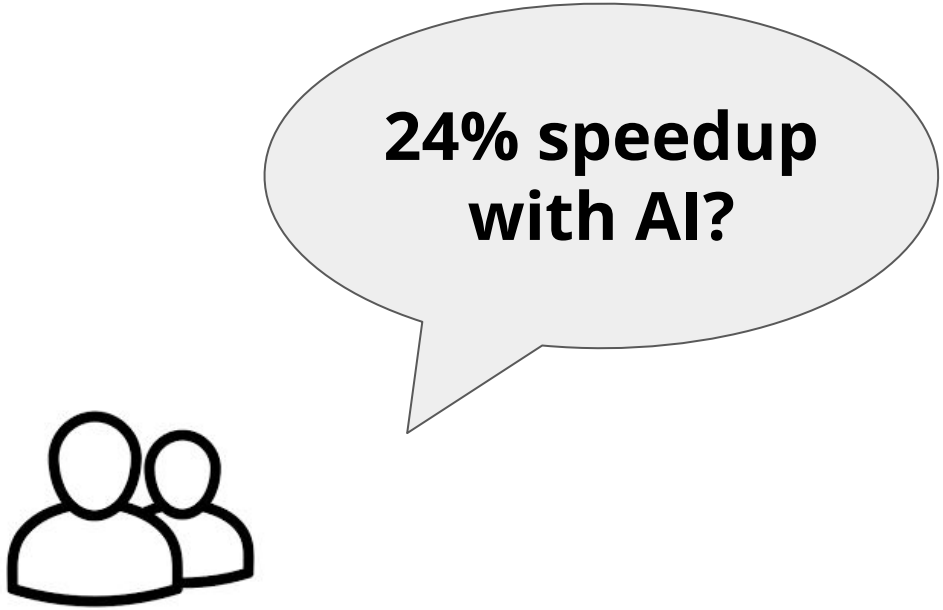
Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity

Joel Becker*, Nate Rush*, Beth Barnes, David Rein
Model Evaluation & Threat Research (MSTR)
July 2025

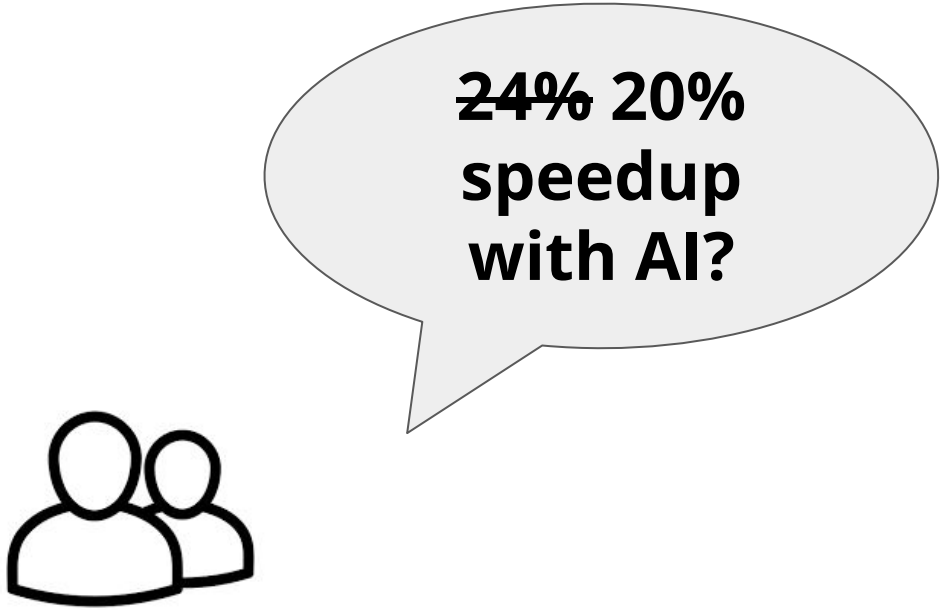
16 Developers

246 Tasks

Before task



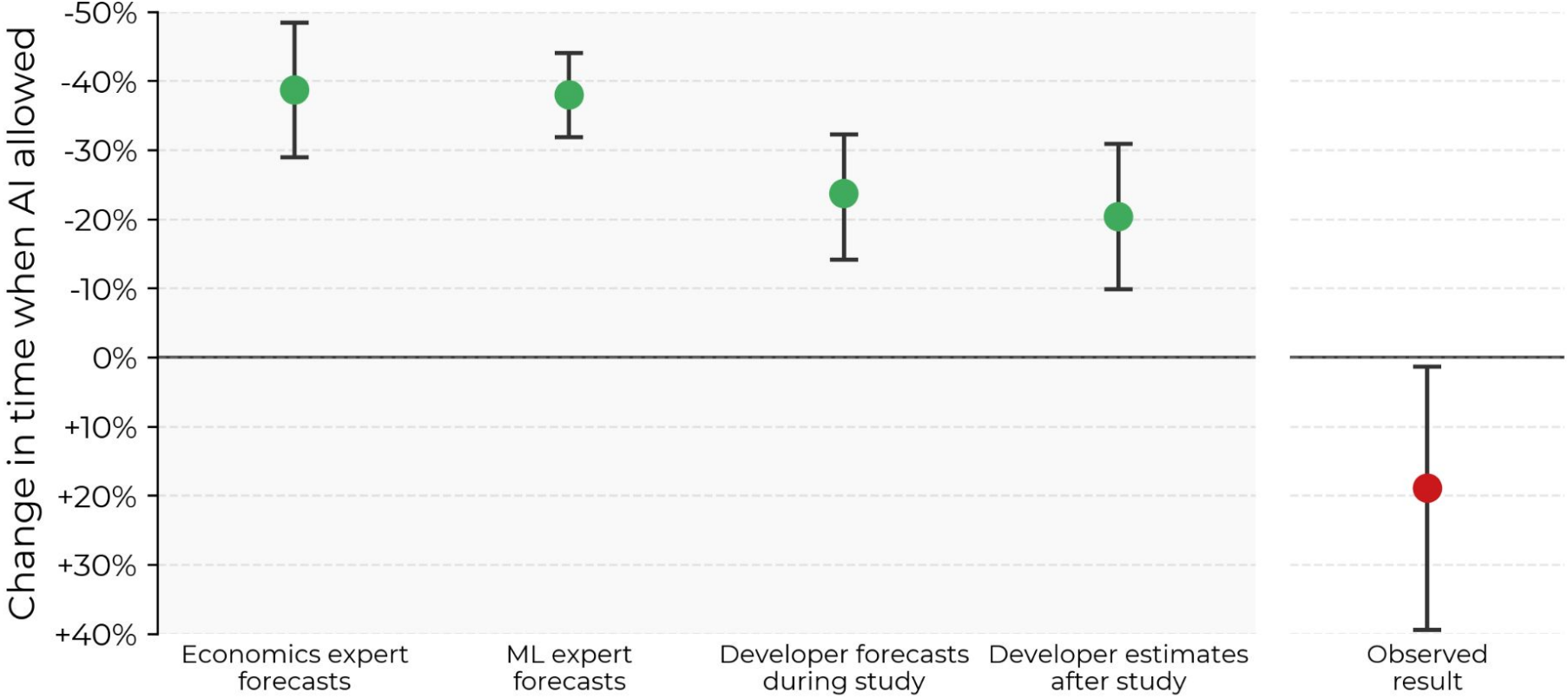
After task



“AI actually increases completion time by 19% —

AI tooling slowed developers down!”

Experts and Study Participants Misjudge AI Speedup



How?

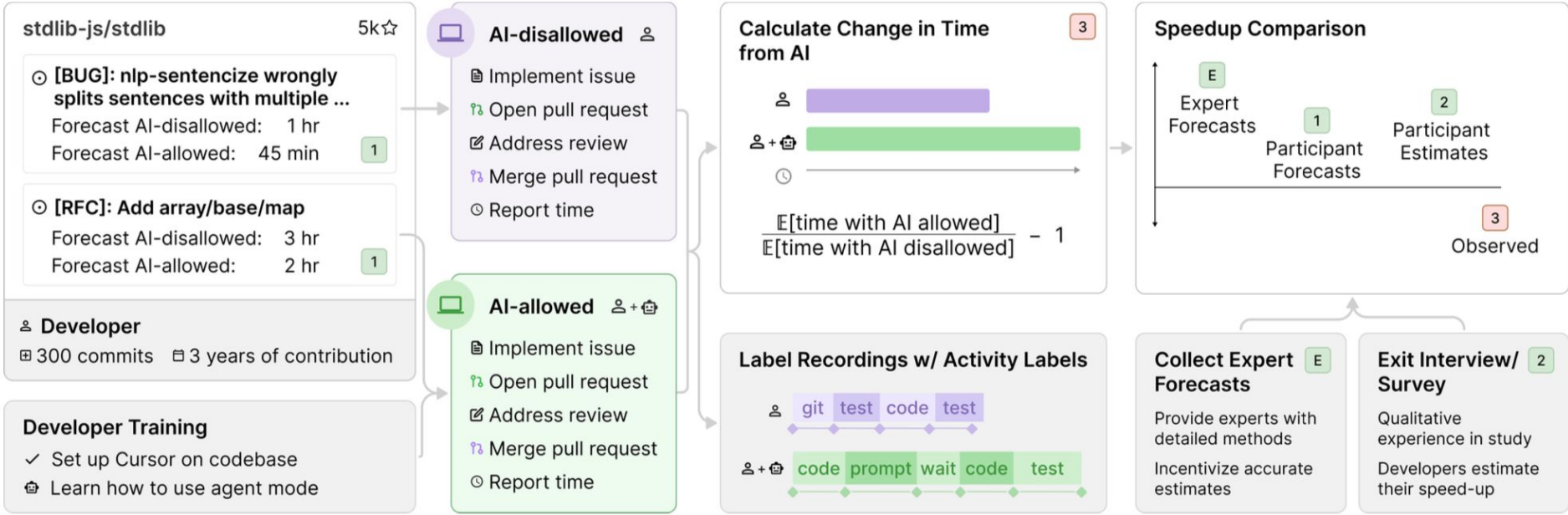


Figure 2: *Our experimental design. Tasks (referred to as issues) are defined before treatment assignment, screen recordings let us verify compliance (and provide a rich source of data for analysis), and forecasts from experts and developers help us measure the gap between expectations and observed results.*

Developers & Repositories

Experienced developers —
Typically 10+ y experience

Regular contributors to the repo —
Average 5y experience (59% of the repo's lifetime)
Have made 1500+ commits

Developers & Repositories

Experienced developers —
Typically 10+ y experience

Regular contributors to the repo —
Average 5y experience (59% of the repo's lifetime)
Have made 1500+ commits

Large and mature —

Average 23,000 stars,
1,100,000 LoC,
4,900 forks,
20,000 commits,
and 710 committers

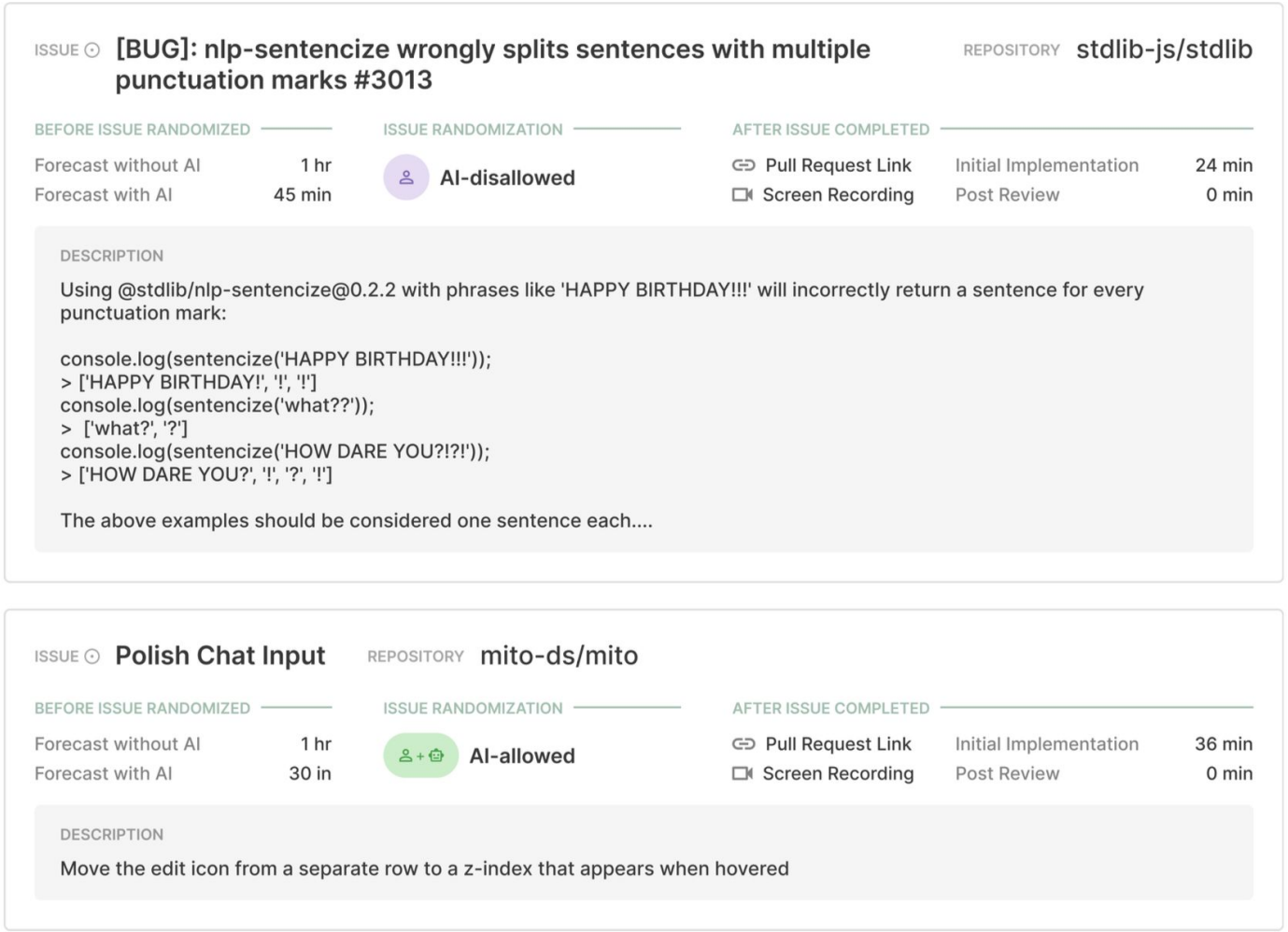


Figure 3: *Real issues completed during the study from the* `stdlibjs` *and* `mito` *repositories*

Experimental Design

Where do the tasks come from?

Each developer provides a list of real issues (bug reports, feature requests, or work items used to coordinate development) in their repo to work on.

They range from brief problem descriptions to detailed analyses and represent work ranging from minutes to hours (max 2h).

Experimental Design

Developers forecast how long each issue would take if they were to complete it both with and without AI assistance.

The forecasts are used as a proxy for **issue difficulty**, and to measure **per-issue speedup anticipated by the developer**.

These issues are then randomized to AI or no-AI via a simulated fair coin flip.

Experimental Design

What do AI and no-AI mean?

If AI is allowed, developers can use **any AI tools or models** they choose, including ***no AI tooling if they expect it to not be helpful***.

If AI is not allowed, no generative AI tooling can be used, including auto-completion. **Search engines, which sometimes use AI under the hood, remain allowed in the AI-disallowed condition.**

Experimental Design

Developers work on the assigned issues in their preferred order (can also work on multiple issues at a time).

-> Submit a PR

-> PR reviewed, make changes if needed

-> Merge

-> Self-report how long it took them on the issue before and after the PR

Data Collection

Developers work on the assigned issues in their preferred order (can also work on multiple issues at a time).

-> Submit a PR

-> PR reviewed, make changes if needed

-> Merge

-> Self-report how long it took them on the issue before and after the PR

-> Typically record their screen as they work

-> Reported that they were not significantly inconvenienced or affected by the differences of the IDE compared to their normal workflows

AI Tools and Training

93% have prior experience with tools like ChatGPT,
but only 44% have experience using Cursor

AI Tools and Training

We provide developers with Cursor Pro subscriptions and conduct live basic training, validating that developers are able to prompt Cursor effectively to edit files in their own codebase, accept changes, and revert to previous checkpoints.

However, we don't require that they use Cursor specifically.

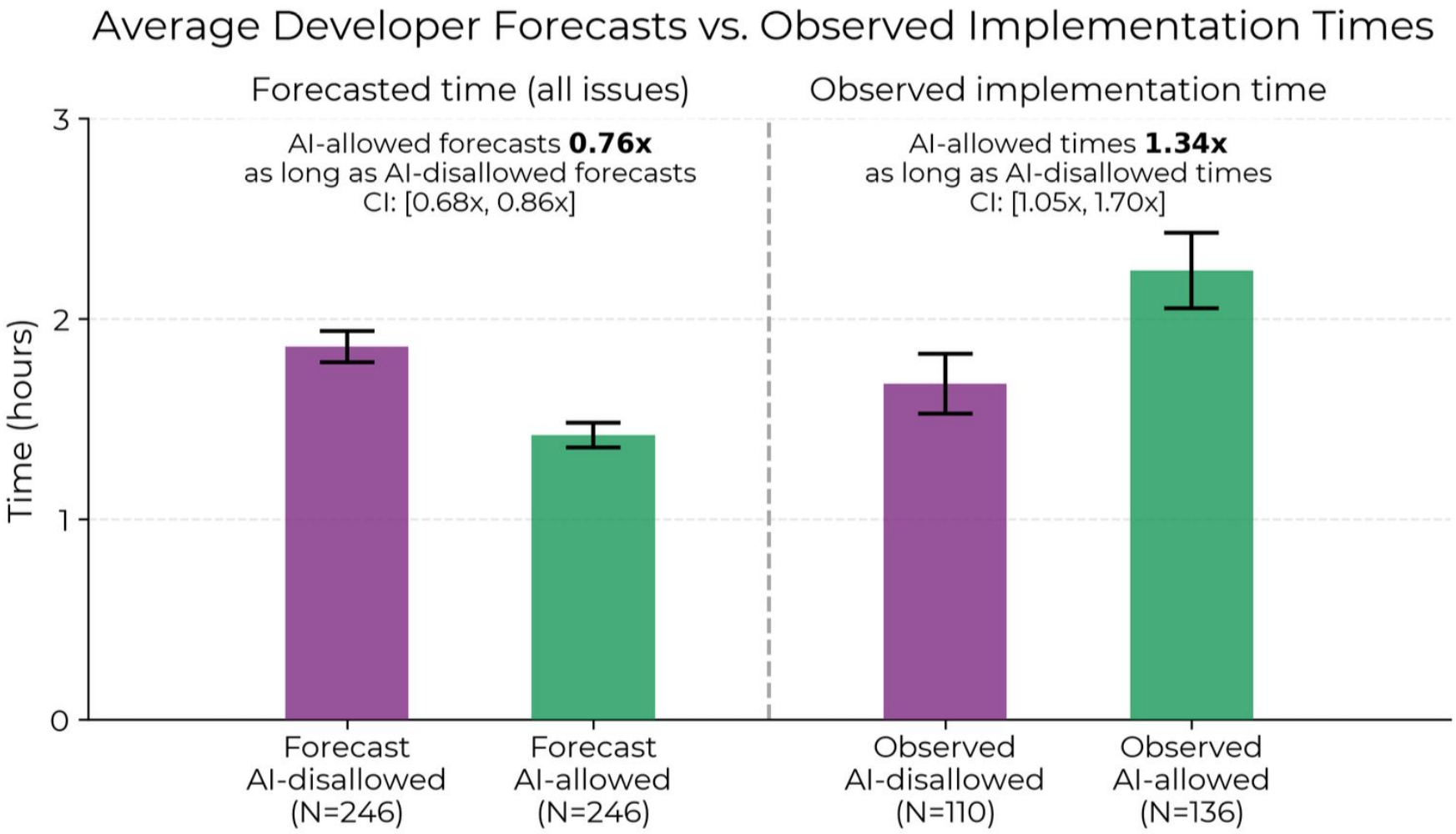
Effect Estimation

$$S = \frac{\mathbb{E}[\text{completion time with AI allowed}]}{\mathbb{E}[\text{completion time with AI disallowed}]} - 1$$

-50% means AI-allowed issues take half the time of AI-disallowed; -> **Speedup!**

0% means they take the same time;

100% means AI-allowed issues take twice as long. -> **Slowdown!**


Average Developer Forecasts vs. Observed Implementation Times
Forecasted time (all issues)
Observed implementation time
AI-allowed forecasts 0.76x
as long as AI-disallowed forecasts
CI: [0.68x, 0.86x]
AI-allowed times 1.34x
as long as AI-disallowed times
CI: [1.05x, 1.70x]
Time (hours)
3
2
1
0
Forecast AI-disallowed (N=246)
Forecast AI-allowed (N=246)
Observed AI-disallowed (N=110)
Observed AI-allowed (N=136)

Why?

Activity Labels

Manually labeled valid recordings for 74 issues on the activities that the developers engage in while they work.

Activity Labels - with AI

Manually labeled valid recordings for 74 issues on the activities that the developers engage in while they work.

When allowed to use AI, developers spend a **smaller proportion of their time actively coding and reading/searching for information**. Instead, they spend time reviewing AI outputs, prompting AI systems, and waiting for AI generations.

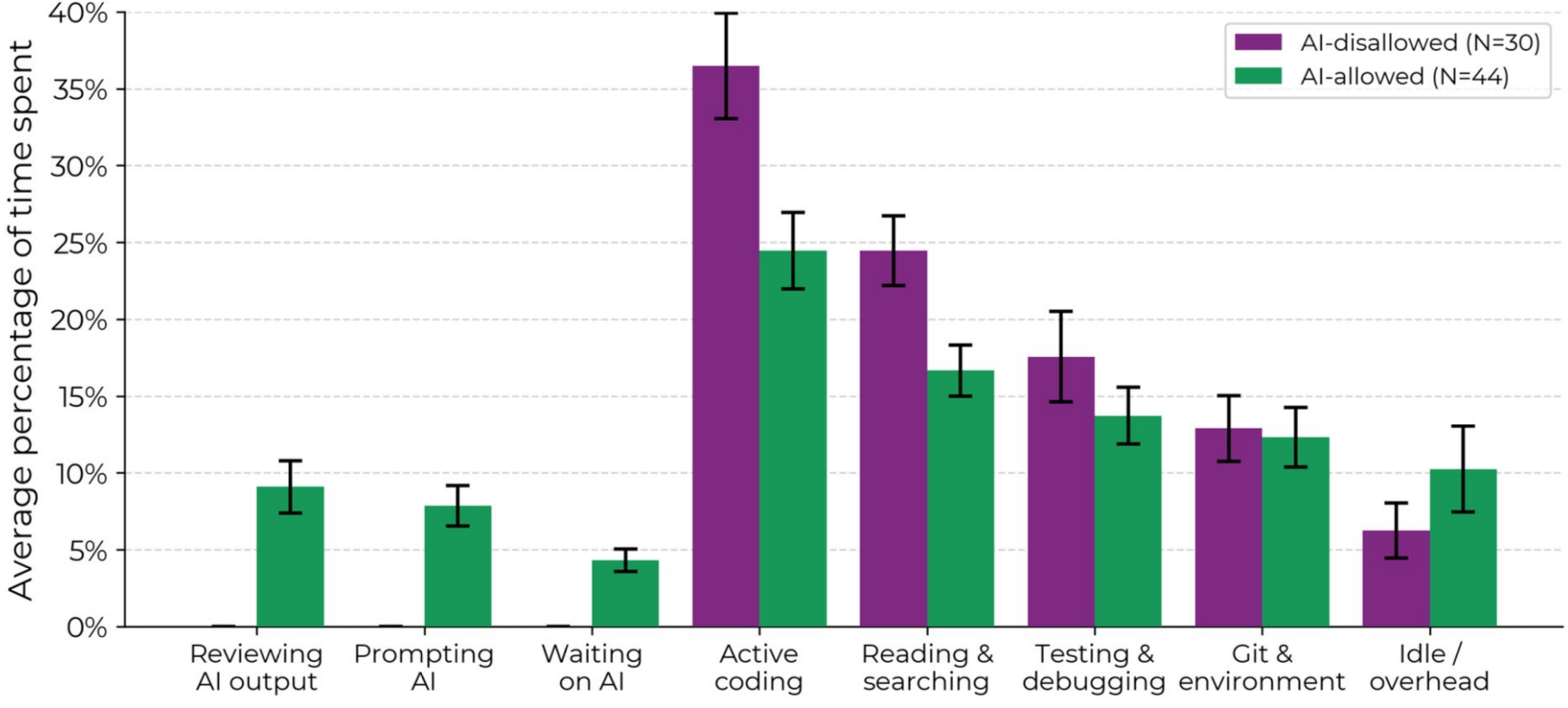
Activity Labels - with AI

Manually labeled valid recordings for 74 issues on the activities that the developers engage in while they work.

When allowed to use AI, developers spend a smaller proportion of their time actively coding and reading/searching for information. Instead, they spend time reviewing AI outputs, prompting AI systems, and waiting for AI generations.

They also spend a somewhat higher proportion of their time idle, where their screen recording doesn't show any activity.

Average Percentage of Time Spent per Activity Among Labeled Screen Recordings



Factors likely to contribute to slowdown

Factor	Type	Relevant Observations
Over-optimism about AI usefulness (C.1.1)	⏳	• Developers forecast AI will decrease implementation time by 24% • Developers post hoc estimate AI decreased implementation time by 20%
High developer familiarity with repositories (C.1.2)	👤	• Developers slowed down more on issues they are more familiar with • Developers report that their experience makes it difficult for AI to help them • Developers average 5 years experience and 1,500 commits on repositories
Large and complex repositories (C.1.3)	🤖	• Developers report AI performs worse in large and complex environments • Repositories average 10 years old with >1,100,000 lines of code
Low AI reliability (C.1.4)	🤖	• Developers accept <44% of AI generations • Majority report making major changes to clean up AI code • 9% of time spent reviewing/cleaning AI outputs
Implicit repository context (C.1.5)	🤖👤	• Developers report AI doesn't utilize important tacit knowledge or context

Factors likely to contribute to slowdown

Overuse AI?

Factor	Type	Relevant Observations
Over-optimism about AI usefulness (C.1.1)		• Developers forecast AI will decrease implementation time by 24% • Developers post hoc estimate AI decreased implementation time by 20%
High developer familiarity with repositories (C.1.2)		• Developers slowed down more on issues they are more familiar with • Developers report that their experience makes it difficult for AI to help them • Developers average 5 years experience and 1,500 commits on repositories
Large and complex repositories (C.1.3)		• Developers report AI performs worse in large and complex environments • Repositories average 10 years old with >1,100,000 lines of code
Low AI reliability (C.1.4)		• Developers accept <44% of AI generations • Majority report making major changes to clean up AI code • 9% of time spent reviewing/cleaning AI outputs
Implicit repository context (C.1.5)		• Developers report AI doesn't utilize important tacit knowledge or context

Factors likely to contribute to slowdown

Factor	Type	Relevant Observations
Over-optimism about AI usefulness (C.1.1)	⏳	• Developers forecast AI will decrease implementation time by 24% • Developers post hoc estimate AI decreased implementation time by 20%
High developer familiarity with repositories (C.1.2)	👤	• Developers slowed down more on issues they are more familiar with • Developers report that their experience makes it difficult for AI to help them • Developers average 5 years experience and 1,500 commits on repositories
Large and complex repositories (C.1.3)	🤖	• Developers report AI performs worse in large and complex environments • Repositories average 10 years old with >1,100,000 lines of code
Low AI reliability (C.1.4)	🤖	• Developers accept <44% of AI generations • Majority report making major changes to clean up AI code • 9% of time spent reviewing/cleaning AI outputs
Implicit repository context (C.1.5)	🤖👤	• Developers report AI doesn't utilize important tacit knowledge or context

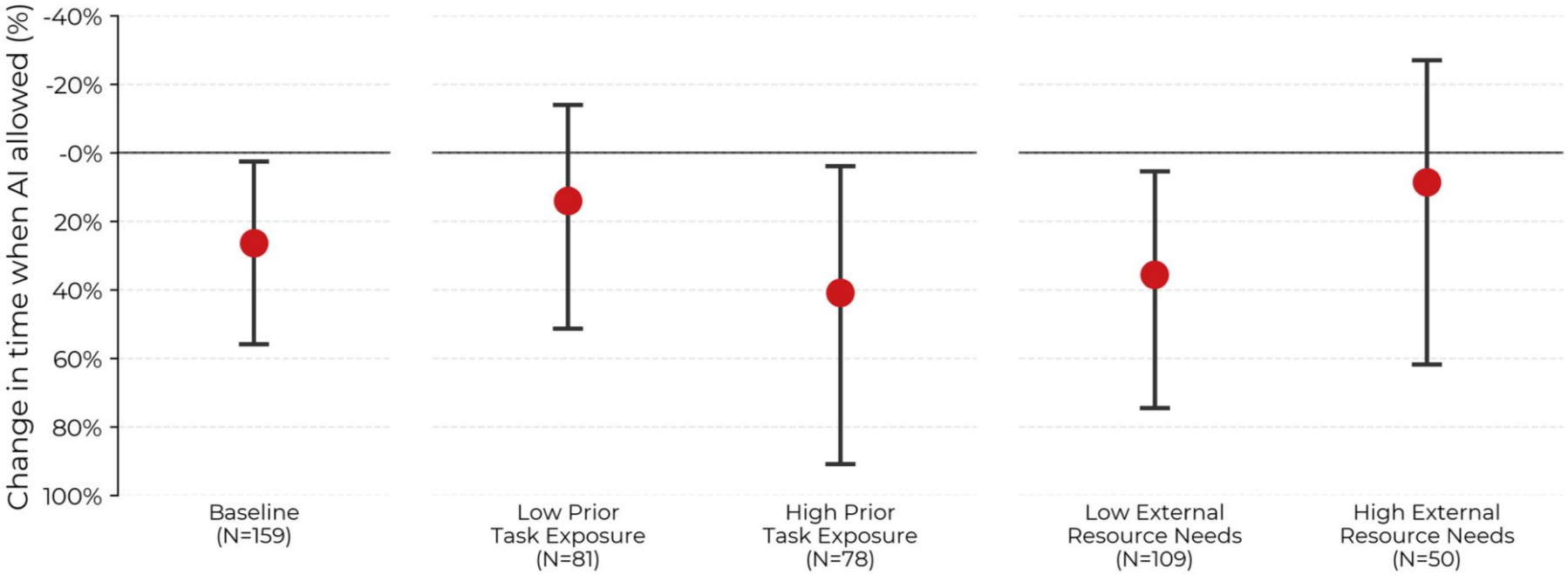


Figure 7: *Developers are slowed down more on issues where they self-report having significant prior task exposure, and on issues where they self-report having low external resource needs (e.g. documentation, reference materials). We only collected this data for the latter half of issues completed in the study—this is why we have a smaller number of issues for our baseline slowdown estimate. See Section D.4 for details on how we estimate heterogeneous treatment effects.*

Factors likely to contribute to slowdown

Factor	Type	Relevant Observations
Over-optimism about AI usefulness (C.1.1)	⏳	• Developers forecast AI will decrease implementation time by 24% • Developers post hoc estimate AI decreased implementation time by 20%
High developer familiarity with repositories (C.1.2)	👤	• Developers slowed down more on issues they are more familiar with • Developers report that their experience makes it difficult for AI to help them • Developers average 5 years experience and 1,500 commits on repositories
Large and complex repositories (C.1.3)	🤖	• Developers report AI performs worse in large and complex environments • Repositories average 10 years old with >1,100,000 lines of code
Low AI reliability (C.1.4)	🤖	• Developers accept <44% of AI generations • Majority report making major changes to clean up AI code • 9% of time spent reviewing/cleaning AI outputs
Implicit repository context (C.1.5)	🤖👤	• Developers report AI doesn't utilize important tacit knowledge or context

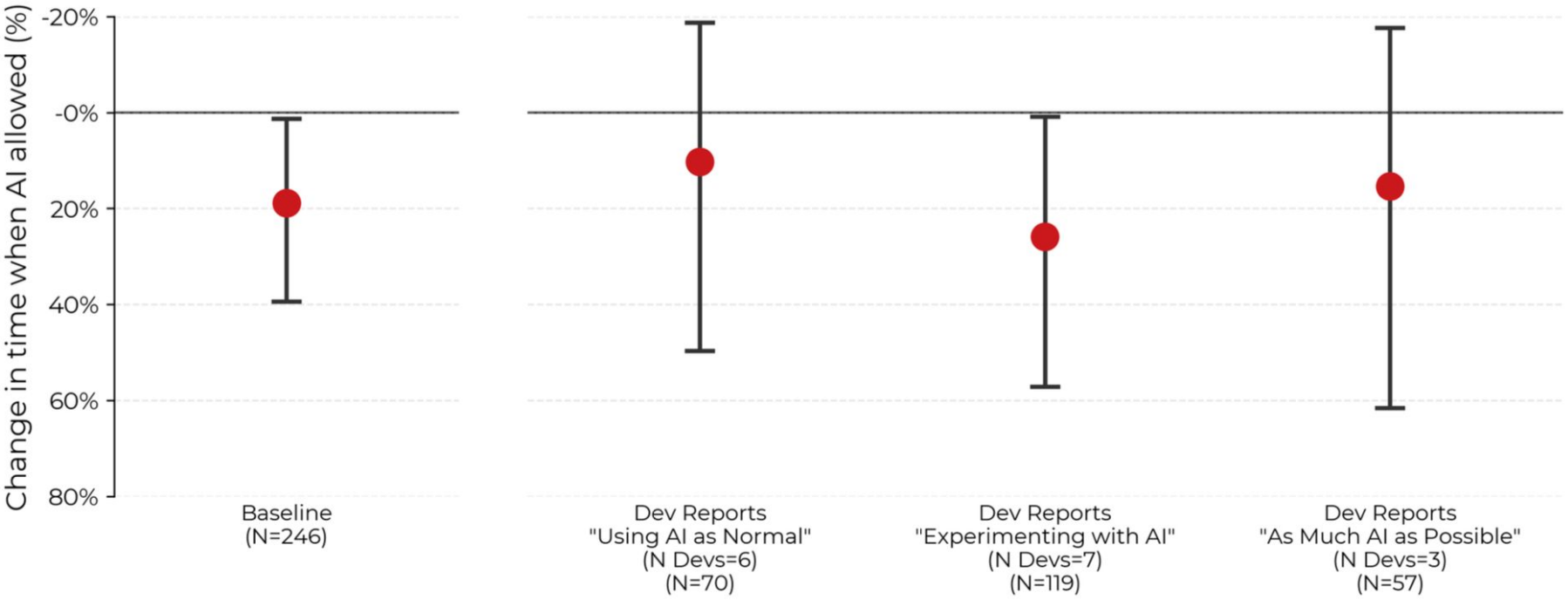


Figure 8: *Developers who report that they were experimenting with AI or using AI as much as possible see greater slowdown than developers who report using AI as they normally would. See Section D.4 for details on how we estimate heterogeneous treatment effects.*

Other factors that don’t contribute to the slowdown in this setting

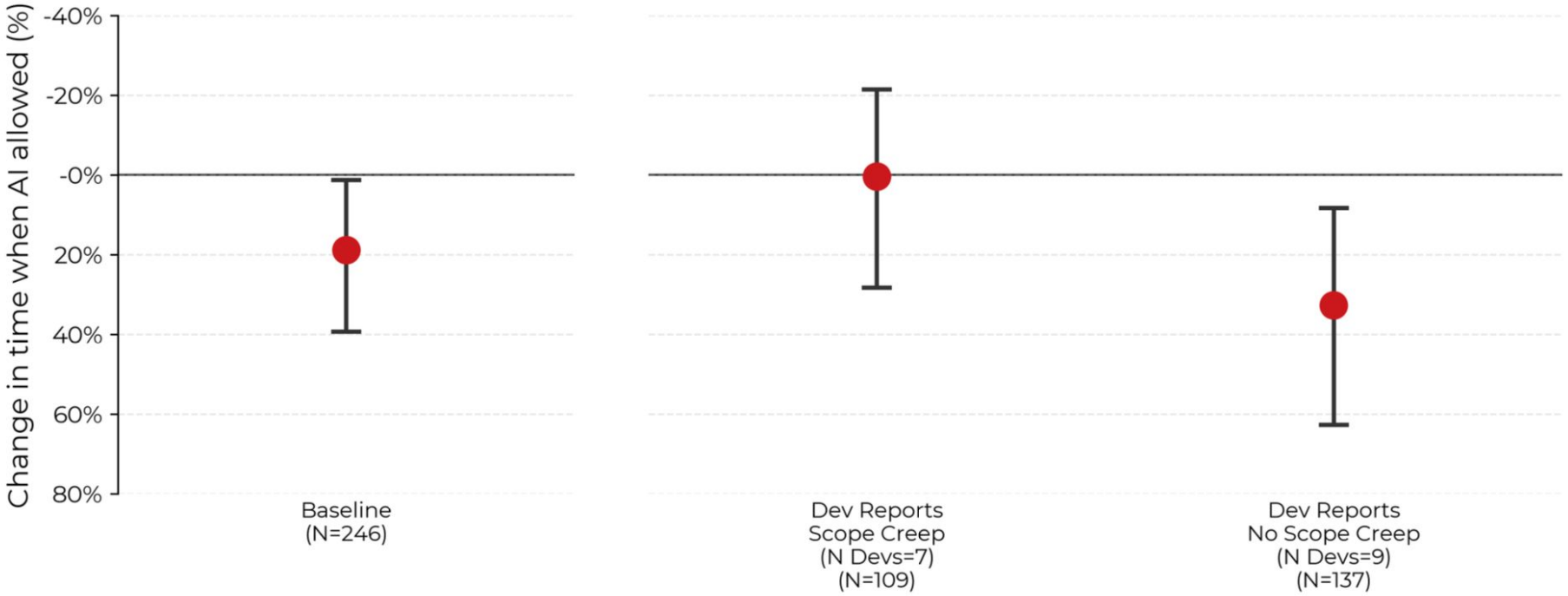


Figure 9: *Speedup on issues broken down by whether the developer completing the issue reports scope creep when using AI. See Section D.4 for details on how we estimate heterogeneous treatment effects.*

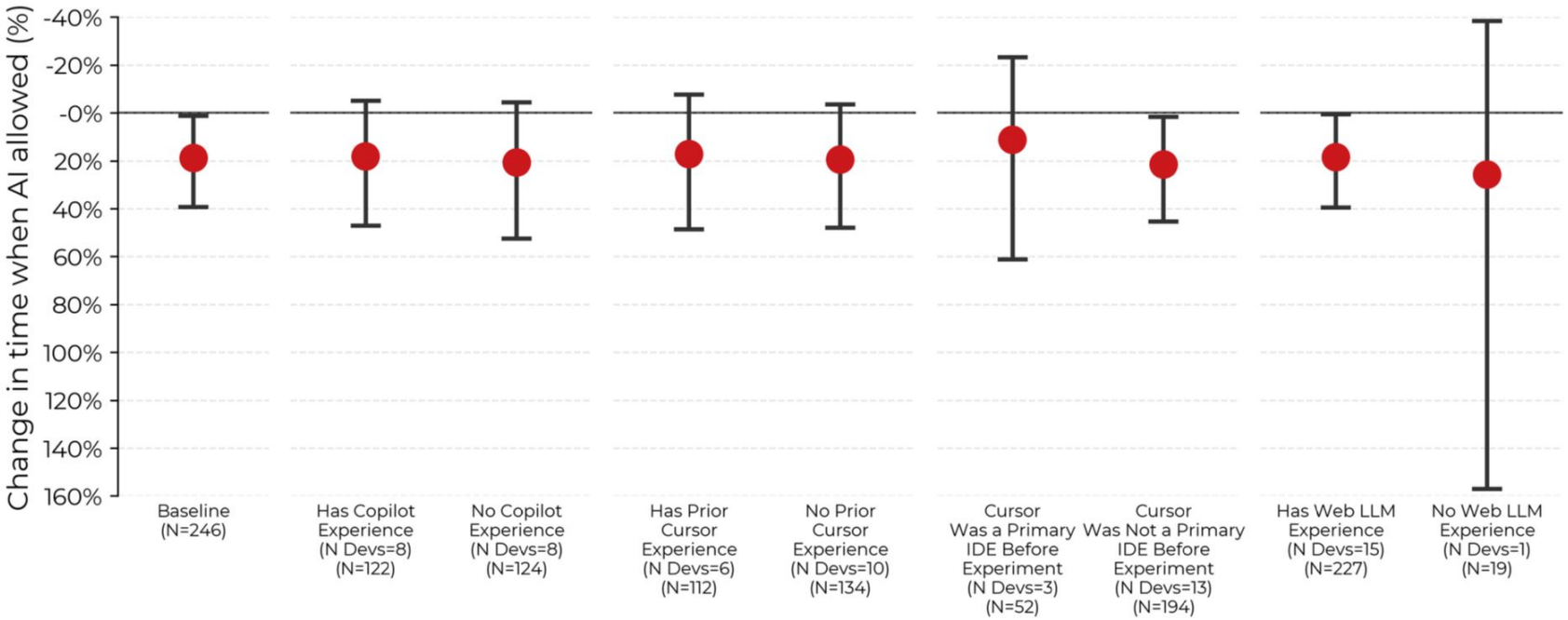


Figure 10: *We evaluate speedup on various subsets of developers' prior experience with GitHub Copilot, Cursor, and web LLMs (e.g. ChatGPT). Developers with prior Cursor experience (who use Cursor in the study) are slowed down similarly to developers without prior Cursor experience, and we see no difference between developers with/without Copilot or web LLM experience. See Section D.4 for details on how we estimate heterogeneous treatment effects.*

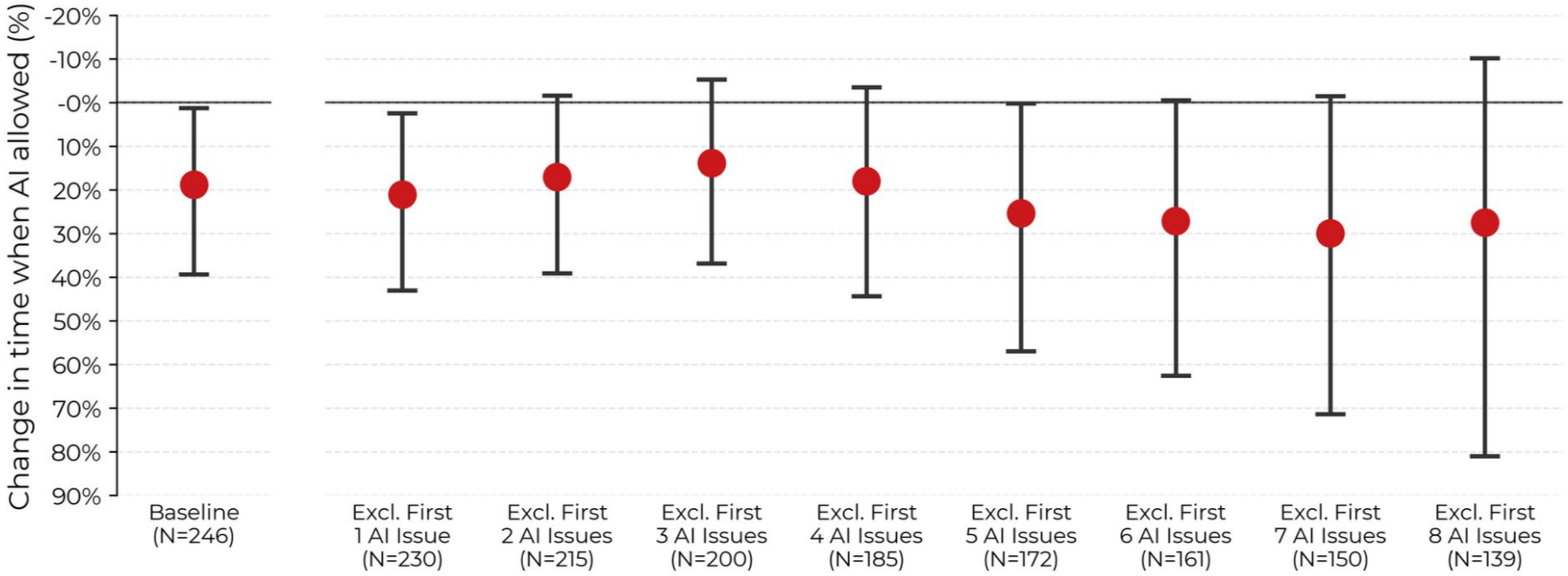


Figure 11: *We see similar slowdown percentages when excluding up to the first eight AI-allowed issues developers work on, suggesting that developers lacking basic skills around using AI effectively does not contribute substantially to the slowdown result. See Section D.4 for details on how we estimate heterogeneous treatment effects.*

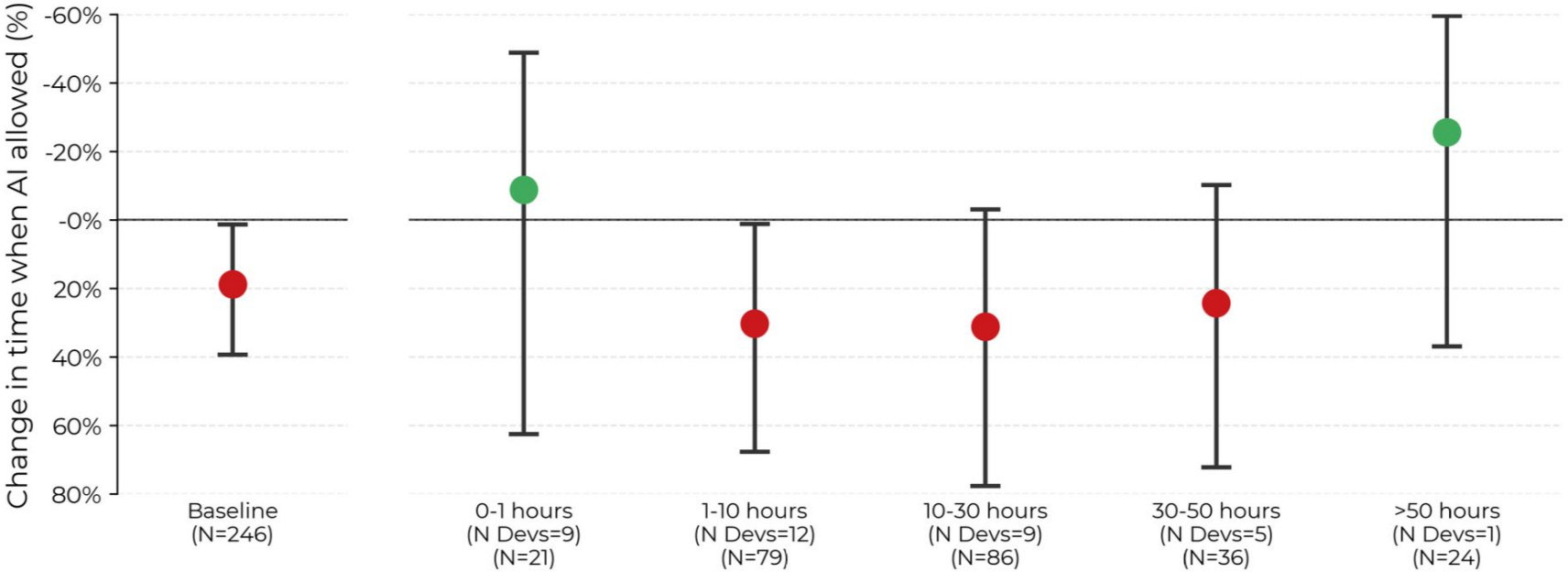


Figure 12: *Speedup on issues where developers have varying hours of experience using Cursor (including prior Cursor experience, plus their usage during the study period). We don't see large differences across the first 50 hours that developers use Cursor, but past 50 hours we observe positive speedup.*[15] *However, we are underpowered to draw strong conclusions from this analysis. See Section D.4 for details on how we estimate heterogeneous treatment effects.*