IJDC / *Conference Paper*

Who writes scholarly code?

Sarah Nguyễn
University of Washington

Vicky Rampin
New York University

Abstract

This paper presents original research about the behaviours, histories, demographics, and motivations of scholars who code, specifically how they interact with version control systems locally and on the Web. By understanding patrons through multiple lenses – daily productivity habits, motivations, and scholarly needs -- librarians and archivists can tailor services for software management, curation, and long-term reuse, raising the possibility for long-term reproducibility of a multitude of scholarship.

Submitted March 12, 2022 ~ *Accepted* April 22, 2022

Correspondence should be addressed to Vicky Rampin, 70 Washington Sq South, New York, NY 10012. Email: vs77@nyu.edu

This paper was presented at International Digital Curation Conference IDCC22, online, 13-16 June, 2022

The *International Journal of Digital Curation* is an international journal committed to scholarly excellence and dedicated to the advancement of digital curation across a wide range of sectors. The IJDC is published by the University of Edinburgh on behalf of the Digital Curation Centre. ISSN: 1746-8256. URL: http://www.ijdc.net/

Introduction

Computational scholarship is dependent on software; from specialty boutique scripts created by individuals or groups for one-off analyses to full-fledged software packages distributed worldwide (Hettrick, 2014). Reproducibility requires more than just access to the immediate code being used; it also requires its dependencies, down to the operating system (Steeves, Rampin, & Chirigati, 2018). Software preservation is therefore key to long-term reproducibility; holding other research materials (such as data) without the code that transforms it into the intellectual or analytical result has less utility, and may even make those materials more difficult to understand (Chassanoff & Altman, 2019).

With the rise in scholars coding as a part of their research and teaching/learning, version control systems (VCS) have also become more integrated into the academic toolkit (Milliken, Nguyễn, & Steeves, 2021). VCS are "system[s] that record changes to a file or set of files over time so that you can recall specific versions later" (Chacon & Straub, 2014). Version control is done on a repository which contains all the metadata and history for all tracked files. The most popular VCS today is Git (StackOverflow, 2018), which was created in 2005 as an attempt to create a more user-friendly VCS (Torvald, n.d.).

An ecosystem has emerged around these VCS to promote wide sharing of repositories on the Web. These source code "forges" are web-based platforms that provide support for open and collaborative software development as well as bug tracking and wikis. When Git was created, a separate category was formed—a Git Hosting Platform (GHP), which either exclusively supports Git (such as GitHub and GitLab) or foregrounds its support (like SourceForge) (Milliken et al., 2021). With GHP, there are now social features in addition to the local version control that Git offers. The friendly web-based interface with additional functionality such as wikis, task management, and static hosting of web pages, also expands its possible use cases and makes the platform attractive to academics creating and working with a variety of materials and computational research.

However, GHP do not provide long-term preservation of the repositories that they host, and in fact, some have depreciated, taking all the hosted work with them (Squire, 2017). With more scholarship and knowledge being produced using Git as a tool and being hosted and accessed on GHP, there is a dearth of understanding scholars' practices and behaviors using Git and GHP and how libraries and archives can preserve research software, with its necessary components (such as wikis documenting how to use it), for longer term reuse. We present original research on academics using version control, specifically how these scholars are using version control locally as well as via GHP, as individuals and in collaboration. We ask:

RQ1: Who is using version control and GHP? What are their experiences learning and using it, and what are their motivations? What are key characteristics of scholars' use of version control and GHP?

RQ2: How are users setting up their work to be archived later on? Where are they archiving their work?

RQ3: How do the collaborations facilitated by hosting code on GHP contribute to long-term usability of scholarly code?

Background

The use of VCS and GHP have been integrated into academic work, and scholars have shared the benefits and drawbacks to using them. These toolkits are influential to the creation and sharing of knowledge now more than ever, and as knowledge creation moves towards FAIR principles for both data (Wilkinson et al., 2016) and software (Lamprecht et al., 2019), we explore how GHP supports (or does not) tasks scholars need to carry out within the academic ecosystem:

- **Version control**: A history of changes allows scholars to gather a more comprehensive understanding on how computations, scripts, and analyses have evolved through a study by viewing line-by-line changes in the code base, as well as who in the research team to contact for questions (Hrynaszkiewicz, 2013; Kaki, Sivaramakrishnan, & Jagannathan, 2019; Meloni, 2019).
- **Community & collaboration**: When researchers publish data, scripts, and software used in their study onto a GHP, there are lower barriers for reuse "and [it can] serve as a powerful catalyst to accelerate progress" (Langille, Ravel, & Fricke, 2018; Ram, 2013; Vanderplas, 2014/2021).
- **Method tracking**: Scholars are writing protocols to aggregate, manage, and analyze data into lab and field notebooks. Git commit history can give researchers and team members reference to understand the iterative progress of a study. Data analysis pipelines, also known as processing workflows, can benefit from version control systems for exposure, collaborative feedback, and reproducibility of the methodology (CARPi, Minges, & Piel, 2017; Dallas, 2015/2019; Raj, 2016; Ram, 2013).
- **Education**: Syllabi, open education resources (OER), and course management can be hosted on GHP. Students and teachers also benefit from GHP as the platform enhances collaboration, communication, feedback, provenance of course materials and submitted work, and exposes students to industry experience (Britton, 2014; Croxall, 2012; Glassey, 2019; Laadan, Nieh, & Viennot, 2010; Lawson, 2013; Metz, 2015; Ross, 2019).
- **Quality assurance**: GHP allows for merging small changes of code often throughout the development cycle and provides testing for bugs in data structures and code changes. This ensures quality experimentation and analyses (Benedetto, Podstavkov, Fattoruso, & Coimbra, 2019; Krafczyk, Shi, Bhaskar, Marinov, & Stodden, 2019; Widder, Hilton, Kästner, & Vasilescu, 2019; Yenni et al., 2018).
- **Reproducibility**: Through publishing, hosting code, and versioning for transparent methodologies, reviewers are able to obtain scripts to replicate analyses, “permit[ting] others to fully understand what a researcher has done” (Stodden, Leisch, Peng, Leisch, & Peng, 2018). When researchers cite their own Git repository within their paper, reviewers can navigate to that repository and access the source code, computational environment (when provided), and data in order to replicate the study (Vanderplas, 2016).
- **Peer production**: As a product of communal and collaborative tasks, scholars can take a wiki-like approach to crowdsourcing and commons-based development practices on GHP. This makes use of the assignment and bug tracking features (aljedaxi, 2019/2019; Astropy, 2011/2022; Zotero, 2011/2022; Perez, 2012).
- **Publishing**: Publishing on GHP “can build up well-documented and robust code libraries that may be reused for future projects and for teaching purposes” (Allen & Mehler, 2019). Common features in GHP, such as web hosting and continuous integration (CI), can help communicate the full scholarship pipeline that can be integrated into a research object bundle (Soiland-Reyes, Gamble, & Haines, 2014).

Academic research on the uses of version control and GHP in scholarly experiences tends to focus on the beneficial possibilities of using the tool throughout the research process, which leaves a gap in understanding who is using Git in academia and their general usage. This article offers a necessary perspective to understand what scholars report to use, ignore, and champion, as well as addresses how curators can help prepare to preserve source code repositories for future reuse, aiding in the long-term reproducibility of the scholarship.

Methodology

To understand scholars who use Git and GHP, we undertook a three-part mixed methods study. The first part included focus group sessions with self-identified minimal Git users, followed by a

survey which targeted Git users of all levels. The last part was in-depth interviews with survey participants who opted in. All participants self-identified as people who contribute to academic activities and are embedded in scholarly environments. The results presented here were derived from the survey only (Steeves & Nguyễn, 2020) in order to provide a quantitative perspective to research findings.

The goal of the survey was to better understand the landscape of academic users of VCS and GHP, in order to gather a wide-ranging and comparable census across the spectrum of their behaviors. The survey was open for four months and two days. There were 54 questions divided into seven sections: Background, Learning Git, Teaching Git, Features on GHP, Scholarship, Sustainability, and Demographics. The full survey and resulting data are freely available (Steeves & Nguyễn, 2020). Hosted on Qualtrics, the survey included branching logic to determine participant eligibility and to display probing questions based on participant answers. Multiple choice options within the survey were influenced by patterns understood from the literature review and analysis of focus group data, such as the list of popular GHP options and terminology that beginners and non-users could comfortably comprehend.

Recruitment for the survey was conducted through social media, listservs, targeted outreach, and snowball sampling—participants shared the survey with their colleagues. The survey was open to an international audience. We aimed to collect at least 500 survey submissions, and received 496 total survey submissions. A sample email that was sent out to recruit participants and the list of distribution channels we contacted is available in the OSF[1].

The survey was open during the COVID-19 pandemic, therefore, the sample population of those who participated in the survey may lack representation from potential participants who had experienced extenuating circumstances. Those who volunteered their time to submit their self-reflections on the survey also self-identified as scholars and/or academics which may have left less representation to students who do not yet self-identify as full-time scholars. Also, this survey came from the perspective of libraries and archival practices, and as is common in library-hosted surveys, there is a potential loss in the sample population from those who do not already make use of the library's resources and/or are not already slightly motivated or interested to join a conversation about Git.

This paper represents a statistical analysis of the survey results to provide a broad overview of scholars' use of VCS and GHP. The data analysis includes frequency distributions and cross-tabulation. The analysis was done using R and tidyverse packages (Müller & Bryan, 2020; Pedersen, 2020; Wickham, 2007; Wickham et al., 2019), and the source code is also openly available (Rampin, 2022).

Results

The survey results represent broad-scale behaviors, motivations, and workflows of academic coders who use a VCS. 496 participants took the survey overall. On average, the survey took 15 minutes to complete and had a 90% completion rate. Of these participants, 358 fit the four criteria for inclusion in analysis: 1) they currently work or study at an academic institution, 2) they write or (re)use code in their work, and 3) they use a version control system for that code, and 4) they consent to participate.

Participant population

At the end of the survey, we asked participants to share some basic information about their current work and academic status. This included their field of study, current role, length of time in their current role, and the type of institution in which they work. Using this information, we can better understand how using VCS manifests in different contexts.

[1] https://osf.io/nu649

Of those who answered, there were 140 staff members (45 of which were postdocs), 81 students (58 doctoral, 19 Masters, 6 undergraduate), and 76 faculty members (49 tenure-track/tenured, 27 continuing contract), with 24 selecting "other". The length of time in their current role varied, though of the survey respondents who answered most were starting out. 269 respondents indicated that they were in their current position for up to seven years (172 were in years 0-3, 97 in years 4-7), while 53 were in their positions for more than eight years (25 in years 8-10, 28 selected 11+ years). 195 of these participants were situated in a public university, 93 in a private university, and 33 in "other" (popular text write-ins included "research institute," "government agency," and "non-profit organization").

Additionally, participants were asked to choose which discipline they align with the most. The list of 83 disciplines was drawn from NSF Science and Engineering Degrees: 1966–2010 Classification of Fields of Study ("National Center for Science and Engineering Statistics Appendix B Science and Engineering Degrees: 1966–2010," 2013, pp. 1966–2010) and Wikidata academic discipline query ("Wikidata Query: Academic Discipline," 2020). When grouped into larger categories, of those who answered 226 were from STEM, 74 from Social Sciences, 13 from the Humanities, and 10 selected "Multidisciplinary."

Table 1. Distribution of field and status among participants

	Continuing contract faculty	Tenure track faculty	Staff	Postdoc	Doctoral student	Masters student	Undergrad student	Other
Humanities	1	4	3	2	1	0	0	2
Multi	0	1	4	3	0	0	0	2
Social Science	13	13	27	4	7	3	0	7
STEM	13	31	61	36	50	16	6	13

The majority of the participants are public university staff members and students early in their roles. When looking at earlier survey answers, these participants reported that they started their careers using Git or were trained to practice version control as a part of their job expectations. This direct job expectation led scholars to Git and GHP, and training in the context of a position (with discrete outcomes) was a factor that can lower the barrier-to-entry to the tools.

Version Control Use

Git is the most popular VCS across all groups. It is the VCS that most participants began using (217 participants), and also the one that most participants currently use (354 participants). Fig. 1 shows a comparison across different local VCS. When using VCS locally, more participants use a terminal (297) than a graphical user interface (GUI, 107 participants). 43 participants selected "other" and wrote text responses that included many mentions of extensions to popular development environments (such as RStudio, VS Code, and Emacs), as well as a web browser (e.g. only uploading documents to web-based repositories on GHP).

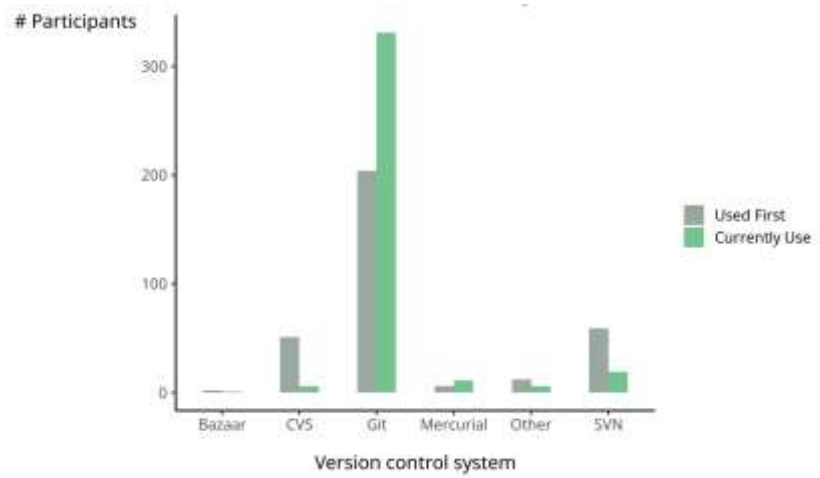

Figure 1. First and current VCS used by participants.

When using a GHP on the Web, an overwhelming number of participants use GitHub as their primary GHP—343. The next most popular was GitLab.com (140 participants), then Bitbucket (74), then self-hosted solutions (57), and finally 7 participants did not use a web-hosted platform for storing and working on their source code (write-in text suggested this was largely for security/confidentiality reasons). Of those who self-hosted a GHP, text responses included self-hosted GitLab, Gerrit, Gitea, and Git-on-a-Linux server (no GUI). Of those that selected "Other," text responses included sourcehut, codeberg, darcshub, and dropbox.

Many participants use both Git and their GHP of choice on a daily basis (177, as seen in Table 2). Most of the time, participants used Git and GHP equally (88 weekly, 15 monthly, 10 quarterly, 1 yearly). Some participants used Git daily but GHP weekly (21), and to a lesser extent the opposite (14 participants used a GHP daily and Git weekly). In the case of the latter, this could be indicative of making more contributions to the scholarly ephemera around code (see Table 3), rather than code contributions. Scholarly ephemera are rich material which provides context to the source code development process, including content that provides insights into the genesis of a project, communications between its contributors and collaborators, and the procedures and interactions that brought it to its most current state (Milliken, Nguyễn, & Rampin, 2019). These artifacts are important when understanding the history of a repository, the relationships between repositories, the branching and network formation of repositories and their contributors, and the ways in which this information could be used to track derivatives of current work. An example would be bug trackers—a GHP feature which combines discussions, to-do lists, and notifications. An example is the "issue" feature in GitHub and GitLab. Bug trackers are used to do everything from peer review to calls for retraction. We argue that all of these activities need to be preserved in addition to the source code, and that those contributions are significant to the development of scholarly code overall.

Table 2. Participants' frequency in using both Git locally and GHP on the Web.

		Frequency using Git locally					
		Daily	**Weekly**	**Monthly**	**Quarterly**	**Yearly**	**Other**
Frequency using GHP	**Daily**	177	14	1	0	0	0
	Weekly	21	88	13	1	0	1

	Monthly	1	3	15	1	0	0
	Quarterly	2	1	0	10	0	0
	Yearly	0	0	0	0	1	0
	Other	0	0	0	0	0	1

Additionally, participants demonstrated their significant usage of various GHP features that contribute to the creation of scholarly ephemera. In Table 3, the spread of GHP features according to participant status type shows how often each group makes use of each feature. Of all the GHP features, merge requests (e.g. on GitHub, forking and making a pull request), bug tracking, and web hosting (e.g. GitLab Pages) tend to be the most popular. Of note is that postdocs use Continuous Integration (CI, services that automate building, testing, and shipping code) more than web hosting, master's students use code annotation more than bug tracking, and undergrads use code annotation more than web hosting. When looking at postdocs' free-text answers to how they use CI, most popular responses included automating tests (from unit tests and self-harm checks, down to linting), making websites, building documentation, and publishing packages. Understanding where there are contributions to the scholarly ephemera around a project can inform what might be needed alongside the code repository itself in order to make it useful in the long-term.

Table 3. Participants' use of GHP features, by status.

Status	Project Boards	CI	Code annotation	Merge requests	Bug tracker	Web hosting	Wiki	Other
Staff	36	49	56	81	80	63	41	5
Postdoc	10	28	24	36	35	25	12	2
Doctoral student	17	23	31	50	49	36	14	3
Masters student	3	4	8	17	7	8	1	0
Undergrad student	1	3	4	6	4	2	2	0
Continuing contract faculty	9	11	15	22	22	22	10	0
Tenure track faculty	12	24	26	42	34	34	10	4
Other	10	12	13	16	19	15	10	2

In combination with questions about past and current usage, participants were asked about what motivated them to use VCS and GHP. We found collaboration to be the strongest motivator for both local and Web-based use, with openness as a second major motivator. Scholars have a simple need for version control, and there currently isn't a more accepted or widespread system than Git. This was reflected in the "Other" responses for using a Git locally, which included many horror stories relating to a lack of code management that prompted participants to seek out a tool to help them. Participants also wrote about wanting to be able to contribute to open source software and needing to learn Git under that impetus. Many wrote that it was a requirement for a class or a job which is, of course, a significant social and economic pressure.

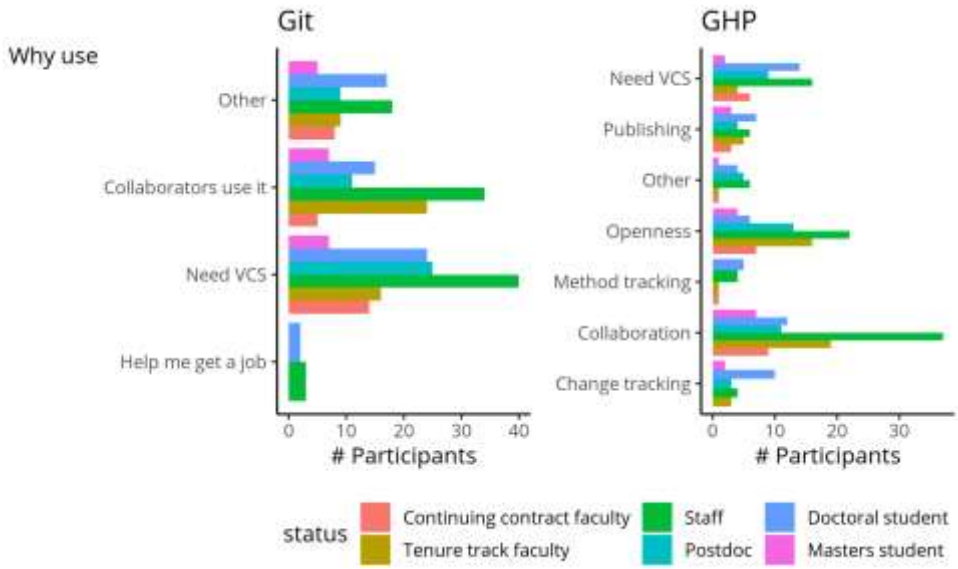

Figure 2. Participants' motivations for using Git locally and GHP, by status.

Figure 2 compares the behavioral motivations of Git and GHP according to participant scholarly status. The pressures that lead participants to adopt Git into their workflow tend to relate to broader skills and social pressures, while the reported reasons for using GHP demonstrates needs for specific features and tasks. There is a larger drive to use GHP due to factors that lead to providing information to others, whether that is for "openness" or "collaboration", whereas the larger drivers to use Git stem from the various personal reasons of "needing a VCS", which in turn could be for the use of engaging collaborators. This also served multiple purposes, as we saw later in the survey when we asked specifically about using GHP as a backup location and provider for code—275 participants reported GHP as a tool to backup their code and 59 did not (potentially using other backup methods or none at all).

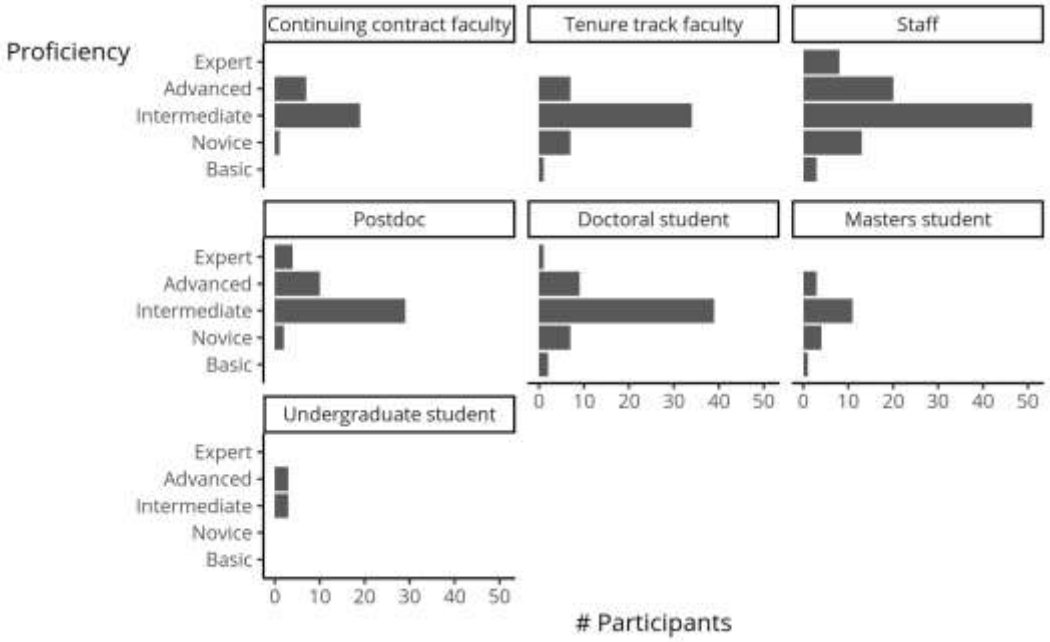

Figure 3. Participants self-reported proficiency with Git, by status.

Finally, participants were also asked to rate their proficiency with using Git overall on a scale of 1 (basic knowledge) to 5 (recognized authority). Most participants rated themselves in the middle, as 'intermediate (practical application)' (see Fig. 3). Staff overwhelmingly reported intermediate to expert proficiency. Some tenure track, staff, and graduate students reported basic and novice proficiency levels, which may provide some insight into their openness to engage in using Git despite their lower proficiency to Git as a tool in their daily workflow.

Teaching & Learning

We were motivated to understand how participants learned Git and GHP and got their proficiency. Further, it was made clear through the literature review and focus groups that many academics teach Git and GHP to each other, through formal and informal means. This is a core academic activity and is embedded throughout, making it vital to understand. This section will show how participants perceive their teaching and learning experiences with Git and GHP whether it was in the classroom, among colleagues, or on their own in documentation or discussion forums.

Of those who answered, 286 participants learned Git after entering postsecondary education, with 40 having learned it before. 22 selected 'Other' with most specifying time periods traditionally after postsecondary education (e.g. on the job). The majority of these participants found Git neutral-to-hard to learn, while GHP were rated from easy to very easy to learn. This may be in part because most GHP are run by for-profit entities who can pay people to run UX studies and write in-depth documentation, driving the value of the product (the GHP).

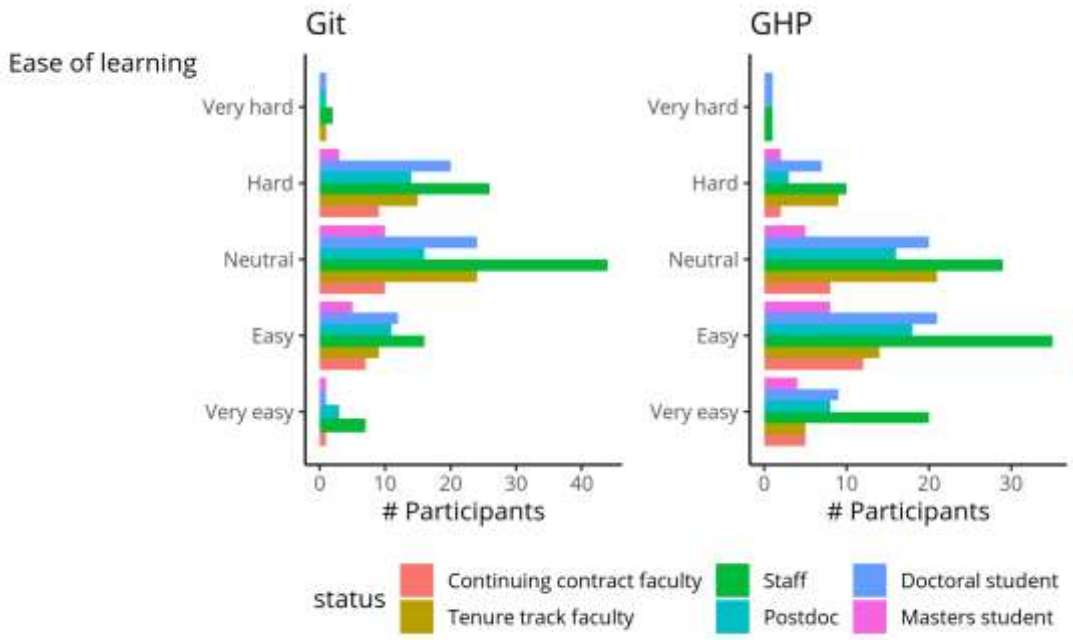

Figure 4. Participants level of ease learning both Git and GHP, by status.

Tenure track faculty participants were the only community that did not mark Git as "very easy" to learn while also being one of few other communities who did not perceive GHP as "very hard" to learn (see Fig. 4). Masters students, postdoctoral students, and continuing contract faculty also did not perceive GHP as "very hard" to learn. Similarly, master's students and continuing contract faculty did not perceive Git as "very hard" to learn.

As said in many Git workshop materials, it is not a tool that someone can learn in one go. To that end, participants were asked to reflect on how often they re-teach themselves Git. The majority of our participants re-teach themselves about once a quarter (see Fig. 5). Additionally, understanding the materials they used to learn Git can shed light on effective (or ineffective) methods. Participants relied mostly on books, articles, or blog posts to guide them through instructions on using Git, while online forums and manuals were the second most referenced learning material, and workshops from resources such as library instruction or Carpentry sessions were the third most popular learning source. Participants were given a free-text box to list their favorite learning resources, and many gave credit to Jenny Bryan, software engineer at RStudio, and her Twitter account for tips and tricks. Additionally, other avenues of learning include trial and error, and colleagues and friends for emotional support.

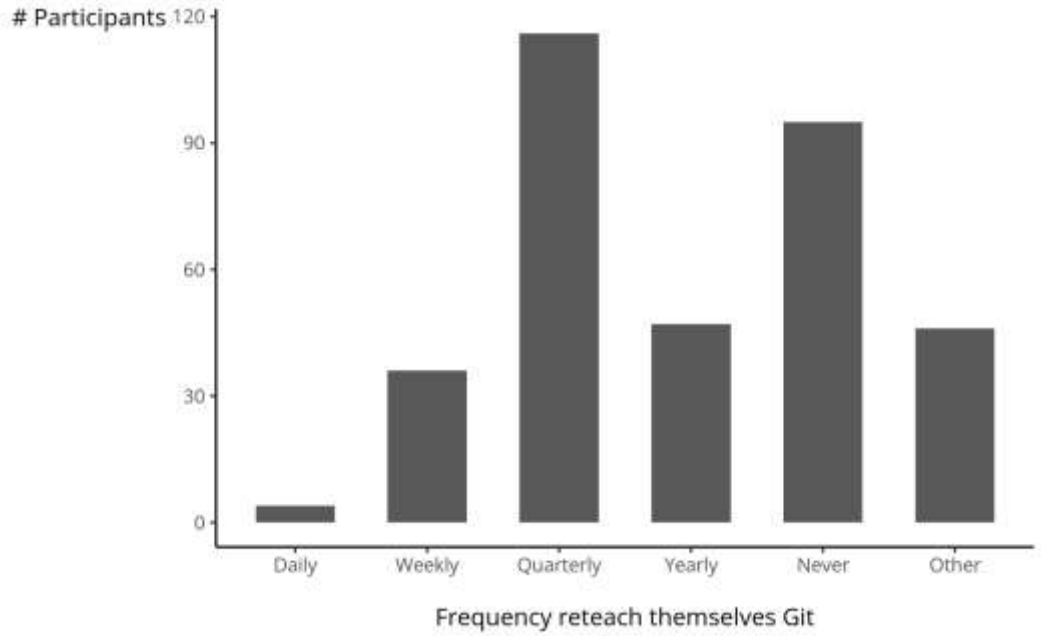

Figure 5. Frequency with which participants re-teach themselves Git.

In addition to questions about how participants learn Git, we also asked about teaching. 198 participants indicated they have taught Git to others (144 indicated they have not). 82% of those participants do not teach regularly (leaving 18% that do teach Git regularly). Of those who teach: 55 make original instructional materials, 97 mix original and secondary materials, and 43 reuse materials from others. These participants teach a lot in-person (180), but there were also those who taught virtually asynchronously (28) and synchronously (43). Participants were asked what resource would help them teach more effectively, and answers included: a helper in the room (online or in-person), hosted/curated computational environments for students (to avoid installation problems), more time to thoughtfully integrate into existing courses, and a concise handout with graphics.

Research & Sustainability

While many participants reported their motivations for using Git in terms of technical features or the facilitation of collaborating with co-researchers, there are also external factors which guide researchers use for such tools. In 2011, the rate of using GHP has increased due to funding agencies like the National Institute of Health, the National Science Foundation (Rubenstein, 2012), and the National Endowment for the Humanities mandating that publications and authors make raw data and/or source code available on an openly accessible repository or archive. Many public and private investors followed suit and included this mandate into their required research practices. In this section we explore participants' reported operations in how Git and GHP are used in research (particularly with respect to collaboration), their funder's expectations, and the scholarly ephemera they most use to accomplish their goals.

Among the scholars and academics we surveyed, of those who answered 44 were required by their funder to publish their research materials openly. 14 were not required to, and 10 were unsure whether or not they had such a requirement. The rough percentages align with the type of funding received; 220 participants had funding from public sources (e.g. federal government), 101 were funded privately, 74 did not have research funding, 18 selected 'other,' and 6 did not know how their research was funded. It is encouraging to know there is more incentive to use Git or GHP in service of open research beside a researcher's funding source.

That said, most participants' primary interest in using Git or a GHP was to facilitate their research workflow in collaboration with others. This brings up concerns on how researchers

onboarded new collaborators to share the source code, expectations among group methods, and standardized language to communicate changes. First, most participants collaborated in the coding process with colleagues using Git (253, with 75 not working collaboratively). From there, the survey drilled down into how these teams set their collaboration environment. Participants reported that when a new collaborator joins their team, the onboarding process or protocol was not likely to be introduced to team members, particularly when introducing expected coding and version control practices (150 participants do not have any form of onboarding procedures in their practice, 84 do, and 16 do not know). This lack of setting agreed upon practices in a research team can be of concern in regards to the long-term legibility and reusability of source code for future research initiatives. It also hampers newcomers in learning the toolkits themselves and how best to go about integrating Git and GHP into their research.

As mentioned earlier, the use of GHP as opposed to Git locally determines the amount of contributions made to artifacts that are considered scholarly ephemera (see Table 3) vs. direct code contributions. These scholarly ephemera provide rich insight into the particular activities and experiences that the participants engaged in within their collaborations—using GHP was a means "to be involved in a larger conversation" (Milliken et al., 2021). As expected, the majority of the participants reported collaboration as their activity to engage on GHP, followed by reproducibility, education and teaching, publishing, then peer review (see Figure 3 in Milliken, et al., 2021).

In the final part of the survey, we asked participants about their archival strategies for this research material, including pre-custodial labor practices and deposit practices. Interestingly, 154 participants deposit their code into a repository or archive for long-term access and reuse (157 did not, and 13 did not know). The most popular repository is Zenodo, followed by the Open Science Framework (OSF), then institutional repositories. Zenodo's official integration with GitHub (Potter & Smith, 2015), the most popular GHP, could be a factor in its popularity amongst scholars.

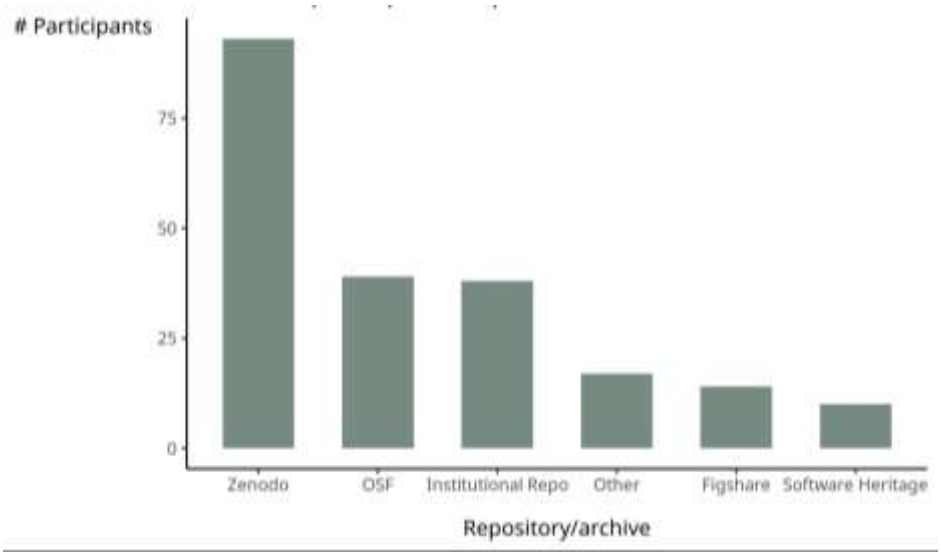

Figure 6. Count of participants' usage of different repositories/archives.

Finally, participants were asked if they would use an academic-specific, user-friendly GHP. It was an overwhelming "maybe" (205 participants)—66 explicitly said no and 53 explicitly said yes. Popularity and ubiquitousness of use are criterion that participants expressed when choosing tools, which may explain this.

Discussion

This study explores a wide-ranging census of academic coders who use Git and GHP as a part of their scholarly work. It is the first of its kind in academia, with a few industry counterparts (GitHub Team, 2021; StackOverflow, 2018). Information gleaned about daily practices, teaching and learning strategies, research workflows, and sustainability of materials can support the interests of GLAM workers concerned about the preservation and long-term access and use of scholarly code.

First and foremost, this survey was paired with a synchronous study to prime the exploration of understanding how source code and annotations on GHP can "move from a phase where they are highly active and collaborative, to a state where they are stable, permanently citable, and under active, professional preservation" (Steeves & Millman, 2018). We are particularly interested in how the preservation process of source code can be operationalized within Galleries, Libraries, Archives, and Museums (GLAMs), alongside other research materials such as manuscripts and data. By understanding who is actively using or ramping up to use GHP or Git, for any of the many benefits which version control and collaboration provide, we are then able to better understand the diversity of preservation methods that have been adopted across the academic landscape and how we, as GLAM workers, can optimize digital preservation services and curation practices for what is important to both the creators and (re)users of scholarly work.

As found in Milliken's (2020) environmental scan of preserving source code and its contextual scholarly ephemera, there are four distinct practical themes through which source code preservation has been explored and operationalized: self-archiving source code, programmatic captures, web archiving, and software preservation. In self archiving work, it is common for scholars to be motivated by open scholarship initiatives and submit their source code and development repositories to the likes of the Open Science Framework, Zenodo, and institutional repositories. While these individualized preservation efforts provide platforms for the source code repositories to be held for long-term storage, their use is not yet ubiquitous in academia. Institutional and programmatic archiving efforts are necessary to fill in this gap and ensure that these important research materials are preserved.

There are projects and institutions which have explored how to programmatically preserve public source code (not specific to research code but inclusive of it) and its ephemera, but these exist separately. Examples include GHTorrent, GH Archive, and Software Heritage. GHTorrent and GH Archive "are open source projects that allow large scale mining of data about repositories on GitHub and provide queryable ways to interact with it" (Milliken, 2020), providing access to the scholarly ephemera without the source code. On the flip side, Software Heritage "collect and preserve software in source code form" without the scholarly ephemera to contextualize it (Inria, 2019). There is potential in nascent projects, such as Software Archiving of Research Artefacts (SARA, 2020), that seek to provide platform integrations to save both the code with the ephemera, though the approach relies on researchers first initiating a deposit (and then changes can be mirrored). Then, there is the Web archiving approach to getting both the research code and its ephemera, since GHP are on the Web. While this approach is encouraging for the ephemera, getting re-runnable code out of WARC or WACZ files remains an area of work.

In a broad sense, all of the above efforts promote the same initiatives to preserve software as a whole. The Software Sustainability Institute is one of a handful of organizations that has formed a community of practice between researchers and funders to advocate for the more sustainable software practices, championing a relatively new role in academia—Research Software Engineers (Software Sustainability Institute, 2019). Through these communities of practice, best practices for software management have been published across researchers and developers. To this point and due to high demand from community members (Chue Hong et al., 2019), GitHub, the most mainstream GHP, now recognizes the significance of citation files (Smith, 2021).

The work in this survey and in IASGE as a whole has led to another project which combines the behavioral data captured during the project (the survey results presented in addition to the interviews and focus groups data) with the environmental scan of archival approaches in order to operationalize archiving Git repositories on the Web at scale, with the long-term of preserving re-

runnable and reproducible research (understanding that the first step is to first have the materials to run). This project, Collaborative Software Archiving for Institutions (CoSAI), will explore and develop "a decentralized and federated platform that will knit together several existing archiving and software preservation tools [... so that] no one institution can be a bottleneck or failure point for archiving workflows", and allowing for shared costs among partners and implementing one of the gold standards in archiving (Rampin & Klein, 2021). CoSAI is a collaborative project, co-PIs being Rampin at NYU and Klein at Los Alamos National Laboratory, with project partners at Old Dominion University and University of Pittsburgh.

By openly publishing this survey data, we hope to provide the means for community researchers to reuse this data and expand upon our findings described in this paper. There is potential to not only investigate scholars' interactions with Git and GHP from a technical perspective of systems and study of scholarship, but also to look at the social implications of who tends to become the software manager, stewarding the sustainability of software, including its preservation, and long term use. We invite others to explore this survey data, as well as any data from the project, in ways that could improve both information and preservation systems, as well as learning, teaching, and maintenance work.

Conclusion

This work explores the behaviors, motivations, histories, and demographics of scholars who code—specifically how they interact with Git locally and on the Web. By better understanding patrons, GLAM workers can tailor services to better serve the goal of preserving reproducible research. Understanding of scholars' behaviors can assist GLAM workers with pre-custodial labor, facilitating the use, learning, and continued reuse of source code repositories using tools such as Git. Instruction and outreach around key areas such as documenting software dependencies (which can make-or-break reproducibility), writing onboarding documentation for collaborators (facilitating intra-group reproducibility), and preparing software for deposit into a repository are particular areas where interventions are necessary. This usage data also serves GLAM workers who want to understand the important facets of research software to preserve for future reuse, as well as the current strategies employed by potential patrons. Research software is foundational to the sustainability of the scholarly record; so much relies on it. Understanding its authors, maintainers, and contributors paves the way for preservation and reuse of research across areas of study.

Acknowledgements

The Investigating & Archiving the Scholarly Git Experience project was funded by the Alfred P. Sloan Foundation.

References

aljedaxi. (2019). Aljedaxi/peer_production_license [TeX]. Retrieved from https://github.com/aljedaxi/peer_production_license (Original work published 2019)

Allen, C., & Mehler, D. M. A. (2019). Open science challenges, benefits and tips in early career and beyond. PLOS Biology, 17(5), e3000246. https://doi.org/10.1371/journal.pbio.3000246

Astropy [Python]. (2022). The Astropy Project. Retrieved from https://github.com/astropy/astropy (Original work published 2011)

Benedetto, V. D., Podstavkov, V., Fattoruso, M., & Coimbra, B. (2019). Continuous Integration service at Fermilab. EPJ Web of Conferences, 214, 05009. https://doi.org/10.1051/epjconf/201921405009

Britton, J. (2014, February 11). GitHub goes to school. Retrieved March 13, 2022, from The GitHub Blog website: https://github.blog/2014-02-10-github-goes-to-school/

CARPi, N., Minges, A., & Piel, M. (2017). eLabFTW: An open source laboratory notebook for research labs (Version 1.4.0) [PHP]. https://doi.org/10.21105/joss.00146

Chacon, S., & Straub, B. (2014). Pro Git Book. Apress. Retrieved from https://git-scm.com/book/en/v2

Chassanoff, A., & Altman, M. (2019). Curation as "Interoperability With the Future": Preserving Scholarly Research Software in Academic Libraries. Journal of the Association for Information Science and Technology, 0(0). https://doi.org/10.1002/asi.24244

Chue Hong, N. P., Allen, A., Gonzalez-Beltran, A., de Waard, A., Smith, A. M., Robinson, C., ... Pollard, T. (2019). Software Citation Checklist for Authors. Zenodo. https://doi.org/10.5281/zenodo.3479199

Croxall, B. (2012, March 22). Forking Your Syllabus [News]. Retrieved April 25, 2019, from ProfHacker: Teaching, tech, and productivity. website: https://www.chronicle.com/blogs/profhacker/forking-your-syllabus/39137

Dallas, T. (2019). Taddallas/LabNotebook. Retrieved from https://github.com/taddallas/LabNotebook (Original work published 2015)

GitHub Team. (2021). The State of the Octoverse. GitHub. Retrieved from GitHub website: https://octoverse.github.com/#sustainable-communities

Glassey, R. (2019). Adopting Git/Github Within Teaching: A Survey of Tool Support. Proceedings of the ACM Conference on Global Computing Education, 143–149. New York, NY, USA: ACM. https://doi.org/10.1145/3300115.3309518

Hettrick, S. (2014, December 4). It's impossible to conduct research without software, say 7 out of 10 UK researchers. Retrieved from Software Sustainability Institute website: https://www.software.ac.uk/blog/2014-12-04-its-impossible-conduct-research-without-software-say-7-out-10-uk-researchers

Hrynaszkiewicz, I. (2013, February 28). Social coding and scholarly communication – open for collaboration [Blog]. Retrieved from Research in progress blog website: http://blogs.biomedcentral.com/bmcblog/2013/02/28/github-and-biomed-central/

Kaki, G., Sivaramakrishnan, K., & Jagannathan, S. (2019). Version Control Is for Your Data Too. 19. https://doi.org/10.4230/LIPIcs.SNAPL.2019.8

Krafczyk, M., Shi, A., Bhaskar, A., Marinov, D., & Stodden, V. (2019). Scientific Tests and Continuous Integration Strategies to Enhance Reproducibility in the Scientific Software Context. 2nd International Workshop on Practical Reproducible Evaluation of Computer Systems (P-RECS'19). Presented at the ACM Federated Computing Research Conference, Phoenix, Arizona. https://doi.org/10.1145/3322790.3330595

Laadan, O., Nieh, J., & Viennot, N. (2010). Teaching Operating Systems Using Virtual Appliances and Distributed Version Control. Proceedings of the 41st ACM Technical Symposium on Computer Science Education, 480–484. New York, NY, USA: ACM. https://doi.org/10.1145/1734263.1734427

Lamprecht, A.-L., Garcia, L., Kuzak, M., Martinez, C., Arcila, R., Martin Del Pico, E., ... Capella-Gutierrez, S. (2019). Towards FAIR principles for research software. Data Science, Preprint, 1–23. https://doi.org/10.3233/DS-190026

Langille, M. G. I., Ravel, J., & Fricke, W. F. (2018). "Available upon request": Not good enough for microbiome data! Microbiome Journal, 6(8). https://doi.org/10.1186/s40168-017-0394-z

Lawson, K. (2013, March 26). Forks and Pull Requests in GitHub [News]. Retrieved April 25, 2019, from ProfHacker: Teaching, tech, and productivity. website: https://www.chronicle.com/blogs/profhacker/forks-and-pull-requests-in-github/47753

Meloni, J. (2019, March 25). A Gentle Introduction to Version Control [News]. Retrieved April 25, 2018, from ProfHacker: Teaching, tech, and productivity. website: https://www.chronicle.com/blogs/profhacker/a-gentle-introduction-to-version-control/23064

Metz, C. (2015, September 22). GitHub Open Sources a Tool That Teaches Students to Code. Wired. Retrieved from https://www.wired.com/2015/09/github-open-sources-tool-teaches-students-code/

Milliken, G. (2020). Environmental Scan. New York University. Retrieved from New York University website: https://osf.io/wx6zs/

Milliken, G., Nguyễn, S., & Rampin, V. (2019, October 18). Defining Scholarly Ephemera. Retrieved June 20, 2022, from IASGE website: https://investigating-archiving-git.gitlab.io/updates/define-scholarly-ephemera/

Milliken, G., Nguyen, S., & Steeves, V. (2021, January 5). A Behavioral Approach to Understanding the Git Experience. Presented at the Hawaii International Conference on System Sciences. https://doi.org/10.24251/HICSS.2021.872

Müller, K., & Bryan, J. (2020). here: A Simpler Way to Find Your Files (Version 1.0.1). Retrieved from https://CRAN.R-project.org/package=here

National Center for Science and Engineering Statistics Appendix B Science and Engineering Degrees: 1966–2010. (2013, June). Retrieved June 13, 2022, from US National Science Foundation (NSF) website: https://www.nsf.gov/statistics/nsf13327/content.cfm?pub_id=4266&id=4

Pedersen, T. L. (2020). patchwork: The Composer of Plots (Version 1.1.1). Retrieved from https://CRAN.R-project.org/package=patchwork

Perez, F. (2012, January 8). The IPython notebook: A historical retrospective. Retrieved March 13, 2022, from http://blog.fperez.org/2012/01/ipython-notebook-historical.html

Potter, M., & Smith, T. (2015). Making code citable with Zenodo and GitHub. Retrieved July 8, 2019, from Software Sustainability Institute website: https://www.software.ac.uk/blog/2016-09-26-making-code-citable-zenodo-and-github

Raj, A. (2016, March 3). From over-reproducibility to a reproducibility wish-list [Blog]. Retrieved April 25, 2019, from RajLab website: http://rajlaboratory.blogspot.com/2016/03/from-over-reproducibility-to.html

Ram, K. (2013). Git can facilitate greater reproducibility and increased transparency in science. Source Code for Biology and Medicine, 8(7), 8. https://doi.org/10.1186/1751-0473-8-7

Rampin, V. (2022). IASGE Survey Analysis. Retrieved from https://gitlab.com/investigating-archiving-git/survey-analysis

Rampin, V., & Klein, M. (2021). Collaborating on Software Archiving for Institutions. Retrieved June 20, 2022, from https://sloan.org/grant-detail/9628

Ross, J. (2019). Introduction to Programming for Data Science and Visualization: INFX 511 [Git repository]. Retrieved May 12, 2019, from Infx511/infx511.github.io website: https://github.com/infx511

Rubenstein, M. A. (2012, October 4). Dear Colleague Letter—Issuance of a new NSF Proposal & Award Policies and Procedures Guide [Letter]. Retrieved from https://www.nsf.gov/pubs/2013/nsf13004/nsf13004.jsp?WT.mc_id=USNSF_109

SARA. (2020). SARA - Software Archiving of Research Artefacts. Retrieved November 24, 2019, from https://www.sara-service.org/

Smith, A. (2021, August 19). Enhanced support for citations on GitHub. Retrieved June 13, 2022, from The GitHub Blog website: https://github.blog/2021-08-19-enhanced-support-citations-github/

Software Sustainability Institute. (2019). The Software Sustainability Institute. Retrieved July 1, 2019, from https://www.software.ac.uk/

Soiland-Reyes, S., Gamble, M., & Haines, R. (2014). Research Object Bundle 1.0. https://doi.org/10.5281/ZENODO.12586

Squire, M. (2017). The Lives and Deaths of Open Source Code Forges. Proceedings of the 13th International Symposium on Open Collaboration, 1–8. Galway Ireland: ACM. https://doi.org/10.1145/3125433.3125468

StackOverflow. (2018). Stack Overflow Developer Survey. Retrieved July 15, 2020, from Stack Overflow website: https://insights.stackoverflow.com/survey/2018

Steeves, V., & Millman, D. (2018). Investigating & Archiving the Scholarly Git Experience. Retrieved June 20, 2022, from https://sloan.org/grant-detail/8723

Steeves, V., & Nguyễn, S. (2020). Investigating and Archiving the Scholarly Git Experience [Data set]. Qualitative Data Repository. https://doi.org/10.5064/F6VOIB8H

Steeves, V., Rampin, R., & Chirigati, F. (2018). Using ReproZip for Reproducibility and Library Services. IASSIST Quarterly, 42(1), 14–14. https://doi.org/10.29173/iq18

Stodden, V., Leisch, F., Peng, R. D., Leisch, F., & Peng, R. D. (2018). Implementing Reproducible Research. New York, NY: Chapman and Hall/CRC. https://doi.org/10.1201/9781315373461

Torvald, L. (n.d.). Git Documentation. Retrieved March 7, 2022, from Git website: https://git-scm.com/docs/git

Vanderplas, J. (2016, April). Driving Reproducibility at UW. Presentation presented at the NYU Reproducibility Symposium, Brooklyn, NY.

Vanderplas, J. (2021). Mutiband Lomb-Scargle Periodograms [TeX]. Retrieved from https://github.com/jakevdp/multiband_LS (Original work published 2014)

Wickham, H. (2007). Reshaping Data with the reshape Package. Journal of Statistical Software, 21, 1–20. https://doi.org/10.18637/jss.v021.i12

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L., François, R., ... Yutani, H. (2019). Welcome to the Tidyverse. Journal of Open Source Software, 4(43), 1686. https://doi.org/10.21105/joss.01686

Widder, D. G., Hilton, M., Kästner, C., & Vasilescu, B. (2019). A Conceptual Replication of Continuous Integration Pain Points in the Context of Travis CI. Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 647–658. Tallinn, Estonia: Association for Computing Machinery. https://doi.org/10.1145/3338906.3338922

Wikidata Query: Academic discipline. (2020). Retrieved June 13, 2022, from Wikidata website: https://query.wikidata.org/embed.html#SELECT%20%3Facademic_discipline%20%3FacademicdisciplineLabel%20WHERE%20%7B%0A%20%20SERVICE%20wikibase%3Alabel%20%7B%20bd%3AserviceParam%20wikibase%3Alanguage%20%22%5BAUTO_LANGUAGE%5D%2Cen%22.%20%7D%0A%20%20%3Facademic_discipline%20wdt%3AP31%20wd%3AQ11862829.%0A%7D%0ALIMIT%20100

Wilkinson, M. D., Dumontier, M., Aalbersberg, Ij. J., Appleton, G., Axton, M., Baak, A., ... Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data, 3(1), 1–9. https://doi.org/10.1038/sdata.2016.18

Yenni, G. M., Christensen, E. M., Bledsoe, E. K., Supp, S. R., Diaz, R. M., White, E. P., & Ernest, S. K. M. (2018). Developing a modern data workflow for evolving data. BioRxiv, 344804. https://doi.org/10.1101/344804

Zotero [JavaScript]. (2022). Zotero. Retrieved from https://github.com/zotero/zotero (Original work published 2011)