

# Generative AI and Distributed Work: Evidence from Open Source Software

Manuel Hoffmann\* Sam Boysel\* Frank Nagle\* Sida Peng† Kevin Xu‡

\*Harvard Business School, Harvard University

†Microsoft Corporation

‡GitHub Inc.

This version: *July 8, 2024*

**Abstract:** Work is increasingly performed by distributed teams in which some linchpin contributors shoulder a disproportionate burden. As these arrangements bring inherent risk in the production process, the critical question of how to relieve this burden arises. Recent advances in artificial intelligence (AI) technology demonstrate considerable potential to complement human capital intensive activities. While an emerging literature documents wide-ranging productivity effects of AI, relatively little attention has been paid to how AI might change the nature of distributed work, especially in the context of linchpins. Using the setting of open source software (OSS), we study individual level effects that AI has on task allocation. We exploit a natural experiment arising from the deployment of GitHub Copilot, a generative AI code completion tool geared towards software developers. Leveraging millions of work activities over a two year period, we use a program eligibility threshold to investigate the impact of AI technology on maintainer (OSS linchpin contributors) task allocation within a quasi-experimental regression discontinuity design. We find that having access to Copilot induces “linchpin” maintainers to shift task allocation towards coding activities and away from project management. Further, linchpins with relatively low ability are more likely to benefit from the coding AI. Overall, our estimates point towards a large potential for AI to transform work processes and to ameliorate the linchpin problem in the digital economy.

**JEL-Classification:** H4, O3, J0

**Keywords:** Generative Artificial Intelligence, Digital Work, Open Source Software, Digital Economy

**Acknowledgement:** The authors are grateful for financial and administrative support from GitHub and, in particular, for generous advice from Peter Cihon. The authors are also indebted for comments by seminar participants at the research seminars at Harvard LISH in Cambridge, MA. We are further grateful for feedback from participants at the “Labor in the Age of Generative AI” conference at the University of Chicago, the 22nd ZEW Economics of ICT conference, in Mannheim, Germany as well as the ACM Collective Intelligence Conference in Boston.

# 1 Introduction

In the modern era, work of all types is increasingly done by teams of individuals collaborating towards a common goal - whether it be within the boundaries of a firm, across loosely organized collections of volunteers, or even by students in a classroom (Charness and Holder, 2019; Emanuel, Harrington, and Pallais, 2023; Fischer, Rilke, and Yurtoglu, 2023). Despite the frequent focus on the output of the team as a whole, team production often hinges on so-called “linchpins”: highly skilled workers that are not easily substituted for and therefore become indispensable components of the production process (Ballester, Calvó-Armengol, and Zenou, 2006; Godin, 2010). However, such individuals can easily become overburdened or burnt out from their heavy workload leading to a bottleneck that prevents the team from functioning optimally.<sup>1</sup> This linchpin problem can be particularly salient in the context of distributed work (Hinds and Kiesler, 2002), where team members are not co-located (increasingly the case in the post-COVID world) and communication and coordination costs are high. It can be further exacerbated when the output of a team is a public good as these are often produced largely through the disproportionate efforts of a relatively small number of highly skilled individuals acting as part of a larger team (Fisher, Smith, and Welser, 2006; Panciera, Halfaker, and Terveen, 2009). With recent advances in technology, a potentially novel solution to the linchpin problem may be to provide these critical team members with artificial intelligence (AI) tools to reduce their burden and alleviate the bottleneck.

Throughout human history, there have been a handful of technological innovations that fundamentally shift how the economy works. The printing press, internal combustion engine, and computers are oft-cited examples of such general purpose technologies. Although AI has existed for some time, many have argued that recent advances may push it into this elite category of technologies that alter the course of history (Crafts, 2021; Goldfarb, Taska, and Teodoridis, 2023; Eloundou et al., 2024). However, if AI — broadly defined as the use of computers and machines to mimic human intelligence — is destined to have such a substantial impact, we are likely still at

---

<sup>1</sup>In the extreme, this can also lead to the challenge known as “key person risk” where one individual leaving a team would mean the team would cease to be able to function. In our analysis, we focus on the overburdening of these individuals rather than the possibility of them leaving the team.

the beginning of this technological revolution that is slowly and steadily reaching all sectors of the economy (Acemoglu et al., 2022). Importantly, the highest economic impact of AI is predicted to be on productivity growth through the labor market (Bughin and Manyika, 2018; Sachs, 2023). However, due to the novelty and breadth of AI, research is only starting to elucidate its impact on the nature of work and task allocation in distributed production settings. This is particularly true of generative AI (GenAI) — a subset of AI built on large-language machine-learning models (LLMs) — which exploded on to the scene in 2022 and currently represents the cutting-edge of AI. These models, including OpenAI’s GPT4, Google’s Gemini,<sup>2</sup> Meta’s LLaMa, and numerous others, are trained on massive, Internet-scale databases and use billions of parameters to construct a probabilistic model that predicts what the next word in an answer to a prompt from a user should be. These models can also be trained on datasets that are more focused on specific contexts — e.g., health, finance, customer service, software development, etc. Whether and how these new technologies will shape the nature of work remain open questions. Further, whether AI can be a complement to skilled workers (Autor, 2024) and help address critical aspects of team production like the linchpin problem, especially in the context of distributed work has gone under-explored.

Answering such questions is of the utmost importance to better understand how distributed digital work processes are affected when AI is introduced and how it can assist in workforce optimization as a goal for strategic human capital management (Wright, Coff, and Moliterno, 2014). Although some early studies on generative AI have shown positive high-level productivity impacts (Brynjolfsson, Li, and Raymond, 2023; Dohmke, Iansiti, and Richards, 2023; Noy and Zhang, 2023; Peng et al., 2023), it is less clear what the mechanisms behind these improvements are. Does the use of generative AI shift users to focus on particular types of tasks that lead to those productivity improvements? If so, which tasks? How exactly does the work process change when using generative AI? Does AI present a potential solution to the linchpin problem? Understanding these impacts informs labor strategy relevant to both firms (Tamayo et al., 2023) and policymakers (U.S. Department of Labor, 2024), including hiring policies, work training programs, and upskilling or

---

<sup>2</sup>Formerly known as Bard.

reskilling efforts for current employees.

The key challenge in assessing how AI changes the nature of distributed work is to identify a setting where (1) work patterns are observable and (2) an AI tool consequential to linchpin workers has been introduced in a pseudo-exogenous manner. Our setting - the introduction of GitHub Copilot - a software development focused GenAI tool - for key developers (known as maintainers) in open source software (OSS) projects addresses both of these criteria. OSS - software whose source code is publicly available for use and modification and that is frequently developed by distributed teams of developers - is a classic example of a product that is produced through the distributed work of teams and is generally free (Moon and Sproull, 2002). Further, OSS also suffers from the linchpin problem as a small set of maintainers are the driving force behind the widely used and incredibly valuable digital infrastructure that has come to underlie software development and the modern economy as a whole (Eghbal, 2020; Geiger, Howard, and Irani, 2021; Hoffmann, Nagle, and Zhou, 2024). The linchpin problem in OSS production can be exacerbated by a number of factors which shift larger workloads onto maintainers. First, considerable demand for OSS scales in part due to its zero price and cost of reproduction. Second, the public nature of OSS gives rise to strong incentives for non-contributors to free ride on the efforts of maintainers who cannot exclude them. Finally, communication technologies and platforms designed to reduce collaboration frictions in distributed work have dramatically lowered barriers to participate in OSS ecosystems. In practice, an influx of non-experts creates an additional burden on maintainers, who must triage support requests, review contributions, and otherwise manage their project's growing community. Indeed survey evidence documents that maintainers tend to be overburdened with too little of their time spent on their core work (coding) and too much on managerial (project management) tasks (Nagle et al., 2020a). With these factors in mind, interventions with the potential to relax constraints on linchpins are of great interest to the distribution production setting of OSS.

GitHub's Copilot is based on OpenAI's Generative Pre-trained Transformer (GPT) models, which also underlie ChatGPT. Copilot was developed jointly by OpenAI (an AI research organization) and GitHub, a subsidiary of Microsoft that offers cloud-based software development and

version control services and is also the world’s largest host of open source software (OSS) projects. Copilot was trained on vast amounts of publicly available software code, including code hosted as OSS on GitHub itself. Originally released to a subset of users as a technical preview starting in June 2021, it became more broadly available as a full product in June 2022. As part of this roll out process, GitHub offered free access to a subset of users, including students and maintainers of top OSS projects hosted on GitHub. This roll-out process assists us in identifying the causal effects of access to Copilot.

We exploit aspects of the general access launch of Github Copilot to the broader public. In particular, a segment of linchpin developers who maintain the top OSS projects on GitHub receive free Copilot access. Eligibility is determined by an internal ranking known to GitHub, and not the individual developers, allowing us to make credible claims about non-manipulation to establish a causal identification strategy to recover the effect of generative AI on developer activity. In the most basic version, maintainers with a ranking below zero become eligible to receive free access to Copilot and those above do not. We use a panel of 187,489 distinct maintainers observed weekly from July 2022 through July 2024, which results in millions of maintainer-week observations for activity levels in public GitHub repositories and Copilot usage. Within the data set of linchpin top maintainers, we find that those who receive access to Copilot during the general access period increase their relative share of coding tasks while reducing their relative share of project management activities. The dynamics of the treatment effects are stable for our two year period. When decomposing these categorizations into more granular activities, we find that increased coding activities are primarily driven by maintainers increasing their relative volume of code contributions<sup>3</sup> while project management reductions stems from the reduction in problem solving activities with others. The results are stable not only on the individual level but also consistent when using a supplementary identification on the repository level. Additional heterogeneity assists us in understanding the linchpin problem. In particular, we find that linchpins with relatively lower ability are more sensitive to the treatment. Lower ability maintainers who receive access to AI increase

---

<sup>3</sup>On the GitHub platform, this is measured when a contributor uploads or “pushes” a change of code (i.e. commit) to a codebase.

coding and reduce project management to a greater extent compared with their higher ability peers.

Our results contribute to a growing literature on the productivity impacts of AI in important ways. Early work in this area posits general productivity gains ([Agrawal, Gans, and Goldfarb, 2019](#); [Corrado, Haskel, and Jona-Lasinio, 2021](#); [Raj and Seamans, 2018](#)), but that the gains may not be evenly distributed ([Brynjolfsson, Rock, and Syverson, 2018](#); [Furman and Seamans, 2019](#)). Empirical work confirmed such predictions and found wide-ranging productivity benefits to using AI, at both the firm level ([Czarnitzki, Fernández, and Rammer, 2023](#)) and the individual level ([Fügener et al., 2022](#)). Our work is consistent with this prior research but adds additional nuance to the labor augmenting technical change literature ([Acemoglu, 2003](#)) by going beyond productivity when focusing on the nature of work as we provide one of the largest natural experiments of generative AI and its impact on highly disaggregated measures of distributed work processes for a two year time horizon “in the wild”.

Our main findings identify changes in the nature of work of linchpin AI adopters in their digital work processes. We show that when software developers leverage AI more, they reallocate their efforts towards technical coding activities and away from auxiliary project management activities that involve social interactions with other maintainers. This is a sign that the maintainers intensify contributions to the public goods of open source software through AI. It is also consistent with reducing collaborative frictions during the problem solving process of work and a change in the way voluntary workers interact with each other on the platform. We complement the current literature that leverages IT and consultancy chat support AIs and focuses on high-level productivity impacts through experimentation ([Brynjolfsson, Li, and Raymond, 2023](#); [Dell’Acqua et al., 2023](#)) by investigating the nature of work through changes in work activities and human interaction processes over two years after the introduction of the programming LLM.

Beyond the identification of causal effects that generative AI has on decentralized work, our results suggest important implications for the future of OSS. OSS has received growing attention ([Lerner and Tirole, 2002](#)) as it has become an increasingly critical part of the modern economy, such that 96% of corporate codebases include some OSS ([Synopsis, 2023](#)). Further, recent studies

estimate the value of OSS to be on the order of billions of dollars for the supply side (Blind et al., 2021; Robbins et al., 2021) and trillions of dollars when accounting for usage (Hoffmann, Nagle, and Zhou, 2024). Additionally, firm usage of, and contribution to, OSS has important implications for firm productivity (Nagle, 2018, 2019), firm competition (Boysel, Hoffmann, and Nagle, 2024) and entrepreneurial activity (Wright, Nagle, and Greenstein, 2023). However, despite the importance of OSS, many critical projects are under-resourced (Nagle et al., 2020b) as numerous firms free-ride on the efforts of others without giving back (Lifshitz-Assaf and Nagle, 2021) leaving volunteer maintainers burnt out and overwhelmed (Raman et al., 2020). As our results show, generative AI may offer a solution to help address these concerns and allow top maintainers to more easily contribute to the common good by solving issues faster. Prior research has shown that OSS developers and maintainers generally contribute to OSS because it gives them a creative outlet and they do not want to spend their time on managerial tasks like security and documentation (Nagle et al., 2020a). AI-powered tools may make it easier to quickly address such managerial tasks, so developers can spend time in a manner they prefer, while still ensuring the security, stability, and usability of OSS.

The remainder of this paper proceeds as follows. In Section 2 we discuss the environment within which the study occurs. In Section 3 we characterize our dataset and discuss the construction of our sample. We hone into the set of developers that are obtaining Copilot eligibility for free via an internal ranking from GitHub and present our estimation strategy in Section 4. We then present our results using a regression discontinuity design (Section 5) while exploring mechanisms. We discuss the limitations, implications, and a back-of-the-envelope calculation to understand how the results may apply to private software development activity in Section 6. Section 7 concludes.

## 2 Institutional Background

### 2.1 The GitHub Platform

GitHub was established in 2008 and is the world’s largest hub for OSS development.<sup>4</sup> It is a “social coding” platform that offers cloud-based software development and version control services. Importantly, it is specifically designed for dispersed teams to collaborate on software development projects, and it chronicles all activities performed on the system to ensure any contributor can observe all prior activity. Activity on the GitHub platform can therefore provide the researcher unique and granular insights into patterns of distributed work, which are increasingly becoming the norm in all areas of knowledge work. Furthermore, the platform allows us to observe the decentralized production of OSS as a public good. Although the details can be quite intricate, the primary workflow of a GitHub contributor is straightforward.

A user who wants to start a new project creates a repository and then writes their code within this repository.<sup>5</sup> Alternatively, a user may “fork” another repository, which entails copying everything into another repository to have the same exact information, but allows them to take the project in a different direction than the primary repository. When the user modifies project code in a local copy of the repository on their machine, these changes to the codebase are condensed into a “commit” that attributes authorship to a user. Uploading these commits from the local repository to the remote GitHub repository is called a “push”. GitHub also has popularized the “pull request” paradigm for OSS contribution in which users contribute to projects by requesting the project’s maintainer to integrate their proposed changes. For example, if user A maintains a repository and user B wants to add a new feature to it but does not have permission to edit the code directly, user B can issue a “pull request” which includes the suggested changes as a sequence of commits to a

---

<sup>4</sup>According to the Engagement platform [6Sense](#), GitHub had a market share of 78.81% on March 18, 2024. Further, 93% of individuals are using Git as a version control system which underlies GitHub. Other alternatives: GitLab, Bitbucket, Codeberg, Gitee, SourceForge, SrcHut, to name a few.

<sup>5</sup>A project “repository” is a focal point for collaboration over a particular codebase. While a repository technically refers to the collection of source code files for the project, the GitHub platform adds important social and project management features to a repository’s profile, including a detailed version control viewer, a forum to raise issues and discussions, and a formal contribution system based on “pull requests”. Throughout this paper, we will refer to a “repository”, “project”, or “codebase” somewhat interchangeably.



fork of the original project. If user A accepts the proposed change, the changes proposed by user B are integrated or merged into the codebase. Finally, any user can report an “issue” for a particular repository (e.g., identifying a bug or asking for a new feature) and when the issue is addressed, it is considered “closed.”

## 2.2 GitHub Copilot

The empirical focus of this paper centers on the introduction of the generative AI coding tool GitHub Copilot. Copilot was built collaboratively between OpenAI and Microsoft/GitHub and was based on models similar to those that underlie ChatGPT.<sup>6</sup> The version of the Copilot AI under consideration in this study is based on the Generative Pre-trained Transformer 3 series (GPT-3) from OpenAI. Copilot can be used by programmers to obtain code-snippets for work while they are coding, and it allows them to easily integrate those snippets into their coding projects. As a large language model, Copilot operates on the idea of next word prediction and instead of a text completion tool it is a code completion tool with the goal of assisting programmers to code faster, solve problems more quickly, and learn code that they previously did not know. Figure A1 provides an example of how the generative AI Copilot can be used to complete a full function after a user generated only the function header.

— Figure 1 about here —

Figure 1 shows the timeline of the introduction of the AI for programmers. GitHub’s Copilot was first launched on June 29, 2021, as a “technical preview” (TP), and then fully launched for “general access” (GA) on June 21, 2022.<sup>7</sup> GitHub users can access Copilot through several pathways. During the technical preview period, every individual was eligible to access Copilot for free. During the general access period, individuals could access Copilot by obtaining a free trial for 60 days and later for 30 days. After the trial period, they must pay \$10 per month (or \$100 dollars per

---

<sup>6</sup>GitHub Copilot is not the same as Microsoft Copilot which was launched after GitHub Copilot. In June 2023, 92% of programmers had used some coding AI tool according to a [GitHub programmer survey](#).

<sup>7</sup>Start date of general access announcement: [GitHub blog post](#).

year) for continued access. For some individuals, GitHub provides free access after the technical preview period. Students can obtain Copilot and pay \$0 per month. Similarly, GitHub rewards top OSS maintainers with free Copilot access based on an internal eligibility ranking. Finally, individuals can obtain access through their company starting on February 14, 2023. Companies can sponsor their employee access to Copilot with the employer paying \$19 per employee for each month.

In this study, we leverage the internal eligibility ranking from GitHub which determines that a sub-population of top maintainers, i.e. linchpins, receive complimentary Copilot access which we leverage as a natural experiment. GitHub ranks selected repositories based on an internal ranking that is unknown to the community. Any individual who has been added as a “collaborator” to those repositories is considered a maintainer and is therefore potentially eligible for free access to GitHub Copilot.<sup>8</sup> Maintainers for projects below a given threshold of the eligibility ranking are granted complimentary Copilot access for one year. After a year, GitHub will check if you are still eligible through the top maintainer channel and continue granting access if that is the case.

A key feature of this program is that the exact ranking system is largely unknown to maintainers. Public discussions between users suggests there exists a considerable degree of uncertainty over what makes a given project eligible.<sup>9</sup> As GitHub does not reveal the exact composition and ingredients of the ranking, the maintainers can only guess. This vagueness in the composition and ingredients of the eligibility ranking lends additional credibility to our identification strategy as it is virtually impossible to manipulate the ranking as a maintainer. Moreover, GitHub does not engage in any additional messaging to communicate the eligibility status to maintainers. Each maintainer has to check whether they are eligible to access the AI for free when they apply for Copilot.

---

<sup>8</sup>The nominal owner of the repository, i.e. the GitHub user who created the repository, can add collaborators in the repository’s settings page.

<sup>9</sup>See [YCombinator](#) and [GitHub](#) discussions.

### 3 Data

We use a mixture of openly available and proprietary data from GitHub to understand the effects of the Copilot AI program for linchpin maintainers. Maintainer activity is publicly available while Copilot AI usage is proprietary to GitHub. In collaboration with GitHub, we link the public platform activity to Copilot usage for a set of pseudonymized developers to form a panel of linchpin maintainer-week observations. From the granular activities data observed on the GitHub platform, we develop a set of two broad classifications of maintainer activities that are essential to collaborative software development. We consider the extent to which maintainers engage in (1) coding and (2) project management activities. Precise definitions for each classification can be found in Appendix B. We detail these categories in turn in the following sections.

First, coding includes maintainer activities that form the core of OSS contribution: pushing commits from local repositories to GitHub, forking existing projects to begin new development directions, and submitting pull requests for other maintainers to integrate proposed changes. These activities characterize the more technical process of writing lines of software code. Second, one can think of all other activities as non-coding activities. We classify an important subset of these non-coding activities as “project management”, process-oriented community interactions that seek to progress project development and require a more than purely technical skill set. A critical component of OSS collaboration is engagement within the repository’s “issues board”. It is here where project maintainers engage with the wider community to assist with software issues, introduce new ideas, and discuss longer term project objectives.<sup>10</sup> Other activities related to project management include the creation of project boards, which assist maintainers with project organization and “road-mapping”, and systematically reviewing the proposed contributions stemming from pull requests. Finally, we aggregate all items to obtain a measure of the total activities for each maintainer on the platform.

Table A1 shows univariate statistics for the Copilot AI treatment, the work activities and the

---

<sup>10</sup>For example, maintainers can open issues themselves, assign other maintainers to investigate certain issues, give issues labels for organization, and close the issue itself, thereby signaling that it is no longer an outstanding concern.

ranking for the first year.<sup>11</sup> We use a sample of active top maintainers who are collaborators on a repository that receives an eligibility ranking during the general access period. To leverage meaningful variation for the sub-sample of maintainers who received access for the first time, we exclude any maintainers who had Copilot experience during the technical preview. All individuals in our top maintainer dataset engaged in at least one of the activities mentioned above during our time window and, thus, are considered active users. This leads to a sample of 50,032 unique “top maintainers” — those who are maintainers of OSS projects that are amongst the most popular and widely used on GitHub. In this dataset, the average developer uses Copilot for approximately 8 days and on average 16% of the top maintainers use Copilot at some point within the general access period from July 2022 to July 2023. During each week, usage is relatively low, with 3% weekly exposure and 0.17 days of Copilot adoption per week. For coding and project management, we create a cumulative share measure defined as the cumulative sum of the activity over the maintainers overall total cumulative activity.<sup>12</sup> By conditioning on overall individual activity levels, we are better able to directly compare work allocation across maintainers and identify any potential reallocation in response to AI adoption. The average maintainer allocates 44% of their engagement towards coding and, 37% towards project management.<sup>13</sup> Finally, our measure for our natural experiment is the best (minimum) normalized ranking across eligible repositories that maintainers are connected to. For example, if an individual is a maintainer for two OSS projects, then the ranking of the lower ranked repository is used. The ranking distribution is right skewed with an average maintainer rank around -364 implying that a substantial amount of variation can be found before the normalized threshold of zero.

---

<sup>11</sup>A honed version of the descriptive statistics for the top maintainer within the bandwidth of  $h \in [-100, 100]$  can be found in Table A3. More granular statistics on work items are displayed in Table A4.

<sup>12</sup>Specifically, we use the following cumulative share version:  $\frac{\sum_{s \leq t} Activity_{it}}{\sum_{s \leq t} TotalActivities_{it}}$ .

<sup>13</sup>Table A2 provides the same statistics for a two year time period. AI usage increases substantially across all statistics.

## 4 Methodology: Natural Experiment

If we were to simply investigate the correlation between AI adoption and work patterns, a natural concern is that more able or motivated linchpins sort into higher usage of AI. This would immediately lead to an overestimation of the influence of AI on the nature of distributed work. To mitigate concerns over selection bias, we instead exploit quasi-random variation in Copilot access through GitHub’s “top maintainer” program. To credibly establish a causal effect of Copilot on OSS activity, we leverage a natural experiment. GitHub awarded free Copilot access to maintainers of the most popular public repositories according to an internal ranking. The top maintainer program was launched at the beginning of the general access period and was at least partially motivated by a desire to support critical OSS infrastructure. GitHub determines that a certain number of repositories are worthy of ranking and those repositories receive a ranking. The exact calculation of the ranking that determines which repositories receive rankings is not publicly disclosed by GitHub.

Since the ranking occurs at the project level, we assign each maintainer a weekly rank from the projects for which they are designated as collaborators. Specifically, the rank assigned to each maintainer is the running best (i.e. cumulative minimum) rank received across all repositories through the current observation period. Maintainers who receive a ranking below 0 are eligible for free access to Copilot while others are not. Maintainers only become aware of their eligibility status when applying for Copilot access. Since the maintainers have to check themselves whether they are eligible for free Copilot access in contrast to receiving a message from the platform, we expect the adoption of the AI to be less sharp than it would have been under alternative program implementations. However, we believe that a regression discontinuity design (RDD) is the best methodology in this environment allowing us to test not only its underlying assumptions while coming close to a randomized control trial from an internal validity perspective but also being able to study developers in their natural environment over a long time horizon.

This setting allows us to employ an RDD which in practice can be interpreted as a localized randomized control trial close to the threshold of AI eligibility. Whether a top maintainer receives

a ranking just above or just below the threshold can be considered to be as-good-as-random. Fortunately, the regression discontinuity design allows us to test its assumptions. For example, it is important to have no manipulation across the ranking for the RDD to be valid.<sup>14</sup> We have checked the credibility of this assumption through frequency plots and statistical tests of the running variable as well as covariate checks along the running variable and find evidence that is consistent with this assumption.<sup>15</sup>

To estimate the first stage of Copilot adoption, we employ the following model:

$$Copilot_{it} = \alpha_0 + \alpha_1 \mathbb{1}Eligible_{it} + \alpha_2 Ranking_{it} + \alpha_3 Eligible_{it} \times Ranking_{it} + \epsilon_{it}$$

where  $Ranking_{it}$  is the cumulative minimum Ranking across each eligible repository of a maintainer.  $Eligible_{it}$  is defined as  $\mathbb{1}(Ranking < 0)$  which is a parameter of interest from which we identify a change of Copilot usage at the normalized ranking threshold while considering a bandwidth of  $h \in [-100, 100]$ .

By construction, a change in access and, therefore, ultimately an exogenous change in adoption of GitHub Copilot is the only policy intervention operating on a linchpin maintainer as one crosses the ranking threshold of zero. We can therefore interpret any activity change across this threshold as caused by Copilot through the following intent-to-treat (ITT) estimation:

$$Activity_{it} = \beta_0 + \beta_1 \mathbb{1}Eligible_{it} + \beta_2 Ranking_{it} + \beta_3 Eligible_{it} \times Ranking_{it} + \epsilon_{it}$$

where  $\beta_1$  shows the causal effect of crossing the threshold — and through it, increased Copilot access — on the work activity of interest.

---

<sup>14</sup>In other words, neither the maintainer can manipulate the ranking to receive Copilot access nor should GitHub be able to manipulate the ranking to the benefit of some maintainers.

<sup>15</sup>Figure A2 visually shows a the distribution of the ranking without bunching at the threshold of zero. Table C1 shows the McCrary test and statistically does not identify any bunching, which adds to the idea that there is no evidence for a manipulation at the threshold. Visual depictions of covariate smoothness are shown in Figure C1

## 5 Main Results

### 5.1 Adoption of the Copilot Generative AI

As a society, we are at the beginning of the S-shaped adoption curve of generative AI overall. This is not any different for the programming AI Copilot from GitHub. 19 percent of maintainers used Copilot at least once (see Table A1). Further, GitHub was not engaging in any kind of advertising informing the top maintainers about their eligibility for free Copilot access. Top maintainers had to visit the GitHub Copilot website and check by themselves whether they were eligible or not. Fortunately, due to the large scale of our data based on the weekly level, we are able to identify even small changes.

— Figure 2 about here —

Figure 2 shows how crossing the threshold of zero from left to right alters programming AI adoption. Maintainers that are below the threshold of zero are eligible to receive Copilot for free with certainty while those above are not eligible for free through this channel. We find a significant drop of AI adoption based on the total number of days that AI has been used over the sample period when we cross the threshold, which implies that maintainers with free access are more likely to use the generative AI.<sup>16</sup>

— Table 1 about here —

Table 1 shows the exact coefficients on generative AI adoption of the top maintainer population across those who are eligible to access Copilot for a prize of zero dollars (i.e. ranking below zero) and those who are ineligible to access Copilot through the top maintainer channel (i.e. ranking above zero). The first two columns show estimates from the cross-section and the last two columns show estimates from the panel. We find an increase of AI adoption independent of the measure that is employed. For the cross-section, we observe an increase of 6.90 days (321% relative to

---

<sup>16</sup>The first stage is robust to a polynomial of degree two (see Figure C2)

the baseline) overall in AI adoption and an increase of 6.14 percentage points (61.2% relative to the baseline) of maintainers that ever adopted Copilot. For the panel, we find an increase of 0.12 days per week (223% relative to the baseline) and an exposure increase of 1.8 percentage points (214% relative to the baseline). Despite not widely advertising Copilot maintainer access, we find very strong effects ranging from 61% to 321% adoption, depending on the definition of the first stage. Independent of the definition of AI adoption, we find that our first stage is relevant with an F-value substantially beyond the recommended threshold of 10 where the cross-section F-values range from around 17 to 18 and the panel from 435 to 557.

## 5.2 Consequences of GenAI Usage

In this step, we analyze how generative AI causally changes work behavior in the digital world, namely the distributed work behavior of linchpin programmers that contribute to the public good of open source software.

— Table 2 about here —

Table 2 shows the intent-to-treat estimates of the Copilot generative AI on work activities. The table displays the reduced form effect of AI on activities that relate directly to programmer work. We find that coding activities increase by 5.4 percentage points (12.37% relative to the baseline) while project management drops by 10 percentage points (24.93% relative to the baseline). This indicates that overall coding activity is increasing due to the availability of Copilot (Figure 3 shows the same effects graphically).<sup>17</sup> In line with Copilot being a teacher and problem solver tool well suited to address problems or inquiries that would typically arise in the repository’s issues page, we observe a substantial drop in the average maintainers relative allocation towards activities related to project management. It implies that maintainers are less likely to seek out assistance from other humans. In this spirit, generative AI is helping the public good of open source programming since

---

<sup>17</sup>The results are robust to different bandwidth selections and functional form as shown in Figure C4. They are also robust to different kernel selections, as shown in Table C2.



linchpin maintainers have to solve fewer issues and they can focus on the core work: writing software code.

— Figure 4 about here —

Figure 4 shows the dynamic impact of a heightened propensity to use Copilot for free in the two years following general access. We find relatively stable coefficients across each period albeit with some ramp up and attenuation. In the first quarters, the effects are slightly weaker (in absolute terms) for the effect of Copilot on coding and project management relative to the first quarters where effects on increases up to 10 percent and -27 percent, respectively. The impacts of the generative coding tool Copilot continuously attenuate from the fourth quarter onward and stabilize around 2.5 percent for coding and 8 percent for project management. Hence, the strongest treatment effects of Copilot arise in the first year which is consistent with the idea that eligibility of the maintainers is re-evaluated after year one. Generally speaking, the pattern indicates that the benefits of accessing Copilot seem to be arising quite instantaneously and after some experimentation of the developers with it, the impacts are stable up to approximately two years for the sample of top maintainers.

To better understand the mechanisms of our treatment effect, we decompose our measures of coding and project management. Figure A3 Panel A shows that the positive effect of coding is driven mainly by pushes and some creation of repositories. In contrast, Copilot slightly reduces forking and the creation of pull requests. Illustrating some background on OSS contribution can help interpret these results. Following the contribution pattern popularized by the GitHub platform, if a developer wishes to make a contribution to an OSS project, they first fork the project, make their changes in the forked copy, and then formally ask the maintainer of the original project to integrate their changes (e.g. a “pull request”). This paradigm, while the basis of OSS development, is naturally beset with some degree of collaboration frictions. In contrast, pushes and the creation of repositories can happen autonomously by the linchpin maintainer. Together, these coefficient estimates indicate that generative AI enables linchpins to bypass collaboration frictions and more easily make unilateral code contributions to projects. The AI seems to increase not only new

innovations, but since adding code to already established projects exhibits the highest coefficient, the AI leads to a substantial upward shift in code contributions. This implies that the Copilot AI allows developers to shift their attention towards their core work activity while working more by themselves and less with others.

Figure A3 Panel B shows the decomposed treatment effects of the generative programming AI on project management. The itemized treatment effects are similarly heterogeneous for project management relative to coding. While most of the coefficients are negative, a few are zero (or close to it) and two are even positive. Linchpin maintainers with Copilot access close and merge issues at a higher rate than those that do not have access to the programming AI. In this process, they require fewer outside assistance which is consistent with a lower rate of requests for reviewers to other maintainers or the assignment of issues to others. They also have to review fewer pull requests and are subscribed to fewer issues which is consistent with those issues having been closed at a faster rate. Overall, the effect of Copilot on each component of project management measures is consistent with linchpins substituting away from work patterns that rely on others to solve coding issues and instead utilizing Copilot in place of human capital required in the previous interactions.<sup>18</sup>

In a next step, we leverage the same identification from the individual level and apply it to the project (repository) level. Figure A5 shows outcomes on coding and project management at the higher level. For coding, we know the amount of forking for each repository and we find that forking increases when a project provides AI access to linchpin maintainers. This implies that the individual-level coding effects are visible on the same repositories that the maintainers are connected to. Conversely, we observe project watching behavior as well as the average days issues are open and closed go down. Both the coding and the project management effects on the repository level are consistent with the individual level effects.

— Figure 5 about here —

---

<sup>18</sup>Another robust channel that would strengthen the Project Management effect if included is "Commenting". Linchpins with AI access seem to comment substantially less than those without (result upon request).

Finally we want to understand who benefits the most and if there is the potential to reduce inequality in the open source setting that is highly dependent on a few highly skilled developers. Figure 5 shows that the AI treatment effects across ability proxies where we conduct a median split across the measures of the centrality of the maintainer to a repository (where centrality is defined as the maintainer’s share of overall commits within a repository), their follower count and their organizational affiliation. We find that individuals that we classify as low on these measures exhibit larger increases of coding and larger reductions in project management. It indicates that the low ability linchpins benefit more from the Copilot programming AI and that the AI has the potential to reduce inequality of contributions for our digital infrastructure of open source software.

## 6 Discussion

Our empirical setting allows us to push the boundaries of existing knowledge on the impact of AI on the nature of work. The primary benefit of this natural experiment is that we are able to observe activities that are driven by AI for a longer period of time than most prior studies (two years) while still using very granular level work activity data. On the other hand, a limitation of this study is that we are not observing the exact code that individuals are programming and we are not observing precisely how individuals are using Copilot. While randomized controlled trials can focus on this aspect, they often have to restrict themselves to a small set of individuals. In contrast, we are able to study the long-run effects of AI on many daily work activities for a large group of programmers that are extremely important linchpin contributors to the public good open source software.

Another limitation of this study is that we are only observing contributions to public repositories. Much of the activity with AI may happen in private repositories and as such, we are likely substantially underestimating the impact that generative AI has on coding and project management behavior. Based on our data, we find that 45.51% of Copilot usage happens in weeks where no public (OSS) work activity is observed and GitHub has confirmed that the Copilot activity must

therefore be related to private repository activity. Further, there is likely an even higher fraction of private activity since this activity - while not measurable - is also likely occurring during weeks where public activities are also occurring. As such, we expect that around 50 percent of the activity of highly active programmers happens on private and 50 percent on public repositories. Hence, a reasonable estimate for the overall private-public effort could be doubling our estimates conditional on private behavior being affected in similar ways to publicly observable behavior. A further interesting feature of the open source software platform is that private repositories can become public over time, which implies that not only will learning from private repositories spill over to public ones due to maintainers programming with the AI in public repositories but also AI generated code may appear at an increased rate over time in the public sphere.

## **7 Conclusion**

This study seeks to shine light on the importance of AI, and in particular generative AI and its consequences on distributed work. Going beyond the first level understanding of whether or not it increases productivity, we dig deeper to understand how it changes the nature of digital work processes of linchpin adopters. We find that top developers of open source software are coding more and are engaging less in project management tasks. The decrease in project management activities is largely driven by maintainers addressing fewer problems that would arise on the project issue boards. It implies that those problems might have been solved by generative AI instead of a human, reducing overall collaboration frictions and other transaction costs characteristic of distributed work. We further find that the programming AI Copilot assists developers more with low ability relative to high ability.

Overall, our results are amongst the first that are able to illuminate the deeper level changes in a decentralized work processes instigated by AI over a long time period with very granular level information using a natural experiment. Furthermore, our study yields early insight into how generative AI shapes the voluntary private provision of critical digital public goods infrastructure

and how it ameliorates the linchpin problem. Indeed the scope for positive externalities inherent to the public goods setting suggests the efficiency gains that arise from introducing generative AI to the OSS production process can generate far-reaching spillover benefits to downstream users. We believe they will help managers and policy makers better understand the nuances of this nascent yet transformational technology.

## References

- Acemoglu, Daron. 2003. “Labor-and capital-augmenting technical change.” *Journal of the European Economic Association* 1 (1):1–37.
- Acemoglu, Daron, David Autor, Jonathon Hazell, and Pascual Restrepo. 2022. “Artificial intelligence and jobs: Evidence from online vacancies.” *Journal of Labor Economics* 40 (S1):S293–S340.
- Agrawal, Ajay, Joshua Gans, and Avi Goldfarb. 2019. “Economic policy for artificial intelligence.” *Innovation policy and the economy* 19 (1):139–159.
- Autor, David. 2024. “Applying AI to rebuild middle class jobs.” Tech. rep., National Bureau of Economic Research.
- Ballester, Coralio, Antoni Calvó-Armengol, and Yves Zenou. 2006. “Who’s who in networks. Wanted: The key player.” *Econometrica* 74 (5):1403–1417.
- Blind, Knut, Mirko Böhm, Paula Grzegorzewska, Andrew Katz, Sachiko Muto, Sivan Pätsch, and Torben Schubert. 2021. “The impact of Open Source Software and Hardware on technological independence, competitiveness and innovation in the EU economy.” *Final Study Report. European Commission, Brussels, doi 10:430161.*
- Boysel, Sam, Manuel Hoffmann, and Frank Nagle. 2024. “Labor Competition and Open Innovation.” *Working Paper* .
- Brynjolfsson, Erik, Danielle Li, and Lindsey R Raymond. 2023. “Generative AI at work.” Tech. rep., National Bureau of Economic Research.
- Brynjolfsson, Erik, Daniel Rock, and Chad Syverson. 2018. “Artificial intelligence and the modern productivity paradox: A clash of expectations and statistics.” In *The economics of artificial intelligence: An agenda*. University of Chicago Press, 23–57.
- Bughin, Jaques and J Manyika. 2018. “The promise and challenge of the age of artificial intelligence.” *McKinsey Global Institute, May. <https://www.mckinsey.it/idee/the-promise-and-challenge-of-the-age-of-artificial-intelligence>* 30 (10).
- Charness, Gary and Patrick Holder. 2019. “Charity in the laboratory: Matching, competition, and group identity.” *Management Science* 65 (3):1398–1407.
- Corrado, Carol, Jonathan Haskel, and Cecilia Jona-Lasinio. 2021. “Artificial intelligence and productivity: an intangible assets approach.” *Oxford Review of Economic Policy* 37 (3):435–458.
- Crafts, Nicholas. 2021. “Artificial intelligence as a general-purpose technology: an historical perspective.” *Oxford Review of Economic Policy* 37 (3).
- Czarnitzki, Dirk, Gastón P Fernández, and Christian Rammer. 2023. “Artificial intelligence and firm-level productivity.” *Journal of Economic Behavior & Organization* 211:188–205.

- Dell'Acqua, Fabrizio, Edward McFowland, Ethan R Mollick, Hila Lifshitz-Assaf, Katherine Kellogg, Saran Rajendran, Lisa Kraye, 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." *Harvard Business School Technology & Operations Mgt. Unit Working Paper* 24 (013).
- Dohmke, Thomas, Marco Iansiti, and Greg Richards. 2023. "Sea Change in Software Development: Economic and Productivity Analysis of the AI-Powered Developer Lifecycle." *arXiv preprint arXiv:2306.15033* .
- Eghbal, Nadia. 2020. *Working in public: the making and maintenance of open source software*. Stripe Press.
- Eloundou, Tyna, Sam Manning, Pamela Mishkin, and Daniel Rock. 2024. "GPTs are GPTs: Labor market impact potential of LLMs." *Science* 384 (6702):1306–1308.
- Emanuel, Natalia, Emma Harrington, and Amanda Pallais. 2023. "The power of proximity to coworkers: training for tomorrow or productivity today?" Tech. rep., National Bureau of Economic Research.
- Fischer, Mira, Rainer Michael Rilke, and B Burcin Yurtoglu. 2023. "When, and why, do teams benefit from self-selection?" *Experimental Economics* 26 (4):749–774.
- Fisher, Danyel, Marc Smith, and Howard T Welser. 2006. "You are who you talk to: Detecting roles in usenet newsgroups." In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, vol. 3. IEEE, 59b–59b.
- Fügener, Andreas, Jörn Grahl, Alok Gupta, and Wolfgang Ketter. 2022. "Cognitive challenges in human–artificial intelligence collaboration: Investigating the path toward productive delegation." *Information Systems Research* 33 (2):678–696.
- Furman, Jason and Robert Seamans. 2019. "AI and the Economy." *Innovation policy and the economy* 19 (1):161–191.
- Geiger, R Stuart, Dorothy Howard, and Lilly Irani. 2021. "The labor of maintaining and scaling free and open-source software projects." *Proceedings of the ACM on human-computer interaction* 5 (CSCW1):1–28.
- Godin, Seth. 2010. "Linchpin: Are You Indispensable?" *Teacher Librarian* 37 (4):77.
- Goldfarb, Avi, Bledi Taska, and Florenta Teodoridis. 2023. "Could machine learning be a general purpose technology? a comparison of emerging technologies using data from online job postings." *Research Policy* 52 (1):104653.
- Hinds, Pamela and Sara Kiesler. 2002. *Distributed work*. MIT press.
- Hoffmann, Manuel, Frank Nagle, and Yanuo Zhou. 2024. "The Value of Open Source Software." *Harvard Business School Strategy Unit Working Paper* 24 (038).

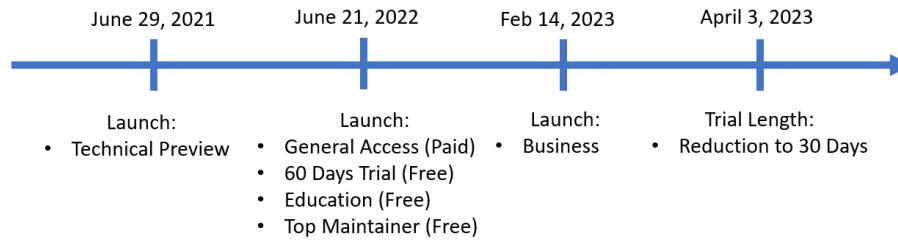
- Lerner, Josh and Jean Tirole. 2002. “Some simple economics of open source.” *The journal of industrial economics* 50 (2):197–234.
- Lifshitz-Assaf, H and F Nagle. 2021. “The digital economy runs on open source. Here’s how to protect it.” *Harvard Business Review* :1–7.
- McCrary, Justin. 2008. “Manipulation of the running variable in the regression discontinuity design: A density test.” *Journal of econometrics* 142 (2):698–714.
- Moon, Jae Yun and Lee Sproull. 2002. “Essence of Distributed Work: The Case of the Linux Kernel.” In *Distributed Work*. The MIT Press. URL <https://doi.org/10.7551/mitpress/2464.003.0023>.
- Nagle, Frank. 2018. “Learning by contributing: Gaining competitive advantage through contribution to crowdsourced public goods.” *Organization Science* 29 (4):569–587.
- . 2019. “Open source software and firm productivity.” *Management Science* 65 (3):1191–1215.
- Nagle, Frank, David A Wheeler, H Lifshitz-Assaf, H Ham, and J Hoffman. 2020a. “Report on the 2020 foss contributor survey.” *The Linux Foundation Core Infrastructure Initiative* .
- Nagle, Frank, Jessica Wilkerson, James Dana, and Jennifer L Hoffman. 2020b. “Vulnerabilities in the Core: Preliminary Report and Census II of Open Source Software.” *The Linux Foundation & The Laboratory for Innovation Science at Harvard* .
- Noy, Shakked and Whitney Zhang. 2023. “Experimental evidence on the productivity effects of generative artificial intelligence.” *Available at SSRN 4375283* .
- Panciera, Katherine, Aaron Halfaker, and Loren Terveen. 2009. “Wikipedians are born, not made: a study of power editors on Wikipedia.” In *Proceedings of the 2009 ACM International Conference on Supporting Group Work*. 51–60.
- Peng, Sida, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. 2023. “The impact of ai on developer productivity: Evidence from github copilot.” *arXiv preprint arXiv:2302.06590* .
- Raj, Manav and Robert Seamans. 2018. “Artificial intelligence, labor, productivity, and the need for firm-level data.” In *The economics of artificial intelligence: An agenda*. University of Chicago Press, 553–565.
- Raman, Naveen, Minxuan Cao, Yulia Tsvetkov, Christian Kästner, and Bogdan Vasilescu. 2020. “Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions.” In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*. 57–60.
- Robbins, Carol, Gizam Korkmaz, Ledia Guci, J Bayoán Santiago Calderón, and Brandon Kramer. 2021. “A First Look at Open-Source Software Investment in the United States and in Other Countries, 2009-2019.” In *Paper presented the IARIW-ESCoE Conference, Measuring Intangible Capitals and their Contribution to Growth (November, RSA House, London)*. 1–30.



- Sachs, Goldman. 2023. “Generative AI could raise global GDP by 7%.” *GoldmanSachs.com* .
- Synopsys. 2023. “023 OSSRA: A deep dive into open source trends.” *Synopsys, May 1 2023*.  
<https://www.synopsys.com/blogs/software-security/open-source-trends-ossra-report/> 05 (1).
- Tamayo, Jorge, L Doumi, S Goel, O Kovács-Ondrejko, and R Sadun. 2023. “Reskilling in the age of AI.” *Harvard Business Review* 21.
- U.S. Department of Labor. 2024. “Artificial Intelligence and Worker Well-being: Principles for Developers and Employers.” URL <https://www.dol.gov/general/AI-Principles>.
- Wright, Nataliya Langburd, Frank Nagle, and Shane Greenstein. 2023. “Open source software and global entrepreneurship.” *Research Policy* 52 (9):104846.
- Wright, Patrick M, Russell Coff, and Thomas P Moliterno. 2014. “Strategic human capital: Crossing the great divide.” *Journal of Management* 40 (2):353–370.

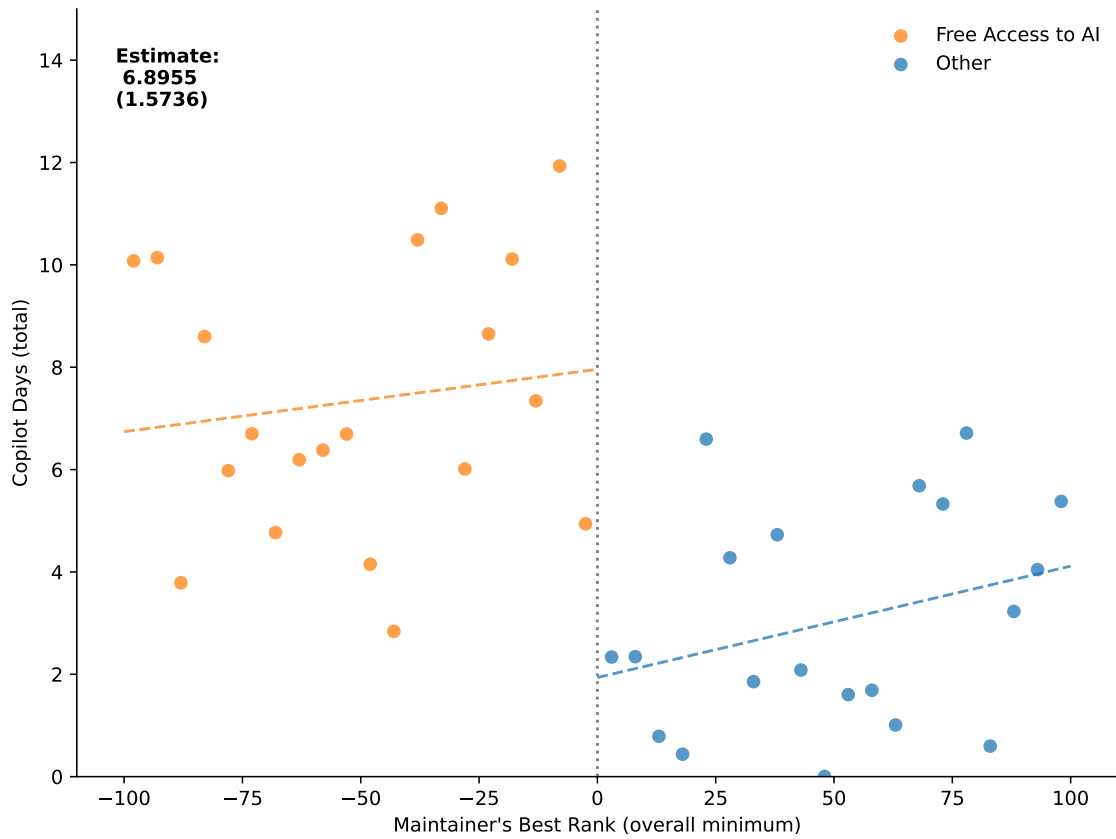
# Figures and Tables

Figure 1: GITHUB COPILOT AI DEPLOYMENT TIMELINE



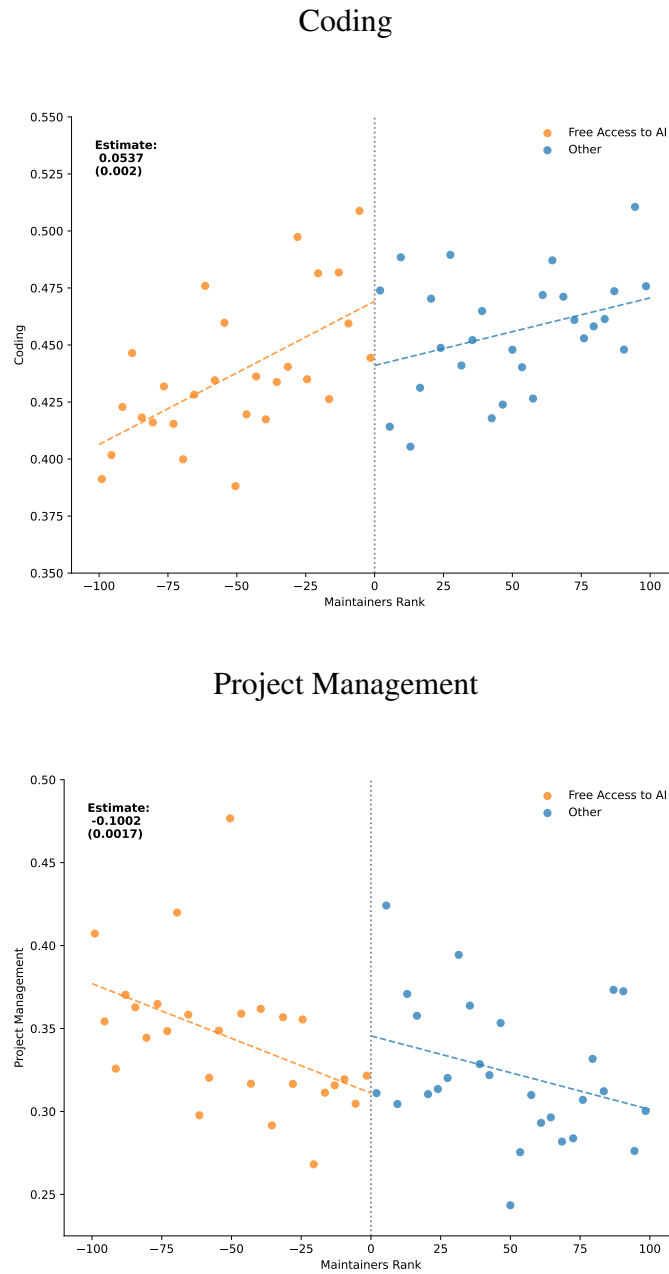
*Note:* The figure shows the timeline for the introduction of the generative AI GitHub Copilot.

Figure 2: COPILOT AI ADOPTION ACROSS RANKS



*Note:* The figure shows the total number of days the Copilot GenAI was used across the overall normalized minimum ranking on GitHub using a linear fit on either side of the threshold within the bandwidth of  $h \in [-100, 100]$ . Maintainers with rankings below zero receive free access to the AI through the top maintainer channel while those above do not. Time frame: July, 2022 to July 2023. Robust standard errors are in parentheses.

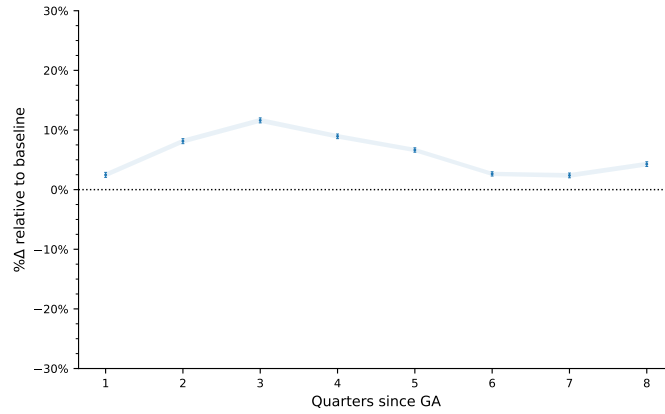
Figure 3: INTENT-TO-TREAT AVERAGE EFFECTS OF COPILOT AI



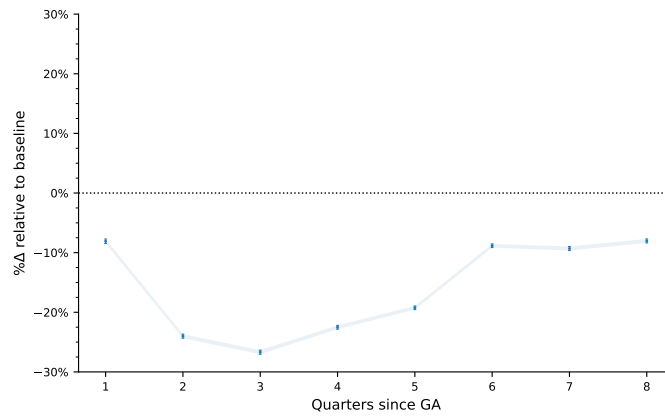
*Note:* The figure shows the intent-to-treat (ITT) effect of the GitHub Copilot GenAI on the outcomes of Coding and Project Management using the overall normalized minimum ranking on GitHub via a linear fit on either side of the threshold within the bandwidth of  $h \in [-100, 100]$ . Maintainers with rankings below zero receive free access to the AI through the top maintainer channel while those above do not. Time frame: July, 2022 to July, 2023. Robust standard errors are in parentheses.

Figure 4: INTENT-TO-TREAT EFFECTS OF COPILOT OVER TWO YEARS

Panel A: Coding



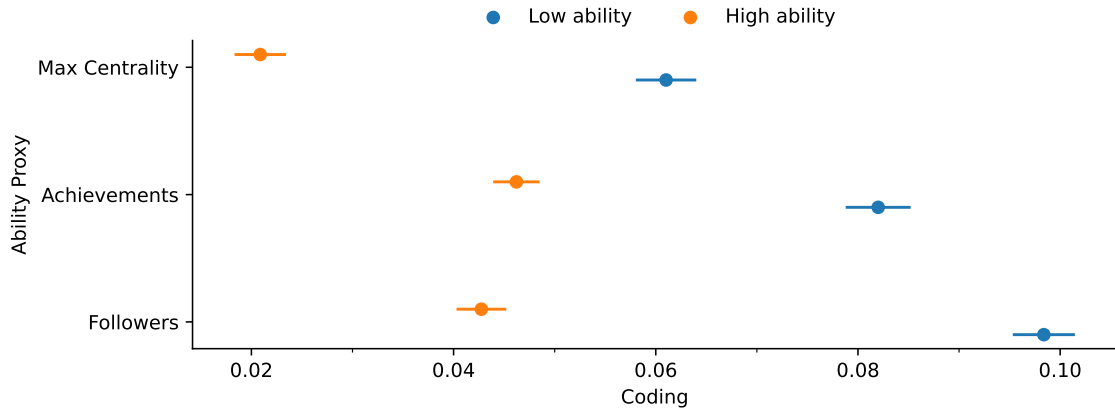
Panel B: Project Management



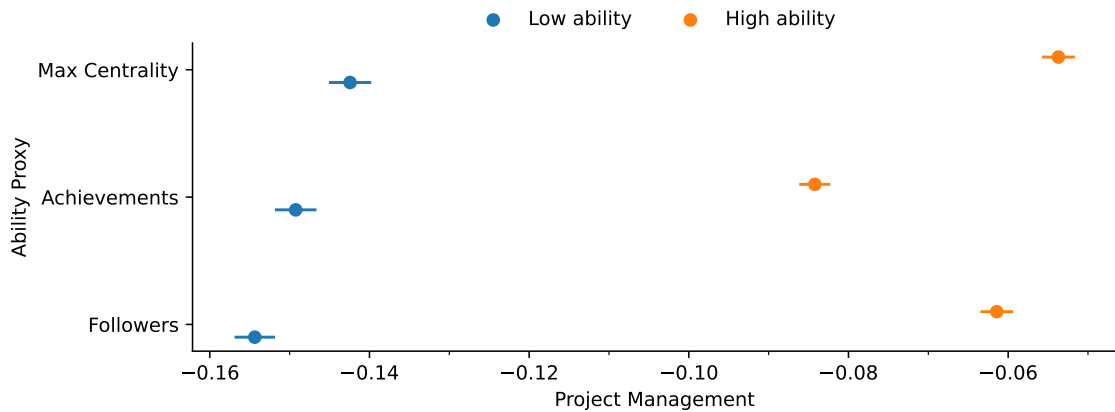
*Note:* The figures plots intent-to-treat (ITT) coefficient estimates and confidence intervals for top maintainers being ranked below zero relative to above a zero ranking on the outcomes of Coding and Project Management by pooling observations in quarters since the general access (GA) period. Time frame: July, 2022 to July, 2024. The confidence intervals are based on robust standard errors.

Figure 5: INTENT-TO-TREAT EFFECTS OF COPILOT ACROSS ABILITY

Panel A: Coding



Panel B: Project Management



*Note:* The figures show intent-to-treat (ITT) coefficient estimates and confidence intervals on the outcomes of Coding and Project Management for top maintainers being ranked below zero relative to above a zero ranking across low and high ability developers. Ability is measured based on ability proxies of developer centrality, follower count and achievements split by the median since the general access period. Low ability individuals are below the median for each ability proxy while high ability are above the median. Time frame: July, 2022 to July, 2023. The confidence intervals are based on robust standard errors.

Table 1: AI ADOPTION ACROSS MAINTAINER RANKING

	AI Adoption			
	Cross Section		Panel	
	Total Days	Ever Adoption	Days/Week	Exposure Share
$\mathbb{1}(Eligible)$	6.896*** (1.574)	0.0614** (0.023)	0.1247*** (0.008)	0.0180*** (0.001)
Baseline	2.146*** (0.593)	0.1004*** (0.013)	0.0557*** (0.003)	0.0084*** (0.000)
Rel. TE (%)	321.4	61.2	223.8	214.3
F	18.00	17.75	435.7	557.5
N	4,268	4,268	215,169	215,169

Note: This table shows the first stage uptake of the generative AI tool Copilot for top maintainers that are eligible based on the internal GitHub ranking relative to those who are not eligible through this pathway using a bandwidth of ranks  $h \in [-100, 100]$ . The first two columns show cross-sectional estimates on the total days of AI adoption and whether a maintainer ever adopted the AI. The last two columns show the panel estimates of the days per week of AI adoption and exposure shares, i.e. the cumulative days of AI adoption over the cumulative total days from the general access period onward. Time frame: July, 2022 to July, 2023. Robust standard errors are in parentheses. \*\*\* $p < 0.001$ , \*\* $p < 0.01$ , \* $p < 0.05$

Table 2: INTENT-TO-TREAT EFFECT OF COPILOT AI ON WORK ACTIVITIES

Panel A:	Coding	
$\mathbb{1}(Eligible)$	0.0537*** (0.002)	0.0354*** (0.002)
Baseline	0.4338*** (0.001)	0.4338*** (0.001)
Rel. TE (%)	12.37	8.16
N	269,546	248,054
Controls		✓

Panel B:	Project Management	
$\mathbb{1}(Eligible)$	-0.1002*** (0.002)	-0.0969*** (0.002)
Baseline	0.4019*** (0.001)	0.4103*** (0.001)
Rel. TE (%)	-24.93	-23.61
N	269,546	248,054
Controls		✓

Note: This table shows the coefficient estimates from the intent-to-treat specification across each activity classification for top maintainers that are eligible based on the internal GitHub ranking relative to those who are not eligible through this pathway using a bandwidth of ranks  $h \in [-100, 100]$ . Covariate controls include the maintainer's number of GitHub achievements, number of followers, and a measure of their share of repository activity (i.e. centrality). Time frame: July, 2022 to July, 2023. Robust standard errors are in parentheses. \*\*\* $p < 0.001$ , \*\* $p < 0.01$ , \* $p < 0.05$



# Appendices

## Appendix A Additional Tables and Figures

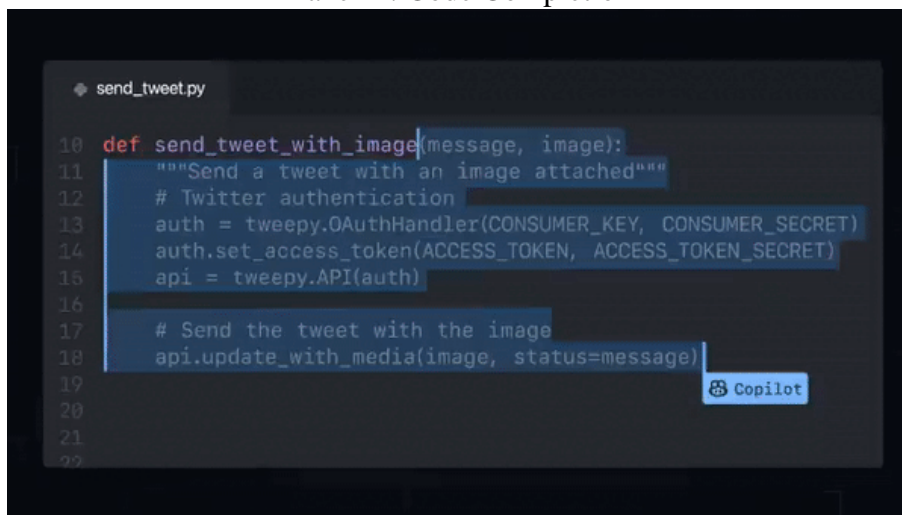
Figure A1: GITHUB COPILOT: ARTIFICIAL INTELLIGENCE IN ACTION

Panel A: Function Header



```
send_tweet.py
10 def send_tweet_with_image|
11
12
13
14
15
16
17
18
19
20
21
22
```

Panel B: Code Completion

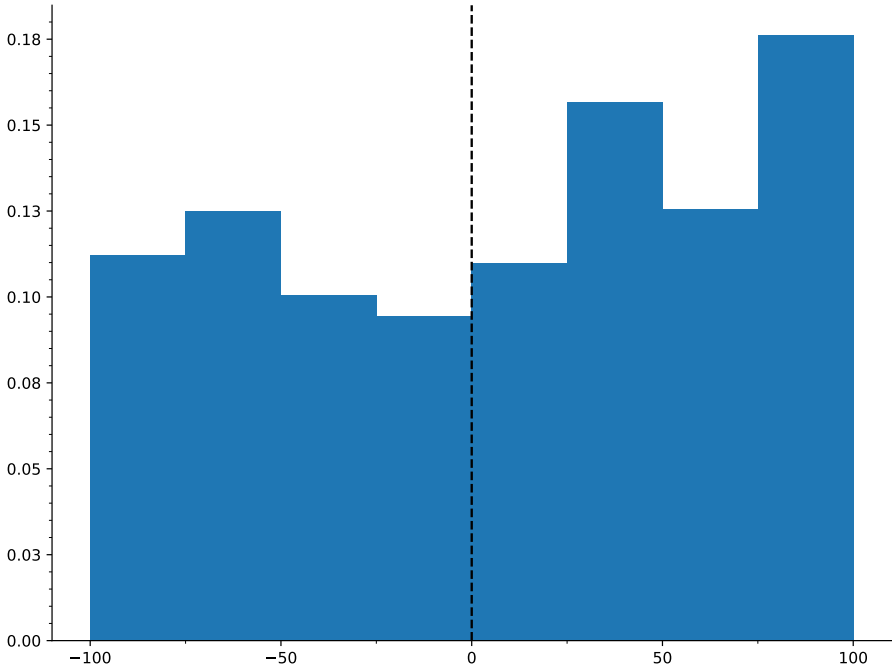


```
send_tweet.py
10 def send_tweet_with_image(message, image):
11     """Send a tweet with an image attached"""
12     # Twitter authentication
13     auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
14     auth.set_access_token(Access_token, Access_token_secret)
15     api = tweepy.API(auth)
16
17     # Send the tweet with the image
18     api.update_with_media(image, status=message)
19
20
21
22
```

*Note:* The figure shows a function generated by GitHub Copilot. The user wrote the function header (Panel A) and Copilot suggested the rest of the function (Panel B).

*Source:* GitHub 2022

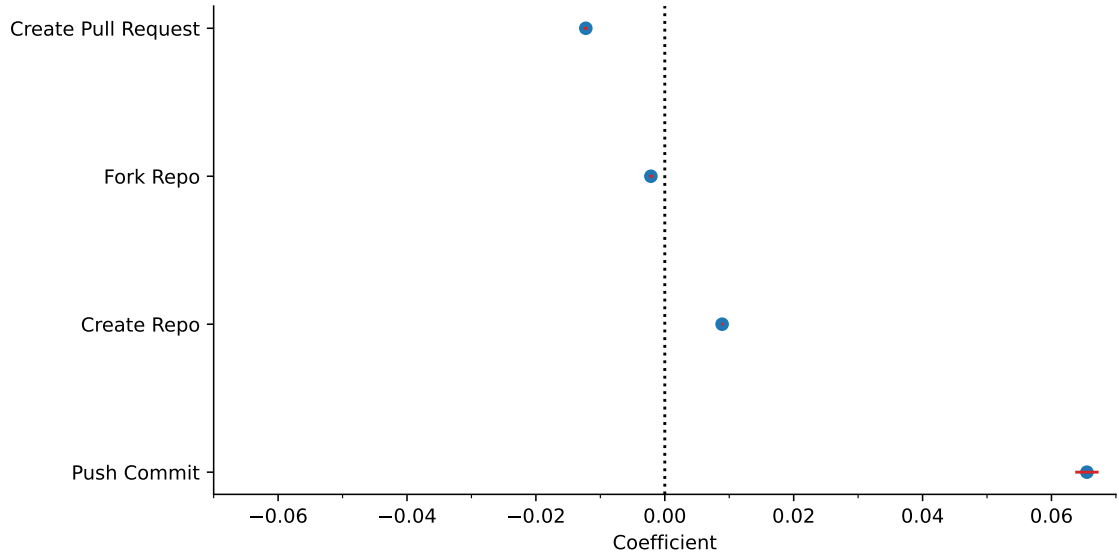
Figure A2: DISTRIBUTION OF MINIMUM RANKINGS



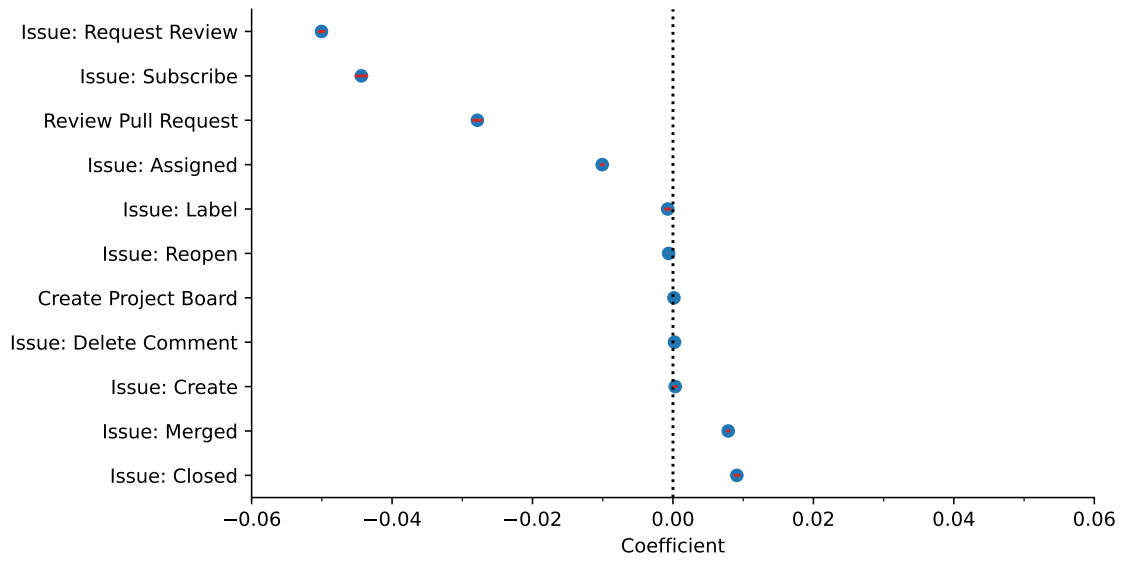
*Note:* The figure shows the distribution of the minimum rankings for the top maintainer based on the internal ranking system from GitHub. The dashed line is at the eligibility threshold of the rank zero.

Figure A3: GRANULAR INTENT-TO-TREAT EFFECTS OF COPILOT

Panel A: Coding



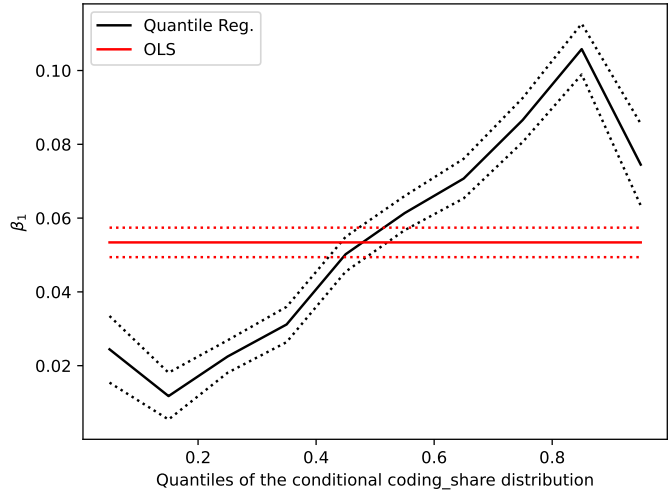
Panel B: Project Management



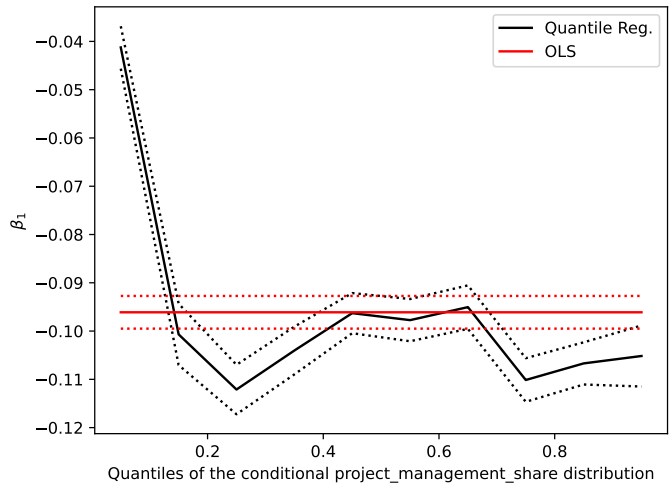
*Note:* The figures show the intent-to-treat (ITT) coefficient estimates on the granular outcomes of Coding and Project Management since the general access period. Time frame: July, 2022 to July, 2023. Confidence intervals are based on robust standard errors.

Figure A4: INTENT-TO-TREAT QUANTILE EFFECTS OF COPILOT AI

### Coding



### Project Management



*Note:* The figure shows the quantile intent-to-treat (ITT) effects of the Copilot GenAI on the outcomes of Coding and Project Management using the overall normalized minimum ranking on GitHub via a linear fit on either side of the threshold. Maintainers with rankings below 0 receive free access to the AI through the top maintainer channel while those above do not. Time frame: July, 2022 to July, 2023. All confidence intervals are based on robust standard errors.

Table A1: DESCRIPTIVE STATISTICS OF TOP MAINTAINERS FOR 1 YEAR

	Mean	SD	Min	Max
<b>AI Treatment</b>				
AI Total Days	8.12	29.17	0	349
AI Ever Used	0.16	0.37	0	1
AI Exposure Share	0.03	0.11	0	1
AI Days Used / Week	0.17	0.87	0	7
<b>Work Activities</b>				
Coding	0.44	0.22	0	1
Project Management	0.37	0.20	0	1
All Activities	25.79	267.70	0	8,665
<b>Ranking</b>				
Normalized Ranking	-363.94	560.98	-999	1000

Note: The table shows univariate descriptive statistics for top maintainers over one year with the arithmetic mean (mean) in the first column, followed by the standard deviation (SD), the lowest value of a variable (min), the highest value of a variable (max) and the number of observations (N). Our full balanced panel contains 2,422,916 observations for 50,032 maintainers within a time period from July 2022 - July 2023.

Table A2: DESCRIPTIVE STATISTICS OF TOP MAINTAINERS FOR 2 YEARS

	Mean	SD	Min	Max
<b>AI Treatment</b>				
AI Total Days	27.66	76.52	0	717
AI Ever Used	0.24	0.43	0	1
AI Exposure Share	0.05	0.14	0	1
AI Days Used / Week	0.29	1.12	0	7
<b>Work Activities</b>				
Coding	0.45	0.22	0	1
Project Management	0.36	0.19	0	1
All Activities	23.74	253.10	0	198,498
<b>Ranking</b>				
Normalized Ranking	-368.22	558.72	-999	1000

Note: The table shows univariate descriptive statistics for top maintainers over two years with the arithmetic mean (mean) in the first column, followed by the standard deviation (SD), the lowest value of a variable (min), the highest value of a variable (max) and the number of observations (N). Our full balanced panel contains 5,381,132 observations for 55,496 maintainers within a time period from July 2022 - July 2024.

Table A3: DESCRIPTIVE STATISTICS OF TOP MAINTAINERS WITHIN MAIN BANDWIDTH

	Mean	SD	Min	Max
<b>AI Treatment</b>				
AI Total Days	4.46	19.06	0	264
AI Ever Used	0.13	0.34	0	1
AI Exposure Share	0.02	0.09	0	1
AI Days Used / Week	0.11	0.70	0	7
<b>Work Activities</b>				
Coding	0.44	0.22	0	1
Project Management	0.37	0.19	0	1
All Activities	17.28	51.50	0	4,942
<b>Ranking</b>				
Normalized Ranking	0.62	60.33	-100	100

Note: The table shows univariate descriptive statistics for top maintainers within a bandwidth of  $h \in [100, 100]$  with the arithmetic mean (mean) in the first column, followed by the standard deviation (SD), the lowest value of a variable (min), the highest value of a variable (max) and the number of observations (N) for top maintainer within the bandwidth of  $h \in [-100, 100]$ . Our full balanced panel contains 215,169 observations for 5,521 maintainers within a time period from July 2022 - July 2023.

Table A4: DESCRIPTIVE STATISTICS OF TOP MAINTAINER FOR ALL WORK ACTIVITIES

	Mean	SD	Min	Max
<b>Coding</b>				
Create Repository	0.04	0.32	0	173
Fork Repository	0.07	1.40	0	1,327
Pull Request	1.03	15.52	0	5,468
Push	10.15	236.12	0	85,302
<b>Project Management</b>				
Issue: Assigned	0.48	3.79	0	1,616
Issue: Closed	1.97	17.21	0	4,889
Issue: Created	0.30	3.97	0	1,591
Issue: Comment Deleted	0.01	0.49	0	225
Issue: Labeled	1.94	43.13	0	10,161
Issue: Merged	1.18	9.04	0	4,338
Issue: Reopened	0.04	1.02	0	340
Issue: Review Requested	1.08	12.06	0	2,687
Issue: Subscribed	1.47	8.91	0	1,550

Note: The table shows univariate descriptive statistics with the arithmetic mean (mean) in the first column, followed by the standard deviation (SD), the lowest value of a variable (min), the highest value of a variable (max) across all individual work activities that are included in the categories from Table A1. Our full balanced panel contains 2,422,916 observations within a time period from July 2022 - July 2023.



Table A5: REPOSITORY LEVEL EFFECTS OF COPILOT AI

	<b>Coding</b>	<b>Project Management (Issues)</b>	
	Forks	Avg. Days Open	Avg. Days Closed
$\mathbb{1}(\text{Eligible})$	0.051*** (0.010)	-43.881*** (4.776)	-3.162*** (0.680)
N	196,539	493,271	507,820

Note: The table shows treatment effects of at the repository level on outcomes for repositories that have a ranking just below zero and provided free access to Copilot relative to those above the threshold which did not. Time frame: July 2022 - July 2023. Robust standard errors are in parentheses.

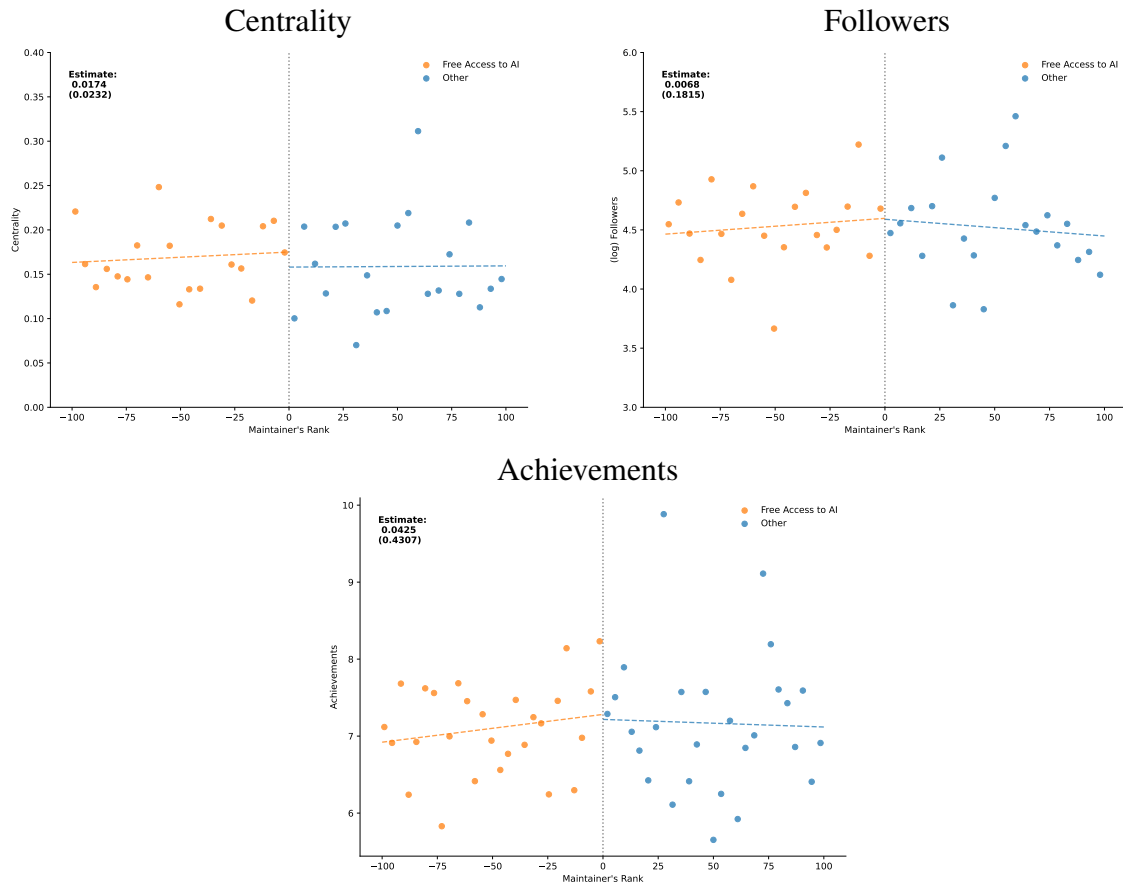
## Appendix B Classification of Work Activities

Each activity is defined as the sum of its disaggregated constituent activities. See below:

<b>Coding</b>	<b>Project Management</b>
Create Repository	Created Project Board
Fork Repository	Issue Assigned
Pull Request	Issue Closed
Push	Issue Comment Deleted
	Issue Closed
	Issue Labeled
	Issue Merged
	Issue Reopened
	Issue Review Requested
	Issue Subscribed
	Reviewed Pull Request

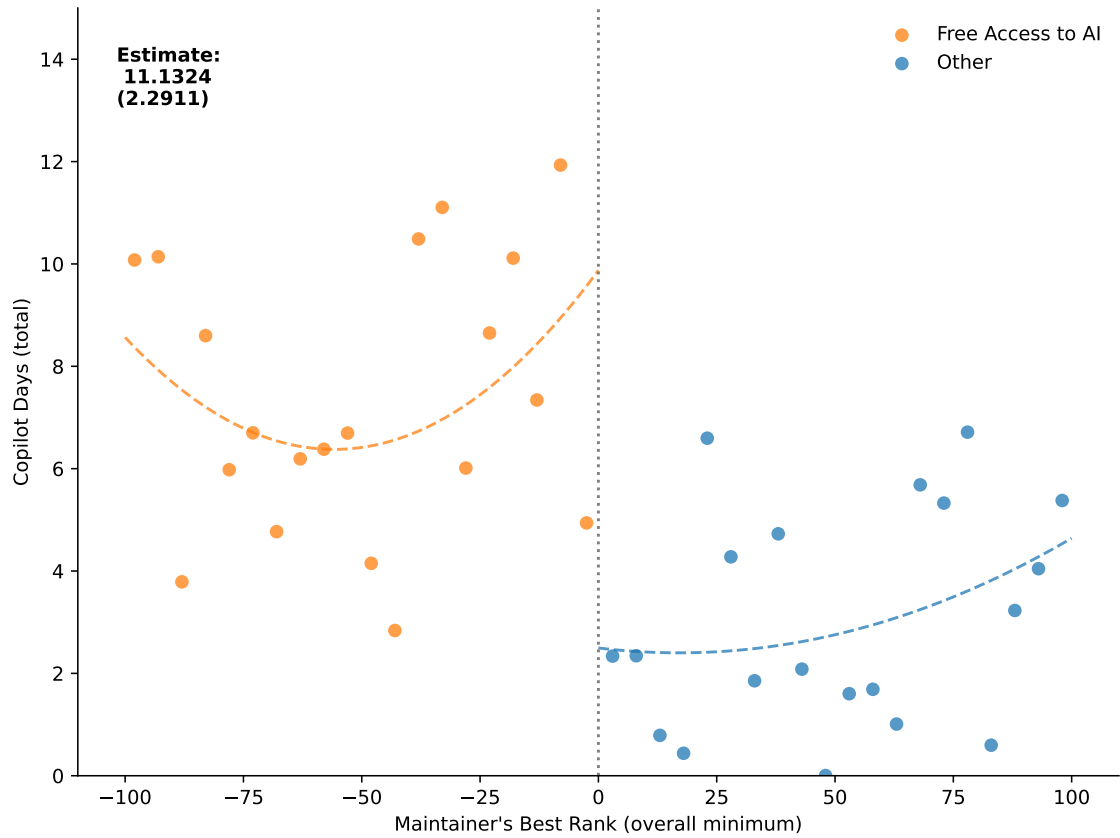
# Appendix C Robustness Checks

Figure C1: COVARIATE SMOOTHNESS



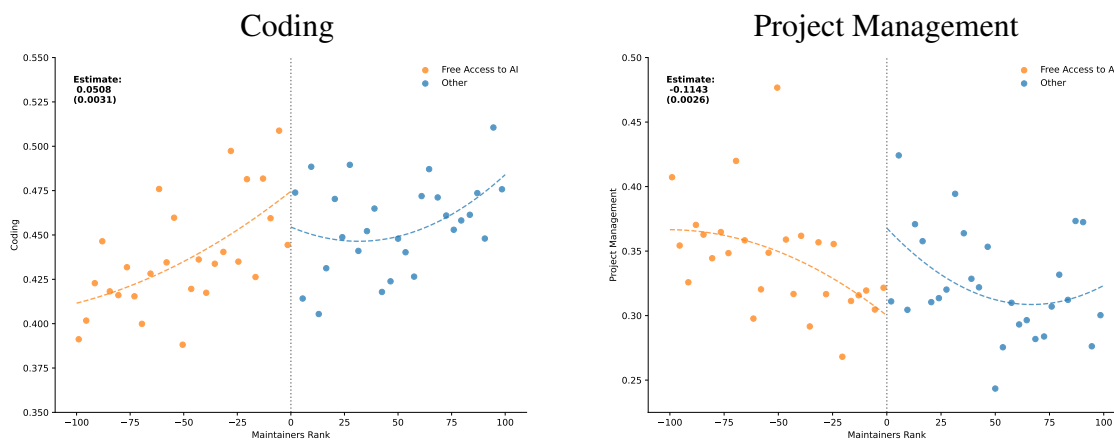
*Note:* The figure shows the covariate smoothness check across the overall minimum ranking on GitHub for measures of maintainer platform achievements, follower counts, and a measure of the maintainer's centrality within their repository. After the general access period, maintainers with rankings below 0 receive free access to the AI through the top maintainer channel while those above do not. Time frame: Covariates are observed between January, 2021 and July, 2023. Maintainer rankings are observed from July 2022 through July 2023. Robust standard errors are in parentheses.

Figure C2: COPILOT AI ADOPTION ACROSS RANKS WITH POLYNOMIAL OF DEGREE 2



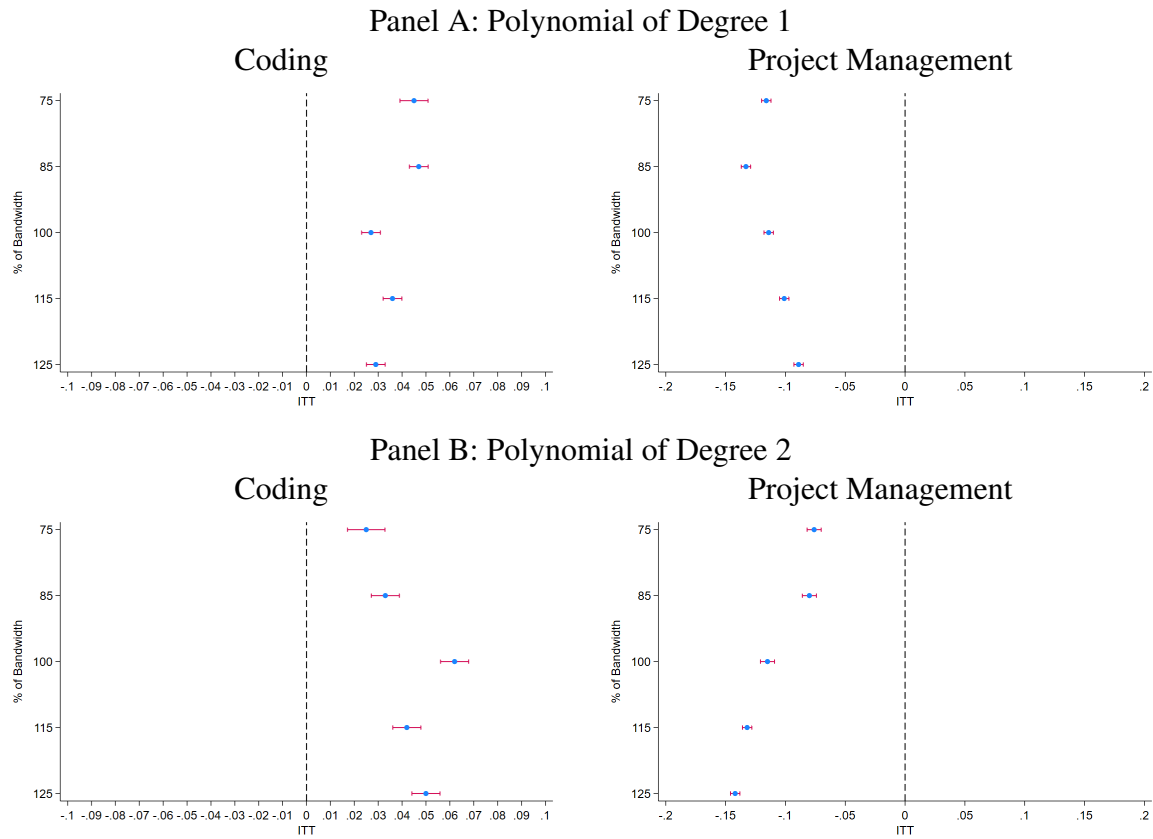
*Note:* The figure shows the total number of days the Artificial Intelligence (AI) GitHub Copilot was used across the overall minimum ranking based on the six top languages on GitHub with a quadratic fit on either side of the threshold. Maintainers with rankings below 0 receive free access to the AI through the top maintainer channel while those above do not. Time frame: July, 2022 to July, 2023. Robust standard errors are in parentheses.

Figure C3: INTENT-TO-TREAT EFFECTS OF COPILOT AI WITH POLYNOMIAL OF DEGREE 2



*Note:* The figure shows the total number of days the generative AI tool Copilot is used across the overall minimum ranking on GitHub using a linear fit on either side of the threshold. Maintainers with rankings below zero receive free access to the AI through the top maintainer channel while those above do not. Time frame: July, 2022 to July, 2023. Robust standard errors are in parentheses.

Figure C4: ITT OF VARYING BANDWIDTHS, POLYNOMIAL OF DEGREE ONE



*Note:* The figure shows the ITT effects when crossing the ranking threshold from the right to the left. Panel A (B) displays linear (quadratic) ITT effects. The bandwidth of 100 is our baseline estimation using  $h \in [-100, 100]$ . Time frame: July, 2022 to July, 2023. Confidence intervals are based on robust standard errors.

Table C1: McCRARY DENSITY TEST

	Ranking Frequency
$\mathbb{1}(Eligible)$	-477.607 (417.169)
N	201

Note: This table shows results from the McCrary test ([McCrary 2008](#)) for the frequency of a ranking  $\in [-100, 100]$  bandwidth, aggregated to the rank level. Robust standard errors are in parentheses. \*\*\* $p < 0.001$ , \*\* $p < 0.01$ , \* $p < 0.05$

Table C2: DIFFERENT KERNELS FOR COPILOT INTENT-TO-TREAT EFFECTS

Coding	Uniform	Triangular	Epanechnikov
$\mathbb{1}(Eligible)$	0.0537*** (0.002)	0.0523*** (0.002)	0.0539*** (0.002)
N	215,169	215,169	215,169
Project Mgmt.	Uniform	Triangular	Epanechnikov
$\mathbb{1}(Eligible)$	-0.1002*** (0.002)	-0.1067*** (0.001)	-0.1101*** (0.002)
N	215,169	215,169	215,169

Note: This table shows the intent-to-treat effects of the generative AI tool Copilot on work activities of coding and project management using a uniform kernel, a triangular kernel and an Epanechnikov kernel in column one, two, and three. Our full balanced panel is observed within a time period from July, 2022 to July, 2023. Robust standard errors are in parentheses. \*\*\*  $p < 0.001$ , \*\*  $p < 0.01$ , \*  $p < 0.05$



Table C3: OPTIMAL BANDWIDTHS FOR COPILOT INTENT-TO-TREAT EFFECTS

Coding	MSE	CER
$\mathbb{1}(Eligible)$	0.0442*** (0.003)	0.0300*** 0.003
N	97,367	71,222
Optimal bandwidth	[-47, 47]	[-35, 35]
Project Management	MSE	CER
$\mathbb{1}(Eligible)$	-0.0964*** (0.002)	-0.0580*** 0.003
N	105,027	74,668
Optimal bandwidth	[-51, 51]	[-38, 38]

Note: This table shows the intent-to-treat effects of the generative AI tool Copilot on work activities of coding and project management using the optimal bandwidth mean squared error (MSE) and the coverage error rate (CER) estimator in column 1 and 2. Our full balanced panel is observed within a time period from July, 2022 to July, 2023. Robust standard errors are in parentheses. \*\*\*  $p < 0.001$ , \*\*  $p < 0.01$ , \*  $p < 0.05$