rOpenSci Community Call - Maintaining an R Package

This doc - bit.ly/ropensci-commcall-maintaining - is a place for collaborative note taking.
It includes:

- Agenda
- Attendees list. Please add yourself
- Questions from moderator
- Questions from attendees. All are encouraged to ask and answer in this doc.
- Call to action!
- Resources list. Add your favorites.

Information on today's topic, with speaker bios:
https://ropensci.org/blog/2020/03/04/commcall-mar2020/

This **call will be recorded** and posted along with any other resources within three business days at https://ropensci.org/commcalls.

Agenda

1. Stefanie Butland, rOpenSci Community Manager - welcome (3 min)
2. Julia Silge presentation (~10 min)
3. Panel Q & A with Elin Waring, Erin Grand, Leonardo Collado-Torres, Scott Chamberlain, moderated by Julia Silge, (40 min)

This is a unique session full of pre-selected questions for the panel. They will not be taking impromptu questions from attendees, but you all have expertise to share! I invite everyone to add your own questions by typing them below, and sharing your thoughts by answering others' questions. Together we can make this a rich resource for all of us.

In attendance (please add yourself here):

- *Name; Affiliation; Country*

1. Stefanie Butland; rOpenSci; Canada
2. Julia Silge; RStudio; USA
3. Scott Chamberlain; rOpenSci; USA
4. Leonardo Collado Torres; Lieber Institute for Brain Development; USA (here are my notes prior to the call https://gist.github.com/lcolladotor/602fc9ede91a45a27ac8cc8febd778de)

5. Erin Grand; Uncommon Schools; USA
6. Elin Waring; Lehman College CUNY; USA
7. Ciara Gribben; NHS NSS/Public Health Scotland; Scotland
8. Kim Cressman; Grand Bay National Estuarine Research Reserve; USA
9. Jasmine Lai; University of British Columbia; Canada
10. Tamas Koncz; Emarsys; Hungary
11. Sam Dunn; Public Health England; UK
12. Pierrette Lo; Oregon Health & Science University; USA
13. Kelsey Montgomery; Sage Bionetworks; USA
14. Sam Albers: BC Government Ministry of Citizens' Service
15. Matthias Grenié; University of Montpellier; France
16. Kelli Johnson; NWFSC NOAA NMFS; USA
17. Max Joseph; CU Boulder; USA
18. Lennert Schepers; Flanders Marine Institute (VLIZ); Belgium
19. Heidi Tebbe; North Carolina State University; USA
20. Janani Ravi; Michigan State University & R-Ladies East Lansing; USA
21. Alvaro Uzaheta; ETH Zurich; Switzerland
22. Daniel Chen; Virginia Tech (quarantined in NY); USA
23. Jesse Sadler; LMU; USA
24. Steffi LaZerte; rOpenSci Community Intern / R consultant; Canada
25. Kenneth Tay; Stanford University; USA
26. Daniel Sjoberg; MSKCC; NYC; New York
27. Nicole Kauer; Sage Bionetworks; USA
28. Jan Savinc; Edinburgh Napier University; Edinburgh, Scotland
29. Andy Teucher; British Columbia Government
30. Matt Dray; UK Government; UK
31. A. Márcia Barbosa; modtools.wordpress.com; Portugal
32. Mauro Lepore; 2 Degrees Investing Initiative; USA
33. Catherine Starnes; Belmont University; USA
34. Tamás Stirling; Budapest; Hungary
35. Stephany Orjuela; University of Zurich; Switzerland
36. Rainer Krug; University of Zurich; Switzerland
37. Georgina Anderson; Public Health England; UK
38. Mike Smith; EMBL; Heidelberg, Germany
39. Kristina Riemer; University of Arizona; USA
40. Marc Dotson; Brigham Young University; USA
41. Vratika Chaudhary; University of Florida; USA
42. Salar Khaleghzadegan; Johns Hopkins University; USA
43. Andrea Sánchez-Tapia - Rio de Janeiro Botanical Garden
44. Ellis Hughes; Fred Hutch Cancer Research Center; USA
45. Jeff Hollister; US EPA;, Narragansett, RI USA
46. Kathryn Doering; Caelum in support of NOAA Fisheries; Seattle, WA, USA
47. Mara Avierck; RStudio; United States of America

48. Sina Rüeger; University of Helsinki; Finland
49. Sharla Gelfand; Freelance #rstats; Toronto, Canada
50. Ian Flores Siaca; Jumping Rivers; Puerto Rico
51. Noam Ross; rOpenSci / EcoHealth Alliance; NYC, USA
52. Mark Padgham; rOpenSci; Münster, Germany
53. Aaron Wolen; TileDB; TN, USA
54. Maëlle Salmon; rOpenSci
55. Danica Lewis; North Carolina State University Libraries; USA
56. Karissa Whiting; MSKCC; New York, NY; USA
57. Hugo Gruson; University of Montpellier; France
58. Jim Tyhurst; Tyhurst Technology Group; Portland, Oregon, USA
59. Amanda Devine; Smithsonian Institution; Washington, DC, USA
60. Athanasia Monika Mowinckel; University of Oslo; Norway
61. Chris Umphlett; TechSmith; Lansing, Michigan, USA
62. Helen Miller; Fred Hutchinson Cancer Research Center; US
63. Sarah Dalrymple; Liverpool John Moores University; UK
64. Klaus Schliep; Graz University of Technology; Austria
65. Jeroen Ooms; rOpenSci
66. Sumedh Panchadhar; MSKCC; USA
67. Noushin Nabavi, BCGov, Canada
68. Vedha Viyash; ZoomRx; Chennai, India
69. Paul Oldham; One World Analytics; UK
70. Sara Stankiewicz; BWH; Boston, MA; USA
71. Gabor Kocsis; Emarsys; Hungary
72. David Patchett; HSBC; UK
73. Ana Laura Diedrichs; Universidad Tecnológica Nacional; Mendoza, Argentina
74. Christine Stawitz; ECS Federal in support of NOAA Fisheries; Seattle USA
75. Erik Sapper; Cal Poly; San Luis Obispo, CA, USA
76. Alyssa Columbus; Johns Hopkins University; USA
77. Ian Taylor; Northwest Fisheries Science Center; NOAA, Seattle USA

90 total participants at peak

Julia Silge's presentation

Slides: https://speakerdeck.com/juliasilge/maintaining-an-r-package

Long tails of small contributors -> many contributors contribute few commits

Changing the classical contribution model: Users -> Contributors -> Committers
And instead have various way of contributing and have different maintainers on different parts of the project (education material, docs, etc.) and have users swim around different subject

Tips to welcome contributions:

- Have a code of conduct
- Be kind and respectful
- Have a public roadmap so newcomers know where you plan on going and where contributions might be needed

Have legitimate forms of peripheral participation; make it easy for newcomers to get started & acknowledge all contributors

Questions from moderator to panelists

We encourage you to capture comments from panelists responding to these questions or add your own responses to these questions.

What does it mean to “maintain an R package”?

Scott : A constant learning process, always thinking how to make things better; constantly learning how to better design function interfaces
Erin: ownership around package’s community
Leo: User-driven Qs, changes, commits, anticipate changes, be up-to-date with CRAN and Bioconductor guidelines

How do you encourage user feedback and/or manage a firehose of user feedback or requests for help?

Elin: (based on skimr exp.) #howto Qs, suggestions, issue-reports, relationships with users. It can be useful to create a tag on stackoverflow to regroup all questions at a specific place. 5 or more similar Qs -- create a tag -- easy to keep track! Balancing how much to ask contributors to fix, especially style, and how a maintainer fixes.
Leo: actively switch users from emails to bioconductor with reprex -- easier than authors creating a blogpost!

How do you manage issues / feature requests? What workflows do you use to do this?

Scott: respond to all issues quickly to engage user interest; define scope clearly and communicate to users asap

What is a path for new contributors to R packages? How can healthy norms be passed on? (Related: What should someone do if they want to start helping maintain one of your packages? First step?)

Erin: Qs/comments/suggestions come via Slack -- onboard them quickly to the package repo and allow contributions to come in

Hugo: I love Julia's comment on having a public roadmap. That's actually how I did my very first contribution to an R package. I visited the list of issues of a package I was regularly using and noticed they had an unresolved issue about how to read a specific file format. Turns out I had the same problem a couple of months back and I knew how to deal with it. The package maintainers were kind enough to guide me through the structure of the project and what was the best way to integrate this new functionality. Fast forward a couple of months and I am now one of the maintainers of this package :)

What considerations go into decisions around dependency management? APIs that change?

Leo: chooses R packages that are more likely to be stable, e.g., tidyverse packages

What do you consider when other packages depend on your work?

What does the process of changing maintainers look like? What sets up this transition for success?

Scott: Be available (at least for a little while) for the new maintainer to get oriented.Be prepared to give up your control of the package. It's key for new maintainer to be familiar with the domain area of the package.

Erin: Have clear guidelines around what the package does, the style of the code, where questions re the package tend to come from.

We've talked about a lot of different facets of package management. Which are the same vs. different for internal packages?

How do you decide to submit to a centralized repository like CRAN or Bioconductor? Peer review like JOSS or rOpenSci? Stay only on GitHub?

Elin: Once you say "v1.0", you are giving a promise that it works.
Elin: Once you're on CRAN, you might start having other packages using you as a dependency. That creates a level of social obligation.

Leo: some more notes about submitting to Bioconductor at https://gist.github.com/lcolladotor/602fc9ede91a45a27ac8cc8febd778de

rOpenSci recommends in rOpenSci Packages: Development, Maintenance, and Peer Review: "We strongly suggest submitting your package for review before publishing on CRAN or submitting a software paper describing the package to a journal. Review feedback may result in major improvements and updates to your package, including renaming and breaking changes to functions."

What does someone need to know or skills they need to have to start maintaining a package?

Leo: regular communication; practice patience + empathy
Erin: communication + documentation
Elin: you need to like your package first! :)
Scott: Learn to write & use functions!

Questions Part B (from attendees).

This is a unique session full of pre-selected questions for the panel. They will not be taking impromptu questions from attendees, but you all have expertise to share! **Add your own questions** by typing them below, and share your thoughts by **answering others' questions**. Together we can make this a rich resource for all of us.

(Your Name) Your question.

(Question) Daniel Sjoberg

Something I have found to be increasingly frustrating is the minimum R version supported. I have a few packages in `Suggests:` I use in my unit tests only. For example, I want to ensure proper behavior of my package functions when using cubic splines from the {Hmisc} package. {Hmisc} depends on {latticeExtra}, which is a min R version requirement of 3.6. **What is the best practice for ensuring continued support for older R versions when dependencies of dependencies of dependencies keep upping the minimum version of R required (and in this case, for a dep I only use in unit tests)?** Currently using Travis CI to test on various versions of R. Thanks!

Klaus Schliep
The rhub package and https://github.com/r-hub offers a lot of infrastructure, also testing on different platforms (different Linux versions, Windows, OS X, Solaris, ...).

Rainer Krug:
An important aspect in developing is the consideration of which packages one depends on / imports, and which comes back in the maintenance. The less packages your own package depends on, the less affected it is to changes in other packages.

Klaus Schliep
A bigger problem is when a lot packages depend on your package to check all reverse dependencies

Rainer Krug:
True - but you could use the ggplot approach, and release a xxx2 package when there are severe breaking changes. But it is not a very nice approach.

Klaus: And you have to maintain 2 packages afterwards

(Question) Athanasia Monika Mowinckel

Part 1: Do any of you have a package that depends on software that is not from R, for instance another command line tool. I.e. your R package wraps system calls to the command line software and uses output from that in R.

Janani Ravi:
Yes, I do! My workflow involves a few shell scripts (bioinformatics) to start with before generating the dataframe that goes into R for subsequent analyses. Still figuring out how to go about this.
Also, using system2 without compromising speed and getting it to work across platforms seems to be critical to make it fully compatible with R -- thoughts anyone?

Athanasia Monika Mowinckel:
Yes particularly cross platform compatibility is very tricky. Some of the tools are specific for linux or macos, but wont work on windows. So how to make install work. For instance, only parts of the program will depend on the command line tool, while other critical functions do not and work fine on windows.

Mike Smith:
You can use a call to Sys.info() to identify what platform the code is running on. You could use that at the start of a function and only proceed if you find you're on a platform that function make sense to run on, and print a message to the user otherwise.

Klaus Schliep:
Doesn't fix the problem, but describing in a README file how to install the dependencies on different platforms can be very useful for users who look in there.

Janani Ravi:
Also, I have a similar Q for running a local C script (dependency). Added it to src to make sure installation is taken care, but to run it, I use system2, and feel this may not work with windows.

Athanasia Monika Mowinckel:
I run CI on appveyor, also on development branches, which helps me at least see when funcs are failing because the system can’t handle it. Though running CI with other command line tools is another thing I can’t seem to get working. Perhaps if I make a docker image with the software needed. But I’m not there yet.

Part 2: How do you improve user experience when the program needs environment variables R does not pick up with Sys.getenv()?

Scott Chamberlain:
I'd imagine a good error message saying what environment variable was looked for and not found, and give the docs manual page to go to for help

Mike Smith:
I'll echo this. If R can't find the environment variable there's very little you can do except catch that gracefully and inform the user. You can either direct them to your own documentation in the package vignette or that of the third-party tool depending on what's most appropriate.

Erin Grand:
I put in a check that looks for the variable, and if it can't find it, the user gets a note saying "Please add XXX to your env with usethis::edit_r_environ()"

Sharla's talk at RStudio this year had some helpful comments on how to write notes to your users.

Athanasia Monika Mowinckel:
Yes, I have been considering this approach. I think I'll give that a go, thanks Erin

(Question) Janani Ravi

Q similar to Athanasia's. I would also like to know how to incorporate a significant part of non-R scripts within the R package workflow -- functions that use system/system2 calls, for instance! Are there any systematic ways and good examples that could point to this?

Scott Chamberlain:
For these system calls, in case you haven't seen it, sys
https://cloud.r-project.org/web/packages/sys/ is a good option

Scott Chamberlain:
What language are you talking about?

Janani Ravi:
1. shell scripts that work on HPC
2. C/Py scripts but those could be run with reticulate or equivalent packages.

Scott Chamberlain:
I don't know best practices, i'll ask around

(Question) Lennert Schepers

Part 1: If I want to start with fixing an issue/contributing to a well-documented package, what are essential parts that I should know before fixing? **Should I learn package documentation *and* testing/awcontinuous integration *and* all other aspects of R packages**,... or are there parts that are necessary and others that are less urgent?

Erin Grand:
It depends on the issue. If it's an error in documentation, you should learn how to edit it - but probably don't need to know about CI. If you're adding or changing a full function, then there may be more complex parts of R packages to learn.

I found it helps to start in my own package as a place to learn.

Lennert Shepers:
Thanks!

Part 2: Are there things that should be absolutely avoided? (things that can break the whole package 😱)?

Scott Chamberlain:
Be careful with while loops

Mike Smith:
Hopefully your own testing will identify if a change totally breaks a package before pushing those changes to a released version. I'd definitely recommend including unit tests early on in package development; a comprehensive suite of tests can highlight when you change things inadvertently, and the earlier you include them the less onerous the task is!

Mike Smith:
I think the more scary situation is when your change breaks someone else's package or workflow. I think that falls under Elin's "social contract" to try and be nice to other developers and users.

Lennert Schepers:
So testing is highly recommended, but something that I still have to learn... thanks!

Klaus Schliep:
Having lots of working(!!!) examples in the man pages (*.Rd files) is often as good or even better than unit tests. These also gets checked and the users have more documentation. I always type example(function_name) for the main functions if I explore a new package and see what happens.

Athanasia Monika Mowinckel:
Not all functions can have proper running examples, for instance working towards an API or outputs that should be evaluated not just return something and therefore pass.

So unit tests are superior in testing and making sure the package functions have expected outputs (and error messages etc). While obviously examples are great for users and should exists and be good for their sake.

Tests built with testthat for instance also run on build checks, and will error when assumptions are not met, and therefore also give you a good idea of what is breaking, where and why.

Klaus Schliep:
I still do think that examples can also function like a small test. When I look into a new package I always first try to run some of the example. Also most R user are not native English speakers so I decided for myself the more examples the better. And some developer seem to use unit tests to get a nice code coverage badge, not to make the API more robust. But still better than no tests I assume.

If someone is looking into tests I prefer the still pretty new tinytest package over testthat. Most import it has really nice documentation, depends only on 2 R base packages (one is parallel, so it is fast and safe to use), familiar syntax, tests for side effects. And you don‘t need to trust my poor judgement, R guru Dirk Eddelbuettel (e.g. Rcpp) seems to make the transition.

(Question) Eric Scott

I missed the call, but if anyone is still looking at this, I have a question. What are some best practices for making big changes to a package? For example, changing the output of a function from a list to a tibble. Should this be a new function? Should there be an argument to get the old behavior back? Warnings to users to update their code? I'd appreciate any thoughts or examples of how to make these kinds of changes gracefully.

Call to action!

Do you want help maintaining **your** rOpenSci package?
Interested in helping someone maintain **their** rOpenSci package?

- Start by asking a question in the Package Maintenance category in our forum https://discuss.ropensci.org/c/package-maintenance/

- In every rOpenSci newsletter we have a “Call for Maintainers” section, keep an eye out for that if you are interested in helping to maintain or maintaining a package yourself
- The rOpenSci developer guide has a thorough treatment of our advice and policies on package development and maintenance

Resources

- Public discussion of this topic with > 20 comments from folks you know and love
- Taking over maintenance of a software package, by Scott Chamberlain, Maëlle Salmon & Noam Ross
- Relaunching the qualtRics package, by Julia Silge
- Package evolution - changing stuff in your package, Chapter 14 in the rOpenSci software development guide
- The mail must get through, by Eric Steven Raymond
- Learning a new codebase with Patrcia Aas
- add your favorites here

What would **YOU** like to hear about on a future rOpenSci Community Call?

Add your votes and ideas for topics and speakers by 1) browsing the issues in this public repository; 2) giving a 👍🏽 or commenting on an issue; 3) opening a new issue if your idea doesn’t fit in any others.