METR 2025 RCT: AI Tools Slowed Experienced Open-Source Devs by 19% — Analysis

> Saved from a study session on 2026-07-08. Source: arXiv 2507.09089v2 — "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity" (Becker, Rush, Barnes & Rein, METR, July 2025).
>
> PDF: Clippings/2507.09089v2.pdf

What the study is

METR ran a **randomized controlled trial** — the first real RCT of AI coding tools on *real* work — between February and June 2025:

- **16 experienced open-source developers**, working on repos they'd contributed to for ~5 years (avg. 1,500 commits). These are big, mature projects: ~23k stars, ~1.1M lines of code, strict review standards.
- They brought **246 real issues** from their own repos — bugs, features, ~2 hours of work each.
- **Each issue was randomized by coin flip**: AI-allowed (mostly Cursor Pro + Claude 3.5/3.7 Sonnet) or AI-disallowed. Devs recorded their screens and reported time spent.

The clever part of the design is *when* things get measured. Before randomization, developers forecast how long each issue would take both with and without AI. That does two jobs: it gives a difficulty measure that's independent of the coin flip, and it captures expectations. The outcome — time to complete the issue — is **fixed before treatment assignment**, so AI can't game the metric. Compare that to earlier studies which measured lines of code or number of PRs — things AI can inflate without any real productivity gain (more verbose code, work split into more PRs).

The headline result

Three numbers, and the gap between them is the whole story:

Who	Expected effect of AI
Economics experts	39% faster
ML experts	38% faster
The developers themselves (before)	24% faster
The developers themselves (*after doing the work*)	20% faster
What actually happened	**19% slower**

Everyone got the *sign* wrong. And the most unsettling row is the fourth one: after spending hours using the AI on real tasks, developers still believed it had sped them up by 20% — while it had actually slowed them down. Interestingly, the devs' forecasts of *task duration* were well-calibrated (correlation ~0.6 with actual times) — they know how long work takes; what's inverted is specifically their belief about AI's contribution.

Where the time went

They hand-labeled 143 hours of screen recordings at ~10-second resolution. With AI allowed, developers spent *less* time actively coding and searching, and instead spent time **prompting the AI, waiting on generations, reviewing AI output** — and notably more time *idle*. The AI didn't remove work; it swapped coding time for AI-babysitting time, and the swap was net negative in this setting.

Why? The factor analysis

They examined 21 possible explanations, grouped into four types (direct productivity loss / experimental artifact / raises human performance / limits AI performance). Five factors had real evidence **for** contributing to slowdown:

1. **Over-optimism** — devs believed AI was helping, so they kept using it past the point of usefulness.
2. **High familiarity** — these devs are experts *in these exact repos*; it's hard for AI to beat someone with all the context already in their head. Slowdown was worse on tasks devs knew best.

3. **Large, complex repos** — AI performs worse in million-line codebases than in greenfield projects.
4. **Low AI reliability** — devs accepted **<44%** of generations; 75% read every line of AI output; 56% often made major cleanups. ~9% of AI-allowed time went just to reviewing/cleaning AI output.
5. **Missing tacit context** — "AI acts like a new contributor": it doesn't know the undocumented constraints, the weird backwards-compat case, which location is the *right* place for an edit.

Six factors had evidence *against* (cheating, dropout, non-frontier models, unfamiliar IDE, etc.), and ten were unclear. The authors are careful: they can't fully rule out experimental artifacts, but the slowdown was robust across many alternative analyses.

The caveat they insist on

This does **not** say "AI doesn't speed up developers." It says: for *experts* working in *codebases they know deeply* with *high quality bars*, early-2025 AI slowed them down. The same paper explicitly says results are consistent with big speedups on greenfield projects or unfamiliar code — and their appendix quotes back this up: devs found AI *most* helpful precisely on tasks they'd never done before ("first time with Git hooks, AI saved me 3 hours").

Open discussion threads

1. **The perception gap** — developers finished the study still believing they'd been sped up 20%. How can a tool slow you down while feeling helpful? (Candidate hypothesis: "trading speed for ease" — effort vs. time.)
2. **Self-mapping** — for a heavy AI-tool user: which side of the familiarity line are you usually on, and do you recognize the review-and-cleanup tax in your own sessions?
3. **Design critique** — 16 developers, issues capped at ~2 hours: what does that exclude?

Revision #2
Created 9 July 2026 11:58:22 by EMB
Updated 10 August 2026 09:56:38 by EMB