Top Ten Reasons to *Not* Share Your Code (and why you should anyway)

Randall J. LeVeque[1]

Version of December 20, 2012

There is no ... Mathematician so expert in his science, as to place entire confidence in any truth immediately upon his discovery of it... Every time he runs over his proofs, his confidence encreases; but still more by the approbation of his friends; and is raised to its utmost perfection by the universal assent and applauses of the learned world.

— David Hume[2], 1739

1 Introduction

I am an advocate of sharing the computer code used to produce tables or figures appearing in mathematical and scientific publications, particularly when the results produced by the code are an integral part of the research being presented. I'm not alone, and in fact the number of people thinking this way seems to be rapidly increasing, see for example [1, 2, 3, 6, 7, 8, 10]. But there is still much resistance to this idea, and in the past several years I have heard many of the same arguments repeated over and over. So I thought it might be useful to write down some of the arguments along with counter-arguments that may be worth considering.

In this article I am thinking mostly of relatively small-scale code of the sort that might be developed to test a new algorithmic idea or verify that a new method performs as claimed, the sort of codes that might accompany articles in many SIAM journals. It can be at least as important to share and archive large scale simulation codes that are used as tools to do science or make policy decisions, but the issues are somewhat different and not all of the arguments below apply directly. However, as computational mathematics becomes increasingly important outside the ivory tower because of these large simulation codes, it is also worth remembering that the way we present our work can play a role in the ability of other scientists and engineers to do credible and reliable work that may have life-or-death consequences. Reproducibility is a cornerstone of the scientific method, and sharing the computer code used to reach the conclusions of a paper is often the easiest way to insure that all the details needed to reproduce the results have been provided.

This article grew out of a talk I gave with the same title in a minisymposium at the 2011 SIAM CSE meeting in Reno, organized by Jarrod Millman, on "Verifiable, Reproducible Research and Computational Science". (Slides from my talk and others are available at http://jarrodmillman.com/events/siam2011.html.)

[1] Department of Applied Mathematics, University of Washington, Box 352420, Seattle, WA 98195-2420, rjl@uw.edu.

[2] A Treatise of Human Nature, http://www.gutenberg.org/files/4705/4705-h/4705-h.htm

2 An alternative universe

Before discussing computer code, I'd like you to join me in a thought experiment. Suppose we lived in a universe where the standards for publication of mathematical theorems are quite different: papers present theorems without proofs, and readers are expected to simply believe the author when it is stated that the theorem has been proved. (In fact our own universe was once somewhat like this, but fortunately the idea of writing detailed proofs grew up along with the development of scientific journals. See, for example, Chapter 8 of [9] for a historical discussion of openness in science. I highly recommend the rest of this book as well.)

In this alternative universe the reputation of the author would play a much larger role in deciding whether a paper containing a theorem could be published. Do we trust the author to have done a good job on the crucial part of the research not being shown in the paper? Do we trust the theorem enough to use it in our own work in spite of not seeing the proof? This might be troubling in several respects. However, there are many advantages to not requiring carefully written proofs (in particular that we don't have to bother writing them up for our own papers, or referee those written by others) and so the system goes on for many years.

Eventually some agitators might come along and suggest that it would be better if mathematical papers contained proofs. Many arguments would be put forward for why this is a bad idea. Here are some of them (and yes, of course I hope you will see how similar they are to arguments against publishing code, *mutatis mutandis*, and will come up with your own counter arguments):

1. *The proof is too ugly to show anyone else.* It would be too much work to rewrite it neatly so others could read it. And anyway it's just a one-off proof for this particular theorem, and not intended for others to see, or to use the ideas for proving other theorems. My time is much better spent proving another result and publishing more papers rather than putting more effort into this theorem, which I've already proved.

2. *I didn't work out all the details.* Some tricky cases I didn't want to deal with, but the proof works fine for most cases, such as the ones I used in the examples in the paper. (Well, actually I discovered that some cases don't work, but they will probably never arise in practice.)

3. *I didn't actually prove the theorem, my student did.* And the student has since graduated, moved to Wall Street, and thrown away the proof, since of course dissertations also need not include proofs. But the student was very good, so I'm sure it was correct.

4. *Giving the proof to my competitors would be unfair to me.* It took years to prove this theorem, and the same idea can be used to prove other theorems. I should be able to publish at least 5 more papers before sharing the proof. If I share it now my competitors can use the ideas in it without having to do any work, and perhaps without even giving me credit since they won't have to reveal their proof technique in their papers.

5. *The proof is valuable intellectual property.* The ideas in this proof are so great that I might be able to commercialize them some day, so I'd be crazy to give them away.

6. *Including proofs would make math papers much longer.* Journals wouldn't want to publish them and who would want to read them?

7. *Referees would never agree to check proofs.* It would be too hard to determine correctness of long proofs and finding referees would become impossible. It's already hard to find enough good referees and get them to submit reviews in finite time. Requiring them to certify the correctness of proofs would bring the whole mathematical publishing business crashing down.

8. *The proof uses sophisticated mathematical machinery that most readers/referees don't know.* Their wetware cannot fully execute the proof, so what's the point in making it available to them?

9. *My proof invokes other theorems with unpublished (proprietary) proofs.* So it won't help to publish my proof — readers still will not be able to fully verify correctness.

10. *Readers who have access to my proof will want user support.* Anyone who can't figure out all the details will send email requesting that I help them understand it, and asking how to modify the proof to prove their own theorem. I don't have time or staff to provide such support.

3 Back to the real world

Of course there are some differences between sharing code and publishing proofs. So let's return to the real world and examine some of these in more detail.

It's not software designed for others to use, it's just a research code. There is obviously a difference between general-purpose software that is designed to be user-friendly and research code developed to test an idea and support a publication. However, most people recognize this difference and do not expect every code found on the web to come with user support. Nor do people expect every code found on the web to be wonderfully well written and documented. The more you clean it up, the better, but people publish far more embarrassing things on the web than ugly code, so perhaps it's best to get over this hangup [2]. Whatever state it is in, the code is an important part of the scientific record and often contains a wealth of details that do not appear in the paper, no matter how well the authors attempt to describe the method used. Parameter choices or implementation details can often be crucial and the ability to inspect the code if necessary can greatly facilitate efforts of other researchers to confirm the results or to adapt the methods presented to their own research problems.

Moreover I believe that it is actually extremely valuable to the author to clean up any code used to obtain published results to the point where it is not an embarrassment to display it to others. All too often I have found bugs in code when going through this process, and I suspect I am not alone. Almost everyone who has published a theorem and its proof has found that the process of writing up the proof cleanly enough for publication uncovers subtle issues that must be dealt with, and perhaps even major errors in the original working proof. Writing a research code is often no easier, so why should we expect to do it right the first time? It's much better for the author to find and fix these bugs before the paper is submitted for publication than have someone else rightfully question the results later.

It's forbidden to publish proprietary code. It is often true that research codes are based on commercial software or proprietary code that cannot be shared for various reasons. However, it is also often true that the part of the code that relates directly to the new research being published has been written by the authors of the paper and they are free to share this much at least. This is also the part of the code that is generally of most interest to referees or readers who want to understand the ideas or implementation described in the paper or obtain details not included in the article. The ability to execute the full code and replicate exactly the results in the paper is often of much less interest than the opportunity to examine the most relevant parts of the code.

Some authors may work for employers who don't allow them to share any code they write. However, if they are being allowed to publish this piece of research in the open literature, then my view is that they should be allowed to publish the parts of the code that are an essential part of the research. After all, employers cannot forbid authors from publishing a proof along with a theorem — referees would not put up with it. A change in expectations may lead to a change in what's allowed.

Moreover, publishing code can take various forms. Some employers may forbid sharing executables or source code in electronic form, but have far fewer restrictions on publishing an excerpt of the code (or even the entirety) in a pdf file. Again it is worth reiterating that for many readers or referees being able to inspect the relevant part of the code is often more valuable than being able to run the code.

If you do publish code, in an article or on the web, it is worth thinking about what sort of copyright or licensing agreement you attach to the code. This may affect the ability of others to reuse your code and the extent to which they must give you credit or propagate your license to derivative works [12].

The code may only run on certain systems today, and nowhere tomorrow. Even apart from the question of proprietary software, many codes have certain hardware or software dependencies that may make it impossible for the average reader to run — perhaps it only runs on a supercomputer or requires a graphics package that's only available on certain operating systems. Moreover, even if everyone can run it today, there is no guarantee that it will run on computers of the future, or with newer versions of operating systems, compilers, graphics packages, etc. In this way a code is quite different from a proof, so what use is it to archive the code? As in the case of proprietary software dependencies, I would argue that being able to examine the code is often extremely valuable even if it cannot be run, and is a critical component in making the research independently reproducible even if the tables or plots in the paper can't be reproduced with the push of a button.

Of course, authors should also attempt to make it easy to run the code whenever this is possible. From a purely selfish standpoint any effort put into cleaning up the code so that it does reproduce all the plots in the paper with a push of the button often pays off for the author down the road — when the referees ask for a revision of the way things are plotted, or when picking up the research project again years later. Or, heaven forbid, when some results in the paper come into question and the co-author who wrote the code has graduated or retired and is no longer available to explain where they actually came from. To minimize difficulties associated with software dependencies and versions, authors might consider using techniques such as a virtual machine to archive the full operating system along with the code [5] or use a code hosting site that simplifies the process of running an archived code without downloading or installing software [4].

Code is valuable intellectual property. It is true that some research groups spend years developing a code base to solve a particular class of scientific problems and their main interest is in "doing science" with these codes and publishing new results that are obtained from simulation. Expecting them to freely share the full code might be seen as the computational equivalent of requiring anyone publishing an experimental result to not only describe their methods in detail, but to also welcome any reader into their laboratory to use their experimental apparatus. This concern should be respected when advocating reproducibility in computational science, and I don't claim to have a good solution for all such cases.

However, for many research codes developed by applied mathematicians, the goal is to introduce and test new computational methods with the hope that others will use them (and cite their papers). For such codes I see little to be gained by not sharing. The easier it is for a reader to understand the details and implement it themselves, or even borrow code, the more likely it is to be adopted and cited by others.

Some people worry they will not receive proper credit from those who adapt code to their own research. But if everyone were expected to share code in publications it would be much easier to see what code has been used, and to compare it to the code archived with earlier publications. Citing the original source would then be easy and become standard operating procedure, leading to more citations for the original author. Readers of mathematics papers can judge for themselves how original the ideas in a published proof are, and if code development were equally transparent, those developing the original algorithms and code would ultimately receive more credit, not less.

Today the idea of publishing a theorem without its proof seems laughable to most mathematicians, in spite of the fact that many great mathematicians of the past apparently found it quite natural. Mathematics has since matured in healthy ways and it seems inevitable that computational mathematics will follow a similar path, no matter how inconvenient it may seem. I sense growing concern among young people in particular about the way we've been doing things and the difficulty of understanding or building on past work, and some funding agencies and journals now require sharing code that is used to obtain published results (see the *Science* guidelines for authors [11], for example). SIAM journals are not currently contemplating such a requirement, but the capability will soon be available for accepting and publishing unrefereed supplementary materials (including code) in conjunction with papers. I believe there is much to be gained, for authors as well as readers and the broader scientific community, by taking advantage of this and rethinking the way we present our work. We can all help our field mature by making the effort to share the code that supports our research.

References

[1] K. A. Baggerly and D. A. Barry. Reproducible research, 2011. `http://magazine.amstat.org/blog/2011/01/01/scipolicyjan11/`.

[2] N. Barnes. Publish your computer code: it is good enough. *Nature*, 467:753, 2010. `http://www.nature.com/news/2010/101013/full/467753a.html`.

[3] S. Fomel and J. F. Claerbout. Guest editors' introduction: Reproducible research. *Computing in Science and Engineering*, 11:5–7, 2009. `http://csdl2.computer.org/comp/mags/cs/2009/01/mcs2009010005.pdf`.

[4] J. Freire, P. Bonnet, and D. Shasha. Exploring the coming repositories of reproducible experiments: Challenges and opportunities. *Proceedings of the VLDB Endowment*, 4:1494–1497, 2011.

[5] B. Howe. Virtual appliances, cloud computing, and reproducible research. *Computing in Science and Engineering*, 14:36–41, 2012.

[6] J. Kovačević. How to encourage and publish reproducible research. *Proc IEEE Int. Conf. Acoust. Speech, and Signal Proc.*, pages IV:1273–1276, 2007.

[7] R. J. LeVeque, I. M. Mitchell, and V. Stodden. Reproducible research for scientific computing: tools and strategies for changing the culture. *Computing in Science and Engineering*, 14:13–17, 2012.

[8] J. Mesirov. Accessible reproducible research. *Science*, 327:415–416, 2010.

[9] M. Nielsen. *Reinventing Discovery: The New Era of Networked Science.* Princeton University Press, Princeton and Oxford, 2012.

[10] R. D. Peng. Reproducible research in computational science. *Science*, 334:1226–1227, 2011.

[11] Science magazine information of authors: Data and materials availability. http://www.sciencemag.org/site/feature/contribinfo/prep/gen_info.xhtml#dataavail, 2012.

[12] V. C. Stodden. The legal framework for reproducible scientific research: Licensing and copyright. *Comput. in Sci. Eng.*, 11:35–40, 2009.