HHS Public Access

Author manuscript

Socius. Author manuscript; available in PMC 2023 June 12.

Published in final edited form as:

Socius. 2019 ; 5: . doi:10.1177/2378023119849803.

Successes and Struggles with Computational Reproducibility: Lessons from the Fragile Families Challenge

David M. Liu[1], **Matthew J. Salganik**[1]

[1]Princeton University, Princeton, NJ, USA

Abstract

Reproducibility is fundamental to science, and an important component of reproducibility is computational reproducibility: the ability of a researcher to recreate the results of a published study using the original author's raw data and code. Although most people agree that computational reproducibility is important, it is still difficult to achieve in practice. In this article, the authors describe their approach to enabling computational reproducibility for the 12 articles in this special issue of *Socius* about the Fragile Families Challenge. The approach draws on two tools commonly used by professional software engineers but not widely used by academic researchers: software containers (e.g., Docker) and cloud computing (e.g., Amazon Web Services). These tools made it possible to standardize the computing environment around each submission, which will ease computational reproducibility both today and in the future. Drawing on their successes and struggles, the authors conclude with recommendations to researchers and journals.

Keywords

computational reproducibility; computational social science; cloud computing; software containers; open and reproducible research

There is broad agreement that reproducibility is generally important for establishing a scientific claim. In this article, we focus on a very narrow form of reproducibility: a researcher's ability to recreate the results in a published article using the raw data and code of the original author. Following Stodden (2015), we call this "computational reproducibility," although this same concept has also been called "verifiability" (Freese and Peterson 2017) and "repeatability" (Collberg and Proebsting 2016; Easterbrook 2014).[1]

Although computational reproducibility is widely considered to be a desirable goal, it is a standard that a lot of scientific research does not currently meet. There are two main

Corresponding Author: David M. Liu, 145 Wallace Hall, Princeton, NJ 08544, USA. dml3@alumni.princeton.edu.

Supplemental Material

Supplemental material for this article is available online.

[1]For more on the somewhat inconsistent terminology in this area, see Goodman et al. (2016), Patil et al. (2016), Clemens (2017), Plesser (2018), Barba (2018), and Benureau and Rougier (2018).

factors that contribute to this shortcoming. First, the data and code used to create results in published articles are often not available to other researchers, despite requirements by some journals, professional organizations, and funding agencies. Second, even if the data and code are available, independent researchers are often not able to reproduce the results. These two patterns have been empirically established in the social sciences (Chang and Li 2018; Dewald, Thursby, and Anderson 1986; Gertler, Galiani, and Romero 2018; Hardwicke et al. 2018; In'nami and Koizumi 2010; McCullough, McGeary, and Harrison 2006; Stockemer, Koehler, and Lenz 2018; Vanpaemel et al. 2015; Wicherts, Bakker, and Molenaar 2006; Wicherts et al. 2011; Wicherts and Crompvoets 2017; Wood, Müller, and Brown 2018), as well as other scientific fields (Alsheikh-Ali et al. 2011; Andrew et al. 2015; Campbell, Micheli-Campbell, and Udyawer 2019; Collberg and Proebsting 2016; Gilbert et al. 2012; Hardwicke and Ioannidis 2018; Ioannidis et al. 2009; Konkol, Kray, and Pfeiffer 2019; Naudet et al. 2018; Ostermann and Granell 2017; Rowhani-Farid and Barnett 2016; Savage and Vickers 2009; Stodden, Seiler, and Ma 2018; Vandewalle, Kovacevic, and Vetterli 2009; Vines et al. 2014). Although the magnitude of the problem differs by field and across time, the limited availability of replication materials and the limited usefulness of the materials that are available are consistent patterns. In other words, a lack of computational reproducibility is itself a reproducible finding.

There are many admirable attempts to improve the computational reproducibility. One of these efforts is the Transparency and Openness Promotion (TOP) Guidelines (Nosek et al. 2015). The TOP Guidelines provide a framework journals can use to assess and improve their efforts to make scientific communication more open. Despite being introduced quite recently, more than 1,000 journals have already adopted the TOP Guidelines in some form (Center for Open Science n.d.).

The TOP Guidelines consist of eight standards, including one called "Data, Analytic Methods (Code), and Research Materials Transparency." When focused on computational reproducibility, this standard has three levels:

> Level 1 ("available upon request"): replication materials available upon request from authors
>
> Level 2 ("access before publication"): replication materials submitted to a trusted archive before publication
>
> Level 3 ("verification before publication"): replication materials verified by journal and then submitted to a trusted archive before publication

For this special issue, we attempted to meet the TOP Level 3 standard ("verification before publication"). Furthermore, in addition to complying with the standard as written,[2] we attempted to give replication materials a more central role in the process of scholarly

[2]The introductory paragraph for the "Data, Analytic Methods (Code), and Research Materials Transparency" standard is as follows: "The policy of the [Journal of Research] is to publish papers only if the data, methods used in the analysis, and materials used to conduct the research are clearly and precisely documented and are maximally available to any researcher for purposes of reproducing the results or replicating the procedure. All materials supporting the claims made by the author must be made available to the journal prior to publication. The journal, or an entity acting on behalf of the journal, will verify that the findings are replicable using the authors data and methods of analysis. Failure to replicate at this stage may result in the paper not being published." See https://osf.io/9f6gx/ for the most recent version.

communication. Our thinking was heavily influenced by what could be called Donoho's dictum[3]:

> An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.
>
> (Buckheit and Donoho 1995)

As we attempted to give replication materials a more central role in the process of scholarly communication, we made three interrelated changes to standard practice. First, inspired by insights from other computational fields, we took a broader view of what should be included in replication materials. In particular, rather than focusing on just data and code, we expanded our focus to include the computing environment (Stodden et al. 2016; Tatman, VanderPlas, and Dane 2018). Second, rather than merely verifying the results in a manuscript, we focused our efforts toward making the replication materials as useful as possible to future scholars. Finally, rather than assessing computational reproducibility after completing peer review of a manuscript, we made computational reproducibility a part of the review process.

We found that computational reproducibility was difficult to achieve in practice. Only 2 of the 14 manuscripts that were submitted to the special issue were computationally reproducible initially, even though we had access to code provided by the authors. After an intensive revision process, 7 of the 12 accepted manuscripts eventually became computationally reproducible. These aggregate results mask important heterogeneity. Some submissions to the special issue used computational approaches that are relatively common in sociology and the other social sciences today, for example, Ahearn and Brand (2019) and McKay (2019). For these submissions, computational reproducibility was relatively easy. Some other submissions, however, drew on more computationally complex approaches from machine learning, for example, Rigobon et al. (2019) and Davidson (2019). These submissions were substantially more difficult to reproduce.

We think this heterogeneity has two important lessons. First, our results suggest that relatively modest efforts could yield large increases in computational reproducibility for the methods that are typically used today. The second important lesson is about the future. We expect that sociologists and other social scientists will increasingly use larger data sets and more computationally intensive methods, such as the methods used in this special issue. Our experience suggests that as research becomes more computationally intensive, the problems with computational reproducibility will likely get worse unless researchers make changes to the process of scientific publishing.

The remainder of this article is organized as follows. In the next section we provide more background on the Fragile Families Challenge and computational reproducibility. Then we describe our process for ensuring computational reproducibility for this special issue. Next

[3]Although this quotation is attributed to David Donoho, Donoho was himself influenced by Jon Claerbout and colleagues (Claerbout and Karrenbach 1992). See Barba (2018) for more historical background.

we offer recommendations for authors wanting to make their work more computationally reproducible, and we offer recommendations for journals wanting to promote computational reproducibility. We also provide supporting online materials that include a summary of the computational reproducibility status of each article in the special issue; the reproducibility memo that we sent to authors during the peer-review process; and instructions for using the Docker images we created to accompany the articles in this special issue.

Background

Fragile Families Challenge

The Fragile Families Challenge is a scientific mass collaboration designed to yield insights that can improve the lives of disadvantaged children in the United States. The Challenge is described in more detail in Salganik et al. (2019), but we describe it here briefly with a focus on issues that are particularly relevant to computational reproducibility. The Fragile Families Challenge builds on the Fragile Families and Child Wellbeing Study, which collects data about a probability sample of births from 1998 to 2000 in large U.S. cities, with an oversample of nonmarital births (Reichman et al. 2001). During the course of the Fragile Families and Child Wellbeing Study, information has been collected from mothers, fathers, and children, as well as other people in the children's lives (e.g., teachers). The Challenge set out the goal of using data collected during the child's first 9 years to predict six outcomes when the child was 15 years old (Figure 1).[4]

To this end, the Challenge used a research design from machine learning called the common task method (Donoho 2017). The common task method involves three main elements: a common target for predictive modeling, a common error metric (in our case, mean squared error), and a common data set with both training data available to participants and held-out data used for evaluation. Following this structure, we split the year 15 data for these six outcomes into three groups: (1) a training set that we provided to participants, (2) a leaderboard set participants could access during the Challenge, and (3) a holdout set participants could not access until the first stage of the Challenge was complete (Hardt and Blum 2015) (Figure 1). Participants in the Challenge received the training set and a specially constructed background data file that contained information about the family from birth to age 9; it included 4,242 families and 12,942 variables about each family.[5]

Computational Reproducibility

The benefits of computational reproducibility—and increased access to data and code, more generally—have already been articulated many times by researchers in many different fields: archaeology (Marwick 2017), bioinformatics (Mangul et al. 2018), cell biology (Grüning et al. 2018), computational fluid mechanics (Mesnard and Barba 2017), computer systems research (Collberg and Proebsting 2016), economics (Anderson et al. 2008; Koenker and Zeileis 2009; Orozco et al. 2018), epidemiology (Coughlin 2017; Peng, Dominici, and

[4]Participants in the Challenge attempted to predict six outcomes: grade point average (GPA) of the child, grit of the child, material hardship of the household, whether the household was evicted from their home, whether the primary caregiver participated in job training, and whether the primary caregiver lost his or her job. The choice of these outcomes was driven by our scientific goals and ethical considerations; each outcome is described in more detail elsewhere (Lundberg et al. 2019; Salganik et al. 2019).
[5]For more on the construction of the Fragile Families Challenge background file, see Lundberg et al. (2019).

Zeger 2006; Shepherd et al. 2017), geosciences (Claerbout and Karrenbach 1992; Gil et al. 2016; Konkol et al. 2019), high-energy physics (Chen et al. 2018), hydrology (Hutton et al. 2016), mathematical and computational biology (Schnell 2018), machine learning (Hutson 2018; Tatman et al. 2018), neuroscience (Crook, Davison, and Plesser 2013; Eglen et al. 2017; Manninen et al. 2017; Mikowski, Hensel, and Hohol 2018), political science (Alvarez, Key, and Nez 2018; King 1995; Lupia and Elman 2014), psychology (Clyburne-Sherin and Green 2018), signal processing (Vandewalle et al. 2009), sociology (Freese 2007), and statistics (Donoho 2010; Gentleman and Lang 2007; Stodden 2015). We find these arguments convincing. So, for the purposes of this article, we simply assume that computational reproducibility is a desirable goal.[6] We acknowledge, however, that this is a relatively low standard; it does not guarantee correctness of the code or the scientific validity of the claims (Leek and Peng 2015).[7]

There are many wonderful efforts under way to increase computational reproducibility, and our approach shares many features with these other efforts. However, there are three things that partially differentiate our approach from other approaches in the social sciences. First, we were not motivated by the "reproducibility crisis" or issues related to "questionable scientific practices." The structure of the Challenge—with explicit holdout data and a clear prespecified evaluation metric—makes concerns about questionable scientific practices less relevant. Rather, our approach was focused on increasing research impact through reuse.[8]

The second way our approach differs from some other approaches in the social sciences is that it focuses on the computing environment, in addition to data and code. Currently, a significant subset of published social science research is based on relatively small data sets using methods that can be deployed in a fairly basic computing environment. In other words, it is not uncommon for many social scientists to do their computing on a laptop computer. However, some social scientists are beginning to use much larger data sources and more resource-intensive computational methods from machine learning. These are potentially exciting developments, but as computational complexity increases, it becomes increasingly difficult to reproduce results using data and code running in a different computing environment (Boettiger 2015; Boisvert 2016; Tatman et al. 2018). Instead, for computationally complex work, reproducibility often requires access to the same

[6]This is not to say that access to data and code is always desirable (Levy and Johns 2016). There are of course exceptions, such as data that contains personally identifiable information or code that contains proprietary business information. However, these exceptions do not call into question the general idea that for scientific research, computational reproducibility is a desirable goal.

[7]Although computational reproducibility does not prevent errors, it does make it easier to diagnose and understand these errors if they are later discovered (see, e.g., Heuberger 2019).

[8]To illustrate the value of reuse, we briefly sketch three research projects that are enabled by the computational reproducibility of the articles in the special issue. First, one could imagine trying to create a super-model that combines the best pieces of each of the individual papers. For example, imagine combining the imputation approaches of Goode et al. (2019), the approach to missing data of Carnegie and Wu (2019), the approach to adding expert knowledge of Filippova et al. (2019), and the machine-learning approach in Rigobon et al. (2019). Might this super-model, or some other combination of existing strategies, perform better than the best strategy used during the Challenge? Second, rather than combining modeling components, future researchers could explore different ways to ensemble predictions. Fragile Families Challenge Team (2019) shows that one such approach to combining predictions—stacking (Breiman 1996; Wolpert 1992)—did not lead to large improvements. Might other approaches, particularly approaches that require running code, lead to better ensembling results? Third and finally, the predictions submitted in the Challenge did not include any estimate of uncertainty. However, as predictive models get deployed for high-stakes social decisions (Rudin 2018), it would be desirable for them to also produce estimates of uncertainty. Using existing ideas from the literature (e.g., King, Tomz, and Wittenberg 2000), would it be possible to produce confidence intervals from these models? What might there coverage properties be? We emphasize that these are just three possibilities, and we expect that many other ideas might occur to other researchers.

computing environment, both hardware and software. Fortunately, software engineers have already developed tools and techniques that make it possible to standardize a computing environment, and for our Challenge reproducibility work, we borrowed liberally from these practices. In particular, for our computational reproducibility work in the Fragile Families Challenge we used software containers (e.g., Docker) to standardize the software environment and cloud computing (e.g., Amazon Web Services) to standardize the hardware environment.

Software containers and cloud computing are not yet common for computational reproducibility in the social sciences, so we will briefly review them.[9] Software containers, such as Docker, are designed to ensure that software that runs on one computer can run in exactly the same way on another computer. Containers ensure portability through encapsulation. Roughly, software containers create an appropriately configured minicomputing environment within a larger computing environment and then make it easy to move this minicomputing environment from one computer to another, independent of the operating systems. This minicomputing environment can include all the supporting software the research code depends on, and it can keep these dependencies separate from all the other code that runs on the host computer. For example, the results in Stanescu et al. (2019) were created with code that requires version 0.7.4 of the R package dplyr. Without a software container, anyone wishing to reproduce the results in the article would need to install version 0.7.4 of dplyr, and all the packages on which it depends, on his or her computer. This installation process is easy with just one relatively new, high-quality package. But when code requires many packages, particularly old and complex packages, installing all the necessary dependencies can be difficult, frustrating, and error prone. These difficulties become even more extreme if the researcher attempting to reproduce these results is also working on another project that requires a different version of the same package. The ugly process of installing dependencies and managing competing dependencies across projects is sometimes referred to as “dependency hell.” Containers prevent dependency hell. They do this by making an appropriately configured minicomputing environment (e.g., with the necessary version of dplyr) that can easily be installed on a new computer and hidden from the other software running on the new machine.

There are two main things a researcher can do with containers: create them and work inside them. To create a container, a researcher starts by making a list of the all the things to be included in the computing environment. With Docker, this configuration file is called a “Dockerfile.” Figure 2 shows the Dockerfile for Stanescu et al. (2019). Then the researcher executes the Dockerfile, and Docker creates a bundle of all the necessary software. With Docker, this bundle of software is called a “Docker image.” The Dockerfile for Stanescu et al. (2019) is relatively simple, but some were much more difficult to create. For example, the Dockerfile for Davidson (2019) required 87 packages.[10]

[9]For more about software containers and virtual machines (a closely related concept), we recommend Merkel (2014), Boettiger (2015), and Clyburne-Sherin and Green (2018). For more about cloud computing in the context of scientific research, we recommend Dudley and Butte (2010) and Howe (2012).

[10]The 87 required packages and version numbers for Davidson (2019) are as follows: absl-py v0.6.1, appnope v0.1.0, astor v0.7.1, backcall v0.1.0, bleach v3.0.2, cloudpickle v0.6.1, cvxpy v1.0.10, cycler v0.10.0, dask v0.20.0, decorator v4.3.0, defusedxml v0.5.0, dill v0.2.8.2, ecos v2.0.5, entrypoints v0.2.3, fancyimpute v0.4.0, fastcache v1.0.2, future v0.17.1, gast v0.2.0, grpcio v1.16.0, h5py v2.8.0, ipykernel v5.1.0, ipython v7.1.1, ipython-genutils v0.2.0, ipywidgets v7.4.2, jedi v0.13.1, Jinja2 v2.10, jsonschema v2.6.0,

Once a Docker image is created and published online, a new researcher can install it to his or her computer and use the image to create an appropriately configured container. The new researcher can then go inside of the container and work in a computing environment that was created to support the replication of the original scientific results. Furthermore, this computing environment is separate from the larger computing environment on the new researcher's computer. A researcher downloading and using a Docker image does not need to understand exactly how it was created, which means that it is much easier to use an image than create one. Although Docker itself may be replaced in the future by alternative tools, we believe that the idea of software containers will likely be important for computational reproducibility for the foreseeable future.

Although software containers standardize the software environment, cloud computing can be used to standardize the hardware environment. Roughly, a cloud computing environment is a huge collection of computing resources, which can be subdivided so that different users can access different amounts of computing resources. Creating and managing cloud computing environments is extremely complex, but from the perspective of a user, they are relatively simple. At the time of the Fragile Families Challenge, large companies such as Amazon, Microsoft, and Google offered cloud computing environments that enabled people to rent computing resources by the hour. For example, at the time of the Challenge, Amazon Web Services, Amazon's cloud computing environment, allowed people to rent computational resources with names like "m5.large" and "t3.medium." These different environments came with different types of processors, random-access memory, and other computing resources. We chose to perform our reproducibility work on Amazon Web Services because we thought this would increase the chance that other researchers will be able to access a very similar hardware environment for the foreseeable future. Having access to a similar hardware environment is helpful because the performance and output of complex software can sometimes depend on the hardware that is used to execute it.[11] Future researches, however, will not be required to use Amazon Web Services or any other cloud computing provider to use the Docker images that we created. Together, software containers and cloud computing make it easier for researchers to have access to a very similar computing environment to the one we used when verifying computational reproducibility, both today and for years to come.

The third and final way our approach differed from common approaches in social science is timing. To the best of our knowledge, most work on computational reproducibility review in social science journals happens after a manuscript has already been conditionally accepted.

jupyter v1.0.0, jupyter-client v5.2.3, jupyter-console v6.0.0, jupyter-core v4.4.0, Keras v2.2.4, Keras-Applications v1.0.6, Keras-Preprocessing v1.0.5, kiwisolver v1.0.1, knnimpute v0.1.0, lime v0.1.1.32, Markdown v3.0.1, MarkupSafe v1.1.0, matplotlib v3.0.1, mistune v0.8.4, multiprocess v0.70.6.1, nbconvert v5.4.0, nbformat v4.4.0, networkx v2.2, notebook v5.7.0, np-utils v0.5.5.2, osqp v0.4.1, pandas v0.23.4, pandocfilters v1.4.2, parso v0.3.1, patsy v0.5.1, pexpect v4.6.0, pickleshare v0.7.5, Pillow v5.3.0, prometheus-client v0.4.2, prompt-toolkit v2.0.7, protobuf v3.6.1, ptyprocess v0.6.0, Pygments v2.2.0, pyparsing v2.3.0, python-dateutil v2.7.5, pytz v2018.7, PyWavelets v1.0.1, PyYAML v3.13, pyzmq v17.1.2, qtconsole v4.4.2, scikit-image v0.14.1, scikit-learn v0.20.0, scipy v1.1.0, scs v2.0.2, Send2Trash v1.5.0, six v1.11.0, sklearn v0.0, statsmodels v0.9.0, tensorboard v1.12.0, tensorflow v1.12.0, termcolor v1.1.0, terminado v0.8.1, testpath v0.4.2, toolz v0.9.0, tornado v5.1.1, traitlets v4.3.2, wcwidth v0.1.7, webencodings v0.5.1, Werkzeug v0.14.1, and widgetsnbextension v3.4.2.

[11]For example, the runtime typically depends on the amount of random-access memory, the number of processors, and the type of processor (e.g., central processing unit vs. graphics processing unit). For more about how the underlying computing hardware can affect the results of scientific computing, see Diethelm (2012).

Our approach, however, involved computational reproducibility review simultaneous to manuscript review. We think this simultaneous review appropriately emphasizes that the published article and code work together to communicate a scientific finding.

Our Process for the Special Issue

Although we had some experience with computational reproducibility in our roles as researchers, we had no experience with computational reproducibility from the perspective of a journal editor. This inexperience meant that we ended up using a process that we now realize was not ideal in several ways. Before offering our recommendations for the future, however, we will describe the process that we used for the special issue.

In our call for papers for the special issue, we stated that manuscripts needed to be accompanied by open-source code that would allow another researcher to reproduce all the figures and tables in the article.[12] However, we did not provide specific guidelines about the form that the code should take. Furthermore, our call for papers had a specific deadline by which manuscripts needed to be submitted (October 16, 2017).

Fourteen manuscript were submitted. While these manuscripts were undergoing peer review, we attempted to reproduce the results in each one. Having computational reproducibility reviews happening at the same time as manuscript peer reviews was consistent with our goal of considering replication materials to be a central part of scholarly communication.

Our process of computational reproducibility review generally involved five steps: (1) create a Docker image with the necessary software and packages, (2) move the authors' code into the Docker image and adjust the file paths to match those of the container, (3) add the Challenge data file to the container,[13] (4) run the authors' code inside the container in our cloud computing environment,[14] and (5) compare the results created within our computing environment with the results shown in the manuscript. We considered a manuscript to be computationally reproducible if we could run the author's code in our computing environment and regenerate (1) all the prediction.csv files (these are the files participants created to record their predictions of all six outcomes for all 4,242 families) within our error tolerance,[15] (2) all the figures that were generated by the author's code, and (3) all the tables that were generated by the author's code.

For some manuscripts, this process went smoothly, but for many others it was quite painful. It was often extremely difficult to even run the authors' code. In some cases, authors sent

[12]The full call for papers is included in Salganik et al. (2019), and here is the part most related to computational reproducibility: "What are the requirements for the open source code? The code must take the Fragile Families Challenge data files as an input and produce (1) all the figures and tables in your manuscript and supporting online materials and (2) your final predictions. The code can be written in any language (e.g., R, Stata, Python). The code should be released under the MIT license, but we will consider other permissive licenses in special situations."

[13]To help protect the privacy of participants in the Fragile Families and Child Wellbeing Study, authors were not supposed to include the Challenge data file in their computational reproducibility materials. For more about the data access process during the Challenge, see Lundberg et al. (2019).

[14]As our computational reproducibility review improved, we would run the same code twice and compare the results. If these results did not match, we did not even attempt to compare with the manuscript.

[15]We considered two numbers the same if they differed by less than 10^{-10}. We acknowledge that this is an arbitrary standard, but this threshold was effective for us because most differences were either much larger or smaller. In other situations a different threshold might be reasonable, and we hope that community standards develop for when two numbers should be considered the same.

a lot of unnecessary code, and in other cases, authors did not send all the code that was necessary. Furthermore, the code that was provided was often disorganized and poorly documented. Even if we were able to get the code to run, it was also often difficult to compare the results produced by the code with the results in the manuscript.

During this initial computational reproducibility review, we were able to approximately reproduce the results in 2 of the 14 manuscripts. We had this low rate of success despite being in frequent e-mail contact with some authors and often devoting many hours trying to reproduce the results of each manuscript.[16]

Initially, we had hoped to provide each author with individual feedback on his or her project's computational reproducibility, much as each author receives individual feedback on his or her manuscript. Unfortunately, this proved to be too time consuming for us given our desire, and that of the editors of *Socius*, to provide rapid feedback to the authors. Ultimately, each author received individual feedback on their manuscript from editors of the special issue and about three anonymous reviewers, and each author received collective feedback on how computational reproducibility could be improved. This collective feedback —reproduced in the supporting online materials of this article—was developed inductively on the basis of our struggles with the initial code submissions, as well as on our review of existing articles about computational reproducibility.

All 14 manuscripts received a decision of "revise and resubmit," and authors were instructed to improve both their manuscripts and code on the basis of the feedback that was provided. Fortunately, we saw dramatic improvements in code structure and documentation between the initial submissions and the revised versions. We do not know, however, how much of this improvement should be attributed to the guidelines we provided to authors and how much should be attributed to other factors, such as the authors realizing that we were treating computational reproducibility as a part of the review process.[17]

After the authors received this first round of feedback, the synchronization of the manuscript review and the computational reproducibility review broke down. In some cases, our computational reproducibility review would proceeded while the manuscript was under review, and in other cases it would proceeded while the authors were working on their revisions. This second stage of the computational reproducibility review was particularly time consuming because it required us to precisely identify, and attempt to eliminate, the discrepancies between the results in the manuscript and the computationally reproducible results. This process was occasionally frustrating for us, and we are sure that it was occasionally frustrating for the authors.

[16]This low success rate roughly matches the rate for the *Quarterly Journal of Political Science*. From September 2012 to November 2015, the replication materials for 20 of the 24 accepted empirical articles were found to require some kind of modification. Furthermore, 14 of 24 accepted empirical articles were found to have discrepancies between the results generated by authors' code and those in their articles (Eubank 2016). However, the journal *International Interactions* reports higher success rates. Between 2014 and 2016, they successfully replicated the results in 22 of 33 articles without needing to contact the authors (Colaresi 2016).

[17]One way to assess the impact of these factors and others would be some kind of randomized controlled trial, where different authors are randomly assigned to different review processes. See, for example, Shah et al. (2018).

Eventually, the editors of *Socius* informed us that we needed to bring the review process for the special issue to an end. We gave all authors with outstanding manuscripts a deadline of November 13, 2018, to submit their final manuscripts and final code. In the end, we completely reproduced the results of 7 manuscripts; for the other manuscripts, we ran out of time (although for many of these, we did make substantial progress). For each of the 12 manuscripts, the code will be released along with an accompanying Docker image with the necessary dependencies. Thus, all articles in the special issue reached TOP Level 2 ("access before publication"), and most reached TOP Level 3 ("verification before publication"). The supporting online materials summarize the computational reproducibility of each article in the special issue.

The most important lesson that we take from our experience is that ensuring computational reproducibility was harder than we had anticipated. There was not one specific thing that caused huge difficulties, but many of the difficulties came from a combination of three interrelated factors: many moving parts, long feedback cycles, and communication difficulty. To illustrate these factors, we will briefly summarize our attempts to replicate the results in Altschul (2019), which was neither particularly easy nor particularly difficult.

At first glance, the code for Altschul (2019) appeared straightforward to reproduce because it is a single R file. Yet this code required 13 R packages to be installed, all of which had to be built from source, a more error-prone and time-consuming form of package installation, because our cloud computing environment used Linux.[18] Furthermore, these packages in turn depended on many other packages which also had to be installed. Although each package can be built from source with a high probability of success p, the entire process will fail if just one package fails, and the probability that at least one package fails is $1 - p^n$, where n is the number of packages. So, as the number of packages increases, the probability of at least one failure increases exponentially. Beyond just installing packages, many of the submissions had a large number of components and a failure of just one component caused a failure of the entire process. This is what we call the "many moving parts problem."

The difficulty of the many moving parts problem was exacerbated by a long delay between execution and feedback. The process of debugging computational reproducibility issues often involves attempting to isolate the problem and then use trial and error to solve the problem. The longer we had to wait to receive feedback, the longer it would take for us to complete many iterations of this isolation and trial-and-error process. For example, for Altschul (2019) it took about one hour of computing time to build the Docker image and many attempts to isolate and eliminate discrepancies required building a new Docker image and then rerunning the code, which would often have to run for several hours. These long computing times made the process of debugging very slow.

Finally, our slow debugging often had to go through many iterations because of communication difficulties. In the 13 months between initial submission and final publication, we exchanged a total of 28 e-mails with Altschul. These exchanges spanned long periods of time not only because we were also communicating with many other authors

[18]Precompiled R package binaries are available only for computers using Windows and MacOS.

at the same time but also because each reply could require hours of additional work. Although not true for Altschul (2019), we also noticed that in some cases, as time went on, it became increasingly difficult for the authors themselves to respond to us quickly because it had been such a long time since they had worked with their own code.

We describe these difficulties replicating Altschul (2019) not because it was a particularly hard case but to illustrate the kind of difficulties that we faced as a result of the many moving parts problem, long feedback times, and communication difficulties. Undoubtedly many of these difficulties were caused by our inexperience with computational reproducibility, and later in the article we offer suggestions for handling these issues more efficiently.

Stepping back, our experience with the Challenge was unusual in a number of ways, and other editors may have different experiences with computational reproducibility in other settings. In fact, we can imagine some reasons to expect that computational reproducibility will be easier in other settings and some reasons to expect the opposite. We think there are three main reasons that ensuring computational reproducibility should be easier in other settings. In sociology and the social sciences more generally, most researchers use simpler data processing and statistical modeling pipelines than were used in the Challenge. We had the easiest time reproducing the results in manuscripts that use methods more common in the social sciences, such as Ahearn and Brand (2019) and McKay (2019). We had more difficulty, however, with manuscripts that included more machine-learning techniques, such as Davidson (2019), Rigobon et al. (2019), and Carnegie and Wu (2019). Given the relatively small number of manuscripts, it is hard to know precisely why those that used more machine-learning methods were harder, but some of the difficulties we encountered included much longer run times and the use of parallelization. Second, the Challenge involved deadlines: there was a hard deadline for submitting to the Challenge and another hard deadline for submitting to the special issue. We think these deadlines, which are uncommon in the social sciences, may have caused code quality to suffer. Finally, we think that in the future, authors and editors will have more experience with computational reproducibility (and we hope that some of the ideas in this article can help with that).

On the other hand, we think achieving computational reproducibility could be harder in other settings for three main reasons. First, in other settings there may be less buy-in from authors and editors; in this case, we think the fact that the Fragile Families Challenge was a mass collaboration helped set the tone of open and reproducible research. Second, in other settings authors may have less experience writing code, which could create barriers to computational reproducibility. Third, in other settings, authors and editors will have to deal with the difficulties related to data access, which did not arise here because all Challenge participants were using the same data set that we had provided.[19]

[19]For more on data sharing, privacy, and reidentification risk, we recommend Lundberg et al. (2019), Meyer (2018), Salganik (2018), and Crosas et al. (2018).

Recommendations for Authors

On the basis of our experience with the special issue—as well as the suggestions of others (Alvarez et al. 2018; Barnes 2010; Benureau and Rougier 2018; Bowers and Voors 2016; Eubank 2016; Fehr et al. 2016; Grüning et al. 2018; Konkol et al. 2019; Marwick, Boettiger, and Mullen 2018; Nagler 1995; Peng 2009; Sandve et al. 2013; Stodden et al. 2016; Tatman et al. 2018; Wilson et al. 2017)—we have some recommendations for authors who wish to improve the computational reproducibility of their results now and in the future. We believe that these recommendations are applicable to all authors, even if they are not using software containers and cloud computing.

We find it helpful to conceptualize the code used to create the results in a manuscript as a *research pipeline* (Peng and Eckel 2009). The research pipeline accepts the rawest data and produces all the final outputs in the manuscript (Figure 3). To promote computational reproducibility, we recommend that researchers make their research pipelines automated, modular, and friendly. Before describing these three ideas in more detail, and offering a five-item checklist, we want to emphasize that our experience with the Challenge suggest that most researchers do not create automated, modular, and friendly research pipelines without effort. In the Challenge, the code that was most effective had repeatedly gone through a process that software engineers call *refactoring*. Roughly, refactoring involves taking code that already works and improving its internal structure without changing its external behavior (Fowler 1999). In professional software engineering, refactoring is not considered a sign of poor work; rather it is considered a best practice that acknowledges the reality that software projects evolve over time and that design decisions made early in the project may need to be change as the project evolves. In the context of the computational reproducibility of an academic article, refactoring might include changes such as removing code that is no longer used, improving variable names, and adding a loop or writing a function to eliminate redundant blocks of code. Just as a researcher often writes, rewrites, and re-rewrites a manuscript to communicate ideas as clearly as possible, we think that researchers should also expect to write, rewrite, and re-rewrite their code to end up with a computationally reproducible research pipeline.

Three Guiding Principles

Automated.—A research pipeline should run from start to finish without human intervention; it should be fully automated. The desire for end-to-end automation is not often common during the research process, because authors are often focused on creating a specific component of the pipeline, such as a specific figure. However, once a manuscript is complete, it is important that all the individual software components seamlessly communicate with each other and can execute from start to finish without any manual intervention to produce all the results that are presented in the manuscript.

Modular.—As code becomes more complex, it becomes increasingly important to break it into independent modules, such that each module serves a specific purpose and the inputs and outputs of each module are clear. Modular code offers many benefits during the research process and during the process of preparing the code for others (as well as for your

future self). While doing research, modularity benefits authors by making the code easier to understand, maintain, and improve. Furthermore, for analyses that involve long run times, such as many of the research pipelines in the Fragile Families Challenge, modularity lets authors run only specific sections of their code.

Additionally, modularity helps computational reproducibility in three ways. First, it helps future researchers understand how the code works; a well-designed, modular code architecture is a wonderful form of documentation. Second, if computational reproducibility is not successful, having modular code dramatically improves the chances to find and then fix problems. Finally, modular code promotes refinement and reuse by future researchers, because it enables them to easily focus on only the parts of the code that are most important to them.

Friendly.—The final characteristic of a good research pipeline is that it is user-friendly. In other words, even thought the code is written to be executed by a computer it should also be written to be read by a person.[20] A user-friendly research pipeline is organized, clean, and clear for someone else without detailed knowledge of the code base. In the special issue, we learned that code that was clear to authors was not always clear to us. To promote friendliness, therefore, we encourage authors to attempt to put themselves in the place of others and ask, "What would I want if I was trying to understand, reproduce, and then improve this code?"

Five-Item Checklist

The three principles that are described above—automated, modular, and friendly—are designed to provide high-level guidance to authors. However, these principles are intentionally abstract, and they do not always provide clear action-guiding advice. Therefore, we now present a simple five-item checklist that is designed to help authors and that could be used by journals. This checklist does not cover every possible situation, but it is designed to be helpful to a wide range of social scientists, impose minimal costs, and be independent of the programming language used by the researcher.

Run-All Script.—We recommend that authors include a single "run-all" script that contains a sequence of commands that reproduces all the results in the article. For example, during the Challenge, we suggested that authors write a bash script to call their Python, R, or Stata files.[21] The run-all script serves as a project blueprint; it provides a high-level understanding of the relationships among the scripts, without the need to understand each script individually. The run-all script was one of the first things we reviewed when attempting to reproduce a result, and so it should be considered a key piece of documentation. For example, Figure 4 reproduces the run-all scripts from Filippova et al. (2019). The script first executes an R Markdown file that contains data-cleaning code before executing a series of imputation and then modeling scripts. Without this run-all script it would be nearly impossible for someone trying to reproduce these results to know the proper

[20]Here we are inspired by the idea of literate programming in Knuth (1984): "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."
[21]For more complex scripts, authors could also consider using a makefile (Piccolo and Frampton 2016).

order of execution of these 17 scripts and the appropriate way to move the output of some scripts into another directory.

Informative Directory Structure.—To supplement the run-all script, we recommend that authors organize their files into a clear directory structure. For example, we asked authors to organize their files in three major directories: code, data, and output. A standardized directory structure is helpful during the execution process because some complex pipelines involve creating multiple intermediate files and the directory structure helps maintain order. Authors can adjust their directory structures to fit the nature of their code. For example, the directory structure in Rigobon et al. (2019), shown in Figure 5, made many common operations more convenient. First, all of the raw Challenge data files, which are not included in the replication materials because of privacy concerns, were meant to be placed in a clearly specified directory (/data). Second, all of the final results created by the script were placed in a single directory named “final_pred” This enabled us to save a copy of these results in “final_pred_author” before rerunning the script, which made it easy for us to compare the two sets of results.

Requirements Files.—We recommended that authors include a file listing all of the software and add-on packages used, including the exact version numbers (e.g., “ggplot2 v3.1.0”). Understanding the software requirements is helpful for reproducing results today, and it is necessary for reproducing results in the future. Many of the articles in the special issue used open-source software with a large number of add-on packages, and these packages are regularly updated with improvements. Although these improvements are generally good for users, they are problematic for reproducibility. These new version sometimes introduce what are called *breaking changes*, which cause code that used to work to suddenly break. For example, the improvements between Python 2 and Python 3 introduced some breaking changes into Compton (2019), which we discovered when we attempted to use Python 3 to import preprocessed data files that had been serialized with Python 2. Furthermore, new versions sometimes cause much more subtle errors. For example, there was a change in Python’s pseudorandom number generator between Python 3.2 and Python 3.3 that could cause some results to fail to be computationally reproducible (Benureau and Rougier 2018).

The deadline for initial submission of manuscript and code to the special issue was October 17, 2017, and the deadline for the final, revised submissions was November 13, 2018. During this 13-month period many of the packages used by participants during the Challenge were improved. However, these improvements also led to problems for computational reproducibility. For example, we when we attempted to reproduce Raes (2019) using the most recent version of all the packages used, the results matched for only 2 of the 11 prediction files. For the other files, the differences were sometimes off by a constant and sometimes off by random noise (Figure 6). However, after matching package versions exactly, we were able to reproduce the results exactly (Figure 6). We close by noting that the problems we had with package versions during the relatively short time window of the special issue will grow over time. Without knowing specific package

numbers, we think it will be nearly impossible for researchers in the future to reproduce Challenge results, even if the data and code are otherwise available.

Helpful Code Headers.—We recommended that researchers clearly document their code. Unfortunately, we found that some authors struggled with this very general advice. Therefore, more specifically, we recommend that authors produce code headers that contain the following information:

- Purpose (in 280 characters or fewer)
- Inputs (i.e., What files does this piece of code require? Is there any code that needs to be run before this code?)
- Outputs (i.e., What files, tables or figures does this piece of code create as outputs?)
- Machine used (i.e., computer type [laptop, desktop, cluster], operating system)
- Expected runtime (e.g., seconds, minutes, hours, days)

For example, Figure 7 presents the code header from the data-cleaning script for Ahearn and Brand (2019). Each piece of information helps the replicator understand what should happen upon execution.

Run Twice.—Once authors have completed the other four items on their checklist, we recommend that they run the code twice and ensure that it produces the same results. Running the code twice is necessary if it uses a pseudorandom number generator. Roughly, every time a computer generates a sequence of random numbers, the results will be different. But, these results are not truly random; rather they are a pseudorandom sequence of numbers that are determined by a single seed value that was used to initiate the sequence. If the code sets the seed value explicitly, then the results will be reproducible, but if it does not explicitly set a seed value, the exact same code, using the exact same data, and run on the exact same computer can produce different results.[22] For example, initially we were unable to reproduce the results of Stanescu et al. (2019), because their code did not explicitly set a seed value. However, after they explicitly seeded their code, we were able to reproduce their results. For some more computationally complex submissions, multiple seed values had to be set for different packages that used different seeds. For example, Davidson (2019) required five different seeds, ranging from the Python standard library's seed to Tensor-Flow's own graph-level seed. Thus, for some complex submissions, such as Davidson (2019) and Roberts (2019), it was often difficult to detect if the code had been properly seeded, and an empirical test, such as testing for consistency across runs in the same computing environment, was most effective.

We hope that these three principles and this five-item checklist can help authors who want to make their work more computationally reproducible for other researchers. There are also steps that journals can take to promote reproducibility, and we offer some of those recommendations next.

[22]For more on random number generation and seeding, see Press et al. (2007).

Recommendations for Journals

Although we believe that individual authors have strong incentives to practice open and reproducible research, journals and other institutions also play a key role in accelerating progress toward open and reproducible science (Freese and King 2018). Building on our experience with computational reproducibility in this special issue, we offer a menu of four options that journals can take to promote computational reproducibility, we make specific suggestions related to TOP Level 3, and we present our preferred process.

Menu of Options

Journals can promote computational reproducibility with sticks and carrots. That is, journals can force authors to comply with TOP Level 2 or TOP Level 3 (sticks). Or they can induce authors to comply these guidelines with benefits, such as faster review time or increased probability of acceptance (carrots). Of the carrots available to journals, we believe that a badging system is a particularly promising option. Badges are small symbols that appear on published articles that quickly and clearly indicate the open-science practices associated with those articles (Figure 8). Badges reward authors for following open-science practices, and there is some evidence that badging systems have had their intended effect of promoting open and reproducible research (Kidwell et al. 2016; Rowhani-Farid, Allen, and Barnett 2017). Social scientists are probably most familiar with the badging system developed by the Center for Open Science and first used by the journal *Psychological Science* starting in January 2014 (Eich 2014; Kidwell et al. 2016; Rowhani-Farid et al. 2017). However, similar badging systems have also been developed in other fields. The journal *Biostatistics* introduced a badging system in 2009 (Peng 2009, 2011), and the Association for Computing Machinery, a prominent professional association of computer scientists, recently developed a badging system specifically focused on computational reproducibility (Boisvert 2016). Badging systems are often opt-in (i.e., authors can choose to participate or not), which makes them easier to implement than other systems.[23]

Given the two main levels of computational reproducibility—TOP Level 2 ("access before publication") and TOP Level 3 ("verification before publication")—and two main policy options—requirements and badges—we think that journals could consider four main options: (1) badge to TOP Level 2, (2) require TOP Level 2, (3) require TOP Level 2 and badge to TOP Level 3, and (4) require TOP Level 3. In this special issue, we attempted to require TOP Level 3, and we were not able to achieve this goal in the time available. Therefore, we ended up requiring TOP Level 2, although we note that many of the articles in this special issue actually did achieve TOP Level 3.

Suggestions Regarding TOP Level 3

For journals considering either requiring TOP Level 3 or badging to TOP Level 3, we have five suggestions. First, journals should have a clear plan for who is going to do the

[23]Given that defaults can have a big impact on behavior (Johnson and Goldstein 2003), we think that journals could also consider a badging system that is opt-out. In other words, authors would be assumed to follow open-science practices, and badges would mark articles for which these practices are not followed. To the best of our knowledge, no journal currently uses this kind of opt-out badging system.

additional work that is required to ensure computational reproducibility; we think the most likely possibilities include authors, reviewers, editors, existing journal staff members, and new journal staff members.[24] In our case, we did not realize how much extra work would be required, and we did not have a clear plan of who would be responsible for it. Therefore, we ended up doing a lot of work helping authors make their work more reproducible. This was in part because we did not want to reject any articles that failed to achieve computational reproducibility. In retrospect, it is clear that we should have done a better job clarifying the allocation of responsibilities with the authors.

Second, we believe that badging to TOP Level 3 will be substantially easier than requiring TOP Level 3. In our case, it was relatively easy to verify that some of the manuscripts were computationally reproducible but others were extremely difficult. We ended up spending roughly 80 percent of our time on 20 percent of the manuscripts. If we had chosen to badge to TOP Level 3, we expect that authors interested in computational reproducibility, who generally had results that were easier to reproduce, would have attempted to earn the badge and others would not. Furthermore, a badging system would have enabled us to set clear time limits on the amount of time we would spend on each submission, a process sometimes called "timeboxing." For example, we could have told authors that we would spend six hours attempting to reproduce their results, and if we could not do it within that time, the authors would not receive a badge. We think that a timeboxed system would help promote best practices by authors and would save a lot of time for people verifying reproducibility.

Third, we think that journals, or perhaps publishers, should invest in tools that can facilitate the assessment of computational reproducibility. The online manuscript-handling system used by *Socius* does not currently provide tools to support sharing and review of code, so we did most of our computational reproducibility work through e-mail, a process that introduced numerous inefficiencies. As a first step, journals could build connectors to existing tools such as GitHub and create simple computational reproducibility checklists that would become a part of the manuscript submission process. Efforts to automate and standardize the process of assessing computational reproducibility will, in the long run, decrease its cost and increase its probability of success.

Fourth, we think that journals just starting to deal with computational reproducibility should draw insights from other journals that already have developed computational reproducibility and open-science policies, such as *Biostatistics* (Peng 2009), *Psychological Science* (Eich 2014), *ACM Transactions on Mathematical Software* (Heroux 2015), *International Interactions* (Colaresi 2016), the *Quarterly Journal of Political Science* (Eubank 2016), the *Journal of the American Statistical Association* (Baker 2016), the *American Journal of Political Science* (Jacoby, Lafferty-Hess, and Christian 2017), *ReScience* (Rougier et al. 2017), *Political Analysis* (Alvarez et al. 2018), and *Methods in Ecology and Evolution*

[24]We wish to emphasize that the people needed to do this additional work related to computational reproducibility will need skills that are already in high demand in research and industry. For example, doing the computational reproducibility work for this special issue required basic familiarity with bash scripting, Git, Docker, and cloud computing; in-depth familiarity with R and Python; and a strong background in statistics and machine learning. People with this skill set are currently in high demand, and we expect that for the foreseeable future, people who have the skills needed to computationally verify scientific research are likely to have many other demands on their time.

(Freckleton 2018). We made a mistake by not including detailed instructions about computational reproducibility in our initial call for papers, and we could have easily developed such instructions based on those used by these other journals.

Finally, for journals thinking of requiring TOP Level 3, we would encourage editors to consider whether they have full support from all important stakeholders, including the publisher, the journal's editorial board, and the pool of potential reviewers and authors. One way to assess this support concretely is with the following question: "Are we willing to reject a manuscript that we would otherwise publish because it is not computationally reproducible?" If the answer is not a resounding yes, we would not recommend attempting to require TOP Level 3. Furthermore, editors should assess whether stakeholders are willing to bear the costs that may come with requiring TOP Level 3, which could be assessed with questions such as "Are we willing to have our turn-around time for manuscripts increase by one week (or one month or three months)?" and "Are we willing to have the number of submissions decrease by 10 percent (or 20 percent or 50 percent)?" In our case, a core value of *Socius* is "rapid dissemination of high-quality, peer-reviewed research, produced in time to be relevant to ongoing research and public debates" (Keister and Moody 2015), and that value did not always align with the time requirements of our computational reproducibility verification process.

Our Preferred Model

For journals that currently have no policies on computational reproducibility, we think that a possible first step would be to require TOP Level 2 and badge to TOP Level 3. Badging to TOP Level 3 would require code review, so we briefly sketch our preferred workflow, one that elevates the code to a first-class status alongside each article (Figure 9). When authors submit manuscripts, they also submit the code, data, and computing environment needed to reproduce all the results in their manuscripts. The manuscript and replication materials would then be sent to several peer reviewers and to a reproducibility reviewer, who could be a student, a peer, or a professional software engineer. The peer reviewers would write reports, much as they do now; they would be free to assess the replication materials or not. The reproducibility reviewer would, however, focus entirely on the replication materials. He or she would spend a fixed maximum amount of time, say 6 hours, attempting to reproduce the results in the manuscript. Then the reviewer would write a reproducibility report, which would go to the editor and be considered along with the manuscript reviews. If an author is encouraged to submit a revised manuscript, he or she would need to reply to the comments of the peer reviewers and reproducibility reviewer. We think that adding a reproducibility review into the editorial process appropriately draws attention to the code and allows community standards to vary by field and evolve over time. Furthermore, the introduction of a reproducibility reviewer who will work for a fixed maximum amount of time on each submission makes the costs of reproducibility reviews predictable and manageable.

Conclusions

In this article we have described our experience with promoting computational reproducibility for the special issue of *Socius* on the Fragile Families Challenge. Our approach was inspired by previous efforts, but it differed from other approaches in the social sciences in terms of focus, motivation, and timing. On the basis of our success and struggles, we have offered recommendations to authors who want to increase the computational reproducibility of their work and to journals that want to promote computational reproducibility.

One important aspect to our approach to computational reproducibility was to expand the focus and include the computing environment in addition to code and data. As sociologists and other social scientists increasingly publish articles based on larger data sets and more computationally intensive methods, we expect the need to have this broader focus will increase. In addition to the tools that we chose to use, there are a number of other tools that may be useful to future researchers interested in computational reproducibility.[25]

Our work also raises questions about the costs of computational reproducibility. Our experience suggests that ensuring computational reproducibility of complex computational approaches is going be time consuming for authors and journals. Who should bear these costs? Should we have different computational reproducibility standards for different kinds of research? In what situations would access to code be sufficient, even if that code cannot be rerun or if it could be rerun but produces slightly different results? What about research that is computationally reproducible, but where the code is extremely hard to understand? What if results are computationally reproducible in some computing environments but not others? How long after publication should we reasonably expect results to be reproducible? Ultimately these are questions that have to be answered by the scientific community, and we expect that the answers will vary from field to field. Despite these important unresolved questions, we hope the approach that we took to promote computational reproducibility will increase the impact of the work published in the special issue, further the research goals of the Fragile Families Challenge, and contribute to broader efforts to create a more transparent and open system of scientific publishing.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

We thank the Board of Advisers of the Fragile Families Challenge. Research reported in this publication was supported by the Russell Sage Foundation and the Eunice Kennedy Shriver National Institute of Child Health and Human Development of the National Institutes of Health under award number P2-CHD047879. Funding for the Fragile Families and Child Wellbeing Study was provided by the Eunice Kennedy Shriver National Institute of Child Health and Human Development through grants R01-HD36916, R01-HD39135, and R01-HD40421 and by a

[25]Some examples of tools that might be useful to other researchers are GUIdock (Hung et al. 2016), Algo-Run (Hosny et al. 2016), ReproZip (Chirigati et al. 2016)/ReproServer (Rampin et al. 2018), Singularity (Kurtzer et al. 2017), Rocker (Boettiger and Eddelbuettel 2017), Jug (Coelho 2017), archivist (Biecek and Kosiski, 2017), encapsulator (Pasquier et al. 2018), Binder (Project Jupyter et al. 2018), Packrat (Ushey et al. 2018), and Whole Tale (Brinckman et al. 2019).

consortium of private foundations, including the Robert Wood Johnson Foundation. We thank Caitlin Ahearn, Drew Altschul, Nicole Carnegie, Connor Gilroy, Brian Goode, Seth Green, Jake Hofman, Alex Kindel, Dawn Koffman, Daniel Rigobon, Julia Rohrer, and Janet Xu for helpful feedback on drafts of this article. This article draws on David Liu's senior thesis for the Department of Computer Science at Princeton University (Liu 2018). The content of this article is solely the responsibility of the authors and does not necessarily represent the views of anyone else.

Author Biographies

David M. Liu graduated from Princeton University in 2018 with a concentration in computer science and a certificate in statistics and machine learning. In college, David's interest in exploring interdisciplinary boundaries led him to take courses in applied statistics and intern at the *New York Times*. His senior thesis was about computational reproducibility in the Fragile Families Challenge. He is now a software engineer working on systems infrastructure in the financial data industry.

Matthew J. Salganik is a professor of sociology at Princeton University, and he is affiliated with several of Princeton's interdisciplinary research centers: the Office for Population Research, the Center for Information Technology Policy, the Center for Health and Wellbeing, and the Center for Statistics and Machine Learning. His research interests include computational social science, social networks, and methodology. He is the author of *Bit by Bit: Social Research in the Digital Age* (Princeton University Press, 2018).

References

Ahearn Caitlin E., and Brand JE. 2019. "Predicting Layoff among Fragile Families." Socius 5(1).

Alsheikh-Ali Alawi A., Qureshi Waqas, Al-Mallah Mouaz H., and Ioannidis John P. A.. 2011. "Public Availability of Published Research Data in High-Impact Journals." PLoS ONE 6(9):e24357. [PubMed: 21915316]

Altschul Drew M. 2019. "Leveraging Multiple Machine Learning Techniques to Predict Major Life Outcomes from a Small Set of Psychological and Socioeconomic Variables: A Combined Bottom-Up/Top-Down Approach." Socius 5(1).

Alvarez R. Michael, Ellen M. Key, and Lucas Nez. 2018. "Research Replication: Practical Considerations." PS: Political Science & Politics 51(2):422–26.

Anderson Richard G., Greene William H., McCullough BD, and Vinod HD. 2008. "The Role of Data/Code Archives in the Future of Economic Research." Journal of Economic Methodology 15(1):99–119.

Andrew Rose L., Albert Arianne Y. K., Renaut Sebastien, Rennison Diana J., Bock Dan G., and Vines Tim. 2015. "Assessing the Reproducibility of Discriminant Function Analyses." PeerJ 3:e1137. [PubMed: 26290793]

Baker Monya. 2016. "Why Scientists Must Share Their Research Code." Nature News. Retrieved May 5, 2019. https://www.nature.com/news/why-scientists-must-share-their-research-code-1.20504.

Barba Lorean A. 2018. "Terminologies for Reproducible Research." arXiv. Retrieved May 5, 2019. https://arxiv.org/abs/1802.03311.

Barnes Nick. 2010. "Publish Your Computer Code: It Is Good Enough." Nature 467(7317):753. [PubMed: 20944687]

Benureau Fabien C. Y., and Rougier Nicolas P.. 2018. "Re-run, Repeat, Reproduce, Reuse, Replicate: Transforming Code into Scientific Contributions." Frontiers in Neuroinformatics 11:69. [PubMed: 29354046]

Biecek Przemysław, and Kosiski Marcin. 2017. "archivist: An R Package for Managing, Recording and Restoring Data Analysis." Journal of Statistical Software 82(11).

Boettiger Carl. 2015. "An Introduction to Docker for Reproducible Research." SIGOPS Operating Systems Review 49(1):71–79.

Boettiger Carl, and Eddelbuettel Dirk. 2017. “An Introduction to Rocker: Docker Containers for R.” R Journal 9(2):527–36.

Boisvert Ronald F. 2016. “Incentivizing Reproducibility.” Communications of the ACM 59(10):5.

Bowers Jake, and Voors Maarten. 2016. “How to Improve Your Relationship with Your Future Self.” Revista de Ciencia Poltica (Santiago) 36(3):829–48.

Breiman Leo. 1996. “Stacked Regressions.” Machine Learning 24(1):49–64.

Brinckman Adam, Chard Kyle, Gaffney Niall, Hategan Mihael, Jones Mathew B., Kowalik Kacper, Kulasekaran Sivakumar, Ludäscher Bertram, Mecum Bryce D., Nabrzyski Jarek, Stodden Victoria, Taylor Ian J., Turk Matthew J., and Turner Kandace. 2019. “Computing Environments for Reproducibility: Capturing the Whole Tale.” Future Generation Computer Systems 94:854–67.

Buckheit Jonathan B., and Donoho David L.. 1995. “WaveLab and Reproducible Research.” Pp. 55–81 in Wavelets and Statistics, edited by Antoniadis A and Oppenheim G. New York: Springer.

Campbell Hamish A., Micheli-Campbell Mariana A., and Udyawer Vinay. 2019. “Early Career Researchers Embrace Data Sharing.” Trends in Ecology & Evolution 34(2):95–98. [PubMed: 30573193]

Carnegie NB, and Wu J. 2019. “Variable Selection and Parameter Tuning for BART Modeling in the Fragile Families Challenge.” Socius 5(1).

Center for Open Science. N.d. “TOP Guidelines.” Retrieved May 5, 2019. https://cos.io/our-services/top-guidelines/.

Chang Andrew C., and Li Phillip. 2018. “Is Economics Research Replicable? Sixty Published Papers from Thirteen Journals Say Often Not.” Critical Finance Review 7.

Chen Xiaoli, Dallmeier-Tiessen Sünje, Dasler Robin, Feger Sebastian, Fokianos Pamfilos, Gonzalez Jose B., Hirvonsalo Harri, Kousidis Dinos, Lavasa Artemis, Mele Salvador, Rodriguez Diego R., Šimko Tibor, Smith Tim, Trisovic Ana, Trzcinska Anna, Tsanaktsidis Ioannis, Zimmermann Markus, Cranmer Kyle, Heinrich Lucas, Watts Gordon, Hildreth Michael, Iglesias Lara L., Kati Lassili-Perini, and Neubert Sebastian. 2018. “Open Is Not Enough.” Nature Physics, P. 1.

Chirigati Fernando, Rampin Rémi, Shasha Dennis, and Freire Juliana. 2016. “ReproZip: Computational Reproducibility with Ease.” Pp. 2085–88 in Proceedings of the 2016 International Conference on Management of Data, SIGMOD ‘16, New York: Association for Computing Machinery.

Claerbout Jon F., and Karrenbach Martin. 1992. “Electronic Documents Give Reproducible Research a New Meaning.” Pp. 601–604 in SEG Technical Program Expanded Abstracts 1992. Tulsa, OK: Society of Exploration Geophysicists.

Clemens Michael A. 2017. “The Meaning of Failed Replications: A Review and Proposal.” Journal of Economic Surveys 31(1):326–42.

Clyburne-Sherin April, and Green Seth A.. 2018. “Computational Reproducibility via Containers in Social Psychology.” PsyArXiv. Retrieved May 5, 2019. https://psyarxiv.com/mf82t.

Coelho Luis P. 2017. “Jug: Software for Parallel Reproducible Computation in Python.” Journal of Open Research Software 5(1).

Colaresi Michael. 2016. “Preplication, Replication: A Proposal to Efficiently Upgrade Journal Replication Standards.” International Studies Perspectives 17(4):367–78.

Collberg Christian, and Proebsting Todd A.. 2016. “Repeatability in Computer Systems Research.” Communications of the ACM 59(3):62–69.

Compton Ryan. 2019. “A Data-Driven Approach to the Fragile Families Challenge: Prediction through Principal Components Analysis and Random Forests.” Socius 5(1).

Coughlin Steven S. 2017. “Reproducing Epidemiologic Research and Ensuring Transparency.” American Journal of Epidemiology 186(4):393–94. [PubMed: 28830078]

Crook Sharon M., Davison Andrew P., and Plesser Hans E.. 2013. “Learning from the Past: Approaches for Reproducibility in Computational Neuroscience.” Pp. 73–102 in 20 Years of Computational Neuroscience, edited by Bower JM. New York: Springer.

Crosas Mercè, Gautier Julian, Karcher Sebastian, Kirilova Dessi, Otalora Gerard, and Schwartz Abigail. 2018. “Data Policies of Highly-Ranked Social Science Journals.” SocArXiv.

Davidson Thomas. 2019. “Black Box Models and Sociological Explanations: Predicting High School GPA Using Neural Networks.” Socius 5(1).

Dewald William G., Thursby Jerry G., and Anderson Richard G.. 1986. “Replication in Empirical Economics: The Journal of Money, Credit and Banking Project.” American Economic Review 76(4):587–603.

Diethelm Kai. 2012. “The Limits of Reproducibility in Numerical Simulation.” Computing in Science Engineering 14(1):64–72.

Donoho David. 2017. “50 Years of Data Science.” Journal of Computational and Graphical Statistics 26(4):745–66.

Donoho David L. 2010. “An Invitation to Reproducible Computational Research.” Biostatistics 11(3):385–88. [PubMed: 20538873]

Dudley Joel T., and Butte Atul J.. 2010. “Reproducible in Silico Research in the Era of Cloud Computing.” Nature Biotechnology 28(11):1181–85.

Easterbrook Steve M. 2014. “Open Code for Open Science?” Nature Geoscience 7:779–81.

Eglen Steven J., Marwick Ben, Halchenko Yaroslav O., Hanke Michael, Sufi Shoaib, Gleeson Padraig, Silver R. Angus, Davison Andrew P., Lanyon Linda, Abrams Mathew, Wachtler Thomas, Willshaw David J., Pouzat Christophe, and Poline Jean-Baptiste. 2017. “Towards Standard Practices for Sharing Computer Code and Programs in Neuroscience.” bioRxiv. Retrieved May 5, 2019. https://www.biorxiv.org/content/10.1101/045104v3.

Eich Eric. 2014. “Business Not as Usual.” Psychological Science 25(1):3–6. [PubMed: 24285431]

Eubank Nicholas. 2016. “Lessons from a Decade of Replications at the *Quarterly Journal of Political Science*.” PS: Political Science & Politics 49(2):273–76.

Fehr Jörg, Heiland Jan, Himpe Christian, and Saak Jens. 2016. “Best Practices for Replicability, Reproducibility and Reusability of Computer-Based Experiments Exemplified by Model Reduction Software.” AIMS Mathematics 1(3):261–81.

Filippova Anna, Gilroy Connor, Kashyap Ridhi, Kirchner Antje, Morgan Allison C., Polimis Kivan, Usmani Adaner, and Wang Tong. 2019. “Humans in the Loop: Incorporating Expert and Crowdsourced Knowledge for Predictions Using Survey Data.” Socius 5(1).

Fowler Martin. 1999. Refactoring: Improving the Design of Existing Code. Reading, MA: Addison-Wesley Professional.

Fragile Families Challenge Team. 2019. “Measuring the Predictability of Life Outcomes with a Scientific Mass Collaboration.” Working paper.

Freckleton Robert P. 2018. “Accessibility, Reusability, Reliability: Improving the Standards for Publishing Code in *Methods in Ecology and Evolution*.” Methods in Ecology and Evolution 9(1):4–6.

Freese Jeremy. 2007. “Replication Standards for Quantitative Social Science: Why Not Sociology?” Sociological Methods & Research 36(2):153–72.

Freese Jeremy, and King Molly M.. 2018. “Institutionalizing Transparency.” Socius 4:1–7.

Freese Jeremy, and Peterson David. 2017. “Replication in Social Science.” Annual Review of Sociology 43(1):147–65.

Gentleman Robert, and Lang Duncan T.. 2007. “Statistical Analyses and Reproducible Research.” Journal of Computational and Graphical Statistics 16(1):1–23.

Gertler Paul, Galiani Sebastian, and Romero Mauricio. 2018. “How to Make Replication the Norm.” Nature 554(7693):417.

Gil Yolanda, David Cédric H., Demir Ibrahim, Essawy Bakinam T., Fulweiler Robinson W., Goodall Jonathan L., Karlstrom Leif, Lee Huikyo, Mills Heath J., Oh Ji-Hyun, Pierce Suzanne A., Pope Allen, Tzeng Mimi W., Villamizar Sandra R., and Yu Xu. 2016. “Toward the Geoscience Paper of the Future: Best Practices for Documenting and Sharing Research from Data to Software to Provenance.” Earth and Space Science 3(10):388–415.

Gilbert Kimberly J., Andrew Rose L., Bock Dan G., Franklin Michelle T., Kane Nolan C., Moore Jean-Sébastien, Moyers Brook T., Renaut Sébastien, Rennison Diana J., Veen Thor, and Vines Timothy H.. 2012. “Recommendations for Utilizing and Reporting Population Genetic Analyses: The Reproducibility of Genetic Clustering Using the Program Structure.” Molecular Ecology 21(20):4925–30. [PubMed: 22998190]

Goode Brian J., Datta Debanjan, and Ramakrishnan Naren. 2019. “Imputing Data for the Fragile Families Challenge: Identifying Similar Survey Questions with Semiautomated Methods.” Socius 5(1).

Goodman Steven N., Fanelli Danielle, and Ioannidis John P. A.. 2016. “What Does Research Reproducibility Mean?” Science Translational Medicine 8(341):341ps12.

Grüning Björn, Chilton John, Köster Johannes, Dale Ryan, Soranzo Nicola, van den Beek Marius, Goecks Jeremy, Backofen Rolf, Nekrutenko Anton, and Taylor James. 2018. “Practical Computational Reproducibility in the Life Sciences.” Cell Systems 6(6):631–35. [PubMed: 29953862]

Hardt Moritz, and Blum Avrim. 2015. “The Ladder: A Reliable Leaderboard for Machine Learning Competitions.” Pp. 1006–14 in Proceedings of the 32nd International Conference on Machine Learning. New York: Association for Computing Machinery.

Hardwicke Tom E., and Ioannidis John P. A.. 2018. “Populating the Data Ark: An Attempt to Retrieve, Preserve, and Liberate Data from the Most Highly-Cited Psychology and Psychiatry Articles.” PLoS ONE 13(8):e0201856. [PubMed: 30071110]

Hardwicke Tom E., Mathur Maya B., MacDonald Kyle, Nilsonne Gustav, Banks George C., Kidwell Mallory C., Mohr Alicia H., Clayton Elizabeth, Yoon Erica J., Tessler Michael H., Lenne Richie L., Altman Sara, Long Bria, and Frank Michael C.. 2018. “Data Availability, Reusability, and Analytic Reproducibility: Evaluating the Impact of a Mandatory Open Data Policy at the Journal *Cognition*.” Royal Society Open Science 5(8):180448. [PubMed: 30225032]

Heroux Michael A. 2015. “Editorial: ACM TOMS Replicated Computational Results Initiative.” ACM Transactions on Mathematical Software 41(3):13:1–13:5.

Heuberger Simon. 2019. “Insufficiencies in Data Material: A Replication Analysis of Muchlinski, Siroky, He, and Kocher (2016).” Political Analysis 27(1):114–18.

Hosny Abdelrahman, Vera-Licona Paola, Laubenbacher Reinhard, and Favre Thibaud. 2016. “AlgoRun: A Docker-Based Packaging System for Platform-Agnostic Implemented Algorithms.” Bioinformatics 32(15):2396–98. [PubMed: 27153722]

Howe Bill. 2012. “Virtual Appliances, Cloud Computing, and Reproducible Research.” Computing in Science Engineering 14(4):36–41.

Hung Ling-Hong, Kristiyanto Daniel, Lee Sung Bong, and Yeung Ka Yee. 2016. “GUIdock: Using Docker Containers with a Common Graphics User Interface to Address the Reproducibility of Research.” PLoS ONE 11(4):e0152686. [PubMed: 27045593]

Hutson Matthew. 2018. “Artificial Intelligence Faces Reproducibility Crisis.” Science 359(6377):725–26. [PubMed: 29449469]

Hutton Christopher, Wagener Thorsten, Freer Jim, Han Dawei, Duffy Chris, and Arheimer Berit. 2016. “Most Computational Hydrology Is Not Reproducible, so Is It Really Science?” Water Resources Research 52(10):7548–55.

In’nami Yo, and Koizumi Rie. 2010. “Can Structural Equation Models in Second Language Testing and Learning Research be Successfully Replicated?” International Journal of Testing 10(3):262–73.

Ioannidis John P. A., Allison David B., Ball Catherine A., Coulibaly Issa, Cui Xiangqin, Culhane Aedín C., Falchi Mario, Furlanello Cesare, Game Laurence, Jurman Giuseppe, Mangion Jon, Mehta Tappan, Nitzberg Michael, Page Grier P., Petretto Enrico, and van Noort Vera. 2009. “Repeatability of Published Microarray Gene Expression Analyses.” Nature Genetics 41(2):149–55. [PubMed: 19174838]

Jacoby William G., Lafferty-Hess Sophia, and Christian Thu-Mai. 2017. “Should Journals Be Responsible for Reproducibility?” Inside Higher Ed. Retrieved May 5, 2019. https://www.inside-highered.com/blogs/rethinking-research/should-journals-be-responsible-reproducibility.

Johnson Eric J., and Goldstein Daniel. 2003. “Do Defaults Save Lives?” Science 302(5649):1338–39. [PubMed: 14631022]

Keister Lisa A., and Moody James W.. 2015. “Editorial.” Socius 1:1.

Kidwell Mallory C., Lazarević Ljiljana B., Baranski Erica, Hardwicke Tom E., Piechowski Sarah, Falkenberg Lina-Sophia, Kennett Curtis, Slowik Agnieszka, Sonnleitner Carina, Hess-Holden Chelsey, Errington Timothy M., Fiedler Susann, and Nosek Brian A.. 2016. “Badges

to Acknowledge Open Practices: A Simple, Low-Cost, Effective Method for Increasing Transparency." PLoS Biology 14(5):e1002456. [PubMed: 27171007]

King Gary. 1995. "Replication, Replication." PS: Political Science & Politics 28(3):444–52.

King Gary, Tomz Michael, and Wittenberg Jason. 2000. "Making the Most of Statistical Analyses: Improving Interpretation and Presentation." American Journal of Political Science 44(2):347–61.

Knuth Donald E. 1984. "Literate Programming." Computer Journal 27(2):97–111.

Koenker Roger, and Zeileis Achim. 2009. "On Reproducible Econometric Research." Journal of Applied Econometrics 24(5):833–47.

Konkol Markus, Kray Christian, and Pfeiffer Max. 2019. "Computational Reproducibility in Geoscientific Papers: Insights from a Series of Studies with Geoscientists and a Reproduction Study." International Journal of Geographical Information Science 33(2):408–29.

Kurtzer Gregory M., Sochat Vanessa, and Bauer Michael W.. 2017. "Singularity: Scientific Containers for Mobility of Compute." PLoS ONE 12(5):e0177459. [PubMed: 28494014]

Leek Jeffrey T., and Peng Roger D.. 2015. "Opinion: Reproducible Research Can Still Be Wrong: Adopting a Prevention Approach." Proceedings of the National Academy of Sciences 112(6):1645–46.

Levy Karen E. C., and Johns David M.. 2016. "When Open Data Is a Trojan Horse: The Weaponization of Transparency in Science and Governance." Big Data & Society 3(1):1–6.

Liu DM 2018. Computational Reproducibility and the Fragile Families Challenge: Lessons Learned and Suggestions for the Future. B.S.E. Thesis, Department of Computer Science, Princeton University, Princeton, NJ.

Lundberg Ian, Narayanan Arvind, Levy Karen, and Salganik Michael J.. 2019. "Privacy, Ethics, and Data Access: A Case Study of the Fragile Families Challenge." Socius 5(1).

Lupia Arthur, and Elman Colin. 2014. "Openness in Political Science: Data Access and Research Transparency: Introduction." PS: Political Science & Politics 47(1):19–42.

Mangul Sergei, Mosqueiro Thiago, Duong Dat, Mitchell Keith, Sarwal Varuni, Hill Brian, Brito Jaqueline, Littman Russell Jared, Statz Benjamin, Lam Angela, Dayama Gargi, Grieneisen Laura, Martin Lana S., Flint Jonathan, Eskin Eleazar, and Blekhman Ran. 2018. "A Comprehensive Analysis of the Usability and Archival Stability of Omics Computational Tools and Resources." bioRxiv. Retrieved May 5, 2019. https://www.biorxiv.org/content/10.1101/452532v1.

Manninen Tiina, Havela Riikka, and Linne Marja-Leena. 2017. "Reproducibility and Comparability of Computational Models for Astrocyte Calcium Excitability." Frontiers in Neuroinformatics 11:11. [PubMed: 28270761]

Marwick Ben. 2017. "Computational Reproducibility in Archaeological Research: Basic Principles and a Case Study of Their Implementation." Journal of Archaeological Method and Theory 24(2):424–50.

Marwick Ben, Boettiger Carl, and Mullen Lincoln. 2018. "Packaging Data Analytical Work Reproducibly Using R (and Friends)." American Statistician 72(1):80–88.

McCullough BD, McGeary Kerry Anne, and Harrison Teresa D.. 2006. "Lessons from the *JMCB* Archive." Journal of Money, Credit and Banking 38(4):1093–1107.

McKay Stephen. 2019. "When 4 ≈ 10,000: The Power of Social Science Knowledge in Predictive Performance." Socius 5(1).

Merkel Dirk. 2014. "Docker: Lightweight Linux Containers for Consistent Development and Deployment." Linux Journal 2014(239).

Mesnard Olivier, and Barba Lorena A. 2017. "Reproducible and Replicable Computational Fluid Dynamics: It's Harder Than You Think." Computing in Science Engineering 19(4):44–55.

Meyer Michelle N. 2018. "Practical Tips for Ethical Data Sharing." Advances in Methods and Practices in Psychological Science 1(1):131–44.

Mikowski Marcin, Hensel Witold M., and Hohol Mateusz. 2018. "Replicability or Reproducibility? On the Replication Crisis in Computational Neuroscience and Sharing Only Relevant Detail." Journal of Computational Neuroscience 45(3):163–72. [PubMed: 30377880]

Nagler Jonathan. 1995. "Coding Style and Good Computing Practices." PS: Political Science & Politics 28(3):488–92.

Naudet Florian, Sakarovitch Charlotte, Janiaud Perrine, Cristea Ioana, Fanelli Daniele, Moher David, and Ioannidis John P. A.. 2018. "Data Sharing and Reanalysis of Randomized Controlled Trials in Leading Biomedical Journals with a Full Data Sharing Policy: Survey of Studies Published in the *BMJ* and *PLoS Medicine*." BMJ 360:k400. [PubMed: 29440066]

Nosek BA, Alter G, Banks GC, Borsboom D, Bowman SD, Breckler SJ, Buck S, Chambers CD, Chin G, Christensen G, Contestabile M, Dafoe A, Eich E, Freese J, Glennerster R, Goroff D, Green DP, Hesse B, Humphreys M, Ishiyama J, Karlan D, Kraut A, Lupia A, Mabry P, Madon T, Malhotra N, Mayo-Wilson E, McNutt M, Miguel E, Paluck EL, Simonsohn U, Soderberg C, Spellman BA, Turitto J, VandenBos G, Vazire S, Wagenmakers EJ, Wilson R, and Yarkoni T. 2015. "Promoting an Open Research Culture." Science 348(6242):1422–25. [PubMed: 26113702]

Orozco Valérie, Bontemps Christophe, Maigné Elise, Piguet V, Hofstetter A, Lacroix Anne, Levert F, and Rousselle JM. 2018. "How to Make a Pie: Reproducible Research for Empirical Economics & Econometrics." Technical report, Toulouse School of Economics.

Ostermann Frank O., and Granell Carlos. 2017. "Advancing Science with VGI: Reproducibility and Replicability of Recent Studies using VGI." Transactions in GIS 21(2):224–37.

Pasquier Thomas, Lau Matthew K., Han Xueyuan, Fong Elizabeth, Lerner Barbara S., Boose Emery, Crosas Mercè, Ellison Aaron M., and Seltzer Margo. 2018. "Sharing and Preserving Computational Analyses for Posterity with encapsulator." arXiv.

Patil Prasad, Peng Roger D., and Leek Jeffrey. 2016. "A Statistical Definition for Reproducibility and Replicability." bioRxiv. Retrieved May 5, 2019. https://www.biorxiv.org/content/10.1101/066803v1.

Peng Roger D. 2009. "Reproducible Research and Biostatistics." Biostatistics 10(3):405–408. [PubMed: 19535325]

Peng Roger D. 2011. "Reproducible Research in Computational Science." Science 334(6060):1226–27. [PubMed: 22144613]

Peng Roger D., Dominici Francesca, and Zeger Scott L.. 2006. "Reproducible Epidemiologic Research." American Journal of Epidemiology 163(9):783–89. [PubMed: 16510544]

Peng Roger D., and Eckel Sandrah P.. 2009. "Distributed Reproducible Research Using Cached Computations." Computing in Science & Engineering 11(1):28–34. [PubMed: 21998556]

Piccolo Stephen R., and Frampton Michael B.. 2016. "Tools and Techniques for Computational Reproducibility." GigaScience 5(1):30. [PubMed: 27401684]

Plesser Hans E. 2018. "Reproducibility vs. Replicability: A Brief History of a Confused Terminology." Frontiers in Neuroinformatics 11:76. [PubMed: 29403370]

Press William H., Teukolsky Saul A., Vetterling William T., and Flannery Brian P.. 2007. Numerical Recipes: The Art of Scientific Computing, 3rd ed. Cambridge, United Kingdom: Cambridge University Press.

Project Jupyter, Bussonnier Matthias, Forde Jessica, Freeman Jeremy, Granger Brian, Head Tim, Holdgraf Chris, Kelley Kyle, Nalvarte Gladys, Osheroff Andrew, Pacer M, Panda Yuvi, Perez Fernando, Ragan-Kelley Benjamin, and Willing Carol. 2018. "Binder 2.0—Reproducible, Interactive, Sharable Environments for Science at Scale." Pp. 113–20 in Proceedings of the 17th Python in Science Conference.

Raes L 2019. "Predicting GPA at Age 15 in the Fragile Families and Child Wellbeing Study." Socius 5(1).

Rampin Remi, Chirigati Fernando, Steeves Vicky, and Freire Juliana. 2018. "ReproServer: Making Reproducibility Easier and Less Intensive." arXiv.

Reichman Nancy E., Teitler Julien O., Garfinkel Irwin, and McLanahan Sara S.. 2001. "Fragile Families: Sample and Design." Children and Youth Services Review 23(4):303–26.

Rigobon Daniel E., Jahini Eaman, Suhara Yoshihiko, AlGhoneim Khaled, Alghunaim Abdulaziz, Pentland Alex "Sandy", and Almaatouq Abdullah. 2019. "Winning Models for GPA, Grit, and Layoff in the Fragile Families Challenge." Socius 5(1).

Roberts Claudia V. 2019. "Friend Request Pending: A Comparative Assessment of Engineering and Social Science Inspired Approaches to Analyzing Complex Birth Cohort Survey Data." Socius 5(1).

Rougier Nicolas P., Hinsen Konrad, Alexandre Frédéric, Arildsen Thomas, Barba Lorean A., Benureau Fabien C. Y., Brown C. Titus, Pierre de Buyl Ozan Caglayan, Davison Andrew P., Delsuc Marc-André, Detorakis Georgios, Diem Alexandra K., Drix Damien, Enel Pierre, Girard Benoît, Guest Olivia, Hall Matt G., Henriques Rafael N., Hinaut Xavier, Jaron Kamil S., Khamassi Mehda, Klein Almar, Manninen Tiina, Marchesi Pietro, McGlinn Daniel, Metzner Christoph, Petchey Owen, Plesser Hans E., Poisot Timothée, Ram Karthik, Ram Yoav, Roesch Etienne, Rossant Cyrille, Rostami Vahid, Shifman Aaron, Stachelek Joseph, Stimberg Marcel, Stollmeier Frank, Vaggi Federico, Viejo Guillaume, Vitay Julien, Vostinar Anya E., Yurchak Roman, and Zito Tiziano. 2017. "Sustainable Computational Science: The ReScience Initiative." PeerJ Computer Science 3:e142.

Rowhani-Farid Anisa, Allen Michelle, and Barnett Adrian G.. 2017. "What Incentives Increase Data Sharing in Health and Medical Research? A Systematic Review." Research Integrity and Peer Review 2(1):4. [PubMed: 29451561]

Rowhani-Farid Anisa, and Barnett Adrian G.. 2016. "Has Open Data Arrived at the British Medical Journal (BMJ)? An Observational Study." BMJ Open 6(10):e011784.

Rudin Cynthia. 2018. "Please Stop Explaining Black Box Models for High Stakes Decisions." arXiv. Retrieved May 5, 2019. https://arxiv.org/abs/1811.10154.

Salganik Matthew J. 2018. Bit by Bit: Social Research in the Digital Age. Princeton, NJ: Princeton University Press.

Salganik Matthew J., Lundberg Ian, Kindel Alexander T., and McLanahan Sara. 2019. "Introduction to the Special Collection on the Fragile Families Challenge." Socius 5. doi:10.1177/2378023119871580.

Sandve Geir K., Nekrutenko Anton, Taylor James, and Hovig Eivind. 2013. "Ten Simple Rules for Reproducible Computational Research." PLoS Computational Biology 9(10):e1003285. [PubMed: 24204232]

Savage Caroline J., and Vickers Andrew J.. 2009. "Empirical Study of Data Sharing by Authors Publishing in PLoS Journals." PLoS ONE 4(9):e7078. [PubMed: 19763261]

Schnell Santiago. 2018. "Reproducible Research in Mathematical Sciences Requires Changes in Our Peer Review Culture and Modernization of Our Current Publication Approach." Bulletin of Mathematical Biology 80(12):3095–3105. [PubMed: 30232583]

Shah Nihar B., Tabibian Behzad, Muandet Krikamol, Guyon Isabelle, and Luxburg von Ulrike. 2018. "Design and Analysis of the Nips 2016 Review Process." Journal of Machine Learning Research 19(49):1–34.

Shepherd Bryan E., Peratikos Meridith Blevins, Rebeiro Peter F., Duda Stephany N., and McGowan Catherine C.. 2017. "A Pragmatic Approach for Reproducible Research with Sensitive Data." American Journal of Epidemiology 186(4):387–92. [PubMed: 28830079]

Stanescu Diana, Wang Erik H., and Yamauchi Soichiro. 2019. "Using LASSO to Assist Imputation and Predict Child Wellbeing." Socius 5(1).

Stockemer Daniel, Koehler Sebastian, and Lenz Tobias. 2018. "Data Access, Transparency, and Replication: New Insights from the Political Behavior Literature." PS: Political Science & Politics 51(4):799–803.

Stodden Victoria. 2015. "Reproducing Statistical Results." Annual Review of Statistics and Its Application 2(1):1–19.

Stodden Victoria, McNutt Marcia, Bailey David H., Deelman Ewa, Gil Yolanda, Hanson Brooks, Heroux Michael A., Ioannidis John P. A., and Taufer Michela. 2016. "Enhancing Reproducibility for Computational Methods." Science 354(6317):1240–41. [PubMed: 27940837]

Stodden Victoria, Seiler Jennifer, and Ma Zhaokun. 2018. "An Empirical Analysis of Journal Policy Effectiveness for Computational Reproducibility." Proceedings of the National Academy of Sciences 115(11):2584–89.

Tatman Rachael, VanderPlas Jake, and Dane Sohier. 2018. "A Practical Taxonomy of Reproducibility for Machine Learning Research." Presented at the Reproducibility in Machine Learning Workshop at ICML 2018, Stockholm, Sweden.

Ushey Kevin, McPherson Jonathan, Cheng Joe, Atkins Aron, and Allaire JJ. 2018. “packrat: A Dependency Management System for Projects and their R Package Dependencies.” Retrieved May 5, 2019. https://rdrr.io/cran/packrat/.

Vandewalle Patrick, Kovacevic Jelena, and Vetterli Martin. 2009. “Reproducible Research in Signal Processing.” IEEE Signal Processing Magazine 26(3):37–47.

Vanpaemel Wolf, Vermorgen Maarten, Deriemaecker Leen, and Storms Gert. 2015. “Are We Wasting a Good Crisis? The Availability of Psychological Research Data after the Storm.” Collabra: Psychology 1(1):Article 3.

Vines Timothy H., Albert Arianne Y. K., Andrew Rose L., Dbarre Florence, Bock Dan G., Franklin Michelle T., Gilbert Kimberly J., Moore Jean-Sébastien, Renaut Sébastien, and Rennison Diana J.. 2014. “The Availability of Research Data Declines Rapidly with Article Age.” Current Biology 24(1):94–97. [PubMed: 24361065]

Wicherts Jelte M., Bakker Marjan, and Molenaar Dylan. 2011. “Willingness to Share Research Data Is Related to the Strength of the Evidence and the Quality of Reporting of Statistical Results.” PLoS ONE 6(11):e26828. [PubMed: 22073203]

Wicherts Jelte M., Borsboom Denny, Kats Judith, and Molenaar Dylan. 2006. “The Poor Availability of Psychological Research Data for Reanalysis.” American Psychologist 61(7):726–28. [PubMed: 17032082]

Wicherts Jelte M., and Crompvoets Elise A. V.. 2017. “The Poor Availability of Syntaxes of Structural Equation Modeling.” Accountability in Research 24(8):458–68. [PubMed: 29140742]

Wilson Greg, Bryan Jennifer, Cranston Karen, Kitzes Justin, Nederbragt Lex, and Teal Tracy K.. 2017. “Good Enough Practices in Scientific Computing.” PLoS Computational Biology 13(6):e1005510. [PubMed: 28640806]

Wolpert David H. 1992. “Stacked Generalization.” Neural Networks 5(2):241–59.

Wood Benjamin D. K., Müller Rui, and Brown Annette N.. 2018. “Push Button Replication: Is Impact Evaluation Evidence for International Development Verifiable?” PLoS ONE 13(12):e0209416. [PubMed: 30576348]

Figure 1.

Fragile Families Challenge data structure. Participants built models predicting the age 15 outcomes using data collected between the time the focal child was born and 9 years old. We provided participants with the data represented by the white boxes. Submissions were scored on the basis of their predictive performance (mean squared error) for the observations represented by the gray boxes, which were available only to organizers. The leaderboard set contained one eighth of all observations and was used to provide instant feedback on submissions. The holdout set contained three eighths of observations and was used to produce final scores for all submitted models at the end of the Challenge.

```
#### Begin with a base image containing a specific version of R ####
FROM r-ver:3.4.2

#### Copy the code files from the local file system into the Docker image ####
COPY . /stanescu
WORKDIR /stanescu

#### Install system dependencies ####
RUN apt-get update
RUN apt-get install -y libcurl4-openssl-dev=7.58.0-2
RUN apt-get install -y libssl-dev=1.1.0h-2

#### Install the required R packages ####
RUN Rscript -e "install.packages('readr', dependencies=TRUE, repos='http://cran.rstudio.com/')"
RUN Rscript -e "install.packages('https://cran.r-project.org/src/contrib/Archive/readr/readr_1.1.1.tar.gz')"

RUN Rscript -e "install.packages('Amelia', dependencies=TRUE, repos='http://cran.rstudio.com/')"
RUN Rscript -e "install.packages('https://cran.r-project.org/src/contrib/Archive/Amelia/Amelia_1.7.4.tar.gz')"

RUN Rscript -e "install.packages('dplyr', dependencies=TRUE, repos='http://cran.rstudio.com/')"
RUN Rscript -e "install.packages('https://cran.r-project.org/src/contrib/Archive/dplyr/dplyr_0.7.4.tar.gz')"

RUN Rscript -e "install.packages('tidyr', dependencies=TRUE, repos='http://cran.rstudio.com/')"
RUN Rscript -e "install.packages('https://cran.r-project.org/src/contrib/Archive/tidyr/tidyr_0.8.0.tar.gz')"

RUN Rscript -e "install.packages('glmnet', dependencies=TRUE, repos='http://cran.rstudio.com/')"
# If 2.0-16 is no longer the latest version of 'glment', uncomment the line below:
# RUN Rscript -e "install.packages('https://cran.r-project.org/src/contrib/Archive/glmnet/glmnet_2.0-16.tar.gz')"

#### Downgrade dependencies installed above to the specified version ####
RUN Rscript -e "install.packages('https://cran.r-project.org/src/contrib/Archive/Matrix/Matrix_1.2-11.tar.gz')"
RUN Rscript -e "install.packages('https://cran.r-project.org/src/contrib/Archive/Rcpp/Rcpp_0.12.16.tar.gz')"
# If 1.4.4 is no longer the latest version of 'foreach', uncomment the line below:
# RUN Rscript -e "install.packages(https://cran.r-project.org/src/contrib/Archive/foreach/foreach_1.4.4.tar.gz')"

#### Instruct Docker to start a shell process when an instance of the image is launched  ####
CMD ["sh"]
```

Figure 2.

This Dockerfile specifies the dependencies required for Stanescu et al. (2019). In addition to R, we install system packages, such as libssl-dev, and R packages, such as dplyr. For the R packages, we first installed the newest version of the required packages along with the necessary dependencies. However, sometimes these newest versions did not match the versions used by the authors. In these situations, we then overwrote the newest version with the one that was specified by the authors.

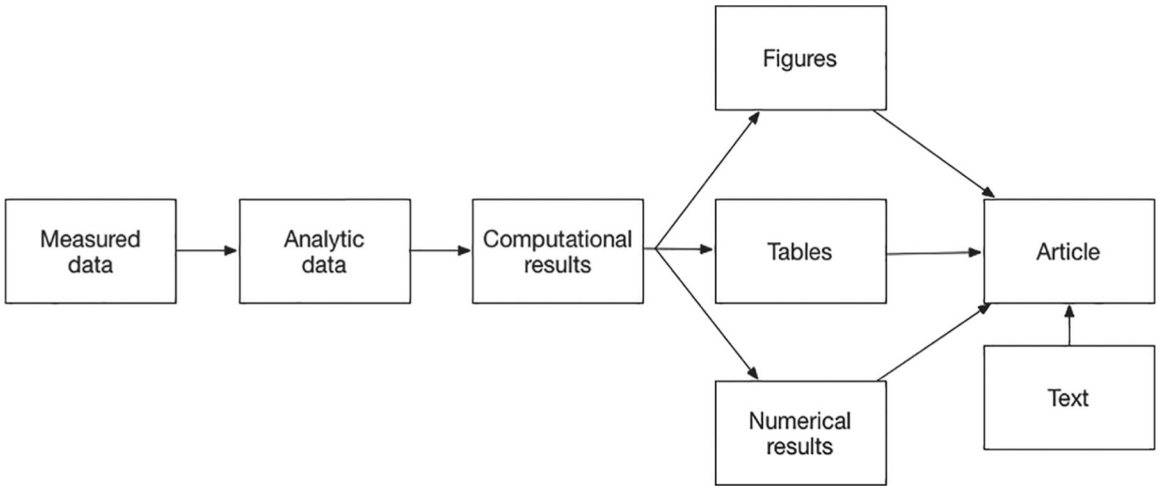


Figure 3.

Research pipeline based on Figure 1 in Peng and Eckel (2009). When writing code, and refactoring code before submissions, authors can imagine a research pipeline that starts with the rawest form of the data, what Peng and Eckel called the "measured data," and ends with all the figures, tables, and numerical results in an article. To promote computational reproducibility, we recommend that authors make their research pipelines automated, modular, and friendly.

```
#!/bin/bash
# create lists of variable metadata
Rscript -e "rmarkdown::render('code/data_processing/variable_metadata.Rmd', output_file='variable_metadata.html')" &&
mv code/data_processing/variable_metadata.html doc/vignettes/variable_metadata.html &&

# set up and do imputations
Rscript code/imputation/setup_mi_data.R &&
Rscript code/imputation/mi.R &&
Rscript code/imputation/mi_constructed.R &&
Rscript code/imputation/reg_lm_typed.R &&
Rscript code/imputation/meanmode_typed.R &&
Rscript code/imputation/reg_lasso_untyped.R &&
Rscript code/imputation/reg_lm_untyped.R &&
Rscript code/imputation/reg_lasso_untyped_constructed.R &&

# run models
Rscript code/runs/run_lasso_mi.R &&
Rscript code/runs/run_lasso_mi_constructed.R &&
Rscript code/runs/run_lasso.R &&
Rscript code/runs/run_lasso_constructed.R &&
Rscript code/runs/run_lasso_constructed_lm.R &&
Rscript code/runs/run_lasso_all.R &&
Rscript code/runs/run_lasso_all_mean.R &&
Rscript code/runs/run_lasso_all_lasso.R &&

echo "Done"
```

Figure 4.

This “run-all” script from Filippova et al. (2019) clarifies the exact commands needed to execute all of the scripts as well as the order these scripts should be executed.

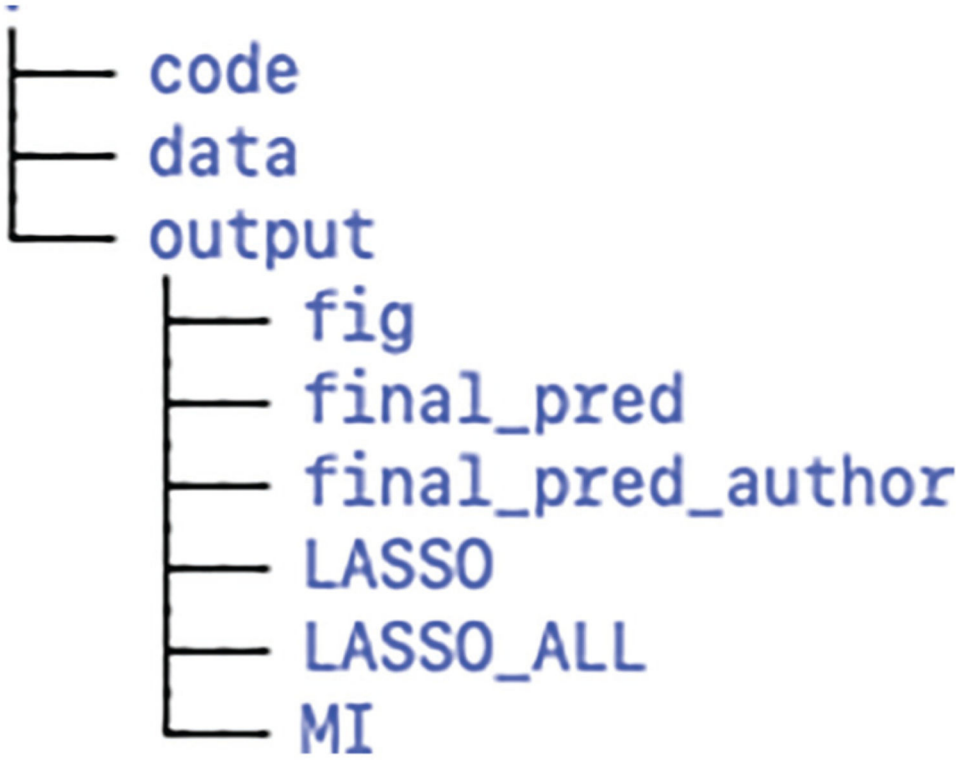
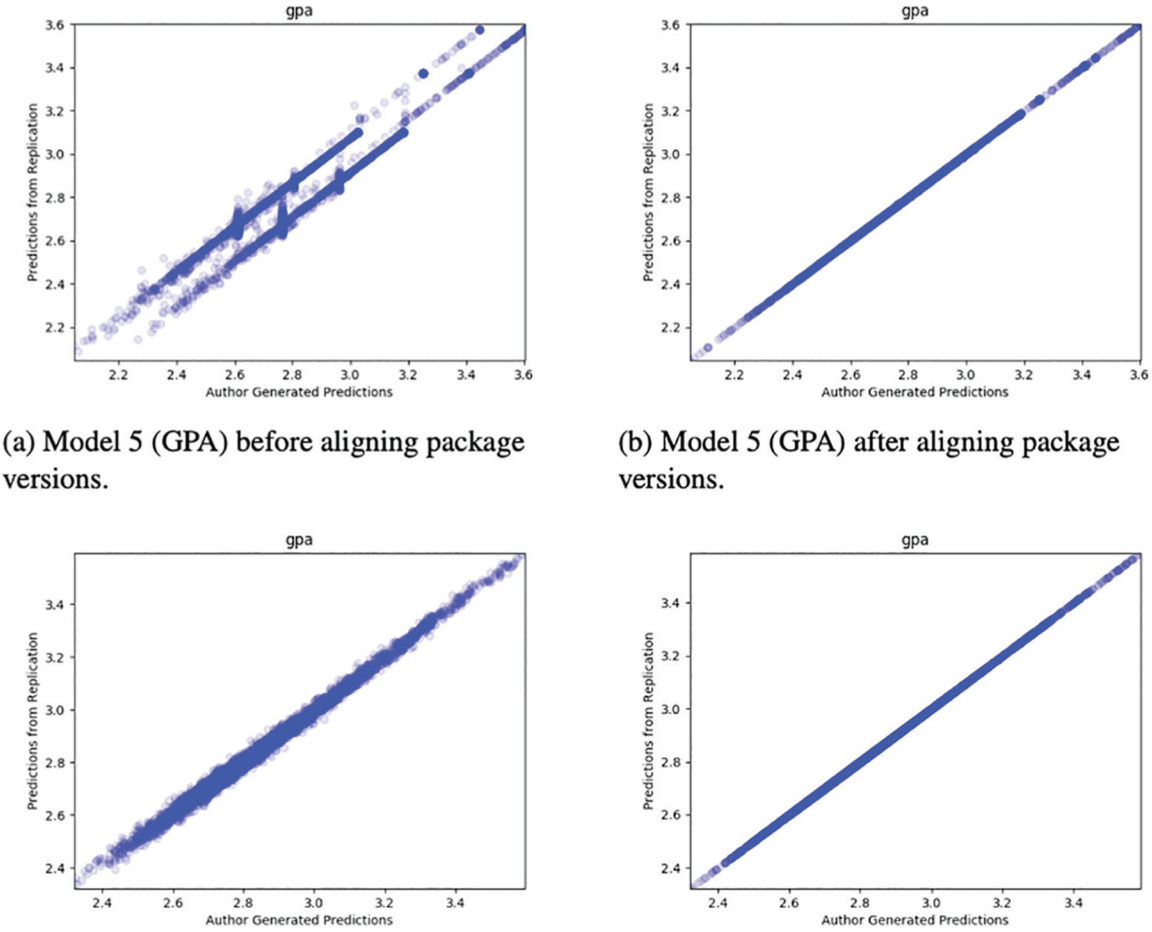


Figure 5.
Directory structure for Rigobon et al. (2019). Because all of the prediction files the authors generated resided in a single subdirectory, we were able to easily save a copy of the original results ("final_pred_author") before generating our own version ("final_pred") for comparison.



(a) Model 5 (GPA) before aligning package versions.

(b) Model 5 (GPA) after aligning package versions.

(c) Model 6 (GPA) before aligning package versions.

(d) Model 6 (GPA) after aligning package versions.

Figure 6.

Examples from Raes (2019). Without specifying package versions, we could not replicate the results, even though we were using the exact same data and code. Furthermore, there was no clear pattern to the errors: sometimes they were off by a constant, and sometimes they were off by random noise. However, once we specified the exact package versions in our container, we were able to reproduce the original results exactly.

```
/*******************************************************************************
        PURPOSE: CLEAN DATA TO PREPARE THEM FOR FINAL ANALYSES
                1: MERGE THE TRAINING AND PREDICTION DATA
                2: RECODE MISSING VALUES
                3: ASSIGN PRIMARY CAREGIVER TO MOTHER OR FATHER
                4: PRIMARY CAREGIVER DEMOGRAPHICS
                5: PRIMARY CAREGIVER HEALTH AND BEHAVIOR
                6: PRIMARY CAREGIVER WORK INDICATORS
                7: SINGLE IMPUTATION

        Input and output files:  selected_vars.dta --> cleaned.dta
        Machine: Mac laptop
        Runtime: less than a minute

*******************************************************************************/
```

Figure 7.
This code header is taken from the data-cleaning script from Ahearn and Brand (2019). The information clearly outlines the code's structure and characteristics.

(a) Badges from the Center for Open Science.

(b) Badges from the Association for Computing Machinery.

Figure 8.

Example badges from the Center for Open Science (Kidwell et al. 2016) and the Association for Computing Machinery (Boisvert 2016). Badges are small symbols that appear on published articles that indicate the open-science practices that apply to this particular article. They are an inducement journals can use to promote different research practices.

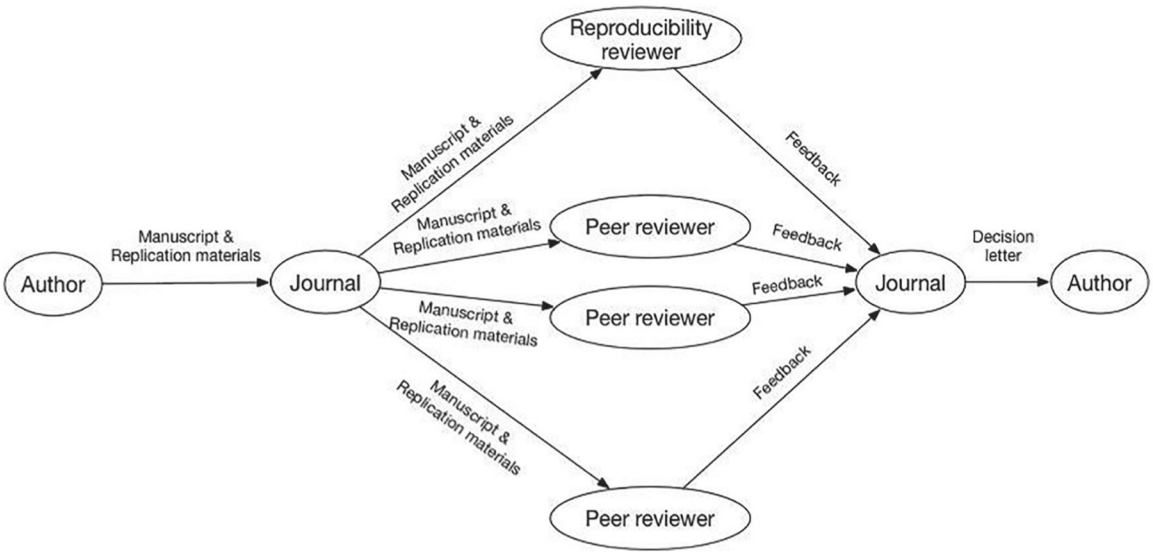


Figure 9.

Schematic of our preferred workflow for journals that are badging to Transparency and Openness Promotion Level 3 (“verification before publication”). Adding a reproducibility review into the editorial process elevates the code to a first-class status alongside each article and allows community standards to vary by field and evolve over time.