

Reproducible Research for Scientific Computing: Tools and Strategies for Changing the Culture

This article considers the obstacles involved in creating reproducible computational research as well as some efforts and approaches to overcome them.

“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.”

—Jonathan Buckheit and David Donoho,
paraphrasing Jon Claerbout¹

It’s increasingly recognized that computational science is facing a credibility crisis: it’s impossible to verify most of the computational results presented at conferences and in papers today.² We believe that addressing this credibility crisis requires a change in the culture of scientific publishing. However, publishing truly reproducible research isn’t a new idea. Our opening quote dates from 1995, and it paraphrases efforts dating back more than 20 years ago at the lab of Stanford University geosciences professor Jon Claerbout (see <http://sepwww.stanford.edu/sep/jon/reproducible.html>). Here we give a brief overview of some of the issues concerning reproducibility in this field, and summarize a workshop and community forum held in Vancouver in July 2011 on this topic. Other articles in this special issue grew out of talks from that workshop, as summarized in the guest editor’s introduction.

The Need for Reproducibility

The notion of reproducibility as a scientific standard began with Robert Boyle and discussions

within the Invisible College in the 1660s. The extensive use of computation in scientific discovery affects the implementation of these standards: Parameter values, function invocation sequences, and other computational details are typically omitted from published articles but are critical for replicating results or reconciling sets of independently generated results. Consequently, researchers from fields as diverse as geoscience, neuroscience, bioinformatics, applied mathematics, psychology, and computer science are calling for data and code to be made available in such a way that published computational results can be conveniently reproduced.³

A number of recent workshops, conference sessions, and committee reports have been devoted to this topic. To choose just a few examples, the annual Society for Industrial and Applied Mathematics (SIAM) Computational Science and Engineering conference featured a multispeaker session on reproducible research

1521-9615/12/\$31.00 © 2012 IEEE
COPUBLISHED BY THE IEEE CS AND THE AIP

RANDALL J. LEVEQUE

University of Washington

IAN M. MITCHELL

University of British Columbia

VICTORIA STODDEN

Columbia University

in 2011; a panel discussion at the International Biometric Society meeting in 2011 featured reproducibility; and the 2011 SIAM Conference on Mathematical and Computational Issues in the Geosciences held a session on reproducible research (see <http://jarrodmillman.com/events/siam2011.html> and www.siam.org/meetings/gs11). In 2010, the Institute of Medicine convened a committee called “Review of Omics-Based Tests for Predicting Patient Outcomes in Clinical Trials” to examine publication standards for computational work that leads to clinical trials. In March of 2012 the committee released a report recommending the release of the software and data underlying the findings (see <http://iom.edu/Activities/Research/OmicsBasedTests.aspx>).

Recently, prestigious journals such as *Science*⁴ and the *Proceedings of the National Academy of Sciences* (PNAS) have made data and code disclosure a requirement for publication (see www.pnas.org/site/misc/iforc.shtml#submission for PNAS’s data availability requirements). The machine learning community in computer science and statistics has created a platform for data sharing called Machine Learning Open Source Software (MLOSS.org), and Kitware provides open source software for the neuroscience community. In 2009, community members from bioinformatics, applied mathematics, computer science, law, and many other fields came together to create a declaration on data and code sharing in support of reproducible computational research, outlining steps forward.³ Funding agencies have also joined the discussion; for example, the National Science Foundation (NSF) created the Sustainable Digital Data Preservation and Access Network Partners (DataNet) program to provide an infrastructure for data-driven research in 2007, and they added a data management plan requirement for all grant applications in 2011.

These different approaches arise in part because the best way to share data and code depends on the research context. Different scientific communities use different software and hardware, structure and access their data in different ways, and use software that varies from short scripts thrown together for one-off tasks to complex combinations of packages developed over decades and containing millions of lines of code. Beyond this, different research areas face different pressures to commercialize aspects of the research, diverse modalities, norms, and constraints on code or data sharing, and different degrees of training in methods that enable effective and efficient sharing (such as documentation, version control, shell scripting, and repository use).

In a recent survey of the machine learning community—a community that is generally well-informed about tools and techniques for software and data management—respondents reported that the single biggest barrier to sharing code and data was the time it takes to clean up and document the work to prepare it for release and reuse (56 percent of respondents cited this reason for not sharing data and 78 percent cited this reason for not sharing code).⁵ Preparation time was also cited as a significant barrier to data sharing in a broader survey of scientists.⁶ One approach to reducing this barrier is to develop tools that more easily capture experimental details and facilitate the communication of the environment, algorithm, data, and reasoning to collaborators and the public when findings are published.

The Workshop

The articles in this special issue came out of a “Reproducible Research: Tools and Strategies for Scientific Computing” workshop we organized in July 2011 at the University of British Columbia as a part of Applied Mathematics Perspectives, a satellite conference to the International Congress on Industrial and Applied Mathematics (ICIAM 2011).⁴ The workshop was sponsored by the Canadian Applied and Industrial Mathematics Society (CAIMS/SCMAI), the Pacific Institute of Mathematical Sciences (PIMS), the Banff International Research Station (BIRS), Mitacs, and the NSF. The goal was to bring together scientists and software developers who’ve created approaches to support reproducible research in the computational sciences and thereby encourage this nascent community.

Day one of the workshop included tutorials on version control, testing, documentation, and intellectual property issues. Days two and three consisted of a series of 14 talks by invited speakers. All of these talks were videotaped, and high-quality recordings with the accompanying slides are available at <http://stodden.net/AMP2011>, which also contains the abstracts. The fourth and final day of the workshop included additional tutorials on the tools that speakers presented and an opportunity to experiment with them. A community forum on the final evening focused on policy issues and the role of journals and funding agencies (which we discuss further in the next section).

Three themes emerged from the workshop talks. The first, and perhaps the one that’s primarily driving the need for improved reproducibility, is the changing nature of science as the quantity of available data and processing power drives a

shift to computational and data-driven modes of discovery.⁷

The second theme is the challenge of defining, interpreting, reducing barriers to, improving incentives for, and providing examples of reproducible research in various research fields. As an example of the lack of a common nomenclature, two sequential speakers provided opposite definitions for *replicable* and *reproducible*. (We believe the first refers to the ability to run a code and produce exactly the same results as published, and the second refers to the ability to create a code that independently verifies the published results using the information provided.⁸)

The third major theme—and the focus of this special issue—is the development of tools and best practices for reproducibility. This requires capturing the computational environment (What executables and libraries were used?), the provenance (What source code versions, execution parameters, and datasets were used?), and the scientific narrative (Why were these particular choices made?). It's not yet clear how best to capture or present all of this information, but a number of interesting approaches were advanced, some of which are discussed in this special issue.

The Community Forum on Reproducible Research Policies

A community forum, funded in part by the Sloan Foundation and SIAM, was a unique part of the workshop. Held on the final day, it drew a broad and distinguished cross-section of the computational science community (see <http://faculty.washington.edu/rjl/rrforum>). Many people coming to the ICIAM meeting the following week made an effort to attend the forum—which brought together more than 40 researchers and stakeholders from editorial boards, funding agencies, and leadership positions in professional societies—to discuss policies that facilitate reproducible research. We divided the forum into two discussions: journal policy and funding agency policy, which we'll summarize here to the best of our ability.

Discussion 1: Journal Policy and Reproducible Research

The discussion on journal policies highlighted the spectrum of opinions in the community on how best to handle code and data that form the basis of the research behind journal publications. A minority took the view that the current model works fine, that most research codes don't need to be made public, and that releasing such codes, which might be poorly documented, badly written, or

just plain wrong, would be irresponsible without a level of review that's unlikely to be attained. Moreover, even correct code might be put to uses for which it wasn't intended, possibly with dangerous results. Others disagreed strongly and felt that only access to the code will reveal all of the details of the computation necessary to reproduce experiments or to allow the discovery of bugs that might affect the results.

Among those in favor of publishing code, there was no clear consensus on the role that traditional journals should play or the appropriate level of peer review for code and data. Although all agreed that code review is exceptionally difficult, some felt that this is a crucial part of the scholarship contained in a research paper in computational science, while others felt that this is beyond the scope of the traditional journal model and that other mechanisms must be found for code review. The point was raised that a distinction must be made between small codes that are relatively easy to verify and share, and large, evolving project-based codes. The current mission of scientific journals is to disseminate good science through the traditional form of an archival paper, and assigning new roles such as reviewing large-scale codes could dilute this goal. Requiring refereeing of code at any scale would almost certainly make it even harder to find a sufficient number of good referees.

As a possible alternative, participants suggested encouraging the development of open source software communities similar to those that exist outside of academia (such as the Mozilla and Linux kernel communities). These communities make heavy use of public code repositories with version control and issue tracking to rapidly identify and fix bugs. Some scientific software efforts have evolved in this direction as well, such as NumPy, SciPy, Octave, and Sage. However, concerns were raised that the scientific coding community's small size might not permit such a development for the specialized code that accompanies the average paper. Another possible barrier to the formation of such communities is that requiring an open source license could prevent some results from being published, and perhaps a new license is needed that permits inspection but restricts execution for published scientific codes.

Underlying the discussion was the broad agreement of the vital importance of appropriate citation when published code and data are reused. Such citation not only encourages the release of data and code but also generates a mechanism by which such contributions to science can be assessed. A major role of journals is to provide a time

THE NEXT STEPS

Without concerted effort and broad agreement on goals and procedures, both individual scientists and scientific institutions face considerable challenges and disincentives for implementing reproducible research. Nevertheless, we call upon all computational scientists to practice reproducibility, even if only privately and for the benefit of your current and future research efforts: use version control, write a narrative, automate your process, track your provenance, and test your code. Keep in mind during this process that reproducibility is not an all-or-nothing affair, but rather a social construct with a spectrum of meanings that supports a gradual learning curve. Furthermore, from private reproducibility it's only a small effort to achieve public reproducibility if circumstances warrant: simply release the code and data under a suitable license.

We also call upon all interested computational scientists to tackle institutional and community challenges. This effort can take a variety of forms—for example, train your students and postdocs in reproducibility, publish examples of reproducible research in your field, request code and data when reviewing, submit to and review for journals that

support reproducible research, critically review and audit data management plans in grant proposals, and consider reproducibility wherever possible in hiring, promotion, and reference letters. Such efforts convince our representatives at funding agencies, journal editorial boards, universities, and scientific societies that reproducibility is a worthwhile goal, and provide ammunition to bring these efforts to the attention of broader and higher audiences.

Last, we call upon all stakeholders to consider code a vital part of the digitization of science. A focus on data policies alone not only misses the unique features of code and its importance to reproducibility but fails to see that code is integral to all stages of data use. Digital datasets are not only analyzed by code, they're also deposited, made available, collated, filtered, and sometimes even created by code. An exclusive emphasis on open data is a missed opportunity to resolve the current credibility crisis facing computational science and engineering.

If we seek to elevate computation into a third pillar of the scientific method alongside theory and experiment, we must overcome relaxed attitudes toward reproducibility. Changing a culture isn't a simple task, but it can be accomplished through individual and small group efforts.

stamp and narrative for discoveries, and similar mechanisms are needed for code and data. There was also a related discussion, and agreement, on the vital importance of versioning for shared data and code, and the need for better infrastructure beyond the commercially supported hosting sites currently available.

Discussion 2: Funding Agency Policy and Reproducible Research

The discussion of funding agency policy began by reiterating the need for sustainable repositories for the long-term availability of code and data. Much of the discussion focused on the data management plan requirement recently introduced by the NSF that's required for all new NSF proposals. These plans could cover code as well as more traditional data used for funded research, although this requirement has been driven by the experimental sciences. Details of what's required have been left purposefully vague, providing an opportunity for the community to influence the expectations. The point was made that data and code management is often a long-term process that happens over decades, whereas funding provided in a grant might only last three years. There was also a discussion of ways in which grant agencies might better recognize the effort required to share code and data—for example, by encouraging the inclusion of software packages or databases along with

journal publications in the biosketches submitted with proposals.


The role of advocacy by computational scientists was then discussed. Legislative decisions are often influenced by special interest groups that might not be speaking for the interests of the broader scientific community. Regulatory agencies might not be impartial and could have interests in encouraging or limiting data exposure that differ from scientific interests, such as job creation or watchdog activities. Congress debates issues and passes regulations that affect the practice of science, often with little or no input from the computational science community. The forum discussion outlined a role for computational scientists in the debate about transparency and open data, and encouraged more involvement from the community.

The principal goal of these discussions and workshops is to develop publication standards akin to both the proof in mathematics and the deductive sciences, and the detailed descriptive protocols in the empirical sciences (the “methods” section of a paper describing the mechanics of the controlled experiment and hypothesis test). Computational science is only a few decades old and must develop similar standards, so that other researchers in the field can independently verify published results (see “The Next Steps” sidebar for

more information). The discussions on standards have bifurcated into two approaches: the first by those who see the issue as a dissemination of data for reuse and the second by those who see reproducibility as the driving concern, requiring the sharing of data and code for verification purposes. These two approaches could indicate different policy prescriptions and different scientific standards. We believe the second approach will best promote scientific progress, as it subsumes data sharing as part of reproducible publishing, rather than establishing open data as an end goal in itself.

An example of the first approach is NSF's Data-Net program, which is targeted at "creating a set of exemplar national and global data research infrastructure organizations" (see www.nsf.gov/funding/pgm_summ.jsp?pims_id=503141). Grantees have focused on understanding and improving the way scientists manage both large and small datasets. The emphasis on scientific data is important, given the rate at which we're collecting it and its centrality in the scientific method, but this focus short-changes computational science, which we define as scientific endeavors in which the software for generating or analyzing data is complex and evolving (computational and data science aren't mutually exclusive by this definition).

Some have argued that bytes are bytes and hence code is data; however, such a viewpoint ignores many important properties of software.⁹ After all, nobody stores books as a movie of the pages being turned. As an example of the unusual form of software as a type of data, consider that the metadata required to execute scientific code—in the form of the computing environment (such as libraries, compilers, the operating system, and hardware)—are often orders of magnitude larger than the scientific code itself.

As showcased by the "Reproducible Research: Tools and Strategies for Scientific Computing" workshop, a nascent and growing community of developers is providing tools and systems for sharing and maintaining academic codes and data. It's clear that open and reproducible science and engineering will need an integrated approach to code and data management, as both are complex and evolving. We believe such systems will become a core component of computational research, and integral to the dissemination and sharing of computational results. In short, reproducible computational science must be recognized as standard practice. 

References

1. J. Buckheit et al., *About Wavelab*, tech. report EFS-NSF-491, Dept of Statistics, Stanford Univ., 1995; [http://statistics.stanford.edu/~ckirby/techreports/NSF/EFS NSF 491.pdf](http://statistics.stanford.edu/~ckirby/techreports/NSF/EFS%20NSF%20491.pdf).
2. D. Donoho et al., "Reproducible Research in Computational Harmonic Analysis," *Computing in Science & Eng.*, vol. 11, no. 1, 2009, pp. 8–18.
3. Yale Law School Roundtable on Data and Code Sharing Roundtable, "Reproducible Research," *Computing in Science & Eng.*, vol. 12, no. 5, 2010, pp. 8–13.
4. B. Hanson, A. Sugden, and B. Alberts, "Making Data Maximally Available," *Science*, vol. 331, no. 6018, 2011, p. 649; www.sciencemag.org/content/331/6018/649.summary.
5. V. Stodden, *The Scientific Method in Practice: Reproducibility in the Computational Sciences*, MIT Sloan research paper no. 4773-10; http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1550193#%23.
6. C. Tenopir et al., "Data Sharing by Scientists: Practices and Perceptions," *PLoS One*, vol. 6, no. 6, 2011, e21101; doi:10.1371/journal.pone.0021101.
7. M. Nielsen, *Reinventing Discovery: The Era of Networked Science*, Princeton Univ. Press, 2011.
8. V. Stodden, "Trust Your Science? Open Your Data and Code," *Amstat News*, 1 July 2011; <http://magazine.amstat.org/blog/2011/07/01/trust-your-science/>.
9. B. Matthews et al., "A Framework for Software Preservation," *Int'l J. Digital Curation*, vol. 5, no. 1, 2010; <http://ijdc.net/index.php/ijdc/article/view/148>.

Randall J. LeVeque is the Founders' Term Professor of Applied Mathematics at the University of Washington. His interests include the development of algorithms and software for solving wave-propagation problems arising in a variety of applications. LeVeque holds a PhD in computer science from Stanford University. Contact him at rjl@uw.edu.

Ian M. Mitchell is an associate professor of computer science at the University of British Columbia. His research interests include the development of algorithms and software for nonlinear differential equations, formal verification, and control and planning in cyberphysical and robotic systems. Mitchell has a PhD in scientific computing and computational mathematics from Stanford University. Contact him at mitchell@cs.ubc.ca.

Victoria Stodden is an assistant professor of statistics at Columbia University. Her current research focuses on how pervasive and large-scale computation is changing our practice of the scientific method—in particular, regarding the reproducibility of computational results and the role of legal framing for scientific advancement. Stodden has a PhD in statistics from Stanford University. Contact her at vcs@stodden.net; <http://blog.stodden.net>.