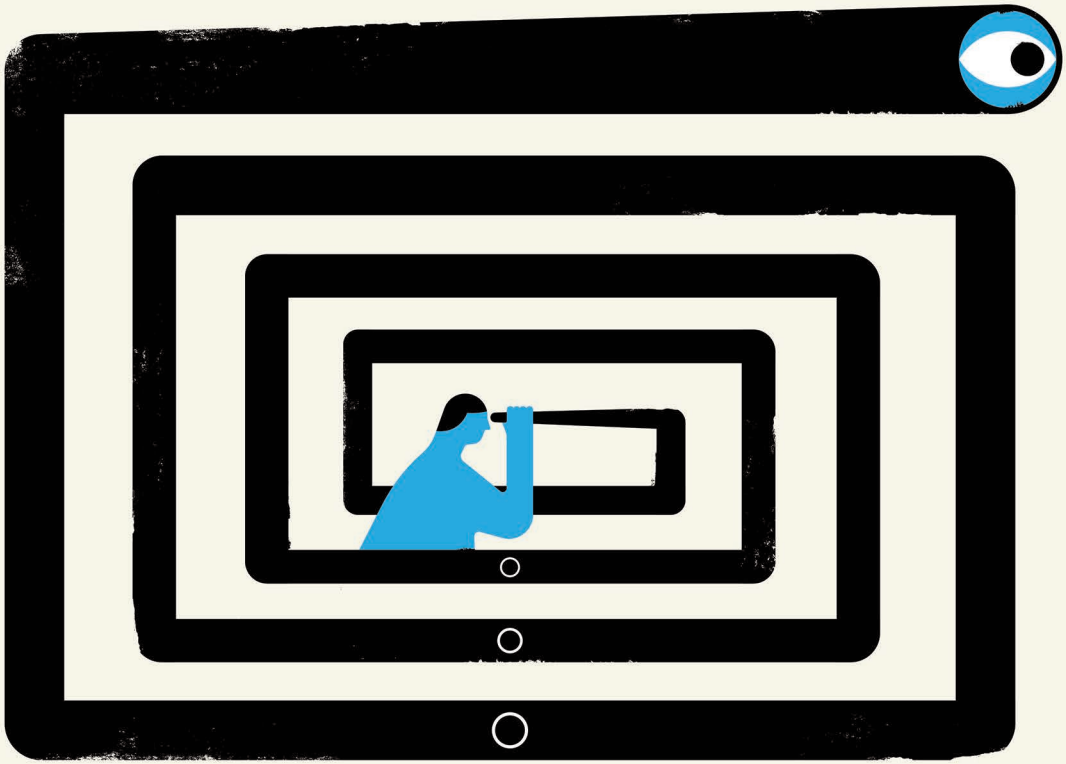

in Washington DC. Mario Jurić, an astronomer at the University of Washington in Seattle who coordinates the LSST's data-management team, says: "I've never seen any migration this fast. It's just amazing."

DATA EXPLORATION

Computational notebooks are essentially laboratory notebooks for scientific computing. Instead of pasting, say, DNA gels alongside lab protocols, researchers embed code, data and text to document their computational methods. The result, says Jupyter co-creator Brian Granger at California Polytechnic State University in San Luis Obispo, is a "computational narrative" — a document that allows researchers to supplement their code and data with analysis, hypotheses and conjecture.

For data scientists, that format can drive exploration. Notebooks, Barba says, are a form of interactive computing, an environment in which users execute code, see what happens, modify and repeat in a kind of iterative conversation between researcher and data. They aren't the only forum for such conversations — IPython, the interactive Python interpreter on which Jupyter's predecessor, IPython Notebook, was built, is another. But notebooks allow users to document those conversations, building "more powerful connections between topics, theories, data and results", Barba says.

Researchers can also use notebooks to create tutorials or interactive manuals for their software. This is what Mackenzie Mathis, a systems neuroscientist at Harvard University in Cambridge, Massachusetts, did for DeepLabCut, a programming library her team developed for behavioural-neuroscience research. And they can use notebooks to prepare manuscripts, or as teaching aids. Barba, who has implemented notebooks in every course she has taught since 2013, related at a keynote address in 2014 that notebooks allow her students to interactively engage with — and absorb material from — lessons in a way that lectures cannot match. "IPython notebooks are really a killer app for teaching computing in science and engineering," she said.

SPEAK MY LANGUAGE

The Jupyter notebook has two components. Users input programming code or text in rectangular cells in a front-end web page. The browser then passes that code to a back-end 'kernel', which runs the code and returns the results (see our example at go.nature.com/2yqq7ak). By Pérez's count, more than 100 Jupyter kernels have been created, supporting dozens of programming languages. Normally, each notebook can run only one kernel and one language, but workarounds exist. One demo notebook, for instance, speaks Python, Julia, R and Fortran.

Importantly, the kernels need not reside on the user's computer. When future users of the LSST use Jupyter notebooks to analyse their data, the code will be running on a supercomputer in Illinois, providing computational muscle no desktop PC could match. Notebooks can also run in the cloud. Google's Colaboratory project, for instance, provides a Google-themed front-end to the Jupyter notebook. It enables users to collaborate and run code that exploits Google's cloud resources — such as graphical processing units — and to save their documents on Google Drive.

Jupyter's newest variant is JupyterLab, which launched as a beta in January 2018 and is available (like the Jupyter notebook) either as a stand-alone package or as part of the free Anaconda scientific-computing environment.

Jason Grout is a software engineer at the financial-services company Bloomberg in San Francisco, California, and a member of the JupyterLab team. He calls JupyterLab a "next-generation web interface" for the Jupyter notebook — one that extends the familiar notebook metaphor with drag-and-drop functionality, as well as file browsers, data viewers, text editors and a command console. Whereas the standard Jupyter notebook assigns each notebook its own kernel, JupyterLab creates a computing environment that allows these components to be shared. Thus, a user could view a notebook in one window, edit a required data file in another, and log all executed commands in a third — all within a single web-browser interface.

"We went from Jupyter notebooks not existing some six years ago to in essence everybody using them today."

Users can also customize JupyterLab to fit their workflow. Built-in viewers exist for image, text and CSV files, for instance, but users can build custom components as well. These could display things such as genomic alignments or geospatial data. An attendee on a course taught by Pérez even created a component to display 3D brain-imaging data. "This is a completely [neuroscience] domain-specific tool, obviously — the Jupyter team has no business writing these things. But we provide the right standards, and then that community in 24 hours can come back and write one," he says.

Two additional tools have enhanced Jupyter's usability. One is JupyterHub, a service that allows institutions to provide Jupyter notebooks to large pools of users. The IT team at the University of California, Berkeley, where Pérez is a faculty member, has deployed one such hub, which Pérez uses to ensure that all students on his data-science course have identical computing environments. "We cannot possibly manage IT support for 800 students, helping them debug why the installation on their laptop is not working; that's simply infeasible," he says.

The other development is Binder, an open-source service that allows users to use Jupyter notebooks on GitHub in a web browser without having to install the software or any programming libraries. Users can also execute Jupyter notebooks on the Google cloud by inserting https://colab.research.google.com/github before the URL of a notebook on GitHub, or using the commercial service Code Ocean. In September, Code Ocean rolled out a new user interface for its cloud-based code-sharing and code-execution service, also based on Jupyter.

PROBLEMS NOTED

Such tools foster computational reproducibility by simplifying code reuse. But users still need to know how to use notebooks correctly.

Joel Grus, a research engineer at the Allen Institute for Artificial Intelligence in Seattle, Washington, gave a presentation titled 'I don't like notebooks' at the Jupyter developers' conference earlier this year in New York City. He says he has seen programmers get frustrated when notebooks don't behave as expected, usually because they inadvertently run code cells out of order. Jupyter notebooks also encourage poor coding practice, he says, by making it difficult to organize code logically, break it into reusable modules and develop tests to ensure the code is working properly.

Those aren't insurmountable issues, Grus concedes, but notebooks do require discipline when it comes to executing code: for instance, by moving analysis code to external files that can be called from the notebook, by defining key variables at the top of the notebook and by restarting the kernel periodically and running the notebook from top to bottom. As one Twitter user quipped, "Restart and run all or it didn't happen."

That's a lesson Barba tries to instil in her students. "I explain to my students from day one that they can interact with a notebook in a nonlinear fashion, and that gives them great power for exploration," she says. "But with great power comes great responsibility."

One tool that might help is Verdant, a plug-in that captures a history of a user's actions in Jupyter. "The authors built an extension that allows a flexible user workflow while also capturing the specific code executed, in what order and on what specific data," says Carol Willing, a member of the Jupyter team at California Polytechnic State University.

Jake VanderPlas, a software engineer at Google in Seattle, Washington, and a member of the Colaboratory team, says notebooks are like hammers: they can be misused, and aren't appropriate for every application. But for data exploration and communication, notebooks excel. The astronomy community seemingly agrees. "We went from Jupyter notebooks not existing some six years ago to in essence everybody using them today," says Jurić. "And we're a community that still has Fortran 77" — as in 1977 — "sticking around. It's something." ■

Jeffrey M. Perkel *is technology editor at* Nature.