TOOLBOX

INTERACTIVE NOTEBOOKS: SHARING THE CODE

The free IPython notebook makes data analysis easier to record, understand and reproduce.

ILLUSTRATION BY THE PROJECT TWINS

BY HELEN SHEN

Flying high above the Pacific Ocean, Titus Brown is taking a deep dive into his students' research code. The long journey from Michigan State University in East Lansing to a conference in Melbourne, Australia, provides the perfect chance for the bioinformatician to scrutinize his lab's new algorithm for removing errors from RNA sequencing data.

Three years ago, Brown might have waited until he was back in his office. It is difficult to dig into other researchers' code without them being present to explain it, make changes and produce updated results. But these days, Brown can work with his lab from afar using a free, open-source software package called IPython, which helps researchers to keep a detailed lab notebook for their computational work.

Brown's students write explanatory text and intersperse it with raw code and the charts and figures that their algorithms generate. Sitting in the aeroplane with an IPython notebook downloaded to his laptop, Brown can interact with the work. He tweaks and re-runs the code, which executes directly in the document he is reading — allowing him to see instantly whether his changes are improving the algorithm. "I can go through their notebook, understand exactly what they did and modify it, explore different parameters and look at different views," he says. "I can do this from anywhere in the world."

Designed to make data analysis easier to share and reproduce, the IPython notebook is being used increasingly by scientists who want to keep detailed records of their work, devise teaching modules and collaborate with others. Some researchers are even publishing the notebooks to back up their research papers — and Brown, among others, is pushing to use the program as a new form of interactive science publishing.

BETTER BOOKKEEPING

The IPython notebook was developed in 2011 by a team of researchers led by Fernando Pérez, a data scientist at the University of California, Berkeley, and computational physicist Brian Granger at California Polytechnic State University in San Luis Obispo. "We built it by solving problems that we ourselves had as researchers and educators," says Pérez.

PROGRAMMING

IPython for beginners

Getting the tools
An interactive *Nature* demonstration of an IPython notebook is available at go.nature.com/ohkjks. The software for the notebook can be downloaded from the IPython website at go.nature.com/mq8nip.

Learning the ropes
Instructions for proficient coders can be found at go.nature.com/sdbolb; example notebooks are at go.nature.com/awtkxn.

For people unfamiliar with Python, a host of resources exist online. OpenTechSchool's educational unit on data analysis with Python (go.nature.com/gpuyxp) includes an introduction to the IPython notebook.

For hands-on training, Software Carpentry, a volunteer organization that teaches basic software skills, can help. The group will run two-day workshops on the IPython notebook by invitation throughout the world (go.nature.com/fj6sza).

Sharing the goods
IPython notebooks can be shared in online repositories such as GitHub, or over e-mail. Recipients need IPython software to view and edit the notebooks. Notebook authors can also use a program called nbviewer to create an online version of their notebook that will be viewable, but not editable (go.nature.com/ry6g4j).

Pérez and Granger saw that data scientists found it hard to share detailed but understandable descriptions of their raw code that would allow others to build on their research. That is partly because many scientists in computation-intensive fields write code in an iterative and piecemeal fashion as each analysis reveals new insight and spins off multiple lines of inquiry. Keeping track of the different versions of code that produce various figures, and linking those files with explanatory notes, is a headache. And what gets published is usually not detailed enough for the reader to follow up on. "In my own computational physics work," says Granger, "a high-level description of the algorithm that goes into the paper is light years away from the details that are written in the code. Without those details, there is no way that someone could reproduce it in a reasonable time scale."

The IPython notebook addresses both issues by helping scientists to keep track of their work, and by making it easy to share and for others to explore the code. The 'I' in IPython refers to an 'interactive' command window that helps users to run code, access variables, call up data analysis packages and view plots, while the Python refers to the popular programming language that the notebook is based on. (Pérez, Granger and their colleagues are now moving the notebook into a project called Jupyter, which aims to make IPython more compatible with other languages, including Julia and R).

CODE CHOPS

At the University of Texas at Austin, Tal Yarkoni uses the IPython notebook to run automated meta-analyses of brain imaging studies to uncover patterns of neural activity involved in language processing, emotion and other processes. The psychoinformatician plans to publish the notebooks as companions to his future journal articles. "The more complicated the analyses, the greater the benefits of being able to convey all that in one simple document," he says.

Applications similar to the IPython notebook already exist for various programming languages. Mathematica and Maple — commercial analysis packages popular among mathematicians — include notebooks or notebook-like programs. MATLAB, a commercial package used heavily in signal processing, engineering and medical-imaging research, also supports a notebook application. Each of these notebooks is specialized for its corresponding proprietary programming language.

A number of notebooks and notebook-like programs exist in the open-source world; knitr works with the R coding language, which is especially powerful for statistical analysis. And the Sage mathematical software system, which is also based on the Python language, supports its own notebook. Dexy is a notebook-like program that focuses on helping users to generate papers and presentations that incorporate prose, code, figures and other media.

But the IPython notebook has become one of the most widely adopted programs of its kind, says Ana Nelson, the creator of Dexy. "So many people have heard of it who haven't heard of any other tool," she says. Granger and Pérez do not know how many people have tried their software, but say that traffic to the website suggests that roughly 500,000–1.5 million people actively use the program. Nelson says that it is the best-designed of the digital notebooks, and attracts many users because it is free and open source. The application also benefits from the popularity of the Python language, which boasts a robust scientific community that meets for an annual international conference, and is (relatively) easy for novice programmers to learn.

NATURE.COM
For more on scientifc software, apps and online tools, visit:
nature.com/toolbox

Although a growing number of researchers are publishing their notebooks alongside papers (see go.nature.com/mqonbm for examples), it may still be some time before journals accept the documents as full journal articles.

A handful of IPython notebooks have been published as books, and many professors use the program to make interactive curricula. But so far, the notebooks seem to have been published only as addenda to papers — often to provide analysis code and additional explanation in method sections.

"Publishers, I would say, still aren't convinced that they want to go the whole way," says Granger. The data format may be too new, he says, for journals to recognize the notebook as an official document format, such as html or pdf. But the IPython team has begun talks with a few publications.

START FROM SCRATCH

Most IPython notebook users are skilled programmers, but experts are helping to introduce beginners to coding through the software (see 'IPython for beginners'). Yan Song, a postdoc at the University of California, San Diego, had no programming experience until about three months ago. She works on the 'wet' side of a cellular and molecular medicine lab, where she designs experiments and collects data that computational scientists — on the 'dry' side — help her to mine for information.

Song looks for changes in RNA expression in mouse and human stem cells as they differentiate into various types of neuron. In the past, she used Excel to compare expression patterns between groups of cells at different developmental stages. But earlier this year, she began to examine RNA sequencing data from single cells and her data sets exploded in size and complexity. Instead of analysing a few groups of cells, she had to compare hundreds of cells at once, and in each one she examined around 1,500 separate genes related to neural development.

Olga Botvinnik, a bioinformatics graduate student in that lab, started to generate the results in an IPython notebook, so Song began playing with the analysis code — out of a mixture of curiosity and impatience, she says. "It seemed to be an easy interface. You can code one line and you can see whether it works right away."

Within a few weeks, Song had picked up some basic IPython programming skills, finding support through online tutorials and message-boards. Botvinnik has also written some customized menus and widgets to let Song explore her data using different clustering algorithms.

Song still relies on Botvinnik for help with intensive computational analyses, but says that she is now starting to explore the data on her own, using her biological knowledge to examine particular subsets of cells or genes, which she can suggest to Botvinnik as leads for further analysis. "We used to speak two different languages. I would talk about the biology and she would talk about coding. Now we have common ground; we can communicate to each other better. This accelerates our research," she says. ■