CSCCE Tech Case Study: Using GitHub to plan community calls for rOpenSci

About this case study

In 2023, CSCCE hosted a series of community "Tools Trials" that focused on tools commonly used for community management in open-source and open-source-supporting communities. The goal of the series - which consisted of several live webinars with Q&A and this accompanying tip sheet collection - was to share existing knowledge and provide any scientific community manager with options about tools and use cases they could explore further.

One of the most commonly used tools by these communities is GitHub. In this case study, we summarize how rOpenSci uses GitHub to organize their community calls. This is a very similar case study to the way The Carpentries uses GitHub to crowdsource and manage session proposals for their conference, CarpentryCon.

What is GitHub?

GitHub is a version control platform for storing, collaborating on, and tracking projects. It is primarily used for software development, but can also be used for other collaborative work. It utilizes Git, which is an open-source version control system that can track which users make changes to computer files (e.g., code, documents) and when. Although there are other software platforms that use Git, GitHub is the most popular. Check out our introductory tip sheet for more information.

Overview of rOpenSci

rOpenSci is a community of scientific open-source software users and developers. rOpenSci members participate in the community by sharing their knowledge, vetting other users' packages via a software peer review process, and maintaining packages within rOpenSci. The organization fosters a culture that values open and reproducible research using shared data and reproducible software for everyone, built by everyone, by creating technical and social infrastructure.

rOpenSci community members convene on GitHub, where they can interact by commenting and contributing on each other's code or by asking and answering questions. rOpenSci also runs a set of interrelated activities around capacity building, community development, and software development practices (including a champions program), as well as hosting regular community calls for its members to discuss and learn about shared interests. These community calls are the focus of this case study.

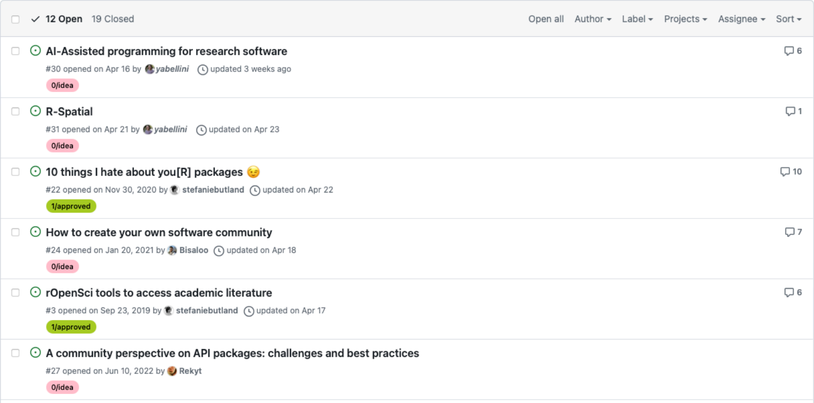
Using GitHub to manage rOpenSci's community call

rOpenSci's Community Calls are quarterly, one-hour online events with two to five speakers, one moderator, and between 20 to 40 minutes of Q&A time.

The main goal of these calls is to strengthen connections between members of the rOpenSci community, but they also serve to inform people about technical and social topics, to bring different stakeholders' perspectives to light, and to identify unmet needs on a topic. Generally, rOpenSci's calls attract a global audience of R users and developers, with representation from academia, government, non-profit, and industry.

To manage these community calls, rOpenSci utilizes a public repository on GitHub for crowdsourcing ideas (Figure 1). The general workflow includes the following steps:

- The community manager or a member of the community opens an issue in the repository for a potential topic
    - There is an issue template (Figure 2) for this, so that each issue contains enough information for others to understand the proposed topic. The template asks for:
        - Information about the proposed topic
        - Who is the audience?
        - Why is this important?
        - What should be covered?
        - Suggested speakers or contributors
        - Resources you'd recommend to the audience
- Members can then comment on these issues or vote using emojis
- The community manager uses labels to annotate the status of these issues (e.g., "idea," "approved," and "working on it")



☐ ⊙ AI-Assisted programming for research software 🗨 6
#30 opened on Apr 16 by yabellini updated 3 weeks ago
0/idea

☐ ⊙ R-Spatial 🗨 1
#31 opened on Apr 21 by yabellini updated on Apr 23
0/idea

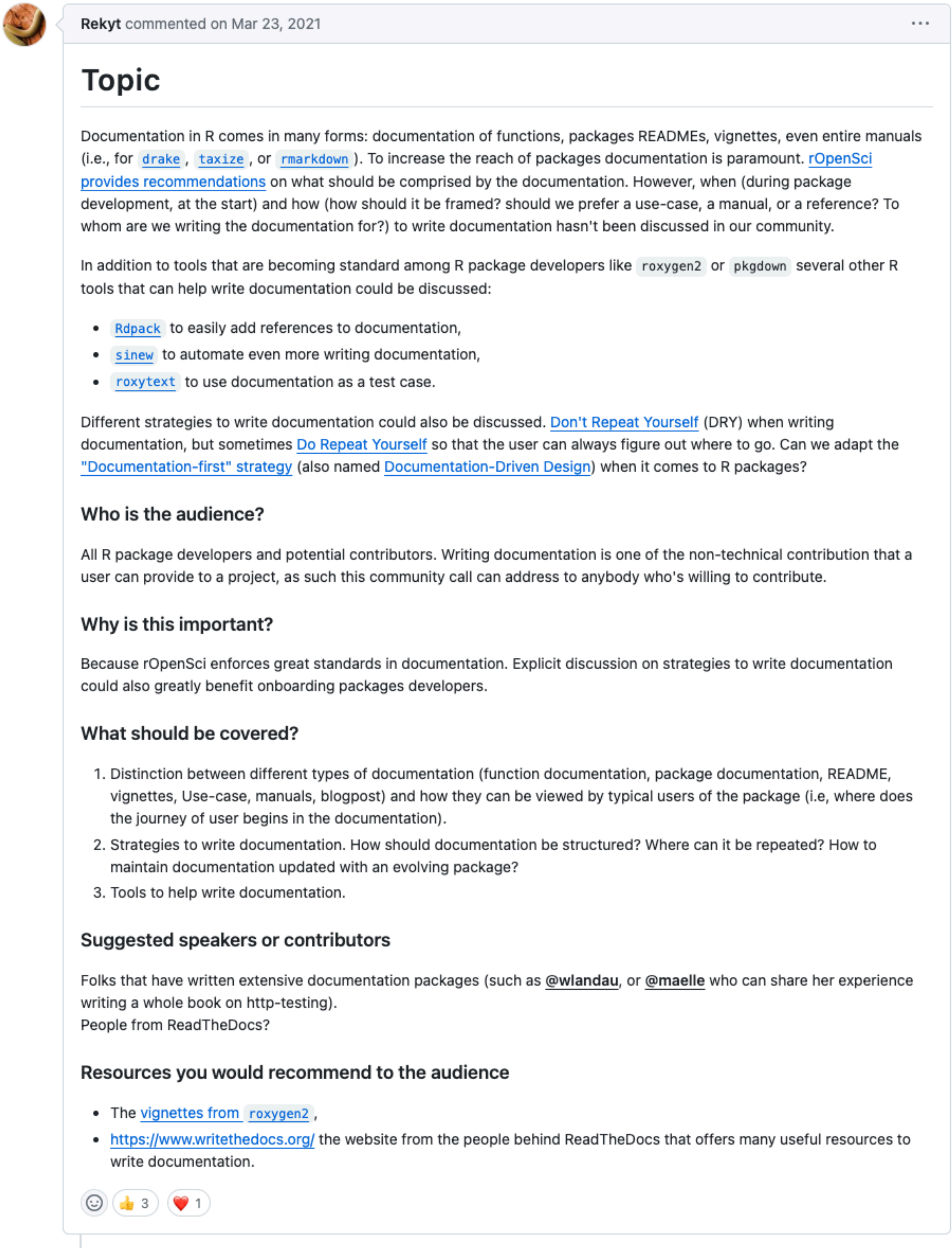
☐ ⊙ 10 things I hate about you[R] packages 😉 🗨 10
#22 opened on Nov 30, 2020 by stefaniebutland updated on Apr 22
1/approved

☐ ⊙ How to create your own software community 🗨 7
#24 opened on Jan 20, 2021 by Bisaloo updated on Apr 18
0/idea

☐ ⊙ rOpenSci tools to access academic literature 🗨 6
#3 opened on Sep 23, 2019 by stefaniebutland updated on Apr 17
1/approved

☐ ⊙ A community perspective on API packages: challenges and best practices
#27 opened on Jun 10, 2022 by Rekyt
0/idea

Figure 1
The rOpenSci community call public repo in GitHub.



Rekyt commented on Mar 23, 2021

Topic

Documentation in R comes in many forms: documentation of functions, packages READMEs, vignettes, even entire manuals (i.e., for `drake`, `taxize`, or `rmarkdown`). To increase the reach of packages documentation is paramount. rOpenSci provides recommendations on what should be comprised by the documentation. However, when (during package development, at the start) and how (how should it be framed? should we prefer a use-case, a manual, or a reference? To whom are we writing the documentation for?) to write documentation hasn't been discussed in our community.

In addition to tools that are becoming standard among R package developers like `roxygen2` or `pkgdown` several other R tools that can help write documentation could be discussed:

- `Rdpack` to easily add references to documentation,
- `sinew` to automate even more writing documentation,
- `roxytext` to use documentation as a test case.

Different strategies to write documentation could also be discussed. Don't Repeat Yourself (DRY) when writing documentation, but sometimes Do Repeat Yourself so that the user can always figure out where to go. Can we adapt the "Documentation-first" strategy (also named Documentation-Driven Design) when it comes to R packages?

Who is the audience?

All R package developers and potential contributors. Writing documentation is one of the non-technical contribution that a user can provide to a project, as such this community call can address to anybody who's willing to contribute.

Why is this important?

Because rOpenSci enforces great standards in documentation. Explicit discussion on strategies to write documentation could also greatly benefit onboarding packages developers.

What should be covered?

1. Distinction between different types of documentation (function documentation, package documentation, README, vignettes, Use-case, manuals, blogpost) and how they can be viewed by typical users of the package (i.e, where does the journey of user begins in the documentation).
2. Strategies to write documentation. How should documentation be structured? Where can it be repeated? How to maintain documentation updated with an evolving package?
3. Tools to help write documentation.

Suggested speakers or contributors

Folks that have written extensive documentation packages (such as **@wlandau**, or **@maelle** who can share her experience writing a whole book on http-testing).
People from ReadTheDocs?

Resources you would recommend to the audience

- The vignettes from `roxygen2`,
- https://www.writethedocs.org/ the website from the people behind ReadTheDocs that offers many useful resources to write documentation.

👍 3 ❤️ 1

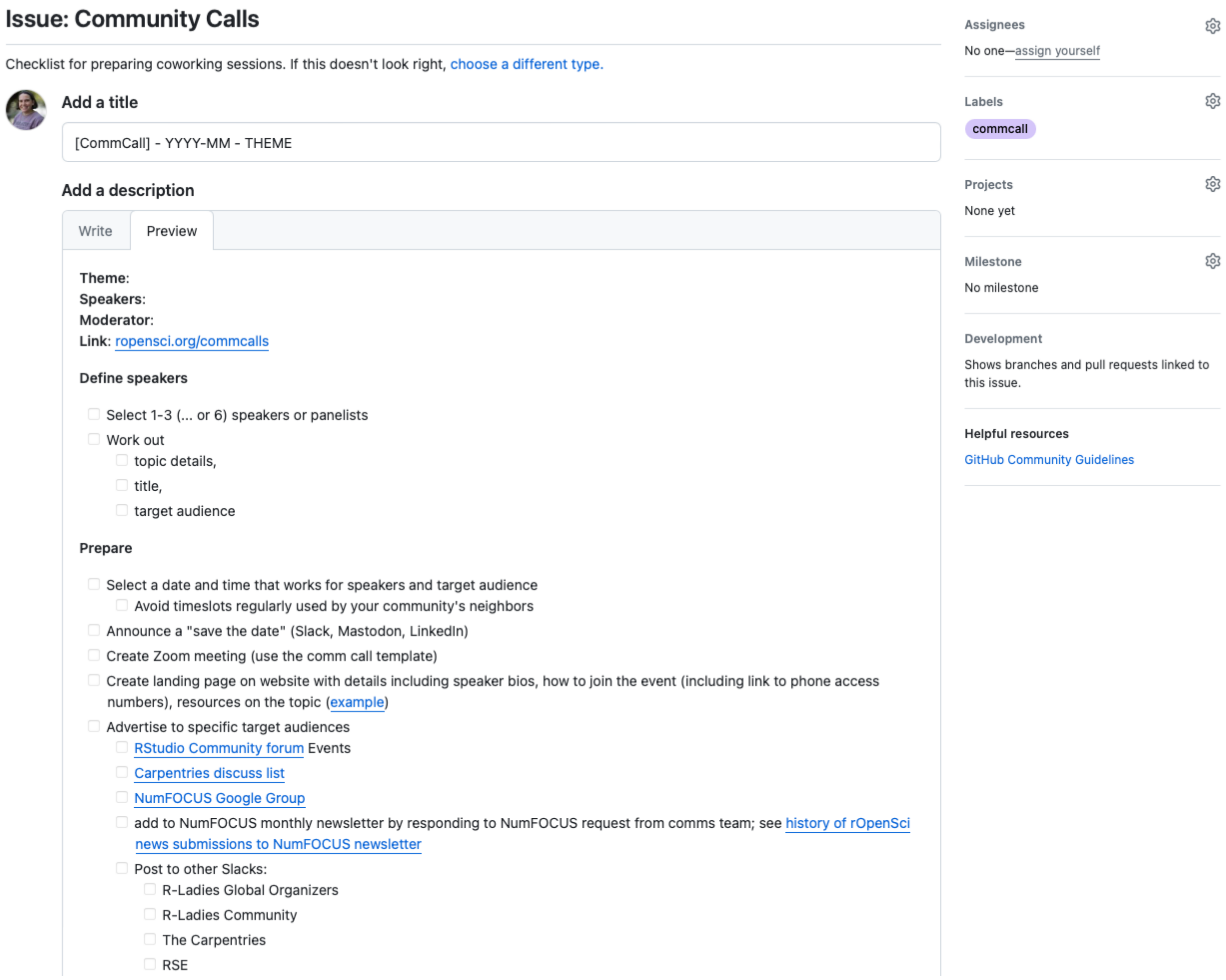
Figure 2

An example of a completed issue containing details about a session proposal for an rOpenSci community call. This is a customized issue template to gather the same types of information from all proposers.

This community-engaged process helps the community manager prioritize topics, and also receive input and suggestions about potential speakers.

Behind the scenes

rOpenSci also maintains a private repository for managing community calls, which only includes issues for call ideas where a decision has been made to include the topic in a future call. These issues are also created with a template (Figure 3), which generates a complete and detailed checklist with all the necessary steps to organize the community call and with links to all the necessary materials, such as the template to the shared document or presentation slide, sample social media posts, and the software used to process subtitles.



Issue: Community Calls

Checklist for preparing coworking sessions. If this doesn't look right, choose a different type.

Add a title

[CommCall] - YYYY-MM - THEME

Add a description

Theme:
Speakers:
Moderator:
Link: ropensci.org/commcalls

Define speakers

- □ Select 1-3 (... or 6) speakers or panelists
- □ Work out
  - □ topic details,
  - □ title,
  - □ target audience

Prepare

- □ Select a date and time that works for speakers and target audience
  - □ Avoid timeslots regularly used by your community's neighbors
- □ Announce a "save the date" (Slack, Mastodon, LinkedIn)
- □ Create Zoom meeting (use the comm call template)
- □ Create landing page on website with details including speaker bios, how to join the event (including link to phone access numbers), resources on the topic (example)
- □ Advertise to specific target audiences
  - □ RStudio Community forum Events
  - □ Carpentries discuss list
  - □ NumFOCUS Google Group
  - □ add to NumFOCUS monthly newsletter by responding to NumFOCUS request from comms team; see history of rOpenSci news submissions to NumFOCUS newsletter
  - □ Post to other Slacks:
    - □ R-Ladies Global Organizers
    - □ R-Ladies Community
    - □ The Carpentries
    - □ RSE

Assignees
No one—assign yourself

Labels
commcall

Projects
None yet

Milestone
No milestone

Development
Shows branches and pull requests linked to this issue.

Helpful resources
GitHub Community Guidelines

Figure 3

An example of an empty issue with the check list and links to resources for organizing a rOpenSci community call. This is a customized issue template that lists all the tasks to do before, during, and after the community call

Pros and cons of using GitHub for this use case

[For general pros and cons of using GitHub for community management, please refer to our overview tip sheet.]

Pros:

- The public repo offers a clear way for community members to engage in identifying and discussing ideas for future community calls and the status of each topic.
- It empowers community members to set the programming agenda by proposing and prioritizing topics of interest to them.
- The private repo standardizes the process for the community management team.
- Issue templates standardize and simplify the community submission process.

Cons:

- There is a risk of publicly proposing an idea which results in disagreements among community members, which underlines the importance of adopting and enforcing a Code of Conduct so that contributors feel safe.
- A member may want to submit their idea in a non-public forum (in rOpenSci's case, they can also submit ideas via Slack or email).
- You need to be logged in to GitHub to propose a topic or engage in the conversation and, therefore, have some level of comfort using the platform. Someone without a profile or unfamiliar with the platform may not feel included to participate.

Additional resources

- Yanina Bellini Saibene's full presentation on GitHub for rOpenSci's community call - Watch this case study being presented during CSCCE's Tools Trial in this short video
- rOpenSci's Community Call Webpage
- Community call repository: https://github.com/ropensci-org/community-calls/issues
- GitHub automation: https://ropensci.org/commcalls/dec2021-automation/
- GitHub for translations: https://ropensci.org/commcalls/nov2023-multilingual/

Acknowledgements

This case study is based on a presentation by Yanina Bellini Saibene, Community Manager at rOpenSci, who also contributed to the content of this case study. CSCCE uses the CRediT contributor roles taxonomy to show how the authors listed contributed to the creation of this guide:

YANINA BELLINI SAIBENE - Writing - Original draft preparation, Writing - Reviewing and Editing

KATIE PRATT - Conceptualization, Supervision, Writing - Original draft preparation, Writing - Reviewing and Editing, Visualization

EMILY LESCAK - Supervision, Writing - Original draft preparation

ALYCIA CRALL - Writing - Reviewing and Editing

LOU WOODLEY - Conceptualization, Funding acquisition, Supervision, Writing - Reviewing and Editing

Citing and reusing this tip sheet

CITATION AND REUSE

“Tech Case Study: Using GitHub to plan community calls for rOpenSci,” by Yanina Bellini Saibene, Katie Pratt, Emily Lescak, Alycia Crall, and Lou Woodley is licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.

Cite as: Center for Scientific Collaboration and Community Engagement. (2024) Tech Case Study: Using GitHub to plan community calls for rOpenSci. Bellini Saibene, Pratt, Lescak, Crall, and Woodley doi: 10.5281/zenodo.12117430

Please clearly attribute any modified use of this material as “Adapted from an original by the Center for Scientific Collaboration and Community Engagement (CSCCE) under a CC BY 4.0 license, doi: 10.5281/zenodo.12117430. We request that the CSCCE logo is clearly visible next to the attribution, that you indicate which is the original text, and include the full citation as described above. If you have any questions, please email info@cscce.org.

Last updated June 2024