CESifo WORKING PAPERS

11479
2024

November 2024

Generative AI and the Nature of Work

Manuel Hoffmann, Sam Boysel, Frank Nagle, Sida Peng, Kevin Xu

CESifo

Impressum:

CESifo Working Papers
ISSN 2364-1428 (electronic version)
Publisher and distributor: Munich Society for the Promotion of Economic Research - CESifo GmbH
The international platform of Ludwigs-Maximilians University's Center for Economic Studies and the ifo Institute
Poschingerstr. 5, 81679 Munich, Germany
Telephone +49 (0)89 2180-2740, Telefax +49 (0)89 2180-17845, email office@cesifo.de
Editor: Clemens Fuest
https://www.cesifo.org/en/wp
An electronic version of the paper may be downloaded
· from the SSRN website: www.SSRN.com
· from the RePEc website: www.RePEc.org
· from the CESifo website: https://www.cesifo.org/en/wp

CESifo *Working Paper No. 11479*

Generative AI and the Nature of Work

Abstract

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 work itself. How do individuals, especially those in the knowledge economy, adjust how they work when they start using AI? Using the setting of open source software, 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 for 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 the task allocation of software developers within a quasi-experimental regression discontinuity design. We find that having access to Copilot induces such individuals to shift task allocation towards their core work of coding activities and away from non-core project management activities. We identify two underlying mechanisms driving this shift - an increase in autonomous rather than collaborative work, and an increase in exploration activities rather than exploitation. The main effects are greater for individuals with relatively lower ability. Overall, our estimates point towards a large potential for AI to transform work processes and to potentially flatten organizational hierarchies in the knowledge economy.

JEL-Codes: H400, O300, J000.

Keywords: generative artificial intelligence, digital work, open source software, knowledge economy.

Manuel Hoffmann
Harvard Business School, Harvard University
Boston / MA / USA
mhoffmann@hbs.edu

Sam Boysel
Harvard Business School, Harvard University
Boston / MA / USA
sboysel@hbs.edu

Frank Nagle
Harvard Business School, Harvard University
Boston / MA / USA
frank@hbs.edu

Sida Peng
Microsoft Corporation
Redmond / WA / USA
sidpeng@microsoft.com

Kevin Xu
GitHub Inc.
k3vn.xu@gmail.com

This version: October 27, 2024
The authors are grateful for financial and administrative support from GitHub and, in particular, for generous advice from Peter Cihon. We thank Shane Greenstein, Tim Simcoe, David Autor, and Sam Ransbotham for their feedback. The authors are also indebted for comments by seminar participants at the research seminars from the Harvard Laboratory for Innovation Science, Boston University, the Massachusetts Institute of Technology, and the University of Passau. We are further grateful for feedback from participants at the "Labor in the Age of Generative AI" conference at the University of Chicago, the NBER SI 2024 Digital Economics and Artificial Intelligence in Cambridge, MA, the 2024 NBER Productivity Seminar in Cambridge, MA, the 2024 Academy of Management Science in Chicago, IL, the 22nd ZEW Economics of ICT conference, in Mannheim, Germany, the 20th Symposium on Statistical Challenges in Electronic Commerce Research in Lisbon, Portugal, the ACM Collective Intelligence Conference in Boston, MA, the MIT Code Conference in Cambridge MA and the CESifo Area Conference on Economics of Digitization 2024 in Munich, Germany.

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 artificial intelligence (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). 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 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, especially in knowledge intensive industries (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 production settings. This is particularly true of generative AI (generative AI) — 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,[1] 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, especially in the context of distributed work, has gone under-explored.

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

[1]Formerly known as Bard.

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? To answer these questions, we develop a theoretical model that leads to testable hypotheses that offer insights into where and why the most salient impacts are likely to occur. Understanding these impacts informs labor strategy in a manner 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 reskilling efforts for current employees.

The key challenge in testing our hypotheses and assessing how AI changes the nature of work is to identify a setting where (1) work patterns are observable and (2) an AI tool specifically tailored for workers has been introduced in a quasi-exogenous manner. Our setting — the introduction of GitHub Copilot, a software development generative AI tool, for key developers (known as maintainers) in open source software (OSS) projects — addresses both of these criteria. OSS source code is publicly available and permissively licensed for use, modification, and redistribution. Frequently developed by distributed teams of developers, OSS is a classic example of a product that is produced through the distributed work of teams and is generally free (Moon and Sproull, 2002). Although OSS creates societal value on the order of trillions of dollars (Hoffmann, Nagle, and Zhou, 2024) and is therefore important in its own right, we argue and provide suggestive evidence that the findings in this setting generalize to the broader set of work activities that occur in the knowledge economy. Further, as with many team production settings, OSS also suffers from the "linchpin" problem (Ballester, Calvó-Armengol, and Zenou, 2006; Godin, 2010) as a small set of developers 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). In practice, an influx of non-experts enabled by decreasing communication costs (Altman, Nagle, and Tushman, 2015) creates an additional burden on developers, who must triage support requests, review contributions, and otherwise manage their project's growing community. Indeed, survey evidence documents that those 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 key individuals are of great interest to the distributed production setting of OSS and are likely to generalize to numerous other settings as distributed work has become increasingly common.

We exploit aspects of the general access launch of Github Copilot to the broader public in June 2022 to establish causal effects of generative AI where some developers below a certain threshold of an internal ranking received free access to the coding AI and others did not. We start with a panel of 187,489 distinct developers observed weekly from July 2022 through July 2024, which results in millions of developer-week observations for Copilot usage and activity levels in public GitHub repositories.[2] Within the data set of top developers, we find that those who receive free 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. We dig further into the mechanisms underlying these effects and find that they are driven by an increase in autonomous behavior (and a related decrease in collaborative behavior) and an increase in exploration behavior (rather than exploitation). Further, we find lower ability developers who receive access to AI increase coding and reduce project management to a greater extent compared with their higher ability peers. The results are robust to the standard regression discontinuity design tests and to different estimation procedures such as difference-in-difference and matching. Further, the results are consistent when considering whether developers are working on behalf of their employers or as volunteers, adding support to the likelihood that these findings generalize beyond the OSS setting to a broader set of workers.

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).

[2] A GitHub repository is a location where all aspects of a project are stored including its source code, documentation, and revision history.

Subsequent empirical work has largely confirmed these 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). Particularly related to this study, research focused on Copilot specifically has either been conducted using a much smaller sample of workers within firms (Cui et al., 2024) or relying on observational data without the benefit of knowing precisely which contributors to OSS were given free access to Copilot (Yeverechyahu, Mayya, and Oestreicher-Singer, 2024). 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 to explore how technology changes the nature of work, we provide one of the largest natural experiments of generative AI and it's impact on highly disaggregated measures of work processes "in the wild" over a two year time horizon.

Our main findings identify changes in the nature of work of AI adopters in their knowledge 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 developers. This is a sign that the workers likely will intensify their core contributions to public goods, such as open source software, when leveraging skill augmenting technology like generative AI. It is also consistent with reduced collaborative frictions during the problem solving process of work and a change in the way 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 the two years following 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, to the point where 96% of corporate codebases contain some open source code (Synopsys, 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 (Eghbal, 2020; Nagle et al., 2020b) as numerous firms free-ride on the efforts of others without giving back (Lifshitz-Assaf and Nagle, 2021) leaving volunteer developers 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 developers to more easily contribute to the common good by solving more issues. Prior research has shown that OSS developers 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. Section 1 develops a model of the impact of generative AI on individual workers leading to testable hypotheses. 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 obtain 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 also exploring the mechanisms at play, and offering empirical support for our hypotheses. We discuss the limitations, implications, and a back-of-the-envelope calculation to understand how the results are likely to generalize beyond our empirical setting in Section 6. Section 7 concludes.

1 Theoretical Framework

In the knowledge economy - which is an increasingly large sector of the overall economy -, highly productive individuals can often become victims of their own success. A common pattern relevant to our study occurs when a developer does exceptional core work, they are often assigned more managerial work as a result. For example, in the context of academia, where research and teaching are core work, the result of doing a good job on these is to get promoted and then to be given more managerial tasks including department and school committee assignments. This can be summed up by tweaking the well-known phrase "The reward for good work is more work." to be "The reward for good core work is more managerial work." This is particularly true in the context of public goods which, as public good projects become more successful and more widely used, new users request more from those that are creating the good.[3] Thus, the introduction of an AI tool that can help reduce some of this burden may play an important role in the creation of public goods.

In the following section, we develop the exposition of our empirical setting by using a simple economic framework where individual workers choose between two activities to maximize their utility: core work c and project management m. Let the worker's preferences $u_\theta(\cdot)$ be indexed by the parameter vector θ. In each period, each worker chooses c and m to solve the following static utility maximization problem:

$$\begin{aligned} \underset{c,m}{\text{maximize}} \quad & u_\theta(c,m) \\ \text{subject to} \quad & p_c c + p_m m \leq \omega \end{aligned} \tag{1}$$

where $c, m \geq 0$ and $p_c, p_m > 0$. The choice is constrained by relative costs of each activity, $p = (p_c, p_m)$, and units of an endowment resource, ω.[4] In line with simple economic models, we assume that preferences are time-invariant and that there are no externalities.

[3] In our empirical context of OSS, this "burden" of being an open source developer (Geerling, 2022) has been cited as significant driver of burnout and abandonment of open source development (Nagle et al., 2020a; Raman et al., 2020). Thus, alleviating this burden is of critical importance.

[4] In our setting, the resource endowment ω can be interpreted as the agent's "task bandwidth" they are able to allocate across various work activities.

To improve our understanding of the environment that the worker is in, we assume a constant-elasticity of substitution (CES) utility function

$$u_\theta(c, m) = \left(\beta_c^{1/\sigma} c^{\frac{\sigma-1}{\sigma}} + \beta_m^{1/\sigma} m^{\frac{\sigma-1}{\sigma}}\right)^{\frac{\sigma}{\sigma-1}} \tag{2}$$

where for $\theta = (\sigma, \beta_c, \beta_m)$, σ is the elasticity of substitution between c and m, and β_c, β_m are CES share parameters. Without loss of generality, after normalizing $p_m = 1$, p_c becomes the relative cost of doing core work. Under the optimal choice of these two activities, the Marshallian demands for core work and project management can be expressed as functions of these productivity, preference, and endowment parameters:

$$c^\star = \frac{\omega p_c^{-\sigma}}{p_c^{1-\sigma} + \frac{\beta_m}{\beta_c}} \tag{3}$$

$$m^\star = \frac{\omega}{\frac{\beta_c}{\beta_m} p_c^{1-\sigma} + 1} \tag{4}$$

Consistent with prior literature (Acemoglu, Kong, and Restrepo, 2024), we choose to model the intervention of generative AI as a reduction in the cost of core work, p_c.[5] As such, the comparative statics with respect to p_c are of interest. Details on the comparative statics for a change in p_c can be found in Appendix D. A consequence of the CES demand system is that a reduction in p_c increases the optimal level of core work under any value of the elasticity of substitution $\sigma > 0$. Further empirical support for this relationship comes from prior literature in the field. Beyond AI, automation and information systems technologies have been shown to complement skilled labor and lead to a reshaping of organizational practices that allows workers to engage in more complex and strategic activities (Autor, Levy, and Murnane, 2003; Orlikowski, 2007; Zammuto et al., 2007). Further, when technology reduces the cost or effort associated with certain tasks, economic and management theory suggests that workers will increase the amount of that task they perform (Acemoglu and Restrepo, 2018; Bloom et al., 2014). As such, we arrive at the following primary

[5]Technically, in a model-free world, it is not ex ante clear how individuals allocate efforts after gaining new time when they become more productive since they can allocate their time in whichever way they like.

hypothesis:

Hypothesis 1a (H1a) *After the adoption of an AI tool that assists with core work tasks, a worker's core work tasks increase as a percentage of all tasks.*

In contrast to the impact on core work tasks, the impact of generative AI on managerial tasks is less clear and dependent on the elasticity of substitution σ. Adoption of the tool may lead to no change in the share of project management when the elasticity of substitution $\sigma = 1$. Alternatively, project management may drop when the price of core work drops given a $\sigma > 1$ (project management is a substitute), or may increase when $0 < \sigma < 1$ (project management is a complement). This is consistent with prior literature that has shown that while automation and technology tend to reduce the burden of routine tasks, they do not necessarily eliminate managerial responsibilities, which may require human judgment, creativity, and interpersonal coordination (Autor, Levy, and Murnane, 2003; Mintzberg, 1994). Consequently, even as AI can reduce the time spent on routine tasks, workers may still engage in high-level decision-making and team leadership, leaving the net effect on managerial tasks uncertain and best determined empirically.

Hypothesis 1b (H1b) *After the adoption of an AI tool that assists with core work tasks, the change to a worker's managerial tasks as a percentage of all tasks is ambiguous.*

We next seek to better understand the mechanisms that are driving these effects. What is the effect of AI technology on task allocation across specific kinds of core work and project management? To this end, we extend the baseline 2-good CES model into a nested CES model, under which core work and project management are instead modeled as composites of more disaggregated goods.

$$u(c_1, c_2, m_1, m_2) = \left(\beta_c^{1/\sigma} u(c_1, c_2)^{\frac{\sigma-1}{\sigma}} + \beta_m^{1/\sigma} u(m_1, m_2)^{\frac{\sigma-1}{\sigma}}\right)^{\frac{\sigma}{\sigma-1}} \tag{5}$$

This allows us to further decompose composite goods into its components where $u(c_1, c_2)$ and $u(m_1, m_2)$ are also CES functions similar to equation 2 but with their respective within-nest elasticities of substitution σ_c and σ_m that correspond to relative substitution between disaggregated goods c_1, c_2 and m_1, m_2 respectively. Hence the nested CES extension to the baseline model permits both more refined definitions of work patterns and richer substitution patterns between these disaggregated goods. Details on the full nested CES model, as well as the comparative statics for a change in p_c can be found in Appendix D.

We use this model to consider two mechanisms through which the primary relationship operates. In the first mechanism, we consider whether workers engage in work that is more autonomous (less interaction with others working on the project) or more collaborative (more interaction with others working on the project). Individuals can engage in either autonomous core work, c_1 or collaborative core work, c_2 or the managerial equivalents, m_1 and m_2, We find that a reduction in the cost of core work through AI, p_c can increase the demand of core work (as in Hypothesis 1a, but it does not necessarily need to happen through both autonomous core work and collaborative core work simultaneously. Indeed, assuming that the elasticity of substitution $\sigma_c > 1$ and that the price of autonomous work is lower than the price of collaborative work, $\frac{p_{c_1}}{p_{c_2}} < 1$ implies that the worker will shift their efforts towards autonomous core work and away from collaborative core work since autonomous core work is less costly than collaborative core work. The same holds true for managerial work such that $\sigma_m > 1$ and $\frac{p_{m_1}}{p_{m_2}} < 1$. While there are reasons to find the alternative parameter spaces, $\frac{p_{c_1}}{p_{c_2}} > 1$ (and $\frac{p_{m_1}}{p_{m_2}} > 1$) are credible, we find this restricted parameter space with the pre-existing wedge of prices generally plausible in the context of workers that are already working in a highly collaborative setting like the increasingly common paradigm of distributed work. We hypothesize that their main issues — collaborative frictions such as the cost of coordination, requests from others to solve problems, or personal conflicts — may be more costly than solving problems by themselves when they have AI as a substitute available at any time.

The predictions of the nested model extension can similarly be derived from the literature. This mechanism builds on the idea that generative AI tools reduce (or even eliminate) much of

the cognitive and communicative friction inherent in distributed work, enabling workers to tackle complex tasks autonomously. Prior research has shown that technologies that streamline communication and decision-making processes reduce the overhead of collaboration, freeing workers to focus on their own work in isolation (Faraj, Jarvenpaa, and Majchrzak, 2011; Aral and Van Alstyne, 2011). However, with generative AI, many of these collaborative costs are simply eliminated as work that previously required communication between multiple people can now be done without any interaction at all. In the context of OSS, a quintessential example of distributed work and our empirical setting, research by Crowston et al. (2008) highlights the importance of collaboration and coordination in distributed work, but also points out that tools that reduce coordination costs (or circumvent the need for coordination altogether) can lead to a shift toward individual, autonomous contributions. Hence, we hypothesize:

Hypothesis 2 (H2) *A worker's change in task allocation resulting from the introduction of an AI tool is driven by an increased focus on autonomous tasks and a decrease in collaborative ones.*

The second mechanism we consider is whether workers that use AI alter their relative intensity of exploration versus exploitation in task allocation. When the cost of core work falls, workers may choose to increase their efforts in established projects or branch out into smaller, more nascent projects. In the nested CES framework, we can decompose both core work and managerial work into two components where c_1 and m_1 relate to experimentation with new competencies and projects (exploration) while c_2 and m_2 relate to engaging further in pre-existing competencies and projects (exploitation). The logic, implications, and parameter space are similar to those from Hypothesis 2 and are not repeated for brevity. To bound our predictions, we make the assumption that the cost of experimentation is smaller than the cost of exploitation, which likely holds true in the context of distributed work given the complexities and interdependencies that persist in existing projects versus those that are starting from scratch.

The distinction between exploration and exploitation is central to organizational learning and innovation theory, as first articulated by March (1991). Exploration involves searching for new

knowledge, competencies, and opportunities, while exploitation focuses on refining and optimizing existing capabilities. Prior research suggests that when the costs of experimentation decrease, individuals and organizations tend to shift their focus toward exploratory activities (Benner and Tushman, 2003; Levinthal and March, 1993). Further, research has shown that information technology investments, including digital tools, automate routine tasks and facilitate rapid feedback, and thereby promote experimentation and flexibility in task allocation (Bresnahan, Brynjolfsson, and Hitt, 2002; Zammuto et al., 2007). AI in particular has been shown to encourage "learning by doing," where individuals are more likely to engage in experimentation because AI tools provide real-time feedback and help them assess the feasibility of new ideas or projects (Ransbotham et al., 2017). While exploitation remains essential, the newfound ease of exploration and experimenting with new competencies and projects provided by AI tools makes the latter a more attractive and feasible focus for workers. As such, we hypothesize:

Hypothesis 3 (H3) *A worker's change in task allocation resulting from the introduction of an AI tool is driven by an increased focus on exploration activities and a decrease in exploitation.*

To better understand who benefits most from the introduction of generative AI, a small extension of the baseline model (CES utility as in Equation 2) introduces heterogeneity by allowing the response to a change in the relative cost of coding to vary by worker ability: $\sigma = \{\sigma^H, \sigma^L\}$. We assume that a low ability worker has a relatively higher elasticity of substitution between core work tasks and managerial tasks than a high ability worker: $\sigma^H < \sigma^L$.[6] This modelling choice can be motivated in different ways. On one hand, lower ability workers may stand to gain more from generative AI technology. In particular, for generative AI to function best, the data it is trained on must be of high-quality (Wladawsky-Berger, 2023). Indeed, for generative AI's that are context specific, the literature shows that input data filtered for higher quality leads to higher quality output (Chen et al., 2021). Thus, when using generative AI, lower ability workers are able to receive

[6]Heterogeneity could alternatively be introduced into this framework if for a common elasticity of substitution, generative AI reduces the cost of core work more for lower ability workers. Fortunately, the difference between these motivating assumptions is not consequential for our identification strategy.

a bigger benefit than their higher ability peers (Brynjolfsson, Li, and Raymond, 2023). Alternatively, variation in substitution by ability could arise if higher ability workers find core work and project management relatively more complementary. Conversely, lower-ability workers may view core and managerial tasks as substitutes rather than complements, as they may find it challenging to balance the demands of both. For these individuals, managerial tasks, which require multitasking, coordination, discretion, and interpersonal communication (Finkelstein and Hambrick, 1990; Hambrick and Finkelstein, 1987), can detract from their ability to focus on core work, thus making them substitutes for each other. In this sense, lower ability workers can be considered "specialists" while higher ability workers are more likely to be "generalists". Technological innovation has been shown to influence the composition of generalists and specialists in team production settings (Teodoridis, 2018). In the context of the model extension, this assumption implies that as the cost of core work drops, lower ability individuals will increase their proportion of activity that is core work more than higher ability individuals, leading to the following hypothesis:

Hypothesis 4a (H4a) *A worker's level of ability moderates the relationship between the adoption of an AI tool and task allocation such that lower ability workers will increase their core work tasks as a percentage of all tasks more than higher ability workers.*

Since the baseline effect of AI adoption on managerial tasks is ambiguous (Hypothesis 1b), predicting the moderating effect of ability on managerial work is less clear. However, using a similar reasoning to the discussion above, it is likely that the enhancement of the effect for lower ability workers found in Hypothesis 4a will also be at play in managerial work. Thus,

Hypothesis 4b (H4b) *A worker's level of ability moderates the relationship between the adoption of an AI tool and task allocation such that lower ability workers will have a larger effect on their managerial tasks as a percentage of all tasks more than higher ability workers.*

2 Institutional Background

To test the hypotheses constructed above, we must find a setting where distributed work is both common and where an individual's engagement in distinct work tasks can be observed with granularity. We find such a setting in the case of open source software, a quintessential example of distributed work. Furthermore, to give a causal interpretation of any recovered effects, we need a plausibly exogenous introduction of an AI tool that assists with core work. In particular, we examine the GitHub platform, where the bulk of OSS activity takes place, and their roll-out of the generative AI software development tool Copilot.

2.1 The GitHub Platform

GitHub is the world's largest hub for OSS development.[7] Launched in 2008, 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.[8] Alternatively, a user may "fork" another repository, which entails copying every-

[7] 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 include GitLab, Bitbucket, Codeberg, Gitee, SourceForge, SrcHut, to name a few, though all have much smaller market shares than GitHub.

[8] 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 hosts the repository and adds important social and project management features for users. Among these features are 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.

thing from that repository into a new repository so it has the exact same information, but allows the copier 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 without the authority to commit directly to a codebase can 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 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 is based on predictive models similar to those that underlie ChatGPT.[9] 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 generate code snippets for work while they are coding that can be easily integrated into the codebase they are working on. 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

[9]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.

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.[10] 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 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 developers 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 developers receive complimentary Copilot access which we leverage as a natural experiment. To determine a user's eligiblity for the program, GitHub creates an internal ranking for OSS repositories based on criteria that remains unknown to the wider public. Any developer who has been added as a "collaborator" to a repository is considered a maintainer and is therefore potentially eligible for free access to GitHub Copilot.[11] Developers for projects below a given threshold of the eligibility ranking are granted complimentary Copilot access for one year. After one year, GitHub verifies the user's current eligibility through the top developer program and complimentary access continues if the developer's repository remains below the threshold.

[10] Start date of general access announcement: GitHub blog post.

[11] The nominal owner of the repository, i.e. the GitHub user who created the repository, can add collaborators in the repository's settings page.

A key feature of this program is that the exact ranking system is largely unknown to developers. Public discussions between users suggests there exists a considerable degree of uncertainty over what makes a given project eligible.[12] As GitHub does not reveal the exact composition and ingredients of the ranking, nor the eligibility threshold the developers 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 developer. Moreover, GitHub does not engage in any additional messaging to communicate the eligibility status to developers. Each developer has to check whether they are eligible to access the AI for free when they apply for Copilot.

3 Data

We use a mixture of openly available and proprietary data from GitHub to understand the effects of the Copilot AI program for developers. Developer activity is publicly available while Copilot AI usage and eligibility for complimentary access are 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 developer-week observations. From the granular activities data observed on the GitHub platform, we develop a set of two broad classifications of developer activities that are essential to collaborative software development. The core of our analysis considers the extent to which developers 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 developer 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 developers to integrate proposed changes. These activities characterize the more technical process of writing lines of software code. Second, all

[12] See YCombinator and GitHub discussions.

remaining activities are considered 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 developers engage with the wider community to assist with software issues, introduce new ideas, and discuss longer term project objectives.[13] Other activities related to project management include the creation of project boards, which assist developers 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 developer on the platform.

Table A1 shows univariate statistics for the Copilot AI treatment, the work activities and the ranking for the first year.[14] We use a sample of active top developers 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 developers who received access for the first time, we exclude any developers who had Copilot experience during the technical preview. All individuals in our top developer 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 developers" — those who are maintainers of OSS projects that are among 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 developers 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.[15] For coding and project management, we create a cumulative share measure defined as the cumulative sum of the activity over the developers overall

[13] For example, developers can open issues themselves, assign other developers to investigate certain issues, give issues labels for organization, and close the issue itself, thereby signaling that it is no longer an outstanding concern.

[14] Descriptive statistics for the subset of top developers within the ranking bandwidth of $h \in [-100, 100]$ can be found in Table A3. More granular statistics on work items are displayed in Table A4.

[15] AI usage is measured directly when a developer uses GitHub Copilot as a plugin within their integrated developer interface by observing the developer's transactions with the Copilot API.

total cumulative activity.[16] By conditioning on overall individual activity levels, we are better able to directly compare work allocation across developers and identify any potential reallocation in response to AI adoption. The average developer allocates 44% of their engagement towards coding and, 37% towards project management.[17] Finally, our measure for our natural experiment is the best (minimum) normalized ranking across eligible repositories that developers 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 developer rank around -364 implying that a substantial amount of variation can be found below the normalized threshold of zero.

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 workers 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 developer" program. To credibly establish a causal effect of Copilot on OSS activity, we leverage a natural experiment. GitHub awarded free Copilot access to developers of the most popular public repositories according to an internal ranking. The top developer 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.

[16] Specifically, we use the following cumulative share version: $\frac{\sum_{s \leq t} Activity_{it}}{\sum_{s \leq t} TotalActivities_{it}}$. While the theoretical framework uses absolute measures, we operationalize it via shares due to (1) the sparsity of the measures, (2) to allow for comparisons of different magnitudes that can be substantially different, and (3) to focus attention on relative task allocation.

[17] Table A2 provides the same statistics for a two year time period. AI usage increases substantially across all statistics.

Since the ranking occurs at the project level, we assign each developer a weekly rank from the projects for which they are designated as collaborators. Specifically, the rank assigned to each developer is the running best (i.e. cumulative minimum) rank received across all repositories through the current observation period. Developers who receive a ranking below 0 are eligible for free access to Copilot while others are not. Developers only become aware of their eligibility status when applying for Copilot access. Since the developers 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 developer 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 it's assumptions. For example, it is important to have no manipulation across the ranking for the RDD to be valid.[18] 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.[19]

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 de-

[18] In other words, neither the developer can manipulate the ranking to receive Copilot access nor should GitHub be able to manipulate the ranking to the benefit of some developers.

[19] 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

veloper. $Eligible_{it}$ 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 developer 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 β_1 shows the causal effect of crossing the threshold — and through it, increased Copilot access — on the work activity of interest.

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. 16 percent of developers used Copilot at least once (see Table A1). Further, GitHub was not engaging in any kind of advertising informing the top developers about their eligibility for free Copilot access. Top developers had to visit the GitHub Copilot website and check for 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. Developers that are below the threshold of zero (on the left) 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 developers with free access are more likely to use the generative AI.[20]

— Table 1 about here —

Table 1 shows the exact coefficients on generative AI adoption of the top developer population across those who are eligible to access Copilot for a price of zero dollars (i.e. ranking below zero) and those who are ineligible to access Copilot through the top developer channel (i.e. ranking above zero). We try a number of alternative measures for Copilot takeup. The first two columns show estimates from the cross-section and the last two columns show estimates from the panel. Across each alternative measure, we find an increase of AI adoption for eligible developers near the rank threshold. For the cross-section, we observe an increase of 6.90 days (321% relative to the baseline) overall in AI adoption and an increase of 6.14 percentage points (61.2% relative to the baseline) of developers 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 developer 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 Generative AI induces a reallocation towards core work

Having established that GitHub's top developer program increases Copilot usage for eligible users, we next begin to explore the causal impact of access to generative AI has on patterns of distributed work. First, we use the identification strategy established in Section 4 to empirically evaluate Hypotheses 1a and 1b.

[20]The first stage is robust to a polynomial of degree two (see Figure C2)

— 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 as a percentage of all activity increase by 5.4 percentage points (12.37% relative to the baseline) while project management as a percentage of all activity 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 about here —

Figure 3 shows the same effects graphically. 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 developer's relative project management intensity. It implies that developers 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 developers have to solve fewer issues and they can focus on their core work - writing software code - which is what surveys have revealed is how they prefer to spend their time when working on OSS (Nagle et al., 2020a). Hence, these findings support the prediction of Hypothesis 1a and provide evidence on Hypothesis 1b that core work and project management are gross substitutes ($\sigma > 1$).

We establish robustness for these results in a number of different ways. First, the results are robust to different bandwidth selections and functional forms as shown in Figure C4. Second, the results hold under different kernel selections, as shown in Table C2. Third, they are also robust under optimally chosen bandwidths suggested by Calonico, Cattaneo, and Farrell (2020), which we present in Table C3. [21] Finally, we show in Table A6 that the reallocation of core work under Copilot also holds when estimating the intent-to-treat effects using differences-in-differences and matching identification strategies. In a next step, we establish the generalizability of our results.

[21] The results are further robust to using the absolute measures (see Table C4) and we find that the residual is moving in the expected direction (see Table C5). We additionally find that the effect for the compliers is qualitatively the same as the ITT approach but with substantially higher scaled up coefficients in absolute terms.

We estimate the core results for the subset of developers who are "firm-affiliated" in that their author commits use a company issued-email address and present the results in Table A7. The effects of Copilot are qualitatively similar for these two sub-populations, albeit with some minor differences, suggesting that our core results have some external validity and likely hold within firms.

— 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 two years albeit with some ramp up and attenuation. In the first quarter, the effects are slightly weaker (in absolute terms) for the effect of Copilot on coding and project management relative to the peak of the third quarter where effects increases up to 10 percent and 27 percent in absolute terms, respectively. The impacts of the generative coding tool Copilot continuously attenuates 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 developers is re-evaluated after year one. Generally speaking, the pattern indicates that the benefits of accessing Copilot seem to arise very quickly and after some experimentation with it, the impacts are stable up to approximately two years.

5.3 Generative AI enables more autonomous work

We next consider Hypothesis 2 to address the question of whether generative AI induces workers to work more collaboratively or autonomously. As a start, we can decompose our measures of coding and project management into more granular activity components, which can in turn be classified according to the extent to which they are more collaborative or autonomous in nature.

— Figure 6 about here —

Figure 6 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 developer 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 developer. Together, these coefficient estimates indicate that generative AI enables developers to bypass collaboration frictions and more easily make unilateral code contributions to projects. 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 6 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. Developers 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 interactions which is in line with a lower rate of requests for reviewers to other developers or the assignment of issues to others. They also have to review fewer pull requests and are subscribed to fewer issues which is indicative of those issues having been closed at a faster rate. Overall, the effect of Copilot on each component of the project management measure is consistent with workers 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.[22]

A more direct way of measuring developer engagement in autonomous work is by assessing how Copilot access influences the size of peer groups a developer opts to work with. We construct a cumulative mean of the distinct number of collaborators the developer interacts with across all public repositories over time. We can further disaggregate this measure by granular activity types. Figure A3 contains intent-to-treat coefficient estimates for the effect of Copilot access on a measure

[22] Another robust channel that would strengthen the project management effect if included is "Commenting". Developers with AI access seem to comment substantially less than those without (result upon request).

of distinct collaborators based on each specific activity. We can see that for nearly all activities, in both core work and project management, that Copilot eligible developers refocus their engagement towards smaller communities. One can interpret this as a switch towards more autonomous and less collaborative distributed work patterns.

We also consider a version of the distinct collaborators measure that is aggregated by taking the maximum number of distinct collaborators across all activity types. The coefficient estimates for this measure are presented in Table A8 and suggest that Copilot eligibility induces developers to reduce the number of peers they collaborate with dramatically: eligible developers work in repositories with 17 fewer peers relative to a baseline of 22 (79.3% lower). This drop off is significant and suggests that developers with Copilot access are substituting work in larger repositories for work in smaller projects. In aggregate, all of this evidence offers support for Hypothesis 2.

5.4 Generative AI encourages experimentation

We are further interested in whether workers are more likely to continue working in established projects ("exploit") or branch out into more experimental work ("explore") when they are provided with free access to coding generative AI. Hypothesis 3 predicts they will engage in more experimentation as the generative AI allows them to explore new areas more.

— Table 3 about here —

Table 3 contains intent-to-treat coefficients for free Copilot eligibility on several measures of relative exploration or experimentation and exploitation. In Panel A, we investigate how Copilot influences (1) the developer's cumulative exposure to repositories that they have never engaged with before (exploration) and (2) the extent to which the developer revisits repositories they've previously interacted with (exploitation). The estimates are consistent with Hypothesis 3: generative AI encourages exploration and diminishes exploitation. To put these effects into context, Copliot eligible developers engage with an additional 15 new repositories on average relative to ineligible peers. Beyond simply interacting with a new set of repositories, we also find evidence

that generative AI enables developers to gain exposure to a wider range of technologies.

In Panel B, we can see that eligible developers increase their cumulative exposure to new programming languages by 21.79% relative to the baseline. We also estimate a version of this cumulative programming language exposure measure weighted by the median salary reported by software developers who use that language.[23] Access to Copilot induces developers to experiment with programming languages that command a 1.41% higher salary relative to a baseline of $119,371 (an increase of $1,683). Together, these estimates indicate that Copilot eligible developers are both exploring new languages and choosing languages with greater labor market return. Overall, these results suggests that Copilot eligibility increases exploratory and experimental work activities undertaken by project developers across several dimensions, while it reduces tendencies to exploit established projects, and establish credible evidence for Hypothesis 3.

5.5 Generative AI responses are stronger for lower ability workers

Finally, we want to understand the extent to which a worker's ability level can moderate the impact of generative AI adoption. By further disaggregating our sample of developers, we can assess the potential for AI tools to reduce inequality in a setting that is overly dependent on a small number of highly skilled individuals (Hoffmann, Nagle, and Zhou, 2024) and explore the implications for improving the health of critical digital infrastructure (Lifshitz-Assaf and Nagle, 2021).

— Figure 5 about here —

Figure 5 shows the AI treatment effects across median splits for a number of alternative proxies for ability.[24] Our set of ability proxies include the developer's maximum centrality across ranked repositories, the number of GitHub platform achievements, their follower count, and their tenure[25]

[23] We use reported median salaries by programming languages from the Stack Overflow (2023) Developer Survey.

[24] Table 4 contains the estimates plotted in Figure 5 as well as relative treatment effect and sample size comparisons.

[25] GitHub achievements are badges that appear on a GitHub user's public profile upon completion of some kind of event, such as successfully merging their first PR, creating a popular repository, or donating to a repository via GitHub Sponsors. See https://githubachievements.com/ for more details. Tenure, in particular, has been used as proxy for ability in similar settings (Bonabi et al., 2024).

on GitHub. Summary statistics for these proxies and a description of how they are measured are contained in Table A5. This diverse set of proxies was chosen to capture developer characteristics such as project workload incidence, diversity of contribution, popular interest from peers, and experience. For each of these measures, we find that individuals below the median (low ability) exhibit larger increases of coding activities and larger reductions in project management activities, as a percentage of all activities, than those above the median (high ability). It indicates that the low ability workers benefit more from the Copilot programming AI and therefore AI has the potential to reduce inequality of contributions for OSS.

6 Discussion

This research has a number of implications related to the impact of generative AI for working individuals, managers, and society more broadly. The main findings of this paper imply that hands-on managers may be aided by generative artificial intelligence and that this novel technology has the potential to flatten organizational hierarchies. It not only shows that generative AI may allow talented workers to refocus their attention to core work that is more aligned with their preferences but also that this shift could lead to more exploratory work and innovation. While we have not shown it directly in this paper, one may speculate that managers who no longer code after rising up in the hierarchy may be able to now more easily get back into the core work, which could allow them to connect more directly with their team.

Our work has further implications related to technology adoption for distributed organizations, as generative AI may bring about a more streamlined production process. However, we argue that this study goes beyond distributed organizations and also speaks to technology within non-distributed organizations and worker decisions within the information economy more broadly. Finally, our salary-weighted language exposure measure allows for a back-of-the-envelope calculation of the monetary impact of Copilot if one extends it to the set of all top developers. The language-specific labor market potential is around $1,683 per year. When developers pay a price

for Copilot of $120 per year, we arrive at an average net potential of $1,563. Extrapolating this average effect to the set of all 300,000 developers suggests that GitHub Copilot can improve the aggregate value of labor market potential by roughly $468 million per year. While this measure is purely a back-of-the-envelope without taking into account general equilibrium effects, we believe that this estimate substantially underestimates the true value since it does not account for any value derived from improvements that are not due to language-specific experimentation or due to the productivity benefits of generative AI.

Our empirical setting allows us to push the boundaries of existing knowledge about 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 writing and we are not precisely observing 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 by the individuals. 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% of the activity of highly active programmers happens on private and 50% 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 developers 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 it's 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 work processes of adopters. We find that top developers of open source software are engaging more in their core work of coding and are engaging less in their non-core work of project management. Both of these main effects are driven by two underlying mechanisms — an increase in autonomous behavior (and a related decrease in collaborative behavior) and an increase in exploration behavior (and a related decrease in exploitation behavior). In particular, the reduction of the need to collaborate with other humans, leads to humans circumventing collaborative frictions and transaction costs that would otherwise occur during their work. We further find that the programming generative AI Copilot shifts the task allocation of developers with lower ability more than those with higher ability.

Overall, our results are among the first to illuminate the deeper level changes in a decentralized work process instigated by AI over a long time period with very granular level information using a natural experiment. Furthermore, our study yields early insights 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 OSS production process can generate far-reaching spillover benefits to downstream users. We believe our study 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.

Acemoglu, Daron, Fredric Kong, and Pascual Restrepo. 2024. "Tasks At Work: Comparative Advantage, Technology and Labor Demand." Tech. rep., National Bureau of Economic Research.

Acemoglu, Daron and Pascual Restrepo. 2018. "Artificial intelligence, automation, and work." In *The economics of artificial intelligence: An agenda*. University of Chicago Press, 197–236.

Agrawal, Ajay, Joshua Gans, and Avi Goldfarb. 2019. "Economic policy for artificial intelligence." *Innovation policy and the economy* 19 (1):139–159.

Altman, Elizabeth J, Frank Nagle, and Michael Tushman. 2015. *Innovating without information constraints: Organizations, communities, and innovation when information costs approach zero*. Oxford University Press New York.

Aral, Sinan and Marshall Van Alstyne. 2011. "The diversity-bandwidth trade-off." *American Journal of Sociology* 117 (1):90–171.

Autor, David. 2024. "Applying AI to rebuild middle class jobs." Tech. rep., National Bureau of Economic Research.

Autor, David H, Frank Levy, and Richard J Murnane. 2003. "The skill content of recent technological change: An empirical exploration." *The Quarterly Journal of Economics* 118 (4):1279–1333.

Ballester, Coralio, Antoni Calvó-Armengol, and Yves Zenou. 2006. "Who's who in networks. Wanted: The key player." *Econometrica* 74 (5):1403–1417.

Benner, Mary J and Michael L Tushman. 2003. "Exploitation, exploration, and process management: The productivity dilemma revisited." *Academy of Management Review* 28 (2):238–256.

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.

Bloom, Nicholas, Luis Garicano, Raffaella Sadun, and John Van Reenen. 2014. "The distinct effects of information technology and communication technology on firm organization." *Management Science* 60 (12):2859–2885.

Bonabi, Sardar Fatooreh, Sarah Bana, Vijay Gurbaxani, and Tingting Nian. 2024. "Navigating the Generative AI Blackout: The Role of Generative AI in Software Development Industry." Working paper.

Boysel, Sam, Manuel Hoffmann, and Frank Nagle. 2024. "Labor Competition and Open Innovation." *Working Paper* .

Bresnahan, Timothy F, Erik Brynjolfsson, and Lorin M Hitt. 2002. "Information technology, workplace organization, and the demand for skilled labor: Firm-level evidence." *The Quarterly Journal of Economics* 117 (1):339–376.

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-ofartificial-intelligence* 30 (10).

Calonico, Sebastian, Matias D Cattaneo, and Max H Farrell. 2020. "Optimal bandwidth choice for robust bias-corrected inference in regression discontinuity designs." *The Econometrics Journal* 23 (2):192–210.

Chen, Mark, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman et al. 2021. "Evaluating large language models trained on code." *arXiv preprint arXiv:2107.03374* .

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).

Crowston, Kevin, Kangning Wei, James Howison, and Andrea Wiggins. 2008. "Free/Libre open-source software development: What we know and what we do not know." *ACM Computing Surveys (CSUR)* 44 (2):1–35.

Cui, Zheyuan Kevin, Mert Demirer, Sonia Jaffe, Leon Musolff, Sida Peng, and Tobias Salz. 2024. "The Effects of Generative AI on High Skilled Work: Evidence from Three Field Experiments with Software Developers." *Available at SSRN* .

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 Krayer, François Candelon, and Karim R Lakhani. 2023. "Navigating the jagged technological frontier: Field experimental evidence of the effects of AI on knowledge worker productivity and quality." *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.

Faraj, Samer, Sirkka L Jarvenpaa, and Ann Majchrzak. 2011. "Knowledge collaboration in online communities." *Organization Science* 22 (5):1224–1239.

Finkelstein, Sydney and Donald C Hambrick. 1990. "Top-management-team tenure and organizational outcomes: The moderating role of managerial discretion." *Administrative science quarterly* :484–503.

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.

Geerling, Jeff. 2022. "The burden of an Open Source maintainer." URL https://www.jeffgeerling.com/blog/2022/burden-open-source-maintainer.

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.

Hambrick, DC and S Finkelstein. 1987. "Managerial discretion: A bridge between polar views of organizational outcomes." *Research in Organizational Behavior* 9:369–406.

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.

Levinthal, Daniel A and James G March. 1993. "The myopia of learning." *Strategic Management Journal* 14 (S2):95–112.

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.

March, James G. 1991. "Exploration and exploitation in organizational learning." *Organization Science* 2 (1):71–87.

McCrary, Justin. 2008. "Manipulation of the running variable in the regression discontinuity design: A density test." *Journal of econometrics* 142 (2):698–714.

Mintzberg, Henry. 1994. "Rounding out the manager's job." *Sloan Management Review* 36:11–11.

Moon, Jae Yun and Lee Sproull. 2002. "Essence of Distributed Work: The Case of the Linux Kernel." In *Distributed Work*. The MIT Press, 21. 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* .

Orlikowski, Wanda J. 2007. "Sociomaterial practices: Exploring technology at work." *Organization Studies* 28 (9):1435–1448.

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.

Ransbotham, Sam, David Kiron, Philipp Gerbert, and Martin Reeves. 2017. "Reshaping business with artificial intelligence: Closing the gap between ambition and action." *MIT sloan management review* 59 (1).

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* .

Stack Overflow. 2023. "Stack Overflow Developer Survey 2023." URL https://survey.stackoverflow.co/2023.

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-Ondrejkovic, and R Sadun. 2023. "Reskilling in the age of AI." *Harvard Business Review* 21.

Teodoridis, Florenta. 2018. "Understanding team knowledge production: The interrelated roles of technology and expertise." *Management Science* 64 (8):3625–3648.

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.

Wladawsky-Berger, Irving. 2023. "Why Human Input Matters to Generative AI." *MIT Initative on the Digital Economy* .

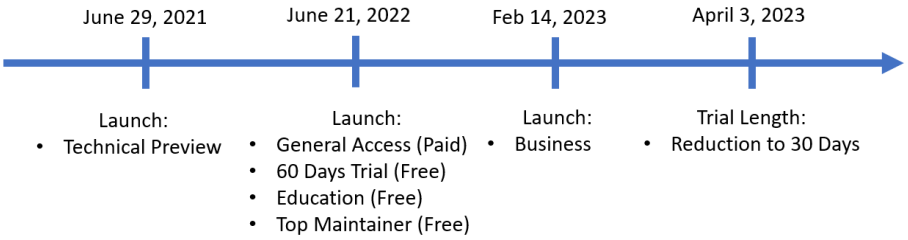
Wright, Nataliya Langburd, Frank Nagle, and Shane Greenstein. 2023. "Open source software and global entrepreneurship." *Research Policy* 52 (9):104846.

Yeverechyahu, Doron, Raveesh Mayya, and Gal Oestreicher-Singer. 2024. "The Impact of Large Language Models on Open-source Innovation: Evidence from GitHub Copilot." *arXiv preprint arXiv:2409.08379* .

Zammuto, Raymond F, Terri L Griffith, Ann Majchrzak, Deborah J Dougherty, and Samer Faraj. 2007. "Information technology and the changing fabric of organization." *Organization Science* 18 (5):749–762.

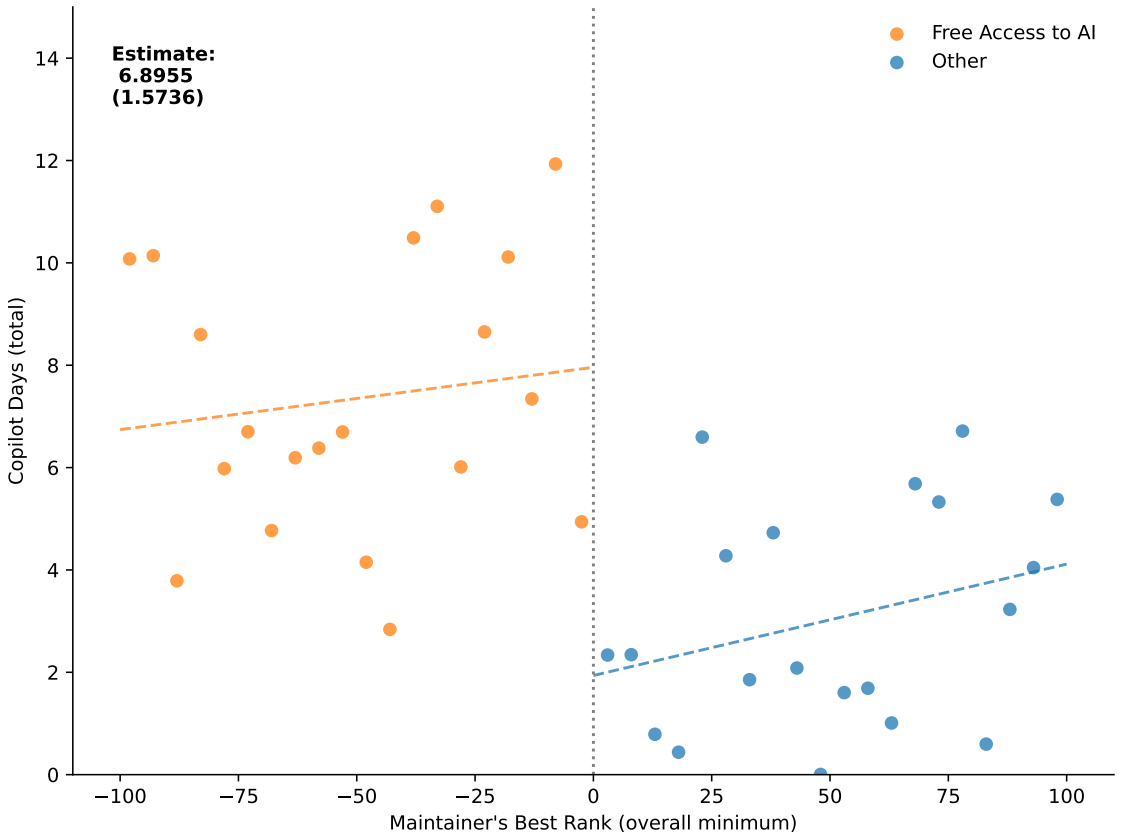
Figures and Tables

Figure 1: GITHUB COPILOT AI DEPLOYMENT TIMELINE



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

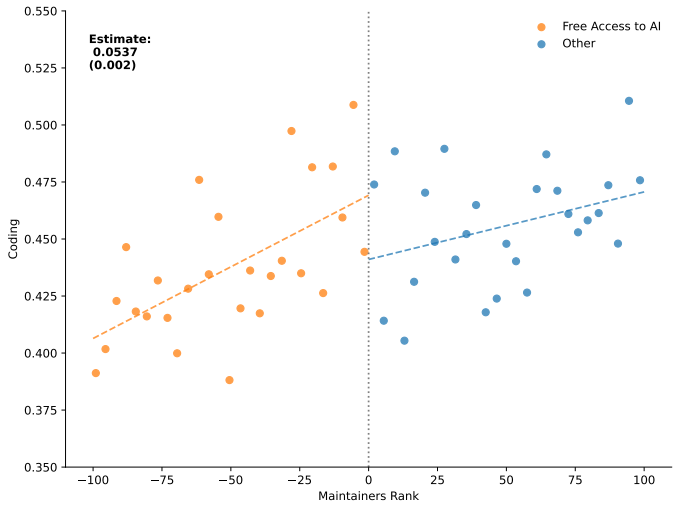
Figure 2: COPILOT AI ADOPTION ACROSS RANKS



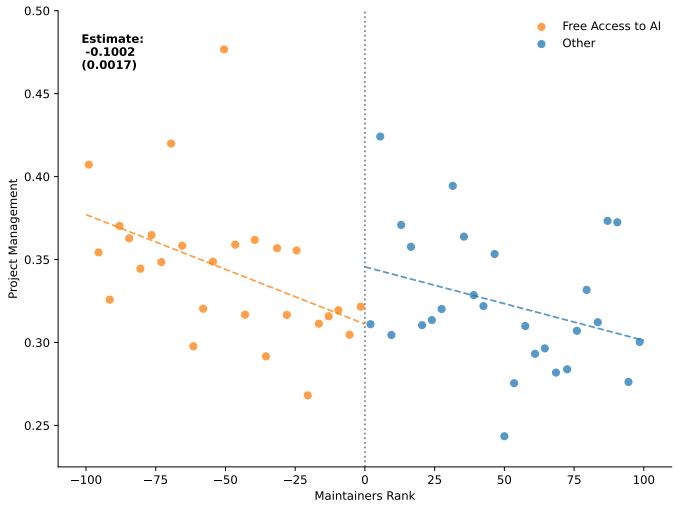
Note: The figure shows the total number of days the Copilot generative AI was used across the overall normalized minimum ranking on GitHub using a linear fit on either side of the threshold within the bandwidth of $h \in [-100, 100]$. Developers with rankings below zero receive free access to the AI through the top developer 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

Coding



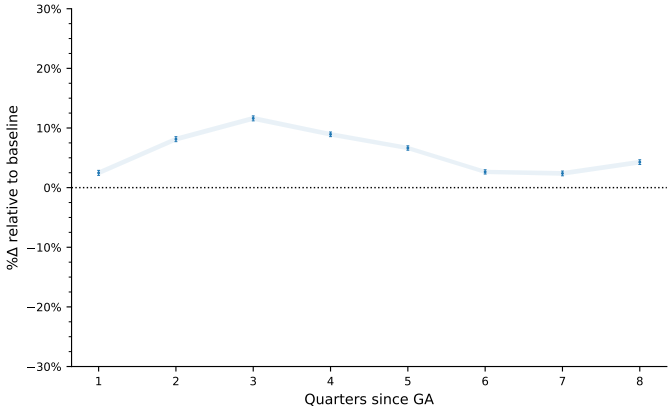
Project Management



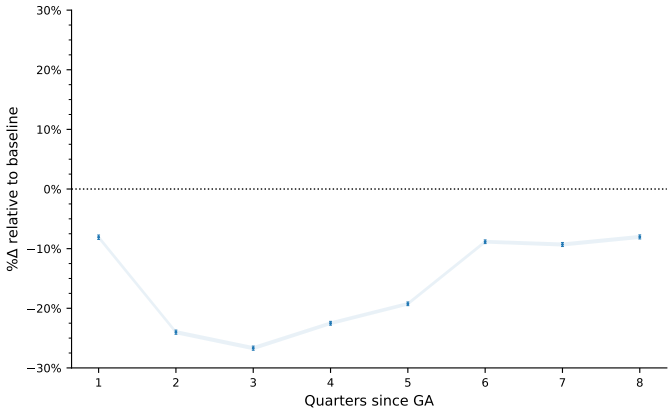
Note: The figure shows the intent-to-treat (ITT) effect of the GitHub Copilot generative AI on the outcomes of Coding and Project Management using the overall normalized minimum ranking on GitHub via a linear fit on either side of the threshold within the bandwidth of $h \in [-100, 100]$. Developers with rankings below zero receive free access to the AI through the top developer 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



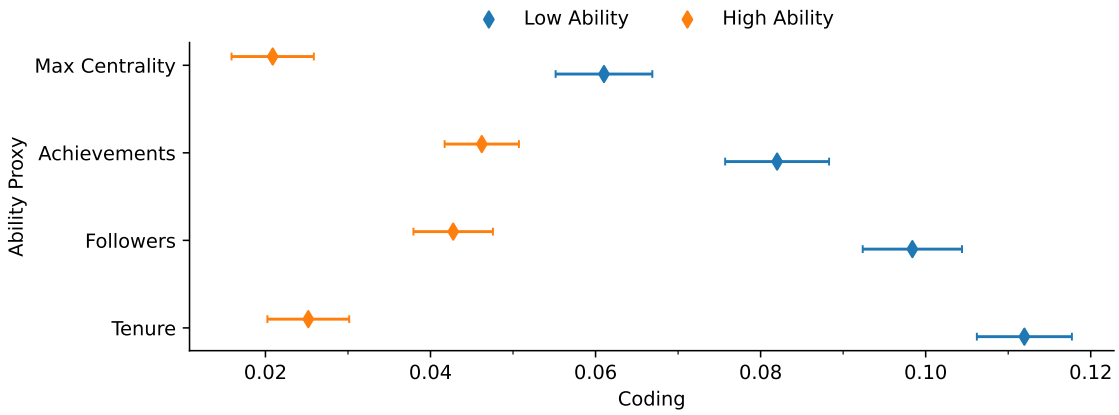
Panel B: Project Management



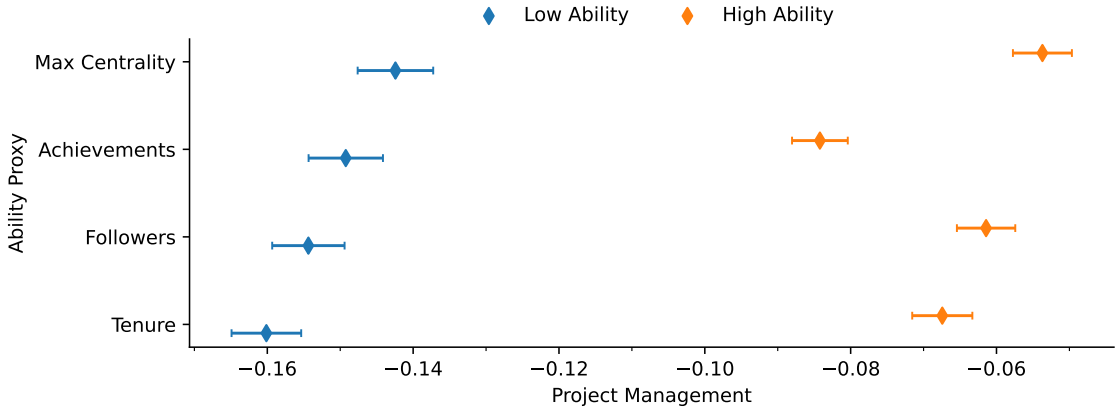
Note: The figures plots the relative treatment effects (in percentage terms relative to the baseline) derived from the intent-to-treat (ITT) coefficient estimates and confidence intervals for top developers being ranked below zero relative to above a zero ranking on the outcomes of coding and project 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



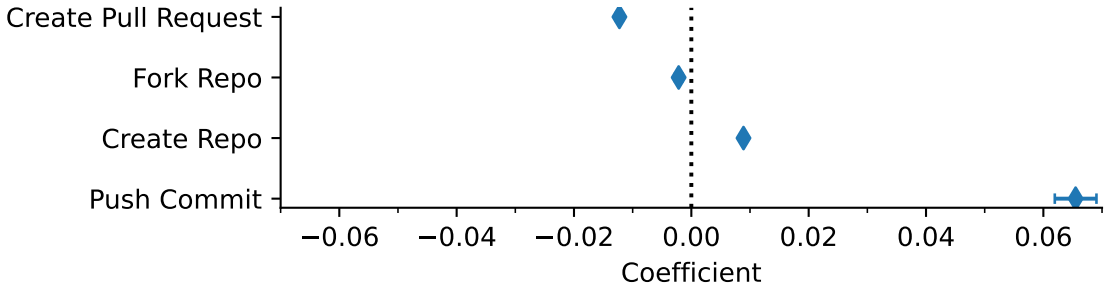
Panel B: Project Management



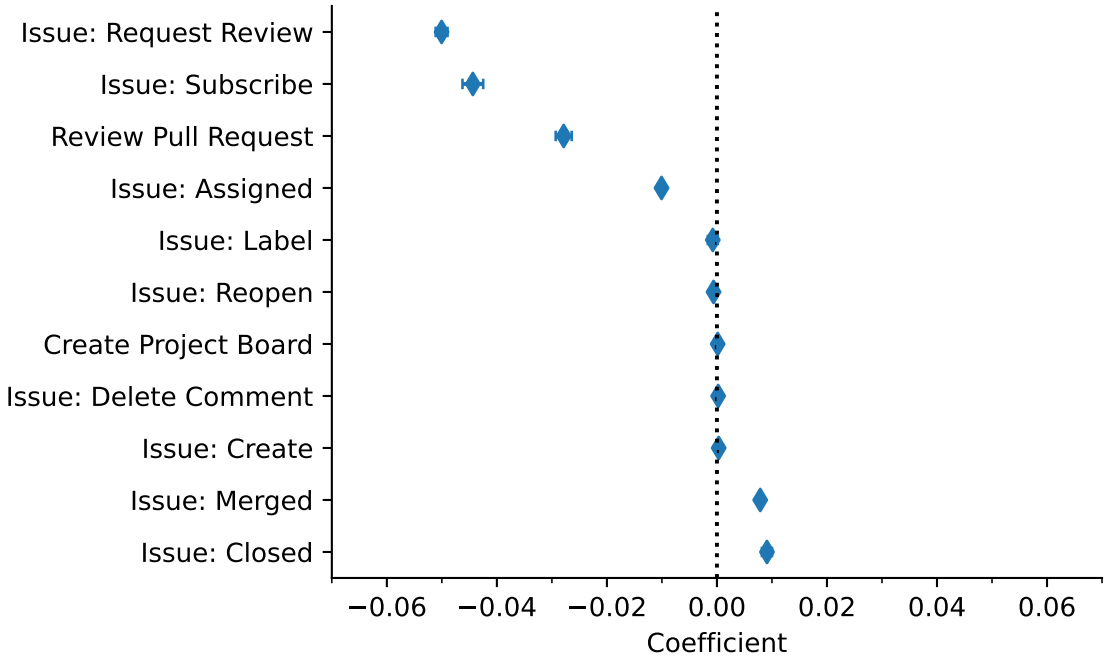
Note: The figures show intent-to-treat (ITT) coefficient estimates and confidence intervals on the outcomes of coding and project management for top developers being ranked below zero relative to above a zero ranking across low and high ability developers. Ability is measured based on ability proxies of developer centrality, follower count, achievements, and tenure split by the median up to the 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.

Figure 6: 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. Horizontal error bars represent 95% condfidence intervals for the coefficient estimate and are based on robust standard errors.

Table 1: AI ADOPTION ACROSS DEVELOPER RANKINGS

	Cross Section		Panel	
	Total Days	Ever Use	Days/Week	Exposure
$\mathbb{1}(Eligible)$	6.896^{***}	0.061^{**}	0.125^{***}	0.018^{***}
	(1.574)	(0.023)	(0.008)	(0.001)
Baseline	2.146^{***}	0.100^{***}	0.056^{***}	0.008^{***}
	(0.593)	(0.013)	(0.003)	(0.000)
Rel. TE (%)	321.3	61.0	223.2	214.3
F	18.0	17.8	435.7	557.5
N	4,268	4,268	215,169	215,169

[a] *Note:* This table shows the first stage uptake of the generative AI tool Copilot for top developers 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 developer 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

	Coding		Project Management	
$\mathbb{1}(Eligible)$	0.054^{***}	0.041^{***}	-0.100^{***}	-0.089^{***}
	(0.002)	(0.002)	(0.002)	(0.002)
Baseline	0.434^{***}	0.457^{***}	0.402^{***}	0.443^{***}
	(0.001)	(0.001)	(0.001)	(0.001)
Rel. TE (%)	12.4	9.0	-24.9	-20.1
N	269,546	248,032	269,546	248,032
Controls		✓		✓

Note: This table shows the coefficient estimates from the intent-to-treat specification across each activity classification for top developers 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 developer's GitHub account age in days, 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: $^{***}\,p < 0.001$; $^{**}\,p < 0.01$; $^{*}\,p < 0.05$

Table 3: INTENT-TO-TREAT EFFECT OF COPILOT ON EXPERIMENTATION

Panel A:

	New Repositories		Old Repositories	
$\mathbb{1}(Eligible)$	0.484^{***}	0.367^{***}	-0.087^{***}	-0.114^{***}
	(0.010)	(0.010)	(0.011)	(0.011)
Baseline	3.097^{***}	2.272^{***}	4.289^{***}	3.645^{***}
	(0.007)	(0.007)	(0.007)	(0.008)
TE (%)	62.3	44.3	−9.1	−12.0
N	269,546	248,032	269,546	248,032
Controls		✓		✓

Panel B:

	Language Exposure		Salary Exposure	
$\mathbb{1}(Eligible)$	1.753^{***}	1.152^{***}	0.014^{***}	0.013^{***}
	(0.075)	(0.076)	(0.000)	(0.000)
Baseline	9.245^{***}	5.127^{***}	11.691^{***}	11.685^{***}
	(0.047)	(0.054)	(0.000)	(0.000)
TE (%)	19.0	22.5	1.4	1.3
N	181,798	170,411	181,798	170,411
Controls		✓		✓

[a] *Note:* This table shows the coefficient estimates from the intent-to-treat specification for several measures of experimentation for top developers within the bandwidth of ranks $h \in [-100, 100]$. Panel A compares the effect of Copilot AI on a measure of exploration, the developers (log) cumulative exposure to new repositories, against its effect on a measure of exploitation, the developer's (log) cumulative count of repositories that they have previously interacted with. In Panel B, we present the ITT estimates for two additional measures of developer experimentation, the cumulative number of distinct programming languages the developer has been exposed to and an alternative version weighted by the salary language practitioners are typically paid. Salary exposure is a version of the language exposure measure where each distinct language is weighted by the median reported salary reported by software engineers who use the language (Stack Overflow Developer Survey 2023). We take the natural log of the resulting quantity plus one. Since the specifications for New Repositories, Old Repositories, and Salary Exposure are log-linear, we interpret the coefficients directly to derive a treatment effect in percentage terms. For Language Exposure, we present the treatment effect relative to the baseline. Robust standard errors are in parentheses: $^{***}\, p < 0.001$; $^{**}\, p < 0.01$; $^{*}\, p < 0.05$

Table 4: INTENT-TO-TREAT EFFECTS OF COPILOT ON WORK ACTIVITIES BY ABILITY

Panel A:

Coding	Centrality		Achievements		Followers		Tenure	
	Low	High	Low	High	Low	High	Low	High
$\mathbb{1}(Eligible)$	0.061^{***}	0.021^{***}	0.082^{***}	0.046^{***}	0.098^{***}	0.043^{***}	0.112^{***}	0.025^{***}
	(0.003)	(0.003)	(0.003)	(0.002)	(0.003)	(0.002)	(0.003)	(0.003)
Baseline	0.431^{***}	0.443^{***}	0.427^{***}	0.438^{***}	0.422^{***}	0.435^{***}	0.421^{***}	0.443^{***}
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
Rel. TE (%)	14.2	4.7	19.2	10.6	23.3	9.8	26.6	5.7
N	124,077	123,977	147,814	121,732	136,011	133,535	134,999	134,496

Panel B:

Project Management	Centrality		Achievements		Followers		Tenure	
	Low	High	Low	High	Low	High	Low	High
$\mathbb{1}(Eligible)$	-0.142^{***}	-0.054^{***}	-0.149^{***}	-0.084^{***}	-0.154^{***}	-0.061^{***}	-0.160^{***}	-0.067^{***}
	(0.003)	(0.002)	(0.003)	(0.002)	(0.003)	(0.002)	(0.002)	(0.002)
Baseline	0.432^{***}	0.348^{***}	0.417^{***}	0.375^{***}	0.448^{***}	0.336^{***}	0.434^{***}	0.355^{***}
	(0.002)	(0.001)	(0.002)	(0.001)	(0.002)	(0.001)	(0.002)	(0.001)
Rel. TE (%)	-33.0	-15.4	-35.8	-22.5	-34.5	-18.3	-36.9	-19.0
N	124,077	123,977	147,814	121,732	136,011	133,535	134,999	134,496

Note: This table shows the coefficient estimates from the intent-to-treat specification across each activity classification for top developers 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 core sample is split along the median to create high and low ability subsamples for four different proxies: a measure of their share of repository activity (i.e. centrality), the developer's number of GitHub achievements, number of followers, and GitHub account age (i.e. tenure). Time frame: July, 2022 to July, 2023. Robust standard errors are in parentheses: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$

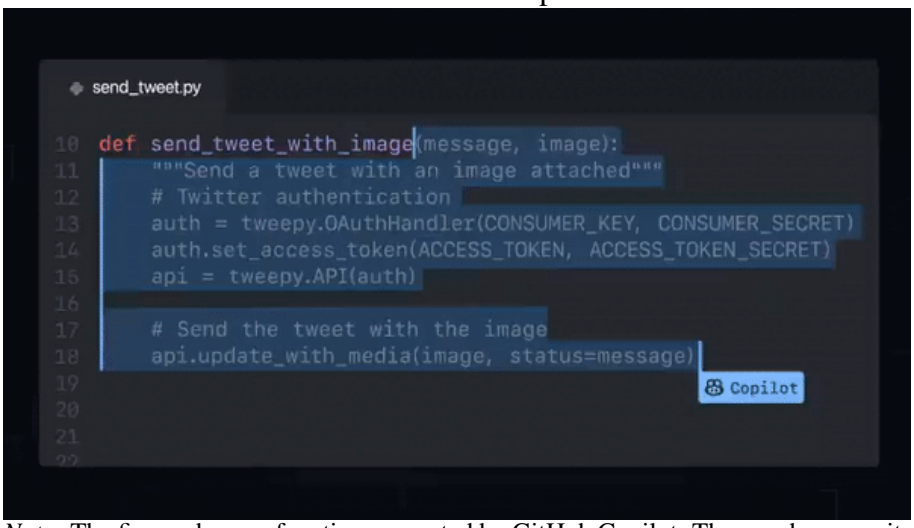
Appendices

Appendix A Additional Tables and Figures

Figure A1: GITHUB COPILOT: ARTIFICIAL INTELLIGENCE IN ACTION

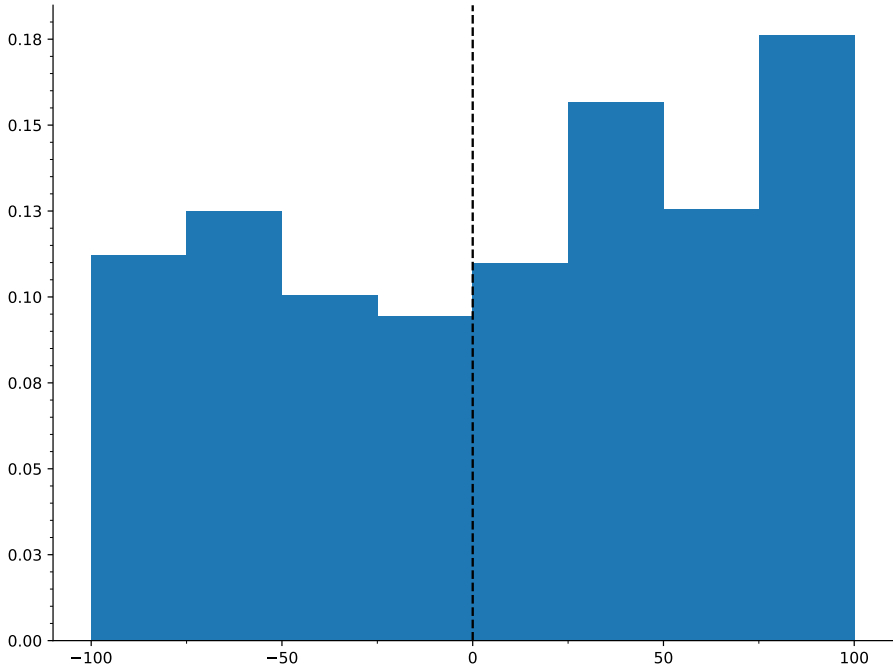
Panel A: User Input

Panel B: Code Completion



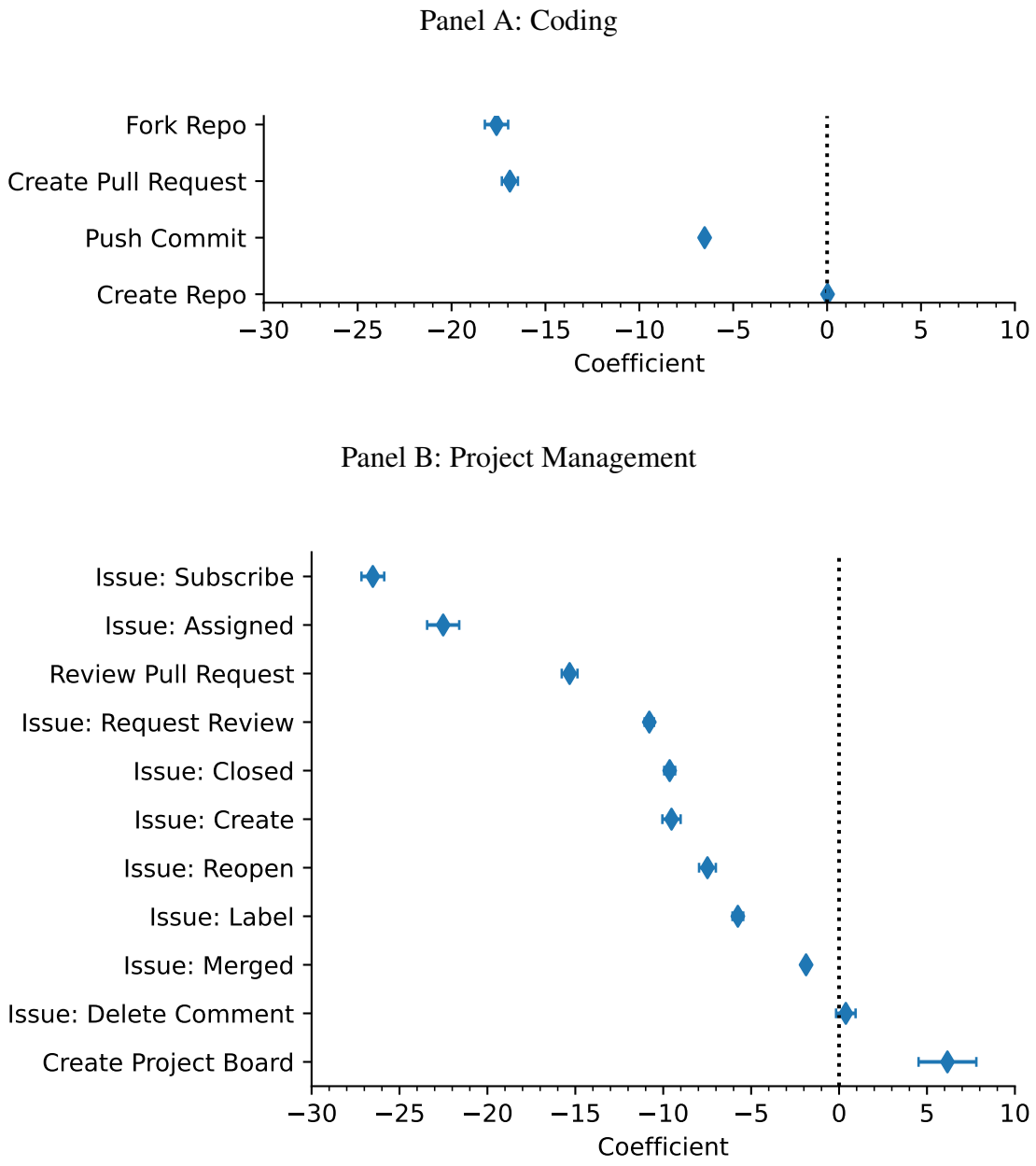
Note: The figure shows a function generated by GitHub Copilot. The user began writing a function (Panel A) and Copilot suggested the rest based on this initial suggestion (Panel B).
Source: GitHub 2022

Figure A2: DISTRIBUTION OF MINIMUM RANKINGS



Note: The figure shows the distribution of the minimum rankings for the top developer 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 ON THE NUMBER OF DISTINCT COLLABORATORS ACROSS EACH ACTIVITY TYPE



Note: The figure shows intent-to-treat coefficient estimates where the outcome variable is the cumulative mean of distinct collaborators a developer interacts with in public repositories. This measure is calculated for each granular activity. Panel A contains activities related to coding and Panel B contains activities related to project management. The estimates uses developers within the ranking bandwidth of $h \in [-100, 100]$. Developers with rankings below zero receive free access to the AI through the top developer channel while those above do not. Time frame: July, 2022 to July, 2023. Horizontal error bars represent 95% condfidence intervals for the coefficient estimate and are based on robust standard errors.

Table A1: DESCRIPTIVE STATISTICS OF TOP MAINTAINERS FOR 1 YEAR

	Mean	SD	Min	Max
AI Treatment				
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
Work Activities				
Coding	0.44	0.22	0	1
Project Management	0.37	0.20	0	1
All Activities	25.79	267.70	0	$8,665$
Ranking				
Normalized Ranking	-363.94	560.98	-999	1000

Note: The table shows univariate descriptive statistics for top developers 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 developers within a time period from July 2022 - July 2023.

Table A2: DESCRIPTIVE STATISTICS OF TOP MAINTAINERS FOR 2 YEARS

	Mean	SD	Min	Max
AI Treatment				
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
Work Activities				
Coding	0.45	0.22	0	1
Project Management	0.36	0.19	0	1
All Activities	23.74	253.10	0	198,498
Ranking				
Normalized Ranking	−368.22	558.72	−999	1000

Note: The table shows univariate descriptive statistics for top developers 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 developers within a time period from July 2022 - July 2024.

Table A3: DESCRIPTIVE STATISTICS OF TOP MAINTAINERS WITHIN MAIN BANDWIDTH

	Mean	SD	Min	Max
AI Treatment				
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
Work Activities				
Coding	0.44	0.22	0	1
Project Management	0.37	0.19	0	1
All Activities	17.28	51.50	0	$4,942$
Ranking				
Normalized Ranking	0.62	60.33	-100	100

Note: The table shows univariate descriptive statistics for top developers 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 developer within the bandwidth of $h \in [-100, 100]$. Our full balanced panel contains 215,169 observations for 5,521 developers within a time period from July 2022 - July 2023.

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

	Mean	SD	Min	Max
Coding				
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
Project Management				
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: SUMMARY STATISTICS FOR ABILITY PROXIES

	Mean	SD	Min	Median	Max
Max Centrality	0.129	0.222	0	0.0256	1
Number of Achievements	8.26	4.29	1	8	26
Number of Followers	91.74	402	1	15	17,650
GitHub Tenure	706	345	2	713	1,420

Note: This table contains summary statistics for the set of ability proxies considered for Hypotheses 4a and 4b. Max centrality the the largest share of commits the developer is responsible for across all public repositories where they have made at least one commits. Number of Achievements counts the cumulative total of "badges" the developer has earned on the GitHub platform. Number of followers count peers who subscribe to notifications that track the developer's public activity. GitHub tenure is measured by counting the number of days since the developer first created their GitHub account.

Table A6: DIFFERENCES-IN-DIFFERENCES AND PROPENSITY SCORE MATCHING

Panel A: Differences-in-differences

	Coding		Project Management	
$\mathbb{1}(Eligible)$	0.004^{***}	0.008^{***}	-0.011^{***}	-0.006^{***}
	(0.001)	(0.001)	(0.002)	(0.001)
Baseline	0.472^{***}	0.461^{***}	0.309^{***}	0.312^{***}
	(0.001)	(0.001)	(0.001)	(0.001)
TE (%)	0.8	1.7	-3.6	-1.9
N	278,040	249,165	278,040	249,165
Controls		✓		✓

Panel B: Propensity Score Matching

	Coding		Project Management	
$\mathbb{1}(Eligible)$	0.038^{***}	0.042^{***}	-0.099^{***}	-0.083^{***}
	(0.003)	(0.003)	(0.003)	(0.003)
Baseline	0.413^{***}	0.429^{***}	0.417^{***}	0.467^{***}
	(0.001)	(0.002)	(0.001)	(0.002)
TE (%)	9.3	9.8	-23.7	-17.8
N	108,376	108,376	108,376	108,376
Controls		✓		✓

[a] *Note:* This table collects intent-to-treat coefficient estimates using differences-in-differences (Panel A) and matching (Panel B). In Panel A, we select a subsample of developers that were ranked $h \in [-100, 0]$ in *the first week of the general access period* and compare them with a set of developers ranked $h \in (0, 100]$ who were *never eligible for complementary Copilot AI* in the first year following general access. We compare the difference in work patterns between these two groups up to one year before GA with the difference up to one year after. The specifications are two-way fixed effect (TWFE) models and include developer and week fixed effects. Treatment effects are calculated relative to baseline mean task allocation for ineligible developers. In Panel B, we take a sample of developers ranked $h \in [-100, 0]$ and match them with a developer ranked $h \in (0, 100]$ using propensity scores. Propensity scores are derived from a logistic regression of the developers Copilot eligibility status on their rank and covariate controls (platform experience, centrality, follower counts, and platform achievements). The results in Panel B are robust to alternative matching procedures (nearest neighbor, full matching, and coarsened exact matching). Robust standard errors are in parentheses: $^{***}\, p < 0.001$; $^{**}\, p < 0.01$; $^{*}\, p < 0.05$

Table A7: HETEROGENEITY BY FIRM AFFILIATION

	Coding		Project Management	
Firm Affiliated	No	Yes	No	Yes
$\mathbb{1}(Eligible)$	0.0590^{***}	0.0681^{***}	-0.1178^{***}	-0.0953^{***}
	(0.002)	(0.005)	(0.002)	(0.004)
Baseline	0.4337^{***}	0.4296^{***}	0.3952^{***}	0.4342^{***}
	(0.001)	(0.003)	(0.001)	(0.003)
Rel. TE (%)	13.6	15.9	-29.8	-21.9
N	240,825	28,721	240,825	28,721

Note: This table shows the coefficient estimates from the intent-to-treat specification across each activity classification for top developers 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 second and fourth columns contain the subset of developers that are affiliated with a firm while the first and third columns contain the subset of developers that are not affiliated with a firm. Time frame: July, 2022 to July, 2023. Robust standard errors are in parentheses: $^{***}\, p < 0.001$; $^{**}\, p < 0.01$; $^{*}\, p < 0.05$

Table A8: INTENT-TO-TREAT EFFECTS OF COPILOT ON AUTONOMOUS WORK

	Number of Distinct Collaborators	
$\mathbb{1}(Eligible)$	-17.251^{***}	-14.697^{***}
	(0.346)	(0.351)
Baseline	21.759^{***}	29.947^{***}
	(0.279)	(0.246)
Rel. TE (%)	-79.3	-49.1
N	161,155	155,113
Controls		✓

Note: This table shows the coefficient estimates from the intent-to-treat specification that compares top developers 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 outcome is the cumulative mean number of distinct collaborators the developer interacts with across all public projects through the current period. A larger value for this measure indicates that the developer interacts in repositories with more peers. Covariate controls include the developer's GitHub account age in days, 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: $^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

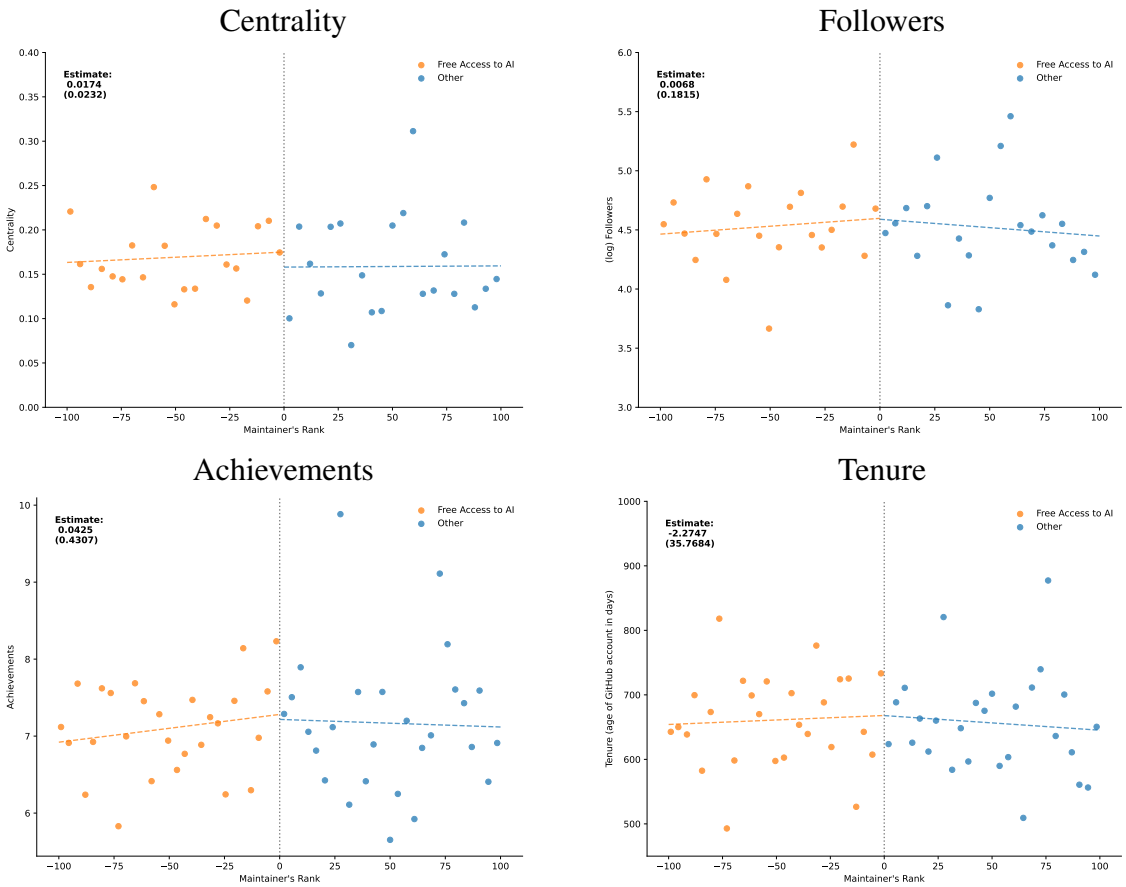
Appendix B Classification of Work Activities

Coding	Project Management
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

Note: Each high-level activity, coding and project management, is defined as the sum of its disaggregated, granular activities.

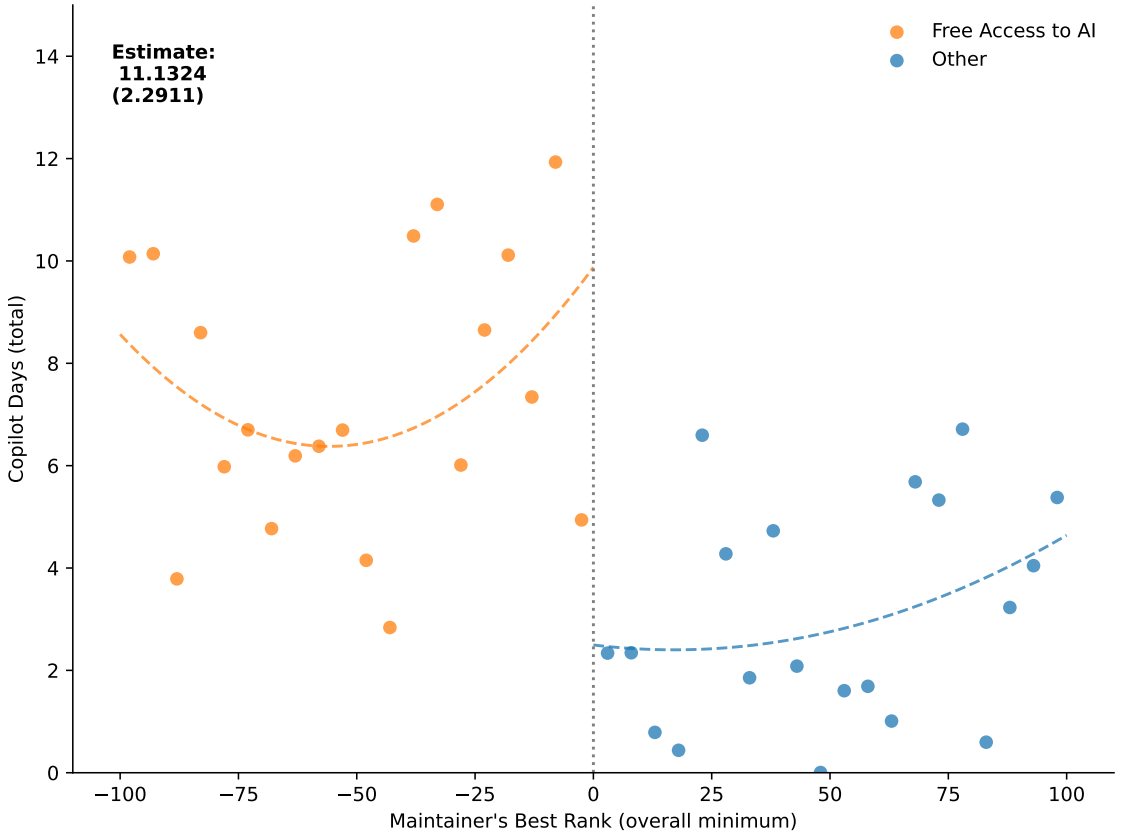
Appendix C Robustness Checks

Figure C1: COVARIATE SMOOTHNESS



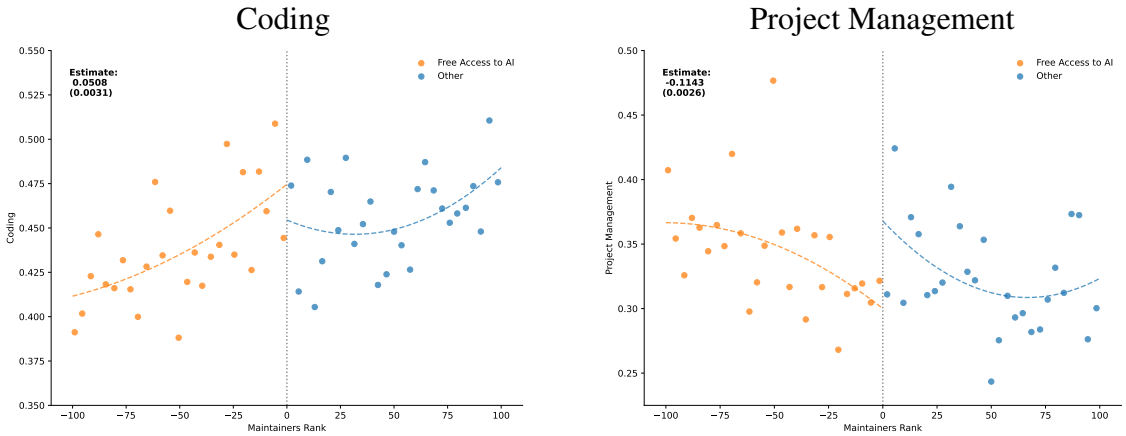
Note: The figure shows the covariate smoothness check across the overall minimum ranking on GitHub for (clockwise from upper left) a measure of the developer's centrality within their repository, follower counts, age of the developer's GitHub account in days (tenure), and developer platform achievements. After the general access period, developers with rankings below zero receive free access to the AI through the top developer channel while those above do not. Time frame: Covariates are observed between January, 2021 and July, 2023. Maintainer rankings are observed form July 2022 through July 2023. Robust standard errors are in parentheses.

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



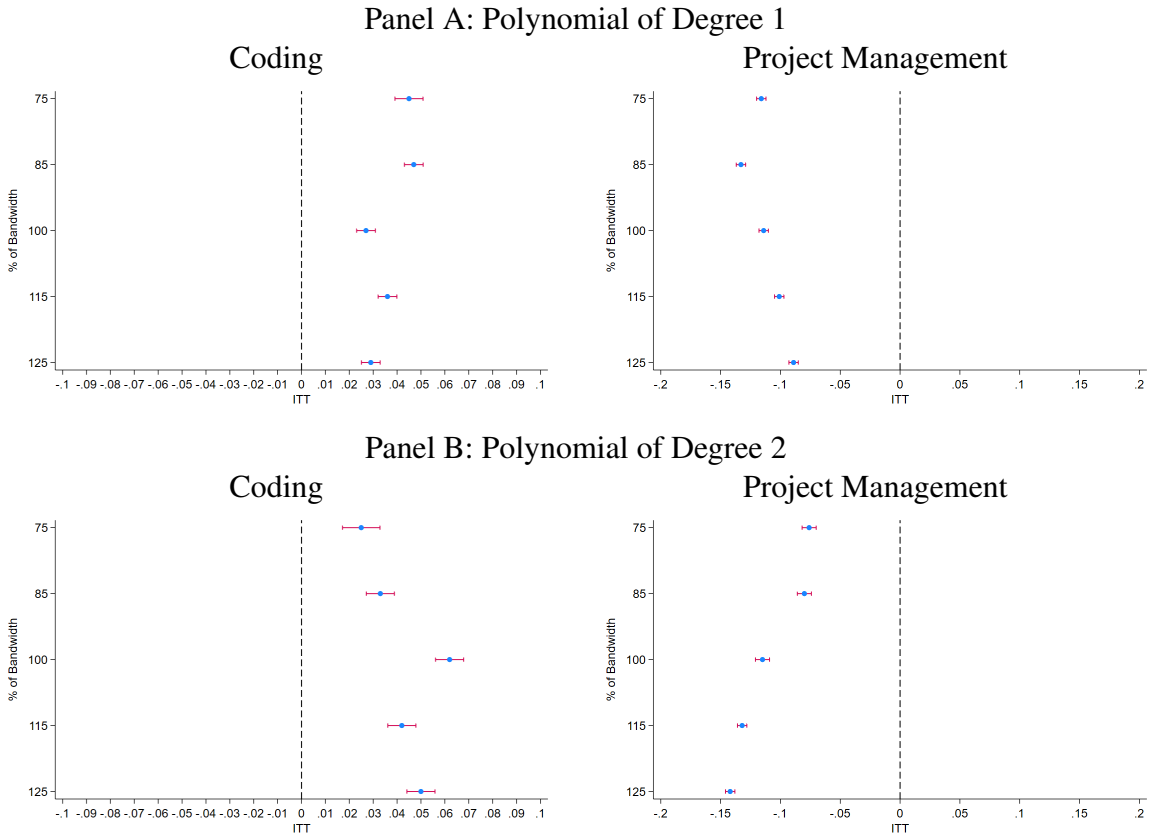
Note: The figure shows the total number of days the generative AI tool GitHub Copilot was used across the overall minimum ranking based on the six top languages on GitHub with a quadratic fit on either side of the threshold. Developers with rankings below 0 receive free access to the AI through the top developers 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 quadratic fit on either side of the threshold. Developers with rankings below zero receive free access to the AI through the top developers 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 intent-to-treat (ITT) effects of the generative AI GitHub Copilot 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 a McCrary (2008) test of eligibility manipulation for ranking $\in [-100, 100]$ bandwidth. The outcome variable is a count of the developer-week observations 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			Project Management		
	Uniform	Triangular	Epanechnikov	Uniform	Triangular	Epanechnikov
$\mathbb{1}(Eligible)$	0.054^{***}	0.052^{***}	0.054^{***}	-0.100^{***}	-0.107^{***}	-0.110^{***}
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
N	215,169	215,169	215,169	215,169	215,169	215,169

[a] *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, triangular, and Epanechnikov kernels. 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

	Coding		Project Management	
	MSE	CER	MSE	CER
$\mathbb{1}(Eligible)$	0.0429^{***}	0.0233^{***}	-0.0997^{***}	-0.0509^{***}
	(0.002)	(0.002)	(0.002)	(0.003)
Baseline	0.4549^{***}	0.4458^{***}	0.3541^{***}	0.3393^{***}
	(0.002)	(0.003)	(0.002)	(0.002)
Rel. TE (%)	9.4	5.2	-28.2	-15.0
Optimal Bandwidth	$[-47, 47]$	$[-35, 35]$	$[-51, 51]$	$[-38, 38]$
N	123,026	90,938	132,057	95,116

[a] *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 derived from the mean squared error (MSE) in columns 1 and 3 and the coverage error rate (CER) estimator in columns 3 and 4 (Calonico, Cattaneo, and Farrell, 2020). 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 C4: ABSOLUTE MEASURES

	Coding	Project Management
$\mathbb{1}(Eligible)$	94.377^{***}	-68.497^{***}
	(22.772)	(10.761)
N	269,546	269,546

[a] *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 absolute measures. 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 C5: RESIDUAL

	Comment	Reaction	Starring	Unstarring	Residual
$\mathbb{1}(Eligible)$	-0.0136^{***}	0.0184^{***}	0.0452^{***}	0.0060^{***}	0.0561^{***}
	(0.001)	(0.001)	(0.001)	(0.000)	(0.002)
Baseline	0.1040^{***}	0.0264^{***}	0.0346^{***}	0.0033^{***}	0.1683^{***}
	(0.001)	(0.000)	(0.001)	(0.000)	(0.001)
N	269,546	269,546	269,546	269,546	269,546

[a] *Note:* This table shows the intent-to-treat effects of the generative AI tool Copilot on work activities of the residual decomposed into comments, reactions, starring and unstarring and jointly in the last column. Our full balanced panel is observed within a time period from July, 2022 to July, 2023. Robust standard errors are in parentheses: $^{***}\,p < 0.001$; $^{**}\,p < 0.01$; $^{*}\,p < 0.05$

Appendix D Details of the Economic Model

D.1 Baseline model: Comparative Statics

How will the optimal choice of these activites change as p_c changes (e.g. $\partial c^\star/\partial p_c$ and $\partial m^\star/\partial p_c$)? Both Marshallians depend on the relative cost of core work, p_c. The elasticity of substitution σ is the key parameter which determines the sign of the relationship between core work and project management, i.e. whether they are gross substitutes or complements.

$$\frac{\partial c^\star}{\partial p_c} = \frac{-\omega(\frac{\beta_c}{\beta_m}\sigma p_c^{-(\sigma+1)} + p_c^{-2\sigma})}{(p_c^{1-\sigma} + \frac{\beta_m}{\beta_c})^2} \tag{6}$$

$$\frac{\partial m^\star}{\partial p_c} = \frac{\omega(\sigma-1)\frac{\beta_c}{\beta_m}p_c^{-\sigma}}{(\frac{\beta_c}{\beta_m}p_c^{1-\sigma}+1)^2} \tag{7}$$

This leads to two cases for coding and three cases for project management

$$\frac{\partial c^\star}{\partial p_c}\begin{cases} = 0 & if \sigma = 0, \\ < 0 & if \sigma > 0, \end{cases} \tag{8}$$

$$\frac{\partial m^\star}{\partial p_c}\begin{cases} = 0 & if \sigma = 1, \\ > 0 & if \sigma > 1, \\ < 0 & if 0 \leq \sigma < 1. \end{cases} \tag{9}$$

For coding, an increase in the price is independent of σ and will always lead to a reduction in the coding share, while the sign for project management is ambiguous. When $\sigma = 1$, then coding and project management are neither perfect substitutes nor perfect complements (Cobb Douglas case). When $\sigma > 1$ then the coding and project management are substitutes. When $0 \leq \sigma < 1$ then coding and project management are complements and in the equal case perfect complements.

D.2 Model Extension

We extend the simple CES model to get insights into the mechanisms by splitting core work and management through a nested CES approach. An agent chooses between four activities to maximize utility, they are: autonomous core work c_1, collaborative core work c_2, autonomous project management m_1 and collaborative project management m_2. We can similarly think about the dimensions of exploratory core work c_1, exploitative core work c_2, exploratory project management m_1 and exploitative project management exploration m_2. The nested CES utility function is then set up as follows:

$$u(c_1, c_2, m_1, m_2) = \left(\alpha_c^{1/\sigma} u(c_1, c_2)^{\frac{\sigma-1}{\sigma}} + \alpha_m^{1/\sigma} u(m_1, m_2)^{\frac{\sigma-1}{\sigma}}\right)^{\frac{\sigma}{\sigma-1}} \tag{10}$$

where the utility for the composite goods are defined as follows:

$$u(c_1, c_2) = \left(\beta_{c_1}^{1/\rho} c_1^{\frac{1-\rho}{\rho}} + \beta_{c_2}^{1/\rho} c_2^{\frac{1-\rho}{\rho}}\right)^{\frac{\rho}{1-\rho}} \tag{11}$$

$$u(m_1, m_2) = \left(\beta_{m_1}^{1/\tau_m} m_1^{\frac{\tau_m-1}{\tau_m}} + \beta_{m_2}^{1/\tau_m} m_2^{\frac{\tau_m-1}{\tau_m}}\right)^{\frac{\tau_m}{\tau_m-1}} \tag{12}$$

Since the relevant comparative static is around P_c, we keep P_c and P_m as CES aggregators of subgood composite prices. After obtaining Marshallian demands for the composite goods $u(c_1, c_2) = u(c)$ and $u(m_1, m_2) = u(m)$, and within each nested composite good, we can obtain the final Marshallian demands:

$$c_1^* = \frac{\omega}{\left[\beta_{c_1}^{\frac{1}{\rho}} + \beta_{c_2}^{\frac{1}{\rho}} \left(\left(\frac{P_{c_1}}{P_{c_2}}\right)^{\rho} \frac{\beta_{c_2}}{\beta_{c_1}}\right)^{\frac{\rho-1}{\rho}}\right]^{\frac{\rho}{\rho-1}}} \times \frac{1}{\left(\frac{\alpha_c}{\alpha_m} P_m \left(\frac{P_m}{P_c}\right)^{\sigma} + P_c\right)} \tag{13}$$

$$c_2^* = \frac{\omega}{\left[\beta_{c_2}^{\frac{1}{\rho}} + \beta_{c_1}^{\frac{1}{\rho}} \left(\left(\frac{P_{c_2}}{P_{c_1}}\right)^{\rho} \frac{\beta_{c_1}}{\beta_{c_2}}\right)^{\frac{\rho-1}{\rho}}\right]^{\frac{\rho}{\rho-1}}} \times \frac{1}{\left(\frac{\alpha_c}{\alpha_m} P_m \left(\frac{P_m}{P_c}\right)^{\sigma} + P_c\right)} \tag{14}$$

$$m_1^* = \frac{\omega}{\left[\beta_{m_1}^{\frac{1}{\tau}} + \beta_{m_2}^{\frac{1}{\tau}} \left(\left(\frac{P_{m_1}}{P_{m_2}}\right)^{\tau} \frac{\beta_{m_2}}{\beta_{m_1}}\right)^{\frac{\tau-1}{\tau}}\right]^{\frac{\tau}{\tau-1}}} \times \frac{1}{\left(\frac{\alpha_m}{\alpha_c} P_c (\frac{P_c}{P_m})^{\sigma} + P_m\right)} \tag{15}$$

$$m_2^* = \frac{\omega}{\left[\beta_{m_2}^{\frac{1}{\tau}} + \beta_{m_1}^{\frac{1}{\tau}} \left(\left(\frac{P_{m_2}}{P_{m_1}}\right)^{\tau} \frac{\beta_{m_1}}{\beta_{m_2}}\right)^{\frac{\tau-1}{\tau}}\right]^{\frac{\tau}{\tau-1}}} \times \frac{1}{\left(\frac{\alpha_m}{\alpha_c} P_c (\frac{P_c}{P_m})^{\sigma} + P_m\right)} \tag{16}$$

The elasticity ρ determines how allocation shifts across the broader categories of core work and project management, while σ and τ determine the within core work and within project management allocation, respectively. If we consider the same price shock, due to AI on P_c, we arrive at the same conclusion as in the baseline model. The relative demand of the overall composite good c increase since the cost of coding has dropped. To understand the relative magnitude, we have to compare

the comparative statics for the within composite good comparison of c_1 and c_2. When the costs of either one of the two goods is relatively lower (or the preferences are higher) then, we will shift relatively more of our allocation towards that good. For example, when the price of autonomous core work P_{c_1} is relatively lower than the price of collaborative core work, P_{c_2}, then individuals are more likely to switch demand to autonomous work. The same logic holds true for project management as well as exploratory and exploitative work.