nature biotechnology

In need of an upgrade

Improving the integration of computational analysis into biology will require better documentation, validation and accessibility of software associated with papers.

Biology today increasingly relies on techniques that generate vast amounts of data. And yet it seems that whereas high-throughput approaches have made great strides, all too often researchers are running to keep up when it comes to parsing and interpreting the resulting data. A small but growing number of laboratories around the world have the requisite expertise in both biology and computation. Other groups may be lucky enough to have a biologically savvy 'computational expert' on hand to delegate such tasks to (all too rare outside the major research centers). But for the majority of biologists, computational knowledge is self-taught, and frequently long hours are spent completing tasks at the computer that should really take minutes. Many have emphasized the need for greater educational resources for those interested in understanding software. But an equally important aspect that until now has received little attention is the need for better documentation, validation and prominence of software in papers as a means of galvanizing progress in computational biology.

In this issue, *Nature Biotechnology* asks several creators of widely used computational biology software to describe the factors that contributed to their success (p. 894). Their secret sauce appears to boil down to five ingredients: developers must possess sufficient proximity to and understanding of the research problem at hand; timing of the software release should correspond with the emergence of the problem in the research community that it addresses; software should have extensibility and interoperability; the algorithm implemented by the software should ideally be novel and indicative of profound insight; and, finally, a broad range of users should be able to run and operate the program.

The interviewees were almost unanimous in their view that most biologists fail to appreciate computation and the craft of making software. The underappreciation of computational science is manifest in several ways. First, developing a mathematical algorithm to answer a research question is seen as more intellectually valuable than developing a software implementation for a broad community of users (in fact, both sets of skills are needed, but rarely found in the same person). Second, current research funding structures fail to provide funding for software and database projects, let alone maintain them. Third, computational scientists tend to face an uphill struggle in obtaining faculty positions in biology departments because bioinformatics is seen more as an adjunct to the other main disciplines, rather than a discipline all in itself. Because computational candidates tend to work as part of an interdisciplinary group, they are not credited with first or senior authorship on a multi-authored paper. Computational science is the means to the biological ends that gets the credit—not the other way round.

Conversely, it could be said that computational scientists often appear oblivious to the need to make their tools accessible and comprehensible to a wider biology audience. Why are so many computational tools inaccessible to all but the most expert practitioners? And why is so little effort spent either on verifying and/or validating software that accompanies algorithms in papers or on creating central repositories where versions of it can be stored in perpetuity?

A case in point is the Assemblathon 2 paper published earlier this year, which evaluates the performance of a whole raft of *de novo* software methods for genome assembly in three different organisms (*GigaScience* **2,** 10, (2013)). The paper comes to the conclusion that each of the tens of assembly software packages produces very different results for the same genome with just small parameter tweaks; what's more, the same assembler with the same parameters performs differently on different sequence data sets. It is sobering that in the second decade of the twenty-first century, there is no means of automatically evaluating the performance of all assemblers on a particular sequence data set that can provide a nonexpert with the best-performing output.

On the other hand, the Assemblathon illustrates the power of crowdsourced challenges—pioneered by competitions like CASP and DREAM—in assessing software performance. In these challenges, teams from the community compete with one another to verify software methodologies against carefully chosen benchmarks. These initiatives are important as they provide side-by-side comparisons of software and highlight the pros and cons associated with certain programs.

But so much more could be done to verify, validate and highlight software in research papers. At *Nature Biotechnology*, for example, we request code for software to be placed in Supplementary Data if that software is central to a paper's main claims. Even so, little systematic feedback is obtained from referees on whether a code implementation matches the mathematical algorithm in the paper, whether the software can be used on common operating systems (e.g., Windows or UNIX) or whether code is readable and sufficiently documented to allow another researcher to follow the algorithm. And when a paper is accepted for publication, no minimum set of specifications about software (e.g., version information or parameter settings) need be present in the final publication. All this, of course, means more work for referees and editors, and we would thus welcome reader feedback on whether such changes to the handling of software are needed.

On balance, it is our view that both biological research and software development would benefit from these changes. For the biological user, ensuring that the software associated with a report has met rigorous standards will mean more people can access it, more people can understand it and more people can adopt it. And for the software developer, a greater emphasis on software quality in publications will translate to career benefits and appropriate credit. Best of all, more stringent peer review of software associated with papers will bring biologists closer to computational scientists by better recognizing what software brings to the research endeavor.