Praxis of Reproducible Computational Science

Lorena A. Barba[1]

[1]George Washington University

November 13, 2018

Abstract

Among the top challenges of reproducible computational science are: (1) creation, curation, usage and publication of research software; (2) acceptance, adoption and standardization of open-science practices; (3) misalignment with academic incentive structures and institutional processes for career progression. I will address here mainly the first two, proposing a praxis of reproducible computational science.

The importance of software for research

At the 2015 SIAM Conference on Computational Science and Engineering (CSE), I was invited to speak at a panel titled "The Future of CSE as a Discipline." My remarks there ended with one provocative statement: *We're not a discipline, until we value software.* Until we value it enough to share it, to demand it as part of our publication process. Until we recognize it as core instrument and method of our endeavor, and give viable careers within academia for scientific software developers. Until we seek quality assurance in software by using version control and testing, and we have training in place for the younger generation of computational and data scientists to produce reliable claims to new knowledge.

Software is far more important in research than common wisdom would have it. According to a 2014 survey of researchers in the UK (417 respondents), 92% of academics use research software, 69% say that their research would not be practical without it, and 56% develop their own software (Hettrick, 2014). In a similar survey targeting members of the US National Postdoctoral Association, 95% of the 209 respondents said they use research software, and 63% stated that it would not be practical to conduct their work without software (Nangia and Katz, 2017). The fact is that, today, software written specifically for research purposes is ubiquitous—in data acquisition (from instruments), data analysis, data visualization, modeling and simulation, transforming or "cleaning" data, and automating various digital tasks. Like academic writing, researchers' professional development ought to include training in research computing.

As I narrated in my short piece in *Science*, "A Hard Road to Reproducibility" (Barba, 2016a), I struggled as a doctoral student when working with code left over by another student in our lab. I received no training in programming, and likely neither did the previous student, although his ability was surely superior to mine. I was and still am a second-rate programmer. But I have come to understand that a key challenge to raise the standards of reproducibility in science rests on how we develop, curate, use and publish research software.

Open-source software

Reproducible research is vitally connected to open-source software, open data and open science. Claerbout and Karrenbach (1992), pioneers of the reproducible-research movement, advocated for the merging of a research publication with the underlying computational analysis, and proposed that the ideal in this context is to use a public license that allows others to use, copy and redistribute the software.

> **Standard public licenses** are the popular and widespread method for sharing code. They are necessary because software and its underlying source code are protected by copyright, automatically, upon creation. If you don't attach a license to source code that you post on a hosting site (such as Bitbucket, GitHub or GitLab), readers must assume it is "all rights reserved," and they cannot do anything with it. Anyone who writes code, or is interested in using code written by others, needs to know a few things about open-source licenses. For a quick overview, click through the slides of "A short lecture on Open Licensing" (Barba, 2017); also recommended is the article by Morin, Urban, and Sliz (2012).
>
> **Free and open-source software (FOSS)** is under a license that grants the users freedom: to access, use or modify the software for their purposes. The most important distinction between the various FOSS licenses is whether they are permissive or copyleft. These terms are often confused. A permissive license gives more freedoms: the only restriction of use is that the original authors receive credit in any distribution of the software or any derivative works. Copyleft licenses (like GPL) require that any derivative works be under the same license (a.k.a., "share-alike"), and thus they are more restrictive. Academic software benefits most from permissive licenses, which originated at academic institutions, including the Berkeley Software Distribution or BSD License, the MIT License and the Apache License. As a researcher, you benefit from the widest dissemination to more users—including commercial users, who in general are averse to copyleft. I recommend the BSD-3 clause license for all research software (`https://opensource.org/licenses/BSD-3-Clause`).

The term open-source software (OSS) should be applied only to software released under a public license, and generally only one approved by the Open Source Initiative (OSI, `https://opensource.org`). Be aware that just because some source code is available on a website doesn't mean that the software is open source! It must have a license to be OSS. In addition to a licensing model, however, open source is also a *development model*. The crux of this model is expressed by "Linus' Law" (for Linus Torvalds, the creator of Linux): "Given enough eyeballs, all bugs are shallow." Taking advantage of the Internet for distributed collaboration, and cultivating a community of users and co-developers, the quality of the software will persistently improve. This perspective is altogether consistent with the ideals of scientific practice, and the quest for truth. As researchers committed to open source software, we not only reap the benefits of transparency, reproducibility and the ability to build from each other's computational work. We also learn to adopt collaboration practices that lead to higher-quality results.

An essential technology in collaborative code development is version control: a system to automatically record and manage changing versions of source code stored as text. It preserves a complete history of changes to a text file, it identifies who made the changes and when, and it allows one to roll back to a previous version. Because version control works on raw text, you can use it with other types of files, not just source code. In our research group, we use it when writing internal preliminary reports (using Markdown, or Jupyter notebooks), and for all our manuscripts (using LaTeX). Our system of choice to version-control files is `git` (a program written by Linus Torvalds), which we combine with GitHub for hosting and coordination functions. GitHub not only hosts your files (source code and other content), it helps coordinate your team

and your interactions with readers and users of your research artifacts. The issue tracker allows anyone with a GitHub account to leave a message, pointing to a bug or suggesting an improvement. You and others can post responses, and link seamlessly to places in the files where you've made changes that relate to the issue. You can make checklists and add labels to organize your work-in-progress; and when a conversation is over, you can close the issue. Over the last few years, we began using the issue tracker to organize our responses to peer review on our manuscripts, which themselves are written publicly in a GitHub repository!

GitHub is a tool-of-the-trade in the open-source world that supports its workflow, and promotes a culture of collaboration. It not only reflects the culture of open source, it brilliantly delivered structure to it. The exemplar of this is the *pull request*: a vital coordination to work in the open model. In a pull request, a user or another third party wants the owner of a project to perform some action, like making a change to a piece of code. They start by making a copy of the origin repository to their own GitHub account (via a "fork"), they make the changes in their local copy, and open a pull request with the origin. The owner can decline the request—and that's the end of it—, they can accept it and merge the changes within the original repository, or they can request changes before merging. In the process of reviewing the contribution, the project owner can start a conversation around code, via comments to the pull request. We use this pattern to coordinate all code development in our group: one graduate student making changes to an in-house code base makes a pull request, and another member of the group must review it (and sometimes a discussion ensues) before the changes are merged. Adopting this practice makes us more effective as a team and as a result our work is of higher quality and more reproducible.

Open data and open science

In reproducible computational research, "all details of the computations — code and data — are made conveniently available to others" (Donoho et al., 2009). Strictly speaking, a promise in a published paper to make code and/or data "available upon request" is *not* a reproducible practice: digital artifacts should already be in a suitable repository. For code, we have the framework of open-source software, with its licensing and tooling ecosystem. The case of data is, in many ways, different. Raw data are considered by US law as facts, and not protected by copyright. But the manner of "selection, coordination or arrangement" of data can be considered an original work, and copyright applies. Principles and practices for open data are better established than is the case for code, with several funding agencies now mandating a Data Management Plan as part of grant proposals.

Good data management is a basic responsibility of all researchers, but it remained a fuzzy concept until recently. Through the coordinated work of representatives from interested communities (researchers, publishers, funders, archivists, and more), we now have the *FAIR Principles*: digital artifacts of research should be Findable, Accessible, Interoperable and Reusable (FAIR, `https://www.force11.org/group/fairgroup/fairprinciples`) for machines and for people (Wilkinson et al., 2016). These high-level principles guide technology choices, practices, and standards. For example, the Findable principle leads to the practice of describing data with metadata, assigning to it a unique and persistent identifier, and registering the data in a searchable index. To be Accessible, data should be retrievable at any time using its unique identifier. By being Interoperable, data can integrate with other data and with standard workflows, which means using a shared language in metadata. The Reusable principle implies that data products should be released with a standard reuse license—typically a Creative Commons Attribution license (CC-BY, `https://creativecommons.org/licenses/by/4.0/`) or a Creative Commons public-domain dedication (CC0, `https://creativecommons.org/publicdomain/zero/1.0/`). Of course, exceptions to sharing data must be made for research involving human subjects, when privacy concerns take precedent.

Data repositories: when choosing an archival repository, the minimum requirements are that it provide a unique global identifier for your data (typically a digital object identifier, DOI), and that it offer long-term preservation guarantees (at least 10 years). Example repositories satisfying these criteria include: Figshare, Zenodo, DataVerse, and Dryad. It's important to note that GitHub is not adequate as an archival repository: it provides no guarantee that the artifacts will be preserved, as a project owner can at any time delete a code repository on GitHub.

Digital object identifier, DOI (https://www.doi.org), is a unique string assigned to a digital object by a registration agency, identifying the object and providing a persistent link to it on the Internet. You should deposit your data (and code releases) in a site that assigns a DOI (or an equivalent), and if a DOI is available for an item you cite, you should always include it in the reference list.

Figshare (http://figshare.com/) is a general-purpose repository for all kinds of digital artifacts of research. Any file format can be uploaded, up to 5GB in size. It is free and unlimited for public items, and offers private space limited to 20GB. We use it to deposit presentation slides, research figures, posters, course syllabi, lecture notes, and reproducibility packages to accompany our papers. You retain copyright on all deposited artifacts, and release them under the license of your choice. You can connect your GitHub account and import directly from your repositories there.

Zenodo (https://zenodo.org) is a data repository created by CERN and the European open-access infrastructure project called OpenAIRE. It is free and non-commercial. You can log in with your ORCID, and can deposit large files—up to 50GB by default, but you can request to deposit larger ones. We use it to deposit larger research datasets, and also to archive a full code base from its GitHub repository, to get a DOI for the code at the time of a release or publication. Our lab group has a Zenodo Community, where we collect our joint artifacts: https://zenodo.org/communities/barbagroup/

In comparison with open-source software, open-access publishing has a more tenuous relationship with reproducibility. The Yale Law School Roundtable on Data and Code Sharing (CiSE, 2010) included a recommendation to publish under open-access conditions (or post pre-prints), without making an explicit argument for this inclusion. Our practice is to always post a pre-print on arXiv for our research manuscripts, at the time of submission to a journal (or before). This makes the work available during the sometimes months-long peer-review process. In physics, mathematics, astronomy, and computer science, arXiv is a way of life! We consider having a friendly pre-print policy a basic criterion to choose a journal. Most journals accept manuscripts previously posted to preprint servers, including almost all journals by Elsevier and Nature Publishing Group, most in Springer and Taylor & Francis, and the majority of professional society journals, including 85% in the American Chemical Society (ACS)—chemistry being historically one of the most hard-nosed communities towards preprints, they've now founded their own preprint server: ChemRxiv. We've seen a semi-explosion of 'Xiv sites in the last few years, with engrXiv, bioRxiv, socArxiv, PeerJ Preprints, Authorea Preprints, and OSF Preprints (from the Open Science Framework). IEEE policy says that "Prior to submission to an IEEE publication: Authors may post their article anywhere at any time, including on preprint servers such as arXiv.org. This does not count as a prior publication."

Besides posting pre-prints, our signature open-science practice is to prepare what we call *reproducibility packages*, or "repro-packs," in short. The concept was part of my "Reproducibility PI Manifesto" (Barba, 2012). For the main results in a research manuscript, we share a file bundle of data, plotting script & figure under a CC-BY license. We deposit the file bundle as a Figshare object and get a DOI, then cite this DOI in the caption of the same figure in the manuscript. Ostensibly, this was a measure to increase

the reproducibility of our results: we also add a "reproducibility statement" in the paper, explaining that the figures, plotting scripts and data were shared just for this purpose. But in conference talks and hallway conversations, I often stressed the fact that this strategy also means that we are asserting the copyright on the figures, releasing them under CC-BY, and reusing them in our own paper under the terms of this license. That leaves the figures open for future reuse by ourselves and others under the license terms, without the need to ask permission from the journal (even if copyright of the paper was transferred to them for publication).

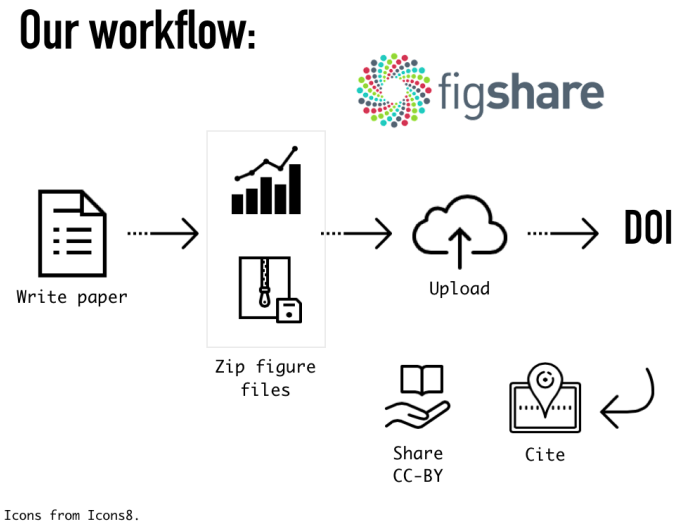

Figure 1: Sketch of our workflow for making "repro-packs" for each figure in a research manuscript.

Publishing research software

Many savvy open scholars are working to slash the hurdles for researchers to receive academic credit for all their output, including software and data. The *Journal of Open Source Software* (JOSS, http://joss.theoj.org) is a developer friendly journal for publishing papers about research software. On the face of it, writing papers about software is a weird thing to do, especially if the code is in a public software repository, with documentation and perhaps even a website for users of the software. But writing papers about software is currently the only sure way for authors to gain career credit, as it creates a citable object (a paper) that can be referenced by other authors, and the citations counted by indexing services. The primary purpose of a JOSS paper is to enable citation credit to be given to authors of research software. But its peer-review process, committed editorial board, and reviewers experienced in building and reviewing research software, lead to improving the quality of the submitted software. The JOSS publication process rests on existing infrastructure: the journal lives on GitHub and uses the issue tracker for open peer review; the post-peer-review software archive is deposited in Zenodo or Figshare for archival and a DOI, ORCID provides author identification, a CrossRef membership enables minting DOIs for the published papers, and a custom web app and Ruby bot (all open source) automate many editorial tasks. JOSS is an affiliate of the Open Source Initiative, and it is a fiscally sponsored project of NumFOCUS, a 501(c)3 nonprofit in the US—also home to NumPy, SciPy, Jupyter, Julia, Fenics and several other open-source projects for science. I am both a founding member of the JOSS Editorial Board, and a member of the Board of Directors of NumFOCUS.

How to run a research project for reproducibility

You might be asking, gentle reader: how to adopt this praxis of reproducibility in your project, or in your lab? Start by making a public commitment to reproducible research—what this means for you could differ from others, but an essential core is common to all. Transparency is an all-important value, and embracing open science is the best route to realize it. Onboarding every lab member with a deliberate group "syllabus" for reproducibility sets the expectations high. Compile a list of must-read literature on reproducible research; I can share mine with you: my lab members helped to make it (Barba, 2016b). For collaborating efficiently and building community, take inspiration from the open-source world, adopt its technology platforms to work on software and to communicate, openly and collaboratively. Key to the open-source culture is to give credit—give lots of credit for every contribution: code, documentation, tests, issue reports! The tools and methods require training, but running a project or lab for reproducibility is your decision. Start taking leadership in building up the ethical commitments in your research group, and your professional communities. Then take chances doing more, raising your standards of quality in research software, data curation, and open science.

> **Bonus advice:** Write into your grant proposals that your research software will be released under an OSI-approved license. If you're lucky to have your grant funded, the condition of open-source code release becomes part of the funding contract. This can save you some grief in trying to explain to staff at the university technology transfer office why your software needs to be open source. They often don't understand the field, and want to default to a proprietary license, imagining some commercial value may exist in your research code. Default to open!

References

L. A. Barba. The hard road to reproducibility. *Science*, 354(6308):142–142, oct 2016a. doi: 10.1126/science.354.6308.142. URL https://doi.org/10.1126%2Fscience.354.6308.142.

Lorena A. Barba. Reproducibility PI Manifesto. https://doi.org/10.6084/m9.figshare.104539.v1, 2012. URL https://figshare.com/articles/Reproducibility_PI_Manifesto/104539. Accessed on Thu, October 11, 2018.

Lorena A. Barba. Barba-group reproducibility syllabus – Hacker Noon. Blog post on Medium, archived on Figshare, 2016b. URL https://hackernoon.com/barba-group-reproducibility-syllabus-e3757ee635cf. Accessed on Thu, October 11, 2018.

Lorena A. Barba. A short lecture on Open Licensing. https://doi.org/10.6084/m9.figshare.4516892.v1, 2017. URL https://figshare.com/articles/A_short_lecture_on_Open_Licensing/4516892. Accessed on Thu, October 11, 2018.

CiSE. Reproducible Research. *Computing in Science & Engineering*, 12(5):8–13, sep 2010. doi: 10.1109/mcse.2010.113. URL https://doi.org/10.1109%2Fmcse.2010.113.

Jon F. Claerbout and Martin Karrenbach. Electronic documents give reproducible research a new meaning. In *SEG Technical Program Expanded Abstracts 1992*. Society of Exploration Geophysicists, jan 1992. doi: 10.1190/1.1822162. URL https://doi.org/10.1190%2F1.1822162.

David L. Donoho, Arian Maleki, Inam Ur Rahman, Morteza Shahram, and Victoria Stodden. Reproducible Research in Computational Harmonic Analysis. *Computing in Science & Engineering*, 11(1):8–18, jan 2009. doi: 10.1109/mcse.2009.15. URL https://doi.org/10.1109%2Fmcse.2009.15.

S. J. Hettrick. UK Research Software Survey 2014. https://doi.org/10.5281/zenodo.14809, 2014. URL https://zenodo.org/record/14809. Accessed on Thu, October 11, 2018.

Andrew Morin, Jennifer Urban, and Piotr Sliz. A Quick Guide to Software Licensing for the Scientist-Programmer. *PLoS Computational Biology*, 8(7):e1002598, jul 2012. doi: 10.1371/journal.pcbi.1002598. URL https://doi.org/10.1371%2Fjournal.pcbi.1002598.

U. Nangia and D. S. Katz. Track 1 Paper: Surveying the U.S. National Postdoctoral Association Regarding Software Use and Training in Research. https://doi.org/10.6084/m9.figshare.5328442, 2017. URL https://figshare.com/articles/Track_1_Paper_Surveying_the_U_S_National_Postdoctoral_Association_Regarding_Software_Use_and_Training_in_Research/5328442. Accessed on Thu, October 11, 2018.

Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C 't Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3:160018, mar 2016. doi: 10.1038/sdata.2016.18. URL https://doi.org/10.1038%2Fsdata.2016.18.