from mid 2016 to late 2024, showing no substantial changes after implementing both "substantial scholarly effort" scope and new GenAI scope policy in January 2026.

documented, test-driven systems becomes even more valuable, not less. Software should be developed and maintained in the open (ideally with open decisions) rather than developed privately and deposited online with an OSI-approved license only to meet minimal open-source requirements. With respect to community, meaningful research software typically emerges from sustained collaborative effort over time. JOSS therefore examines a project's development timeline and commit history as indicators of genuine community engagement and iterative refinement. The repository must have a public record of more than six months of open development, ideally from the beginning of the project with evidence of active development, such as releases, public issues, and pull requests. Projects that do not meet this six-month threshold are out of scope. Even for projects with longer histories, reviewers will look closely at why the contribution is credit-worthy, examining the problem framing, design decisions, and abstractions that reflect genuine scholarly contribution rather than rapid AI-assisted code generation. Finally, engagement from individuals beyond the core team or paper authors, whether through code contributions, issues, reviews, or other feedback, is a strong signal of impact beyond the originating team.

5 Citing JOSS articles

As previously noted [DSK+25], JOSS started with an 11-person editorial team, and published its first paper in May 2016. There are currently 9 track editors on the main editorial team, and 105 topic editors. Over 3500 papers have been published as of May 2026. Of these, around 800 have zero citations, while around 600 have 20+ citations. JOSS as a whole has an *h-index* of 125; 125

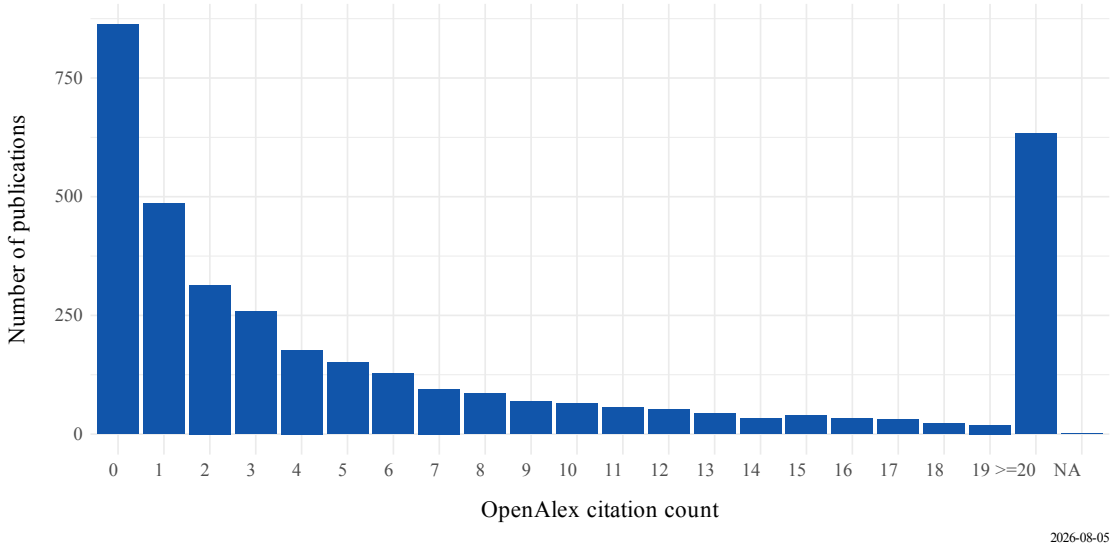


Figure 5: Citation counts per paper, as measured by OpenAlex.

papers have been cited 125 times.

The five most cited papers as of 27 May 2026, using data from OpenAlex, are:

1. Wickham *et al.*, (2019). Welcome to the Tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686, 21 172 citations

2. McInnes *et al.*, (2018). UMAP: Uniform Manifold Approximation and Projection. Journal of Open Source Software, 3(29), 861, https://doi.org/10.21105/joss.00861, 9430 citations

3. Lüdecke *et al.*, (2021). performance: An R Package for Assessment, Comparison and Testing of Statistical Models. Journal of Open Source Software, 6(60), 3139, https://doi.org/10.21105/joss.03139, 5100 citations

4. Lüdecke, (2018). ggeffects: Tidy Data Frames of Marginal Effects from Regression Models . Journal of Open Source Software, 3(26), 772, https://doi.org/10.21105/joss.00772, 2609 citations

5. McInnes *et al.*, (2017), hdbscan: Hierarchical density based clustering, Journal of Open Source Software, 2(11), 205, https://doi:10.21105/joss.00205, 2608 citations

Citations for JOSS depend not only upon availability of the papers for researchers, but also on institutional buy-in. JOSS is not yet indexed in Scopus or the Web of Science, although we have repeatedly applied. JOSS is recognized by several national registers and research indicators [DSK+25].

6 Conclusion

JOSS's record shows that it successfully met its original goal of creating a means by which open source research software developers could have their software peer-reviewed as part of a process that also gave them a publication about the software, which can be used as a placeholder in the scholarly publication system to give them credit for their work. JOSS's peer-review system is very open, and has led to the development of a community that includes JOSS volunteer editors, reviewers, and authors, working together to define and improve research software.

There has always been some tension in JOSS between rewarding all software developers and those whose software significantly impacts the scholarly research community. When this was first realized, JOSS responded by creating a minimal effort requirement, judged in large part by the size of the software that was submitted. The increased use of generative AI in JOSS submissions, making it easier to quickly generate large amounts of code, becoming clear through 2025, led JOSS editors to change this requirement from simply code size to human effort and software impact. While this initially has led to an increased rejection rate of submissions as authors continue to submit software that doesn't meet these new requirements, we believe that, over time, JOSS will publish better, more significant software packages, though perhaps fewer.

Supplementary materials

Acknowledgements: The U.S. Department of Energy supported this work through the Los Alamos National Laboratory. Los Alamos National Laboratory is operated by Triad National Security, LLC, for the National Nuclear Security Administration of the U.S. Department of Energy (Contract No. 89233218CNA000001). Approved for public release with LA-UR-26-24097.

Data availability: The dataset used to generate the figures in this paper is available on Zenodo[7] and GitHub[8].

Declaration on the use of AI: No AI was used in the creation of this article.

Bibliography

[BCHR15] C. Boettiger, S. Chamberlain, E. Hart, K. Ram. Building Software, Building Community: Lessons from the rOpenSci Project. *Journal of Open Research Software*, Nov 2015.
doi:10.5334/jors.bu

[7] https://doi.org/10.5281/zenodo.21496776
[8] https://github.com/openjournals/joss-analytics

[Cro26] Crossref. Crossref REST API: Works Endpoint. https://api.crossref.org/works, 2026. Accessed: 2026-07-29.

[Dat14] Data Citation Synthesis Group. Joint Declaration of Data Citation Principles. Technical report, Force11, 2014.
doi:10.25490/A97F-EGYK

[DSC+21] S. Druskat, J. H. Spaaks, N. Chue Hong, R. Haines, J. Baker, S. Bliven, E. Willighagen, D. Pérez-Suárez, A. Konovalov. Citation File Format. Aug. 2021.
doi:10.5281/zenodo.5171937

[DSK+25] P. Diehl, C. Soneson, R. C. Kurchin, R. Mounce, D. S. Katz. The Journal of Open Source Software (JOSS): Bringing Open-Source Software Practices to the Scholarly Publishing Community for Authors, Reviewers, Editors, and Publishers. *Journal of Librarianship and Scholarly Communication* 12, 2 2025.
doi:10.31274/jlsc.18285

[KBH+19] D. S. Katz, D. Bouquin, N. P. C. Hong, J. Hausman, C. Jones, D. Chivvis, T. Clark, M. Crosas, S. Druskat, M. Fenner, T. Gillespie, A. Gonzalez-Beltran, M. Gruenpeter, T. Habermann, R. Haines, M. Harrison, E. Henneken, L. Hwang, M. B. Jones, A. A. Kelly, D. N. Kennedy, K. Leinweber, F. Rios, C. B. Robinson, I. Todorov, M. Wu, Q. Zhang. Software Citation Implementation Challenges. 2019.
doi:10.48550/arXiv.1905.08674

[KBNS19] D. S. Katz, L. A. Barba, K. Niemeyer, A. M. Smith. Cost Models for Running an Online Open Journal. June 2019.
doi:10.59349/g4fz2-1cr36

[Mac26] J. MacFarlane. Pandoc. 2026.
https://pandoc.org/

[Nie17] K. Niemeyer. JOSS publication flowchart. figshare, 2017. The JOSS Flowchart.
doi:10.6084/m9.figshare.5147773

[Por26] Portico. Portico Digital Preservation Archive. https://www.portico.org/, 2026. Accessed: 2026-07-29.

[RBC+19] K. Ram, C. Boettiger, S. Chamberlain, N. Ross, M. Salmon, S. Butland. A Community of Practice Around Peer Review for Long-Term Research Software Sustainability. *Computing in Science & Engineering* 21(2):59–65, 2019.
doi:10.1109/MCSE.2018.2882753

[SKN16] A. M. Smith, D. S. Katz, K. E. Niemeyer. Software Citation Principles. *PeerJ Computer Science* 2:e86, Sept. 2016.
doi:10.7717/peerj-cs.86

[SNK+18] A. M. Smith, K. E. Niemeyer, D. S. Katz, L. A. Barba, G. Githinji, M. Gymrek, K. D. Huff, C. R. Madan, A. Cabunoc Mayes, K. M. Moerman, P. Prins, K. Ram,

A. Rokem, T. K. Teal, R. Valls Guimera, J. T. Vanderplas. Journal of Open Source Software (JOSS): design and first-year review. *PeerJ Computer Science* 4:e147, Feb. 2018.
doi:10.7717/peerj-cs.147