A Few Easy Pieces: How to enhance your relationship with your future self.

Jake Bowers[*]

February 27, 2011

> Do I contradict myself?
> Very well then I contradict myself,
> (I am large, I contain multitudes.)
> (Whitman, 1855)

An idea is born in a coffee shop, a seminar, a quiet walk. It is a cold winter in 2011, but the idea energizes the student. The idea is why he went to graduate school.

This idea inspires a seminar paper in the spring. And a conference paper arises from the seminar paper in collaboration with another student in 2012. A dissertation chapter descends from the conference paper in 2013. Other dissertation chapters take up 2014. A submission to a journal with the original co-author and a new collaborator happens in 2015. Revision and resubmission wait until 2017 while harried editors, reviewers and authors strive to balance teaching, service, and life. The three collaborators have luckily found work in three different universities around the world. 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

[*]Many thanks to Kevin Quinn and Cara Wong 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. The source code for this document may be freely downloaded from `https://github.com/jwbowers/workflow`.

the paper. What would happen if only the first 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?

The first author convenes a three way video conference with the other collaborators while commuting home in his flying car. The group must go back to the analyses. Which ones? The ones from 2011? Or 2012? Or 2018? Where are the files? The next day, one member of the group finds some of the files. Now the checks and re-analyses should be easy. Right? The student-now-faculty member is the same person and should remember the reason for those bits of code (or at least should remember which series of mouse clicks were used to produce the numbers for that crucial table — and those mouse clicks should not be time consuming to redo exactly as they were done in 2011, or was it 2015?). And, of course, the way Microsoft Word/Stata/SPSS/R/LISREL stores files and the way that your machine reads and writes them is the same, right — since Windows and Mac OS X have always existed and will always continue to exist more or less as they current exist. And the group knows exactly which bit of code produced which table and which figure, right? And they wrote their code following Nagler's Maxims Nagler (1995) and being following King's Replication Standard King (1995) right?

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

1 Meaningful code requires 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/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 Type, don't click.

All tables and graphs should be 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

[1] The command `please-fit` and some other R functions used in this essay come from the `MayIPleaseDoStatistics` package which emphasizes politeness in data analysis. The commands `please-fit` and `please-plot` can be blocked and `would-you-please-fit` or `may-I-please-have-a-plot` can be required using `options(politeness=99)`

so with just a few edits to the code.

Using R, for example, I might specify that the file `fig1.pdf` was produced by the command, where the `please-plot` function was loaded elsewhere.

```
pdf('fig1.pdf')
please-plot(outcome by explanatory)
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, to produce the Figure 1, I insert the following chunk of code into the LaTeX document, and then I asked R to process the document, and then produce a pdf file in which the code has disappeared to be replaced with an actual figure.[2]

```
##  Make a scatterplot of Protest by Inequality
with(good.df,plot(Gini2004/100,protac2000,
                  xlab='Gini Coefficient 2004 (UNDP)',
                  ylab='Mean Protest Activities\n(World Values Survey 1980-2000)',
                  cex=.8))
##  Label a few interesting points
with(good.df[c("EGY","JOR","USA","SWE","CHL"),],
     text(Gini2004/100,protac2000,labels=Nation,srt=0,cex=.6,pos=3,offset=.1))
```

[2] One can insert the same code chunk in an OpenOffice document and use the odfWeave package to produce OpenOffice documents in which figures and tables replace code. One may also use the LyX interface to LaTeX to write LaTeX documents within a WYSIWYG environment.

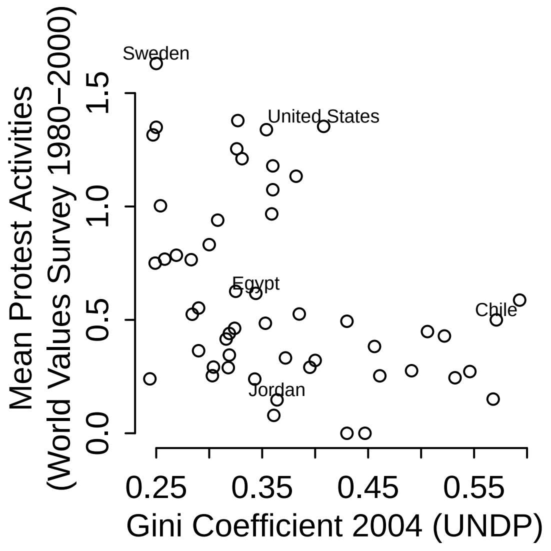

Figure 1: Protest activity by income inequality (Norris, 2009, from).

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

3 Enjoy literate data analysis.

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) (Leisch, 2005). 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 a word processor (like Open Office or Word) — 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 in Figure 1, those who view the code for this essay will see how the Table 1 was also generated directly from a regression object.

	Coef	Std. Err.	95% CI	
Intercept	1.51	0.16	1.19	1.83
Income Inequality (lower=more equal)	−1.04	0.43	−1.89	−0.18
Mean Political Rights (lower=more rights)	−0.15	0.02	−0.19	−0.11
	n: 53, resid.sd:0.28, R^2:0.57			

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, adjusting for average political rights as measured by Freedom House 1980–2000. Data from (Norris, 2009).

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 to prevent clobbering.

Group work requires version control.[3] Many people are familiar with the “Track Changes” feature in modern WYSIWYG word processors or the fact that Dropbox allows one to recover previous versions of files. These are both kinds of version control. When collaborating with yourself or others, it is

[3] See X for more discussion of what version control is.

useful to see what has changed, to feel free to experiment and then to dump the experiment in favor of previous work, to have multiple "releases" of the same document (one to MPSA, one to APSR, one to your parents) without requiring that spawn many copies of the same document and risk confusion and clobbering. Clobbering is what happens when your future self or your other collaborators saves an old version of a file over a new version — erasing good work by accident.

Of course if you rely on Dropbox or "track changes" for version control, you must communicate with other folks in your group before you edit existing files. Only one of you can edit and save 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. (Notice that this is different from directly working on files within your Dropbox managed directories.)

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. [See Y in this issue for much more about version control and collaboration.]

Fancy version control systems are not required, however. I suspect that Google Docs allows a kind of version tracking and collaboration as well. An excellent, simple, and robust version control system is to merely rename your files with the date and time of saving them: thedoc.tex becomes thedoc25-12-2011-23:50.tex. Be sure to include year in the file names — remember, the life of an idea is measured in years. Also, if you use this method, spend a little extra time to ensure that you do not clobber files when you make typos in the file-names.

If you are wise enough to have saved your documents as plain text (Such as LaTeX source (with or without R/Stata code chunks) [4] then you can easily compare documents using the many utilities available for comparing text files [FileMerge on OS X is pretty, but ediff and diff for Unix are very useful, I am sure that Windows has many other options]. Adobe Acrobat allows one to compare differences in pdf files. OpenOffice supports a "Compare Documents" option.

When you reach certain milestones you can rename the file accordingly: thedocAPSA2009.tex — for the one sent to discussants at APSA — or thedocAPSR2015.tex — 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 much more elegant and capable, 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 No data analyst is an island.

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 as well a technical comments in the code itself. If you are not using a strictly literate programming practice then use whatever commenting protocol exists in your analysis language.

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.

```
##  Repeat the regression run for Table~\ref{tab:protest}
##  Do political rights predict protest reporting?
lm1<-lm(protac2000 ~ I(Gini2004/100) + meanpr, data=good.df)
##  Q: Was it worthwhile controlling for political rights?
lm2<-lm(protac2000~I(Gini2004/100),data=good.df)
coef(lm2)[2]-coef(lm1)[2]
```

[4] A quick Google search of "Sweave for Stata" turned up lots of resources for literate programming with Stata.

```
I(Gini2004/100)
    -0.751
 ##  A: Yes (around 3/4 an act of protest difference)
```

If you need to track 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.

Notice one another benefit of coding for an audience: we learn by teaching. By assuming that others will look at your code, you will be more likely to write clearer code, or perhaps even to learn more about what you are doing as you do it.

6 Minimize error by testing.

Imagine that you reported bootstrap confidence intervals in your famous article. Now the statisticians at the UN are worrying about your bootstrap confidence interval. You'd like to evaluate your bootstrap procedure. Although nice code exists for bootstrapping linear models, no nice code exists to bootstrap the bootstrap. Of course, the code required is not complex, but since you are writing your own code you worry about getting it right. Now, 9 years after the idea, you've had lots of time to appreciate problems arising from bugs and errors in data analysis code.

Now, if you had a moment to think in between teaching that new class, reading books for the awards committee, reading application files for the admissions committee, staying home with a sick child, and your own current research, you might say to yourself, "Before I write new code, I should write a test of the code. I should write a little bit of code that let's me know that my double-bootstrap procedure actually does what it is supposed to do."

Of course, this idea, like most others is not new. When large groups of programmers write code for multi-million dollar programs the question about avoiding error looms large. The idea of test driven development and the idea that one ought to create tests of small parts of one's code arose to address such concerns. For the social scientist collaborating with her future self and/or a small group of collaborators.

Here is an example of this idea in a very simple form. I want to write a function to multiply a number by 2. If my function works, when I give it the number 4, I should see it return the number 8 and when I give it -4, I should get -8.

```
  ##  The test function:
  test.times.2.fn<-function(){
    ##  This function tests times.2.fn
    if(times.2.fn(thenumber=4) == 8 &
       times.2.fn(thenumber=-4) == -8){
      print("It works!")
    }else{print("It does not work!")}
  }
 ##  The function:
 times.2.fn<-function(thenumber){
   ##  This function multiplies a scalar number by 2
   ##  thenumber is a scalar number
   thenumber+2
 }
 ##Use the test function
 test.times.2.fn()
[1] "It does not work!"
```

Ack! I mistyped "+" for "*". Good thing I wrote the test!

7 Take advantage of file names to convey information to your future self.

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 "inequality-and-protest-analyses.R" than if your files are called "temp9" or "supercalifragilisticexpialidocious" (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.)

8 Copy and improve on others' examples.

Lots of people are thinking about "reproducible research" and "literate programming" these days. Google those terms. Of course the devil is the details: Here I list a few of my own attempts at enabling reproducible research.

You'll find many other inspiring examples on the web. Luckily, the open source ethos aligns nicely with academic incentives, so we are beginning to find more and more people offering their files online for copying and improvement. By the way, if you do copy and improve, it is polite to alert the person from whom you made the copy about your work.

I have experimented with three systems so far: (1) for one paper we simply included a Sweave document and data files into a compressed archive Bowers and Drake. (2005); (2) for another more computing intensive paper we assembled a set of files which enabled reproduction of our results using the "make" system (Bowers et al., 2008); and (3) recently I have tried the "compendium" approach (Gentleman, 2005; Gentleman and Temple Lang, 2007) which embeds an academic paper with the R package system Bowers (2011). The benefit of this last system is that one is not required to have access to unix: the compendium is downloadable from within R using `install.packages()` and it viewable using the `vignette()` function. The idea that one ought to be able to install and run and use an academic paper just as one installs and uses statistical software packages is very attractive and I anticipate that it will become ever easier to turn papers into R packages as creative and energetic folks turn their attention to the question of reproducible research.

9 Remember that research ought to be credibile communication.

> [I]f the empirical basis for an article or book cannot be reproduced, of what use to the discipline are its conclusions? What purpose does an article like this serve? (King, 1995, 445)

We all always collaborate. Many of us collaborate with groups of people at one moment in time as we race against a deadline. All of us collaborate with ourselves over time.[5] The time-frames over which collaboration are required — whether among a group of people working together or within a single scholar's productive life or probably both — are much longer than any given version of any given software will easily exist. Plain text is the

[5] What is a reasonable time-span for which to plan for self-collaboration on a single idea? Ask your advisers how long it took them from idea to dissertation to book (articles).

exception. 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 sentient cockroaches finally join political science departments (by then dominated by cetaceans after humans are mostly uploads).[6]

Now, imagine that the UN never hears of your work, or even that your article does not spawn debate. Why then would you spend time to communicate with your future self and others? My own answer to this question is because I want my work to be credible and useful to myself and other scholars even if each article does not immediately change the world. What I report in my data analyses should have two main characteristics: (1) the findings of the work should not be 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).

Assume we want others 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 in some collection of stuff. Now, it might be easy to convince others 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 broke it up into understandable parts and reorganized and summarized it. 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 and future people about our analyses.

Now, say your collections of stuff are large or complex and 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? This essay has suggested a few concrete ways to enhance the believeability of such scholarly work. In addition, these actions (as summarized in the section headings of this essay) make collaboration within research groups more effective. Believeability comes in part from re-

[6] The arrival of the six-legged social scientists revives Emacs and finally makes Ctrl-a Ctrl-x Esc-x Ctrl-c a reasonable key combination.

producibility and research groups often need to be able to reproduce in part or in whole what different people in the group have done.

In the end, following these practices and those recommended by X and Y in this issue allows your computerized analyses of your collections stuff of to be credible. Finally, if the UN quibbles with your analyses, your future self can shoot the archive required to reproduce your work (in still intelligible plain text, analyzed using commented code so that folks can translate to whatever system succeeds R, or since you used R, you can include a copy of R and all of the R packages you used in your final analyses in 2018 in the archive itself.) You can say, ”Here is everything you need to reproduce my work.” To be extra helpful you can add ”Read the README file for futher instructions.” And then you can get on with your life: maybe the next great idea will occur when your 4 year old asks a wacky question after stripping and painting her overly cooperative 1 year old brother purple, or teaching a class, or in a coffee shop, or on a quiet walk.

References

Bowers, J. (2011), “Reproduction Compendium for: “Making Effects Manifest in Randomized Experiments”,” http://hdl.handle.net/1902.1/15499.

Bowers, J. and Drake., K. (2005), “Reproduction archive for: ‘EDA for HLM:Visualization when Probabilistic Inference Fails’,” http://hdl.handle.net/1902.1/13376.

Bowers, J., Hansen, B., and Fredrickson, M. (2008), “Reproduction archive for: ‘Attributing Effects to A Cluster Randomized Get-Out-The-Vote Campaign’,” http://hdl.handle.net/1902.1/12174.

Gentleman, R. (2005), “Reproducible research: A bioinformatics case study,” *Statistical Applications in Genetics and Molecular Biology*, 4, 1034.

Gentleman, R. and Temple Lang, D. (2007), “Statistical analyses and reproducible research,” *Journal of Computational and Graphical Statistics*, 16, 1–23.

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

Leisch, F. (2005), *Sweave User Manual.*

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

Norris, P. (2009), "Democracy Crossnational Data, Release 3.0," http://www.hks.harvard.edu/fs/pnorris/Data/Data.htm, data file.

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