GOLDEN HELIX
Accelerating the Quest for Significance™

Is Free Software Really Free?

Examining the hidden costs of open-source bioinformatic software tools

Executive Summary

When considering conducting genetic analysis, researchers often begin by looking at open-source tools or consider building a program themselves. In some instances, these are the only options available. While open-source tools are a vital and necessary component of any bioinformatics toolbox, many researchers never consider the hidden costs of open-source software when a proprietary option is available.

In this white paper, the true cost of "free" software is explored, including time allocation, potential damage to one's reputation, and distraction from scientific discovery. Finally, five questions to consider in purchasing commercial software are discussed, as researchers weigh benefits and costs.

Background

There are many free things out there, and most people will ask, "Why should I pay for something that I can get for free?" This is a good question. The open-source movement has gained a lot of momentum in the last decade. In higher education, open-source programs such as Moodle (course management system: *www.moodle.org*), Kuali (financial system for academics: *www.kuali.org*), and Sakai (another course management system: *www.sakaiproject.org*) have gained popularity. The genetic research field is no different with open-source tools such as Bioconductor (*http://www.bioconductor.org*) and PLINK (*http://pngu.mgh.harvard.edu/~purcell/plink*).

There are many great benefits to using open-source software, and many genetic research analysis projects would not have been completed without them. Burton Group, now a part of the Gartner research firm, describes the tendency to go open-source (OSS): "Cost is often cited as the primary motivator for using OSS. After all, if the software is free, doesn't that mean that [the user] saves money?"[i]

With scientific research at universities dependent almost entirely on grants, a culture of frugality and resourcefulness perpetuates. In recent history, even fewer projects are being funded given the National Institute of Health budget cuts. Thus, genetic researchers live in an environment where budgets are tight, grants are limited, and competition is fierce.

And when a grant is obtained, there is a tendency to spend the money on generating the data as explained by Mark Gerstein, a Professor of Biomedical Informatics at Yale University: "Historically, analysis has always been underfunded relative to data production… Previously [researchers] saw the data as the valuable thing and the analysis was an afterthought and easy to do."[ii] When new sequencing machines have price tags in the millions, it's easy to see why.

With no money and such an "easy" task, a common belief runs rampant: that some graduate or post-doctoral student must be available to handle the bioinformatics (and they are cheap, right?). So Principal Investigators often rely on students to figure out the downstream analysis, with which they have little training or experience.

In an academic culture of self-reliance and required innovation, the prevalence of relying on open-source code or programming a one-off bioinformatics project from scratch seems logical.

But is "free" really free?

Jonathan Schwartz, former President of Sun Microsystems (acquired by Oracle in 2010) characterized open-source as a "free puppy",[iii] i.e. the puppy may be free, but the food, veterinary bills, and toys are costs that would not have incurred otherwise.

Open-source programs carry "hidden" costs that many researchers never consider: time and resources, reputation, and purpose. Let's examine each.

Hidden Cost 1: Time and Resources

Ask any bioinformatician what skills you should learn to go into their field, and you're likely to get myriad answers beginning with programming languages such as Perl, Python, R, C/C++, Java and general familiarity with a command-line environment. Even for those who aren't bioinformaticians, basic genetic analysis today requires researchers to be quasi-computer programmers. Why? The reliance on free command-line tools that require "code writing" skills.

Programming languages such as Perl and R are free, but they are, of course, command-line based. Commercial software companies generally focus on usability and intuitiveness of their products in order to properly serve their customers. When purchasing software, users typically have higher expectations for how easy software is to use and learn. There is general agreement in the industry that proprietary systems have better user interfaces.

By relying on "free," scientists are often stuck with command-line tools. If there is a program out there that already does what they are looking for, a researcher can use that and not start from scratch (although they will still have to understand the coding language to run the analyses). If developing a new method, they may instead decide to write a whole new program themselves instead of investing time understanding a complex, ill-engineered code base.

Learning a programming language is no easy task. Picking up Perl, for example, isn't something that can be done in an afternoon.

Open-source software is like a "free puppy."

Learning any computer coding language takes time, effort, and a lot of frustration. One case study is PhD student, Sander van der Laan, who completed an internship at Erasmus working with data for over 800 patients using a free, command-line tool. After six months of trying to wrangle the data into something that could be used for analysis, he realized the data wasn't good enough for reliable association results, and he had to start all over again.[iv]

James Schnable, a graduate student at University of California-Berkeley, warns: "A large number of [grad students] will eventually drop out. Staring at command line and struggling through introductory books on scripting languages… isn't how they picture spending their time in grad school." [v]

While the attitude of grad students being cheap labor is prevalent, stipends paid to these students add up. A comprehensive review of graduate school stipends found most to be between $20,000 and $30,000 a year for biological sciences.[vi]

And it only goes up from there. Genetic Researchers' salaries average $54,000, Professors of Genetics get a bit more at $57,000, and Principal Investigators receive $116,000 annually.[vii] Having researchers (or post-docs or lab personnel) learn a coding language is costly when salaries are considered.

Often, this pain is felt most keenly when newly minted programmers (i.e. genetic researchers) have a question. Given the nature of open-source programs relying on their community's free time for support, other individuals using the program are frequently left trying to figure out the answer to their question or waiting for someone to get back to them. Documentation may be scarce or non-existent. If these types of questions happen frequently, a project may stall. Another outcome is when resources (such as documentation, discussion forums, etc) don't outline a clear path to follow. The frustration can be overwhelming.

Program interoperability has also become a problem as many one-off programs exist for performing one specific analysis, and each program has its own data format. As high-throughput sequencing has entered the scene, even more software—and thus, file extensions—is available. Conversion has become a nightmare as scientists start with one file format, change to another, and then translate to a third. "Data management" is now written into genetic researchers' job descriptions, along with all the headaches and time it requires. Once again, time and

resources are squandered on menial tasks just to get data to a place where it can be analyzed.

And a cursory overview of genomic software shows just how big a problem this can be. The North Shore LIJ Research Institute maintains a comprehensive list (formerly managed by the Rockefeller University) of over 500 genetic analysis software packages as of August 2010, the great majority of which are free.[viii]

However, examining the NSLIJ list reveals another problem—software viability. In January 2011, a study of approximately half of the packages showed that over 85% were not maintained.[ix] In this case, after finally finding a program that can conduct the analysis required, time now has to be spent researching whether a project was abandoned due to problems with the software, shifting priorities, etc. If the choice is made to use the program anyway, all questions and modifications will have to be "figured out" on one's own without the help of professional support.

There is also the risk of a lab becoming dependent on a free tool only to have it go stale when its author loses interest in the program or has new priorities. The time spent learning and investing in that software is wasted as researchers are forced to find a new program that is staying current with the fast-paced field of genetics.

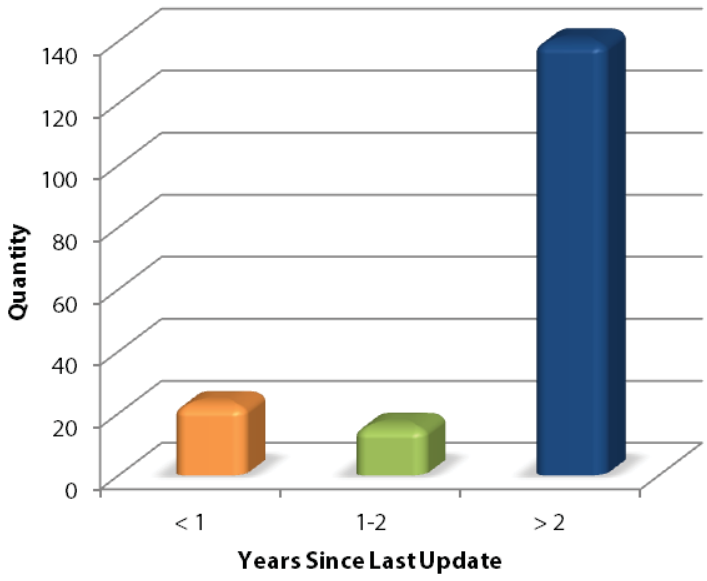
Relying on post-docs to perform the analysis presents challenges as well, as the constant churn of students results in duplicative efforts, reinventing the wheel, and the inability to reproduce results. Not to mention, of course, the cost to train new post-docs and get them up to speed.

On the flip side, researchers may publish custom one-off packages, which they then become committed to maintaining, supporting, and updating, requiring more time and effort than they originally intended. Questions about format capability, requests for additional functionality, and such will be a continual issue.

All the cost savings benefits have suddenly disappeared as paid staff invests large amounts of time (and thus their salary) in learning, researching, writing, converting, and supporting command-line tools.

The idea of proprietary software and the use of resources is summarized by Dr. Brad Wheeler (Vice President for IT, CIO, and Professor at Indiana University) who stated at the EDUCAUSE 2008 annual conference session entitled "The Community Source Model: Promise or Peril for Higher Ed?":

> "It would seem prudent if I can go buy [software] off the shelf, break the plastic, and install it and only customize it a little bit if I have to, that is a far better use of my staff time than to take my… people and have them sit in laborious meetings deciding [details of a new software program to be built]. Maybe that's not a real good use of resources."[x]



Over 85% of genetic analysis software programs listed by the North Shore LIJ Research Institute have not been updated in over two years.[ix]

Hidden Cost 2: Reputation

With the reality of "publish or perish" in academia, reputation is the measure of a scientist's success in the field with lasting consequences such as the ability to get tenure and receive future grant funding. One misstep and that reputation could be badly damaged.

What does the discussion of reputation have to do with free software? Take the example of Dr. Geoffrey Chang at the Scripps Research Institute. His group used a free program supplied by another lab for their biological analysis, but found out later that the software had an error and provided inaccurate results. Dr. Chang's group was forced to retract five papers from *Science*, the *Journal of Molecular Biology*, and *Proceedings of the National Academy of Sciences*.[xi]

The Science *journal cover of one of Dr. Chang's retracted articles.*

Professional software engineers are trained in the disciplines of writing clean code with descriptive comments, creating detailed documentation, and testing to make sure the software is solid and behaves as anticipated even for edge cases. While these practices do not guarantee perfectly accurate results, chances are much better when they are employed.

Many scientists, on the other hand, aren't formally trained in computer science and may lack the necessary skills to thoroughly test and annotate their programs. Not knowing the degree to which a program follows programming best practices, other scientists use the program and publish their results. As in the case of Dr. Chang, the results may be disastrous.

John Cook, a Research Statistician at MD Anderson Cancer Center, describes the difference between how scientists view software they write versus programmers:

> "To a scientist, the software is done when they get what they want out of it, such as a table of numbers for a paper. Professional programmers give more thought to reproducibility, maintainability, and correctness."[xii]

According to the *Nature* article, "Error...Why Scientific Programming Does Not Compute" published in 2010,

> "As a general rule, researchers do not test or document their programs rigorously... It [is] almost impossible to reproduce and verify published results generated by scientific software, say computer scientists. At best, poorly written software programs cause researchers… to waste valuable time and energy. But the coding problems can sometimes cause substantial harm, and have forced some scientists to retract papers."[xi]

Hidden Cost 3: Purpose

Dr. Adrian Sannier, Vice President of Product for Pearson eCollege (formerly Arizona State University's Vice President and University Technology Officer) is passionate about buying software. In the EDUCAUSE 2008 session with Dr. Wheeler, he proclaims:

> "I think it's really interesting that you don't see the CIOs of the petroleum industry [for example] saying… 'we really need to be spending our talent and energy and management building a financial system [for example]'… What they are doing instead is saying, 'we need to get out of this business entirely and find a provider…' That's the model of the future. Not writing your own code."[x]

He concludes his statements with:

> "What I want to do is make sure that we start to tackle the problems that are really problems and get away from the problems that [don't matter]… Let's get to the business of teaching and learning."[x]

Most genetic researchers didn't go into their field of study wanting to write code. Applying Dr. Sannier's comments, scientists should focus on science and leave building tools to others. When time, energy, and brain power are spent coding computer software, not much is left for scientific theory and exploration, the areas that scientific researchers excel at.

Although open-source tools have their advantages, such as the ability for researchers to easily introduce novel methods, the hidden costs of time and resources, reputation, and purpose make using open-source programs for all instances questionable. Yet, choosing proprietary software (also called "commercial software") has its trade-offs.

As you contemplate options, there are many things to consider.

Five Things to Consider in Choosing Proprietary Software

1. Vendor lock-in

A common concern for choosing proprietary software is becoming "locked in" to a specific vendor once project files and results are in the vendor's proprietary file format. Vendors that require proprietary formats and other measures that lock users in may do so based on the fear that their software cannot compete with other products on functionality or ease-of-use. In a survey conducted by Computer Economics in 2005, 44% of respondents replied that the most important advantage of open-source was "less dependence on vendors."[xiii]

Some commercial companies have responded to this fear by offering the ability to import and export data from their software into many different formats, including open-source tools. This functionality allows users to utilize other programs when necessary while also alleviating the data management overhead of dealing with numerous different packages.

Additionally, some proprietary vendors offer a free "viewer" so that datasets can still be viewed and exported even after you no longer have access to the software. Having a way to view their data, even after a vendor relationship has ended, is invaluable to researchers. When software has extensive import and export capabilities and a free viewer, researchers feel more freedom to use what is best for their institution and not be "locked in" to a vendor.

2. Ability to customize

Another argument commonly made for using open-source tools is the ability to customize the platform as desired. Seventeen percent of Computer Economics' survey respondents agreed that the ease of customization was open-source software's biggest advantage.[xiv]

While most proprietary software code is not made public, vendors have found other ways to allow users the ability to customize the platform. One way is through well-documented API's or a scripting interface, such as Python, where researchers who desire to do so can code custom functionality into the program. These customizations can even be shared among the community to get the best of open-source and commercial software combined.

3. Training and support

Upon deciding to use a new tool, training is essential to get up and running. Open-source tools often lack any type of training materials and lack the personnel to conduct any training. But this is often true for proprietary software as well. Be sure to ask what type of training a vendor provides on the software, methodology, and applications as well as what channels it is provided through—online tutorials, one-on-one web meetings, and in-person training.

Once a project is going, timely support can make or break a project, given the complexity of genetic analysis. While open-source communities can be a great resource when a user has a question, responses rely on other users and may not be timely or reliable. A huge benefit to commercial software is having a support channel. However, not all support is created equal. If a vendor has poor support, expected benefits may not be realized, and time may be wasted.

In choosing a vendor, be sure to ask:

- What are the hours for support?
- How long does it take support to respond to a user?
- How qualified are the staff?
- What ways is support offered (phone, email, webinar, etc.)?
- What types of questions does support handle?
- What is the reputation of support at company 'x'?

Training and support are vital to get up and running.

4. Frequency of updates

Open-source tools rely on the author(s) and community base for updates, and most of the time, this software is a side project. Updates may be scarce, and projects can go stale when an author changes focus or finds something new.

However, proprietary software can also go stale if a company's priorities shift. Any mature product will have regular updates with bug fixes and additional features.

Given the ever-changing landscape of genomic analysis, it is important for researchers—and software—to stay current. If a product is only updated once a year or less, it can lag behind on more powerful methods and new innovations.

5. Features

In the Burton Group report cited earlier, author Gary Hein observes: "An underlying theme is emerging: 'good enough.' OSS doesn't have to be the best or most innovative solution; rather, it must be good enough for the task at hand."[i] Genetics researchers will often release their own open-source tools once they are "good enough"—that is, they perform the function at hand in a manner that will suffice. Little concern is paid to anything beyond that.

"Good enough" is usually not "good enough" for most software vendors. Because they are commercial organizations, time is spent on polish, usability, and the making of a turn-key solution for marketplace viability. A robust, end-to-end solution is the goal and not just something that will "make do" for today.

However, when choosing proprietary software, researchers should take care that the tool performs all functions necessary to complete their work. Defining what capabilities are "must haves" versus "nice to haves" will assist users up front when considering software, as no package can do everything one could hope for. If software lacks a core functionality, researchers may be stuck using one-off tools for analyses that are not supported in their commercial tool.

Don't assume that just because a software vendor has released a finished product that it does everything.

"Free" is a Misnomer

With the growing popularity of next-generation sequencing, researchers will have more data than ever before to explore and analyze. Many haven't yet considered—how will all this data be analyzed?

And genetic analysis isn't easy when using command-line tools. It isn't straight-forward. It can't be taught in a one-hour course. It can't be explained in a one-size-fits-all wizard. It can't be given to a green post-doc with no instructions.

In summary, academia relies on "free" as budgets are tight and funding scarce. This fact has led to a dependence on command-line, open-source tools. These tools are often a cornerstone of bioinformatics and can provide some great benefits. However, there are hidden costs associated with using these "free" tools for every project.

- **Time and Resources:** Staff salaries of post-docs, researchers, and lab personnel are wasted in learning a computer-programming language, waiting to hear back on questions from open-source communities, managing data format interoperability, replacing programs that no longer have a user base, and supporting their own programs to others. Is learning a coding language a good use of resources? What does your time really cost?

- **Reputation:** Most scientists write suboptimal code. It isn't their fault—they aren't trained computer programmers. Using open-source programs that haven't been vetted is a gamble when publishing. Can your reputation take the hit of a retracted article due to bad software? In a world where reputation will make a difference in your position, your funding, and your lab are you willing to take the risk with unknown, free software?

- **Purpose:** Most researchers want to do research. Computer coding takes up energy and focus that could be used on your scientific purpose instead. What is your purpose as a scientist? Does free software help you or hinder you from that purpose?

However, proprietary software is not without drawbacks either. As researchers chose a vendor, there are five questions to make sure to keep in mind:

1. Will you be locked into this vendor?
2. Can you customize the software?
3. How extensive is the training and support they provide?
4. How often is the software updated?
5. Does the software have the features that you require to complete your analysis?

A final thought to consider. James Schnable quoted earlier, says: "Just because your dataset was expensive to generate doesn't mean you don't have to worry about the competition stealing the glory if you take more than a year to publish."[v] Are you willing to risk being "scooped?"

As the saying goes, "There's no such thing as a free lunch." Don't fall into the trap of ignoring everything but dollars and cents. "Free" software could actually cost you much, much more.

About Golden Helix

Founded in 1998, Golden Helix is known for helping genetic research groups working with large-scale DNA-sequencing or microarray data overcome the frustration and challenges of bioinformatic roadblocks: delayed projects, lack of quality findings, and low productivity. By empowering researchers with highly effective software tools, world-class support, and an array of complementary analytic services, we refute the notion that analysis has to be difficult or time consuming. Golden Helix's flagship software product, SNP & Variation Suite (SVS), is an integrated collection of powerful data management, quality assurance, visualization, and tertiary analysis tools for genetic data. SVS is delivered in a user-friendly, scalable platform and is supported by a team of highly trained bioinformaticians, statistical geneticists, and computer scientists that together make advanced statistical and bioinformatic methods accessible to scientists of all levels.

References

[i] Hein, Gary. (2005) Open Source Software: Risk and Rewards. *Burton Group.* Retrieved October 13, 2011, from http://www.immagic.com/eLibrary/ARCHIVES/GENERAL/BURGRPUS/B050803H.pdf.

[ii] *Bioinform*. (2011) Q&A: Yale University's Mark Gerstein on the Real Cost of Sequencing. Retrieved October 17, 2011, from http://www.genomeweb.com/informatics/qa-yale-universitys-mark-gerstein-real-cost-sequencing.

[iii] Donoghue, Andrew. (2005) Open Source 'is free like a puppy is free' says Sun boss. *ZDNet UK.* Retrieved on October 17, 2011, from http://www.zdnet.co.uk/news/application-development/2005/06/08/open-source-is-free-like-a-puppy-is-free-says-sun-boss-39202713.

[iv] Vionas, Jessica. (2011) Wondering what SVS can do for a PhD student? Just ask Sander. *Our 2 SNPs…* Retrieved October 21, 2011, from http://blog.goldenhelix.com/?p=845.

[v] Schnable, James. (2011) What NOT to do with your fresh RNA-seq dataset (a rant). *James and the Giant Corn.* Retrieved October 21, 2011, from http://www.jamesandthegiantcorn.com/2011/07/13/what-not-to-do-with-your-fresh-rna-seq-dataset-a-rant.

[vi] Chao, Wendy. (2010) America's Best Graduate School Stipends 2009-2010 Edition. Retrieved on October 21, 2011, from http://www.wendychao.com/science/stipends/2009-10.html.

[vii] Genetic Researcher and Professor of Genetics salary data from Simply Hired. (2011) Retrieved October 21, 2011, from http://www.simplyhired.com. Principal Investigators salary data from the US Bureau of Labor Statistics. (2010) Retrieved April 25, 2012, from http://www.bls.gov/ooh/management/natural-sciences-managers.htm.

[viii] An Alphabetic List of Genetic Analysis Software. (Last Update: October 21, 2011) *North Shore LIJ Research Institute.* Retrieved on October 24, 2011, from http://linkage.rockefeller.edu/soft.

[ix] Lambert, Christophe. (2011) "Dammit Jim, I'm a doctor, not a bioinformatician!" *Our 2 SNPs…* Retrieved October 24, 2011, from http://blog.goldenhelix.com/?p=652.

[x] Wheeler, Brad and Adrian Sannier. (2008) *EDUCAUSE 2008 Annual Conference.* Session: "The Community Source Model: Promise or Peril for Higher Ed?" Retrieved on October 17, 2011, from http://hosted.mediasite.com/mediasite/Viewer/?peid=ab4e38da501f4c3292dcffbee2a8fd92.

[xi] Merali, Zeeya. (2010) …Error … why scientific programming does not compute. *Nature*, 467:775-777, doi:10.1038/467775a. Retrieved October 24, 2011, from http://www.nature.com/news/2010/101013/full/467775a.html.

[xii] Cook, John D. (2011) Software Exoskeletons. The Endeavour. Retrieved October 25, 2011, from http://www.johndcook.com/blog/2011/07/21/software-exoskeletons.

[xiii] *Computer Economics.* (2005) Key Advantages of Open Source is Not Cost Savings. Retrieved on October 26, 2011, from http://www.computereconomics.com/article.cfm?id=1043.

Additional resources:

LinkedIn discussion. (2011) Changing to Bioinformatics. *Bioinformatics Geeks group.* Retrieved on October 21, 2011, from http://www.linkedin.com/groupItem?view=&gid=65325&type=member&item=63720228

Marx, Vivien. (2011) Bioinformatics Job Market Tug of War: Heavy Demand for Data Analysis vs. Tightening Budgets. *BioInform.* Retrieved on October 17, 2011, from http://www.genomeweb.com/informatics/bioinformatics-job-market-tug-war-heavy-demand-data-analysis-vs-tightening-budge.

Trappler, Thomas. (2009) Is There Such a Thing as Free Software? The Pros and Cons of Open-Source Software. Retrieved October 21, 2011, from http://www.educause.edu/EDUCAUSE+Quarterly/EDUCAUSEQuarterlyMagazineVolum/IsThereSuchaThingasFreeSoftwar/174575.