Sharing the value of software freedom with non-programmers

Presented to the ImageJ User and Developer Conference 2015
by Pariksheet Nanda

September 4, 2015

Most of you here in this room have a special appreciation for ImageJ. Obviously it's very useful to the work you do. As Curtis Rueden mentioned, a lot of that usefulness comes from the community's feedback, their voluntary help, and the code contributions over the last 25 years or so.

On the other end of the spectrum, there are a number of scientists, who need to use imaging, but who have many other things to worry about and would rather pay a commercial contractor to take on that responsibility. In principle, there's nothing wrong with that. Biology requires close attention to many things, so it makes sense to off-load some of types of work to spend time on things that you are an expert in. Within this group that needs imaging expertise, there's still great potential to create wonderful little imaging geeks. But the problem is this group traditionally doesn't actively participate in the community. Let's talk about why that is.

Today, the capstone of our educational systems teach us how to be independent thinkers, but it does very little to encourage independent learning in the use of scientific computing. What do I mean by *independent learning*? Well, one can actually imagine 3 types of students:

1. Those who are just not interested in learning. It's difficult to teach them,

2. Those who are curious and want to learn, but need some help getting there. Teachers are ideally suited to helping them. And finally,

3. Independent learners who already know how to learn, and don't need help. They don't really need a teacher. This last group of *independent learners* has achieved transcendence. You can give them nearly any problem and they can manipulate existing tools or create their own to solve it.

It is possible for any student to reach this highest level of independent learning if they invest the time and have the right type of help on hand.

But our way of typically interacting with most commercial software companies discourages independent problem solving. The interaction encourages dependence on them, by making us passive and disengaged. Let's see a few examples. Companies hire *Application Scientists* who often were biologists before joining industry. The job of an Applications Scientist is to an expert in your job. When you need help, you call on an Application Scientist and have instant answers to complex computing questions. Another example. Fixing a software issue these days, very often will involve a remote desktop. Someone will log in over the internet, see your computer screen, and control your mouse and keyboard. It saves companies a lot of time, and it's easy for you.

I think these two examples: the Application Scientist, and the remote desktop really symbolize the difference in approach between proprietary and free and open source software.

When you ask a question on the ImageJ mailing list you create a public record. Correct answers to those questions save time for other people in the same situation. If there is a bug in the software it will more than likely be noticed and fixed quickly. One often gets valuable advice from other scientists along the lines of "you know this other approached for your problem would be more robust". As we try out suggestions and explore new things we learn through our fingers. The unique selling point of working openly and transparently, is students work their way up to becoming independent learners and contributing back to the community.

Then consider the opposite. No public community and no transparency. Instead of communicating with the 2000 ImageJ subscribers, you communicate privately with your one Application Scientist. This one person acts as the gate keeper of all knowledge and isolates you from your own large, healthy community. You're only ever going to be as intelligent as that application scientist. If you need help, you have your remote desktop. But in both cases, one rarely learns the thought process, or has access to the resources of solving the problem. And there is no public, searchable record. In fact there is a strong incentive for these companies to never have public records of any kind because that would tarnish the illusion of the perfect product you're buying. To be successful in a low-volume, competitive market, their goal is to divide and conquor; To make you spend years getting comfortable using their product only so that you will want to buy that at your next position, and in nearly every case they succeed.

In fact, the imagery I would like to put in all your minds, is of these types of companies being like heliopter parents. Hovering around, taking care of your every desire. But you know what the problem with helicopter parents is? They want you to remain 5 years old forever. They don't want you to ever grow up. They don't want you to reach imaging transcendence.

As scientists pay between $5,000 and $30,000 dollars for the comfortable blanket of helicopter parenting. We want to be able to reach someone when things go wrong. Money spent in this way is a real shame. Free and open source software forms the super highways of research. ImageJ uses the most efficient financial model possible. We pay just for the development and the support. Pay once; use infinite. Yet scientists are diverted off the highway, into secretive side roads and waste money on proprietary software. The main reason this happens is because if you're hungry, you eat what's placed in front of you.

More scientists need to be careful about what they eat.

There are two things we as a community need can do, to foster the skills of independent learning:

1) Instil the confidence to work through technical problems and not giving up at the first sign of trouble, and

2) Fix the social problem of becoming an active participant, and an enthusiastic community member.

It's possible to achieve both these things by helping scientists grow as programmers. When you write software and you first start to get help from mailing lists and IRC, most people will have a religious experience. Total strangers, who are increadibly smart, take minutes and hours out of their day to help you; you, a novice. When that happens a few times, you get into karma debt, and that fuels the urge to contribute back to the community. Helping other people out is the best way to learn, and it starts you on that path to become an independent learner. I know people in this room have had this experience. Tiago Ferreira said yesterday, in regard to all the help he got from Johnnes Schindelin, "it would be cheaper if I open a BAR to repay all those beers". That was his inspiration for the name of the *BAR* update site[1] in FIJI.

To grow as programmers, there are 3 volunteer social organizations that help university students become better scientists.

How many of you here have heard of Software Carpentry[2]? For those of you who don't know, Software Carpentry was started at Los Alamos about 17 years ago, because the scientists and engineers using

[1] `http://imagej.net/BAR`

[2] `https://software-carpentry.org/`

the computing cluster did not have strong grounding in writing maintainable software. Today Software Carpentry runs over a hundred workshops a year to teach basic computing skills for reproducible research: Git version control, writing tests, using Makefiles, the Shell, beginner programming with Python, domain specific tools like R and MATLAB. The typical introductory workshop runs for 2 days. Volunteer instructors are trained by Software Carpentry, and are formed of grad students, postdocs, professors, people from industry. Any one of you can sign up to be an instructor, and if you don't have a strong background in one area it will even out from instructor training. There is a lot of focus on pedagogy for keeping students active, and help engage uninterested students and make them curious.

A more recent organization is OpenHatch, which specifically teaches the tools and culture of free software development with their *Open Source Comes to Campus* workshops[3]. The format is a 1 day immersion workshop, you start off with learning about IRC, Mailing lists, Git version control. In the afternoon there is a career panel where you can interact with 3 or 4 developers who are paid to work on open source software. But the highlight of he day is the contribution session. Attendees of the workshop participate on GitHub. They do testing, making sure they follow all the steps to build the project, and report them on the issue tracker where there are problems. They pick out and fix low hanging issues from the tracker. None of this requires a programming programming background.

For both Software Carpentry and OpenHatch, the class size is about 40 students, and they both have TAs or helpers. Generally, one helper for every 5 students. Being a helper is also a great way to transition into helping instructing one of these events.

Software Carpentry and OpenHatch are great, but there is one more requirement. What workshops do is create a scaffold of some knowledge and skills, but you still have to fill that in with your own hard work. Or, to use another analogy, you don't lose weight by watching other people run. Greg Wilson, over at Software Carpentry, refers to this requirement for practice as the "other 90%".

To be inclusive of students who want to practice their new programming skills, we need regular and informal group meetings where students can come and get help. For about a year now, Mozilla Science Lab[4] has been supporting Study Group events[5] to meet this type of need . Many Mozilla Study Groups organize[6] around a local required skill like Python or R. But the Study Groups are much more than simply meetings for open programming tools. A typical Study Group meeting can cover anything from code review, to digital lab notebook practices; all the habits that make you successful as an open scientist[7]. This is what's exciting about these Mozilla Science events: many of the practical benefits of open source software are being combined with the benefits of open science.

Explaining this idea of open source software is hard. Last year one of the themes of LibrePlanet[8] was *messaging* or, in other words, how to explain this idea of free and open source software to people outside of the community. There, at the birthplace of the Free Software Foundation, the take home message was it's difficult to explain the appeal of free software to someone who has never had any experience with it. The appreciation for the philosophy, the community, and the independent learning comes in stages. For ImageJ, we need a bridge to the online community, and that means giving people exposure to those subtle advantages in person. Because that's where we're losing ground to these companies. If you spend 5 hours a week helping out on the mailing list, spend one hour in person.

In conclusion:

1. Proprietary software is terrible because there are hidden costs and it has practical problems.

[3]`http://campus.openhatch.org/`

[4]`https://www.mozillascience.org/training`

[5]`https://www.mozillascience.org/introducing-mozilla-science-study-groups`

[6]`https://mozillascience.github.io/studyGroupHandbook/`

[7]`https://github.com/mozillascience/studyGroupLessons/issues/7`

[8]`https://libreplanet.org/2014/program/sessions.html`

   (a) It denies you from making copies and modifications.
   (b) There is no transparency when getting help in our low-volume, competitive market.
   (c) And worst of all, it denies you the freedom to learn.

2. Industry is supposed to do is commodotize: to make items inexpensive to buy. But in the case of scientific software many companies do the opposite; they charge exorbitant sums to repackage, because we are hungry for programming resources. They intentionally divide the community and sell us black boxes. How can we compare results if we can't see what the software is doing?

3. You don't change a system by complaining about it. You make it obsolete. At the BAR workshop, yesterday, Tiago Ferreira said, "scripting is an open door to Open Science" as it makes your work accessible and reproducible. Sharing software freedom with non-programmers, means turning them into programmers. But unlike traditional 1-to-1 help in collaborations, the exit strategy should be getting people used to working 1-to-many. Help scientists ask questions the right way[9], that would make a volunteer be able to help them. This means reaching out in person. There are fantastic social organizations which help us teach and practise the skills of open science and programming: Software Carpentry, OpenHatch, Mozilla Science. We have also our in-house experts: the Core Facility Managers. Students who need imaging and programming help, often talk to their Core Managers first. Give your Core Managers the confidence to say, "you know what, while I don't have the time to offer you this programming service, I know all these students who got help from this Study Group. They will be able to help you."

4. Finally, we're still sticking to the master plan of helping people. As Kevin Eliceiri said at the beginning of the conference, a lot of the value of ImageJ is the time spent giving high quality help. We can involve more scientists in that experience. There is high demand for learning programming, and starting more people down on that path makes for a great community.

Any questions?

[9] `http://www.catb.org/esr/faqs/smart-questions.html`