Reproducible software vs. reproducible research

Fernando Pérez
Helen Wills Neuroscience Institute,
University of California at Berkeley.
http://fperez.org
Fernando.Perez@Berkeley.edu

January 21, 2011

Note: this text should be considered a working draft and is *not* a complete academic article (e.g. it currently lacks a proper set of bibliographic references). It is offered here as a complement to a talk delivered at the 2011 annual meeting of the AAAS.

As an active member of both the scientific research and the open-source software development communities, I have observed that the latter often lives up better than the former to our ideals of scientific openness and reproducibility. I will explore the reasons behind this, and I will argue that these problems are particularly acute in computational domains where they should be in fact less prevalent. I will discuss how we can draw specific lessons from the open source community both in terms of technical approaches and of changing the structure of incentives, to make progress towards a more solid base for reproducible computational research.

1 A contrast in cultures

Open source software development uses public fora for most discussion and systems for sharing code and data that are, in practice, powerful provenance tracking systems. There is a strong culture of public disclosure, tracking and fixing of bugs, and development often includes exhaustive automatic validation systems, that are executed automatically whenever changes are made to the software and whose output is publicly available on the internet. This helps with early detection of problems, mitigates their recurrence, and ensures that the state and quality of the software is a known quantity under a wide variety of situations (operating systems, inputs, parameter ranges, etc). Additionally, the very systems that are used for sharing the code track the authorship of contributions. All of this ensures that open collaboration does not dilute the merit or recognition of any individual developer, and allows for a meritocracy of contributors to develop while enabling highly effective collaboration.

In sharp contrast, the incentives in computational research are strongly biased towards the rapid publication of papers without any realistic requirement of validation. The outcome is that results from publications in computationally-based research (applied to any specific field of inquiry) are often, in practice, impossible to reproduce. Sometimes this is due to the code not being available at all in the first place. Often, however, authors do make codes available –thus fulfilling a token requirement of disclosure– but in such a state that it is not a practical solution to the reproducibility problem. A static archive of source code that has never been tested in a computer or operating system outside of the author's, that has never been audited by external eyes, that has no automatic testing built into it, is highly unlikely to work reliably when used in a completely new environment.

2 The realities of transplanting approaches

Notwithstanding the above, there are real issues with simply attempting to transplant the practices of open source development verbatim to computational research. The open source model ends up being one where, in practice, the copyright and authorship of any large collaborative project is spread amongst many authors, possibly thousands. While the source control tools in use do allow for a relatively precise provenance analysis to be performed if desired, this is rarely done and its success is contingent on the community having followed certain steps rigorously to ensure that attribution was correctly recorded during development.

This is not a major issue in open source, as the rewards mechanisms tend to be more informal and based on the overall recognition of any one contributor in the community. Sometimes people contribute to open source projects as part of their official work responsibilities, and in that case a company can enact whatever policies it deems necessary; often contributions are made by volunteers for whom an acknowledgment in the project's credits is sufficient recognition.

In contrast, the academic world overwhelmingly weighs the authorship of scholarly articles and conference proceedings as the main driver of all forms of professional advancement and reward. In this system, the pecking order of authorship matters enormously (with the many unpleasant consequences familiar to all of us), and so does the total number of authors in a publication. While in certain communities papers with thousands of authors do exist (experimental high-energy physics being the classic example), most scientists need the prominent visibility they can achieve in a short author list. This dilution of authorship that can result from a largely open collaborative development model is an important issue that must be addressed by any proposal we present.

Furthermore, the notion of a fully open development model typical of open source projects is at sharp odds with another aspect of the scientific publication and reward system: the "first to publish" race. Most scientists would, understandably, be very leery of exposing their projects to an openly accessible website when in their embryonic stages. The fear of being scooped by others is very real, and again we must properly address it as we consider how to apply the lessons of open source development to the scientific context.

3 The limits of scientific computational reproducibility

As we seek ways to learn from the ways in which the open source praxis can inform our scientific work, we must recognize that the ideal of scientific reproducibility is by necessity a reality of shades. We can see a gradation that goes from a pure mathematical result whose proof should be accessible to any person skilled in the necessary specialty, to one-of-a-kind experiments such as the Large Hadron Collider or the Hubble Space Telescope, that can't be reproduced in any realistic sense. At each point in this spectrum, however, we can always find ways to improve our confidence in the results: whether we re-analyze the same unique datasets with independently developed packages run by separate groups or we re-acquire partial samplings of critical data multiple times, we should never completely renounce the ideals of reproducibility because of practical difficulties.

Similarly, in computational research we also have certain areas where complete reproducibility is more challenging than others. Some projects require computations carried on the largest supercomputers on the planet, and these are very expensive resources that can't be arbitrarily allocated for repeated executions of the same problem. Others may require access to enormous datasets that can't easily be transferred to the desktop of any researcher wishing to re-execute an analysis. But again, alternatives exist: it should be possible to validate scaled versions of the largest problems run independently, against scaled specimens created on the supercomputers for this very purpose, and sub-sampled datasets can be used to collect at least validation statistics that may be informative of the trust we place on the published analysis.

4 Some ideas moving forward

Ultimately, while it is true that there are real issues with applying the ideals of computational reproducibility from open source software development to computational research, we can and must do better. We sketch here some ideas on concrete lessons we can learn from software development in this direction:

- Pervasive version control: research codes should be developed, while still in-house, *always* using version control systems that track the actual history of everyone's contributions.

- Journals should mandate that upon paper *approval* (but before actual publication and with said publication being conditioned on the author meeting this last condition), authors must expose their version control system to the public, and that the publicly available version can faithfully reproduce (within the limitations discussed above) the published results. This public version then becomes available for the scientific community not only for download, but also as a starting point for further contribution and development.

- By using a distributed version control system, authors can continue to maintain a private branch where new work (say leading to a new publication) is conducted while tracking the public development. This will enable them to maintain exclusive access to their new work until it is published, while continuing to develop the openly accessible code with the rest of the scientific community. Once the code is published, since it was developed using the same version control machinery of the public branch, the new contributions can be seamlessly merged with the public version and their entire provenance (including information such as time of invention and individual credit within the lab) becomes available for inspection.

In summary, we think that a few simple lessons can be learned from the practices of the open source world which, if carefully assimilated, can lead to significant improvements in the state of reproducible computational research.