Collaboration

Jake Bowers[*]

February 24, 2011

> Do I contradict myself?
> Very well then I contradict myself,
> (I am large, I contain multitudes.)
> (Whitman, 2008)

An idea is born in a coffee shop, a seminar, a quiet walk in 2011. This idea inspires a seminar paper in 2011. And a conference paper arises from the seminar paper in 2012. A dissertation chapter descends from the conference paper in 2013. Other dissertation chapters take up 2014. A submission to a journal happens in 2015. Revision and resubmission wait until 2017 while harried editors, reviewers and author strive to balance teaching, service, and life. In 2018 a child is born and a paper is published. In 2020 a graduate student in a coffee shop reads the paper and has an idea that challenges the analysis in the paper. What would happen if only the author had thought to control for X? Or run one more MCMC chain? Or chosen a different likelihood function? Will the United Nations (now eager to act based the paper) make a wrong move? Time to go back to the analyses run sometime between 2011 and 2020 to check. Easy. Right? Collaboration between the past self and the present self should be simple — after all you are the same person in 2011 and 2020, right? And the way Microsoft Word/Stata/SPSS/R/LISREL stores files and the way that your Windows/Mac machine reads and writes them is the same, right? And you know exactly which bit of code produced which table and which figure, right? And you wrote your code following Nagler's Maxims Nagler (1995) and being following King's Replication Standard King (1995) right?

[*] Many thanks to Kevin Quinn for direct help on this document, Mika LaVaque-Manty, Ben Hansen, and Mark Fredrickson for many useful discussions on this topic, and Kieran Healy for inspiration from afar.

Just in case one has some doubt about answering these questions, this piece aims to amplify some of what we already ought to know Nagler (1995), and to update some of those ideas given current practices, platforms, and propensities.

When we call some piece of work "scientific" we are talking about how it was done. By calling it "scientific" we are also using a short-hand to suggest that (1) the findings of the work are not a matter of opinion and (2) other people could reproduce the findings. That is, what the work represents is not a matter of opinion but is a shared experience — and an experience shared without respect to the identities of others (although requiring some shared technical training and research resources). This short-hand means that "scientific" work should make us change how we act — or at least, it ought to stand on stronger epistemological ground than other claims about experiences which ought to be shared or shareable.

Assume we want other scientists to believe us when we say something. More narrowly, assume we want other people to believe us when we say something about data: "data" here can be words, numbers, musical notes, images, ideas, etc . . . The point is that we are making some claims about patterns that we see in some collection of stuff. Now, it might be easy to convince other people that "this collection of stuff is different from this collection of stuff" if those people were looking over our shoulders the whole time that we made decisions about collecting the stuff and analyzed it [where "analyze" means something like "break it up into understandable parts"]. Unfortunately, we can't assume that people are willing to shadow a researcher throughout her career. Rather, we do our work alone or in small groups and want to convince other distant people about our analyses.

Now, say your collections of stuff are large or complex and where your chosen tools of analyses are computer programs. How can we convince people that what we did with some data with some program is credible: not a matter of whim or opinion, and reproducible by others who didn't shadow us as we wrote our papers? Here are a few concrete actions that people can take to enhance the believeability of their work. In addition, these actions make collaboration within research groups more effective (since believeability comes in part from reproducibility and research groups often need to be able to reproduce in part or in whole what different people in the group have done).

Here is some advice that is not currently emphasized but which might be profitably added to previous advice.

1 Show me the data.

All files containing commands operating on data must refer to a data file. A reference to a data file is a line of code the analysis program will use to operate on ("load"/ "open" / "get" / "use") the data file. One should not have to edit this line on different computers or platforms in order to execute this command. Using R, for example, all analysis files should have load('thedata.rda') or `read.csv('http://www.mywebsite.org/Data/thedata.csv')` or some equivalent line in them, and 'thedata.csv' should be stored in some place easy to find (like in the same directory as the file or perhaps in 'Data/thedata.rda').

Where should you store the data files? A very obvious solution is to always make sure that the data file used by a command file is in the same directory as the command file. More elegant solutions (pointed to below) require all co-authors to have the same directory structure so that load('Data/thedata.rda') means the same thing on all computers used to work on said project.

The principle of modularity Nagler (1995) suggests that you separate data cleaning, processing, recoding, and merging from analysis in different files. So, perhaps your analysis oriented files will load('cleandata.rda') or something but with a comment in the code telling the future you (among others) that cleandata.rda was created from create-cleandata.R which in turn begins with read.csv('dirtydata.csv') which, from a comment in the file, was downloaded from `http://www.greatfreedata.gov/thedata?dirtydata.csv` on April 1, 2011. Such a data processing file will typically end with something like `save('cleandata.rda')` so that we are doubly certain about the provenance of the data.

Now, if in 5 years we wonder where 'cleandata.rda' came from, we might `grep cleandata *.R` to search for occurrences of this file on our Unix/OS X system. However, if such searching among files is a burden, and even nicer solution is to maintain a file for each project called "MANIFEST.txt" or "INDEX.txt" or "README.txt" which lists the data and command files with brief descriptions of their functions and relations.

The principle regarding data is to know where the data came from and what operations were performed on which set of data. I have seen command files which do not begin with a call to load data and I have despaired. What is the meaning of `please-fit var1 with var2` if I do not know where var1 var2 come from.[1]

2 Code not Clicking

All tables and graphs should be as much as possible generated from code and not clicking (let alone copying and pasting). If I wanted to re-create the figure you created but with red lines, and only for people in the South, I should be able to do so with just a few edits to the code.

Using R, for example: I would write

```
pdf('fig1.pdf')
plot(stuff)
dev.off()
```

Now, in the future if I wonder how "that plot on page 10" was created, if I suitably commented my manuscript or inserted text into my MANIFEST.txt file I will know (1) "that plot" is from a file called "fig1.pdf" and (2) fig1.pdf was created in `figs1and2.R`. In the future, if I wanted to quickly change fig1.pdf, I could edit the commands quickly and easily to do so. [Another good practice is to name the files descriptively, so that we don't in fact have "fig1.pdf" but "political participation by education in the south.pdf" although such a name may prove burdensome to operating systems which do not care for spaces in file names or to folks who are used to typing full filenames rather than using command line completion facilities in modern shells.]

An even nicer way to ensure that figures and tables are reproducible or easily modifiable by your older and wiser but more harried self is to use literate programming. For example, one may insert the following chunk of code into either a LaTeX or an OpenOffice document, ask R to process the document, and then produce a pdf file in which the code has disappeared to be replaced with an actual figure:

```
 <<fig1,fig=TRUE>>==
 plot(stuff)
 @
```

[1] The command `please-fit` comes from the MayIPleaseDoStatistics package for R which emphasizes politeness in data analysis. It is highly recommended for use with toddlers.

The principle here is to know where tables and figures came from — to know what commands created which figures and tables.

3 Fishing rods and not fish

The source code of any pdf document exchanged by the group must be available and executable. Say you run some command and then make a table or a graph. Then you produce a nice little report on your work for use among the working group. Eventually we want to use pieces of that report (tables, graphs, paragraphs) in a publishable paper. Thus, that report must not have been created using copying and pasting. It must be created using LaTeX (or HTML or OpenOffice or something else that the working group standardizes on). And, in fact, it must be created using the reproducible markup system like Sweave (if you are using LaTeX and R as your working group tools). Alternatively, you can put comments into your report detailing where the figures and tables came from (i.e. which command file produced them). Or include details about which code produced which figures in your MANIFEST.txt.

In order to know that "that plot on page 10" is from a file called "fig1.pdf" you need to document it somehow. I can imagine a system with very disciplined use of Word/OpenOffice — perhaps using special auxiliary "listoffigures.txt" files or something. In my experience, it is much easier to just use LaTeX (which can be used in a very user friendly way via LyX).

The Sweave system is even nicer because you can have a LaTeX file with R code right inside of it! I showed Sweave with plotting above so here is an example in which a regression table is produced automagically:

	Coef	Std. Err.	95% CI	
(Intercept)	1.51	0.16	1.19	1.83
Gini2004	-0.01	0.00	-0.02	-0.00
meanpr	-0.15	0.02	-0.19	-0.11
n: 53 , $\hat{\sigma}$: 0.278				

Table 1: People living in countries with unequal income distributions report less protest activity to World Values Survey interviewers than people living in countries with relatively more equal income distributions, controlling for mean political rights as measured by Freedom House 1980–2000

Right now it is hard to beat Sweave as a system for enhancing the collaboration of research groups doing quantitative work in political science. Something will make our lives easier than Sweave (or odfWeave) in the future. Then we'll change.

4 Enabling revision. Preventing clobbering.

Group work requires version control. If you are using some kind of weak version control system (which is much much better than none at all), like Dropbox, then you must communicate with other folks in your group before you edit existing files. That is, only one of you can edit a given file at a time. This prevents your work (or your colleagues work) from getting lost when you both try to save the same file on top of each other. If you find that you are needing to work on the same files at the same time, then you should work on establishing your own shared version control system. Free options include launchpad, github, sourceforge for open source projects (i.e. papers you are writing which you are happy to share with others). Each of those services include paid versions too. One may also use Dropbox as a kind of server for version control: checking out files from the Dropbox directory into a local working directory.

We use subversion with our own research group, and I use it for all of my own projects. Subversion and bazaar and git are all great. They mainly differ in the extent to which you need to run a server. Subversion requires a server and we are lucky that the National Center for Supercomputing Applications provides such a server for us. However, the price of such hosting may not be that much using one of the many webhosting services out there.

An excellent, simple, and robust version control system is to merely rename your files with the date and time of saving them: thedoc.txt becomes thedoc25092009-22:40.txt. And then when you reach certain milestones you can rename the file accordingly: thedocAPSA2009.txt [for the one sent to discussants at APSA] or thedocAPSR2015.txt [for the version eventually sent to the APSR six years after you presented it at APSA]. The systems I mentioned above all allow this kind of thing and are more elegant, but you can do it by hand too as long as you don't mind taking up a lot of disk space and having many many "thedoc..." files around.

5 Obey Nagler's Maxims

Read Nagler (1995). Comment your code. Luckily, if you are using a literate programming practice (Sweave, odfWeave, using R2HTML, etc..), you can write paragraphs to surround your code. If not, then use whatever commenting system exists in your analysis system.

A given analysis of a given collection is a long series of decisions. You need to be able to justify them. Often you justify them in the paper you are writing (in the text surrounding the R code chunks in your Sweave document). Some decisions will be too small and technical for inclusion in the paper itself. These need to be documented in the code itself.

Using R within Sweave, for example, the comment is marked using the pound sign. Notice that I've left in a regression not run for the paper directly, but run as a part of the due diligence process of writing the paper.

```
<<reg1>>
##Fitting the regression with x/1000 for ease of interpretation
thelm<-lm(y~I(x/1000))
##Fits using a quadratic term as suggested by Blah(1965) did not enhance the fit of t
##thelmsq<-lm(y~I(x/1000)^2)
```

If you need to sleuth down something more than a small detail, but yet the sleuthing is really a footnote to the article, then you can create another file to explore and report on those questions for the research group. The writing in such a memo can be more informal, but the norms of reproducibility ought to be the same as for the main paper. After all you will be releasing both documents (probably) publicly upon publication of the article if not before.

(and read Jonathan Nagler's piece on this too)

6 File Naming

(6) Name your files with evocative and descriptive names. Your collaborators are less likely to call you at midnight asking for help if your files are named "regressions.R" than if your files are called "temp" (note the use of the extension .R to tell us that the file contains R commands. use extensions like this as a standard practice to help you and your computer get along.)

(other stuff I've forgotten?)

[examples: Examples of reprodible research using R vignette system [jake's Manifest Effects paper?], and a more complex system that required more time instensive simulation using a Makefile [jake and ben and marks's JASA archive]?]

7 Constructive Collaboration

We all always collaborate. Many of us collaborate with groups of people at one moment in time as we race to produce a paper in time for some deadline. All of us collaborate with ourselves over time. An idealized version of self-collaboration goes something like this: We begin a project with a few notes. We write a conference paper. We revise this paper and send it to a journal. The journal asks for revisions. We revise and resubmit. Our paper is accepted. We revise a bit more. The paper is published. Later another scholar reacts to our paper. We write another paper based on said reactions (draft, conference paper, submission, revision, acceptance, revision). These processes occur over years of time. So, I collaborate with a self 5 or 10 or even more years hence when I write a paper. [Ask anyone who has turned their dissertation into a book how long it took from them from dissertation prospectus to physical published book for an example of a common time span in which self-collaboration is required.]

The time-frames over which collaboration are required — whether among a group of people working together or within a single scholar's productive life — are much longer than any given version of any given software will easily exist. All except for plain text. Thus, even as we extol version control systems, one must have a way to ensure future access to them in a form that will still be around when the cockroaches take over political science departments (and thereby finally make C-x M-x C-c a reasonable way to instruct Emacs to do something [insert url citing this argument — that Emacs was really made for cockroaches or other six legged creatures.]

In the end, following these practices allows your analyses of your collections to be credible — and, in the case of criticism, you can always tell antagonists to go redo your analysis if they really don't believe you.

"[I]f the empiricalbasis for an articleor book cannotbe reproduced,of what use to the discipline are its conclusions? What purposedoes an articlelike this serve? (King, 1995, 445)

References

King, G. (1995), "Replication, replication," *PS: Political Science and Politics*, 28, 444–452.

Nagler, J. (1995), "Coding Style and Good Computing Practices," *PS: Political Science and Politics*, 28, 488–492.

Whitman, W. (2008), *Leaves of Grass [1855]*, Project Gutenberg, chap. Song of Myself (51).