Some thoughts on open-source software development *

Jamie Collins
School of Oceanography and eScience Institute
University of Washington

james.r.collins@aya.yale.edu

🐦, 🐙 @jamesrco

http://jamesrco.github.io

* Presumes you intend to develop something for public consumption at project outset, but most advice useful in retroactive scenario as well

Thanks

- Members of Data Stewardship and Scientific Software clusters at ESIP (Federation of Earth Science Information Partners): http://esipfed.org/
- eScience colleagues
- WHOI Ocean Informatics Initiative, particularly Stace Beaulieu: http://www.whoi.edu/DoR/special-projects/ocean-informatics-working-group
- Cara Manning (UBC, formerly WHOI)

Perspective

- Oceanography & the geosciences
- Mostly R (own recent experience developing LOBSTAHS package)
    - https://github.com/vanmooylipidomics/LOBSTAHS
    - http://bioconductor.org/packages/LOBSTAHS
- Some advice specific to R development, but almost all the advice here applies to package/library development in Python as well

Experience & motivation?

Informal survey of eScience students and postdocs *

Descriptor	No. responses
Have you developed formally packaged open-source software?	5
Contributed to a larger, ongoing open-source project?	4
If one of the above, what language?	
Python	1
R	3
C	1
Are you a package or library maintainer?	2
Project genesis:	
Direct outgrowth of research project?	3
Or a side project?	2
Reason for developing?	Multiple responses, including (1) desire to make performing repetitive easier, (2) desire to share tool with broader research community
Level of training in best practices:	
None/self-taught	8
Published journal articles	1
Classroom or other formal training	1

* Survey of those attending the eScience student & postdoc lunch on 16 January 2018 (N = 10)

At the beginning…

- Ask yourself up front:
  - Will a loose collection of scripts suffice?
  - Do I really need/want to develop a package/library?
  - Target audience (size and type) instructive here

If a package it is…

- Again, think about audience
- Version control critical (obvious to this audience)
    - Facilitates collaboration
    - Makes it easy to walk back mistakes & float new features without breaking everything

If a package it is…

- Code hygiene will be important; don’t wait until later to clean up your mess

- **Documentation** may be most time-consuming component, so start early

    - Build docs/manual pages for functions as you go

    - Make them useful (don’t do the minimum only to spoof a package checker)

    - Good documentation will make your software more appealing and useful

As you work toward the goal

- Unit tests (make lots of them, use lots of them)
- Trial and error (on your development branch, of course)
- The #opensource, #openaccess community is ready to help; find a good listserv or forum and ask away
    - …but pay it forward and assist in the future when you're the one who knows the answer
- If in R: Run R CMD check, R CMD BiocCheck early & often

Before first official release

- Choose the right license (depends on objective): https://choosealicense.com/
    - Patent or other IP implications?
- Use testing?

Once it’s out there

- Support your work & be willing to fix bugs (especially early on)

- What happens after you leave: Do you have a **software sustainability** plan?

    - Some repositories will sunset or mothball your package after years of inactivity, but it might still be useful to someone

More on **sustainability**

- Some resources:
  - Best practices from the software Sustainability Institute: https://www.software.ac.uk/blog/2017-11-29-best-practices-scientific-software
  - “Community recommendations” from some ESIP colleagues: https://openresearchsoftware.metajnl.com/articles/10.5334/jors.bt/

Once it’s out there

- Consider creating a companion data package containing a *validated* example/demonstration dataset that you understand
    - Helps users learn your software
- Promote: A published paper helps, but so does accession to a repository (CRAN or Bioconductor for R)
    - Social media?
- Get credit: Can use Zenodo to archive and obtain DOI for each release of your package; new solutions on horizon

Some general resources

- ESIP Software Guidelines, draft October 2016: https://esipfed.github.io/Software-Assessment-Guidelines/guidelines.html

- A few useful papers (credit C. Titus Brown, Stace Beaulieu):

  - http://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.1002303

  - https://arxiv.org/pdf/1609.00037v2.pdf

Some R resources

- https://hilaryparker.com/2014/04/29/writing-an-r-package-from-scratch/
- https://cran.r-project.org/doc/manuals/r-release/R-exts.html
- Bioconductor best practices (even if not submitting to Bioconductor):
  - https://www.bioconductor.org/developers/how-to/buildingPackagesForBioc/
  - https://www.bioconductor.org/developers/how-to/coding-style/
- Anything by Hadley Wickham, but particularly:
  - http://r-pkgs.had.co.nz/man.html (guidance on documentation)

Future initiatives

- Some activity afoot to develop a scientific software repository/directory for the geosciences
  - NSF EarthCube, ESIP
- Incorporating formal instruction in best practices into undegrad/early graduate level education