Jupyter Notebooks—a publishing format for reproducible computational workflows

Thomas KLUYVER[a,1], Benjamin RAGAN-KELLEY[b,1], Fernando PÉREZ[c], Brian GRANGER[d], Matthias BUSSONNIER[c], Jonathan FREDERIC[d], Kyle KELLEY[e], Jessica HAMRICK[c], Jason GROUT[f], Sylvain CORLAY[f], Paul IVANOV[g], Damián AVILA[h], Safia ABDALLA[i], Carol WILLING[d] and Jupyter Development Team[j]

[a] *University of Southampton, UK*
[b] *Simula Research Lab, Norway*
[c] *University of California, Berkeley, USA*
[d] *California Polytechnic State University, San Luis Obispo, USA*
[e] *Rackspace*
[f] *Bloomberg LP*
[g] *Disqus*
[h] *Continuum Analytics*
[i] *Project Jupyter*
[j] *Worldwide*

Abstract. It is increasingly necessary for researchers in all fields to write computer code, and in order to reproduce research results, it is important that this code is published. We present Jupyter notebooks, a document format for publishing code, results and explanations in a form that is both readable and executable. We discuss various tools and use cases for notebook documents.

Keywords. Notebook, reproducibility, research code

1. Introduction

Researchers today across all academic disciplines often need to write computer code in order to collect and process data, carry out statistical tests, run simulations or draw figures. The widely applicable libraries and tools for this are often developed as open source projects (such as NumPy, Julia, or FEniCS), but the specific code researchers write for a particular piece of work is often left unpublished, hindering reproducibility.

Some authors may describe computational methods in prose, as part of a general description of research methods. But human language lacks the precision of code, and reproducing such methods is not as quick or as reliable as it should be. Others provide code separately as supplementary material, but it may be difficult for readers to cross reference between code and prose, and there is a risk that the two become inconsistent as the author works on them.

Notebooks—documents integrating prose, code and results—offer a way to publish a computational method which can be readily read and replicated.

[1] Corresponding Author.

2. Notebooks

Notebooks are designed to support the workflow of scientific computing, from interactive exploration to publishing a detailed record of computation. The code in a notebook is organised into cells, chunks which can be individually modified and run. The output from each cell appears directly below it and is stored as part of the document. This is an evolution of the interactive shell or REPL (read-evaluate-print loop) which has long been the basis of interactive programming (Iverson, 1962; Spence, 1975). However, whereas the direct output in most shells can only be text, notebooks can include rich output such as plots, formatted mathematical equations, and even interactive controls and graphics. Prose text can be interleaved with the code and output in a notebook to explain and highlight specific parts, forming a rich computational narrative.

The notebook interface first became popular among mathematicians. The proprietary computer algebra systems Mathematica and Maple both feature notebook interfaces, as does the open source SageMath.

Jupyter aims to bring notebooks to a broader audience. Jupyter is an open source project, which can work with code in many different programming languages. Different language backends, called kernels, communicate with Jupyter using a common, documented protocol; over 50 such backends have already been written, for languages ranging from C++ to Bash. Jupyter grew out of the IPython project (Pérez & Granger, 2007), which initially provided this interface only for the Python language. IPython continues to provide the canonical Python kernel for Jupyter.

The Jupyter Notebook is accessed through a modern web browser. This makes it practical to use the same interface running locally like a desktop application, or running on a remote server. In the latter case, the only software the user needs locally is a web browser; so, for instance, a teacher can set up the software on a server and easily give students access. The notebook files it creates are a simple, documented JSON format, with the extension '.ipynb'. It is simple to write other software tools which access and manipulate these files.

3. Sharing and reproducibility

Notebooks record a computation in order to explain it in detail to others, and a variety of tools help users to conveniently share notebooks. The Jupyter project includes *nbconvert*, which converts notebook files into a variety of file formats, including HTML, LaTeX and PDF, so that they are accessible without needing any Jupyter software installed. Nbconvert uses a powerful templating engine (Jinja), so the conversion process can be completely customised to produce different kinds of output.

Another Jupyter project, *nbviewer*, is a hosted web service built around nbconvert. nbviewer provides an HTML view of notebook files published anywhere on the web. The primary instance runs at https://nbviewer.jupyter.org/, but as it is open source, anyone can run their own instance—for example on an internal network, to view notebooks which should not be made public. These HTML views have a major advantage over publishing converted HTML directly: they link back to the notebook file, so interested readers can download it, run it and modify it themselves.

While nbconvert and nbviewer facilitate sharing statically rendered notebooks, a new project called *Binder* (http://mybinder.org/) enables sharing of live notebooks,

including a computational environment in which users can execute the code. Authors can publish notebooks on GitHub along with an environment specification in one of a few common formats. By pointing the Binder web service at the repository, a temporary environment is automatically created with the notebooks and any libraries and data required to run them. This allows authors to publish their code in an interactive and immediately verifiable form.

Together, these tools allow the preservation and reuse of scientific code, the computational environment to run that code, and data within the size constraints of a git repository. Third party tools such as *noWorkflow* can integrate with this to track provenance: how inputs, code and generated files relate to one another. noWorkflow captures the execution of a marked notebook cell, or a script run through its command line tool, as a 'trial', recording in a database the code that was used, the environment in which it ran, the versions of modules that were used, and the files read and written.

4. Notebooks in academic publishing

Several papers have been published with supporting notebooks to reproduce the analysis, or the creation of key plots. The detection of gravitational waves by the LIGO experiment (LIGO Scientific Collaboration and Virgo Collaboration et al., 2016), announced earlier this year, is one such: the researchers posted a notebook on their website illustrating in detail how to filter and process the data to reveal the signature of a distant black hole merger (LIGO collaboration). Others quickly made this available through Binder, as described above (https://github.com/minrk/ligo-binder), allowing anyone to replicate the analysis even without downloading or installing anything. Other papers published in fields from geology to genetics to computer science have used notebooks as supporting material (e.g. Sylvester et al., 2013; Olson & Roberts, 2015; Brown et al., 2012).

Authors have also written books as a collection of IPython notebooks. Some of these have been published in hard copy (e.g. Unpingco, 2014; Davidson-Pilon, 2015; Rossant, 2014), but with the internet blurring traditional categorisations, similar collections of notebooks are being published purely online. Of these, course materials are a notable group, both to accompany teaching and for learners to work through independently (e.g. Caporaso; Barba; Johansson).

It is not yet very practical to write academic papers themselves as notebooks, but we are working towards this. One tricky point is inserting academic citations, which require structured data about sources to be formatted in a very precise way which may depend on the journal. One of us (TK) has an experimental plugin cite2c (https://github.com/takluyver/cite2c), which allows the author to search their reference library stored in the Zotero service, and insert citations into a Markdown cell. The citations and bibliography are rendered by the citeproc-js package (Bennett), using the common Citation Style Language format (http://citationstyles.org/).

Notebooks also fit well into novel publishing paradigms, such as post publication review. Digital objects such as GitHub repositories, which may contain notebooks, and blog posts, which may be made from notebooks, can now be archived and given permanent DOI references (GitHub; Yarkoni, 2015), making it practical to cite them in other publications. The Jupyter Project is part of the coalition around Hypothes.is, an open source tool to annotate documents on the web (Perkel, 2015; Hypothes.is, 2015).

Finally, work is under way to support real-time collaboration in notebooks. This will let multiple authors work on a notebook together, with the changes instantly visible to all, reducing the chance of two people trying to change the same thing in different ways.

References

Barba, L.A. CFD Python: 12 Steps to Navier-Stokes, Available from: <http://lorenabarba.com/blog/cfd- python-12-steps-to-navier-stokes/> [Accessed: 4 March 2016]

Bennett, F. Citeproc-Js, Available from: <https://bitbucket.org/fbennett/citeproc-js> [Accessed: 4 March 2016]

Brown, C.T., Howe, A., Zhang, Q., Pyrkosz, A.B. & Brom, T.H. (2012) A Reference-Free Algorithm for Computational Normalization of Shotgun Sequencing Data, arXiv:1203.4802 [q-bio] Available from: <http://arxiv.org/abs/1203.4802> [Accessed: 4 March 2016]

Caporaso, G. An Introduction to Applied Bioinformatics, Available from: <http://readiab.org/> [Accessed: 4 March 2016]

Davidson-Pilon, C. (2015) Bayesian Methods for Hackers: Probabilistic Programming and Bayesian Inference, New York: Addison-Wesley Professional GitHub Making Your Code Citable, Available from: <https://guides.github.com/activities/citable-code/> [Accessed: 4 March 2016]

Hypothes.is (2015) Annotating All Knowledge, Available from: <https://hypothes.is/annotating-all-knowledge/> [Accessed: 4 March 2016]

Iverson, K.E. (1962) A Programming Language, New York, NY, USA: John Wiley & Sons, Inc.

Johansson, R. QuTiP Lectures as IPython Notebooks, Available from: <https://github.com/jrjohansson/qutip-lectures> [Accessed: 4 March 2016]

LIGO collaboration Signal Processing with GW150914 Open Data, Available from: <https://losc.ligo.org/s/events/GW150914/GW150914_tutorial.html> [Accessed: 4 March 2016]

LIGO Scientific Collaboration and Virgo Collaboration, Abbott, B.P., Abbott, R., Abbott, T.D., Abernathy, M.R., Acernese, F., et al. (2016) Observation of Gravitational Waves from a Binary Black Hole Merger, Physical Review Letters 116 (6): 061102

Olson, C.E. & Roberts, S.B. (2015) Indication of Family-Specific DNA Methylation Patterns in Developing Oysters, bioRxiv: 012831

Pérez, F. & Granger, B.E. (2007) IPython: A System for Interactive Scientific Computing, Computing in Science Engineering 9 (3): 21–29

Perkel, J.M. (2015) Annotating the Scholarly Web, Nature 528 (7580): 153

Rossant, C. (2014) IPython Interactive Computing and Visualization Cookbook, Packt Publishing Spence, R. (1975) APL Demonstration, Imperial College London Available from: <https://www.youtube.com/watch?v=_DTpQ4Kk2wA> [Accessed: 4 March 2016]

Sylvester, Z., Pirmez, C., Cantelli, A. & Jobe, Z.R. (2013) Global (latitudinal) Variation in Submarine Channel Sinuosity: COMMENT, Geology 41 (5): e287–e287

Unpingco, J. (2014) Python for Signal Processing, Springer

Yarkoni, T. (2015) Now I Am Become DOI, Destroyer of Gatekeeping Worlds, The Winnower Available from: <https://thewinnower.com/papers/282-now-i-am-become-doi-destroyer-of-gatekeeping-worlds> [Accessed: 4 March 2016]