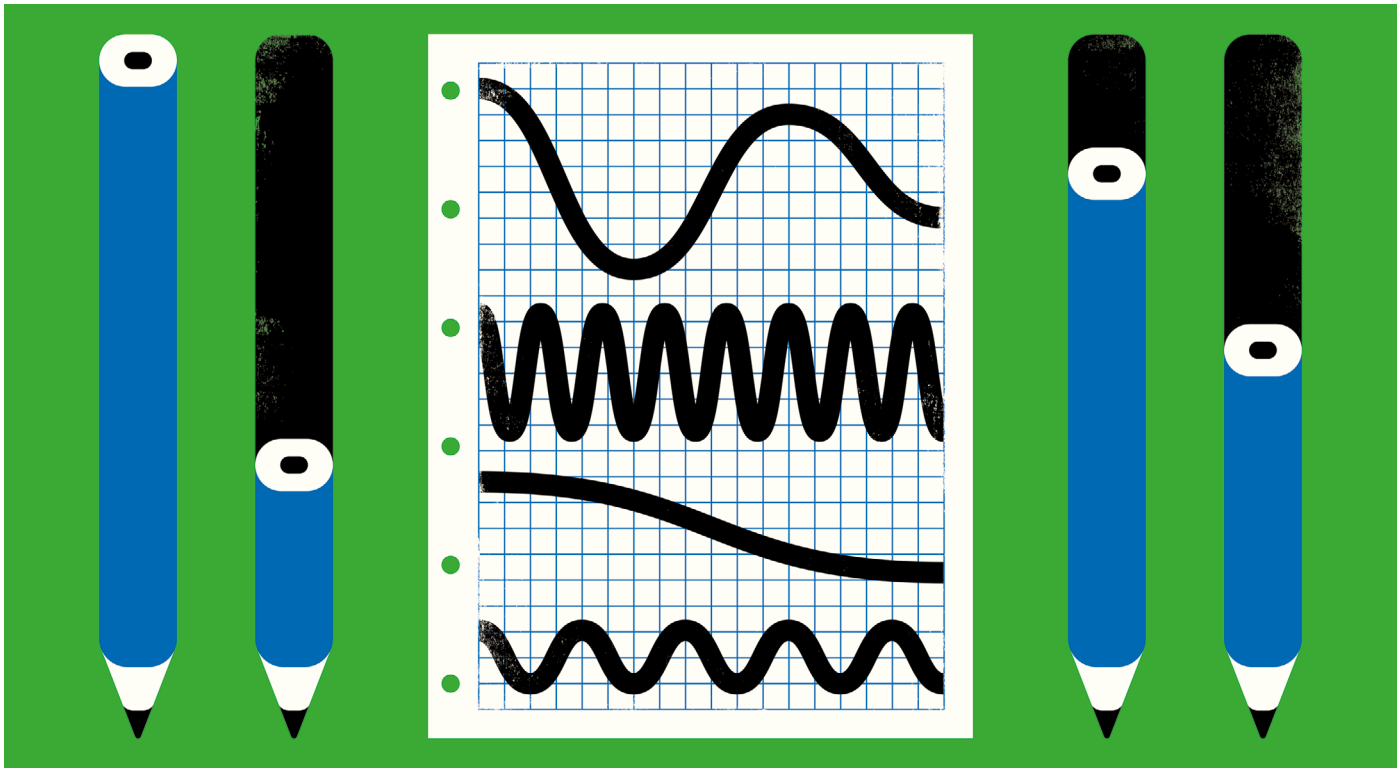

ILLUSTRATION BY THE PROJECT TWINS

REACTIVE, REPRODUCIBLE, COLLABORATIVE: COMPUTATIONAL NOTEBOOKS EVOLVE

A new breed of notebooks is taking data visualization and collaborative functionality to the next level, with spreadsheet simplicity. **By Jeffrey M. Perkel**

This year marks ten years since the launch of the IPython Notebook. The open-source tool, now known as the Jupyter Notebook, has become an exceedingly popular piece of data-science kit, with millions of notebooks deposited to the GitHub code-sharing site.

Computational notebooks combine code, results, text and images in a single document, yielding what Stephen Wolfram, creator of the Mathematica software package, has called a "computational essay". And whether written using Jupyter, Mathematica, RStudio or any other platform, researchers can use them for iterative data exploration, communication, teaching and more.

But computational notebooks can also be confusing and foster poor coding practices. And they are difficult to share, collaborate on and reproduce. A 2019 study found that just 24% of 863,878 publicly available Jupyter notebooks on GitHub could be successfully re-executed, and only 4% produced the same results (J. F. Pimentel *et al.* in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)* 507–517; IEEE, 2019).

"Notebooks are messy," says Anita Sarma, a computer scientist at Oregon State University in Corvallis who studies human–computer interaction. "You write stuff, you keep old crusty code behind, and it's hard to kind of figure out which cells to execute in which order, because you were trying different things."

But a growing suite of platforms and tools aims to smooth these rough edges. Some make notebooks 'reactive', so that code re-executes whenever software variables change; others focus on collaboration and version control. But all provide researchers with innovative ways to explore, document and share their data with colleagues and the world.

For Sergei Pond, notebooks have provided an outlet for documenting the genetics of the pandemic. Pond, a computational biologist at Temple University in Philadelphia, Pennsylvania, has created some three dozen documents related to SARS-CoV-2, the virus that causes COVID-19. "My default setting", he says, is to "write up an interactive notebook and send it to my collaborators so they can play with the data, [so] they can immediately see what's there."

His notebook platform of choice is called Observable. It's based in San Francisco, California, and was founded in 2019 by two Google alumni: Mike Bostock, developer of the D3 JavaScript library that powers many of the interactive data visualizations on the web today, and Melody Meckfessel. The company's web-based notebook system allows users to create, share and reuse sophisticated, interactive visualizations written in JavaScript, the

programming language understood by web browsers. According to Meckfessel, "hundreds of thousands" of users do so every month.

Unlike Jupyter, which passes code to an external 'kernel' that executes it, Observable code runs in the browser itself. That makes the platform fast and responsive, Bostock says. But because JavaScript is not a typical data-science language, researchers often use Observable not for data processing but for visualization.

Pond, for instance, uses Observable to share colourful maps, graphs, protein structures and sequence alignments that represent data that he generates in other software. Observable's modular structure means that other programmers can easily apply those visualizations to their own data. But Pond's notebooks also take advantage of another key Observable feature: reactivity.

Suppose you have a Jupyter notebook that plots a line. In one code cell, you define the slope and *y*-intercept; in the next, you draw the graph. The notebook structure allows coders to return to the earlier cell to change the slope after the plot has been rendered. But that change does not cause the figure to be automatically redrawn; the user must manually re-execute the cell that plots it.

This workflow can lead to 'state problems', in which a notebook's output does not reflect its code — as would happen, for instance, if the user deletes the cell that defines a variable after it has been executed. In 2018, Joel Grus, then a software engineer at the Allen Institute for Artificial Intelligence in Seattle, Washington, highlighted this behaviour, and the ensuing confusion, in a widely viewed talk entitled "I don't like notebooks". But, "to a large degree, having fully reactive notebooks eliminates that feature," he says now.

Reactive notebooks are similar to spreadsheets, Bostock explains. Just as Microsoft Excel knows to recalculate a formula if the underlying cells change, reactive notebooks track how code cells relate to one another to ensure that a notebook's output always reflects its variables.

Combined with visual widgets, such as sliders and pull-down lists, such behaviour makes notebooks interactive, allowing readers to explore how changing variables or assumptions can affect results. Herb Susmann, a biostatistics PhD student at the University of Massachusetts Amherst, for instance, uses reactive documents to explain statistical concepts. "It really helps me get more of a visceral feel for how these statistical things work," he says. (That said, reactivity isn't always desirable, particularly if cells take a long time to execute, or when data sets are very large.)

React and collaborate

Other reactive notebook systems exist for researchers who don't use JavaScript. Susmann, for instance, has built a reactive notebook for R programmers, called Reactor. And Fons van der Plas, a software engineer in Berlin, created Pluto, a reactive notebook platform for the programming language Julia. Henri Drake, a graduate student of climate physics at the Massachusetts Institute of Technology in Cambridge, uses Pluto to demonstrate concepts in climate science. "Coding it up as an interactive Pluto notebook makes it a way more engaging experience for a first-time user," Drake says, "and can really help people understand the models that I'm building."

Fernando Pérez, a co-founder of Project Jupyter at the University of California, Berkeley, notes that Jupyter itself "is agnostic on the topic of reactivity". Most kernels so far have been non-reactive, but they don't have to be: Richa Gadgil, a former Jupyter intern at California Polytechnic State University in San Luis Obispo, for instance, spent her internship co-developing an experimental reactive kernel for Python. "It was a test of the Jupyter architecture and the Jupyter architecture passed that test," says Brian Granger, who directed her work.

"It really helps me get more of a visceral feel for how these statistical things work."

Another open-source system, called Vizier, focuses on data-driven reactivity, says Juliana Freire, a computer scientist at New York University, who co-directed the project. With built-in data validation functions and a spreadsheet interface, Vizier users can massage their data to fix inconsistencies — such as those caused by a column that contains both 'Y/N' and 'yes/no' responses. As they do so, the notebook re-executes. "You analyse, you clean, you analyse, you clean," Freire says. "And as you do that, you save the whole provenance of the process." As a result, users can revert to an earlier stage of clean-up and try again, all the time logging the changes that they have made. (Vizier notebooks can handle Python, SQL and Scala code.)

Some commercial reactive systems, including Observable, Deepnote and JetBrains' Datalore (the last two of which are based in the Czech Republic), also emphasize another notebook pain point: collaboration. Observable, for instance, allows real-time collaborative editing, much as Google Docs does, as well as commenting. There are two plan tiers: Personal (free for up to 5 members in the same interactive document) and Teams (for 6 or more members: US$15 per editor, per month; free for viewers).

Gábor Csányi, who studies molecular modelling at the University of Cambridge, UK, uses Deepnote (free for up to 3 collaborators, then $12 per user, per month) in his teaching. With his university's previous system, a student seeking help could share a copy of a notebook with Csányi, but it wasn't possible for both of them to view and edit the same document at the same time. "It was sort of a pain," he says. But with Deepnote, he can help students to debug their code in real time. "Just like you do with Google Docs, we see each other's cursors. We are editing the same notebook, and as they press shift-enter on a cell, I see the result. That was an incredible experience in how personalized support could be done efficiently."

Real-time collaboration is "a topic of massive activity" in the Jupyter project as well, says Pérez, and an in-development prototype is available on GitHub. "I'm pretty optimistic that this will happen soonish," he says.

Version control

Many commercial platforms also address another notebook challenge: version control. The file format of Jupyter notebooks includes code, metadata, and computational output. As those outputs are often binary images, version control — the process that developers use to track how files change, which is optimized for plain-text files — can become difficult. Complicating matters, programmers can struggle to adapt standard version-control workflows to the fast, iterative nature of data exploration. As a result, crucial experimental details can be lost.

Commercial platforms tend to provide built-in notebook versioning. For those who prefer to stick with Jupyter, two plug-ins are available: nbdime, which provides an intelligent, structured view of file changes, including of graphical output; and Verdant, which offers a graphical interface that tracks how cells are modified, reordered and executed.

According to developer Mary Beth Kery, who studies human–computer interaction at Carnegie Mellon University in Pittsburgh, Pennsylvania, Verdant can smooth interactions with collaborators and peer reviewers. "Somebody will say, oh, did you try this in the model, or did you try this analysis?" she says. Many times, the answer is yes, but because the analysis didn't work, the code is deleted. "What you want to do during the meeting is just pull it back up and be like, oh, yeah, I did, and here's why it didn't work. And our tool lets you actually do that."

Such features can make an already user-friendly computing paradigm even friendlier — and easier to share. And that makes them even more powerful vehicles for scientific communication. "If you do really great science but no one understands it or no one gets access to it, then what's the point?" Drake says. "These kinds of notebooks can really get people excited and expose people to concepts that are otherwise kind of impenetrable."

Jeffrey M. Perkel is *Nature*'s Technology Editor.