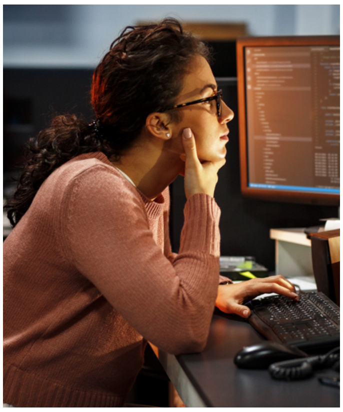
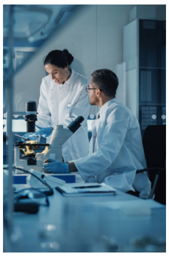




Australian Research Data Commons

Research Software Visibility Infrastructure Priorities Report

By Dr. Karthik Ram, Dr. James Howison
for the *Australian Research Data Commons*
October 2023

Australian Research Data Commons

10.5281/zenodo.8404846

Suggested citation:

Ram & Howison (2003).
Research Software Visibility Infrastructure Priorities Report.
https://doi.org/10.5281/zenodo.8404846

Cover Images Clockwise
BalanceFormCreative - 188531911 / stock.adobe.com
Mario Purisic - jG1z5o7NCq4 / unsplash.com
Gorodenkoff - 402341966 / stock.adobe.com
Hero Images - 316159695 / stock.adobe.com
Gorodenkoff - 236237514 / stock.adobe.com
themorningglory - 391783296 / stock.adobe.com
Associate Professor Sandra Philips - Marc Grimwade / ARDC

CONTENTS

EXECUTIVE SUMMARY

Research software has become critically important for modern research. Not only do researchers increasingly rely on software for their work, but many are involved in the creation, development, and maintenance of such tools as part of their research activity and output. The visibility of research software is crucial to research effectiveness. In this report, we provide insight into infrastructure opportunities to advance the visibility of research software, based on a consultation consisting of six focus groups of stakeholders, including participants across the ARDC's three thematic research data commons (health and medical research, earth and environmental science research, and humanities, arts, social science and indigenous research), as well as our experience in these topics. Highlights of our focus group findings include:

- Researchers struggle with the **discovery and evaluation** of tools; participants highlighted the **importance of fully worked use cases** for onboarding different skill levels and for evaluation to avoid investing costly effort into creating their own software.
- Citation was emphatically recognized as important. Recent work, including that of the ARDC, has helped practitioners understand *how* to cite software. Yet there was widespread **uncertainty about community norms on *which* software to cite within the available space for references in publications**.
- Tracking the impact of scholarly work such as software was seen by researchers as tedious, manual, and incomplete. There is a desire among researchers to have more **infrastructure to track the impact** of software for use in promotion and funding packets. Focus group participants also recognized **the potential of web analytics data** as a supplemental way of demonstrating the impact and reach of software and associated contributions.
- **Doing the work needed for software archiving was not well motivated for researchers** and many expressed concerns that the burden of the additional effort required for long-term archiving was not compensated or recognized.

We analyzed these results, particularly for differences across thematic areas, and have highlighted different emphases below. Nonetheless, we did not find these differences large enough to make separate recommendations for different disciplinary or research topic areas.

In the body of this report, based on the focus groups and our experience in these topics, we make and explain 8 recommendations:

1. **Work with research domain leaders to build norms around which software to cite 11**

   Software often goes uncited in academic publications, leading to challenges for researchers in recognizing its contribution. We recommend that the ARDC continue to collaborate with journal editors and scientific societies to establish field-specific guidelines on how to decide which software should be cited within the limited space of reference lists, rather than guidelines about the citation format.

2. **Support the effort to create and disseminate software bill of materials (SBOMs) 14**

   Focus group discussions highlighted the challenge of distinguishing between software crucial for understanding research results and other underlying tools. A potential solution is the "software bill of materials" (SBOM), which documents all software dependencies and can enhance reproducibility and software acknowledgment, though its implementation and uptake still faces challenges that the ARDC could help address.

EXECUTIVE SUMMARY

3. **Create a software use infrascope as an observatory based on software mentions in publications to highlight particular areas of strength or opportunity in Australia** **17**

   Software lacks visibility in research, making it challenging to identify crucial software for specific research areas. We recommend that the ARDC play a role in developing a “software use infrascope” based on recognizing software in the full text of publications to systematically understand the software used by Australian researchers, categorized by research topics. This tool would enable stakeholders to identify software trends, support training programs, and assess the impact of software packages, enhancing research collaboration and funding opportunities.

4. **Promote the creation and dissemination of fully worked example use cases aimed at different skill levels**

   Research software users often struggle to decide which tools to adopt due to insufficient documentation and real-world use cases. To address this, we recommend that software producers provide detailed use cases catering to various skill levels, ensuring these examples are easily accessible and up-to-date. Initiatives like small grants can foster collaboration between software creators and users, enhancing documentation and promoting best practices.

5. **Support existing technology approaches to software archiving** **21**

   Research software is often poorly archived, making it hard to access post-study. Focus group participants largely assumed platforms like GitHub were sufficient for long-term storage despite potential risks. We recommend that the ARDC promote the use of existing large-scale automated archives like Software Heritage and Zenodo to ensure the long-term archiving of Australia’s software.

6. **Support specialized communities of practice (online and offline)** **22**

   Researchers often require assistance with software discovery and usage. Focus group participants emphasized the value of local support venues like “hacky hours” and online communities centered around specific software ecosystems. We recommend that the ARDC continue supporting local communities of practice and support community activators for specific research software areas, promoting active engagement and organic growth within these communities.

7. **Provide guidance for implementing web analytics to understand usage** **23**

   Software projects require reliable data to showcase their impact and understand their user base. While traditional metrics like download counts have credibility issues, focus group participants emphasized the potential of website analytics to gauge impact. We recommend that the ARDC promote the use of web analytics on the documentation and websites of research software projects, offering guidance, training, and resources to maximize their benefits.

8. **Provide a low-friction way for researchers to link to software alongside data submissions** **24**

   Focus group participants emphasized the importance of software discovery but showed more interest in comprehensive example case studies rather than software-specific catalogs. Our recommendation is for data catalogs to integrate easy-to-create links to software frequently used to work with that data.

CONTEXT

The research software landscape has evolved considerably in recent years. Compared to a decade ago, it is significantly easier to develop new open source software thanks to the widespread availability of training (Wilson, 2014), communities of practice (Ram et al., 2019), advances in tooling (Salmon & Ram, 2021), and the advocacy efforts of organizations like the UK Software Sustainability Institute (Crouch et al., 2013), Workshop on Sustainable Software for Science: Practice and Experiences (Katz et al., 2014), the US Research Software Sustainability Institute (Carver et al., 2018) and the Research Software Alliance (ReSA)[1]. The Research Software Engineer (RSE) movement (Baxter et al., 2012) has normalized the role of software development in an academic context. It has provided a career pathway for many that struggled to find the right home within their institutions. However, while research software creation is much easier, many challenges remain if software is to reach its true potential including discovery, use, evaluation, and sustainability.

First and foremost, it is still exceedingly difficult to demonstrate the impact of software on research. While efforts like the Journal of Open Source Software have worked to lower barriers to publishing papers on research software (Smith et al., 2018), software is still rarely cited (Du et al., 2022) and cultural change in norms will take a long time. This lack of visibility has impacts on careers and the sustainability of the software itself. Visibility and credit are also linked to the long-term archiving of software and code, as well as the discovery of existing software products and their reuse in other research contexts.

Action required to improve software visibility should be both bottom-up (educating the research community about software citation practices) and top-down (using modern technical and social infrastructure).

The ARDC's National Agenda for Research Software (Australian Research Data Commons, 2022) puts the challenge clearly:

> The vision for the National Agenda for Research Software: Research software is recognised as a first-class output of research. Together we can make this vision a reality through concerted, coordinated action to see, shape and sustain research software.

This report builds on existing work from the ARDC within the program "Implementing the Research Software Agenda for Australia", including the reports: Understanding How Researchers Find Research Software for Research Practice (Stevens, 2022) and Research Software Capability in Australia (Barker & Buchhorn, 2022). Within the ARDC's emphasis on work to "see, shape, and sustain" research software, this report focuses primarily on infrastructure to improve the visibility of research software ("see"), drawing connections to the role of infrastructure in efforts to "shape, and sustain" research software where appropriate.

SUMMARY OF FOCUS GROUPS

Between May and June 2023, we interviewed stakeholders from the Australian research community through online focus groups. The groups were based on the classifications in the national agenda for research software, drawing on both the three-part classification of software tools (analysis code, prototype tools, and research software infrastructure), and the seven-part creators/stakeholders classification (creators, authors, maintainers, supporters, infrastructure providers, and policymakers). Participants were identified by the ARDC and, through discussion with the authors of this report, were organized into six groups: domain researchers, research software engineers, model makers, methods makers, ARDC services staff, and ARDC senior leadership team members experienced with research community liaison. Each focus group had representatives familiar with working with the ARDC thematic research data commons focused on health and medical research; earth and environmental science; and humanities, arts, social science, and indigenous research.

In each of these focus groups, we seeded discussion with questions about 4 themes related to software visibility: discovery, use, credit/acknowledgment, and archiving of software. We chose not to present examples of potential infrastructure that might be recommended, reasoning that a deeper understanding of participants' perspectives would better inform infrastructure recommendations and avoid demand bias. Focus group participants gave permission for the sessions to be recorded and transcribed for analysis by the authors. To encourage frank discussion we undertook not to identify comments by name and not to release raw transcripts. We summarized results using qualitative research techniques, including thematic analysis, creation of tables, memoing, and discussion. In particular, each author reviewed the full transcripts after an initial draft, seeking to identify elements of the discussion that could challenge interim conclusions. Given the authors' experience in this domain, we paid particular attention to topics or perspectives that were different from the existing discourse around research software visibility and infrastructure.

In this report, we summarize our findings along with a series of recommendations for the ARDC to implement. These recommendations will elevate the recognition of software, which is the interface to scientific data and methods and therefore a crucial element in the ARDC's purpose of "providing Australian researchers with competitive advantage through data."

Software Discovery

Confirming the findings of the ARDC report on software discovery (Stevens, 2022), participants generally said that software discovery was typically "*informal, non-systematic*", using searches on Google, GitHub (finding "*the right keywords*"), Twitter/X (albeit with resignation that Twitter/X is increasingly becoming less useful), StackOverflow, social connections in labs, conferences, as well as literature review (two participants mentioned software-specific publications).

> “
>
> It's pretty shocking the number of times people just stumble upon something that could have saved them many months and years of work because it was the exact thing they needed. But they didn't even know it existed. And they obviously didn't even know how to go about finding [a better] way to search for it.

Discussion affirmed the importance of social networks and research community interactions including *"hackathons … like 30 people together, mainly sort of new people to the field"*. Humanities-oriented participants particularly highlighted the importance of social networks, particularly in identifying that particular research approaches can be assisted by software, but participants across focal areas mentioned the importance of learning directly from other researchers. Further discussion argued that learning from networks of researchers could be a slower way to discover new software or approaches than its appearance in publications:

> As we push forward in new methods, new data science methods, some of the discoverability of new data science methods, new software [to support them] isn't as quick as expected, because it needs to filter through the network.

Participants did discuss collections of software, but they did not mention stand-alone software catalogs or listings as an important resource. Rather, when collections of software were referenced, participants highlighted services beyond software listing and description, from hosting development (searching GitHub) to including locating software via package managers such as PyPi[2], BioConductor[3], and CRAN[4] *"where it is externally audited before it is accepted… the code is run … the patches are built … every two days … you get notified so you can fix your software"*.

Discussion indicated that organizations providing and hosting research data could do a better job of highlighting the tools that were frequently used to analyze the data, as well as highlighting the venues where participants could get training and support in using and analyzing the data.

Use

Focus group discussion of software use was centered on the identification and evaluation of software.

Participants were consistent and emphatic in calling for fully worked example use cases *"with data, real data"* that would show different on-ramps for possible users, *"sort of different entry points for users with different levels of skills and experience"*. Analysis of these comments suggests developing use cases that consider personas of potential users that differ in their familiarity and needed depth of engagement with the packages (as well as operating systems or high performance computing environments). Fully worked use cases were also cited as important for those assessing whether to use software or whether to write their own; one participant made it clear that having to invest large quantities of time just to discover the capabilities of the package made them much more likely to write their own, and much less likely to even attempt to contribute code to that project (less *"worth my time to contribute"*). Participants from earth and environmental science research argued this indicated that high-quality on-boarding documentation was expected with proprietary software, but often lacking with open source software.

Seeking help with software (installation, use, selection) frequently came up. In addition to general platforms such as StackOverflow, research-specific venues were mentioned. One participant attributed the success of their software to a specific online forum, "*which was already very popular when we launched our software … if it wasn't, I don't know what we would have done"*.

One somewhat surprising distinction emerged: while ARDC staff expressed confidence that researchers would be happy and quick to ask questions online, researchers and research software engineers were more likely to describe being disinclined to ask questions in open forums, highlighting their preference for searching for existing questions and asking questions in more closed settings. In addition to seeking help in their own lab, these participants frequently cited "hacky hours" as valuable settings with high trust, low risk, quick turnaround, and expectation of reciprocation. Downsides of public forums were discussed, from the clumsy overhead of "*having to create yet another account*" to a disinclination to participate in them.

> We would love our users to just be calm, just be not afraid to post a GitHub issue and just explain the problem, because our response rate is pretty fast. But I know for certain that people don't do that, because if something doesn't work, they just, they just fail because they're not. They're not ready to post stuff online on a public forum.
>
> And for my own stuff, I mean, I tried to provide [many channels, but] despite all of that, it's usually mainly people just emailing me, which is my least preferred way of getting requests for help, but it's the way people seem most comfortable with.

Important also to highlight is the self-censorship reported as a result of women being treated dismissively when asking questions in open source oriented public forums:

> I mean, it's much easier to hide things like gender online, but you still are going to read answers in a gendered way. And feel like a stupid woman. When, you know, somebody with some male username tells you that you're an idiot, you're gonna feel like some idiot woman stepping into some male space.

Credit

Researchers and research software engineers were unanimous and clear in their emphasis on the crucial value of citation as appropriate credit, one group quickly agreeing that *"citation is king"*. Conversely, a number of participants employed as professional staff indicated that citations were not crucial to making their case for impact, arguing that more traditional measures of user uptake and satisfaction were important for them.

Discussion around citation revealed a particular tension, sometimes clear within individual speakers (rather than a disagreement between speakers).

On one hand, participants wanted the software to be acknowledged through citation, or at least felt it was the only useful currency, and that it did not occur enough: *"people won't even cite the package, let alone a really important dependency, and sometimes that can be really critical"* and *"citations are important for software people as well, you know, if we're in the research space, that's how we build our careers"*.

On the other hand, there was discomfort with the idea that *all* software used should be cited. This was expressed in two forms. First, clarity that the sheer number of citations needed would "*overwhelm*" the publishing system. Participants reinforced this with stories of attempting to include citations to all software used, but having colleagues or editors require them to be trimmed.

> And they cite everything about the [field], like, you know, from 1920, or whatever. And then when it comes to, we write the method section, and we cite every single package we use, that adds quite a lot of citations, they get quiet … [saying] sort of, well, 'we have to cut down on our citations. So we don't think these are as important' [or] 'can we move them to a supplementary material?

Second, some participants worked towards a distinction drawing on an understanding of norms in scientific fields about what kinds of contributions rise to the level that they ought to be cited. These distinctions were nascent and evolving in the discussion. They included criteria such as novelty, uniqueness, whether the author needed scientific credit (if the code was *"based on my research"*), whether the software merely implemented well-known or straightforward techniques like *"data wrangling"*, or implemented techniques that advanced scientific practice. One participant phrased it as *"a distinction between sort of engineering content and intellectual or research content in deciding [what to put in limited citation lists]"*. Another developed a distinction between *"novelty"* building on another's comment regarding *"intellectual content"* and strongly contrasted that with *"usefulness [in making] everyone's life 25% easier"*. They argued that citations (properly guided and improved) could measure *"novelty"* but *"usefulness"* needed a *"different metric … an alternative metric"*. In the words of another participant *"… the citation system just doesn't fit software, and we're trying to ram it in there"*, suggesting that, *"a national organization can … come up with an alternative metric [of actual use], which then people could put forward on their CVs".*

Participants were emphatic and unanimous in calling for greater guidance on these principles: saying *"There are no real established norms [for which software to cite]"* and *"If there's a really important dependency that is worth citing, [it is] not clear whether you cite that or not"*. They invoked strong crediting norms in research outside software saying that there is *"not the same stigma"* for leaving out contributions via software. When prompted for whom they should expect to provide these guidelines, many diverse suggestions were made including scientific societies, well-known scientific leaders, libraries, software language communities: *"some sort of consortium or group or task force or whatever the name is … working with … professional [scholarly] bodies."*

During the discussion of alternative metrics for usefulness, participants were not confident that measures like download counts were meaningful, citing multiple issues with their credibility: *"So you put in your promotion application, I've done this package, it's been downloaded x times. And we know that's a very bad metric. But I don't even know if that sentence in a promotion application is making a difference."* There was more confidence in the idea of linking usefulness to when the software was used in published research, albeit a lack of confidence that the citation system can be repurposed to gather that information: *"You know, you could have thousands and thousands of people using it. But if there's no publications that come out of that, potentially research institutions would not be super interested in providing funds."*

Finally, research software engineers mentioned using website analytics tools installed on project websites and documentation to gain insights into possible users that would be more credible than download statistics, highlighting both unique users and data on their location and institutional sector (for example, research vs government). There were no perceptible differences in this topic across the three ARDC thematic areas.

Long-Term Archiving

Discussion about archiving focused primarily on two approaches: inclusion in code collaboration platforms such as GitHub, together with services such as Zenodo (which archives repositories and provides a digital object identifier, or DOI). Participants emphasized that the frequently changing nature of software required automated solutions for creating DOIs, *"it's never really finished. So there will always be a new DOI."* and *"it's not just having places to put stuff, it's also having the mechanisms that enable that to happen automatically"*.

Participants, in general, did not indicate dissatisfaction with these approaches, perhaps reflecting a lack of motivation for longer-term, more reliable, and sustained archives of software, *"I hope that I would have enough time to move everything to whatever new tool was available. But yeah, it's kind of not something that I really consider when I choose whether to put something on GitHub or not."*. We prompted them with questions about institutional repositories and global archiving services such as Software Heritage. No participant indicated that they had directly used these long-term software archiving repositories or searched for software in them. There were no perceptible differences across the three ARDC thematic areas.

Additional Topics

At the end of the focus groups, we created time for participants to emphasize anything they thought relevant. Three additional topics came up: feedback on the typology of research software from the National Agenda, comments on the need for clear career paths for research software engineers, and the overall importance of specific funding for research software work.

Feedback on the ARDC three-part typology of research software (analysis code, prototype tools, and research software infrastructure) was obtained in non-ARDC focus groups. This typology is presented in the ARDC National Agenda for Research Software (Australian Research Data Commons, 2022). In general, there was good support for this typology, with each of the focus groups easily identifying themselves within the typology. That said, one participant pointed out that porting research software across operating systems or execution environments was an important activity that could occur in any of the three general categories.

Improving and clarifying career paths for research software work were important to multiple participants: "*to get to this point has basically been a purely voluntary journey for many, many people, and I don't think that's sustainable long term"* and these comments reinforce the recommendations from the 2021 ARDC report Research Software Capability in Australia (Barker & Buchhorn, 2022) that "there is a sizable and growing community research software support capability that needs to be developed, retained, sufficiently skilled and valued" (p. 4).

SUMMARY OF FOCUS GROUPS

Participants made it clear that specific and increased funding for software work was fundamental. The understanding we obtained from these discussions was that the issues discussed were important but improvements in those areas were likely to be minimal unless it was accompanied by specific funding for software. This theme was repeated across all ARDC thematic focus areas and focus groups.

> Always having to put a [domain science] spin on it or whatever ... it's really crap, because writing the paper, and getting it through review, is really time-consuming... and that time working on the next cool thing, or writing more software or making our software even better. And in Australia, it's actually really hard to find funding for software.
>
> **Method Makers Focus Group**

> I'm really, really pleased to see ... national statistical software awards. But like, more funding to provide ongoing maintenance ... I would love to be able to tap into small amounts of money for that. ... But that is something I was sort of hoping to discuss.
>
> **Research Software Engineer Focus Group**

> Talking about maintaining that code after the projects wrapped up... creating a community that would maintain it after the project and having that being part of the, you know, the grant application.
>
> **Researcher Focus Group**

RECOMMENDATIONS

1 Work With Research Domain Leaders to Build Norms Around What Software to Cite

Problem: Software often goes unrecognized since it is not cited in the same way as to peer-reviewed papers in the reference section of a publication. Unlike papers, software as a scholarly output also poses unique challenges for citation in that the same software can have multiple things that can be cited, each with varying authors (for example, a paper about software, a persistent identifier for each version of the software, a collaboration repository without a persistent identifier). Since most software creators don't provide specific guidelines, researchers are left struggling to figure out what to cite. Even if a preferred citation is easily located, some additional challenges arise. Journals often limit the number of references, forcing authors to make hard decisions on what to cite. Not all the software used in a paper will be referenced in the text, which poses another challenge for referring to these citations in the paper. (see quote).

> “
>
> I have encountered times in my research career where I tried to cite every package that I use, every time I called like a library, this package. [colleagues said,] ‘that's too many ... you have to choose [the most important].

Focus group outcomes: Participants noted uncertainty around what to cite and agreed that it would be too impractical to cite every piece of software used. Instead, participants expressed a preference to cite only domain-specific code that was relevant to the results and discussion. The rationale for this is that there are numerous ways to accomplish more general tasks such as reading, tidying, and visualizing data. The lack of detail about these tools would not necessarily impede reproducibility but would allow researchers to focus on citing only the highly relevant domain code used in a paper. In contrast to citing all software used, this approach would ensure that readability is maintained. High level descriptions of relevant models or algorithms provide a natural place for citation. This recommendation closely matches the FORCE11 implementation group recommendations “You should cite software that has a significant impact on the research outcome presented in your work, or on the way the research has been conducted” and their later recommendations to avoid citing software that is “commonplace in your field” (Chue Hong et al., 2019).

Nonetheless, participants also expressed disquiet, acknowledging that the limited space for formal citations, and even other practices of mentioning software in the full-text, meant that there was limited visibility for software they knew had made their work better. We address this tension through recommendation #2.

Our recommendation: The ARDC has already carried out quite a bit of work around software citations (Liffers & Honeyman, 2021)[5] and should continue the work of normalizing formal software citations. Yet the focus groups in this study made it clear that the issue was not syntactical (how to cite) but related to norms of intellectual writing (what to cite, and why to cite some things but not others).

In particular, we recommend that the ARDC continue to work with journal editors and scientific societies to accelerate the adoption of the minimal set of practices (Chue Hong et al., 2019) for their communities, but also to develop field-specific guidelines on when and which software to cite within the limited room available for references. **The core of this recommendation is to help researchers decide on what and why to cite, not how to cite**. Several of the focus group participants said they were happy to cite software but would like guidelines from some set of credible scientific authorities. The ARDC thematic focus areas could assist with outreach to specific research societies and organizing statements from particular leaders with thematic areas.

2 Support the Effort to Create And Disseminate Software Bill of Materials (SBOMs)

Problem: Without visibility in publications, research software use will continue to be unacknowledged. Without acknowledgement, the work of building and maintaining software is difficult to sustain. Yet citation practices are governed by deeply held scientific norms and are limited in number, making it hard to acknowledge the large quantities of individual components used in research workflows. This particularly undermines visibility for infrastructural software (especially software used across disciplines), reducing incentives for its production and maintenance as well as undermining efforts to shape, improve and sustain software infrastructure.

Focus group outcomes: Participants' discussion of software citation revealed a tension. On the one hand participants emphatically endorsed the importance of citation for credit, linking clearly to the ability to sustain and improve software. Yet they also related episodes in which software was excluded from the limited space of reference lists, going on to acknowledge that attempting to include all relevant software would *"overwhelm"* that system and fail to meet scholarly norms.

After analysis, we developed a distinction between two purposes of software visibility in research papers. The first purpose is to explain choices relevant to understanding the research results of a paper (and thus to future research). The second purpose is to acknowledge the usefulness of software. These two purposes are overlapping: some software is both useful and relevant to explaining the intellectual choices of the research, while other software was useful (possibly crucial) but was not linked to understanding the intellectual challenge addressed in the research. In the first category participants tended to discuss software with *"domain-specific" or "intellectual content"* where lack of clarity on what tools were used can make or break attempts by others to extend the work, while in the second category, participants tended to place software doing *"general purpose"* tasks (like *"data wrangling")* or well understood analyses (like *"linear algebra"*) which can be accomplished in many ways through an array of interchangeable tools.

In grappling with the question of which pieces of software ought to be included in the limited space of the reference list, our understanding is that it was easier to argue for inclusion of software relevant to intellectual explanation than other software (although the more a piece of software was specific to a domain, or produced by members of that domain, the more inclusion in a limited reference list would meet norms.) Militating against inclusion in reference lists was that the sheer quantity of software used, most of which was not relevant to the intellectual explanation and was considered far from the specific domain of the paper authors and audience, was simply too great for useful visibility within scholarly papers. It seems reasonable that those that make software that intentionally cuts across fields, such as those in our Method Makers focus group, would be particularly likely to fall into this structural exclusion, and be less likely to be cited in formal reference lists.

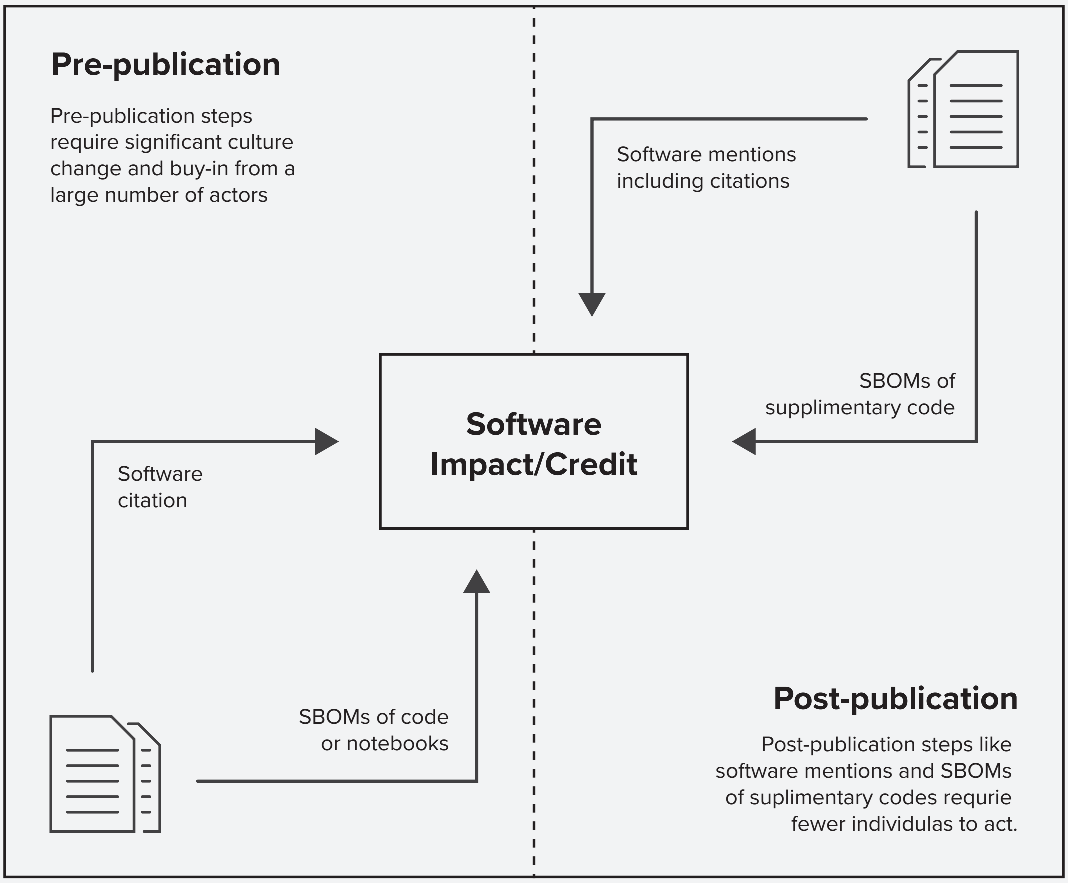
For software that falls into the second category — useful but not seen as important to the intellectual explanation of the specific paper — the situation is partially analogous to citation outside software: some concepts become so fundamental to fields that they are no longer directly referenced. Garfield (1977) called this "the obliteration phenomenon" in scholarly citation practices. Accordingly, unmentioned software becomes like the infrastructure, and the work of maintaining it becomes "invisible work" (Geiger et al., 2021; Star & Strauss, 1999; Lee et al., 2006). While participants wanted to give credit to the authors of all the software they used, the limits of space and the norm of intellectual explanation combined to make citations (or even mentions) less useful for visibility of software important to overall scholarship.

In contrast, publishing the entire research workflow, including all data and a full software stack, would document all software used in research and thus include both kinds of software discussed about. In addition to benefits to reproducibility, this would facilitate analysis to demonstrate impact and improve incentives to production and maintenance of infrastructural software. This recommendation has been made many times (De Roure et al., 2010; Stodden, 2010; Stodden et al., 2013; Strijkers et al., 2011).

Yet uptake is slow. We speculate that this is because of difficulties in sharing data, whether for privacy or competitive reasons, challenges in sharing proprietary code, sections of workflows being undertaken by hand (and not amenable to being scripted), the effort required to "clean up" code for distribution, and the perception of future support liabilities created by releasing code that others may misunderstand to be for general use (Howison & Herbsleb, 2011; Strijkers et al., 2011; Trainer et al., 2015).

Recommendation: Recent progress in creating a "software bill of materials" may offer a pathway forward. SBOMs are now required for software suppliers to the US Federal Government[6], highlighting the dependencies incorporated in delivered code, giving insights into likely pathways for security issues (and informing downstream users of what dependencies they should be looking at for security issues). Existing SBOM analysis tools[7, 8] can run against code without sharing the code itself; publications could require an SBOM to be submitted along with the publication. SBOMs can also be generated post hoc from publications with code supplementary materials. SBOMs offer acknowledgment of the usefulness of software infrastructure, and also the promise of supporting reproducibility efforts by documenting exact versions of software used in a research effort.

There are still several unresolved issues with the implementation of SBOMs for acknowledging the use of software. Some of these issues include building community consensus on a schema for the files, linking these files with publications via metadata, and demonstrating the benefits of generating additional artifacts for researchers. While this is still in the early stages of development, the ARDC could take a leading role in shaping the discourse.


Pre-publication
Pre-publication steps require significant culture change and buy-in from a large number of actors
Software mentions including citations
SBOMs of supplimentary code
Software Impact/Credit
Software citation
SBOMs of code or notebooks
Post-publication
Post-publication steps like software mentions and SBOMs of suplimentary codes requrie fewer individulas to act.

Comparing recommendations #1 and #2, citation versus SBOMs

Citations and reference lists are crucial venues for software visibility, but their limited size means they will always be contested spaces. Software bill of materials (SBOMs) are the opposite: they are unlimited in size, but sacrifice the value of specific mention within the flow of writing, which enhances reputations because it indicates that the authors and editors want to acknowledge that this software met the contributions norms in a way similar to other citations.

Change to pre-publication practices requires buy-in from a large number of different actors (researchers, journals, editors and so on), and achieving this level of culture change can take a lot of time. Improving software citation (recommendation #1) relies heavily on these changes, while the introduction of SBOMs (recommendation #2) relies less heavily on pre-publication changes.

Thus we view recommendation #1 as important, but with an impact over the longer term. Conversely, Recommendation #2 can be more readily implemented in the short term, by building SBOMs from software mention extractions and existing supplementary code (complementing the work envisioned for recommendation. #3: software use infrascope). This will enable rapid demonstration of value, and requires buy-in from fewer more centralized actors. In the longer term, the value of SBOMs for software visibility can be enhanced by changes to pre-publication activities, such as generating SBOMs at the time of article submission or during the peer review process (a middle ground between software mentions and seeking publication of entire research workflows).

3 Create a Software Use Infrascope as an Observatory Based on Software Mentions in Publications to Highlight Particular Areas of Strength or Opportunity in Australia

Problem: The visibility of the software used by researchers is relatively low, and it is, therefore, difficult to know what software is crucial for particular research topics. If researchers, developers, and funders could more easily see trends in software use by research topic and by national origins, they would be able to focus support, training, and funding more effectively. This would aid the ARDC in its efforts to make software visible, as outlined in the ARDC's national agenda for research software.

Focus Group outcomes: Participants described approaches to identifying where their software was mentioned in the literature, making it clear that these were inadequate, laborious, and incomplete. For example, *"I tracked them and I put a link to a page where I tried to collect them, which is, most of them are not sort of formal citations … It's just me googling or … going into Google Scholar and searching for [title of my package]"*.

Recommendation: The ARDC should support infrastructure that enables a systematic understanding of the software being used by Australian researchers, broken down by specific research topics. We term this a "software use infrascope" because it enables inspecting and thus makes visible the software infrastructure that underlies research areas and the work of Australian researchers. As a microscope makes the microscopic visible, a software use infrascope would make software visible.

For any publication, it should ideally be possible to identify the research area, associated researchers and institutions (including their national connection), and software used within the research. Each of these is challenging but tractable. Research areas can be assessed through indices associating publication venues with areas, keywords provided by authors, or topic modeling using the full text of the papers. Researcher national connections can be assessed through lists of publications associated with nationals (such as the Excellence in Research for Australia evaluation rounds, or ERA), institutions listed in CrossRef metadata, or through parsing institutions from full-text and connecting those with lists of institutions within Australia, as has been recently done for France by the French Open Science Monitor (Bassinet et al., 2023). Software used in publications can be discovered using machine learning systems such as SoMeSci (Schindler et al., 2021), which includes trained models able to identify software packages mentioned in full text and distinguish between those merely mentioned and those actually used by researchers.

Together this dataset could be exposed through a web interface, which would enable stakeholders to execute queries such as "which software packages are used by Australians researching wildfires?" or "which Australian institutions have clusters of researchers using a certain software package, regardless of the research area"? These data would then be actionable for undertakings such as creating training programs that connect package developers with their users or creating local support communities for users of related packages. When data shows change over time, this may indicate opportunities to reach out to users of packages with declining use and seek to understand their reasons, potentially matching training or targeted funding to update local dependencies that make migration difficult. Research area focus could be assessed with varying levels of specificity, from individual fields or topical areas up to broad thematic emphasis areas, such as the ARDC thematic focus areas.

A straightforward extension would be to add the ability to identify software that was created by the authors of a paper, thus enabling queries such as, "which Australian researchers are contributing software that is gaining use within indigenous research?" Such insight would enable targeted funding support for software on the rise.

Combining the data on users of software with the creators of software would enable innovative research support approaches. For example, if the data showed that creators and users were nearby within Australian institutions, small funding efforts for collaboration-creating workshops could yield an outsized impact. Conversely, the data could show that while particular packages are widely used in strategically important research areas in Australia, there are no Australian researchers contributing to these packages, thus creating an opportunity to enhance synergy by encouraging Australian researchers to work with these packages (something that could be done through funding or recruitment).

Further, a software use infrascope can develop into a resource that assists in making the case for the impact of software packages. In our focus groups, we discussed many ad hoc efforts that developers were using to track where their packages were occurring in the literature.

> I don't use metrics provided by citations, because their use/their benefit to me is hypothetical at the moment. I don't have a platform that tracks that. I don't know how I can track that. Right. Yeah, maybe I'm missing something here. But I don't know how you track that.

Finally, a software use infrascope has synergies with other recommendations in this report. Recommendation #1 encourages clearer guidelines to which software ought to be formally cited due to its special significance to the research field; this can provide a new source of data about scientific contribution beyond use. Recommendation #2 encourages the creation and promotion of a research software bill of materials as a standardized online appendix, a data source that goes beyond the informal practices of software mentions to provide additional data on which software is used in research. Recommendation #4 encourages the ARDC to provide online community infrastructural support for the overlap of the research area and software used; the infrascope would provide guidance on which overlaps can provide high benefit.

Alternative approaches to identifying software use during research were discussed in the focus groups, such as instrumenting national supercomputing infrastructure to identify dependencies used in computing jobs. In discussion and on reflection, we think there are significant technical, social, and institutional challenges to implementing this approach (which was characterized as similar to spyware during discussion). In any case, this approach would not apply beyond very specific infrastructures. Therefore we do not recommend incorporating this approach.

4 Promote the Creation and Dissemination of Fully Worked Example Use Cases Aimed At Different Skill Levels

Problem: A common challenge for model makers, methods makers, researchers, and research software engineers is deciding which tools to use. There was considerable consensus on the steps that these individuals take when evaluating software for use. These include:

- evidence that the software is still being maintained
- mentions of the software in published peer-reviewed papers
- mentions in recent tutorials and other training materials and courses
- documentation (like README files and other long-form documentation) containing several examples that can be quickly run on small datasets.

In these discussions, a common need surfaced: the desire for available use cases to help consumers quickly evaluate if the tool is right for their needs. Several focus group participants noted that the lack of clear fully worked example use cases (with "real data") drove more advanced users to create their own tools, or novices to fall back on known but inefficient approaches.

Focus group outcomes: Even when software contained introductory documentation, some participants, especially the methods makers, noted that the presence/absence of fully worked-out examples was key to deciding whether or not to use such software. Bioconductor (Gentleman et al., 2004) was named as an exemplar community where long-form documentation (Vignettes) are reviewed as part of the software evaluation.

Recommendation: To increase the adoption of research software tools, software producers should be encouraged to provide use cases aimed at researchers with different skill levels from novice to expert in order to help researchers quickly evaluate and adopt relevant tools. Tool documentation should contain multiple fully worked example use cases, and these should be made available as separate units that can be linked to directly, potentially with separate digital object identifiers. The ability to directly link to example use cases will aid in their discoverability via search engines, which will then contribute to software discovery.

If the use cases are made available as part of software documentation (for example, vignettes in the R ecosystem) or on code collaboration platforms such as GitHub, they should ideally be included in the build process or built with continuous integration to ensure that the examples still work with the current version of the software. We offer two suggestions to achieve this outcome:

1. Educate research software producers in curating use cases and include this as part of training efforts, featuring and promoting best practice examples.
2. The ARDC, in partnership with funders or domain-specific groups, should encourage the creation of small grants similar to Google Summer of Code (GSOC)[9] or Google Season of Docs (GSOD)[10] to pair software creators with users to publish high-quality use cases. An added outcome of these small grants will be increased communication between users and developers. Recommendation #3 (software use infrascope) can be used to identify packages for this support.

5 Support Existing Technology Approaches to Software Archiving

Problem: Software developed by researchers is poorly archived (Collberg et al., 2015) and difficult to locate after a study has been completed.

Focus group outcomes: Many of the focus group participants we spoke to had not thought deeply about archiving. Several assumed that GitHub was a (sufficiently) permanent archive. While some acknowledged that future reductions in free services or accidental deletions could lead to permanent loss of research software, there was also inertia against doing more. Some believed that if GitHub were to disappear, there would be sufficient time to migrate to a future offering. Thus, discussion within the focus groups did not reveal clear motivations for the long-term archiving of software, which perhaps contributed to the assumption that commercial working repositories such as Github were sufficient. Some participants also expressed reluctance to do the additional work because similar efforts were not rewarded or recognized, saying *"This extra time is never compensated"*.

Recommendation: Since many excellent solutions already exist[11], the ARDC should not invent anything new in this space. Training and consulting services should promote the use of large-scale, automated, archives like Software Heritage and Zenodo and encourage the use of the GitHub to Zenodo integration, which automatically archives each new release. By relying on automation long-term archiving can occur, even while education efforts seek to increase the motivation of individual researchers for this work, the ARDC can ensure that more of Australia's software products are archived for the long term.

There are synergies here with other recommendations: The software use infrascope would create data on packages created and highly used in Australia which could be used to create a focal collection on Zenodo or Software Heritage.

6 Support Specialized Communities of Practice (Online and Offline)

Problem: Research software can be relatively complex, and research needs are frequently emergent and difficult to predict at the early stages of research projects. Together these factors mean that researchers need help with software discovery and with software use.

Focus Group Outcomes: Throughout the focus groups, participants highlighted the importance of two kinds of support venues.

The first was local venues for interaction and support, with many mentioning "hacky hours"[12] specifically. They made it clear that these venues create high trust, well-focused, environments for advice and help-seeking that were contrasted positively against public, general purpose, online software "question and answer" and support venues.

The second was online communities around research area-specific software ecosystems, established to bring together the online discussions around a group of related packages, and supported with staff time to activate and manage the community. Examples given included Images.sc (funded by the US National Institutes of Health) and the Rstudio/posit forums[13] around the R Tidyverse (Wickham et al., 2019). The Images.sc model, in particular, seems promising as a venue for reaching potential users and finding support for the complexities of research use of software. The frequently asked questions section for Images.sc describes enrolling projects as "community partners" with the requirement that the forums are then the primary advertised support venue for all those projects.[14]

Recommendations

#6.1 Continue support for local communities of practice at research institutions

The ARDC should continue its tradition of creating and supporting communities of practice, including across the three thematic focus areas, encouraging the creation and maintenance of local hybrid communities that incorporate researchers, research software engineers, and students.

#6.2 Fund community activators around research area-specific software ecosystems

The ARDC should support online communities of practice in specific research software areas, bringing together related software (either by technique or components used in workflows in specific fields). Crucial to this recommendation is funding the involvement of community managers (or activators) who act to create outreach, invitations, content, and engagement, thus seeding activity. A success measure for continued support should be a demonstration that the venues are primary discussion areas for research software in the specified field and that the seeded activity is increasingly matched and eventually exceeded by organic activity driven by researchers and research software engineers research software engineers in the area (see "Buzzing Communities" (Millington, 2012) and "The Art of Community"/"People Powered") (Bacon, 2019) With that success metric in place, the ARDC could provide hosted platforms (and engage with the US Center for Scientific Collaboration and Community Engagement "tools trials"[15]), or provide support to groups already providing platforms internationally; but the core of this recommendation is funding community managers and activators as human community infrastructure.

Both recommendations have synergies with identifying areas of strategic use or need through the software use infrascope (Recommendation #3) and the ARDC's existing work in creating and supporting research communities of practice.

7 Provide Guidance for Implementing Web Analytics to Understand Usage

Problem: Software projects need to be more visible, and they need reliable data to demonstrate their impact, and to understand their user communities. Software-specific metrics (such as downloads) are relatively specialized and problematic (Howison et al., 2015), and website analytics (data on visits to project-specific websites) may offer a useful alternative.

Focus group outcomes: In our focus groups, popularity measures such as download counts (from package managers), GitHub "stars", or installations via package managers, were discussed. Yet these comments were immediately followed by concerns over poor data quality undermining the credibility and thus usefulness of these measures. Specific issues discussed included the impossibility of separating bot-like installs such as those from continuous integration systems and package managers.

> If you can get to download stats, that's another thing [to add to citation, but] are they downloaded from some chatbot in Russia? … being downloaded might just mean that somebody got frustrated, couldn't get it to work.

Participants did, however, highlight their use of website metrics (such as Google Analytics or Matamo) to build their case for impact. Projects can then draw on the wealth of work on identifying and discounting non-user accesses to websites (for example, search bots and other scrapers). Indeed, website analytics applied to documentation, and fully worked use cases could open new avenues of insight between software developers and potential users. However, to realize value here, understanding the appropriate use of web metrics would be yet another skill that academic software developers would need to learn.

Recommendation: The ARDC should encourage Australian-based research software projects to make systematic use of web analytics platforms around project websites and documentation. This support could involve producing guides, training, and dashboards, and connecting experienced project users with those seeking to implement web analytics. The ARDC may also be able to identify existing discount contracts for which projects may be eligible, such as through Australian government contracts or hosted web analytic services and also include links to software produced by others that is frequently used to analyze this dataset..

8 Provide a Low-Friction Way for Researchers To Link to Software Alongside Data Submissions

Problem: Scientific societies have noted that data sharing efforts are falling short of their promise because the code and analysis scripts are not shared or made easily discoverable (Jenkins et al., 2023).

Focus Group Outcomes: Focus group participants highlighted the importance of software discovery, lamenting as all too frequent their discovery that researchers were not using software that could directly solve their problems. Nonetheless, echoing the results of the ARDC published report Understanding How Researchers Find Research Software for Research Practice (Stevens, 2022), participants did not point to software-specific catalogs (such as that provided by the Netherlands eScience Center's Research Software Directory[16]) as a promising avenue for improvement. In fact, software repositories came up only when highlighting the importance of services beyond simple listings, such as package management, code audit, version control, or collaboration tools. Much more enthusiasm was displayed by participants for fully worked example case studies "with real data" to support the ability to explore software and its capabilities through well-contextualized use.

Overall, participants were conscious of the additional unrewarded work involved in promoting software; our interpretation is that efforts to promote software are most likely to be perceived as worthwhile if they are a small amount of additional burden undertaken within another task (ideally one with its own clear motivation or resources).

Recommendation: Data catalogs should be enhanced with simple-to-create links to the software frequently used to work with the data. The data repository Dryad implemented changes to the submission workflow providing authors the opportunity to deposit any code associated with the datasets. The code and software assets are then deposited in Zenodo at the time of data archiving, producing a DOI that can then be linked with the deposited data[17]. This model could be adopted in Australian data repositories allowing data catalogs to surface more links to software. The ARDC could work with the data providers for the ARDC Research Data Australia platform, providing support to incorporate the submission of related software during dataset curation.

ACKNOWLEDGMENTS

We thank Dr Tom Honeyman and Dr Andrew Treloar for providing us with the opportunity to develop the study and carry out the focus groups. The recommendations we make in this report are a result of numerous conversations with various experts in the field.

Conflict Of Interest Statement

The authors of this report, Karthik Ram and James Howison have undertaken related funded work to create infrastructure for software visibility, resulting in work with the R-universe and Softcite projects mentioned in recommendation #3 (software use infrascope). In addition, we work closely with Patrice Lopez, author of the Grobid PDF extraction software, collaborator on the Softcite project, and paid participant in the creation of the French Open Science Monitor. We managed these conflicts in the focus groups by choosing not to present infrastructure possibilities and carefully analyzing discussion to highlight infrastructure opportunities beyond our existing work.

FEEDBACK

We welcome your feedback on this guide. Please email contact@ardc.edu.au with any comments or questions.

ABOUT THE AUSTRALIAN RESEARCH DATA COMMONS

The Australian Research Data Commons (ARDC) enables the Australian research community and industry access to nationally significant, data intensive digital research infrastructure, platforms, skills and collections of high quality data.

The ARDC is supported by the Australian Government through the National Collaborative Research Infrastructure Strategy (NCRIS).

APPENDIX A: AUTHOR BIOS

Dr. Karthik Ram

Dr. Karthik Ram is a Research Associate Professor at UC Berkeley's Institute for Data Science and the Berkeley Initiative for Global Change Biology. His research is interdisciplinary and focuses on various areas such as understanding the impacts of climate change on ecological communities, reproducible research, sustainable scientific software, and open science.

Karthik is also known for his involvement in several organizations and projects related to open science and scientific software development. He is the co-founder and director of The rOpenSci Project (https://ropensci.org/) and leads the US Research Sustainability Institute (https://urssi.us/). He has played a crucial role as founder and founding editor of the Journal of Open Source Software (https://joss.theoj.org/) and The rOpenSci Software Review. Moreover, Karthik is on the editorial boards of ReScience and Research Ideas and Outcomes.

Due to his extensive knowledge and experience in scientific software development, Karthik has been an advisor to several organizations, including the Research Software Alliance, Chan Zuckerberg Initiative's Essential Open Source Software for Science Program, and The UK Software Sustainability Institute.

Karthik has served as a consultant for infrastructure providers such as Data Dryad and for scientific organizations like the American Geophysical Union, National Science Foundation's DataONE, Code for Science and Society, and more. Karthik and James have run similar focus groups at the Chan Zuckerberg Essentials of Open Source Conference 2020 https://github.com/karthik/software-mapping-workshop. Karthik and James are also currently funded by the Gordon and Betty Moore Foundation's Data-Driven Discovery Program to build infrastructure for research software that incorporates software metrics (especially software mentions).

Dr. James Howison

Dr. James Howison is an Associate Professor in the Information School of the University of Texas at Austin, where he has been since August 2011. James studies open collaboration, particularly in software development, including open source software development and the development of software in science. His work has been supported by grants from the National Science Foundation (NSF), including a 2015 NSF CAREER award and a 2019 PECASE award, as well as the Sloan Foundation. James has published in the fields of information systems, computer-supported cooperative work, and information sciences (for example, MIS Quarterly, ACM CSCW Conference, and JASIST), as well as keynotes for industry and funding agency advisory events, including RSECon (the Research Software Engineer conference in the UK), AstroPy, Earth Sciences Information Partners (ESIP) and presenting on invited panels at the NSF Software Infrastructure for Sustained Innovation PI meeting and the Chan Zuckerberg Initiative Essential Open Source Software PI meetings.

James has published studies of scientists using and developing software, the evaluation of research software development projects, studies about practices of software citation, and developed a gold standard dataset of software mentions in the research literature that has formed the basis for groups (including us) to develop systems to automatically recognize software mentions in the literature. James has organized and participated in many workshops addressing questions in the production of scientific software, including a Dagstuhl workshop on engineering academic software, The "Working towards sustainable software for science: practice and experiences" (WSSSPE) series of workshops, workshops on software citation, and the US Research Software Sustainability Institute (URSSI) workshops.

REFERENCES

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

Bacon, J. (2019). People Powered: How Communities Can Supercharge Your Business, Brand, and Teams. HarperCollins Leadership. https://play.google.com/store/books/details?id=y1GNDwAAQBAJ

Barker, M., & Buchhorn, M. (2022). Research Software Capability in Australia. https://doi.org/10.5281/zenodo.6335998

Bassinet, A., Bracco, L., Jeangirard, E., Lopez, P., & Romary, L. (2023). Large-scale Machine-Learning analysis of scientific PDF for monitoring the production and the openness of research data and software in France. https://hal.science/hal-04121339

Baxter, R., Hong, N. C., Gorissen, D., Hetherington, J., & Todorov, I. (2012, September 10). The Research Software Engineer. Digital Research 2012. https://www.research.ed.ac.uk/files/65195747/DR2012_12_1_.pdf

Carver, J. C., Gesing, S., Katz, D. S., Ram, K., & Weber, N. (2018). Conceptualization of a US Research Software Sustainability Institute (URSSI). Computing in Science & Engineering, 20(3), 4–9. https://doi.org/10.1109/MCSE.2018.03221924

Chue Hong, N. P., Allen, A., Gonzalez-Beltran, A., de Waard, A., Smith, A. M., Robinson, C., Jones, C., Bouquin, D., Katz, D. S., Kennedy, D., Ryder, G., Hausman, J., Hwang, L., Jones, M. B., Harrison, M., Crosas, M., Wu, M., Löwe, P., Haines, R., … Pollard, T. (2019). Software Citation Checklist for Authors. https://doi.org/10.5281/zenodo.3479199

Collberg, C., Proebsting, T., & Warren, A. M. (2015). Repeatability and benefaction in computer systems research. University of Arizona TR, 14(4). http://reproducibility.cs.arizona.edu/v2/RepeatabilityTR.pdf

De Roure, D., Goble, C., Aleksejevs, S., Bechhofer, S., Bhagat, J., Cruickshank, D., Fisher, P., Hull, D., Michaelides, D., Newman, D., Procter, R., Lin, Y., & Poschen, M. (2010). Towards open science: the myExperiment approach. Concurrency and Computation: Practice & Experience, 22(17), 2335–2353. https://doi.org/10.1002/cpe.1601

Du, C., Cohoon, J., Lopez, P., & Howison, J. (2022). Understanding progress in software citation: a study of software citation in the CORD-19 corpus. PeerJ. Computer Science, 8, e1022. https://doi.org/10.7717/peerj-cs.1022

Geiger, R. S., Howard, D., & Irani, L. (2021). The Labor of Maintaining and Scaling Free and Open-Source Software Projects. Proc. ACM Hum.-Comput. Interact., 5(CSCW1), 1–28. https://doi.org/10.1145/3449249

Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M., Rossini, A. J., … Zhang, J. (2004). Bioconductor: open software development for computational biology and bioinformatics. Genome Biology, 5(10), R80. https://doi.org/10.1186/gb-2004-5-10-r80

Howison, J., Deelman, E., McLennan, M. J., Ferreira da Silva, R., & Herbsleb, J. D. (2015). Understanding the scientific software ecosystem and its impact: Current and future measures. Research Evaluation, 24(4), 454–470. https://doi.org/10.1093/reseval/rvv014

Howison, J., & Herbsleb, J. D. (2011). Scientific software production: incentives and collaboration. Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work, 513–522. https://doi.org/10.1145/1958824.1958904

Jenkins, G. B., Beckerman, A. P., Bellard, C., Benítez-López, A., Ellison, A. M., Foote, C. G., Hufton, A. L., Lashley, M. A., Lortie, C. J., Ma, Z., Moore, A. J., Narum, S. R., Nilsson, J., O'Boyle, B., Provete, D. B., Razgour, O., Rieseberg, L., Riginos, C., Santini, L., … Peres-Neto, P. R. (2023). Reproducibility in ecology and evolution: Minimum standards for data and code. Ecology and Evolution, 13(5), e9961. https://doi.org/10.1002/ece3.9961

REFERENCES

Katz, D. S., Choi, S.-C. T., Lapp, H., Maheshwari, K., Löffler, F., Turk, M., Hanwell, M. D., Wilkins-Diehr, N., Hetherington, J., Howison, J., Swenson, S., Allen, G. D., Elster, A. C., Berriman, B., & Venters, C. (2014). Summary of the First Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE1). In arXiv [cs.SE]. arXiv. http://arxiv.org/abs/1404.7414

Liffers, M., & Honeyman, T. (2021). ARDC Guide to making software citable. https://doi.org/10.5281/zenodo.5003989

Millington, R. (2012). Buzzing communities: How to build bigger, better, and more active online communities. BookBaby.

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

Salmon, M., & Ram, K. (2021). The R developer community does have a strong software engineering culture. The R Journal, 13(2), 673. https://doi.org/10.32614/rj-2021-110

Smith, A. M., Niemeyer, K. E., Katz, D. S., Barba, L. A., Githinji, G., Gymrek, M., Huff, K. D., Madan, C. R., Mayes, A. C., Moerman, K. M., Prins, P., Ram, K., Rokem, A., Teal, T. K., Guimera, R. V., & Vanderplas, J. T. (2018). Journal of Open Source Software (JOSS): design and first-year review. PeerJ Preprints, 4, e147. https://doi.org/10.7717/peerj-cs.147

Star, S. L., & Strauss, A. (1999). Layers of silence, arenas of voice: The ecology of visible and invisible work. Computer Supported Cooperative Work: CSCW: An International Journal. https://doi.org/10.1023/A:1008651105359

Stevens, F. (2022). Understanding how researchers find research software for research practice. https://doi.org/10.5281/zenodo.7340034

Stodden, V. (2010). The Scientific Method in Practice: Reproducibility in the Computational Sciences. https://doi.org/10.2139/ssrn.1550193

Stodden, V., Guo, P., & Ma, Z. (2013). Toward Reproducible Computational Research: An Empirical Analysis of Data and Code Policy Adoption by Journals. PloS One, 8(6), e67111. https://doi.org/10.1371/journal.pone.0067111

Strijkers, R., Cushing, R., Vasyunin, D., de Laat, C., Belloum, A. S. Z., & Meijer, R. (2011). Toward executable scientific publications. Procedia Computer Science, 4, 707–715. https://doi.org/10.1016/j.procs.2011.04.074

Trainer, E. H., Chaihirunkarn, C., Kalyanasundaram, A., & Herbsleb, J. D. (2015). From Personal Tool to Community Resource: What's the Extra Work and Who Will Do It? Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing, 417–430. https://doi.org/10.1145/2675133.2675172

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T., Miller, E., Bache, S., Müller, K., Ooms, J., Robinson, D., Seidel, D., Spinu, V., … Yutani, H. (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686. https://doi.org/10.21105/joss.01686

Wilson, G. (2014). Software Carpentry: lessons learned. F1000Research, 3, 62. https://doi.org/10.12688/f1000research.3-62.v2

Crouch, S., Chue Hong, N., Hettrick, S., Jackson, M., Pawlik, A., Sufi, S., Carr, L., De Roure, D., Goble, C., & Parsons, M. (2013) "The Software Sustainability Institute: Changing Research Software Attitudes and Practices," Computing in Science & Engineering, 15(6) 74–80, https://doi.org/10.1109/MCSE.2013.133

Lee, C. P., Dourish, P., & Mark, G. (2006). The human infrastructure of cyberinfrastructure. Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work, 483–492. https://doi.org/10.1145/1180875.1180950

Schindler, D., Bensmann, F., Dietze, S., & Krüger, F. (2021). SoMeSci—Software Mentions in Science [dataset]. Zenodo. https://doi.org/10.5281/zenodo.4968738

ENDNOTES

1 https://upstream.force11.org/the-research-software-alliance-resa/

2 https://pypi.org/

3 https://www.bioconductor.org/

4 https://cran.r-project.org/

5 https://ardc.edu.au/program/research-software-program/

6 https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

7 https://spectralops.io/blog/top-10-sbom-tools-in-2023/

8 https://jenkins-x.io/blog/2022/07/24/sbom-tools/

9 https://summerofcode.withgoogle.com/

10 https://developers.google.com/season-of-docs

11 https://www.softwareheritage.org/save-and-reference-research-software/

12 https://hackyhour.github.io/

13 https://community.rstudio.com/

14 https://forum.image.sc/t/frequently-asked-questions/18729

15 https://www.cscce.org/2023/07/20/investigating-open-source-community-platforms-cscce-tools-trials-return

16 https://www.esciencecenter.nl/research-software-directory/

17 https://blog.datadryad.org/2023/07/14/leaders-in-open-data-journal-editors-boost-standards-for-data-sharing/

ARDC

Australian Research Data Commons

CONTACT

ardc.edu.au

+61 3 9902 0585

contact@ardc.edu.au

FOLLOW

@ardc_au

australian-research-data-commons

ARDC_AU

Subscribe to the ARDC Connect Newsletter
https://ardc.edu.au/subscribe/

NCRIS
National Research
Infrastructure for Australia
An Australian Government Initiative

The ARDC
is enabled
by NCRIS