Computation and biology: a partnership

Computation goes hand in hand with contemporary biological studies. We describe a few trends in computational science that are helping drive new biological knowledge.

From instrument control, to data analysis and visualization, to simulation and prediction studies, to computational notebooks used for record keeping, computation is an essential part of the majority of contemporary biological studies. Since our very first issue in October 2004, *Nature Methods* has been publishing computational methods and tools, as well as software performance comparisons, that we think will be of broad interest to life scientists.

New experimental technologies create opportunities for computational scientists to develop tools to exploit the data generated by such techniques. For example, the diversity of nucleic acid sequencing technologies now available has necessitated new computational tools and/or adaptations of existing tools for analyzing the resulting data, as reviewed for single-cell RNA-seq analysis in this issue. The importance of software tools to analyze data from techniques as diverse as mass spectrometry-based metabolomics and magnetic resonance imaging cannot be underscored enough. Computational advances can fundamentally change and improve how biologists interact with their data, and even propel new biological insights. New algorithms being applied to cryo-electron microscopy datasets, for instance, are allowing researchers to reconstruct heterogeneous structural ensembles of protein complexes. Computational methods that help researchers integrate disparate types of datasets can yield new biological inferences, making such datasets even more valuable than the sum of their parts.

Machine learning and its buzzworthy relative deep learning are here to stay, already having had a profound impact across multiple fields, especially in image analysis (as exemplified by our recent Focus issue on deep learning in microscopy), in neuroscience, and in genomics. Models with more sophisticated architecture and improved expressivity and interpretability are being developed at a rapid clip. Their applications are being explored to tackle some of the most daunting challenges in modern data sciences, such as high dimensionality, noise and sparsity.

Biology can often be computation-intensive, especially when crunching huge datasets or running detailed simulations. Supercomputers are all too rare (and expensive), however, so computational scientists have taken advantage of workarounds. Distributed computing has facilitated the intensive process of protein structure prediction, as exemplified by Rosetta@home. Many algorithms have been implemented to run on graphics processing units (GPUs) to take advantage of parallel computing, allowing performance gains of several orders of magnitude. This has helped accelerate computationally demanding all-atom molecular dynamics simulations, for example, making millisecond and longer time scale simulations a reality. The rise of cloud computing, exemplified by popular platforms such as Galaxy, allows a researcher to choose from a plethora of tools using infrastructure maintained by a service provider.

Another newsworthy computational trend, quantum computing, is also poised to make an impact in biology, as discussed in a Comment in this issue. Quantum computing may yet help with difficult search problems that are so computationally intensive that they are essentially impossible with classical computers. One area with potential to greatly benefit from this technology is molecular design; another is the analysis of population-scale datasets. However, applying quantum computing in life science is not simply a matter of porting an existing algorithm to a quantum computer—it is a fundamentally different computational paradigm. And not all biological problems will benefit from quantum computing. These caveats and more are discussed in this month's Technology Feature.

Over the years we have continually improved how we handle computational papers. Since the early days of the journal, we have asked reviewers to evaluate tool performance and code, and required that code central to new methods we publish be made available upon publication. We have also aimed to educate our authors and readers about the importance of naming software and ensuring that it is properly cited. With popular code repositories such as GitHub and DOI-minting repositories such as Zenodo now in common use, software tools may be made readily accessible and discoverable. We have also partnered with Code Ocean to facilitate the peer review of code, without reviewers (and, eventually, readers) needing to download a frustrating number of dependencies to run a program.

It's clear that computational tools for biology are no longer solely the domain of experts: another growing trend has been that of packaging tools in containerized platforms with easy-to-use graphical user interfaces. This has enabled life scientists without serious computational know-how to apply sophisticated software tools in their research. But this 'black boxing' of computational tools comes with a risk: life scientists must ensure they are sufficiently knowledgeable to understand how the tools they apply function, lest they apply the tools improperly or without a full understanding of their caveats.

On the flip side, software developers must consider what biologists need to know about how a tool functions without overwhelming them with details. We believe that computational methods papers intended to be read and used by biologists should in fact be readable by biologists. This is why such papers we publish tend to have a relatively brief description of the underlying algorithm in the main part of the text, supported by figures that demonstrate strong validation and an application to a challenging biological problem. Computationally savvy readers who are interested in looking under the hood at the algorithmic details will still find them accessible in the Methods section and in the Supplementary Information.

Over the years, we have been pleased to see a culture shift towards greater openness, with many researchers now habitually making software tools for biological research freely available and providing source code and detailed documentation. Going beyond improving the reproducibility and transparency of results generated using computational tools, such practices are more likely to facilitate greater community uptake. Making code open source and providing appropriate licenses allows other developers to adapt and further build on existing code, advancing science. As always, we welcome your feedback about how we can improve our editorial standards and processes to better serve both computational tool developers and tool users. ❐

Published online: 8 July 2021
https://doi.org/10.1038/s41592-021-01215-2