git can facilitate greater reproducibility and increased transparency in science.

Karthik Ram, Ph.D.
Environmental Science, Policy, and Management.
University of California, Berkeley.
Berkeley, CA 94720. USA.
karthik.ram@berkeley.edu

Abstract

Reproducibility is the hallmark of good science. Maintaining a high degree of transparency in scientific reporting is essential not just for gaining trust and credibility within the scientific community but also for facilitating the development of new ideas. Sharing data and computer code associated with publications is becoming increasingly common, motivated partly in response to data deposition requirements from journals and mandates from funders. Despite this increase in transparency, it is still difficult to reproduce or build upon the findings of most scientific publications without access to a more complete workflow.

Version control systems (VCS), which have long been used to maintain code repositories in the software industry, are now finding new applications in science. One such open-source VCS, git, provides a lightweight yet robust framework that is ideal for managing the full suite of research outputs such as datasets, statistical code, figures, lab notes, and manuscripts. For individual researchers, git provides a powerful way to track and compare versions, retrace errors, explore new approaches in a structured manner, while maintaining a full audit trail. For larger collaborative efforts, git and git hosting services make it possible for everyone to work asynchronously and merge their contributions at any time, all the while maintaining a complete authorship trail. In this paper I provide an overview of git along with use-cases that highlight how this tool can be leveraged to make science more reproducible and transparent, foster new collaborations, and support novel uses.

Keywords

reproducible research, version control, open science.

Background

Reproducible science provides the critical standard by which published results are judged and central findings are either validated or refuted [1]. Reproducibility also allows others to build upon existing work and use it to test new ideas and develop methods. Advances over the years have resulted in the development of complex methodologies that allow us to collect ever increasing amounts of data. While repeating expensive studies to validate findings is often difficult, a whole host of other reasons have contributed to the problem of reproducibility [2,3]. One such reason has been the lack of detailed access to underlying data and statistical code used for analysis, which can provide opportunities for others to verify findings [4,5]. In an era rife with costly retractions, scientists have an increasing burden to be more transparent in order to maintain their credibility [6]. While post-publication sharing of data and code is on the rise, driven in part by funder mandates and journal requirements [7], access to such research outputs is still not very common [8,9]. By sharing detailed and versioned copies of one's data and code researchers can not only ensure that reviewers can make well-informed decisions, but also provide opportunities for such artifacts to be repurposed and brought to bear on new research questions.

Opening up access to the data and software, not just the final publication, is one of goals of the open science movement. Such sharing can lower barriers and serve as a powerful catalyst to accelerate progress. In the

era of limited funding, there is a need to leverage existing data and code to the fullest extent to solve both applied and basic problems. This requires that scientists share their research artifacts more openly, with reasonable licenses that encourage fair use while providing credit to original authors [10]. Besides overcoming social challenges to these issues, existing technologies can also be leveraged to increase reproducibility.

All scientists use version control in one form or another at various stages of their research projects, from the data collection all the way to manuscript preparation. This process is often informal and haphazard, where multiple revisions of papers, code, and datasets are saved as duplicate copies with uninformative file names (e.g. *draft_1.doc, draft_2.doc*). As authors receive new data and feedback from peers and collaborators, maintaining those versions and merging changes can result in an unmanageable proliferation of files. One solution to these problems would be to use a formal Version Control System (VCS), which have long been used in the software industry to manage code. A key feature common to all types of VCS is that ability save versions of files during development along with informative comments which are referred to as commit messages. Every change and accompanying notes are stored independent of the files, which obviates the need for duplicate copies. Commits serve as anchor points where individual files or an entire project can be safely reverted to when necessary. Most traditional VCS are centralized which means that they require a connection to a central server which maintains the master copy. Users with appropriate privileges can check out copies, make changes, and upload them back to the server.

Among the suite of version control systems currently available, **git** stands out in particular because it offers features that make it ideal for managing artifacts of scientific research. The most compelling feature of git is its decentralized and distributed nature. Every copy of a git repository can serve either as the server (a central point for synchronizing changes) or as a client. This ensures that there is no single point of failure. Authors can work asynchronously without being connected to a central server and synchronize their changes when possible. This is particularly useful when working from remote field sites where internet connections are often slow or non-existent. Unlike other VCS, every copy of a git repository carries a complete history of all changes, including authorship, that can be viewed and searched by anyone. This feature allows new authors to build from any stage of a versioned project. git also has a small footprint and nearly all operations occur locally.

By using a formal VCS, researchers can not only increase their own productivity but also make it for others to fully understand, use, and build upon their contributions. In the rest of the paper I describe how git can be used to manage common science outputs and move on to describing larger use-cases and benefits of this workflow. Readers should note that I do not aim to provide a comprehensive review of version control systems or even git itself. My goal here is to broadly outline some of advantages of using one such system and how it can benefit individual researchers, collaborative efforts, and the wider research community.

How git can track various artifacts of a research effort

Before delving into common use-cases, I first describe how git can be used to manage familiar research outputs such as data, code used for statistical analyses, and documents. git can be used to manage them not just separately but also in various combinations for different use cases such as maintaining lab notebooks, lectures, datasets, and manuscripts.

Manuscripts and notes

Version control can operate on any file type including ones most commonly used in academia such as Microsoft Word. However, since these file types are binary, git cannot examine the contents and highlight sections that have changed between revisions. In such cases, one would have to rely solely on commit messages or scan through file contents. The full power of git can best be leveraged when working with plain-text files. These include data stored in non-proprietary spreadsheet formats (e.g. comma separated files versus `xls`), scripts from programming languages, and manuscripts stored in plain text formats (`LaTeX` and `markdown` versus Word documents). With such formats, git not only tracks versions but can also highlight which sections of a file have changed.

In Microsoft Word documents the *track changes* feature is often used to solicit comments and feedback. Once those comments and changes have either been accepted or rejected, any record of their existence also disappears forever. When changes are submitted using git, a permanent record of author contributions remains in the version history and available in every copy of the repository.

Datasets

Data are ideal for managing with git. These include data manually entered via spreadsheets, recorded as part of observational studies, or ones retrieved from sensors. With each significant change or additions, commits can record a log those activities (e.g. "*Entered data collected between 12/10/2012 and 12/20/2012*", or "*Updated data from temperature loggers for December 2012*"). Over time this process avoids proliferation of files, while the git history maintains a complete provenance that can be reviewed at any time. When errors are discovered, earlier versions of a file can be reverted without affecting other assets in the project.

Statistical code and figures

When data are analyzed programmatically using software such as `R` and `Python`, code files start out small and often become more complex over time. Somewhere along the process, inadvertent errors such as misplaced subscripts and incorrectly applied functions can lead to serious errors down the line. When such errors are discovered well into a project, comparing versions of statistical scripts can provide a way to quickly trace the source of the problem and recover from them.

Similarly, figures that are published in a paper often undergo multiple revisions before resulting in a final version that gets published. Without version control, one would have to deal with multiple copies and use imperfect information such as file creation dates to determine the sequence in which they were generated. Without additional information, figuring out why certain versions were created (e.g. in response to comments from coauthors) also becomes more difficult. When figures are managed with git, the commit messages (e.g. "*Updated figure in response to Ethan's comments regarding use of normalized data.*") provide an unambiguous way to track various versions.

Complete manuscripts

When all of the above artifacts are used in a single effort, such as writing a manuscript, git can collectively manage versions in a powerful way for both individual authors and groups of collaborators. This process avoids rapid multiplication of unmanageable files with uninformative names (e.g. *final_1.doc, final_2.doc, final_final.doc, final_KR_1.doc* etc.) as illustrated by the popular cartoon strip PhD Comics.
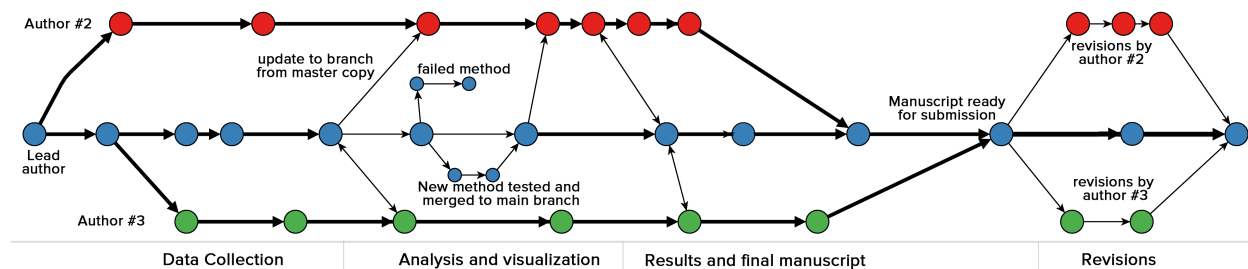

Figure 1: A hypothetical git workflow for a scientific collaboration involving three authors. Each circle represents a commit and colors denote author specific commits. Two way arrows indicate a sync (a push and pull in git terminology). One way arrows indicate an update to one branch from another. Horizontal arrows indicate development along a particular branch.

Use cases for git in science

1. Lab notebook

Day to day decisions made over the course of a study are often logged for review and reference in lab notebooks. Such notebooks contain important information useful to both future readers attempting to replicating a study, or for thorough reviewers seeking additional clarification. However, lab notebooks are rarely shared along with publications or made public although there are some exceptions [11]. Git commit logs can serve as a proxies for lab notebooks if clear yet concise messages are recorded over the course of a project. One of the fundamental features of git that make it so useful to science is that every copy of a repository carries a complete history of changes available for anyone to review. These logs can be be easily searched to retrieve versions of artifacts like data and code. Third party tools can also be leveraged to mine git histories from one or more projects for other types of analyses.

2. Facilitating Collaboration

In collaborative efforts, authors contribute to one or more stages of the manuscript preparation such as collecting data, analyzing them, and/or writing up the results. Such information is extremely useful for both readers and reviewers when assessing relative author contributions to a body of work. With high profile journals now discouraging the practice of honorary authorship [12], git commit logs can provide a highly granular way to track and assess individual author contributions to a project.

When projects are tracked using git, every single action (such as additions, deletions, and changes) is attributed to an author. Multiple authors can choose to work on a single branch of a repository (the '*master*' branch), or in separate branches and work asynchronously. In other words, authors do not have to wait on coauthors before contributing. As each author adds their contribution, they can sync those to the master branch and update their copies at any time. Over time, all of the decisions that go into the production of a manuscript from entering data and checking for errors, to choosing appropriate statistical models and creating figures, can be traced back to specific authors.

With the help of a remote git hosting services, maintaining various copies in sync with each other becomes effortless. While most changes are merged automatically, conflicts will need to be resolved manually which would also be the case with most other workflows (e.g. using Microsoft Word with track changes). By syncing changes back and forth with a remote repository, every author can update their local copies as well as push their changes to the remote version at any time, all the while maintaining a complete audit trail. Mistakes or unnecessary changes can easily undone by reverting either the entire repository or individual files to earlier commits. Since commits are attributed to specific authors, error or clarifications can also be appropriately directed. Perhaps most importantly this workflow ensures that revisions do not have to be emailed back and forth. While cloud storage providers like Dropbox alleviate some of these annoyances and also provide versioning, the process is not controlled making it hard to discern what and how many changes have occurred between two time intervals.

In a recent paper led by Philippe Desjardins-Proulx https://github.com/PhDP/article_preprint/network all of the authors successfully collaborated using only git and GitHub (https://github.com/). In this particular git workflow, each of us cloned a copy of the main repository and contributed our changes back to the lead author. Figures 2 and 3 show the list of collaborators and a network diagram of how and when changes were contributed back the master branch.

3. Backup and failsafe against data loss

Collecting new data and developing methods for analysis are often expensive endeavors requiring significant amounts of grant funding. Therefore protecting such valuable products from loss or theft is paramount. A recent study found that a vast majority of data and code are stored on lab computers or web servers both of
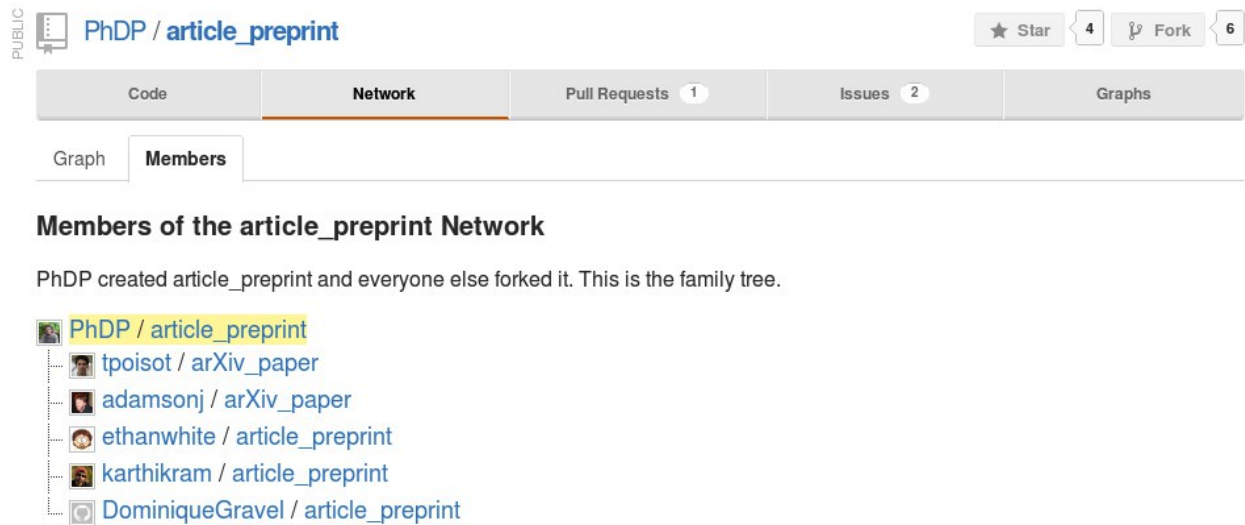
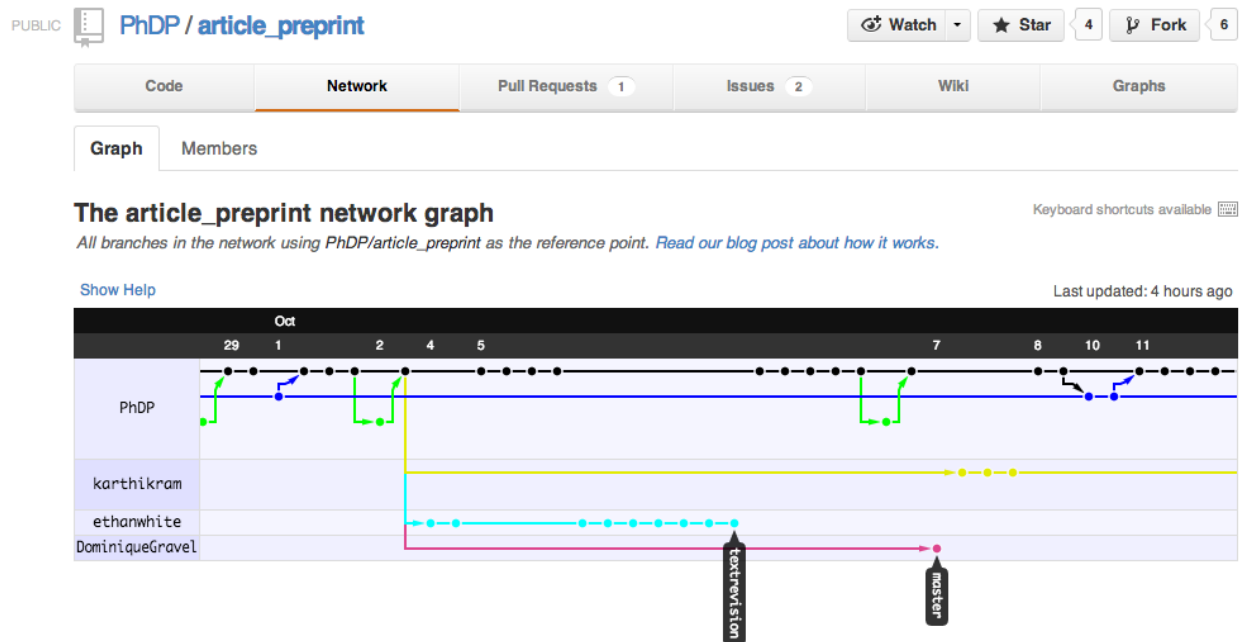Figure 2: A list of contributions to a project on GitHub

Figure 3: git makes it easy to track individual contributions through time ensuring appropriate attribution and accountability. This screenshot shows subset of commits (colored dots) by four authors over a period spanning October 29th - November 11th, 2012.

which are prone to failure and often become inaccessible after a certain length of time. One survey found that only 72% of studies of 1000 surveyed still had data that were accessible [13,14]. Hosting data and code publicly not only ensures protection against loss but also increases visibility for research efforts and provides opportunities for collaboration and early review [15].

While git provides a powerful features that can leveraged by individual scientists, git hosting services open up a whole new set of possibilities. Any local git repository can be linked to one or more **git remotes**, which are copies hosted on a remote cloud severs. Git remotes serve as hubs for collaboration where authors with write privileges can contribute anytime while others can download up-to-date versions or submit revisions with author approval. There are currently several git hosting services such as SourceForge, Google Code, GitHub, and BitBucket that provide free git hosting. Among them, Github has surpassed popular provides like Google Code and SourceForge and hosts over 2 million public repositories at the time of this writing [16,17]. While these services are usually free for publicly open projects, some research efforts, especially those containing embargoed or sensitive data will need to be kept private. There are multiple ways to deal with such situations. For example, certain files can be excluded from git's history, others maintained as private sub-modules, or entire repositories can be made private and opened to the public at a future time. Some git hosts like BitBucket offer unlimited public and private accounts for academic use.

Managing a research project with git provides several safe guards against short-term loss. Frequent commits synced to remote repositories ensure that multiple versioned copies are accessible from anywhere. In projects involving multiple collaborators, the presence of additional copies makes even more difficult to lose work. While git hosting services protect against short-term data loss, they are not a solution for more permanent archiving since none of them offer any such guarantees. For long-term archiving, researchers should submit their git-managed projects to academic repositories that are members of CLOCKSS (http://www.clockss.org/). Output stored on such repositories (e.g. figshare) are archived over a network of redundant nodes and ensure indefinite availability across geographic and geopolitical regions.

4. Freedom to explore new ideas and methods

git tracks development of projects along timelines referred to as *branches*. By default, there is always a master branch (line with blue dots in figure 1). For most authors, working with this single branch is sufficient. However, git provides a powerful branching mechanism that makes it easy for exploring alternate ideas in a structured and documented way without disrupting the central flow of a project. For example, one might want to try an improved simulation algorithm, a novel statistical method, or plot figures in a more compelling way. If these changes don't work out, one could revert changes back to an earlier commit when working on a single master branch. Frequent reverts on a master branch can be disruptive, especially when projects involve multiple collaborators. Branching provides a risk-free way to test new algorithms, explore better data visualization techniques, or develop new analytical models. When branches yield desired outcomes, they can easily be merged into the master copy while unsuccessful efforts can be deleted or left as-is to serve as a historical record (illustrated in figure 1).

Branches can prove extremely useful when responding to reviewer questions about the rationale for choosing one method over another since the git history contains a record of failed, unsuitable, or abandoned attempts. This is particularly helpful given that the time between submission and response can be fairly long. Additionally, future users can mine git histories to avoid repeating approaches that were never fruitful in earlier studies.

5. Mechanism to solicit feedback and reviews

While it is possible to leverage most of core functionality in git at the local level, git hosting services offer additional services such as issue trackers, collaboration graphs, and wikis. These can easily be used to assign tasks, manage milestones, and maintain lab protocols. Issue trackers can be repurposed as a mechanism for soliciting both feedback and review, especially since the comments can easily be linked to particular lines of code or blocks of text. Early comments and reviews for this article were also solicited via GitHub Issues https://github.com/karthikram/smb_git/issues/

6. Increase transparency and verifiability

Methods sections in papers are often succinct to adhere to strict word limits imposed by journal guidelines. This practice is especially common when describing well-known methods where authors assume a certain degree of familiarity among informed readers. One unfortunate consequence of this practice is that any modifications to the standard protocol (typically noted in internal lab notebooks) implemented in a study may not available to the reviewers and readers. However, seemingly small decisions, such as choosing an appropriate distribution to use in a statistical method, can have a disproportionately strong influence on the central finding of a paper. Without access to a detailed history, a reviewer competent in statistical methods has to trust that authors carefully met necessary assumptions, or engage in a long back and forth discussion thereby delaying the review process. Sharing a git repository can alleviate these kinds of ambiguities and allow authors to point out commits where certain key decisions were made before choosing certain approaches. Journals could facilitate this process by allowing authors to submit links to their git repository alongside manuscripts and sharing them with reviewers.

7. Managing large data

git is extremely efficient with managing small data files such as ones routinely collected in experimental and observational studies. However, when the data are particularly large such as those in bioinformatics studies (in the order of tens of megabytes to gigabytes), managing them with git can degrade efficiency and slow down the performance of git operations. With large data files, the best practice would be to exclude them from the repository and only track changes in metadata. This protocol is especially ideal when large datasets do not change often over the course of a study. In situations where the data are large *and* undergo frequent updates, one could leverage third-party tools such as git-annex http://git-annex.branchable.com/ and still seamlessly use git to manage a project.

8. Lowering barriers to reuse

A common barrier that prevents someone from reproducing or building upon an existing method is lack of sufficient details about a method. Even in cases where methods are adequately described, the use of expensive proprietary software with restrictive licenses makes it difficult to use [18]. Sharing code with licenses that encourage fair use with appropriate attribution removes such artificial barriers and encourages readers to modify methods to suit their research needs, improve upon them, or find new applications [10]. With open source software, analysis pipelines can be easily *forked* or branched from public git repositories and modified to answer other questions. Although this process of depositing code somewhere public with appropriate licenses involves additional work for the authors, the overall benefits outweigh the costs. Making all research products publicly available not only increases citation rates [19–21] but can also increase opportunities for collaboration by increasing overall visibility. For example, Niedermeyer & Strohalm [22] describe their struggle with finding appropriate software for comprehensive mass spectrum annotation, and eventually found an open source software which they where able to extend. In particular, the authors cite availability of complete source code along with an open license as the motivation for their choice. Examples of such collaboration and extensions are likely to become more common with increased availability of fully versioned projects with permissive licenses.

A similar argument can be made for data as well. Even publications that deposit data in persistent repositories rarely share the original raw data. The versions submitted to persistent repositories are often *cleaned* and finalized versions of datasets. In cases where no datasets are deposited, the only data accessible are likely mean values reported in the main text or appendix of a paper. Raw data can be leveraged to answer questions not originally intended by the authors. For example, research areas that address questions about uncertainly often require messy raw data to test competing methods. Thus, versioned data provide opportunities to retrieve copies before they have been modified for use in different contexts and have lost some of their utility.

Discussion

Wider use of git has the potential to revolutionize scholarly communication and increase opportunities for reuse, novel synthesis, and new collaborative efforts. With disciplined use of git, individual scientists and labs can ensure that the entire timeline of events that occur over the development of a research project are securely logged in a system that provides security against data loss and encourages risk-free exploration of new ideas and approaches. In an era with shrinking research budgets, scientists are under increasing pressure to produce more with less. If more granular sharing via git reduces time spent developing new software, or repeating expensive data collection efforts, then everyone stands to benefit. Scientists should note that these efforts don't have to viewed as entirely altruistic. In a recent mandate the National Science Foundation [23] has expanded its merit guidelines to include a range of academic products such as software and data, in addition to peer-reviewed publications. With the rise in use of altmetric tools that track and credit such efforts, then everyone can benefit [24].

Although I have laid out various arguments for why more scientists should be using git, one should be careful not to view git as a one stop solution to all the problems facing reproducibility in science. Although the basic features of git can be readily used without any knowledge of command line tools, leveraging the full power of git, especially when working on complex projects where one might encounter unwieldy merge conflicts, comes at a significant learning cost. There are also comparable alternatives to git (e.g. Mercurial) which offer less granularity but are more user-friendly. While time invested in becoming proficient in git would be valuable in the long-term, most scientists do not have the luxury of learning software skills that do not address more immediate problems. Despite the fact that scientists spent considerable time using and creating their own software to address domain specific needs, good programming practices are rarely taught [25]. Therefore wider adoption of useful tools like git will require greater software development literacy among scientists. On a more optimistic note, such literacy is slowly becoming common in the new generation of academics, driven in part by efforts such as Software Carpentry http://software-carpentry.org/ and newer courses taught in graduate curricula (e.g. Programming for biologists taught at Utah State University).

List of Abbreviations

VCS: Version Control System; NSF: National Science Foundation; CSV: Comma Separated Values.

Acknowledgements

Author contributions

KR wrote the manuscript.

Competing interests

I declare that I have no competing interests.

Funding support

Literature Cited

1. Vink CJ, Paquin P, Cruickshank RH (2012) Taxonomy and Irreproducible Biological Science. BioScience 62: 451–452. Available: http://www.bioone.org/doi/abs/10.1525/bio.2012.62.5.3.

2. Peng RD (2011) Reproducible Research in Computational Science. Science 334: 1226–1227. Available: http://www.sciencemag.org/cgi/doi/10.1126/science.1213847.

3. Begley CG, Ellis LM (2012) Drug development: Raise standards for preclinical cancer research. Nature 483: 531–3. Available: http://dx.doi.org/10.1038/483531a.

4. Schwab M, Karrenbach M, Claerbout J (2000) Making Scientific Computations Reproducible. Computing in Science Engineering 2: 61–67. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=881708.

5. Ince DC, Hatton L, Graham-Cumming J (2012) The case for open computer programs. Nature 482: 485–8. Available: http://dx.doi.org/10.1038/nature10836.

6. Van Noorden R (2011) The trouble with retractions. Nature: 6–8.

7. Whitlock MC, McPeek MA, Rausher MD, Rieseberg L, Moore AJ (2010) Data archiving. The American naturalist 175: 145–6. Available: http://www.jstor.org/stable/10.1086/650340.

8. Vines TH, Andrew RL, Bock DG, Franklin MT, Gilbert KJ, et al. (2013) Mandated data archiving greatly improves access to research data. FASEB journal official publication of the Federation of American Societies for Experimental Biology. doi:10.1096/fj.12-218164

9. Wolkovich EM, Regetz J, O'Connor MI (2012) Advances in global change research require open science by individual researchers. Global Change Biology 18: 2102–2110. Available: http://apps.webofknowledge.com/full_record.do?produ

10. Neylon C (2013) Open access must enable open use. Nature 492: 8–9.

11. Wald C (2010) Issues & Perspectives Scientists Embrace Openness. Available: http://sciencecareers.sciencemag.org/career_ Accessed 16 Jan 2013.

12. Greenland P, Fontanarosa PB (2012) Ending honorary authorship. Science (New York, N.Y.) 337: 1019. Available: http://www.sciencemag.org/content/337/6098/1019.short.

13. Schultheiss SJ, Münch M-C, Andreeva GD, Rätsch G (2011) Persistence and availability of Web services in computational biology. PloS one 6: 24914. Available: http://dx.plos.org/10.1371/journal.pone.0024914.

14. Wren JD (2004) 404 not found: the stability and persistence of URLs published in MEDLINE. Bioinformatics (Oxford, England) 20: 668–72. Available: http://bioinformatics.oxfordjournals.org/content/20/5/668.abstract.

15. Prlić A, Procter JB (2012) Ten Simple Rules for the Open Development of Scientific Software. PLoS Computational Biology 8: 1002802. Available: http://dx.plos.org/10.1371/journal.pcbi.1002802.

16. Pearson DP (2013) GitHub sees 3 millionth member account. Available: http://www.gamesindustry.biz/articles/2013-01-17-github-sees-3-millionth-member-account. Accessed 18 Jan 2013.

17. Finley K (2011) Github Has Surpassed Sourceforge and Google Code in Popularity. Available: http://readwrite.com/2011/06/02/github-has-passed-sourceforge. Accessed 15 Jan 2013.

18. Morin A, Urban J, Sliz P (2012) A quick guide to software licensing for the scientist-programmer. PLoS computational biology 8: 1002598. Available: http://dx.plos.org/10.1371/journal.pcbi.1002598.

19. Piwowar HA, Day RS, Fridsma B (2007) Sharing Detailed Research Data Is Associated with Increased Citation Rate. PLOS One.

20. Piwowar H a (2011) Who shares? Who doesn't? Factors associated with openly archiving raw research data. PloS one 6: 18657. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3135593\&tool=pmcentrez\&render

21. Alsheikh-Ali A a, Qureshi W, Al-Mallah MH, Ioannidis JP a (2011) Public availability of published research data in high-impact journals. PloS one 6: 24357. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=316848

22. Niedermeyer THJ, Strohalm M (2012) mMass as a software tool for the annotation of cyclic peptide tandem mass spectra. PloS one 7: 44913. Available: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3441486\&tool=pmce

23. US NSF - Dear Colleague Letter - Issuance of a new NSF Proposal & Award Policies and Procedures Guide (NSF13004) (2012) US NSF - Dear Colleague Letter - Issuance of a new NSF Proposal & Award Policies and Procedures Guide (NSF13004). Available: http://www.nsf.gov/pubs/2013/nsf13004/nsf13004.jsp?WT.mc_id=USNSF_109. Accessed 11 Nov 2012.

24. Piwowar H (2013) Altmetrics: Value all research products. Nature 493: 159–159. Available: http://www.nature.com/doifinder/10.1038/493159a.

25. Wilson G, Aruliah DA, Brown CT, Hong NPC, Davis M, et al. (2012) Best Practices for Scientific Computing. Arxiv: 6.