Supplementary Materials: How (Not) To Reproduce – Practical Considerations to Improve Research Transparency in Political Science

July 19, 2021

1 Journal Research Transparency Requirements

.

Today, many journals in the political sciences request or require authors to submit reproduction material to data archives. The journals vary, however, in the degree to which these requests or requirements are enforced. In Table 1 we provide a summary of the research transparency policies currently in place at ten of the top quantitative research journals in political science. Note that we find that all of these journals have some type of research transparency policy that is easily accessible on each journal's website. However, only six of ten use some type of permanent and public archive for research transparency materials. Furthermore, while all ten have policies that require the provision of data and code prior to a paper's publication, as best as we can determine only two of these ten journals actually check to determine that the provided code and data successfully reproduce the paper's quantitative claims.

Table 1: Research Transparency Requirements in Top Political Science Journals

	Accessible policy*	Collective archive†	Data/Code required‡	Reproduced #
AJPS	✓	✓	✓	✓
APSR	✓	✓	✓	✗
BJPS	✓	✓	✓	✗
EJPR	✓	✗	✓	✗
JOP	✓	✓	✓	✗
PA	✓	✓	✓	✓
POQ	✓	✗	✓	✗
PRQ	✓	✗	✓	✗
PS	✓	✓	✓	✗
QJPS	✓	✗	✓	✗

* Journal research transparency policy is easily accessible on the journal website

† Journal archive contains all data submissions and is permanently publicly available

‡ Provision of data and code is required prior to publication

Code is run during review to reproduce and verify manuscript results

2 What Do Good Reproduction Materials Look Like?

Figure 1 shows a sample README covering all five sections needed for reproducibility. Figure 2 displays well-organized and commented code: The code is split into clearly labelled sections, no native functions are overwritten, and all data sources, `R` objects and output objects are succinctly named. The code saves Table 1 in `.tex` format to ease comparison with the corresponding manuscript version.

```
REPLICATION MATERIAL FOR SMITH 2019 "BRILLIANT METHODS ANALYSIS"

Author: Jane Smith, John Smith
Article: "Brilliant Methods Analysis"
Journal: Political Analysis

For any questions/inquiries, please email jane.smith@researchuniversity.com


#### Folders and files in the material ####

material.Rproj
/data
   /results.csv
/figures
   /Figure_1.pdf
   /Figure_2.pdf
   /Figure_A.1.pdf
README.txt
/scripts
   /analysis.R
   /figures.R
   /functions.R
   /master.R
   /simulations.R
   /tables.R
/tables
   /Table_1.tex
   /Table_A.1.tex
   /Table_A.2.tex

/data contains all data source files. /figures contains all code-produced figures
in .pdf form. /scripts contains the code files to be run. /tables contains all
code-produced tables in .tex form.


#### Hardware specifications ####

MacBook Pro Early 2015, 2.7 GHz Intel Core i5, 16 GB RAM


### Software used ####

macOS High Sierra, 10.13.6
R 3.6 (2019-04-26) "Planting of a Tree"
R packages: abind (1.4-5), caret (6.0-76), dplyr (0.7.4), foreach (1.4.3),
ggplot2 (2.2.1), here (0.1), lme4 (1.1-13), magic (1.5-6), xtable (1.8.4)


#### Code files ####

* master.R (sources all other code files)
* functions.R (sets up needed functions)
* simulations.R (runs simulations)
* analysis.R (runs analyses)
* figures.R (creates and saves figures)
* tables.R (creates and saves tables)

Users only need to run master.R in order to reproduce the results of the paper.


#### Approximate running times ####

* functions.R: ~30 seconds
* simulations.R: ~10 hours
* analysis.R: ~1 hour
* figures.R: ~2 minutes
* tables.R ~1 minute
```

Figure 1: Sample README

```
# load libraries and set the working directory with 'here'
library(here)
library(xtable)
library(ggplot2)

# load data source
results <- read.csv(here('data','results.csv'))

# data analysis
modA <- lm(PBC ~ Resp, data = results)

# save figure 1
pdf(filename = here('figures', 'Figure_1.pdf'))
ggplot(data = results, aes(x = label, y = percent, fill = origin)) +
  geom_bar(stat = "identity", position = position_dodge())
dev.off()

# save table 1
print(xtable(modA, type = "latex"), file = here('tables', 'Table_1.tex'))
```

Figure 2: Sample R code

Table 2 shows the resulting LaTeXoutput of Table 1 next to its corresponding manuscript content. All numbers and names are identical and easily matched. Figure 3 shows the folder content listing and the R code utilizing a master file. These samples are of course rudimentary, but they illustrate what needs to be present for successful reproduction of political science data analysis.

```
\begin{table}[ht]
\centering
\begin{tabular}{rrrr}
  \hline
 & Estimate & Std. Error & t-value \\
  \hline
(Intercept) & 1.1205 & 1.2752 & 0.88 \\
Resp & -10.2483 & 3.3437 & -3.06\\
   \hline
\end{tabular}
\end{table}
```

	Estimate	Std. Error	t-value
(Intercept)	1.1205	1.2752	0.88
Resp	-10.2483	3.3437	-3.06

Table 1: Regression Results

Table 2: Sample .tex table produced with R (left), manuscript table (right)

```r
library(here)
source(here('scripts', 'functions.R'))
source(here('scripts', 'simulations.R'))
source(here('scripts', 'analysis.R'))
source(here('scripts', 'figures.R'))
source(here('scripts', 'tables.R'))
```

Figure 3: Sample R master file sourcing all script files

3 The Future of Replication? Docker Containers

Most political science journals currently employ data repositories in the style of Dataverse, where researchers deposit their data and code for others. In order to reproduce researchers' original results, users need to download the material onto their local machines and run the provided code files there. This can lead to all manners of problems, as it is highly unlikely that a user's local computational environment matches the original researchers'. For example, even a slight change of `R` versions from `3.5` to `3.6` could affect results in large-scale simulations, since the RNG method in the `sample()` function changed (Smith, 2019). Similarly, changing the `R` package `dplyr` from version `0.4.0` to `0.5.0` could lead to differing results because of the introduction of functionality breaking changes (Przytula, 2017). As dangerous as such seemingly small changes are, one could argue that they are easily fixed by installing the correct `R` version and using package management packages such as `packrat` (Marwick et al., 2018; Ushey et al., 2018). What do users do, however, if problems can be traced back to differing operating systems? Something as basic as string sorting leads to differing results when performed on Linux and Windows systems (Przytula, 2017). This is not to mention under-the-hood compilers such as `CLANG`, which are notoriously difficult to set up across operating systems (CLANG, 2020).

It is not researchers' responsibility to provide material that is perfectly reproducible across various operating systems, different versions within operating systems, or different versions of software packages. This would be an impossible task, as it would entail predicting and accounting for all potential system variations between operating systems and versions, of which we have merely mentioned a few in our paper. It is generally expected in replication, though, for material to return identical results when reproduced under the same conditions that the original research took place. In theory, any researcher who sets up the same software and computational environment as the original researchers can then reproduce the analysis with identical results. In practice, however, setting up the same software and computational environment is next to impossible, as we have outlined. From a reproduction, a journal, and an author perspective, the data repository style is thus unfortunate as the almost unavoidable mismatch between the original researchers' and the user's environment often proves to be very time-consuming and obtrusive.

With the continuing evolution of computing power and technology, one possible solution to this dilemma comes from data science in the form of Docker containers. A Docker container is an independent image that packages up operating system and software all in one. It is a virtual, self-contained computer that can be accessed through the browser (Merkel, 2014; Hung et al., 2016). Users can install software, upload data, and run their code in a remote container from the web. This eliminates the need to download and install software on local computers and ensures full reproducibility of code and results across all platforms and software (Gallagher, 2018; Boettiger, 2015). Docker containers can thus greatly increase efficiency and effectiveness in the reproduction of quantitative results in social science (Liu and Salganik, 2019; Anderson et al., 2008; Clyburne-Sherin et al., 2019). To our knowledge, however, no major social science journal currently uses Docker containers to conduct replications. *Political Analysis* is set to become the first to do so. It is currently setting up a reproduction structure with the provider *Code Ocean* to make Docker containers its main vehicle for data reproduction later this year. This will ensure full reproducibility, remove vast computational hindrances, and speed up replication and publication processes. It will also greatly ease the replication of material requiring a large amount of computing power, as Docker containers typically provide ten times the power of standard laptops. In our view, the use of Docker containers thus represents a win-win for authors and journals alike. We thus hope more social science journals will utilize the potential of this technology in the years to come.

References

Anderson, R. G., W. H. Greene, B. D. McCullough, and H. D. Vinod (2008). The Role of Data/Code Archives in the Future of Economic Research. *Journal of Economic Methodology 15*, 99–119.

Boettiger, C. (2015). An Introduction to Docker for Reproducible Research. *SIGOPS Operating Systems Review 49*, 71–79.

CLANG (2020). Cross-Compilation using `CLANG`. https://clang.llvm.org/docs/CrossCompilation.html.

Clyburne-Sherin, A., X. Fei, and S. A. Green (2019). Computational Reproducibility via Containers in Social Psychology. *Meta-Psychology 3*, 1–9.

Gallagher, M. J. (2018). Cloud Computing in Social Sciences and Humanities. In S.-H. Chen (Ed.), *Big Data in Computational Social Science and Humanities*, pp. 289–296. New York, NY: Springer.

Hung, L.-H., D. Kristiyanto, S. B. Lee, and K. Y. Yeung (2016). `GUIdock`: Using Docker Containers with a Common Graphics User Interface to Address the Reproducibility of Research. *PLoS ONE 11*, 1–14.

Liu, D. and M. Salganik (2019). Successes and Struggles with Computational Reproducibility: Lessons from the Fragile Families Challenge. *Socius: Sociological Research for a Dynamic World 5*, 1–21.

Marwick, B., C. Boettiger, and L. Mullen (2018). Packaging Data Analytical Work Reproducibly Using `R` (and Friends). *American Statistician 72*, 80–88.

Merkel, D. (2014). Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux Journal 239*, 1–5.

Przytula, P. (2017). Reproducible Research - When Your Results Can't Be Reproduced? https://www.r-bloggers.com/reproducible-research-when-your-results-cant-be-reproduced/.

Smith, D. (2019). What's New in `R` 3.6.0. https://blog.revolutionanalytics.com/2019/05/whats-new-in-r-360.html.

Ushey, K., J. McPherson, J. Cheng, A. Atkins, and J. J. Allaire (2018). `packrat`: A Dependency Management System for Projects and their `R` Package Dependencies. https://rdrr.io/cran/packrat/.