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 (Vink et al., 2012). 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 novel science. 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 changes as required. The decentralized nature ensures that this rich history, including authorship, is available in every copy of a git repository for anyone to review, contribute, or build upon. In this paper I review reasons why scientists should leverage the power of git in their day to day research workflow to increase reproducibility and transparency, foster collaboration, and support novel synthesis.

Introduction

One of the original motivations behind publishing scientific articles was to ensure that results could be reproduced, validated and extended. Articles with detailed methods sections were necessary not just for evaluating the soundness of the central findings but also to ensure that readers could validate such findings and improve upon them. Over time, the scientific process has become quite complex, requiring increasing amounts of data collected using complex methods. The volume of research has also grown considerably over the years, resulting in a shortening of the length of articles (first because space was limiting in publications but later continued even though e-papers don't have any limitations). Now most papers contain brief methods. To reproduce the central findings of any paper, a reader not only requires access to detailed methods, but also the underlying data and code used to arrive at them. While this practice has become more common, there are several problem. Many articles still do not share their data and or code. There is also the problem that they are not shared in adequate detail (derived data versus raw data) or all authors do not share all the decisions they used to arrive at their final conclusions. So, some decisions, such as removing outliers, may not be adequately in the steps.

Another problem is the rise in retractions in recent years. (Van Noorden, 2011). Talk about why retractions are costly, hurt science, and are a bad thing overall.

Open science is needed to accelerate research. In the era of declining funding, we need to leverage existing data and code in new ways. Not only that we must also ensure that our data come with appropriate licenses that not only support fair use but also credit original authors (Neylon, 2013). Sharing git repositories with a full history of changes can be one way to lower barriers and accelerate open science. In this paper I describe why attributes of the distributed version control system **git** make it ideal for managing scientific workflows.

Version control systems

Version control systems (VCS) have long been used in software development for managing code bases. A key feature common to all such systems is that ability 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. 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.

Distributed version control systems like **git** offer several 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 rom 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, including any of the branches used to explore alternate methods or ideas. git also has a small footprint and nearly all operations occur locally. Rather than maintaining multiple versions of the same file, git uses changesets (i.e. changes that occur between revisions), which makes it possible for every file to have its own history. When changes are synchronized to remote copies, files are compressed before being transmitted further reducing the need for expensive hardware or high-speed internet connections.

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 and their authors. 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, check for errors among various other applications. In the remainder of the article I describe how key attributes of git can be applied to improve the scientific process. Readers should note that the article is not meant to be a complete or thorough review of all of git's features.

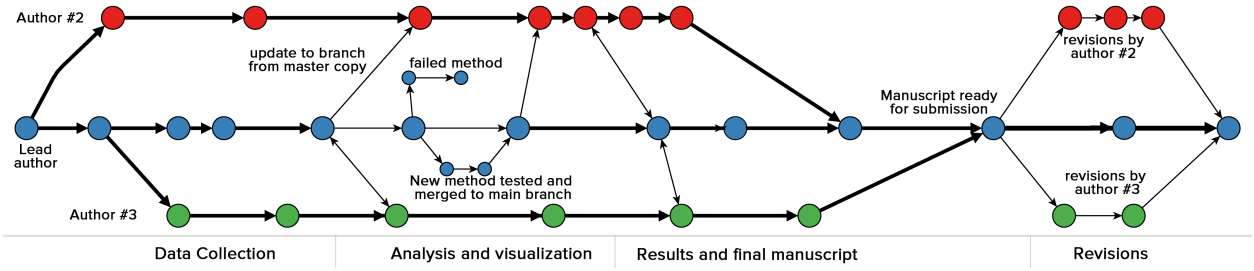

Figure 1: A hypothetical git workflow for a scientific collaboration involving three authors. Each circle represents a commit and colors denote author specific commits. Two way arrows indicate a sync (a push and pull in git terminology). One way arrows indicate an update to one branch from another. Horizontal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

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 exceptions (suggestions for citations?). git commit logs can serve as a proxies for lab notebooks if clear yet concise messages are included 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 be a highly granular way to track and assess individual author contributions over the course of 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 clone copies and work asynchronously. As each author adds their contribution, they can choose to sync those to the master branch and also 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 the most appropriate 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 merges are automatic, 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. In addition to facilitating easy collaboration, this process also makes it easy to assess relative author contributions when determining the order of authorship. Error or clarifications can also be directed to the appropriate author.

In a recent paper led by Philippe Desjardins-Proulx https://github.com/PhDP/article_preprint/network all of the authors (including me) successfully collaborated using only git and GitHub. 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.

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 on web servers and often inaccessible after a certain period of time. One survey found that only from 72% of studies of 1000 surveyed 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 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. There are currently several git hosting services such as SourceForge, Google Code, GitHub, and BitBucket that provide free

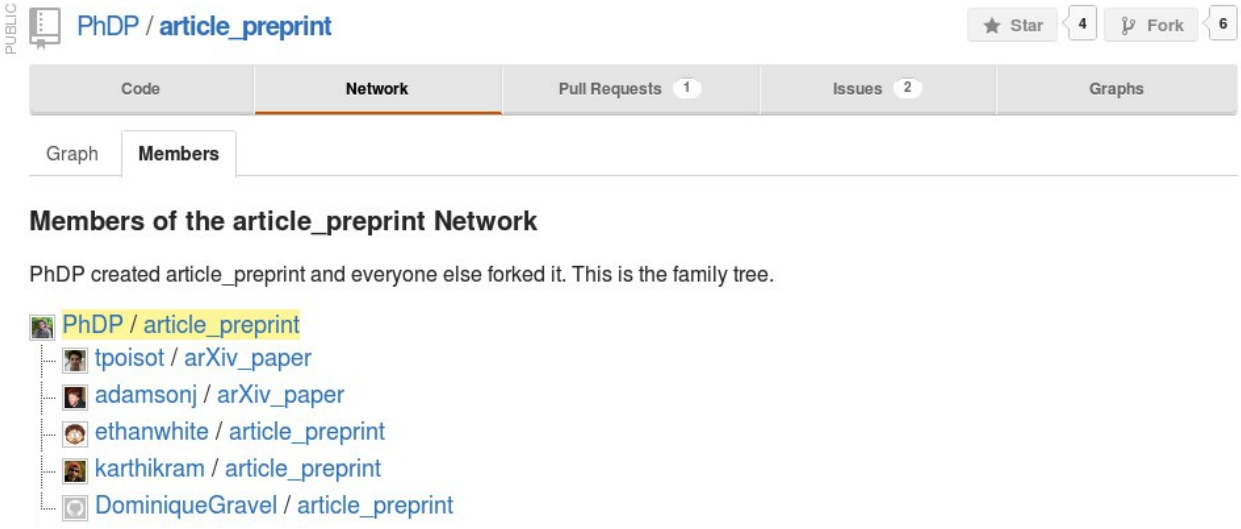

Figure 2: A list of contributions to a project on GitHub

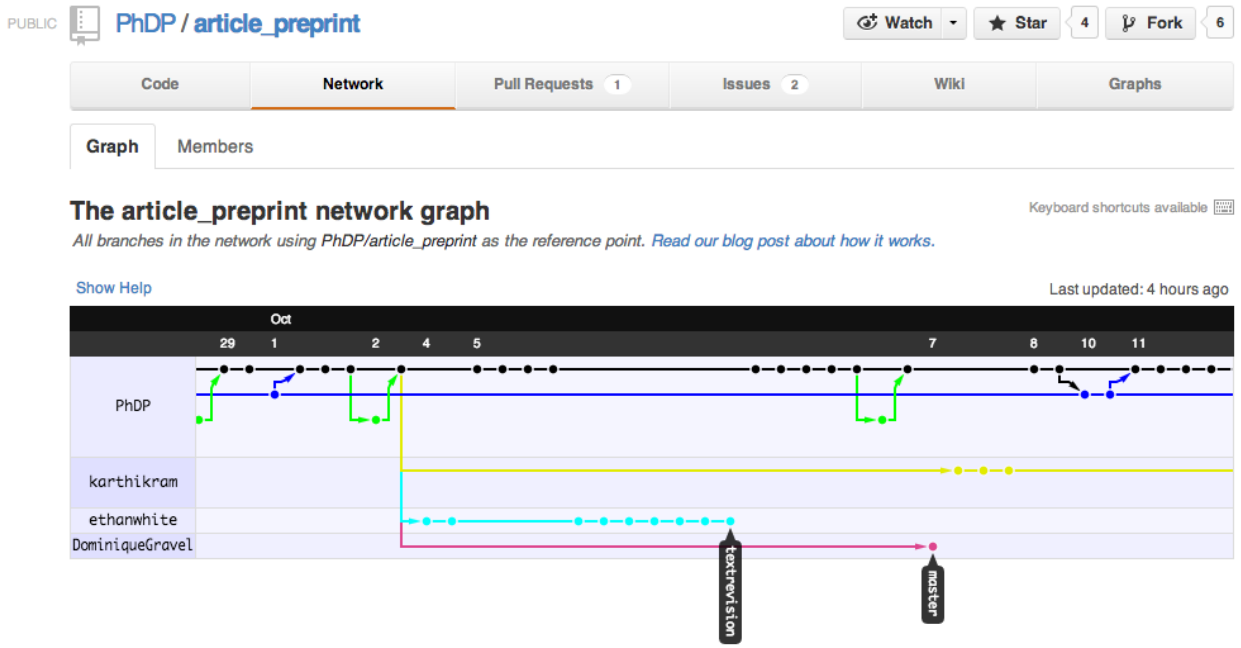

Figure 3: git makes it easy to track individual contributions through time ensuring appropriate attribution and accountability. This screenshot shows subset of commits (black circles) by four authors over a period spanning October 29th - November 11th, 2012.

git hosting. Among them, Github has surpassed other popular provides like Google Code and SourceForge 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 maintained privately. There are multiple ways to deal with such situations. For example, certain files can be excluded from git's history, others can be maintained as private sub-modules, or entire repositories can be kept private (with paid plans) and made public at a future time. A detailed discussion of the entire range of possibilities are beyond the scope of this article.

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 are archived over a network of redundant nodes and ensure indefinite available 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, or develop new analytical models. 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 comments about the rationale for choosing one method over another since the git history contains a record of failed, inappropriate or abandoned attempts. This is particularly useful since the time between submission and response can be fairly long. Additionally, future users can mine git histories to avoid repeating approaches that were never pursued for valid reasons.

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 also solicit review. Issue trackers can provide a mechanism for both feedback and review, especially since they can easily be linked to particular lines or blocks of code or text.

6. Transparency, verifiability

Methods sections in papers are often brief and succinct to adhere to strict word limits imposed by journal guidelines. This practice is particularly 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 are not available to the reviewers. 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 conclusion 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 use where 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.

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 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 the metadata. This protocol is especially ideal when such 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/.

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 do so. 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). Analysis pipelines can be easily *forked* or branched from public git repositories and modified for new applications. Although this process of depositing code somewhere public with appropriate licenses creates 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 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 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* datasets. In cases where no datasets are deposited, the only data accessible are mean values reported in the text. 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 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 research efforts are securely logged while maintaining author contributions in a system that provides security against data loss and encourages exploration of new 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 (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 efforts to share, then everyone can benefit (Piwowar, 2013).

Although I have laid out numerous arguments for wider use of a distributed version control system like git, it should not be viewed as a solution to all the problems facing science today. 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 those efforts 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, knowledge of good programming practices, including the use of version control, are rarely taught (Wilson et al., 2012). Therefore wider adoption of useful tools like git will require greater software development literacy among scientists. On an optimistic note, such literacy is becoming increasingly common with 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 (Does Ethan W. have a class I can cite?).

Acknowledgements

I was supported by NSF DEB-1021553 while preparing this manuscript. Comments from 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 changes) 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.

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.

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.