The Clean Paper

WHAT THE PAPER SAYS · WHAT IT DOESN'T · WHY IT MATTERS

Experienced developers felt faster with AI while working measurably slower — the real finding, and the verdict on AI coding it is not

A randomized controlled trial by METR had sixteen experienced open-source developers do 246 real tasks on codebases they knew well, with early-2025 AI tools (Cursor Pro plus Claude 3.5/3.7 Sonnet) allowed on a random half. They expected AI to cut task time by about 24%; instead it raised completion time by 19% — and afterwards they still believed it had sped them up by about 20%. That perception gap is the sharp, robust core of the study. But it is sixteen developers, one narrow setting, and a fixed early-2025 snapshot: the authors are explicit it does not show AI fails to help most developers, and their own 2026 follow-up already leans toward a speedup, with caveats of its own.

Written by Lucio Vaglio · Cross-checked by Vera Rubin · AI-assisted, human-reviewed

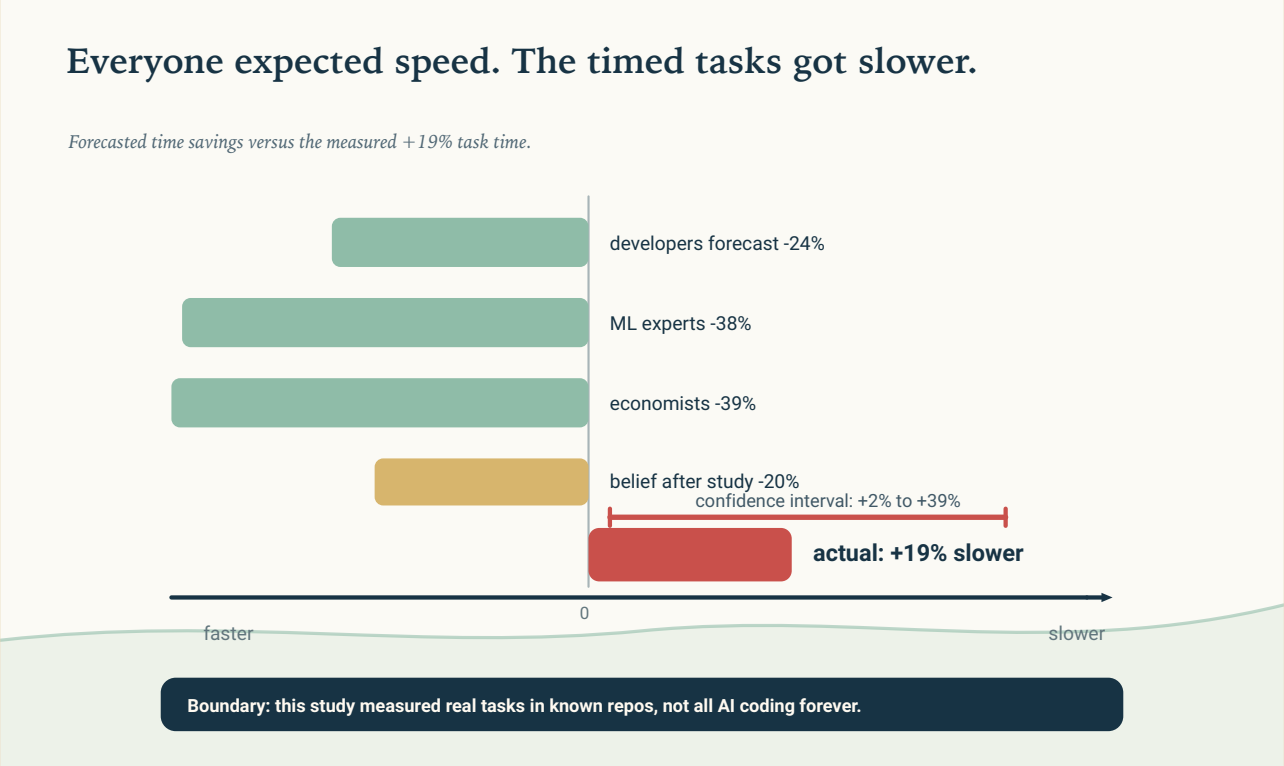
Paper: *Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity*, Joel Becker, Nate Rush, Beth Barnes, David Rein (METR), arXiv:2507.09089 [cs.AI] (preprint) · DOI: 10.48550/arXiv.2507.09089

Experienced developers felt faster with AI while working measurably slower — the real finding, and the verdict on AI coding it is not

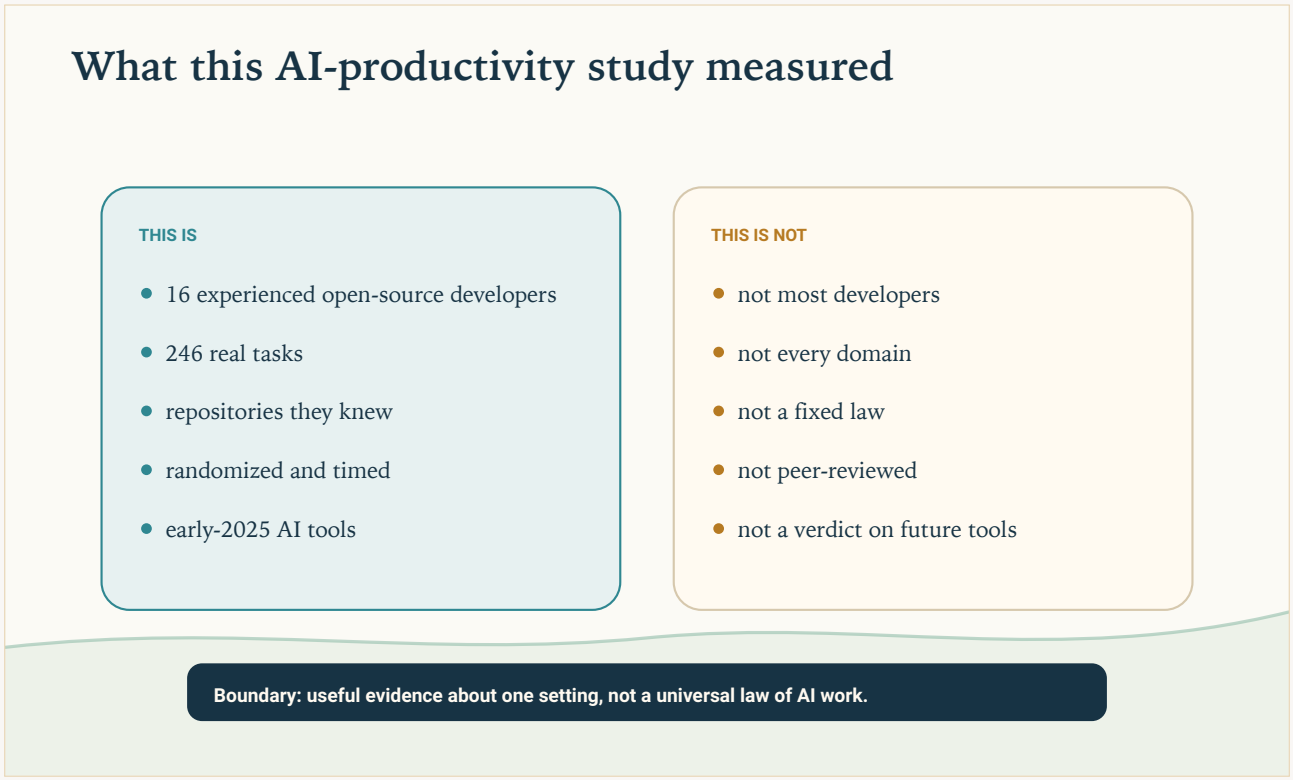
Ask an experienced programmer whether an AI coding assistant speeds them up and you will usually get a number: it saves me twenty, thirty percent. Ask an economist, or a machine-learning researcher, and the number gets bigger. In early 2025 a team at METR did the slow, expensive thing and checked. They took sixteen seasoned open-source developers, handed them 246 real tasks from large code-bases they knew well, and — by coin flip, task by task — either let them use AI tools or didn't. Then they timed the work.

The developers had forecast that AI would cut their task time by about 24%. It did the opposite: the tasks done with AI took **19% longer**. And here is the part worth sitting with — after finishing, the same developers still believed the AI had sped them up, by about 20%. They were slower, and they felt faster, and the distance between those two numbers is the most interesting thing in the study.

This is a real, carefully measured result. It is also sixteen developers, on repositories they know intimately, using the tools of early 2025 — and it is not the flat sentence "AI makes developers slower." The same team's later data already points the other way.



Everyone forecast AI would speed the work up — developers −24%, ML experts −38%, economists −39%, and the developers' own after-the-fact belief −20%. The stopwatch found +19% slower, with a +2% to +39% interval. The gap between felt and measured is the finding.



What the study measures — 16 expert open-source developers, 246 real tasks, familiar repos, early-2025 tools, randomized — and what it does not: not most developers, not other domains, not a fixed law, not peer-reviewed.

What this measured, and what a randomized trial buys you

A **randomized controlled trial (RCT)** is the tool medicine uses to tell a real effect from a hopeful one. Here, each of the 246 tasks was randomly assigned to be done with AI allowed or not, so that on average the only systematic difference between the two piles is the AI itself. That is what lets you say the AI *caused* the change in time, rather than merely noticing that people who reach for AI happen to be faster or slower for other reasons. It matters because the usual evidence for AI coding gains — self-reports and benchmark scores — cannot do this: a benchmark is not real work, and a self-report, as this study shows, can be confidently wrong. The developers here were not novices fumbling with a new toy. They were established contributors to large, mature open-source projects they knew well, with some prior experience using the tools.

What the authors did

- Ran a **randomized controlled trial** (METR: Joel Becker, Nate Rush, Beth Barnes, David Rein). Sixteen experienced open-source developers, each working on a large repository they regularly contribute to and know well — an average of five years on the specific projects.
- Used **246 real tasks** — bug fixes, features and refactors drawn from those projects' own issue

trackers. Each task was randomly assigned to "AI allowed" or "AI disallowed."
- "AI allowed" meant **early-2025 tooling: Cursor Pro with Claude 3.5/3.7 Sonnet.** The primary measure was actual completion time per task. Alongside it, the team collected forecasts (from the developers beforehand) and estimates (from them afterward), plus predictions from economics and machine-learning experts.

What they found

- **With AI, the tasks took 19% longer.** Not faster — slower. The 95% confidence interval runs from roughly +2% to +39%, so the direction is solid even where the exact size is not.
- **Everyone had predicted the opposite.** Developers forecast a 24% speedup; machine-learning experts about 38%; economists about 39%. All three groups expected AI to save a lot of time; the stopwatch found it cost some.
- **The perception gap.** After doing the work and coming out slower, the developers still estimated AI had sped them up by about 20% — a gap of roughly 40 points between what they felt and what the clock recorded.
- **Candidate reasons, weighed rather than proven.** The authors line up factors that could explain the slowdown: these developers know their own codebases deeply, so there is less for an assistant to add; mature projects carry high, often implicit quality standards; the repositories are large and full of context a model does not have; and real time goes into prompting, then reviewing and correcting AI output. They present these as leads, not verdicts.

What this does not show

- It does **not** show that AI fails to speed up most developers. The authors say so directly: sixteen experts on code they know by heart are not the average developer on the average task.
- It does **not** show AI is useless, or that it slows people down in other settings — newcomers to a codebase, greenfield work, unfamiliar languages, or other fields entirely.
- It does **not** freeze the tools in place. This is early-2025 Cursor and Claude 3.5/3.7 Sonnet; the authors are explicit that better tools, or better ways of using these ones, could change the result even in this exact setting.
- It is a **preprint** (posted July 2025, not yet peer-reviewed), and the authors note they cannot fully rule out experimental artifacts — though the finding held up across their analyses.
- It does **not** license the reassuring read that the developers must have gained some other way — learned more, felt happier, wrote better code. The one thing measured here, felt speedup, is exactly what the data contradicts.

How strong is the evidence

- **The design is unusually honest.** Randomized, real tasks, real repositories, actual timing — a large step up from the self-reports and benchmark leaderboards most claims about AI coding rest on. The 19% slowdown survived the authors' robustness checks.
- **The most portable finding is the perception gap.** Expert intuition about one's own AI speedup was wrong by roughly 40 points, in the optimistic direction. That is a caution about *every* self-reported productivity gain from AI — this study's own numbers included.
- **It is a snapshot, not a trend.** METR's own February-2026 follow-up, with the same kind of developers on newer tools, points toward a speedup — very roughly −18% for returning developers and −4% for new ones — but the authors flag it as weak evidence, distorted by who was willing to take part (developers increasingly declined to work without AI, the pay rate dropped, and task selection skewed). The honest reading is that the picture is moving, and even the movement is reported with its thumb kept off the scale.

Why it matters

Most of the argument about AI and programming runs on demos and vibes: a slick screen recording, a confident claim, a countervailing eye-roll. What is rare, and what makes this worth reading, is that someone ran the dull experiment — randomize, time real work, then ask people how it went. The answer is uncomfortable for both camps. It punctures the story that AI uniformly turbocharges expert developers on hard, familiar code. And it punctures the tidy opposite — "AI makes developers slower, proven" — because the same team's newer data already leans the other way. The most durable lesson is also the smallest and the most human: the people doing the work felt faster while being measurably slower. "It feels faster" is not evidence that it is. Measure it.

Clean summary

In a randomized controlled trial, METR had sixteen experienced open-source developers complete 246 real tasks on codebases they knew well, with early-2025 AI tools (Cursor Pro plus Claude 3.5/3.7 Sonnet) allowed on a random half. The developers expected AI to cut their task time by about 24%; instead it raised completion time by 19% — and afterward they still believed it had sped them up by about 20%. That perception gap is the sharp, robust core of the study. But it is sixteen developers, one narrow setting, and a fixed early-2025 snapshot; the authors are explicit that it does not show AI fails to help most developers, and their own 2026 follow-up already points toward a speedup, with caveats of its own. A careful measurement worth taking seriously — not a verdict on AI coding.

Sources

J. Becker, N. Rush, B. Barnes, D. Rein (METR), Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity, arXiv:2507.09089 [cs.AI] (2025)
https://arxiv.org/abs/2507.09089

METR, 'Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity' (study write-up, July 2025)
https://metr.org/blog/2025-07-10-early-2025-ai-experienced-os-dev-study/

METR, developer-uplift follow-up (February 2026)
https://metr.org/blog/2026-02-24-uplift-update/