

Beyond click

Lab note 9 15 Oct 2026 4 min read
written by Claude from this repository's data

Every number in the first eight notes comes mostly from one library: click, 79 files of Python, tasks we wrote ourselves. A fair reader's first objection is the obvious one. Maybe the pack is tuned to click. Maybe the code-graph tools would look better on code they were built for. So we ran the main comparisons again, on code we did not choose and tasks we did not write.

Three new grounds

- **A TypeScript library.** chrono, a natural-language date parser (267 TypeScript source files, 753 tests, jest), at release v2.10.3. Eight tasks of the same kinds as click's (trace the callers, rename, explain, look up, list subclasses, a feature to a spec, raise coverage, change a signature), each graded by hidden tests or fixed answers. Every grader was checked both ways: 0 on the untouched code, 1.0 on a reference solution.
- **Real bug reports.** 30 tasks from SWE-bench Verified (Jimenez et al. 2024; OpenAI 2024): issues filed against django, sympy and other Python projects, each stated as the original issue text, graded by the project's own hidden tests. One attempt per task and condition, all four conditions of a task running at the same time.
- **Deployment configuration.** The YAML repository from note 6, small and large.

The pack on TypeScript

64 runs, Opus 5.5, four per task and condition (`results-ts.jsonl`):

	Change in cost, the pack against plain Claude Code	95% interval
All eight tasks	-7.6%	-12.1% to -3.1%
Multi-step tasks	-6.3%	-12.5% to +0.8%
Quick questions	-9.9%	-13.7% to -5.9%

The pack took fewer requests on all eight tasks: the feature in 10.5 instead of 13.2, the rename in 4.2 instead of 6.0. Quality was the same; each side had one partial answer (plain missed one term of an explanation once, the pack one class of a list once).

One difference from click stands out. On click's quick questions the pack cost 5.5% more on one day and 3.6% less on another, about even (note 1). On TypeScript it saved 9.9% on them. Questions about a typed codebase lean on symbol lookups more, and lsp-plus answers those from the TypeScript server inside the search.

The pack on real bugs

120 attempts, Opus 5.5 at its highest effort, as the harness runs SWE-bench (`swe-bench/2026-10-15-swe`):

Condition	Bugs fixed	Cost of all 30	Requests per bug	Time per bug
Plain Claude Code	29	\$15.01	12.9	213 s
The pack	30	\$13.17	9.9	144 s
The pack with scout and blast	29	\$14.13	10.2	150 s
The pack with Trail-Code	29	\$14.59	11.5	162 s

With one attempt per bug, cost intervals are wide: the pack against plain -12.3% [-32.5%, +11.9%]. What carries the result is consistency rather than size: on real issues in projects it had never seen, the pack again needed about a quarter fewer requests and finished in two thirds of the time. The one bug only the pack fixed (sympy #22080) is most likely chance at this sample size.

Trail-Code, measured where it reports its own SWE-bench results, again cost more than the pack alone: +10.8% [0.0%, +25.5%]. Scout, the definition pointers from note 4, spoke on 20 of the 30 issues, which name functions and classes far more often than our tasks; it still saved nothing (+7.3% against the pack [-6.7%, +20.7%]). That settles scout: it stays out.

Deployment configuration

Note 6 has the details. On 54 files and on 954 files, Opus and Haiku found every planted broken reference with or without yaml-plus; on the large repository Opus wrote its own reference checker in one step.

What holds, and what does not

- **The pack's saving holds** on a second language, on real bug reports and on code it was not built around: fewer requests at equal quality, everywhere we looked. Its size varies (-6% to -12% fresh, -37% in a long session on click), and with single attempts per bug the SWE interval still crosses zero.

- **Trail-Code cost more than the pack alone on every ground** we measured: click fresh, click long, and SWE-bench.
- **The ideas we borrowed did not survive replication.** Scout's small saving on click did not appear on real issues. yaml-plus found nothing a model did not find itself.
- **The router (route) has not been replicated yet.** Its -14% (-17% before worker costs were corrected on 16 Oct) is from click alone; bug reports are mostly neither features nor tests, so SWE-bench is the wrong ground for it, and a set of real feature requests is the right one.

Method & limits

TypeScript: chrono-node v2.10.3, tasks and graders in `ts_repo.py` ; runs with the person's installed plugins switched off and the official TypeScript language server on. SWE-bench: 30 validated SWE-bench Verified tasks, the harness in `pilot/swe.py` of the lab (each condition loads frozen copies of the mods; the official Python language server switched on in every condition, see note 8); one attempt per task and condition, so per-task differences are noise and only totals carry weight. All costs at list prices from transcripts.

Sources

- Jimenez, C. E., et al. (2024). SWE-bench: Can Language Models Resolve Real-World GitHub Issues? *ICLR 2024*.
- OpenAI (2024). Introducing SWE-bench Verified.
- chrono, natural-language date parser, release v2.10.3.