Six steps to a better relationship with your future self.

Jake Bowers*

March 24, 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. What would happen if the authors had controlled for X? Or included information now available but

*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`.

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 2020 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 answering "wrong" to the questions posed here then reproducing, updating, or changing the analyses will take a lot of time. If reproduction is hard to do, then the reputations of the scholars will suffer and, more importantly, the UN will be unable to implement a master plan for world peace. 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 from Nagler (1995) and King (1995), and to add to some of those ideas given current practices, platforms, and propensities.

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

[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.

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, 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.)
dev.off() ## stop writing to the pdf file
```

Now, in the future if I wonder how "that plot on page 10" was created: (1) "that plot" is from a file called `fig1.pdf` and `fig1.pdf` was created in `fig1.R`. In a future where R still exists, if I wanted to change the figure, I could edit the commands quickly and easily to do so (since R commands are written in plain text, and plain text is a format that will be around for a long time to come).

Principle Know the provenance of your results so that your future self or current collaborators can quickly and easily reproduce your work (and make changes).

2 No data analyst is an island for long.

Data analysis involves a long series of decisions. Each decision requires justification. The text itself will justify the most important decisions. Some decisions will be too small and technical for inclusion in the paper itself. These need to be documented in the code itself (Nagler, 1995).

2.1 Comment your code.

Comments — unexecuted text inside of a script — are a message to your future self and other consumers of your work. Luckily, if you are using a literate programming practice (ex. Sweave or odfWeave), you can write paragraphs to surround your code as well as technical comments in the code

[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)`

itself. If you are not using a strictly literate programming practice then use whatever commenting protocol exists in your analysis language. R, for example, considers text marked with “#” as a comment. Here for example I comment about a linear model run for a paper and also include comments about another model run for due diligence.

```
##  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]
```

```
I(Gini2004/100)
         -0.751
```

```
##  A: Yes (around 3/4 an act of protest difference)
```

Messages left for your future self (or near-future others) help retrace and justify the steps of your data-analytic journey in case you ever need to revist them as the work moves from seminar paper to conference paper to poster back to paper to dissertation and onwards. 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.

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

2.2 Programming can be literate.

Sometimes simple comments in the main command file are not enough: imagine if the authors had arrived at the bootstrap only after running simulations to assess the coverage of that procedure compared to a profiled likelihood based approach. Rather than include time consuming simulations in the body of their main Sweave document, they might have written some memos that discuss the tradeoffs and implementation decisions involved in such a comparison. In the main text they would have just read in the results of the simulation study with an appropriate comment:

```
## We assessed the coverage of the bootstrap intervals against
## direct inspection of the likelihood profile in the file
## Memos/shouldwebootstrap.Rnw which produced bsresults.rda

##Here we read in the results of the bootstrap procedure:
load("Memos/bsresults.rda") ## using a compressed binary format here for speed
```

Now that memo itself (or perhaps the main document) probably involves sentences and paragraphs explaining an analysis. Literate programming or literate data analysis happens when the code of the analysis is mixed side by side with the paragraphs explaining, interpreting, foreshadowing, discussing the analysis.[4]

If comments are guideposts, literate programs are full fledged travelogues. Why might one care to use a literate programming practice?

Say you run some command, discover something new (or confirm something old) and then make a claim backed up by table or a figure. You produce a nice little report on your work for use in discussions of the working group. The report itself is a pdf file or some other format which emphasizes reading.[5] Eventually we want to use pieces of that report (tables, graphs, paragraphs) in a publishable paper. It would be a waste of time if you had to re-create those analyses by pointing, clicking, copying, or pasting. It would also be a waste of time if you couldn't remember why you decided to use a quadratic instead of a cubic term and had to re-create that argument. So, we might use literate programming because we don't want to waste time.

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." If one were using Sweave, one could even more easily have written the following to produce the desired figure.

```
\begin{figure}[H]
\begin{center}
 <<fig1plot,echo=FALSE,fig=TRUE,width=3,height=3>>=
 par(bty="n",xpd=TRUE,pty="s",tcl=-.25)
 ##  Make a scatterplot of Protest by Inequality
 with(good.df,plot(Gini2004/100,protac2000,
```

[4]This document is written in such a way using LaTeXand R code mixed together in what is known as an Sweave file (Leisch, 2005). You can download and view and edit this document as a plain text file from https://github.com/jwbowers/workflow.

[5]like html, postscript, dvi or even the wordprocessor formats like rtf or doc or odt, or the web formats like html.

```
                    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))
 @
\caption{Protest activity by income inequality \citep[from][]{norris2009data}.}\label{fig:ginipr
\end{center}
\end{figure}
```

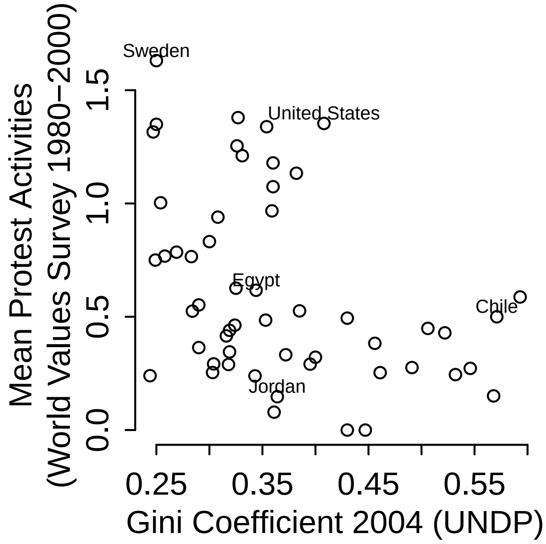

Figure 1: Protest activity by income inequality (from Norris, 2009).

By using `\label{fig:giniprot}`, I do not need to keep track of the figure number. Nor do I need a separate `fig1.R` file or `fig1.pdf` file. Tables and other numerical results are also possible to generate within "code chunks" in 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.[6]

[6] See [cite to Beck TPM] for the inspiration for this particular presentation of a linear model.

	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).

If your workflow does not involve LaTeXand R, you can still implement some of the principles here (and if you use OpenOffice or LyX and R you can use `odfWeave` do directly do literate data analysis in a WYSIWYG environment).[7]

Literate data analysis is not the same as Sweave, even if Sweave is a nice implementation.[8] For example, the person most often credited with these ideas used plain TeXas the "document formatting" language and Pascal as the programming language [cite]. See [X] for an example of using the Emacs org-mode as literate programming. And, one could imagine a Word or HTML file in which code chunks are inserted by commented out and run one at a time (or perhaps extracted by a clever script from the XML source file of a modern Microsoft Word Document) — or an HTML file in which hyperlinks are inserted to plain text files containing the relevant code chunks.

Principle 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.

In order to know that "that plot on page 10" is from a file called "fig1.pdf"

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

[8] See `http://cran.r-project.org/web/views/ReproducibleResearch.html` for many of the different approaches to literature programming for R.

you need to document it somehow. I can imagine a system with very disciplined use of a word processor perhaps using special auxiliary "listoffigures.txt" or "MANIFEST.txt" or "README" or "Makefile" files or something. Right now it is hard to beat Sweave or some plain text combination of LaTeX(or some other markup based document formatting system with support for citations, etc...) with some plain text based statistical programming environment (like R or Stata or Matlab, etc.) as a system for enhancing the collaboration of research groups doing quantitative work in political science. Something other than than Sweave will make our lives easier in the future. Then we'll change.[9]

2.3 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-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.)

Principle Comments and other documentation enable you to learn through teaching, code efficiently, and avoid duplicating your effort. Literate programming also enables you to avoid the kinds of errors which occur when you re-type numbers in to tables or graphic-producing systems or even into a document itself. Finally, and most importantly, a document that can be "run" to reproduce all of the results is a document that can more effectively spur discussion and learning and cumulation of research as people no longer need spend weeks attempting to reproduce research on which they desire to build.

`http://en.wikipedia.org/wiki/Literate_programming`

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

[9]For example, X uses the org-mode markup system in this issue.

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'`). 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.

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.

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.

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

Let me note that this principle was required when all our data analyses occurred in batch mode on VAX (and later Unix) machines. The fact that I need to articulate this principle at all arises because of the rise of interactive data analysis and graphical user interfaces: it is all too easy to use the mouse to load a data file into 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.[10] 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 useful to see what has changed, to 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, and to 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, risking 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). 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 changes by hand before copying over existing files. (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 (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 and we are lucky that the NCSA provides such a server for us. However, the price of such hosting may not be that much using one of the many webhosting services

[10] See X and Y in this issue for more discussion of what version control is.

out there or perhaps may be available for free at your university.[11]

Fancy version control systems are not required, however, to get many of the benefits of formal version control. 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. 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. Adobe Acrobat allows one to compare differences in pdf files. OpenOffice supports a "Compare Documents" option.

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. And, you will find yourself spending extra time reconciling changes made by different collaborators by hand that modern version control systems take care of quickly and easily.

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 (and which are described in more depth in X and Y) 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.

Principle 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, git, etc...) or with formal systems of file naming, file comparing, and communication or, even better, with both.

[11] For example, if you already pay to have a website, you may already have the right to run a subversion server there.

5 Minimize error by testing.

Now, back to that confidence interval in the famous article of 2018. The statisticians at the UN worry about your use of the bootstrap. The authors would like to evaluate their bootstrap procedure — and they did so originally in their memo comparing it to a profile likelihood based approach. 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 are writing custom code they worry about getting it right. Now, 9 years after the idea, they've had lots of time to appreciate problems arising from bugs and errors 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, reading application files for the admissions committee, staying home with a sick child, and undertaking the odd bit of your own current 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. 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!

Principle You cannot forsee all of the ways that your code could be used, but you can at least make sure it does what it is supposed to do in your particular case.

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 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 with the R package system Bowers (2011). The benefit of this last system 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. 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.

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.[12] 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 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).[13]

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

[12]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 publication.

[13]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.

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 reproducibility 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 of stuff 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, 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*

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.

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

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*

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*