ALFRED P. SLOAN FOUNDATION

www.sloan.org | proposal guidelines

PROPOSAL COVER SHEET

Project Information

Principal Investigator
James Howison, Assistant Professor
UTA 5.404
1616 Guadalupe St Austin TX 78722
(315) 395 4056
jhowison@ischool.utexas.edu

Grantee Organization: University of Texas at Austin

Amount Requested: 635,261

Requested Start Date: 1 October 2016

Requested End Date: 30 September 2018

Project URL (if any):

Project Goal

Our goal is to improve software in scholarship (science, engineering, and the humanities) by raising the visibility of software work as a contribution in the literature, thus improving incentives for software work in scholarship.

Objectives

We seek support for a three year program to develop a manually coded gold-standard dataset of software mentions, build a machine learning system able to recognize software in the literature, create a dataset of software in publications using that system, build prototypes that demonstrate the potential usefulness of such data, and study these prototypes in use to identify the socio-technical barriers to full-scale, sustainable, implementations. The three prototypes are: CiteSuggest to analyze submitted text or code and make recommendations for normalized citations using the software author's preferred citation, CiteMeAs to help software producers make clear request for their preferred citations, and Software Impactstory to help software authors demonstrate the scholarly impact of their software in the literature.

Proposed Activities

Manual content analysis of publications to discover software mentions, developing machine-learning system to automate mention discovery, developing prototypes of systems, conducting summative socio-technical evaluations (including stakeholder interviews).

Expected Products

Published gold standard dataset of software mentions. Open source machine learning system for recognizing software mentions. Published dataset of software mentions from PubMed Open Access, SSRN, RePEC and ADS literature corpora. Open source prototypes of CiteSuggest, CiteMeAs, and Software Impactstory. Socio-technical roadmaps for their implementation as sustainable systems.

Expected Outcomes

Datasets for insight by scholars and science policy-makers; Improved citation of software; Improved resources for scholars to demonstrate the impact of their software work..

Improving the visibility of scholarly software work

1. What is the main problem and why is it important?

When preparing publications, scholars too rarely cite the software they used in their research. For this reason software, and work on software, is relatively invisible as a form of scientific contribution. This lack of visibility reduces the incentive to share and support scholarly software. Without adequate incentives, academia underinvests in software work, and the end result is that the software used by scientists, engineers, and humanists is not as good as it could be and the effectiveness of scholarship is thus undermined.

Scholars too rarely cite the software they used in their research. Researchers who build software have long pointed out that their contributions do not appear in publications (Katz, 2013; Katz et al., 2014; Stodden, 2010; Stodden, Guo, & Ma, 2013). Empirical research confirms this issue. For example, Howison and Herbsleb (2011) examined the work leading to three high-quality papers by interviewing the authors about the software used, then interviewing the authors of that software and outward to all dependencies; very few of those packages (and none of their dependencies) were actually mentioned in the original papers. Interviews with scientists make clear that they feel that their software contributions are not visible in the scientific literature, an area that counts most for the reputations of scientists. One informant laughingly estimated that "less than 10%" of use results in actual citations (Howison, Deelman, McLennan, Silva, & Herbsleb, 2015). Sloan supported research has found similar issues with data and data citation (Borgman, 2007; Edwards, Mayernik, Batcheller, Bowker, & Borgman, 2011).

Even when software use is mentioned in articles, those mentions are too often informal. Howison and Bullard (2015), attached as Appendix F, examined 90 randomly selected biology articles and

manually examined them for mentions of any kind, formal citations or informal mentions. They found that software is mentioned informally more frequently than it is cited formally. In fact, only 37% of mentions involved formal citations (either to domain papers or to "software papers" written to describe software). The remaining 63% of mentions were informal, such as just mentioning project names in the full-text, project URLs in footnotes, and citations to non-standard publications such as user manuals and project websites.

Software, and work on software, is relatively invisible as a form of scholarly contribution. The absence of software mentions in publications, and the informality of mentions when they do occur, means that systems that measure impact through bibliometrics based on formal citations, such as Google Scholar, Scopus, and Web of Science don't help scholars that contribute via software make their case for scholarly contribution and impact. Perhaps even more importantly, qualitative stories of impact, of great value when making the case for support to funders, are hard for software-contributing scholars to find and report. For example, NanoHub developed a procedure for identifying and categorizing articles that mentioned their software, removing false positives, and categorizing the extent of use and type of impact (the full protocol is described in Howison et al, 2015). The system, while functional and innovative, was time-consuming, involving content analysis by multiple undergraduate students and review by post-docs and PIs.

This lack of visibility reduces the incentive to share and support scholarly software. Scholarly software is rarely shared openly (Stodden 2010; Ince, 2012). While scholars hold legitimate competitive concerns, research demonstrates that they perceive the substantial work implied by sharing, and see the rewards to be insufficient and therefore have trouble prioritizing the work needed to realize the potential for software in scholarship. Sharing software is undoubtedly extra work; from adapting the software for use outside individual computers, to

documenting the limits of the code, managing user requests, to encouraging and testing code contributions (Trainer, Chaihirunkarn, Kalyanasundaram, & Herbsleb, 2015). Encouragingly, Trainer et al. also found evidence that scientists are interested in sharing, but wary of extra, unrewarded, effort. Their interviewees did not suggest that they resented providing support (in fact they felt an obligation to provide support). Rather they feared being unable to do support well, given other demands on their time, and that releasing their code might therefore slow down the work of other scientists. Thus, the issue is not that scientists, or other scholars, are selfish per se; indeed scientists are keen to share and to realize communitarian values of science (Stodden, 2010). Rather the issue is that realizing the benefits of sharing requires more than simply uploading code. If science, engineering, and the humanities want well-supported software, then they must provide incentives and rewards to those undertaking this work. We argue that the best incentive is to acknowledge software work as a scholarly contribution and to do so through citations in the scholarly literature.

Without adequate incentives, academia underinvests in software work, and the end result is that scholarly software is not as good as it could be. All is not well with software, even as software grows rapidly in importance across scholarship (Atkins, 2003; Joppa et al., 2013). Users frequently find it frustrating and poorly documented (Joppa et al., 2013; Katz et al., 2014). Software is often written using monolithic architectures, rather than adopting modular architectures (Boisvert & Tang, 2001). Software has been a source of concern in many fields, from retractions in biomedicine (Miller, 2006), to the identification of errors in important results in economics (the Reinhart-Rogoff Excel formula issue), and recent concerns over the validity of fMRI studies (Eklund, Nichols, & Knutsson, 2016). Even software deposited in repositories has not fared well over time: the Journal of Money, Banking, and Finance found less than 10% of

their repository was reproducible, in part because no maintenance had been done to keep software up to date (McCullough, McGeary, & Harrison, 2006). Researchers have found that sharing and studying software code is crucial for replication and understanding, over and above abstract descriptions of algorithms (Ince, Hatton, & Graham-Cumming, 2012).

Beyond concerns over specific pieces of software there are concerns over the software practices of scholars, including scientist's ability to manage versions, as shown in log files of efforts showing difficulties in recovering software and data released from the Climate Research Unit in the UK. Even if the results produced are accurate, the devaluation of software work and processes costs significant time and threatens the credibility of scholarly work. Perhaps more insidiously, evidence from the "Workshop on Sustainable Software for Science: Practice and Experiences" (WSSSPE) series of workshops makes clear that scientists do not do a good job of developing software in open communities, as is common in non-scientific open source software world (Katz et al., 2014, 2016). The work of monitoring downstream and upstream dependencies, encouraging outside contributions, timing and managing releases is substantial (Bietz, Baumer, & Lee, 2010; Trainer, Chaihirunkarn, Kalyanasundaram, & Herbsleb, 2015). If building quality software is not acknowledged as a scholarly contribution then the even more removed work of building software communities will be even more poorly rewarded and thus motivated.

In summary, we argue that it is problematic that software is so rarely visible in the scholarly literature and in the reputation systems built on that literature. We identify the literature (journal articles, conference papers, and pre-print papers) as crucial because the literature is acknowledged as the most important representation of contribution (Merton, 1988). If software work is to be seen as an equally valid form of scholarly contribution, then it ought to be visible in

the manner that other scholarly contributions are made visible and accounted for: through citations in articles. Moreover, addressing the visibility of software work in the literature allows us to tap into the existing institutional valuation of citations, rather than attempt to build a parallel accounting system for software contributions.

Thus we seek to address the problem of the invisibility of software in the scholarly literature, in order to alter incentives and, ultimately, to improve the software available to science, engineering, and the humanities.

2. What is the major related work in this field?

Most previous work attempting to raise the visibility of software work have been focused outside the scholarly literature. Many systems examine and report on code dependencies (http://scisoft-net-map.isri.cmu.edu/, http://depsy.org, http://libraries.io, https://www.versioneye.com). Some repository-specific tools have used download statistics to rank packages or have provided a place for users to rate and comment (http://pypi-ranking.info/alltime, http://www.crantastic.org/, http://ascl.net). Unfortunately, these approaches miss software and scripts that aren't released as formal packages (common in scholarly work) and they bypass the major mechanism of scholarly communication, the research paper.

It is challenging to build incentives based on software use in the scholarly literature because of there has been no standardized citation practice to date, so software citation use is invisible to existing citation tools and incentive structures. There are two approaches to improving this situation: prospective and retrospective. The prospective approach seeks to improve software citation practices in the future by designing standardized ways to cite software, building tools to make it easy to cite software, and working to drive use and uptake among scholars and publishers. This will make software references in papers published in the future visible to

existing citation tools. The retrospective approach takes current and historical practices as given and seeks to mine the literature to make the best use of those mentions of software that are present. With this approach we can. In this section we detail this major related work, before turning to our approach, which combines the retrospective and the prospective.

The earliest efforts have favored prospective approaches. Language-specific formats like Python's DueCredit (Halchenko & Matteo Visconti di Oleggio Castello, 2016) and R's citation() function (Wickham, n.d.) embed authors' preferred citation formats directly into source code. Others have proposed language-agnostic metadata formats to do the same thing, including GitHub CITATION files (Wilson, 2013) and the promising CodeMeta standard (CodeMeta Collaboration, n.d.).

The FORCE11 Working Group on Software Citation (https://www.force11.org/group/software-citation-working-group) is leveraging the experience of the FORCE11 organization in addressing data citation. They are defining a set of metadata elements that ought to be in ideal software citations and bringing together stakeholders including authors, style guide writers, scholarly societies, citation software producers, and publishers to design and implement new standardized citation formats and guidelines. Similarly, other players are working to provide new paths forward for software citation: for example, Mozilla Science, Github, and Zenodo are providing a way to archive a software release and obtain a DOI to reference it. Publication venues like the Journal of Open Research Software, the Journal of Open Source Software are providing new ways to obtain a citation target for software contributions, and some well established journals like ACM Transactions on Mathematical Software are providing peer reviewed publications of software itself. The Astrophysics Source Code Library (ASCL) is another important demonstration of the prospective approach. ASDL provides "landing pages," identifiers, and

suggested citations for astrophysics software and is working with the Astrophysics Data System to improve indexing of software citations in the Astrophysics literature (Allen et al., 2015). We believe that these prospective efforts are important; indeed the PIs in this grant participate in these efforts and are working toward their success.

In contrast, the retrospective approach seeks to leverage existing practice, analyzing the literature to identify mentions of software use and operationalize these mentions as incentives. A retrospective approach is possible, as mentioned above, because while formal software citation practices are problematic, software use is informally mentioned in the literature. Authors mention project names in the full-text, place project URLs in footnotes, cite user manuals and project websites, and more (Howison & Bullard, 2015).

If we can automatically detect these informal mentions, in the same way that commercial services like Google Scholar automatically detect formal citation, we can significantly increase the visibility of research software in the literature. A retrospective approach, combined with the interventions we describe below, can quickly jumpstart a software credit ecosystem, while we wait for prospective approaches to gradually build a best-practice citation culture.

However, there are challenges to the retrospective approach. In particular, it is difficult for machines to parse the diverse morphology of informal mentions, making them harder to detect than traditional, formal citations. Largely because of this, researchers have only recently begun to tackle the problem of automatically finding software mentions at a meaningful scale.

Their approach to date has been mostly rule-based. This straightforward natural language processing (NLP) technique uses a set of rules to score each word on its likelihood of being a software mention. For instance, the name “FooPlot” in the phrase “we used the FooPlot 1.2 program” could score points for preceding “program” and having a version number next to it.

Words with enough points count as software mentions. Using the rule-based technique, Duck et al identify software mentions with a precision of 0.58 and recall of 0.68 (Duck, Nenadic, Brass, Robertson, & Stevens, 2013). In a later paper they improve this to 0.80 and 0.64 respectively (Duck et al., 2016). Priem and Piwowar (2016) employ a related approach in the Depsy application (http://depsy.org), using preset search phrases to find mentions. All these efforts rely on researcher intuition to guide the selection and weighting of rules, limiting the ability to optimize them systematically. Pan et al. (2015) address this limitation by generating the rules automatically, using a machine-learning bootstrapping technique. Their approach sacrifices recall (0.43) but results in greatly improved precision in identifying software mentions (0.94). However, it still relies on bootstrapping from a discreet ruleset.

It is likely that much better results can be obtained by discarding the rule-based approach altogether. There is significant evidence for this in the literature around named entity recognition (NER), and particularly biomedical named entity recognition (BNER) (Settles, 2004). Like the nascent research in automatically extracting software mentions, BNER uses NLP to extract important terms from scholarly literature. However, unlike the software mention literature, BNER has evolved beyond its early use of rule-based techniques (Tang et al., 2014). Rather, today's BNER research uses machine-learning approaches that have no explicit rules, and instead train a computer model to recognize entities in text. Several machine learning models are in use, but the most successful of these today is conditional random fields, or CRF (Lafferty, McCallum, & Pereira, 2001).

Although of course identifying proteins in scientific articles is different from identifying software, but the problem space is similar enough to believe that applying state-of-the-art NER techniques, especially CRF, to the problem of automatically detecting software mentions will

produce much better results than rule-based efforts to date. This information can then be used in a variety of systems to promote culture change by making software more visible. For example, researchers have used NLP techniques to identify missing citations of traditional, non-software, articles (McNee et al., 2002), an idea that we extend to software below.

3. Why are the proposers qualified to address this problem?

Our team is well qualified to undertake this project, with experience and expertise in content analysis, machine learning, system implementation, and socio-technical studies of science.

James Howison is an Assistant Professor at the UT Austin Information School, joining in 2011 after a post-doctoral position at CMU (with James D. Herbsleb) and his PhD at the Information School at Syracuse. He studies software work, both Free and Open Source Software development and Scientific Software work. His thinking is socio-technical, equally focused on incentives and social dynamics as on technical structures in the software. In 2015 he was awarded an NSF CAREER grant to study scientific software projects attempting to transition from grant-funding to open source-like peer production. In 2015 he published a study of software mentions in the biology literature, building a reliable content analysis scheme and manually analyzing 90 randomly selected articles (Howison & Bullard, 2015).

Dr Heather Piwowar is a cofounder of Impactstory. Prior to Impactstory, Heather was a PhD student in Biomedical Informatics at the University of Pittsburgh in Dr. Wendy Chapman's Natural Language Processing (NLP) group, and a postdoc at Duke University. A focus of her dissertation and later research has been using NLP to identify instances of data sharing and reuse in the published biomedical literature (Piwowar, 2011; Piwowar & Chapman, 2008a, 2008b, 2009). Piwowar's 2007 paper on data sharing citations used NLP to identify more than 10,000 instances of data sharing in the research literature; it has received more than 400 citations

(Piwowar, Day, & Fridsma, 2007). As a postdoc, Heather also served as a paid NLP consultant to UBC's NeuroEthics department (Garnett, Whiteley, Piwowar, Rasmussen, & Illes, 2011). She holds Bachelor and Master's degrees in Electrical Engineering and Computer Science from MIT, focussing on signal processing and artificial intelligence. She has 15 years of professional software experience. Together, she and Jason Priem designed, implemented and support Impactstory and Depsy.

Jason Priem is cofounder of Impactstory, a nonprofit that grew out of his doctoral studies in *altmetrics*, scholarly impact metrics based online activity. Jason coined the word altmetrics, authored its founding document, with more than 300 citations (Priem, Taraborelli, Groth, & Neylon, 2010), and organized its first workshop. Since then the field has become a major and fast-growing subdiscipline of scientometrics. Jason's experience in uncovering and understanding metrics of nontraditional scholarly products provides a solid background for the work proposed in this project. He is also a software developer of ten years experience, and has successfully designed, implemented and supported production-level, web-scale applications including Depsy and Impactstory. These applications have supported tens of thousands of users, and handle millions of rows of data daily.

4. What is the approach?

Our approach is a three year program to develop a manually coded gold-standard dataset of software mentions, build a machine learning system able to recognize software in the literature, and create a dataset of software in publications using that system. We will build three prototypes that demonstrate the potential usefulness of such data in addressing incentives for software work in science: CiteSuggest, CiteMeAs, and Software Impactstory. We will study these prototypes in use to identify the socio-technical barriers to full scale, sustainable implementations.

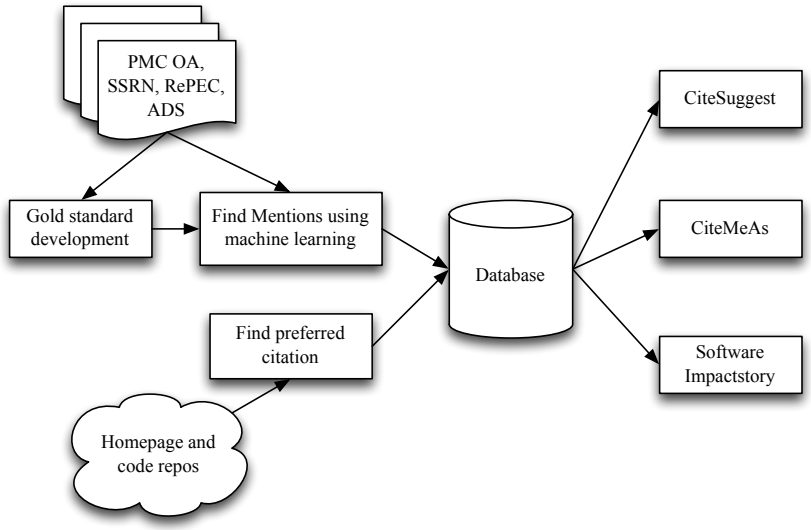

Figure 1: Overall architecture. The ML module ("find mentions") scans the literature, after being trained on the annotated gold standard. Mentions are found and stored in the database, along with contextual information. The citation finder ("find preferred citation") scrapes and parses webpages and repositories. The three prototype apps (CiteSuggest, CiteMeAs, and Impactstory) all present data from the ML database to users, covering different use cases around software credit and attribution.

Gold standard development

We will create and publish a gold standard dataset of software mentions. The dataset will support the work described below and provide a resource to the wider publication mining community.

We have a reliable content analysis scheme in place (created for Howison and Bullard, 2015) and two trained coders. The scheme is described in Howison and Bullard (attached as an appendix, see Tables 3 and 4 on page 5 and 6 of that article). The scheme includes codes for the name of a piece of software, whether it was used or just discussed, whether it includes a URL, whether it includes a version indication (either version number or date), configuration details and coding for the creator (usually a person or company). The scheme also contains codes for any citation and codes for identifying the type of reference linked to in the bibliography (publication, software paper, project page/name, manual etc).

The content analysis work will be undertaken by a team of undergraduates, managed by a doctoral student from University of Texas at Austin and overseen by James Howison, as PI. We

intend to train between 5 and 10 undergraduates. To avoid errors caused by fatigue we will limit the number of articles assigned at a time and include a percentage of articles assigned to multiple coders (to periodically check accuracy). We may arrange joint analysis sessions, such as bringing the coders together for content analysis "sprints." The undergraduates will also be included in lab meetings to introduce them to the process of research. We plan to include undergraduate students from nearby Huston-Tillotson College, a historically black and minority serving institution in East Austin.

We aim to build the dataset out to cover diverse fields: biomedicine, astronomy, and economics, with 350-500 articles from each field in the dataset using a random sampling strategy, stratified by venue. This work will be completed early in the grant, ideally in the first year. We will publish the Gold Standard dataset openly on Zenodo, under a CC-0 license. Appendix E (Information Products) shows an example of our anticipated published data.

Machine Learning Implementation

Using the gold standard dataset as training data, Piwowar and Priem will design and implement a machine learning system capable of automatically recognizing and extracting software mentions from full-text publications. By running this system over our selected corpora (see Information Products Appendix), we will create a uniquely comprehensive dataset of software mentions in the disciplines of economics, astronomy, and biomedicine.

We will implement such a system using machine learning named entity recognition (NER). We will use the conditional random fields (CRF) algorithm, using the popular CRFSuite implementation via the NLTK Python library (Okazaki, n.d.). The CRF classifier is particularly strong for NER applications because it accounts for context of mentions by leveraging the sequence of words (Lafferty et al., 2001).

We will supply the CRF model with three types of features. First, we will employ a variety of standard features including bag-of-words, part of speech, capitalization, and others. To these we will add several document-level features like indexing terms, since these will improve identification of discipline-specific software packages.

Second, we will also leverage previous rules-based work (Duck et al., 2016; Pan et al., 2015) including published "this is a software word" heuristics as features. A preprocessing step will to find which tokens (if any) are matched by a given rule, with the result being fed to the CRF model. Some rules may turn out to be redundant within the model, and we will experiment to see which ones can be removed to improve performance.

Finally, we use unsupervised word representations to add a set of additional features. We will run Google's Word2Vec (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) algorithm over the entire corpus. Word2Vec is a two-layer neural net that produces a word representation vector for each word in the corpus; this vector expresses the word's meaning in ways that are often surprisingly robust. We then feed this vector as another feature into the CRF model. This approach has shown good results in both general (Turian, Ratinov, & Bengio, 2010) and domain-specific NER tasks (Tang et al., 2014).

Intervention Prototypes

Prototype 1: CiteSuggest is a tool to recommend software citations based on submitted article text or source code. Authors will be prompted to upload their Word .doc, text, or latex manuscript. We then run the manuscript through our machine learning algorithm to find software this author mentioned.

The next step will use data from a module that identifies the preferred citation using available data. This PreferredCitation module will use any URLs we are able to parse out of the literature

as part of the mention detection, particularly ones pointing to the software's homepage, source repo, or package repo. To supplement these automatically discovered URLs, we will also experiment with using undergrads to manually look up and store URLs. We will automatically resolve these URLs, and then parse pages to find preferred citation information, examining using CodeMeta annotations, source code annotations (R CITATION, Python DueCredit, etc), as well as heuristic-identified plain text citations. ("please cite this package as <target>"). We will also check CrossRef to see if any software has a DOI assigned, and if so resolve and return the relevant metadata (author, name, date, etc as well as DOI string).

We will identify the most appropriate formal citation using these steps, returning the first one found:

- Is there a DOI for this software? Return the resolved metadata in citation format.
- Is there an author-preferred software paper to cite? Return it.
- Is there a most-commonly cited software paper to cite? Return it.
- Else, return a generic citation template, appropriate for use in the reference list, populated by the software name and URL.

After creating preferred citations for each software instance, we then present this list to the user, also with option to download in multiple reference manager formats.

In addition to suggesting best-practice citations based on existing software mentions in papers, we'll attempt to identify related software that the author may have neglected to mention at all, by matching patterns from our ML database (similar to Amazon's "customers who made similar purchases to you also bought" feature and the citation recommendation approach of (McNee et al., 2002). These will be presented to the user with the list of preferred citations discussed above.

The CiteSuggest prototype will also allow authors to submit Python or R scripts that represent a processing workflow for a research project. Using components already implemented for Depsy, we'll read the first-level dependencies (the libraries that the user composed into their workflow) for the software package, using mechanisms appropriate for the language (e.g., import for Python and library() for R). We will deliver a list of appropriate formal citations using the same method described above.

We will pilot this prototype with publication venues that we have worked with before. For paper submission we intend to pilot this with Ubiquity Press (working with Brian Hole) and the eLife journal (working with Mark Patterson). For code submission, we intend to work with the Journal of Open Research Software (JORS) (working with Neil Chue Hong and Matt Turk). We will demonstrate the tool to these venues, seek introductions to their lead users, and ask them to include the tool as a suggestion in their submission guidelines. In addition we will make the tool publically available and promote it as described below.

Prototype 2: CiteMeAs for GitHub will help software projects on GitHub make clear requests for their preferred citations. To do this, we will start by identifying all projects in our ML database with a GitHub URL. We will then automatically submit pull requests to each GitHub repo with an updated README file that includes the sentence “Please cite this as: <citation>.” We will find the correct information to put in the “<citation>” field using the same techniques described above for the CiteSuggest prototype. The pull request will also include a codemeta.jsonld file filled out using data we hold in our ML database, particularly the "softwarePaperCitation" and "softwareCitation" fields (see example of these files at https://github.com/codemeta/codemeta/blob/master/example-codemeta.json). We will then monitor how the project reacts, updating our records of preferred citation if they edit our

suggested citations in the pull request or repository. The information sent in the pull request will also be exposed online in a web service.

To pilot this prototype we intend to work with a sub-set of projects that have already indicated their interest in making explicit requests for citation, either by implementing R's citation() command or participating in the Python DueCredit process. Our pilot will enable us to fine-tune how we make pull-requests and the language we will use to introduce our pull-request. After these small pilots we will proceed with broad-based outreach via pull-request to the full population of projects identified by our publication mining and which are on github.

Prototype 3: Software Impactstory is an interface to help scholars that contribute software to to identify their software impact in existing literature. We will prototype this as a component of the existing Impactstory platform (http://impactstory.org), using the data from the corpora we have analyzed above. The Impactstory prototype will use this data to identify all of a user's software products and papers related to those products. The database will be used again to pull in all literature mentions of each software project or software paper. Further, software dependencies and downloads will be retrieved from Depsy (http://depsy.org; Depsy PIs are Priem and Piwowar), so that mention metrics are part of a full software impact picture.

Profiles will display which publications have mentioned a user's software, as well as patterns and trends over time. Embeddable badges will be created that highlight the reuse of the software, supported by hard numbers and percentiles; Priem and Piwowar have several years of experience in data visualization of alternative impact data in this fashion. In addition, the system will learn from observed practice of scholarly software projects, described in Howison et al. (2015), and provide context for each mention (e.g., surrounding sentences) allowing users to understand how their software was used, supporting both quantitative and qualitative stories of impact.

We will pilot the Software Impact Story prototype with existing users of Impactstory. Among the regular users of Impactstory is a sizeable community of academics who write software and have contacted us to express interest in tracking the impact of their software. We will invite them to join a pilot group to elicit feedback and generate engagement and publicity.

We will work to build broad conversation around these prototypes, both to attract interested users and to help nurture and guide important early conversations in this area. In addition to our pilots, identified above, we will demonstrate prototypes at appropriate venues, such as presentations at conferences like the Research Data Alliance conference, FORCE11, and the Research Software Engineers conference. We will engage in regular updates and conversations via our Twitter accounts, which reach around 10,000 followers, and reach out to Depsy's and Impactstory's very engaged user bases. Finally, we will leverage our contacts with high-visibility publications including Science, Nature, and the Chronicle of Higher Education to encourage research press coverage, a strategy that has been consistently successful with Impactstory and Depsy.

Socio-technical evaluations

Finally, we will conduct socio-technical evaluations to understand the potential of our prototype interventions. We seek to understand the technical and social opportunities and barriers for their implementation and success. Accordingly we will undertake a qualitative, socio-technical study around each prototype, drawing on our technical experience in developing the prototypes and on interviews with stakeholders. The stakeholders for each prototype are different, and will build on and extend beyond the pilot groups we identify above. For example, for CiteSuggest we will contact publishers, representatives of scientific societies, and scholarly publishing technologists

(e.g., CrossRef). We will recruit participants in part through responses to the publicity generated by our prototypes.

The interviews will be based around demonstrating our prototype and asking them to interact with it. We will, of course, ask straightforward questions about usability, but our emphasis will be much wider. Accordingly we will elicit answers to questions such as: What would it take to bring a prototype to full-scale implementation? Would a full-scale implementation have the impact that we anticipate? Who would benefit from such an implementation? Who would be motivated to provide the ongoing resources to realize a sustainable implementation? How might such a system be disruptive or undermine current desirable arrangements? We will identify potential users and beneficiaries and understand their incentives for ongoing support of a system. For example, our citation suggestion system may be useful within a publishing workflow, helping to normalize citations and helping reviewers assess whether articles are citing the software correctly. Our evaluations will understand whether there is real demand and potential for publishers or scientific societies to implement a sustainable system of this type, as well as a technical roadmap for implementation. We anticipate undertaking between 5 and 8 interviews with appropriate stakeholders for each prototype, for a total of between 25 and 40 interviews. In addition we will gain insight from informal reactions to our prototype demonstrations at conferences and workshops.

The output of this work will be a technical and institutional roadmap for each prototype intervention. In addition we will improve our understanding of the stakeholders, capabilities, and incentives in the practice of science, the publishing system, and systems of evaluation and reward in science. Through this process we will generate a summative evaluation of our prototype interventions and provide empirical understanding of the practice of science, which we

will publish in fields such as scholarly communications (e.g., JASIST) and science and technology studies (e.g., Science, Technology and Human Values.).

5. What will be the output from the project?

1) A gold standard dataset of software mentions, published for text and data mining (TDM) community. 2) A protocol for conducting manual content analysis, including formats for adding to the gold standard dataset. 3) A trained machine learning system able to identify software mentions, available as open source. 4) A publically available dataset of Software Mentions extracted from corpora in biomedicine, economics, and astronomy. 5) Three prototype interventions, each with technical and institutional map for their implementation. Code for each proto type will available as open source. 6) Publications about challenges and opportunities for sustainable interventions, based on the socio-technical studies of the prototypes.

6. What is the justification for the amount of money requested?

The budget for this project totals $635,261 over 3 years (approximately $210,000 a year) and includes support at UT Austin and a sub-contract for Impactstory. UT Austin is primarily responsible for the content analysis and the socio-technical assessment. Impactstory is primarily responsible for the implementation of machine learning to find software mentions and the creation of the intervention prototypes. The project requests funding for three years effort, sufficient time to build out the dataset, take advantage of it for the creation of the prototypes, and undertake socio-technical assessments.

Funds at UT Austin total include 2 months of summer salary support for PI Howison, who is employed in a 9 month tenure track position, a full graduate assistantship (20 hours week stipend, benefits, and full tuition costs), and funding for part-time hourly employment of 5-10

undergraduate students for content analysis. The graduate student will, under the direction of PI Howison, manage the team of undergraduate content analysis coders. Howison and the graduate student will undertake the socio-technical assessment interviews together (with Piwowar and Priem when appropriate). Additional amounts are budgeted travel for dissemination and research trips for the socio-technical assessment, and for publication costs.

Funds at Impactstory total $307,796, over three years. The budget provides for 4.5 months of salary support plus benefits each year for each of PIs Piwowar and Priem. In addition the amount includes a modest budget for computing costs, to facilitate the large-scale computation underlying the machine learning. The budget includes travel and open access publication costs.

7. What other sources of research support does the proposer have in hand or has he/she applied for to support the research team?

In 2013 Impactstory was awarded a $300k from the National Science Foundation for a software-citation related project: EAGER Award #1346575 "Investigating the value of automatically-gathered software impact data." This grant has resulted in the design, release, and support of Depsy (http://depsy.org/), a web application that helps build the software-intensive science of the future by promoting credit for software as a fundamental building block of science. It has been featured in a Nature article, multiple blog posts, and spurred much discussion in conferences and online. The Impactstory research team will continue to be funded by this grant at 50% FTE through September 2017, under a no-cost extension (in process). We are in early talks with several private foundations that plan to enter the open science/altmetrics space; we will be submitting grant proposals to them in the next few months for funding in 2018 and beyond.

8. What is the status and output of current and/or previous Sloan grants?

PI Howison was previously funded by the Sloan Foundation to organize a small workshop. The workshop brought together researchers undertaking empirical research on software and data practices, including the research groups of Christine Borgman (UCLA), Jim Herbsleb (CMU), James Howison (Texas at Austin), and the Sloan/Moore Data Science Studies groups. Many of the participants had not met before because they represent different fields of research (Information Science, Computer Science/CSCW, and Science and Technology Studies, as well as the emerging field of Data Science Studies). The workshop was successful, sharing research results and approaches and seeding ongoing interactions and collaborations among the 18 participants. Josh Greenberg attended for the Sloan Foundation.

Piwowar and Priem were previously funded by the Sloan Foundation with two grants. The first, through Duke University in 2012/13 was for $125k, with the purpose of developing a prototype altmetrics web application into a robust application and a formal nonprofit organization to help transform research assessment and review. This was successful: Impactstory was rewritten into a mature application, and Impactstory Inc was incorporated and granted 501(c)3 charity status. A second Sloan grant, for $500k, was awarded to Impactstory Inc in 2013 to "support the scaling and further development to sustainability of ImpactStory." Impactstory has successfully scaled and been developed since then: it currently has more than 3,500 users and is growing quickly, supported by strong word-of-mouth: 43% of surveyed users say they would strongly recommend Impactstory to a colleague. It has been an important touchstone of the altmetrics community with over 7,500 Twitter followers, and been featured in Nature, Science, The Chronicle of Higher Education, and on BBC Radio. Impactstory has experimented with several business models during the grant period, as was the goal of the grant, and remains financially healthy.

Appendicies

Appendix A: Curriculum Vitae

1. James Howison
2. Heather Piwowar
3. Jason Priem

James Howison, Ph.D.

School of Information
University of Texas at Austin
Austin, TX

jhowison@ischool.utexas.edu
http://james.howison.name
Phone: +1 315 395 4056

Education

Syracuse University	Information Science and Technology	Ph.D. 2009
University of Sydney	Economics & Social Science	B.Ec. 1998

Appointments

Aug 2011–Present Assistant Professor, School of Information, University of Texas at Austin

Jan 2009–Aug 2011 Post-doc, School of Computer Science, Carnegie Mellon University

Five Related Products

Howison, J., & Bullard, J. (2015). 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 (JASIST)*, Article first published online: 13 May 2015. http://doi.org/10.1002/asi.23538

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

Howison, J and Crowston, K (2014) Collaboration through Open Superposition. MIS Quarterly. 38(1) 29-50.

Howison, J & Herbsleb, J. D. (2013) Scientific Software Production: Incentives and Integration In Proceedings of ACM Conference on Computer Supported Cooperative Work (CSCW 2013).

Howison, J., & Herbsleb, J. D. (2011). Scientific software production and collaboration. In Proceedings of ACM Conference on Computer Supported Cooperative Work (CSCW 2011).

Five Other Products

Crowston, K., Wei, K, Howison, J., and Wiggins, A. (2012). Free (Libre) Open Source Software Development: What We Know and What We Do Not Know. ACM Computing Surveys. 44(2)

Howison, J., Wiggins, A., & Crowston, K. (2012). Validity Issues in the Use of Social Network Analysis with Digital Trace Data. Journal of the Association for Information Systems, 12(12)

Howison, J., Conklin, M., and Crowston, K. (2006). FLOSSmole: A collaborative repository for FLOSS research data and analysis. International Journal of Information Technology and Web Engineering, 1(3):17–26.

Berente, N., Howison, J. & King, J. (2013) Five Models for Interaction Between Science Enterprises and Organization Scientists" Atlanta Conference on Science and Innovation Policy.

Wiggins, A., Howison, J., & Crowston, K. (2009). Heartbeat: Measuring Active User Base and Potential User Interest in FLOSS Projects. In Proceedings of IFIP Open Source Software Conference

Synergistic Activities

- NSF CAREER award (2015) "Sustaining Scientific Infrastructure: Researching Transition from Grants to Peer Production" (1453548)
- VOSS Research Coordination Network with Nick Berente and John King
- SBE SciSiP award "The Scientific Network Map" researching techniques to measure usage and impact of scientific software, in conjunction with Jim Herbsleb at CMU
- Co-organizer of "Sharing, Re-use and Circulation of Resources in Cooperative Scientific Work"
- Program Committee for WSSSPE2 workshop on Software Sustainability
- Co-founded FLOSSmole, an NSF funded data and analysis sharing repository for research on free and open source software development. http://ossmole.sf.net
- Teach "Building and Managing Online Communities" and "Data Wrangling" at UT Austin iSchool

List of Collaborators

Nicolas Berente (UGA), John King (Michigan), James D. Herbsleb (CMU), Kevin Crowston (Syracuse University), Robert Heckman (Syracuse University), Megan Conklin (Elon University), Lee McKnight (Syracuse University), Andrea Wiggins (Syracuse University), Jungpil Hahn (NUS), Gwanhoo Lee (American), Ewa Deelman (ISU/USC), Susan Winter (Maryland), Brian Butler (Maryland)

Graduate Students and Advisors

Advisors: Kevin Crowston (Syracuse University, Ph.D.),

James D. Herbsleb (Carnegie Mellon University, Post-doc)

Graduate Students: Eunyoung Moon, Julia Bullard, Nicholas Gottschlich, Johanna Cohoon.

Heather A. Piwowar

heather@impactstory.org

Professional Preparation

MIT	Electrical Engineering and Computer Science	B.S.	1995
MIT	Electrical Engineering and Computer Science	M. Engineering	1996
U of Pittsburgh	Biomedical Informatics	Ph.D.	2010

Appointments

2012-present	Cofounder and employee of Impactstory, Carrboro NC.
2010-2013	Postdoctoral Researcher, DataONE, National Evolutionary Synthesis Center NC.
2010	NLP Consultant for Neuroethics group at UBC, Canada
2001-2005	Senior Systems Developer, Precision Therapeutics Inc., Pittsburgh PA
1998-2001	Senior Software Developer, Vocollect Inc., Pittsburgh PA
1996-1998	Software Engineer, Ascend Communications Inc., Alameda CA

Relevant Products

1. Piwowar, Chapman (2008) Identifying data sharing in biomedical literature. AMIA Annual Symposium. (28 citations)
2. Piwowar, Chapman (2008) Linking database submissions to primary citations with PubMed Central. *BioLINK* 2008
3. Garnett, Piwowar, Rasmussen, Illes (2010) Formulating MEDLINE queries for article retrieval based on PubMed exemplars. Nature Precedings.
4. Harkema, Piwowar, Amizadeh, Dowling, Ferraro, Haug, Chapman (2008) A Baseline System for the i2b2 Obesity Challenge In: The Second i2b2 Workshop on Challenges in Natural Language Processing for Clinical Data November 7-8 2008, Washington DC
5. Piwowar, Day, Fridsma (2007) Sharing Detailed Research Data Is Associated with Increased Citation Rate. *PLOS ONE* 2: 3. e308 (418 citations)
6. Priem and Piwowar. Depsy (2015) web application and open source software. http://depsy.org and https://github.com/impactstory/depsy.
7. Priem and Piwowar. Impactstory (2013) web application and open source software. http://impactstory.org and https://github.com/impactstory/impactstory-tng.

Other Significant Products

1. Piwowar (2013) Value all research products. *Nature* 493,159. (181 citations)
2. Priem, Piwowar, Hemminger (2012) Altmetrics in the wild: Using social media to explore scholarly impact. *arXiv*:1203.4745 (178 citations)
3. Piwowar and Vision. (2013) Data reuse and the open data citation advantage. *PeerJ* (88 citations)

Biosketch for Heather A. Piwowar, continued.

Synergistic Activities

1. Committed to open research products: open datasets are highly viewed, downloaded, and have been reused in research papers; open source has been reused: pypub (reused by a researcher), Impactstory (reused by commercial company); open APIs used by many.
2. Given more than 30 invited talks on research data, textmining, and altmetrics since Jan 2011. Co-organized a AAAS panel on Open Research Data at AAAS; also a speaker.
3. Active online research presence: research blog at http://researchremix.wordpress.com is frequently highlighted in OA News and has been referenced in Nature News; research twitter account has more than 5000 followers.
4. Research (data sharing and reuse; altmetrics) and advocacy (open science policies; negotiating groundbreaking textmining access with Elsevier) have been covered by Nature News, The Guardian, The Chronicle of Higher Education, CBC radio, Peter Suber's OA newsletter, the New Yorker blog, and other venues.
5. Recently featured in a metrics-driven list of "100 Awesome Women in the Open-Source Community You Should Know" http://bit.ly/1XUDLu2

Collaborators and Co-Editors

Allard, Suzie (University of Tennessee, Knoxville); Asper, Vernon (University of Southern Mississippi); Becich, Michael J. (University of Pittsburgh); Bilofsky, Howard (University of Pennsylvania); Carlson, Jonathan D. (University of Wisconsin–Madison); Chapman, Wendy W. (University of San Diego); Cook, Robert B. (Oak Ridge National Laboratory); Crowley, Rebecca S. (University of Pittsburgh); Enriquez, Valerie (Simmons College); Garnett, Alex (University of British Columbia); Guralnick, Robert (University of Colorado at Boulder); Hemminger, Bradley (University of North Carolina at Chapel Hill); Herendeen, Patrick S. (Chicago Botanic Garden); Helgen, Kristofer M. (National Museum of Natural History); Hill, Andrew (University of Colorado at Boulder); Holmberg, Kim (Åbo Akademi University); Illes, Judy (University of British Columbia); Jameson, Mary Liz (Wichita State University); Judson, Sarah W (Brigham Young University); Lapinski, Scott (Harvard); McDade, Lucinda A. (Rancho Santa Ana Botanic Garden); Maddison, David R. (Oregon State University); Pikas, Christina K (University of Maryland); Priem, Jason (University of North Carolina at Chapel Hill); Rasmussen, Edie (University of British Columbia); Sandusky, Robert J. (University of Illionois at Chicago); Vickers, Andrew (Memorial Sloan-Kettering Cancer Center); Vis, Morgan L. (Ohio University); Vision, Todd J. (University of North Carolina – Chapel Hill); Weber, Nicholas (University of Illinois, U-C); Whiteley, Louise (University of British Columbia); Whitlock, Michael J. (University of British Columbia); Wilson, Bruce E. (Oak Ridge National Laboratory)

Graduate and Postdoctoral Advisors:

Chapman, Wendy W. (was University of Pittsburgh, now University of San Diego)
Vision, Todd J. (University of North Carolina – Chapel Hill)

Jason Priem

jason@Impactstory.org

Professional Preparation

University of Florida	History	B.A.	2001
University of Florida	Social Studies Education	M.Ed.	2002
University of North Carolina-Chapel Hill	Information Science	PhD (unfinished)	2009-

Appointments

2012-	Co-founder of Impactstory, Carrboro NC
2009-	PhD student, UNC-Chapel Hill School of Information and Library Science
2008-2009	Project manager and Instructional Designer, University of Florida
2007-2008	Web designer, freelance
2002-2007	Teacher, Union Grove Middle School, McDonough GA

Relevant Products

1. Priem and Piwowar (2012) Impactstory web application. http://impactstory.org and open source at https://github.com/total-impact (184k unique visitors, >15k registered users, API users include PeerJ, eLife, DataCite, PMC Europe)
2. Priem and Piwowar (Nov 2015) Depsy web application. http://depsy.org and open source at https://github.com/impactstory/depsy (>20k unique visitors)
3. Priem (2013) Beyond the paper. *Nature* 495:7442 (99 citations)
4. Priem, J., & Hemminger, B. H. (2010). Scientometrics 2.0: Toward new metrics of scholarly impact on the social Web. *First Monday, 15(7)*. (238 citations)
5. Priem, J., Taraborelli, D., Groth, P., & Neylon, C. (2010). alt-metrics: a manifesto. Retrieved August 15, 2011, from http://altmetrics.org/manifesto/ (336 citations)

Other Significant Products

1. Bar-Ilan, Haustein, Peters, Priem, Shema, Terliesner (2012). Beyond citations: Scholars' visibility on the social Web. *17th International Conference on Science and Technology Indicators*. Montreal, Canada, 5-8 Sept. 2012. (109 citations)
2. Priem, Piwowar, Hemminger (2012) Altmetrics in the wild: Using social media to explore scholarly impact. arXiv:1203.4745 (178 citations)
3. Priem and Costello (2010) How and why scholars cite on Twitter. *Proceedings of the 73rd ASIS&T Annual Meeting*. Pittsburgh, PA, USA. doi:10.1002/meet.14504701201 (152 citations)
4. Priem (2014) Altmetrics. Book chapter in *Beyond bibliometrics: Harnessing multidimensional indicators of scholarly performance*, Sugimoto and Cronin, Eds. MIT Press. (46 citations)
5. Priem and Hemminger. (2012). Decoupling the scholarly journal. *Frontiers in Computational Neuroscience* 6:19. (42 citations)

Biosketch for Jason Priem, continued.

Synergistic Activities

1. 15 years of experience and coursework in visual design and web usability.
2. Wrote 50% of Depsy code and Impactstory code, including Angular.js UI and Python backend.
3. Early leader in altmetrics community: coined the term, organized first two workshops, wrote key manifesto.
4. Engaged in outreach via blog and Twitter feed (4.5k followers).
5. Work profiled by BBC Radio, Forbes blog, Wired blog, and the Chronicle of Higher Education.
6. Has given more than 30 talks and interviews on alternative scholarly products (including software) and the future of scholarly communication since 2011. These have addressed diverse audiences, including at the NSF, NIH SMRB, SXSW, BBC Radio, the NIH Library, AAAS, DataCite, ORCID and others.

Collaborators and Co-Editors

Bar-Ilan, Judit (Bar-Ilan University); Black, Erik (University of Florida); Costello, Kaitlin (UNC-Chepel Hill); Dawson, Kara (University of Florida); Dzuba, Tyler (UNC-Chapel Hill); Garnett, Alex (University of British Columbia); Groth, Paul (VU Univ, Amsterdam); Haustein, Stefanie (Universite de Montreal); Hemminger, Bradley (UNC-Chapel Hill); Holmberg, Kim (Åbo Akademi University); Neylon, Cameron (UK Science and Technology Facilities Council); Parra, Cristhian (University of Trento); Peters, Isabella (Heinrich-Heine-University); Pikas, Christina K (University of Maryland); Piwowar, Heather (NESCent); Shema, Hadas (Bar-Ilan University); Taraborelli, Dario (Wikimedia Foundation); Terliesner, Jens (Heinrich-Heine-University); Waagmeester, Andra (Maastricht University); Weber, Nicholas (University of Illinois, Urbana-Champaign)

Graduate Advisor:

Bradley M. Hemminger (University of North Carolina at Chapel Hill)

Appendix B: Conflicts and Interest/Sources of Bias

Howison, Piwowar, and Priem do not have conflicts of interest or sources of bias related to this project. UT Austin and Impactstory Inc. do not have conflicts of interest or sources of bias related to this project.

Students employed on the project will undergo UT Austin's responsible conduct of research and conflict of interest training and make appropriate declarations required by UT Austin.

Appendix C: Attention to Diversity

Traditionally underrepresented groups will play key roles in several components of this grant.

Minority students will form a vital part of the team pursuing the grant. To create the gold-standard set of tagged software literature mentions, we will recruit undergraduates from Huston-Tillotson, an historically black university in Austin. Howison has taught and advised Autumn Caviness, a UT Austin doctoral student, who is an adjunct instructor there and who has organized hackathons at Huston-Tillotson; we will work with her to recruit content analysis students from Huston-Tillotson. This is important since a key to graduate student recruitment is to provide opportunities for legitimate peripheral participation (Wenger, 1998), which is well served by providing apprenticeship opportunities in research projects.

Although women are currently underrepresented in software development, and particularly in open-source software, half our two-person development team (Heather Piwowar) will be female. Heather has been recognized as one of the top 100 women in open-source software. (http://blog.sourced.tech/post/100-awesome-women-in-the-open-source-community-you-should-know/). Moreover, the UT Austin Information School, despite being the smallest school at UT Austin, graduates the highest number of women trained in technology, and our doctoral program is over 60% female. We have identified Johanna Cohoon, an incoming doctoral student at UT Austin, as likely funded under this award.

Appendix D: Bibliography

Allen, A., Berriman, G. B., DuPrie, K., Mink, J., Nemiroff, R., Robitaille, T., … Wallin, J. (2015). Improving Software Citation and Credit. *arXiv:1512.07919 [astro-Ph]*. Retrieved from http://arxiv.org/abs/1512.07919

Atkins, D. (2003). *Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure*. Retrieved from http://www.nsf.gov/od/oci/reports/toc.jsp

Bietz, M. J., Baumer, E. P., & Lee, C. P. (2010). Synergizing in Cyberinfrastructure Development. *Computer Supported Cooperative Work*, *19*(3-4), 245–281. http://doi.org/10.1007/s10606-010-9114-y

Boisvert, R. F., & Tang, P. T. P. (Eds.). (2001). *The Architecture of Scientific Software*. Retrieved from http://www.springer.com/computer/programming/book/978-0-7923-7339-1

Borgman, C. L. (2007). *Scholarship in the Digital Age: Information, Infrastructure, and the Internet*.

CodeMeta Collaboration. (n.d.). codemeta/codemeta [Github Project Page]. Retrieved August 4, 2016, from https://github.com/codemeta/codemeta

Duck, G., Nenadic, G., Brass, A., Robertson, D. L., & Stevens, R. (2013). bioNerDS: exploring bioinformatics' database and software use through literature mining. *BMC Bioinformatics*, *14*, 194. http://doi.org/10.1186/1471-2105-14-194

Duck, G., Nenadic, G., Filannino, M., Brass, A., Robertson, D. L., & Stevens, R. (2016). A Survey of Bioinformatics Database and Software Usage through Mining the Literature. *PLOS ONE*, *11*(6), e0157989. http://doi.org/10.1371/journal.pone.0157989

Edwards, P. N., Mayernik, M. S., Batcheller, A. L., Bowker, G. C., & Borgman, C. L. (2011). Science friction: Data, metadata, and collaboration. *Social Studies of Science*, *41*(5), 667–690. http://doi.org/10.1177/0306312711413314

Eklund, A., Nichols, T. E., & Knutsson, H. (2016). Cluster failure: Why fMRI inferences for spatial extent have inflated false-positive rates. *Proceedings of the National Academy of Sciences*, *113*(28), 7900–7905. http://doi.org/10.1073/pnas.1602413113

Garnett, A., Whiteley, L., Piwowar, H. A., Rasmussen, E., & Illes, J. (2011). Neuroethics and fMRI: Mapping a Fledgling Relationship. *PLOS ONE*, *6*(4), e18537. http://doi.org/10.1371/journal.pone.0018537

Halchenko, Y., & Matteo Visconti di Oleggio Castello. (2016, July 7). duecredit/duecredit [Github Project Page]. Retrieved August 4, 2016, from https://github.com/duecredit/duecredit

Howison, J., & Bullard, J. (2015). 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 (JASIST)*, Article first published online: 13 MAY 2015. http://doi.org/10.1002/asi.23538

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

Howison, J., & Herbsleb, J. D. (2011). Scientific software production: incentives and collaboration. In *Proceedings of the ACM Conference on Computer Supported*

Cooperative Work (pp. 513–522). Hangzhou, China. http://doi.org/10.1145/1958824.1958904

Ince, D. C., Hatton, L., & Graham-Cumming, J. (2012). The case for open computer programs. *Nature*, *482*(7386), 485–488. http://doi.org/10.1038/nature10836

Joppa, L. N., McInerny, G., Harper, R., Salido, L., Takeda, K., O'Hara, K., … Emmott, S. (2013). Troubling Trends in Scientific Software Use. *Science*, *340*(6134), 814–815. http://doi.org/10.1126/science.1231535

Katz, D. S. (2013). Citation and Attribution of Digital Products: Social and Technological Concerns. In *Papers presented at WSSSPE (Working towards Sustainable Software for Science: Practice and Experiences) at Supercomputing 2013*. Denver, CO. http://doi.org/10.6084/m9.figshare.791606

Katz, D. S., Choi, S.-C. T., Lapp, H., Maheshwari, K., Löffler, F., Turk, M., … Venters, C. (2014). Summary of the First Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE1). *Journal of Open Research Software*, *2*(1). http://doi.org/10.5334/jors.an

Katz, D. S., Choi, S.-C. T., Wilkins-Diehr, N., Hong, N. C., Venters, C. C., Howison, J., … Littauer, R. (2016). Report on the Second Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE2). *Journal of Open Research Software*, *4*(1). http://doi.org/10.5334/jors.85

Lafferty, J. D., McCallum, A., & Pereira, F. C. N. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning* (pp. 282–289). San Francisco,

CA, USA: Morgan Kaufmann Publishers Inc. Retrieved from http://dl.acm.org/citation.cfm?id=645530.655813

McCullough, B. D., McGeary, K. A., & Harrison, T. D. (2006). Lessons from the JMCB Archive. *Journal of Money, Credit, and Banking*, *38*(4), 1093–1107.

McNee, S. M., Albert, I., Cosley, D., Gopalkrishnan, P., Lam, S. K., Rashid, A. M., … Riedl, J. (2002). On the Recommending of Citations for Research Papers. In *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work* (pp. 116–125). New York, NY, USA: ACM. http://doi.org/10.1145/587078.587096

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *arXiv:1310.4546 [cs, Stat]*. Retrieved from http://arxiv.org/abs/1310.4546

Miller, G. (2006). A Scientist's Nightmare: Software Problem Leads to Five Retractions. *Science*, *314*(5807), 1856–1857. http://doi.org/10.1126/science.314.5807.1856

Okazaki, N. (n.d.). *CRFsuite: a fast implementation of conditional random fields (CRFs)*. Retrieved from http://www.chokkan.org/software/crfsuite/

Pan, X., Yan, E., Wang, Q., & Hua, W. (2015). Assessing the impact of software on science: A bootstrapped learning of software entities in full-text papers. *Journal of Informetrics*, *9*(4), 860–871. http://doi.org/10.1016/j.joi.2015.07.012

Piwowar, H. A. (2011). Who Shares? Who Doesn't? Factors Associated with Openly Archiving Raw Research Data. *PLOS ONE*, *6*(7), e18657. http://doi.org/10.1371/journal.pone.0018657

Piwowar, H. A., & Chapman, W. (2008a). Linking database submissions to primary citations with PubMed Central (p. 4). Presented at the BioLINK Workshop at ISMB.

Piwowar, H. A., & Chapman, W. W. (2008b). Identifying data sharing in biomedical literature. *AMIA ... Annual Symposium Proceedings / AMIA Symposium. AMIA Symposium*, 596–600.

Piwowar, H. A., & Chapman, W. W. (2009). Using open access literature to guide full-text query formulation. Presented at the DBMI training retreat.

Piwowar, H. A., Day, R. S., & Fridsma, D. B. (2007). Sharing Detailed Research Data Is Associated with Increased Citation Rate. *PLOS ONE*, *2*(3), e308. http://doi.org/10.1371/journal.pone.0000308

Piwowar, H. A., & Priem, J. (2016). *Depsy: valuing the software that powers science*. Retrieved from https://github.com/Impactstory/depsy-research

Priem, J., Taraborelli, D., Groth, P., & Neylon, C. (2010). Altmetrics: A manifesto. Retrieved from http://www.citeulike.org/group/17557/article/11877310

Settles, B. (2004). Biomedical Named Entity Recognition Using Conditional Random Fields and Rich Feature Sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications* (pp. 104–107). Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from http://dl.acm.org/citation.cfm?id=1567594.1567618

Stodden, V. (2010). *The Scientific Method in Practice: Reproducibility in the Computational Sciences* (SSRN Scholarly Paper No. ID 1550193). Rochester, NY: Social Science Research Network. Retrieved from http://papers.ssrn.com/abstract=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–e67111. http://doi.org/10.1371/journal.pone.0067111

Tang, B., Cao, H., Wang, X., Chen, Q., Xu, H., Tang, B., … Xu, H. (2014). Evaluating Word Representation Features in Biomedical Named Entity Recognition Tasks, Evaluating Word Representation Features in Biomedical Named Entity Recognition Tasks. *BioMed Research International, BioMed Research International*, *2014, 2014*, e240403. http://doi.org/10.1155/2014/240403, 10.1155/2014/240403

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? In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing* (pp. 417–430). New York, NY, USA: ACM. http://doi.org/10.1145/2675133.2675172

Turian, J., Ratinov, L., & Bengio, Y. (2010). Word Representations: A Simple and General Method for Semi-supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (pp. 384–394). Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from http://dl.acm.org/citation.cfm?id=1858681.1858721

Wickham, H. (n.d.). Installed files · R packages. Retrieved August 4, 2016, from http://r-pkgs.had.co.nz/inst.html

Wilson, R. (2013, August 30). Encouraging citation of software – introducing CITATION files « Robin's Blog. Retrieved from http://blog.rtwilson.com/encouraging-citation-of-software-introducing-citation-files/

Appendix E: Information Products Appendix

This grant will generate four types of Information Products, each with their own dissemination needs:

Manually-derived dataset of Software Mentions (gold standard)

Description: Here is an example of a possible csv formatted serialization of our dataset. Note the connection of in-text citation with the details from the reference list.

> (10.234/1111, astropy, software_reuse, "We used astropy to do this", "")
>
> (10.234/1111, FooCountr3k, software_create, "We have made FooCountr3k available for download at", "Brown et al, 'FooCountr3k: a novel software for the counting of foo.' 2008, Foomatics, Volume 1 Number 1")
>
> (10.234/1111, astropy, software_reuse, "We the standard software package[1] to do this", "Brown et al, 'Astropy: its Debut. 2008, Astroinformatics Volume 1 Number 1")
>
> (10.234/2222, astropy, software_idle_mention, "There is such a thing as astropy", "")

Management, Dissemination, Archiving & Stewardship: We will publish our gold standard dataset on Zenodo using a CC-0 license. We will also explore donation to machine learning repositories such as the UCI repository (https://archive.ics.uci.edu/ml/datasets.html).

Automatically-derived dataset of Software Mentions

Description: The machine learning system will be trained and tuned on the manually-derived gold standard software mentions dataset described above. Once trained, we will run the machine learning system on millions of papers spanning three fields: biomedicine, economics, and astronomy (see Table below). The results of this automatic curation form an additional valuable information product: a huge dataset of automatically-identified software mentions, much larger than could ever be developed manually, and with higher precision and recall than any previous automatically-derived dataset of software mentions.

field	corpus name	approx number of documents	notes
Biomedicine	PubMed Central Open Access subset	400k	~2% of PubMed
Economics	Research Papers in Economics (RePEc)	2M	
Economics	SSRN	unpublished	We are have an existing relationship with CEO Gregg Gordon and are in discussions now
Astronomy	SAO/NASA Astronomy Data System (ADS)	11M	Includes non-astronomy papers we will need to remove

Management, Dissemination, Archiving & Stewardship: We will publish the automatically-derived dataset of Software Mentions on Zenodo using a CC-0 license. Although we expect the dataset to be well over 10 million rows, this is well within Zenodo's data size limits (2GB).

Prototype Services

Description: Three prototype software services will be developed to address the research questions: the CiteMeAs service (which automatically submits pull requests to GitHub repositories, as well as exposing a web service), the CiteSuggest web service, and a software mention module for Impactstory. All code will be written in Python.

Management and Dissemination: All software will be openly developed on GitHub under an MIT open source license. Each prototype software service will be made available for use on the open internet for the duration of the grant, as a proof-of-concept.

Archiving & Stewardship: The prototypes have no funding beyond the end of the grant, and so will cease to be available as operating web services on the internet once the grant funding ends. The fact these are limited-duration prototypes only will be clearly indicated on the web services themselves. However, the source code will be remain openly available on GitHub and the final release of each prototype will be permanently archived at Zenodo under an MIT open source license.

Scholarly publications

Description: A key output of the project will be scholarly publications summarizing our machine learning and socio-technical evaluation findings. These publications are subject to organization-wide OA policies at Impactstory (no related policies at the University of Texas).

Management: We will write our manuscripts as executable, reproducible papers, thereby linking text and results with supporting code and data. These knitr manuscripts will be openly available on GitHub throughout the writing process.

Dissemination, Archiving & Stewardship: All publications will be made open access immediately upon publication, either through publishing in an Open Access journal or through self-archiving in Zenodo, using a CC-BY license. All supporting statistical scripts, software, and datasets will be archived and made available in Dryad and Zenodo, through a CC-0 license (for data) and an MIT open source license (for scripts and software). We have included a budget line for open access fees.

Appendix F: Howison and Bullard software citation article.

Software in the Scientific Literature: Problems with Seeing, Finding, and Using Software Mentioned in the Biology Literature

James Howison
School of Information, University of Texas at Austin, 1616 Guadalupe Street, Austin, TX 78701, USA. E-mail: jhowison@ischool.utexas.edu

Julia Bullard
School of Information, University of Texas at Austin, 1616 Guadalupe Street, Austin, TX 78701, USA. E-mail: julia.a.bullard@gmail.com

Software is increasingly crucial to scholarship, yet the visibility and usefulness of software in the scientific record are in question. Just as with data, the visibility of software in publications is related to incentives to share software in reusable ways, and so promote efficient science. In this article, we examine software in publications through content analysis of a random sample of 90 biology articles. We develop a coding scheme to identify software "mentions" and classify them according to their characteristics and ability to realize the functions of citations. Overall, we find diverse and problematic practices: Only between 31% and 43% of mentions involve formal citations; informal mentions are very common, even in high impact factor journals and across different kinds of software. Software is frequently inaccessible (15%–29% of packages in any form; between 90% and 98% of specific versions; only between 24%–40% provide source code). Cites to publications are particularly poor at providing version information, whereas informal mentions are particularly poor at providing crediting information. We provide recommendations to improve the practice of software citation, highlighting recent nascent efforts. Software plays an increasingly great role in scientific practice; it deserves a clear and useful place in scholarly communication.

Introduction

Software is increasingly crucial to scholarship; it is a key component of our knowledge infrastructure (Edwards et al., 2013). Software underlies many scientific workflows and incorporates key scientific methods; increasingly, software is also key to work in humanities and the arts, indeed to work with data of all kinds (Borgman, Wallis, & Mayernik, 2012). Yet, the visibility of software in the scientific record is in question, leading to concerns, expressed in a series of National Science Foundation (NSF)- and National Institutes of Health–funded workshops, about the extent that scientists can understand and build upon existing scholarship (e.g., Katz et al., 2014; Stewart, Almes, & Wheeler, 2010). In particular, the questionable visibility of software is linked to concerns that the software underlying science is of questionable quality. These quality concerns are not just technical, but extend to the appropriateness of software for wide sharing, and its ability to facilitate the codevelopment that would make efficient use of limited scholarly funding (Howison & Herbsleb, 2013; Katz et al., 2014).

The link is two-fold: First, when software is not visible, it is often excluded from peer review; second, its lack of visibility, or the particular form of visibility, means that incentives to produce high-quality, widely shared, and codeveloped software may be lacking. A well-functioning system would assist not only the goals of understanding and transparency, but also the goals of aiding replication (Stodden et al., 2010), complementing the availability of publications such that "the second researcher will receive all the benefits of the first researcher's hard work" (King, 1995, p. 445).

The situation with software is broadly analogous (but not identical) to that of data in publications; indeed, all data are processed by software in some form (Borgman et al., 2012). Nonetheless, there are relevant differences. Accordingly, our inquiry into the visibility of software in scholarly communication is complementary to recent interest in data citation. In sum, then, the relationship of software to the scholarly

Received August 20, 2014; revised February 19, 2015; accepted February 20, 2015

 DOI: 10.1002/asi.23538

publication ought to be of key concern to those interested in scholarly communication, data in scholarship, and, indeed, the overall functioning of scholarship, knowledge infrastructures, and innovation.

In this article, we ask how software is currently visible in the literature and the extent to which this visibility contributes to achieving the normative ideals of scientific practice. Citations to a formal bibliography are important, yet formal citations are not the only form of visibility: Software is also visible in less-formal ways, including footnoted URLs to web pages maintained by software projects, parenthetical notes akin to those used for purchased scientific consumables, and simply discussed in the text in passing. Therefore, we write of software "mentions," intentionally choosing a word with casual and wide-ranging connotations, including the full spectrum of formal to informal visibility. While we were interested in cases where it was apparent that software was used, but not mentioned at all, such as statistical analyses, indeed some software authors claim this to be a very common problem (Howison & Herbsleb, 2011); but, for this study, we focused only on explicit mentions.

Specifically, we undertake a content analysis of a random sample of 90 journal articles from *Biology*, stratified by journal impact factor. We develop a reliable content analytic scheme to identify mentions of software in the literature and to understand how well these mentions achieve desirable functions, such as identification of an artifact, providing credit to its creators, and assisting others to build on the scholarship. We use this scheme to examine each identified software mention for its ability to realize these functions. Overall, we aim to provide a systematic motivation and basis for the pressing task of designing improved systems of visibility for software in the scientific literature.

Literature Review

Much of the foundational literature on scholarly citation examines the practice of citing, particularly the relationship indicated between scholarly publications (Cano, 1989; Lipetz, 1965; Moravcsik & Murugesan, 1975). Studies in the meaning of citation have attempted to clarify the possible relationships between citations and the works cited, providing typologies of credit giving (Moravcsik & Murugesan, 1975), associating the location of the citation with the type of credit given (Cano, 1989), and identifying the relevant element of the cited work (Lipetz, 1965). These have been used for automatic classification to identify relevant works (Pham & Hoffmann, 2003) and augment impact factor calculations (Teufel, Siddharthan, & Tidhar, 2006). In general, this scholarship is a practice literature that examines the nuances of an established practice to interpret these acts and improve our understanding of how science works or our information retrieval systems for science.

More recently, though, changes in publication technology have returned the discussion to other basic functions, such as identification and assistance in finding cited objects. Achieving these functions, long since addressed in standardized citation formats for print publications, require new methods for digital works. A familiar example of this trend is the citation of online works and the phenomenon of "link rot" (Klein et al., 2014; Koehler, 1999). To the extent that the location of online works is not fixed, citations cannot reliably facilitate access to cited works (Lawrence, 2001; Sellitto, 2005), undermining the verifiability and repeatability integral to the scientific method (Goh & Ng, 2007). As publication technology changed, the literature shifted back from studying the meaning of citations to addressing questions of design: How ought scholars reference other scholarly works?

This article thus continues the traditions of citation scholarship, seeking to contribute to both a literature of practice ("How do scientists mention software?") and a literature of design through assessment ("How well do the current practices do their job") leading to proposals for improvement ("How ought scientists mention software?"). Finally, we seek to raise, even if we cannot yet answer them, questions of change ("How best can the practices relevant to software visibility be altered?" and "How might proposed citation practices influence other areas of scientific conduct"?).

Data Citation

Design questions are at the heart of the literature on data citation, including how citations can provide identification of, location of, and access to, data, including data sharing, verification, and replicability (Mooney & Newton, 2012). Recently, the discussion has gained more urgency given the possibilities of data sharing online, the present "data deluge" of available data sets (Borgman et al., 2012), the possibilities of the linked data movement (Mayernik, 2012), and the adoption of data-sharing policies by granting agencies and journals (Borgman et al., 2012, p. 1060).

The practices of data citation and data sharing are intertwined; data sharing is motivated by the credit-giving apparatus of data citation, but data citation practices can only develop in a scholarly culture of data sharing (Mooney & Newton, 2012). The practice literature of data citation has examined how this dilemma is playing out in contemporary publications, finding that data citation is still an emergent practice, neither pervasive nor consistently applied (Simons, Visser, & Searle, 2013). Findings such as these have led scholars to call for cultural change in scholarly communication (Mayernik, 2012) and institutional mandates for data sharing (Simons et al., 2013).

Even if the need for citation of shared data was clear, the mechanisms are not yet so clear. Studies of the technical apparatus of data citation seek to identify the necessary criteria of adequate citation, such as specificity regarding the version and granularity of what is being cited (Borgman et al., 2012; Simons et al., 2013). In particular, scholars are concerned that data citation include the elements necessary to provide adequate identification and access to the data set (Altman & King, 2007; Konkiel, 2013).

DOI: 10.1002/asi

From these discussions of the necessary criteria for functional data citations, a design literature emerges that seeks to identify the criteria necessary to data citations, assesses to what extent these are used in contemporary practice, and proposes design improvements. Criteria include specificity regarding versions and granularity (Borgman et al., 2012; Simons et al., 2013) and findability supported by stable locators (Konkiel, 2013). Empirical studies of data citation in contemporary scholarship find that data citations tend to be minimal and incomplete when present at all (Mooney & Newton, 2012).

Suggestions to improve current practice include both cultural and technological changes. For example, technical proposals, such as digital object identifiers (DOIs) for data sets (Simons et al., 2013), as well as new citation standards (Altman & King, 2007; CODATA, 2013), will allow authors to cite in a way that supports the findability of data sets. Design improvements include integrating data citation counts into altmetrics to motivate data sharing (Konkiel, 2013).

Software Citation

Software citation requires both a practice and a design literature of its own. Software use and reuse are important for contemporary scientific methods and scholarly communication, and verifying, replicating, and building upon these studies requires adequate, consistently adopted modes of software citation. The small existing practice literature of software citation enumerates a number of challenges for meeting the criteria for credit and location. The barriers to software citation include all of those identified for data citation—such as difficulty with versioning and lack of citation standards—along with complications specific to this form. For example, Howison and Herbsleb (2013) report that the constant incremental improvements typical to software development are incongruent with structures of recognition and credit in academia. As with the chicken and egg dilemma in data citation identified by Mooney and Newton (2012), software citation suffers from a mismatch between the incentives for software development and sharing and science outcomes (Howison & Herbsleb, 2011). To the extent that software development is often proprietary rather than open, distribution models often run counter to the ideals of the "Republic of Science," endangering the verification and replication functions of citation (Gambardella & Hall, 2006; Ince, Hatton, & Graham-Cumming, 2012).

Some design improvements have been proposed. As with data citation, proposed solutions are both cultural and technological in nature; an example of a cultural change is the push toward adoption of permissive, open licenses for scientific software (Gambardella & Hall, 2006; Ince et al., 2012), whereas technological solutions include infrastructure for code sharing and metrics for software contributions (e.g., Goble, Roure, & Bechhofer, 2013; Katz, 2014; Stodden, Hurlin, & Perignon, 2012). We will return to suggestions for improvement in our discussion.

One mode of assessing both current practice and proposed solutions is to compare them against the criteria for citation identified earlier. Extending the criteria for data citation to software citation is appropriate given that the practices share technological challenges and relative novelty in scholarly communication. The practices are also intertwined: A full reference to data reuse requires mention of the software transformations applied to the set (Borgman et al., 2012, p. 1073). From these similarities and the foundational criteria from traditional citations, we identify the functions of crediting, identification (including versioning), and access (the ability to obtain the software). The requirement for identification, in the case of scientific software, also involves the configuration settings applied to the program—answering the question of which elements of the program were used.

Software also introduces some novel requirements for citations in order to support verification, replication, and building on others' work. Verification and replication, in the case of scientific software, requires not only the ability to locate the referenced material, but also access and permission to run the program. In particular, even special purpose descriptions of algorithms in articles have been found to be insufficient to replicate analyses; direct access to source code is vastly preferred (Ince et al., 2012; Stodden et al., 2010). Further, to build on others' work requires not just access to the source code, but also permission to extend the work, particularly to modify the program or combine it with other code in particular ways. As we will show, we develop these characteristics into a specific coding scheme.

Method

We identified a balanced and representative sample of the biology literature and undertook classic content analysis based on our development of two reliable content analytic schemes.

We chose to confine our analysis to a single domain, trading off broad scientific coverage against achieving a larger sample size. Biology is a leading domain for the importance of software in science, given the importance of computerized data analysis and the rise of bioinformatics. Some of the most well-cited papers of any kind in any science are biology software related (Science Watch, 2003). Because we are interested in contemporary practices, we confined our sample frame to articles published between 2000 and 2010 (the last complete year when we took the sample). Scientific attention is concentrated toward certain journals, albeit different journals in different fields and subfields; overall, the hierarchy of scientific journals forms a non-normal, exponential-like distribution, such as in Bradford's law (Bradford, 1934; Brookes, 1985). Such distributions are difficult to sample from: There is no "typical" item in such a distribution. It would be problematic to only study widely read ("top") journals, but equally problematic to study only less-well-read journals. Accordingly, we sought to study a sample balanced for overall coverage and likely influence.

DOI: 10.1002/asi

We identified a set of 18 biology-related subject headings in biology using the 2010 Institute for Scientific Information (ISI) Web of Science (WoS). We took all of the 1,455 journals included in these headings and sorted them by their journal impact factor. Previous research has found differences in practices between higher and lower impact factors (e.g., Stodden, Guo, & Ma, 2013), and journal impact factor seemed an appropriate proxy for overall influence or breadth of readership. Though there are many criticisms of journal impact factor, particularly for assessing influence of specific articles or authors, the journal unit of analysis is well suited for our study given that the policies of journals seem likely to affect the form of articles. Thus, we divided our journal list into three groups: The first group of journals included those ranked 1 through 10 (10 journals), the second had those ranked 11–110 (100 journals), and the third had the rest of those ranked 111–1,455 (1,345 journals). We combined the journals with strings for the years (2000–2010) and weeks (1–52) to yield a sampling frame that covered each of the journals across the whole time period. We then randomly selected 90 journal-year-week tuples for each strata. We worked through this list taking the first 30 issues listed that appeared to be from journals that publish original research, as opposed to review journals.

We manually retrieved the issue from the journal website that was current in the year and week number. When an issue was dated during or after the chosen week, we chose the issue that came out before that week. We found two journals in the sample that we did not have library access to and discarded these, taking the next journal-year-week tuple. We also found 12 tuples that were before the first published volume of the journal (e.g., we sought a 2001 article from a journal that began publishing in 2006); in those cases, we discarded that tuple and used the next from the list of 90, rather than taking the first issue of the journal on the basis that first issues might be systematically different.

We assessed the content of the chosen issue, identifying research articles (as opposed to letters, editorials, perspectives, review/survey articles, and other publications, such as "plant registrations"). In two cases, where our chosen issue did not have any research articles, we went to the issue immediately following. From the research articles in the selected issue, we used a random number generator to choose one article. We continued this process until we had 30 research articles from each strata, a total data set of 90 biology research articles, as shown in Table 1.

We obtained portable document formats (PDFs) of the articles and of any supplemental materials (these were often "methods and materials" online supplements with their own text and references lists). During coding, we found one article that was not a biology article (it was a pure mathematics article) and we replaced it with an article derived from the next tuple in the original random selection for that strata. Appendix A includes a full list of categories in our sample frame and journals in our sample; Table 2 shows a distribution of articles from well-known journals in the top strata by impact factor (we did not intend to only choose articles from 5 of the top 10; that was simply a result of the method of randomization).

Our random selection of articles enables us to use our sample to make estimates about software mentions in the overall biology literature, because undertaking random sampling means it is reasonable to believe that sampling errors resulting from our specific sample are normally distributed. Accordingly, we are able to present 95% confidence intervals (CIs), for the population around the characteristics of the sample we report, providing upper and lower bounds for the results we report in the population of biology articles. These estimates treat each mention as independent, not adjusting for the reality that ways of mentioning software may be influenced by authors and journals (i.e., within articles). This is not ideal, but given that authors are not necessarily consistent (even within articles) and, more importantly, readers read widely across journals and articles by different authors, readers are going to encounter many varying ways of mentioning software, even if there is some consistency within specific journals or authors. We conducted the statistics with the R functions, prop.test and chisq.test (based on Hope, 1968; Newcombe, 1998). The data set and full analysis scripts are available at http://github.com/jameshowison/softcite/.

In the analysis to follow, we present results both in aggregate and, in some cases, broken out by journal impact factor strata. In many cases, our statistical analysis shows no statistically significant differences between strata, but we do not rely on those results for our main conclusions. Indeed, the contribution of this article is toward informing policy making and prompting the emergence of a design literature for software mentions in scientific articles; in that context, it is unclear that any specific size of difference (effect size) between strata would matter, and without that, it is hard to estimate the statistical power needed for reliable between-strata comparisons.

TABLE 1. Summary of sample and sample frame.

	Strata 1	Strata 2	Strata 3
Journals in sample frame	10	100	1,345
Articles in sample	30	30	30
Journals in sample	5	23	30

TABLE 2. Numbers of articles included from strata 1 journals

Journal name	Article count
Science	7
Nature	5
Cell	7
Nature Biotechnology	5
Nature Genetics	5

DOI: 10.1002/asi

Coding Scheme Development

Our coding scheme development proceeded in three rounds: identifying software mentions; coding their characteristics; and coding their functions. In each case, we developed our coding scheme by iterating between reading the text of the articles and the existing literature described earlier.

Identifying software mentions. In round 1, we analyzed the full text of the articles to identify mentions of software within an article. We were exhaustive in seeking locations of possible mentions, including not only the main text of the article, but also table and figure captions, reference list, and supplemental materials. We considered coding for situations where it was apparent that software was used, but not mentioned at all, such as when an article presents statistics or figures, but with no mention of the software almost definitely used to create them. Unfortunately, whereas this would be very interesting, we concluded that this would be too speculative and difficult to achieve reliability in coding; accordingly, we confined our coding to identifying explicit mentions of software.

We tested reliability of our ability to recognize software mentions by having two coders independently code subsets of articles and then comparing their coding. Reporting agreement is complicated in this case because the coding units are not predefined; rather, the coders are picking them out from the text of the articles; these are thematic coding units that may be whole paragraphs, sentences, or phrases. Coders are thus only identifying units they think mention software, not identifying units they think do not. Further, software mentions are sparse in the data set. In this sense, using agreement statistics on, say, a sentence level would substantially inflate agreement owing to the many sentences coded as not mentioning software. Given the sparseness of the thematic units, it is also not necessary to adjust for the very unlikely case of chance agreement, and therefore we report straight percentage agreement (and not, say, Cohen's kappa), calculated using the "irr" package for the R statistics program (Gamer, Lemon, Singh, & Fellows, 2012). We tested the reliability in this way twice: once at the beginning of coding and once when we trained a new coder.

The first test included 12 articles in the subsample. Both coders agreed that there were no software mentions in 7 of the 12 articles. In the remaining five articles, coders achieved percentage agreement of 68.2%. We identified the reasons for disagreement in discussion and resolved them with coding rules (e.g., sentences with two citations for one software package should be coded as two mentions). The most complex source of disagreement revolved around whether a sentence referred to a piece of software or the abstract scientific model; we discussed rubric to determine the difference, including brief online searching.

The second test occurred when we trained a third coder, using a new subsample of eight articles. There was agreement by both coders that six articles contained no software mentions. Agreement in the two remaining articles was 83.3%, with a single instance where one coder failed to identify a mention; on inspection, we ascribed this to coder fatigue and not conceptual disagreement. The high agreement in this second round of training provides confidence that the issues discussed in the first round were adequately resolved.

Software mention characteristics. Our second coding scheme identified characteristics of software mentions. These codes are shown in Table 3. We tested the reliability of this scheme by applying them to the mentions coded in the 12-article subsample discussed earlier; this set included 32 mentions drawn from the five articles that mentioned software. Because this coding involved applying codes to a preagreed set of mentions, we report intercoder reliability using Cohen's kappa. Specifically, we use "Byrt's kappa" because it adjusts for unbalanced prevalence (i.e., when one value, negative or positive, is rarely used) (Byrt, Bishop, & Carlin, 1993).

Owing to the fact that many mentions come as in-text citations with references in the bibliography, we linked the in-text citation and the reference in the data set. We then applied codes to each element separately. For references, we used the additional codes shown in Table 4, but, for comparison in reporting purposes, we treat a citation + reference pair as a single mention, which has all of the codes applied to either element. For example, if one mention included a creator name in text, whereas another included the creator name in the reference, this distinction is retained in the data

TABLE 3. Coding scheme for mentions of software.

Code	Definition	Agreement (kappa)
Software name	The name of the software package	k = 1
URL	A web address for the software or project	k = 1
Version number	A version number (or source code label) identifying a specific version of the software	k = 1
Date	A date used to indicate a version of the software (not date of article or reference)	k = 1
Configuration details	Any mention of configuration of the software	k = 0.75
Software used	For mentions of software that was used in the research	k = 0.875
Software not used	For mentions of software that the authors did not use (e.g., they discuss why they did not use particular software, or the method realized in the software)	k = 1
Creator	A mention of the creator of the software (could be applied to in text mention or reference)	k = 1

DOI: 10.1002/asi

TABLE 4. Additional codes for references in software mentions.

Software publication	Formal publication primarily describing software
Domain publication	Formal publication primarily describing mainline domain science
Users guide/manual	Project documentation, typically online but not published in a journal/conference proceeding or similar
Project name	Reference with just project name
Project page	Reference to URL of project

TABLE 5. Codes for functions.

Code	Explanation
Identifiable	Can we identify which software has been mentioned (e.g., Is there a name used at all, beyond "a program we wrote?" Can we find references to that software, even if we cannot find the software itself?)?
Findable	Given an identifiable piece of software, can we find an online source that details the software (not necessarily the software itself, but any official presence) (e.g., a project page or online manual)?
Findable version	Can we find the specific version listed in the article, if there was one?
Access	Can we access the software now? Can take three values: No Access, Purchase Access, Free Access.
Source available	Can we access the source code in any way?
Permission to modify	Do the creators give permission to modify the program (if no mention of modification, assume no)?; if permission only by contact, then no.
Matches preferred citation	If the project page lists a preferred citation, does the mention match it?

set, but, in the analysis reported on in this article, both would be reported as a single mention that included a creator name.

We standardized the software names by clustering the raw names using Jaro-Winkler distance, as implemented by the R stringdist package (Van der Loo, 2014), and manually inspecting the clusters (e.g., standardizing "Image J" and "ImageJ" and components of a single package, such as BLAST, BLASTP, BLASTN, and so on).

Functions of software mentions. In the third round, we coded to assess the extent to which the mention performed the functions of citation identified earlier (e.g., location, credit-giving; see Table 5 for full set of codes and explanations). We were generous in seeking relevant information across the full article when assessing the functions of citations. That is, we combined all the information supplied across all mentions of a piece of software in the article in order to find the software. Once we had sufficient identifying information, we went outside the article text and used web searching to attempt to locate the software and assess the possibility of access (for reproducibility), access type (free or for purchase), source code availability (for transparency), and ability to modify the code (for building on the work of others).

Examples of software mentions with codes. From the article "The seasonal phenology of Bactrocera tryoni (Froggatt) (Diptera: Tephritidae) in Queensland" in the *Australian Journal of Entomology*, we identified this sentence:

> The DYMEX model we used was as described and parameterised by Yonow et al. (2004).

Which we coded as follows:

> An in-text mention to software used by the authors, with a reference. The software name was "DYMEX"; there were no configuration details (in the focal text) and no version number, date, or URL given. The reference was coded as a domain publication that cited a creator (the authors of the reference). The software was identifiable and a web search showed it to be findable. It is accessible in that it is available for purchase. The source code is not available and there is no permission to modify the code. The project does not make a request for a specific citation.

From the article "Insights into assembly from structural analysis of bacteriophage PRD1" in *Nature*, we identified this mention:

> Data were analysed with DENZO [41] and the resolution limit was determined with TRIM_DENZO (D.I.S., unpublished program).

Which was coded as follows:

> Two software mentions, one for "DENZO" (with a reference) and one for "TRIM_DENZO." Both were coded as software used by the authors; neither included version numbers, configuration details, dates, or URLs. Both were coded as providing creator information (For TRIM_DENZO, the initials D.I.S. match the author's initials, the reference provides creator information for DENZO). DENZO was found to be identifiable and findable, but there was no access to the software (which also implies no source code or permission to modify). TRIM_DENZO was coded as identifiable but unfindable (implying no source access or permission to modify).

From the article "Yeast Cbk1 and Mob2 activate daughter-specific genetic programs to induce asymmetric cell fates" in *Cell*, we identified this sentence as mentioning software:

> We captured and analyzed images using a SPOT2e CCD camera (Diagnostic Instruments, Inc., Sterling Heights, MI) coupled to MetaMorph imaging software (Universal Imaging Corporation, Downingtown, PA).

Which was coded as follows:

> This was coded as a software mention of software used by the authors. The software name was "MetaMorph." There were no

DOI: 10.1002/asi

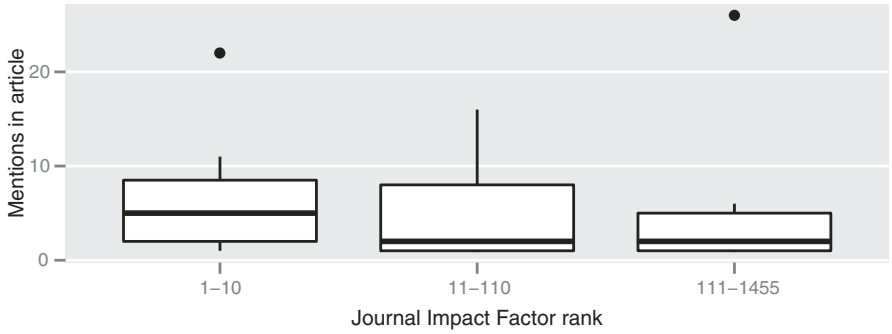

FIG. 1. Counts of mentions in articles, broken down by impact factor strata.

TABLE 6. Types of software mentions in publications.

Mention type	Count ($n = 286$)	Proportion (95% CI)	Example
Cite to publication	105	0.37 (0.31–0.43)	. . . *was calculated using biosys (Swofford & Selander 1981).*
Cite to users manual	6	0.02 (0.01–0.05)	. . . *as analyzed by the BIAevaluation software (Biacore, 1997).* Reference List has: Biacore, I. (1997). BIAevaluation Software Handbook, version 3.0 (Uppsala, Sweden: Biacore, Inc)
Cite to project name or website	15	0.05 (0.03–0.09)	. . . *using the program Autodecay version 4.0.29 PPC (Eriksson 1998).* Reference List has: *ERIKSSON, T. 1998. Autodecay, vers. 4.0.29 Stockholm: Department of Botany.*
Instrument-like	53	0.19 (0.14–0.24)	. . . *calculated by t-test using the Prism 3.0 software (GraphPad Software, San Diego, CA, USA).*
URL in text	13	0.05 (0.03–0.08)	. . . *freely available from* http://www.cibiv.at/software/pda/.
In-text name mention only	90	0.31 (0.26–0.37)	. . . *were analyzed using MapQTL (4.0) software.*
Not even name mentioned	4	0.01 (0.00–0.04)	. . . *was carried out using software implemented in the Java programming language.*

configuration details and no URL, version_number, or date, but the mention included a creator ("Universal Imaging Corporation, Downingtown, PA"). The software to be identifiable and findable. Access was possible through purchase, but the source was unavailable and modifications were prohibited.

Results

Overview

From the 90 articles total, we identified 59 that mentioned software and 31 that did not (65% of articles mentioned software). In our sample, articles in higher impact factor strata were more likely to mention software (77% in strata 1, 63% in strata 2, and only 57% in strata 3). In total, we found 286 distinct mentions in the 59 articles that mentioned software. The distribution of mentions across articles is shown in Figure 1; most articles that mentioned software had relatively few mentions. The two articles with the highest number of software mentions have over 20 mentions. We retained these within our data set.

Classification of mentions. We classified references according to the scheme previously described. In our sample, the mentions range in form quite widely. Only 44% of software mentions involve an entry in a references list, with only 37% being a citation to a formal publication (another 7% are informal entries in a reference list, including either the name or website of the project). Of the 56% of mentions that do not include references, 31% mention only the name of the project. Another 18% mention software in a manner similar to scientific instruments or materials, typically mentioning the name in text followed by the author or company and a location in parentheses. Finally, some 5% of mentions provide a URL in the text or in a footnote and 1% mention using some software, but provide no additional details. Our categorizations, with examples, are shown in Table 6 and Figure 2, where we provide 95% CIs for the likely proportion of these types of mentions in the population of biology articles.

These categories of mentions are useful for understanding the overall diversity of practice, but somewhat fine-grained for further analysis. Accordingly, we collapsed these categories into three: Cite to publication (including cite to user manual); Like instrument; and Other (including Cite to name or website, URL in text, Name only, or Not even name). These categories correspond well to two formalized forms of mentioning in the literature and a collection of informal techniques that scientists are using. The results are shown in Figure 3.

DOI: 10.1002/asi

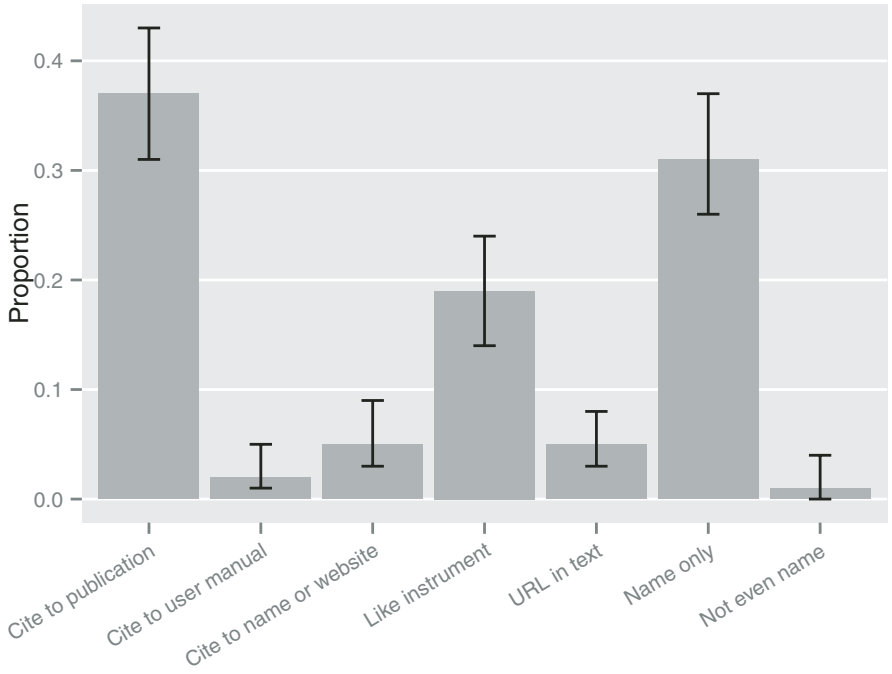

FIG. 2. Types of software mentions. Errorbars show 95% CIs.

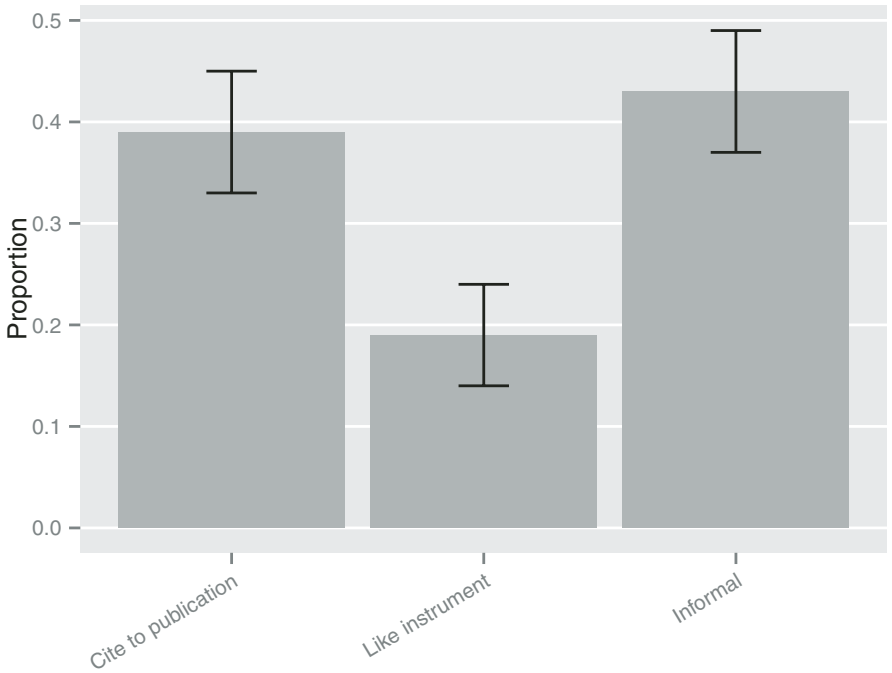

FIG. 3. Classification of software mentions (collapsed categories).

Using these categories, 39% (95% CI: 0.33–0.45) of mentions cite a publication, 19% (95% CI: 0.14–0.24) refer to software following the guidelines for instruments, and 43% (95% CI: 0.37–0.49) use some form of other, informal way of mentioning software.

Figure 4 shows these categories of mentions broken out by strata. Whereas there are no differences in the use of cites to publications, we can see that there are significantly fewer mentions that look like instruments in the low journal impact strata. The data tend to show higher use of

DOI: 10.1002/asi

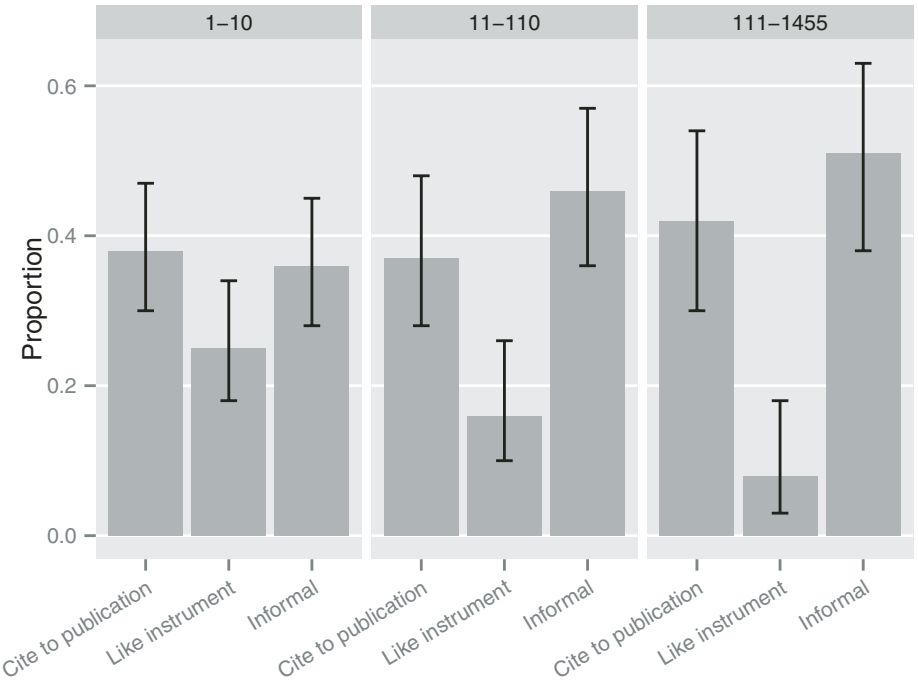

FIG. 4. Major software mention types by journal strata. Error bars show 95% CIs.

informal citations in lower journal impact strata, but the 95% CIs overlap. These results are consistent with the idea that journals in higher strata have more formalized mention styles; nonetheless, even in the top strata alone, 36% (95% CI: 0.29–0.44) of mentions were informal (categorized as "Other").

Characteristics of software mentioned. The mentions we found were to 146 distinct pieces of software. The majority of pieces were only mentioned in a single article, with the most mentioned software being mentioned in only four articles. We provide the full list of software mentioned in articles in Appendix B, but given the broad distribution of software in the literature, our sample size does not allow us to claim representativeness sufficient to create a "league table" of software use in science; we include the Appendices to help readers assess the face validity of our content analysis results.

We classified the type of software using the codes described earlier (the result of seeking the software online, using data provided with any mention within an article): whether the software was accessible; whether one has to pay money for a license; whether the source code is available; and whether the software provides explicit permission to modify and extend the source code (i.e., a free software or open source license). As illustrated in Figure 5, we were able to access only 79% (95% CI: 0.71–0.85) of the software mentioned. Forty-seven percent of the software mentioned was available without payment (95% CI: 0.39–0.56), whereas only 32% had source code available (95% CI: 0.24–0.40) and only 20% gave explicit permission for others to modify or extend the source code (95% CI: 0.14–0.27).

The characteristics of software are important results because they reflect the usefulness of software to other scientists, but they do not provide intuitive labels to discuss types of software. Accordingly, we combine these categories to produce intuitive labels. The first is "Not accessible." The second is for software that must be paid for and for which the source code is held as a proprietary secret; these we call "Proprietary" (rather than "Commercial," emphasizing the unavailability of source code). At the other end, we place "Open source" software that is available without payment, provides access to the source code, and provides explicit permission to modify the code. Falling between is the "Noncommercial" software category for software available without payment, but that does not provide explicit permission to modify the code; most, but not all, provide access to source code. This includes many packages written by scientists and made available for other scientists, but either without specifying license conditions or specifying licenses that restrict modification. As illustrated in Figure 6, we found 21% of software to be Not accessible (95% CI: 0.15–0.29), 32% to be Proprietary (95% CI: 0.24–0.40), 27% to be Noncommercial (95% CI: 0.21–0.36), and 20% to be Open source (95% CI: 0.14–0.27).

Our classification of software and mention types enables us to explore whether particular types of software are referred to in different ways. For example, it seems reasonable that Proprietary software would be more likely to be mentioned using the Like instrument style, given that it is less likely to have a publication associated with it and was purchased perhaps from the same budgets as equipment. Figure 7 shows the relationship between types of software and types of mentions, which is statistically significant

DOI: 10.1002/asi

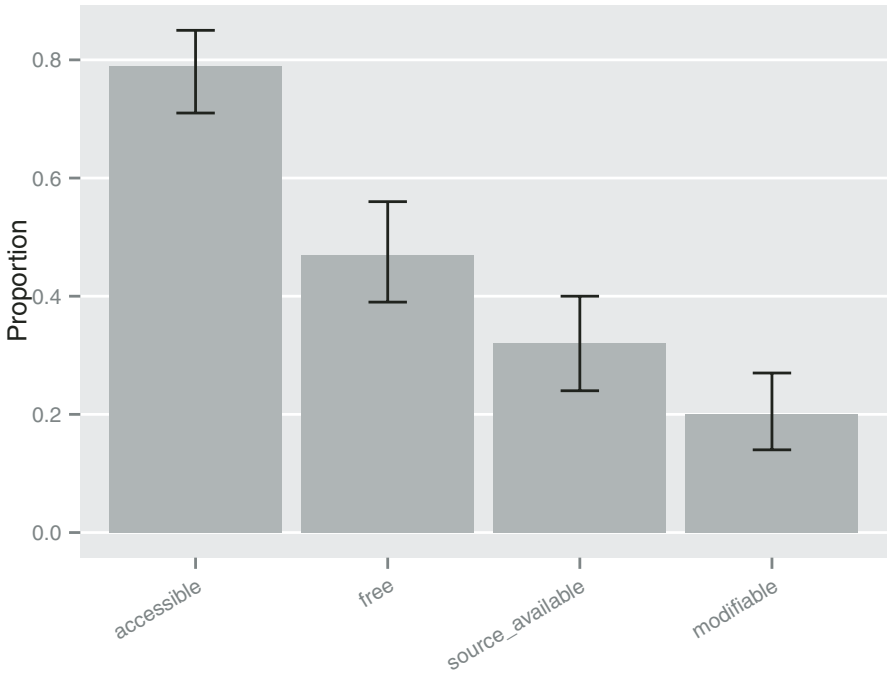

FIG. 5. Characteristics of mentioned software.

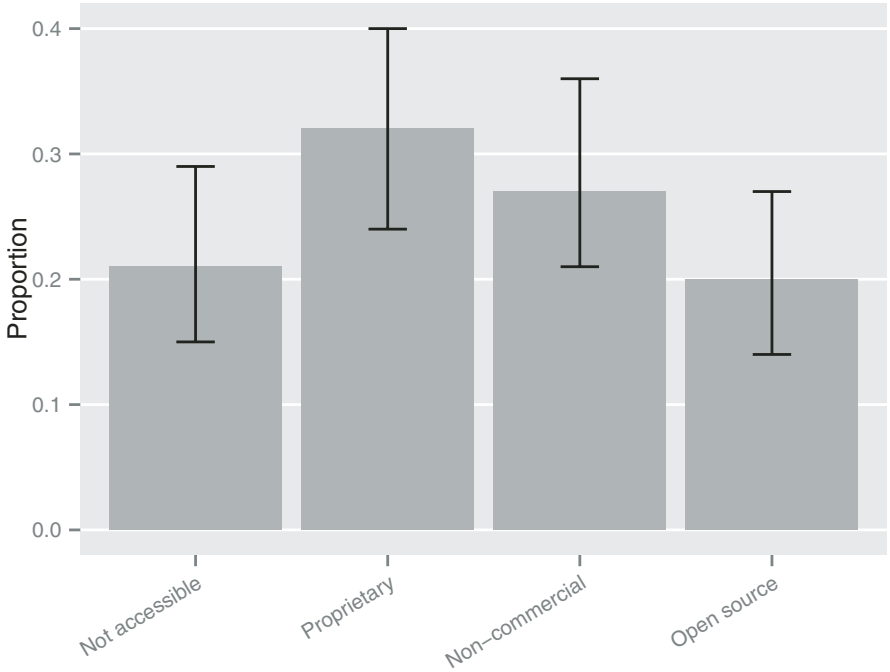

FIG. 6. Types of software mentioned.

($\chi^2(6, N = 274) = 49.248$, $p < .05$). Indeed, Proprietary software is far more likely to be mentioned using the Like instrument style than other kinds of software; 35% of mentions of proprietary software use the Like instrument style (95% CI: 0.26–0.46), whereas the Like instrument style was used for less than 10% of mentions of Noncommercial and Open source software. Similarly, there is greater use of the Cite to publication style in our Noncommercial and Open source categories, understandable given that many of these packages are written by scientists for scientists and include a citable article. Yet, the clearest takeaway from this analysis is that there is still considerable diversity in styles; even for

DOI: 10.1002/asi

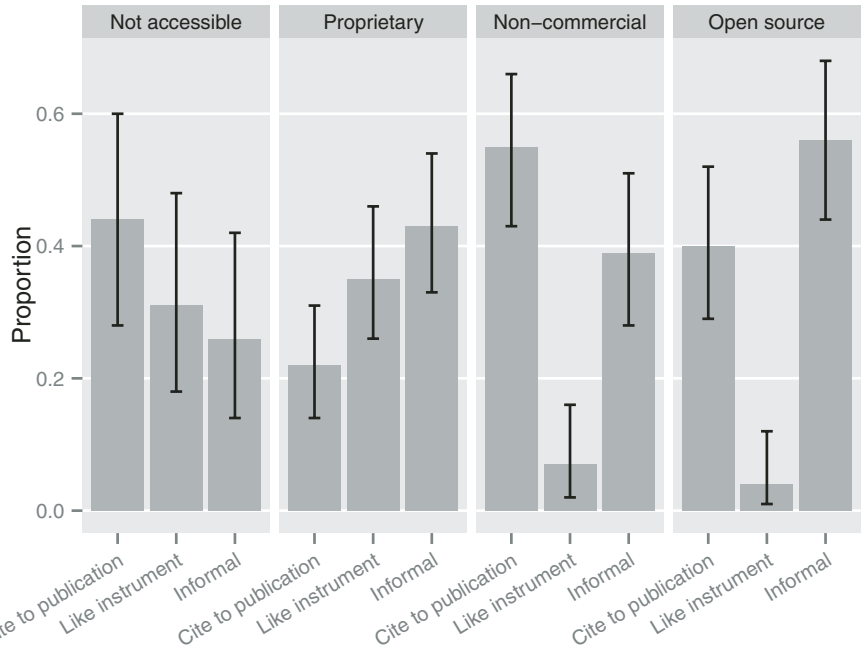

FIG. 7. Relationship between software types and mention types.

Proprietary software, use of informal Other style is statistically indistinguishable from the use of Like instrument style and for the other categories of software the Other style is statistically indistinguishable from the use of the Cite to publication style.

Citation Functions

Identifying and finding software. We assessed our data set to see whether the mentions gave sufficient information for identifying and finding software. We also assessed how well authors do in providing credit to the authors of software. Owing to the fact that pieces of software are mentioned in multiple articles, our data set for this section is larger than the overall number of pieces of software; a single piece of software could be mentioned in a functional way in one article, but without the same functionality in another. Accordingly, there are 182 unique combinations of software and articles. As shown in Figure 8, overall, 93% of the software was identifiable (95% CI: 0.88–0.96) and 86% provided enough information for us to find the software online (95% CI: 0.80–0.90), however, this means that at least 1 in 10 software packages mentioned in articles are simply not providing sufficient information to find the software package. Some 77% (95% CI: 0.70–0.83) provided some information about the creators of the packages, meaning that one in five did not.

Information on specific versions was much less frequently provided. Overall, only 28% provided version information (95% CI: 0.22–0.35). Yet, many of those projects did not provide access to earlier versions, meaning that only in 5% of cases (10 actual combinations of articles and software) were we able to find the specific versions of software mentioned in articles (95% CI: 0.03–0.10).

As shown in Figure 9, there were essentially no significant differences in these functions across strata.

We sought to understand our findings in more detail by examining whether different ways of mentioning software were more likely to perform each function of citation. We illustrate this in Figure 10. For the basic functions of identification and providing the ability to find the software, our data show no statistically significant differences. This analysis, however, does show that mentions that are cites to publications are much less likely to include version data (only 11% do; 95% CI: 0.06–0.20). Similarly, the issues with not providing any credit information appear to be almost entirely driven by informal mentions: Only 43% do (95% CI: 0.31–0.56), whereas both the Like instrument and Cite to publication categories all provide at least some credit information.

Discussion

The evidence presented in this article clearly shows that the practices of mentioning software are diverse, with substantial problems in achieving the functions of citation. It seems that scientists are addressing software primarily by analogy with other elements that appear in publications, sometimes treating software as though it were an instrument or material commercially purchased, sometimes as akin to a scientific protocol, sometimes treating software as a pair with a published article, and sometimes simply including whatever is at hand, from user manuals to URLs and the names of projects.

DOI: 10.1002/asi

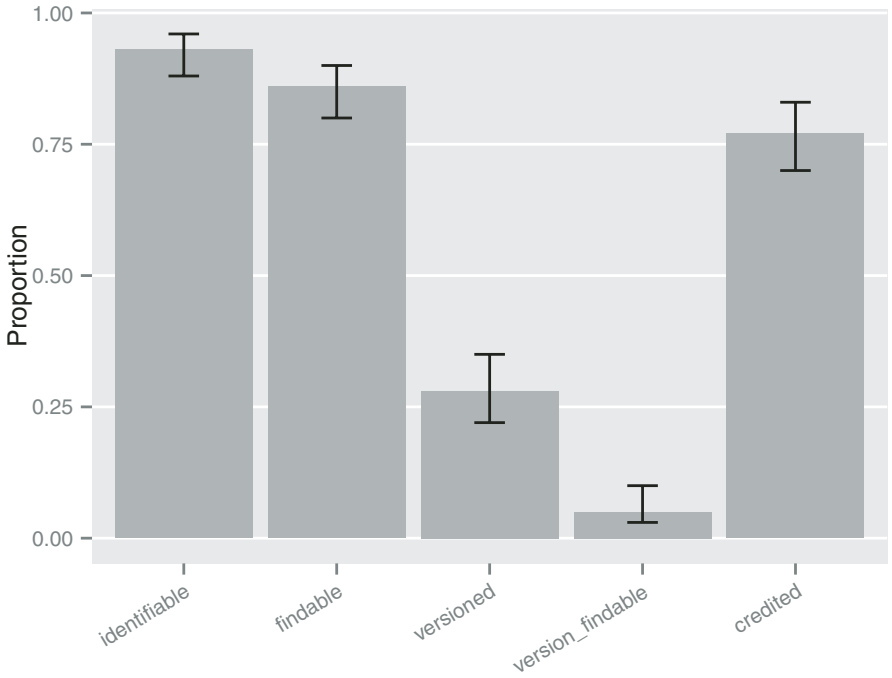

FIG. 8. Functions of citation.

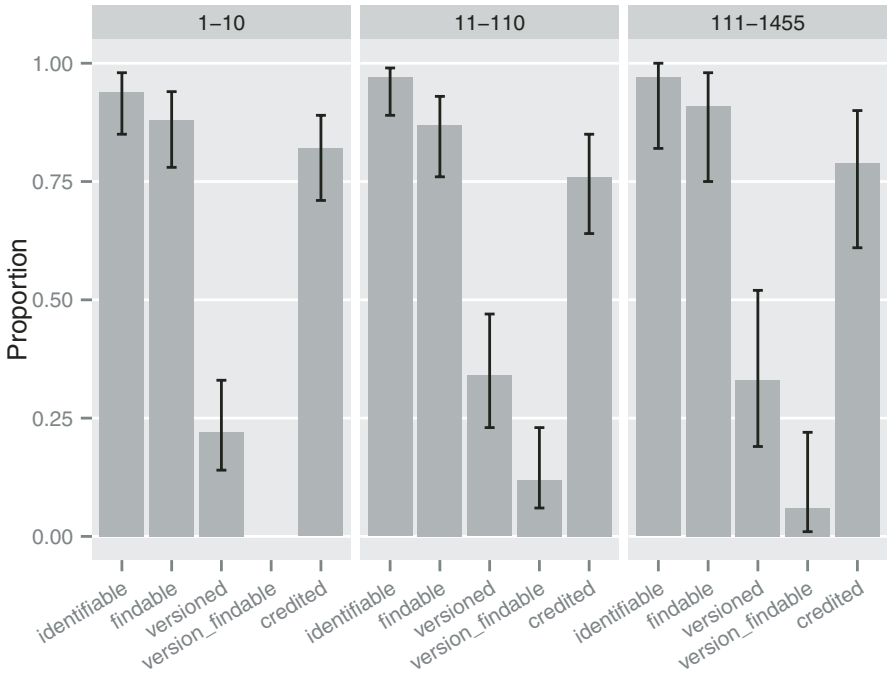

FIG. 9. Functions of citation by strata.

These diverse ways of mentioning software are, from a scholarly communications perspective, certainly better than nothing, but often fail to accomplish many of the functions of citation.

Whereas almost all mentions allow for identification of the software discussed, only between 80% and 90% provide sufficient information to find that software (meaning 1 in 10 software packages could not be found). Yet, software, unlike almost all articles, typically changes over time, the ability to find a particular version is more important, and only between 22% and 35% of software mentions provide that information; moreover, in only between 2% and 10% of cases can that specific version be found.

DOI: 10.1002/asi

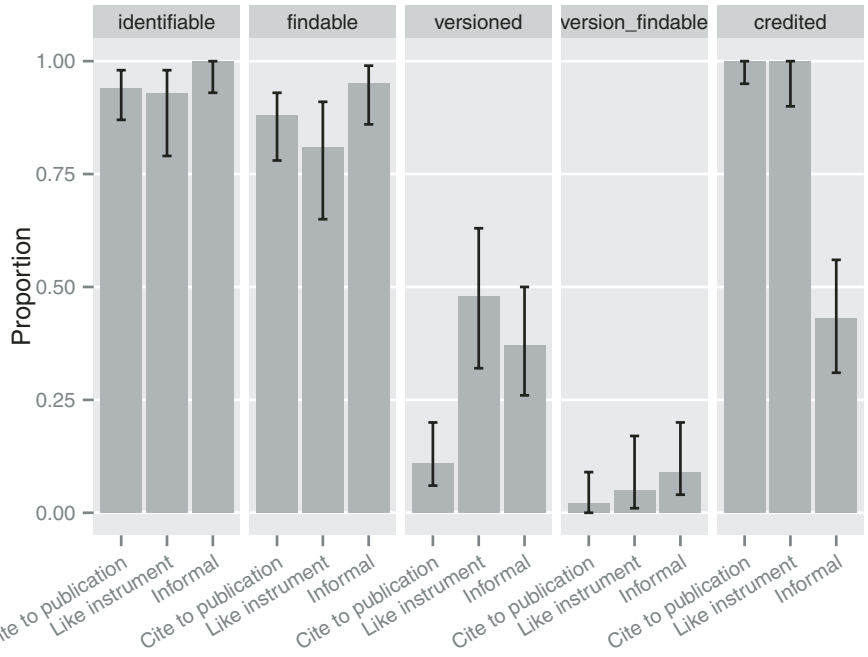

FIG. 10. Functions of citation by mention category.

Turning to the second, but no less important, function of providing credit for scientific contribution, and thus rewarding the effort required to build reusable software, we find that between 70% and 83% of mentions attempt to give credit in some form, primarily through reference to accompanying publications or parenthetical mentions of authors or companies.

As we move further up the list of attributes necessary for reproducibility and for efficient innovation through building on the work of others, the situation worsens even further. Only between 71% and 85% of software is available, whereas only between 24% and 40% of the software mentioned is available in source form, facilitating inspection by those interested in replicating the research. Finally, only between 14% and 27% of the software mentioned provides the most basic condition for extension: permission to reuse and/or modify the software provided.

What Is to Be Done?

Improving the situation presented in this article requires action across a number of domains of scientific practice, both in design and then in driving change. Certainly, one area is to design and standardize improved forms for describing software use in scientific articles, reducing the acceptability of using the variety of informal forms of mentioning software. Improved standards should tackle the functions of identification and findability (including at the level of specific versions) as well as giving credit in a manner that motivates excellent software work. Yet, moving beyond those basic functions requires change not in how articles are written, but in how software is made available, changes that have to occur outside the process of writing articles, at the projects that build software.

In this section, we move code by code, considering the causes of the issues, potential solutions, techniques to encourage uptake of the solutions, and describing "green shoots" indicating progress in these areas.

Improving identification and findability. The most basic function of mentioning software in an article is to allow readers, including reviewers, to identify and locate the software used. This function is directly analogous to the ability to identify and find a specific publication, or the ability to identify and find a specific material or instrument. In the case of software, which, unlike a typical publication, continues to change after its initial release, this also involves specific version numbers. Whereas we do not have specific data on authors' intentions, the fact that they mentioned the software at all indicates that the problem in this area appears not to be motivation, but a lack of clear standards and norms for mentioning software. The way forward, then, seems fairly straightforward: First, we need clear and consistent practices for citing software, and second, we need to disseminate, encourage, and enforce their use.

We are, of course, not the first to make this point. Indeed, many citation style guides offer forms for citing software, including the American Psychological Association (APA). Recent efforts in this space include work analogous to data citation, such as that undertaken by DataOne (Mayernik, 2012) and the ESIP organization (Earth Sciences Information Partner, 2012). For software, a promising way

DOI: 10.1002/asi

to incorporate version information is to link directly to the source code repositories that development teams use to track their development, as well as automating the creation of a DOI or other Handles. Systems with this approach have been developed at CERN (Purcell, 2014) and by the Mozilla Science Project, Github, and Figshare (http://mozillascience.github.io/code-research-object/).

The way forward here clearly involves journals and conferences adopting specific forms of citation and enforcing them as a condition of publication. We examined the "instructions to authors" for the journals in our sample, and found that only 24% had specific policies on citing software. Unsurprisingly, journals in higher stratas seemed more likely to have such policies (three of five journals in the first strata (60%), 10 of 23 in the second strata (43%), and 1 of 30 in the third strata (3%), with strata 3 showing statistically significant differences from strata 1 and strata 2 ($p = .005$). It may be that with clearer standards that are more broadly expected by authors, reviewers, editors, and readers that journals' provision of relevant policies will improve. On the other hand, it may be appropriate to build systems that automatically check the form of software citations, ensuring that they follow the required styles and that they correctly resolve to a specific version in a repository.

Improving crediting. Authors appear committed to providing information about the origins of software. As discussed, for authors seeking to make scientific contributions, credit is vital; it may be less so for those selling their software. Yet, some forms of mentions offer more potential than others; as we saw, the absence of crediting information is driven almost entirely by the incidence of informal mentions. Ironically, the Like instrument citation form preferentially used with commercial software (see Figure 7) (and thus less likely to be driven by a need for credit) is relatively effective in ascribing credit, at least to the level of the company selling the software.

Similarly, the Cite to publication form definitely encourages the inclusion of crediting information; yet the reuse of publication style citations may undermine the usefulness of these mentions or actually produce undesirable results. At first, cites to publications would seem to most directly enable contributors to demonstrate their scientific impact, reusing existing bibliographic analysis systems. Yet, using citations to articles can run counter to the need to identify and find the software itself, particularly because the publication citations remain static while software changes, including changing name. Further, however, these citations can "fix" the contributor list at a particular time, creating a disincentive for later potential participants to contribute their changes to a project and contributing to the tendency for scientific software to "fork" (Howison & Herbsleb, 2013). Finally, because software is typically constructed by integrating code of others, it is not clear that simply citing the authors of the package used actually credits those who have provided the functionality; indeed, a desire to be recognized might encourage authors of software to avoid having their code integrated.

Thus, there is a need for a form of crediting that identifies and rewards contributors in a manner most useful to them and least likely to undermine desirable collaboration and integration. The proposals discussed, linking to software repositories, offer advantages in this area, potentially facilitating tracing contribution to specific versions by post-hoc examination of commits and their authorship in the source code repository. Katz (2014) addresses the question of integration by suggesting a system of indirect credit, dividing citation credit accruing to top-level projects between their developers and the developers of the components they draw on. Other approaches take an altmetrics approach and focus not on the appearance of code in publications, but on metrics such as downloads or use, including analysis of traces, such as downloads and analysis of workflow repositories (e.g., McConahy, Eisenbraun, Howison, Herbsleb, & Sliz, 2012; McLennan & Kennell, 2010; Piwowar & Priem, 2013; Stodden et al., 2012).

One approach achievable in the short term is for projects themselves to specify the manner in which they would like to be mentioned; with journals or styles providing "fallback" guidelines to be used when the software does not. Some of the projects in our sample indeed did this, providing "preferred citations," which were themselves a mix of citations to domain and software articles and forms with corporate authorships (e.g., "The R project Team"). Most of these requests were contained on the home page of the project or, in a few cases, in a "splash screen" or other part of the software interface. We recorded whether a project made such a request and coded, at the article level, whether authors appeared to follow the request. We found that only 27 of our 146 software packages (18%; 95% CI: 0.13–0.30) made a specific request to be mentioned in some way. These packages were mentioned in 15 articles, resulting in 31 combinations of these packages and articles. We found 21 cases where the requested citation was used (68%, across 11 articles; 95% CI: 0.49–0.83), leaving 10 cases where the request was not followed (32%, occurring across eight articles; 95% CI: 0.17–0.51). We can only speculate, but this may be a combination of not being aware of the request, publishers' style guides, or simple inattention on the author's behalf. Certainly, the paucity of specific requests for citation, combined with their inconsistent usage, suggests that measuring the research impact of software solely by searching for specific citations has serious validity concerns.

One possibility to improve the situation is for authors to make correct acknowledgment a requirement of permission to use the software; all but one of the examples we observed were phrased as requests and not as requirements. In our interviews and discussions with producers of scientific software (Howison & Herbsleb, 2011, 2013), authors hesitate to make such requirements, both in fear of

DOI: 10.1002/asi

losing users and in the belief that such requirements violate principles of scientific sharing. There is precedent for using licenses (and thus contract law) to require specific acknowledgments within the domain of open source software and open cultural production, although such requirements are controversial. The GNU GPL and the Apache license requires software users to retain all attribution notices in the code, and the original Berkeley Software Distribution (BSD) license required acknowledgment of the University of California; the Open Source Initiative approves licenses requiring attribution, such as the "Common Public Attribution License" used for the code behind Reddit (Wilson, 2008). All Creative Commons licenses require attribution (other than the Public Domain Dedication, CC0) as a condition of use, and the project provides guidelines on appropriate forms of attribution, including tools to automate attributions (see Creative Commons, 2014). Nonetheless, as with any system, end users may not follow the license; indeed, in our data set, one package used a license that required users to cite a specific article, but the mention of that software in our data set did not.

Finally, it seems likely that any standards should address the question of whether to handle commercial software differently from software written for academic credit. The prevalence of Instrument like citations suggests that authors see software as similar to other equipment; this may be appropriate, especially if those writing the software are merely interested in selling software and not in earning academic reputation. However, a standard that differentiated in this way would need to help authors know when to use which form, and our suggestion of packages themselves providing this information seems pertinent.

Improving accessibility. We found that 21% of software packages in our sample simply could not be accessed; at 95% CI, this suggests that between 16% and 28% of software mentioned in publications is unavailable. One approach for improving the availability of software associated with an article is to require that it be deposited with the publication itself. This approach often combines a requirement for depositing data and analysis code, sometimes in the form of "workflows" (e.g., Goble et al., 2013; Roure et al., 2009; Stodden et al., 2012) or perhaps "virtual machines" replicating the entire analysis execution environment. An extension of this approach is the "executable article" (Strijkers et al., 2011), which calls for bundling all the data and software needed to produce the results and the article, right through to plots and, ultimately, the article PDF. These are promising approaches, which avoid the reproducibility issue from incomplete software and workflow descriptions demonstrated by failed attempts at replication by Ince et al. (2012), and they have been quite successful in some fields; an increasing number of journals and conferences have these requirements. Yet, as with citation standards, such repositories have compliance, monitoring, and maintenance issues, as described in *Econometrics* by McCullough, McGeary, and Harrison (2006). The *Journal of Money, Banking, and Finance* has had a data and software repository for many years; yet, an attempt to use the contents of the repository for replication showed that only 69 of the 193 articles that should have had entries actually did, and the authors were only able to use code to successfully replicate the analysis in 14 cases. Clearly, a policy is only as good as its enforcement.

In fact, much of the question of accessibility depends not on the actions of authors of articles that use the software, but on the behavior of a much larger group, including software component producers and intermediaries, such as software publishers and repositories. This is particularly true when one seeks to access source code and integrate or modify it. Accordingly, a series of workshops and publications have argued that nothing less than software that is developed and made available as fully open source software is sufficient for the aims of science policy (Ince et al., 2012; Katz et al., 2014). This means choosing and using a specific open source software license and committing to continually making software available through public repositories. Just as in data advocates for openness have reasoned "public money, public data," so, too, comes advocacy for "public money, public code." The arguments for openness, however, need to interact with requirements for software sustainability over time. In some cases, openness and sustainability are well aligned, as with well-executed open source projects. If, however, the project chooses to pursue sustainability through commercial sales, then the situation is more complex. For example, some code of great usefulness to scientists is supported by sales to the commercial market, in effect cross-subsidizing scientific use and making greater resources available to science. Blanket policies, such as "public money, public code," preclude models like this. Nonetheless, hybrid models are possible, such as is common with MATLAB code: a for-profit, closed source engine, but a great deal of open sharing of analysis code.

Conclusion and Future Research

We have examined the manner in which software is mentioned in scientific articles, and we conclude that the practices are varied and appear relatively ad hoc. It is not too surprising, then, that we also find that the way that software is mentioned, and the way that it is made accessible to users of the scientific literature, fails to accomplish many of the intended functions of citations in scholarly communication. Certainly, it is clear that studies of software in publications, or efforts to assess the impact of software through bibliometrics, must look beyond formal citations or reference lists given that the data in this article provide evidence that these, at least in the biology literature, constitute only between 31% and 43% of software mentions.

DOI: 10.1002/asi

There are a great number of interesting research questions that ought to be pursued. Certainly, efforts are needed in the design and testing of improved software citation approaches and standards. These efforts need to assess potential influence on collaboration. For example, how does the reuse of the publication system through "software articles" as requested citations influence the willingness of future developers to cooperate? How might a software citation system acknowledge the many contributors to software dependencies on which user-facing components are built (providing indirect credit)? Can scholarly articles bear the sheer amount of citations that such a system would call for? Further, we know little about how scientists reason about what ought to be cited and how they make these decisions; in particular, we know almost nothing about when scientists choose not to mention software they have used at all and we know little about how to influence scientists toward new practices.

Software is both similar and different to other elements mentioned in scientific papers: It is at once an artifact, an instrument, a protocol, sometimes a publication, and the focus of ongoing activity. In short, software is both an artifact and a practice. This varied nature renders the question of how software ought to be mentioned in publications surprisingly complex. Yet, it also provides an opportunity: Addressing the issues reported in this article would go a great distance to improving the efficacy of both scholarly communications and scientific practice.

Acknowledgments

The full data set and analysis code for this article is available online at https://github.com/jameshowison/softcite/. This version was commit 1c37d938f1349cdd72ab7586ceb398e1e8372ab7 or tag vR2.1.

The graphs in this article were created using ggplot2 software (Wickham, 2009), version 1.0.0, running in the R statistics environment (R Development Core Team, 2009), version 3.1.1. Data storage and manipulation was done with the Apache Jena software (version 2.11.1) (https://jena.apache.org/, written by the Apache Jena team, https://jena.apache.org/about_jena/team.html) and the spin framework (version 1.4.0) (http://spinrdf.org/spin.html, written by Holger Knublauch), supported by the Hamcrest Library (version 1.3) and JUnit (version 4.11) (credit information for both at https://github.com/junit-team/junit/blob/master/acknowledgements.txt). Jena and R were linked using the rrdf library (Willighagen, 2013), version 2.0.2. Additional data manipulation used the dplyr (version 0.2.0.99) and reshape2 (version 1.4) R libraries, both written by Hadley Wickham.

We would like to thank Catherine Grady for her assistance with content analysis.

This material is based upon work supported by the NSF under Grant No. SMA-1064209.

References

Altman, M., & King, G. (2007). A proposed standard for the scholarly citation of quantitative data. D-Lib Magazine, 13(3/4).

Borgman, C.L., Wallis, J.C., & Mayernik, M.S. (2012). Who's got the data? Interdependencies in science and technology collaborations. Computer Supported Cooperative Work (CSCW), 21(6), 485–523.

Bradford, S.C. (1934). Sources of information on specific subjects. Engineering, 137, 85–86.

Brookes, B.C. (1985). "Sources of information on specific subjects" by S.C. Bradford. Journal of Information Science, 10(4), 173–175.

Byrt, T., Bishop, J., & Carlin, J.B. (1993). Bias, prevalence and kappa. Journal of Clinical Epidemiology, 46(5), 423–429.

Cano, V. (1989). Citation behavior: Classification, utility, and location. Journal of the American Society for Information Science, 40(4), 284–290.

CODATA. (2013). Out of cite, out of mind: The current state of practice, policy, and technology for the citation of data. Data Science Journal, 12(September), CIDCR1–CIDCR75.

Creative Commons. (2014). Best practices for attribution—CC Wiki. Retrieved from https://wiki.creativecommons.org/Best_practices_for_attribution

Earth Sciences Information Partner. (2012). Data citation guidelines for data providers and archives. *ESIP Working Document*. doi: 10.7269/P34F1NNJ

Edwards, P.N., Jackson, S.J., Chalmers, M.K., Bowker, G.C., Borgman, C.L., Ribes, D., . . . Calvert, S. (2013). Knowledge infrastructures: Intellectual frameworks and research challenges. doi: 2027.42/97552

Gambardella, A., & Hall, B.H. (2006). Proprietary versus public domain licensing of software and research products. Research Policy, 35(6), 875–892.

Gamer, M., Lemon, J., Singh, P., & Fellows, I. (2012). irr: Various coefficients of interrater reliability and agreement. Retrieved from http://CRAN.R-project.org/package=irr

Goble, C., Roure, D.D., & Bechhofer, S. (2013). Accelerating scientists' knowledge turns. In A. Fred, J.L.G. Dietz, K. Liu, & J. Filipe (Eds.), Knowledge discovery, knowledge engineering and knowledge management (pp. 3–25). New York, NY, USA: Springer Berlin Heidelberg. doi:10.1007/978-3-642-37186-8_1

Goh, D., & Ng, P. (2007). Link decay in leading information science journals. Journal of the American Society for Information Science and Technology, 58(2002), 15–24.

Hope, A.C. (1968). A simplified Monte Carlo significance test procedure. Journal of the Royal Statistical Society: Series B (Methodological), 30(3), 582–598.

Howison, J., & Herbsleb, J.D. (2011). Scientific software production: Incentives and collaboration. In Proceedings of the ACM Conference on Computer Supported Cooperative Work (pp. 513–522). Hangzhou, China March 19–23. doi: 10.1145/1958824.1958904

Howison, J., & Herbsleb, J.D. (2013). Incentives and integration in scientific software production. In Proceedings of the ACM Conference on Computer Supported Cooperative Work (pp. 459–470). San Antonio, TX: February 23–27. doi: 10.1145/2441776.2441828

Ince, D.C., Hatton, L., & Graham-Cumming, J. (2012). The case for open computer programs. Nature, 482(7386), 485–488.

Katz, D.S. (2014). Transitive credit as a means to address social and technological concerns stemming from citation and attribution of digital products. Journal of Open Research Software, 2(1), e20.

Katz, D.S., Choi, S.-C.T., Lapp, H., Maheshwari, K., Löffler, F., Turk, M., . . . Venters, C. (2014). Summary of the first workshop on sustainable software for science: Practice and experiences (WSSSPE1). Journal of Open Research Software, 2(1).

King, G. (1995). Replication, replication. Political Science and Politics, 28, 444–452.

Klein, M., Van de Sompel, H., Sanderson, R., Shankar, H., Balakireva, L., Zhou, K., & Tobin, R. (2014). Scholarly context not found: One in five articles suffers from reference rot. PLoS ONE, 9(12), e115253.

Koehler, W. (1999). An analysis of web page and web site constancy and permanence. Journal of the American Society for Information Science, 50(2), 162–180.

Konkiel, S. (2013). Tracking citations and altmetrics for research data: Challenges and opportunities. Bulletin of the American Society for Information Science and Technology, 39(6), 27–32.

Lawrence, S. (2001). Free online availability substantially increases a paper's impact Nature, 411(6837), 521–521. doi: 10.1038/35079151

Lipetz, B. (1965). Improvement of the selectivity of citation indexes to science literature through inclusion of citation relationship indicators. American Documentation, 16(2), 81–90.

Mayernik, M.S. (2012). Data citation initiatives and issues. Bulletin of the American Society for Information Science and Technology, 38(5), 23–28.

McConahy, A., Eisenbraun, B., Howison, J., Herbsleb, J.D., & Sliz, P. (2012). Techniques for monitoring runtime architectures of socio-technical ecosystems. In Workshop on Data-Intensive Collaboration in Science and Engineering (CSCW 2012), February 11–12, Seattle, WA.

McCullough, B.D., McGeary, K.A., & Harrison, T.D. (2006). Lessons from the JMCB archive. Journal of Money, Credit, and Banking, 38(4), 1093–1107.

McLennan, M., & Kennell, R. (2010). HUBzero: A platform for dissemination and collaboration in computational science and engineering. Computing in Science and Engineering, 12(2), 48–53.

Mooney, H., & Newton, M. (2012). The anatomy of a data citation: Discovery, reuse, and credit. Journal of Librarianship and Scholarly Communication, 1(1), 1–6.

Moravcsik, M.J., & Murugesan, P. (1975). Some results on the function and quality of citations. Social Studies of Science, 5(1), 86–92. doi: 10.2307/284557

Newcombe, R.G. (1998). Interval estimation for the difference between independent proportions: Comparison of eleven methods. Statistics in Medicine, 17(8), 873–890.

Pham, S., & Hoffmann, A. (2003). A new approach for scientific citation classification using cue phrases. AI 2003: Advances in Artificial Intelligence. doi: 10.1007/978-3-540-24581-0_65

Piwowar, H., & Priem, J. (2013). The power of altmetrics on a CV. Bulletin of the American Society for Information Science and Technology, 39(4), 10–13.

Purcell, A. (2014, March 5). Tool developed at CERN makes software citation easier. International science grid this week. Retrieved from http://www.isgtw.org/spotlight/tool-developed-cern-makes-software-citation-easier

R Development Core Team. (2009). R: A language and environment for statistical computing. Vienna: R Foundation for Statistical Computing.

Roure, D.D., Goble, C., Aleksejevs, S., Bechhofer, S., Bhagat, J., Cruickshank, D., . . . Poschen, M. (2009). Towards open science: The myExperiment approach. Concurrency and computation: Practice and experience, 22(17), 2335–2353.

Science Watch. (2003). Twenty years of citation superstars. Science Watch, 14(5).

Sellitto, C. (2005). The impact of impermanent web-located citations: A study of 123 scholarly conference publications. Journal of the American Society for Information Science and Technology, 56(7), 695–703.

Simons, N., Visser, K., & Searle, S. (2013). Growing institutional support for data citation: Results of a partnership between Griffith University and the Australian National Data Service. D-Lib Magazine, 19(11/12).

Stewart, C.A., Almes, G.T., & Wheeler, B.C. (2010). NSF cyberinfrastructure software sustainability and reusability workshop report. doi: 2022/6701

Stodden, V., Donoho, D., Fomel, S., Friedlander, M., Gerstein, M., LeVeque, R., . . . Wiggins, C. (2010). Reproducible research. Computing in Science and Engineering, 12(5), 8–13.

Stodden, V., Hurlin, C., & Perignon, C. (2012). RunMyCode.org: A novel dissemination and collaboration platform for executing published computational results. In 2012 IEEE 8th International Conference on E-Science (e-Science) (pp. 1–8). doi: 10.1109/eScience.2012.6404455

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.

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.

Teufel, S., Siddharthan, A., & Tidhar, D. (2006). Automatic classification of citation function. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (pp. 103–110). Sydney, Australia, July 22–23. doi: 10.3115/1610075.1610091

Van der Loo, M.P.J. (2014). The stringdist package for approximate string matching. The R Journal, 6(1), 111–122.

Wickham, H. (2009). ggplot2: Elegant graphics for data analysis. New York, NY, USA: Springer.

Willighagen, E. (2013). Accessing biological data with semantic web technologies. *Peer J Pre-Prints*. doi: 10.7287/peerj.preprints.185v1

Wilson, R. (2008). Common public attribution license—An overview. Retrieved from http://oss-watch.ac.uk/resources/cpal

Appendix A: Details of Sampling Frame and Journals in Sample

The sampling frame included all journals in the 2010 edition of the ISI WoS, using the *Journal Citation Reports* tool. We included these categories:

TABLE 7. All categories in sample.

Biochemistry & Molecular Biology	Biology
Biotechnology & Applied Microbiology	Cell Biology
Developmental Biology	Entomology
Evolutionary Biology	Genetics & Heredity
Marine & Freshwater Biology	Mathematical & Computational Biology
Microbiology	Multidisciplinary Sciences
Mycology	Ornithology
Parasitology	Plant Sciences
Reproductive Biology	Zoology

TABLE 8. All journals in sample.

1–10	11–110	111–1,455
Nature Genetics	*Nucleic Acids Research*	*Applied Biochemistry and Biotechnology*
Science	*Nature Cell Biology*	*BMC Plant Biology*
Nature Biotechnology	*Molecular Systems Biology*	*Academie des Sciences. Comptes Rendus. Biologies*
Cell	*Molecular Ecology*	*American Journal of Botany*
Nature	*The FASEB Journal*	*Israel Journal of Plant Sciences*
	Genome Research	*Advances in Complex Systems*
	Molecular Therapy	*Biochimica et Biophysica Acta. Proteins and Proteomics*
	Nature Structural and Molecular Biology	*Journal of Molecular Neuroscience*
	Developmental Cell	*BMC Molecular Biology*
	Cladistics	*Turkish Journal of Biochemistry*
	The Plant Journal	*Phytomedicine*
	Systematic Biology	*Molecular Diagnosis and Therapy*
	Acta Crystallographica. Section D: Biological Crystallography	*Zoological Studies*
	Human Molecular Genetics	*Journal of Molecular Catalysis B: Enzymatic*
	Stem Cells	*Australian Journal of Entomology*
	Nanomedicine	*Journal of Computer—Aided Molecular Design*
	New Phytologist	*Waterbirds*
	Cell Research	*The Journal of Parasitology*
	PLoS Biology	*Acta Parasitologica*
	National Academy of Sciences. Proceedings	*Biochimica et Biophysica Acta. General Subjects*
	The Journal of Infectious Diseases	*Journal of Thermal Biology*
	The Journal of Cell Biology	*Protoplasma*
	Molecular Psychiatry	*Aquatic Ecosystem Health & Management*
		Turkish Journal of Zoology
		Arthropod Structure & Development
		Cytotechnology
		Undersea & Hyperbaric Medicine
		Systematic Botany
		Nucleosides, Nucleotides and Nucleic Acids
		Journal of Integrative Plant Biology

DOI: 10.1002/asi

Appendix B: Software Packages Mentioned in Articles

TABLE 9. Software packages mentioned in sample.

CCP4	4	LSM510	1
ClustalW	4	LSQKAB	1
Excel	4	MacClade	1
PAUP	4	MapMaker	1
Adobe Photoshop	3	MapQTL	1
BLAST	3	MATLAB	1
HKL	3	Mfold	1
ImageJ	3	Minitab	1
MetaMorph	3	MitoProt	1
NIH Image	3	MOLREP	1
O	3	MOLSCRIPT	1
SPSS	3	MorphoCode	1
CNS	2	MrBayes	1
ModelTest	2	NeuroZoom	1
R	2	NormFinder	1
REFMAC	2	NTSYS-pc	1
SAS	2	Opticon Monitor 2	1
SOLVE	2	OPUS	1
Stereo Investigator	2	PC-ORD	1
Treeview	2	PHASE	1
Adobe INDesign CS	1	PHASER	1
Agilent 2100 Expert Software	1	Phred/Phrap/Consed	1
AMoRe	1	PHYLIP	1
AMOVA	1	PHYML	1
Autodecay	1	PONDR	1
BeadStudio	1	POST	1
BIAevaluation	1	PREDATOR	1
BioDataFit	1	Prism	1
BioEdit	1	PROCHECK	1
BioNJ	1	PSORT	1
BIOSYS	1	qBasePlus	1
BLAT	1	QUANTA	1
BOXSHADE	1	Quantity One	1
cactus online smiles translator	1	RACE	1
CAD	1	RASTER3D	1
Calcusyn	1	RESOLVE	1
CALPHA	1	RIBBONS	1
Chart 5	1	SCALEPACK	1
CHIMERA	1	SCAMP	1
ChipViewer	1	Sedfit	1
Cluster	1	Sednterp	1
COLLAPSE	1	Sequence Navigator	1
COOT	1	SHELLSCALE	1
DatLab	1	SHP	1
DENZO	1	SIGMAA	1
DYMEX®	1	Sigmaplot	1
EIGENSTRAT	1	Software for Zeiss LSM 510	1
Ensembl	1	software-Unknown-a2003-22-CR_BIOL-C01-mention	1
EnzFitter	1	software-Unknown-a2003-44-SCIENCE-C09-mention	1
EPMR	1	software-Unknown-a2003-44-SCIENCE-C10-mention	1
ESCET	1	software-Unknown-a2006-05-SYST_BIOL-C05-mention	1
GAP	1	software-Unknown-a2006-05-SYST_BIOL-C08-mention	1
GDE	1	software-Unknown-a2006-47-SYST_BIOL-C02-mention	1
Gelworks 1D Advanced	1	software-Unknown-a2007-11-GENOME_RES-C09-mention	1
GenePix	1	software-Unknown-a2008-06-NAT_GENET-C04-mention	1
GENESPRING	1	Staden	1
Genome Analyser II	1	STATA	1
geNorm	1	Statistica	1
GoMiner	1	Statview	1
Grafit	1	Swiss-Model	1
Graph Pad Prizm	1	SYSTAT	1
GraphPad Prism	1	TargetP	1
GRASP	1	TIMAT2	1
GRID	1	TMHMM	1
GRIN	1	TRIM_DENZO	1
IDEG6	1	tRNAScan-SE	1
Jalview	1	TRUNCATE	1
JAZZ	1	Useq	1
JMP(R)	1	WinNONLIN	1
jMRUI	1	X-PLOR	1
Kodak Digital Science 1D	1	X-Score	1
KS300	1	XPREP	1
limma R package	1	Zeiss LSM Image Browser	1

DOI: 10.1002/asi