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

 • Published online 13 May 2015 in Wiley Online Library (wileyonlinelibrary.com). 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

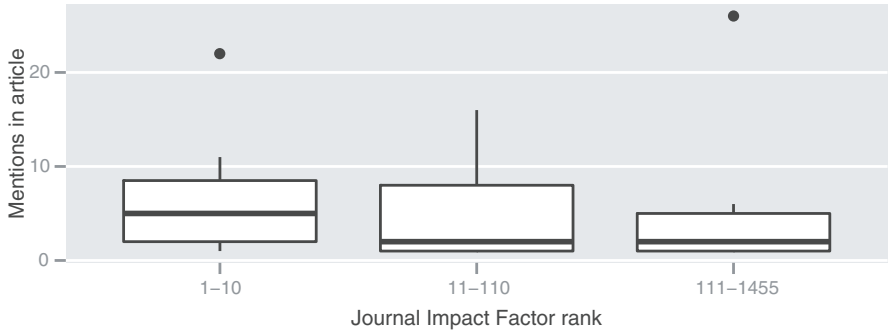

FIG. 1. Counts of mentions in articles, broken down by impact factor strata.

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

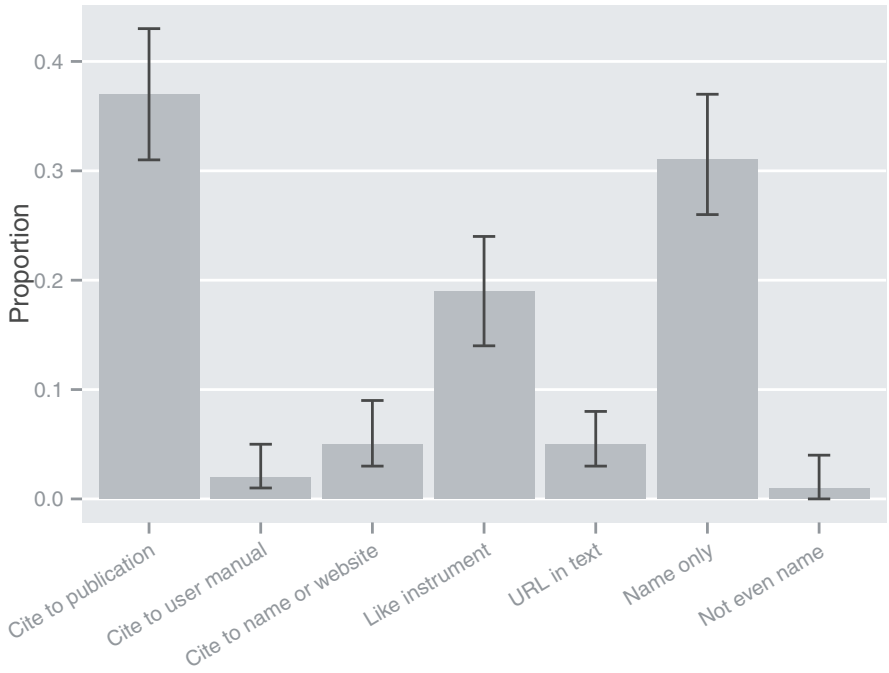

FIG. 2. Types of software mentions. Errorbars show 95% CIs.

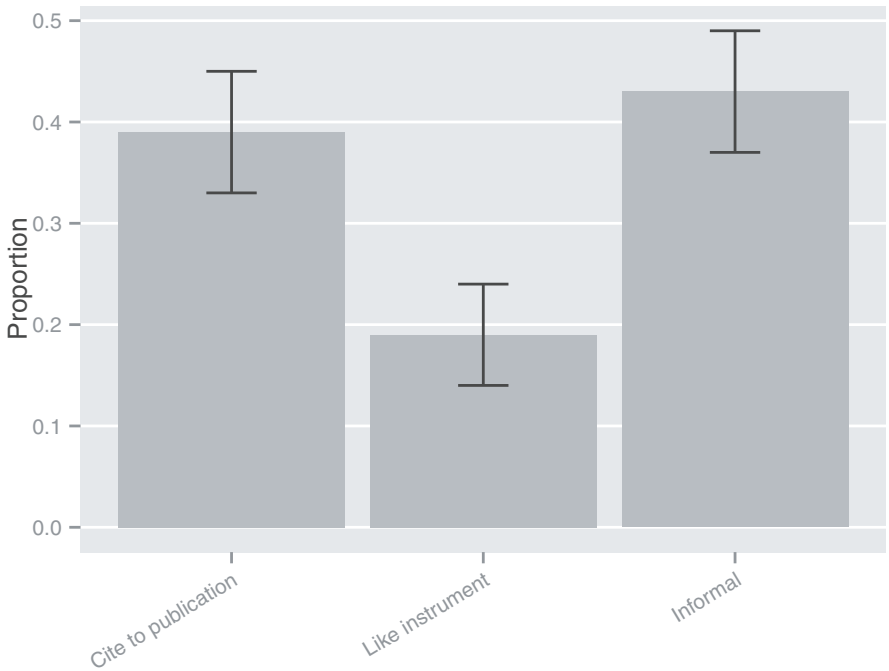

FIG. 3. Classification of software mentions (collapsed categories).

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

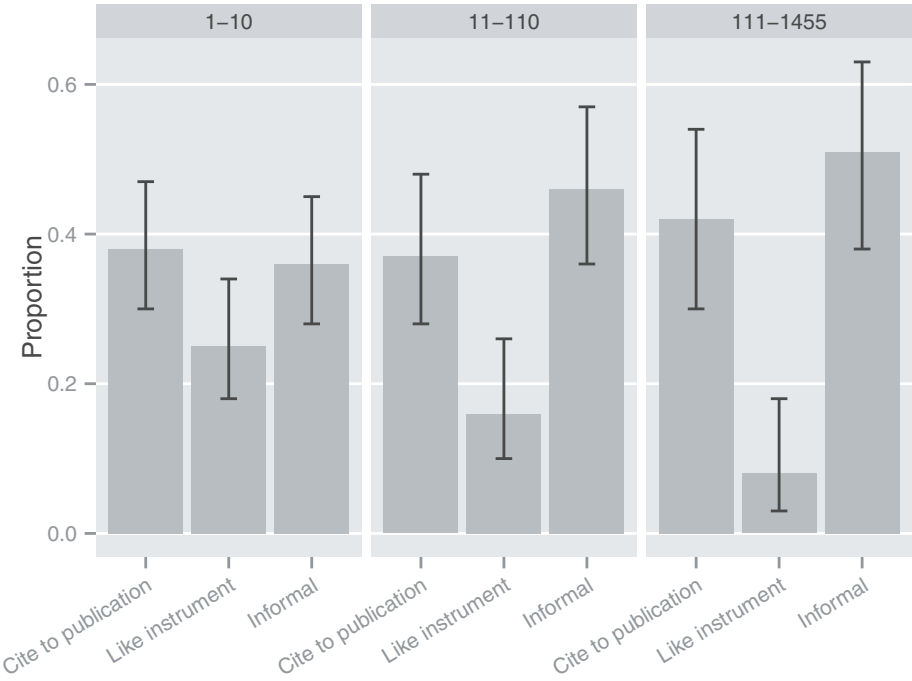

FIG. 4. Major software mention types by journal strata. Error bars show 95% CIs.

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

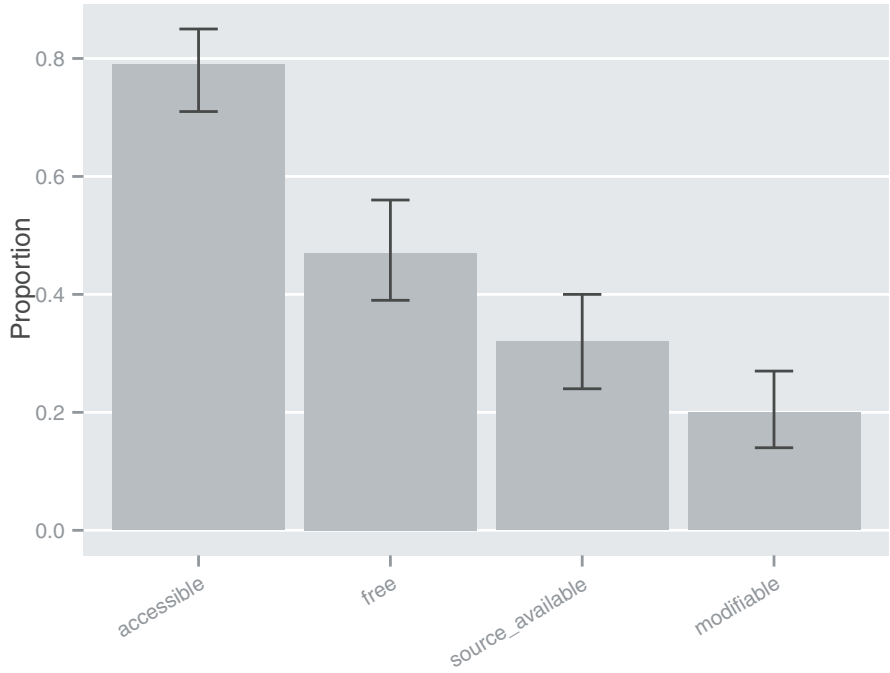

FIG. 5. Characteristics of mentioned software.

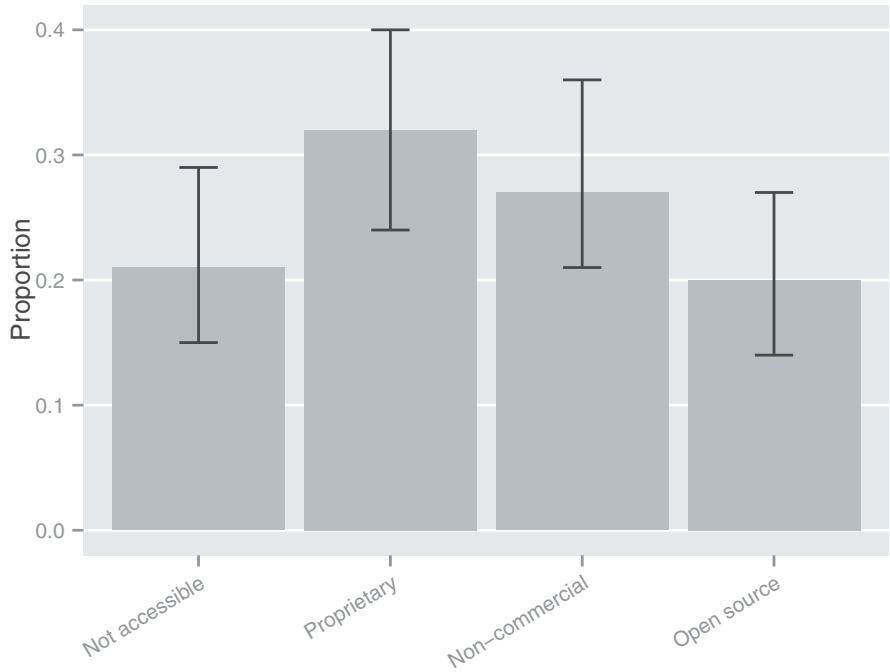

FIG. 6. Types of software mentioned.

($\chi^2(6, N = 274) = 49.248$, $p < .05$). 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

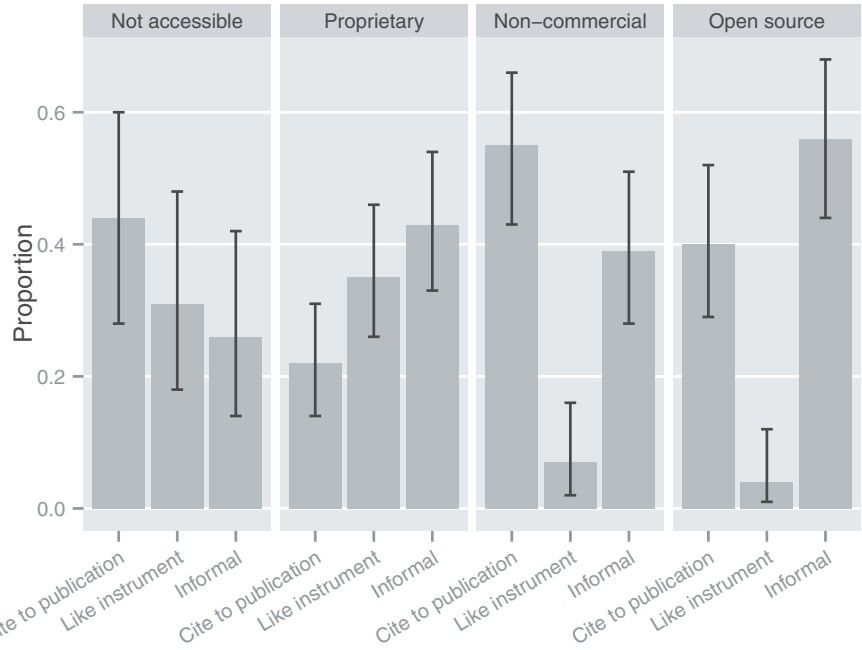

FIG. 7. Relationship between software types and mention types.

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

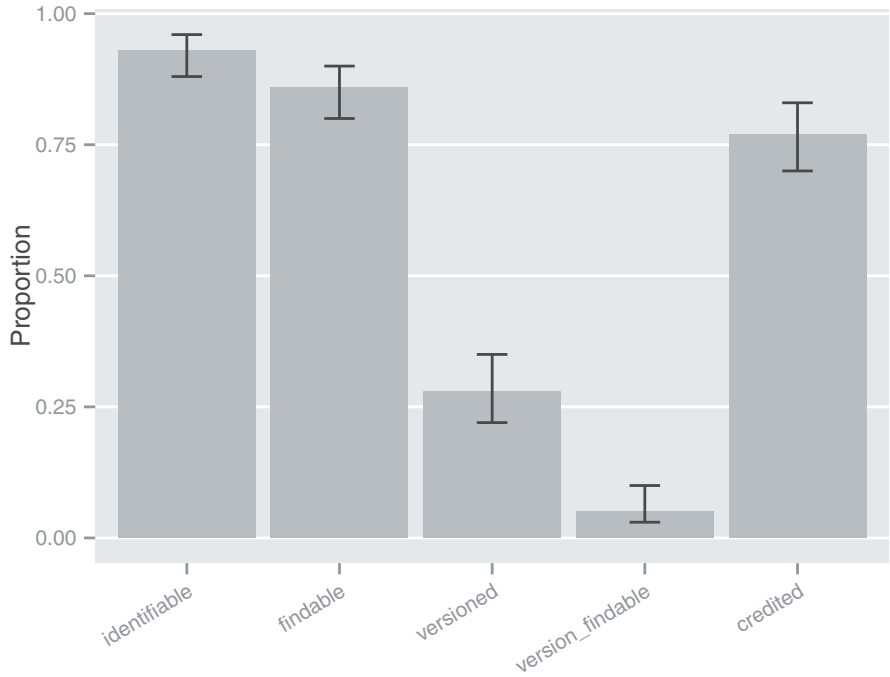

FIG. 8. Functions of citation.

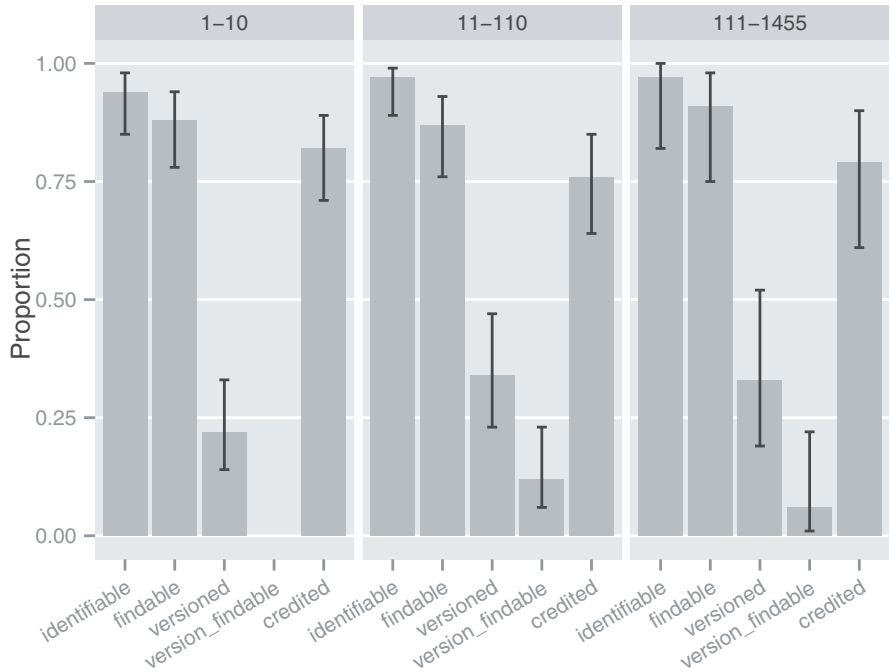

FIG. 9. Functions of citation by strata.

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

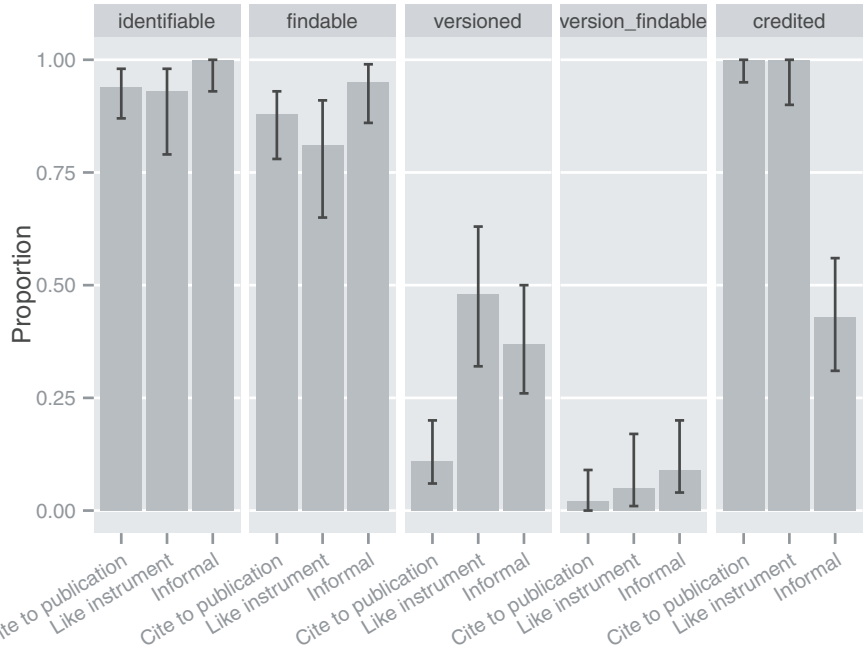

FIG. 10. Functions of citation by mention category.

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

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