

Reviewing a pull request

Developer quick-sheet · for the maintainer and for reviewers · version 2, 21 October 2026

A pull request (PR) is ready to merge when **the checks are green, the diff holds only what this kind of PR should hold, and a person has read what a check cannot judge**. Who may merge is in `AGENTS.md`: contributors do not merge their own PR; the maintainer may merge their own; Kitty Mickelson may merge a feature PR into `develop` after a review by someone else; release and hotfix PRs into `master` are merged only by the maintainer. **An AI agent never merges without the maintainer's explicit go-ahead.**

1. The frame (one minute)

- **Base.** It is `develop`. Only `release/*` and `hotfix/*` branches target `master`.
- **Checks.** *Student notes* (notes, and the pinned links), *Note pins*, *Library tests* (when `python/` changed) and *Check instructions* (the guides' setup steps on Windows, macOS and Linux). Skipped is fine; failing is not. A job that never got a runner and was cancelled after 15 minutes is not a failure: re-run it.
- **Files changed.** Is there anything the kind of PR in section 2 does not need? Ask why before reading further.
- **Behind develop?** If GitHub says the branch is out of date and there are no conflicts, press **Update branch** (a merge commit) and let the checks run again. If there are conflicts, ask the author, or resolve them in GitHub's editor and say so.

2. What to read, by kind of PR

Kind	Look for
Student note (<code>student-<name></code> , <code>notes/src/</code>)	Only <code>notes/src/*.md</code> , <code>notes/src/img/</code> and the student's own folder under <code>data_analysis/R76/student/</code> . Run <code>python tools/check_note.py notes/src/<note>.md</code> (or read the job summary): the structure and the pinned links must be fine; look at each number listed NOT FOUND (advice only: a value from a paper is fine if the text says so, and it carries <code><!-- numbers-ok --></code>). Read the note: the title is a question, each figure has a bold "Figure N" caption, it says what was <i>not</i> checked. Header: <code>id</code> (section 4), status, and, if it replaces an earlier note, that note's status and link change in the same PR. Sources edited, never the HTML or PDF by hand. The rebuilt pages and PDFs of the changed guides are committed with the sources, and only those (restore the others the build touched). Version and date bumped in the header and <i>Session Info</i> . No text past the margin in the PDF, and the check "PDFs copy as plain text" is green.
Student guide (<code>docs/students/src/</code>)	Tests added; every public function has a <code>@status</code> tag and a docstring; the generated reference page rebuilt (a test fails if it is stale); a breaking change updates the notebooks or marks them deprecated; a changelog line.
Library (<code>python/</code>)	One row in <code>docs/index.html</code> , newest first; every link pinned to a commit (<code>python tools/check_note_pins.py check</code>); a Complete note gets only a minimal correction.
Hand-written note (<code>docs/notes/</code>)	It runs on its own PR; permissions are as small as they can be; it works on all three systems if it checks the guides; a new check for notes goes into <code>tools/check_note.py</code> ; the tool has tests and a changelog line.
CI or tools (<code>.github/</code> , <code>tools/</code>)	Maintainer only. Follow <code>RELEASING.md</code> .
Release or hotfix	

3. Saying it

Comment on the line, one point per comment. Say what to change and why, and for a student point to the step of the guide that explains it. Use **Request changes** only for something that blocks the merge. The author answers in the PR and pushes to the same branch; the PR and its checks update themselves. You may commit a small fix to the author's branch (the **Edit file** menu on the PR); say so in a comment, so that they pull before they edit again.

4. Merge, then check

1. **A student note first needs its number.** The *Student notes* check posts the next free number on the PR as a suggestion on the id: "TBD" line. Press **Commit suggestion**, wait for the checks, then merge (no checkout: `python tools/build_notes.py` next prints the same number). Two note PRs open at once are offered the same number: merge one, then re-run the check of the other. The build refuses a note that still says TBD, and nothing is published.
2. **Merge with a merge commit** (not squash or rebase).
3. **After the merge** the push to develop builds the notes: a bot commit adds the page and its table row, and the page is live in 2 to 3 minutes at <https://villano-lab.github.io/NSDF-Data/>. Open it. `git pull` on develop before more work, because the bot commits there. Guides and quick-sheets are live in about two minutes.
4. **Clean up.** Delete a branch only if you created it and it is merged. A `student-<name>` branch is long-lived: never delete it. Remove any worktree you made for the review.

Tip

An agent can prepare the review, not close it. An agent may read the diff, run the checks, write the comments and fix what a reviewer asks for (Auto-fix does the last for failing checks, conflicts and comments). It does not merge, delete other people's branches or change repository settings without the maintainer's go-ahead.