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 papers is essential not just for gaining trust and credibility within the scientific community but also essential 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 robust, feature-rich framework that is ideal for managing various artifacts of scientific endeavors such as data, code, figures, and manuscripts. In particular, git is a lightweight and distributed system that records a complete timeline of events that occur over the evolution of a research project. Since the system is distributed, collaborators can work asynchronously and merge their contributions as needed. The decentralized nature ensures that this rich history, including authorship trail, is available with every copy of a repository for anyone to review, contribute, or build upon. In this paper I review reasons why scientists should leverage this software technology in their day to day research workflow to increase reproducibility and transparency, foster collaboration, and support novel synthesis.

Introduction

Reproducible science provides the critical standard by which published results are judged and central findings are validated or refuted (Vink et al., 2012). Reproducibility also allows others to build upon existing work and use it to develop new ideas and methods. Advances in technology over the years have resulted in application of complex methodologies that have enabled us to collect ever increasing amounts of data. While repeating expensive studies is not always possible, a whole host of other reasons have contributed to the problem of reproducibility (Peng, 2011). One such reason has been the lack of sufficient access to underlying data and code used for analysis, which can provide opportunities for others to verify claims. In an era rife with costly retractions, scientists have an increasing burden to be more transparent to maintain a high level of credibility (Van Noorden, 2011). When data and code are shared in sufficient detail, these artifacts can also be repurposed and brought to bear on new research questions not anticipated by the original authors.

Unrestricted sharing of data and code, which is the primary aim of open science, lowers the barriers needed to increase transparency and can also serve as a powerful catalyst that can accelerate progress. In the era of limited funding, we 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 (Neylon, 2013). Besides overcoming social challenges to these issues, existing technologies can also be leveraged to increase reproducibility.

Version control systems (VCS) have long been used in software development for managing code bases. A key feature common to all types of VCS is the ability to save versions of files during development along with informative comments which are referred to as *commit messages*. Commits serve as anchor points where individual files or an entire project can be safely reverted to when necessary. Additionally, VCS also allow for creation of branches where new approaches or ideas can be safely tested without disrupting a project's development and incorporated once all issues have been resolved. Traditional VCS (such as *cvs* and *svn*) 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, **git** stands out in particular because it offers features that make it more 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 unavailable. 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 in the development of a project. git also has a small footprint and nearly all operations occur locally.

The features that make git the popular choice among software developers also make it ideal for managing scientific products. git repositories maintain a complete timeline of events along with notes describing such changes. Perhaps the biggest reason that makes git so well suited for tracking scientific research is that it maintains a history of authorship not just in the original repository but also in every single copy cloned by anyone. This rich history can be searched or mined by future scholars to extract useful bits of code or data, review methods, and check for errors among various other applications.

How git can track various artifacts of a research project

git can be used to manage not only individual items like data, code, figures and text, but also various combinations for different use cases such as maintaining lab notebooks, lectures, and complete manuscripts.

Managing notes and manuscripts

Although git can be used to version any file type, its usefulnesss declines when used with binary files (such as `MS Word` documents), with which git can only detect that the file has changed but not identify the changes. When working with plain text formats such as `markdown` or `LaTeX`, git can track and highlight differences in text between any two versions.

Managing 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 addition, commit messages can record a log of these 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 unmanagable files (the folder or directory only contains a single copy with changes stored in a hidden git folder), while the git history provides a way to browse such changes and revisit files in earlier states when necessary. When errors are discovered, earlier versions of a file can be extracted and corrected without affecting other assets in the project. Furthermore, git saves the differences between versions rather than a full copy of each version, and uses modern compression algorithms, and is therefore very efficient in disk usage.

Managing statistical analyses 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 used functions can lead to serious errors down the line. When such errors are discovered well into a project, comparing earlier versions of the code can provide a way to quickly trace bugs and recover from them. In addition, one can easily experiment with analysis, knowing that he can always revert back to the last commited version of the code.

Figures that are published in a paper often undergo multiple revisions before resulting in a final production copy. Without version control, one would have to deal with multiple copies and use imperfect information such as file creation dates to determine the order in which they were created. Without additional information, working out why certain versions were created (e.g. in response to comments from coauthors) also becomes harder to track. 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.

Managing 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 single authors and for projects involving multiple collaborators. This protocol avoids proliferation of unmanagable files with uninformative names (e.g. *final_1.doc, final_2.doc, final_final.doc, final_KR_1.doc* etc.)

The features that make git the popular choice among software developers also make it ideal for managing scientific products. git repositories maintain a complete timeline of events along with a history of changes. Perhaps the biggest reason that makes git so well suited for tracking scientific research is that it maintains a history of authorship not just in the original repository but also in every single copy cloned by anyone. This rich history can be searched or mined by future scholars to extract useful bits of code or data, review methods, and check for errors among various other applications.

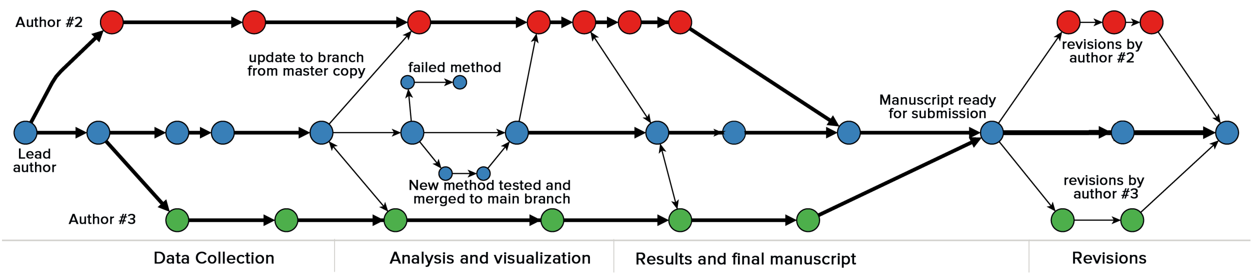

Author #2
update to branch from master copy
failed method
Manuscript ready for submission
revisions by author #2
Lead author
New method tested and merged to main branch
revisions by author #3
Author #3
Data Collection
Analysis and visualization
Results and final manuscript
Revisions

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

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 replicate a study, or for thorough reviewers seeking additional clarification. However, lab notebooks are rarely shared along with publications or made public although there are exceptions (Wald, 2010). Git commit logs can serve as 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.

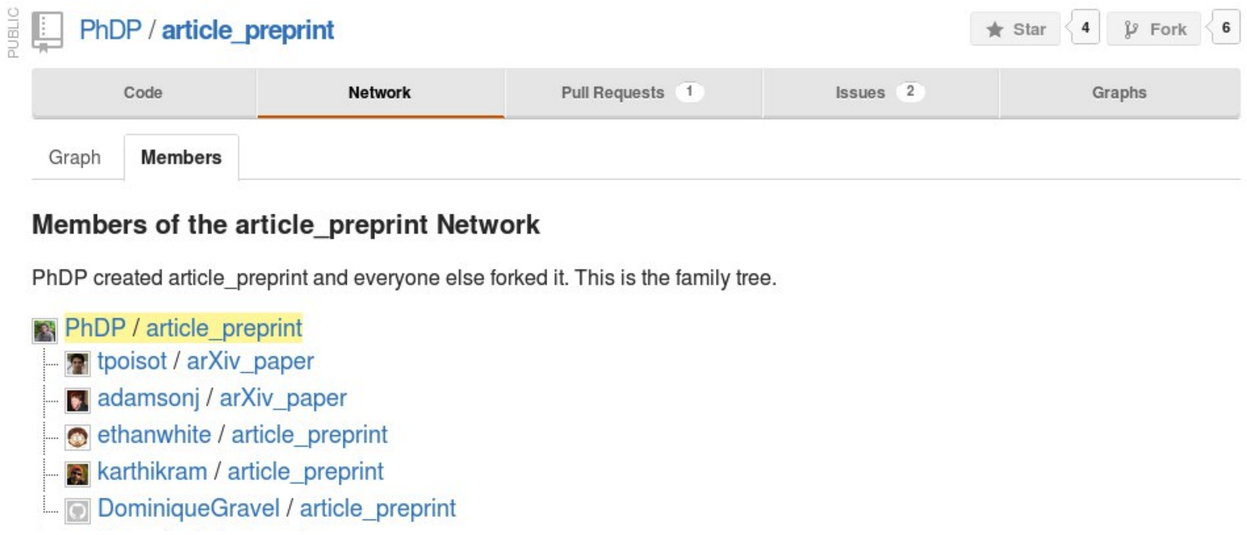
2. Tracking Collaboration

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 (Greenland and Fontanarosa, 2012), 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. As each author adds their contribution, they can choose to sync those to the master branch and/or 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 remote git hosting services, maintaining various copies in sync with each other becomes effortless. While most merges are automatic, conflicts will need to be resolved manually which would also be the case with most other workflows (e.g. using `MS 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 undone easily 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.

In a recent paper led by Philippe Desjardins-Proulx (https://github.com/PhDP/article_preprint/network) all of the authors (including myself) successfully collaborated using only git and the populaer git hosting service GitHub ([https://github.com/](Vink et al., 2012)). In this particular git workflow, each of us cloned a copy of the main repository and contributed our changes back to the original 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.

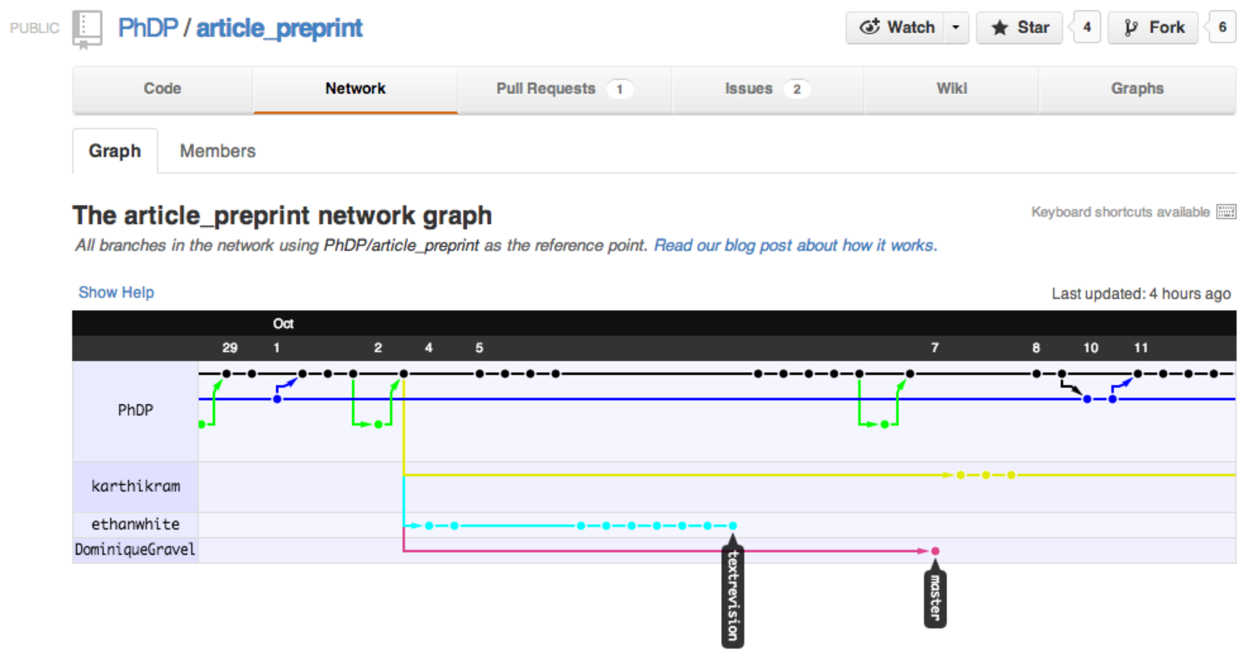


Members of the article_preprint Network

PhDP created article_preprint and everyone else forked it. This is the family tree.

PhDP / article_preprint
- tpoisot / arXiv_paper
- adamsonj / arXiv_paper
- ethanwhite / article_preprint
- karthikram / article_preprint
- DominiqueGravel / article_preprint

Figure 2: A list of contributions to a project on GitHub. An online version is available at `https://github.com/PhDP/article_preprint/network/members`


The article_preprint network graph

All branches in the network using PhDP/article_preprint as the reference point. Read our blog post about how it works.

Last updated: 4 hours ago

Figure 3: git makes it easy to track individual contributions through time ensuring appropriate attribution and accountability. This screenshot shows a subset of commits (black circles) by four authors over a period spanning October 29th - November 11th, 2012. The full network graph is available online at https://github.com/PhDP/article_preprint/network

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 such products are stored on lab computers or web servers both of which are prone to failure and often inaccessible after a certain length of time. One survey found that only 72% of 1,000 surveyed studies still had data that were accessible (Schultheiss et al., 2011; Wren, 2004). 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 (Prlić and Procter, 2012).

While git provides a powerful feature set that can be 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 server. Git remotes serve as hubs for collaboration where authors with write privileges can contribute anytime while others can browse and download up-to-date versions or submit revisions with author approval. There are currently several git hosting services that provide free git hosting, such as SourceForge, Google Code, GitHub and Bitbucket. Among them, Github has surpassed other popular providers and hosts over 2 million public repositories at the time of this writing (Finley, 2011). 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 (with paid plans **although Bitbucket has unlimited free private repos and GitHub offers 5 free private repos to academics) and opened to the public at a future time.

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 provide protection 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. 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

With git's effortless branching mechanism, there is almost no cost to creating branches for exploring alternate ideas in a structured and documented way without disrupting the central flow of a project. Branches provide a risk-free way to test new algorithms, explore better data visualization techniques, develop new analytical models or changing the organization of a manuscript. When branches yield successful outcomes, they can easily be merged into the master copy while unsuccessful efforts can be left as-is to serve as a historic record. 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 not fruitful in earlier efforts.

5. Mechanism to solicit feedback and reviews

While it is possible to leverage most of the core functionality of git at the local level, git hosting services offer additional features such as issue trackers, collaboration graphs, online editing, web sites 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 because comments can easily be linked to particular lines of code or blocks of text.

6. Transparency, verifiability

Methods sections in papers are often brief and succinct to adhere to strict word and page 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 implemented in a study may not be 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 findings of a paper. Without access to a detailed history, a reviewer competent in statistical methods has to give the benefit of the doubt to the authors and assume that assumptions of methods used were clearly met. Sharing a git repository can remove these kinds of ambiguity and allow authors to point out commits where certain key decisions were made before using particular analytic approaches. Journals could facilitate this process by allowing authors to submit links to their git repository along with their submitted manuscript.

7. Managing large data

git is extremely efficient in managing small data files such as those 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. When large data files do not change often, the best practice would be to exclude those data 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. 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 (Neylon, 2013). 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 (Piwowar et al., 2007) but can also increase opportunities for collaboration. For example, Niedermeyer and Strohalm (2012) describe how they struggled 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.

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 questions that aim at addressing 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.

Conclusions

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 full version history and security against data loss and encourages 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 be viewed as entirely altruistic. In a recent mandate the National Science Foundation (US NSF - Dear Colleague Letter - Issuance of a new NSF Proposal & Award Policies and Procedures Guide (NSF13004), 2012) 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, everyone stand to benefit (Piwowar, 2013).

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. 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 spend considerable time using and creating their own software to address domain specific needs, good programming practices are rarely taught (Wilson et al., 2012). 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 scientists, 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).

Acknowledgements

I was supported by `NSF DEB-1021553` while preparing this manuscript. Comments from Carl Boettiger, Scott Chamberlain, `...`, and `...` on earlier drafts greatly improved the final version of this article. This manuscript is available both as a git repository with a full history of versions at `https://github.com/karthikram/smb_git.git` and also as a permanent archived copy on figshare (`http://figshare.com/.....`) (I'll add a link to figshare URL once a final version of the paper is accepted). I also thank the rOpenSci project (`http://ropensci.org`) for helping me gain a greater appreciation for git as a tool for advancing science.

Literature Cited

Finley,K. (2011) Github Has Surpassed Sourceforge and Google Code in Popularity.

Greenland,P. and Fontanarosa,P.B. (2012) Ending honorary authorship. *Science (New York, N.Y.)*, **337**, 1019.

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

Niedermeyer,T.H.J. and Strohalm,M. (2012) mMass as a software tool for the annotation of cyclic peptide tandem mass spectra. *PloS one*, **7**, 44913.

Peng,R.D. (2011) Reproducible Research in Computational Science. *Science*, **334**, 1226–1227.

Piwowar,H. (2013) Altmetrics: Value all research products. *Nature*, **493**, 159–159.

Piwowar,H.A. et al. (2007) Sharing Detailed Research Data Is Associated with Increased Citation Rate. *PLOS One*.

Prlić,A. and Procter,J.B. (2012) Ten Simple Rules for the Open Development of Scientific Software. *PLoS Computational Biology*, **8**, 1002802.

Schultheiss,S.J. et al. (2011) Persistence and availability of Web services in computational biology. *PloS one*, **6**, 24914.

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).

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

Vink,C.J. et al. (2012) Taxonomy and Irreproducible Biological Science. *BioScience*, **62**, 451–452.

Wald,C. (2010) Issues & Perspectives Scientists Embrace Openness.

Wilson,G. et al. (2012) Best Practices for Scientific Computing. 6.

Wren,J.D. (2004) 404 not found: the stability and persistence of URLs published in MEDLINE. *Bioinformatics (Oxford, England)*, **20**, 668–72.