Title: Current and future considerations for a NASA Open-Code Policy

Authors: Adam Kellerman, Steve Morley, & Alexa Halford with suggestions from the Assessment of Understanding and Quantifying Progress Toward Science Understanding and Operational Readiness Working group.

Summary:

We believe that open source code is something that our scientific community and NASA should embrace, but not require, as the benefits are many, however a requirement may limit use of commercial/controlled source codes and/or be impractical in some cases.

There are two main items which are covered in the sections below: (1) Availability and (2) Reproducibility. It should be a requirement that codes used to produce the results in a given study are archived, documented, and/or referenced appropriately, given the type and restrictions of the codes used (see the acknowledgment section below).

We strongly encourage open source, but there are cases where this will not be possible or feasible. The main aims here are to make the codes, or at least the functionality of the codes, available, so that any results may be reproduced with the information given.

Reproducibility is a fundamental part of the scientific process, and critical to scientific progress. Encouraging open source code and open source programing languages also lowers the buy in for scientists from smaller institutions, and less research cash flushed countries. Making it easier and cheaper for more scientists to participate in our field will help enable a more diverse scientific community.

There are some potential perceived pitfalls to requiring all code be made open source. We suggest that, for cases where developing open-source code is possible, and feasible, funding be made available for researchers to develop the code, document it, put it under source control, archive versions used for publication, and describe it appropriately in the literature.

NASA has already encouraged the first steps towards reproducibility and making it easier for all to participate in research efforts through requiring data from NASA funded missions to be made public, and through code of large models to be able to be run on request through the CCMC. We believe the next step is to highly encourage open source coding, and in the following paragraphs we outline the potential mechanisms to boost buy in from the scientific community.

Acknowledgement:

Just as data sources, and instruments used to collect those data, are cited and acknowledged in peer reviewed journal papers, it is important that code used to produce research results is also correctly described, and cited. Although not feasible for every study, a robust archival and citation process should be strongly encouraged. One possibility is that larger codebases may be given DOIs, are archived appropriately, which will help to keep track of which version of the code or coding package was used to produce results (https://guides.github.com/activities/citable-code/).

DOI's are issued by publishers, and thus will not be assigned by default for many codes. Along with providing DOIs, or another unique reference for the relevant code or coding packages, instrument or model like papers should be written for the purpose of introducing the new coding tools available to the community. The responsibility of ensuring or encouraging the proper citation of codes or coding packages would fall to the community and the journals in much the same way that they attempt to ensure that data and previous published work are properly cited.

It should be required that any codes used to create data or figures used in peer reviewed work be archived appropriately, and at a minimum be available on request. It should be encouraged to make such codes open source to allow faster access to, and reproducibility of paper results.

Here we provide two examples that illustrates why citations and open source should be encouraged, but should not be required. The first example includes Dr. X, who has used the Python scientific stack (numpy, scipy, scikit-learn) to fit a support vector machine to open data. The question is how far down the stack of software should they cite? This also begs the question, what is the preferred method for citing each of the libraries/codes? Not all codes have a DOI, and developers of each code/library may request the user to cite a paper, rather than the existing DOI's.

A second example is the widely used AE9/AP9/SPM package. The executable for the code is publicly available, however the source code is controlled. Hence, this code, although an industry standard would not satisfy the "open source software" requirement, and if we were to adopt a requirement for all codes to be open source, and further, require a DOI reference, it would limit the capability of researchers to conduct any studies that utilize commercial, proprietary, or export-controlled codes. There are several other codes that fall under the same umbrella.

One suggestion to get around this is to have a hierarchy of recommendations for open source code, and suggestions for best practices in the case of code that isn't or can't be open sourced:

Case 1 - The ideal case. The code used is archived open source with a DOI and indexed by a service like code.nasa.gov or OpenAIRE. The code should be cited in the literature, where used, with links to the code, and the the correct reference for the release utilized in the study.

Case 2 - The code is made public, open source and under version control. There is no archive of the code (no DOI). In this case, the code should be acknowledged and referenced by a revision number (SVN) tag (Git) or another commit identifier. The paper describing the code (if available) should be cited. The authors should consult the developer for the appropriate reference method for the codes used. A paper should be produced by the code authors to describe the code in this case.

Case 3 - The code is commercial, proprietary, or otherwise controlled, and hence the source is closed. In this case, The publisher/developer and the version number should be used to reference the software (if available). Information on where and how to obtain the code should be provided, along with links to any published documentation.

Case 4 - The author of the paper developed the code, and would like to make it open source, but there are restrictions in place. The open source approval may be slow/lengthy, or not available for some researchers. Such requests for release can be denied for commercial or legal reasons, and in the case that a request is approved it may take 6 months to a year for the code to be released. At times this process can be even longer. In this case, the requirement should be that the code is described in sufficient detail so that the reader can reproduce the results from the text in the paper. A follow-up paper describing the open source code should be made available.

For this reason, it is suggested that DOI's be adopted, but not required. Codes should be archived somewhere, as described in the next section, and referenced as appropriate/suggested by the authors in each case.

The code developer should also be compensated for developing, documenting, and making available the open source code.

Ensuring Longevity:

Open source data policies have led to advancements of science through the generation of long term data, and continued research on no longer operational data sets which previously may have been lost as funding ran out to house them at their original PI institution. We expect that open source code will have a similar transformational impact. Having access to the code's used to process and analyze the older data sets will allow for new generations of researchers to understand the process historically used as well as use new processing and analysis techniques as

they become available, enabling new insights with the older data sets such as correct cross calibrations between missions.

Sharing mission software, even as small as read routines, can save a lot of time for the community. This has perhaps been best shown by the efforts of the THEMIS team with their very successful SPEDAS software. It is very common now for mission teams to provide a base set of programs which make use of already written functions in SPEDAS. This greatly enables the ability for non-mission scientists to use the data quickly and appropriately, and gives some control to the instrument teams to make sure that any finalized advancement in the code or data products are immediately available to all users.

The one downfall, which the THEMIS team has in many ways mitigated, is that the base software language used in SPEDAS is IDL which is very expensive. Smaller efforts have been made using Python, a free software. Many of the heliophysics python packages are available through Git, SVN, or other online repositories. This is fantastic, but it does make it harder for people not familiar with the different packages to find the ones relevant to their needs. We suggest that NASA provide a single version control repository, in much the same way that NASA provides a data repository. Having a one stop shop for finding coding packages would help get more people using and contributing to the open source code, and promote community interaction.

Source control packages/repositories such as Github, SourceForge do not provide DOIs and they aren't archival. So while we recommend they are used for development and release of codes, they should not be used for archival/reproducibility of published results. The best practice in this case would be to have an archived release (e.g. Zenodo) as well as a paper about the software package/release associated with the research results.

Funding

One of the potential pitfalls is the amount of time that it can take to convert code that is run on a personal computer into something that will be usable for the community. It also requires that the code be well documented and tested. We see a few potential avenues to provide funding for code to be made to be open source. Perhaps the easiest would be to include these efforts in proposal work plans in much the same way that data management is currently included. For larger code packages, it would be useful to have a separate call, which would adequately cover the time and resources required to transition the codes to an open source format and platform. Currently there are year long calls for data to be processed and made publicly available, we suggest that there could be a similar set of calls for making code open source. Equally important, is that the funding should cover the publication costs associated with describing the open source code in sufficient detail.