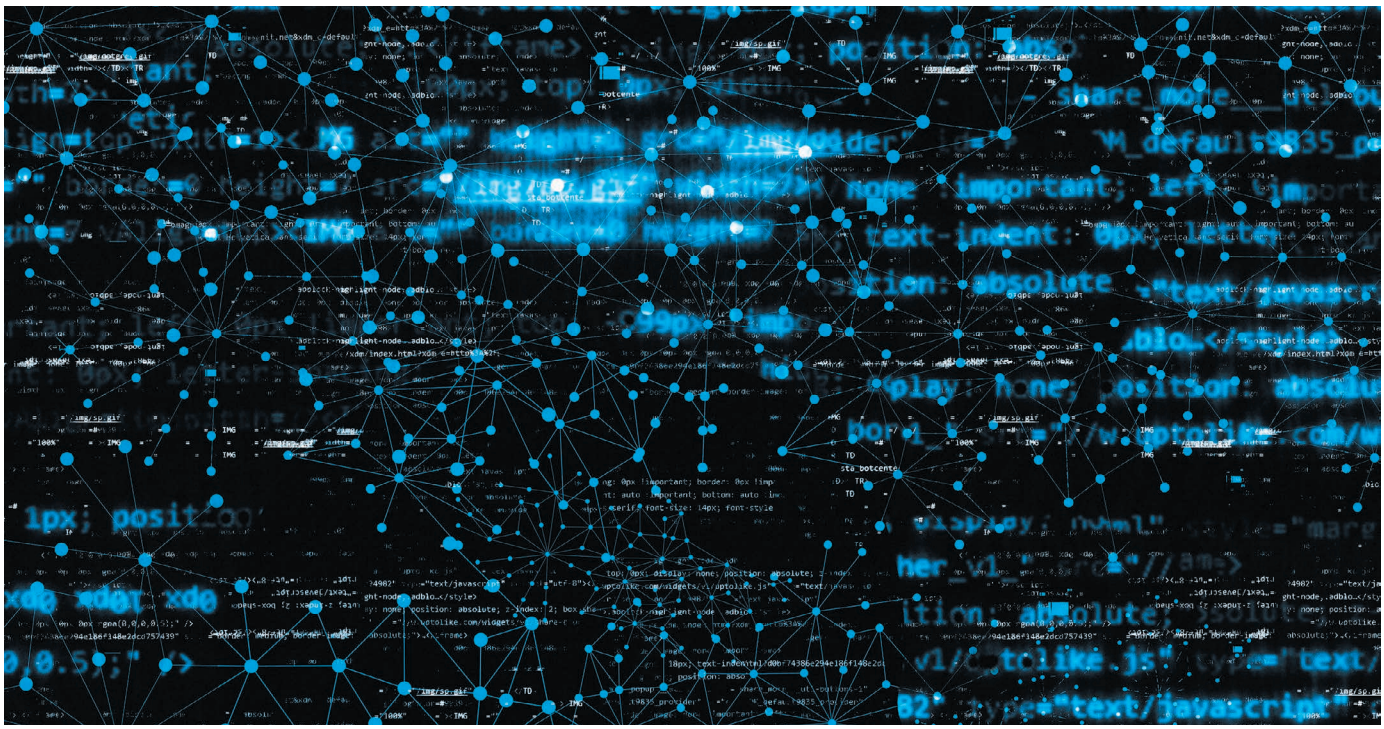
TECHNOLOGY FEATURE

A TOOLKIT FOR DATA TRANSPARENCY

A simple software toolset can help to ease the pain of reproducing computational analyses.



MEHAU KULYK/SPL/GETTY

BY JEFFREY M. PERKEL

Julia Stewart Lowndes studied metre-long Humboldt squid (*Dosidicus gigas*), tagging them to track their dives, as a graduate student at Stanford University in California in 2011. When she wrote up her dissertation, she had data on five animals. Then, two months before the project was due, another tag was located in Hawaii.

"I got a sixth of my data within months of finishing," Lowndes says — a lucky break. But luck favours the prepared. Each data set contained hundreds of thousands of points; Lowndes was able to use the new one easily because she had conducted her analyses with an eye on reproducibility. Computational reproducibility is the ability to repeat an analysis of a particular data set and obtain the same or similar results, says Victoria Stodden, who studies reproducibility at the University of Illinois at Urbana–Champaign. In practice, it means that researchers who publish scientific findings based on computational analyses should release 'digital artefacts', including data and code, so that others can test their findings.

That hasn't always been the case. Figures have long been published with no way of viewing the underlying data, and algorithms have been described in plain English. But there are many ways to implement an algorithm, and many 'knobs and dials' that can be tweaked to produce subtly different results. Without the actual code, parameters and data used, it is impossible for other researchers to be certain whether the published implementation is correct, or whether their own data are consistent with the published results.

Reproducibility doesn't necessarily mean that the results are scientifically accurate, Stodden emphasizes. "It just says we have this level of computational transparency that permits verification, so that we know that those computational steps, combined with all the workflow information, including the data and any input parameters, actually did produce the results in the article."

That transparency comes from a handful of key tools, especially version control, scripting and programming. Similar to many researchers, Lowndes — now a marine data scientist at the US National Center for Ecological Analysis and Synthesis at the University of California, Santa Barbara — was trained to work with point-and-click graphical applications such as Microsoft Excel.

But precisely documenting one's analytical steps in such software is difficult, which complicates reproducibility. When Lowndes's first data sets arrived, she discovered that they were too large to open in Excel, so she had to learn to crunch her numbers in the programming language MATLAB. It wasn't easy. But by the time that sixth tag washed up on a beach in Maui, she could update her entire analysis by running her code again.

Computational reproducibility, Lowndes says, is about "thinking ahead". She adds: "It's not thinking, 'This is easiest for myself right now.' It's thinking, 'When I'm working on this next week, next month, right before I graduate — how do I set myself up so that it's easier later?'"

SHARING KNOWLEDGE

David Donoho, a statistician at Stanford, co-authored a 1995 paper describing a library of signal-processing functions called WaveLab, in which he explained the history of computational reproducibility so far (J. B. Buckheit and D. L. Donoho *Wavelets Statist.* **103**, 55–81; 1995). Paraphrasing Stanford geologist Jon Claerbout, a leader in the reproducibility field, Donoho wrote: "An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures."

Put simply, this means that researchers should make their computational workflow and data available for others to view. They should include the code used to generate published figures and omit only data that cannot be released for privacy or legal reasons. Lorena Barba, an aerospace engineer at George Washington University in Washington DC, for instance, creates 'reproducibility packs' — including the code, data and final image — for every figure she publishes. She archives those packages on digital-repository sites such as Figshare, and includes the digital object identifiers (DOIs) in the papers themselves.

But reproducibility is incredibly complex, because neither software nor data are static.

"When you write a program to analyse a piece of biological data, for example, you're sitting on the pinnacle of a huge pile of software, the top bit of which you wrote, and underneath it is this enormous pile of stuff which other people have written," says Les Hatton, a mathematician at Kingston University in London, who has been studying computational reproducibility since the 1970s (and who also releases reproducibility packages for his publications).

In this era of open-source, community-developed software, the data and software are constantly in flux. Seemingly simple code optimizations can alter output values. But even the same code can fail to produce the same outcome if it is run on different hardware or compiled using different settings, or because it makes assumptions about the user's file-directory structure.

"The sum total of all of this is that unquantifiable errors are extremely simple to introduce into the results," Hatton says.

SOFTWARE STRUGGLES

Take Chris Baldassano, for instance. Baldassano was a postdoctoral fellow doing memory research at Princeton University in New Jersey when he developed MATLAB code to analyse brain-activity data. Now at Columbia University in New York City, he shared that code with his collaborator, who was using a different version of MATLAB and so got errors instead of results. "There was something about how the underlying system dealt with large numbers that was slightly different on my software and theirs," he says. Baldassano circumvented the problem by encapsulating his code on Code Ocean, a cloud-based service that allowed him to share his computational environment and code in a web browser, where people could also execute the scripts. (On 1 August, *Nature Methods*, *Nature Biotechnology* and *Nature Machine Intelligence* announced a pilot programme that allows authors "to share fully-functional and executable code accompanying their articles and to facilitate peer review of code by the reviewers" using Code Ocean. See go.nature.com/2akz52b for details.)

> ***"Users often find themselves struggling to replicate other scientists' research — and even their own."***

Data sets can also evolve over time. Morgan Ernest, an ecologist at the University of Florida in Gainesville, leads a research project that has been documenting biodiversity in the Arizona desert for more than 40 years. "It's 'living data' because it's still changing," she says. In such cases, pinpointing and documenting the precise state of the data at the time the analyses are run becomes crucial, because the data set might well have evolved by the time researchers want to revisit it. Ernest published a strategy for handling such data using GitHub and continuous integration — a form of automated code checking — in June (see go.nature.com/2o2lsza).

The upshot of all these layers is that users often find themselves struggling to replicate other scientists' research — and even their own. As part of his postdoc, Casey Greene, a bioinformatician at the University of Pennsylvania in Philadelphia, ran a computational analysis of gene-expression data. He documented every aspect of his work, except the specific version of a key database. Four years later, those findings were irreproducible, at least in the fine details. "That was troubling," he says.

Today, journals increasingly consider reproducibility when evaluating and publishing manuscripts. The *Journal of the American Statistical Association* has appointed editors specifically to assess computational reproducibility. At the *American Journal of Political Science*, a recent emphasis on reproducibility

Getting reproducible

There's no one-size-fits-all solution for computational reproducibility. But these practices can help.

- **Use code.** Instead of pointing and clicking, use programming languages to download, filter, process and output your data, and command-line scripts to document how those tools are executed.
- **Go open-source.** Code transparency is key to reproducibility, so use open-source tools whenever possible. "If you give me a black box with no source code and it just gives me numbers, as far as I am concerned, it's a random-number generator," says mathematician Les Hatton of Kingston University in London.
- **Track your versions.** Using version-control software such as Git and GitHub, researchers can document and share the precise version of the tools that they use, and retrieve specific versions as necessary.
- **Document your analyses.** Use computational notebooks such as Jupyter to interleave code, results and explanatory text in a single file.
- **Archive your data.** Freeze data sets at key points — when submitting an article for publication, for example — with archiving services such as Zenodo, Figshare or the Open Science Framework.
- **Replicate your environment.** Software 'containers', such as Docker, bundle code, data and a computing environment into a single package; by unboxing it, users can recreate the developer's system. ReproZip, developed in the lab of New York University computer scientist Juliana Freire, simplifies container creation by watching program execution to identify its requirements. The commercial service Code Ocean and an open-source alternative, Binder, enable researchers to create and share executable Docker containers that users can explore in a web browser.
- **Automate.** Automation provides reproducibility without users really having to think about it, says bioinformatician Casey Greene at the University of Pennsylvania in Philadelphia. Continuous integration services such as Travis CI can automate quality-control checks, for instance, and the Galaxy biocomputing environment automatically logs details of the jobs it runs.
- **Get help.** Resources abound for interested researchers; see practicereproducibleresearch.org, for instance, or find a Software Carpentry workshop near you to learn basic computing skills.

extended average publication times by more than seven weeks, requiring (on average) two resubmissions of author materials and eight hours of employee time per manuscript "to replicate the analyses and curate the materials for public release", according to a 2017 analysis published in *Inside Higher Ed* (see go.nature.com/2vhvv3x). Thu-Mai Christian, a data archivist at the H. W. Odum Institute for Research in Social Science at the University of North Carolina at Chapel Hill, which performs the journal's reproducibility assessments, says the biggest issue with submitted material is poor documentation.

All articles published in Nature Research journals, and in *Nature* itself, require a statement indicating where the data can be found, and those with custom code require a code-availability statement. Code is reviewed on a case-by-case basis. At the moment, reviewers must install the software themselves to test it, and hiccups are not infrequent, says *Nature Methods* chief editor Natalie de Souza. "It's not 1–5%" of cases, she says. "It is more common."

TOOLKIT TEST

Reproducibility advocates are converging around a tool set to minimize these problems. The list includes version control, scripting, computational notebooks and containerization — tools that allow researchers to document their data, the steps they follow to manipulate it, and the computing environment in which they work (see 'Getting reproducible').

Without these pieces, says Ben Marwick, an archaeologist at the University of Wollongong in Australia, tantalizing connections lie abandoned because they're impossible to explore. "It's like we're a roomful of hungry people handing around tins of canned food, and nobody has a can opener. And then we're asking each other, why doesn't anyone eat anything?"

Reproducibility can be a tough sell. Researchers aren't software engineers, and learning to work that way is an intimidating prospect, says Lowndes. Reproducibility work can also be time-consuming, and there's little incentive to pursue it, adds Rich FitzJohn, a research software engineer at Imperial College London. "If researchers are spending all of their time stressing about how long something is going to stay working for them, that's just time they're not spending doing the thing that they're good at, which is solving science problems."

Academia in many ways disincentivizes reproducibility, as most institutions reward high-profile publications, and making code and data freely available could expose researchers to criticism and the possibility of getting scooped, Stodden says. Speaking of how much academics can realistically achieve, she adds: "Whatever people feel like they can fit in their regular day-to-day job of being a researcher, I think that's enough for right now". Even if researchers cannot build executable containers, for instance, making code available on GitHub or Zenodo repositories allows others to see what they did. (Stodden, Barba and Donoho are members of a US National Academies panel on reproducibility and replicability in science. The panel's final report is anticipated by early 2019.)

Such approaches might be most useful for researchers themselves. Computational ecologist Christie Bahlai ran a course at Michigan State University in East Lansing that taught graduate students in ecology to work with data. One year, her students submitted a paper describing a model of firefly activity, including their code. A reviewer actually tested the code and pointed out that the team was basing its conclusions on an assumption; if they changed that assumption, they might draw a different conclusion.

"He was able to make this insightful comment because of the higher level of reproducibility of the paper," says Bahlai, who is now at Kent State University in Ohio. The team expanded the article's discussion to elaborate.

"This is how science is supposed to work, isn't it?" she says. "Because of reproducibility, people were able to critique the work at a deeper level." ■

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

Q&A Harriet Alexander
Software training in Antarctica

Harriet Alexander is a postdoctoral fellow in oceanography bioinformatics at the University of California, Davis. In January, she travelled to McMurdo Station in Antarctica to take part in this year's Antarctic Biology Training Program, a month-long course sponsored by the US National Science Foundation. When bad weather delayed her flight home, she was able to host a workshop for the non-profit training organization Software Carpentry, its first in Antarctica.

What is Software Carpentry?

Software Carpentry is one branch of The Carpentries, a project based in San Francisco, California, that trains researchers in data science and software. For US$2,500, plus travel and accommodation, volunteer instructors present a 2-day workshop at your location, on topics such as the command line, programming, and how to work with data. Lessons can be tailored to specific fields, and are constantly being improved in line with best practices.

I have led two workshops, and I really love the format. That sort of intensive, full-day immersion in the technology, with lots of hands-on material, is a really great model.

How did the workshop go?

We had about 20 students. The first day, I taught the Unix command-line introduction and it went really well. The second day was Python, which we typically teach by introducing some packages using the Conda packaging system. Anaconda is about 300 megabytes. That doesn't sound like much — but when you're in Antarctica, where the Internet is very slow, it's huge. I ended up downloading Miniconda, a stripped-down version of 35 megabytes, and it took me probably 2.5 days. But I forgot that Miniconda has to download other stuff, so there was no way it was going to work.

Bottom line: if you're going to be teaching somewhere remote, download what you need before you get there.

How was Antarctica?

It's a completely different world — very stark, yet absolutely gorgeous. McMurdo is like a mining town plopped down in the middle of this barren ice. So it's like moving to a small town for a month and living in a dorm. It was a wonderful experience. ■

INTERVIEW BY JEFFREY M. PERKEL

This interview has been edited for length and clarity.