

Software Citations in Political Science

Vincent Arel-Bundock, *Université de Montréal, Canada*

Joshua McCrain, *University of Utah, USA*

ABSTRACT

Political scientists rely on complex software to conduct research, and much of the software they use is written and distributed for free by other researchers. This article contends that creating and maintaining these public goods is costly for individual software developers but that it is not adequately incentivized by the academic community. We demonstrate that statistical software is used widely but rarely cited in political science, and we highlight a partial solution to this problem: software bibliographies. To facilitate their creation, we introduce `softbib`, an R package that scans analysis scripts, detects the software used in those scripts, and automatically creates bibliographies. We hope that recognizing the contribution of software developers to science will encourage more scholars to create public goods, which could yield important downstream benefits.

Political scientists rely on complex software to conduct research, and much of the software they use is written and distributed for free by other researchers. This software enables efficient data collection, manipulation, visualization, and analysis using cutting-edge methodologies. Without software, the work of most political scientists would be much more difficult or outright impossible.

Research software makes major contributions to the advancement of knowledge in political science, but those contributions usually go unacknowledged; software almost never is cited in our field, even if it is used widely. In a review of 804 articles published by *American Political Science Review* between 2010 and 2021, we found that more than 40% make no mention of any software. Only 23% of the articles that we surveyed included a formal citation of software in their bibliography (McCrain and Arel-Bundock 2023).

Undercitation of software creates a host of problems for the discipline. First—and most obvious—there is little incentive for academics to create free software without the reward of citation. Unless software is cited formally in a bibliography, it can be difficult for authors to convey the importance of their contributions to science to hiring committees, tenure letter writers, and promotion committees. Second, without professional incentives, existing software is less likely to be maintained or updated following new

methodological advancements. Third, the underprovision of this public good increases the costs and fragility of research. It forces many researchers to constantly “reinvent the wheel” by writing error-prone code to execute the same common tasks.

We propose a partial solution to this problem: journal editors should require the inclusion of a software bibliography in every article that they agree to publish.¹ To facilitate this, we introduce `softbib`, an R package that can scan analysis scripts, detect the software used in those scripts, and automatically create bibliographies.² We hope that recognizing the contribution of software developers to science will encourage more scholars to create public goods, which could yield important downstream benefits. We also suggest other improvements to the status quo that when combined with requiring citations to software could ameliorate the problems documented in this article.

BENEFITS OF RESEARCH SOFTWARE

In our view, most political scientists should seek to limit the number of lines of code they write for many of their research projects. Instead, they should use (and contribute to) open-source, publicly accessible, well-tested, and well-documented research software. Using such code has many benefits.³

First, research software can improve the reliability of our scientific findings. A common source of errors in research is programming mistakes made by a researcher.⁴ This is not surprising: any nontrivial software will include mistakes, and scientists are not necessarily programmers by profession. To be sure, research software is not immune to bugs either. However, when many researchers use the same code, there are more opportunities

Vincent Arel-Bundock  is an associate professor of political science at the Université de Montréal. He can be reached at Vincent.arel-bundock@umontreal.ca.

Joshua McCrain  is an assistant professor of political science at the University of Utah. He can be reached at Joshua.mccrain@utah.edu.

© The Author(s), 2023. Published by Cambridge University Press on behalf of the American Political Science Association. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (<http://creativecommons.org/licenses/by/4.0>), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

to detect and correct problems. This is doubly true in the case of open-source software, in which the code can be inspected freely. Moreover, good-quality software tends to be developed with extensive unit tests and detailed documentation. Research software thus can mitigate some of the scientific errors caused by the inadvertent but unavoidable coding mistakes found in almost any complex bespoke code base.

Second, research software is convenient, and it makes us more efficient. Many packages are simply tools that permit easy access to complex, difficult-to-use datasets—or just simple functionality

...journal editors should require the inclusion of a software bibliography in every article that they agree to publish.

that allows users to not re-download a commonly used dataset every time they want to use it. Producing tables and figures is similarly valuable: it allows us to iterate over several complex analyses and to produce aesthetically pleasing reports quickly while avoiding transcription errors. Producing tools to format results into common formats, without extensive programming knowledge, greatly accelerates the production of research.

Third, research software is empowering. Only a small proportion of political scientists have the technical skills to implement cutting-edge methods. When researchers do implement such methods, they risk making errors that may be subtle, difficult to check, and impossible to detect during peer review. Most researchers do not need to understand the exact programming concepts that underlie the implementation of the statistical method they need to use. In fact, some of the skills needed to develop well-tested, documented, and user-friendly software are orthogonal to the actual statistical knowledge needed to develop or apply statistical techniques. Therefore, empowering scientists to use cutting-edge methods must be viewed as part of a true community effort, involving those who invent statistical techniques, those who implement them, and those who apply them.

...the current norm in political science is to largely ignore the work of those who create the tools that make research possible. To illustrate this, we conducted full-text searches in all articles published in American Political Science Review between 2010 and 2021.

UNDERACKNOWLEDGMENT OF RESEARCH SOFTWARE

Although there are clear benefits to using research software, there also are few incentives for its creation. Those who choose to create these public goods face high costs through development, testing, documenting, bug triage, feature request implementation, and user support. Whereas the costs of production are high, the rewards for software contributors remain very low.

Indeed, the current norm in political science is to largely ignore the work of those who create the tools that make research possible. To illustrate this, we conducted full-text searches in all articles published in *American Political Science Review* between 2010 and 2021.⁵ We used the `readtext` package to read individual articles in PDF format and the `quanteda` package for R to extract all sentences that matched one of the following case-insensitive expressions: CRAN, github, library, package, SAS, software, SPSS,

Stata, R Core Team, R Foundation (Benoit and Obeng 2021; Benoit et al. 2018; R Core Team 2022). A human coder then read each sentence to cull false positives and to classify software references as “in text” or “in bibliography.” A second coder reviewed all classification and exclusion decisions.

We found that of the 804 articles in our sample, less than 25% gave credit to software developers by formally citing them in the bibliography. Among these papers, the majority included references to only one or two pieces of software—and a considerable proportion of those were self-citations. A substantial percentage of

articles (35%) mentioned the name of a software package in the text without formal attribution in the bibliography. This included many generic mentions of R, Stata, and Python, among others. Finally, more than 40% of authors did not even mention the software they had used.⁶

It is difficult to obtain reliable data on the popularity of different software packages because download statistics are a flawed measure of use and usefulness.⁷ Nevertheless, it is easy to find examples of packages developed by political scientists, which are popular and downloaded often but rarely cited: `panelView`, with zero citations (Liu, Xu, and Xu 2018); `interplot`, with three citations (Solt, Hu, and Kenkel 2021); and `countrycode`, a popular tool in international relations and comparative politics that was cited 65 times but only once in a political science peer-reviewed journal (Arel-Bundock, Enevoldsen, and Yetman 2018).⁸

Anecdotally, we found that software that tends to generate an appreciable number of citations usually accompanies new statistical methodology papers or is used in disciplines other than political science, in which norms around attribution are friendlier to software developers. For example, consider the `mediation`

package (Tingley et al. 2014), with more than 2,200 citations, and the text-analysis packages `stm` (Roberts, Stewart, and Tingley 2019) and `quanteda` (Benoit et al. 2018), which have generated more than 1,000 and 700 citations, respectively. These success stories are encouraging, but we want to see more (and different types of) software be recognized by our community: that is, code that helps us to access data, manipulate it, estimate models, and communicate results.

Of course, not every piece of code deserves to be cited; much software is published but never used. As with other research output, we should expect the distribution of software citations to be skewed and right tailed. Nevertheless, our exploration of citation patterns in a flagship journal shows that the current norms in the discipline are problematic, and it suggests that they could lead to the underprovision of an important public good.

MANDATORY SOFTWARE BIBLIOGRAPHIES

A partial solution to this problem is to leverage the existing incentive structure of academia and to ensure that researchers who write software receive proper credit for their contributions to science (i.e., software citations). Specifically, journals editors should make it mandatory for authors to include a software bibliography in every article that is accepted for publication. By including a full list of references for the primary software and its dependencies, authors would properly give credit to the contributions that all categories of software make to the different stages of the research process. In addition to giving credit for public-goods provision, these software bibliographies could play a useful role in promoting the reproducibility of research by consigning a permanent record of software versions used in a given project. This could complement more comprehensive strategies, such as containerization.

Two objections to our proposal may be that software bibliographies would constrain the word-count limits imposed on print articles and that preparing them would impose undue costs on authors and copyeditors. We address each concern in turn.

First, publishers may be concerned that including a complete software bibliography would increase the word count of articles. In principle, this constraint should apply only to the print versions of articles; however, given the current state of academic publishing, the length of the bibliography may be a concern for some publishers. When the constraint is binding, we recommend a simple three-step approach: (1) include a limited number of references to the primary tools in the main bibliography (i.e., a subjective call by the researcher); (2) archive a full software bibliography along with the article's data-replication package; and (3) include the full software bibliography in the primary PDF of manuscripts archived on preprint servers (e.g., arXiv and the Open Science Framework). This approach involves a judgment call by researchers, but would move in the direction of more recognition for software contributions. Archiving the full list of software used in a preprint repository also would allow Google Scholar to crawl the full bibliography and count citations.

The second potential objection to a mandatory software bibliography is that it takes time to prepare. Of course, the time it takes to add a few references to a bibliography typically is shorter than the time the software has saved for researchers; however, this observation does not invalidate the concern. To address the problem more directly, the next section introduces `softbib`, an R package that automates the creation of software bibliographies.⁹

SOFTBIB: AUTOMATED SOFTWARE BIBLIOGRAPHIES IN R

`softbib` is an R package that combines commands from other existing packages to automate the production of software bibliographies.¹⁰ `softbib` works in three steps. First, the `renv` package is

By typing nine characters in their R session—`softbib()`—authors can automatically obtain bibliographies in PDF, Microsoft Word, RMarkdown, and BibTeX formats. `softbib` also supports the Citation Style Language (CSL) standard, which means that bibliographies can be formatted in thousands of different styles.¹¹ This solution is almost costless for researchers, reviewers, and publishers.

COMPLEMENTARY REFORMS

Mandatory software bibliographies are by no means a panacea to the problem of undercitation that we documented. We believe that facilitating and requiring software bibliographies will improve the status quo, but we also recommend complementary reforms. First, many political science journals already require a listing of basic software used in the replication stage.¹² A possible addition to this process is that the replication officer determines that software is cited appropriately in either the main list of references or the supplementary information. This is not dissimilar to, for instance, journals checking for Institutional Review Board documentation or information on funding sources.¹³ We envision two primary issues with this reform. First, there is still some subjectivity (as mentioned previously) in *which* software *must* be cited—and, at this stage, the replication officer would be responsible. However, given the technical skills needed by replication officers to do their primary task, we also suggest that they may be best positioned to know which software *must* be cited. Second, this reform might marginally increase the duration of the replication process. We cannot state definitively the degree to which this is a concern, but journals would have to make adjustments after implementation.

Next—and, again, complementary to requiring software bibliographies—is journals offering flexibility in how bibliographies are counted toward article length and word counts. On the one hand, authors already are attentive to not overciting existing literature due to these binding constraints at many journals. Thus, requiring authors to also include citations to key software used in their work is not that different than strategically citing the most important literature on which their research builds. In other words, there is no inherent reason to think that citing relevant literature is any more or less important than citing the tools used to conduct the analyses. On the other hand, we recognize that enforcing authors to limit citations due to length constraints has negative externalities on the discipline (e.g., Dion, Sumner, and Mitchell 2018). For this reason, it may be strictly better for journals, for instance, to not count bibliography length toward article length. Ultimately, however, we suggest that a net improvement would be to require the software citations at the very least in the online materials submitted for publication—similar to the supplemental information and/or pre-analysis plans.¹⁴

Political scientists should use research software. Political scientists should write research software. Political scientists should cite research software.

used to scan the contents of a project directory to identify which software is used in the analysis scripts (Ushey 2022). Second, the `bibtex` package parses and formats the bibliographic entries for each package in use (Francois 2020). Third, the `rmarkdown` package creates bibliographies in several formats (Allaire et al. 2022).

CONCLUSION

This article contends that the contribution of software to political science is underrecognized and that this situation likely leads to an underprovision of public goods. Political scientists should use research software. Political scientists should

write research software. Political scientists should cite research software.

This article focuses on software bibliographies as a partial solution to the problem of attribution and credit claiming, but this is only a first step. As a discipline, we also should consider other more institutional mechanisms to encourage our colleagues to write and publish more free software. For instance, we could create more prizes like the Statistical Software Award of the Society for Political Methodology¹⁵; encourage hiring, tenure, and promotion committees to recognize the importance of software contributions; and establish new sources of funding to hire research-software professionals. Ultimately, we recognize that our proposed solution, including the developed package, is not going to fully solve this problem. There still will be subjectivity, and journals will need to flexibly adjust their policies. We believe that policies along the lines that we are recommending, facilitated by the accompanying package, are a net improvement toward increasing incentives for the creation of public goods.

ACKNOWLEDGMENTS

We thank Gabrielle Boyer for research assistance and Marco Mendoza Aviña, Ryan Briggs, and Arthur Spirling for helpful comments. We are grateful to three anonymous reviewers and the editor, Justin Esarey, for thoughtful comments and suggestions.

DATA AVAILABILITY STATEMENT

Research documentation and data that support the findings of this study are openly available at the *PS: Political Science & Politics* Harvard Dataverse at <https://doi.org/10.7910/DVN/PYKIUN>.

CONFLICTS OF INTEREST

The authors declare that there are no ethical issues or conflicts of interest in this research. ■

NOTES

1. A second-best solution includes a software bibliography in supplementary materials as well as in preprint manuscripts archived by public repositories such as arxiv.org and osf.io. Because preprints typically are indexed by Google Scholar, references would be counted in that site's citation metrics. A downside of this approach is that software citations still would not be counted by other providers of citation statistics.
2. See <https://github.com/vincentarelbundock/softbib>.
3. Researchers must remember that a tradeoff comes with using libraries instead of hand-coding procedures: this increases code dependencies and may hamper reproducibility in the long run.
4. Famous examples include Reinhardt and Rogoff, <https://academic.oup.com/cje/article-abstract/38/2/257/1714018>, and Excel autocorrect error in genomics. "Autocorrect errors in Excel still creating genomics headache: 30% of published papers contain mangled gene names in supplementary data." See www.nature.com/articles/d41586-021-02211-4.
5. The sample also includes articles that were published online as "First View" at the time of data collection.
6. There are many software packages designed for qualitative scholars, such as QCA (Thiem and Dusa 2013) and corporaexplorer (Gjerde 2019).
7. In the R world, for example, the RStudio company publishes download statistics for packages published on the CRAN repository. However, the published statistics consider only the downloads from a small subset of CRAN servers (i.e., mirrors), and they do not allow us to identify individual users, users who

8. have reinstalled a package multiple times, or automated installations made in the context of continuous integration testing frameworks.
9. Citation statistics were obtained from Google Scholar on September 29, 2022.
10. Currently, the software that we developed supports only the creation of bibliographies for researchers who use R. Similar software solutions could be developed relatively easily for other programming environments, such as Stata, SPSS, and Python.
11. `softbib` is free and open source. See <https://github.com/vincentarelbundock/softbib>.
12. CSL-style files can be downloaded freely from the Zotero Style Repository. See www.zotero.org/styles.
13. These requirements are highly idiosyncratic to a specific journal's replication process and often do not include the listing of specific packages. Furthermore, we are not aware of any replication guidelines that require the citations to the software and/or packages used in the replication archive.
14. We are grateful to the editor for this idea.
15. According to its own documentation, Google Scholar is most likely to pick up a bibliography of software citations when it is included in the main paper. However, Google Scholar indexes any formatted citations that are provided in standard formats and linked from websites—exactly what happens when journals provide direct links to online information on the pages of the published articles. For more detail, see <https://scholar.google.com/intl/en/scholar/inclusion.html#overview>. A separate but important question beyond the scope of this article is the degree to which our profession relies on Google Scholar.
16. See <https://polmeth.org/statistical-software-award>.

REFERENCES

- Allaire, Joseph J., Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2022. *RMarkdown: Dynamic Documents for R*. R Package Version 2.16. <https://github.com/rstudio/rmarkdown>.
- Arel-Bundock, Vincent, Nils Enevoldsen, and CJ Yetman. 2018. "Countrycode: An R Package to Convert Country Names and Country Codes." *Journal of Open Source Software* 3 (28): 848. <https://doi.org/10.21105/joss.00848>.
- Benoit, Kenneth, and Adam Obeng. 2021. *readtext: Import and Handling for Plain and Formatted Text Files*. R Package Version 0.81. <https://github.com/quanteda/readtext>.
- Benoit, Kenneth, Kohei Watanabe, Haiyan Wang, Paul Nulty, Adam Obeng, Stefan Müller, and Akitaka Matsuo. 2018. "quanteda: An R Package for the Quantitative Analysis of Textual Data." *Journal of Open Source Software* 3 (30): 774. <https://quanteda.io>.
- Dion, Michelle L., Jane Lawrence Sumner, and Sara McLaughlin Mitchell. 2018. "Gendered Citation Patterns Across Political Science and Social Science Methodology Fields." *Political Analysis* 26 (3): 312–27.
- Francois, Romain. 2020. *BibTeX: BibTeX Parser*. R Package Version 0.4.2.3. <https://github.com/romainfrancois/bibtex>.
- Gjerde, Kristian Lundby. 2019. "corporaexplorer: An R Package for Dynamic Exploration of Text Collections." *Journal of Open Source Software* 4 (38): 1342.
- Liu, Licheng, Yiqing Xu, and Maintainer Yiqing Xu. 2018. "Package 'PanelView.'" <https://cran.r-project.org/web/packages/panelView/index.html>.
- McCraen, Joshua, and Vincent Arel-Bundock. 2023. "Replication Data for 'Software Citations in Political Science.'" *PS: Political Science & Politics*. DOI:10.7910/DVN/PYKIUN.
- R Core Team. 2022. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. www.R-project.org.
- Roberts, Margaret E., Brandon M. Stewart, and Dustin Tingley. 2019. "stm: An R Package for Structural Topic Models." *Journal of Statistical Software* 91:1–40.
- Solt, Frederick, Yue Hu, and Brenton Kenkel. 2021. "Package 'InterPlot.'" <https://cran.r-project.org/web/packages/interplot/vignettes/interplot-vignette.html>.
- Thiem, Alrik, and Adrian Dusa. 2013. "QCA: A Package for Qualitative Comparative Analysis." *The R Journal* 5 (1): 87. DOI:10.32614/RJ-2013-009.
- Tingley, Dustin, Teppei Yamamoto, Kentaro Hirose, Luke Keele, and Kosuke Imai. 2014. "Mediation: R Package for Causal Mediation Analysis."
- Ushey, Kevin. 2022. *renv: Project Environments*. R Package Version 0.15.5. <https://rstudio.github.io/renv>.