Digital science

Technology is transforming science. We can calculate faster than ever before, and gather more precise data, of an ever-expanding variety. We now store staggering volumes of information, and draw on it ever more creatively. Communication between scientists is changing too, even if the scientific paper still remains the standard element of information transfer and recording. The web, e-mail and blogs, as well as social media including Facebook and Twitter, have completely altered how we exchange thoughts.

The standard scientific paper increasingly offers only a limited glimpse of today's scientific process. After all, computer software of innumerable kinds has established itself implicitly within the fabric of scientific practice — in the structuring of databases, and in computer algorithms used to process such data, or to carry out mathematical analyses. How the intricate details of software and algorithms influence results isn't always clear.

Indeed, one might worry that the standards of science are quietly being eroded. Can we trust the results of a new paper if they depend on calculations carried out by proprietary software with non-public source code? Just recently, mathematicians reported the proof of a long-standing conjecture — the Boolean Pythagorean triples problem. The proof resides in some 200 terabytes of computer-generated output. If only computers running other algorithms can check the result, how can we really know it's all ok?

These issues have been raised recently by biophysicist Konrad Hinsen (preprint at https://arxiv.org/abs/1605.02960; 2016), who has drawn motivation from years of relying on software to run biomolecular simulations. As he points out, computing applications have become so utterly familiar to all of us that we treat them as practical tools not requiring further analysis. We see tools for integrating, simulating, sorting and transforming as the equivalent of forks, spoons and other tableware; as unquestioned things to be used and judged only by their capacity to produce results.

Yet beyond results, science aims to record the process of logic and argument that leads to a deduction or interpretation. Scientists judging work need to scrutinize the path leading to results. An experiment, for example, only leads to new understanding if we understand all the details of how the

Computer software of innumerable kinds has established itself implicitly within the fabric of scientific practice.

apparatus was set up and used; we accept theoretical arguments only if each step is made explicit. Likewise, Hinsen argues, trusting the results of any calculation run on scientific software requires full understanding of the functioning of that software. Unfortunately, most software used today never gets any peer review.

Hinsen offers an illustration of the problem. Consider using Newton's equations to calculate the celestial mechanics of planets and predict their future trajectories. The scientific knowledge relevant to the problem is embodied in Newton's laws of motion and gravitation, the mathematics of calculus, as well as a considerable body of astronomical observation and experimental knowledge (on telescopes and the interpretation of their output, for example) that yields accurate initial conditions for the bodies in question at some moment. The calculation of future motion can then be carried forward, in principle using only pencil and paper.

In practice, we use computers, as otherwise we could calculate little of practical interest. This introduces another layer of required specification, for the numerical algorithm used to approximate the continuous dynamical equations with finite difference analogues, as well as for the software that determines how real numbers get approximated by floating point operations. Calculations relying on software need to be explicit on such details if they are to fully describe the pathway to a set of results, so other scientists can reproduce them. Today, this is often — probably even typically — not the case. Our casual familiarity with software is partially to blame.

But the nature of software, Hinsen argues, makes the problem worse. The programming languages behind it all aren't anything like human languages, and only permit writing instructions for a computer, and nothing else. This disrupts the ordinary process by which scientists record what they're doing in a way that can be easily understood by others, as anything scientific has to be reduced to numerics. The result is a series of diabolical tendencies all too familiar to computational scientists. As he notes, scientists can develop software using the best practices of software engineering and the result may still compute something different from what its users think it computes. Of course, concerned users can in principle consult the documentation, but how do they know that documentation is complete and accurate?

All of this would be avoided if computing source code were intelligible to human readers, but that's very far from the case. At the moment, Hinsen suggests, "most scientific software source code is unintelligible to its users, and sometimes it even becomes unintelligible to its developers over time."

Hinsen got concerned about the issue after having to abandon a research project when he could not reproduce the most important prior work on the topic, which relied on software that no longer existed. He wrote his own software, obtained very different results, yet couldn't explore where the differences might come from. He couldn't publish on the topic, given the lack of real information to make any comparisons.

What's the solution? Hinsen doesn't offer any specific solution, but merely raises the question and outlines possible routes to better practice. At the moment, many people don't even realize that there is a problem. It can be solved, he suggests, if scientists relying on computation — and this is increasingly just about everyone — take the issue seriously. Yet it will require insight from computer science itself to find the right techniques. Hinsen has started an open science project in the hope of finding collaborators to start working toward better practices (see https://www.guaana.com/projects/scientific-notations-for-the-digital-era).

Scientists haven't learned yet that software represents more than just tools for doing things; it's part of the precious repository of knowledge. It's currently being built up out of language that the majority of human scientists can't read or understand, and this is a big problem, even if it is largely invisible. There's no guarantee that the cumulative and self-correcting history of science will automatically continue. ❐

MARK BUCHANAN

© 2016 Macmillan Publishers Limited. All rights reserved.