Publishing scientific software matters

Christophe Pradal, Gaël Varoquaux, Hans Peter Langtangen

Publishing scientific software matters

Christophe Pradal[a,b], Gaël Varoquaux[c,d], Hans Peter Langtangen[e,f]

[a] *Virtual plants, INRIA Sophia-antipolis, France*
[b] *CIRAD / UMR DAP, Avenue Agropolis, TA 40/02, 34398 Montpellier, France*
[c] *Parietal project-team, INRIA Saclay-île de France, France*
[d] *CEA/Neurospin bât 145, 91191 Gif-Sur-Yvette, France*
[e] *Simula Research Laboratory, P.O. Box 134 NO-1325 Lysaker, Norway*
[f] *Department of Informatics, University of Oslo, P.O. Box 1080 Blinder, NO-0316, Norway*

Software is a central part of modern scientific discovery. Software turns a theoretical model into quantitative predictions; software controls an experiment; and software extracts from raw data evidence supporting or rejecting a theory. As of today, scientific publications seldom discuss software in depth, maybe because it is both highly technical and a recent addition to scientific tools. But times are changing. More and more scientific investigators are developing software and it is important to establish norms for publication of this work [7].

Producing scientific software is an important part of the landscape of research activities. Very visible scientific software is found in products developed by private companies, such as Mathwork's Matlab or Wolfram's Mathematica, but let us not forget that these build upon code written by and for academics [3]. Scientists writing software contribute to the advancement of Science via several factors.

First, software developed in one field, if written in a sufficiently general way, can often be applied to advance a different field if the underlying mathematics is common. Modern scientific software development has a strong emphasis on generality and reusability by taking advantage of the general properties of the mathematical structures in the problem. This feature of modern software help close the gap between fields and accelerate scientific discovery through packaging mathematical theories in a directly applicable way.

Second, the public availability of code is a corner stone of the scientific method, as it is a requirement to reproducing scientific results: *"if it's not open and verifiable by others, it's not science, or engineering, or whatever*

it is you call what we do." (anonymous in [8]). Emphasizing code to an extreme, Buckheit and Donoho [2] have challenged the traditional view that a publication was the valuable outcome of scientific research: *"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 [...]".*

It is important to keep in mind that going beyond replication of results requires reusable software tools: code that is portable [4], comes with documentation, and, most of all, is maintained throughout the years. Indeed, software development is a major undertaking [5] that must build upon best practices and a quality process [1, 9]. Reversing Buckheit and Donoho's argument, publications about scientific software play an increasingly important part in the scientific methodology. First, in the publish-or-perish academic culture, such publications give an incentive to software production *and maintenance*, because good software can lead to highly-cited papers. Second, the publication and review process are the *de facto* standards of ensuring quality in the scientific world. As software is becoming increasingly more central to the scientific discovery process, it must be subject to these standards. We have found that writing an article on software leads the authors to better clarify the project vision, technically and scientifically, the prior art, and the contributions. Last but not least, scientists publishing new results based on a particular software need an informed analysis of the validity of that software. Unfortunately, much of the current practice for adopting research software relies on ease of use of the package and reputation of the authors [6]. Peer-reviewed papers on the software will help improve this situation, and the goal of the current special edition of the *Journal of Computational Science* is to document several interesting software projects as referred journal papers.

This special edition is focused on scientific software that arose from presentations at the 2011 Euroscipy meeting. The Euroscipy conference[1] is the annual European meeting on the scientific usage of the Python language, which is now experiencing a substantally increasing popularity in computational science as a high-level language alternative to MATLAB. The Euroscipy meeting is not focused on a specific scientific field, but on software production and usage. As such it provides many examples of software developments for science that illustrate the various needs and challenges that arise in research, from experimentation to theory. Walter *et*

[1] `http://www.euroscipy.org`

al. [NEEDREF] tackle the cycle between modeling and experimentation in the context of nanophotonics. Wilbert *et al.* [NEEDREF] provide a data-processing package. Rossant *et al.* [NEEDREF] present a framework for distributed computing. Fiers *et al.* expose a state-of-the-art computational method, algorithmic differentiation, to a wider public via an easy-to-use software package. And finally, Bianchi *et al.* address the ever-present challenge of efficient software engineering in a scientific research context. These five papers are a mere drop in the ocean of scientific software development, but they paint a picture of this emerging landscape.

Today, software is to scientific research what Galileo's telescope was to astronomy: a tool, combining science and engineering. It lies outside the central field of principal competence among the researchers that rely on it. Like the telescope, it also builds upon scientific progress and shapes our scientific vision. Galileo's telescope was a leap forward in optics, a field of investigation that is now well established, with its own high-impact journals and scholarly associations. Similarly, we hope that visibility and recognition of scientific software development will grow.

References

[1] S.M. Baxter, S.W. Day, J.S. Fetrow, S.J. Reisinger, Scientific software development is not an oxymoron, PLoS Computational Biology 2 (2006) e87.

[2] J.B. Buckheit, D.L. Donoho, Wavelab and reproducible research, Springer, 1995.

[3] J.J. Dongarra, E. Grosse, Distribution of mathematical software via electronic mail, Communications of the ACM 30 (1987) 403–407.

[4] P.F. Dubois, T. Epperly, G. Kumfert, Why johnny can't build [portable scientific software], Computing in Science & Engineering 5 (2003) 83–88.

[5] R.L. Glass, Facts and fallacies of software engineering, Addison-Wesley Professional, 2002.

[6] L.N. Joppa, G. McInerny, R. Harper, L. Salido, K. Takeda, K. O'Hara, D. Gavaghan, S. Emmott, Troubling trends in scientific software use, Science 340 (2013) 814–815.

[7] C. Neylon, J. Aerts, C. Brown, S.J. Coles, L. Hatton, D. Lemire, K. Millman, P. Murray-Rust, F. Perez, N.F. Saunders, et al., Changing computational research. the challenges ahead., Source Code for Biology and Medicine 7 (2012) 2.

[8] V. Stodden, The scientific method in practice: reproducibility in the computational sciences (2010).

[9] G.V. Wilson, Where's the real bottleneck in scientific computing?, American Scientist 94 (2006) 5.