



Analytical code review overview

Childhood Cancer
Data Lab

x






Objectives

- Start with *WHY* (ours, anyway)
- Illustrate some practical benefits of code review
- Catalog some practical considerations for code review
- Describe what code review can't do

The Data Lab's why

- **Set and maintain high standards** for analytical code, documentation, and training materials to maximize their value to the pediatric cancer community.
- **Provide training and knowledge sharing opportunities**; people do not have uniform experience with everything they must do as part of their role.
- **Improve our overall efficiency (long-term)** with code we can reuse and docs that make it easier to do our jobs.

Q: If you practice code review, do you know your team's why? If you don't practice code review, are there part of your team's overall objectives that might be addressed by adopting that practice?





Practical benefits of code review



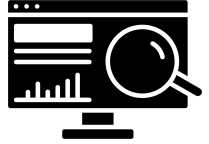


What's the point of an analysis, anyway?

The primary purpose of an analysis is to **obtain and communicate a result**.

When we obtain a result, we want it to be **accurate or correct**. Ideally, we'd also like it if we could:

- Reliably get the same result when re-running the code with the same data (i.e., be computationally **reproducible**)
- **Reuse the code** for other datasets
- Get the results **efficiently** (e.g., run time, memory requirements)



What's the point of an analysis, anyway?

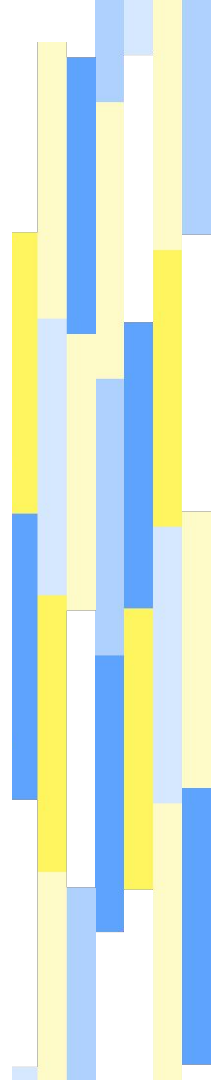
The primary purpose of an analysis is to **obtain and communicate a result**.

We often don't know if we've successfully communicated a result until we explicitly test that our message is received.

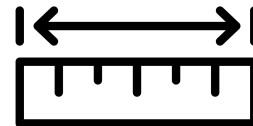
Sometimes we need to write processing or data-cleaning code in service of our analyses, but we ultimately want that to be **accurate, reproducible, reusable, and efficient**, too.

It's a lot easier to be
critical of other
people's work than
your own

Central assumption



Accurate or correct



“The idea than an analysis should be accurate is probably such a strong assumption within a scientific or business setting that it is rarely discussed.” – [Parker. 2017.](#)

How code review helps:

- You can write automatic tests to make sure a metric comes back within the expected range, but sometimes it takes another human to point out an incorrect assumption
- Experienced analysts often have expert “data intuition,” and two heads can be better than one for common sense checks

Very much from [Parker. 2017.](#)



Reproducible

Reproducibility = *“The ability of independent analysts to recreate the results claimed by the original authors using the original data and analysis techniques.”*

Goals of reproducible research:

- Provide a record of the processes used to analyze data underlying a work, which can allow for understanding beyond what’s in a methods section
- Ensure integrity and transparency into an analysis

Quote and goals from [Peng and Hicks. 2021.](#)



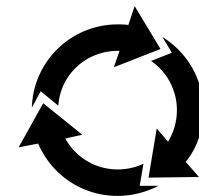
Reproducible

Reproducibility = *“The ability of independent analysts to recreate the results claimed by the original authors using the original data and analysis techniques.”*

How code review helps:

- Can explicitly test if someone else can repeat your analysis and get the same results
- Can explicitly test transparency of the analysis – if something is publicly available but unclear, it’s less transparent by definition
- Experienced analysts might know, for example, when to set a seed

Quote from [Peng and Hicks. 2021.](#); ideas from [Parker. 2017.](#)



Reusable

There are clear scientific reasons to reuse code, for example:

- Eliminate technical differences between samples due to differences in processing
- Repeat an analysis on an independent validation set

How code review helps:

- Can help spot patterns will yield correct results but may limit reuse, such as hardcoding file paths, parameters, or sample identifiers

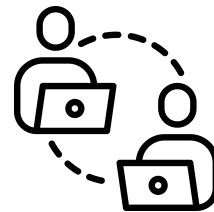
Efficient



It can be helpful if the code we write provides results in a timely manner, and we could use it on machines that are readily available or cost less to use. This is particularly important if an analysis isn't a one-off!

How code review helps:

- Experienced reviewers might have knowledge of language, package, or function quirks that they can pass on by providing feedback on implementations



Effectively communicated

Code review can explicitly test if someone understands your results and agrees with your conclusions long before manuscript peer review.

In practice, having to answer questions about your analysis during review can clarify ideas and decisions, even when (or perhaps especially when) your reviewer has less experience than you do.



Practical considerations

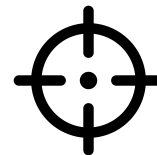


Some questions you should ask yourself

- **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/>

What makes a **good pull request**?



Focused on a single task. Multiple, related fixes are okay, but the cumulative changes should be manageable to review without getting overly fatigued.

For an analysis pull request, that might look like a single notebook.

As a rule of thumb, anything more than **400 lines of changes** is too big! (We'll cover some strategies for taming your diffs later.)



Code review is not a panacea



Code review is not a panacea



- You probably won't catch every bug with code review alone.
- It is certainly not a substitute for effective team communication, but it can play a role in supporting team communication.
 - Coordination is often required to keep things moving (i.e., for timely review that minimizes the chance of merge conflicts).
- Code review is not necessarily the time to have sweeping discussions about the direction of a project or analyses; have those first!

Summary

- Code review can help us ensure that our analytical code is accurate or correct, reproducible, reusable, and efficient.
- Code review can explicitly test whether we've communicated a result effectively.
- Code review can not stand alone. For example, a review is not usually the time to have broad discussions about the direction of a project.

Q: If you practice code review, do you know your team's why? If you don't practice code review, are there part of your team's overall objectives that might be addressed by adopting that practice?

