The Productivity Effects of Generative AI

Evidence from Three Field Experiments with Software Developers

Kevin Zheyuan Cui[1], Mert Demirer[2], Sonia Jaffe[3],
Leon Musolff[4], Sida Peng[3], and Tobias Salz[2]

[1]Princeton [2]MIT [3]Microsoft [4]UPenn

October 2024

This Paper

Question: what are the real-world productivity gains from generative AI in a high-skill work environment?

We analyze experiments run at Dell, Accenture and Microsoft that give developers random access to a coding assistant based on generative AI (GitHub Copilot).

- Experiments involve 5,114 full-time software developers
- Can observe behavior over a long period (up to fourteen months)

Software Development

Software supports much of modern economy

- Productivity improvements have potentially large spillovers

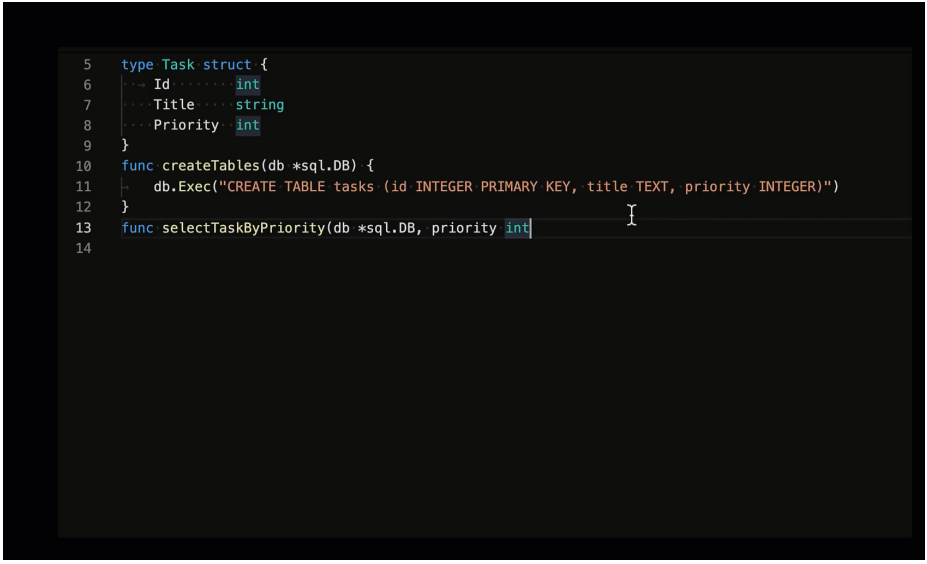
Software development is cognitive, demanding work

- Most previous papers focus on non-cognitive work (e.g., Acemoglu and Restrepo, 2020)
  - Exception: Brynjolffson, Li and Raymond (2023)
- Second-highest paid college major in 2024 (Source: Forbes)

Output of software developers is measurable

- Unit of work is defined and measured through version control

What is GitHub Copilot?



What is GitHub Copilot?

- Released in April 2021
- Based on Codex (GPT 3) developed by OpenAI
- Rapidly adopted
  - 1 million paid individual users of GitHub Copilot
  - 37,000 enterprise subscribers as of November 2023
- Coding assistant tools is a growing market
  - Other available products: Amazon Code Whisperer, Replit Ghostwriter, Google Codey

The Experiments

Microsoft

- 1,749 developers randomized into treatment (50.3%) and control (49.7%)
- Treatment group allowed (but not required) to access Copilot; control group *cannot*
- Started in October 2022

Accenture

- 311 software developers, started in July 2023
- 60% of participants are in the treatment group
- Encouragement design (same as for Microsoft)

Dell

- 3,054 developers, all eventually invited to use Copilot
- But: invitation date randomly assigned
- Started in October 2023

Version Tracking Outcomes

Commits

- Fundamental unit of version-tracking: marks collection of changes you can undo.

Pull Requests

- Can be interpreted as unit of work.
- Pre-defined goal that is reviewed and then added to the code base.

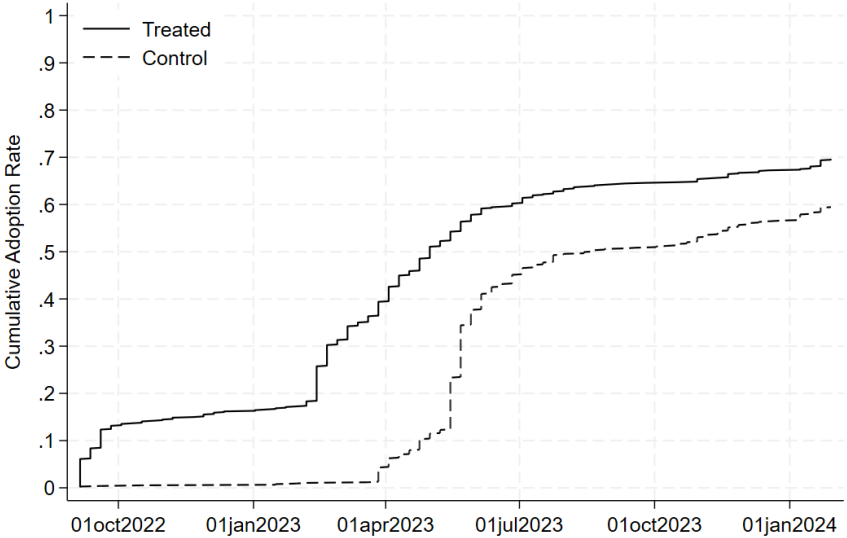
Successful Builds

- Code needs to be *built* (i.e., compiled) regularly.
- Building code allows engineers to check they did not introduce errors.

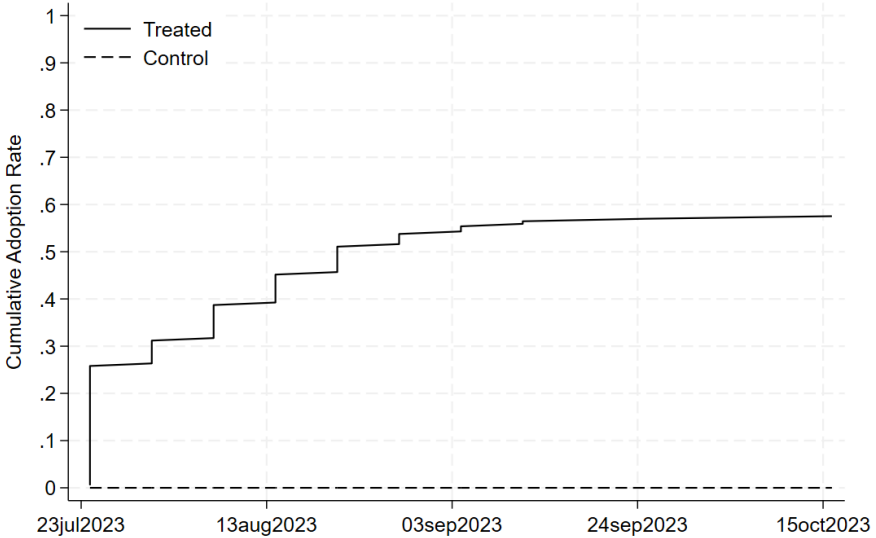
Other Data

- Number of suggestions by Copilot and how many accepted .
- Tenure in the company (i.e., how long since this developer was hired).

Adoption Patterns: Microsoft



Adoption Patterns: Accenture



Empirical Strategy

IV: We exploit the encouragement design and run

$$y_{it} = \beta \times \text{adoption}_{it} + \gamma_i + \delta_t + \varepsilon_{it},$$

where adoption_{it} is a dummy that turns on for engineer i in the week they first use Copilot, and is instrumented with treatment status.

Empirical Strategy

IV: We exploit the encouragement design and run

$$y_{it} = \beta \times \text{adoption}_{it} + \gamma_i + \delta_t + \varepsilon_{it},$$

where adoption_{it} is a dummy that turns on for engineer i in the week they first use Copilot, and is instrumented with treatment status.

Weighted IV: We re-weight the IV estimates to focus on periods where there are adoption differences between treatment and control, weighting by the difference in fraction of adoption in control vs treatment.

- Why? Control group eventually allowed access, which dilutes instrument relevance.
- This follows Coussens and Spiess (2021) and Huntington-Klein (2020).
- Bloom et al (2013) uses a similar strategy.

Headline Results

	Microsoft		Accenture		Dell		Pooled	
	IV	Weighted IV	IV	Weighted IV	IV	Weighted IV	IV	Weighted IV
Pull Requests	14.79 (21.95)	31.87** (14.36)	7.51 (5.46)	8.69** (4.37)	51.42 (40.58)	51.42 (40.58)	24.57 (15.49)	30.66*** (14.42)
Commits	16.36 (17.02)	18.84* (11.09)	1.04 (23.31)	5.50 (23.19)	- -	- -	3.60 (14.43)	12.17 (12.85)
Success Builds	15.61 (22.88)	26.88* (14.77)	89.57*** (24.70)	84.17*** (23.79)	- -	- -	52.59** (14.61)	55.52*** (16.50)
N Developers	1,653		311		3,054		5,018	
N Clusters	688		311		432		1,431	

All reported estimates are expressed as % of pre-treatment mean.

Pooled estimates constructed by averaging across experiments.

Heterogeneity (Microsoft only)

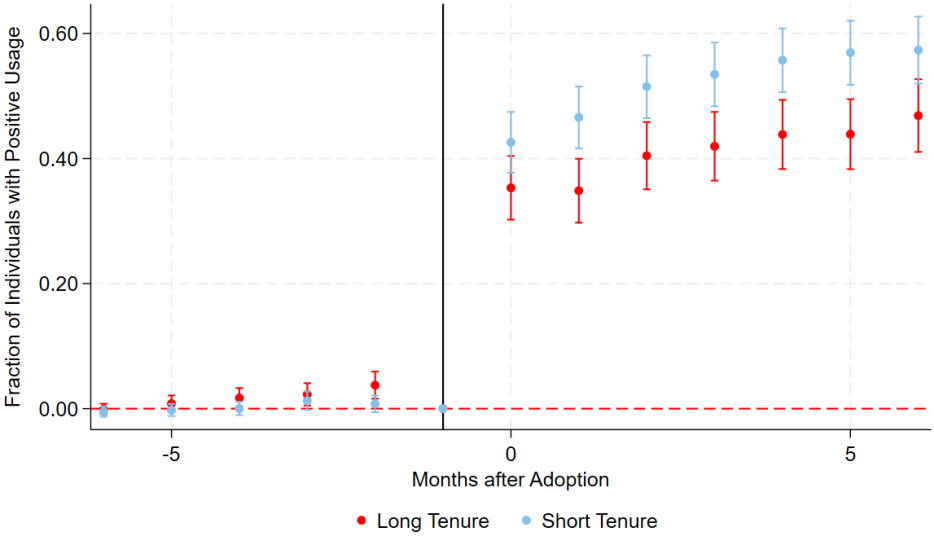
Summary so far:

- Results from all experiments suggest Copilot increased productivity of software developers
- Consistent with other studies in the literature

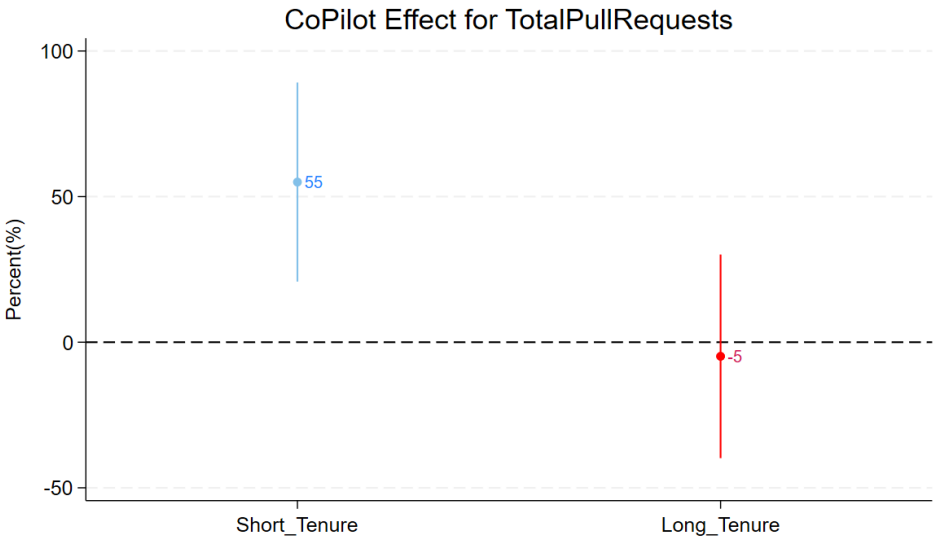
Next: Heterogeneity Analysis

- Split developers into long- and short-tenure
  - We split at median developer tenure, 1,100 days ($\approx$ 3 years)
- To help with power, we only use weighted IVs from now on
  - Unweighted directionally equivalent, larger standard errors

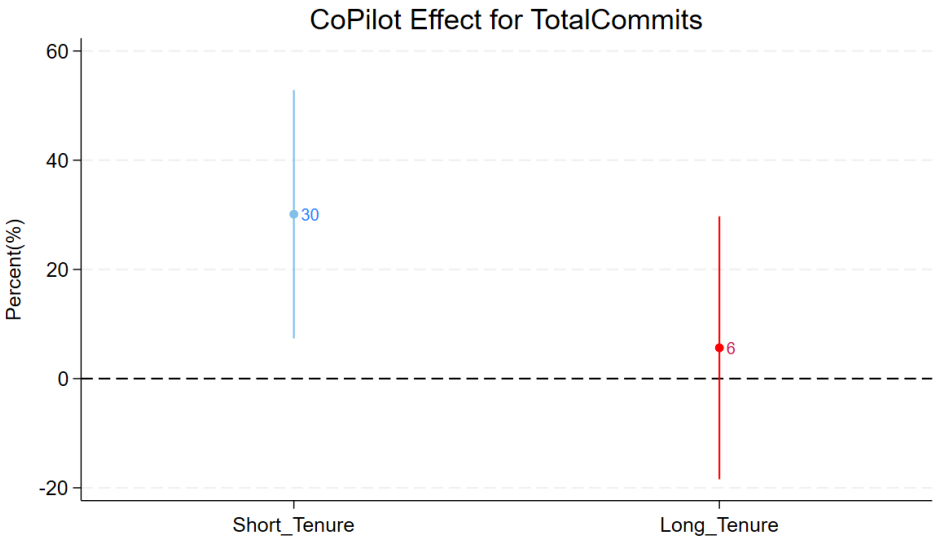
Heterogeneity (Microsoft only): Extensive Margin Usage



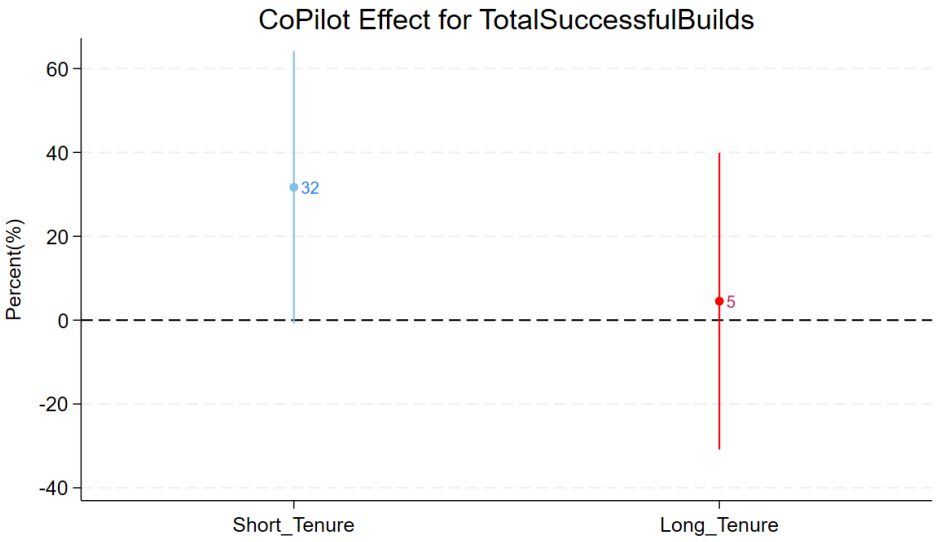
Heterogeneity (Microsoft only): Pull Requests



Heterogeneity (Microsoft only): Commits



Heterogeneity (Microsoft only): Successful Builds



Conclusion

- Examine productivity impact of LLMs through lens of coding
- Find noisey but consistently positive effects across three different companies
  - $\sim$31% increase in completed pull requests (but large standard errors)
- Heterogeneity analysis suggests less experienced developers benefit more
  - $\sim$55% increase for early-career developers, no effect on late-career
  - Consistent with Brynjolfsson et al (2023), Noy and Zhang (2023)