COMPUTER SCIENCE

Accessible Reproducible Research

As use of computation in research grows, new tools are needed to expand recording, reporting, and reproduction of methods and data.

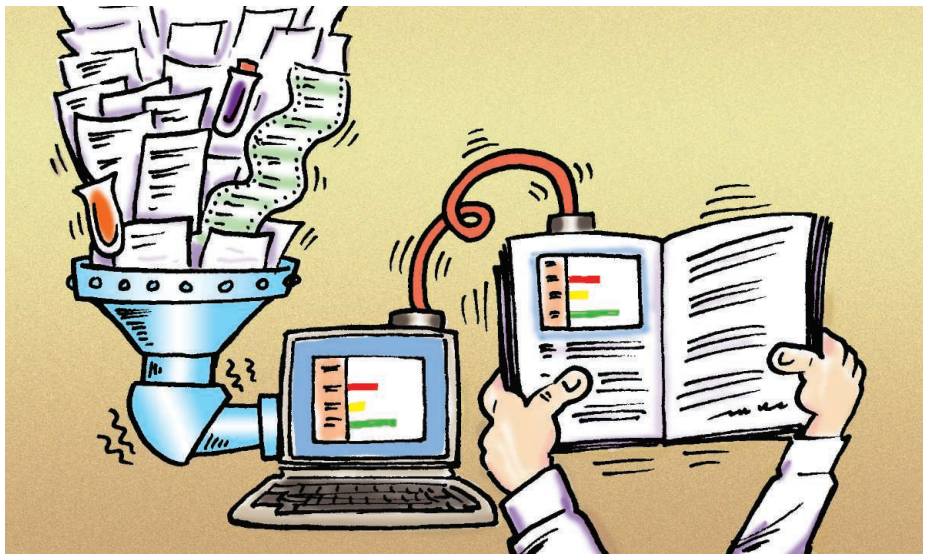
Jill P. Mesirov

Scientific publications have at least two goals: (i) to announce a result and (ii) to convince readers that the result is correct. Mathematics papers are expected to contain a proof complete enough to allow knowledgeable readers to fill in any details. Papers in experimental science should describe the results and provide a clear enough protocol to allow successful repetition and extension.

Over the past ~35 years, computational science has posed challenges to this traditional paradigm—from the publication of the four-color theorem in mathematics (*1*), in which the proof was partially performed by a computer program, to results depending on computer simulation in chemistry, materials science, astrophysics, geophysics, and climate modeling. In these settings, the scientists are often sophisticated, skilled, and innovative programmers who develop large, robust software packages.

More recently, scientists who are not themselves computational experts are conducting data analysis with a wide range of modular software tools and packages. Users may often combine these tools in unusual or novel ways. In biology, scientists are now routinely able to acquire and explore data sets far beyond the scope of manual analysis, including billions of DNA bases, millions of genotypes, and hundreds of thousands of RNA measurements. Similar issues may arise in other fields, such as astronomy, seismology, and meteorology. While propelling enormous progress, this increasing and sometimes "indirect" use of computation poses new challenges for scientific publication and replication. Large data sets are often analyzed many times, with modifications to the methods and parameters, and sometimes even updates of the data, until the final results are produced. The resulting publication often gives only scant attention to the computational details. Some have suggested these papers are "merely the advertisement of scholarship whereas the computer programs, input data, parameter values, etc. embody the scholarship itself" (*2*). However, the actual code or software "mashup" that gave rise to the final analysis may be lost or unrecoverable.

For example, colleagues and I published a computational method for distinguishing between two types of acute leukemia, based on large-scale gene expression profiles obtained from DNA microarrays (*3*). This paper generated hundreds of requests from scientists interested in replicating and extending the results. The method involved a complex pipeline of steps, including (i) preprocessing of the data, to eliminate likely artifacts; (ii) selection of genes to be used in the model; (iii) building the actual model and setting the appropriate parameters for it from the training data; (iv) preprocessing independent test data; and finally (v) applying the model to test its efficacy. The result was robust and replicable, and the original data were available online, but there was no standardized form in which to make available the various software components and the precise details of their use.



CREDIT: JOE SUTLIFF/WWW.CDAD.COM/JOE

Broad Institute of Massachusetts Institute of Technology and Harvard University, Cambridge, MA 02142, USA. E-mail: mesirov@broad.mit.edu

Reproducible Research

This experience motivated the creation of a way to encapsulate all aspects of our in silico analyses (*3*) in a manner that would facilitate independent replication by another scientist (*4*). Computer and computational scientists refer to this goal as "reproducible research" (*5*), a coinage attributed to the geophysicist Jon Claerbout in 1990, who imposed the standard of makefiles for construction of all the figures and computational results in papers published by the Stanford Exploration Project (*6*). Since that time, other approaches have been proposed (*7*–*14*), including the ability to insert active scripts within a text document (*15*) and the use of a markup language that can produce all of the text, figures, code, algorithms, and settings used for the computational research (*16*). Although these approaches may accomplish the goal, they are not practical for many nonprogramming experimental scientists using other groups' or commercial software tools today.

A similar challenge was encountered more than 20 years ago when scientists wanting to access data from remote computers had to write their own retrieval programs. The solution was the invention of the World Wide Web (*17*), together with the concept of "Web browsers" such as MOSAIC (*18*) and its successors. The approach was so effective that we now take it for granted.

In the same spirit, we need a paradigm that makes it simple, even for scientists who do not themselves program, to perform and publish reproducible computational research. Toward this end, we propose a Reproducible Research System (RRS), consisting of two components. The first element is a Reproducible Research Environment (RRE) for doing the computational work. An RRE provides computational tools together with the ability to automatically track the provenance of data, analyses, and results and to package them (or pointers to persistent versions of them) for redistribution. The second element is a Reproducible Research Publisher (RRP), which is a document-preparation system, such as standard word-processing software, that provides an easy link to the RRE. The RRS thus makes it easy to perform analyses and then to embed them directly into a

Published by AAAS

paper. A reader can readily reproduce the analysis and, in fact, can extend it within the document itself by changing parameters, data, filters, and so on.

A simple form of this concept is embedded in most word processors: When one "clicks" on a spreadsheet embedded in a document, an active spreadsheet will "pop up," which allows the reader to fill in new numbers, propagate formulas, and create new charts, all without leaving the document. This can be thought of as a rudimentary RRS, involving linkage between an RRE (the spreadsheet program) and an RRP (the word processor).

GenePattern-Word RRS

In collaboration with researchers at Microsoft, my coauthors and I (*4*) conceived and created a user-friendly version of an RRS. The RRE is the GenePattern computational genomics environment (*19*), and the RRP is an adaptation of Microsoft Word that can link to GenePattern (*20*). Other "quantitative programming environments" (*8*) might serve as the RRE, and other document preparation environments could, with appropriate modification, serve as the RRP (*21*).

GenePattern is an environment that allows analysis of genomic data sets by (i) creating pipelines by "connecting" modules, from a large library of over 120 tools; (ii) defining parameters for analysis; and (iii) specifying the data sets to be used. These "analytic workflows" can be created through a user-friendly, graphical user interface without writing any computer code; they can then be executed on a GenePattern server running on the researcher's desktop, a larger departmental machine, or a high-performance compute farm. GenePattern automatically tracks the versions of modules and pipelines, captures the history of users' analytic sessions, and can generate the corresponding pipelines (including parameters and input data files) from user output files, as well as package them for redistribution. In this way, a nonprogramming scientist can create fully reproducible in silico research.

The combined GenePattern-Word RRS embeds the functionality of GenePattern pipelines within a Microsoft Word document (figs. S1 to S4). By using a menu in the word processor, an author can link text, tables, and figures to previously executed GenePattern pipelines comprising the entire analysis and data that yielded those results (*22*). Pipelines and data can then be stored in their entirety within the document or (for space or runtime considerations) as a pointer to their location on the Web.

Similarly, a reader of the document can open a dashboard within the word processor to view the GenePattern pipelines. When one selects a table or figure, the word processor displays the pipeline that produced it. Just as with opening a spreadsheet, the reader can directly connect to a GenePattern server to rerun the calculation, change parameters, or apply the method to other data. The reader can save the exploratory results within the document, along with their provenance (for replication) and annotated text. The document can then be sent to a colleague.

Conclusion

The GenePattern-Word RRS system described here is intended as an example. Scientists who employ stochastic simulations would benefit from RRSs designed to capture simulations, including the initialization parameters. Commercial vendors of software packages used by the research community could (and should) develop RRS versions of their codes. Critical to a robust RRS is the automated tracking and maintaining of code versions so that, as methods evolve, the computations can still be repeated. High-performance codes may involve special requirements for processing and storage. Equally important may be the hardware configuration, operating system version, compiler version, and so on for full provenance of a complex piece of software. Although we have focused here on new software systems as a foundation for reproducible research, it is important to note that data integrity and persistence are also critical concerns (*23*).

The centrality of the role of computation in science—from molecular biology to the social sciences (*24*)—calls out for a new model for the way we publish our results. Just as it is routine to include references in our papers, we should also include our complete computational methods. Journals can play a key role in making this a requirement for publication. To facilitate this, we need simple, intuitive ways to both capture and embed our computational work directly into our papers. The value of such tools goes beyond mere documentation. They will encourage the next generation of scientists to become "active" consumers of scientific publications—not just looking at the figures and tables, but running computational experiments to probe the results as they read the paper.

References and Notes

1. K. Appel, W. Haken, *Discrete Math.* **16**, 179 (1976).
2. M. Schwab, M. Karrenbach, J. Claerbout, *Comput. Sci. Eng.* **2**, 61 (2000).
3. T. R. Golub *et al.*, *Science* **286**, 531 (1999).
4. M. Reich *et al.*, *Nat. Genet.* **38**, 500 (2006).
5. Reproducible research in this context refers to the ability to repeat the calculations for analyzing the data and obtaining the computational results rather than independent validation by another algorithm or implementation.
6. The Stanford Exploration Project (http://sepwww.stanford.edu) is a 25-year-old project in seismic imaging. To achieve his reproducibility goal, Claerbout leveraged the work of Feldman (*25*), who developed the Make program in the 1980s for maintaining and building executable programs from source code.
7. Special Issue on Reproducible Results, *Comput. Sci. Eng.* **11**, 3 (2009).
8. J. Buckheit, D. Donoho, in *Wavelets and Statistics*, A. Antoniadis, Ed. (Springer-Verlag, Berlin, 1995), pp. 55–81.
9. R. Gentleman, *Stat. Appl. Genet. Mol. Biol.* **4**, 25 (2005).
10. R. Gentleman, D. Temple Lang, *J. Comput. Graph. Statist.* **16**, 1 (2007).
11. Sweave processing of Open Document Format (ODF) files, http://cran.r-project.org/web/packages/odfWeave/index.html.
12. *Notebook Basics*, Wolfram Research, http://reference.wolfram.com/mathematica/guide/NotebookBasics.html.
13. *Using Notebook to Publish to Microsoft Word*, www.mathworks.com/access/helpdesk/help/techdoc/matlab_env/brgbdb8.html.
14. *Scripting Platform Plug-Ins*, http://inference.us/Solution-Platform/Scripting%20Platform%20Plug-Ins.aspx.
15. A script is a programming language interpreted or carried out by another program. Examples of the approach of embedding active scripts in text include (*12*–*14*).
16. This approach leverages the work of Knuth and his typesetting system TeX (*26*), as well his notion of literate programming, introduced in the early 1980s, where a single file produces both source code ("tangle" command) and documentation ("weave" command) (*27*).
17. T. Berners-Lee *et al.*, *Commun. ACM* **37**, 76 (1994).
18. M. Andreessen, "NCSA Mosaic Technical Summary" (National Center for Supercomputing Applications, 1993).
19. GenePattern, www.broadinstitute.org/genepattern.
20. The GenePattern-Word RRS is a freely available open source add-in to the Microsoft Office application; http://genepatternwordaddin.codeplex.com. A technical description and a video of a user session are in the Supporting Online Material.
21. Key features of an RRE are automated provenance tracking and easy packaging of the computational analysis for redistribution. Packages like geWorkbench, MeV, Galaxy, and Accelrys Software's Pipeline Pilot have varying amounts of this functionality and might be modified to add the rest. They do not require the user to script or program. Examples of alternative document preparation software include, Open Office's Writer, Corel's WordPerfect, and Apple's Pages. It is highly unlikely that any one piece of software would support all of science. But providing scientists who use computation with the means to capture the history of their analyses and to embed them into their documents in an easy and accessible manner should encourage a more thorough and interactive manner of publication.
22. An easy, straightforward method to provide reproducibility of the analysis associated with a scientific result increases the probability that authors will adopt it. It takes only a minute or two to link each executed pipeline to the document.
23. Committee on Science, Engineering, and Public Policy. *Ensuring the Integrity, Accessibility, and Stewardship of Research Data in the Digital Age* (National Academies Press, Washington, DC, 2009).
24. G. King, *PS: Polit. Sci. Polit.* **39**, 119 (2006).
25. S. I. Feldman, *Bell Lab.* **9**, 255 (1979).
26. D. E. Knuth, *TEX and METAFONT: New Directions in Typesetting* (American Mathematical Society, Providence, RI, 1979).
27. D. E. Knuth, *Comput. J.* **27**, 97 (1984).

28. Thanks to C. Mundie, R. Hinrich, and T. Hey at Microsoft for funding the development of the Word add-in; to Infusion Development and Persistent Systems for software implementation; to GenePattern team members P. Tamayo, M. Reich, T. Liefeld, H. Thorvaldsdottir, B. Hill, and H. Keuhn; to B. Gross and E. Lander; and to Broad Institute testers. Complete acknowledgments are in the SOM.

Supporting Online Material

www.sciencemag.org/cgi/content/full/327/5964/415/DC1

10.1126/science.1179653