A Model For Peer Review And Onboarding Research Software

Karthik Ram & Scott Chamberlain
rOpenSci and University of California, Berkeley
Noam Ross
EcoHealth Alliance

28 April, 2017

Abstract

Code review, in which peers manually inspect the source code of software written by others, is widely recognized as one of the best tools for improving reliability and finding bugs in software. While this is a standard practice in industry, formal code review is relatively uncommon in *scientific* software development, though. Scientists, despite being familiar with the process of peer review, often have little exposure to code review due to lack of training and historically little incentive to share the source code from their research. So scientific code, from one-off scripts to reusable R packages, is rarely subject to review. Most R packages are subject only to the automated checks required by package managers like CRAN, which primarily ensure that packages can be installed on multiple systems. As such, The burden is on software users to discern well-written and efficient packages from poorly written ones.

rOpenSci is a community of developer-scientists, creating R packages by and for scientists, and our package contributors come from a variety of backgrounds. We aim to serve our users with high-quality software, and also promote best practices among our author base and in the scientific community in general. In this paper we describe our onboarding process and its evolution since we began the pilot in March 2015, along with some lessons learned and our plans for the future.

Peer-review of research software

- Peer-review has been critical for scientific progress. Independent oversight tries to ensure that surious results do not get published, and that at least one or two experts and an editor make an informed decision on a paper.
- Although papers are often thoroughly vetted, the same level of scrunity isn't given to research software or other supplementary material. Most often, the code behind a paper is not shared with reviewers nor is is shared upon publication. The reviewers are expected to just review without further insight into the code. Even when given access to the code, these are rarely reviewed.
- There have long been calls for researchers to simply share their code regardless of quality (Barnes, 2010). However, with the increasing emphasis on software, popularization of data science, and growth in skills from workshops like software and data carpentry, more researchers are releasing their work as software packages.
- There is value to having software reviewed with some detailed criteria. Before creating a more widespread model, we decided to pilot an effort to formalize peer review of software.
- rOpenSci's mission is to develop software that makes is easier for researchers to make their work more reproducible. Instead of every single researcher writing their own bespoke solution towards a data access problem, we try to make that general.

- In this paper we describe our efforts to create a software peer-review framework and share some details from the initial year.

The process

rOpenSci's package review process owes much to the experiments of others (such as Marian Petre (http://mcs.open.ac.uk/mp8/) and the Mozilla Science Lab (https://mozillascience.org/code-review-for-science-what-we-learned)), as well as the active feedback (https://discuss.ropensci.org/t/code-review-onboarding-milestones/180) from our community (https://discuss.ropensci.org/t/how-could-the-onboarding-package-review-process-be-even-better/302).

Here's how it works: When an author submits a package, our editors evaluate it for fit according to our criteria (https://github.com/ropensci/policies#package-fit), then assign reviewers who evaluate the package for usability, quality, and style based on our guidelines (https://github.com/ropensci/packaging_guide#ropensci-packaging-guide). After the reviewers evaluate and the author makes recommended changes, the package gets the rOpenSci stamp in its README and is added to our collection.

We work entirely through the GitHub issue system. To submit authors open an issue (https://github.com/ropensci/onboarding/issues/new). Reviewers post reviews as comments on that issue. This means the entire process is open and public from the start. Reviewers and authors are known to each other and free to communicate directly in the issue thread. GitHub-based reviews have some other nice features: reviewers can publicly consult others by tagging if outside expertise is wanted. Reviewers can also contribute to the package directly via a pull request when this is more efficient than describing the changes they suggest.

This system deliberately combines elements of traditional academic peer review (external peers), with practices from open-source software review. One design goal was to keep reviews non-adversarial - to focus on improving software quality rather than judging the package or authors. We think the openness of the process has something to do with this, as reviews are public and this incentivizes reviewers to do good work and abide by our code of conduct (https://github.com/ropensci/policies#code-of-conduct). We also do not explicitly reject packages, except for turning some away prior review when they are out-of-scope. We do this because submitted packages are already public and open-source, so "time to publication" has not been a concern. Packages that require significant revisions can just remain on hold until authors incorporate such changes and update the discussion thread.

- Back in the early days of rOpenSci, most of the packages that comprised our suite was developed in house. As we did more outreach about the importance of reproducible research, we got more and more enquiries about people wanting to transfer their packages to us.
- We were dealing with a huge diversity in quality and decided to formalize the process. This allowed us to do quality checks, promote the use of best practices, and ensure that our brand would attest to higher than average quality of software.
- Our initial list of requirements was inspired by The Joel Test for software `knitcitations::citep("")`.
- Beyond that checklist, we wanted experienced R developers and R users to install the package and review all of the functionality being exposed. As most authors work in isolation, having a second pair of eyes really helped.
- Packages that are accepted into the rOpenSci suite end up being higher quality because we went beyond the basic CRAN checks and included CI, code coverage, more complete tests, consistent style, and usability.
- We accept packages that fit our mission. That is, to promote reproducibility in all areas of science. So any package that is submitted to our suite must fall under one of these categories.

Packages onboarded so far

Since March 2015 the onboarding effort has accepted 35 packages into the rOpenSci suite. These packages were contributed by 24 unique authors. The packages represent X fields. Only 5 submissions were rejected as

out of scope. Our packages thus far have fallen into the categories of Climate data, data access, earth science, reproducibility, data-extraction, geospatial tools, text-mining, literature and databases.

Lessons learned and future directions

In DATE Noam Ross also recently surveyed (https://docs.google.com/spreadsheets/d/1zaE5MvqXyD0I7LWONh1HlQu98wTIZ0
edit?usp=sharing) our reviewers and reviewees, asking them how long it took to review, their positive and negative experiences with the system, and what they learned from the process.

1. _Reviews are not quick, but on par with traditional peer-review___. Software review takes the same time. While there appear to be no comprehensive studies that have tracked the amount of time researchers spend reviewing manuscripts, at least one paper reports a median time spent reviewing an article at about 5 hours (Ware, 2011). Package reviews take a similar amount of time and more experienced developers were not necessarily faster in their review times.

We asked reviewers to estimate how much time each review took, and here's what they reported: Answers varied from 1-10 hours with an average of 4. This is comparable to how long it takes researchers to review scholarly papers (http://publishingresearchconsortium.com/index.php/112-prc-projects/research-reports/
peer-review-in-scholarly-journals-research-report/142-peer-review-in-scholarly-journals-perspective-of-the-scholarly-communit
but it's still a lot of time, and does not include further time corresponding with the authors or re-reviewing an updated package.

2. _Reviewers find the process highly rewarding___ Pretty much everyone who responded to the survey, which was most of our reviewers, found value in the system. While we didn't ask anyone to rate the system or quantify their satisfaction, the length of answers to "What was the best thing about the software review process?" and the number of superlatives and exclamation marks indicates a fair bit of enthusiasm. Here are a couple of choice quotes:

"I don't really see myself writing another serious package without having it go through code review."

"I learnt that code review is the best thing that can ever happen to your package!"

Authors appreciated that their reviews were thorough, that they were able to converse with (nice) reviewers, and that they picked up best practices from other experienced authors. Reviewers also praised the ability to converse directly with author, expand their community of colleagues and learn about new and best practices from other authors.

Interestingly, no one mentioned the credential of an rOpenSci "badge" as a positive aspect of review. While the badge may be a motivating factor, it seems from the responses that authors primarily value the feedback itself. There has been some argument (http://simplystatistics.org/2013/09/26/
how-could-code-review-discourage-code-disclosure-reviewers-with-motivation/) whether code review will encourage or discourage scientists to publish their code. While our package authors represent a specific subset of scientists - those knowledgeable and motivated enough to create and disseminate packages - we think our pilot shows that a well-designed review process can be encouraging.

3. **The review process feels ambiguous** With traditional peer-review, there is some cultural knowledge passed down from mentor to mentee. Graduate students often shadow review papers that are assigned to their advisors and examples of good and bad reviews are sometimes shared. With package reviews, there hasn't been enough of it for a culture to develop.

A few respondents pointed out we could be better at explaining the review process, both in how to get started and how it is supposed to wrap up. For the former, we've recently updated our reviewer guide (https://github.com/ropensci/onboarding/wiki/For-Reviewers), including adding links to previous reviews. We hope as our reviewer pool gets more experienced, and as software reviews become more common, this gets easier. However, as our pool of editors and reviewers grows, we'll need to ensure that our communication is clear.

As for the end of the review, this can be an area of considerable ambiguity. There's a clear endpoint when a package is accepted, but with no "rejections" some reviewers weren't sure how to respond if authors didn't follow up on their comments. We realize it can be demotivating to reviewers if their suggestions aren't acted upon. (One reviewer pointed out that seeing her suggestions implemented as a positive motivator.) It may be worthwhile to enforce a deadline for package authors to respond.

Future directions

4. _Scaling and reviewer incentives___ Like academic paper review or contributing to free open-source projects, our package review is a volunteer activity. How do we build an experienced reviewer base, maintain enthusiasm, and avoid overburdening our reviewers? We will need to expand our reviewer pool in order to spread the load. As such, we are moving to a system of multiple "handling editors" to assign and keep up with reviews. Hopefully we will be able to bring in more reviewers through their networks.

5. **Improving author incentives** Our small pool of early adopters indicated that they valued the review process itself, but will this be enough incentive to draw more package authors to do the extra work it takes? An area to explore is finding ways to help package authors gain greater visibility and credit for their work after their packages pass review. This could take the form of "badges", such as those being developed by The Center for Open Science (https://osf.io/tvyxz/wiki/home/) and Mozilla Science Lab (https://www.mozillascience.org/projects/contributorship-badges), or providing an easier route to publishing software papers.
6. **Improving automation around the review process** How can we automate more parts of the review process so as to get more value out of reviewer and reviewees time? One suggestion has been to submit packages as pull requests (https://discuss.ropensci.org/t/how-%20could-the-onboarding-package-review-process-be-even-better/302/3) to take more advantage of GitHub review features such as in-line commenting. This may allow us to move the burden of setting up continuous integration and testing away from the authors and onto our own pipeline, and allow us to add rOpenSci-specific tests. We've also started using automated reminders](https://github.com/ropenscilabs/heythere) to keep up with reviewers, which reduces the burden on our editors to keep up with everyone.

Acknowledgements

Research reported in this publication was supported in whole or in part by The Leona M. and Harry B. Helmsley Charitable Trust under award number 2016PG-BRI004 (KR and SC). TODO: Noam who supported you? We also thank the various authors, reviewers, and editors who have generously contributed their time and expertise to the review process.

Literature cited

[1] N. Barnes. "Publish your computer code: it is good enough". In: *Nature* 467.7317 (Oct. 2010), pp. 753-753. DOI: 10.1038/467753a. <URL: https://doi.org/10.1038%2F467753a}.>

[2] M. Ware. "Peer Review: Recent Experience and Future Directions". In: *New Review of Information Networking* 16.1 (May. 2011), pp. 23-53. DOI: 10.1080/13614576.2011.566812. <URL: https://doi.org/10.1080%2F13614576.2011.566812}.>

Notes

Scientists are coding more and more these days. With new coding experience, scientists are getting better are producing their own software Yet this is often an afterthough and not a first class citizen.

We have peer review for papers but we don't have such a thing for software, outside of code review that happens with teams.

The rOpenSci project was founded with the mission of improving the quality of software used in data analysis for science. So our goal from the beginning was to write high quality software for people to use. Initially, much of the software was written by our team. Now a large number of contributions come from others. Over the years we have started to formalize the process.

inspirations from The Joel Test: https://www.joelonsoftware.com/2000/08/09/the-joel-test-12-steps-to-better-code/

paper on code review: http://www.nature.com/news/mozilla-plan-seeks-to-debug-scientific-code-1.13812

practicing code review: https://alexgaynor.net/2013/sep/26/effective-code-review/

How is what we do different than code review? https://alexgaynor.net/2013/sep/26/effective-code-review/

http://ivory.idyll.org/blog/on-code-review-of-scientific-code.html

Carl: http://www.carlboettiger.info/2013/06/13/what-I-look-for-in-software-papers.html

publish your code, even if it's bad: http://www.nature.com/news/2010/101013/full/467753a.html

So on May 2016, we began piloting a project called rOpenSci onboarding. That projects with a broad appeal towards being useful in the data sicence space and potentially become part of the rOpenSci Suite, go through the onboarding process.

rOpenSci's software onboarding process

Lessons learned

Go over same stuff from the blog post Include time it takes for peer-reviewing papers https://scientistseessquirrel.wordpress.com/2015/08/31/how-long-should-peer-review-take/

http://sciblogs.co.nz/code-for-life/2013/09/27/how-long-do-you-take-to-review-a-research-paper/

https://twitter.com/RKPriestley/status/383397036751470592

http://www.aje.com/en/arc/peer-review-process-15-million-hours-lost-time/

http://www.tandfonline.com/doi/abs/10.1080/13614576.2011.566812

says: 3.5 hours spent on reviewing scientific papers.

Better code review:

https://medium.com/@mrjoelkemp/giving-better-code-reviews-16109e0fdd36

Mozilla science lab results:

https://science.mozilla.org/blog/code-review-for-science-what-we-learned

Notes from the reviews:

https://github.com/ropensci/onboarding/issues/82 > For me this has been the first time I have reviewed an R package and I have learned a lot and enjoyed it, so thanks to all of you! I hope to have the opportunity to review more packages in the future.

> Nah. . . , I am the one indebted for getting all that great feadback - thanks for letting me join the club.

Do average time for a review to be done.