The Struggle of Sustainable Community Software and a Proposed Solution

Dr. Demitri Muna
Center for Cosmology and Astroparticle Physics, Ohio State University
muna@astronomy.ohio-state.edu

Perspective

This position paper for the inaugural workshop of the URSSI is written from the perspective of a longtime software developer from the astronomy community. While many of the problems presented here are astronomy-specific, it is extremely unlikely that most other quantitative research fields do not similarly suffer some (or all) of the issues discussed here. The proposed solutions presented are equally applicable broadly across scientific research, and in fact, would be better served in that manner.

Introduction

It is inescapable that any modern quantitative research not only utilizes, but is completely dependent upon software. For the purposes of this position paper, software will be placed into three classes. The first is commercial software, software that either has a price tag (e.g. Adobe Photoshop) or is free but commercially supported (e.g. operating systems like macOS and Linux). The second class is bespoke software specifically written to process or analyze a particular data set. For example, a software pipeline written for a particular telescope but could not be applied unmodified to another telescope would qualify. The third class consists of software research tools that are general purpose tools for a particular field. Examples from the astronomy field would include astronomical tools like Astropy, Starlink, IRAF, and SAOImage DS9.

The first class of software (commercial) is, by its nature, supported and sustainable. It has broad appeal and usage well outside of research. The second class (bespoke) is necessary for the success of any analysis and is generally sufficiently budgeted. It has a very focussed purpose; consequently it is used by a relatively small number of people, often poorly documented, and generally abandoned as projects end or evolve. Even if made publicly available and/or open source, it's not unusual for the software to be written or updated by a small number of people. The last class (community software) is the main focus of this paper. Researchers depend just as strongly on this class of software (maybe more), however it is this class that suffers from a lack of funding and sustainability, a problem that has only grown significantly over the past two decades.

The Astropy Problem

The problem of software sustainability in astronomy is best epitomized (but by no means exclusive to) a software package called "Astropy" (https://astropy.org). I wrote a position paper on this called "The Astropy Problem" published on arXiv (Muna, 2016, https://arxiv.org/abs/1610.03159). I will summarize the main points of the paper as it is likely variations of the story will be recognizable in other disciplines.

It was generally recognized by at least 2010 that the astronomy community would move towards the Python language. It solved many problems (e.g. it is free, it is a widely supported modern, object-oriented language) and was capable for scientific research. The largest barrier of adoption was that the community was dependent on libraries written in another language called IDL. To adopt Python, the libraries would have to be written in Python. It would seem reasonable that since there was wide acknowledgement that this was the inevitable path forward (i.e. there were not several camps advocating divergent paths), that one or more of the major astronomy institutions that have traditionally produced community software, none did (outside of skunkworks efforts hidden from management). Instead, a community of mostly graduate students and early career postdocs self-organized a distributed, grassroots project called Astropy to be the de facto Python package for professional astronomers.

Astropy was, and is, an unqualified success. Nearly every graduate student uses Astropy. Missions like the Hubble Space Telescope, the upcoming JWST, and WFIRST; archives like the Space Telescope Science Institute (STScI), NOAO, and IPAC; ground-based surveys like the Dark Energy Survey (DES), the Sloan Digital Sky Survey (SDSS), and nearly every other not only use Astropy, *they are critically dependent upon it.* Despite this, the project remains largely unfunded. While some people are allowed a small percentage of their time to contribute to Astropy, the majority of the nearly two hundred contributors have never seen compensation for their efforts. I estimated that as of July 2016, the total contribution to the community was worth between $8-9 million (at a rate of ~$1.7M/year).

Tragedy of the Commons

No one in the community disputes the usefulness of Astropy, or more broadly, general purpose research tools. However, to date all disavow responsibility to fund it. University departments claim their mission is to educate, not fund or generate software (and besides, most lack the expertise). Large-scale surveys claim their funding was awarded to generate the data and results of the project. Astronomical archive centers also answer to their funding sources (e.g. NASA, NSF), where every hour must be charged to a specific project/mission. The NSF funds specific projects for limited periods of time (e.g. 2-5 years), which is difficult for longer term development.

Of course, all *depend* on the software. This is a classic example of a tragedy of the commons: a situation where many members of the community enjoy a resource to their benefit (or self-interest), but do not contribute to its sustainability. The "resource" that is used here is the time and expertise (both astronomical and software development) of the contributors. Part of the price is that the more time one spends contributing to such efforts, the more it actually *damage*s one's career as this is time taken away from publishing papers. While there is a growing and commendable effort by journals to accept and encourage submissions of software and methods journal papers, they still do not carry the weight of a "science" paper when competing for academic positions. There are vanishingly few positions in the community to develop community software. Many of these core developers have left academia altogether to get "data science" jobs in industry that pay twice or more than the positions they left. In fact, funding for community software (in astronomy) has almost completely dried up over the past 15 years.

A Solution

The reality is that no current astronomy institution could have created the equivalent to Astropy, even if funding were available. This is partly due to institutional culture, partly due to management, and partly due to the extreme competition for funds. If reasonable funding were made available, likely it would have taken the form of competing efforts of more than one institution with significant overlap in functionality. The Astropy project has succeeded and flourished in an entirely different environment: distributed, open source and open development, and globally collaborative, focussed specifically on the *community* rather than a single project, mission, or institution. A project like this cannot, and should not, be housed in a single (existing) institution. A new model for open, community-developed sustainable research software is required.

I believe the best solution is the creation of, for lack of a better term, a "virtual institute". It would form an umbrella that will house community software development, training, and tutorials. Some of the positions would be full-time, career positions. However, we want to avoid such positions from drifting away from the community and current research needs. Therefore, some of the positions would be hosted by research institutions, e.g. 50% funded by the institute, and 50% by the research institution. This will allow a scientist to pursue their research interests, work directly with their peers, and continually see what software needs exist. The other 50% of their time cannot be dictated by the host institution, but is dedicated to community software development.

Of course, there is often no substitute for in-person communication and collaboration. The virtual institute would regularly host code sprints. but instead of one week at a time (which is usually enough to really just get started on a problem), these would be 2-3 months of focussed effort. Not only would these sprints be extremely productive, but they would be the minimum needed to bring a new person up to speed on a complex project to develop redundant expertise.

Training and workshops are another critical component of the solution. From the personal experience of running an introduction to programming workshop for astronomers annually since 2010, there is a *significant* demand for software instruction specifically targeted to scientists. Computer science courses contain far too much material that is irrelevant to a research scientist, and often lack the tips and tricks that are critical. Both a curriculum that can be given to universities as well as focussed, multi-day workshops (e.g. introduction to Python, machine learning, statistical methods) would be developed and offered by members of the institute. A web site that offers detailed tutorials should be a part of this.

The solution to the tragedy of the commons problem is one of shared responsibility. It is not the responsibility of the NSF to solely and perpetually fund an institute like this. It is the community's. Astronomy, as an example, is a global community. While the NSF might fund the seed to start such an institute, other funding bodies from the UK, Australia, India, China, Japan, etc. can all contribute. Individual university departments should contribute in the same way one would pay for a software subscription. Collaborations and funding from technology companies

(such as Google and Amazon) would create a bi-directional knowledge transfer. These efforts are very much in line with the aims of private funding sources like the Sloan Foundation and Moore Foundation. While none of the sources might contribute at the same level every year, the hope is that the cumulative funding would create a steady level.

Resisting the urge to fill out several more pages, these are the main points that must be addressed to fostering sustainable software:

- career-track, stable positions
- freedom to focus on community needs
- software development training and education for researchers
- room for "blue-sky" development; developing for future problems
- competitive salaries

The ideas presented here are brush strokes of what a viable solution would look like. I welcome suggestions, feedback, and collaboration.