

Writing a note

NSDF-Data student guide 3 — writing a note, for every computer

Version 24 · 23 October 2026

Tip

This is guide 3 of 5. It is about *writing* the note. Guide 4 covers *publishing* it, and guide 5 covers publishing with an AI assistant.

Tip

What a note is. A note is a short, honest write-up of one question you investigated: what you asked, what you did, what you found, and how sure you are. Notes are how the lab keeps a record that someone else can follow and check. You do not need to be an expert to write one. You do need to be specific.

Before you start

- You have finished the setup guide and the first analysis for your computer.
- You know which question you are answering. If you do not, pick a small one from the project's *Next steps* sections (for example, a number you can reproduce from a note).
- You have a notebook (. ipynb) or a script that produces the numbers and figures you will show. Keep it. The note points to it.

What a note is for

A reader should be able to answer four questions after reading your note:

1. **What was the question?**
2. **What data and code did you use?**
3. **What did you find, and how sure are you?**
4. **What would you do next?**

If your note does not answer these, it is not finished, however long it is.

The parts of a note

Only the header is required. Every other section is a suggestion. Use the sections that help your reader, and leave out the ones that do not apply. A short note with a header and one clear result is complete. A long note with no caveats is not.

Most notes open the same way, so a reader knows where to look. Each section you use becomes a heading, with an outline link.

Section	What goes in it	Length
Title	The question, as a sentence. “Why does X happen in Y?” is better than “Analysis of X.”	one line
Header	Author, date, status, and one-sentence description (see below).	a few lines
Code repository	The notebook and library files you used, each pinned to the commit that produced the results.	a short list
Data and setup	Which series, dump and detector; which quantities; which windows and cuts. Define any word a newcomer would not know.	one or two paragraphs
The question	The question again, and why it matters.	a paragraph

After the question, the note is yours. There is no fixed list of headings from here. Let the story decide the order, and name each heading for what is under it: “What the cut removes” tells a reader more than “Results”. Depending on your work, you might include:

- the figures or tables that answer the question, each with a caption that says what to look at;
- a method, or a check you made along the way, if the reader needs it to trust a number;
- what you think the results mean, and what they do *not* show;
- what could be wrong (a small sample, one channel only, a cut you chose by eye);
- what you would do next, as a short list that the next reader can pick up.

Use one ## heading for each part, and leave out any part that has nothing to say.

Notes come in different kinds, and the kind shapes the headings. A few kinds, and what each one tends to need after the opening:

If your note is...	After the opening it might have...	An example here
A result: a question and its answer	what you did; the figures; what it means and does not show; next steps	Note 7
A check: does this method or cut work?	what was checked and how; the outcome; where it might fail on other data; what to try	Note 4
An explanation: how something is built or used	the idea; how it is organized; a worked example; what is still unfinished	Note 1a

If your note is...	After the opening it might have...	An example here
An investigation of where something is, or why	the method; what was found; what was not checked	Note 8
A log of problems and fixes	one short entry per problem: what happened, and what fixed it	none yet
A brainstorm: ideas, without results	the ideas, clearly marked as not results (give it the status <i>Brainstorm</i>)	none yet

A note can be two kinds at once, and a title does not have to be a question when the note is a check or an explanation. Choose the kind that is closest, and let the headings follow it.

Examples from the site. The notes already there do not share one list of headings. These three go like this:

Note	What it is about	Its headings
Note 5	Choosing a cut for real pulses	Goal; The cut; a check made while writing the note; Status / Next steps
Note 6	A veto that removes pulse-like traces	Motivation; The pulse-shape criterion; What it removes; Effect on the noise PSD; The good_noise archive; Status / Next steps
Note 8	Where the UMN data lives	The question; Method; What Processing.xlsx records; What is actually on cdms3 today; Other users' copies of the k100 pipeline; Reading; Next steps

What they share is the shape, not the headings: each says what was asked, shows what was done and found, and ends by saying what comes next.

Tip

Caveats are strongly suggested. A note that states its limits is trusted more than one that does not, however good the plots look. If you leave caveats out, be sure there is nothing you would want a reader to know about your result.

Where and how to write

Write your note in whatever you are comfortable with. Two routes are supported, and guide 4 takes either one onto the site.

Suggested: VS Code, writing Markdown in the project

[VS Code](#) is a free editor. Here you write the note as a **plain text file ending in .md** inside the NSDF-Data folder, next to the notebook that makes its numbers. It can show a preview of the page as you type, and everything stays in Git on your branch, so every version is saved and changing the note later is just editing the file. This is the route we suggest.

1. Install VS Code, start it, and choose **File**, then **Open Folder**, and pick the NSDF-Data folder.
2. In the file list on the left, open notes/src. Copy `_template.md` (right-click it, **Copy**, then right-click the folder and **Paste**) and rename the copy `your-note-name.md`. Use lowercase letters and hyphens, for example `baseline-in-dump-one.md`.
3. Write. To see the preview in its own tab next to the text, press `Ctrl+K`, let go, then press `V` (on macOS: `Cmd+K`, then `V`). The preview follows as you type and scroll. (`Ctrl+Shift+V` also works, but it replaces the text with the preview in the same tab.)
4. Save each figure as a PNG in `notes/src/img/`, with your note's name at the start of the file name (`baseline-in-dump-one-fig1.png`) and refer to it as `![caption](img/baseline-in-dump-one-fig1.png)`. All notes share that one folder, so the name prefix keeps your pictures from being mixed up with someone else's. Put the picture directly in `img/`, not in a subfolder. How to make the plot and write its caption (bold **Figure 1.** label, what is plotted, what to look for) is in [Making a good plot](#) and [The caption](#).
5. Keep your notebook and any scripts in your own folder under `data_analysis/R76/student/`, and commit as you go (guide 2, Step 7).

Any other plain text editor works for this route too (Notepad, TextEdit set to **Format**, then **Make Plain Text**, or gedit). Save the file as `your-note-name.md`, as UTF-8, and check on Windows that the name does not end in `.md.txt` (File Explorer, **View**, then tick **File name extensions**). The GitHub web editor also works, with no install at all.

Also supported: Word or Google Docs

If you write better in Word or Google Docs, write there. Keep the file anywhere outside the project, for example your Research folder. When it is ready, guide 4 turns it into a note, pictures included, with one command, and the same command updates the note after you change the document. Three habits make the conversion clean:

- **Use Heading styles for section titles.** In Word, select a section title and choose **Home**, then **Heading 1** (main sections) or **Heading 2** (sub-sections). Do not just make the title bold or bigger. Only Heading-styled text becomes a heading, and without headings there is no outline.
- **Avoid underline and superscript.** Underlined text and superscripts (the raised “th” in 5th, for example) do not convert. Use plain text instead.
- **Insert pictures into the document with Insert, then Pictures** (or paste them in), and put the caption in the line under each one (see [The caption](#)). Pictures that are only linked, not embedded, are lost.

Keep code out of the document where you can: link to the notebook instead (see *Pinning your code*). Word turns straight quotes into curly ones, which breaks any code you paste in.

The header

Start your note file with a short block like this. These are the fields the site will read:

```

id: "TBD"
title: "Why does the 1000-sample window see more pulses?"
author: "Your Name"
date: "2026-10-05"
status: "In progress"
keywords: "baseline, pretrigger window, dump 1"
description: "One sentence: what you did and what you found."
---
```

Leave id as TBD. The reviewer gives your note its number (the next free one, for example 8) before it is published. Notes have one continuous set of numbers, whoever wrote them.

Status is one of: *In progress* (you are still working on it), *Complete* (the question is answered and you are not going to change it), or *Superseded (reference/link)* (a later note has taken its place; this one stays up for reference and links to the newer note). New notes start as *In progress*.

You do not type the date you finished. When the note is published with the status *Complete*, the front page shows *Complete (date)*, the day the note was first published as *Complete*, taken from Git. If you reopen a note (back to *In progress*) and publish it as *Complete* again, the page shows *Complete (first date), revised (date)*, and puts a red box *Revised (date)* at the top of the note unless you have already put a box there that starts with **Revised.**, **Correction.** or **Update.** Say in that box what changed.

If your note replaces an earlier note

Sometimes a new note redoes an earlier one with a better method. Then the two notes have to point at each other, and the older note's status has to change. Do this:

1. **In your note**, directly under the header, add one line that names the note you replace and links to it. Pages sit in the same folder, so a link to a hand-written note looks like this (note 3 here), and a link to another student note is `student-their-note-name.html`:

```
**Supersedes:** [Note 3](note-03-noise-psd.html)
```
2. **In the pull request description** (guide 4), write `Supersedes Note 3` so your lead sees it at once.
3. **The older note** then changes to the status *Superseded (reference/link)* and gets a link to yours. Your lead does this for a hand-written note, in the same pull request, as an extra commit. For another student note, change `status:` in its header in `notes/src/` yourself, add a line

```
**Superseded by:** [your note](student-your-note-name.html)
```

 under its header, and include that file in your pull request.

Do not delete the older note. It stays up for reference.

Writing the body

- **Say what you did in order**, with enough detail that someone could repeat it: the file, the detector, the window length, the cut.
- **Use the project's words.** A *segment* is a stretch of consecutive events with the same trigger type. A *run* is a whole data-taking period. Do not use *run* for a segment.
- **Define each quantity once**, in the *Data and setup* section, and use the same name every time after. If you call it `bstd` in one place and "baseline noise" in another, the reader will think they are different things.

- **Write short sentences.** Put the number first, then what it means: “The median is 1.53 counts, so a typical trace is flat noise.”
- **Say how sure you are.** “The two groups look the same” is weaker than “the medians are 1.51 and 1.58, from 141 and 38 traces.” Give the counts.
- **Separate what you measured from what you think.** A sentence in *Results* states a number. A sentence in *Reading* says what it might mean.

Tip

If you change your mind, say so. Notes that show a correction are more useful than notes that pretend nothing went wrong. Put the correction in a clearly labelled box, with the date (see *A coloured box*, next).

A coloured box

For a correction, a caveat or anything a reader must not miss, put it in a coloured box. In a Markdown note, write the text between two lines that hold only three colons, with a blank line before and after:

```
::: callout
**Correction, 12 October 2026.** The median in the first table
used 1000 samples; with 500 it is 1.51 counts.
:::
```

Start the box with a bold label such as **Correction**, **Caveat** or **Note**. Most boxes are shaded with a border, and the label tells the reader which kind it is. A box whose label is **Revised**, **Revision**, **Correction**, **Update**, **Warning** or **Caution** is drawn in red, because the reader must not miss it (use it when something in the note changed or is wrong).

Careful

The VS Code preview does not draw the box. It shows the two `:::` lines as plain text. That is expected: the box appears on the published page, not in the preview. Do not delete the `:::` lines.

If you write in Word or Google Docs, a coloured box does not survive the conversion. Use a bold label such as **Correction**. at the start of the paragraph, or add the `:::` lines to the Markdown after converting.

Figures and tables

A figure is often the first thing a reader looks at, and sometimes the only thing. Make each one able to stand on its own.

Making a good plot

1. **One figure, one message.** Decide what you want the reader to see, and make the plot show that. If you have two messages, make two figures, or two panels.
2. **Label both axes with the quantity and its units**, in the form Pulse height (ADC counts), Time (μ s) or Event number. A bare “x” or “counts” is not a label. If an axis is logarithmic, say so in the label or the caption.

3. **Choose the range to show the data.** Do not leave half the plot empty. If you zoom in and cut off data, say so in the caption. Use the same axes on plots the reader will compare.
4. **Make it readable at the size it will be shown.** Text on the figure should be about as large as the text of the note (font size 11 or 12, a figure about 6 inches wide). Hairline lines and tiny markers disappear on the page.
5. **Do not rely on color alone.** Some readers cannot tell red from green, and a printout may be grey. Use different line styles or markers as well, and name each curve ("Detector 0, 12 traces", not "Series 1"). Put the legend where it does not cover data.
6. **Mark what matters.** A dashed line at a threshold, a shaded band for a cut, an arrow at the feature the caption talks about.
7. **Save it from the notebook, not as a screenshot**, so that it can be made again.

A starting point in matplotlib (replace the names with your own arrays and numbers):

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(6, 4))
ax.plot(time_us, trace, lw=1, label="Detector 0, Channel 0")
ax.axhline(threshold, ls="--", color="gray", label="Cut threshold")
ax.set_xlabel("Time (μs)")
ax.set_ylabel("Pulse height (ADC counts)")
ax.legend(frameon=False)
fig.savefig("baseline-in-dump-one-fig1.png", dpi=200, bbox_inches="tight")
```

Save the PNG into the note's image folder (notes/src/img/, named as in *Where and how to write*) or, in a Word document, put it into the document.

The caption

Every figure has a number and a caption directly under it. **Begin the caption with the label in bold, "Figure 1."**, then say, in a sentence or two:

- **what is plotted**, and for which data (series, dump, detector, channel, how many traces);
- **what the lines, colors and symbols mean**, if the plot does not say so itself;
- **what to look for**: where the reader's eye should go.

In Markdown, the label and the caption are the text inside the square brackets:

`! [**Figure 1.** Bstd against event number, dump 1 (1517 traces). Grey: trigger type U`

Number the figures in the order they appear, and refer to each one in the text ("Figure 1 shows...").

In a Word document, type the caption in the line under the picture, and make "Figure 1." bold.

Tables

Use a table when you are comparing numbers, and a figure when you are showing shape.

Formulas

Write a formula in LaTeX between dollar signs: $\Delta t = 1.6\,\mu\mathrm{s}$ inside a sentence, or between double dollar signs on lines of their own for an equation:

$$\Delta t = 1.6\,\mu\mathrm{s}$$

$$S[k] = \frac{2}{\Delta t} \cdot \frac{1}{M} \sum_{m=1}^M |X_m[k]|^2$$

VS Code's Markdown preview and the published page both draw it. Put no space just inside the dollar signs ($\$x\$, not $\$ x \$$), or the text is not read as a formula. The number checker does not look inside formulas, so if a number in a formula is a result, say it in the sentence as well.$

Pinning your code

A note is only useful if the reader can find the code that made its numbers. Link to your notebook and library files at a **specific commit**, not at a branch name such as `develop` or `master`, because branches keep changing:

1. Commit your notebook to your branch (see the first analysis guide).
2. Find the commit's short code, for example `a1b2c3d`.
3. Link to the notebook like this, with your commit in place of `a1b2c3d` and the notebook's path in place of `PATH` (its folder under `data_analysis/R76/student/`, then its file name):

`https://github.com/villano-lab/NSDF-Data/blob/a1b2c3d/PATH`

Say in the note which commit you pinned.

You do not write a second link for the current version. The published note adds one for you: after each pinned link to a file, a small **[latest]** link opens the same file on the `develop` branch as it is today (and, if the file has moved since your commit, at its new place; if it was removed, the link is left out). So the pinned link shows the code that made your numbers and **[latest]** shows where that code is now. It appears only on the published page, not in the VS Code preview, and only for links in the form shown above (full `https://github.com/villano-lab/NSDF-Data/blob/<commit>/...` addresses), so do not shorten them.

Outline and links

You do not write the outline. The published note shows a clickable **Outline** box at the top, with a link to each section, which readers use to jump to what they need. The site builds it from your `##` headings, so use one `##` heading for each section (the template already does). It appears only on the published page, not in the VS Code preview. A note with no `##` headings has no outline.

Checking your note before you submit

Go through this list before you ask for review:

- ☐ The title is a question.
- ☐ Every number in the text can be found in your notebook output (a tool can check this for you, see below).
- ☐ Every figure has a number, a bold label ("Figure 1.") and a caption that says what is plotted and what to look for, and both axes are labelled with units.
- ☐ Every notebook link is pinned to a commit, as a full `https://github.com/villano-lab/NSDF-Data/blob/<commit>/...` address (the published note then adds the **[latest]** link itself).
- ☐ The header is complete.
- ☐ Any section you use says something a reader needs. Empty or filler sections are removed.
- ☐ You have said what you did *not* check, if the note makes a claim.

Checking the numbers with a tool

The command that checks your whole note, `python tools/check_note.py notes/src/your-note-name` includes this check (guide 4, Step 2, says what the rest of it does). To run only the numbers, with your environment active, run this from the NSDF-Data folder, with your own note's file name:

```
python tools/check_note.py notes/src/your-note-name.md --only numbers
```

It reads the notebook your note links to (at the commit you pinned) and looks at each number in your text and captions. For each one it says whether the number is in the notebook's **saved output**, only in its **code**, or **NOT FOUND**. The notebook must have been saved with its outputs, so do not clear them before you commit.

- **Found in the output:** nothing to do.
- **Only in the code:** it is a value you chose (a threshold, a bin width), not a result. Say so in the note.
- **NOT FOUND:** check the number. A typing mistake, a number from an older run of the notebook, a number you worked out by hand, or a number read off a plot all show up here. Fix the note, or add the calculation to the notebook so it prints the number, and run it again. If a number is right and cannot come from the notebook (for example a value from a paper), put `<!-- numbers-ok -->` at the end of its line, and say where it comes from in the text.

The tool is a help, not a judge: it matches numbers at the digits you wrote (0.52 matches 0.5213), and it skips the header, code, links, labels such as “Figure 1”, dates and the reference list. - [] If you wrote next steps, they are concrete. - [] You have read it aloud once, looking for a word a newcomer would not know.

Session Info

Guide version 24, 23 October 2026. Written for every computer. Please tell the project lead where a step was unclear.

Next

When the note is written and checked, go to **student guide 4: publishing a note**.

Where to get help

Anthony Villano, project lead: anthony.villano@ucdenver.edu

When you write, include the note's title, the step or section you are stuck on, and the sentence or number that worries you. Nobody will be annoyed by a question about a note that is not finished.

Key links

- Notes site: <https://villano-lab.github.io/NSDF-Data/>
- Repository: <https://github.com/villano-lab/NSDF-Data>
- Example of a good note with a correction box: <https://villano-lab.github.io/NSDF-Data/notes/note-02a-bstd-vs-time.html>