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, and Tobias Salz

February 2025

Abstract

This study evaluates the impact of generative AI on software developer productivity via randomized controlled trials at Microsoft, Accenture, and an anonymous Fortune 100 company. These field experiments, run by the companies as part of their ordinary course of business, provided a random subset of developers with access to an AI-based coding assistant suggesting intelligent code completions. Though each experiment is noisy, when data is combined across three experiments and 4,867 developers, our analysis reveals a 26.08% increase (SE: 10.3%) in completed tasks among developers using the AI tool. Notably, less experienced developers had higher adoption rates and greater productivity gains.

[*]Cui: Princeton. Demirer: MIT. Jaffe: Microsoft. Musolff: The Wharton School of the University of Pennsylvania. Peng: Microsoft. Salz: MIT. This RCT was post-registered as AEARCTR-0014530 (Peng 2024). We are grateful to Avi Goldfarb, Shane Greenstein, Anton Korinek, Ethan Mollick, Daniel Rock as well as the participants of the IIOC2024 conference and the AI, Cognition, and the Economy Workshop. Thanks also for data help from employees at Microsoft, GitHub, and Accenture: Phillip Coppney, Wen Du, Ya Gao, Lizzie Redford, Ryan J. Salva, Daniel A. Schocke, Amanda Silver, An-Jen Tai, Dan Tetrick, Jeff Wilcox. Mert Demirer and Tobias Salz thank the MIT GenAI initiative for funding.

1 Introduction

Many economists expect generative AI to profoundly affect the organization of economic activity (Agrawal, Gans, and Goldfarb 2019; Frank et al. 2019; Furman and Seamans 2019; Greenstein et al. 2024). Eloundou et al. (2023) estimate that generative AI can perform tasks associated with over 80% of U.S. jobs and that AI task coverage is notably higher for occupations that require advanced degrees. The ability of generative AI to perform tasks required in such high-skilled occupations — allowing it to assist doctors in diagnosing diseases, lawyers in drafting legal documents, and software engineers with code development — has led to predictions of substantial productivity gains from the adoption of such technologies (Baily, Brynjolfsson, and Korinek 2023). Others, however, are less optimistic about such productivity gains (Acemoglu 2024).

Uncertainty around firms' willingness to adopt these technologies and their capacity to make necessary complementary investments (Bresnahan 2024; Brynjolfsson, Rock, and Syverson 2021) makes it currently difficult to empirically assess whether or not optimism about productivity gains is justified.[1] Nevertheless, some applications of generative AI have already matured and are integrated into existing workflows. An example is software development, where commercial coding assistants based on generative AI have gained widespread adoption.[2]

In this project, we ask how generative AI affects the productivity of knowledge workers, using software developers as an example. We analyze three large-scale randomized controlled trials in a real-world environment. These experiments randomly assigned access to Copilot, a coding assistant developed by GitHub in collaboration with OpenAI, to just under five thousand software developers at Microsoft, Accenture, and an anonymous Fortune 100 electronics manufacturing company (henceforth Anonymous Company). These experiments were run as part of the ordinary course of business at these companies to decide whether or how extensively to adopt these technologies, and the companies kindly shared the resulting data with us.[3] These experiments lasted 2 to 8 months after which all groups were granted access to Copilot.

Our preferred estimates from an instrumental variable regression suggest that usage of the coding assistant causes a 26.08% (SE: 10.3%) increase in the weekly number of completed tasks for those using the tool.[4] When we look at outcomes of secondary interest, our results support this interpretation, with a 13.55% (SE: 10.0%) increase in the number of code updates (commits) and a 38.38% (SE: 12.55%) increase in the number of times code was compiled. For Microsoft we observe both the developers' tenure and

[1]In addition, it is hard to predict further breakthroughs in the architecture of these models, which may lead to further improvements in quality or decrease the cost of inference and training.

[2]Prior academic work has shown that generative AI can pass mock interviews for coding jobs at Amazon in the top decile of human performance, performs at human level in a database of coding challenges that measure programming logic and proficiency, and can write entire programs for simple video games from several lines of instructions (Bubeck et al. 2023). Copilot is used by 1.3 million subscribers and more than 50,000 businesses.

[3]The exact implementation of these experiments was rather ad-hoc as they were driven by business considerations at these companies rather than research goals.

[4]Because of imperfect compliance, our preferred estimates use treatment status as an instrument for usage, so this is an estimate of the treatment on the treated.

their seniority as measured by job title. We find that Copilot significantly raises task completion for more recent hires and those in more junior positions but not for developers with longer tenure and in more senior positions. An important question is whether AI will be more beneficial to low-productivity or high-productivity workers. Prior work has shown that when workers are conducting the same tasks, generative AI helps lower-ability or lower-experience workers more (e.g., Brynjolfsson, Li, and Raymond 2023; Noy and Zhang 2023) but others find that both generative AI (Otis et al. 2024) and an AI tool designed to assist material science research (Toner-Rodgers 2024) benefit productive workers the most. Like the first set of papers, we find that generative AI increases the productivity of lower-ability workers more, even when workers are performing tasks according to their tenure or seniority.

Our preferred estimate pools estimates across all three experiments and places more weight on periods with larger differences in treatment status. We make these choices because our analysis must confront challenges related to statistical power despite the large number of software developers in the experiments. These challenges arise due to large variation in measured outcomes and factors that reduce the take-up and duration of the three experiments.[5] The experiment at Microsoft started before Copilot was widely known (and before the release of ChatGPT), and initial uptake was low. Shortly after a larger fraction of developers in the treatment group started using it, the control group was also allowed access. At Accenture, only a few hundred developers participated in the experiment. Lastly, at Anonymous Company, the treatment consisted of a staggered rollout with differences in treatment status lasting only a short time.

Most studies of the impact of generative AI tools on worker productivity have been conducted in controlled lab-like experiments (Campero et al. 2022; Noy and Zhang 2023; Peng et al. 2023; Vaithilingam, Zhang, and Glassman 2022). In a lab-in-the-field experiment on consultants employed by Boston Consulting Group, Dell'Acqua et al. (2023) find that productivity on 18 tasks designed to mimic the day-to-day work at a consulting company increased by 12%-25%. Evidence from these experiments generally suggests significant productivity effects of generative AI. The exception is Vaithilingam, Zhang, and Glassman (2022), which did not find a statistically significant difference in completion time.

While lab experiments offer a valuable opportunity to examine the short-term implications of generative AI, challenges and complex interactions arise when these tools are deployed in real-world environments (Jaffe et al. 2024). There are some observational studies of the effects of generative AI in an actual workplace setting (Hoffmann et al. 2024; Yeverechyahu, Mayya, and Oestreicher-Singer 2024). For instance, Brynjolfsson, Li, and Raymond (2023) find that an AI-based conversational assistant increases the productivity of customer chat support agents by 14%. The drawback of these studies is that they do not have the benefit of random experimental assignment of these technologies.

Our work complements both the literature on lab experiments as well as these observational studies by studying the impact of generative AI using a field experiment in an actual workplace setting. To date, there is still a dearth of experimental studies exam-

[5]We observe large variation in the output of software developers due to significant heterogeneity in their seniority, with more senior managers being less likely to engage in coding activities.

ining the effect of generative AI in a field setting. We fill this gap in the literature by examining a field experiment with high-skilled and highly paid knowledge workers, a group that is particularly relevant given the prediction that high-skilled jobs will be most affected by this technology. Although we examine a different part of the skill distribution and use experimental variation rather than a staggered introduction, we find similar productivity increases as Brynjolfsson, Li, and Raymond (2023). Furthermore, like them, we also find suggestive evidence that these gains are primarily driven by improved output from recent hires and employees in more junior roles.

More generally, we contribute to the literature studying the productivity and on-the-job performance of software developers (Cowgill et al. 2020; Emanuel, Harrington, and Pallais 2023; Murciano-Goroff 2022).

2 Setting and Experiments

2.1 What Is AI-Assisted Software Development?

AI Assistants for software development offer intelligent code suggestions and auto-completion within integrated development environments. Prominent examples include GitHub Copilot, Cursor, and Replit Ghostwriter. In our study, we examine the effects of one of these tools, GitHub Copilot. GitHub Copilot was developed by GitHub in partnership with OpenAI. It was available for "technical preview" in June 2021 and publicly available in June 2022, just a few months before the first of the experiments we analyze.[6] The development of Copilot involved combining advanced machine learning techniques and natural language processing. A substantial amount of code from public GitHub repositories was used to train Copilot. This extensive dataset allowed the AI model to learn from real-world coding practices, patterns, and styles across various programming languages and frameworks. See Nagle et al. (2023) for an in-depth overview of the origins and evolution of GitHub Copilot.

Copilot integrates with the software that developers use for coding. As developers write software code or plain text comments, Copilot analyzes the context and generates relevant code snippets, comments, and documentation. It can autocomplete code that developers might manually type or suggest snippets they would otherwise need to search for online. This capability can save developers time and potentially improve code quality by offering suggestions the developer might not be aware of. However, like all tools based on Large Language Models (LLMs), Copilot can make mistakes. If developers rely on it without review, it could potentially introduce errors or decrease code quality. While general-purpose LLMs like ChatGPT can also help with software development, they are less specialized and don't integrate with standard coding tools.

[6]ChatGPT, the first general-purpose public tool of this class of AI models was released on November 30, 2022.

2.2 Experiments

We analyze three randomized experiments conducted with software developers at Microsoft, Accenture, and Anonymous Company. In the Microsoft and Accenture experiments, one group of developers (the treated group) was randomly assigned to be able to access GitHub Copilot, whereas the other group (the control group) did not have access to the tool for a period of eight (Microsoft) or four (Accenture) months. In the Anonymous Company experiment, all users gained access to the tool over a period of two months, but access dates were randomized, with some teams gaining access six weeks before others.

Microsoft The experiment at Microsoft started in the first week of September 2022, involving a sample size of 1,746 developers primarily located in the United States. Of these developers, 50.4% were randomly selected to receive access to Github Copilot. Randomization was implemented at both the individual and the team levels.[7] In particular, 616 developers were individually randomized and 1,130 developers were randomized at the team level, with an average team size of 6.2. The developers work on building a wide range of software within Microsoft, with tasks that include engineering, designing, and testing software products and services. They occupy various positions in the company, ranging from entry-level developers to team managers. They may work in a team or individually, depending on their task and team structure.

Participants in the treated group were informed via email about the opportunity to sign up for GitHub Copilot. The email also included information introducing GitHub Copilot as a productivity-enhancing tool and outlining its potential impact on their coding tasks (see Figure 9 in Appendix). Beyond this email, treated participants did not receive any specific instructions regarding their workload or workflow to ensure they use GitHub Copilot in their natural work environment. Control group participants did not receive any communication as part of the study, even when they were eventually allowed access.[8] The experiment ended on May 3rd, 2023, as growing awareness of AI-assisted coding tools led the control group participants to seek access to Copilot.

Accenture The Accenture experiment started in the last week of July 2023 and included a number of Accenture offices located in Southeast Asia. Randomization occurred at the developer level, with 61.3% of the 320 developers assigned to the treatment group. Treatment group participants were informed over email that they were eligible to sign up for GitHub Copilot. They also participated in a training session, which explained what GitHub Copilot is, how to use it, and the potential benefits. Finally, the participating managers were asked to encourage their reports' adoption of GitHub Copilot. The control group was granted access to Copilot in December 2023, though uptake was lower than in the treatment group.

Anonymous Company The Anonymous Company experiment started in October 2023. It involved 3,054 developers who were all eventually invited to use Copilot. The invita-

[7]We account for this randomization structure in calculating our standard errors below.

[8]A small number of developers in the control group nevertheless got access to Copilot because they were working on related tools.

tion dates were randomized, with new invites being sent out weekly between September 2023 and October 2023.

2.3 Variables and Outcome Measures

Measuring the productivity of modern knowledge work is notoriously difficult. Our setting has the advantage that almost all professional software development follows a highly structured workflow, where specific tasks are defined and tracked through version control software. This makes internally defined goals quantifiable. All three participating organizations use the version control software GitHub. By observing the developers' GitHub activity, we can observe many of the variables that are part of their workflow.

A main outcome of interest is "pull requests", which can be thought of as a unit of work for software developers. Within an organization, the scope of a pull request is likely to remain relatively stable over time, shaped by organizational norms and conventions, even though different organizations may define this scope differently. For instance, a pull request may be asking for a feature to be added to a larger software project. A pull request will lead to a code review, often by a more senior software developer. If this review is passed, the code will be merged and thereby become part of the larger software project.

We use three additional outcome variables related to the developers' workflow. Before submitting a pull request, a developer will work separately on her code, tracking smaller changes through "commits." Periodically, the developer will "build" the code they are working on, and we can observe whether it compiled successfully. While commits and builds do not directly correspond to a deliverable, we expect them to be nevertheless monotone in the amount of accomplished work. Lastly, we use the build success rate as a measure of code quality.

In addition to these output measures, we observe how developers use Copilot. For each developer who uses Copilot, we observe both the number of suggestions by Copilot and the number of accepted suggestions.

For the Microsoft experiment only, we also see the hire date of developers and their level at the company, allowing us to separate the analysis by tenure and seniority.

Table 1 shows summaries of the treatment and control groups across all three experiments, as well as balance tests for the outcome variables. With the exception of commits in the Accenture experiment, randomization successfully balanced the average pre-treatment outcomes across the control and treatment groups. However, the table also shows that for all outcomes (with the exception of the Build Success Rate), the standard deviation exceeds the pre-treatment mean, and sometimes by a lot. This high standard deviation is driven by a large fraction of developer-weeks where the outcome variables are zero. Figure 1 shows the distribution of pull requests conditioning on weeks with at least one pull request. The high fraction of zeros, however, limits our power to detect effects in the experimental regressions below (and will be reflected in a larger standard error.)

	Control		Treatment			
	Mean	SD	Mean	SD	Difference	p-value
Panel A: Microsoft						
Pull Requests	0.86	1.49	0.87	1.50	0.01	0.88
Commits	9.43	14.86	9.36	14.80	-0.07	0.94
Builds	7.76	12.99	7.67	12.73	-0.09	0.91
Build Success Rate	0.72	0.30	0.75	0.29	0.02	0.33
Short Tenure	0.48	0.50	0.52	0.50	0.04	0.23
Junior Level	0.55	0.50	0.61	0.49	0.06	0.03**
Panel B: Accenture						
Pull Requests	0.13	0.47	0.14	0.47	0.00	0.85
Commits	2.56	6.00	3.64	7.25	1.08	0.01**
Builds	0.96	2.54	1.10	2.68	0.14	0.38
Build Success Rate	0.51	0.37	0.54	0.38	0.03	0.40
Panel C: Anonymous						
Pull Requests	0.73	1.23	0.73	1.19	-0.00	0.99

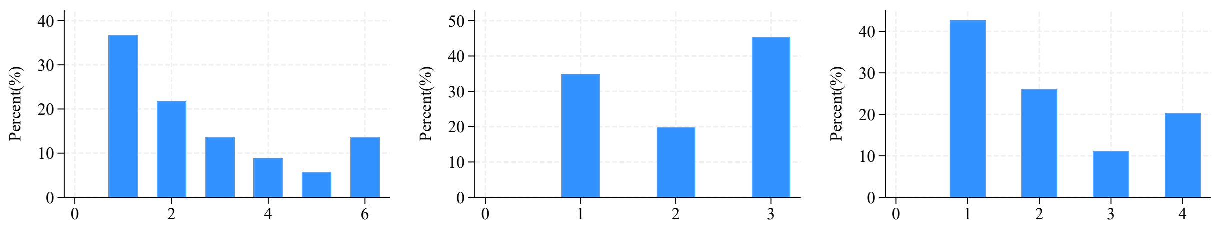
Table 1: Balance Table

Notes: This table presents a comparison of pre-experimental outcomes in control and treatment groups across experiments. 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 a difference in means. We do not present other outcome measures in Panel C because we do not have access to these data. The differences in p-values are calculated using standard errors clustered at the level of treatment assignment, which varies across experiments (Microsoft: mixed team-level and individual assignment; Accenture: individual assignment; Anonymous Company: team-level assignment.)

3 Adoption of Copilot

This section reports the adoption of Copilot in the experiments. Understanding adoption patterns is important for assessing the effectiveness of the experiments in generating random variation in Copilot usage. Furthermore, these patterns offer insights into the adoption of AI tools in the workplace. We define the adoption period as the first time a software engineer uses GitHub Copilot and consider an engineer as having adopted the tool even if they later stop using it. This approach captures the initial willingness to try and use the technology.

Figure 2 presents the cumulative adoption rates for the three experiments. In Panel (a), we observe that during the first two weeks of the Microsoft experiment, only 8.5 percent of the treated group signed up for GitHub Copilot. This low adoption rate might have been due to inattentiveness to the initial email notification. Consequently, Microsoft sent two additional email reminders on Feb 15th, 2023, and Feb 28th, 2023. These additional emails increased the take-up rate to 42.5% within two weeks of being sent. The initial compliance in the control group was not perfect, with 0.5 percent of



(a) Microsoft (37.09% $>$ 0) (b) Accenture (12.90% $>$ 0) (c) Anonymous (34.01% $>$ 0)

Figure 1: Distribution of Pull-Requests (Conditional on Above Zero)

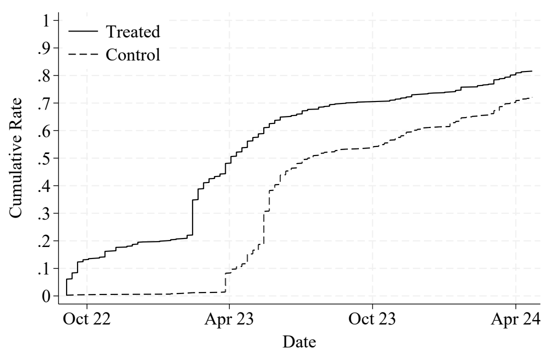
Notes: This figure provides, for each experiment, a bar chart depicting the distribution of our primary outcome variable, the number of completed pull requests. The unit of observation is a developer-week. We plot this number after winsorizing at the 95-th percentile; its unwinsorized maximum is 892 for Microsoft, 70 for Accenture and 876 for Anonymous. Furthermore, we condition on observations with non-zero completed pull requests.

individuals in the control group adopting Copilot. At the conclusion of the experiment in April 2023, the control group was given access to Copilot, and we observed a rapid adoption rate in the control group. However, in Jan 2024, adoption in the control group (64.0%) remained below the adoption in the treated group (75.6%), providing limited long-run variation in adoption generated by the emails encouraging adoption sent as part of the experiment (which were only ever sent to the treatment group.)

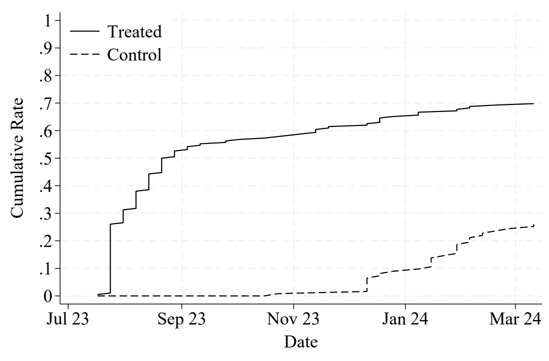
Panel (b) reports the adoption rate of Copilot in the Accenture experiment. Unlike in the Microsoft Experiment, in the Accenture Experiment, the nudge to adopt Copilot was paired with training on how to use the software, and hence, the treated group's adoption rate was more rapid at the start of the experiment, but after 1-2 months, it plateaus at around 60%. When the control group is allowed to adopt in December 2023, we see a slower but steady increase in adoption rate amongst control users. By April 2024, the adoption rate for the treated group is 69.4%, while the adoption rate for the control group is 24.4%.

Panel (c) shows the staggered invitation to access Copilot (represented by the solid line) and the cumulative adoption rate among all participants (dashed line). As evident from the plot, all developers gained access to Copilot after six weeks. During the invitation rollout in September and October, we observed a steady increase in adoption as developers got access to Copilot. Following the initial increase in adoption, the adoption rate plateaued, showing only small, steady increases for the remainder of the experiment.

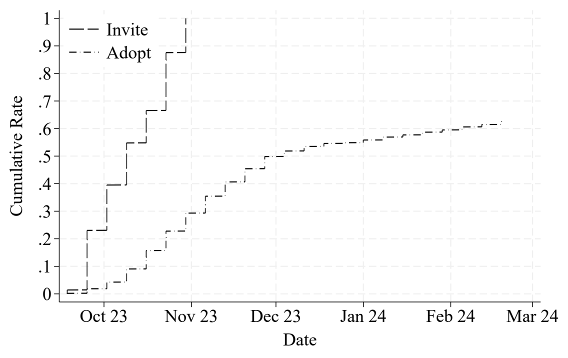
It is worth noting that adopting Copilot is relatively easier and less costly compared to other AI tools in the workplace. It does not require any complementary investment, can be adopted individually, and integrates directly into the existing development environment. Despite this, the adoption rate is significantly below 100% in all three experiments, with around 30-40% of the engineers not even trying the product. Moreover, we observe that the adoption rates are remarkably similar across the experiments. This suggests that factors other than access, such as individual preferences and perceived quality of the tool, play important roles in engineers' decisions to use this tool (Dietvorst, Simmons, and Massey 2015, 2018).



(a) Microsoft Experiment



(b) Accenture Experiment



(c) Anonymous Company Experiment

Figure 2: Cumulative Adoption Rates

Notes: The first two graphs show the cumulative rate of adoption over time for software developers in both the treatment and control groups across various experiments. In the Anonymous Company experiment (Panel c), unlike the other experiments, all developers were granted access to Copilot in a staggered fashion, with the order of access randomized among participants. Hence, we show the cumulative fraction of users invited to participate and who adopted Copilot.

4 Empirical Strategy & Main Results

4.1 Empirical Strategy

Our empirical strategy exploits the experimental variation while accommodating imperfect compliance by utilizing the experimental assignment as an instrument for the adoption of GitHub Copilot. For each experiment, we observe data at the developer-week level. To gain precision, we control for both developer and week-fixed effects (to account for, e.g., differences in developer skills and holidays). This leaves us with the following regression as our main specification:

$$y_{it} = \beta D_{it} + \mu_i + \gamma_t + \epsilon_{it}. \tag{1}$$

Here, β is the coefficient of interest, D_{it} is an adoption dummy that turns on after a developer first uses GitHub Copilot, μ_i is a developer fixed effect, and γ_t is a week fixed effect.[9] We estimate Equation 1 by two-stage least squares (2SLS), instrumenting D_{it} with a dummy Z_{it} that turns on for all developers randomized into the treatment group after the start of the experiment.

Before we move on to the results, we have to discuss a key complication: our data comes from experiments where the control group was eventually also allowed to access GitHub Copilot. This is not a challenge for identification, but it reduces the power of our instrument if we employ the naive strategy detailed above. In particular, consider a hypothetical experiment that lasts just one month: at $t = 0$, developers are randomized into treatment and control groups, where control is not allowed access to Copilot until $t = 4$. Suppose further that, starting at $t = 4$, the differences in uptake between the two groups start declining over time, asymptoting a zero uptake difference. If we naively estimate (1) by 2SLS in this setting, the power of our instrument (and hence the precision of our estimates) will be strictly declining in the number of periods we observe – in the limit, with infinite periods, our instrument violates the relevance condition because initial treatment assignment eventually fails to predict uptake. One potential solution to this dilemma involves cutting out some data and using only $t \in \{1, \ldots, 4\}$ to estimate the model. During this period, the instrument has maximal relevance. However, to the extent that there is still an adoption difference between treatment and control groups at $t = 5$, this strategy is wasteful in that it does not exploit all possible identifying variations.

To avoid ad-hoc decisions about how many periods after the experiment ends should be included in the analyzed data, we weight the 2SLS estimates by the (period-by-period) difference in adoption across treatment and control groups. The resulting weighted IV regression gracefully handles the issue of declining instrument relevance, and it has been previously proposed and analyzed in the context of uptake differences across individuals by (Coussens and Spiess 2021; Huntington-Klein 2020); in the context of uptake differences over time, a similar strategy was employed by Bloom et al. (2012) to improve precision.

The impact of our weighting on the interpretation of our results is straightforward: the weighted regression weights periods based on the difference in Copilot adoption between control and treatment groups. As we can see in Appendix Figure 8, the weighted IV estimates place greater emphasis on treatment effects during periods like March 2023 in the Microsoft experiment, where there was a significant difference in adoption between the treatment and control groups. Thus, to the extent that one would expect treatment effects that are heterogeneous over calendar time (e.g., because Copilot improved over time), the weighting will affect which estimand our estimator is targeting. If, on the other hand, treatment effects are unchanged over the course of the experiment, the weighting will purely improve the precision of our estimates without affecting their interpretation.

[9]For the initial phase of the Microsoft experiment, we do not observe intensive usage data. Hence, we say a developer at Microsoft has adopted Copilot after they either register to use it (relevant in the initial phase) or we see any usage of Copilot (relevant in the later phase.)

4.2 Results

We present our results in Table 2, split by experiment. We divide each coefficient by the mean in the control group for that variable and multiply by 100 so that coefficients in this table can be interpreted as percentage effects.[10] To enable easy comparison to observational studies such as Brynjolfsson, Li, and Raymond (2023), we present both difference-in-difference (DiD) estimates that do not exploit experimental variation and our main (weighted) results (W-IV), which do.[11] Interestingly, we find that our difference-in-difference estimates are sometimes larger and sometimes smaller than the experimental estimates, emphasizing that the reasons for the divergence between experimental and observational estimates may differ across companies.

At Microsoft, we find positive effects of Copilot on the number of completed pull requests, the number of commits made, and the number of times that code was built (i.e., compiled.) Focusing on the more credible experimental estimates, however, only the effect on the number of pull requests is statistically significant at conventional significance levels. We find no negative effect on the build success rate, which would be negatively affected if Copilot was writing code that does not compile and these mistakes are not caught by developers. Although not always statistically significant, we find directionally similar effect sizes at Accenture and Anonymous Company, with the exception of the build success rate, which is negative at Accenture.

In the final column, we combine estimates across experiments (taking the precision-weighted average across our three experiment-by-experiment estimates) to get the most precise estimate of the effect of the coding assistance tool.[12] While standard errors are consistently large and the effect sizes differ across the three different companies, we find evidence of productivity-enhancing effects of GitHub Copilot: on average, the number of weekly pull requests made by developers increases by 26.08% (SE: 10.3%), the number of weekly commits increases by 13.55% (SE: 10.0%), and the number of weekly builds increases by 38.38% (SE: 12.55%).

A less optimistic interpretation of the increase in builds is that developers may engage in more trial-and-error coding, accepting Copilot's suggestions and then compiling the project to check for errors. Such a change in coding style could lead to lower-quality code in the long run and undermine efficiency gains in the quantity of code. However, our results on build success rate only (weakly) support such an interpretation for the Accenture experiment.

Before moving on, we discuss an additional experiment run at Accenture (not exhibited in the above table) that was abandoned due to a large layoff affecting 42% of participants, resulting in a lack of data on Copilot usage (and hence adoption status). Because of these data quality issues, we relegate this experiment to Appendix D. How-

[10]We do not take logs because of a large number of person-weeks that are zero for each variable. However, we report results from a Poisson difference-in-differences regression in Appendix A.

[11]We relegate unweighted results (which are less precise) to Appendix A.

[12]Thus, $\hat{\beta}_{Pooled} = \left(\sum_{e \in E} 1/\hat{\sigma}_e^2\right)^{-1} \sum_{e \in E} \hat{\beta}_e/\hat{\sigma}_e^2$ where $E = \{\text{Microsoft, Accenture, Anon. Comp}\}$, $\hat{\beta}_e$ refers to the estimate in experiment e, and $\hat{\sigma}_e$ refers to the standard error. We combine the estimates in this way instead of running a pooled regression because of the different experimental designs across the three companies – the staggered rollout at Anonymous Company cannot be easily combined with the treatment/control split at the other two.

Outcome	Microsoft		Accenture		Anon. Comp.		Pooled	
	DiD	W-IV	DiD	W-IV	DiD	W-IV	DiD	W-IV
Pull Requests	7.63***	27.38**	52.65***	17.94	1.70	54.03	6.24***	26.08**
	(2.49)	(12.88)	(9.46)	(18.72)	(2.47)	(42.63)	(1.72)	(10.3)
Commits	7.03***	18.32	12.85	-4.48	-	-	7.25***	13.55
	(2.32)	(11.25)	(11.62)	(21.88)	-	-	(2.28)	(10.0)
Builds	7.11***	23.19	39.66***	92.40***	-	-	8.23***	38.38***
	(2.65)	(14.20)	(14.03)	(26.78)	-	-	(2.6)	(12.55)
Build Success	-0.65	-1.34	-20.72***	-17.40**	-	-	-1.13	-5.53
Rate	(0.79)	(4.23)	(5.06)	(7.12)	-	-	(0.78)	(3.64)

Table 2: Experiment-by-Experiment Results

Notes: This table provides difference-in-difference estimates (DiD) and weighted IV (W-IV) estimates of the effect of GitHub Copilot adoption on various productivity measures. Each entry corresponds to an estimate of β in (1) expressed as a percentage of the control mean. DiD estimates instrument adoption D_{it} with itself. W-IV estimates instrument D_{it} with experimental assignment Z_{it} and weight by differences in adoption status across treatment and control (see main text.) Standard errors are clustered at the level of treatment assignment, which varies across experiments (Microsoft: mixed team-level and individual assignment; Accenture: individual assignment; Anonymous Company: team-level assignment.) "Pooled" is the precision-weighted average of the other three estimates. We combine the estimates in this way instead of running a pooled regression because of the different experiment designs – the staggered rollout at Anonymous Company cannot be easily combanied with the treatment/control split at the other two.)

ever, if we conservatively impute Copilot adoption status, we find a negative, statistically insignificant, point estimate of -39.18% (SE: 36.78%) on the number of completed pull requests. The estimates on the number of commits 43.04% (SE: 38.8%) and builds 12.33% (SE: 53.6%) are both positive, though also not statistically significant. The combined results do not change much when we include this second Accenture experiment: the number of pull requests made by developers increases by 21.34% (SE: 9.92%), the number of commits increases by 15.39% (SE: 9.69%), and the number of builds increases by 37.03% (SE: 12.22%).

5 Heterogeneity

Previous literature has noted that productivity enhancements driven by large language models are heterogeneous across skill level and education. In particular, in the context of customer service and professional writing tasks, large language models have been found to help the least educated, least skilled workers the most (Brynjolfsson, Li, and Raymond 2023; Noy and Zhang 2023). In scientific and entrepreneurial contexts, however, the most productive workers have been found to benefit more (Otis et al. 2024; Toner-Rodgers 2024). Because we have access to developer characteristics for the Microsoft experiment, we can contribute to this open question in the literature.

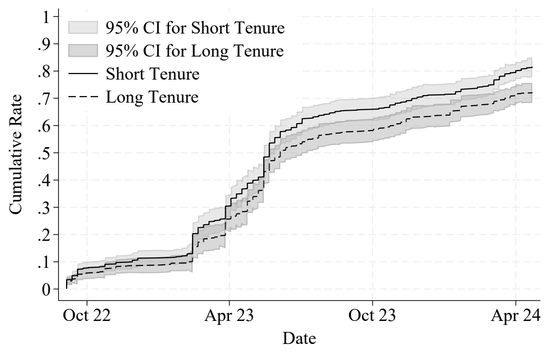
In particular, we now break out results by (i) the tenure, (ii) the level, and (iii) the pre-experiment productivity of employees at Microsoft[13]. We split developers into short and long tenure based on the median observed tenure in our data.[14] Any developers that have been with Microsoft for less than the median time at the start of the experiment are considered "short tenure," and all other developers "long tenure." Similarly, we split developers into "junior" and "senior" based on the level at which they are employed at the company. Finally, we split developers into "low pre-productivity" and "high pre-productivity" based on the number of pre-experiment pull requests we observe from them (using the median as the cutoff.)

We begin our analysis by considering the heterogeneity in adoption patterns in Figure 3. We see that short-tenure developers are 9.4 percentage points (SE: 2.2pp) more likely to adopt Copilot by the end of our sample period (81.6% vs 72.1%), consistent with prior research suggesting that younger workers (who naturally have lower tenure on average) are more likely to adopt new technologies (Meyer 2011). The same effect is at play for the junior developers, who are 4.7 percentage points (SE: 2.2pp) more likely to adopt (79.2% vs. 74.4%), though the adoption difference is slightly smaller in this dimension. Intriguingly, low pre-productivity developers are initially more likely to adopt, but this changes soon after the Control group is allowed to adopt, and by the end of our sample, high pre-productivity are slightly more likely to adopt.

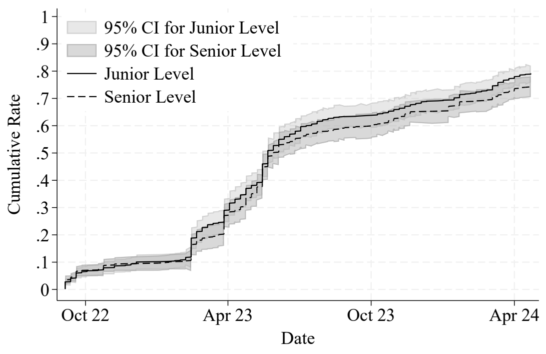
The next figure, Figure 4, reveals that employees of shorter tenure are more likely to continue using Copilot more than one month after initial adoption, suggesting that they

[13]We measure level as of March 1st, 2023, which is the earliest date available in our data.

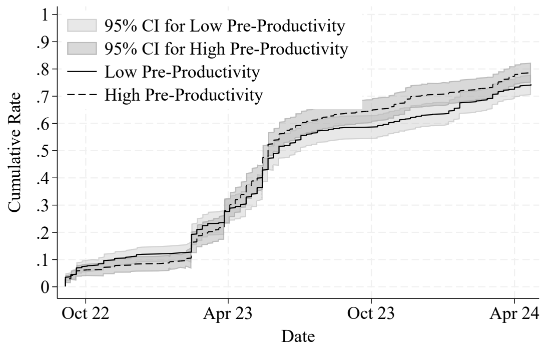
[14]As it is considered sensitive information, we cannot reveal the exact median tenure, but it is between 2 and 4 years.



(a) Adoption by Tenure



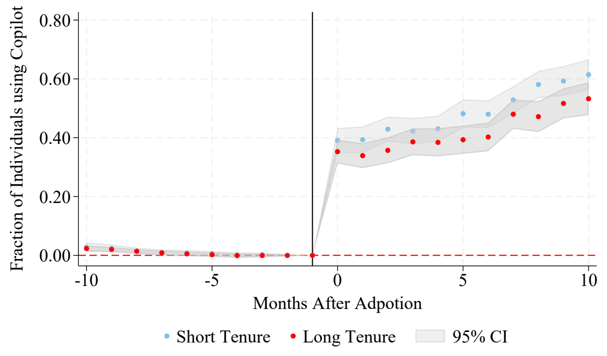
(b) Adoption by Level



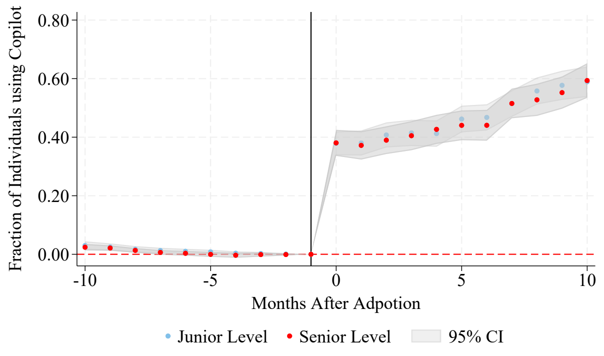
(c) Adoption by Pre-Productivity

Figure 3: Heterogeneity of Adoption of Copilot

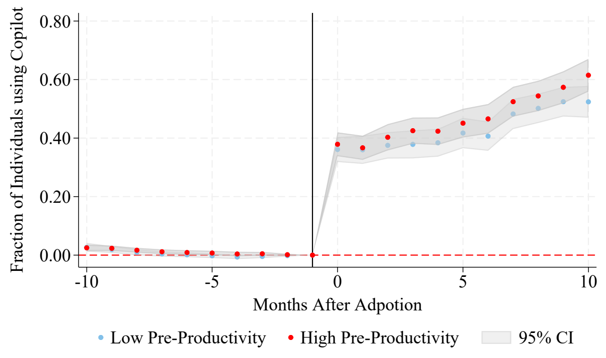
Notes: This figure explores heterogeneous adoption patterns of Copilot across developer tenure, level, and pre-period productivity. Panel (a) provides the adoption of Copilot over time broken out by whether a developer's tenure with Microsoft at the beginning of the experiment was below or above the median; panel (b) does the same for each level. Panel (c) splits out developers with the above- vs. below-median number of pull requests before the start of the experiment.



(a) Usage (Since Adoption) by Tenure



(b) Usage (Since Adoption) by Level



(c) Usage (Since Adoption) by Pre-Productivity

Figure 4: Heterogeneity of Usage of Copilot

Notes: This figure explores heterogeneous usage patterns of Copilot across developer tenure, level and pre-period productivity. We show event studies that detail the extensive margin, i.e., how likely a developer is to have used Copilot at all a given number of months after adopting Copilot; short-tenure developers are more likely to stick with Copilot.

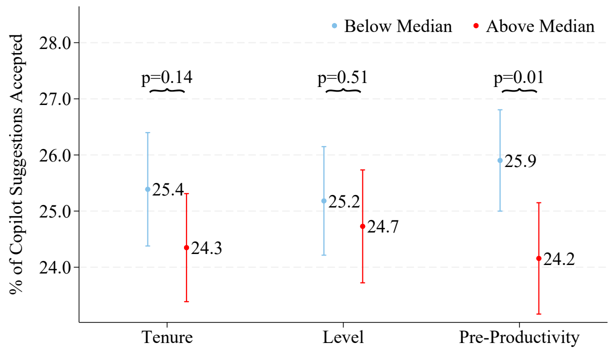


Figure 5: Heterogeneity of Fraction of Suggestions Accepted

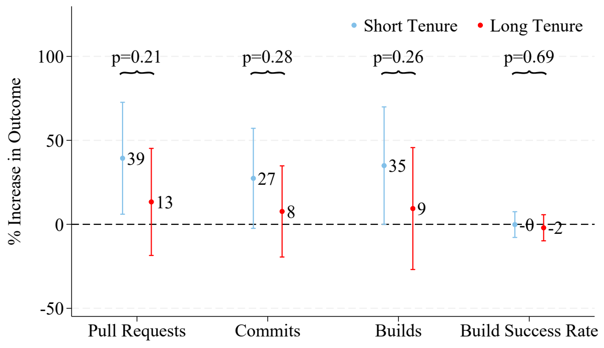
Notes: This figure shows the fraction of Copilot suggestions that are accepted by developers; short-tenure developers are slightly and less productive developers are much more likely to accept suggestions.

perhaps expect larger benefits from the technology than their more experienced counterparts. Judging by Figure 4(b), this effect does not seem present when comparing junior to senior developers. Figure 4(c) shows that, if anything, low-productivity developers are less likely to continue using Copilot, though the difference is (just) not statistically significant (p=.085). Finally, Figure 5 reveals that higher-tenure developers are approximately 4.3% (or 1.0 percentage points) less likely to accept code suggested by Copilot. When comparing junior to senior developers, this difference in acceptance rates is much smaller at 1.8% (or 0.5 percentage points), though it goes in the same direction of senior developers being less likely to accept AI suggestions. Intriguingly, there is a much larger difference across pre-productivity levels: developers who were less productive before the experiment are significantly more likely to accept any given suggestion.

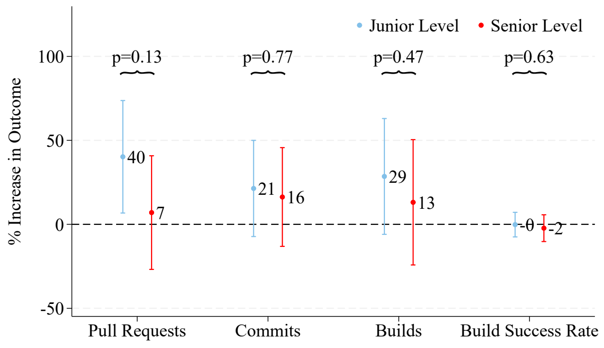
Moving on towards output measures in Figure 6, we see that the productivity-enhancing effects of Copilot are stronger for lower tenure and more junior developers. While our estimates are noisy and not statistically significant at conventional levels, the pattern is the same across all three main outcome measures: short-tenure developers increase their output by 27% to 39% while long-tenure developers have smaller gains of 8% to 13%. However, we note that because of our emphasis on the average effect of *initially* adopting Copilot and the patterns in Figure 4(a), the estimates for longer-tenure developers may be attenuated by a larger extent of developers abandoning the technology after an initial trial phase. Still, there is no difference in usage conditional on adoption between junior and senior developers in Figure 4(b), and still we see in 6(b) that junior developers increase their output by 21% to 40% while senior developers have more marginal gains of 7% to 16%. The estimates for above and below-median productivity developers are even noisier but directionally consistent.

6 Conclusion

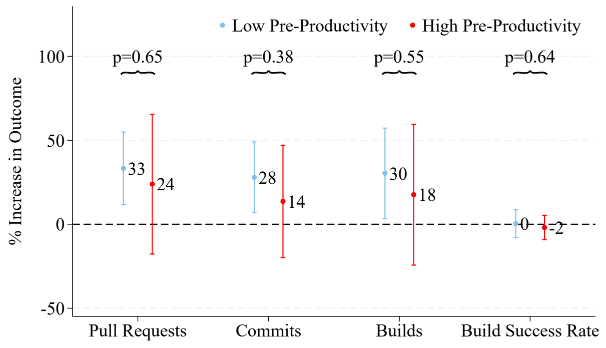
To summarize, we find that usage of a generative AI code suggestion tool increases software developer productivity by 26.08% (SE: 10.3%). We note that this estimate is substantially smaller than the 58% decrease Peng et al. (2023) find for the time to complete a software engineering task in the lab (which would correspond to almost twice as many tasks done in a given amount of time). It is perhaps unsurprising that the effect of the tool in real-world contexts is smaller than in the lab, since some types of coding tasks may be less readily assisted by Copilot. Additionally, coding is only part of a software developer's job, so only some of the time saved on coding tasks may be spent on additional coding. Our estimate is based on observing, partly over the years, the output of almost five thousand software developers at three different companies as part of their regular job, which strongly supports its external validity.



(a) Treatment Effects by Tenure



(b) Treatment Effects by Level



(c) Treatment Effects by Pre-Productivity

Figure 6: Heterogeneity of Copilot Effect (Weighted IV)

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) 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 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 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 all three outcome measures, the effects on productivity are stronger for short-tenure/junior/less productive developers, though the difference is typically not statistically significant.

References

Acemoglu, Daron (Apr. 2024). *The Simple Macroeconomics of AI*. Tech. rep. Prepared for Economic Policy. Massachusetts Institute of Technology.

Agrawal, Ajay, Joshua Gans, and Avi Goldfarb (2019). "Economic policy for artificial intelligence". In: *Innovation policy and the economy* 19.1, pp. 139–159.

Baily, Martin, Erik Brynjolfsson, and Anton Korinek (2023). "Machines of mind: The case for an AI-powered productivity boom". In.

Bloom, Nicholas, Benn Eifert, Aprajit Mahajan, David McKenzie, and John Roberts (Nov. 2012). " Does Management Matter? Evidence from India *". In: *The Quarterly Journal of Economics* 128.1, pp. 1–51. eprint: `https://academic.oup.com/qje/article-pdf/128/1/1/30624754/qjs044.pdf`.

Bresnahan, Timothy (2024). "What innovation paths for AI to become a GPT?" In: *Journal of Economics & Management Strategy* 33.2, pp. 305–316. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1111/jems.12524`.

Brynjolfsson, Erik, Danielle Li, and Lindsey Raymond (2023). "Generative AI at Work". In: *NBER Working Paper* w31161.

Brynjolfsson, Erik, Daniel Rock, and Chad Syverson (2021). "The productivity J-curve: How intangibles complement general purpose technologies". In: *American Economic Journal: Macroeconomics* 13.1, pp. 333–372.

Bubeck, Sébastien, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. (2023). "Sparks of artificial general intelligence: Early experiments with gpt-4". In: *arXiv preprint arXiv:2303.12712*.

Campero, Andres, Michelle Vaccaro, Jaeyoon Song, Haoran Wen, Abdullah Almaatouq, and Thomas W. Malone (2022). *A Test for Evaluating Performance in Human-Computer Systems*. arXiv: `2206.12390 [cs.HC]`.

Coussens, Stephen and Jann Spiess (2021). *Improving Inference from Simple Instruments through Compliance Estimation*. arXiv: `2108.03726 [econ.EM]`.

Cowgill, Bo, Fabrizio Dell'Acqua, Sam Deng, Daniel Hsu, Nakul Verma, and Augustin Chaintreau (2020). "Biased Programmers? Or Biased Data? A Field Experiment in Operationalizing AI Ethics". In: *Proceedings of the 21st ACM Conference on Economics and Computation*. Columbia Business School Research Paper Forthcoming. ACM, pp. 679–681.

Dell'Acqua, Fabrizio, Edward McFowland, Ethan R Mollick, Hila Lifshitz-Assaf, Katherine Kellogg, Saran Rajendran, Lisa Krayer, François Candelon, and Karim R Lakhani (2023). "Navigating the jagged technological frontier: Field experimental evidence of the effects of AI on knowledge worker productivity and quality". In: *Harvard Business School Technology & Operations Mgt. Unit Working Paper* 24-013.

Dietvorst, Berkeley J, Joseph P Simmons, and Cade Massey (2015). "Algorithm aversion: people erroneously avoid algorithms after seeing them err." In: *Journal of experimental psychology: General* 144.1, p. 114.

— (2018). "Overcoming algorithm aversion: People will use imperfect algorithms if they can (even slightly) modify them". In: *Management science* 64.3, pp. 1155–1170.

Eloundou, Tyna, Sam Manning, Pamela Mishkin, and Daniel Rock (2023). *GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models*. arXiv: `2303.10130 [econ.GN]`.

Emanuel, Natalia, Emma Harrington, and Amanda Pallais (2023). *The Power of Proximity to Coworkers: Training for Tomorrow or Productivity Today?*

Frank, Morgan R, David Autor, James E Bessen, Erik Brynjolfsson, Manuel Cebrian, David J Deming, Maryann Feldman, Matthew Groh, José Lobo, Esteban Moro, et al. (2019). "Toward understanding the impact of artificial intelligence on labor". In: *Proceedings of the National Academy of Sciences* 116.14, pp. 6531–6539.

Furman, Jason and Robert Seamans (2019). "AI and the Economy". In: *Innovation policy and the economy* 19.1, pp. 161–191.

Greenstein, Shane, Nathaniel Lovin, Scott Wallsten, Kerry Herman, and Susan Pinckney (Dec. 2024). *A Guide to the Vocabulary, Evolution, and Impact of Artificial Intelligence (AI)*. Technical Note 625-039. Harvard Business School.

Hoffmann, Manuel, Sam Boysel, Frank Nagle, Sida Peng, and Kevin Xu (July 2024). "Generative AI and Distributed Work: Evidence from Open Source Software". In: *Unpublished*. Version: July 8, 2024.

Huntington-Klein, Nick (2020). In: *Journal of Causal Inference* 8.1, pp. 182–208.

Jaffe, Sonia, Neha Parikh Shah, Jenna Butler, Alex Farach, Alexia Cambon, Brent Hecht, Michael Schwarz, and Jaime Teevan (July 2024). *Generative AI in Real-World Workplaces*. Tech. rep. MSR-TR-2024-29. Microsoft.

Meyer, Jeremy (2011). "Workforce age and technology adoption in small and medium-sized service firms". In: *Small Business Economics* 37.3, pp. 305–324.

Murciano-Goroff, Raviv (2022). "Missing women in tech: The labor market for highly skilled software engineers". In: *Management Science* 68.5, pp. 3262–3281.

Nagle, F, S Greenstein, M Roche, NL Wright, and S Mehta (2023). "CoPilots (s): Generative AI at Microsoft and GitHub". In: *Harvard Business School Case* 9, pp. 624–010.

Noy, Shakked and Whitney Zhang (2023). "Experimental evidence on the productivity effects of generative artificial intelligence". In: *Science* 381.6654, pp. 187–192.

Otis, Nicholas, Rowan Clarke, Solène Delecourt, David Holtz, and Rembrand Koning (Feb. 2024). "The Uneven Impact of Generative AI on Entrepreneurial Performance". In: *SSRN Electronic Journal*.

Peng, Sida (Oct. 2024). *The Effects of Generative AI on High Skilled Work: Evidence from Three Field Experiments with Software Developers*. AEA RCT Registry.

Peng, Sida, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer (2023). "The impact of ai on developer productivity: Evidence from github copilot". In: *arXiv preprint arXiv:2302.06590*.

Toner-Rodgers, Aidan (2024). "Artificial intelligence, scientific discovery, and product innovation". In: *arXiv preprint arXiv:2412.17866*.

Vaithilingam, Priyan, Tianyi Zhang, and Elena L Glassman (2022). "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models". In: *Chi conference on human factors in computing systems extended abstracts*, pp. 1–7.

Yeverechyahu, Doron, Raveesh Mayya, and Gal Oestreicher-Singer (July 2024). "The Impact of Large Language Models on Open-source Innovation: Evidence from GitHub

Copilot”. In: *Unpublished*. Available at SSRN: `https://ssrn.com/abstract=4684662` or `http://dx.doi.org/10.2139/ssrn.4684662`.

Supplemental Appendix

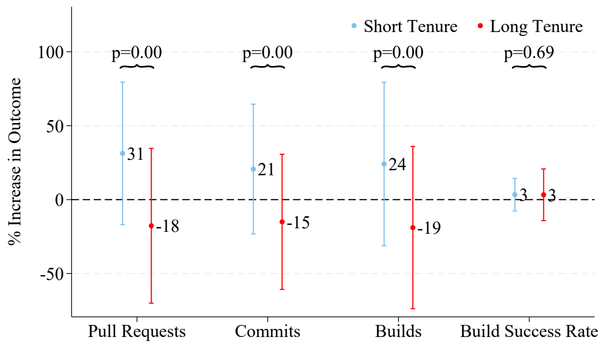
A Alternative Specifications & Robustness

This appendix provides a version of Table 2 that includes additional, alternative specifications, including unweighted IV. We also provide a version of Figure 3 that does not use weights when estimating 1. The resulting results have wider confidence intervals but point in the same direction as our main estimates. We also exhibit the weights that underlie Table 2 and Figure 3.

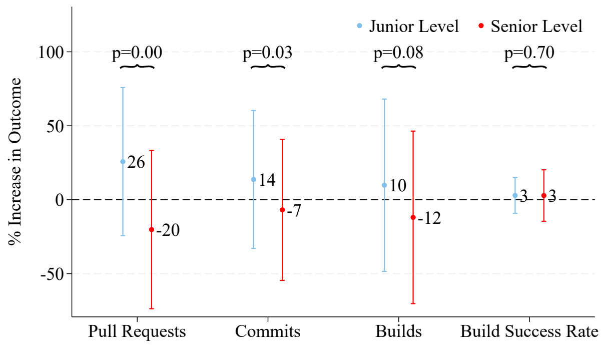
Outcome	Microsoft				Accenture				Anon. Comp.				Pooled			
	DiD	DiD-P	IV	W-IV	DiD	DiD-P	IV	W-IV	DiD	DiD-P	IV	WIV	DiD	DiD-P	IV	W-IV
Pull Requests	7.63***	6.81***	10.53	27.38**	52.65***	22.54**	15.97	17.94	1.70	2.77	54.03	54.03	6.24***	5.23***	18.73	26.08**
	(2.49)	(2.52)	(24.82)	(12.88)	(9.46)	(9.35)	(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	-0.65	-0.66	3.92	-1.34	-20.72***	-19.59***	-18.10*	-17.40**	-	-	-	-	-1.13	-1.05	-5.39	-5.53
Rate	(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 3: Alternative Specifications for Experimental Results

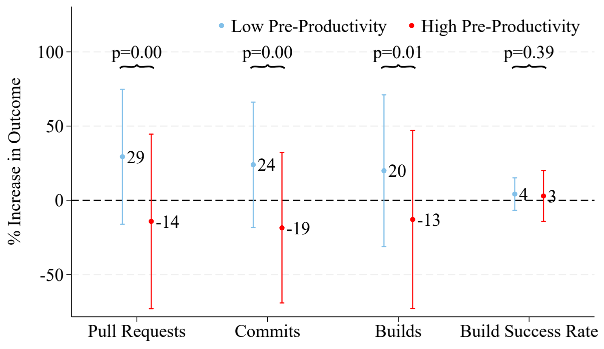
Notes: This table builds on Table 2 by reporting the results of additional specifications. Each entry can be interpreted as an estimate of the percentage effect of adoption of GitHub Copilot. DiD is like in Table 2, DiD-P is like DiD but runs a Poisson model and then reports $100 * (\exp(\beta) - 1))$, IV is like W-IV in Table 2 but without weighting the regression by adoption differences, W-IV is like in Table 2.



(a) Treatment Effects by Tenure



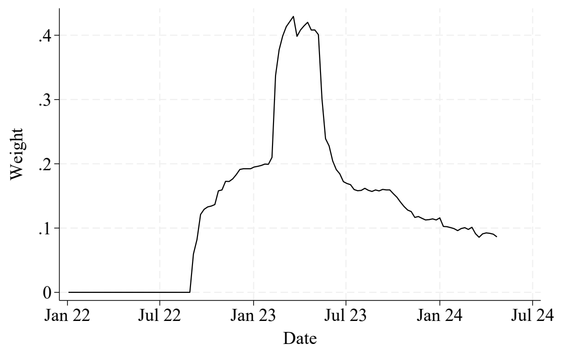
(b) Treatment Effects by Level



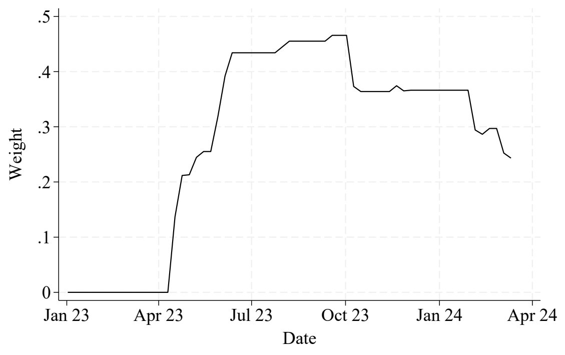
(c) Treatment Effects by Pre-Productivity

Figure 7: Heterogeneity of Effect of Copilot (Unweighted IV)

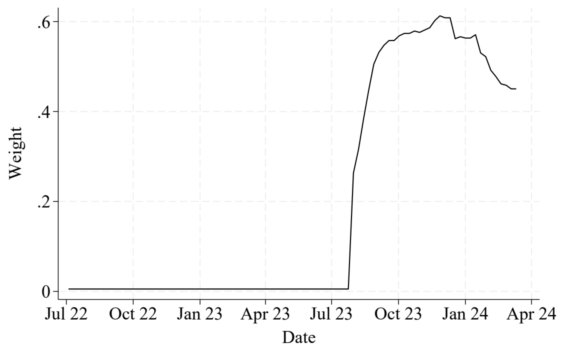
Notes: This figure provides unweighted IV estimates of the effect of adopting Copilot on total number of pull requests, commits, builds, and build success rate broken out by (a) whether a developer's tenure with Microsoft at the beginning of the experiment was below median (short tenure) or above median (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 all 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



(c) Accenture Experiment

Figure 8: Regression Weight

Notes: This figure provides the weights used in the W-IV estimates underlying Table 2. 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.

B Data Cleaning

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

B.1 Microsoft

In the original sample of the dataset, we have 1,746 individuals. We kept only software engineers, which leaves us 1,538, and we dropped people who switched organizations, leaving us 1,522. Finally, we dropped one individual who adopted 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, and they naturally weaken the strength of the instrument.

B.2 Accenture

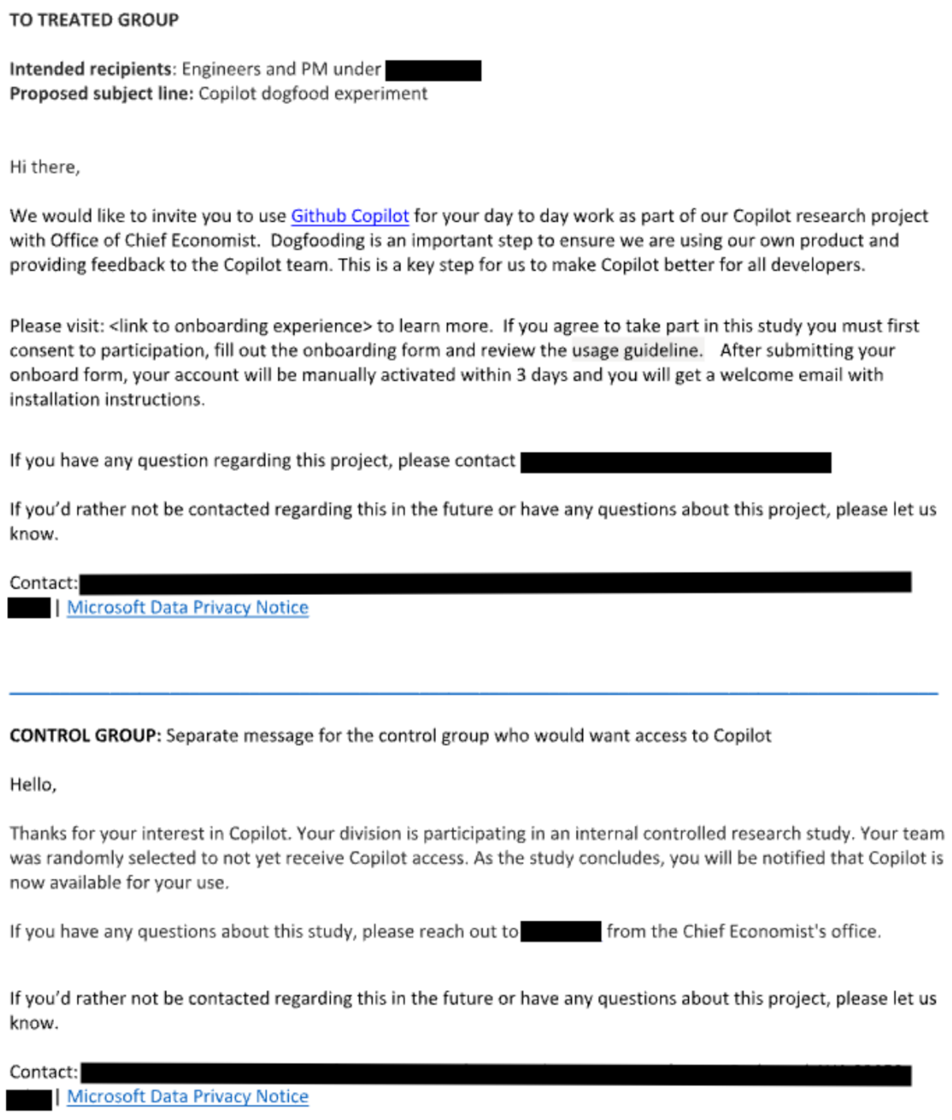
We drop individuals who have no record of data and people who left the company. We start with the original dataset with 369 individuals, and after dropping individuals with no outcome measure found we are left with 320, and after dropping individuals who left the company we are left with a final sample 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.

B.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.

C Experiment Details



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.

	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 4: Balance Table for First Accenture Experiment
Notes: This table presents a comparison of pre-experimental outcomes in control and treatment groups across experiments. 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 differences in p-values are calculated using standard errors clustered at the level of treatment assignment.

D The First Accenture Experiment

We do not discuss at length in the main text another experiment that was run by Accenture in April 2023 and 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 some 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 4 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 10. 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 5. 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 on the number of commits 43.04% (SE: 38.8%) and builds 12.33% (SE: 53.6%) are both positive, though also not statistically significant.

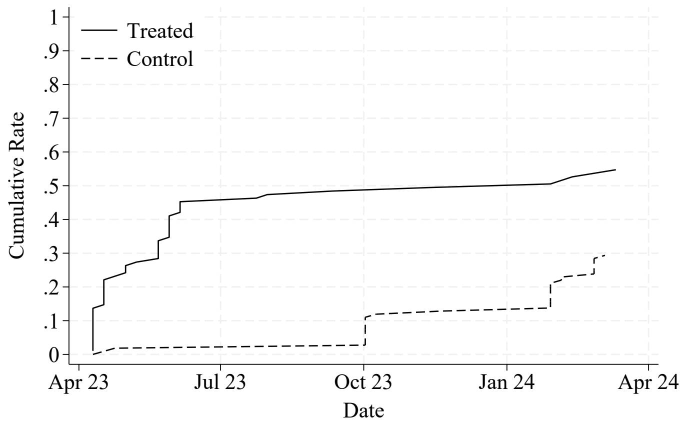


Figure 10: Cumulative Adoption Rates for First Accenture Experiment
Notes: This graph shows the cumulative rate of adoption over time for software developers in both the treatment and control groups in the first Accenture experiment. Note that we are imputing that nobody in the Control group adopts Copilot until October '23.

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 5: 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, and 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 '23.)