By Kristin Sainani, PhD

Tool Dissemination

DOING IT RIGHT

Biomedical computing at academic research centers has been compared to a cottage industry. Lots of individuals work away on their focused research projects, generating useful algorithms. But quite often, the knowledge gained is lost when researchers move on to new projects. Yes, they might post their code on Web sites. But is it useful to anyone else without support and documentation? And how can people find it in the first place?

To overcome the cottage industry mentality, the National Institutes of Health (NIH) is placing a greater emphasis on dissemination as a piece of the National Centers for Biomedical Computing (NCBCs) as well as for other grantees.

But what does it really take to turn an impressive algorithm into a widely disseminated, prolific computational tool? The transition might be harder than you think.

"Today, our software is very widely used, but it didn't take off right away. It took years," says **Klaus Schulten**, **PhD**, speaking about the molecular dynamics simulator NAMD (http://www.ks.uiuc.edu/Research/namd/) and the molecular graphics viewer VMD (http://www.ks.uiuc.edu/Research/vmd/), which together have more than 100,000 users. "We went through a long initial phase where we were close to failure all the time." Schulten is professor of physics at the University of Illinois at Urbana-Champaign and director of the Theoretical and Computational Biophysics Group at the university's Beckman Institute.

> "There's a world of difference between developing code for yourself and developing code that you want to distribute," says Klaus Schulten.

For a tool to spread, it takes more than a good algorithm. From the start, someone has to build "disseminability" into the tool, with robust, flexible, and extensible code. Then, someone has to package the tool in a way that makes it accessible to a wide audience. Finally, someone has to publicize the tool, build a community of users, and support and maintain the tool.

In an ideal world, that "someone" would include a team of people with diverse skills—such as software engineers, technical writers, and marketers. But, in reality, it is often a scientist moonlighting as all of the above. Tool dissemination has traditionally been underappreciated and underfunded, making it hard for researchers to dedicate resources to tools beyond what's needed for their science. Fortunately, this situation is changing—with initiatives such as the NCBCs that recognize the importance of tool development and dissemination—but there is still a long way to go.

So how do scientists manage to do it right? *Biomedical Computation Review* spoke to a panel of individuals who have disseminated popular open source biomedical tools to find out what it takes to succeed and how they pulled it off.

Laying the Ground Work

The ingredients for successful tool dissemination have to be built into the tool's core from the start.

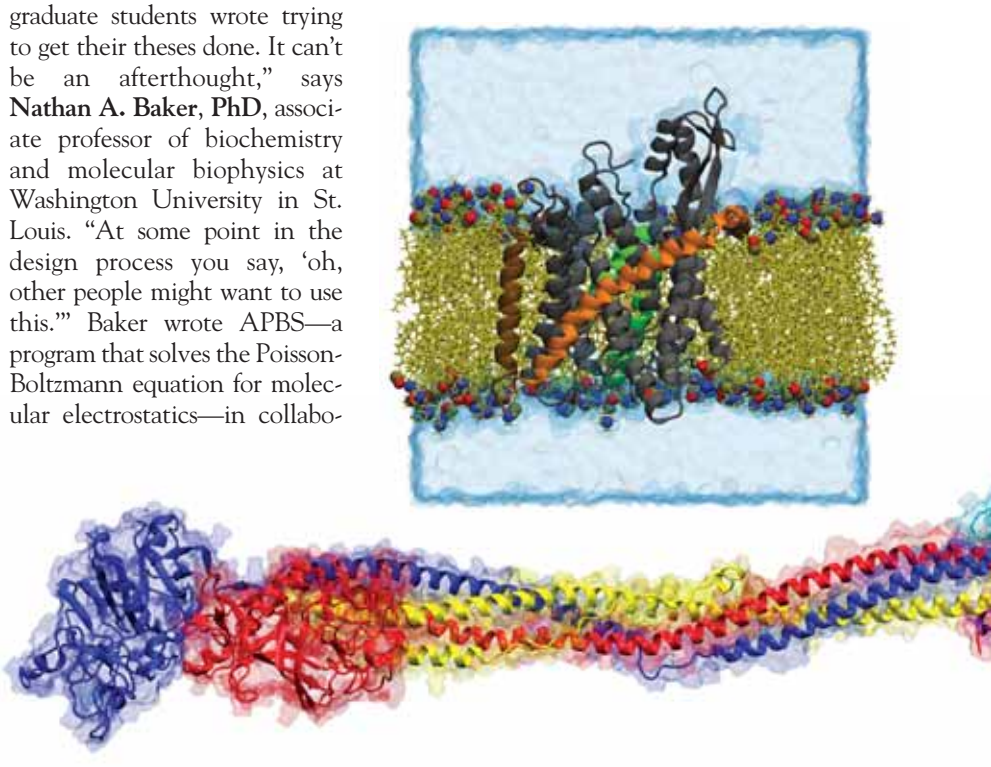
"You can't assemble a software package out of a bunch of code that your graduate students wrote trying to get their theses done. It can't be an afterthought," says **Nathan A. Baker**, **PhD**, associate professor of biochemistry and molecular biophysics at Washington University in St. Louis. "At some point in the design process you say, 'oh, other people might want to use this.'" Baker wrote APBS—a program that solves the Poisson-Boltzmann equation for molecular electrostatics—in collaboration with colleagues at the University of California, San Diego; the program is downloaded about 1000 times a month (http://apbs.sourceforge.net/).

When Baker realized that APBS offered something new that might be widely useful, he says, "I took most of what I'd written at that point and just deleted it and started over." A tool that is going out to others has to be built according to professional software design principles, he says. The code should be clean, bug-free, and robust; and it should be built in a flexible, modular fashion so that others can add to the tool and adapt it to their own problems.

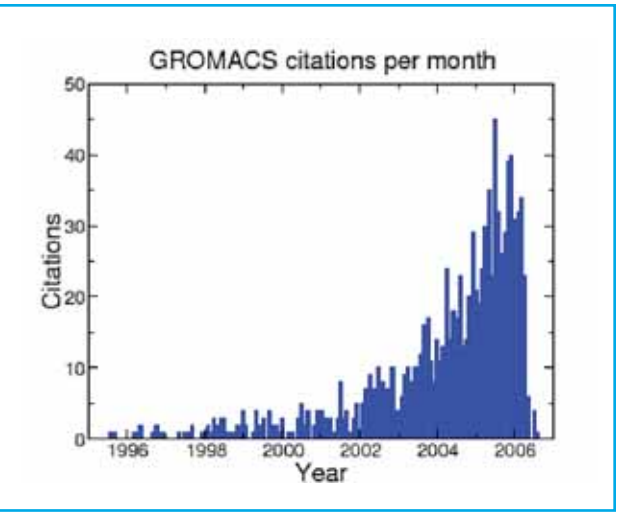
"There's a world of difference between developing code for yourself and developing code that you want to distribute," Schulten agrees. Establishing the proof of concept takes 10 percent of your time, whereas adhering to professional design principles takes 90 percent, he says. "And it is almost impossible to convince any normal scientist to spend that 90 percent." Professional programmers helped design VMD and NAMD, and they were a key factor in the tools' success, he says.

Dressing Your Tool for Success: Accessible, Well Documented, with a GUI

To become widely used, tools also have to be accessible—which means open source, portable, well documented, and user-friendly.



VMD Visuals: (top) secY protein, (lower left) fibrinogen protein,(lower right) polio virus particle. Picture made by the molecular graphics software VMD. Despite initial challenges, VMD is now a clear dissemination success story. The software is even used in high school classrooms. Courtesy of: the Theoretical and Computational Biophysics Group, NIH Resource for Macromolecular Modeling and Bioinformatics, at the Beckman Institute, University of Illinois at Urbana-Champaign.



Growing a Tool. The use of GROMACS software has spiked since 2000: There has been growth every month in the number of citations to one or more of the three GROMACS papers or the manual. Courtesy of Erik Lindahl.

"Science is about getting things out there," says **Erik Lindahl**, **PhD**, associate professor in the Center for Biomembrane Research and the department of biochemistry & biophysics at Stockholm University in Sweden. "Unless you have this great 10 million dollar idea that will make you a fortune, the last thing you want to do is to limit access to your work." Lindahl is a primary developer of GROMACS, a molecular dynamics simulation package developed at the University of Groningen, which has been cited more than 1000 times (http://www.gromacs.org/).

When GROMACS was released in the early 1990s, it was not open source—academic users had to sign a contract and industry users had to pay a fee. But the licenses were a hassle and Lindahl barely broke even paying for the secretary to handle them, he says. "So, we realized this wasn't really very smart."

When they moved GROMACS to open source, their user base quickly jumped from 1000 to 5000 and continued to climb from there. The community also started voluntarily fixing bugs and writing new modules and patches. "Everybody benefits from the openness. So I think overall it's been an incredibly positive experience for us," Lindahl says.

GROMACS follows the GPL-style open source license, which requires those who adapt the software to make their programs open source as well. Other tools in this article follow the less restrictive BSD-style license. "If I was starting from scratch, I'd seriously consider going with this completely open license," Lindahl says.

"BSD actually worked out quite well for us," says **Steve Pieper**, **PhD**, founder and CEO of Isomics, Inc., in Cambridge, MA, and the dissemination core PI for the NCBC NA-MIC (National Alliance for Medical Image Computing). The NA-MIC toolkit includes visualization software: VTK, ITK, and Slicer (http://www.na-mic.org/Wiki/index.php/NA-MIC-Kit). The BSD license has allowed medical imaging companies to incorporate bits and pieces of the software into their equipment—which gets the technology out where it can directly benefit patients, Pieper says.

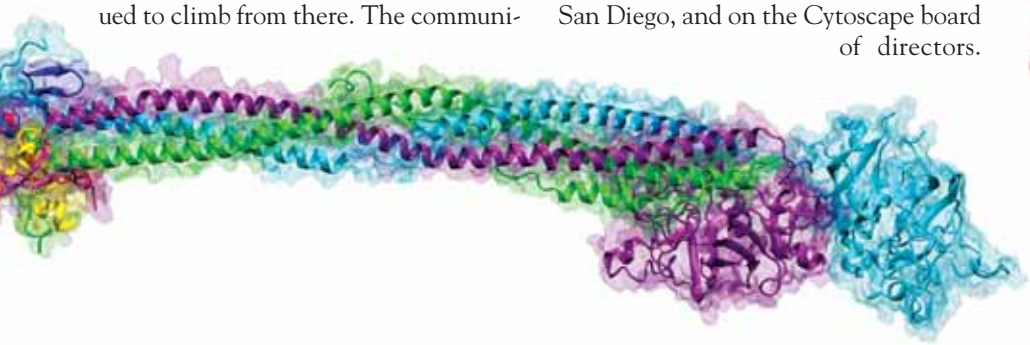
Cytoscape—which also follows the BSD license—has similarly been incorporated into several commercial software applications, says **Trey G. Ideker**, **PhD**, associate professor of bioengineering at the University of California, San Diego, and on the Cytoscape board of directors. Cytoscape is a software platform for modeling molecular interaction networks that gets about 3000 downloads per month (http://www.cytoscape.org/).

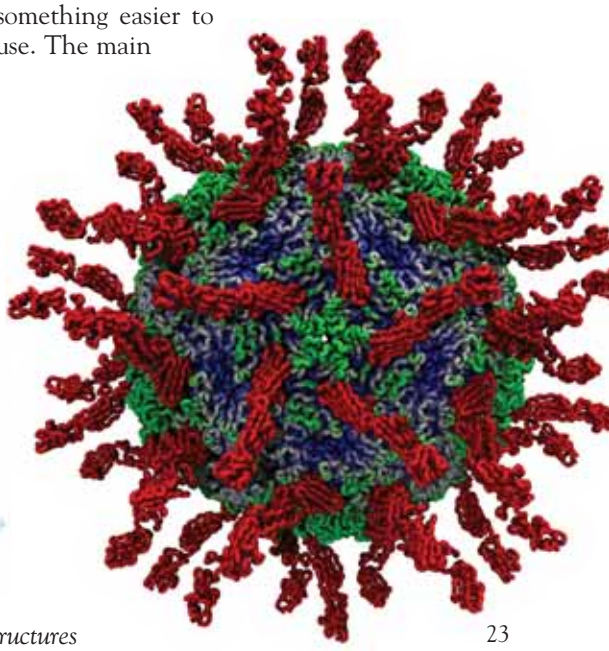
To be accessible, tools not only have to be free but also have to work on the computers that biologists are using, says **Thomas L. Madden**, **PhD**, a scientist at the National Center for Biotechnology Information at the U.S. National Library of Medicine. Madden helped transform UNIX-based BLAST into a tool that runs on multiple platforms, including Windows and Mac OS. BLAST is a sequence alignment tool and an undisputed tool success story—the original BLAST paper was the most highly cited biomedical paper in the 1990s (http://blast.ncbi.nlm.nih.gov/Blast.cgi).

"A lot of bioinformatics tools are only made for Linux or Unix, but we've had just as many downloads of the PC version of BLAST as the Linux version," he says. "I think you can figure that just about every lab has a PC. So I don't think you can underestimate the importance of that."

Once users have a tool in-hand, if it is technically difficult or poorly documented, they are likely to seek out something easier to use. The main

"You can't assemble a software package out of a bunch of code that your graduate students wrote trying to get their theses done. It can't be an afterthought," says Nathan Baker.





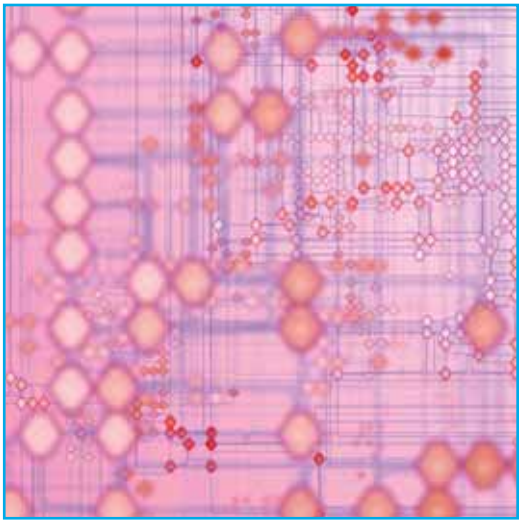
"Unless you have this great 10 million dollar idea that will make you a fortune, the last thing you want to do is to limit access to your work," says Erik Lindahl.

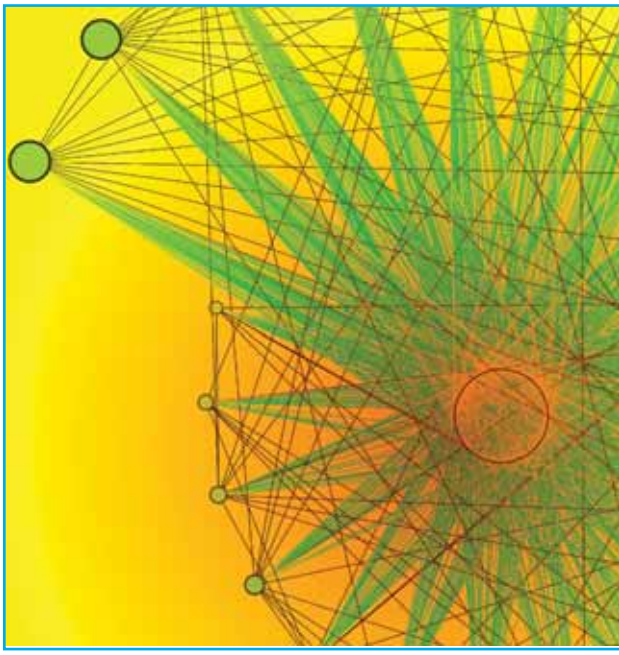
reason scientists flock to commercial alternatives for open source software is not because of superior performance (often the opposite is true), but because of a great user interface and great documentation, Lindahl says. Open source tools often fall short on these aspects.

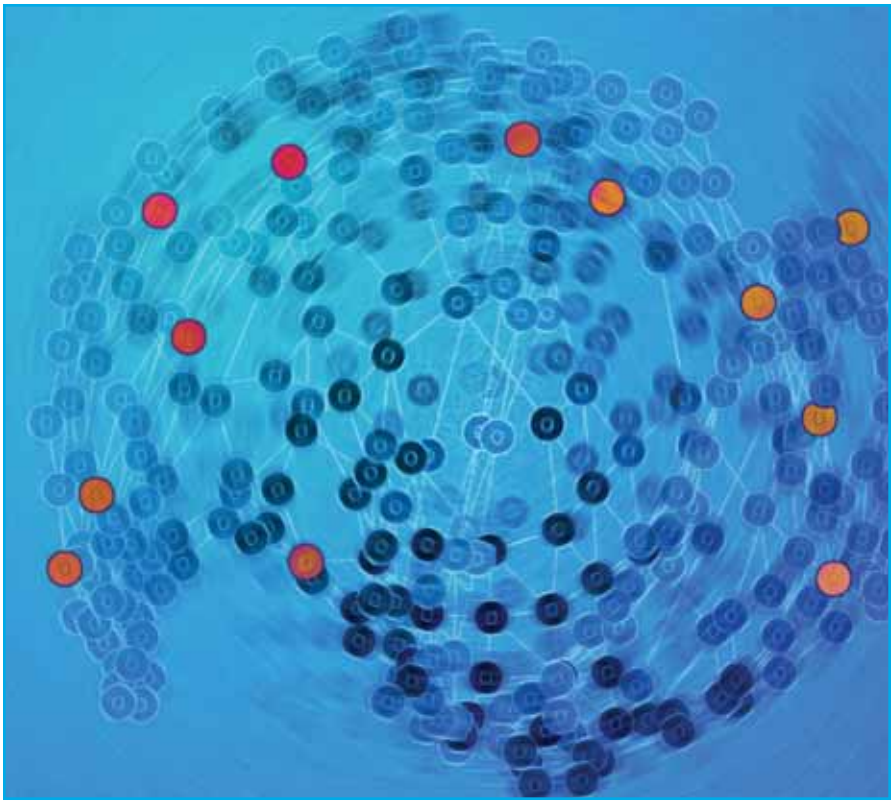
"I'm a sucker for good documentation. If there are not clear PDFs with graphics, I'm extremely unlikely to use it," says **Raymond R. Balise, PhD**, a biostatistical programmer at Stanford University, who uses the open source statistical package R, which has hundreds of thousands of users (http://www.r-project.org/). But the best programmers are usually not the best writers, he says. "So you have brilliantly designed elegant packages—and then good luck reading the documentation."

To help make the documentation more user-friendly, several of our interviewees advocate "learn by example" tutorials, which lead users step-by-step through common research problems.

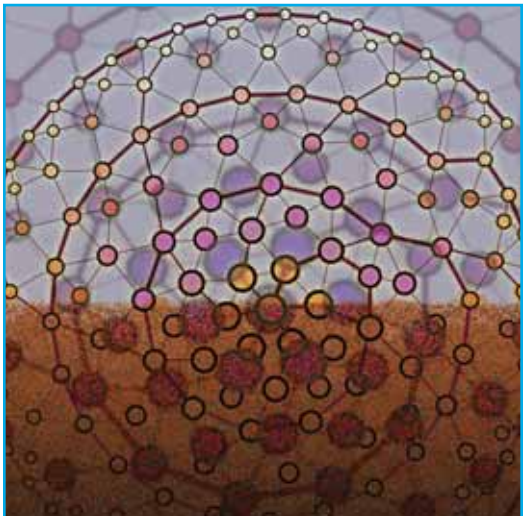
Many potential users are also deterred by the lack of a graphical user interface (GUI). For example, Baker says of APBS: "It's no worse than the other command-line computational biology tools. But I would say that maybe 80 percent of our audience would prefer to interact with it in some other way."







Cytoscape Pathways (Including background image on page 21). Pictures generated from Cytoscape, software for visualizing complex molecular interaction networks. Cytoscape follows a "non-viral" open source license, which allows companies to incorporate the software into their own commercial tools. Many companies now rely on Cytoscape as a critical part of their tools. Courtesy of: Vuk Pavlovic and Benjamin Elliott, the University of Toronto.



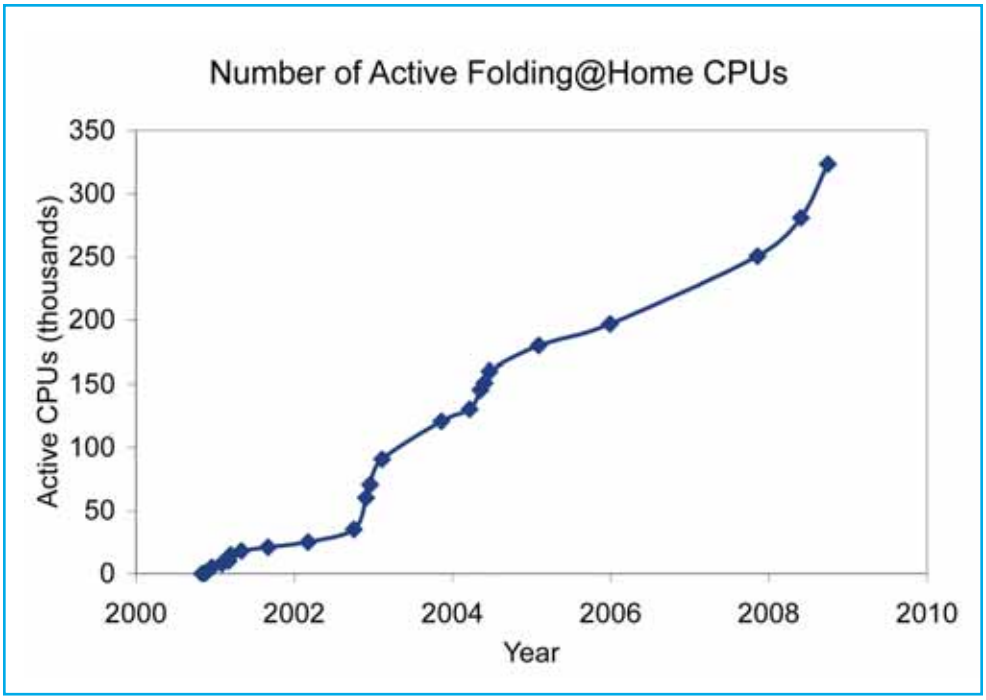
Similarly, R is a great tool for mathematicians and statisticians who are used to difficult programming languages, but telling physicians or biologists to "learn to program" just doesn't fly, Balise says. To make tools accessible to a wider audience, you need to wrap a nice GUI around the package and build in checks and balances to alert users if they're doing something wrong, he says.

Tool developers often resist these steps for fear that they will have to sacrifice power and flexibility for usability. An easy-to-use GUI-based interface is too constraining for research-driven tools, such as R and Bioconductor, that need to keep up with the cutting edge of science, says **Martin Morgan, PhD**, a core developer for Bioconductor, an R-based tool for analyzing high-throughput genomic data that has tens of thousands of users (http://www.bioconductor.org/). These tools may never be a satisfactory solution for a general audience, says Morgan, who is also a staff scientist and director of the Bioinformatics Shared Resource at the Fred Hutchinson Cancer Research Center in Seattle, Washington.

But usability can evolve, even if the tool was designed for expert users. For example, community developers have spontaneously added GUIs onto several programs—including R Commander for R, and PyMOL and VMD plugins for APBS. Core developers may also revisit usability as a tool matures. For example, BLAST's core developers have become more focused on ease of use in recent years, particularly for the BLAST webpage interface, Madden says.

In rarer instances, developers consider usability from the start. This was the case with GenePattern, says **Jill Mesirov, PhD**, director of computational biology and bioinformatics and chief informatics officer at the Broad Institute of MIT and Harvard. GenePattern is an analysis program for genomic and proteomic data, which also captures users' steps in a reproducible pipeline; the package, released in 2004, already has thousands of users (http://www.broad.mit.edu/cancer/software/genepattern/).

From the beginning, GenePattern's developers recognized that they were targeting two audiences: "We have a number of computational scientists who do a lot of their own coding. We also have a lot of bench biologists who want to do analyses but don't want to write code. And why should they?" Mesirov says.



Power Surge. The number of active computers running Folding@home has surged since 2000. Courtesy of: Vijay Pande, Stanford University.

So, they developed the program to be modular and flexible for expert users, including allowing it to interface with standard programming languages such as MATLAB, Java, and R; but they also provided a point-and-click GUI.

"I think it really is the non-programming community that has made the package so popular," she says. "We get emails from both types of users, and we get really effusive ones from the non-programming users, because they say 'Wow, this really lets me use all these sophisticated tools and I can do it on my own,'" Mesirov says.

Connecting to Your Audience

The next step in tool dissemination is the actual dissemination—connecting the tool to users. This means not only getting the word out about the tool but also "selling" it.

"There is a mentality that if the tool is good enough it will speak for itself," says Stanford University's **Joy Ku, PhD**, director of dissemination for Simbios and its tools, including SimTK Core, a toolkit for physics-based biological simulations (http://simtk.org/home/simtkcore), and OpenSim, a package for modeling musculoskeletal movement (http://simtk.org/home/opensim). But, in many cases, particularly for complex tools, you really need active outreach to show people how the tool applies to them and how to use it, she says.

Outreach often starts with a publication that announces the tool. In the early days, people discovered BLAST primarily through the publication and word of mouth, Madden says. BLAST solved a key problem, so it was obvious how it was useful. Nowadays, "lightweight" outreach on the web can also go a long way, he says. You can reach many potential users with little cost through newsgroups, email lists, bloggers, and even random web searches.

"I'm a sucker for good documentation. If there are not clear PDFs with graphics, I'm extremely unlikely to use it," says Raymond Balise.

"One thing that worked very well for us is the web," Schulten agrees, speaking about VMD and NAMD.

"That really was a godsend because it's basically like we have a shop and our shopping window is the web." he says. "It's so easy to do and you reach so many people."

Non-programming users of GenePattern send effusive emails, says Jill Mesirov, "because they say 'Wow, this really lets me use all these sophisticated tools and I can do it on my own.'"

To promote FFTW ("the Fastest Fourier Transform in the West")—a general-purpose tool that performs Fourier transforms, which are often used in molecular dynamics simulations—creators **Steven G. Johnson, PhD**, assistant professor of applied mathematics at MIT, and **Matteo Frigo, PhD**, chief scientist and founder of Cilk Arts, focused heavily on web outreach. They posted benchmarks comparing their software with other FFT implementations; added FFTW links on websites that list FFT programs, as well as on sites that catalog free-software projects (such as freshmeat.net and directory.fsf.org); advertised on mailing lists; created their own mailing list; and answered questions on online discussions about FFTs, including providing links to FFTW and other free FFT software.

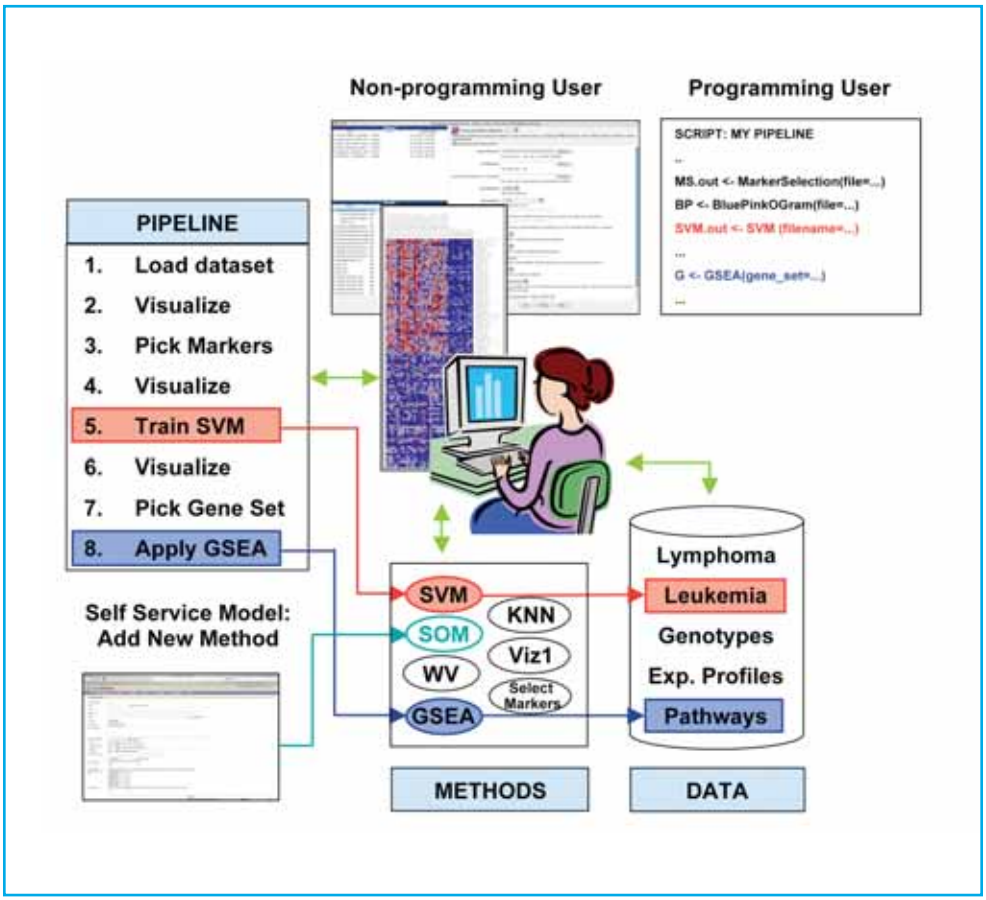
"Eventually, people began posting questions specifically about FFTW, and it was especially important to respond to these—having a support presence on public forums reassures people that the software works and is actively maintained," Johnson says. FFTW is now downloaded about 10,000 times a month (http://www.fftw.org/).

Active mailing lists and online forums help draw in new users, support existing users, and build a sense of community. "I frequently get much better support from open source mailing lists than you get from vendors," Lindahl says. Answering emails about the tool also goes a long way: "We've received over 10,000 email messages about FFTW over the past 10 years, and responded to a large fraction of them," Johnson says.

Beyond the web, more "heavyweight" outreach includes training sessions, workshops, and conferences. For example, Simbios and NA-MIC as well as other NCBCs hold training events at conferences and stand-alone workshops for developers and general users. Cytoscape developers run tutorials at the major bioinformatics conferences and some major disease conferences. It's hard to convince scientists to spend time running training sessions rather than improving the tool, Pieper says. So, it's important to involve people who are specifically interested in and passionate about teaching, he advises. R, Bioconductor, and Cytoscape hold their own annual conferences (funded primarily by corporate sponsors and paying participants), which help advertise the tools as well as bring developers together. "There's definitely a community, and the whole mentality of working as an international team is huge for R," Balise says.

High school teachers and college professors also promote tools in their classrooms. With VMD, "it became so user friendly that it could actually trickle down to college and high school education," Schulten says. "We were very fortunate that these outreach efforts were essentially ripped out of our hands. So now there are many efforts, and we just happily receive the news."

Distributed computing efforts are all about outreach, since researchers must convince the general public to download and run their tool. Coverage in

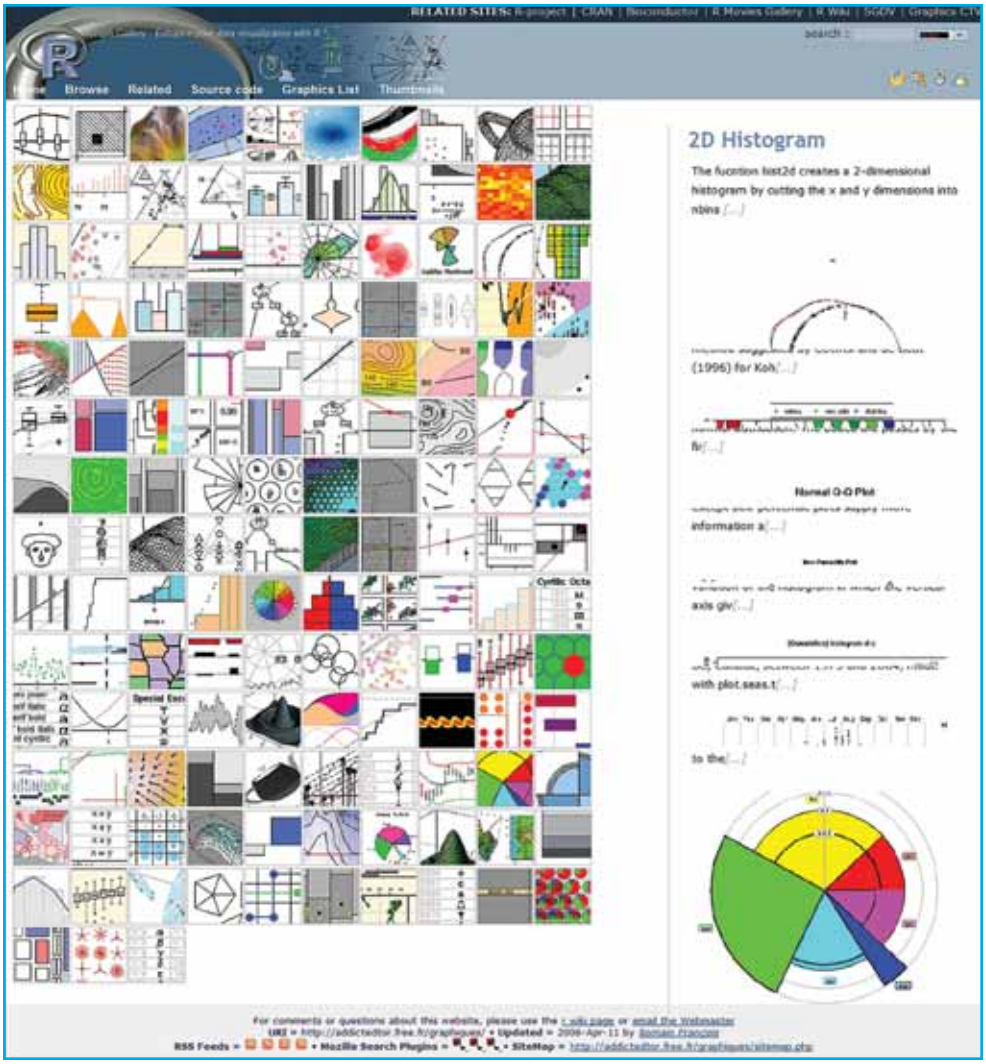


Building a Pipeline. The GenePattern tool helps expert and non-expert users analyze genomic and proteomic data, while capturing the steps in a reproducible pipeline. The tool was built with non-expert users in mind, which has been a major factor in the popularity of the tool. Reproduced from Reich M, GenePattern 2.0,* Nature Genetics *(2006) 38:500-501, supp. fig. 1.

the popular press (Time, CNN, and the New York Times, for example) helped generate buzz for Folding@home (http://folding.stanford.edu/), a distributed computing project at Stanford University led by **Vijay Pande, PhD**, associate professor of chemistry. Distributed computing also uses competition to stir up interest—participants collect points based on the amount of computing power they contribute. Capturing the high score is reminiscent of holding the high score on Asteroids at your local video arcade back in the eighties, but this is on a much grander scale, Pande says. "It's something on a very high profile site, where you can be number one out of hundreds of thousands."

Competitions are something we'd like to explore, Ku says. Already, Simbios runs a traditional grant competition for seed projects, which generates interest in and awareness of their center. "Ultimately you're only going to fund a small percentage of applicants, but all the applicants have to become familiar enough with what you're doing," she says. A similar approach could be used for software.

So, which of these outreach efforts is most effective? Until this year, we've just been going by an intuitive feel for what works, Ku says. But, in an effort to improve dissemination, they collected eight months of data on how people find their software project repository Web sites, simtk.org. The breakdown is: 29% word of mouth; 25% publications and conferences; 24% web search; 13% mailing lists and newsgroups; 9% other mechanisms (including use in the classroom, *Biomedical Computation Review*, and links on other Web sites). Word of mouth leads the way, but it accounts for less than one-third of hits—so more active outreach is vital.



R Gallery. Community developers have written so many graphical programs for data visualization in R that it's hard to keep track of them; here the programs are cataloged visually for easier access. Contributions from the community have been critical to R's growth and success. Screenshot from the R Graph Gallery, http://addictedtor.free.fr/graphiques.

Making it Happen

Successful tool dissemination can be lengthy and costly, and it requires diverse skills, such as programming, writing, marketing, and teaching. So how do scientists support these efforts?

"Up to now it's frequently been the case that you're kind of moonlighting," Lindahl says. "One problem both in Europe and in the States is that it's hard to get funded only for software development." Many tools are supported using bits and pieces of resources scrounged from science-driven grants as well as many hours of volunteerism—from professors, graduate students, postdocs, and community members. Under this piecemeal model, there's no money to hire professional programmers let alone technical writers or outreach coordinators. Lindahl says he'd "nudge" postdocs to turn code they wrote for their research into formal GROMACS modules. Pande says he and his graduate students have to work 60 to 70-hour weeks to keep Folding@home going. "It's just a lot of work to be running something like this," Pande says. Johnson says he and Frigo did most of the legwork for FFTW themselves over the years, despite many other time commitments.

"One problem both in Europe and in the States is that it's hard to get funded only for software development," says Lindahl.

Tool upkeep and dissemination are also undervalued when it comes to academic promotion—making it even harder to justify dedicating scarce time

and resources to these endeavors. "Academic credit for maintaining software is not the same as producing publications," says BioPerl developer **Jason E. Stajich, PhD**, Miller Research Fellow in the department of plant and microbial biology at the University of California, Berkeley. BioPerl is a programming toolkit for processing sequence data. It has been cited more than 500 times (http://www.bioperl.org/wiki/Main_Page). Stajich worked heavily on BioPerl before and during his graduate studies but, as he transitions to a faculty position, he needs to focus more on his science; and many other developers are in the same situation. "We'd like to do more outreach, but it requires a critical mass of people who actually have time to do that," he says.

To augment the piecemeal model of tool dissemination, some groups have formed non-profits. For example, Stajich and his colleagues formed the Open Bioinformatics Foundation, which provides infrastructure for BioPerl and related projects, such as BioJava and BioPython. Similarly, the Cytoscape Consortium provides an umbrella for the institutions involved in Cytoscape core development. The non-profit model can help with logistics, including accepting donations and running conferences.

Other tools in this article have managed to obtain tool-specific funding, which was likely instrumental in their success. For example, APBS, GenePattern, and some members of the Cytoscape Consortium have been funded through NIH's R01 program for "software development and maintenance" (which has been available since 2002). GROMACS has also obtained recent funding through the European Union. The funding gives us the ability to reply to user requests within 24 to 48 hours and to develop tutorials, Baker (of APBS) says. "Without that funding, that just wouldn't happen. It would be like, as with most previous funding, an afterthought in some grant: 'Oh, and by the way, I guess we'll keep this tool limping along.'"

As part of the NCBCs, Simbios and NA-MIC have specific funding for tool maintenance and dissemination. "One of the things that's great about the NCBC program is that there's funding to do actual training events," Pieper says. Finally, Schulten has had long-standing (two decades of) tool-specific funding through an NIH P41 grant—which specifically funds technology development. These funds allow him to hire professional programmers and run training events.

Measuring Success and Reflecting on Failure

The final step in tool dissemination is evaluation—measuring how well the efforts are going.

"It is extremely difficult to measure the popularity of a free software project like FFTW," Johnson says. Citations provide a rigorous measure of success, but these take time to accumulate. So, our interviewees also track softer measures including: registered users, downloads, mailing list subscribers, mailing list activity, Web site visits, conference attendees, and the number of plugins added to a tool.

This article focuses on tools that succeeded. But, for every success story, many more tools have failed. In a recent editorial in *PLoS Computational Biology*, founding editor-in-chief **Philip E. Bourne, PhD**, a professor of pharmacology at the University of California, San Diego, and his colleagues describe their efforts to track down 14 software programs (for partitioning proteins into domains) described in published papers. Eight programs were not even accessible in a usable form, let alone widely used and popular. Given the difficulty of the task and the lack of rewards, it's not surprising that so many tools languish into obscurity—but it is wasteful and reflects poorly on the biomedical computing community. "There's a huge amount of resource that goes into making these things, and so much of it is just lost." Bourne says.

Fortunately, funding agencies and journals are beginning to acknowledge the importance of tool upkeep and dissemination. In the past few years, the National Science Foundation (NSF) and NIH have "come around to the idea that software is not something to be dabbled with," Pande says. Lindahl has also noticed an increase in tool-specific funding. Journals could also help alter the reward system, Bourne says. *PLoS* is contemplating a software section where papers will only be published if the software is deposited in an open source archive such as sourceforge.net or bioinformatics.org. Online journal editors or readers could simply add a comment to papers when the software is no longer available, Bourne says. "That would sort of be a black mark against the author, so I think that might encourage the author to make the software available longer."

Even with more incentives and resources, tool dissemination will still be a challenge. Despite sufficient resources and a proven track record in tool dissemination, Schulten says his latest tool, BioCore (http://www.ks.uiuc.edu/Research/biocore/), is teetering on the edge of failure. BioCore is a collaborative work environment for biomedical research, supporting tasks such as co-authoring papers and sharing molecular visualization results. The program hasn't taken off yet, in part because scientists are reluctant to try new technology, he says. But Schulten is determined to showcase the tool more and run more training events. "We have to put more energy into these efforts," he says.

Success requires persistence, Lindahl agrees. "Don't give up in the beginning. It takes a while to build these communities." □

In the past few years, the National Science Foundation (NSF) and NIH have "come around to the idea that software is not something to be dabbled with," Vijay Pande says.