The Effects of Generative AI on High-Skilled Work: Evidence from Three Field Experiments with Software Developers

Kevin Zheyuan Cui Mert Demirer
Sonia Jaffe Leon Musolff Sida Peng Tobias Salz

Appendix - For Online Publication

A Data Cleaning

We provide details on which individuals we had to exclude from each raw dataset and the reasons for their exclusion.

A.1 Microsoft

In the original sample of the dataset, we have 1,746 individuals. We kept only software developers, which leaves us with 1,538, and we dropped people who switched organizations, leaving us 1,522. Finally, we dropped one individual who adopted Copilot before the experiment started, with a final sample of 1,521.

We also drop the data for the last week of the dataset since the dataset does not record the full week of activity for the last week.

Finally, note that while the restriction for the control group was lifted in April 2023, ten individuals in the control group adopted before that date. We include these individuals in our regressions, which naturally weakens the instrument's strength.

A.2 Accenture

We drop individuals who have no record of data and people who have left the company. We start with the original dataset containing 369 individuals. After dropping individuals with no outcome measures, we are left with 320. After further dropping individuals who left the company, we are left with a final sample of 316.

Finally, we note that while individuals in the control group were allowed to adopt starting December 2023, there was one individual in the control group who adopted in October 2023. We include this individual in our regressions.

A.3 Anonymous Company

The original sample has 3,054 individuals. We drop individuals who have shown/adopted before they were given access, and are left with a final sample of 3,030 individuals.

B Intention-To-Treat Results

In this appendix, we investigate the ITT effects for the Microsoft and Accenture experiments. We first estimate the ITT effect and then conduct a heterogeneity analysis based on ITT estimates.

B.1 ITT Estimates of Productivity Effect

As both experiments feature imperfect compliance because the Control group is eventually granted access to GitHub Copilot—see Figure 2—we drop observations starting just before significant control group adoption. In particular, we restrict attention to the first 29 weeks for the Microsoft experiment and the first 21 weeks for the Accenture experiment. We report the effects of simply being assigned to the treated group (without necessarily adopting Copilot) during those weeks in Table 5. We also report the IV results corresponding to these ITT estimates, noting that they will generally differ from the results reported in Table 9 due to differences in the sample period and weighting.

With the notable exception of the number of builds in the Accenture experiment, we find no statistically significant impact of being assigned to treatment on outcomes. However, we note that adoption rates during these early stages of the experiment were quite small: at Microsoft, only 44.2% of developers adopted Copilot in the first 29 weeks (and this is with adoption significantly accelerating towards the end of this period). At Accenture, adoption was only slightly higher at 61.7% by the end of the initial phase.

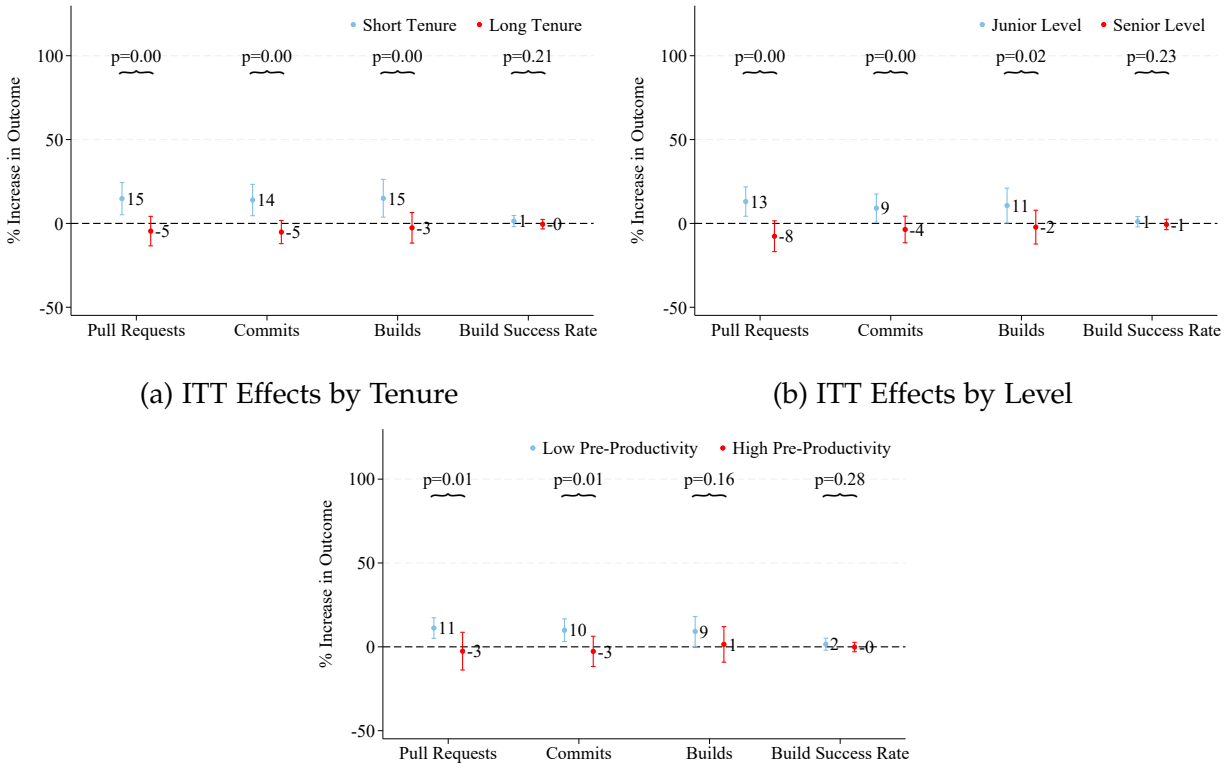
Outcome	Microsoft		Accenture		Pooled	
	ITT	IV	ITT	IV	ITT	IV
Pull Requests	3.91	22.19	9.41	18.65	4.66	20.16
	(3.83)	(22.02)	(9.65)	(18.96)	(3.56)	(14.37)
Commits	3.20	18.19	1.65	3.28	3.08	11.93
	(3.40)	(19.45)	(11.54)	(22.86)	(3.26)	(14.81)
Builds	5.09	28.89	56.45***	111.90***	9.49**	64.84***
	(4.22)	(24.27)	(13.79)	(27.77)	(4.04)	(18.27)
Build Success Rate	0.33	1.81	-9.73*	-17.24*	-0.22	-5.14
	(1.38)	(7.65)	(5.73)	(10.09)	(1.34)	(6.1)

Table 5: ITT Estimates of Effect of Copilot.

Notes: We investigate the ITT effects of assigning a developer to the treatment group in the Microsoft and Accenture experiments, cutting the sample just before the Control group adopts Copilot. With the exception of builds at Accenture, we find no statistically significant effect of being assigned to the treatment group on outcomes. All effects were estimated in linear regressions but are expressed as a % of the pre-treatment mean. *10%, **5%, ***1%

B.2 Heterogeneity in ITT Estimates

In Figure 7, we explore heterogeneity in the ITT effects, finding that these effects are significantly higher for developers with short tenure, at a junior level, or with low pre-productivity. However, we emphasize that this heterogeneity could be driven by both (i) differences in adoption or utilization and (ii) differences in the effect of Copilot conditional on utilization.



(a) ITT Effects by Tenure

(b) ITT Effects by Level

(c) ITT Effects by Pre-Productivity

Figure 7: Heterogeneity of Copilot Effect (ITT)

Notes: This figure provides ITT estimates of the effect of adopting Copilot on the total number of pull requests, commits, builds, and build success rate broken out by (a) whether a developer's tenure with Microsoft at the beginning of the experiment was below median (short tenure) or above median (long tenure), (b) the level at which a developer was employed and (c) the productivity of the developer before the start of the experiment. The dots in each panel are estimates derived from a single regression for each outcome where the ITT effect is allowed to differ by (a) tenure, (b) level, or (c) the developer's productivity in the pre-period as measured by their total number of completed pull requests. The bars provide 95% confidence intervals based on standard errors clustered at the level of treatment assignment. For all three outcome measures, the ITT effects on productivity are stronger for short-tenure/junior/less productive developers. Note that the ITT effects combine heterogeneity in the amount of uptake with potentially heterogeneous treatment effects on adopters.

C Dynamic Treatment Effects

To explore whether there is any evidence that users have to learn how to use GitHub Copilot, we split our sample into (i) the first three months after adoption, and (ii) more than three months since adoption. We then estimate the DiD model and the weighted IV models, allowing for different effect sizes across these periods (but estimate jointly across periods) in Table 6. The DiD estimates suggest that the treatment effect increases over time, but the experimental IV estimates are underpowered to detect any heterogeneity in effects.

Outcome	DiD		WIV	
	Short Term	Long Term	Short Term	Long Term
Pull Requests	2.16	10.55***	32.39*	31.51
	(2.45)	(3.35)	(18.39)	(22.17)
Commits	4.24*	7.13**	29.33	14.83
	(2.37)	(3.24)	(18.41)	(19.41)
Builds	4.43*	7.46**	36.21	19.55
	(2.57)	(3.62)	(22.21)	(23.58)
Build Success Rate	-1.17	-1.52	-5.47	1.39
	(0.81)	(0.97)	(7.73)	(6.57)

Table 6: Dynamic Treatment Effects

Notes: We separate out treatment effects by whether a developer has adopted GitHub Copilot within the last three months ("Short Term") or has used GitHub Copilot for at least three months already ("Long Term"). While DiD estimates suggest the treatment effect may be growing over time, our experimental (weighted IV) estimates are underpowered to confirm this using experimental variation. All effects were estimated in linear regressions but are expressed as a % of the pre-treatment mean. *10%, **5%, ***1%

D Comparison between Experiments

In this Appendix, we provide additional details on the experiments, emphasizing differences in implementation across companies, the distinct roles of software developers within each firm, and how outcome variables—such as pull requests—should be interpreted across these different contexts. To preserve confidentiality, we cannot describe specific internal details from each organization. However, we offer a qualitative depiction of the differences in developers' day-to-day tasks across the three companies based on publicly available information. We also discuss how differences in the experimental designs may have contributed to heterogeneity in the estimates.

D.1 Design Differences

Although all three field experiments share the same objective, their designs necessarily vary to accommodate each firm's workflows and operational constraints. The Microsoft experiment was launched first, leveraging the company's early access to Copilot as its owner. The experiment ended earlier than planned when control-group developers began to request access mid-experiment. Accenture's experiment, by comparison, started later but lasted for a full five months as planned. The experiment at Accenture also incorporated a training component. The anonymous company took a different approach, implementing a randomized cohort rollout instead of random assignment to a treatment and control group.

Because these experiments were run by the companies in collaboration with Microsoft and we only analyzed the data, they were not pre-registered. Nevertheless, the goal of each experiment is the same across companies: to learn the productivity effects of GitHub Copilot and use that information in their business decisions. Even though the objective is the same, implementation was driven by idiosyncratic constraints, such as the number of software developers who could participate, the duration for which the control group's Copilot access could be restricted, how quickly the firm needed the experiment results, and the cost of the experiment.

There are several ways these constraints vary across companies. For instance, the ability of the experiment coordinator to persuade managers to postpone the rollout of Copilot varies depending on the managers involved and the perceived importance of Copilot. This, in turn, affected the duration of each experiment. For example, in the anonymous company, this constraint led to a randomized staggered rollout rather than a full RCT with treated and control groups. Similarly, at Microsoft, the control group gained access to GitHub Copilot earlier than planned because some teams requested access to it since they were working on AI-related products at the time. Another example is the encouragement design: Microsoft used a simple email, while Accenture implemented comprehensive training sessions. These differences reflect variation in both intervention costs and how firms and their employees perceive them.

D.2 Differences in Software Development

While all three companies in our study employ software developers, these developers differ in their day-to-day tasks and workflows—factors that influence how GitHub Copilot

is integrated into their daily work and affect productivity.

At Microsoft, most developers are responsible for maintaining and evolving long-lived first-party products and cloud services. Their code ships directly to users and is updated continuously; therefore, the teams own design, implementation, automated testing, and post-release telemetry in a single, tight feedback loop. Microsoft also employs internal systems that help developers work quickly while maintaining high quality. For example, before any code changes can be added to the main product, they must pass through automated checks and reviews by other team members (gated pull-request policies). The company also uses automated systems that test code and deploy updates (CI/CD pipelines), plus real-time monitoring tools (dashboards) that show how the software is performing.[23]

Accenture is a service-based company that specializes in resolving client issues and providing support for pre-existing applications, primarily to external clients, rather than developing proprietary products. The majority of software development work focuses on client projects across various industries. Developers are assigned to specific client engagements spanning various industries, including banking, insurance, telecommunications, and public services. While Accenture does have some enterprise software products and platforms, these represent a smaller portion of its business. The diverse client base exposes Accenture's software developers to a wide variety of client projects. They might build entirely new software systems, improve existing ones, or fix problems when things break. This means they need to understand the client's business needs, design solutions, write code, and test that everything works properly. Because they work with many different clients, these developers learn multiple programming languages and gain skills across various types of software and computer systems.[24]

Since the manufacturing firm that hosted the third experiment is anonymous, we describe its environment in terms of typical large, hardware-centered manufacturers, rather than providing firm-specific details. In such companies, software developers usually work on three overlapping roles—(i) firmware developers who write code that runs on the computer chips that control hardware components, (ii) factory-automation engineers who build supervisory control software for assembly lines, and (iii) integration engineers who link production equipment with plant-wide IT and quality systems. Each of these categories follows development schedules that combine software updates into planned releases, prioritizing reliability over rapid changes.

We account for these differences between companies in several ways in our empirical analysis. First, throughout the paper, we report percentage changes in the outcome variables rather than levels, since baseline levels are likely to vary across companies. Second, we report estimates for each company separately and only pool them in our mean specification after estimating individual effects. Finally, all of our estimates are based on within-company—and even within-developer—variation, so cross-company differences do not confound the estimates.

[23]Source: Microsoft—How Microsoft Develops DevOps.

[24]Sources: Accenture—Job Details1, Accenture—Job Details2, Accenture—Newsroom Fact Sheet.

D.3 Differences in Outcome Variables

In our baseline results, we used pull requests as the primary outcome variable. Although the primary structure of pull requests remains the same across companies, approaches to pull requests can vary depending on the main product and the company structure. While startups might allow developers to self-merge small changes with minimal review to maintain a rapid pace, established enterprises in regulated industries typically require multiple approvers, comprehensive automated testing, and sometimes security team sign-offs before any code reaches production. The tooling and automation integration also spans a wide spectrum, from companies that run extensive CI/CD pipelines[25] with security scans and performance benchmarks on every PR (code review), to those relying more heavily on manual testing and simpler approval workflows. These differences in merge strategies, branch protection rules, and review culture generally evolve as companies mature, typically moving toward more structured and rigorous processes as team sizes grow and the business impact of code changes increases.

Based on publicly available sources, we provide an overview of Microsoft's internal pull request (PR) practices.[26] Microsoft's PR practices center on Azure DevOps, utilizing a trunk-based workflow. In this approach, developers create short-lived branches off the main branch, and each pull request triggers automated builds and tests to enforce branch policies. Only after these checks pass do team peers review the code; Microsoft requires at least one approval to ensure code quality and architecture standards are met before merging it into the main branch. At Microsoft, pull requests and code reviews are frequent: according to Macleod et al. (2018), 36% of Microsoft developers review code multiple times per day.[27] Such reviews are important in maintaining quality and coordinating tens of thousands of developers on shared codebases.

At Accenture, the most important difference from Microsoft is that pull requests modify a client's live codebase rather than an internal one. This substantially increases the consequences of coding errors. As a result, every PR at Accenture must pass through multiple review and validation stages before approval, making them far less frequent. This is also evident in Table 2, which shows a significantly lower average number of weekly pull requests at Accenture compared to Microsoft (0.13 vs. 0.86).

D.4 The Effects of Company-Level Differences on Results

Because the experiment design choice (timing, training, cohorting) was tailored to specific firm-level constraints, measured treatment effects are inevitably affected by experimental design and company context. Therefore, it is not possible for us to pinpoint the exact sources of heterogeneous effects across these companies. However, we can describe some possible sources of this heterogeneity.

[25] CI/CD stands for Continuous Integration/Continuous Deployment, which refers to automated systems that regularly test, integrate, and deploy code changes to ensure software quality and enable rapid releases.

[26] Sources: Microsoft—How Microsoft Develops DevOps, Microsoft—Transforming Modern Engineering at Microsoft, Greiler—Code Reviews at Microsoft.

[27] MacLeod, Laura, Michaela Greiler, Margaret-Anne Storey, Christian Bird, and Jacek Cz- erwonka (2018). "Code Reviewing in the Trenches: Challenges and Best Practices". In: IEEE Software 35.4, pp. 34–42.

One potential source arises from the nature of the tasks developers perform in these companies. As described above, coding tasks vary significantly across these companies, inherently generating heterogeneity in the effectiveness of coding assistant tools. For example, coding tools typically perform better on programming languages and tasks that are well-represented in the training data, and some companies may use languages or work on tasks that were more extensively covered in the AI model's training. Although we have limited visibility into the languages used by developers (in some experiments, only for Copilot users), it is plausible that Accenture uses a greater variety of languages and more niche languages than other companies, given their work across multiple industries as described above. This diversity could make Copilot less effective on these less common programming languages, potentially explaining the lower impact observed at Accenture relative to Microsoft.

Another example is selection into treatment. As described in Section 4.4, our analysis identifies the LATE, capturing heterogeneous effects driven by selection into treatment. Selection into treatment can vary across companies due to factors such as company culture and policies. Therefore, even if the underlying productivity effects of coding assistants are identical across companies, differences in how participants select into treatment could lead to heterogeneous results.

Finally, the experimental design can indirectly influence the selection into treatment. For instance, in the Accenture experiment, managers encouraged their direct reports to use the tool to enhance compliance, whereas at Microsoft, a simple email was sent. This difference can affect developers' motivations for adoption: at Accenture, developers might use the tool primarily to satisfy managerial expectations, driving the relatively fast and widespread adoption observed in Figure 2b. At Microsoft, by contrast, adoption is initially slower and less extensive, leaving more scope for developers to select into treatment based on anticipated productivity benefits. This can be a source of a higher LATE estimate at Microsoft than at Accenture. Similarly, the timing of the experiment can affect results. GitHub Copilot likely received updates during the experimental period, potentially leading experiments conducted earlier, such as Microsoft's, to observe a smaller impact.

While we think that the sources of heterogeneity across companies are extremely important, we ultimately have a sample size of three, which prevents us from obtaining systematic evidence on the differences across companies.

E The First Accenture Experiment

We do not discuss in detail in the main text another experiment conducted by Accenture in April 2023, which included a number of Accenture offices located in Southeast Asia. This experiment was abandoned by the company after Accenture laid off 19,000 employees that same month (cnn.com), including 42% of the developers participating in this experiment. Still, this attrition was balanced across treatment and control, and we can thus subset to the 204 developers who were not let go for our analysis; indeed, Table 7 confirms that after this subsetting, treatment and control are still balanced. The problem emerges because Microsoft did not log all Copilot usage data for this experiment, as the company considered it abandoned. In particular, we lack adoption data for the control group until October '23. Without this adoption data, any analysis is potentially biased.

Still, because our initial analysis revealed that this experiment was the only experiment across the three in which we have a negative (though statistically insignificant) point estimate for Copilot's effect on productivity, we proceed to analyze this experiment in this appendix by imputing that nobody in the Control group adopts Copilot until October '23, yielding the adoption path in Figure 8. Thus, in the worst-case scenario, it could be that all the adoptions that we attribute to October 2023 already happened right at the beginning of the experiment. This data quality concern means our treatment effect estimates will be conservative (as we may mistakenly count up to 10% of the control group as non-adopters during half of the sampling period).

Keeping in mind this caveat that our treatment effect estimates are potentially conservative, we report the results from this first Accenture experiment in Table 8. We find a negative point estimate of -39.18% (SE: 36.78%) on the number of tasks completed. Still, this estimate has a high degree of statistical uncertainty, and we note that the estimates for the number of commits 43.04% (SE: 43.04%) and builds 12.33% (SE: 53.60%) are both positive, although not statistically significant.

	Control		Treatment			
	Mean	Std. Dev	Mean	Std. Dev	Difference	p-value
Pull Requests	0.08	0.26	0.09	0.29	0.02	0.38
Commits	6.28	11.24	5.28	10.09	-1.00	0.30
Builds	5.32	10.32	5.23	10.52	-0.09	0.93
Build Success Rate	0.49	0.33	0.50	0.33	0.01	0.60

Table 7: Balance Table for First Accenture Experiment

Notes: This table presents a comparison of pre-experimental outcomes in control and treatment groups in the first Accenture experiment. For each measure, we present its mean and standard deviation in the control group and in the treatment group. We also show the mean difference across these groups and the p-value associated with an underlying test of differences in means. The p-values for the differences are calculated using standard errors clustered at the level of treatment assignment.

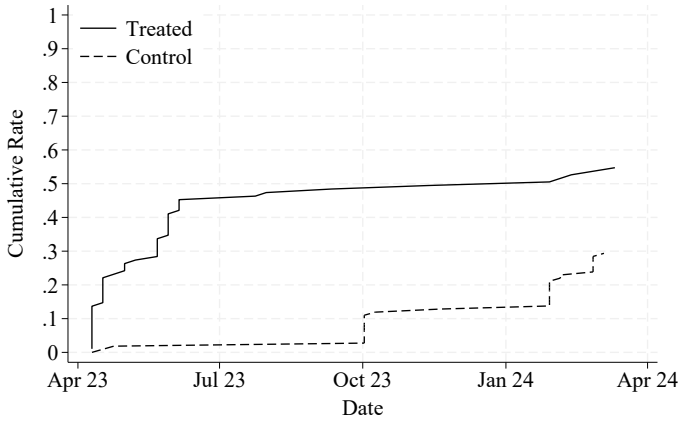


Figure 8: Cumulative Adoption Rates for First Accenture Experiment

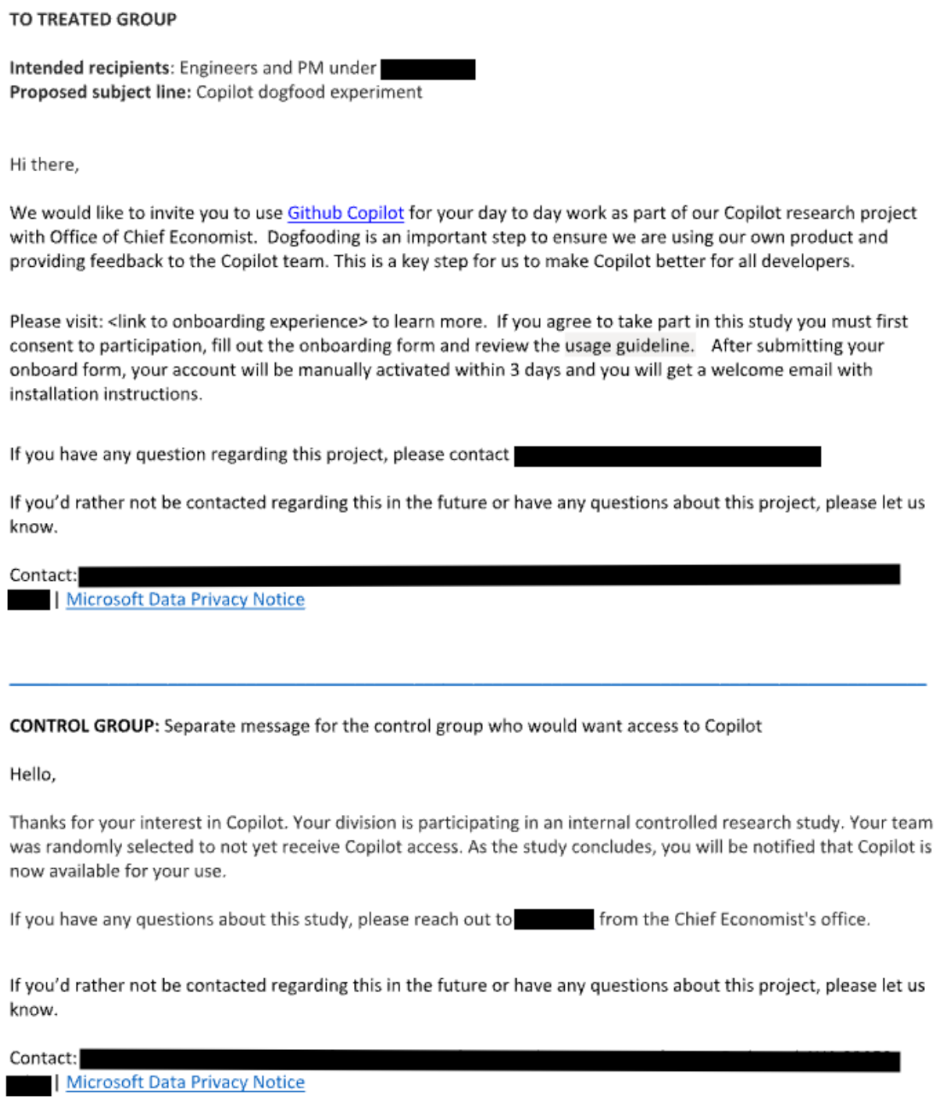
Notes: This graph shows the cumulative rate of adoption over time for software developers in both the treatment and control groups in the first Accenture experiment. Note that we are assuming that nobody in the Control group adopts Copilot until October 2023.

Outcome	Accenture #1
Pull Requests	-39.18
	(36.78)
Commits	43.04
	(38.80)
Builds	12.33
	(53.60)
Build Success Rate	-0.99
	(16.51)
N Developers	204
N Clusters	204

Table 8: Weighted IV Results for First Accenture Experiment

Notes: This table provides estimates of the effect of GitHub Copilot adoption on the number of Pull Requests, Commits, Builds, and Build Success Rates in the first Accenture experiment. Standard errors are clustered at the developer level. The estimates presented in this table are potentially conservative because they require imputing that nobody in the Control group adopts Copilot until October 2023.)

F Additional Exhibits & Robustness Checks



TO TREATED GROUP

Intended recipients: Engineers and PM under [redacted]
Proposed subject line: Copilot dogfood experiment

Hi there,

We would like to invite you to use Github Copilot for your day to day work as part of our Copilot research project with Office of Chief Economist. Dogfooding is an important step to ensure we are using our own product and providing feedback to the Copilot team. This is a key step for us to make Copilot better for all developers.

Please visit: <link to onboarding experience> to learn more. If you agree to take part in this study you must first consent to participation, fill out the onboarding form and review the usage guideline. After submitting your onboard form, your account will be manually activated within 3 days and you will get a welcome email with installation instructions.

If you have any question regarding this project, please contact [redacted]

If you'd rather not be contacted regarding this in the future or have any questions about this project, please let us know.

Contact: [redacted]
[redacted] | Microsoft Data Privacy Notice

CONTROL GROUP: Separate message for the control group who would want access to Copilot

Hello,

Thanks for your interest in Copilot. Your division is participating in an internal controlled research study. Your team was randomly selected to not yet receive Copilot access. As the study concludes, you will be notified that Copilot is now available for your use.

If you have any questions about this study, please reach out to [redacted] from the Chief Economist's office.

If you'd rather not be contacted regarding this in the future or have any questions about this project, please let us know.

Contact: [redacted]
[redacted] | Microsoft Data Privacy Notice

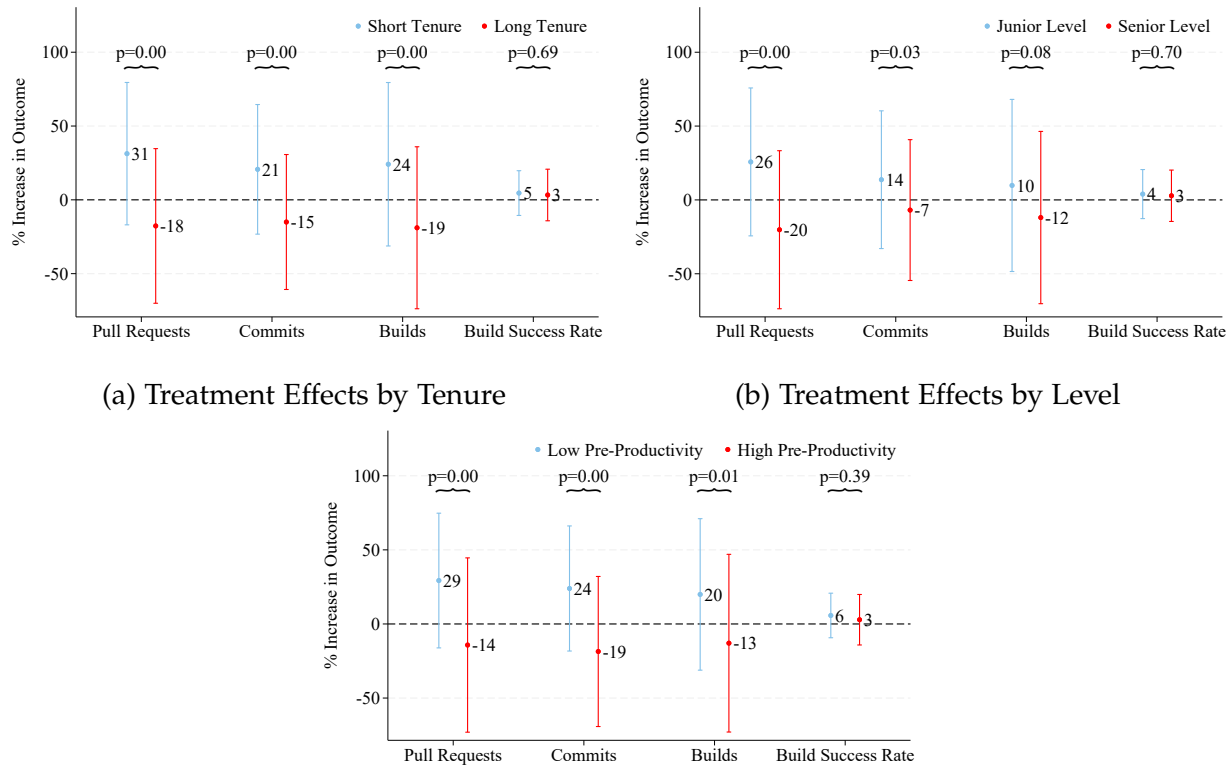
Figure 9: E-mail Sent to Participants in the Microsoft Experiment

Notes: This figure exhibits the copy that was sent to participants in the Microsoft experiment. Only the subset of users in the Control group who explicitly requested access to GitHub Copilot received the second email.

Outcome	Microsoft				Accenture				Anon. Comp.				Pooled			
	DiD	DiD-P	IV	W-IV	DiD	DiD-P	IV	W-IV	DiD	DiD-P	IV	WIV	DiD	DiD-P	IV	W-IV
Pull Requests	7.63***	6.81***	10.53	27.38**	52.65***	22.54**	15.97	17.94	1.70	2.77	54.03	54.03	6.24***	5.23***	18.73	26.08**
	(2.49)	(2.52)	(24.82)	(12.88)	(9.46)	(9.35)	(21.26)	(18.72)	(2.47)	(2.35)	(42.63)	(42.63)	(1.72)	(1.69)	(15.1)	(10.3)
Commits	7.03***	6.42***	5.54	18.32	12.85	-8.67	-3.60	-4.48	-	-	-	-	7.25***	5.82**	0.97	13.55
	(2.32)	(2.46)	(22.20)	(11.25)	(11.62)	(12.08)	(22.19)	(21.88)	-	-	-	-	(2.28)	(2.41)	(15.69)	(10.0)
Builds	7.11***	7.25***	5.87	23.19	39.66***	13.70	96.05***	92.40***	-	-	-	-	8.23***	7.56***	49.66**	38.38***
	(2.65)	(2.74)	(27.25)	(14.20)	(14.03)	(12.10)	(28.05)	(26.78)	-	-	-	-	(2.6)	(2.67)	(19.55)	(12.55)
Build Success Rate	-0.65	-0.66	3.92	-1.34	-20.72***	-19.59***	-18.10*	-17.40**	-	-	-	-	-1.13	-1.05	-5.39	-5.53
	(0.79)	(0.78)	(8.17)	(4.23)	(5.06)	(5.36)	(9.55)	(7.12)	-	-	-	-	(0.78)	(0.77)	(6.21)	(3.64)

Table 9: Alternative Specifications for Experimental Results

Notes: This table builds on Table 3 by reporting the results of additional specifications. Each entry can be interpreted as an estimate of the percentage effect of adoption of GitHub Copilot. DiD is like in Table 3, DiD-P is like DiD but runs a Poisson model and then reports $100 * (\exp(\beta) - 1))$, IV is like W-IV in Table 3 but without weighting the regression by adoption differences, W-IV is like in Table 3.



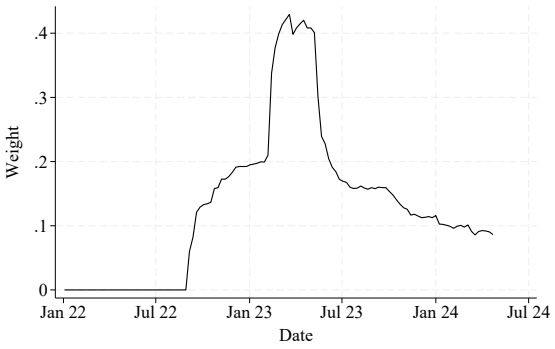
(a) Treatment Effects by Tenure

(b) Treatment Effects by Level

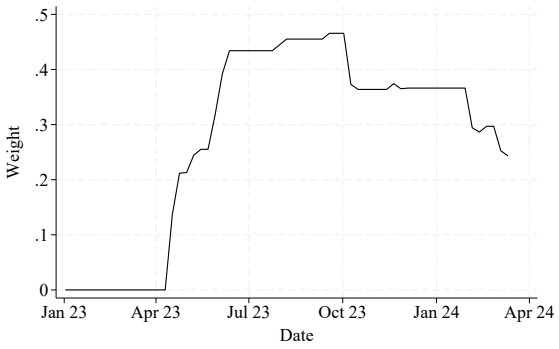
(c) Treatment Effects by Pre-Productivity

Figure 10: Heterogeneity of Effect of Copilot (Unweighted IV)

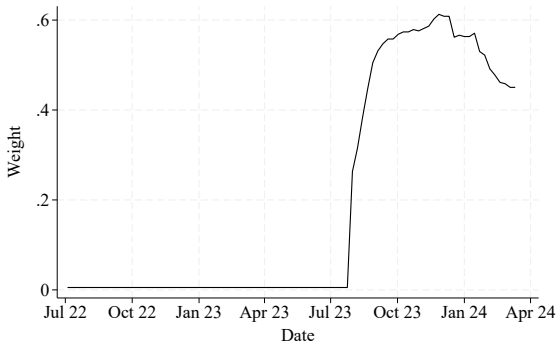
Notes: This figure provides unweighted IV estimates of the effect of adopting Copilot on the total number of pull requests, commits, builds, and build success rate broken out by (a) whether a developer's tenure with Microsoft at the beginning of the experiment was below median (short tenure) or above median (long tenure), (b) which level a developer was employed at, and (c) the developer's pre-experiment productivity. The dots in each panel are estimates derived from a single regression for each outcome where the treatment effect is allowed to differ by (a) tenure, (b) level, or (c) the developer's productivity in the pre-period as measured by his total number of completed pull requests. The bars provide 95% confidence intervals based on standard errors clustered at the level of treatment assignment. For the first three outcome measures, the effects on productivity are stronger for short-tenure/more junior/less productive developers. The p-values for differences between individual coefficient estimates are often very small despite substantial overlap in the confidence intervals due to high correlations (exceeding 0.9) between the estimates.



(a) Microsoft Experiment



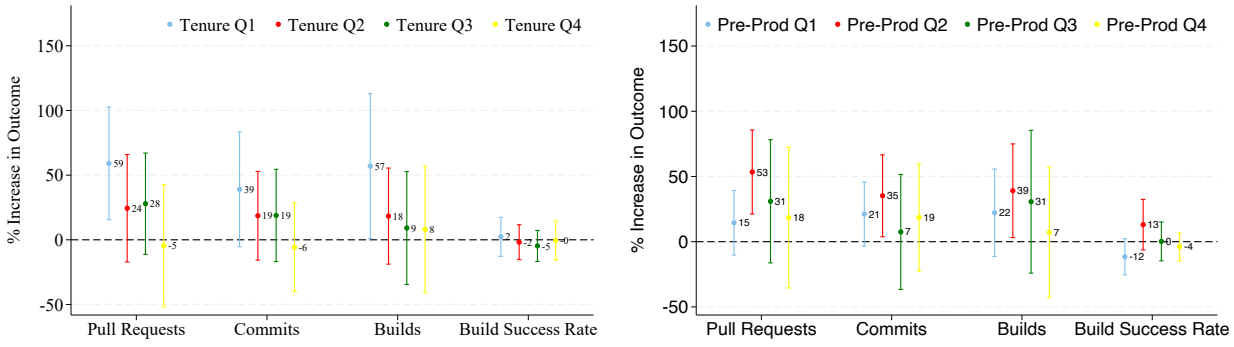
(b) Accenture Experiment #1



(c) Accenture Experiment #2

Figure 11: Regression Weights

Notes: This figure provides the weights used in the W-IV estimates underlying Tables 3 and 8. Recall that we are weighting the IV estimates to exploit information from periods where the instruments predict uptake. Hence, we use the difference in adoption across the control and treatment groups by a given date as our weight. This matters most for the Microsoft experiment, in which the control group adopted at an elevated rate when its access was granted in March 2023. For this experiment, we put extra weight on the period just before the control group was allowed to adopt Copilot.



(a) Treatment Effects by Tenure

(b) Treatment Effects by Pre-Productivity

Figure 12: Heterogeneity of Copilot Effect based on Quantiles

Notes: This figure provides weighted IV estimates of the effect of adopting Copilot on the total number of pull requests, commits, builds, and build success rate, broken out by (a) the developer's tenure with Microsoft at the beginning of the experiment, and (b) the developer's productivity prior to the start of the experiment. In both cases, developers were grouped into four categories based on quartiles of the corresponding variable. The dots in each panel are estimates derived from a single regression for each outcome where the treatment effect is allowed to differ by (a) tenure, and (b) the developer's productivity in the pre-period as measured by his total number of completed pull requests. The bars provide 95% confidence intervals based on standard errors clustered at the level of treatment assignment. For the first three outcome measures, the effects on productivity are stronger for short-tenure/junior/less productive developers, though the difference is typically not statistically significant.