Towards Reproducibility using Open Development: Astropy as a Case Study

Primary Author

Dr. Erik Tollerud
Assistant Astronomer at the Space Telescope Science Institute & Astropy Project Coordinator, 410-338-6761, etollerud@stsci.edu

Contributing Author

- David W. Hogg, Professor of Physics and Data Science and Group Lead at Flatiron Institute. Supporter and Developer of Open Source Tools.

To put the starting point succinctly: if it's not reproducible, it's not science. While the exact definition of "reproducibility" is at times nebulous and is in part a social construct, we believe that most astronomers (and indeed most scientists) agree that the ability to reproduce a key result of a paper or other work is a core tenant of science. This relates to the questions posed in the "Best Practices for a Future Open Code Policy for NASA Space Science" call for White Papers in that there is an intimate connection between open code and reproducibility. A historical position holds that reproducibility can be high-level in the sense that the basic "idea" in a paper can be reproduced. However, in the data-intensive, precision era that astronomy is now firmly in, most significant results are based on extremely complex analysis procedures dependent on task-specific software, and this is unlikely to change in the foreseeable future. This directly implies that for science to be reproducible, the software for the science must be similarly reproducible. The most straightforward way to accomplish this is through Open Code processes and policies.

With this in mind, in this White Paper we consider how Open *Development* models provide such reproducibility, exemplified by the Astropy Project. We hope that this and the experiences of related community projects can guide NASA Open code policies both to facilitate reproducibility as described above, and to improve the overall health of the astronomy software ecosystem.

The Astropy Project has proven to be a successful example of the "open development" model of scientific software development (i.e. greater than 1000 citations to the code paper, which is a *lower limit* of its use in science). This means that Astropy has been developed both *by* and *for* the astronomy science community. While not all contributors are active researchers, contributions are *reviewed* by the scientists, and scientists can (and often do) contribute code and/or learning materials relevant for their science domain. At the same time, the project has adopted software engineering techniques and principles like Continuous Integration with unit tests and industry-standard tools that keep documentation close to code. This has led it to become the main Python software library in astronomy, and is used near-universally by Python users in the astronomy community. It has also led to an "ecosystem" of software packages, e.g. *photutils*, *halotools*, which are developed separately from the Astropy package, but follow the same development model pioneered by Astropy (and typically depend on Astropy for underlying data structures or utilities). All of this is predicated on an open source license that is open enough for contributors to trust the code will always be visible and useful to them, while not detering contributors with possible private interests like "copyleft" licenses such as GPL sometimes do. The exact choice of license for Astropy (3-clause BSD) is not critical, other than that it be

GPL-*compatible* (without being copyleft) so that it can be included in the full swath of existing open software repositories. While an exception process should be allowed, e.g. for ITAR-interfacing software or software that interacts directly with proprietary contractor code, there should be a high bar, and this kind of software should only be allowed where it does not directly impact the science content or reproducibility. By adopting a policy requiring such licenses for new (and, where possible existing) software, the NASA SMD can enable the thriving of comparable projects in other science domains.

This system has also led to improved efficiency in producing software, because astronomers know they can immediately fix something that is "close but not quite". Hence they spend less time writing their own custom software (a perennial problem in astronomy) in favor of using something they can trust because they see both that it works and that they can have a say if something goes wrong. Hence, a NASA policy that encourages Open *development* (i.e., one step beyond Open Code) could serve to both improve reproducibility, as outlined above, and improve the *efficiency* of every NASA dollar that goes into science. While individual PIs may find trouble having the time to support such a community, a wide ecosystem of open-developed software that is grounded on a NASA policy encouraging such practices would allow such PIs to "crowdsource" support if their code is actually of use to a significant chunk of the community. Precisely this process has organically grown up in the Astropy ecosystem, because Astropy provides the tools to make such support easier (the "astropy helpers"). While not all codes using these tools grow a community support network, in many cases this is simply because there is only need for them in a small niche group of scientists. This implies that developing codes in this manner leads to a natural and healthy distribution of resources such that codes with the most usage get the most support.

All that said, it is important to recognize that this Open Development model is *not* necessarily appropriate for small single-investigator projects (e.g., "one off" scripts for simple data analysis tasks). Even though the overhead can be made quite low, there is always *some* overhead to trying to build a community. Hence, any policy should recognize that there is a minimum bar where using this Open Development model may not be worth the effort, so a hard *requirement* of Open Development is not justifiable.

Similarly, this policy should *not* have the side-effect of chilling university contribution to NASA software. Some universities have intellectual property rules that are not obviously compatible with an open code NASA policy, so simply creating a policy without addressing this concern might scare university PIs away from contributing NASA software. However, funder policies on Open Source

enable PIs to work around these restrictions: IP offices at UW Seattle, NYU, and UC Berkeley all accepted the open code policies of the Moore-Sloan Data Science Environments as conditions of funding, even though these required modifications to the Universities' IP policies. A similar policy by NASA would plausibly have the same effect; indeed it could lead to universities adopting comparable policies, multiplying the positive impacts described here beyond NASA-funded science.

With all this in mind, we believe that NASA Science is best supported by a policy that requires (with a rigorous exception process) an open source license, but does *not* require it to be copyleft. It should also encourage (and provide resources for) an open *development* model for any NASA-funded science software, where this model is feasible. Such a policy has the potential to keep space science reproducible in the software-dominated future, while simultaneously improving efficiency through pooling of resources and expertise.

Appendix: Supporters

1. Tom Aldcroft (Chandra X-ray Center/Harvard CfA)
2. Adrian M. Price-Whelan (Princeton University)
3. Thomas Robitaille (Aperio Software)