JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

pubs.acs.org/jcim Viewpoint

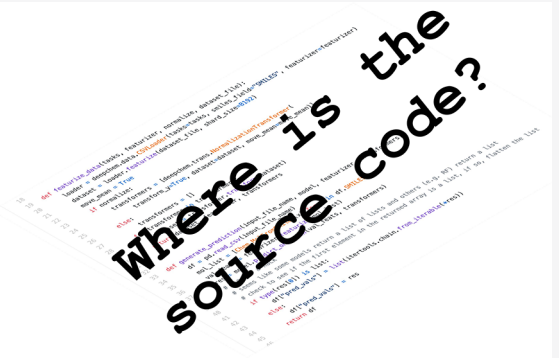
Code Sharing in the Open Science Era

W. Patrick Walters*

Cite This: J. Chem. Inf. Model. 2020, 60, 4417−4420 Read Online

ACCESS | Metrics & More | Article Recommendations

ABSTRACT: Many high-profile scientific journals have established policies mandating the release of code accompanying papers that describe computational methods. Unfortunately, the majority of journals that publish papers in Computational Chemistry and Cheminformatics have yet to define such guidelines. This Viewpoint reviews the current state of reproducibility for the field and makes a case for the inclusion of code with computational papers.



INTRODUCTION

For hundreds of years, reproducibility has been a fundamental tenet of scientific publication. In the experimental sciences, journals require a paper to contain a detailed experimental section that serves two purposes—to demonstrate that the results obtained are genuine and to provide a step by step procedure that will enable others to reproduce and extend the work. While reproducibility is considered essential in experimental disciplines, it is often overlooked in papers describing computational methodologies. This seems odd. In most cases, it is relatively easy to make the source code and the associated validation data for a new method available via code sharing websites such as GitHub. While it is true that papers in Molecular Modeling and Cheminformatics often do not include source code, this is not the case for most other computational disciplines. In fields such as Bioinformatics, Statistics, and Computer Science, the inclusion of source code with papers, and even with conference proceedings, has become the norm. In fact, the highly regarded Neural Information Processing Systems (NeurIPS)[1] conference has created a Code Submission and Reproducibility Policy along with a checklist for included source code.

We have reached a point where technical barriers to the reproducibility of computational methods no longer exist. As a community, we need to take advantage of our ability to share code on the Internet and ensure that the readers of our papers can easily reproduce our results. As the first step in this process, journals such as this one, that publish Computational Chemistry and Cheminformatics methods, need to establish guidelines to ensure that computational methods can be easily reproduced.

APPROACHES TO COMPUTATIONAL REPRODUCIBILITY

In recent years, many prominent journals have taken steps toward ensuring the reproducibility of computational methods. The editors of *Science*[2] have developed a policy requiring the release of code for computational papers.

We require that all computer code used for modeling and/or data analysis that is not commercially available be deposited in a publicly accessible repository upon publication. In rare exceptional cases where security concerns or competing commercial interests pose a conflict, code-sharing arrangements that still facilitate reproduction of the work should be discussed with your Editor no later than the revision stage.

Nature[3] is a bit less prescriptive, but still strongly urges the release of source code.

Upon publication, Nature Research journals consider it best practice to release custom computer code in a way that allows readers to repeat the published results. We also strongly encourage authors to deposit and manage their source code revisions in an established software version control repository, such as GitHub, and to use a DOI-minting repository to uniquely identify the code and provide details of its license of use. We strongly encourage the use of a license approved by the open source initiative. The license of use and any restrictions must be described in the Code Availability statement.

Received: August 26, 2020
Published: September 16, 2020

https://dx.doi.org/10.1021/acs.jcim.0c01000
J. Chem. Inf. Model. 2020, 60, 4417−4420

At least one journal in our field is making an effort to ensure that the methods described in papers can be easily reproduced. The *Journal of Cheminformatics*[4] has the following as part of its editorial policy.

If published, the software application/tool should be readily available to any scientist wishing to use it for non-commercial purposes, without restrictions (such as the need for a material transfer agreement). If the implementation is not made freely available, then the manuscript should focus clearly on the development of the underlying method and not discuss the tool in any detail.

To date, it appears that the other journals in our field, *Journal of Chemical Information and Modeling* (JCIM), *Molecular Informatics*, *Journal of Computer-Aided Molecular Design*, *Journal of Chemical Theory and Computation*, and *Journal of Molecular Graphics and Modeling* have yet to define editorial policies for reproducibility or source code inclusion. JCIM has made some steps in the right direction. As part of the submission process, authors are asked whether the manuscript relies on software that is not generally available. While this is a necessary first step, it does not appear that answers to this question are impacting editorial decisions.

There are two approaches to this issue: the carrot and the stick. As can be seen above, some journals have used the stick and implemented guidelines that mandate the inclusion of source code. Another approach would be to provide specific recognition for papers that include source code. This could be done by attaching a badge or banner in the table of contents. This badge, which will hopefully become something that readers and authors value, could also be a valuable indicator for those who may have to pay to download an article. Another method, which has been adopted by a number of journals is the inclusion of an "Availability of Materials" section in each paper. In this section, authors must either provide pointers to code repositories or clearly justify their reasons for not including code.

In a 2019 perspective,[5] Robert Clark made the point that there is much to be gained from implementing new methods based on descriptions in a paper's Methods section. While it is true that implementation provides one of the best means of understanding an algorithm, there are a number of other factors that should be considered. First, the descriptions of computational methods in many papers are ambiguous or incomplete. Reviewers often do not have the time or the necessary skill to assess whether the descriptions in a paper are sufficient to enable others to reproduce the method. Even if the description is complete, one often needs access to a "gold standard" implementation to ensure that the reimplementation is generating the correct results. In many cases, the reader of a paper may simply want to compare the results of a new method with those generated using an existing technique. Asking the reader to reimplement a method in order to perform a simple comparison seems a bit extreme and may be difficult for those with limited programming skills.

POINT/COUNTERPOINT

Ultimately this is a societal issue as well as a scientific one. As a community, we need to decide what is acceptable for publication and what is not. Of course, there are multiple aspects to this issue, and others have proposed a number of arguments against requiring the inclusion of source code with computational papers. In this section, I review some of these arguments and provide counterpoints.

- **Requiring code will prevent those in industry from publishing**—Similar arguments were made 25 years ago when Biophysics journals began to require crystallographic coordinates to be deposited in the Protein Data Bank (PDB) prior to publication.[6] In the intervening years, we have not seen a reduction in the number of publications from the industry and depositing data in the PDB has become the norm. One area that has benefitted from the sharing of source code is the application of Artificial Intelligence (AI) in drug discovery. Probably due to the fact that the field had its origins in Computer Science, where publications are typically accompanied by code, many authors have provided code repositories to accompany their publications. These publications have not solely come from academic groups. Smaller biotechnology companies such as Benevolent AI[7] and InSilico Medicines,[8] as well as larger pharmaceutical companies like AstraZeneca,[9,10] Bayer,[11] and Janssen,[10,12] have made a practice of releasing code with their papers.
- **Requiring code will prevent software companies from publishing**—It is legitimate to argue that the inclusion of code with papers could negatively impact a software company's finances. It is possible that journals could make provisions to accommodate commercial software. Companies could provide time-limited evaluation licenses that would enable others to evaluate new methodologies and make the necessary comparisons. Given the ability of software vendors to easily post white papers describing advances and reporting benchmarks, one has to ask whether the scientific literature is still the most appropriate venue for these sorts of papers.
- **Code will quickly go out of date and will have limited value**—It is true that we are in the midst of a rapidly evolving computational ecosystem that puts little emphasis on backward compatibility. Today's state of the art deep learning infrastructure is tomorrow's dinosaur. While it is true that today's code may not be easy to run a few years from now, there are still benefits to code release. If the software enables them to perform a useful task, a motivated user will figure out how to port it to a new environment. Ultimately, the value of a software package may be determined by the motivation of a community to maintain it.
- **New methods can be made available via a web interface**—It has become common for academic groups to make their methods available through web interfaces. This approach has the advantage that users do not have to install software, and a user-friendly web interface can make it easier for some to use the software. While a web interface can enable some degree of method validation, it falls short in a couple of areas. First, it is difficult, if not impossible, to learn from or build upon software that is only released through a web interface. Second, the use of a web portal is often not an option for scientists working in industry who are unable to disclose proprietary chemical structures. A web interface is a valuable public service, but it should not be considered a substitute for source code.
- **Code will be made available on request for non-commercial use**—There are a couple of factors that complicate this approach. The first is that authors of the paper and the software have to be motivated to distribute their code. Many potential users, including myself, have been frustrated by authors who never respond to email

https://dx.doi.org/10.1021/acs.jcim.0c01000
J. Chem. Inf. Model. 2020, 60, 4417−4420

requests. The second factor is that the definition of commercial use has changed dramatically over the past few years. Many academic institutions are employing aggressive intellectual property strategies and are licensing their discoveries to commercial entities. Do these institutions qualify as noncommercial?

- **We are not professional programmers**—Many computational groups focus more on science than they do on software development. As such, some groups may be reluctant to release code that is not "polished" to professional standards. While the public release of code may appear daunting, it is a great motivation to do the necessary "house cleaning" and make the code more robust. The public release of code also tends to point out limitations and ultimately makes the code better. Fortunately, repositories like GitHub have builtin issue trackers that enable others to report, and sometimes fix, software bugs.
- **We are competing for grants, and releasing code will give our competitors an unfair advantage**—This argument seems to defy the fundamental purposes of scientific publication and collaboration. It is puzzling that many funding agencies still do not have specific guidelines for the release of software written as part of research funded by a grant.
- **The code relies on large amounts of internal infrastructure, which cannot be released**— Some organizations have a large body of legacy code on which their research is based. While it may be difficult or impossible to release the surrounding infrastructure, it is often relatively easy to reimplement a method using a publicly available software toolkit. One has to ask whether a reader could reproduce the work if the author cannot provide a publicly accessible version of the software.

CONCLUSIONS

This is not a new issue for our field. I published a perspective similar to this one in 2013.[13] In the intervening years, our field has made some progress. As mentioned above, in some factions of our discipline, there seems to be an implicit pressure to release code. However, this is far from uniform, and there are still many papers describing applications of AI in drug discovery that do not include code. As a scientist, I have found open source software to be key to my work. The ability to download a new method, try it on data sets that I know well, and understand its scope and limitations is incredibly powerful. Studying a method by reading the code can often provide insights that can be difficult to glean from even the most well-written description. As someone who has been writing and releasing code for more than 30 years, I get tremendous satisfaction from the fact that others can learn from and hopefully build upon code that I have written. It is exciting to imagine a world where I could read any paper, download the corresponding software, and immediately try the method on my favorite data set.

We have reached a point where the technical aspects of code sharing have been dramatically simplified. It is trivial to set up a repository and make code available on GitHub or similar sites. These sites also make it easy to track changes and release updated versions. Now that the technical limitations have been removed, the only barriers appear to be social and institutional. Collaboration continues to be a critical component of scientific discovery. Open source code makes it easy to collaborate across institutions and continents. It is incumbent not only on journals but on the community to promote this type of collaboration.

Of course, it is easy to complain about the current state of our field and much more difficult to take concrete action. I would suggest the following to the editors of JCIM and hope that other journals will take similar action.

(1) Institute an immediate policy of specifically highlighting papers that include source code.
(2) Include an explicit "Availability of Code and Data" section in every paper. In this section, authors will either provide links to code and data repositories or provide a clear rationale for not including certain materials.
(3) Solicit input from the community and develop a clear policy for reproducibility and inclusion of source code with papers. The policies of numerous other journals such as the *Journal of Cheminformatics* will provide excellent starting points.

There are also a number of topics around data sharing that need to be addressed, but that is a subject for another day.

AUTHOR INFORMATION

Corresponding Author

W. Patrick Walters − *Relay Therapeutics, Inc., Cambridge, Massachusetts 02142, United States;* orcid.org/0000-0001-5528-8839; Email: pwalters@relaytx.com

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.0c01000

Notes

The author declares no competing financial interest.

ACKNOWLEDGMENTS

I would like to thank my friends and colleagues for their feedback on earlier versions of this manuscript. I would also like to thank numerous Twitter users and readers of Practical Cheminformatics[14] for lively discussions on this important topic.

REFERENCES

(1) *NeurIPS 2020 Code Submission Policy*. https://nips.cc/Conferences/2020/PaperInformation/CodeSubmissionPolicy (accessed May 17, 2020).
(2) *Science Journals: Editorial Policies*. https://www.sciencemag.org/authors/science-journals-editorial-policies (accessed May 17, 2020).
(3) *Nature Journals: Reporting Standards and Availability of Data, Materials, Code and Protocols*. https://www.nature.com/nature-research/editorial-policies/reporting-standards (accessed May 17, 2020).
(4) *BioMed Central: Editorial Policies*. biomedcentral.com/getpublished/editorial-policies (accessed May 17, 2020).
(5) Clark, R. D. A Path to next-Generation Reproducibility in Cheminformatics. *J. Cheminf.* **2019**, *11* (1), 62.
(6) Berman, H. M.; Lawson, C. L.; Vallat, B.; Gabanyi, M. J. Anticipating Innovations in Structural Biology. *Q. Rev. Biophys.* **2018**, *51*, No. e8.
(7) Brown, N.; Fiscato, M.; Segler, M. H. S.; Vaucher, A. C. GuacaMol: Benchmarking Models for de Novo Molecular Design. *J. Chem. Inf. Model.* **2019**, *59* (3), 1096−1108.
(8) Polykovskiy, D.; Zhebrak, A.; Sanchez-Lengeling, B.; Golovanov, S.; Tatanov, O.; Belyaev, S.; Kurbanov, R.; Artamonov, A.; Aladinskiy, V.; Veselov, M.; Kadurin, A.; Nikolenko, S.; Aspuru-Guzik, A.; Zhavoronkov, A. Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models. *arXiv:1811.12823v4*, 2018.

(9) Olivecrona, M.; Blaschke, T.; Engkvist, O.; Chen, H. Molecular de-Novo Design through Deep Reinforcement Learning. *J. Cheminf.* **2017**, *9* (1), 48.
(10) Sturm, N.; Mayr, A.; Le Van, T.; Chupakhin, V.; Ceulemans, H.; Wegner, J.; Golib-Dzib, J.-F.; Jeliazkova, N.; Vandriessche, Y.; Böhm, S.; Cima, V.; Martinovic, J.; Greene, N.; Vander Aa, T.; Ashby, T. J.; Hochreiter, S.; Engkvist, O.; Klambauer, G.; Chen, H. Industry-Scale Application and Evaluation of Deep Learning for Drug Target Prediction. *J. Cheminf.* **2020**, *12* (1), 26.
(11) Winter, R.; Montanari, F.; Noé, F.; Clevert, D.-A. Learning Continuous and Data-Driven Molecular Descriptors by Translating Equivalent Chemical Representations. *Chem. Sci.* **2019**, *10* (6), 1692−1701.
(12) Aa, T. V.; Chakroun, I.; Ashby, T. J.; Simm, J.; Arany, A.; Moreau, Y.; Le Van, T.; Dzib, J. F. G.; Wegner, J.; Chupakhin, V.; Ceulemans, H.; Wuyts, R.; Verachtert, W. SMURFF: A High-Performance Framework for Matrix Factorization. *arXiv:1904.02514v3*, 2019.
(13) Walters, W.; Modeling, P. Informatics, and the Quest for Reproducibility. *J. Chem. Inf. Model.* **2013**, *53* (7), 1529−1530.
(14) Practical Cheminformatics. https://practicalcheminformatics.blogspot.com (accessed May 17, 2020).

https://dx.doi.org/10.1021/acs.jcim.0c01000
J. Chem. Inf. Model. 2020, 60, 4417−4420