# Towards computational reproducibility: researcher perspectives on the use and sharing of software

**Yasmin Alnoamany** Corresp., [1] , **John A. Borghi** [2]

[1] University of California, Berkeley, United States
[2] California Digital Library

Corresponding Author: Yasmin Alnoamany
Email address: yasminal@berkeley.edu

Research software, which includes both the source code and executables used as part of the research process, presents a significant challenge for efforts aimed at ensuring reproducibility. In order to inform such efforts, we conducted a survey to better understand the characteristics of research software as well as how it is created, used, and shared by researchers. Based on the responses of 215 participants, representing a range of research disciplines, we found that researchers create, use, and share software in a wide variety of forms for a wide variety of purposes, including data collection, data analysis, data visualization, data cleaning and organization, and automation. More participants indicated that they use open source software than commercial software. While a relatively small number of programming languages (e.g. Python, R, JavaScript, C++, Matlab) are used by a large number, there is a long tail of languages used by relatively few. Between group comparisons revealed that significantly more participants from computer science write source code and create executables than participants from other disciplines. Group comparisons related to knowledge of best practices related to software creation or sharing were not significant. While many participants indicated that they draw a distinction between the sharing and preservation of software, related practices and perceptions were often not aligned with those of the broader scholarly communications community.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.26727v1 | CC BY 4.0 Open Access | rec: 19 Mar 2018, publ: 19 Mar 2018

# Towards Computational Reproducibility: Researcher Perspectives on the Use and Sharing of Software

**Yasmin AlNoamany**[1] **and John A. Borghi**[2]

[1]**University of California, Berkeley**
[2]**California Digital Library**

Corresponding author:
Yasmin AlNoamany[1]

Email address: yasminal@berkeley.edu

## ABSTRACT

Research software, which includes both the source code and executables used as part of the research process, presents a significant challenge for efforts aimed at ensuring reproducibility. In order to inform such efforts, we conducted a survey to better understand the characteristics of research software as well as how it is created, used, and shared by researchers. Based on the responses of 215 participants, representing a range of research disciplines, we found that researchers create, use, and share software in a wide variety of forms for a wide variety of purposes, including data collection, data analysis, data visualization, data cleaning and organization, and automation. More participants indicated that they use open source software than commercial software. While a relatively small number of programming languages (e.g. Python, R, JavaScript, C++, Matlab) are used by a large number, there is a long tail of languages used by relatively few. Between group comparisons revealed that significantly more participants from computer science write source code and create executables than participants from other disciplines. Group comparisons related to knowledge of best practices related to software creation or sharing were not significant. While many participants indicated that they draw a distinction between the sharing and preservation of software, related practices and perceptions were often not aligned with those of the broader scholarly communications community.

## 1 INTRODUCTION

Research software is a important consideration when addressing concerns related to reproducibility (Hong (2011); Hong (2014); Stodden et al. (2014); Goble (2014)). Effective management and sharing of software saves time, increases transparency, and advances science (Prlić and Procter (2012)). At present, there are several converging efforts to ensure that software is positioned as a "first class" research object that is maintained, assessed, and cited in a similar fashion as scholarly publications (e.g. NIH (2016); Katz et al. (2013); Ram et al. (2017); Crouch et al. (2013)). However, while there is a burgeoning literature exploring the actives of researchers in relation to materials like data (Tenopir et al. (2015); Tenopir et al. (2015); Monteith et al. (2014); Kim and Stanton (2016)), those related to software have received less attention. Specifically, we have been unable to find a study that thoroughly examines how researchers use, share and value their software.

In this paper we report the results of a survey designed to capture researcher practices and perceptions related to software. Survey questions addressed a variety of topics including:

1. What are the characteristics of research software?
2. How do researchers use software?
3. To what extent do current practices related to software align with those related to reproducibility?
4. How do researchers share software?

5. How do researchers preserve software?

After filtering, 215 researchers participated in our survey. Overall, our results demonstrate that researchers create software using a wide variety of programming languages, use software for a wide variety of purposes, have adopted some- but not all- practices recommended to address reproducibility, often share software outside of traditional scholarly communication channels, and generally do not actively preserve their software. Participants from computer science reported that they write source code and create executables significantly more than participants from other disciplines. However, other group comparisons largely did not reach statistical significance.

In the following sections, we provide a more detailed description of our findings. We start with an overview of the related literature (Section 2) then a description of our survey instrument and the demographic characteristics of our participants (Section 3; Section 4). In section 5, we describe our findings related to the characteristics of research software and its usage. Responses to questions involving reproducibility-related practices are detailed in Section 6. Section 7 outlines the responses to questions related to software sharing and preservation. We discuss the implications of our findings in Section 8. Finally, Section 9 contains a discussion of future work.

## 2 RELATED WORK

While there is an emerging body of research examining researcher practices, perceptions, and priorities for products like data (Fecher et al. (2015); Kratz and Strasser (2015); Tenopir et al. (2011); Tenopir et al. (2015)), work related to software has generally focused on how it is found, adopted, and credited (Howison and Bullard (2015b); Hucka and Graham (2016); Joppa et al. (2013)). For example, research examining the re-use of software demonstrates that the most common difficulty for users looking for software is a lack of documentation and that finding software is a difficult task even within technology companies (Sadowski et al. (2015)). However, as software is increasingly central to the research process (Borgman et al. (2012)), understanding its characteristics, its use, and the related practices and perceptions of researchers is an essential component of addressing reproducibility.

The term “reproducibility” has been applied to a variety of efforts aimed at addressing the misalignment between good research practices, including those emphasizing transparency and methodological rigor, and the academic reward system, which generally emphasizes the publication of novel and positive results (Nosek et al. (2012); Munafò et al. (2017)). Attempts to provide a cohesive lexicon for describing reproducibility-related activities are described elsewhere (Goodman et al. (2016)) but *computational reproducibility* generally refers to the description and sharing of software tools and data in such a manner as to enable their use and evaluation by others (Stodden et al. (2013)). Efforts aimed at fostering computational reproducibility are often focused on the sharing of source code but may also include the establishment of best practice guidelines related to how software tools are described, cited, and licensed (e.g. Stodden et al. (2016)).

Because of the costs of irreproducibility, there have been numerous calls urging researchers to more thoroughly describe and share their software (Barnes (2010); Ince et al. (2012); Joppa et al. (2013); Morin et al. (2012a)). Such calls are increasingly backed by mandates from funding agencies. For example, the Wellcome Trust now expects that grant recipients make available “any original software that is required to view datasets or to replicate analyses” (Wellcome (2017)). In parallel, a myriad of guidelines, tools, and organizations have emerged to help researchers address issues related to their software. Software-related best practices have been outlined for both individuals working in specific research disciplines (Eglen et al. (2017); Marwick (2017)) and for the research community in general (e.g. Piccolo and Frampton (2016); Sandve et al. (2013); Jimenez et al. (2017)). Literate programming tools such as Jupyter notebooks (Perez and Granger (2007)) allow researchers to combine data, code, comments, and outputs (e.g., figures and tables) in a human-readable fashion, while packaging and containerization platforms such as ReproZip (Chirigati et al. (2013)) and Docker (Boettiger (2015)) enable the tracking, bundling, and sharing of all of the software libraries and dependencies associated with a research project. Through their integration with Github (`https://github.com/`), services like Figshare (`https://figshare.com/`) and Zenodo (`https://zenodo.org/`) allow researchers to deposit, archive, and receive persistent identifiers for their software. Training researchers to better develop, use, and maintain software tools is the primary focus of community organizations including The Carpentries (Wilson (2006); Teal et al. (2015)) and the Software Sustainability Institute (Crouch et al. (2013)) while scholarly communications-focused organizations such as Force11 have published guidelines for describing and citing software (Smith et al. (2016)).

As is evident in the above description, reproducibility-related efforts involving software often, but not always, overlap with those related to data. However, software presents a number of unique challenges compared to data and other research products. Even defining the bounds of the term “software” is challenging. For example, the National Institute of Standards and Technology (NIST) defines software as “Computer programs and associated data that may be dynamically written or modified during execution.” (Kissel et al. (2011)), a definition that is as recursive as it is potentially confusing for researchers without a background in computer science or software development. Software involves highly interdependent source and binary components that are sensitive to changes in operating environment and are difficult to track (Thain et al. (2015)). Evaluating the validity and reliability of software often requires inspecting source code, which is not possible when proprietary licenses are applied (Morin et al. (2012b); Stodden (2009)). Even when source code is technically available, important information about versions, parameters, and runtime environments is often missing from the scholarly record (Howison and Bullard (2015b); Pan et al. (2016); Stodden et al. (2013)). Seemingly small alterations, even for well described and openly available software tools, can lead to significant effects on analytical outputs (McCarthy et al. (2014)), a problem exacerbated by the fact that researchers often have minimal formal training in software development practices (Hannay et al. (2009); Joppa et al. (2013); Prabhu et al. (2011)). The iterative and collaborative nature of software development also means that it does not fit easily within existing academic incentive structures (Hafer and Kirkpatrick (2009); Howison and Herbsleb (2011); Howison and Herbsleb (2013)).

Research software is a growing concern for research service providers, including those affiliated with academic institutions. Often through workshops facilitated by The Carpentries, many have begun to provide guidance and training to researchers looking to create and use software tools. Services related to the preservation of software have also been explored by some academic libraries (e.g. Rios (2016)). However, these activities remain relatively nascent and it is presently unclear what a mature set of services related to research software and computational reproducibility might look like. By identifying the characteristics or research software, its uses, and elucidating the related practices and perceptions of researchers, we hope to establish a benchmark that can be applied to inform the development of such services in the future.

# 3 METHODS

In order to understand researcher practices and perceptions related to software and computational reproducibility, we designed and disseminated an online survey via the Qualtrics platform (`www.qualtrics.com`). The survey was advertised through blog posts, social media, and research-related email lists and listservs. Because the survey was distributed using different communication channels, we could not calculate the response rate. In Section 4, we detail the demographics of the survey’s participants.

All study materials and procedures were approved by the University of California Berkeley Committee for Protection of Human Subjects and Office for the Protection of Human Subjects (protocol ID 2016-11-9358). The full text of the survey can be found in the supplementary materials. Before beginning the survey, participants were required to read and give their informed consent to participate. After reading the informed consent form (see survey), participants indicated their consent by checking a box. Information from participants who did not check this box was removed from all subsequent analyses. An anonymized version of our survey results (AlNoamany and Borghi (2018a)) as well as the code we used for its analysis (AlNoamany and Borghi (2018b)) are also available on Github (`https://github.com/yasmina85/swcuration`).

## 3.1 Survey Design

The survey was developed to capture a broad range of information about how researchers use, share, and value their software. The final survey instrument consisted of 56 questions (53 multiple choices, 3 open response), divided into four sections. In order, the sections focused on:

1. Demographics: Included questions related the participant’s research discipline, role, degree, age, institution, and funding sources (7 questions)

2. Characteristics of research software: Included questions related to how the participants use software and the characteristics of their software (17 questions).

3. Software sharing practices: Included questions related to how participants make their software available to others (18 questions).

4. How researchers assign value to software (14 questions).

Because only sections 2 and 3 addressed topics related to computational reproducibility, this paper is focused on responses to questions in the first three questions. Future work will further delineate how researchers value software.

We hypothesized that study participants would come to our survey with different levels of knowledge about software development practices and terminology. Therefore, we included a brief list of definitions in our survey for terms like “source code”, “executable”, and “open source software” that participants could refer back to at any time. Participants were not required to answer every question in order to proceed through the survey.

### 3.2 Filtering and Exclusion Criteria

We collected 330 responses to an online survey of software usage and sharing practices and perceptions from late January to early April of 2017. We excluded participants who started the survey but did not answer questions beyond the demographic section to have 215 unique responses. Though the majority of our participants indicated that they were from academia (Table 1), we did not exclude any participant due to institution type because of the possibility that participants could be affiliated with an academic or research program while conducting work in another sector. Institution names and disciplines were canonicalized (e.g. UCB and uc berkeley were mapped to UC Berkeley).

**Table 1.** Demographic breakdown for study participants.

| **Discipline** | **Count** | **Percentage** | **Institution** | **Count** | **Percentage** |
|---|---|---|---|---|---|
| Computer Science | 39 | 18.3% | Academic: Research Focused | 164 | 77.0% |
| Biology | 29 | 13.6% | Academic: Teaching Focused | 22 | 10.3% |
| Psychology | 28 | 13.1% | Government | 13 | 6.1% |
| Engineering | 13 | 6.1% | Nonprofit | 7 | 3.3% |
| Interdisciplinary Programs | 12 | 5.6% | Academic: Medical School | 3 | 1.4% |
| Mathematics | 12 | 5.6% | Commercial | 2 | 0.9% |
| Physics | 12 | 5.6% | Other | 2 | 0.9% |
| Earth Science | 9 | 4.2% | **Role** | **Count** | **Percentage** |
| Library Sciences | 9 | 4.2% | Graduate Student | 67 | 31.5% |
| Social Sciences | 9 | 4.2% | Postdoc | 38 | 17.8% |
| others | 41 | 19.20% | Research Faculty | 35 | 16.4% |
| **Highest degree** | **Count** | **Percentage** | Staff | 29 | 13.6% |
| Doctorate | 110 | 51.9% | Principal Investigator | 25 | 11.7% |
| Masters | 72 | 34.0% | Research Assistant | 10 | 4.7% |
| Bachelors | 26 | 12.3% | Undergraduate Student | 2 | 0.9% |
| High school | 3 | 1.4% | Research | 1 | 0.5% |
| Professional degree | 1 | 0.5% | Other | 6 | 2.8% |

## 4 PARTICIPANT DEMOGRAPHICS

We asked participants about their age, professional degrees, professional title (or role) and institutional affiliation, institution type, and the sources of funding. The majority of these questions were multiple choice with an option for open response upon selecting “Other”.

The mean and median age of our participants were 35.8 and 33 years old, respectively. Reflecting the ubiquity of software within the research enterprise, participants were drawn from a wide variety of research disciplines, institution types, and roles. As shown in Table 1, the disciplines most represented in our sample were computer science, biology, and psychology. The majority of our participants were drawn from 129 different research-focused academic institutions (including 12% out of 215 researchers from UC Berkeley). Table 1 also shows that participants had a range of degrees and roles, with the most common being doctorate (51.9%, $N = 215$) and graduate student (31.5%, $N = 215$), respectively. In terms of funding, the most common responses were the National Science Foundation (NSF) (16.7%, $N = 215$) and the National Institutes of Health (7.0%, $N = 215$).

# 5 CHARACTERISTICS AND USE OF RESEARCH SOFTWARE

Before diving deeper into how researchers use their software, we wanted to identify its characteristics. In this section, we describe responses to questions related to the creation and use of source code and executables.

## 5.1 Source Code and Executables

We asked participants about the generation and use of source code and executables (i.e. Do you write source code?, Do you use source code written by others?, Do you create executables?, Do you use executables created by others?). We found that 84.2% out of 215 responding participants write source code and 89.8% out of 215 use source code written by others while 53.7% out of 214 create executables and 80.4% use executables written by others.

Figure 1 shows that participants from computer science were significantly more likely to write source code [$\chi^2$(2,N = 215) $= 8.93, p < 0.05$], create executables [$\chi^2(2, N = 214) = 22.67, p < 0.00001$], and use executables created by others [$\chi^2(2, N = 214) = 6.66, p < 0.05$] than participants from other disciplines. Comparisons related to the use of others' source code did not reach statistical significance [$\chi^2(2, N = 215) = 1.21, p = 0.55$].

We also asked participants about the type of software they use (i.e. Do you use commercial software in the course of your research? Do you use open source software in the course of your research?). As shown in Figure 2 more participants indicated that they use open source software (94.9%, $N = 214$) than commercial software (72.8%, $N = 214$).

## 5.2 Programming Languages

In order to quantify the breadth of programming languages used in a research setting, we asked participants about the languages they use when writing their own code. Table 2 shows the top ten languages, which together account for 86.4% of languages selected. The top used languages in our sample were Python, R, Javascript, C++, Matlab, Java, C, PHP, and Perl. Python and R were the most used languages, selected by 64.0% and 57.0% of participants of respectively. For the most part, these results are in line with previous findings from Hucka and Garaham (Hucka and Graham (2016)) and also match those of a recent study from Stack Overflow (Inc. (2016)). In total, 52 different languages were chosen, with the most common responses outside of the top ten being Ruby, C#, ASP, SAS, XML, XQuery, and Julia. Quantitatively measuring the use programming languages in academic research is difficult due to the variability of reporting practices (Howison and Bullard (2015a)), but our results are largely in line the rapid ascent of R and Python as tools for data science.

**Table 2.** The top 10 programming languages used by the researchers in our sample. A total of 214 participants answered this question. Together these languages represent 86.4% of the languages selected. Note that participants could choose more than one language.

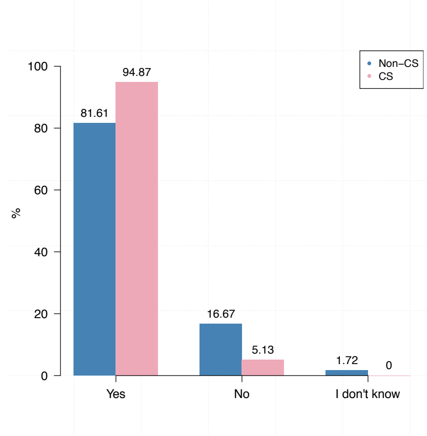
| Language | Python | R | SQL | Javascript | C++ | Matlab | Java | C | PHP | PERL |
|---|---|---|---|---|---|---|---|---|---|---|
| Selection | 137 | 122 | 60 | 57 | 54 | 45 | 35 | 25 | 25 | 21 |
| Percentage | 64.0% | 57.0% | 28.0% | 26.6% | 25.2% | 21.0% | 16.4% | 11.7% | 11.7% | 9.8% |

We also inquired about collaborative code development and the extent to which the same programming languages are used within a lab or a research group. Though 53.3% of participants indicated that they write code collaboratively, we were surprised to see that only 33.0% indicated that everyone in the lab used the same language(s).

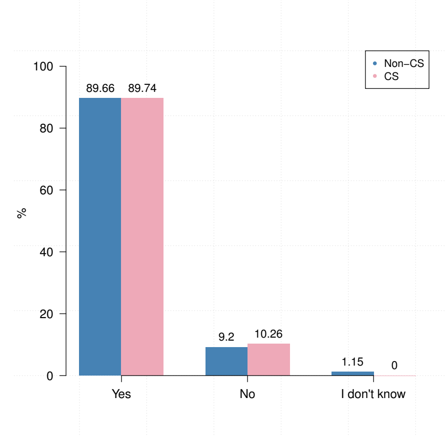
## 5.3 Use of Research Software

Previous scholarship (e.g. Borgman et al. (2012)) has indicated that researchers use software for a wide variety of purposes. To examine the purposes of research software, we asked participants about how they use their code or software (Figure 1). This question allowed them to choose multiple answers from a suggested list or input other answers.

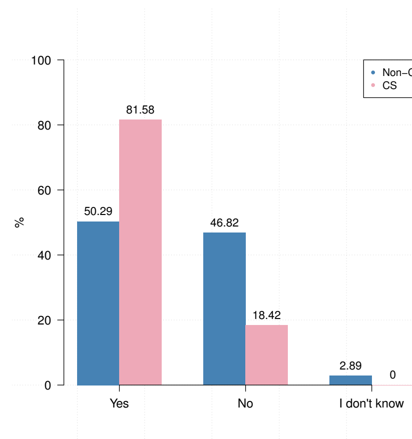
Figure 3(a) shows that our participants use software primarily to analyze data, visualize data, clean and organize data, automate their work, and collect data. A total of 104 participants (55.7% out of 212 participants) responded that they use software for all five. "Other" responses included running simulations, building models, researching algorithms, testing methods, writing compilers, and sharing and publishing



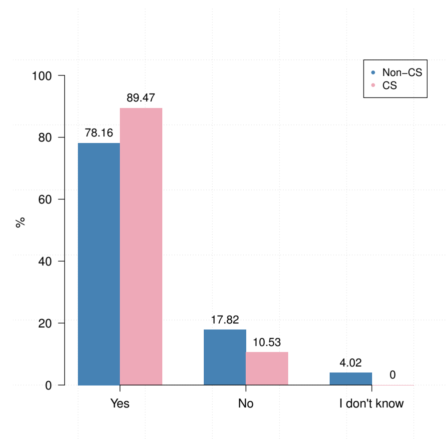

(a) Do you write source code? $N = 215$




(b) Do you use source code written by others? $N = 215$




(c) Do you create executables? $N = 214$




(d) Do you use executables created by others? $N = 214$

**Figure 1.** Significantly more participants from computer science stated that they write source code, create executables, and use executables created by others than participants from other disciplines.

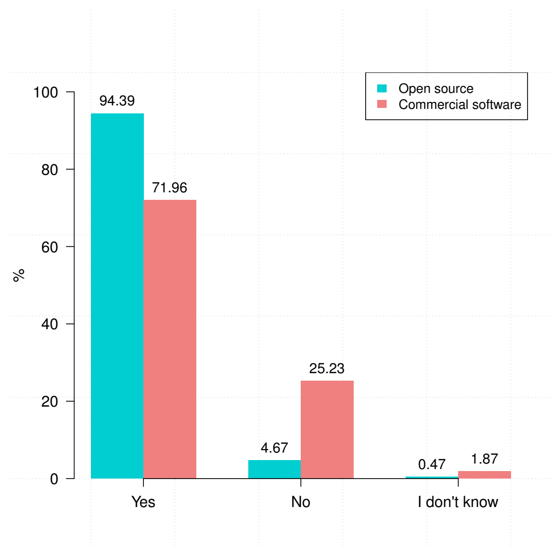
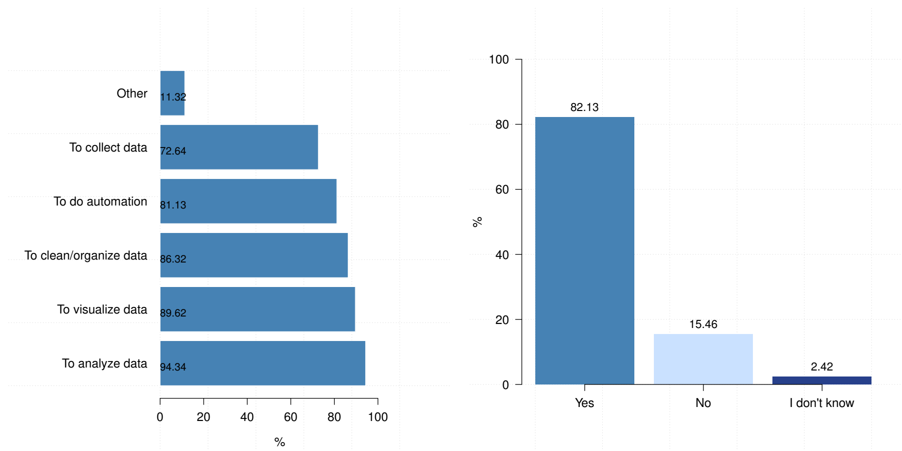



**Figure 2.** The use of open source software versus commercial software. $N = 214$.




(a) How do you use code or software? $N = 212$

(b) Have you ever repurposed your code or software? $N = 208$

**Figure 3.** The purpose of using research software. Note that the first question could be answered with more than one choice.

data. We also asked if researchers repurpose their code (i.e. using it for a project other than the one for which it was originally created) and found that 82% out of 208 participants indicated that they do that.

We investigated how researchers collaborate on code writing within their research labs (Figure 4) (e.g. “Do you write code collaboratively (i.e. with another person or multiple people)?”, “Does everyone in your lab or research group write code using the same programming language(s)?”) We found that 49.8% ($N = 200$) of researchers write code collaboratively (Figure 4(a)), while only 30% ($N = 201$) use the same coding language in their research labs (Figure 4(b)).

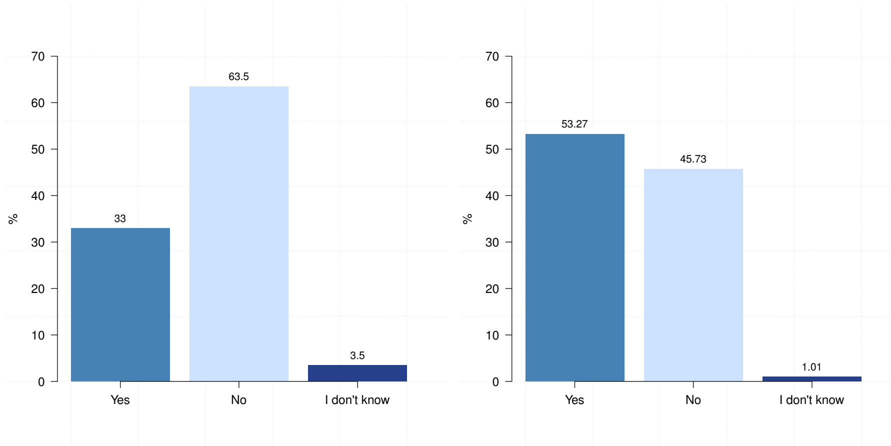
Previous studies on the reuse of research software have focused mainly on licensing, review of code, and user awareness (Joppa et al. (2013); Morin et al. (2012a)). Reinforcing the need to establish best practices (or good enough practices - e.g. Wilson et al. (2017) akin to those related to the management of research data, 79.8% of our participants ($N = 208$) indicated that they repurpose their code.

In an open response question, we asked participants to describe, in their own words, how they use their software and code. Here, again, participants indicated that they use software for a wide variety of purposes. One participant summed the relationship between software and their research succinctly as “I use software for stimulus presentation, data acquisition, and data analysis and visualization - basically my entire research is run via computers (and thus code).” Similarly, another participant described the application of software within the field of computer science: “As a computer scientist, almost every aspect of my research from grant proposal to collecting data to analyzing data to writing up my results involves software. I write software. I use software my collaborators or students write as well as open source and commercial software.

## 6 REPRODUCIBILITY-RELATED PRACTICES

To understand how the practices of our participants align with those related to computational reproducibility, we asked a number of questions about adding comments to source code, generating documentation, communicating information about dependencies, and using “notebook” applications such as Jupyter. We also asked about awareness of coding conventions and best practices. The results of these questions are shown in Figure 5.

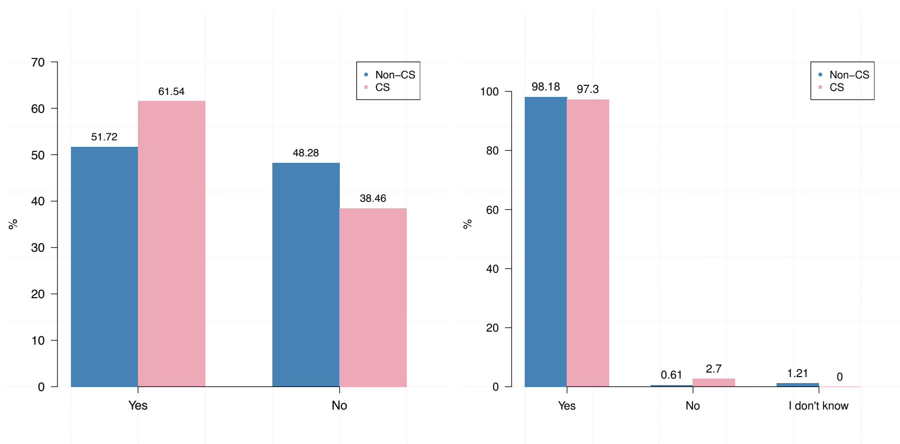
In line with previous research (Hannay et al. (2009); Joppa et al. (2013); Prabhu et al. (2011)), only 53.4% ($N = 215$) of our participants indicated that they have received formal training in coding conventions or best practices. At the same time, we found that many actually employ practices that are commonly cited for establishing computational reproducibility. For example, when asked “Do you include comments in your code?” and “When you share your code or software, do you provide information about dependencies?” the majority of participants (98.0%, $N = 204$, 72.2%, $N = 169$) indicated that they include comments and provide information about dependencies, respectively. However, substantially




(a) Does everyone in your lab or research group write code using the same programming language(s)? $N = 201$

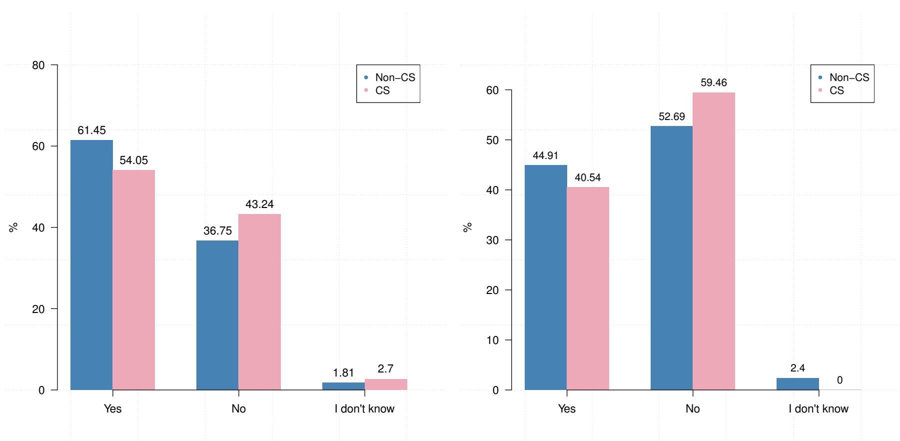
(b) Do you write code collaboratively? $N = 200$

**Figure 4.** Consistency of programming languages within research groups.




(a) Have you received training in coding conventions or best practices? $N = 215$.

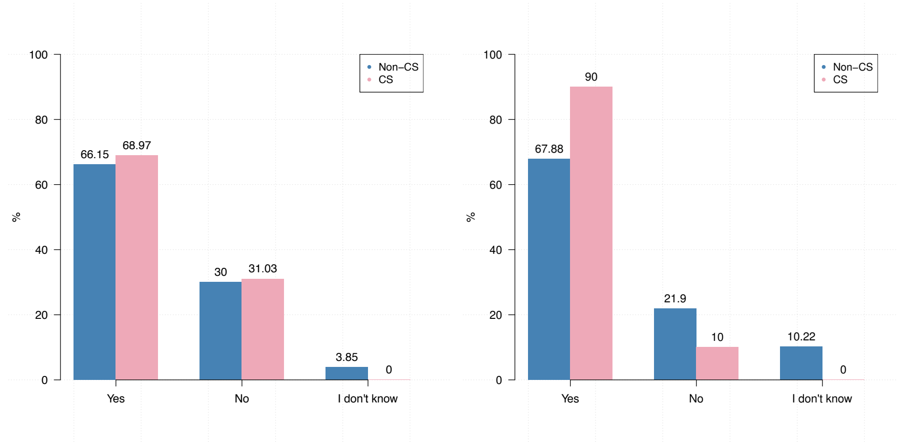
(b) Do you include comments in your code? $N = 204$.




(c) Do you generate documentation for your code? $N = 205$.

(d) Do you write code using a notebook? $N = 206$.

**Figure 5.** Reproducibility practices in research.




(a) When you share your code or software, do you share it alongside related files (e.g. datasets)? $N =$ 161.

(b) When you share your code or software, do you provide information about dependencies? $N =$ 169.

**Figure 6.** CS researchers tend to provide information about dependencies more than other disciplines.

fewer indicated that they employ other practices such as generating documentation (60.0%, $N = 205$). While electronic lab notebooks have been cited as a tool for ensuring reproducibility (Kluyver et al. (2016)), only 43.6% ($N = 206$) of our participants indicated that they use them to write code.

Comparisons of responses by discipline (e.g. computer science versus others) or location (e.g. UC Berkeley versus others) were insignificant even, surprisingly, on questions related to training [discipline: $\chi^2(1, N = 215) = 1.58, p = 0.21$, location: $\chi^2(2, N = 215) = 0.00, p = 1.00$] (Figure 5). The lone exception was in providing information about dependencies. Significantly more respondents from computer science reported that they include information about dependencies when they share their code than participants from other disciplines [$\chi^2(2, N = 169) = 17.755, p < 0.001$].

# 7 SHARING AND PRESERVATION OF THE RESEARCH SOFTWARE

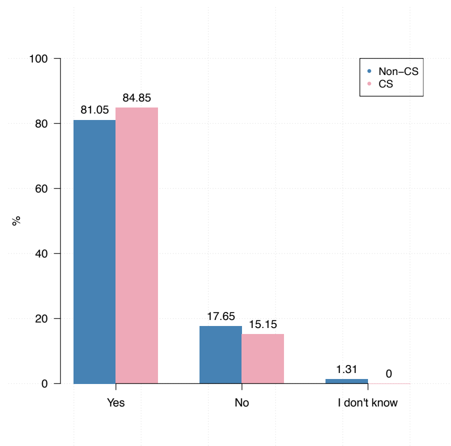
Making materials available for others to evaluate, use, and build upon is an essential component of ensuring reproducibility. Much of the previous work examining the sharing of research software has focused on the degree to which software is cited and described irregularly in the scholarly literature (Howison and Bullard (2015a); Smith et al. (2016)) and the relationship between code sharing and research impact (Vandewalle (2012)). In order to gain a greater understanding of how sharing practices relate to reproducibility, we asked our participants a variety of questions about how they share, find, and preserve software.

## 7.1 Sharing Research Software

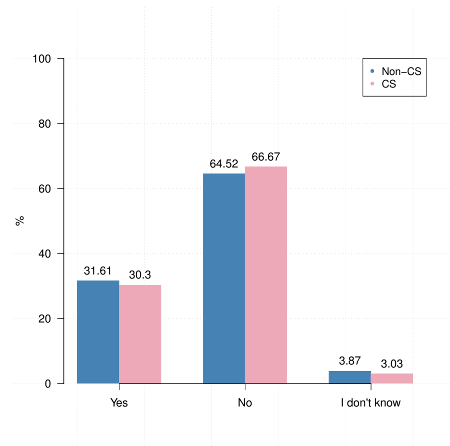
### *Sharing Practices*

While only half (50.5%, $N = 198$) of our participants indicated that they were aware of related community standards in their field or discipline, the majority indicated that they share software as part of the research process (computer science: 84.9%, other disciplines: 81.1% for $N = 187$) (Figure 7). Of 189 participants, 31% indicated that there were reasons their software could not be shared (Figure 7(b)). The most commonly cited restrictions on sharing were the inclusion of sensitive data, intellectual property concerns, and the time needed to prepare code for sharing. Comparisons between computer science and other disciplines on the sharing of code were not statistically significant [$\chi^2(2, N = 187) = 1.5842, p > 0.4529$].

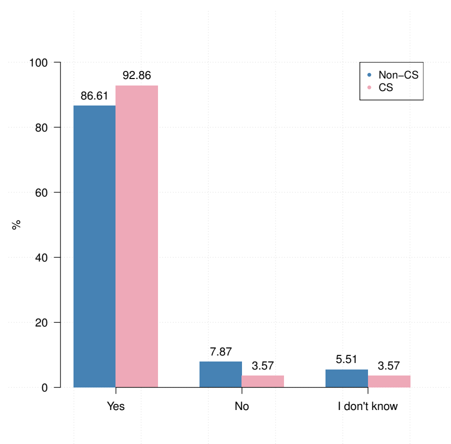
We also checked if participants share new versions of their code and found that 81% ($N = 156$) do so using a version control system. A group comparison related to the sharing of new versions was not statistically significant [CS vs non-CS: $\chi^2(2, N = 156) = 2.2, p > 0.05$] (Figure 7(c)), however significantly more participants from computer science indicated that they share their codes via a version control system than those from other disciplines [$\chi^2(2, N = 185) = 16.4, p < 0.05$] (Figure 7(d)).



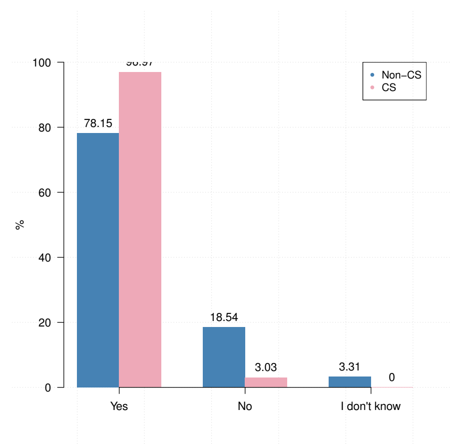

(a) Do you share the code or software created as part of your research? $N = 187$.




(b) Is there any reason your code or software could not be shared? $N = 189$.

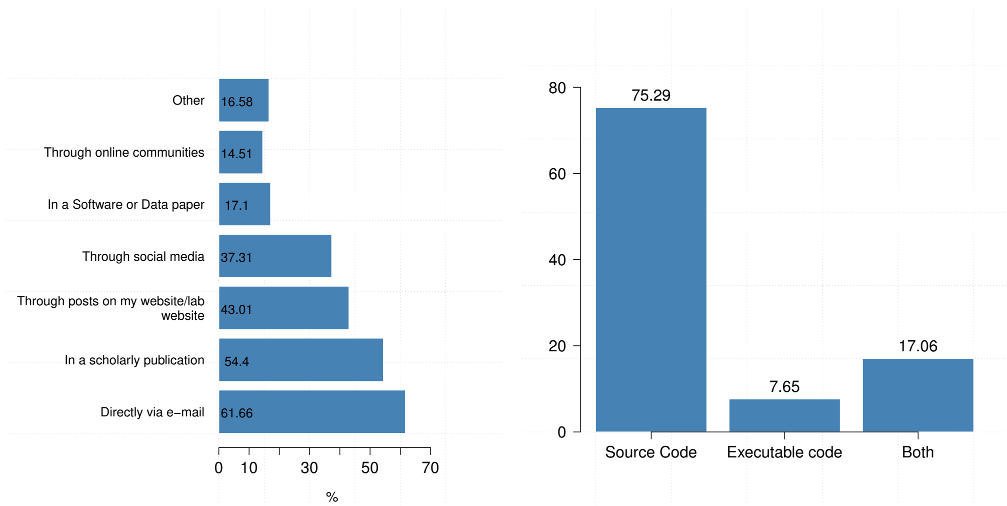



(c) If you make a change to your code, do you share a new version? $N = 156$.




(d) Do you use a version control system (e.g. Git, SVN)? $N = 185$.

**Figure 7.** Practices of code sharing.




(a) How do you tell people about the code or software you've shared? $N = 165$.

(b) In what format do you typically share your code? $N = 175$.

**Figure 8.** Methods and formats for sharing software. Note that both of these questions could be answered with more than one response.

#### *Sharing Format and Platform*

We asked our participants about how they share their code and found that 75.3% of 175 participants share their software in the form of source code, 7.6% share executables only, and 17.1% share both formats (Figure 8). As shown in Figure 8(a), participants indicated that they share their software through a variety of channels, with the most common being e-mail. The figure shows that 73.94% of the time our participants make their code available through direct communication and 50% make their code available through social media platforms. The participants who indicated that they use methods other than those listed in our survey generally responded that they do so using platforms such Github or the Open Science Framework. A few researchers mentioned that they save their code along with the dataset in their institutional repository, while others indicated that they publicize their code via conferences.

### 7.2 Preserving Research Software

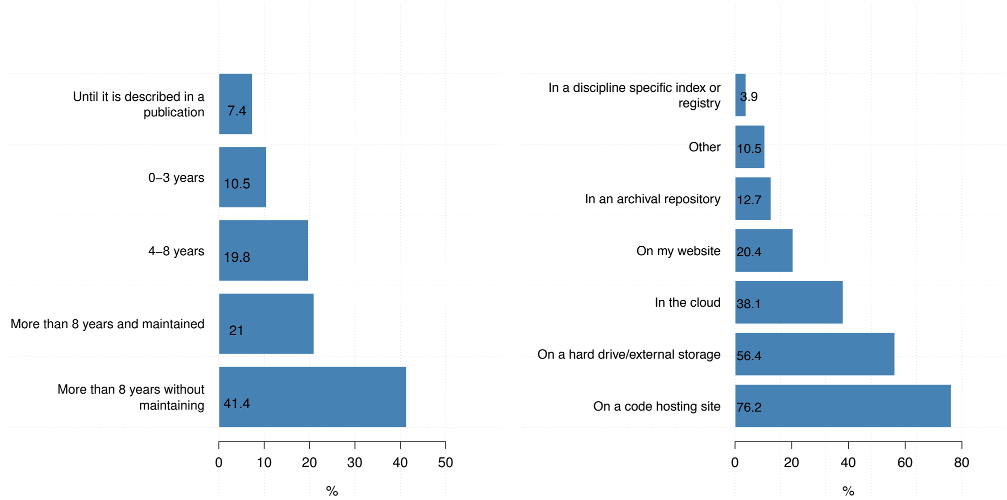
We asked variety of questions about preserving research software (i.e. Do you take any steps to ensure that your code or software is preserved over the long term?, How long do you typically save your code or software?, and Where do you save your code or software so that it is preserved over the long term?). While research software is a building block for ensuring the reproducibility, 39.9% of participants ($N = 183$) do not prepare their code for long-term preservation.

#### *How long do you typically save your code or software?*

Figure 9(a) shows that the majority of our participants (40.4% out of 162) preserve their code for more than eight years, but generally not in a way that maintains its use. In contrast, 7.4% ($N = 162$) of participants keep their codes until it is described in a publication, poster, or presentation. We found 10.5% out of 162 researchers tend to keep their codes 3 years or less and 19.8% tend to keep their codes 4-8 year. Only 21.0% out of 162 researchers tend to keep their codes for 8 years or more with maintaining their codes for future access and use.

#### *Where do you save your code or software so that it is preserved over the long term?*

In terms of where our participants preserve their code, Figure 9(b) shows that 76.2% of the time participants use code hosting sites such as Github. About 56.4% of the time, researchers use hard drives or external storage to preserve their codes and 38.1% of the time they preserve their codes by putting them on the cloud. Only 12.7% of our participants indicated that they use archival repositories (e.g. figshare). The participants who entered "other" responses mentioned that they use a backup system of their lab, organization archive (e.g., University server), their own PC, language package registry (CRAN, PyPi or similar), Internal SVN repository, or project specific websites.




(a) How long do you typically save your code or software? $N = 162$.

(b) Where do you save your code or software so that it is preserved over the long term? $N = 182$.

**Figure 9.** 76.2% of researchers use Github for preserving their codes. Note that the second question could be answered with more than one choice.

We asked participants to define *sharing* and *preserving* in their own words. Their responses generally indicated that they make a distinction between the two concepts. For example, one participant stated succinctly, "sharing is making code available to others, in a readily usable form. Preserving is ensuring to the extent practical that the code will be usable as far into the future as possible." However, several responses indicated that participants did not necessarily regard preservation as an active process that continues even after the conclusion of a particular project (e.g. "sharing means giving access to my code to someone else. Preserving means placing my code somewhere where it can remain and I will not delete it to save room or lose it when I switch computers or suffer a hard drive failure.". In contrast, other responses indicated that participants were aware that preservation is important for reuse purpose and had a knowledge of preservation tools. For example, one researcher defined preserving software as, "branching so that code remains compatible with different versions of overarching libraries (in my case) or with new coding standards and compilers". and another stated "Preserving should be done via a system like LOCKSS that ensures that provides for redundancy. Sharing can be done via the web, but must include a license so that recipients know about their rights."

# 8 DISCUSSION

Scholars throughout the humanities and sciences depend on software for a wide variety of purposes, including the collection, analysis, and visualization of data (Borgman et al. (2012); Hey et al. (2009)). Though ubiquitous, software presents significant challenges to efforts aimed at ensuring reproducibility. Our results demonstrate that researchers not only create and use software in a wide variety of forms and for a wide variety of purposes, but also that their software-related practices are often not completely in line with those associated with reproducibility. In particular, our results demonstrate that, while scholars often save their software for long periods of time, many do not actively preserve or maintain it. This perspective is perhaps best encapsulated by one of our participants who, when completing our open response question about the definition of sharing and preserving software, wrote "Sharing means making it publicly available on Github. Preserving means leaving it on GitHub". We share this anecdote not to criticize our participants or their practices, but to illustrate the outstanding need for support services related to software.

In the broader scholarly communications space, there are several prominent frameworks that relate to the reproducibility of scholarly outputs. As part of an effort to advance data as a "first class" research product, the FAIR (Findable, Accessible, Interoperable, and Reusable) guidelines provide a measurable set of principles related to the management and sharing of research data (Wilkinson et al. (2016)).

The FAIR principles are general enough that they can, with some modification, also be applied to software (Jimenez et al. (2017)). At the level of scholarly publications, the TOP (Transparency and Openness Promotion) guidelines (Nosek et al. (2015)) addresses citation standards and the availability of research materials including data and software. A supplement to TOP, the Reproducibility Enhancement Principles (REP) (Stodden et al. (2016)) specifically targets disclosure issues related to computation and software. However, our results support previous work indicating that software still mostly exists outside the reputation economy of science (Howison and Herbsleb (2011)) which indicates that a more education-based approach, that provides guidance about software before the publication stage is necessary.

The majority of our participants indicated that view code or software as "first class" research product, that should be assessed, valued, and shared in the same way as a journal article. However, our results also indicate that there remains a significant gap between this perception and actual practice. The fact that our participants indicated that they create and use software in a wide variety of forms and for a wide variety of purposes demonstrates the significant technical challenges inherent in ensuring computational reproducibility. In contrast, the lack of active preservation and tendency to share software outside traditional (and measurable) scholarly communications channels displayed by our sample demonstrates the social and behavioral challenges. A significant difficulty in ensuring computational reproducibility is that researchers oftentimes do not treat their software as a "first class" research product. These findings reinforce the need for programs to train researchers on how to maintain their code in the active phase of their research.

At present, there are a number of initiatives focused on addressing the preservation and reproducibility of software. In the United States, the Software Preservation Network (SPN) (Meyerson et al. (2017)) represents an effort to coordinate efforts to ensure the long-term access to software. The focus of SPN is generally on cultural heritage software rather than research software, but their work delineating issues related to metadata, governance, and technical infrastructure has substantial overlap with what is required for research software. In the United Kingdom, the Software Sustainability Institute trains researchers on how to develop better software and make better use of the supporting infrastructure (Crouch et al. (2013)). Befitting the necessity of training and preservation indicated by our study, a similar effort, the US Software Sustainability Initiative was recently awarded funding by the National Science Foundation (NSF Award #1743188). While it is likely not possible for academic institutions to offer support services that cover the broad range of programming languages and applications described in our survey results, collaborating with such groups to create guidance and best practice recommendations may a feasible first step in engaging with researchers about their software and code in the same manner as many research data management (RDM) initiatives now engage with them about their data.

While research stakeholders including academic institutions, publishers, and funders have an interest in tackling issues of computational reproducibility in order to ensure the integrity of the research process, our results demonstrate the complexity of doing so. One participant summed up why their code could not be made re-usable: "Most of my coding is project specific and not reusable between projects because the datasets I encounter are very variable. I typically only generate packages for tasks such as getting data from a database (e.g., PubMed) and keeping RMarkdown templates in an orderly way."

## 9 CONCLUSION AND FUTURE WORK

In this paper, we introduced the results of surveying researchers across different institution on software usage, sharing, and preservation. We also checked the practices used to manage software for ensuring the reproducibility and integrity of the scientific research. Our results point to several interesting trends including the widespread writing of source code and use of source code written by others, the variety of programming languages used and the lack of consistency even within the same lab or research group, the use of open source software over commercial software, and the adoption of some practices assure computational reproducibility, such as adding comments and documentation to code, but not others, specifically the general lack of active preservation. The findings of this paper inform ongoing conversations about research software and reproducibility on the current practices around research software. This will help service providers to deliver the right tools and systems that help researchers to manage their code and help in ensuring the integrity of the reproducibility in the scholarly ecosystem.

The present study was designed to capture a broad picture of how researchers use and share their software. For this reason, we were not able to provide a particularly granular picture of how individual practices relate to reproducible science outcomes. For example, while the majority of our participants responded that they include comments in their source code and generate documentation for their software, we were not able to make any judgment about whether or not the contents of these comments and documentation are sufficient to ensure reproducibility. Follow up research is needed in order to gain a more nuanced understanding of how processes related to the creation and use of research software relate to reproducibility. However, despite these limitations, our results indicate several potential directions for future library services centered on helping researchers create, use, and share their software and assure computational reproducibility.

## ACKNOWLEDGMENTS

We would like to thank our colleagues at UC Berkeley Library and California Digital Library for their valuable suggestions and insightful comments throughout this project.

## REFERENCES

AlNoamany, Y. and Borghi, J. A. (2018a). Data: Researcher perspectives on the use and sharing of software.

AlNoamany, Y. and Borghi, J. A. (2018b). Software study code.

Barnes, N. (2010). Publish your computer code: it is good enough. *Nature*, 467(7317):753–753.

Boettiger, C. (2015). An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79.

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

Chirigati, F., Shasha, D., and Freire, J. (2013). Reprozip: Using provenance to support computational reproducibility. In *Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance*, TaPP '13, pages 1–4. USENIX Association.

Crouch, S., Hong, N. C., Hettrick, S., Jackson, M., Pawlik, A., Sufi, S., Carr, L., Roure, D. D., Goble, C., and Parsons, M. (2013). The software sustainability institute: Changing research software attitudes and practices. *Computing in Science & Engineering*, 15(6):74–80.

Eglen, S. J., Marwick, B., Halchenko, Y. O., Hanke, M., Sufi, S., Gleeson, P., Silver, R. A., Davison, A. P., Lanyon, L., Abrams, M., Wachtler, T., Willshaw, D. J., Pouzat, C., and Poline, J.-B. (2017). Toward standard practices for sharing computer code and programs in neuroscience. *Nature Neuroscience*, 20:770.

Fecher, B., Friesike, S., and Hebing, M. (2015). What drives academic data sharing? *PLOS ONE*, 10(2):1–25.

Goble, C. (2014). Better software, better research. *IEEE Internet Computing*, 18(5):4–8.

Goodman, S. N., Fanelli, D., and Ioannidis, J. P. A. (2016). What does research reproducibility mean? *Science Translational Medicine*, 8(341):341ps12–341ps12.

Hafer, L. and Kirkpatrick, A. E. (2009). Assessing open source software as a scholarly contribution. *Communications of the ACM*, 52(12):126–129.

Hannay, J. E., MacLeod, C., Singer, J., Langtangen, H. P., Pfahl, D., and Wilson, G. (2009). How do scientists develop and use scientific software? In *Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, SECSE '09, pages 1–8. IEEE Computer Society.

Hey, T., Tansley, S., and Tolle, K. (2009). *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research.

Hong, N. C. (2011). Digital preservation and curation: the danger of overlooking software. *The Preservation of Complex Objects*, page 25.

Hong, N. C. (2014). Dealing with software: the research data issues.

Howison, J. and Bullard, J. (2015a). How is software visible in the scientific literature. Technical report, Technical report, Univ. of Texas.

Howison, J. and Bullard, J. (2015b). 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*.

Howison, J. and Herbsleb, J. D. (2011). Scientific software production: Incentives and collaboration.

In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, CSCW '11, pages 513–522. ACM.

Howison, J. and Herbsleb, J. D. (2013). Incentives and integration in scientific software production. In *Proceedings of the ACM 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 459–470. ACM.

Hucka, M. and Graham, M. J. (2016). Software search is not a science, even among scientists. *CoRR*, abs/1605.02265.

Inc., S. E. (2016). Developer survey results 2016.

Ince, D. C., Hatton, L., and Graham-Cumming, J. (2012). The case for open computer programs. *Nature*, 482:485.

Jimenez, R., Kuzak, M., Alhamdoosh, M., Barker, M., Batut, B., Borg, M., Capella-Gutierrez, S., Chue Hong, N., Cook, M., Corpas, M., Flannery, M., Garcia, L., GelpÌ, J., Gladman, S., Goble, C., Gonz·lez Ferreiro, M., Gonzalez-Beltran, A., Griffin, P., Gr¸ning, B., Hagberg, J., Holub, P., Hooft, R., Ison, J., Katz, D., Leskoek, B., Lupez Gumez, F., Oliveira, L., Mellor, D., Mosbergen, R., Mulder, N., Perez-Riverol, Y., Pergl, R., Pichler, H., Pope, B., Sanz, F., Schneider, M., Stodden, V., Suchecki, R., Svobodov· Va?ekov·, R., Talvik, H., Todorov, I., Treloar, A., Tyagi, S., van Gompel, M., Vaughan, D., Via, A., Wang, X., Watson-Haigh, N., and Crouch, S. (2017). Four simple recommendations to encourage best practices in research software [version 1; referees: 3 approved]. *F1000Research*, 6(876).

Joppa, L. N., McInerny, G., Harper, R., Salido, L., Takeda, K., O'Hara, K., Gavaghan, D., and Emmott, S. (2013). Troubling trends in scientific software use. *Science*, 340(6134):814–815.

Katz, D. S., Allen, G., Hong, N. C., Parashar, M., and Proctor, D. (2013). First workshop on sustainable software for science: Practice and experiences (wssspe): Submission and peer-review process, and results. *arXiv preprint arXiv:1311.3523*.

Kim, Y. and Stanton, J. M. (2016). Institutional and individual factors affecting scientists' data-sharing behaviors: A multilevel analysis. *Journal of the Association for Information Science and Technology*, 67(4):776–799.

Kissel, R., Kissel, R., Blank, R., and Secretary, A. (2011). Glossary of key information security terms. In *NIST Interagency Reports NIST IR 7298 Revision 1, National Institute of Standards and Technology*.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., and development team [Unknown], J. (2016). Jupyter notebooks: A publishing format for reproducible computational workflows. In Loizides, F. and Scmidt, B., editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90. IOS Press.

Kratz, J. E. and Strasser, C. (2015). Researcher perspectives on publication and peer review of data. *PLOS ONE*, 10(2):1–21.

Marwick, B. (2017). Computational reproducibility in archaeological research: Basic principles and a case study of their implementation. *Journal of Archaeological Method and Theory*, 24(2):424–450.

McCarthy, D. J., Humburg, P., Kanapin, A., Rivas, M. A., Gaulton, K., Cazier, J.-B., and Donnelly, P. (2014). Choice of transcripts and software has a large effect on variant annotation. *Genome Medicine*, 6(3):26.

Meyerson, J., Vowell, Z., Hagenmaier, W., Leventhal, A., Roke, E. R., Rios, F., and Walsh, T. (2017). The Software Preservation Network (SPN): A Community Effort to Ensure Long Term Access to Digital Cultural Heritage. *D-Lib Magazine*, 23(5/6).

Monteith, J. Y., McGregor, J. D., and Ingram, J. E. (2014). Scientific research software ecosystems. In *Proceedings of the 2014 European Conference on Software Architecture Workshops*, ECSAW '14, pages 9:1–9:6. ACM.

Morin, A., Urban, J., Adams, P. D., Foster, I., Sali, A., Baker, D., and Sliz, P. (2012a). Shining light into black boxes. *Science*, 336(6078):159–160.

Morin, A., Urban, J., and Sliz, P. (2012b). A quick guide to software licensing for the scientist-programmer. *PLOS Computational Biology*, 8(7):1–7.

Munafò, M. R., Nosek, B. A., Bishop, D. V. M., Button, K. S., Chambers, C. D., Percie du Sert, N., Simonsohn, U., Wagenmakers, E.-J., Ware, J. J., and Ioannidis, J. P. A. (2017). A manifesto for reproducible science. *Nature Human Behaviour*, 1(1):0021.

NIH (2016). Strategies for nih data management, sharing, and citation.

Nosek, B. A., Alter, G., Banks, G. C., Borsboom, D., Bowman, S. D., Breckler, S. J., Buck, S., Chambers, C. D., Chin, G., Christensen, G., Contestabile, M., Dafoe, A., Eich, E., Freese, J., Glennerster, R., Goroff, D., Green, D. P., Hesse, B., Humphreys, M., Ishiyama, J., Karlan, D., Kraut, A., Lupia, A., Mabry, P., Madon, T., Malhotra, N., Mayo-Wilson, E., McNutt, M., Miguel, E., Paluck, E. L., Simonsohn, U., Soderberg, C., Spellman, B. A., Turitto, J., VandenBos, G., Vazire, S., Wagenmakers, E. J., Wilson, R., and Yarkoni, T. (2015). Promoting an open research culture. *Science*, 348(6242):1422–1425.

Nosek, B. A., Spies, J. R., and Motyl, M. (2012). Scientific utopia: Ii. restructuring incentives and practices to promote truth over publishability. *Perspectives on Psychological Science*, 7(6):615–631.

Pan, X., Yan, E., and Hua, W. (2016). Disciplinary differences of software use and impact in scientific literature. *Scientometrics*, 109(3):1593–1610.

Perez, F. and Granger, B. E. (2007). Ipython: A system for interactive scientific computing. *Computing in Science Engineering*, 9(3):21–29.

Piccolo, S. R. and Frampton, M. B. (2016). Tools and techniques for computational reproducibility. *GigaScience*, 5(1):30.

Prabhu, P., Jablin, T. B., Raman, A., Zhang, Y., Huang, J., Kim, H., Johnson, N. P., Liu, F., Ghosh, S., Beard, S., Oh, T., Zoufaly, M., Walker, D., and August, D. I. (2011). A survey of the practice of computational science. In *State of the Practice Reports*, SC '11, pages 19:1–19:12. ACM.

Prlić, A. and Procter, J. B. (2012). Ten simple rules for the open development of scientific software. *PLoS Comput Biol*, 8(12):e1002802.

Ram, K., Katz, D., Carver, J., Gesing, S., and Weber, N. (2017). Si2-s2i2 conceptualization: Conceptualizing a us research software sustainability institute (urssi).

Rios, F. (2016). The Pathways of Research Software Preservation: An Educational and Planning Resource for Service Development. *D-Lib Magazine*, 22(7/8).

Sadowski, C., Stolee, K. T., and Elbaum, S. (2015). How developers search for code: a case study. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 191–201. ACM.

Sandve, G. K., Nekrutenko, A., Taylor, J., and Hovig, E. (2013). Ten simple rules for reproducible computational research. *PLOS Computational Biology*, 9(10):1–4.

Smith, A. M., Katz, D. S., and Niemeyer, K. E. a. (2016). Software citation principles. *PeerJ Computer Science*, 2:e86.

Stodden, V. (2009). The legal framework for reproducible scientific research: Licensing and copyright. *Computing in Science & Engineering*, 11(1):35–40.

Stodden, V., Guo, P., and Ma, Z. (2013). Toward reproducible computational research: An empirical analysis of data and code policy adoption by journals. *PLOS ONE*, 8(6):1–8.

Stodden, V., Leisch, F., and Peng, R. D. (2014). *Implementing reproducible research*. CRC Press.

Stodden, V., McNutt, M., Bailey, D. H., Deelman, E., Gil, Y., Hanson, B., Heroux, M. A., Ioannidis, J. P., and Taufer, M. (2016). Enhancing reproducibility for computational methods. *Science*, 354(6317):1240–1241.

Teal, T. K., Cranston, K. A., Lapp, H., White, E., Wilson, G., Ram, K., and Pawlik, A. (2015). Data carpentry: workshops to increase data literacy for researchers. *International Journal of Digital Curation*, 10(1):135–143.

Tenopir, C., Allard, S., Douglass, K., Aydinoglu, A. U., Wu, L., Read, E., Manoff, M., and Frame, M. (2011). Data sharing by scientists: practices and perceptions. *PloS one*, 6(6):e21101.

Tenopir, C., Dalton, E. D., Allard, S., Frame, M., Pjesivac, I., Birch, B., Pollock, D., and Dorsett, K. (2015). Changes in Data Sharing and Data Reuse Practices and Perceptions among Scientists Worldwide. *PLOS ONE*, 10(8):1–24.

Thain, D., Ivie, P., and Meng, H. (2015). Techniques for Preserving Scientific Software Executions: Preserve the Mess or Encourage Cleanliness? *Proceedings of the 12th International Conference on Digital Preservation (iPRES)*.

Vandewalle, P. (2012). Code sharing is associated with research impact in image processing. *Computing in Science Engineering*, 14(4):42–47.

Wellcome (2017). Policy on data, software and materials management and sharing.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., Gonzalez-Beltran, A., Gray, A. J. G., Groth, P., Goble, C., Grethe, J. S., Heringa, J., 't Hoen, P. A. C., Hooft, R., Kuhn, T., Kok, R., Kok, J., Lusher, S. J., Martone, M. E., Mons, A., Packer, A. L., Persson, B., Rocca-Serra, P., Roos, M., van Schaik, R., Sansone, S.-A., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M. A., Thompson, M., van der Lei, J., van Mulligen, E., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft, K., Zhao, J., and Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3:160018.

Wilson, G. (2006). Software carpentry: Getting scientists to write better code by making them more productive. *Computing in Science & Engineering*, 8(6):66–69.

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., and Teal, T. K. (2017). Good enough practices in scientific computing. *PLOS Computational Biology*, 13(6):1–20.