Tools for the lifecycle of computational research

Fernando Perez and Brian E. Granger

August 14, 2013

Contents

Part I

Abstract

We propose to build open source tools to support the various phases of computational work that are typical in scientific research and education. Our tools will span the entire life-cycle of a research idea, from initial exploration to publication and teaching. They will enable reproducible research as a natural outcome and will bridge the gaps between code, published results and educational materials. This project is based on existing, proven open source technologies developed by our team over the last decade that have been widely adopted in academia and industry.

Part II

Proposal

Scientific research has become pervasively computational. In addition to experiment and theory, the notions of simulation and data-intensive discovery have emerged as third and fourth pillars of science [1]. Today, even theory and experiment are computational, as virtually all experimental work requires computing (whether in data collection, preprocessing or analysis) and most theoretical work requires symbolic and numerical support to develop and refine models. Scanning the pages of any major scientific journal, one is hard-pressed to find a publication in any discipline

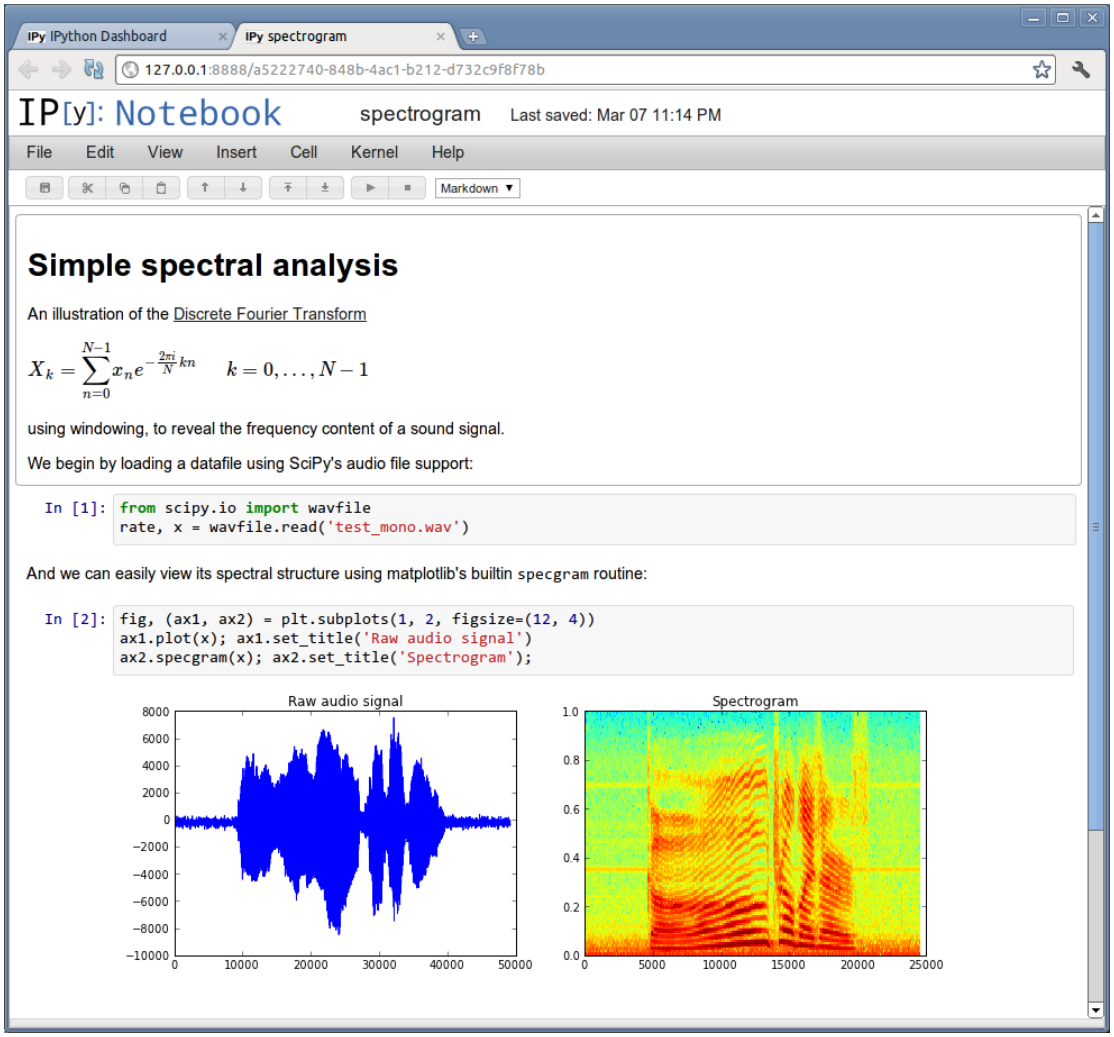
that doesn't depend on computing for its findings.And yet, for all its importance, computing is often treated as an afterthought both in the training of our scientists and in the conduct of everyday research. Most working scientists have witnessed how computing is seen as a task of secondary importance that students and postdocs learn "on the go" with little training to ensure that results are trustworthy, comprehensible and ultimately a solid foundation for reproducible outcomes. Software and data are stored with poor organization, documentation and tests. A patchwork of software tools is used with limited attention paid to capturing the complex workflows that emerge, and the evolution of code is often not tracked over time, making it difficult to understand how a result was obtained. Finally, many of the software packages used by scientists in research are proprietary and closed-source, preventing the community from having a complete understanding of the final scientific results. The consequences of this cavalier approach are serious. Consider, just to name two widely publicized cases, the loss of public confidence in the "Climategate" fiasco [2] or the Duke cancer trials scandal, where sloppy computational practices likely led to severe health consequences for several patients [3].This is a large and complex problem that requires changing the educational process for new scientists, the incentive models for promotions and rewards, the publication system, and more. We do not aim to tackle all of these issues here, but our belief is that a central element of this problem is the nature and quality of the software tools available for computational work in science. Based on our experience over the last decade as practicing researchers, educators and software developers, we propose an integrated approach to computing where the entire life-cycle of scientific research is considered, from the initial exploration of ideas and data to the presentation of final results. Briefly, this life-cycle can be broken down into the following phases:

- **Individual exploration:** a single investigator tests an idea, algorithm or question, likely with a small-scale test data set or simulation.
- **Collaboration:** if the initial exploration appears promising, more often than not some kind of collaborative effort ensues.
- **Production-scale execution:** large data sets and complex simulations often require the use of clusters, supercomputers or cloud resources in parallel.
- **Publication:** whether as a paper or an internal report for discussion with colleagues, results need to be presented to others in a coherent form.
- **Education:** ultimately, research results become part of the corpus of a discipline that is shared with students and colleagues, thus seeding the next cycle of research.

In this project, we tackle the following problem. **There are no software tools capable of spanning the entire lifecycle of computational research.** The result is that researchers are forced to use a large number of disjoint software tools in each of these phases in an awkward workflow that hinders collaboration and reduces efficiency, quality, robustness and reproducibility.These can be illustrated with an example: a researcher might use Matlab for prototyping, develop high-performance code in C, run post-processing by twiddling controls in a Graphical User Interface (GUI), import data back into Matlab for generating plots, polish the resulting plots by hand in Adobe Illustrator, and finally paste the plots into a publication manuscript or PowerPoint presentation. But what if months later the researcher realizes there is a problem with the results? What are the chances they will be able to know what buttons they clicked, to reproduce the workflow that can generate the updated plots, manuscript and presentation? What are the chances that other researchers or students could reproduce these steps to learn the new method or understand how the result was obtained? How can reviewers validate that the programs and overall workflow are free of errors? Even if the researcher successfully documents each program and the entire workflow, they have to carry an immense cognitive burden just to keep track of everything.We propose that the open source IPython project [4] offers a solution to these problems; a single software tool capable of spanning the entire life-cycle of computational research. Amongst high-level open source programming languages, Python is today the leading tool for general-purpose source scientific computing (along with R for statistics), finding wide adoption across research disciplines, education and industry and being a core infrastructure tool at institutions such as CERN and the Hubble Space Telescope Science Institute [5, 6, 7]

In [1]:
```
from IPython.display import Image
Image('ipython-notebook-specgram.png')
```

Out [1]:



References

[1] Tony Hey, Stewart Tansley, and Kristin Tolle, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Redmond, Washington, 2009.

[2] O. Heffernan. 'Climategate' scientist speaks out. *Nature*, 463(7283):860, 2010.

[3] J. Couzin-Frankel. Cancer research. As questions grow, Duke halts trials, launches investigation. *Science*, 329(5992):614–5, 2010.

[4] F. Pérez and B. E. Granger. IPython: a System for Interactive Scientific Computing. *Computing in Science & Engineering*, 9(3):21–29, May 2007. URL: http://ipython.org.

[5] F. Pérez, B. E. Granger, and J. D. Hunter. Python: an ecosystem for scientific computing. *Computing in Science & Engineering*, 13(2):13–21, 2011.

[6] Frederic Brochu, Ulrik Egede, J. Elmsheuser, K. Harrison, R. W. L. Jones, H. C. Lee, Dietrich Liko, A. Maier, Jakub T. Moscicki, A. Muraru, Glen N. Patrick, Katarina Pajchel, W. Reece, B. H. Samset, M. W. Slater, A. Soroko, C. L. Tan, and Daniel C. Vanderster. Ganga: a tool for computational-task management and easy access to grid resources. *CoRR*, abs/0902.2685, 2009.

[7] Science Software Branch at the Space Telescope Science Institute. Space Telescope Science Institute stsci_python. URL: http://www.stsci.edu/institute/software_hardware/pyraf/stsci_python.