An MIT Exploration of Generative AI • From Novel Chemicals to Opera

# The Productivity Effects of Generative AI: Evidence from a Field Experiment with GitHub Copilot

**Kevin Zheyuan Cui[1] Mert Demirer[2] Sonia Jaffe[3] Leon Musolff[4] Sida Peng[3] Tobias Salz[5]**

[1]Princeton University, [2]Sloan School of Management, Massachusetts Institute of Technology, [3]Microsoft, [4]The Wharton School, University of Pennsylvania, [5]Massachusetts Institute of Technology

**MIT**

**Published on:** Mar 27, 2024

**DOI:** https://doi.org/10.21428/e4baedd9.3ad85f1c

**ABSTRACT**

We are providing a preview of a project that analyzes two field experiments with 1,974 software developers at Microsoft and Accenture to evaluate the productivity impact of Generative AI. As part of our study, a random subset of developers was given access to GitHub Copilot, an AI-based coding assistant that intelligently suggests 'completions' for code. Our preliminary results provide suggestive evidence that these developers became more productive, completing 12.92% to 21.83% more pull requests per week at Microsoft and 7.51% to 8.69% at Accenture (depending on specification). Due to low compliance in the Microsoft experiment and internal organizational changes at Accenture, our estimates are not very precise and only reach statistical significance if we weight more heavily periods in which Copilot uptake differs across control and treatment.

**Keywords:** generative AI, large language models, technology adoption, worker productivity



---

# 1. Introduction and Motivation

The recent rapid progress in the development of large language models is seen by many as a watershed moment for the deployment of AI. Many expect that the deployment of advanced AI will have widespread implications for high-skilled labor markets (Webb 2019; Felten et al. 2019; Brynjolfsson and Mitchell 2017). Assistive tools for developers are a particularly promising application of these models (Bubeck et al. 2023), and several such products are already commercially available. Software developers, therefore, serve as an interesting case study to glean valuable lessons that may apply to other high-skilled professions. In this project, we study the impact of generative AI tools on software development by analyzing two 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 about two thousand software developers.

While we are unable to talk about its architecture, Copilot benefits at least in part from the gains in performance that have been achieved with GPT-4. Prior academic work has shown that GPT-4 passes mock interviews for coding jobs at Amazon in the top decile of human performance (taking only several minutes as opposed to the several hours that are typically allotted), performs at human-level accuracy in HumanEval (a database of coding challenges that measure programming logic and proficiency), and is able to write entire programs for simple video games from several lines of instructions (Bubeck et al. 2023). Thus, it would not be unreasonable to expect large potential gains from having developers collaborate with a model of this type.

We study these gains by analyzing results from two experiments conducted at Microsoft and Accenture. The Microsoft experiment started in September 2022 and included 1,663 developers. The Accenture experiment started in July 2022 and included 311 developers. Our preliminary results provide suggestive evidence that developers at both companies became more productive after adopting Copilot, completing 12.92% to 21.83% more pull requests per week at Microsoft and 7.51% to 8.69% at Accenture (depending on specification).

However, both experiments were poorly powered. At Microsoft, there was low initial uptake of Copilot, and shortly after a larger fraction of developers in the treatment group started using it, the control group was also allowed access. At Accenture, internal organizational changes required us to analyze only a (random) subset of the available experimental data.

Our study also has to grapple with the difficulty of measuring the individual productivity of modern service workers. For instance, the output of an 'executive assistant' or a 'management consultant' is poorly defined and (therefore) hard to measure. While there are still challenges in our setting, we benefit from the fact that almost all professional software development follows a highly structured workflow, where specific tasks are defined ex-ante, and the progress of these tasks is tracked through version control software. This allows us to measure the effect of Copilot assistance on the quantity of output, review time, and several measures of the quality of developers' work. In addition, we observe how much developers use Copilot in the form of accepted suggestions.

## 1.1. Related Literature

Most studies of the impact of generative AI tools on worker productivity have been conducted in controlled laboratory-like experiments (Peng et al. 2023; Vaithilingam et al. 2022; Mozannar et al. 2022; Campero et al., 2022; Noy and Zhang 2023). While laboratory 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. For instance, they must be integrated into existing development workflows; developers must adopt and use these tools, which might require additional training; and the effects could be heterogeneous depending on both the developer's experience and the nature of her task.

Evidence from existing (laboratory) experiments generally suggests significant productivity effects of generative AI tools for software developers. Peng et al. (2023) hired 95 professional programmers to develop an HTTP server in JavaScript. The group that used GitHub Copilot completed the task 55.8% faster, yet there was no effect on whether the task was completed. Additionally, Copilot benefited less experienced, high-workload, and older developers. Campero et al. (2022) conducted experiments to assess the impact of GPT-3 on coding performance among both experienced programmers and those with no programming experience. Their findings show that GPT-3 increased performance for both groups. Programmers achieved a 27% increase in speed, whereas nonprogrammers reached performance levels comparable to experienced programmers with the assistance of GPT-3. Vaithilingam et al. (2022) conducted a within-subject study in a randomized controlled

experiment involving GitHub Copilot, in which each participant completed two tasks, one with Copilot access and one without. Contrary to other studies, this study found no statistically significant differences in completion time, although participants preferred using Copilot in their programming work.

Regarding evidence from field experiments, a few studies have found positive productivity effects of generative AI. For customer support via chat interfaces, Brynjolfsson, Li, and Raymond (2023) found that access to an AI-based conversational assistant increases the productivity of customer support agents (measured as issues resolved per hour) by 14% on average. In a laboratory experiment on consultants employed by Boston Consulting Group, Dell'Acqua et al. (2023) found that productivity on 18 tasks designed to mimic the day-to-day work at a consulting company increased by 12% to 25%.

## 1.2. What Is AI-Assisted Software Development?

Some of the earliest commercial applications of large language models are tools to assist software developers in coding. Prominent examples include GitHub Copilot, Amazon CodeWhisperer, and Replit Ghostwriter. In our study, we examine the effects of one of these tools, GitHub Copilot. GitHub Copilot is a code-generation tool developed by GitHub in partnership with OpenAI. It aims to increase the productivity of software developers by offering intelligent code suggestions and autocompletions within integrated development environments. As developers write software code or plain text comments, Copilot analyzes the code context and produces pertinent code snippets, comments, and documentation.

| | Mean | Standard Deviation | 10-th Percentile | 90-th Percentile |
|---|---|---|---|---|
| **Panel A: Microsoft Experiment (N=1,663)** | | | | |
| Pull Requests | 1.19 | 4.21 | 0.00 | 6.00 |
| Successful Builds | 10.55 | 382.69 | 0.00 | 22.00 |
| Lines of Code Changed | 48.50 | 3181.35 | 0.00 | 55.00 |
| Treatment | 0.50 | 0.50 | 0.00 | 1.00 |
| Adoption by Last Period | 0.57 | 0.49 | 0.00 | 1.00 |
| **Panel B: Accenture Experiment (N=311)** | | | | |
| Pull Requests | 0.33 | 1.73 | 0.00 | 0.00 |
| Successful Builds | 1.32 | 5.75 | 0.00 | 3.00 |
| Commits | 5.57 | 18.85 | 0.00 | 16.00 |
| Treatment | 0.61 | 0.49 | 0.00 | 1.00 |
| Adoption by Last Period | 0.35 | 0.48 | 0.00 | 1.00 |

Table 1. Summary statistics. The table presents summary statistics for our outcome variables as well as (varying only at the developer level and not across time) a dummy for whether the developer was assigned to be allowed access to Copilot (“Treatment”) and whether by the final week in our data the developer had used Copilot at least once (“Adoption By Last Period”). Note that our sample includes post-experimental periods for Panel A, and hence developers who are not treated may have adopted Copilot by the last period.

The development of Copilot involved combining advanced machine learning techniques and natural language processing. To train Copilot, a substantial amount of code from public GitHub repositories was used. This extensive dataset allowed the AI model to learn from real-world coding practices, patterns, and styles across various programming languages and frameworks.

## 2. Experimental Design and Outcome Variables

We analyze two randomized experiments conducted with software developers at Microsoft and Accenture. In each experiment, 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. While developers in the treated group had access to Copilot, they were not *required* to use it; thus, our identification strategy relies on an encouragement design and hence employs instrument variables regressions rather than simple comparisons of means across treatment and control. We collected various performance metrics following the experiments to uncover systematic performance differences between the treated and control groups, aiming to gain insights into how GitHub Copilot impacts software developer productivity. Below, we present the specifics of these experiments.  Summary statistics for both experiments are provided in Table 1.

## 2.1. Microsoft Experiment

The experiment at Microsoft started in the first week of September 2022, involving a sample size of 1,749 developers, of which 50.3% were randomly selected to be given access to Github Copilot. Randomization was implemented at both the individual and the team level. 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 3 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.

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. While this large heterogeneity makes it difficult to obtain standardized task-specific performance measures, it reflects their real-world environment.

We gathered data on several output performance metrics, including the total number of pull requests, the number of successful builds, and the number of lines of code changed, which we explain in detail below. These metrics provide accurate insights into the developers' coding activity. Additionally, we gathered data on the total numbers of shown and of accepted suggestions by Copilot to measure actual Copilot usage.

As shown in Figure 1, during the first two weeks of the experiment, only 8.6% of the treated group signed up for GitHub Copilot. This low adoption rate may have been due to inattentiveness to the initial email notification. Microsoft therefore sent two additional email reminders on February 15, 2023, and February 28, 2023. These additional emails increased the take-up rate to 35% within two weeks of being sent. The initial compliance in the control group was not perfect, with 1.2% of individuals in the control group adopting Copilot. This was primarily because a subset of control group developers was engaged in developing Copilot-related products.

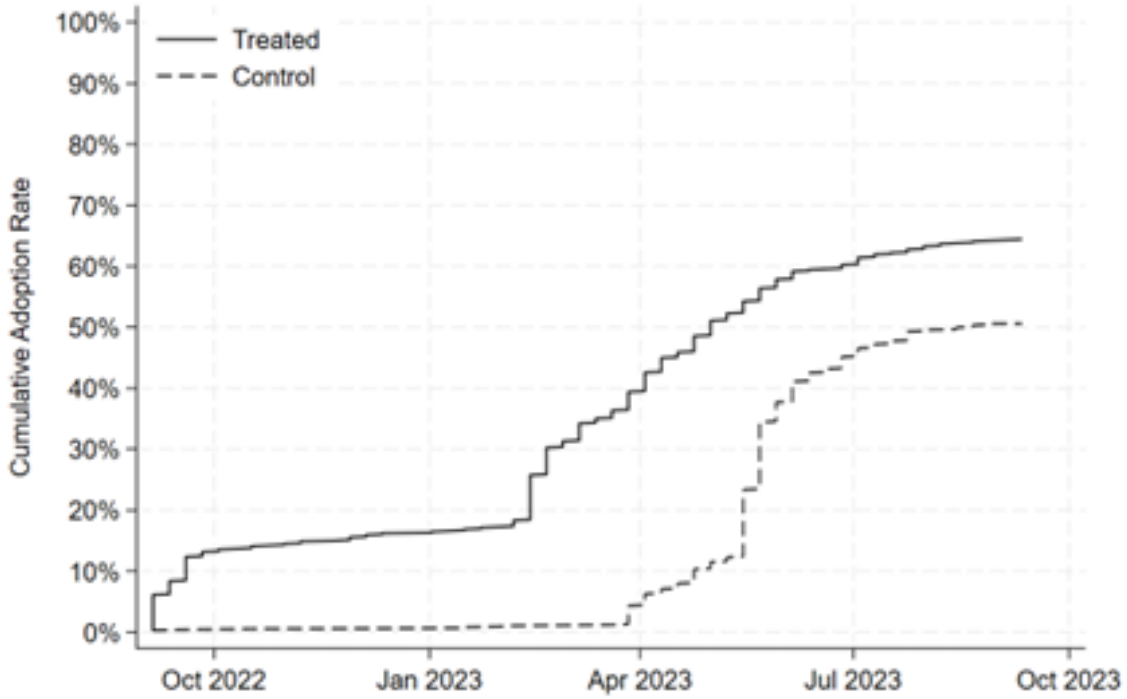


Figure 1. Cumulative adoption of copilot for Microsoft experiment. This graph shows the cumulative rate of adoption across time for software developers at Microsoft in both the treatment and control group over the course of the experiment.

We have access to weekly data over the seven months of the experiment, which lasted from September 2022 to March 2023. Table 2 presents a balance table for the three outcome variables we analyze in the study for the pretreatment period. We observe that all outcomes are balanced between the control and treated groups, suggesting that randomization was successful (we discuss balance for the Accenture experiment separately below). At the end of the experiment in March 2023, the control group was given access to GitHub Copilot. Nevertheless, we continued to receive data after the formal conclusion of the experiment, observing a rapid adoption rate in the control group. By September 2023, adoption in the control group had reached a similar level (50%) as in the treated group (65%).

| | Control Mean | Treatment Mean | Difference | Difference p-value |
|---|---|---|---|---|
| **Panel A: Microsoft Experiment** | | | | |
| Pull Requests | 0.97 | 0.99 | 0.02 | 0.77 |
| Successful Builds | 6.05 | 6.06 | 0.01 | 0.98 |
| Lines of Code Changed | 14.54 | 13.94 | -0.59 | 0.53 |
| **Panel B: Accenture Experiment** | | | | |
| Pull Requests | 0.07 | 0.08 | 0.00 | 0.67 |
| Successful Builds | 0.49 | 0.57 | 0.08 | 0.38 |
| Commits | 2.56 | 3.64 | 1.08 | 0.01 |

Table 2. Treatment/control balance table for pre-experiment outcomes. The table presents the mean values of our primary outcome variables during the pre-experiment phase, separately for both treatment and control groups. Additionally, it includes the *P*-value from the *t*-test assessing the mean difference between treatment and control. To the extent that the randomization was successful, we would expect to not reject the null of a zero difference between treatment and control in any row.

## 2.2. Accenture Experiment

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 60% of 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, Microsoft worked with the participants' managers to encourage them to incorporate GitHub Copilot in their daily workflow.

From these participants, we have access to various metrics that capture the workflow of software development and allow us to measure their productivity both in quantitative and qualitative terms. We also have access to several interviews of software developers, gaining insights into their views about GitHub Copilot and how they affect their productivity.

Data from the Accenture experiment is available from July 2022 to November 2023. We drop developers if their data is always missing (54 participants who do not actively participate in software development) and those who left the company (4 participants), leaving us with a sample of 311 developers. As reported in Figure 2, treated developers adopted Copilot rapidly, with an adoption rate of 57% two months after the experiment. Moreover, in this experiment, there was perfect compliance for the control group, i.e., no developers in the

control group were granted access to GitHub Copilot. This experiment is still running, and we hope to receive more data.

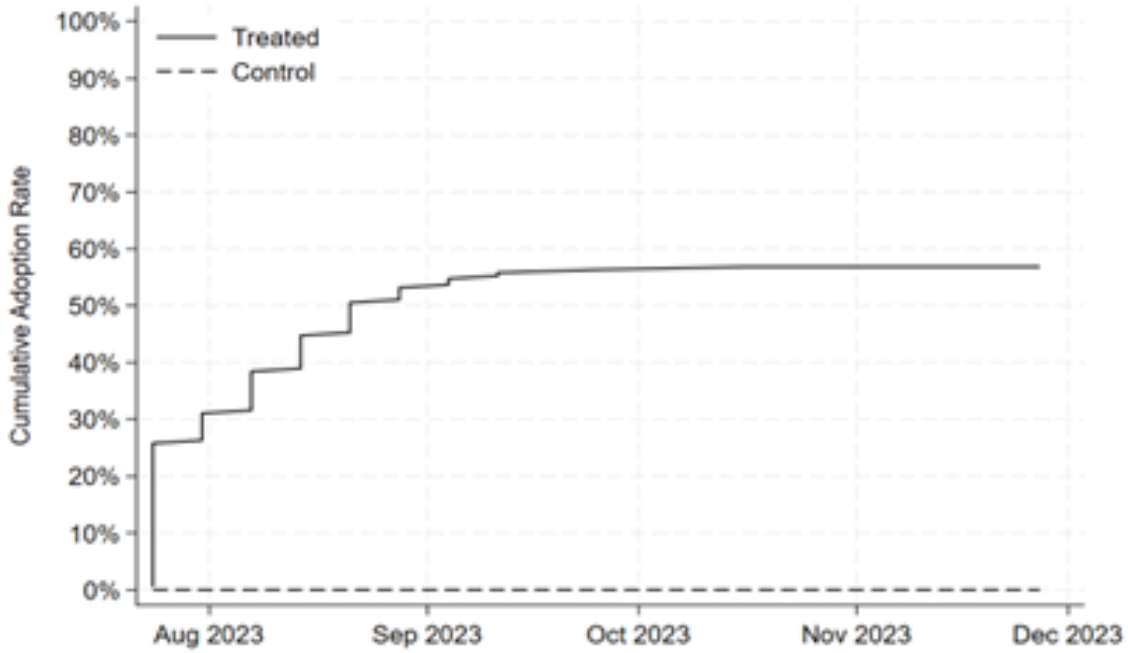


Figure 2. Cumulative adoption of Copilot for Accenture experiment. This graph shows the cumulative adoption rate across time for software developers at Accenture in both the treatment and control over the course of the experiment.

As for the Microsoft experiment, Table 2 presents a balance table for pretreatment outcomes for the three variables we analyze in the Accenture experiment. By contrast to the Microsoft experiment, we reject the null of no pretreatment difference in the number of commits; the other outcomes are balanced between the control and treated groups. As this is a single rejection across six outcomes in the balance table, we cautiously interpret this failure of balance not as a failure to randomize (indeed, randomization occurred without access to these variables) but rather as an unlucky draw. Nevertheless, the imbalance needs to be considered when interpreting our results for the number of commits outcome below, which should be taken with a grain of salt.

## 2.3. Variables and Outcome Measures

Modern software development is organized through version control software. This has the advantage that internally defined goals and tasks are tracked and made visible to outside observers. Both participating organizations use GitHub, a version control software offered by Microsoft.

Pull requests keep track of a unit of work from a software developer. Different organizations may differ in how they define the scope of a pull request, but this scope is likely to be relatively stable over a longer time horizon as it is shaped by organizational norms and conventions. For instance, a pull request may be asking for a

feature to be added to a larger software project. Before a 'pull' is requested, a developer will work separately on her code, tracking smaller changes through 'commits.' The pull request will lead to a review of this code, 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.

Through the developers' GitHub activity, we can observe all the variables that are part of this workflow, i.e., the number of tracked code changes of individual developers (commits), the completion of predefined tasks (pull requests), the time it took to review those completed tasks, and three objective measures of the quality of these completed tasks (the number and rate of successful compilations and the number of errors encountered when compiling). Due to the fat right tail that these variables take on, we winsorize them at the 5th and 95th percentile.

In addition to these quantitative and qualitative productivity measures, we are able to 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. Copilot usage data is particularly helpful due to potentially significant heterogeneity in Copilot usage after the initial sign-up. For example, some software developers may try Copilot for a few weeks and then stop using it, whereas others gradually incorporate it into their workflow.

Lastly, for the experiment participants at Microsoft, we observe much more detailed data on the allocation of the work time of software developers. This data is referred to as telemetry data and records general activity on the work computer of software developers. Through this telemetry data, we can observe the time that developers designate toward code development as opposed to emails, meetings, or other work-related activity. This data also allows us to keep track of total work time. While we hope to use this data in future iterations of this project, we do not explore the effects of Copilot on time usage in this draft due to their high variance.

| Outcome | IV1 | IV2 | SLATE |
|---|---|---|---|
| Pull Requests | 20.49 | 12.92 | 21.83** |
| | (22.14) | (16.60) | (11.12) |
| Successful Builds | 5.55 | 17.31 | 20.77* |
| | (22.53) | (16.77) | (11.32) |
| Lines of Code Changed | 0.95 | 9.52 | 11.53* |
| | (11.41) | (9.19) | (6.06) |
| Pre-Period Included | | ✓ | ✓ |
| Developer FE | | ✓ | ✓ |
| Time FE | ✓ | ✓ | ✓ |

Table 3. Impact of Copilot on productivity in the Microsoft experiment. This table gives the estimated impact of the adoption of GitHub Copilot on various productivity measures (one per row.) All estimates are expressed as percentages of the pre-treatment mean, and parentheses refer to standard errors clustered at the developer level. The first column ("IV1") gives results from a linear regression of the outcome on a dummy for Copilot usage and time fixed effects; the usage dummy has been instrumented by a dummy indicating assignment to the treatment group. The second column ("IV2") supplements pre-experimental data, adds developer fixed effects, and instruments with a dummy that turns on for developers in the treatment group only after the start of the experiment. Finally, the third column ("SLATE") is like "IV2" except that we interact the previous instrument with predicted 'compliance,' i.e., predicted differences in uptake between control and treatment groups. Effectively, the SLATE specification weights periods with large differences in uptake more heavily than other periods.*10%, **5%, ***1%.

# 3. Results

## 3.1. Microsoft Experiment

In Table 3, we show results from three different specifications that each aim to analyze the impact of Copilot access on Microsoft developers. Specifications IV1 and IV2 use the experimental treatment assignment as an instrument for Copilot adoption. The difference between IV1 and IV2 is that IV2 includes pre-experimental data and developer fixed effects to control for developer heterogeneity. The first-stage regressions of these instrumental variable specifications are poorly powered because of the low initial take-up rate in the treatment group.

Finally, the third column ("SLATE") utilizes the experimental variation but further uses compliance weighting as suggested by Coussens and Spiess (2021) to estimate a better-identified Super Local Average Treatment Effect (i.e., the effect of Copilot in the periods in which experimental assignment best predicts Copilot uptake.) Formally, we regress Copilot uptake on a set of time dummies and linear trends separately for the treatment and control groups;[1] the difference in predicted uptake across these groups (where predictions are based

exclusively on the time since the experiment's start) is then multiplied with the treatment dummy to construct our new SLATE IV. We can think of the resulting regression as a weighted IV regression that weights periods based on the difference in Copilot adoption between control and treatment groups. This regression will thus more heavily weight, for example, March 2023, when there was a large difference in adoption between treatment and control groups. In the SLATE column, we find evidence that the total number of pull requests may increase by as much as 22%, with similar-sized changes in the total number of builds. We also find an 11% increase in the number of lines of code changed, suggesting that developers are not simply dividing up the same amount of work into more pull requests.

| Outcome | IV1 | IV2 | SLATE |
|---|---|---|---|
| Pull Requests | 7.75 | 7.51 | 8.69** |
| | (7.05) | (5.46) | (4.37) |
| Successful Builds | 107.02*** | 89.57*** | 84.17*** |
| | (28.29) | (24.70) | (23.79) |
| Commits | 51.58** | 1.04 | 5.50 |
| | (24.73) | (23.31) | (23.19) |
| Pre-Period Included | | ✓ | ✓ |
| Developer FE | | ✓ | ✓ |
| Time FE | ✓ | ✓ | ✓ |

Table 4. Impact of Copilot on productivity in the Accenture experiment. This table gives the estimated impact of the adoption of GitHub Copilot on various productivity measures (one per row.) All estimates are expressed as percentages of the pre-treatment mean, and parentheses refer to standard errors clustered at the developer level. The first column ("IV1") gives results from a linear regression of the outcome on a dummy for Copilot usage and time fixed effects; the usage dummy has been instrumented by a dummy indicating assignment to the treatment group. The second column ("IV2") supplements pre-experimental data, adds developer fixed effects, and instruments with a dummy that turns on for developers in the treatment group only after the start of the experiment. Finally, the third sixth column ("SLATE") is like "IV2" except that we interact the previous instrument with predicted compliance. *10%, **5%, ***1%.

## 3.2. Accenture Experiment

Table 4 summarizes our estimates of the effect of GitHub Copilot on the productivity of software developers at Accenture. The first and second columns ("IV1" and "IV2") provide estimates that utilize the experimental variation as an instrument for the adoption of Copilot; they differ only in that IV2 uses pre-experimental data and developer fixed effects in an attempt to denoise the data.

Regarding measures of pull requests—our preferred measure of productivity for the Microsoft experiment—we see mixed results at Accenture. IV1 and IV2 estimates suggest an increase in the number of pull requests by 7.75% and 7.51%, but these effects are not statistically significant. As for the Microsoft experiment, the SLATE specification[2] increases precision by focusing on periods in which the instrument is maximally predictive of Copilot adoption; hence, the SLATE estimate of 8.69% is statistically significant at the 5% level.

We note that unlike at Microsoft, pull requests at Accenture are usually conducted by a manager, and hence they may be a less reliable indicator of individual developer performance at Accenture than at Microsoft. To ameliorate this concern, we also look at the effect of Copilot on the number of commits and builds, which are more likely to be directly controlled by individual developers.

We find a large positive effect of 51% on commits in IV1, but this result may be driven by a slight imbalance of randomization across the pre-experiment number of commits (see Table 2) Indeed, once we include the pre-experimental data (IV2 and SLATE), the effect vanishes. While we report the result on commits, its interpretation is unclear: It is possible that adopting Copilot makes developers feel more certain about their changes in the code base, hence necessitating fewer commits that could be reverted to in case of problems down the road.

For this reason, our primary effects of interest here are the effects on the total number of successful builds. These effects are positive, statistically significant, and economically meaningful across all specifications. The estimates vary from an 84% increase in builds to a 107% increase in builds. While these effects seem large, they are still quite imprecisely estimated, with standard errors as large as 28% of the pretreatment mean.

## 4. Caveats

We would like to discuss several caveats of our analysis. The first caveat concerns the interpretation of our results. While certain output of software developers is very well measured, we may still be concerned that none of these measures is a silver bullet to capture true developer productivity. In particular, the number of commits may not be monotone in underlying productivity. Furthermore, we are unable to investigate how the above outcomes change *per unit of time* that developers devote to these tasks. It is a possibility that some developers reduced or increased their work hours after adopting Copilot.

Secondly, imperfect compliance limits the precision of our estimates. In particular, being assigned to the treatment group only gave experimental subjects the *option (not the obligation)* to adopt Copilot. However, at least initially, the adoption of Copilot in the Microsoft experiment was slow. Worse, once adoption picks up, the experiment soon concludes, and the control group quickly adopts Copilot. Thus, while these adoption patterns do not present a threat to identification or a potential bias, the experimental assignment lacks power, and to the extent that they limit the precision of our estimates, this lack of precision is reflected in their associated standard errors.

# 5. Conclusion

To summarize, we find suggestive evidence that GitHub Copilot raises the productivity of software engineers. However, these estimates are still preliminary. We are still in the process of collecting data for the Accenture experiment.

# Appendix

**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 A1. Screenshot of the e-mail sent to participants in the Microsoft experiment.

## Footnotes

1. To be exact, we regress uptake on (1) three dummies turning on after first adoption, the nudge emails, and after the control group was allowed access to Copilot as well as (2) two linear time trends after first adoption and after the control group was allowed to access Copilot. Results do not differ substantially when using a full set of time fixed effects. ↩

2. For the Accenture experiment, SLATE uses as weights the difference in predicted uptake across treatment and control group where the prediction comes from a (bivariate) regression of uptake on a linear time trend that measures the number of weeks since the first developer adopted. ↩

## References


- Brynjolfsson, Erik, and Tom Mitchell. 2017. “What Can Machine Learning Do? Workforce Implications.” *Science* 358, no. 6370: 1530–34.

  ↩
- Brynjolfsson, Erik, Li, Danielle, and Lindsey R. Raymond. 2023. “Generative AI at Work” (working paper 31161, National Bureau of Economic Research).

  ↩
- Bubeck, Sébastien, Chandrasekaran, Varun, Eldan, Ronen, Gehrke, Johannes, Horvitz, Eric, Kamar, Ece, Lee, Peter, et al. 2023. “Sparks of Artificial General Intelligence: Early Experiments with GPT-4.” Preprint, submitted March 22, 2023. https://arxiv.org/abs/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. https://doi.org/10.48550/ARXIV.2206.12390. ↩
- Coussens, Stephen, and Jann Spiess. 2021. “Improving Inference from Simple Instruments through Compliance Estimation.” Preprint, submitted August 8, 2021. https://arxiv.org/abs/2108.03726.

  ↩
- Dell’Acqua, Fabrizio, McFowland, Edward, Mollick, Ethan R., Lifshitz-Assaf, Hila, Kellogg, Katherine, Rajendran, Saran, Krayer, Lisa, Candelon, François, and Karim R. Lakhani. 2023. “Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality” (working paper, Harvard Business School, Technology & Operations Management).

  ↩
-

↩

- Mozannar, Hussein, Bansal, Gagan, Fourney, Adam, and Eric Horvitz. 2022. “Reading Between the Lines: Modeling User Behavior and Costs in AI-Assisted Programming.” Preprint, submitted October 25, 2022. https://doi.org/10.48550/arXiv.2210.14306.

↩

- Noy, S. and Zhang, W., 2023. Experimental evidence on the productivity effects of generative artificial intelligence. Available at SSRN 4375283.

↩

- Peng, Sida, Kalliamvakou, Eirini, Cihon, Peter, and Mert Demirer. 2023. “The Impact of AI on Developer Productivity: Evidence from GitHub Copilot.” Preprint, submitted February 13, 2023. https://doi.org/10.48550/arXiv.2302.06590.

↩

- Vaithilingam, Priyan, Zhang, Tianyi, 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, 1–7. New York: ACM. https://doi.org/10.1145/3491101.3519665.

↩

- Webb, Michael. 2019. “The Impact of Artificial Intelligence on the Labor Market.” Preprint, submitted November 6, 2019. http://dx.doi.org/10.2139/ssrn.3482150.

↩