Guide to “A typical (fictional) paper”

Modern DevOps for Systems Biologists

Jun Allard

A guide to `allardjun/a-typical-paper`, the worked example this course keeps coming back to: a fictional paper's repository carrying a real mistake, made the way real ones are made and caught the way real ones are caught.

> **Where this came from**
>
> Everything below is copied from the presenter notes in `a_typical_paper_generator/README.md`, which `build_history.py` regenerates there on every rebuild. The commit hashes are therefore true of the repository as it stands today, and stop being true the moment its history is rebuilt. `scripts/sync_presenter_notes.py` is what brings them back into step; `--check` says whether they have drifted.

The planted error

The figure is Figure 3, `figures/fig3_forceresponse.png`, drawn from `data/force_response.csv`. Activation against force: points with error bars, solid two-state model curve. When the data are bad, the four points at and above 10 pN peel off the curve into a false plateau.

The bad data survived 29 commits and 57 days, and was fixed before submission.

	Commit	Date	Who	Message
Error created	`7d0cdae`	2023-09-18	Tomás Lindqvist	new data processed – preliminary inspection to get a quick sense, …
Error still live	`15c952f`	2023-09-29	Jun Allard	input

	Commit	Date	Who	Message
Error caught	`3b85158`	2023-11-13	Priya Raghunathan	Priya: before I model the plateau, can we check the September trace…
Error fixed	`1cb035c`	2023-11-14	Tomás Lindqvist	re-process the September force-clamp traces through the curation pi…

The manuscript then grew a Results paragraph explaining the artefact (`664e2ca`), which was deleted again at `ae02107`.

Commit-list pages on github.com

The **Commits** button (the one with the counterclockwise arrow) opens the list. These links jump straight to the pages the demo needs:

Page	What is on it	Link
3	caught, fixed	https://github.com/allardjun/a-typical-paper/commits/main?after=4be4c8bdb13959f8fc716e594bb7befd76c4d9a7+69
4	created, still live	https://github.com/allardjun/a-typical-paper/commits/main?after=4be4c8bdb13959f8fc716e594bb7befd76c4d9a7+104

Page 1 is the default view, which is where the fix and the tail of the history are. Pagination assumes github.com still shows 35 commits per page; if it changes, these links land on the wrong page and the offsets in `build_history.py` need updating.

Sturdier fallback, immune to both pagination and rebuilds — the file's own history, three commits:

https://github.com/allardjun/a-typical-paper/commits/main/data/force_response.csv

The image diff (Swipe / Onion Skin)

github.com renders a changed PNG as a **visual** diff rather than a binary blob. Open a commit that modifies one and the image diff carries three buttons at its top right: **2-up**, **Swipe**, and **Onion Skin**.

The CSV and the figure it feeds change in the *same* commit, so a single page carries both halves of the story: the readable data diff at the top, the image diff below it.

	Commit	What you see
Figure breaks	`7d0cdae`	the high-force points drop off the curve
Figure recovers	`1cb035c`	they come back onto it

Those are the same two commits as in the table above, which is the point of merging them.

2-up puts before and after side by side. **Swipe** gives a draggable divider across a single image. **Onion Skin** fades one into the other on a slider, which is the one that makes four points visibly detach from a curve.

This works only for a *modified* image, not a newly added one, and only for formats github.com can render. Both commits above are modifications, so all three modes are available. This is the reason the figures are committed as PNG rather than PDF: a changed PDF shows up as an unreadable binary diff.

Two commands

```
git log --oneline -- data/force_response.csv     # three commits, whole story
git log -p -- data/force_response.csv            # provenance comment flips, numbers move
```

To show the broken figure: `git checkout 15c952f` and open `figures/fig3_forceresponse.png`, then `git checkout main`.