Active Maintenance: A Proposal for the Long-term Computational Reproducibility of Scientific Results

Limor Peer* Lilla Orr† Alexander Coppock†

August 26, 2020

Abstract

Computational reproducibility, or the ability to reproduce analytic results of a scientific study on the basis of publicly available code and data, is a shared goal of many researchers, journals, and scientific communities. Researchers in many disciplines including political science have made strides towards realizing that goal. A new challenge, however, has arisen. Code too often becomes obsolete within just a few years. We document this problem with a random sample of studies posted to the ISPS Data Archive; we encountered nontrivial errors in seven of 20 studies. In line with similar proposals for the long-term maintenance of data and commercial software, we propose that researchers dedicated to computational reproducibility should have a plan in place for "active maintenance" of their analysis code. We offer concrete suggestions for how data archives, journals, and research communities could encourage and reward the active maintenance of scientific code and data.

*Yale University Institution for Social and Policy Studies

†Yale University Department of Political Science

‡We thank Jamie Druckman and Ann Green for helpful feedback on earlier drafts and Yuntian Xia and Brian Shih for excellent research assistance.

Introduction

A study is computationally reproducible if a sufficiently savvy third party could download the code and data from a public repository and successfully execute the analysis with minimal difficulty. Initiatives like DA-RT in political science (Lupia and Elman, 2014) reflect a growing consensus that at least for research that relies on the statistical analysis of quantitative data which can be safely shared, computational reproducibility is a shared goal.

As argued in a National Academies of Sciences, Engineering, and Medicine report "the scientific enterprise depends on the ability of the scientific community to scrutinize scientific claims" (National Academies of Sciences Engineering and Medicine, 2019, p. 6). Evaluating published scientific claims can be difficult because readers must infer from written descriptions of statistical procedures what the authors have actually done. Even when those descriptions are accurate, they are necessarily an approximation of what occurred. The move toward computational reproducibility as a precondition of publication has been successful in revealing analytical errors and ensuring that shared materials are given persistent links (Gertler and Bullock, 2017).

We consider what happens *after* materials have been deposited at public archives. A study that was computationally reproducible on the day of deposit has already cleared a high bar. In addition to rigorous scrutiny of the manuscript, the code and data, or "replication archives," were demonstrated by a trusted party to reproduce the numerical results reported in the paper. However, computational reproducibility is dynamic and can be elusive. Code that successfully executes on the day of deposit may break a year, a decade, or a generation later. Operating systems change and software packages are updated or not maintained at all. Data used for analysis may become unintelligible if the software used to create the data cannot be run. Workflows that depend on particular online resources may fail if the resources are moved or taken down.

In this paper, we propose "active maintenance" as a solution to this problem. In short, we suggest that replication archives should be periodically inspected to ensure that they run on contemporary computing environments according to a maintenance plan. Active maintenance builds on common practices in the data preservation (Conway, 2000) and code development communities (Fowler and Foemmel, 2006). Both recognize that supporting materials is an ongoing effort that requires resources, infrastructure, and standards.

Whether active maintenance is appropriate in a given case depends on the relative costs and benefits. The benefits of active maintenance are data and code that continue to provide

scholars with tools to understand, critique, and reuse existing scientific knowledge. The costs to active maintenance are non-negligible. Debugging old code takes time and expertise. Whether the benefits exceed the costs may ultimately depend on the value of the scientific knowledge contained in the replication archive and on the difficulty of the required maintenance. We argue that studies deemed as important justify an investment in long-term reproducibility and that the decision to invest in active maintenance ought to be supported by policy. Archives, publishers, scholarly communities, funders, universities, or other stakeholders taking responsibility for the digital materials should make explicit at the time of deposit whether and how they engage in active maintenance.

Motivating example: "Shy Trump Voters"

We begin with a case study that illustrates overtime degradation in computational reproducibility and how active maintenance could address it. Coppock (2017) reports the results of a study that attempts to discern whether some portion of the polling misses in the 2016 U.S. Presidential election could be attributed to survey respondents who misreport their support for Donald Trump for fear of being perceived as racist or sexist.

The ISPS Data Archive reviewed the replication materials for Coppock (2017) in July of 2017 as part of its routine process (Peer and Green, 2012). During the initial review process, archive staff confirmed that all files necessary to replicate the reported results were available. These included anonymized and documented survey data, and program files necessary to analyze the data. The analysis consisted primarily of descriptive summaries and a comparison of direct and list experimental estimates with bootstrapped standard errors. Archive staff suggested a few revisions which the author implemented. These materials were archived on May 1, 2018.

In October of 2019, ISPS archive staff revisited these materials as part of a training exercise. Newly hired staff members encountered an error which prevented the code from running in full. The ISPS Data Archive contacted the author, who traced the error to the removal of the "bootstrap" function from the `broom` package for R (Robinson and Hayes, 2019) and rewrote the bootstrap code using the `rsample` package instead (Kuhn, Chow and Wickham, 2020). ISPS staff then confirmed that results reported in the paper could be computationally reproduced and the ISPS Data Archive released the updated materials.

This case study is an example of unplanned active maintenance. Close working relationships between archive staff and ISPS researchers helped to facilitate a simple troubleshooting and update process. Extending the computational reproducibility of this study cost the author

perhaps 10 emails and an hour of debugging and recoding and helped avoid future questions from users.

Computational reproducibility is dynamic

Achieving computational reproducibility at the time data and code are deposited with an archive or repository is in itself a feat. Even before deposit, computational reproducibility typically requires that researchers maintain organized data, use version control, and test code, as per best practices of reproducible research (Bowers, 2011; Christensen, Freese and Miguel, 2019). Reproducibility is challenged *after* deposit if users cannot make use or make sense of the files.

Datasets may be damaged at the bit level, and data are often contained in software-dependent file formats which may become unreadable if the format is incompatible with current computers. After a period of time, the original software might be altogether unavailable (Peng, 2011) or restricted (e.g., closed-source) or the ability to use it constrained by a large number of components and dependencies (Hinsen, 2019; Matthews et al., 2008). Changes to statistical programing languages can ripple down to the many add-on modules or packages that analysts rely on. For example, R updated how it generates random samples in 2019. The commands `set.seed()` and `sample()`, and their underlying functions, which would previously have given the same results across R versions, were updated in R 3.6.0. The default kind of random number generation was adjusted to allow for greater uniformity of uniform random samples, particularly in large samples. To use the previous default, R users can call `RNGkind(sample.kind = ‘‘Rounding’’)` (Smith, 2019). As this example illustrates, poor documentation of software versions can produce failed attempts to computationally reproduce results of a study after its initial release.

Beyond software versions, metadata for replication archives should include information regarding what data were excluded or included in the analysis and how instruments were developed or calibrated, for example. Especially as conventions change within a field, using human- and machine-readable strategies to produce high quality code can improve the potential for computational reproducibility (Katz, 2018). Interpretability is itself dynamic: even if the documentation makes sense to a contemporary user, it may be less interpretable to users 50 years in the future (or even to the original researcher a few years down the line (Bowers, 2011).)

One commonly offered solution is virtualization or “containerization.” The replication archive and the computing environment as it existed at the moment of deposit are placed in a virtual container, thus addressing the versioning problem described above and automating any

data manipulation tasks. This approach can be seen in solutions like Docker and Singularity. A container is essentially a virtual machine running a single particular task. Containers guarantee that, even if run on different machines, the same results will be reproduced. This solution has many advantages, but re-opening the container and running the data and code within can be technically challenging for some users, and it requires virtualized systems which can themselves become obsolete. More importantly, as Katz (2017) observes, containers "provide bitwise reproducibility, but aren't scientifically useful, because as black boxes, you can't really remix the contents." Researchers may want to integrate legacy data with current and future computation methods in order to reuse, reassess, and reproduce results.

Another approach to extending computational reproducibility is to "run the original software under emulation on future computers" (Rothenberg, 1999). Emulation enables running legacy software on modern computers, thus preventing the loss of information and functionality that inevitably results from new generations of software and hardware. However, emulation can be computing- and resource-intensive and subject to software licensing challenges. In addition, emulated systems may not interact with current software and computing environments, limiting the usefulness of the materials.

Digital archivists often employ strategies to enhance usability including copying and migrating. For example, the ISPS Data Archive automatically creates a copy of data files in .csv format in order to preserve a system-independent, nonproprietary format that is likely to be machine readable in the decades to come. This procedure has some advantages, but some information may be lost in the new format. Migration involves some curatorial responsibility, more than backing up a file, or transferring from punch card to tape to disk. It "includes refreshing the media but also addresses the internal structure of the files so that the information within can be read on subsequent computer platforms, operating systems, and software" (Green et al., 1999). With code being instrumental for computational reproducibility, both to extract meaning from data and as an object worthy of study, more archives and repositories are applying these strategies to software. Recent recommendations on how to sustain content overtime – that materials need to be monitored and a combination of strategies deployed – parallel our active maintenance suggestion (Daigle et al., 2018).

Table 1: Computational reproducibility in 20 studies

	Author Provided Code N = 20	Archive Added Code N = 11
No errors	13 (65%)	3 (27%)
Errors which could be resolved	6 (30%)	8 (74%)
Errors which could not be resolved without further information	3 (15%)	0
Scripts to analyze data not archived	3 (15%)	0

Note: Studies are classified as either running with present day hardware and software, or containing an error. We additionally note if studies included analysis of data which is not publicly available.

Evaluation of the long-term computational reproducibility of 20 studies

In order to assess long-term computational reproducibility in the ISPS Data Archive, we attempted to execute the statistical program files associated with a sample of studies, all of which were computationally reproducible when they were initially archived. To select the sample, we compiled a list of all 97 archived studies, excluded three books, then randomly selected 20 studies. The sample included studies archived between 2009 and 2019, with four to 88 files associated with each study. Files associated with these studies were archived using versions of Stata (18), R (1), and SPSS (1) available at the time of deposit. All studies could be characterized as quantitative political science.

Between December 2019 and March 2020, ISPS Data Archive research assistants (RAs) reviewed the program files using Stata 15, R 3.6.1 and 3.6.2, and SPSS 26. For each study, RAs reviewed documentation and downloaded all statistical program files with the data necessary to run them. After adjusting working directories and file names, they attempted to run each program file and recorded any errors. After troubleshooting, errors were classified based on whether someone attempting computational reproducibility might be able to overcome them without substantially updating the script for the statistical analysis, or after receiving help from the study authors, as summarized in Table 1.

Of the 20 studies reviewed, the statistical program files associated with 13 of them could be run fully with present day hardware and software. Of the remaining seven studies, a variety of challenges to computational reproducibility emerged. In six of these studies, program files produced errors which could be resolved by users with fairly substantial knowledge of the

software. Necessary adjustments, such as updating `outreg` to `outreg2` in Stata files written for version 7 or earlier, are typically trivial for users familiar with the statistical software. In three of the studies, program files produced errors which could not be resolved by present day users without further information or documentation. For example, variables which now appear to be missing would most easily be supplied or explained by study authors. In three of the twenty studies, program files were written to process data that was not made publicly available via the archive. It is often necessary to exclude raw data from replication files, so we did not classify the resulting errors as indicators of failed computational replication.

For eleven studies, those archived prior to July 2013, archive staff added R files at the time of deposit as part of an effort to convert statistical code to open source formats. Of the eight instances where the R programs did not run, several included simple typos. However, emerging challenges to computational reproducibility were generally more difficult to overcome and underscore the potential hazard of open source software. For example, three relied on packages that had been removed from CRAN (e.g., `Design`), or updated in ways that made it impossible to run the original syntax (e.g., `Zelig`).

In sum, we find that even in the relatively short period of 10 years, replication materials which originally allowed for computational reproducibility can break down. Most of the challenges were encountered in the older studies. Sophisticated users would have little difficulty resolving many of the errors which appeared, but even some of the user-resolvable errors would have required substantial troubleshooting and re-coding.

Active Maintenance

Our proposed solution to the problem of overtime degradation in computational reproducibility is active maintenance. Active maintenance requires monitoring materials and deploying a combination of strategies drawn from two fields: data preservation and code development.

Data archivists apply principles of digital preservation and curation to ensure files can be rendered, understood, and reused well into the future. Such "active preservation" is an effort to mitigate the risks of digital obsolescence and to make digital materials more discoverable. Strategies include documentation and migrating data files from older formats to current formats. Proper documentation for computational reproducibility involves describing the research method, the research artifacts (e.g., metadata about the data provider, software, dependencies, hardware). Code documentation is especially important if it needs to be recreated in the event of software challenges. All metadata should be made available in both human and machine-readable

formats. These activities are fundamental to enabling computational reproducibility in the near, medium, and long term.

Software developers are keenly aware of how changes to the computational environment affect the performance of source code. Continuous integration (CI) is the practice of recompiling software packages regularly to verify the software runs without errors (Fowler and Foemmel, 2006). Key principles include frequent automatic testing, documenting specifications and dependencies, and saving all components in one location. The CI process typically concludes at some point as it cannot be expected that all software will be maintained indefinitely. Adopting basic CI principles and committing to automatically re-run computational analysis whenever relevant changes are made to any component of the computational environment can increase the probability that the code remains available and usable (Beaulieu-Jones and Greene, 2017).

Conclusions

Computational reproducibility is a special kind of knowledge – it offers future analysts a way to scrutinize and understand scientific studies in a far more intimate and tangible way than the journal article can provide. But given its dynamic nature, computational reproducibility is time-variant. As political science and other fields develop shared values and norms around computational reproducibility, we must recognize that our commitment to this knowledge should not end on the day we deposit the study in the archive.

We propose active maintenance as a way to ensure long-term computational reproducibility. As the name implies, active maintenance requires effort. That effort raises two important questions: is the effort worth it and who should be making the effort?

We think that the answer to the first question – is it worth opening up replication archives every so often to see if they still run? – will differ from study to study. Because studies differ in their scientific value, the returns to computational reproducibility will differ accordingly. We think that at a minimum, important replication archives should be actively maintained, but we refrain from passing judgment over which studies those are.

The answer to the second question – who should do the maintenance? – also depends on collective values, priorities, and incentives. Research teams could take the responsibility of ensuring reproducibility. This puts the onus on authors to adhere to "best practice" guidelines at the time of the initial deposit, and to keep up with advances in contemporary statistical computing. It also assumes researchers have the incentives and the resources to do the work of active maintenance. Active maintenance could also be crowd-sourced to the wider academic

community. Much like user contributions to software made on GitHub, one could imagine the very scholars who use the replication archives in the first place contributing up-to-date analysis scripts. Many courses have replication project assignments – updating analysis code and uploading it to the repository could be a fitting capstone to such an assignment. This approach requires technological and other support. For example, will university libraries preserve all the research software used in their institutions or outsource this responsibility?

Independent third parties such as the Odum Institute, Curate Science, and ReScience could also provide active maintenance service on behalf of journals, universities, or scholarly associations. While a significant step in the direction of improving computational reproducibility, these services currently only certify computational reproducibility at a point in time, usually upon deposit in a repository.

Alternatively, the responsibility for active maintenance could fall to archive or repository staff. Data archives are well-positioned to provide services verifying computational reproducibility as part of standard curation practice (Peer, Green and Stephenson, 2014). Many archives already conduct automated bit-level checks and could potentially expand to perform routine reproducibility checks. A thorough initial review can provide an opportunity to take steps to increase the potential for long-term reuse. Archives can also enforce policies conducive to active maintenance, requiring, for example, particular file formats, or specific information in a readme file.

In light of the dynamic nature of computational reproducibility, we urge the community to recognize that supporting data and code is an ongoing effort that requires resources, infrastructure, and standards. Studies that were computationally reproducible on the day they were archived often cannot be reproduced on modern software and hardware only a few years later – we think the computational reproducibility of those studies is worth actively maintaining.

References

Beaulieu-Jones, Brett K and Casey S Greene. 2017. "Reproducibility of computational workflows is automated using continuous analysis." *Nature biotechnology* 35(4):342–346.

Bowers, Jake. 2011. "Six steps to a better relationship with your future self.".
URL: *https://cpb-us-e1.wpmucdn.com/blogs.rice.edu/dist/d/2418/files/2013/09/tpm_v18_n2.pdf*

Christensen, Garret, Jeremy Freese and Edward Miguel. 2019. *Transparent and reproducible social science research: How to do open science*. University of California Press.

Conway, Paul. 2000. *Overview: Rationale for Digitization and Preservation*. Northeast Document Conservation Center.

Coppock, Alexander. 2017. "Did shy Trump supporters bias the 2016 polls? Evidence from a nationally-representative list experiment." *Statistics, Politics and Policy* 8(1):29–40.

Daigle, Bradley, Karen Cariani, Carol Kussmann, Nathan Tallman and Lauren Work. 2018. "Levels of Digital Preservation.".
URL: *https://ndsa.org/publications/levels-of-digital-preservation/*

Fowler, Martin and Matthew Foemmel. 2006. "Continuous integration.".
URL: *https://martinfowler.com/articles/continuousIntegration.html*

Gertler, Aaron L and John G Bullock. 2017. "Reference rot: An emerging threat to transparency in political science." *PS, Political Science & Politics* 50(1):166.

Green, A G, J A Dionne, M Dennis, M J Dennis and Digital Library Federation. 1999. *Preserving the Whole: A Two-track Approach to Rescuing Social Science Data and Metadata*. Digital Library Federation.

Hinsen, Konrad. 2019. "Dealing with software collapse." *Computing in Science & Engineering* 21(3):104–108.

Katz, Daniel. 2017. "Is software reproducibility possible and practical?".
URL: *https://danielskatzblog.wordpress.com/2017/02/07/is-software-reproducibility-possible-and-practical/*

Katz, Daniel. 2018. "Fundamentals of Software Sustainability.".
URL: *https://danielskatzblog.wordpress.com/2018/09/26/fundamentals-of-software-sustainability/*

Kuhn, Max, Fanny Chow and Hadley Wickham. 2020. *rsample: General Resampling Infrastructure*.
URL: *https://cran.r-project.org/package=rsample*

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

Matthews, Brian, Brian McIlwrath, David Giaretta and Esther Conway. 2008. The significant properties of software: A study. Technical report.
URL: *http://purl.org/net/epubs/manifestation/9506/SignificantPropertiesofSoftware.pdf*

National Academies of Sciences Engineering and Medicine. 2019. *Reproducibility and Replicability in Science*. Washington, DC: The National Academies Press.
URL: *https://www.nap.edu/catalog/25303/reproducibility-and-replicability-in-science*

Peer, Limor and Ann Green. 2012. "Building an Open Data Repository for a Specialized Research Community: Process, Challenges and Lessons." *International Journal of Digital Curation* 7(1):151–162.

Peer, Limor, Ann Green and Elizabeth Stephenson. 2014. "Committing to Data Quality Review." *International Journal of Digital Curation* 9(1):263–291.

Peng, Roger D. 2011. "Reproducible Research in Computational Science." *Science* 334(6060):1226–1227.
URL: *https://science.sciencemag.org/content/334/6060/1226*

Robinson, David and Alex Hayes. 2019. *broom: Convert Statistical Analysis Objects into Tidy Tibbles*.
URL: *https://cran.r-project.org/package=broom*

Rothenberg, Jeff. 1999. *Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation. A Report to the Council on Library and Information Resources.* ERIC.
URL: *https://www.clir.org/pubs/reports/rothenberg/contents/*

Smith, David. 2019. "What's new in R 3.6.0.".
URL: *https://blog.revolutionanalytics.com/2019/05/whats-new-in-r-360.html*