Edinburgh Research Explorer

Software Citation in Theory and Practice

Software Citation in Theory and Practice

Daniel S. Katz[1] and Neil P. Chue Hong[2]

[1] University of Illinois Urbana-Champaign, USA
d.katz@ieee.org,
[2] University of Edinburgh, UK
n.chuehong@epcc.ed.ac.uk,
https://www.software.ac.uk/

Abstract. In most fields, computational models and data analysis have become a significant part of how research is performed, in addition to the more traditional theory and experiment. Mathematics is no exception to this trend. While the system of publication and credit for theory and experiment (journals and books, often monographs) has developed and has become an expected part of the culture, how research is shared and how candidates for hiring, promotion are evaluated, software (and data) do not have the same history. A group working as part of the FORCE11 community developed a set of principles for software citation that fit software into the journal citation system, allow software to be published and then cited, and there are now over 50,000 DOIs that have been issued for software. However, some challenges remain, including: promoting the idea of software citation to developers and users; collaborating with publishers to ensure that systems collect and retain required metadata; ensuring that the rest of the scholarly infrastructure, particularly indexing sites, include software; working with communities so that software efforts count; and understanding how best to cite software that has not been published.

Keywords: software citation, credit, software identifiers, software metadata, software repositories, bibliometrics

1 Introduction

In most fields, computational models and data analysis have become a significant part of how research is performed, in addition to the more traditional theory and experiment. Evidence of the increased role and importance of software in today's research can be found in surveys and in papers, and while neither of these are specific to mathematics, it is likely no exception.

Two recent surveys, one of UK academics at Russell Group Universities [9, 10], and one of members of (US) National Postdoctoral Research Association [15, 14] asked researchers asked how important software is to them, and found that 67% / 63% (UK/US respectively) of respondents said, "my research would not be possible without software." 21% / 31% said, "my research would be possible but harder," while just 10% / 6% said, "it would make no difference." A similar survey of mathematicians would be welcome.

One of the authors of this paper scanned six months of *Science* in mid-2013, and found that about half the papers were software-intensive projects, and most of the other papers also relied on some software. A formal study of 90 randomly selected papers in the biology literature in 2015 found that 80% mentioned software, and that those articles mentioned an average of 4.85 software packages [11]. A more recent study of *Nature* in Jan–Mar 2017 found software mentioned in 32 of 40 research articles, with an average of 6.5 software packages mentioned per article [16]. A similar study could be done of the mathematics literature. And while these studies have been manually performed by humans, natural language processing and machine learning could be used to expand their reach.

The system of publication and credit for theory and experiment (journals and books, often monographs) has developed and has become an expected part of the culture, how research is shared and how candidates for hiring, promotion are evaluated; software (and data) do not have the same history. In order to cite software, we could overload the current citation system to add software or alternatively, we could develop a new citation system that works for all kinds of products. As developing a new citation system would be very difficult, current efforts related to software citation have focused on the overloading approach.

2   Software Citation Principles

FORCE11[3] is a community of scholars, librarians, archivists, publishers and research funders that has arisen organically to help facilitate the change toward improved knowledge creation and sharing. In 2015 and 2016, a FORCE11 Software Citation working group developed a set of software citation principles [19]. The group grew to about 60 members, including researchers, developers, publishers, repository developer and maintainers, and librarians.

The group worked on GitHub[4] and on the FORCE11 web site[5]. It reviewed existing community practices and developed a set of use cases for software citation, and then drafted a software citation principles document. To do this, the group started with previously published data citation principles [5], updated them based on software use cases and related work, and further updated them based on working group discussions. This draft was then subjected to community feedback and review through a variety of channels, including a workshop at FORCE2016 in April 2016. In late 2016, the paper and its reviews were published [19]. The paper includes a set of six principles (general statements), use cases (where the principles should apply), and discussion (suggestions on how to apply the principles).

---

[3] `https://www.force11.org`

[4] `https://github.com/force11/force11-scwg`

[5] `https://www.force11.org/group/software-citation-working-group`

The software citation principles, quoting from [19], are:

1. **Importance**. Software should be considered a legitimate and citable product of research. Software citations should be accorded the same importance in the scholarly record as citations of other research products, such as publications and data; they should be included in the metadata of the citing work, for example in the reference list of a journal article, and should not be omitted or separated. Software should be cited on the same basis as any other research product such as a paper or a book, that is, authors should cite the appropriate set of software products just as they cite the appropriate set of papers.

2. **Credit and Attribution**. Software citations should facilitate giving scholarly credit and normative, legal attribution to all contributors to the software, recognizing that a single style or mechanism of attribution may not be applicable to all software.

3. **Unique Identification**. A software citation should include a method for identification that is machine actionable, globally unique, interoperable, and recognized by at least a community of the corresponding domain experts, and preferably by general public researchers.

4. **Persistence**. Unique identifiers and metadata describing the software and its disposition should persist – even beyond the lifespan of the software they describe.

5. **Accessibility**. Software citations should facilitate access to the software itself and to its associated metadata, documentation, data, and other materials necessary for both humans and machines to make informed use of the referenced software.

6. **Specificity**. Software citations should facilitate identification of, and access to, the specific version of software that was used. Software identification should be as specific as necessary, such as using version numbers, revision numbers, or variants such as platforms.

There are now over 50,000 DOIs that have been issued for software, and more than 60% of them have been issued since the FORCE11 group published the first preprint of the principles paper [20].

3   Practices and Examples

In practice, the adoption of software citation depends on developing community guidelines that implement the software citation principles within the context of existing community scholarly communication and software development norms.

For some commonly used commercial software, there are mandatory citations, e.g. as specified by SAS [17] or Matlab [4]. In other cases, authors of research software may provide a recommended general citation referring to suite of related software, e.g. the HSL Mathematical Software Library [18]. However, in many of these cases, the citations do not provide enough information to allow crediting

of the software authors (Principle 2), a machine actionable unique identifier (Principle 3) and persistent identifiers and metadata (Principle 4) or – in the case of HSL – an understanding of which version of the software was used (Principle 6).

Examples of mandatory and general software citations that do not fully implement the Software Citation Principles:

- The output for this paper was generated using SAS/STAT software, Version 14.1 of the SAS System for Unix. Copyright ©2018 SAS Institute Inc. SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc., Cary, NC, USA.
- MATLAB and Statistics Toolbox Release 2012b, The MathWorks, Inc., Natick, Massachusetts, United States.
- HSL. A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk/

Some software frameworks and platforms provide clear guidance on how to support particular versions or a specific citation for a package (Principle 6), e.g., by using the `citation()` function for R packages or the instructions for citing the GAP system for computational discrete algebra [23]. However these still do not provide persistent, machine actionable identifiers.

Examples of citations of specific packages as recommended by the software platform they are distributed with that mostly implement the principles:

- Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., Hornik, K. (2018). cluster: Cluster Analysis Basics and Extensions. R package version 2.0.7-1.
- Emma J. Moore, Christopher D. Wensley, groupoids - a GAP package, 1.54, 29/11/2017, https://gap-packages.github.io/groupoids/

However most software used in research does not provide guidance on how to cite it properly. If the software's website, or a CITATION file or README file with the source code, specifies how to cite the software, the author should use this information; this might be a reference to a software paper, or other publication. If the source code includes a codemeta.json [12] or Citation File Format (CFF) [7] file, the metadata in these files can be used with appropriate tooling to generate a citation automatically. Otherwise, the following guidance will help to construct a citation that implements the principles:

- For the authors, try to include all contributors to the software or, if this is not clear, name the project as the author. This may encourage some projects to make citation metadata available, including listing the authors.

- Include the name of the software, along with specific version/release information.
- Try to include a method for identification that is machine actionable, globally unique and interoperable. This ideally is a DOI but if there is no DOI, a URL pointing to a specific release might be the next best option.
- If there is a landing page that includes metadata, point to that, not directly to the software. Where you have the choice of pointing to a URL for general landing page including metadata, versus a specific URL (e.g. to a tag of a version) which does not contain sufficient metadata it is preferred to use the URL for the general landing page as the identifier, and clearly state the version.

Examples of citations for software using the suggested guidelines:

- Voevodsky, Vladimir and Ahrens, Benedikt and Grayson, Daniel and others. UniMath — a computer-checked library of univalent mathematics. https://github.com/UniMath/UniMath [accessed 2018-04-27]
- Eigen Project. (2017). Eigen [software] version 3.3.4 Available from https://bitbucket.org/eigen/eigen/ [accessed 2018-04-27]

For developers of a piece of software, there are several things that can be done to make it easier for others to cite the software. At a minimum, the code should be published using a clear version number and license. If the code is in GitHub, the developer can make it easily citable using Github's integration with Zenodo [8]. Alternatively, the developer can manually deposit it in a digital repository such as Zenodo or Figshare – supplying metadata including the authors, title and version – and being provided with a Digital Object Identifier (DOI) and often a recommended citation that adheres to the Software Citation Principles. This information can be used to insert the citation that others should use into the software documentation, preferably as a CITATION file.

Example of a citation generated by Zenodo that implements the principles:

- Vince Knight, & Ria Baldevia. (2018, January 31). drvinceknight/Nashpy: v0.0.13 (Version v0.0.13). Zenodo. http://doi.org/10.5281/zenodo.1163694

Of course, the fact that swMath [21] exists means that citation should be integrated with it, providing suggested citations for software in it, and using it to track and understand citations of math software.

4   Challenges

In May 2017, the FORCE11 Software Citation Working Group ended, and a new Software Citation Implementation Working Group[6] started. This group has the goal of moving the software citation principles to implementation. Those interested in following the new group can join it.

Many challenges remain, including:

– **Encouraging citation of software by authors**. Data citation is still not commonplace in many disciplines, let alone software citation. Author guidance for software citation is varied in the mathematical sciences. Both the Journal of Mathematical and Computer Simulation [22] and Journal of Statistical Software [13] provide guidance that follows the Software Citation Principles, but others - including the International Congress on Mathematical Software - do not. This will require the community to work with journals, conferences, and publishers to implement the Software Citation Principles in a way that they can be adopted by researchers in the area, similar to efforts in astronomy [2]. Tools such as CiteAs [1] may also help.
– **Promoting the idea of software citation to developers**. The benefits of making software more easily citable are not always obvious. The time taken to submit metadata can be reduced by the use of formats such as CodeMeta [12] and Citation File Format [7], particularly as they are adopted by repositories [3] and citation tools.
– **Citing unpublished software**. When authors do not publish their software, there is no archival link a citer can point to. The in-progress work to build a software archive for all source code by Software Heritage [6] may solve this problem.
– **Ensuring quality of information**. Even when information is provided, it may be discarded in the publication process. Collaboration with publishers, funders, and the identifier and citation infrastructure will be required to ensure that systems collect and retain required metadata, making it easier to discover and reuse software.
– **Giving credit for software through citation**. Ultimately, software citation will become widely practiced when the rest of the scholarly infrastructure, particularly indexing sites, includes software, and research communities recognize the value of software as a research output, thus providing an incentive for developers and authors to publish and reuse research software.

5   Conclusions

Although software citation is currently not standardized nor widely practiced, the publication of the Software Citation Principles has acted as a foundation on which to build community guidelines and improved tooling and infrastructure to support citation. The FORCE11 Software Citation Implementation Working

---

[6] `https://www.force11.org/group/software-citation-implementation-working-group`

Group is taking forward work to address the challenges standing in the way of software citation, and looks to the mathematical sciences community to work towards implementing the principles in the future.

References

1. Citeas, `http://citeas.org/`, accessed: 2018-04-27
2. American Astronomical Society: Citing repositories in AAS journals (AJ/ApJ) (2018), `https://github.com/AASJournals/Tutorials/blob/master/Repositories/CitingRepositories.md`, accessed: 2018-04-27
3. Caltech Library: Enhanced software preservation now available in CaltechDATA! (2018), `https://www.library.caltech.edu/news/enhanced-software-preservation-now-available-caltechdata`, accessed: 2018-04-27
4. Croucher, M.: How to cite MATLAB in research papers (2013), `http://www.walkingrandomly.com/?p=4767`, accessed: 2018-04-27
5. Data Citation Synthesis Group: Joint declaration of data citation principles (2014). https://doi.org/10.25490/a97f-egyk, Martone, M. (ed), FORCE11, San Diego, CA
6. Di Cosmo, R., Zacchiroli, S.: Software Heritage: Why and how to preserve software source code. In: iPRES 2017: 14th International Conference on Digital Preservation. Kyoto, Japan (2017), `https://hal.archives-ouvertes.fr/hal-01590958https://hal.archives-ouvertes.fr/hal-01590958/file/ipres-2017-software-heritage.pdf`
7. Druskat, S.: Citation file format (CFF) (2017). https://doi.org/10.5281/zenodo.1003150, `https://github.com/sdruskat/citation-file-format`
8. GitHub: Making your code citable (2018), `https://guides.github.com/activities/citable-code/`, accessed: 2018-04-27
9. Hettrick, S.: It's impossible to conduct research without software, say 7 out of 10 UK researchers (2014), `http://bit.ly/2B8y6Iz`
10. Hettrick, S., Antonioletti, M., Carr, L., Chue Hong, N., Crouch, S., De Roure, D., Emsley, I., Goble, C., Hay, A., Inupakutika, D., Jackson, M., Nenadic, A., Parkinson, T., Parsons, M.I., Pawlik, A., Peru, G., Proeme, A., Robinson, J., Sufi, S.: UK research software survey 2014 (Dec 2014). https://doi.org/10.5281/zenodo.14809
11. Howison, J., Bullard, J.: Software in the scientific literature: Problems with seeing, finding, and using software mentioned in the biology literature. Journal of the Association for Information Science and Technology **67**(9), 2137–2155 (2016). https://doi.org/10.1002/asi.23538
12. Jones, M.B., Boettiger, C., Mayes, A.C., Smith, A., Slaughter, P., Niemeyer, K., Gil, Y., Fenner, M., Nowak, K., Hahnel, M., Coy, L., Allen, A., Crosas, M., Sands, A., Chue Hong, N., Cruse, P., Katz, D.S., Goble, C.: CodeMeta: an exchange schema for software metadata. version 2.0. (2017). https://doi.org/10.5063/schema/codemeta-2.0
13. Journal of Statistical Software: Journal of statistical software style guide, `https://www.jstatsoft.org/pages/view/style`, accessed: 2018-04-27
14. Nangia, U., Katz, D.S.: Survey of National Postdoctoral Association - dataset (Aug 2017). https://doi.org/10.5281/zenodo.843607
15. Nangia, U., Katz, D.S.: Track 1 Paper: Surveying the U.S. National Postdoctoral Association Regarding Software Use and Training in Research. figshare (8 2017). https://doi.org/10.6084/m9.figshare.5328442.v3

16. Nangia, U., Katz, D.S.: Understanding software in research: Initial results from examining Nature and a call for collaboration. In: Proceedings of the 13th IEEE International Conference on eScience (eScience 2017) (2017). https://doi.org/10.1109/eScience.2017.78

17. SAS Institute Inc.: Referencing data analysis performed with SAS software (2015), `https://www.sas.com/en_us/legal/editorial-guidelines.html`, accessed: 2018-04-27

18. Science & Technology Facilities Council: HSL. a collection of Fortran codes for large scale scientific computation, `http://www.hsl.rl.ac.uk/catalogue/`, accessed: 2018-04-27

19. Smith, A.M., Katz, D.S., Niemeyer, K.E., FORCE11 Software Citation Working Group: Software citation principles. PeerJ Computer Science **2**, e86 (Sep 2016). https://doi.org/10.7717/peerj-cs.86

20. Smith, A.M., Katz, D.S., Niemeyer, K.E., FORCE11 Software Citation Working Group: Software citation principles. PeerJ Preprints **4**, e2169v1 (Jun 2016). https://doi.org/10.7287/peerj.preprints.2169v1

21. swMath: swMATH: An information service for mathematical software, `http://www.swmath.org`, accessed: 2018-04-30

22. Taylor & Francis: Taylor & Francis standard reference style — NLM, `https://www.tandf.co.uk//journals/authors/style/reference/tf_NLM.pdf`, accessed: 2018-04-27

23. The GAP Group: How to cite GAP (2018), `https://www.gap-system.org/Contacts/cite.html`, accessed: 2018-04-27