Pull request responsibilities

Childhood Cancer Data Lab x Alex's Lemonade Stand

Objectives

- Cover some common complaints around code review
- Characterize responsibilities of pull request authors and reviewer
- Illustrate how automation and other GitHub features can support authors and reviewers

Common complaints about code review

- **It takes too long**, and long-lived branches result in merge conflicts.
- **It's no fun.** It taxes both the authors and the reviewers.
- **No clear benefit** because reviews are too long and are rushed.

Source: Wikimedia Commons

And your team leader and the rest of your team, with systems and processes, can prevent these complaints

https://blog.arkency.com/disadvantages-of-pull-requests/

Some questions you should ask yourself

- **Who is going to review this?** A good code review will be confident and requires some understanding about the project and problem at hand.[1] But it can also be an opportunity to onboard people to the project. Maybe a pull request would benefit from multiple reviewers playing different roles!

- **What are the stakes?** A quick, one-off analysis for internal purposes needs to be correct, but it doesn't necessarily need to be extensible or efficient.
- **When does this need to be completed?** If you have a code review policy, you should factor in that code review takes time, and something isn't completed until it's merged
- **What is this blocking?** Related to the *when* question – how should review be prioritized?

[1] https://slack.engineering/how-about-code-reviews/

We ask these questions in service of our goal

Generally, leveraging the benefits of code review, such as

- Raising quality
- Sharing knowledge

While making good, sustainable progress

- Working on the right things
- Not grinding to a halt
- Not overburdening our team

Shared responsibilities

Does it accomplish what it needs to accomplish?

Both the author and the reviewer need to understand the purpose of the pull request and who the intended audience of the output is.

Addresses: **Clear benefit**

Does it meet the required standards?

The author's job is to put forth something they believe is valid (correct, reproducible, maybe reusable and efficient), and the reviewer's job is to confirm the work's validity.

Addresses: **Clear benefit**

Are the right people reviewing it?

An author should request a reviewer with enough understanding of the problem and project that can verify the work's validity.

Reviewers should be open and honest about the potential limitations of their reviews.

Addresses: **Clear benefit**

Don’t be a jerk

Hopefully, this goes without saying. Scientific, collaborative work requires team psychological safety.

Sharing your knowledge with your teammates, either as an author or as a reviewer, can be fun!

Addresses: **No fun**

Instead...

- Assume the other person on your team is doing a good job! Maybe they know something you don’t.
- Get curious and ask for clarifications.
- Use wording like “us”, “our”, “we” rather than “you”, “your”, and “mine.”
- Remember to point out what you liked or learned from, too.

From https://slack.engineering/on-empathy-pull-requests/

Keep things moving

Yes, there are consequences to things hanging out in review too long. Some sense of urgency is good.

Does everything need to be addressed now, or can you follow up with new issues and subsequent pull requests instead?

Addresses: **Takes too long**

Summary: Shared responsibilities

- Code review is a collaborative effort and care must be taken to ensure that it’s a net positive.
- Pull requests don’t happen in a vacuum, so it’s helpful to consider the “bigger picture” like what happens when a PR get merged both as an author and a reviewer.
- A team’s code review practice can benefit from some guidelines. What are some guidelines you would write?

Author responsibilities

An **author** is the person that is directly **responsible** and **accountable** for an issue

A good issue is focused

Roughly, the smallest unit of work something could be broken down into to result in a single, functional pull request that is manageable for review.

Rules of thumb for **focused pull requests**

- Represents a single task
- If multiple things are changing, they should be related
- ~400 lines or fewer

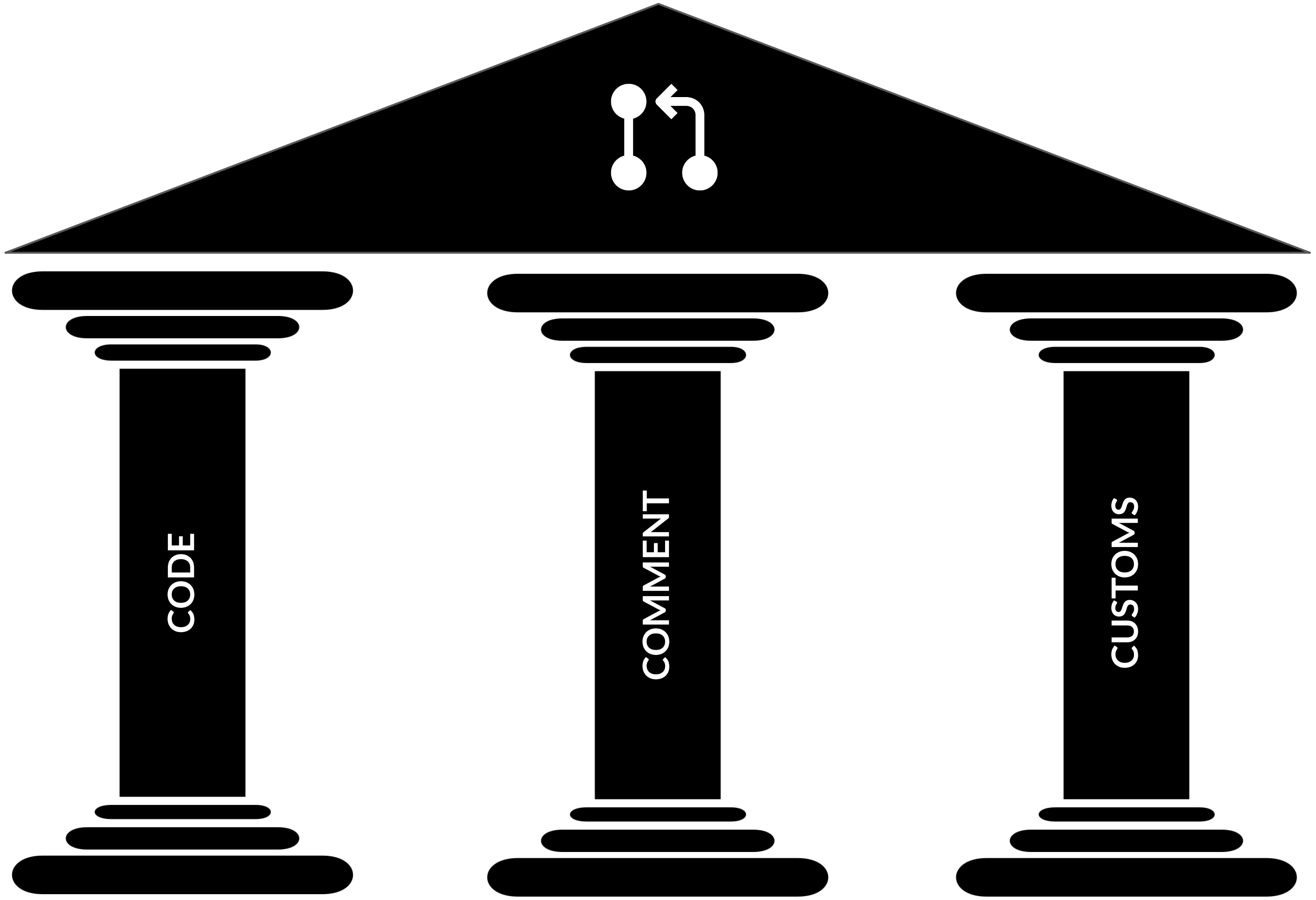
Ideas adapted from *On Empathy & Pull Requests*

Authors are responsible for **more than just code**

Or more than just the files changed.

They also have the following responsibilities:

- Introducing or documenting the pull request for the reviewer
- Requesting a reviewer
- Complying with other policies (e.g., connecting to the relevant issue)


CODE
COMMENT
CUSTOMS

Before you file

Please look at the Files Change tab on GitHub 👀

Is your pull request ***focused***?

> How many lines have changed?
>
> Did some irrelevant files sneak in somehow?

In a later session, we'll cover some techniques for taming your diffs if your Files Changed are likely to strike fear into the heart of your reviewer.

Code related responsibilities

Any files that get sent for review should be **correct** to the best of the author's knowledge and **meet the team standards**, which may include things like:

- Variable or function naming guidance is followed
- File documentation is up to date

Comment related responsibilities

- **Provide context.** PRs should have informative titles. Authors should explain what they did and why they ended up with the solution they're proposing to merge, as well as link to relevant material that was consulted.
  - Remember, your reviewer doesn't know about all of the work you've put in!
- **Structure comments to elicit substantive feedback.**
  - In our experience, reviewers have a tendency to want to jump in and comment line-by-line. But maybe you want comments on the bigger picture. Tell the reviewer what you'd like them to focus on!

Ideas adapted from *On Empathy & Pull Requests* and Parker. 2017.

Pull request templates can help with comment related responsibilities!

Customs related responsibilities

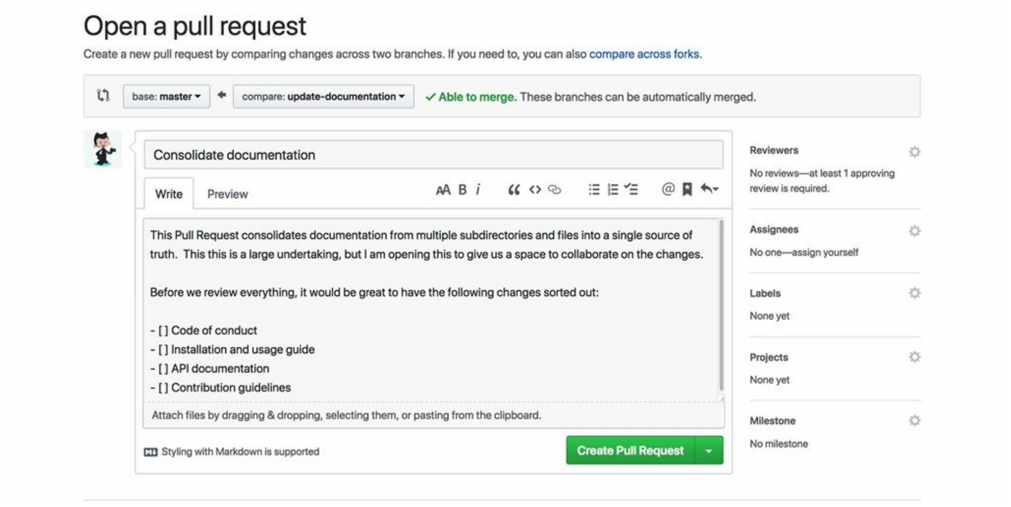
Team policy likely dictates that an author is responsible for **requesting a reviewer**.

Beyond that an author should **adhere to all team procedures, policies, or practices** for pull requests, which might include:

- Using specific branch-naming conventions
- Mentioning the relevant GitHub issue in the text of their comment
- Indicating the status of an item in an external project management system

Draft pull requests as a special case

GitHub has functionality that allows you to file a PR but indicate that it is not ready for review or merging. That's called a draft pull request!



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: master ▾ compare: update-documentation ▾ ✓ Able to merge. These branches can be automatically merged.

Consolidate documentation

This Pull Request consolidates documentation from multiple subdirectories and files into a single source of truth. This this is a large undertaking, but I am opening this to give us a space to collaborate on the changes.

Before we review everything, it would be great to have the following changes sorted out:

- [] Code of conduct
- [] Installation and usage guide
- [] API documentation
- [] Contribution guidelines

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

Styling with Markdown is supported

Reviewers
No reviews—at least 1 approving review is required.

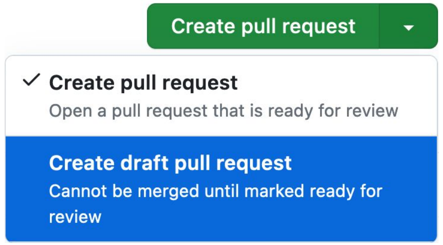
Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Image source: GitHub blog


Create pull request
✓ Create pull request
Open a pull request that is ready for review
Create draft pull request
Cannot be merged until marked ready for review

Draft pull requests as a special case

Draft pull requests allow authors to get high-level feedback on a solution or results while signalling to others that the code is not "ready for primetime."

These can come in handy when it turns out an author didn't fully understand gotchas associated with an issue or if they get surprising results!

Summary: Author responsibilities

- An author is a person directly responsible and accountable for an issue or task.
- Author responsibilities may be broader than you think!
  - **Code** related - Focused, a manageable size and diff
  - **Comment** related - Sets up the reviewer to have a good experience
  - **Customs** related - Requests a reviewer, adheres to other team standards and practices

Systems approaches to facilitating code review

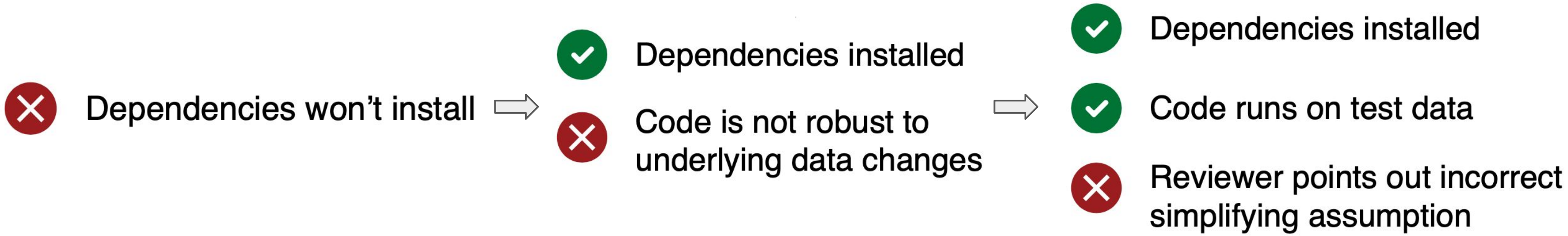
What do I mean by **system approaches**?

Anything that goes beyond the scope of an individual pull request (i.e., applies to all of them) or an individual author's responsibilities.

Alt title: Solving pull request problems at a deeper level!

Let’s imagine a pull request where we want to make sure:

- All the packages required to run the new code are in the project environment
- The new code can be reused on other data sets
- The approach is methodologically sound



❌ Dependencies won't install ⇨

✅ Dependencies installed

❌ Code is not robust to underlying data changes ⇨

✅ Dependencies installed

✅ Code runs on test data

❌ Reviewer points out incorrect simplifying assumption

Better done by robots 🤖

Needs people

Adapted from Shapiro et al. 2023.

If we can learn to **spot tasks for robots**, we can **save reviewers' energy** for the things that need it

We can also leverage automations to codify and enforce policies.

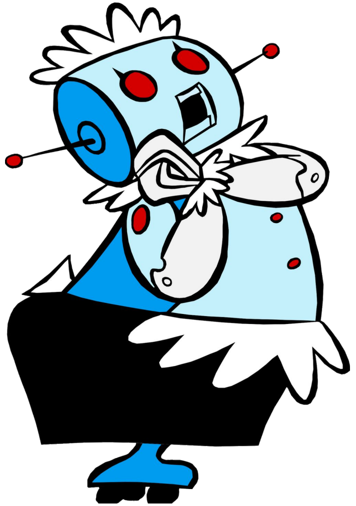


Image from the Jetsons Fandom

Beyond automation, we can use other features of GitHub to **create structure** that appears when people perform the task of **creating a pull request**.

A **GitHub Actions** primer

GitHub Actions (GHA) is a service through GitHub that allows for automation of workflows. GitHub and third parties create “actions,” which are like template workflows (e.g., checking out a repository or pushing to Dockerhub).

The nice part about GHA is that it can be triggered on many different GitHub events like a commit to a pull request targeting a specific branch or when an issue is created.

Energy-saving example: Spell check

Using GitHub Actions to check for spelling errors – failing if any errors are detected – allows pull request authors to correct them before a reviewer looks at it.

Let's take a look at some real world examples:

`training-modules` workflow, Rscript, and custom dictionary

`refinebio-docs` workflow, config, and custom dictionary

Codifying policies example: Styling

Using GitHub Actions to enforce code style relieves authors' and reviewers' burden of styling code and enforcing style, respectively.

Debate about style can happen once – when selecting or implementing the workflow – instead of on every pull request.

Let's take a look at some real world examples:

`refinebio-examples` workflow

`scpcaTools` workflow

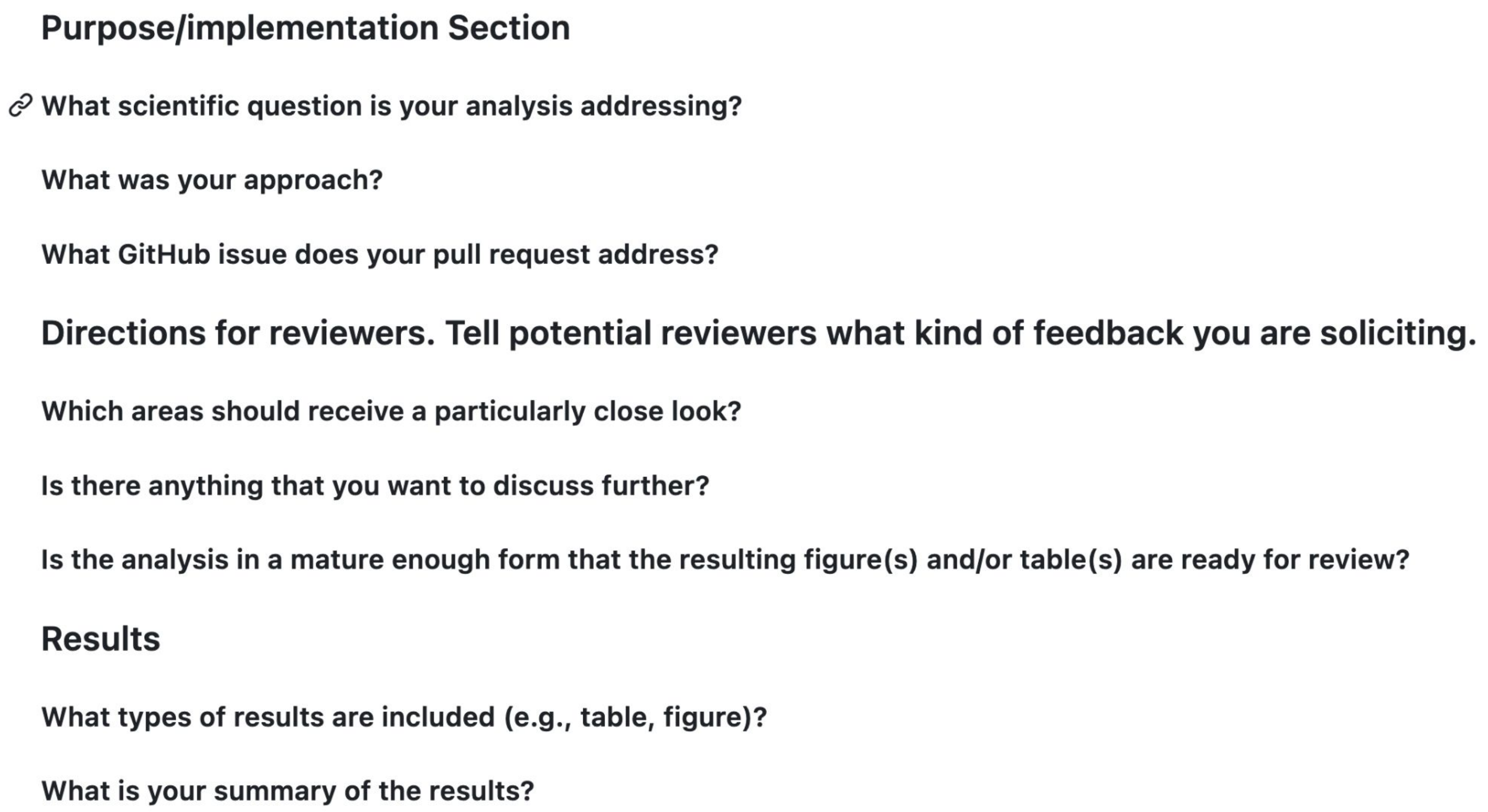
Pull request templates can help prompt authors to satisfy requirements

Whether that's providing context for the pull request, linking to an issue, or a checklist of required steps

Real world examples of **pull request templates**

In the next few slides, we'll review

- A single, universal PR template from the `OpenPBTA-analysis` project
- Multiple PR templates (and how to use them) from a repository called `refinebio-examples`



Purpose/implementation Section

What scientific question is your analysis addressing?

What was your approach?

What GitHub issue does your pull request address?

Directions for reviewers. Tell potential reviewers what kind of feedback you are soliciting.

Which areas should receive a particularly close look?

Is there anything that you want to discuss further?

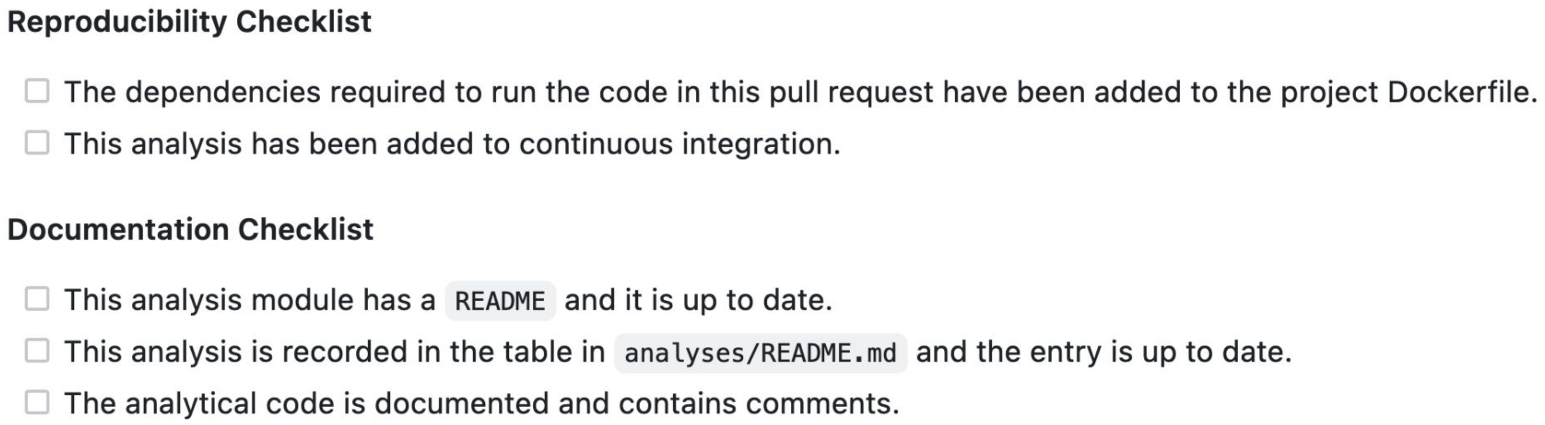
Is the analysis in a mature enough form that the resulting figure(s) and/or table(s) are ready for review?

Results

What types of results are included (e.g., table, figure)?

What is your summary of the results?

https://github.com/AlexsLemonade/OpenPBTA-analysis/blob/master/.github/PULL_REQUEST_TEMPLATE.md



Reproducibility Checklist

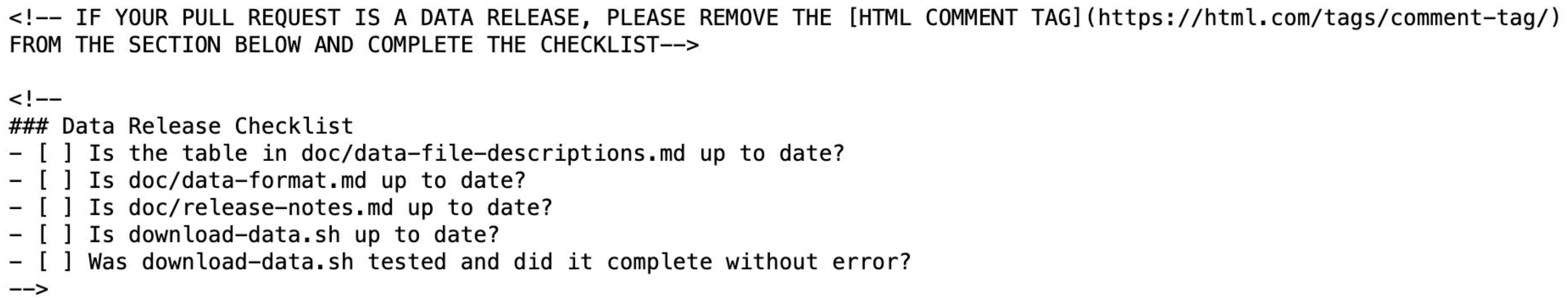
- [ ] The dependencies required to run the code in this pull request have been added to the project Dockerfile.
- [ ] This analysis has been added to continuous integration.

Documentation Checklist

- [ ] This analysis module has a `README` and it is up to date.
- [ ] This analysis is recorded in the table in `analyses/README.md` and the entry is up to date.
- [ ] The analytical code is documented and contains comments.

https://github.com/AlexsLemonade/OpenPBTA-analysis/blob/master/.github/PULL_REQUEST_TEMPLATE.md

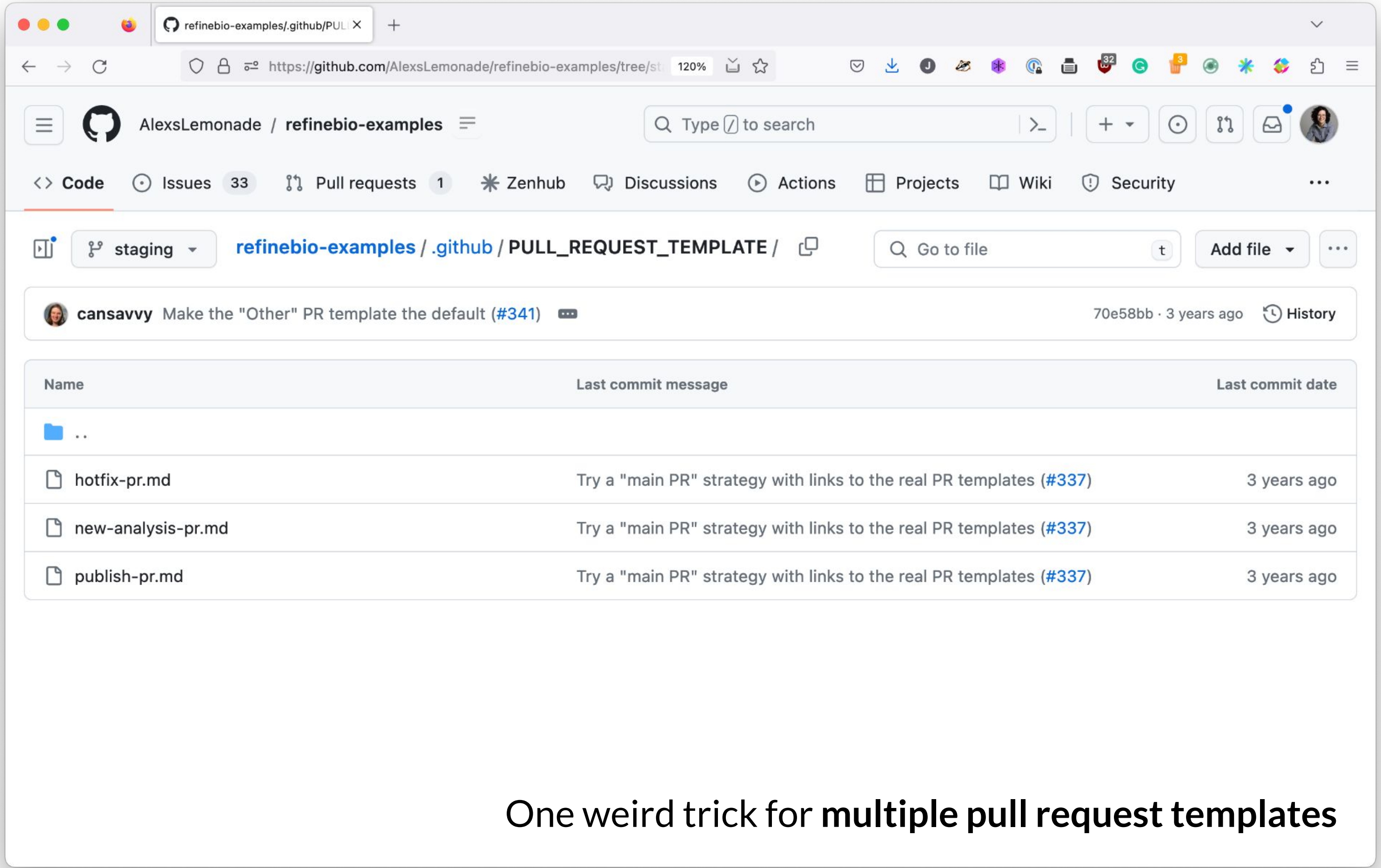
An HTML comment includes another checklist for data releases



```
<!-- IF YOUR PULL REQUEST IS A DATA RELEASE, PLEASE REMOVE THE [HTML COMMENT TAG](https://html.com/tags/comment-tag/)
FROM THE SECTION BELOW AND COMPLETE THE CHECKLIST-->

<!--
### Data Release Checklist
- [ ] Is the table in doc/data-file-descriptions.md up to date?
- [ ] Is doc/data-format.md up to date?
- [ ] Is doc/release-notes.md up to date?
- [ ] Is download-data.sh up to date?
- [ ] Was download-data.sh tested and did it complete without error?
-->
```

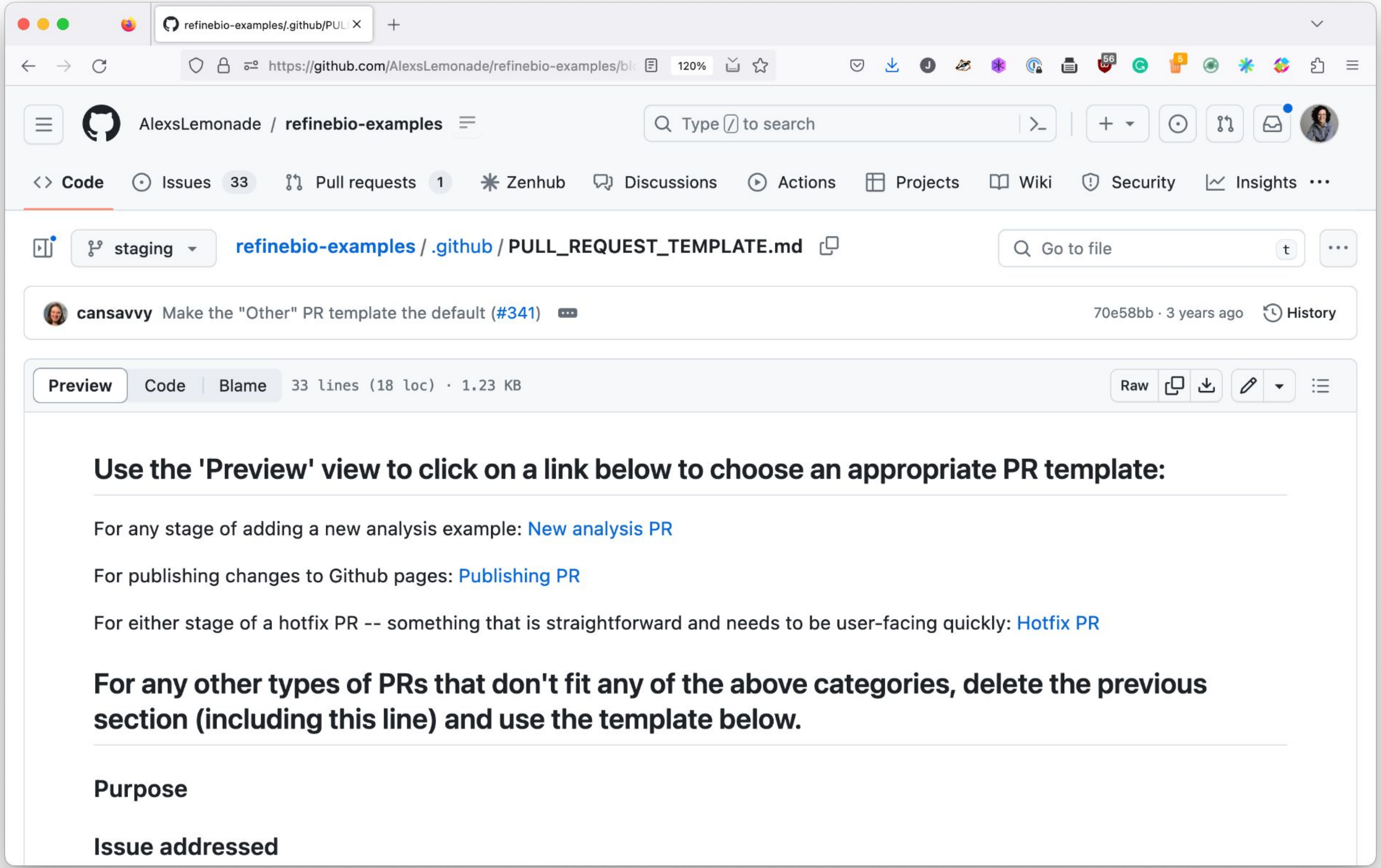
https://github.com/AlexsLemonade/OpenPBTA-analysis
/blob/master/.github/PULL_REQUEST_TEMPLATE.md


refinebio-examples/.github/PUL
https://github.com/AlexsLemonade/refinebio-examples/tree/st
120%
AlexsLemonade / refinebio-examples
Type / to search
Code
Issues 33
Pull requests 1
Zenhub
Discussions
Actions
Projects
Wiki
Security
staging
refinebio-examples / .github / PULL_REQUEST_TEMPLATE /
Go to file
t
Add file

cansavvy Make the "Other" PR template the default (#341) 70e58bb · 3 years ago History

Name	Last commit message	Last commit date
..		
hotfix-pr.md	Try a "main PR" strategy with links to the real PR templates (#337)	3 years ago
new-analysis-pr.md	Try a "main PR" strategy with links to the real PR templates (#337)	3 years ago
publish-pr.md	Try a "main PR" strategy with links to the real PR templates (#337)	3 years ago

One weird trick for **multiple pull request templates**


refinebio-examples/.github/PUL
https://github.com/AlexsLemonade/refinebio-examples/bl
120%
AlexsLemonade / refinebio-examples
Type / to search
Code
Issues 33
Pull requests 1
Zenhub
Discussions
Actions
Projects
Wiki
Security
Insights
staging
refinebio-examples / .github / PULL_REQUEST_TEMPLATE.md
Go to file
t
cansavvy Make the "Other" PR template the default (#341)
70e58bb · 3 years ago
History
Preview
Code
Blame
33 lines (18 loc) · 1.23 KB
Raw

Use the 'Preview' view to click on a link below to choose an appropriate PR template:

For any stage of adding a new analysis example: New analysis PR

For publishing changes to Github pages: Publishing PR

For either stage of a hotfix PR -- something that is straightforward and needs to be user-facing quickly: Hotfix PR

For any other types of PRs that don't fit any of the above categories, delete the previous section (including this line) and use the template below.

Purpose

Issue addressed

Summary: Systems approaches to facilitating code review

- GitHub Actions – or other continuous integration/continuous deployment (CI/CD) services – can automate some things we'd like to check during a review, thereby taking them off a human reviewer's plate.
- Pull request templates support individual authors in meeting their responsibilities by reminding them of what to include in an initial pull request comment right when they write it.

Reviewer responsibilities

Reviewers must **stand behind** the work they approve

If you have a code review policy, that means review is part of your job!

The goal is to provide **constructive feedback** so the whole team gets better

Code review should **not be superficial**

Reviews should not be overly focused on style or formatting. Ideally, you've set up systems (🤖 👋) to help with the surface-level stuff.

Reviews should focus on the big picture.

It's easier to live up to this responsibility when pull requests are small enough bites.

From *How About Code Reviews?* And *Pull Requests—The Good, the Bad and Really, Not That Ugly*

Code review requires **understanding**

Read the pre-existing code if you need to.

Ask clarifying questions.

Tag in someone with more experience or expertise if you’re not the right reviewer!

From *How About Code Reviews?*

Share your knowledge

Maybe you’re very familiar with the foibles of the particular package being used. Now’s your chance to pass this knowledge on to your coworkers!

Did you consult supplementary material from a paper during your review? Go ahead and link that in your comments! (Same goes for StackOverflow, blog posts, etc.)

From *How About Code Reviews?*

Distinguish between must-haves and nice-to-haves

Or preferences, or matters of opinion, etc.

Make it easy for the author to identify what it will take to get this pull request merged.

You can lean on the different options for returning a review to help:

- **Approval** means that the code can be merged.
- **Changes requested** means that the reviewer that requested the changes must re-review and approve.
- **Commenting** means the code can be merged if someone else with the appropriate permissions approves.

From *Pull Requests—The Good, the Bad and Really, Not That Ugly*

Explain how you reviewed

Did you attempt to run the code locally and get an error?

Did you read the code and think through edge cases?

All of this is helpful context for the author!

Pick up the phone if you need to!

If you fundamentally disagree with someone’s approach, (internally) public written communication might be a tricky way to have a productive conversation about it.

You might want to opt for a synchronous discussion instead and record the outcomes on the pull request.

From *How About Code Reviews?*

Summary: Reviewer responsibilities

- If you have a code review policy, review is an important part of your role!
- The point of review is constructive feedback, which dictates that review:
  - Shouldn’t be overly focused on superficial things like style
  - Requires understanding of the existing code base or problem at hand
  - Facilitates knowledge sharing
- Practically speaking, it’s helpful if a reviewer:
  - Communicates the distinction between what it will take for them to approve and nice-to-haves
  - Explains what steps they took as part of their review

What policies might you need to write for code review?