A recommendation for a complete open source policy.

Authors

Steven D. Christe, Research Astrophysicist, NASA Goddard Space Flight Center, SunPy Founder and Board member

Jack Ireland, Senior Scientist, ADNET Systems, Inc. / NASA Goddard Space Flight Center, SunPy Board Member

Daniel Ryan, NASA Postdoctoral Fellow, NASA Goddard Space Flight Center, SunPy Contributor

Supporters

SunPy Board

- **Monica G. Bobra**, Research Scientist, W. W. Hansen Experimental Physics Laboratory, Stanford University
- **Russell Hewett**, Research Scientist, Unaffiliated
- **Stuart Mumford**, Research Fellow, The University of Sheffield, SunPy Lead Developer
- **David Pérez-Suárez**, Senior Research Software Developer, University College London
- **Kevin Reardon**, Research Scientist, National Solar Observatory
- **Sabrina Savage**, Research Astrophysicist, NASA Marshall Space Flight Center
- **Albert Shih**, Research Astrophysicist, NASA Goddard Space Flight Center

Joel Allred, Research Astrophysicist, NASA Goddard Space Flight Center

Tiago M. D. Pereira, Researcher, Institute of Theoretical Astrophysics, University of Oslo

Hakan Önel, Postdoctoral researcher, Leibniz Institute for Astrophysics Potsdam, Germany

Michael S. F. Kirk, Research Scientist, Catholic University of America / NASA GSFC

The data that drives scientific advances continues to grow ever more complex and varied. Improvements in sensor technology, combined with the availability of inexpensive storage, have led to rapid increases in the amount of data available to scientists in almost every discipline. Solar physics is no exception to this trend. For example, NASA's Solar Dynamics Observatory (SDO) spacecraft, launched in February 2010, produces over 1TB of data per day. However, this data volume will soon be eclipsed by new instruments and telescopes such as the Daniel K. Inouye Solar Telescope (DKIST) or the Large Synoptic Survey Telescope (LSST), which are slated to begin taking data in 2020 and 2022, respectively. Each of these telescopes will produce on average 15TB of data per day.

Managing and analyzing these data requires increasingly sophisticated software tools made possible by the emergence of powerful and low-cost computational hardware. The numerical simulations that allow us to effectively interpret these data are also growing in size and complexity. Computing has therefore become the backbone of theory and experiment and is essential in data analysis, interpretation, and inference (Vardi 2010).

Unfortunately, the culture surrounding computational work for scientific research has not kept pace with these developments. Laboratory experimenters are expected to keep detailed and careful records of their work, documenting materials and methods as well as positive and negative results. Paradoxically, no such requirements are currently expected of computational work though the same expectations of scientific rigor are required. This has several negative effects including (1) limiting the ability of scientists to build on past results and (2) a general inefficiency in the field as researchers are forced to redevelop existing software from scratch. Finally and perhaps more importantly, a lack of transparency could potentially lead to a diminished credibility of scientific research.

A number of workshops have already been held on the topic of "reproducible research" in other disciplines (ICERM Workshop on Reproducibility in Computational and Experimental Mathematics 2012, Roundtable at Yale Law School, a workshop at the Applied Mathematics Perspectives 2011 conference, SIAM Conferences on Computational Science and Engineering 2011, ICIAM 2011). They have all come to the same conclusion that software developed for scientific purposes must be made public (V. Stodden et al 2013).

As a funding agency, NASA is in a unique position to effectuate change by promoting policies to encourage reproducibility. **We therefore recommend that all NASA-funded science code be required to be open source**. We believe that such a policy without the support of additional guidance or guidelines will be of limited usefulness. A number of additional recommendations are described here. We believe the following should be part of a complete open source policy to achieve maximum positive impact:

1. The use of open and free software platforms must be encouraged.
2. The development of core software libraries must be directly supported.
3. A culture of computational reproducibility must be promoted as part of the research process.

4. The use of appropriate and modern tools must be taught to the research community.

Each recommendation is discussed in more detail below.

Open and free software platforms

In order for the full value of an open source policy to be realized the code should be accessible. If code is written for a platform that is not accessible because it is proprietary, not financially accessible or is written in an obscure language and therefore hard to decipher, then its release is of significantly reduced value. The easiest way to enable open source code to be accessible is to encourage the use of open and free platforms.

The benefits of accessibility have already been proven. Open and free platforms enable true open collaboration maximizing the number of available participants. One example of this success is the Python language, which is growing in popularity in scientific research. Python is a free, general-purpose, powerful, and easy-to-learn high-level programming language. First released in 1991, it began to thrive once Python 2.0 was released in 2000 and its development became transparent and community-backed. By some measures, it is currently the most popular programming language competing with languages that have been around for much longer such as C, C++, and Java.

Many additional benefits follow from the lack of a financial barrier of entry. The developer community is vast, diverse and enthusiastic. This has led to the development of an incredibly wide array of Python extensions. The Python Package Index currently lists over 113,000 user-submitted packages, providing functionality in such diverse domains as web application programming, numerical computing, artificial intelligence and the physical sciences.

Python is now used by many universities to teach basic computer science. It is now the case that 8 of the top 10 CS departments (80%), and 27 of the top 39 (69%), teach Python in introductory computer science courses (Guo 2014). For comparison, a proprietary and paid-for Matlab is the next most popular language relevant to scientific computing. Similarly useful for teaching basic computer science, it is used by less than 25% of the top 39 schools. Adopting a language such as Python holds the promise of reducing the burden on students to learn a new language for data analysis for their scientific research and therefore accelerate their ability to produce results. Another important advantage of using an open platform is that other large organizations (e.g. Wikipedia, Google, Yahoo!, CERN) make use of and contribute to the language. This means that graduate students and post-docs have developed a skill that can easily be transferred to industry.

The community-development ethos of open and free platforms have benefits for cross-discipline engagement. An example of this is the ongoing collaboration between SunPy (Mumford et al. 2015), a solar physics Python package, and Astropy (Robitaille et al. 2013), a much broader astronomy Python package. Crucially, both Astropy and SunPy have fully adopted the open source worldview that code that solves scientific problems should be shared as widely as

possible. This leads to solution implementations in free, open languages using free and open tools, thus lowering barriers to cooperation and collaboration. Thus, what started as a solar-specific tool can be made more broadly useful and SunPy developers can be more efficient and more productive by drawing on software already developed by Astropy. The consequent cross-fertilization of ideas is to everyone's benefit as members of different communities learn from what has been done outside their field. Finally, open source platforms are not dependent on the company or developer that originally created it. Even if the company fails, the platform can continue to exist and be developed by its users, a useful guarantee for scientific research which must consider longer time horizons than industry.

Core Software Libraries

Core libraries form the foundation upon which more complex analyses are built. They bring together common practices and workflows allowing for increasingly efficient software development and data analysis. Core libraries are where the primary benefits of an open source policy can be focused. As the community develops and releases software tools, they can be reviewed by the community, preferably through the standard peer review process. Code review also helps the integrity of scientific results. By having a transparent process whereby multiple people check code that is likely to be used by the community, it is more likely that mistakes will be caught, reducing the chance of incorrect scientific conclusions being drawn from false data analysis results. Widely used code can then be accepted as "standard practice" and incorporated into a core library. Researchers can then rely on the already reviewed code from the core libraries and build upon it. Core libraries can be seen a cyber-infrastructure on which the science is built. As with other kinds of infrastructure, core libraries must be maintained and managed. Unfortunately, current funding opportunities do not allow specific funding for this purpose and do not provide the correct incentives (Howison 2013)

Two examples of emerging core libraries are Astropy and SunPy. Both of these projects aim to provide a core library in their fields; Astropy for astronomy and SunPy for solar physics. They have both been managed, written, and operated as a grassroots self-organized volunteer effort and are currently being used by many members of the community. Yet both efforts are effectively unfunded. This problem is well described by Muna et. al 2016.

To provide some context to the problem, the experience with SunPy is relevant to this discussion. SunPy currently contains more than 33,000 lines of code from 126 contributors which represents ~9 years of effort (COCOMO model). Assuming an average salary of $100,000 (a rough salary for a highly skilled software developer in industry) along with a 1.5x multiplier for overheads this represents a total value of $1.2M. Much of the development and management has been performed by graduate students, undergraduates and researchers, with some help by scientists with permanent positions. This value is currently being extracted from the careers of young and capable scientists. These contributions are provided at the expense of their research and publication output in order to support the community. Since this work is not funded it is by definition not valued by NASA. Finally, it is difficult for university hiring

committees to quantify the value of community code contributions since there is no funding available to scientist-programmers. Despite this, the use of SunPy and other open source codes in solar physics community demonstrates that it is valued. We therefore strongly recommend that this work be appropriately funded. One important caveat should be attached to this recommendation which is that the traditional funding mechanisms which promote competitive bids should be modified so as to not lead to multiple core libraries being developed with overlapping functionality. Open source software development should be for the benefit of the community, that is to say it should prioritize collaboration over competition.

A Culture of Reproducibility

The key to the advancement of science is that it produces results which are reproducible. This allows multiple scientists to test each others results and reach a consensus. Currently, re-producing a computer-enabled result which is described in the literature can be exceedingly difficult. At best, the steps of an algorithm are described in words and a challenge is presented to a reviewer to implement the algorithm themselves. It is safe to say that most do not and therefore must take other scientists at their word.. As a funding agency, NASA can set the important standard of an open source policy which is the first step toward a robust culture of reproducibility. The community can then develop the appropriate standards to enable this. In order for this to happen we recommend that all proposals be required to provide the following details

- Platforms and software to be used.
- Reasonable standards for dataset and software documentations.
- Reasonable standards for the persistence of resulting software, data preservation and archiving.
- Reasonable standards for sharing resulting software among reviewers and the community.

Many of these recommendations are now very easy to achieve. The use of version control systems easily enables persistence of resulting software. With proper versioning when a result is published, the version numbers or commit hashes of the software can be published along with it. This would enable other scientists wanting to reproduce the result to ensure they use the exact same software as was used in the original study. It also enables the result to be obtained with newer, better software tools at a later stage to investigate whether the result has changed. Many good, free version control systems are already open source and freely available, such as Git (http://git-scm.com).

Software hosting and preservation can be achieved in a number of ways. Many university libraries now allow scientists to deposit their code and data into repositories which are assigned a permanent URL and DOI making it searchable in the library system (e.g. https://sdr.stanford.edu/). A large registry of data repositories exist worldwide (https://www.re3data.org/). Additionally, many free code hosting services are also available such as GitHub, GitLab, BitBucket, and SourceForge. These services provide software management tools that enable contributions and ease collaboration. It is also now possible to

publish software packages such that they can be directly cited in the literature (e.g., J. of Open Research Software, Zenodo). This enables all of the standard impact metrics to be applied to software and would encourage and credit the development of open-source scientific software development.

Educating the Community

A recent survey of software use in the astronomy community carried out between 2014 and 2015 with responses from 1142 astronomers found that 90% of responders across all demographics write their own software. Even though software development is so wide-spread only 8% responded that they received significant training in this area with 43% stating that they received no training at all (Momcheva & Tollerud 2015). A similar informal survey of NASA GSFC Heliophysics division scientists suggest similar results may apply across multiple fields. This lack of formal education in an area so important for the success of a scientist's career is a fundamental problem. **In order to support an open source policy we recommend that significant steps be taken to educate the community on an ongoing basis.** This will lead to more efficient, robust, and transferable code. Basic software development practices such as writing maintainable code, using version control, issue trackers, code reviews, and unit testing should be taught and their use encouraged so that they become community standards. Recommendations for those standards already exist (Wilson et al. 2017, Wilson et al. 2014). There are a number of ways that NASA can support this.

- Funding for dedicated workshops at annual conferences
- Forming a working group to develop community standards
- Providing funding for STEM and computer science integration projects (e.g. NSF's CS for all initiative)
- Funding mentorship programs to pair researchers with research software developers.

The funding of core package development teams should be contingent on a community educational mission. Such teams will be uniquely positioned to set community standards for documentation and coding practices. Open source software lowers the barrier to entry to scientific research for other audiences, such as schools, universities and the general public.

Summary

To support an NASA open source policy, we advocate for the wide adoption of the practices of free and open source software development as the primary tool to optimize scientific return within and across disciplines. Software development is now fundamental to a wide range of scientific disciplines that are important to NASA, but is demonstrably undervalued. We propose that steps be taken to educate the scientific community as to the benefits of the practices of free and open source software development, that funding opportunities for community-led free and open source libraries be developed, and that incentives be developed to encourage these practice. Free and open source software lowers the barrier to entry for multiple audiences and promotes the awareness of cross-disciplinary ideas and capabilities, thereby maximizing the scientific return on investment for everyone.

References

- Guo P, https://goo.gl/zRgE9p, Jul. 2014
- Howison J., “Incentives and Integration In Scientific Software Production,” Collaboration and Sharing in Scientific Work, pp. 1–12, Aug. 2013.
- Momcheva I., Tollerud E. J. , “Software Use in Astronomy: An Informal Survey,” Arxiv, pp. 1–20, Jul. 2015.
- Mumford, S. J., et. al, “SunPy—Python for solar physics,” Comput. Sci. Disc., vol. 8, no. 1, pp. 1–23, Jul. 2015.
- Muna D., et al., “The Astropy Problem.”, Arxiv, 11-Oct-2016.
- Stodden V. et al., ICERM Workshop, “Setting the Default to Reproducible,” pp. 1–19, Feb. 2013.
- T. P. Robitaille, et al., “Astropy: A community Python package for astronomy,” Astronomy and Astrophysics, vol. 558, Oct. 2013.
- Vardi M. Y., “Science has only two legs,” Commun. ACM, vol. 53, no. 9, pp. 5–5, Sep. 2010.
- Wilson G, Aruliah DA, Brown CT, Chue Hong NP, Davis M, et al. (2014) “Best Practices for Scientific Computing.” PLoS Biol 12(1): e1001745. doi:10.1371/journal.pbio.1001745
- Wilson G, Bryan J, Cranston K, Kitzes J, Nederbragt L, Teal TK (2017), “Good enough practices in scientific computing.”, PLoS Comput Biol 13(6): e1005510. https://doi.org/10.1371/ journal.pcbi.1005510