The Atlantic

The Scientific Paper Is Obsolete

Here's what's next.

JAMES SOMERS

PNAS / Richard Goerg / Getty / The Atlantic

APR 5, 2018 | SCIENCE

THE SCIENTIFIC paper—the actual form of it—was one of the enabling inventions of modernity. Before it was developed in the 1600s, results were communicated privately in letters, ephemerally in lectures, or all at once in books. There was no public forum for *incremental* advances. By making room for reports of single experiments or minor technical advances, journals made the chaos of science accretive. Scientists from that point forward became like the social insects: They made their progress steadily, as a buzzing mass.

The earliest papers were in some ways more readable than papers are today. They were less specialized, more direct, shorter, and far less formal. Calculus had only just been invented. Entire data sets could fit in a table on a single page. What little "computation" contributed to the results was done by hand and could be verified in the same way.

The more sophisticated science becomes, the harder it is to communicate results. Papers today are longer than ever and full of jargon and symbols. They depend on chains of computer programs that generate data, and

clean up data, and plot data, and run statistical models on data. These programs tend to be both so sloppily written and so central to the results that it's contributed to a replication crisis, or put another way, a failure of the paper to perform its most basic task: to report what you've actually discovered, clearly enough that someone else can discover it for themselves.

Perhaps the paper itself is to blame. Scientific methods evolve now at the speed of software; the skill most in demand among physicists, biologists, chemists, geologists, even anthropologists and research psychologists, is facility with programming languages and "data science" packages. And yet the basic means of communicating scientific results hasn't changed for 400 years. Papers may be posted online, but they're still text and pictures on a page.

What would you get if you designed the scientific paper from scratch today? A little while ago I spoke to Bret Victor, a researcher who worked at Apple on early user-interface prototypes for the iPad and now runs his own lab in Oakland, California, that studies the future of computing. Victor has long been convinced that scientists haven't yet taken full advantage of the computer. "It's not that different than looking at the printing press, and the evolution of the book," he said. After Gutenberg, the printing press was mostly used to mimic the calligraphy in bibles. It took nearly 100 years of technical and conceptual improvements to invent the modern book. "There was this entire period where they had the new technology of printing, but they were just using it to emulate the old media."

Victor gestured at what might be possible when he redesigned a journal article by Duncan Watts and Steven Strogatz, "Collective dynamics of 'small-world' networks." He chose it both because it's one of the most highly cited papers in all of science and because it's a model of clear

exposition. (Strogatz is best known for writing the beloved "Elements of Math" column for *The New York Times*.)

The Watts-Strogatz paper described its key findings the way most papers do, with text, pictures, and mathematical symbols. And like most papers, these findings were still hard to swallow, despite the lucid prose. The hardest parts were the ones that described procedures or algorithms, because these required the reader to "play computer" in their head, as Victor put it, that is, to strain to maintain a fragile mental picture of what was happening with each step of the algorithm.

Victor's redesign interleaved the explanatory text with little interactive diagrams that illustrated each step. In his version, you could *see* the algorithm at work on an example. You could even control it yourself.

Bret Victor

Strogatz admired Victor's design. He later told me that it was a shame that in mathematics it's been a tradition for hundreds of years to make papers as formal and austere as possible, often suppressing the very visual aids that mathematicians use to make their discoveries.

Strogatz studies nonlinear dynamics and chaos, systems that get into sync or self-organize: fireflies flashing, metronomes ticking, heart cells firing

electrical impulses. The key is that these systems go through cycles, which Strogatz visualizes as dots running around circles: When a dot comes back to the place where it started—that's a firefly flashing or a heart cell firing. "For about 25 years now I've been making little computer animations of dots running around circles, with colors indicating their frequency," he said. "The red are the slow guys, the purple are the fast guys … I have these colored dots swirling around on my computer. I do this all day long," he said. "I can see patterns much more readily in colored dots running, moving on the screen than I can in looking at 500 simultaneous time series. I don't *see* stuff very well like that. Because it's not what it really looks like … What I'm studying is something dynamic. So the representation should be dynamic."

Software is a dynamic medium; paper isn't. When you think in those terms it does seem strange that research like Strogatz's, the study of dynamical systems, is so often being shared on paper, without the benefit of his little swirling dots—because it's the swirling dots that helped him to see what he saw, and that might help the reader see it too.

This is, of course, the whole problem of scientific communication in a nutshell: Scientific results today are as often as not found with the help of computers. That's because the ideas are complex, dynamic, hard to grab ahold of in your mind's eye. And yet by far the most popular tool we have for communicating these results is the PDF—literally a simulation of a piece of paper. Maybe we can do better.

* * *

STEPHEN WOLFRAM published his first scientific paper when he was 15. He had published 10 when he finished his undergraduate career, and by the time he was 20, in 1980, he'd finished his Ph.D. in particle physics from the California Institute of

Technology. His secret weapon was his embrace of the computer at a time when most serious scientists thought computational work was beneath them. "By that point, I think I was the world's largest user of computer algebra," he said in a talk. "It was so neat, because I could just compute all this stuff so easily. I used to have fun putting incredibly ornate formulas in my physics papers."

As his research grew more ambitious, he found himself pushing existing software to its limit. He'd have to use half a dozen programming tools in the course of a single project. "A lot of my time was spent gluing all this stuff together," he said. "What I decided was that I should try to build a single system that would just do all the stuff I wanted to do—and that I could expect to keep growing forever." Instead of continuing as an academic, Wolfram decided to start a company, Wolfram Research, to build the perfect computing environment for scientists. A headline in the April 18, 1988, edition of *Forbes* pronounced: "Physics Whiz Goes Into Biz."

At the heart of Mathematica, as the company's flagship product became known, is a "notebook" where you type commands on one line and see the results on the next. Type "1/6 + 2/5" and it'll give you "17/30." Ask it to factor a polynomial and it will comply. Mathematica can do calculus, number theory, geometry, algebra. But it also has functions that can calculate how chemicals will react, or filter genomic data. It has in its knowledge base nearly every painting in Rembrandt's *oeuvre* and can give you a scatterplot of his color palette over time. It has a model of orbital mechanics built in and can tell you how far an F/A-18 Hornet will glide if its engines cut out at 32,000 feet. A Mathematica notebook is less a record of the user's calculations than a transcript of their conversation with a polymathic oracle. Wolfram calls carefully authored Mathematica notebooks "computational essays."

The notebook interface was the brainchild of Theodore Gray, who was inspired while working with an old Apple code editor. Where most programming environments either had you run code one line at a time, or all at once as a big blob, the Apple editor let you highlight any part of your code and run just that part. Gray brought the same basic concept to Mathematica, with help refining the design from none other than Steve Jobs. The notebook is designed to turn scientific programming into an interactive exercise, where individual commands were tweaked and rerun, perhaps dozens or hundreds of times, as the author learned from the results of their little computational experiments, and came to a more intimate understanding of their data.

"It's incalculable, literally ... how much is lost, and how much time is wasted."

What made Mathematica's notebook especially suited to the task was its ability to generate plots, pictures, and beautiful mathematical formulas, and to have this output respond dynamically to changes in the code. In Mathematica you can input a voice recording, run complex mathematical filters over the audio, and visualize the resulting sound wave; just by mousing through and adjusting parameters, you can warp the wave, discovering which filters work best by playing around. Mathematica's ability to fluidly handle so many different kinds of computation in a single, simple interface is the result, Gray says, of "literally man-centuries of work."

The vision driving that work, reiterated like gospel by Wolfram in his many lectures, blog posts, screencasts, and press releases, is not merely to make a good piece of software, but to create an inflection point in the enterprise of science itself. In the mid-1600s, Gottfried Leibniz devised a

notation for integrals and derivatives (the familiar ∫ and dx/dt) that made difficult ideas in calculus almost mechanical. Leibniz developed the sense that a similar notation applied more broadly could create an "algebra of thought." Since then, logicians and linguists have lusted after a universal language that would eliminate ambiguity and turn complex problem-solving of all kinds into a kind of calculus.

Wolfram's career has been an ongoing effort to vacuum up the world's knowledge into Mathematica, and later, to make it accessible via Wolfram Alpha, the company's "computational knowledge engine" that powers many of Siri and Alexa's question-answering abilities. It is Wolfram's own attempt to create an Interlingua, a programming language equally understandable by humans and machines, an algebra of everything.

It is a characteristically grandiose ambition. In the 1990s, Wolfram would occasionally tease in public comments that at the same time he was building his company, he was quietly working on a revolutionary science project, years in the making. Anticipation built. And then, finally, the thing itself arrived: a gargantuan book, about as wide as a cinder block and nearly as heavy, with a title for the ages—*A New Kind of Science*.

It turned out to be a detailed study, carried out in Mathematica notebooks, of the surprisingly complex patterns generated by simple computational processes—called cellular automata—both for their own sake and as a way of understanding how simple rules can give rise to complex phenomena in nature, like a tornado or the pattern on a mollusk shell. These explorations, which Wolfram published without peer review, came bundled with reminders, every few pages, about how important they were.

The more of Wolfram you encounter, the more this seems to be his nature. The 1988 *Forbes* profile about him tried to get to the root of it: "In the

words of Harry Woolf, the former director of the prestigious Institute for Advanced Study in [Princeton, New Jersey]—where Wolfram, at 23, was one of the youngest senior researchers ever—he has 'a cultivated difficulty of character added to an intrinsic sense of loneliness, isolation, and uniqueness.'"

When one of Wolfram's research assistants announced a significant mathematical discovery at a conference, which was a core part of *A New Kind of Science*, Wolfram threatened to sue the hosts if they published it. "You won't find any serious research group that would let a junior researcher tell what the senior researcher is doing," he said at the time. Wolfram's massive book was panned by academics for being derivative of other work and yet stingy with attribution. "He insinuates that he is largely responsible for basic ideas that have been central dogma in complex systems theory for 20 years," a fellow researcher told the *Times Higher Education* in 2002.

Wolfram's self-aggrandizement is especially vexing because it seems unnecessary. His achievements speak for themselves—if only he'd let them. Mathematica was a success almost as soon as it launched. Users were hungry for it; at universities, the program soon became as ubiquitous as Microsoft Word. Wolfram, in turn, used the steady revenue to hire more engineers and subject-matter experts, feeding more and more information to his insatiable program. Today Mathematica knows about the anatomy of a foot and the laws of physics; it knows about music, the taxonomy of coniferous trees, and the major battles of World War I. Wolfram himself helped teach the program an archaic Greek notation for numbers.

All of this knowledge is "computable": If you wanted, you could set "x" to be the location of the Battle of the Somme and "y" the daily precipitation,

in 1916, within a 30-mile radius of that point, and use Mathematica to see whether World War I fighting was more or less deadly in the rain.

Courtesy of *Stephen Wolfram Blog*

"I've noticed an interesting trend," Wolfram wrote in a blog post. "Pick any field X, from archeology to zoology. There either is now a 'computational X' or there soon will be. And it's widely viewed as the future of the field." As practitioners in those fields become more literate with computation, Wolfram argues, they'll vastly expand the range of what's discoverable. The Mathematica notebook could be an accelerant

for science because it could spawn a new kind of thinking. "The place where it really gets exciting," he says, "is where you have the same transition that happened in the 1600s when people started to be able to read math notation. It becomes a form of communication which has the incredibly important extra piece that you can actually run it, too."

The idea is that a "paper" of this sort would have all the dynamism Strogatz and Victor wanted—interactive diagrams interleaved within the text—with the added benefit that all the code generating those diagrams, and the data behind them, would be right there for the reader to see and play with. "Frankly, when you do something that is a nice clean Wolfram-language thing in a notebook, there's no bullshit there. It is what it is, it does what it does. You don't get to fudge your data," Wolfram says.

To write a paper in a Mathematica notebook is to reveal your results and methods at the same time; the published paper *and* the work that begot it. Which shouldn't just make it easier for readers to understand what you did—it should make it easier for them to replicate it (or not). With millions of scientists worldwide producing incremental contributions, the only way to have those contributions add up to something significant is if others can reliably build on them. "That's what having science presented as computational essays can achieve," Wolfram said.

Wolfram says that he's surprised the computational essay hasn't taken off. He remembers working with Elsevier, the scientific publishing giant, all the way back in the early '80s. "Elsevier hired me to do some consulting thing about 'What would the future of scientific publishing look like?'" This was before the Mathematica notebook, but he gave them a spiel along the same lines. "A few years ago I was talking to some of their upper management again. I realized in this meeting, oh my gosh, I said exactly the same things 35 years ago!"

I spoke to Theodore Gray, who has since left Wolfram Research to become a full-time writer. He said that his work on the notebook was in part motivated by the feeling, well formed already by the early 1990s, "that *obviously* all scientific communication, all technical papers that involve any sort of data or mathematics or modeling or graphs or plots or anything like that, *obviously* don't belong on paper. That was just *completely obvious* in, let's say, 1990," he said.

"It's been a source of ongoing bafflement and consternation for the past 29 years, that with the exception of a few people who get it, the community at large hasn't really adopted it," he said. "It's incalculable, literally … how much is lost, and how much time is wasted, and how many results are misinterpreted or are misrepresented."

* * *

IN EARLY 2001, Fernando Pérez found himself in much the same position Wolfram had 20 years earlier: He was a young graduate student in physics running up against the limits of his tools. He'd been using a hodgepodge of systems, Mathematica among them, feeling as though every task required switching from one to the next. He remembered having six or seven different programming-language books on his desk. What he wanted was a unified environment for scientific computing.

But rather than going off and building a company, he found two far-flung scientists, a German oceanographer and a computer-science graduate student at Caltech, who had been thinking along the same lines. They'd all fallen in love with Python, an open-source, general-purpose programming language, and they'd all independently started building tools to make it work better for scientists: tools that made it easier to manage data sets and draw plots, and that encouraged a more exploratory programming style.

Pérez combined the three projects into one and took the reins. From the very beginning, the project, called IPython (the "I" for "Interactive"), was open-source: It wasn't just that the program was free, but that anyone could inspect its code and modify it, contributing their changes back to the common cause. It was a deliberate decision. "I was interested in both the ethical aspect of being able to share my work with others," Pérez told me—he came from Colombia, where access to proprietary software was harder to come by—"and there was also a more epistemological motivation." He thought that if science was to be an open enterprise, the tools that are used to do it should themselves be open. Commercial software whose source code you were legally prohibited from reading was "antithetical to the idea of science," where the very purpose is to open the black box of nature.

"You're effectively using the computer as a thinking partner."

Hence the allure of Python. Out of the box, Python is a much less powerful language than the Wolfram Language that powers Mathematica. But where Mathematica gets its powers from an army of Wolfram Research programmers, Python's bare-bones core is supplemented by a massive library of extra features—for processing images, making music, building AIs, analyzing language, graphing data sets—built by a community of open-source contributors working for free. Python became a de facto standard for scientific computing because open-source developers like Pérez happened to build useful tools for it; and open-source developers have flocked to Python because it happens to be the de facto standard for scientific computing. Programming-language communities, like any social network, thrive—or die—on the strength of these feedback loops.

The idea for IPython's notebook interface came from Mathematica. Pérez admired the way that Mathematica notebooks encouraged an exploratory style. "You would sketch something out—because that's how you *reason* about a problem, that's how you understand a problem." Computational notebooks, he said, "bring that idea of live narrative out … You can think through the process, and you're effectively using the computer, if you will, as a computational partner, and as a thinking partner."

Instead of building a specialized, stand-alone application, let alone spending man-centuries on it, the IPython team—Pérez was now joined by Brian Granger, a physics professor at California Polytechnic State University at San Luis Obispo; and Min Ragan-Kelley, a Ph.D. student at UC Berkeley working in computational physics—built their notebooks as simple web pages. The interface is missing Mathematica's Steve Jobsian polish, and its sophistication. But by latching itself to the web, IPython got what is essentially free labor: Any time Google, Apple, or a random programmer open-sourced a new plotting tool, or published better code for rendering math, the improvement would get rolled into IPython. "It has paid off handsomely," Pérez said.

The paper announcing the first confirmed detection of gravitational waves was published in the traditional way, as a PDF, but with a supplemental IPython notebook. The notebook walks through the work that generated every figure in the paper. Anyone who wants to can run the code for themselves, tweaking parts of it as they see fit, playing with the calculations to get a better handle on how each one works. At a certain point in the notebook, it gets to the part where the signal that generated the gravitational waves is processed into sound, and this you can play in your browser, hearing for yourself what the scientists heard first, the bloop of two black holes colliding.

“I think what they have is acceptance from the scientific community as a tool that is considered to be universal,” Theodore Gray says of Pérez’s group. “And that’s the thing that Mathematica never really so far has achieved.” There are now 1.3 million of these notebooks hosted publicly on Github. They’re in use at Google, Bloomberg, and NASA; by musicians, teachers, and AI researchers; and in “almost every country on Earth.”

At every turn, IPython chose the way that was more inclusive, to the point where it’s no longer called “IPython”: The project rebranded itself as “Jupyter” in 2014 to recognize the fact that it was no longer just for Python. The Jupyter notebook, as it’s called, is like a Mathematica notebook but for any programming language. You can have a Python notebook, or a C notebook, or an R notebook, or Ruby, or Javascript, or Julia. Anyone can build support for their programming language in Jupyter. Today it supports more than 100 languages.

Theodore Gray, who developed the original Mathematica notebook interface, said that he once as an experiment tried to build support for other programming languages into it. “It never went anywhere,” he told

me. “The company had no interest in supporting this. And also because when you have to support a lot of different languages, you can’t do it as deeply.”

“I’m all in favor of there being a maniac in the middle.”

A 1997 essay by Eric S. Raymond titled “The Cathedral and the Bazaar,” in some sense the founding document of the modern open-source movement, challenged the notion that complex software had to be built like a cathedral, “carefully crafted by individual wizards or small bands of mages working in splendid isolation.” Raymond’s experience as one of the stewards of the Linux kernel (a piece of open-source software that powers all of the world’s 500 most powerful supercomputers, and the vast majority of mobile devices) taught him that the “great babbling bazaar of differing agendas and approaches” that defined open-source projects was actually a strength. “The fact that this bazaar style seemed to work, and work well, came as a distinct shock,” he wrote. The essay was his attempt to reckon with why “the Linux world not only didn’t fly apart in confusion but seemed to go from strength to strength at a speed barely imaginable to cathedral builders.”

Mathematica was in development long before Raymond’s formative experience with Linux, and has been in development long after it. It is the quintessential cathedral, and its builders are still skeptical of the bazaar. “There’s always chaos,” Gray said about open-source systems. “The number of moving parts is so vast, and several of them are under the control of different groups. There’s no way you could ever pull it together into an integrated system in the same way as you can in a single commercial product with, you know, a single maniac in the middle.”

The maniac is, of course, Stephen Wolfram. Gray pointed out that the trains ran on time under Mussolini. "It's a bad analogy," he said, but still, "I'm all in favor of there being a maniac in the middle." The Mathematica notebook is the more coherently designed, more polished product—in large part because every decision that went into building it emanated from the mind of a single, opinionated genius. "I see these Jupyter guys," Wolfram said to me, "they are about on a par with what we had in the early 1990s." They've taken shortcuts, he said. "We actually want to try and do it right."

But it's hard to sell the scientific community on a piece of commercial software. Even though Wolfram Research has given away a free Mathematica notebook viewer for years, and even though most major universities already have a site license that lets their students and faculty use Mathematica freely, it might be too much to ask publishers to abandon PDFs, an open format, for a proprietary product. "Right now if you make a Mathematica notebook and you try to send that to a journal," Gray says, "they're gonna complain: Well, we don't have Mathematica, this is an expensive product—give us something that's more of a standard."

It doesn't help that Wolfram—the man, and to some extent the company—touts Mathematica's superiority, its *necessity*, in a way that even Gray describes as being a bit like how CrossFit people won't shut up about CrossFit. This is, after all, the same Stephen Wolfram who titled a book about his own work on cellular automata *A New Kind of Science*. In his blog post about computational essays, he writes, "At the core of computational essays is the idea of expressing computational thoughts using the Wolfram Language."

Google Trends

Which, who knows, might well be true—maybe computational notebooks will only take root if they're backed by a single super-language, or by a company with deep pockets and a vested interest in making them work. But it seems just as likely that the opposite is true. A federated effort, while more chaotic, might also be more robust—and the only way to win the trust of the scientific community.

Wolfram has a hard time seeing outside Wolfram, and perhaps that's the reason that the Mathematica notebook has remained relatively obscure while a rival system—derivative, yes, and simplistic, but open—looks to be taking over the world.

* * *

IT'LL BE SOME time before computational notebooks replace PDFs in scientific journals, because that would mean changing the incentive structure of science itself. Until journals *require* scientists to submit notebooks, and until sharing your work and your data becomes the way to earn prestige, or funding, people will likely just keep doing what they're doing.

I spoke to a neuroscientist-turned-software-developer who contributed to Jupyter, who told me that the professor in charge of his former lab was originally an electrophysiologist—he actually measured neuronal activity with implanted electrodes. "This kind of data is basically so costly, so expensive, and so good," he said, that nobody would ever share it. "You collect one batch of data and you can mine it for the rest of your career."

"At this point, nobody in their sane mind challenges the fact that the *praxis* of scientific research is under major upheaval," Pérez, the creator of Jupyter, wrote in a blog post in 2013. As science becomes more about computation, the skills required to be a good scientist become increasingly attractive in industry. Universities lose their best people to start-ups, to Google and Microsoft. "I have seen many talented colleagues leave academia in frustration over the last decade," he wrote, "and I can't think of a single one who wasn't *happier* years later."

Pérez told me stories of scientists who sacrificed their academic careers to build software, because building software counted for so little in their field: The creator of matplotlib, probably the most widely used tool for generating plots in scientific papers, was a postdoc in neuroscience but had to leave academia for industry. The same thing happened to the creator of NumPy, a now-ubiquitous tool for numerical computing. Pérez himself said, "I did get straight-out blunt comments from many, many colleagues, and from senior people and mentors who said: Stop doing this, you're wasting your career, you're wasting your talent." Unabashedly, he said, they'd tell him to "go back to physics and mathematics and writing papers."

Still, those who stay are making progress. Pérez himself recently got a faculty appointment in the stats department at Berkeley. The day after we spoke, he was slated to teach an upper-division data-science course, built

entirely on Python and Jupyter notebooks. “The freshman version of that course had in the fall I think 1,200 students,” he said. “It’s been the fastest-growing course in the history of UC Berkeley. And it’s all based on these open-source tools.”

When you improve the praxis of science, the dream is that you’ll improve its products, too. Leibniz’s notation, by making it easier to do calculus, expanded the space of what it was possible to think. The grand scientific challenges of our day are as often as not computational puzzles: How to integrate billions of base pairs of genomic data, and 10 times that amount of proteomic data, and historical patient data, and the results of pharmacological screens into a coherent account of how somebody got sick and what to do to make them better? How to make actionable an endless stream of new temperature and precipitation data, and oceanographic and volcanic and seismic data? How to build, and make sense of, a neuron-by-neuron map of a thinking brain? Equipping scientists with computational notebooks, or some evolved form of them, might bring their minds to a level with problems now out of reach.

At one point, Pérez told me the name Jupyter honored Galileo, perhaps the first modern scientist. The Jupyter logo is an abstracted version of Galileo’s original drawings of the moons of Jupiter. “Galileo couldn’t go anywhere to buy a telescope,” Pérez said. “He had to build his own.”

ABOUT THE AUTHOR

JAMES SOMERS is a contributing editor at *The Atlantic*.