Reproducible Research

Our generation of computational scientists is living in an exciting time: not only do we get to pioneer important algorithms and computations, we also get to set standards on how computational research should be conducted and published. From Euclid's reasoning and Galileo's experiments, it took hundreds of years for the theoretical and experimental branches of science to develop standards for publication and peer review. Computational science, rightly regarded as the third branch, can walk the same road much faster.

What is science anyway? When does computational research become scientific? According to the American Physical Society, "Science is the systematic enterprise of gathering knowledge about the universe and organizing and condensing that knowledge into testable laws and theories. The success and credibility of science are anchored in the willingness of scientists to expose their ideas

SERGEY FOMEL
The University of Texas at Austin
JON F. CLAERBOUT
Stanford University

and results to independent testing and replication by other scientists. This requires the complete and open exchange of data, procedures and materials." (See www.aps.org/policy/statements/99_6.cfm for more information.)

The idea of a "replication by other scientists" in reference to computations is more commonly known as "reproducible research," the term coined by Jon Claerbout, a geophysics professor at Stanford University. Soon after its inception in 2000, *CiSE* published a paper by Claerbout and his students on their experience with creating a reproducible research environment.[1] The open-community Madagascar project (www.ahay.org) is currently extending this environment with modern tools such as Python-based SCons.[2]

A First-Hand Account

After spending many years working in this field, Claerbout has amassed a lifetime of experience in reproducible research. Here's his story in his own words: "From our failures, we learn the most. Being a senior person in this community, I have many failures to recount. In 1987, I spent a sabbatical leave preparing tutorial interactive programs in the most promising environment, Sun's new Sunview. It no longer exists, having been replaced by X-window—a lesson there! Since then, I've learned that interactive programs are slavery (unless they include the ability to arrive in any previous state by means of a script).

"I published textbooks in 1976 and 1985. In those days, inserting illustrations in a book was done by hand with the help of a copy machine. About then I began using the Unix `make` utility and its more logical version, `cake`. With these tools, I prepared a new book in 1992 in which I claimed all its 150 illustrations of signal and image processing could be destroyed, and I could, by typing a single command, rebuild them all, insert them into the LaTex source, and regenerate the book.

"Thus overwhelmed by my extraordinary new power (and being acutely aware that students graduate and leave), I began inflicting this goal upon a team of graduate students—all our research should be reproducible by other people by means of a simple build instruction. We began producing CD-ROMs (an accomplishment in those days) for our industrial sponsors and made the claim that they would easily be able to reproduce all our work on any workstation of Sun, IBM, or HP. Only then did we begin to appreciate the enormity of the task. We ran into the 'versioning problem': different people had different compilers. X-window was constantly changing. The C-shell had different versions. LaTex had differing fonts and versions. But we were undaunted and still enthusiastic. We learned to define and use new concepts when some of our environment wasn't exportable. Our increasing store of reproducible research provided a great testbed for new software or changes in old software. Not long ago, I discovered a 20-year-old working C program suddenly failed.

"Responding to the broader availability of `gmake` over `cake`, we converted to `gmake` in hopes of enlarging the community. (New problems arose because the `gmake` community cared about software building while we cared about building documents

and their illustrations.) Again, in an attempt to increase the size of the community, Matthias Schwab and I submitted a paper to *Computers in Physics,* one of *CiSE*'s forerunners. It was rejected. The editors said if everyone used Microsoft computers, everything would be easily reproducible. They also predicted the imminent demise of Fortran.

"Despite our poor ability to get new adherents to the philosophy of reproducible research, we continued producing maybe 800 pages a year of documents in signal and image processing, steadily honing our skills at doing so. Although I always made the claim (which was true) that reproducibility was essential to pass wisdom on to the next generation, our experience was always that the most likely recipient would be the author herself at a later stage of life."

In this Issue

The articles in this special issue provide independent solutions for practical reproducible research systems. David Donoho and his colleagues describe their 15 years of experience in promoting reproducible research in computational harmonic analysis using Matlab-based tools such as the famous Wavelab and Sparselab packages. In particular, the authors point to the success of the reproducible research discipline in increasing the reliability of computational research and reflect on the effort necessary for implementing this discipline in a research group and overcoming possible objections to it. The article ends with a call to funding agencies to support reproducible research.

The article by Randall LeVeque describes a Python interface to the well-known Clawpack package for solving hyperbolic partial differential equations that appear in wave propagation problems. The author was led to reproducible research by personal frustration with publications in scientific and mathematical journals that are "filled with pretty pictures of computational experiments that the reader has no hope of reproducing." He argues strongly in favor of reproducible computations and shows an example using a simplified Python interface to Fortran code.

Roger Peng and Sandrah Eckel represent the field of bioinformatics, which has been a stronghold of reproducible research.[3] Their article describes the cacher package, which is built on top of the R computing environment. Cacher enables a modular approach to reproducible computations by storing results of intermediate computations in a database.

The special issue ends with an article by Victoria Stodden on the legal aspects of reproducible research, including copyright and licensing issues.

She proposes the Reproducible Research Standard (RRS) as a tool for enabling and promoting reproducible research similar to the way the GNU Public License (GPL) is used to promote free software. RRS encompasses not only software but all other data, procedures, and materials necessary for replicating a computational experiment.

Before you publish your next paper, please ask yourself a question: Have I done enough to allow the readers of my paper to verify and reproduce my computational experiments? Your solution to reproducibility might differ from the those described in this issue, but only with a joint effort can we change the standards by which computational results are rendered scientific.　　　　 $\overset{\text{C}}{\text{SE}}$

References

1. M. Schwab, M. Karrenbach, and J. Claerbout, "Making Scientific Computations Reproducible," *Computing in Science & Eng.,* vol. 2, no. 6, 2000, pp. 61–67.
2. S. Fomel and G. Hennenfent, "Reproducible Computational Experiments Using SCons," *Proc. IEEE Int'l Conf. Acoustics, Speech and Signal Processing,* vol. 4, IEEE CS Press, 2007, pp. 1257–1260.
3. R. Gentleman, "Reproducible Research: A Bioinformatics Case Study," *Statistical Applications in Genetics and Molecular Biology,* vol. 4, no. 1, 2005, pp. 1–23.

Sergey Fomel is an associate professor at the Jackson School of Geosciences, the University of Texas at Austin, with a joint appointment between the Bureau of Economic Geology and the Department of Geological Sciences. His research interests are computational and exploration geophysics, including seismic imaging and data analysis. Fomel has a PhD in geophysics from Stanford University. He recently received the Clarence Karcher Award from the Society of Exploration Geophysicists "for numerous contributions to seismology." Contact him at sergey.fomel@beg.utexas.edu.

Jon F. Claerbout is the Green professor in the Department of Geophysics at Stanford University. He has a PhD in geophysics from MIT, founded the Stanford Exploration Project (SEP) in 1973, and has authored three published books and five Internet books. Claerbout is an elected fellow of the American Geophysical Union, a member of the National Academy of Engineering, and an honorary member of the Society of Exploration Geophysicists and the European Association of Geoscientists and Engineers, which both honored him with their highest awards for his pioneering work in seismic data analysis. Contact him at claerbout@stanford.edu.