My digital toolbox: Nuclear engineer Katy Huff on version-control systems

Git and GitHub are the 'laboratory notebook of scientific computing'.

Sylvia Tippmann

29 September 2014

Denia Djokić

Katy Huff uses software to simulate the behaviour of nuclear reactors in accidents.

In the second of a regular series, Katy Huff, a nuclear engineering postdoc at the University of California, Berkeley, discusses the software and tools that she finds most useful in her research.

How would you describe your research?

I am a computational nuclear engineer interested in problems of nuclear-energy sustainability and nuclear reactor safety. I have worked in the past on the environmental impacts of nuclear fuel-recycling schemes and disposal options. Currently, I am interested in the promise of new, more 'passively safe' reactor technologies. Some nuclear reactors are designed to take advantage of basic physics if there is an accident — for example, the actions of gravity and natural convection — so that they should shut down safely without active intervention (see 'Nuclear energy: Radical reactors').

As a fellow of the Nuclear Science and Security Consortium [a multi-institution network that trains and educates experts in nuclear security] , I contribute to the design of just such a reactor by developing a simulation of the neutron, heat and fluid behaviour during various accident scenarios. Also, as a fellow with the Berkeley Institute for Data Science, I engage in work related to determining and advocating best practices in scientific computing and computing education.

Which software, websites or tools do you use on a regular basis, and why?

The list of tools I rely on day-to-day is incredibly long. Even if I neglect all of those that are specific to nuclear engineering, the list is long and includes: Git, GitHub, LaTeX, the editor vi, Zotero, Google Scholar, NumPy, PyTables, SQLite3, matplotlib, nose, Google Test, CMake...

However, the tool that has most powerfully impacted the reproducibility, transparency and robustness of my work is definitely the combination of Git and GitHub. These are version-control systems; the laboratory notebook of scientific computing.

The purpose of a lab notebook is to provide a lasting record of events in a laboratory. In the same way that a chemistry experiment would be nearly impossible without a lab notebook, scientific computing would be a nightmare of inefficiency and uncertainty without version-control systems.

Visit the Toolbox hub for more articles

Can you explain more about how Git and GitHub work?

Rather than inventing a complex naming scheme for new versions of code, the best practice in software development is to allow the version-control system to track changes over time. Git provides a framework in which those changes are easily stored and must be annotated, just like in a lab notebook, with your own notes. Git automatically attaches a date and time stamp to that annotation and associates your work with your author name.

Also, GitHub brings that laboratory notebook to the world. Its interface for 'pull requests' allows collaborators to peer review code. That transparent archiving and opportunity for review does for scientific software what the peer-reviewed journal system does for scientific papers. Scientific code has historically gone unreviewed and unrecognized by the scientific community. However, by virtue of being archived, transparent and peer-reviewed, software is increasingly a go-to scientific research product — not unlike a journal article. Without the easy interface in GitHub, this would never be possible.

Git has a steep learning curve, so I would recommend a new user to take advantage of material on the topic on the skill-sharing website Software Carpentry or to read the Pro Git book. Whereas Git can be paradigmatically difficult, the GitHub social platform should be easy to get started with. Just create an account and start browsing, forking and cloning code that you find interesting! I should note that there are alternative repository hosting sites: Bitbucket and Launchpad are excellent examples.

Which emerging tools do you have your eye on?

I'm really excited about a computational framework called MOOSE. I am only just starting to use it, but I have high hopes for its computational power in my reactor safety analysis. It is a solution environment developed at Idaho National Laboratory for use in problems that involve many types of physics at once — or 'multiphysics problems'. MOOSE is a highly scalable framework for solving coupled problems, a platform that can easily be run on large computers and solves physical problems that involve multiple disciplines. MOOSE provides the framework, but user–developers are given the power to implement their own extensions to the code for their own custom physics.

Would you recommend any training courses or books for learning more about scientific software?

For many years I have been involved in teaching computing skills to scientists through the websites Software Carpentry and The Hacker Within. Software Carpentry's online tutorials, as well as its in-person workshops, are the best resources available for a scientist interested in best practices for scientific software (see 'Boot camps teach scientists computing skills'.)

My colleague Anthony Scopatz [a computational scientist at the University of Wisonsin-Madison] and I are working on a book that should be published this winter. Its working title is *Effective Computation in Physics* and it will address all of the tools and best practices that are applicable to an early research career in the physical sciences.

What software would you like to see developed in future?

Although I use many programming languages, I deeply love using the Python programming language. However, Python has a severe packaging problem. It is hard to download and install non-conflicting versions of many modules in Python without eventually polluting your programming environment with multiple versions of a module. I know that Python will be a much less intimidating language for new users once this problem is solved, so I am anxious for a solution. A number of tools have attempted to solve this problem, and I think at least a few (for example, Anaconda from Continuum Analytics) are getting very close. So, there's hope!

Nature | doi:10.1038/nature.2014.16014