documentation.html code-review.Rmd

Lessons in ‘Code Review’ {

[Implementation Reflections]

< StatChat November 2023 >
< Regina Lisinker >

}

LASER
Lab Advancing Statistics Education Research

What is ‘Code Review’; {

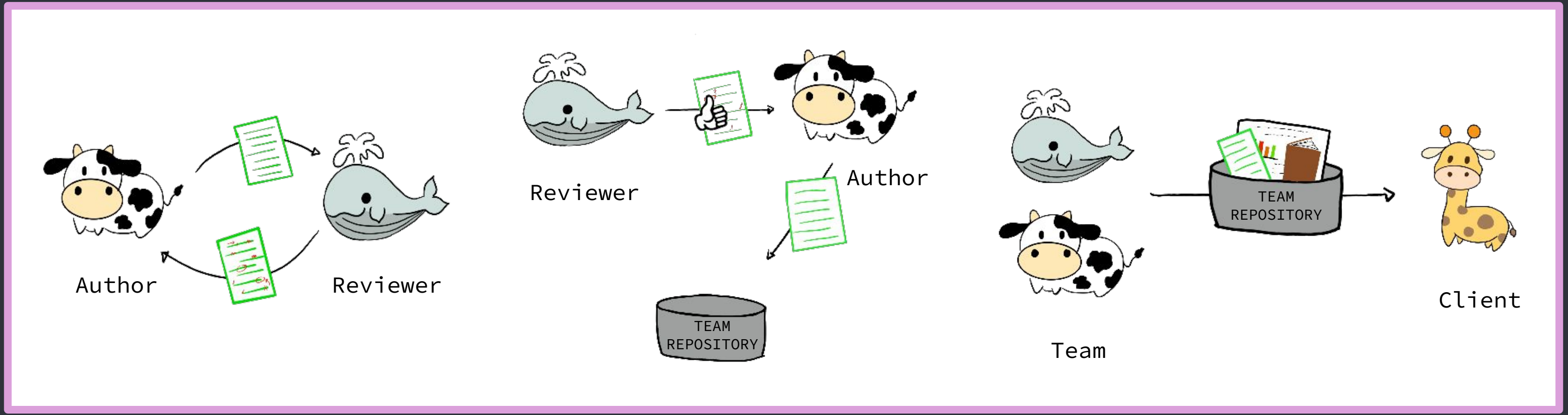
< Manual/systematic line-by-line code inspection >

< Conducted by peers; software exists as well >

< To improve code quality, ensure code adheres to project guidelines, and find bugs >

}

Basic Code Review Process



Code is written by designated author and code review feedback loop between author and reviewer is initiated

Reviewer gives the OK on final review round and code is added to the team's repository

The team provides the completed project along with code and documentation to client for review

documentation.html

code-review.Rmd

Benefits < /1 > {

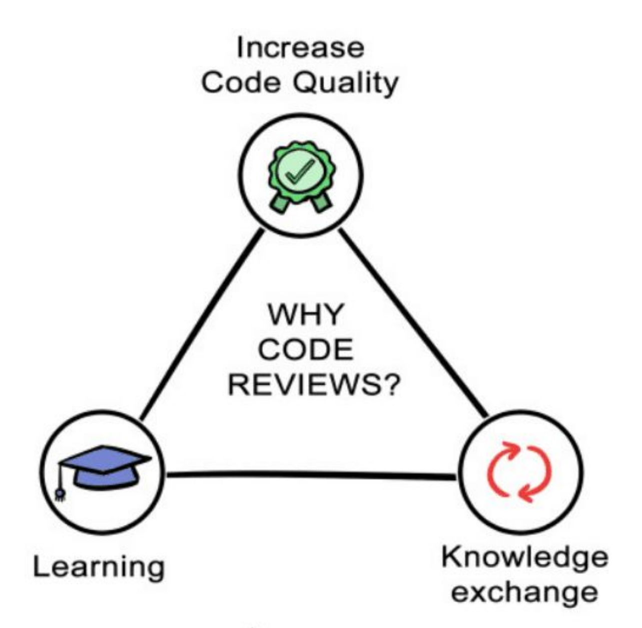
* Consistent feedback
* Knowledge-sharing
* Communication
* Cleaner code

}

Challenges < /2 > {

* Timing
* Establishing process
* Scope/tone of feedback

}



+ Educational Setting

Main Findings in Literature {

< code review research /1 >

* Motivations for conducting code review
* Best practices
* What qualities make for a good code review
* Challenges

< scholarship in educational settings /2 >

* Suggestions for data science curriculum that emphasize communication
* Examples of implementation of code review workflows in courses
* Assessment tools (such as rubrics)
    - Effectiveness
    - Grading

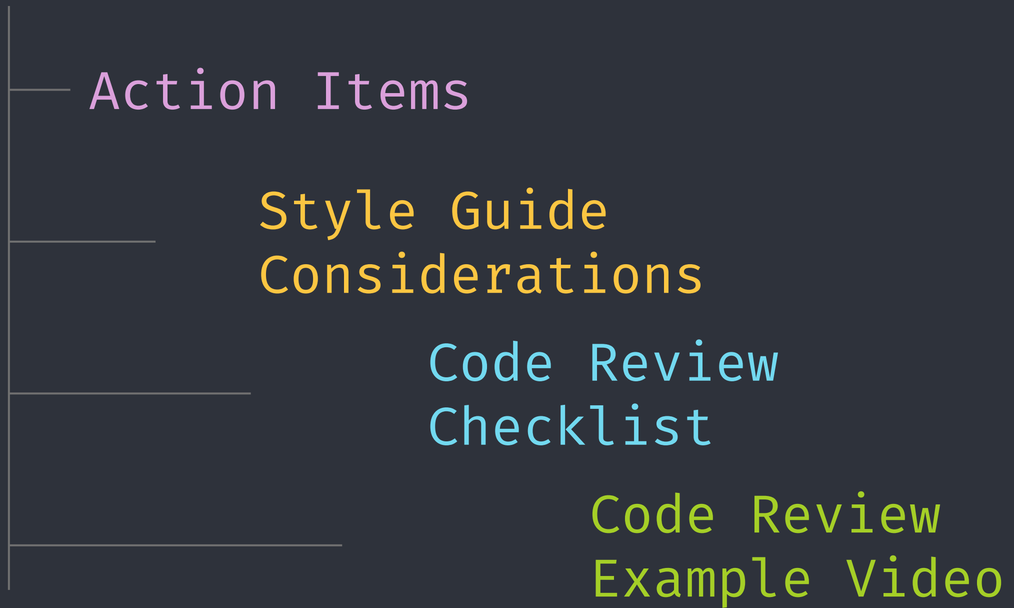
}

Motivation; {

> ‘Existing literature on code review in educational settings has not studied students-designed and driven code review processes.’
>
> - Preferred methods of communication and typical group organization among students has not been documented.
>
> ‘This study hopes to:’
>
> - Garner feedback on newly developed code review resources from both students and instructors
> - Understand how students utilized the resources, their code review processes, and how they communicate via code review

}

Code Review ‘Resources’ {



Action Items

Style Guide Considerations

Code Review Checklist

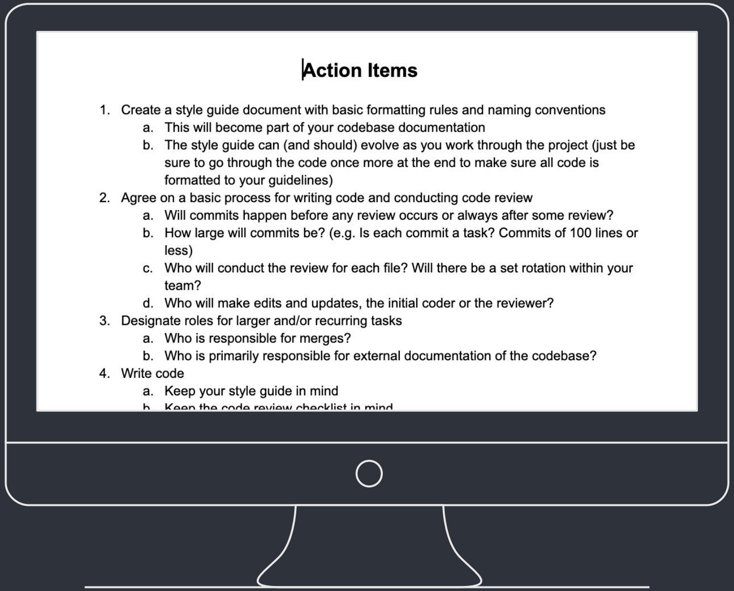
Code Review Example Video

}

Action Items {

Chronological to-do list that provides an overview of the code review process within a typical data science project.

}



Action Items

1. Create a style guide document with basic formatting rules and naming conventions
    a. This will become part of your codebase documentation
    b. The style guide can (and should) evolve as you work through the project (just be sure to go through the code once more at the end to make sure all code is formatted to your guidelines)
2. Agree on a basic process for writing code and conducting code review
    a. Will commits happen before any review occurs or always after some review?
    b. How large will commits be? (e.g. Is each commit a task? Commits of 100 lines or less)
    c. Who will conduct the review for each file? Will there be a set rotation within your team?
    d. Who will make edits and updates, the initial coder or the reviewer?
3. Designate roles for larger and/or recurring tasks
    a. Who is responsible for merges?
    b. Who is primarily responsible for external documentation of the codebase?
4. Write code
    a. Keep your style guide in mind
    b. Keep the code review checklist in mind

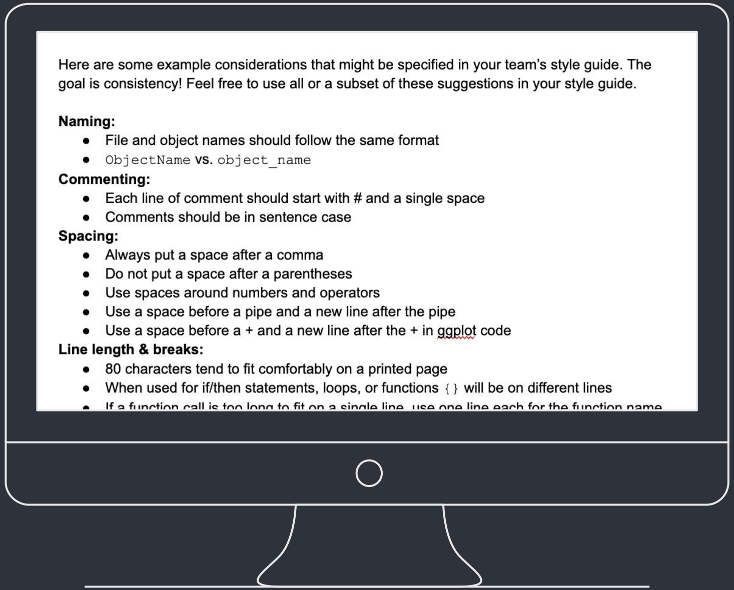
documentation.html

resources.html

Style Guide Considerations {

Brief list of suggestions for student teams who choose to create their own style guides. Suggestions are grouped by naming, commenting, spacing, line length and breaks, indentation, and syntax conventions.

}



Here are some example considerations that might be specified in your team's style guide. The goal is consistency! Feel free to use all or a subset of these suggestions in your style guide.

Naming:

- File and object names should follow the same format
- `ObjectName` vs. `object_name`

Commenting:

- Each line of comment should start with # and a single space
- Comments should be in sentence case

Spacing:

- Always put a space after a comma
- Do not put a space after a parentheses
- Use spaces around numbers and operators
- Use a space before a pipe and a new line after the pipe
- Use a space before a + and a new line after the + in ggplot code

Line length & breaks:

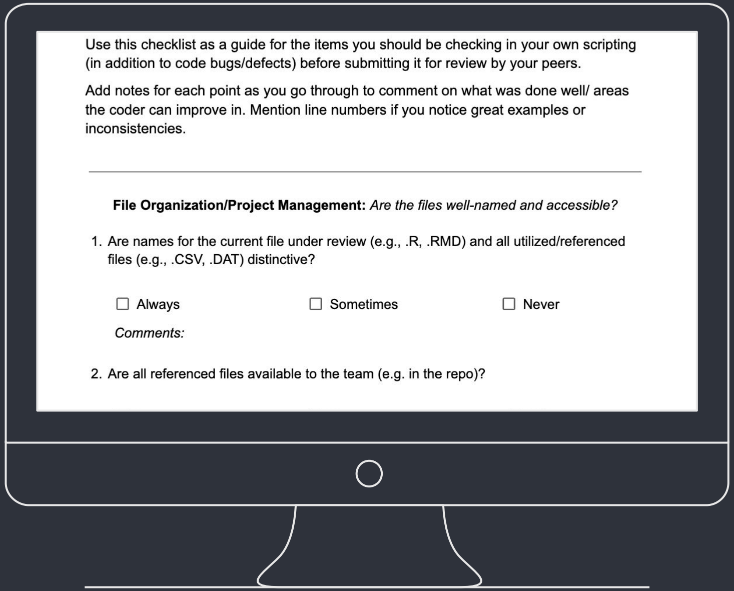
- 80 characters tend to fit comfortably on a printed page
- When used for if/then statements, loops, or functions `{ }` will be on different lines
- If a function call is too long to fit on a single line, use one line each for the function name

Code Review

Code Review Checklist {

Detailed set of questions to use when conducting code review. Provides students with a basic procedure with which to conduct code review and aims to improve their client projects.

}



Use this checklist as a guide for the items you should be checking in your own scripting (in addition to code bugs/defects) before submitting it for review by your peers.

Add notes for each point as you go through to comment on what was done well/ areas the coder can improve in. Mention line numbers if you notice great examples or inconsistencies.

File Organization/Project Management: *Are the files well-named and accessible?*

1. Are names for the current file under review (e.g., .R, .RMD) and all utilized/referenced files (e.g., .CSV, .DAT) distinctive?

   □ Always □ Sometimes □ Never

   Comments:

2. Are all referenced files available to the team (e.g. in the repo)?

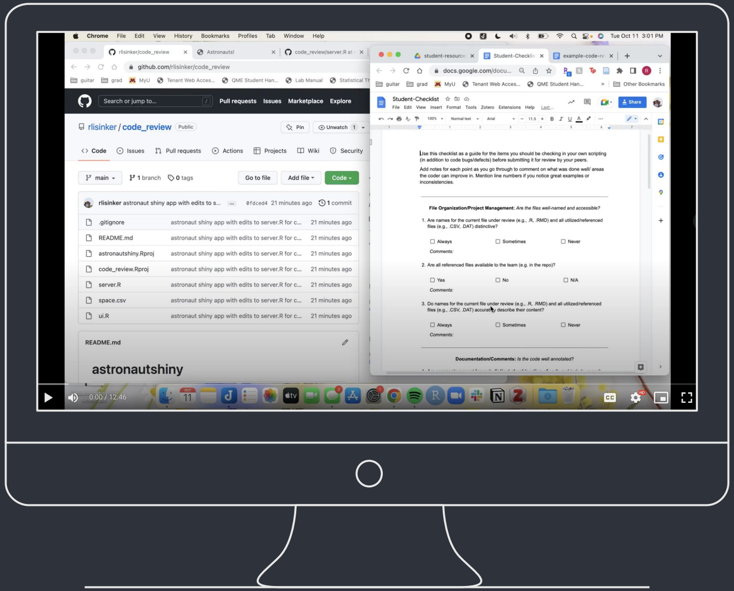
documentation.html

resources.html

Code Review
Example Video {

Screen recording of a completed checklist shown side-by-side with the code being reviewed.

}


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Tue Oct 11 3:01 PM
rlisinker/code_review
Astronauts!
code_review/server.R at r
github.com/rlisinker/code_review
guitar grad MyU Tenant Web Acces... QME Student Han... Lab Manual Statistical Th
Search or jump to...
Pull requests Issues Marketplace Explore
rlisinker / code_review Public
Pin Unwatch 1
Code Issues Pull requests Actions Projects Wiki Security
main 1 branch 0 tags
Go to file Add file Code
student-resource
Student-Checklis
example-code-re
docs.google.com/docu...
Other Bookmarks
Student-Checklist
File Edit View Insert Format Tools Zotero Extensions Help Last...
Share
100% Normal text Arial 11.5
0:00 / 12:46

rlisinker astronaut shiny app with edits to s... 0fdced4 21 minutes ago 1 commit

.gitignore	astronaut shiny app with edits to server.R for c...	21 minutes ago
README.md	astronaut shiny app with edits to server.R for c...	21 minutes ago
astronautshiny.Rproj	astronaut shiny app with edits to server.R for c...	21 minutes ago
code_review.Rproj	astronaut shiny app with edits to server.R for c...	21 minutes ago
server.R	astronaut shiny app with edits to server.R for c...	21 minutes ago
space.csv	astronaut shiny app with edits to server.R for c...	21 minutes ago
ui.R	astronaut shiny app with edits to server.R for c...	21 minutes ago

README.md

astronautshiny

Use this checklist as a guide for the items you should be checking in your own scripting (in addition to code bugs/defects) before submitting it for review by your peers.

Add notes for each point as you go through to comment on what was done well/ areas the coder can improve in. Mention line numbers if you notice great examples or inconsistencies.

File Organization/Project Management: *Are the files well-named and accessible?*

1. Are names for the current file under review (e.g., .R, .RMD) and all utilized/referenced files (e.g., .CSV, .DAT) distinctive?

   □ Always □ Sometimes □ Never

   Comments:

2. Are all referenced files available to the team (e.g. in the repo)?

   □ Yes □ No □ N/A

   Comments:

3. Do names for the current file under review (e.g., .R, .RMD) and all utilized/referenced files (e.g., .CSV, .DAT) accurately describe their content?

   □ Always □ Sometimes □ Never

   Comments:

Documentation/Comments: *Is the code well annotated?*

Code Review

Data Collection {

Insights into students’ code review habits and implementation of code review resources were gained through:

Student Survey:

* Created to understand how the code review intervention and supplementary resources changed how students conduct code reviews.

Faculty Interviews:

* Conducted to provide more detail and corroborate student survey responses.

}

Results

Inconsistent code review “norms”

One student admitted that “when [they] did the code review, everyone had a different strategy to approach the code.”

Documentation was added after the fact

“I think that we wrote more comments after.”

“When we make a new feature, we work on getting it to work and then add comments after”

Results

Defect-finding activity

”top priority was just getting the thing working.”

Faculty involvement matters!

“helped [the team] get organized at the beginning, but after that… was more of a facilitator”

Communicative Growth

“Most growth in [communication]... with the practice from doing this regularly they developed the skills to say ‘this is not working for me in the project right now’”.

reflections.Rmd results.html

Lessons Learned

Students do not conduct code review on their own

Faculty involvement is make or break

Providing structure is helpful

Code review is important!

Thank You!

Questions?
Email me: lisin003@umn.edu