Six steps to a better relationship with your future self.

Jake Bowers*

March 25, 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. On this gray day in 2011, the idea dispells February's doldrums. The student rushes home, mind racing, the cold ignored.

This idea inspires a seminar paper in the spring. 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 research, teaching, service, and life. By now, the three lucky collaborators work as professors in three different universities. In 2018 a child is born and a paper is published. The United Nations takes an interest in the paper in 2019 and hosts a conference to discuss implications of the research. In 2020 a first year graduate student in a coffee shop has an idea that challenges the results in the now famous paper. She presents her paper at a conference in 2021. What would happen if the authors had controlled for X? Or included information

*I owe many thanks to Mark Fredrickson, Brian Gaines, Kieran Healy, Kevin Quinn and Cara Wong for direct help on this document and to Mika LaVaque-Manty and Ben Hansen for many useful discussions on this topic. The source code for this document may be freely downloaded and modified from https://github.com/jwbowers/workflow.

now available but missing in 2012? 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 during his homeward commute after putting his flying car in auto-drive mode.[1] 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 who has kept some hard-drives around out of nostalgia finds some of the files.[2] Now re-analyses should be easy. Right? The student, now professor, 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 that mousing should not be time consuming to redo exactly as it was done in 2011 . . . or was it 2015?). Right? And, of course, the way Microsoft Word/Stata/SPSS/R/LISREL understands files and the way that machines in 2021 read and write them is the same — since Windows and Mac OS X have always existed and will always continue to exist more or less as they currently exist? Right? 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 King's Replication Standard (King, 1995) right?

Right? Wrong? If the collaborators find themselves say "Wrong" in answer to the questions posed here then reproducing, updating, or changing the original analyses will take a lot of time. If reproduction is hard to do, then the reputations of the scholars will suffer and, more importantly, world peace will have been delayed. This essay provides some suggestions for practices which will make such reproduction occur much more easily and quickly in the event that famous papers require special scrutiny. Specifically, this piece aims to amplify some of what we already ought to know (King, 1995; Nagler, 1995), and to add to some of those ideas given current practices, platforms, and possibilities.

[1] One assumes that video chatting during manual driving of flying cars will have been outlawed in his state by 2020.

[2] This is the same guy who still owns cassette tapes and compact discs.

1 Data analysis is computer programming.

All results (numbers, comparisons, tables, figures) should arise from code, not series of mouse clicks or copying and pasting. If I wanted to re-create the figure you created but including a new variable or specification, I should be able to do so with just a few edits to the code rather than knowledge of how you used your pointing device in your graphical user interface.

Using R (R Development Core Team, 2011), for example, I might specify that the file `fig1.pdf` was produced by the following commands in a file called `fig1.R`.[3]

```
thedata <- read.csv("Data/thedata-15-03-2011.csv") ## Read the data
pdf('fig1.pdf') ## begin writing to the pdf file
please-plot(outcome by explanatory using thedata. red lines please.)
please-add-a-line(using model1)
## Note to self: a quadratic term does not add to the substance
## model2<-please-fit(outcome by explanatory+explanatory^2 using thedata
## summary(abs(fitted(model1)-fitted(model2)))
dev.off() ## stop writing to the pdf file
```

Now, in the future if I wonder how "that plot on page 10" was created, I will know: (1) "that plot" is from a file called `fig1.pdf` and (2) `fig1.pdf` was created in `fig1.R`. In a future where R still exists, changing the figure will require quick edits of commands already written. In a future where R does not exist, I will at least be able to read the plain text R commands and use them to write code in my new favorite statistical computing language: R scripts are written in plain text, and plain text is a format that will be around as long as computer programmers write computer programs. [4]

Moreover, realize that file names send messages 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-figures.R` than if your files are called `temp9` or `supercalifragilisticexpialidocious`. The extension `.R` tells us and the operating system that the file contains R commands. This part of the filename

[3] The command `please-plot` and some other R functions used in this essay come from the `MayIPleaseDoStatistics` package which emphasizes politeness in data analysis. Functions like `please-plot` can be blocked and more polite versions such as `may-I-please-have-a-plot` can be required using `options(politeness=99)`

[4] What is more, since R is open source, I will be able to download an old version of R, download an old fashioned open-source operating system (like Ubuntu 10), and run the old-fashioned statistical computing environment in the old-fashioned operating system in a virtual machine on my new-fashioned actual machine.

enables us to quickly search our antique hard drives for files containing R scripts.

Step 1 Know the provenance of your results so that your future self or current collaborators can quickly and easily reproduce and thus change your work.

2 No data analyst is an island for long.

Data analysis involves a long series of decisions. Each decision requires justification. Some decisions will be too small and technical for inclusion in the published article itself. These need to be documented in the code itself (Nagler, 1995). Paragraphs and citations in the publication will justify the most important decisions. So, one must code to communicate with yourself and others. There are two main ways to avoid forgetting the reasons you did something with data: comment your code and tightly link your code with your writing.[5]

2.1 Code to communicate: Comment your code.

Comments — unexecuted text inside of a script — are a message to collaborators (including your future self) and other consumers of your work. In the above code chunk, I used comments to explain the lines to readers unfamiliar with R and to remember that I had tried a different specification but decided not to use it because adding the squared term did not really change the substantive story arising from the model.[6] Messages left for your future self (or near-future others) help retrace and justify your decisions as the work moves from seminar paper to conference paper to poster back to paper to dissertation and onwards.

Notice one other 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 think more deeply about what you are doing as you do it.

[5] One can also try the R command `put-it-in-my-mind(reason,importance='high')` to firmly place a reason for a decision into the mind of the analyst. I myself have not had much luck with this function.

[6] R considers text marked with `#` as a comment.

Comment liberally. Comments are discarded when R runs analysis or LaTeX turns plain text into page images, so only those who dig into your work will see them.

2.2 Code to communicate: Literate programming.

Imagine you discover something new (or confirm something old). You produce a nice little report on your work for use in discussions of your working group or as a memo for a web or reviewer appendix. The report itself is a pdf file or some other format which displays page images to ease reading rather than to encourage reanalysis and rewriting. Eventually pieces of that report (tables, graphs, paragraphs) ought to show up in, or at least inform, the publishable paper. Re-creating those analyses by pointing, clicking, copying, or pasting would invite typing error and waste time. Re-creating your arguments justifying your analysis decisions would also waste time. More importantly, we and others want to know why we did what we did. Such explanations may not be very clear if we have some pages of printed code in one hand and a manuscript in the other.

How might one avoid these problems? Literate programming is the practice of weaving code into a document — paragraphs, equations, and diagrams can explain the code, and the code can numbers, figures, and tables (and diagrams and even equations and paragraphs). Literate programming is not merely fancy commenting but is about enabling the practice of programming itself to enable easy reproduction and communication.

For example, in §1, I suggested that we know where "that plot on page 10" comes from by making sure we had a `fig1.pdf` file produced from a clearly commented plain text file called something like `fig1.R`. An even easier solution would be to directly include a chunk of code to produce the figure inside of the paper itself. This paper, for example, was written in plain text using LaTeX markup with R code chunks to make things like Figure 1: this combination of LaTeXand R is called Sweave (Leisch, 2005).[7]

```
This paper, for example, was written in plain text using \LaTeX
markup with R code chunks to make things like
Figure~\ref{fig:giniprot}: this combination of \LaTeX and R is
called Sweave \citep{Leis:2005}.\footnote{Support for Sweave is
  included with R.}
\begin{figure}[h]
```

[7] Support for Sweave is included with R.

```
\centering
 <<fig1plot,fig=TRUE>>=
 ##  Make a scatterplot of Protest by Inequality
 with(good.df,plot(gini04,protac00,xlab='Gini Coefficient 2004 (UNDP)',
                   ylab='Mean Protest Activities\n(World Values Survey 1980-2000)'))
 ##  Label a few interesting points
 with(good.df[c("EGY","JOR","USA","SWE","CHL"),],
      text(gini04,protac00,labels=Nation))
 @
\caption{Protest activity by income inequality \citep[from][]{norris2009data}.}
\label{fig:giniprot}
\end{figure}
```

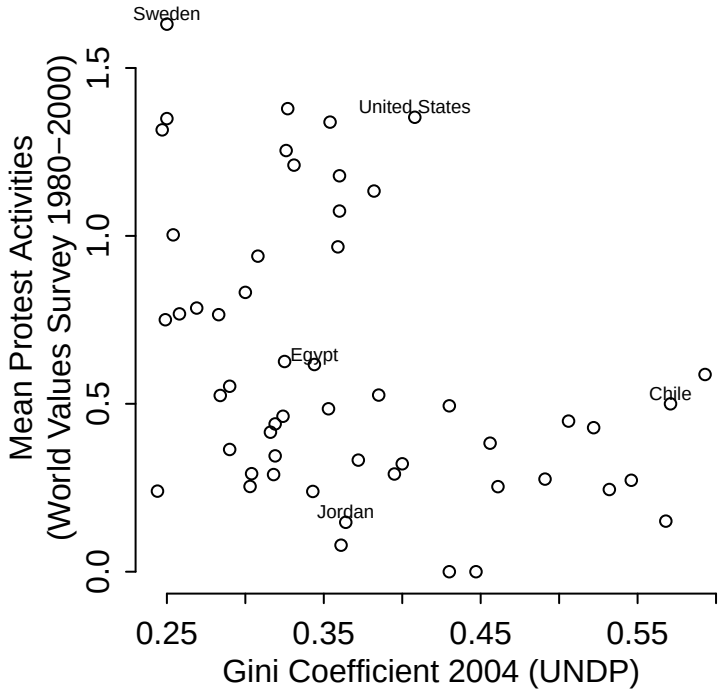

Figure 1: Protest activity by income inequality (from Norris, 2009).

By using `\label{fig:giniprot}`, I do not need to keep track of the figure number, nor do extra work when I reorganize the document in response to reviewer suggestions. Nor do I need a separate `fig1.R` file or `fig1.pdf` file.

Tables and other numerical results are also possible to generate within the source code of a scholarly paper. Those who view the code for this essay will see how Table 1 was also generated directly from a regression object.[8]

	Coef	Std. Err.	95% CI	
Intercept	1.5	0.2	1.2	1.8
Income Inequality (lower=more equal)	−1.0	0.4	−1.9	−0.2
Mean Political Rights (lower=more rights)	−0.2	0.0	−0.2	−0.1
	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).

Literate data analysis is not the same as Sweave, even if Sweave is a nice implementation.[9] LyX offers a WYSIWYG environment for LaTeX which supports Sweave. And the `odfWeave` package in R allows the use of OpenOffice documents in exactly the same way.[10] If your workflow does not involve LaTeX and R, you can still implement some of the principles here. Imagine creating a style in Microsoft Word called "code" which hides your code when you print your document, but which allows you do at least run each code chunk piece by piece [or perhaps there are ways to extract all text of style "code" from a Microsoft Word document]. Or imagine just using some other kind of indication linking paragraphs to specific places in code files. There are many ways that creative people can program in a literate way.

Literate programming need not go against the principle of modular data analysis (Nagler, 1995): in my own work I routinely have several different Sweave files that fulfill different functions, some of them create LaTeXcode that I can `\input` into my `main.tex` file, others setup the data, run simulations, or allow me to record my journeys down blind alleys. Of course, when we

[8] Beck (2010) inspired this particular presentation of a linear model.

[9] The R project has a task view devoted to reproducible research listing many of the different approaches to literate programming for R.

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

have flying cars running on autopilot, perhaps something other than than Sweave will make our lives even easier. Then we'll change.

Step 2

> Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do. (Knuth, 1984, p. 97)

We analyze data in order to explain something about world to other scholars and policy makers. If we focus on explaining how we got our computers to do data analysis to human beings, we will do a better job with the data analysis itself: we will learn as we focus on teaching, and we will avoid errors and save time as we ensure that others (including our future selves) can retrace our steps. A document than can be "run" to reproduce all of the analyses also instills confidence in readers and can more effectively spur discussion and learning and cumulation of research as people no longer need spend weeks attempting to reproduce research which inspired them and on which they desire to build.

Keep in mind the distinction between the "source code" of a document (i.e. what computation was required to produce it) and the visible, type-set page image. Page images are great for reading, but not great for reproducing or collaborating. The source code of any document exchanged by the group must be available and executable.

3 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('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'`). Of course, it never hurts to drop in a comment pointing to the data file.

Where should one store data files? A obvious solution is always to make sure that the data file used by a command file is in the same directory as the command file. More elegant solutions 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 the project. This kind of solution is one of the things that formal version control systems do well (as discussed a bit more in §4).

The principle of modularity suggests that you separate data cleaning, processing, recoding, and merging from analysis in different files (Nagler, 1995). So, perhaps your analysis oriented files will `load('cleandata.rda')` and a comment in the code will alert the future you (among others) that `cleandata.rda` was created from `create-cleandata.R` which in turn begins with `read.csv(\url('http://www.greatfreedata.gov/dirtydata.csv'))`. 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.[11]

Now, if in the future we wonder where `cleandata.rda` came from, we might search for occurrences of 'cleandata' in the files our system. However, if such searching among files is a burden, an 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.

Step 3 We should know data where the data came from and what operations were performed on which set of data.

In the good old days, when we executed our LISREL code in batch mode, we had no choice but to tell the machine clearly, in a few easy to understand and informative lines, what files (with filenames no longer than 8 characters) to use. For example,

```
DA NI=19 NO=199 MA=CM
LA=basic.lab
CM FI=basic.cov
```

The fact that I need to articulate this idea at all arises because of graphical user interfaces: it is all too easy to use the mouse to load a data file into

[11] Of course, if you need math or paragraphs to explain what is happening in these files, you might prefer to make them into Sweave files, for which the conventional extension is .Rnw. So you'd have `create-cleandata.Rnw` which might explain and explore the different coding decisions you made, perhaps involving a factor analysis and diagnostic plots.

memory and then to write a script to analyze this file without ever noting the actual name or location of the data file.

4 Version control prevents clobbering and reconciles history.

Group work requires version control.[12] 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. More generall, when we collaborate, we'd like to do a variety of actions with our shared files. Collaboration on data analytic projects is more productive and easier when (1) it is easy to see what has changed between versions of files; (2) members of the team feel free to experiment and then to dump parts of the experiment in favor of previous work while merging the successful parts of the experiment into the main body of the paper; (3) the team can produce have multiple "releases" of the same document (one to MPSA, one to APSR, one to your parents) without spawning many possibly conflicting copies of the same document; (4) when people can work on the same files at the same time without conflicting with one another (and can reconcile their changes without too much confusion and clobbering). Clobbering is what happens when your future self or your current collaborator 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 need 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 as you write). Each of those services include paid versions too. One may also use Dropbox as a kind of server for version control: for example, one may copy files from the Dropbox directory into a local working directory so as to avoid clobbering and then working on merging

[12] Fredrickson, Testa and Weidmann (2011) and Healy (2011) in this issue also explain what version control is and why we might want to use it.

changes by hand before copying back to the Dropbox directory and replacing existing files.

We use subversion with our own research group, and I use it for all of my own projects (except this one, for which I am experimenting with git). 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. [13]

Of course, one may take advantage of many of the benefits of formal version control systems with some disciplined systems for file and directory organization. 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. If you are wise enough to have saved your documents as plain text then you can easily compare documents using the many utilities available for comparing text files.[14] 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 formal version control 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 "thedoc..." files around. If you do version control by hand, spend a little extra time to ensure that you do not clobber files when you make mistakes typing in the file-names. And, you will find yourself spending extra time reconciling changes made by different collaborators by hand; a task that modern version control systems take care of quickly and easily.

Step 4 Writing is rewriting. Thus, all writing involves versions. When we collaborate with ourselves and others we want to avoid clobbering and we want to enable graceful reconciliation of rewriting. One can do these things with formal systems of software (like subversion or git or bazaar) or with

[13] If you already pay to a hosting provider for website, you may already have the right to run a subversion or git server there. Your university or institute may have a version control system running somewhere on campus. And google will direct you to many helpful people who have installed such servers on their own diverse desktop machines. Github requires that you pay to host private repositories.

[14] Adobe Acrobat allows one to compare differences in pdf files. OpenOffice supports a "Compare Documents" option. And Google Docs will report on the version history of a document.

formal systems of file naming, file comparing, and communication or, even better, with both. In either case, plain text files will make such tasks easier, will take up less disk space, and be easier to read for the future you.

5 Minimize error by testing.

Now, back to that famous article of 2018. After reading the conference paper critique of 2021 (that came from the seminar paper of 2020), the statisticians at the UN worry about the bootstrap confidence intervals presented in the original paper.[15] So, now authors would like to evaluate their 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 they are writing custom code they worry about getting it right. As they've struggled to respond to the critiques of their paper, they've had lots of time to appreciate problems arising from bugs, errors, and typos in data analysis and code.

Now, if they had a moment to think in between teaching that new class, reading books for the awards committee, evaluating application files for the admissions committee, feeding popsicles to a sick child, and undertaking the odd bit of research, they might say to themselves, "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: Say, 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.

[15] Perhaps they should be worried about the deeper substantive critiques offered by the student, but they are statisticians and so focus on the stats. The policy makers of 2021 were cowed by the methodological virtuosity of the 2018 article, and so, even though they had the same substantive concerns as the student, they kept their mouths shut at the mini-conference to avoid looking dumb in front of their bosses.

```
##  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![16]

Step 5 No one can forsee all of the ways that a computer program can faile. One can, however, at least make sure that it succeeds in doing the task motivating the writing of the code in the first place.

6 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

[16] A more common example of this kind of testing occur everyday when we recode variables into new forms but look at a crosstab of the old vs. new variable before proceeding.

we assembled a set of files which enabled reproduction of our results using the `make` system (Bowers, Hansen and Fredrickson, 2008); and (3) recently I have tried the "compendium" approach (Gentleman, 2005; Gentleman and Temple Lang, 2007) which embeds an academic paper within the R package system (Bowers, 2011). The benefit of this last approach is that one is not required to have access to a command line for `make`: the compendium is downloadable from within R using `install.packages()` and it viewable using the `vignette()` function in any operating system than runs R.[17] 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.

Step 6 We all learn by doing. When we share reproduction materials we improve both cumulation of knowledge as articulated in (Freese, 2007; King, 1995) and our methods for doing social science. As we copy and improve upon each others modes of doing work we enhance our collective ability to believe each other for future scholars to believe us, too.

7 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.[18] 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

[17] Notice that my reproduction archives and/or instructions for using them are hosted on the Dataverse which is another system to enhance academic collaboration across time and space.

[18] 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 to move from idea to dissertation to publication.

of any given software will easily exist. Plain text is the 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).[19]

But what if the UN never hears of your work, or, by some cruel fate, 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 that 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 should be able to reproduce the findings. That is, the work represents a shared experience — and an experience shared without respect to the identities of others (although requiring some common 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

[19]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.

groups more effective. Believeability comes in part from reproducibility and researchers often need to be able to reproduce in part or in whole what different people in the group have done or what they, themselves, did in the past.

In the end, following these practices and those recommended by Fredrickson, Testa and Weidmann (2011) and Healy (2011) in this issue allows your computerized analyses of your collections of stuff to be credible. Finally, if the UN quibbles with your analyses, your future self can shoot them the archive required to reproduce your work.[20] 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

Beck, Nathaniel. 2010. "Making Regression and Related Output More Helpful to Users." *The Political Methodologist* 18(1):4–9.

Bowers, Jake. 2011. "Reproduction Compendium for: "Making Effects Manifest in Randomized Experiments"." http://hdl.handle.net/1902.1/15499.
URL: *http://hdl.handle.net/1902.1/15499*

Bowers, Jake, Ben B. Hansen and Mark M. Fredrickson. 2008. "Reproduction archive for: 'Attributing Effects to A Cluster Randomized Get-Out-The-Vote Campaign'." http://hdl.handle.net/1902.1/12174.
URL: *http://hdl.handle.net/1902.1/12174*

Bowers, Jake and Katherine W. Drake. 2005. "Reproduction archive for: 'EDA for HLM:Visualization when Probabilistic Inference Fails'." http://hdl.handle.net/1902.1/13376.
URL: *http://hdl.handle.net/1902.1/13376*

[20] Since you used plain text, the files will still be intelligible, 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 even throw in a copy of Ubuntu 10 and an open source virtual machine running the whole environment.

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).

Freese, J. 2007. "Replication standards for quantitative social science." *Sociological Methods & Research* 36(2):153.

Gentleman, Robert. 2005. "Reproducible research: A bioinformatics case study." *Statistical Applications in Genetics and Molecular Biology* 4(1):1034.

Gentleman, Robert and Duncan Temple Lang. 2007. "Statistical analyses and reproducible research." *Journal of Computational and Graphical Statistics* 16(1):1–23.

Healy, Kieran. 2011. "Choosing Your Workflow Applications." *The Political Methodologist* 18(2).

King, Gary. 1995. "Replication, replication." *PS: Political Science and Politics* 28(3):444–452.

Knuth, Donald E. 1984. "Literate programming." *The Computer Journal* 27(2):97–111.

Leisch, Friedrich. 2005. *Sweave User Manual*.
URL: *http://www.ci.tuwien.ac.at/ leisch/Sweave*

Nagler, Jonathan. 1995. "Coding Style and Good Computing Practices." *PS: Political Science and Politics* 28(3):488–492.

Norris, Pippa. 2009. "Democracy Crossnational Data, Release 3.0." http://www.hks.harvard.edu/fs/pnorris/Data/Data.htm. Data file.
URL: *http://www.hks.harvard.edu/fs/pnorris/Data/Data.htm*

R Development Core Team. 2011. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. ISBN 3-900051-07-0.
URL: *http://www.R-project.org*

Whitman, Walt. 1855. *Leaves of Grass*. Project Gutenberg [2008] chapter Song of Myself, p. 51.
URL: *http://www.gutenberg.org/files/1322/1322-h/1322-h.htm*