

GitHub: Putting the social in social science data set development and verification

Christopher Gandrud
Yonsei University*

January 11, 2013

I can't send you my data b/c I think you might find out I made an error. #overly-honestmethods (tweeted by Charlisle Rainey, 1/10/2013)

A data set is created, analyses are run, an article is published, the data set languishes. The researchers who create the data set move on to other projects. Maybe someone else is able to access the data and they update part of it for their own work, but the original is not changed and the updates not widely known about. Maybe another researcher finds a mistake in the original data set. They email the original authors suggesting corrections. The original authors may or may not make the changes. If changes are made there are no easily accessible records of this. Some researchers may even be reluctant to make their data available at all because of a fear that someone may find a mistake in it. These are all examples of ways that the development and management of social science data sets do not take advantage of knowledge distributed in the social science community. These problems are partially caused by the data storage tools social scientist use. Despite rapid recent advances in social technologies—Twitter, Facebook, Skype, and so on—the tools many social scientists use do not make it easy to collaboratively develop and verify data sets, especially for people not involved in creating the original data set.

In this brief article I show that GitHub¹ offers a comprehensive data storage service for social scientists with unique tools for *social data set development and verification* (correcting errors). GitHub also fits

directly into a highly reproducible workflow, particularly one that also includes R.²

GitHub was originally created and is widely used as a tool for software developers to work together on software projects using the Git³ version control system. Though GitHub is often used by social scientists for statistical package development⁴ it initially seems strange to suggest that this service would be useful for social scientists for building and maintaining data sets. However, a software program and many social science data sets are fundamentally similar. They are collections of text files. GitHub is a means of storing and version controlling text files. So if a social scientist's data is in a plain-text format, such as comma-separated values (CSV),⁵ and has accompanying variable description files also in a plain text format—e.g. TXT plain text or Markdown (MD),⁶ they can take full advantage of GitHub's features for social data set collaboration and verification.⁷

In this article I first discuss basic features that we would want from a cloud service to store social science data: stable storage, access, version control, and collaboration. Then I examine how strong these features are in three methods currently used to store data in social science. Though each methods has strengths none of them encourages social data set development and verification.

²GitHub can be used to store and develop entire social science research projects, not just data. However, I do not directly address these other topics here.

³<http://git-scm.com/>

⁴For example the *Zelig* R package (Imai, King and Lau, 2008; Owen et al., 2012) is hosted on GitHub. See <https://github.com/IQSS/Zelig>.

⁵All major statistical programs as well as Microsoft Excel and Apple's Numbers can save and open files in plain-text formats like CSV.

⁶<http://daringfireball.net/projects/markdown/>

⁷See Bowers (2011, 3) for a discussion of the advantages of storing research files in plain-text format.

*Lecture in International Relations. Email: gandrud@yonsei.ac.kr.

[†]The files used to create this article are available on GitHub at: <https://github.com/christophergandrud/PolisciGitData>.

¹<https://github.com/>

A few brief notes before getting started: To set up Git and GitHub see the GitHub page: <https://help.github.com/articles/set-up-git>. In this article I give examples of how to take advantage of GitHub using their website and from the command line. However, there are also very good graphical user interface (GUI) versions of GitHub for Mac and Windows (see the set up page). I focus on GitHub, but you can also use other services that work with Git such as Bitbucket⁸ for many of the same functions. Finally, the tools I discuss in this article are suitable for small to medium size data sets (i.e. of about 100,000 observations or less) common in social science. Much larger data sets often cannot be efficiently stored in plain-text files, so GitHub is probably not a suitable storage service for them.

1 What do we need from cloud data storage?

A cloud storage service for social science data needs to enable at least four things: *stable storage*, *access*, *version control*, and *collaboration*. Obviously a cloud storage service needs to be a stable and reliable platform for keeping data sets on. Data stored on it needs to be accessible to both coauthors and people who wish to reproduce an analysis (Fomel and Claerbout, 2009) or use the data set in a new project (Kelly, 2006; King, 1995). Another key part of access is that the data set, both development and use, can be easily tied into researchers' workflows. The service needs to include version control—similar to track changes in a word processing program—so that the development of the data set can be understood and researchers are able to revert to old versions. See Bowers (2011), Healy (2011) and Fredrickson, Testa and Weidmann (2011) for discussions of version control. Finally, a cloud data storage service should make collaboration easy. Collaboration should not be limited to coauthors. It should be possible to collaborate with non-coauthors to take advantage of knowledge (and motivation) distributed throughout the social science community. Ideally, social data set development and verification will keep data sets more up-to-date and more accurate.

⁸<https://bitbucket.org/>

2 Data storage: the social science status quo

Social science data is often stored in at least three ways. It may be stored locally on a researcher's computer, on a general purpose cloud storage service such as Dropbox⁹ or Google Cloud Drive,¹⁰ or data may be stored on a specialized research hosting service, notably Dataverse.¹¹ Each of these data storage methods have their strengths in terms of stable storage, access, version control, and collaboration. Nonetheless each are lacking in at least one important way and none of them promote social data set development and verification.

Local storage The least robust form of data storage currently used, is only storing data on an individual researcher's computer. This is not a very stable form of storage as anyone knows if they have lost all of their files when their computer died. The research with access to the computer can easily access the data and use it in their workflow, but access from other computers and for co-authors can be limited. Access by non-coauthors is very cumbersome. Files must be emailed in response to individual requests. These files are not automatically updated with new versions of the data. Version control with Git or some other program is possible with locally stored data. Because access is limited to users of the computer the data is stored on, collaboration, especially by non-coauthors is extremely limited.

General purpose cloud storage Dropbox and other general purpose cloud storage services offer a much more robust form of storage. These services generally work by syncing files stored on individual computers with those on cloud servers. Access is also much better. These services usually can be accessed via desktop programs, mobile apps, and websites. Because files stored on individual computers are automatically synced with cloud serves it is very easy to incorporate them into a workflow. General purpose cloud storage services also make it possible to share files and folders via URL links. Dropbox has a basic version control system. Every time you save a file a new version is saved on Dropbox. If you are using Dropbox for free old, versions are only saved for 30

⁹<https://www.dropbox.com/>

¹⁰<https://drive.google.com/>

¹¹<http://thedata.org/>

days.¹² In addition you could use Git with a data set stored on Dropbox. Collaboration with coauthors is very easy because folders can be shared. This means that official collaborators (those given write access to the folder) can easily make changes to files in the folder and these changes are automatically synced for all users.¹³ However non-coauthors, without write access to the shared folder cannot easily make changes. They must email one of the researchers with write access to suggest updates. This is no better than the situation when files are stored locally.

Dataverse Many journals—Political Analysis for example—require data used in articles they publish to be uploaded to a service like Dataverse. Dataverse is a stable form of storage that is easily accessible for anyone with an internet connection. It is difficult to incorporate Dataverse into a research workflow using the standard version.¹⁴ Unlike Dropbox, for example, there is no way to automatically update a data set on Dataverse. You have to point and click to upload data files for each version. It is difficult to access data through a statistical program such as Stata or R. The data needs to be downloaded by pointing and clicking and then loaded into these programs. It does have some version control features. It saves each version that is uploaded. Finally it is difficult to collaborate using Dataverse. Changes to a data set must be uploaded to the site manually, then manually downloaded. There are also no direct ways for non-coauthors to suggest changes.

Dataverse is good for what it is designed to do: store a snapshot of a data set for replicating specific published results. However, it is difficult to use it to store and access data as part of an ongoing workflow. In addition it does not easily enable social data set development and verification.

3 Data Storage on GitHub

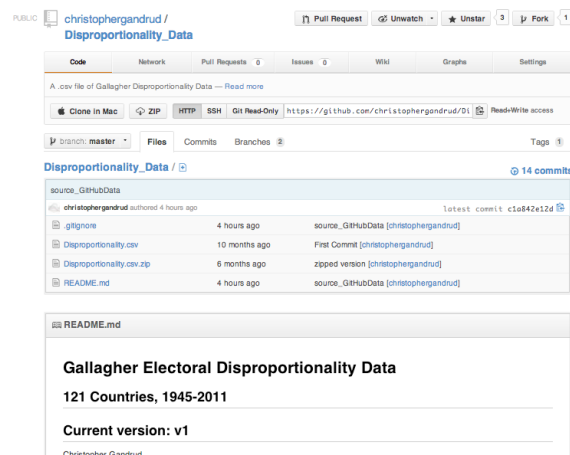
Though admittedly requiring a steeper learning curve, GitHub meets the four criteria for data set storage. In this section I will demonstrate how to use GitHub for data set storage, version control, access, and collaboration. Rather than being a complete in-

troduction, I focus on specific particularly important aspects of each feature in Git and GitHub. In this section I refer to a data set stored on GitHub of countries' Gallagher Electoral Disproportionality index that I compiled (see Gallagher, 1991; Carey and Hix, 2011). The data set and more information is available at: http://christophergandrud.github.com/Disproportionality_Data/.

3.1 Storing and Version control

Version control is an integral part of how GitHub stores files. GitHub remotely stores files in what Git terms “repositories”, repos for short. You can think of repos as the main folder a data set is in. They could contain the data and description files. Git version controls these files and GitHub hosts them remotely in the cloud.¹⁵

It is free to store files on GitHub as long as they are in what GitHub calls “public repositories”. Anyone can see the contents of public repositories, including all previous versions.¹⁶ Users can have an unlimited number of public repositories. There is a soft storage size limit of one gigabyte per repository. This should be more than enough for most social science data sets if they are stored in plain-text formats. Here is a portion of the electoral disproportionality repository's main page:



Git is a very powerful version control system, especially for text files. For these types of files, Git only saves the actual changes when you **add** a file to the repository and then **commit** a version of the file to

¹²Old versions are stored for longer with paid accounts.

¹³This can create problems if multiple authors are making changes to the same files at the same time. For a discussion of file conflicts see Fredrickson, Testa and Weidmann (2011).

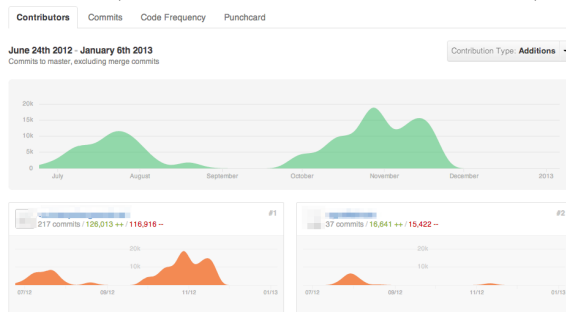
¹⁴It is possible to build a custom version of Dataverse with R integration.

¹⁵Specifically, GitHub files are stored on Rackspace (<http://www.rackspace.com/>).

¹⁶Private repositories are available allowing only official contributors to see their contents. They require a paid account.

the repository. This is different from the other cloud storage services we have discussed. They save the whole file for each version. Each commit in a repo is given a unique SHA-1 number identifying it. Once changes made on your computer are committed you **push** them to the remote GitHub version of the repo. A repo's GitHub website allows you to view all of the changes that have been made to it.

Contributor analytics An important component of version control on GitHub for us here is that because Git only saves changes and uniquely identifies the changer, it is possible to properly attribute every individuals' contribution to a data set. GitHub has analytic capabilities for organizing and displaying this data. You can find these by clicking on the **Graphs** button located near the top of each repository. This will give you access to graphs such as the ones below (I've blurred the contributors names):



Binary files I mentioned that Git treats non-plain-text files—binary files—like those Stata uses to store data differently. Rather than only the specific changes being saved with each commit, the entire file is stored again. If the files are very very large this could take up a lot of storage space.¹⁷ This is also a problem for all of the other data storage methods with version control discussed here. For very large data sets you will need to use a totally different type of cloud storage system like Rackspace or Amazon's S3.¹⁸

If you absolutely must have binary files in a repository one solution to the space constraints problem is to have Git ignore it. You do this by including a text file called `.gitignore` in your repository. In `.gitignore` simply type the binary file's name. Git will not version control it. This unfortunately also means that

the file will not be loaded on to GitHub when you push your files.

Describe the data set with README files

Each folder in a GitHub repository can contain a file called README.¹⁹ README files for data sets can contain information about sources, variable descriptions, and so on. GitHub automatically displays the README file on the repository's website in full. If it is written in the Git Flavored Markdown²⁰ mark-up language and called README.md it will also be automatically formatted. You can see part of the disproportionality data set's README.md above.

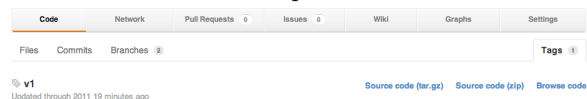
Tagging versions Git **tags** function as bookmarks for major versions of a Git repository. They are particularly useful for demarcating the version of a data set used in a particular publication, for example. creating tags is simple. Imagine we want to tag the second major version of a data set, the one we used for a publication:

```
git tag -a v2 -m 'Citation Information'
```

The option `-a` means add, `v2` is the version number, and `-m` adds a message, in this case the publication's citation information. Then simply **push** the tags to GitHub:

```
git push --tags
```

Now on GitHub there will be a list including the tag and an option to view and download this specific version of the data. For example:



3.2 Accessing data

There are many ways to access data stored on GitHub and incorporate it into your workflow. The simplest way is to use the GitHub website to actually edit files and commit changes. This can be handy for small changes and accessing the data from mobile devices. As I mentioned, changes can be committed on your computer and pushed to GitHub. You can use the command line version of Git or the GUI version of

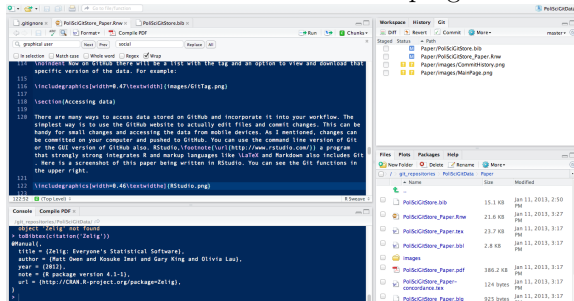
¹⁷Though it is important to remember that previous commits are compressed.

¹⁸<http://aws.amazon.com/s3/>

¹⁹Actually each folder in a repository can contain one as well.

²⁰<http://github.github.com/github-flavored-markdown/>

GitHub also. RStudio,²¹ a program that strongly integrates R and mark-up languages like \LaTeX and Markdown, also includes Git. Here is a screenshot of this paper being written in RStudio. You can see the Git functions in the upper right. So it is possible to make changes to a repository, commit them and push them to GitHub all within the same program.



Cloning a repo Repositories can be downloaded in full. This is called cloning. You can do this by clicking the **Clone in . . .** button on the GitHub repositories' website. It can also be done in the command line using the repository's address. For example, the disproportionality data's clone-able address is: https://github.com/christophergandrud/Disproportionality_Data.git.

```
git clone https://github.com/christophergandrud/Disproportionality_Data.git
```

Note that in real life the address needs to be on the same line. Also, before cloning a repository remember to change the working directory to the place where you want to have it saved. In the command line use the `cd` command to do this.

Once you have cloned the repository you can role back to any previous version with the `checkout` command. For example if you wanted to role back to a tag called "v2":

```
git checkout v2
```

Access data directly from R Loading data stored on GitHub into R for use in statistical analysis is very easy. I have created a function that loads plain-text formatted data from GitHub into an R data frame.²² It's called `source_GitHubData` and is stored in a GitHub Gist²³ at: <https://gist.github.com/4466237>.

²¹<http://www.rstudio.com/>

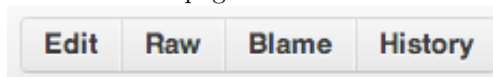
²²The function is based on *devtools*' `source_url` command.

²³Gists host code snippet. See: <https://gist.github.com/>.

gist.github.com/4466237. It can be loaded into R using the `source_gist` command from the *devtools* package (Wickham and Chang, 2012).

```
# Load source_GitHubData
# The functions' gist ID is 4466237
devtools::source_gist("4466237")
```

The main argument in `source_GitHubData` is the `url`. You just set this as the URL for the *raw* version of data file you want to download. The raw version is the version with only the text-file (no extra HTML mark-up). You get this by clicking the **Raw** button on the file's GitHub page:



The raw URLs for each commit and tagged version of the data are also accessible if you want to always link to a particular version. The URL for the raw version of the electoral disproportionality data is <http://bit.ly/Ss6zD0>.²⁴ To download the data using `source_GitHubData` simply type.

```
# Download data
Data <- source_GitHubData(
  url = "http://bit.ly/Ss6zD0")
```

Note that by default `source_GitHubData` loads CSV data. You can add the argument `sep = "\t"` for tab-separated data files. You can also specify `header = TRUE` (default) or `header = FALSE`.

Citing GitHub data When a researcher uses data they accessed via GitHub how can they cite it? A common practice is to cite the publication the data set was originally used in. However, this is incomplete in at least two ways. First, the version of the data used in the original publication may be different from that used later on. Second, it would be difficult to use this citation to acknowledge contributions made by contributors who did not work on the original data set, but contributed to later versions. One solution is to use the standard set by Altman and King (2007).²⁵ They propose that data set citations should included:

- the authors name,

²⁴I used bitly (<http://bitly.com/>) to shorten the URL so that it would fit on the page more easily. The full URL is: https://raw.githubusercontent.com/christophergandrud/Disproportionality_Data/master/Disproportionality.csv.

²⁵Dataverse uses a version of this standard.

- the date,
- the data set's title,
- a unique global identifier (UGI),
- a universal numeric fingerprint (UNF),
- a bridge service.

The first three are self explanatory and shared with standard citations for other types of materials. Examples of UGI include Document Object Identifiers (DOI) and the Handel System.²⁶ They uniquely identify the data set. UNF's uniquely number a particular version of the data set. Finally a bridge service allows you to use the DOI and UNF to link to the actual data set.

We can easily use Altman and King's citation standards for GitHub repositories. I would suggest citing specific tags of the data set. If these are not available you can cite the specific commit you used to produce results with.

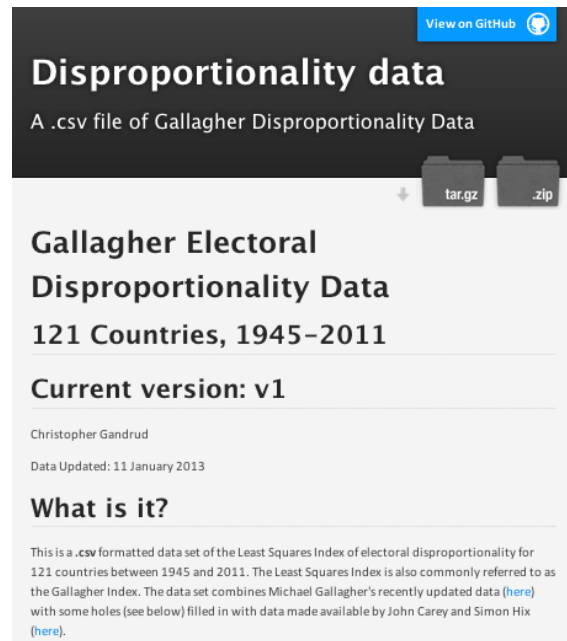
Showcase a data set with GitHub Pages Each public repository's GitHub website allows anyone full access to the data. However, these pages may be confusing for those without experience using a version control system. To solve this problem GitHub Pages²⁷ allows you to very easily create a simple webpage for the repository. These pages allow you to describe the data and include links to download the entire repository. You can of course also add links to specific files.

To create your repository's page navigate to the repository's normal webpage. Then click **Settings** → **Automatic Page Generator**. By default it will load the README file as the content of the new page. You can change this, add a Google Analytics tracking ID²⁸ to gather information on who visits the page, and choose an aesthetic style for the page. Here is a sample of the disproportionality data's page:

²⁶<http://www.handle.net/>

²⁷<http://pages.github.com/>

²⁸<http://www.google.com/analytics/>



3.3 Collaboration

Compared to the status quo social science data storage methods, GitHub is uniquely strong for enabling collaboration both between co-authors and even non-co-authors who may be able to help develop and verify the data set.

Co-authors Public repositories can have an unlimited number of what it calls “collaborators”.²⁹ Collaborators have the same privileges to make changes to the repository. They all do so in the same way. There is one important thing to note. If multiple collaborators are actively working on a data set they need to add an extra step to their Git workflow.³⁰ Each person needs to **pull** their collaborator's changes and resolve any merge conflicts before pushing the changes to GitHub. Here is a full example:

```
# Add new files to Git
git add .

# Commit the changes
git commit -am "A message"
```

²⁹On the repositories' GitHub site click **Settings** → **Collaborators** and add the co-authors GitHub username.

³⁰This is also true if you make changes to both the GitHub version of the repository and the copy on your computer.

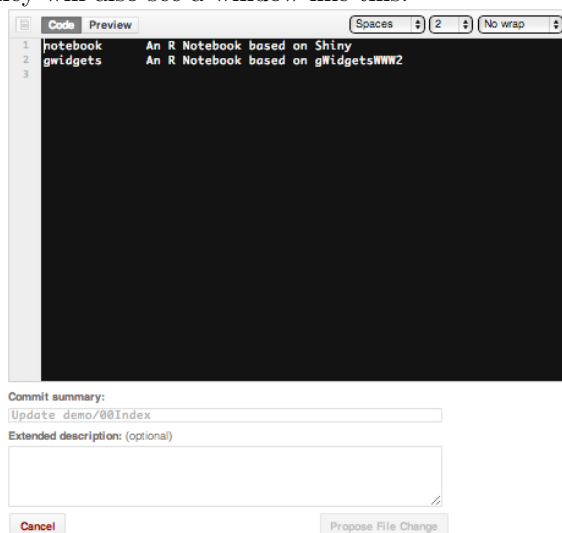
```
# Pull collaborator's commits
git pull
```

```
# Push changes to GitHub
git push origin master
```

Zachary, are you thinking of discussing some of these commands in more detail?

Collaborating with non-Co-authors: Pull requests People who want to make changes to a repository, but are not collaborators can make pull requests. All they need are GitHub accounts. Pull requests are basically specific changes that non-collaborators suggest. Requesters can also add comments about why they are making the request. GitHub also has a discussion forum to discuss these comments. It is then up to the repository's owners to decide whether or not to accept the requested changes. If they accept the changes, they are made instantly and a full record of who made the changes is kept.

If someone notices a small error—e.g. a misspelling, a misclassification—or other improvement that they think should be made to a data set they can make a pull request by navigating to the GitHub page for the specific file they think needs to be changed. Then they click the **Edit** button located next to the **Raw** button (see above). Clicking this button “forks” the repository, i.e. gives them a copy they can change. They will also see a window like this:



In this window they can make their proposed changes and add a comment about what the changes are before clicking **Propose File Change**.

For longer file changes, e.g. a major addition to the data set to bring it up-to-date, it is better to work with the forked repository rather than the Edit window. To directly fork the data set's repository click the **Fork** button on the upper right corner of its webpage. You can then make changes to the forked repository as if it was your own. When you are ready to suggest the changes be included, you click the **Pull Request** button at the top center of the forked repository's webpage. For more information on forking and pull requests see the GitHub article on the topic: <https://help.github.com/articles/using-pull-requests>.

Issues GitHub repositories also have an “Issues” area that allows any other GitHub user to make a comment on the repo. These tend to include general suggestions for how to improve the files or questions about the repo that may be of general interest. Anyone can respond in the Issues area.

Repo Wiki GitHub provides a very easy tool for creating nicely formatted repository wiki's. For example, a data set repo's wiki could include short articles with details about how the data set was created. Non-official collaborators can add to repo wikis. Like pull requests, their changes are subject to approval by one of the repository's owners.

Follow a repository If you are a GitHub member you can follow a public repository, even if you aren't a collaborator. This gives you a Facebook-style news-feed³¹ where you can see any updates made to the data set repository as well as discussions in the Issues area and pull requests.

Passing on the data set Because of changing time commitments, professional interests, and so on, no one can maintain and update a data set forever. GitHub makes it easy to pass on control of a data set while maintaining its entire version history. Simply add the new data set maintainer as a collaborator. They will then have the same privileges to manage the repository as you did. To completely transfer control simply have the new maintainer fork the repository and leave a link to the forked version at the previous location.

³¹ See: <https://www.facebook.com/help/327131014036297/>.

Selective incentives Why would people actively contribute to improving publicly available data sets, especially those primarily associated with other authors? If a repository had many non-official contributors it would be impractical to cite all of them whenever someone used the data set, for example.

GitHub not only provides technology—particularly pull requests—for social data development and verification, but it also gives selective incentives that can motivate people to do so. By keeping track of who made what changes and providing numerous ways to quantify and visualize each member’s contribution, GitHub provides strong selective incentives to collaborate. Perhaps one day social scientists could even use GitHub’s descriptive statistics when they go in front of hiring and promotion committees.

4 Conclusion

In this article I have tried to show that GitHub is a very good option for storing social science data in the cloud. Admittedly it has more of a learning curve than the incumbent options. However, I hope that I demonstrated that it is at least as good as the alternatives in terms of stable storage, access, version control, and collaboration. It is by far better at enabling social data development and verification. Hopefully by using a service like GitHub that both makes collaboration easy and provides selective incentives to do so we can improve the social science community’s ability to utilize its collective knowledge to have more complete and robust data sets. Better data sets will allow us to better answer our questions.

References

- Altman, Micah and Gary King. 2007. “A Proposed Standard for the Scholarly Citation of Quantitative Data.” *D-Lib Magazine* 13(3/4).
- Bowers, Jake. 2011. “Six Steps to a Better Relationship with Your Future Self.” *The Political Methodologist* 18(2):2–8.
- Carey, John M. and Simon Hix. 2011. “The Electoral Sweet Spot: Low Magnitude Proportional Electoral Systems.” *American Journal of Political Science* 55:383–397.
- Fomel, Sergey and Jon F Claerbout. 2009. “Reproducible Research.” *Computing in Science & Engineering* 11(1):5–7.
- Fredrickson, Mark M., Paul F. Testa and Nils B. Weidmann. 2011. “Collaboration for Social Scientists, or Software is the Easy Part.” *The Political Methodologist* 18(2):19–23.
- Gallagher, Michael. 1991. “Proportionality, Disproportionality, and Electoral Systems.” *Electoral Studies* 10(1):33–41.
- Healy, Kieran. 2011. “Choosing your workflow applications.” *The Political Methodologist* 18(2):9–18.
- Imai, Kosuke, Gary King and Olivia Lau. 2008. “Toward A Common Framework for Statistical Analysis and Development.” *Journal of Computational and Graphical Statistics* 17(4):892–913.
- Kelly, Clint D. 2006. “Replicating Empirical Research in Behavioral Ecology: How and Why it Should be Done But Rarely Ever Is.” *The Quarterly Review of Biology* 81(3):221–236.
- King, Gary. 1995. “Replication, Replication.” *PS: Political Science and Politics* 28(3):444–452.
- Owen, Matt, Kosuke Imai, Gary King and Olivia Lau. 2012. *Zelig: Everyone’s Statistical Software*. R package version 4.1-1.
URL: <http://CRAN.R-project.org/package=Zelig>
- Wickham, Hadley and Winston Chang. 2012. *devtools: Tools to make developing R code easier*. R package version 0.8.
URL: <http://CRAN.R-project.org/package=devtools>