Task Efficiency and Signaling in the Age of GenAI: Effort Reallocation and Firm Value Effects

Shiwei Ye*

Job Market Paper

This Version: December 26, 2025

Link to the Latest Version

Abstract

I study how Generative AI (GenAI) reshapes effort allocation and firm value by affecting both productivity and signaling. Using developer-level data from U.S. public firms' open-source projects, I construct a novel AI exposure measure and exploit GitHub Copilot's launch as a shock. I find that GenAI increases coding productivity for senior developers while juniors create more valuable projects. This is consistent with seniors capturing productivity gains while career-concerned juniors shift toward creative work that better signals ability, as GenAI erodes coding's signaling value. Changes in signaling incentives are reflected in project selection, programming language choices, and job moves across firms. Stock market reactions to GitHub Copilot's launch show that non-innovative firms with senior-heavy teams, where signaling incentives align with productivity goals, benefit most. A multitask signaling model rationalizes these patterns. These findings shed new light on the dual role of GenAI as a productivity tool and a force reshaping labor market signaling.

Keywords: Generative AI, Productivity, Innovation, Signaling, Firm value
JEL Classification: G10, J24, O33, O36

Acknowledgements: I am indebted to Thomas Lambert and Daniel Metzger for their mentorship and generous support. I am also grateful for Ashwini Agrawal, Elliott Ash, Dion Bongaerts, Jean-Edouard Colliard, Robin Döttling, Daniel Ferreira, Luca Henkel, Milena Nikolova, Mikael Paaso, and participants at Corporate Finance Day, ERIM PhD seminar, European Conference on Corporate R&D and Innovation, SGF Conference, and Zurich Workshop in AI+Economics for helpful comments. All errors are my own. Part of this research was conducted during my visit to the London School of Economics. I am grateful for their hospitality and the financial support from the Erasmus Trustfonds.

*Rotterdam School of Management, Erasmus University. Email: ye@rsm.nl.

1 Introduction

Generative AI (GenAI) is reshaping work across all experience levels, but its distributional effects remain contested. Research documents substantial productivity gains for junior employees (Brynjolfsson, Li and Raymond, 2023; Kogan, Papanikolaou, Schmidt and Seegmiller, 2023; Noy and Zhang, 2023; Cui, Demirer, Jaffe, Musolff, Peng and Salz, 2024; Hoffmann, Boysel, Nagle, Peng and Xu, 2024b). Yet by making it easier for anyone to produce competent-looking work, GenAI also weakens employers' ability to assess genuine skill (Cowgill, Hernandez-Lagos and Wright, 2024; Colliard and Zhao, 2025; Cui, Dias and Ye, 2025; Wiles and Horton, 2025). Consequently, GenAI may hurt the very workers it helps most. Hiring has shifted from early-career to more experienced candidates in AI-exposed occupations,[1] with industry reports citing employer uncertainty about assessing ability without AI as one contributing factor.[2] This tension between productivity gains and signaling applies to many knowledge work outputs that juniors use to signal ability and work ethic, from valuation modeling in finance to polished writing and data analysis in academia. Yet how signal erosion affects worker effort allocation and, in turn, firms' returns to AI adoption, remains unclear.

I investigate how employees at different seniority levels allocate effort between AI-assisted tasks and creative activities, and its implication for firm value. Because AI provides less assistance for creative work, such tasks retain their signaling value. I use open-source software (OSS) projects made available by U.S. public firms on GitHub, the most popular open-source platform, as my empirical laboratory.

This empirical setting is relevant because GenAI pronouncedly affects software de-

[1] The declining trend is documented in administrative data (Brynjolfsson, Chandar and Chen, 2025; Lichtinger and Hosseini Maasoum, 2025).

[2] See HackerRank's 2025 Developer Skills Report: https://www.hackerrank.com/reports/developer-skills-report-2025. In practice, some firms have begun restricting applicants' use of AI in interviews, citing difficulty in assessing genuine ability. See https://www.businessinsider.com/amazon-stop-people-using-ai-cheat-job-interviews-2025-2 for Amazon and https://fortune.com/2025/06/11/goldman-sachs-students-ai-chatgpt-interviews-amazon-anthropic/ for Goldman Sachs.

velopment, and signaling is an important driver for participating in open-source projects (Lerner and Tirole, 2005a). OSS is particularly well suited for studying signaling because contributions are visible to the entire labor market, not just within a single firm. However, this visibility also means that signaling-driven behavior may disproportionately appear in OSS, while productivity gains remain in proprietary codebases. Furthermore, OSS represents an economically important context for studying value creation, as it generates economic value for both firms that publish OSS projects and the broader society through private value creation (Emery, Lim and Ye, 2024) and substantial externalities (Hoffmann, Nagle and Zhou, 2024a).[3]

To investigate this question, I use a generalized difference-in-differences (DID) approach to study the causal effects of GenAI on coding and innovation outcomes of developers working for firms. I exploit the official launch of GitHub Copilot, a coding tool powered by OpenAI's large language models (LLMs) and widely adopted since then, on June 21st, 2022.[4]

Because LLMs are trained on unequal quantities of code across programming languages, some languages (such as Python) benefit more from GenAI than others. I exploit this variation to construct a novel developer-level measure of GenAI exposure based on the programming languages developers use in their *ex ante* codebase portfolio. I validate this measure in three ways. First, a higher AI usefulness score predicts a higher comment-to-code ratio in scripts after GenAI's introduction, consistent with developers using comments as prompts for LLMs or better documentation practices induced by GenAI. Second, languages with higher AI usefulness scores experience slower question growth on Stack Overflow, consistent with GenAI reducing the need to post common

[3] In practice, firms increasingly choose to make their innovation open source. As of 2022, 90% of Fortune 100 companies use GitHub. See https://octoverse.GitHub.com/2022/.

[4] The tool is integrated seamlessly into development environments and assists developers by suggesting code snippets in real time as they type code or natural language instructions. Since its introduction, developers have quickly adopted the tool, with over one million paid subscribers in 2023 and one third Fortune 500 adopters as of December 2022. See https://GitHub.com/features/copilot (September 2024).

queries. Third, I show the AI usefulness score reflects how actively a language is discussed in connection with coding using GenAI on Reddit.

I then calculate AI exposure scores for each developer by taking the weighted average of their language-level AI usefulness scores, and compare the top quartile of developers (treatment group) with the bottom three quartiles (control group) before and after GenAI's introduction.

I find that GenAI improves productivity in AI-assisted tasks for senior developers (those with above-median tenure on GitHub). Developers with high AI exposure are 1.16 percentage points more likely to contribute code to firm-owned projects each month. Effects are driven by senior developers: the triple-difference estimate shows expected coding events for AI-exposed seniors increase by about 9.2% more than for juniors. These productivity gains cannot be purely explained by more working hours or lower quality of outputs, as proxied by issues reported. Dynamic effects observed from event-study analysis show that firms' developers immediately react to the introduction of GenAI, that the effects persist over time, with no evidence of pre-treatment trend differences.

Next, I turn to studying how GenAI affects innovation outcomes. While GenAI does not affect the probability of creating new projects on average, it increases community adoption and interest in new projects, as measured by GitHub forks (a citation-like measure) and stars (a bookmarking system that serves as a widely-used proxy for popularity and quality) received by projects as of February 2024. However, the effect on private firm value depends critically on team composition. Overall, projects from high-AI-exposure teams experience negative stock market reactions at release, driven primarily by senior-heavy teams. All-junior teams with high AI exposure see 14% higher project value, while adding senior developers reduces this gain by approximately 12 percentage points per senior added to a five-person team. Projects led by junior developers with high AI exposure are significantly more novel than those led by seniors, as measured by both

LLM-based conceptual assessment and text-similarity-based distinctiveness from prior work.

In contrast to many studies,[5] my findings suggest that less experienced developers do not engage more in AI-assisted tasks, even though they stand to benefit the most from GenAI. Instead, they generate higher-value innovation. One possible explanation is that AI-generated code reduces the signaling value of coding activities, especially for developers with shorter tenure.[6]

I develop a theoretical model adapting the frameworks of Holmström and Milgrom (1991) and Holmström (1999) that shows two competing effects. While productivity gains incentivize more coding, signal dilution due to AI-induced noise motivates future employers to weight more on alternative signals and career-concerned developers to shift to creativity tasks. These concerns are now widespread in practice. For example, on the labor demand side, Amazon banned AI use in interviews, citing inability to accurately assess candidates. On the labor supply side, Baird, Mar, Xu and Xu (2024) show that existing developers of and developers newly hired by firms adopting GitHub Copilot add more non-programming skills to their LinkedIn profiles. However, signaling effects are less observable in lab settings, internal firm activities, or among top-ranked open-source developers who typically have short tenure with homogeneous career concerns,[7] as studied in previous research on the productivity effect of GenAI.

[5] Studies at the individual level consistently find that junior workers benefit more from GenAI than senior workers. They include, but are not limited to, Brynjolfsson et al. (2023); Dell'Acqua, McFowland, Mollick, Lifshitz-Assaf, Kellogg, Rajendran, Krayer, Candelon and Lakhani (2023); Noy and Zhang (2023), and Hoffmann et al. (2024b).

[6] As conceptualized in Colliard and Zhao (2025), for instance, AI can have a free-riding effect as the probability of success increases even when the worker shirks. Using experiment with pitch writing tasks, Cowgill et al. (2024) find that AI reduces evaluators' ability to identify true expertise by about 4 to 9 percent.

[7] For example, Hoffmann et al. (2024b) focus on developers contributing to the most popular public repositories for identification purposes. Tenure in their study has a sample mean of 706 days and a maximum of 1,420 days. In comparison, the sample used in this paper has a mean tenure of 2,690 days and a maximum of 5,274 days.

Consistent with the theoretical prediction, signaling incentives manifest in labor market outcomes. Matching GitHub developers to LinkedIn employment records, I find that junior developers with greater AI exposure are 1.44 percentage points more likely to change employers per quarter. Among those switching firms, more innovative juniors experience promotion while pure coders face demotion, consistent with signal reweighting. Importantly, these effects appear only in external (across-firm) moves; there is no effect on internal transitions. This distinction supports the signaling interpretation: public signals inform prospective employers who lack direct observation, while current employers already know their workers' true ability. In contrast, senior developers show no effects, consistent with established track records.

I show that the signaling channel also drives project selection and language choices. Senior developers target popular team projects for visibility benefits, while junior developers seek peer monitoring over popularity by avoiding solo work amid AI-introduced noise. Moreover, senior developers increase activities for both new and familiar languages, while juniors focus only on new ones. This is consistent with signaling value being more equal where neither group has track records. Overall, the labor market and behavioral evidence is difficult to reconcile with alternative mechanisms, including experience complementarity, task specialization, skill revaluation, and displacement; the signaling mechanism best explains the full set of findings.

The effect of signaling incentives extends to firms, where the incentive alignment between firms and their workforce composition plays a crucial role. Using stock market data for IT firms with GitHub presence, I define innovative firms as those with above-median R&D intensity. Non-innovative firms with a higher proportion of senior developers, who rely less on signaling through alternative tasks, are more likely to benefit from adopting GenAI tools to increase efficiency in AI-assisted tasks. To test this prediction, I conduct an event study examining cumulative abnormal returns (CARs) around the official launch of GitHub Copilot. I find that AI-exposed firms with aligned incentives experience significantly higher CARs in the 10- to 30-day windows following Copilot's launch, with the effect concentrated among non-innovative firms. These results suggest that investors perceive firms whose workforce composition aligns with signaling incentives as better positioned to benefit from GenAI tools.

Literature. This study makes several contributions to the literature. First, this paper adds to the growing body of research on the impact of artificial intelligence (AI), and more recently, GenAI on labor market inequality. Early research on automation and AI often focused on job displacement and wage polarization (Acemoglu and Autor, 2011). More recent studies highlight labor-augmenting capabilities of GenAI and suggest it disproportionately benefits less-experienced or lower-skilled workers, potentially reducing skill inequality (Brynjolfsson et al., 2023; Dell'Acqua et al., 2023; Kogan et al., 2023; Noy and Zhang, 2023; Cui et al., 2024; Hoffmann et al., 2024b). Yet, large-scale administrative data suggest that AI exposure is associated with higher unemployment (Ozkan and Sullivan, 2025). In particular, since the adoption of GenAI, entry-level job hiring has declined where AI tends to automate labor, while senior positions remain unaffected (Brynjolfsson et al., 2025; Lichtinger and Hosseini Maasoum, 2025). This paper contributes to the discussion by showing that in addition to automation, concerns about adverse selection may also explain the seniority-biased labor effect of GenAI. As GenAI brings information distortion to ability signals, junior workers who signal through alternative tasks see better labor market outcomes. My findings therefore highlight that signal dilution, not just augmentation or automation, shapes how different workers fare as AI is introduced.

Second, this paper speaks to the literature on the role of AI in firm value and growth. Several studies have examined how AI may affect firm value through labor productivity (Eisfeldt, Schubert and Zhang, 2023; Kogan et al., 2023), labor composition (Babina, Fedyk, He and Hodson, 2023; Berger, Cai, Qiu and Shen, 2024), product innovation (Babina, Fedyk, He and Hodson, 2024), and entrepreneur decision making (Otis, Clarke, Delecourt, Holtz and Koning, 2024). To my knowledge, this paper is among the first to show that employer-employee alignment in signaling incentives shapes firms' returns to AI adoption.

Conceptually, this paper is related to the literature centering on signaling and career concerns and how technology advancements reshape the dynamics. In the seminal career concerns model of Holmström (1999), forward-looking employees balance current output with actions that reveal their talent to improve future opportunities. OSS developers contribute partly to signal their ability and advance their careers (Lerner and Tirole, 2005b; Gupta, Nishesh and Simintzi, 2024). However, GenAI makes it easier to produce outputs that appear competent (Cowgill et al., 2024; Colliard and Zhao, 2025; Cui et al., 2025), making AI-assisted signals less reliable. This paper is one of the first to provide empirical evidence of the trade-off between productivity and signaling on the labor supply side, showing that career-concerned workers may shift to producing signals that are less affected even if GenAI increases productivity for AI-assisted tasks. Over time, this could change career development trajectories, where human capital investment may focus more on areas where human creativity and insight remain paramount.

Methodologically, this study also contributes to the literature that uses GenAI to generate new data and construct measurements to overcome various data challenges in academic research. For example, researchers have leveraged large language models (LLMs) to summarize or classify unstructured data (Cheng, Lee and Tambe, 2022; Beckmann, Beckmeyer, Filippou, Menze and Zhou, 2024; Chen and Wang, 2024; Kim, Muhn and Nikolaev, 2024) and construct measures for variables that require less subjective evaluation, such as occupational AI exposure scores (Eisfeldt et al., 2023; Eloundou, Manning, Mishkin and Rock, 2023; Kogan et al., 2023). This paper develops and validates a novel LLM-based AI exposure score for programming languages, enabling GenAI-related research on developer data at a more detailed level.

The remainder of the paper is organized as follows. Section 2 describes the institutional setting of open-source software development and GitHub Copilot. Section 3 develops a conceptual framework that formalizes the trade-off between productivity gains and signal dilution. Section 4 presents the data, AI exposure measure, and identification strategy. Section 5 reports the empirical findings on productivity, innovation, and signaling effects. Section 6 concludes.

2 Institutional Background

2.1 Open Source Software and Commercial Engagement

Systems granting excludability, such as patents, have been seen to be important to incentivize innovation (Arrow, 1962; Crouzet, Eberly, Eisfeldt and Papanikolaou, 2022). Yet, there has been an increasing trend in open-source innovations, particularly in the software industry. Based on the definition of the Open Source Initiative, "open source" means not only access to the source code but also allowing free redistribution and modification under terms defined by open-source licenses. Therefore, when an innovation is "open-sourced," it is made publicly available to all parties at little or no cost. Because of potential knowledge spillovers and the reduction of replacement costs for open-source software (OSS) adopters, OSS can generate large externalities and facilitate innovation in society as a whole (Fershtman and Gandal, 2011; Nagle, 2019; Hoffmann et al., 2024a; Chen, Shi and Srinivasan, 2024). The recent debates over open-source large language models further show the increasing importance and impact of open-source innovation.

While open-source innovations contribute to social welfare, they can also generate private value for firms.[8] Indeed, many firms choose to make their innovation open source.

[8] There is a broad literature studying the incentives for commercial firms to reveal their innovations in an open-source way, see Allen (1983), Lerner and Tirole (2002), Harhoff, Henkel and von Hippel (2003), Dahlander and Gann (2010), Henkel, Schöberl and Alexy (2014), Parker, Van Alstyne and Jiang

A recent survey finds that 90% of Fortune 100 companies use GitHub, the largest platform for developing open-source innovation.[9] Emery et al. (2024) document an increasing trend of open-source activity by U.S. public firms, with these firms representing 68% of the stock market by market capitalization by the end of 2023. We show open-source innovation can generate private value for firms, and this value is a predictor of future sales growth, profitability, employment growth, and patent innovation.

2.2 Software Development Activities on GitHub

GitHub operates on the Git system, which supports a distributed and collaborative framework for software development. Although not all open-source projects are developed on GitHub, it remains the largest platform for such efforts and is closely associated with the concept of open-source software. This section will outline key terms and activities related to software development on GitHub.

To share their innovations on GitHub, firms begin by setting up organization accounts. Within these accounts, they can establish repositories (projects), with administrators determining whether these will be publicly accessible or restricted to selected organization or project members with appropriate permissions. The creation and maintenance of public repositories incur minimal costs, whereas managing private repositories may require GitHub Team or GitHub Enterprise subscriptions for additional support and features. Importantly, despite previous charges for private repositories before GitHub's 2015 shift from a repository-based to a user-based pricing model, public repository hosting has been free since GitHub's launch.

The development process starts with developers making modifications to the codebase, committing these changes locally with concise descriptions. These "commits" are

(2017), Alexy, West, Klapper and Reitzig (2018), Nagle (2018), Teece (2018) and Lin and Maruping (2022). For reviews of the open-source literature, see von Hippel and von Krogh (2003), Goldfarb and Tucker (2019), and Dahlander, Gann and Wallin (2021).

[9] See https://octoverse.GitHub.com/2022/.

then “pushed” to remote branches, making the updates accessible to other contributors and users.

Users who want to follow a repository’s progress can “star” a repository, essentially bookmarking it for future reference. Those who have questions or suggestions can also “open issues,” which are addressed by the development team and the broader community.

Additionally, users can contribute by “forking” the repository, creating a personal copy to work on independently. If the changes made in the fork are considered beneficial to the original project, users can submit “pull requests.” These pull requests are formal proposals to merge their changes back into the original repository. These pull requests are reviewed, and if accepted, the modifications are integrated into the main codebase, further advancing the open-source project.

2.3 GitHub Copilot

GitHub Copilot is a cloud-based AI-powered code completion tool developed by GitHub in collaboration with OpenAI. Specifically, it is built on OpenAI’s Codex model, a large language model trained on vast datasets of public code repositories. The tool integrates seamlessly into popular Integrated Development Environments (IDEs), and is designed to assist developers by suggesting code snippets and entire functions in real-time as they write code. Initially, it was launched in June 2021 in preview, available with a limited number of spots. It has later become generally available to all developers since June 21st, 2022, approximately five months before the public release of ChatGPT. While GitHub Copilot is freely available for verified students and maintainers of popular open-source projects, for most individual developers it is priced at $10 per month. There is also an Enterprise option for business. The tool is widely adopted since then. There are over one million paid subscribers in 2023, and one third of Fortune 500 companies use GitHub Copilot as of December 2022.[10]

[10] See https://GitHub.com/features/copilot (September 2024).

Developers use GitHub Copilot by installing it as an extension in supported IDEs. As they type, Copilot analyzes the code context and offers autocomplete suggestions. It can also generate code based on natural language descriptions, allowing users to input comments describing desired functions or algorithms, and Copilot will output the corresponding code. Therefore, it significantly enhances developer productivity by reducing the time spent on routine coding tasks, lowering the cognitive load, and minimizing common errors. In addition, by offering creative coding solutions and suggesting best practices, it enables developers to learn new coding techniques and languages. In March 2023, GitHub Copilot further offers GPT-4-powered chat feature, which allows developers to engage in a dialogue with the AI assistant to get feedback and suggestions.

3 Conceptual Framework

The desire for peer recognition and career advancement often motivates developers to contribute to open-source projects (Lerner and Tirole, 2005a). As discussed in Section 2, GitHub serves as the largest platform for hosting open-source projects and is widely recognized by developers and employers as a source of productivity signals, particularly from firm-owned projects.[11] For example, Gupta et al. (2024) show that high-skill developers from small firms who reveal their coding activities are more likely to be hired by large firms and promoted to senior roles.

The introduction of GenAI coding tools such as GitHub Copilot can have two opposing effects on productivity signal generation. On one hand, GenAI increases productivity of AI-assisted tasks such as writing and reviewing code. On the other hand, GenAI can dilute the signal from coding outputs. If the second effect dominates, junior developers (whose talent is less known) might choose to spend the time saved on AI-assisted tasks on areas less influenced by GenAI, such as initiating new projects that demonstrate

[11] According to GitHub, most first-time internal and external contributors of open source projects on GitHub chose bigger, company-run repositories. See https://octoverse.GitHub.com/2022/state-of-open-source.

creativity and leadership.

Combining the multitask signaling framework in Holmström and Milgrom (1991) and career concern dynamics in Holmström (1999), I develop a parsimonious framework to conceptualize the trade-off between productivity gain and reduction in signal value of AI-assisted tasks when GenAI becomes available. I allow GenAI to both increase marginal return to effort and dilute the information that hiring markets extract from AI-assisted outputs. While the productivity channel always incentivizes effort reallocation to AI-assisted tasks, the model predicts that the signaling channel will drive developers with less established reputations to shift towards alternative tasks that the market weights more. The following subsections present the core setup and key results; full derivations and extensions are in Appendix A.

3.1 Model Setup

Consider a developer with unobservable talent θ who allocates effort (e_1, e_2) between two tasks. (θ, e_1, e_2) are known to the developer and the current employer, but the labor market is uninformed. Task 1 (e.g., coding) is assisted by GenAI, while task 2 (e.g., creativity) is not. The intensity of AI use in task 1, $\lambda \in [0,1]$, is exogenously given, conditional on availability. The observable outputs from these tasks serve as public signals of the developer's performance:

$$y_1 = \underbrace{\big((1-\lambda)e_1 + \theta\big)}_{\text{Human contribution}} + \underbrace{\lambda(b_g e_1 + g)}_{\text{GenAI contribution}} + \varepsilon_1 = A(\lambda)e_1 + \theta + \lambda g + \varepsilon_1, \quad b_g > 1,$$

$$y_2 = e_2 + \theta + \varepsilon_2.$$

Where $A(\lambda) := 1 + \lambda(b_g - 1) > 1$. Here, $g, \varepsilon_1, \varepsilon_2$ are independent, zero-mean noise terms. GenAI's impact on task 1 is twofold. First, it increases the marginal production of effort through the term $A(\lambda)$. This creates a productivity gain effect. Second, it introduces AI-specific noise, λg, into the output. This makes it harder for the market to discern the developer's true talent θ from y_1, creating a signal-diluting effect. The output of task 2 serves as a stable, human-centric signal.

The developer is career-concerned and maximizes a utility function that includes short-term and long-term rewards, net of a convex effort cost, $C(e_1, e_2) = \frac{1}{2}(e_1^2 + e_2^2 + 2\gamma e_1 e_2)$. The tasks are assumed to be substitutes, so the coefficient on interaction term $\gamma \in (0, 1)$.

In the current period, the developer's risk-neutral employer in a competitive market can perfectly observe talent θ and effort (e_1, e_2), leading to a compensation that equals the expected contribution, $B = A(\lambda)e_1 + e_2 + 2\theta$. In the future period, however, the wider labor market must infer the developer's talent solely from the public outputs (y_1, y_2). This inference forms the developer's reputation, defined as the market's posterior belief $\hat{\theta} = \mathbb{E}[\theta | y_1, y_2]$.

Aware of this future evaluation, the developer's total utility is:

$$U = \underbrace{A(\lambda)e_1 + e_2 + 2\theta}_{\text{Current Compensation}} + \underbrace{\mathbb{E}[\hat{\theta} | \theta, e_1, e_2]}_{\text{Value of Future Reputation}} - C(e_1, e_2).$$

Importantly, the intensity of reputational incentives is governed by the precision of the market's prior belief about talent, $\tau_\theta = 1/\sigma_\theta^2$. This parameter captures what Holmström (1999) terms "career concerns": when the prior is noisy (low τ_θ), the market updates heavily on observed signals, creating strong reputational stakes; when precise (high τ_θ), new signals barely move beliefs. Since track records accumulate over time, τ_θ maps to career stage, with juniors facing strong career concerns and seniors facing weak ones.

3.2 Equilibrium Analysis

The equilibrium is found by solving for the developer's optimal effort allocation, considering how the uninformed future market will interpret their output signals. I start with the market inference process.

Market inference process. The uninformed market infers the developer's talent by observing the outputs (y_1, y_2). As described in Holmström (1999), in a rational expectations equilibrium, the market anticipates the equilibrium efforts (e_1^*, e_2^*) and mentally subtracts them from the outputs to isolate the signals concerning talent, which are $x_1 := \theta + \lambda g + \varepsilon_1$ and $x_2 := \theta + \varepsilon_2$.

The market then assesses the quality of these signals using their precision (the inverse of the noise variance). The precision of the AI-assisted signal, $\tau_1(\lambda) = 1/(\lambda^2\sigma_g^2 + \sigma_1^2)$, is crucially decreasing in the intensity of AI use, λ. The precision of the human-centric signal, $\tau_2 = 1/\sigma_2^2$, is constant. The precision of the prior belief is τ_θ.

Under Bayesian updating with normal distributions, the market's posterior belief about talent, $\hat{\theta}$, is a weighted average of the signals. The informational weights are determined by each signal's relative contribution to the total precision, $T(\lambda) = \tau_1(\lambda) + \tau_2 + \tau_\theta$. The weights are thus given by:

$$\alpha_1(\lambda) = \frac{\tau_1(\lambda)}{T(\lambda)} \quad \text{and} \quad \alpha_2(\lambda) = \frac{\tau_2}{T(\lambda)}.$$

The key finding is that as AI use intensifies, the AI-assisted signal becomes less informative, causing the market to reduce the weight it places on this signal ($\alpha_1(\lambda)$ falls) and increase the relative weight on the stable, human-centric signal ($\alpha_2(\lambda)$ rises). Crucially, the magnitude of these weightsand thus the strength of incentivesdepends inversely on τ_θ: juniors face high weights (strong incentives), while seniors face low weights.

Equilibrium effort allocation. Anticipating the market's inference process, the developer chooses efforts (e_1, e_2) to maximize their total utility. The optimal effort levels are derived from the first-order conditions as follows:

$$\begin{aligned}\frac{\partial U}{\partial e_1}: &\quad A(\lambda)+\alpha_1(\lambda)A(\lambda)=e_1+\gamma e_2,\\ \frac{\partial U}{\partial e_2}: &\quad 1+\alpha_2(\lambda)=e_2+\gamma e_1.\end{aligned}$$

Solving the system of linear equations yields the equilibrium effort levels (e_1^*, e_2^*):

$$\begin{aligned} e_1^* &= \frac{1}{(1-\gamma^2)}\{A(\lambda)[1+\alpha_1(\lambda)]-\gamma(1+\alpha_2(\lambda))\},\\ e_2^* &= \frac{1}{(1-\gamma^2)}\{1+\alpha_2(\lambda)-\gamma A(\lambda)[1+\alpha_1(\lambda)]\},\end{aligned}$$

Comparative statics. The central question of the model is the following: does the availability of GenAI tools incentivize developers to work more on AI-assisted tasks, or does it cause them to shift focus to purely human-centric tasks to better signal their talent? To answer this, I differentiate the equilibrium efforts (e_1^*, e_2^*) with respect to λ and look at how these derivatives depend on τ_θ.

As shown in Appendix A, the model reveals a trade-off between a direct productivity channel, which encourages effort in the AI-assisted task, and a signaling channel, which creates reputational incentives that can shift effort away from it. As AI use (λ) increases, the AI-assisted output y_1 becomes a noisier signal of talent. In the next subsection, I describe the main empirical predictions from the model.

3.3 Main Results

Proposition 1 (Effort in AI-Assisted Tasks). *The effect of GenAI adoption (λ) on effort in the AI-assisted task (e_1^*) is determined by the precision of the market's belief (τ_θ). There exists a unique threshold $\tau_\theta^* > 0$ such that:*

- *For developers with noisy priors (e.g., juniors, $\tau_\theta < \tau_\theta^*$), GenAI adoption decreases effort in the assisted task ($\frac{\partial e_1^*}{\partial \lambda} < 0$).*

- *For developers with precise priors (e.g., seniors, $\tau_\theta > \tau_\theta^*$), GenAI adoption increases effort in the assisted task ($\frac{\partial e_1^*}{\partial \lambda} > 0$).*

This proposition highlights the core trade-off. The productivity gain from GenAI always encourages more effort. However, this is countered by the reputational cost of signal dilution. For juniors (low τ_θ) who face strong reputational incentives, the long-term negative signaling effect outweighs the short-term productivity gain, leading them to reduce effort in the task whose output has become a noisy measure of their talent. This is consistent with the empirical findings in Section 5.2 that junior developers engage *less* in coding than senior developers.

Proposition 2 (Effort Substitution)**.** *The effect of GenAI adoption (λ) on effort in the non-assisted task (e_2^*) is also determined by the precision of the market's belief (τ_θ). There exists a unique threshold $\tau_\theta^{**} > 0$ such that:*

- *For developers with noisy priors (e.g., juniors, $\tau_\theta < \tau_\theta^{**}$), GenAI adoption increases effort in the non-assisted task ($\frac{\partial e_2^*}{\partial \lambda} > 0$).*
- *For developers with precise priors (e.g., seniors, $\tau_\theta > \tau_\theta^{**}$), GenAI adoption reduces effort in the non-assisted task ($\frac{\partial e_2^*}{\partial \lambda} < 0$).*

This proposition predicts strategic effort substitution. As the AI-assisted task becomes a less reliable signal of ability, juniors shift their focus to the human-centric task, which now serves as a relatively clearer and more valuable signal. Instead, for seniors with low signaling motives, the negative productivity spillover dominates, causing them to reduce their creativity effort as AI makes coding more attractive. Thus, the model predicts junior developers, after GenAI adoption, will increase their efforts in creativity tasks, while the effort of senior developers becomes lower. The empirical findings in Section 5.3 are consistent with this prediction.

Proposition 3 (Signal Re-weighting Mechanism)**.** *An increase in GenAI use (λ) lowers the quality of the AI-assisted signal, causing the market to:*

1. *Decrease the informational weight on the AI-assisted signal* ($\frac{\partial \alpha_1(\lambda)}{\partial \lambda} < 0$)*.*

2. *Increase the relative informational weight on the human-centric signal* ($\frac{\partial \alpha_2(\lambda)}{\partial \lambda} > 0$)*.*

The behavioral responses in Propositions 1 and 2 are driven by a rational re-weighting of signals by the uninformed market. This proposition formalizes the underlying mechanism and predicts that the market will reward juniors who generate more signals from task 2. Hence, it means more-innovative junior developers, after GenAI adoption, may see better labor market outcomes than their coder peers. I test these predictions empirically in Section 5.

4 Data and Methodology

4.1 Data

4.1.1 GitHub Activity of U.S. Public Firms' Developers

To test the predictions developed in Section 3, I construct a comprehensive dataset on GitHub activity of developers working for U.S. public firms. I begin by linking GitHub organization accounts with firms. Following the methodology of Conti, Peukert and Roche (2021), I first collect websites of organization accounts via the GHTorrent project and the GitHub API. I then match these domains with the web URLs of U.S. public firms and their subsidiaries from Compustat or Orbis.[12] I then manually search for firms' open-source organization accounts to complement the domain-based matching.[13] Following this, I compile a comprehensive list of public repositories owned by the identified organization accounts through the GHArchive database, which records and archives timestamped public activity of GitHub repositories. In total, I match 1,281 firms with 3,314 organization accounts and 168,085 public repositories up to the year 2023.

[12] Domains that are indicative of hosting or social media services, such as "GitHub.com" and "facebook.com.", are excluded.

[13] Specifically, I query the firm names together with the term "open source" via Google to locate official web pages that list their open source projects, and search the firm names on GitHub to identify associated organization accounts.

Upon establishing a link between U.S. public firms and their respective GitHub organization accounts and public repositories, I use the GHArchive database to gather additional information on the public footprints of these repositories. Most importantly, I define firms' developers as internal contributors, i.e., individuals authorized to add code directly to firm-owned repositories. Overall, my sample spans from January 1st, 2021 to December 31st, 2023, 18 months before and after the introduction of GitHub Copilot.

4.1.2 Developer and Repository Characteristics

I use the GitHub API to collect static characteristics of developers as of March 2024. In particular, I obtain the account create date and self-reported names. I use the user account create date to calculate tenure and proxy for seniority. Figure 1 illustrates the distribution of account create month in my data. For self-reported names, I use OpenAI's API to interact with the GPT-3.5 turbo model to exclude users with account name containing "bot" or with "bot" account type identified to ensure bot accounts will not contaminate my sample. This results in 26,026 GitHub individual accounts during the sample period. I then match GitHub developers in my sample to their LinkedIn profile from Revelio Labs using their names and employers. In total, 12,858 developers are matched.

Similarly, I collect static characteristics of repositories extant as of February 2024 via GitHub API. This includes an array of attributes from descriptive repository metadata, such as programming languages and their corresponding byte sizes, to quantitative measures of community engagement, including the number of stars, watchers, and forks.

To measure project novelty, I use two complementary approaches. First, I employ an LLM-based measure that evaluates how novel or groundbreaking each repository is compared to existing solutions, based on its description and topics. This measure captures conceptual novelty as perceived by a state-of-the-art language model; see Emery et al. (2024) for methodology details. Second, as a robustness check, I construct a text-similarity-based measure following Kelly, Papanikolaou, Seru and Taddy (2021). For each repository, I compute TF-BIDF (term frequency–backward inverse document frequency) vectors using only repositories created in the prior 12 months, then calculate novelty as one minus the average cosine similarity to the ten most similar prior repositories in the same programming language. The two measures are positively correlated. See Section Internet Appendix A.4 for implementation details of the text-similarity measure.

4.1.3 Repository Value and Firm Characteristics

I estimate the forward-looking value of repositories using a stock market-based approach. Specifically, the value is calculated based on the stock market reaction within three days after a project is made public. In Emery et al. (2024), we provide methodology details and validation of the value measure. Stock return data comes from CRSP and other firm financial characteristics are obtained from Compustat.

4.2 GenAI Exposure Measure

To compare users with relatively higher *ex ante* exposure to GenAI with users with lower exposure, I leverage variation in the programming languages used by a user from June 2019 to June 2021, which ends right before the GitHub Copilot preview and one year before the introduction of GitHub Copilot to ensure that the AI exposure score does not reflect selection effects. The idea is that some languages (such as Python) benefit more from GenAI than others (such as Stata) because there are more training data in certain languages available for LLMs.[14] For each language, I assign an AI usefulness score (0-1) based on ChatGPT's assessment of how helpful GenAI coding tools are for that language. Because ChatGPT is trained on similar public data as GitHub Copilot, its assessment

[14] As GitHub officially stated, "the language support for GitHub Copilot varies depending on the volume and diversity of training data for that language." See https://docs.github.com/en/get-started/learning-about-github/github-language-support#core-languages-supported-by-github-features (Last accessed: December 2025).

plausibly reflects the underlying variation in training data availability across languages. Section Internet Appendix A.3 provides prompt details. While this paper is among the first to use ChatGPT to assign AI usefulness scores to programming languages, LLM-based scores have been largely implemented for occupational AI exposure (Eisfeldt et al., 2023; Eloundou et al., 2023; Kogan et al., 2023). Table 1 lists selected languages and their AI usefulness scores, with Python ranked first with a score of 1 and Stata and TeX ranked among the lowest with a score of 0.5. Other languages irrelevant for coding, such as CSV, do not have an AI usefulness score. Section 5.1 presents a set of validation tests of the AI usefulness score.

Because there is no directly available information on language usage over time at user-level, I take two steps to approximately measure user-level AI exposure. First, I calculate the total language byte size for each user (b_i^l) based on user activities in firm-owned repositories and the byte size of languages in each repository (b_r^l) between June 2019 and June 2021. For each repository, I calculate user's fraction of contribution of each language in terms of the user's share of "PushEvent" and then sum it up to user-language level. Specifically, I calculate:

$$b_i^l = \sum_r \frac{a_{i,r}}{\sum_j a_{j,r}} b_r^l,$$

where b_i^l is the byte size of language l contributed by user i, $a_{i,r}$ is the total number of PushEvent activity of user i in repository r, and b_r^l is byte size of language l used in repository r.

Then for each user, I calculate the weighted AI exposure score, where the weight is the byte size of a given language to the byte size of all code contribution by user i among the two-year period one year prior to the introduction of GitHub Copilot. Specifically, I construct the user-level AI exposure score as follows (using language-level AI usefulness scores):

$$s_i = \sum_l \frac{b_i^l}{\sum_l b_i^l} s^l,$$

where s_i is the weighted AI exposure score of user i, b_i^l is the byte size of language l contributed by user i, and s^l is the AI usefulness score of language l provided by ChatGPT. Lastly, I define users with s_i in the 4th quartile as having high exposure to GenAI.

Before proceeding, I address a concern about the weights in the exposure measure. One could argue that developers with less experience in an AI-assisted language benefit more, so weighting by usage share would understate their exposure. However, as Acemoglu (2024) points out, code autocompletion tools perform subtasks, but completing the overall task requires handling complementary human subtasks. Two conditions must hold for productivity gains: first, human coding costs must exceed AI costs for substitution within subtasks; second, the developer must be sufficiently specialized to complete the complementary subtasks. See Internet Appendix B for a formal model.

Table IA3 provides supporting evidence at the developer-language level: a higher usage share of AI-exposed language predicts greater coding activity, especially in non-primary languages, where developers are less efficient (satisfying the first condition) yet possess foundational skills (satisfying the second). Therefore, the assumption that a higher usage share of AI-exposed language indicates greater AI exposure within a developer is empirically supported.

4.3 Identification Strategy

I use a generalized difference-in-differences (DID) approach to study the reactions of labor productivity and innovation outcomes of firms' developers to the introduction of GitHub Copilot, a code autocompletion and chat tool powered by OpenAI's GPT models. Using the shock of GitHub Copilot's public release has several advantages. First, GitHub Copilot is designed for coding tasks and is seamlessly integrated with major IDEs (integrated development environments), making it particularly relevant and easy to use for software developers. Second, the tool was officially launched for individual developers on June 21st, 2022,[15] five months before the release of ChatGPT on November 30th, 2022. Therefore, any initial reaction observed is likely to be driven by GenAI powering job-specific coding tasks of developers rather than changes in activities of other tasks unobservable in the software development context. Third, while there was a period of technical preview since June 29th, 2021,[16] the preview was strictly limited to a small number of spots, and the tool's performance was relatively poor. The general availability of the tool can therefore serve as an ideal shock for the main purpose of this paper.

For baseline regressions, I use the following specification:

$$Y_{i,t} = \beta_1 Post_t \times AI\ Exposure_i + \mu_i + \theta_t + \epsilon_{i,t}, \tag{1}$$

where $Post_t$ indicates periods after the introduction of GitHub Copilot. Specifically, it equals one since July 2022 for monthly analysis or the third quarter of 2022 for quarterly analysis. $AI\ Exposure_i$ equals one for the group with relatively high GenAI exposure, i.e., the user's *ex ante* AI exposure score is in the fourth quartile. In addition, I include individual (μ_i) and time (θ_t) fixed effects to control for time-invariant individual characteristics and common time trends. The outcomes of interest $Y_{i,t}$, are individual-level outcomes, such as engagement in AI-assisted coding tasks or creativity tasks and job changes.

I further explore the heterogenous effects of GenAI on employees along the seniority dimensions. To do this, I conduct a triple difference-in-differences (DDD) analysis using

[15] For official announcement, see https://GitHub.blog/news-insights/product-news/GitHub-copilot-is-generally-available-to-all-developers/

[16] See https://GitHub.blog/news-insights/product-news/introducing-GitHub-copilot-ai-pair-programmer/.

the following specification:

$$
\begin{aligned}
Y_{i,t} = & \beta_1 Post_t \times AI\ Exposure_i + \beta_2 Post_t \times Char_i \\
& + \beta_3 Post_t \times AI\ Exposure_i \times Char_i + \mu_i + \theta_t + \epsilon_{i,t},
\end{aligned} \tag{2}
$$

where $Char_i$ is a dummy indicating the the characteristics of developer i. For example, the dummy for seniority equals one if the tenure of the developer on the GitHub platform, approximated based on the account's create date, is above median. The coefficient of interest is therefore β_3.

Additionally, I conduct an event-study analysis for individual-level reactions to the introduction of the GenAI. While the generalized DID approach gives an estimate of the average impact over the time horizon after the AI shock, the event-study approach allows for examining dynamic effects and checking whether the parallel trend assumption is violated or not. The event-study specification is as follows:

$$
Y_{i,t} = \sum_{l=\underline{l}+1}^{\overline{l}-1} \gamma_l D^l_{i,t} + \gamma_{\underline{l}} D^{\underline{l}}_{i,t} + \gamma_{\overline{l}} D^{\overline{l}}_{i,t} + \mu_i + \theta_t + \epsilon_{i,t}, \tag{3}
$$

where D_l are leads and lags of treatment for short-run effects, and $D^{\underline{l}}$ ($D^{\overline{l}}$) accounts for periods before $\underline{l}$ (after $\overline{l}$) periods relative to treatment for all longer-run effects. D^{-1} is omitted for normalization, that is, one month before the introduction of GitHub Copilot based on the panel frequency. For monthly analysis, I set $\underline{l} = -7$ and $\overline{l} = 13$.

Lastly, I conduct DID analysis in repeated cross-sections for project-level innovation outcomes, measured by stock-market-based project value, adoption (forks), community interest (stars), and novelty (LLM-rated). Specifically, I estimate the following:

$$\begin{aligned} Y_{r,f,t} = &\beta_1 AI\ Exposure_r + \beta_2 Post_t \times AI\ Exposure_r \\ &+ Controls_{r,f,t-1} + \alpha_f + \theta_{topic,t} + \epsilon_{r,f,t}, \end{aligned} \tag{4}$$

where $Y_{r,f,t}$ is the dependent variable for repository value, estimated through stock market reaction, adoption (forks), community interest (stars), and an LLM-based novelty score. I include team-level AI exposure ($AI\ Exposure_r$) if one project initiator has high AI exposure . I include lagged firm-year level controls, including the natural logarithms of one plus cumulative number of firm-owned repository, market capitalization, number of employees, and one plus value of patent portfolio. I also control for return on assets, R&D expenditure as a share of assets, whether R&D expenditure is missing, and innovator team size, and include firm and repository-topic-time fixed effects ($\theta_{topic,t}$).[17] The repository-topic-time fixed effects control for any topic-specific time-varying shocks, including potential GenAI-induced hype around AI-related projects. Similar to developer-level analysis described above, I exploit heterogeneity in team composition by seniority.

4.4 Summary Statistics

I provide an overview of monthly open-source activities of firm's developers before the introduction of GitHub Copilot in Table 2. First, Panel (a) shows that the average AI exposure score in my sample is around 0.81, with little difference between junior and senior developers. Coding activities account for the majority of activity records.[18] Specifically, 69% developer-month has code contribution, and an average developer contributes code around 33 times per month, showing that these developers are active contributors. De-

[17] These controls have been shown to be significant determinants of repository value as documented in Emery et al. (2024).

[18] See Section Internet Appendix A.1 for activity classification.

velopers on average contribute code to 2.9 projects per month, although they show active public footprint in 4.2 projects. Firms' developers work mostly for firm-owned projects (1.8 projects per month), but they are also active for individual projects (1.5 projects per month) and projects owned by non-firm organizations (0.8 projects per month). Exploiting heterogeneity in developer's characteristics, I show that before the introduction of the GenAI coding tool, junior developers contribute less in terms of intensity and frequency than senior developers across all types of activities, and they work on fewer projects concurrently.

Panel (b) of Table 2 compares activities and characteristics between developers with high and low exposure to GenAI. High-exposure developers contribute less code and participate in fewer projects, yet they do not differ in gender or seniority. Given that these developers are more likely to work for larger firms (see below) but display otherwise similar individual attributes, the observed differences in GitHub activity likely reflect variation in firm-level engagement on the platform rather than fundamental differences across developers.[19]

Table IA2 compares characteristics between firms with high and low exposure to GenAI, based on the average AI exposure scores of their developers active on GitHub prior to GitHub Copilot's launch. These firms show similarity across many financial dimensions, including revenue growth, profitability, foreign revenue share, leverage, interest expense to total assets ratio, and R&D expenditure as a share of total assets. High-exposure firms, however, have smaller market capitalization (significant at 10% level). Despite having fewer active developers on GitHub, high-exposure firms maintain nearly identical ratios of activities associated with senior developers (63% versus 61%). Thus, the two groups appear broadly comparable in fundamentals.

[19] In Emery et al. (2024), we show that large firms represent 89% of repositories in the sample.

5 Empirical Results

This section presents the main empirical findings. I begin by validating the AI exposure measure. I then examine GenAI's heterogeneous effects on coding and innovation activities across developer seniority levels. Finally, I test predictions from the signaling framework and analyze how these micro-level changes translate into firm value through employer-employee incentive alignment.

5.1 Validation of AI Usefulness Scores

The reliability of the AI usefulness score is essential for the empirical analysis in this paper. Anecdotal evidence suggests the score reflects how well languages integrate with GitHub Copilot, since those highlighted by GitHub, including Python, JavaScript, TypeScript, Ruby, and Go, all show very high scores (>= 0.8). To formally test its validity, I conduct three tests using GitHub code scripts, Stack Overflow data, and Reddit posts, as described below.

GitHub Copilot is designed as a code autocompletion tool, where a developer prompts LLMs with code context and instructions in comment lines. This design naturally leads to more comment lines in a file. In addition, LLMs are known for generating well-documented code. Thus, if a GenAI tool assists a language, the file will contain a higher ratio of comment lines.

To test this prediction, I examine a subsample of code scripts committed to firm repositories on GitHub. I obtain the data from the GitHub Repos dataset on BigQuery. These files are under 1 MB on the HEAD branch and were updated between January 1, 2022 and November 26, 2022. To ensure sufficient coverage, I focus on widely used languages that appear in Stack Overflows Developer Surveys. I then apply regular expressions to match and count lines with comment signs in each file and exclude files above the 95th percentile of the comment line ratio, as these typically represent license headers, configuration templates, or documentation rather than functional code. This results in more than 8 million files.

I report the results in Panel (a) of Table 3. Column (1) shows that files in languages with higher AI usefulness scores have a greater ratio of comment lines after the introduction of GitHub Copilot. An increase of 0.1 in the AI usefulness score raises the comment line ratio by 0.267 percentage point relative to the pre-treatment sample average of 6.44% (a 4.14% increase). This evidence suggests developers comment more to prompt the tool or use it to generate additional documentation. GenAI, however, does not produce longer code scripts, as shown in Columns (2)-(3). The same file size with more comment lines implies that code in AI-assisted languages has become more concise, which may indicate improved quality. Aggregating the data at the language-day level shows that more files and lines are coded in languages with higher AI usefulness scores (Columns (4)-(6)). These results align with prior studies on GitHub Copilot that document its productivity benefits.

I further validate the measure using Stack Overflow data, with results reported in Panel (b) of Table 3. Stack Overflow, the largest question-and-answer platform, is where programmers solve coding problems and learn from one another. I hypothesize that languages strongly supported by GenAI would see slower question growth, since many easy-to-medium problems can be handled by AI tools. This would leave harder, niche, or novel questions, along with low-effort posts such as pasted AI-generated code that draw little interest, leading to a higher share of unanswered questions.

As shown in Columns (1)-(3), languages with higher AI usefulness scores experience slower growth in Stack Overflow questions after the launch of GitHub Copilot, especially among widely used languages in Stack Overflow's Developer Surveys. A 0.1 increase in the AI usefulness score corresponds to a 2.49 percentage point drop in question growth (Column (1)). The no-answer rate also rises by 0.54 percentage points for the same increase in AI usefulness (Column (4)). This evidence suggests the AI usefulness score effectively captures both the impact of AI assistance and its intensity of use for a given language.

Lastly, I examine the correlation between the AI usefulness score and the number of posts mentioning a programming language in the subreddit r/ChatGPTCoding.[20] The sample period spans from December 6, 2022, to December 31, 2023. I extract posts, including both submissions and comments, that mention programming languages using regular expressions, and count the number of posts for each language.[21]

I report the results for Reddit posts in the Panels (c)-(d) of Table 3. As shown in Panel (c), the AI usefulness score is strongly correlated with the total number of posts mentioning each programming language, explaining more than 20% of the variation in mentions. The popularity of a language may lead to more posts, especially for Python, which is the primary language for AI development spurred by the recent AI hype. However, as shown in Columns (3)-(5), the correlation remains even after accounting for language usage or excluding Python. Moreover, languages without an AI usefulness score are rarely mentioned, averaging 2.19 mentions compared to 23.24 for those with an AI usefulness score. Thus, the AI usefulness score appears to be a strong indicator of a language's relevance in discussions related to coding with GenAI.

Overall, the evidence from GitHub, Stack Overflow, and Reddit confirms the validity of the AI usefulness score. On GitHub, languages with higher scores show a greater ratio of comment lines after Copilot's release, consistent with developers using comments to prompt the tool or generate documentation, while code length remains unchanged and more files created, suggesting improved conciseness and productivity. On Stack

[20] The "ChatGPTCoding" subreddit is used because it is among the top 40,000 subreddits with bulk data available from Watchful1 (n.d.). While coding with ChatGPT is not identical to coding with GitHub Copilot, both largely rely on the same large language models developed by OpenAI.

[21] To ensure the regular expressions accurately capture programming languages, I exclude those consisting of a single letter, such as R, and those with names used in everyday English. These include Text, Less, Clean, Click, Cool, Just, DM, Reason, Self, Io, Processing, Max, and BASIC.

Overflow, these languages experience slower question growth and higher no-answer rates, reflecting the shift of routine problems to AI tools. On Reddit, the AI usefulness score closely aligns with the extent to which a language appears in conversations related to coding with GenAI. Together, these results show that the AI usefulness score in this paper captures the extent and impact of AI assistance across programming languages.

5.2 AI-Assisted Tasks

In this section, I examine the impact of GenAI on the productivity of AI-assisted tasks, specifically coding, and explore how the effects vary among developers with different tenure lengths. I start by investigating the extensive margin, i.e., whether developers have any open-source coding activity related to firm-owned projects within a given month. Columns (1) and (2) of Table 4 presents results estimated from equations 1 and 2. The findings indicate that the GenAI-powered coding tool significantly increases the likelihood of coding-related events. Developers with high AI exposure are 1.16 percentage points more likely to contribute code to firm-owned projects after the introduction of GitHub Copilot.[22] Moreover, this effect is driven by senior developers: the total effect (main effect plus interaction) is 1.67 percentage points. For junior developers, the main effect alone is 0.28 percentage points and statistically insignificant.

Next, I compare the quantity of coding activity between developers with high and low AI exposure before and after the introduction of GenAI. Columns (3)-(6) of Table 4 report the results, confirming that GenAI similarly boosts coding activity within firm-owned projects, with the effect again stronger among senior developers. Based on Poisson estimates, GenAI increases expected coding events for AI-exposed seniors by about 9.2% $((e^{0.88} - 1))$ relative to AI-exposed junior developers.

Figure 2 plots the event study results for coding activity engagement related to

[22] These results also hold at the individual-language level, as reported in Table IA4.

firm-owned projects, with coefficients estimated using equation 3. The pre-launch coefficients confirm that the parallel trend assumption is not violated. Moreover, following GenAI's introduction, an immediate increase in coding activity occurs. Specifically, in the short-term, developers with high exposure become 2-4 percentage points more likely to contribute code to firm-owned projects immediately after the launch, and this elevated activity persists for up to nine months (three quarters).

Beyond parallel trends, a concern is that developers more exposed to AI differ systematically from those less exposed. Several considerations address this. First, language selection is largely exogenous, determined by job requirements, existing codebases, and project needs rather than individual choice. AI exposure is also measured pre-treatment (June 20192021), so selection cannot respond to AI availability. Second, balance checks in Panel (b) of Table 2 show that high- and low-exposure developers do not differ on observable characteristics, including gender and seniority. Third, results hold at the developer-language level with individual-time fixed effects (Table IA4), comparing the same developer across languages with different AI usefulness scores. This within-developer comparison rules out the concern that developers more exposed to AI differ from those less exposed.

5.2.1 Ruling Out Alternative Interpretations

The observed increase in coding activity could reflect factors other than genuine productivity gains. I consider two alternative explanations.

Output quality. If GenAI merely inflates output quantity at the expense of quality, the productivity interpretation would be misleading. This alternative predicts that quality-adjusted output should not increase, or that quality metrics should decline. To test this, I examine two proxies for quality: the number of stars and the number of issues opened attributed to each developer. "Starring" indicates direct community interest, whereas users open issues to report bugs or provide suggestions. Since developers naturally accumulate more stars with increased contributions, I also compute the cumulative ratio of stars per code push. Additionally, I calculate the cumulative ratio of issues opened per star, considering that popular projects typically foster more active discussions.

Table 5 reports the regression results, while Figure IA1 visualizes the event study estimates. The findings indicate that GenAI usage increases both the number of stars and issues attributable to developers, and similarly, these effects are more pronounced for senior developers. However, the stars-per-push ratio remains largely unchanged. By contrast, the ratio of issues opened per star actually decreases. This suggests that GenAI-driven productivity enhancements primarily target maintenance (fixing bugs) rather than increasing product popularity.

Working hours. Alternatively, the increase in coding activity may reflect longer working hours rather than efficiency gains. This alternative predicts that developers should work more outside common hours or on weekends. Since only the timestamp of event completion is available, I examine three specific outcomes to assess changes in input and the output-to-input ratio: work completed outside common hours, work completed on weekends, and work completed per hour, where work is defined as coding activity associated with firm-owned projects.[23] Common hours are determined as hours during which a developer completes events that constitute more than 5% of all events on a given weekday, based on activity records from 2020 of developers with at least 100 coding events.[24]

[23] Specifically, I look at the cumulative ratio of coding events occurring outside common hours ($\frac{\text{cumulative number of coding events outside common hours}_{i,t}}{\text{cumulative total number of coding events}_{i,t}}$), the cumulative ratio of coding events occurring on weekends ($\frac{\text{cumulative number of coding events on weekends}_{i,t}}{\text{cumulative total number of coding events}_{i,t}}$), and the cumulative number of coding events per hour ($\frac{\text{cumulative total number of coding events}_{i,t}}{\text{cumulative total number of hours}_{i,t}}$).

[24] For example, if a developer completes 100 coding events on Mondays throughout 2020, with only 2 at 8 pm and 5 at 2 pm, then 2 pm on Monday qualifies as a common hour, whereas 8 pm on Monday is considered outside common hours. Additionally, any hour on weekends is deemed outside common hours.

Table 6 presents the results. It appears that GenAI only weakly increases seniors' input, as measured by overtime work (Columns (1)-(4)). Consistent with increased output and unchanged input, developers complete more events per hour, as shown in Columns (5)-(6). Similar to the output results above, the effects are stronger for senior developers. The findings show that GenAI leads to efficiency gains by enabling developers to complete more work within standard working hours without increasing overtime or weekend work.

Overall, I find that after the introduction of GitHub Copilot, a coding tool powered by GenAI models, developers with high AI exposure show higher productivity, an effect not explained by more working hours or lower quality of outputs. This productivity gain is in line with the idea that GeneAI tools like GitHub Copilot reduce the cost of subtasks that complement those performed by humans (Acemoglu, 2024).

Yet, unlike many prior studies, where junior workers are more likely to adopt and benefit more from using GitHub Copilot or other GenAI tools (Brynjolfsson et al., 2023; Dell'Acqua et al., 2023; Kogan et al., 2023; Cui et al., 2024; Gambacorta, Qiu, Shan and Rees, 2024; Hoffmann et al., 2024b), this paper finds stronger effects on AI-assisted tasks among senior developers. As outlined in Section 3, this result may reflect the declining signaling value of coding for junior workers, a key reason why they contribute to open-source projects on GitHub. If so, they may shift to other signals less influenced by GenAI, such as activities related to creativity and leadership. In the next section, I examine whether GenAI exposure leads to more product innovation, particularly among junior developers.

5.3 Innovation Tasks

Improving labor productivity in AI-assisted tasks captures only part of GenAI's economic value. Beyond these direct productivity gains, GenAI may contribute to firm value and growth through two innovation channels. First, AI tools can directly assist ideation, stimulating new ideas and products (Babina et al., 2024). Second, AI-augmented humans, freed from routine work, can redirect their time and cognitive capacity toward creative activities that were previously constrained.

I study the impact of GenAI on firms' open-source innovation, focusing on the likelihood of developers becoming innovators, the community's interest in the innovation, and the value of the innovation. I define innovators as those who initiate new projects owned by the firm. Specifically, I identify innovators who publicly contribute code to newly created projects in the same month of projects' first publicly visible dates.

My analysis starts with the likelihood of producing innovation and the number of new projects initiated each quarter. Table 7 reports the results. Overall, introducing GenAI does not affect either the probability of innovation or the number of new projects. The effects also do not differ between senior and junior developers. These results suggest that the GenAI tool in this study has a limited role in generating new ideas. Additionally, more innovative developers equipped with GenAI may move to other firms and leave the sample, as discussed below in Section 5.4.

Next, I turn to project-level outcomes, starting with the private value each repository generates for its firm, estimated from stock market reactions to public releases. Repository-topic-time fixed effects are included to control for any topic-specific time-varying shocks, including potential GenAI-induced hype around AI-related projects. I then explore channels behind the value changes, including adoption (number of forks received as of February 2024), community interest (number of stars received as of February 2024), and LLM-based and text-similarity-based novelty.[25]

Table 8 reports the results. Columns (1)-(2) show how GenAI affects the firm's private value from open-source projects. After GitHub Copilot's launch, projects created by high-AI-exposure teams declined 20.3% in value (exp(-0.2262)-1). Examining team

[25] See Emery et al. (2024) for methodology details for value estimation, repository topic classification, and LLM-based novelty evaluation.

seniority reveals that senior developers drive this negative effect. By contrast, projects by all-junior teams with high AI exposure show 14% higher value. However, increasing the senior developer share by 0.2 (e.g., replacing one junior with one senior in a five-person team) reduces this positive effect by 12% (exp(-0.6414/5)-1).

Columns (3)-(6) of Table 8 provide evidence on whether improved project quality explains the value effects. After GitHub Copilot's launch, projects from high-AI-exposure teams show higher adoption rates, as measured by forks. Columns (4) shows that all-junior teams drive this effect, while increasing the senior share leads to less forking. Community interest, as proxied by stars, shows similar patterns, though coefficients remain insignificant. These results parallel the value findings, as junior teams with high AI exposure generate both higher value and higher adoption, while seniors reduce both effects. This suggests that improved project quality helps explain the value patterns observed across different team compositions.

Project quality may improve through novelty or maintenance commitment. Columns (7)-(10) of Table 8 investigate whether GenAI-equipped teams create more novel projects using two complementary measures. The LLM-based measure (Columns (7)-(8)) shows that all-junior teams with high AI exposure produce higher novelty scores, though the effect is not statistically significant. The text-similarity-based measure following Kelly et al. (2021) (Columns (9)-(10)) confirms this pattern: while the main effect is insignificant, senior developers significantly reduce novelty. Combined with earlier findings that senior developers with high AI exposure engage more in coding activities that imply better maintenance, these results suggest that the quality improvements in projects created by GenAI-exposed junior teams more likely reflect conceptual innovation than maintenance commitment.

In summary, GenAI does not increase the rate of new project initiation. However, it improves the quality of projects created by juniors. All-junior teams with high AI exposure create higher firm value, attract more community adoption, and produce more novel innovations as measured by both LLM-based and text-similarity-based approaches. Senior developers, while benefiting more from GenAI in routine productivity tasks, reduce these innovation gains. The conceptual framework in Section 3 reconciles these results through the lens of signaling incentives.

5.4 The Effects of GenAI on Signaling

As outlined in Section 3, the signaling channel may explain why junior developers engage less in coding activities, despite they might benefit more from the GenAI coding tool. However, the predictions may not be realistic if the productivity channel dominates while the signaling channel barely matters. In this section, I first examine whether signaling predictions translate into labor market outcomes. I then provide supporting evidence from developer behavior in project selection and programming language choice. After considering alternative mechanisms that could explain the observed patterns, I discuss implications for firm value creation.

5.4.1 Labor Market Evidence for the Signaling Channel

As predicted by the theoretical framework, the market will put more weight on signals from tasks less affected by GenAI and shift away from signals diluted by GenAI. To test this prediction, I compare developers' job mobility and promotions before and after the official launch of GitHub Copilot. Empirically, I match GitHub developers to their LinkedIn profiles from Revelio Labs using their names and employers. In total, 12,858 developers are matched. Table IA6 presents summary statistics on job changes among firm developers with matched GitHub profiles at the developer-year-quarter level from January 2021 to December 2023.

As shown in Panel (a), the average probability of a job move is 7% per quarter, with 3% occurring within the same firm and 4% across firms. The probability of an across-firm promotion is 3%, suggesting most developers move across firms for better opportunities rather than layoffs. Developers with longer GitHub tenure make up 65% of the sample, indicating they are more likely to create LinkedIn accounts earlier. Average job seniority is 3.21 on a 1-7 scale, and total yearly compensation averages $192,110.

Panel (b) compares developers based on their tenure and whether they are classified as innovators (i.e., those who initiated at least one GitHub project owned by their employers before the introduction of GitHub Copilot). On average, junior developers are 1.5 percentage points more likely to change job positions and 0.6 percentage points more likely to move to other firms. They also hold less senior positions with lower compensation. The differences in job mobility between innovators and non-innovators are less pronounced, though innovators tend to hold more senior positions and receive higher total compensation.

I first present evidence on the impact of GenAI on employee mobility, controlling for previous job's characteristics, individual fixed effects, firm-time fixed effects, and origin-destination-time fixed effects to control for time-varying shocks within country pairs. Panel (a) of Table 9 provides the estimates. As shown in Column (2), junior developers active on GitHub with greater AI exposure are 1.52 percentage points more likely to change jobs per quarter. However, GenAI has little impact on internal job changes, as shown in Columns (3) and (4). Instead, the effects are driven by across-firm job moves. Specifically, junior developers with greater AI exposure are 1.44 percentage points more likely to change employers per quarter (Column (6)). In contrast, senior developers show no significant effect, consistent with them having fewer career concerns.

I now compare promotion and demotion probabilities across firms for junior developers between innovators and non-innovators. The results are presented in Panel (b) of Table 9. Consistent with the signal-reweighting mechanism, junior innovators with greater AI exposure are 1.00 percentage points more likely to be promoted across firms

(Column (1)), while their non-innovative peers are 0.59 percentage points more likely to be demoted when moving across firms (Column (4)). Moreover, these effects differ by destination firm type. Junior innovators with greater AI exposure are promoted primarily by moving to private firms (Column (3)) but not to public firms (Column (2)), while non-innovative junior developers with greater AI exposure move to other public firms and get demoted (Column (5)).

Overall, the results suggest that firms value innovation signals from AI-affected developers, particularly those with less established records. This signal reweighting drives labor market segmentation: innovators sort to private firms with career advancement, while non-innovators concentrate in public firms with deteriorating positions. Consequently, signal reweighting may risk deepening inequality, as GenAI's benefits accrue primarily to developers capable of generating observable innovation signals.

5.4.2 Behavioral Evidence for the Signaling Channel

The labor market patterns documented above should be underpinned by changes in developer behavior. Signaling incentives predict that developers exposed to GenAI will increase contributions to more popular projects because the visibility and reputational payoff from working on well-known repositories is greater. However, junior developers facing higher AI-introduced noise may prioritize working with larger teams where peer monitoring moderates the distortion, making project popularity secondary. Linking to the conceptual framework (Section 3), peer monitoring (like code review) corresponds to $\sigma_g^2 \downarrow$, reducing information distortion by verifying AI-generated code quality. Table IA5 confirms these predictions. All developers work more on team projects relative to solo projects after GenAI introduction, but the underlying mechanisms differ by seniority. Senior developers drive this increase partly through preference for more popular projects (Columns (4) and (8)), which also tend to be managed by larger teams. In contrast, junior developers show no responsiveness to popularity and actually decrease contributions to solo projects, where noise reduction through peer monitoring is absent.[26]

Developer-language level analysis provides complementary evidence for the signaling channel. Table IA4 shows that senior developers increase coding activities for both familiar and new languages with high AI exposure after GenAI introduction. Junior developers, however, only increase coding activities for new languages with high AI exposure. The differences between effects on senior and junior developers substantially narrow for new languages. This can be explained by the similar signaling value that new languages offer across seniority levels, as neither group has established track records. Overall, these divergent responses across project types and programming languages provide evidence that signaling motives drive developer behavior in my sample.

5.4.3 Alternative Mechanisms

The patterns documented above, seniors increasing coding while juniors shift toward innovation, could reflect mechanisms other than signaling. I consider four alternatives.

Experience complementarity. If GenAI complements developer experience, seniors should benefit most from AI in their areas of expertise. The evidence suggests the opposite. First, within-developer gains concentrate in secondary languages, where developers have less language-specific experience (Table IA3). Second, juniors expand into new languages at rates comparable to seniors, indicating that general programming experience does not drive differential benefits. Third, this hypothesis cannot explain why juniors reduce solo project contributions, as they could simply revert to manual workflows if AI were unhelpful. These patterns suggest substitution within AI-assisted subtasks rather

[26] One might wonder why, for junior developers with high AI exposure, contributions to solo projects decrease and contributions to team projects remain unchanged, yet total contributions across all projects are also unchanged (Table 4). This occurs because the sample includes only projects that existed before GenAI introduction, since team size classifications require pre-treatment activity data. Contributions to newly created projects are therefore excluded from these measures. This pattern further implies that junior developers reallocate effort toward new projects launched after GenAI's introduction, consistent with their increased investment in innovative activities.

than experience complementarity.

Task specialization. Perhaps juniors are naturally better at innovation, and firms optimally assign them to creative tasks after GenAI arrives. If so, internal employers should also respond to these productivity differences. Instead, effects appear only in external moves, consistent with public signals informing prospective rather than current employers. Moreover, rational firms would not assign high-quality juniors to visible innovation work that facilitates their departure to competitors, yet the promotion effects concentrate in across-firm moves. Similar patterns appear in non-firm-owned projects, where firm assignment plays no role.

Skill revaluation. GenAI may have revalued pre-existing skills rather than inducing effort reallocation, with juniors already better at innovation simply becoming more valuable. This alternative predicts patterns that contradict the data. First, the junior-senior gap in innovation should exist before Copilot, yet seniors actively initiated projects pre-GenAI. Second, juniors should shift to innovation in familiar languages where they have demonstrated skills, yet the senior-junior coding gap narrows specifically in new languages where neither group has track records.

Displacement. GenAI might displace junior developers rather than induce strategic reallocation. Several patterns point to reallocation. First, there is no internal demotion, as current employers do not view these workers as less valuable. Second, juniors code at similar rates to seniors in new languages, suggesting they are not pushed out of coding entirely. That said, the juniors in my sample are mid-level developers with several years of experience, so these findings may not generalize to truly entry-level workers.

Having established evidence for signaling-driven behavior and ruled out alternative mechanisms, I now examine implications for firm value.

5.4.4 Employer-Employee Alignment and Firm Value

If the signaling channel is important, the impact of GenAI on firm value will depend on employer-employee alignment around signaling incentives. Senior developers readily exploit GenAI's efficiency gains. Instead, junior developers need to establish credibility in their career profiles and might prioritize tasks where human contributions stand out. As a result, a firm focusing on AI-assisted coding benefit more if its workforce is primarily composed of senior developers than a junior-heavy firm. In contrast, innovative firms with more junior developers benefit from the aligned incentives, since junior developers can pursue innovation tasks that match both their signaling needs and the firm's objectives. However, this alignment matters less overall because innovative firms' core business models are less affected by GenAI.

To test this hypothesis, I use an event study approach, examining cumulative abnormal returns following the official launch of GitHub Copilot. To ensure a relevant and meaningful sample, I include only firms in the information technology industry (SIC code 737) and those with more than 100 code push events up to the event date. I calculate a firm's AI exposure score based on the language composition of its repositories and classify a firm as AI-exposed if its score falls in the fourth quartile. I assess firm-developer compatibility by considering a firm's innovativeness and workforce tenure. Specifically, incentives are aligned if an innovative firm (with R&D expenditure as a share of assets above the median) has an average developer tenure in the first quartile or if a non-innovative firm has an average tenure in the fourth quartile. Thus, if the hypothesis holds, incentive-aligned firms should experience higher abnormal returns after the introduction of the GenAI coding tool.

Table 10 report the event study results. Panel (a) examines all active GitHub firms in the information technology industry. On the day of Copilot's launch, there is little market reaction, suggesting that the market took time to process information about disruptive technologies like GenAI. In the event windows from 10 days to 30 days, consistent with the hypothesis, AI-affected firms with aligned incentives experience higher cumulative abnormal returns. Panel (b) further divides the sample into innovative and non-innovative firms. The positive effect of incentive alignment is concentrated in non-innovative firms. Consistent with previous prediction, these results suggest that investors perceive firms compatible with employees' signaling incentives benefit more from GenAI tools, particularly those whose businesses are more exposed to GenAI.

Summary. The findings suggest that the signaling channel may help explain the surprisingly small change in coding activity among junior developers, who are supposed to benefit more from GenAI tools. Instead, junior developers who are able to generate observable innovation signals see better labor market outcomes. However, the observed labor market segmentation because of the GenAI-induced signal-reweighting mechanism suggests risks of deepening inequality. At the firm level, GenAI's impact on firm value is not driven solely by technological advancement but also by how well a firm aligns with signaling incentives of its employees.

6 Conclusion

This paper examines how GenAI reshapes effort allocation between AI-assisted tasks and creative work, with effects that differ by employee tenure. Using a developer-level measure of AI exposure and the launch of GitHub Copilot, I find that GenAI boosts coding productivity but primarily for senior developers. Junior developers, despite potentially benefiting more from AI augmentation, do not increase their coding activity.

One potential source where the asymmetry arises from is signaling. AI-generated code weakens the signal value of routine tasks, particularly for less experienced workers who rely on open-source contributions to demonstrate ability. In response, juniors shift toward activities less influenced by GenAI. Projects initiated by junior-heavy teams with high AI exposure generate significantly higher community adoption and private value and are measurably more novel. This suggests juniors improve innovation quality through conceptual creativity rather than implementation. Supporting the signaling mechanism, I show that senior developers increase contributions to high-visibility projects, while juniors prioritize peer-monitored settings. The tenure gap in coding activity also narrows for new programming languages where neither group has established reputations.

Labor market response is consistent with signal reweighting. Junior developers with greater AI exposure are more likely to change employers, and career outcomes diverge by innovation signals: more innovative juniors are more likely to be promoted when switching firms, while non-innovative juniors face higher demotion rates. Firms' returns to AI adoption seem to depend on workforce composition. Less-innovative firms with senior-heavy workforces experience positive abnormal returns following GitHub Copilot's introduction, as established reputations reduce signaling frictions and allow full capture of AI productivity gains on routine tasks.

While this study focuses on software developers, the underlying mechanisms likely extend to other knowledge work contexts where early-career workers rely on productive outputs to signal ability to employers. Consider junior analysts in finance building valuation models, graduate students producing polished academic writing, or associates drafting legal documents. In each case, GenAI can both boost output quantity and erode the signal value of that output by making it easier for anyone to produce competent-looking work. The tension between productivity gains and signaling dilution, and the resulting incentive to shift toward tasks where human judgment remains essential, are not unique to software development but rather a general feature to knowledge work in the age of GenAI.

The study has implications for firms and policymakers. For firms, this implies they may not fully capture AI's productivity gains on routine tasks. When AI dilutes the signaling value of routine work, junior employees, who stand to benefit most from augmentation, shift effort toward human-centric activities such as innovation. Yet firms may still benefit from greater output in these activities. In the setting of this paper, which firms gain depends on strategic alignment: innovation-focused firms can leverage junior reallocation toward creative work, while efficiency-focused firms benefit most with senior-heavy workforces whose established reputations reduce signaling pressures, allowing them to fully capture AI-assisted productivity gains.

For policymakers, the signaling perspective suggests that policies preserving the value of human work deserve consideration. Transparency requirements around AI usage, or certification that verifies human input for high-stake decisions, could help labor markets maintain the ability to screen talent. When employers can identify genuine human contributions, fair compensation and accurate talent detection become easier. However, such policies involve a trade-off: mandates that slow AI adoption may dampen productivity gains. The optimal balance depends on the relative importance of screening versus efficiency in each domain.

References

Acemoglu, Daron, “The Simple Macroeconomics of AI,” May 2024.

_ and David Autor, “Skills, Tasks and Technologies: Implications for Employment and Earnings*,” in David Card and Orley Ashenfelter, eds., *Handbook of Labor Economics*, Vol. 4, Elsevier, January 2011, pp. 1043–1171.

Alexy, Oliver, Joel West, Helge Klapper, and Markus Reitzig, “Surrendering control to gain advantage: Reconciling openness and the resource-based view of the firm,” *Strategic Management Journal*, 2018, *39* (6), 1704–1727.

Allen, Robert C., “Collective invention,” *Journal of Economic Behavior & Organization*, 1983, *4* (1), 1–24.

Arrow, Kenneth, “Economic Welfare and the Allocation of Resources for Invention,” in “The Rate and Direction of Inventive Activity: Economic and Social Factors,” Princeton University Press, 1962, pp. 609–626.

Babina, Tania, Anastassia Fedyk, Alex He, and James Hodson, “Artificial intelligence, firm growth, and product innovation,” *Journal of Financial Economics*, January 2024, *151*, 103745.

_ , _ , Alex X. He, and James Hodson, “Firm Investments in Artificial Intelligence Technologies and Changes in Workforce Composition,” June 2023.

Baird, Matthew, Carpanelli Mar, Brian Xu, and Kevin Xu, “Early Evidence on the Impact of GitHub Copilot on Labor Market Outcomes for Software Engineers,” 2024.

Beckmann, Lars, Heiner Beckmeyer, Ilias Filippou, Stefan Menze, and Guofu Zhou, “Unusual Financial Communication: ChatGPT, Earnings Calls, and Financial Markets,” January 2024.

Berger, Philip G., Wei Cai, Lin Qiu, and Cindy Xinyi Shen, “Employer and Employee Responses to Generative AI: Early Evidence,” February 2024.

Brynjolfsson, Erik, Bharat Chandar, and Ruyu Chen, “Canaries in the Coal Mine? Six Facts about the Recent Employment Effects of Artificial Intelligence,” 2025. Working paper.

_ , Danielle Li, and Lindsey R. Raymond, “Generative AI at Work,” April 2023.

Chen, Mark A. and Joanna (Xiaoyu) Wang, “Displacement or Augmentation? The Effects of AI on Workforce Dynamics and Firm Value,” April 2024.

Chen, Wilbur, Terrence Tianshuo Shi, and Suraj Srinivasan, “The Value of AI Innovations,” May 2024.

Cheng, Zhaoqi, Dokyun Lee, and Prasanna Tambe, "InnoVAE: Generative AI for Mapping Patents and Firm Innovation," March 2022.

Colliard, Jean-Edouard and Junli Zhao, "Artificial Intelligence and the Rents of Finance Workers," June 2025.

Conti, Annamaria, Christian Peukert, and Maria Roche, "Beefing IT up for your Investor? Open Sourcing and Startup Funding: Evidence from GitHub," 2021. Working paper, IE Business School, Harvard University, University of Lausanne.

Cowgill, Bo, Pablo Hernandez-Lagos, and Nataliya Langburd Wright, "Does AI Cheapen Talk? Theory and Evidence From Global Entrepreneurship and Hiring," July 2024.

Crouzet, Nicolas, Janice C. Eberly, Andrea L. Eisfeldt, and Dimitris Papanikolaou, "The Economics of Intangible Capital," *Journal of Economic Perspectives*, August 2022, *36* (3), 29–52.

Cui, Jingyi, Gabriel Dias, and Justin Ye, "Signaling in the Age of AI: Evidence from Cover Letters," September 2025. arXiv:2509.25054 [econ].

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

Dahlander, Linus and David M. Gann, "How open is innovation?," *Research Policy*, 2010, *39* (6), 699–709.

_ , _ , and Martin W. Wallin, "How open is innovation? A retrospective and ideas forward," *Research Policy*, 2021, *50* (4), 104218.

Dell'Acqua, Fabrizio, Edward McFowland, Ethan R. Mollick, Hila Lifshitz-Assaf, Katherine Kellogg, Saran Rajendran, Lisa Krayer, Franois Candelon, and Karim R. Lakhani, "Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality," September 2023.

Eisfeldt, Andrea L., Gregor Schubert, and Miao Ben Zhang, "Generative AI and Firm Values," May 2023.

Eloundou, Tyna, Sam Manning, Pamela Mishkin, and Daniel Rock, "GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models," August 2023. arXiv:2303.10130 [cs, econ, q-fin].

Emery, Logan P., Chan Lim, and Shiwei Ye, "The Private Value of Open-Source Innovation," 2024.

Fershtman, Chaim and Neil Gandal, “Direct and indirect knowledge spillovers: the social network of open-source projects,” *The RAND Journal of Economics*, March 2011, *42* (1), 70–91.

Gambacorta, Leonardo, Han Qiu, Shuo Shan, and Daniel Rees, “Generative AI and labour productivity: a field experiment on coding,” September 2024.

Goldfarb, Avi and Catherine Tucker, “Digital Economics,” *Journal of Economic Literature*, 2019, *57* (1), 3–43.

Gupta, Abhinav, Naman Nishesh, and Elena Simintzi, “Big Data and Bigger Firms: A Labor Market Channel,” October 2024.

Harhoff, Dietmar, Joachim Henkel, and Eric von Hippel, “Profiting from voluntary information spillovers: how users benefit by freely revealing their innovations,” *Research Policy*, 2003, *32* (10), 1753–1769.

Henkel, Joachim, Simone Schöberl, and Oliver Alexy, “The emergence of openness: How and why firms adopt selective revealing in open innovation,” *Research Policy*, 2014, *43* (5), 879–890.

Hoffmann, Manuel, Frank Nagle, and Yanuo Zhou, “The Value of Open Source Software,” January 2024.

_ , Sam Boysel, Frank Nagle, Sida Peng, and Kevin Xu, “Generative AI and Distributed Work: Evidence from Open Source Software,” 2024.

Holmström, Bengt, “Managerial Incentive Problems: A Dynamic Perspective,” *The Review of Economic Studies*, January 1999, *66* (1), 169–182.

_ and Paul Milgrom, “Multitask Principal-Agent Analyses: Incentive Contracts, Asset Ownership, and Job Design,” *Journal of Law, Economics, & Organization*, 1991, *7*, 24–52. Publisher: Oxford University Press.

Kelly, Bryan, Dimitris Papanikolaou, Amit Seru, and Matt Taddy, “Measuring Technological Innovation over the Long Run,” *American Economic Review: Insights*, 2021, *3* (3), 303–320.

Kim, Alex, Maximilian Muhn, and Valeri V. Nikolaev, “From Transcripts to Insights: Uncovering Corporate Risks Using Generative AI,” July 2024.

Kogan, Leonid, Dimitris Papanikolaou, Amit Seru, and Noah Stoffman, “Technological Innovation, Resource Allocation, and Growth*,” *The Quarterly Journal of Economics*, May 2017, *132* (2), 665–712.

_ , _ , Lawrence D.W. Schmidt, and Bryan Seegmiller, “Technology and Labor Displacement: Evidence from Linking Patents with Worker-Level Data,” November 2023.

Lerner, Josh and Jean Tirole, "Some Simple Economics of Open Source," *Journal of Industrial Economics*, 2002, *50* (2), 197–234.

_ **and** _ , "The Economics of Technology Sharing: Open Source and Beyond," *Journal of Economic Perspectives*, 2005, *19* (2), 99–120.

_ **and** _ , "The Economics of Technology Sharing: Open Source and Beyond," *Journal of Economic Perspectives*, June 2005, *19* (2), 99–120.

Lichtinger, Guy and Seyed Mahdi Hosseini Maasoum, "Generative AI as Seniority-Biased Technological Change: Evidence from U.S. Rsum and Job Posting Data," August 2025.

Lin, Yu-Kai and Likoebe M. Maruping, "Open Source Collaboration in Digital Entrepreneurship," *Organization Science*, 2022, *33* (1), 212–230.

Nagle, Frank, "Learning by Contributing: Gaining Competitive Advantage Through Contribution to Crowdsourced Public Goods," *Organization Science*, 2018, *29* (4), 569–587.

_ , "Open Source Software and Firm Productivity," *Management Science*, March 2019, *65* (3), 1191–1215.

Noy, Shakked and Whitney Zhang, "Experimental evidence on the productivity effects of generative artificial intelligence," *Science*, July 2023, *381* (6654), 187–192. Publisher: American Association for the Advancement of Science.

Otis, Nicholas, Rowan Clarke, Solne Delecourt, David Holtz, and Rembrand Koning, "The Uneven Impact of Generative AI on Entrepreneurial Performance," February 2024.

Ozkan, Serdar and Nicholas Sullivan, "Is AI Contributing to Rising Unemployment? Evidence from Occupational Variation," *On the Economy*, August 2025. Number: 101478 Publisher: Federal Reserve Bank of St. Louis.

Parker, Geoffrey, Marshall Van Alstyne, and Xiaoyue Jiang, "Platform Ecosystems: How Developers Invert the Firm," *MIS Quarterly*, 2017, *41* (1), 255–266.

Teece, David J., "Profiting from innovation in the digital economy: Enabling technologies, standards, and licensing models in the wireless world," *Research Policy*, 2018, *47* (8), 1367–1387.

von Hippel, Eric and Georg von Krogh, "Open Source Software and the "Private-Collective"' Innovation Model: Issues for Organization Science," *Organization Science*, 2003, *14* (2), 107–225.

Watchful1, "Subreddit comments/submissions 2005-06 to 2024-12."

Wiles, Emma and John J. Horton, "Generative AI and Labor Market Matching Efficiency," 2025. Available at SSRN.

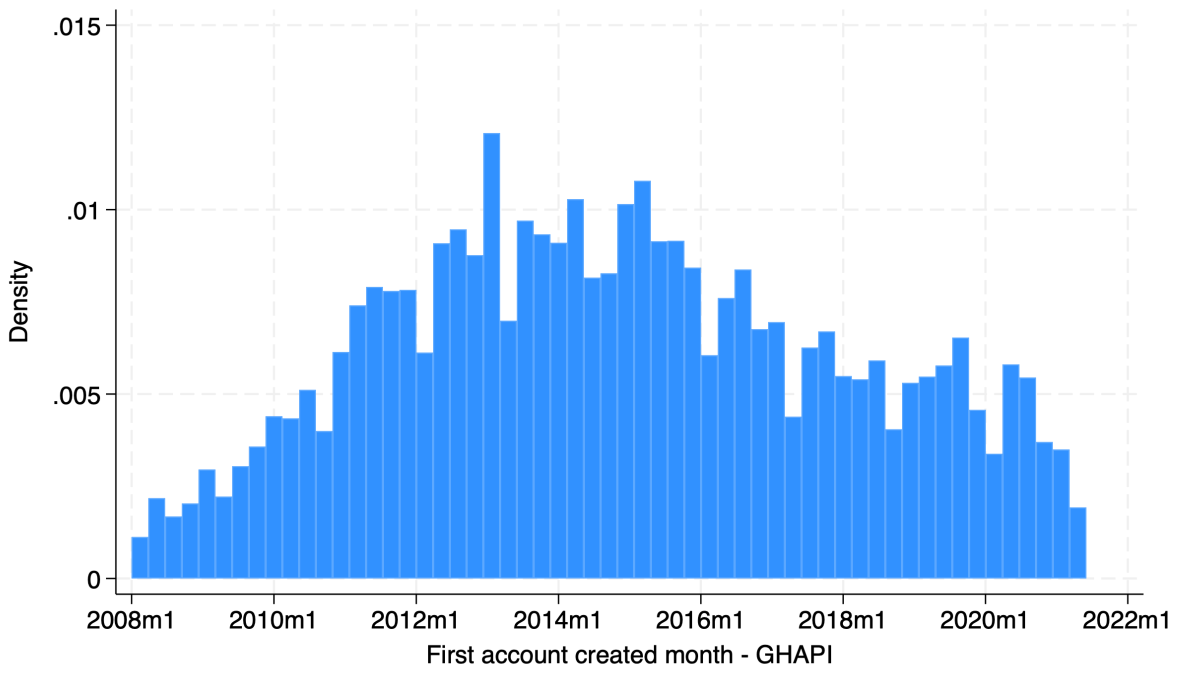

Figure 1. Density of Account Created Month

This figure plots the density of account create months of firms's developers, which is obtained via GitHub API.

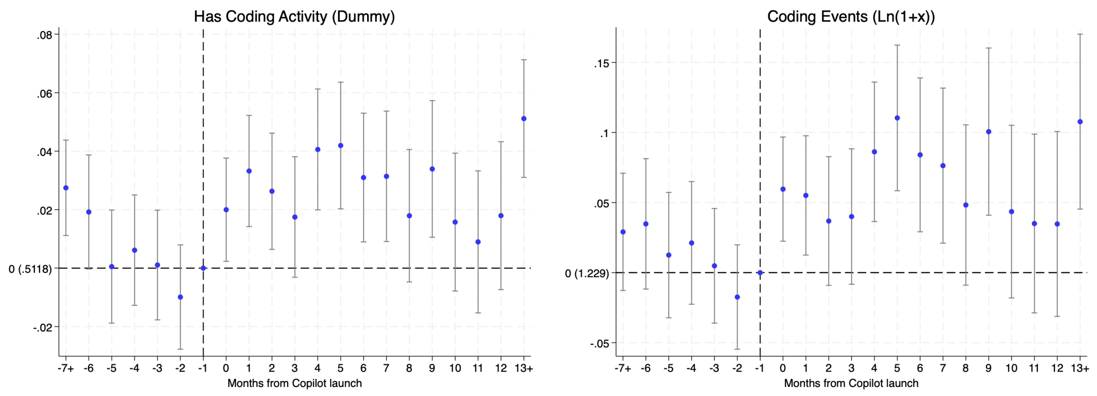

Figure 2. Firm-Related GitHub Coding Activity After Copilot Launch

This figure plots event study coefficients comparing developers with high versus low AI exposure around the GitHub Copilot launch (June 2022), with 95% confidence intervals. The specification includes individual and time fixed effects, with the month before launch as the reference period. The outcome variables are a dummy variable that equals one if a developer has any public activity in firm-owned repositories in a given month (left) and the $log(1 + x)$ transformation of the number of coding activities in firm-owned repositories in a given month (right). Standard errors are clustered at developer level.

Table 1. LLM-Based AI Usefulness Score of Selected Languages

This table lists the LLM-based AI usefulness scores of selected languages, which is later used to calculate developer-level AI exposure. The score ranges from 0 to 1. See Section Internet Appendix A.3 for the prompt used to obtain AI usefulness scores for programming languages.

High AI Exposure Languages		Low AI Exposure Languages		Random without AI Exposure
language	score	language	score	language
Python	1.0	BASIC	0.4	BrighterScript
C#	0.9	LiveScript	0.4	CSV
Java	0.9	Visual Basic 6.0	0.4	Cadence
JavaScript	0.9	ASP	0.5	DTrace
Jupyter Notebook	0.9	Cython	0.5	Futhark
TypeScript	0.9	Markdown	0.5	Inno Setup
CSS	0.8	SAS	0.5	Lex
Go	0.8	Stata	0.5	Oxygene
HTML	0.8	TeX	0.5	Self
PHP	0.8	VBA	0.5	TOML

Table 2. Summary Statistics of User-Month GitHub Activity (January 2021-June 2022)

This table presents summary statistics of user-month GitHub activity before the official launch of GitHub Copilot, i.e., from January 2021 to June 2022. Panel (a) summarizes main outcome variables used in the regression analysis by seniority. Panel (b) summarizes main outcome variables and developer characteristics by AI-exposure level. A developer is considered as senior if the tenure of the developer on the GitHub platform, approximated based on the account's create date, is above median. *High AI Exposure* is a dummy that equals one if the developer's AI exposure score is in the fourth quartile. Gender is inferred based on developer name and LLM-based gender likelihood score. A developer is considered to be male/female when the likelihood score is above 0.5. See Internet Appendix A.2 for the methodology. Activities are grouped based on their related skill requirements. See Section Internet Appendix A.1 for classification details. Repository ownership can be firm or non-firm. The latter includes repositories owned by organization accounts (*org*) or individuals (*ind*). Count variables are winsorized at 99% level by month.

(a) By Seniority

	All			Senior			Junior		
	Mean	Median	SD	Mean	Median	SD	Mean	Median	SD
Coding events	30.10	5.00	58.13	33.12	6.00	61.77	25.24	4.00	51.33
General skill events	9.30	1.00	22.88	11.19	1.00	25.46	6.26	0.00	17.53
Mixed events	5.84	0.00	14.91	6.43	0.00	15.73	4.90	0.00	13.44
Has coding	0.69	1.00	0.46	0.71	1.00	0.45	0.66	1.00	0.47
Has general	0.50	1.00	0.50	0.55	1.00	0.50	0.43	0.00	0.50
Has mixed	0.35	0.00	0.48	0.38	0.00	0.48	0.31	0.00	0.46
AI exposure from push events	0.81	0.84	0.13	0.82	0.84	0.13	0.81	0.84	0.14
Active repositories (total)	3.90	2.00	5.55	4.53	2.00	6.10	2.88	1.00	4.32
Active repositories (coding events)	2.64	1.00	3.76	2.97	2.00	4.06	2.11	1.00	3.14
Active repositories (general events)	1.26	1.00	2.05	1.50	1.00	2.28	0.88	0.00	1.54
Active repositories (mixed events)	0.61	0.00	1.06	0.68	0.00	1.13	0.50	0.00	0.92
Active repositories (total) (firm)	1.62	1.00	2.16	1.70	1.00	2.25	1.48	1.00	2.00
Active repositories (total) (org)	0.74	0.00	1.88	0.95	0.00	2.15	0.38	0.00	1.25
Active repositories (total) (ind)	1.40	0.00	2.52	1.70	1.00	2.79	0.91	0.00	1.92

(b) By GenAI Exposure

	High AI Exposure			Low AI Exposure		
	Mean	Median	SD	Mean	Median	SD
Female (inferred)	0.13	0.00	0.34	0.13	0.00	0.34
Senior developers (GHAPI)	0.60	1.00	0.49	0.62	1.00	0.48
Coding events	22.63	3.00	50.19	32.34	6.00	60.13
General skill events	6.24	0.00	17.83	10.22	1.00	24.12
Mixed events	4.38	0.00	13.00	6.28	0.00	15.42
Has coding	0.64	1.00	0.48	0.71	1.00	0.45
Has general	0.44	0.00	0.50	0.52	1.00	0.50
Has mixed	0.28	0.00	0.45	0.37	0.00	0.48
Active repositories (total) (firm)	1.30	1.00	1.81	1.71	1.00	2.24
Active repositories (total) (org)	0.59	0.00	1.63	0.78	0.00	1.94
Active repositories (total) (ind)	1.19	0.00	2.29	1.47	0.00	2.59

Table 3. Validation of AI Usefulness Scores for Programming Languages

This table presents the results of the validation tests on language-level AI usefulness scores using a subsample of GitHub code scripts (Panel (a)), Stack Overflow data (Panel (b)), and posts from the subreddit r/ChatGPTCoding (Panel (c)-(d)). Panel (a) uses non-binary files committed to firm repositories under 1 MB on the HEAD branch. The sample covers January 1, 2022 to November 26, 2022. Only languages appearing in Stack Overflow surveys are included. The file comment line ratio is trimmed at the 95% level. Columns (1)-(3) are at the file level, while Columns (4)-(6) aggregates data to the language-day level. *Comment Ratio* is the ratio of lines with comment signs of a language. *Total Lines* is the natural logarithm of code lines. *File Size* is the natural logarithm of the file byte size. For Panel (b), the Stack Overflow data is aggregated into two periods for each language: January 1, 2021 to June 21, 2022 as the pre-treatment period, and June 22, 2022 to December 2023 as the post-treatment period. The outcome variables in Columns (1)-(3) are growth rate of Stack Overflow questions of a given language. The outcome variables in Columns (4)-(6) are the share of questions created in each period but do not receive any answer as of September 2025. Columns (1) and (4) use all languages with data available. Other Columns use languages that appear in Stack Overflow's Developer Survey from 2020 to 2023. Growth variables are winsorized at 95% level. For Panel (c)-(d), posts mentioning programming languages from the subreddit r/ChatGPTCoding up to December 31, 2023, are identified using regular expressions. Languages with names commonly used in everyday English are excluded. Panel (c) examines the natural logarithmic transformation of total posts mentioning each programming language, while Panel (d) compares the summary statistics of total mentions for languages with and without an AI usefulness score. *Post* is a dummy that equals one if the time period is after June 22, 2022. *AI Score* is the raw LLM-based AI usefulness score of a language. *Language Usage Share* is the lagged share of developers indicating they use the given language extensively in Stack Overflow Developer Surveys. Standard errors are clustered at language level except for Panel (c) which uses robust standard errors. Significance: *, $p < 0.1$; **, $p < 0.05$; ***, $p < 0.01$.

(a) GitHub Code Scripts

	File Level			Day-Level Aggregate		
	Comment Ratio	Total Lines	File Size	File Count	Total Lines	File Size
	(1)	(2)	(3)	(4)	(5)	(6)
Post×AI Score	0.0267**	0.3562	0.2774	0.4507**	0.6352*	0.6737*
	(2.20)	(1.25)	(1.26)	(2.05)	(1.83)	(1.68)
N	8,466,108	8,911,683	8,911,683	21,450	21,450	21,450
Adj. R2	0.3104	0.1819	0.2116	0.8596	0.8132	0.7889
Language FE	Y	Y	Y	Y	Y	Y
Date FE	Y	Y	Y	Y	Y	Y

(b) New Questions on Stack Overflow

	Questions Growth			Share of No-Answers (%)		
	(1)	(2)	(3)	(4)	(5)	(6)
Post	0.0051	0.1163	0.1120	-0.9268	-1.3975	-1.3465
	(0.06)	(1.16)	(1.12)	(-0.56)	(-0.58)	(-0.56)
Post×AI Score	-0.2492*	-0.3926**	-0.3732**	5.4205**	5.6389	5.4044
	(-1.71)	(-2.44)	(-2.33)	(2.09)	(1.65)	(1.58)
Language Usage Share			-2.8788**			34.8038
			(-2.05)			(1.19)
N	242	124	124	242	124	124
Adj. R2	0.4074	0.6359	0.6408	0.8215	0.8907	0.8902
Language FE	Y	Y	Y	Y	Y	Y
Sample	All	SO Survey	SO Survey	All	SO Survey	SO Survey

(c) Posts From the Subreddit R/ChatGPTCoding

Dep. Var.	Ln (1+Mentions)	Ln(Total Mentions)			
Language Excluded				Ex. Python	
	(1)	(2)	(3)	(4)	(5)
AI Score	6.7237***	5.8442***	5.0322**	5.2298***	4.6553**
	(6.60)	(4.84)	(2.54)	(4.37)	(2.32)
Language Usage Share			0.0461**		0.0416**
			(2.55)		(2.21)
N	143	79	45	78	44
Adj. R2	0.3199	0.2571	0.4364	0.2066	0.3546

(d) Summary Statistics of Total Mentions From the Subreddit R/ChatGPTCoding

	Mean	Median	SD	Min	Max	N
Has AI Score	23.24	1.00	99.44	0	831	143
Missing AI Score	2.19	0.00	8.70	0	85	266
Total	9.55	0.00	59.93	0	831	409

Table 4. Firm-Related GitHub Coding Activities After Copilot Launch

This table reports regression results of equation 1 and equation 2. In Columns (1)-(2), the outcome variables are dummy variables that equal one if a developer has any public coding activity in firm-owned repositories in a given month. In Columns (3)-(4), the outcome variables are logarithm transformations of one plus the number of coding activities in firm-owned repositories in a given month. In Columns (5)-(6), the outcome variables are the number of coding activities in firm-owned repositories in a given month. Columns (1)-(4) present OLS estimates while Columns (5)-(6) use Poisson regressions. *Post* is a dummy that equals one if the time period is after July 2022 (or the third quarter of 2022). *AI Exposure* or *AI* are dummy variables that equal one if the developer's AI exposure score is in the fourth quartile. *Senior* is a dummy that equals one if the developer's tenure is above median. Standard errors are clustered at developer level. Significance: *, p < 0.1; **, p < 0.05; ***, p < 0.01.

	Has Coding Events		Ln(1+Coding Events)		Coding Events	
	(1)	(2)	(3)	(4)	(5)	(6)
Post×AI Exposure	0.0116**	0.0028	0.0479***	-0.0025	0.0310	-0.0266
	(2.39)	(0.35)	(2.82)	(-0.09)	(1.40)	(-0.71)
Post×Senior		-0.0157***		-0.0693***		-0.0719***
		(-3.37)		(-4.04)		(-3.76)
Post×AI×Senior		0.0139		0.0797**		0.0880*
		(1.39)		(2.25)		(1.90)
Total Effect (Senior)		0.0167***		0.0772***		
		(2.71)		(3.63)		
N	563,656	563,582	563,656	563,582	563,656	563,582
Adj. R2	0.4040	0.4040	0.6433	0.6434		
Pseudo R2					0.6844	0.6844
Individual FE	Y	Y	Y	Y	Y	Y
Year-Month FE	Y	Y	Y	Y	Y	Y
Regression	OLS	OLS	OLS	OLS	Poisson	Poisson

Table 5. Quality Change After Copilot Launch

This table reports regression results of equation 1 and equation 2. In Panel (a), the outcome variables are the $ln(1+x)$ transformations of the number of stars and issues opened that are associated with developers' work each month. In Panel (b), the outcome variables are the cumulative number of stars, scaled by the number of pushes, and the cumulative number of issues opened, scaled by the number of stars. *Post* is a dummy that equals one if the time period is after July 2022. *AI Exposure* or *AI* are dummy variables that equal one if the developer's AI exposure score is in the fourth quartile. *Senior* is a dummy that equals one if the developer's tenure is above median. The total effects for senior developers (sum of the coeffcients of the post treatment indicator and the interaction term) are reported underneath. Standard errors are clustered at developer level. Significance: *, p < 0.1; **, p < 0.05; ***, p < 0.01.

(a) Log Number of Stars and Issues Opened

	Ln(1+Stars)		Ln(1+Issues Opened)	
	(1)	(2)	(3)	(4)
Post×AI Exposure	0.0473***	0.0164	0.0211***	-0.0080
	(4.91)	(1.14)	(2.89)	(-0.70)
Post×Senior		-0.0477***		-0.0351***
		(-5.24)		(-4.59)
Post×AI×Senior		0.0488**		0.0461***
		(2.54)		(3.11)
Total Effect (Senior)		0.0652***		0.0381***
		(5.08)		(4.02)
N	563,877	563,803	563,877	563,803
Adj. R2	0.5883	0.5885	0.6215	0.6216
Individual FE	Y	Y	Y	Y
Year-Month FE	Y	Y	Y	Y

(b) Scaled Number of Stars and Issues Opened

	Stars Per Push		Issues Opened Per Star	
	(1)	(2)	(3)	(4)
Post×AI Exposure	0.0141	0.0001	-0.0287***	-0.0112
	(1.46)	(0.00)	(-2.81)	(-0.65)
Post×Senior		-0.0167*		0.0244**
		(-1.74)		(2.43)
Post×AI×Senior		0.0222		-0.0268
		(1.07)		(-1.26)
Total Effect (Senior)		0.0222*		-0.0381***
		(1.92)		(-2.98)
N	452,142	452,075	404,663	404,630
Adj. R2	0.9736	0.9736	0.9599	0.9599
Individual FE	Y	Y	Y	Y
Year-Month FE	Y	Y	Y	Y

Table 6. Hours Spent on Coding Activity of Firm-Owned Projects After Copilot Launch

This table reports regression results of equation 1 and equation 2. The outcome variables include the cumulative ratio of core events occurring outside common hours, the cumulative ratio of core events occurring on weekends, and the cumulative number of core events per hour. Core coding events are defined in Section Internet Appendix A.1. Common hours are defined as hours during which a developer completes events that constitute more than 5% of all events on a given weekday, based on 2020 activity records (with at least 100 events). Only activities related to firm-owned repositories are considered. *Post* is a dummy that equals one if the time period is after July 2022. *AI Exposure* or *AI* are dummy variables that equal one if the developer's AI exposure score is in the fourth quartile. *Senior* is a dummy that equals one if the developer's tenure is above median. The total effects for senior developers (sum of the coeffcients of the post treatment indicator and the interaction term) are reported underneath. Standard errors are clustered at developer level. Significance: *, p < 0.1; **, p < 0.05; ***, p < 0.01.

	Outside Common Hours		Weekends		Events Per Hour	
	(1)	(2)	(3)	(4)	(5)	(6)
Post×AI Exposure	0.0041	0.0003	0.0012	-0.0006	0.0109**	-0.0015
	(1.58)	(0.08)	(1.06)	(-0.36)	(2.02)	(-0.17)
Post×Senior		0.0002		0.0001		-0.0307***
		(0.11)		(0.07)		(-5.54)
Post×AI×Senior		0.0059		0.0031		0.0190*
		(1.12)		(1.30)		(1.70)
Total Effect (Senior)		0.0062*		0.0024		0.0176**
		(1.91)		(1.57)		(2.55)
N	165,364	165,344	509,468	509,401	509,468	509,401
Adj. R2	0.8946	0.8946	0.8067	0.8067	0.9101	0.9101
Individual FE	Y	Y	Y	Y	Y	Y
Year-Month FE	Y	Y	Y	Y	Y	Y

Table 7. Firm-Owned Open-Source Innovation Activity After Copilot Launch

This table reports regression results of equation 1 and equation 2. In Columns (1)-(2), the outcome variables are a dummy that equals one if a developer initiated at least one new firm-owned repository (project) in a given quarter. In Column (3)-(4), the outcome variables are number of newly initiated projects of a developer in a given quarter. *Post* is a dummy that equals one if the time period is after the third quarter of 2022. *AI Exposure* or *AI* are dummy variables that equal one if the developer's AI exposure score is in the fourth quartile. *Senior* is a dummy that equals one if the developer's tenure is above median. The total effects for senior developers (sum of the coeffcients of the post treatment indicator and the interaction term) are reported underneath. Standard errors are clustered at developer level. Significance: *, p < 0.1; **, p < 0.05; ***, p < 0.01.

	Initiated Project		# of Initiated Project	
	(1)	(2)	(3)	(4)
Post×AI Exposure	-0.0015	-0.0069	-0.0012	-0.0306
	(-0.47)	(-1.36)	(-0.06)	(-0.68)
Post×Senior		-0.0033		-0.0151
		(-1.13)		(-0.63)
Post×AI×Senior		0.0088		0.0475
		(1.36)		(0.95)
Total Effect (Senior)		0.0018		0.0170
		(0.45)		(1.11)
N	191,182	191,157	191,182	191,157
Adj. R2	0.1682	0.1683	0.4300	0.4300
Individual FE	Y	Y	Y	Y
Firm-Year-Quarter FE	Y	Y	Y	Y

Table 8. Value of Firm's Open-Source Innovation After Copilot Launch

This table reports regression results of equation 4. In Columns (1)-(2), the outcome variables are the natural logarithm of repository (project) value estimated based on stock-market reaction within three days of the project's public release. In Columns (3)-(4), the outcome variables are the natural logarithm of one plus the number of forks received as of February 2024. In Columns (5)-(6), the outcome variables are the natural logarithm of one plus the number of stars received as of February 2024. In Columns (7)-(8), the outcome variables are an LLM-based novelty measure. In Columns (9)-(10), the outcome variables are a text-similarity-based novelty measure following Kelly et al. (2021). See Emery et al. (2024) for methodology details for value estimation, repository topic classification, and LLM-based novelty evaluation. *Post* is a dummy that equals one if the time period is after July 2022. *AI Exposure* or *AI* are dummy variables that equals one if the AI exposure score of at least one member of the developer team is in the fourth quartile. *Senior* is the share of developers with tenure above median. Control variables include the natural logarithms of one plus cumulative number of firm-owned repository, market capitalization, number of employees, and one plus value of patent portfolio. I also control for return on assets, R&D expenditure as a share of assets, whether R&D expenditure is missing, and project team size. All firm-year control variables are one-year lagged and winsorized at 1% and 99% levels. See Table IA1 for variable descriptions and sources. Standard errors are clustered at firm level. Significance: *, $p < 0.1$; **, $p < 0.05$; ***, $p < 0.01$.

	Ln(Repo Value)		Ln(1+Forks)		Ln(1+Stars)		Novelty (LLM)		Novelty (Kelly)	
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
AI Exposure	0.0404***	0.0263	-0.0377	-0.1709	-0.0071	-0.1451	-0.0143	-0.0208	-0.0553	-0.0812
	(2.70)	(0.67)	(-0.45)	(-1.46)	(-0.05)	(-0.81)	(-0.82)	(-0.84)	(-1.62)	(-1.45)
Post×AI Exposure	-0.2262**	0.1310**	0.2108**	0.3071**	0.0616	0.1965	0.0197	0.0265*	0.0126	0.0717*
	(-2.41)	(2.14)	(2.59)	(2.31)	(0.39)	(1.21)	(1.18)	(1.84)	(0.35)	(1.86)
Senior		-0.0333		0.1657**		0.3640***		0.0240***		0.0263***
		(-1.21)		(2.09)		(3.94)		(3.43)		(5.07)
Post×Senior		0.0535		-0.0457		-0.1750**		-0.0104		-0.0046
		(0.93)		(-0.74)		(-2.39)		(-1.60)		(-0.99)
AI Exposure×Senior		0.0302		0.2108**		0.2061		0.0082		0.0474
		(0.52)		(2.07)		(1.33)		(0.47)		(1.15)
Post×AI×Senior		-0.6414**		-0.2346*		-0.2958		-0.0107		-0.1044***
		(-2.59)		(-1.73)		(-1.52)		(-0.62)		(-2.64)
N	11,955	10,123	11,955	10,123	11,955	10,123	11,951	10,119	11,706	9,955
Adj. R2	0.8824	0.8847	0.2987	0.3078	0.3819	0.4012	0.3547	0.3550	0.2333	0.2535
Firm FE	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Topic-Year-Month FE	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Table 9. Job Changes of Firm Developers on GitHub After Copilot Launch

This table reports regression results of equation 1 and 2 at the individual-year-quarter level. Panel (a) examines the impact of GenAI on developers' job changes. In Columns (1)-(2), the outcome variable equals one if a developer starts a new job in a given quarter. In Columns (3)-(4), the outcome variable is limited to internal job changes within the same firm. In Columns (5)-(6), the outcome variable is limited to job changes outside the previous employers. Panel (b) explores heterogeneous effects on promotion and demotion among junior innovators (i.e., GitHub project initiators) and junior non-innovators. This panel focuses on junior developers with below-median tenure on GitHub. Columns (2)-(3) and (5)-(6) divide the junior subsample into those moving to public and private firms. The outcome variables are either *Promotion*, which equals one if the new job position offers higher total compensation or higher seniority, or *Demotion* if the new job position has lower total compensation and a lower seniority rank. *Post* is a dummy that equals one if the time period is after 2022Q3. *AI Exposure* is a dummy variable that equals one if the language's AI exposure score is in the fourth quartile. *Senior* is a dummy that equals one if the developer's tenure is above median. *Innovator* is a dummy that equals one if the developer has initiated at least one project prior to 2022Q3. Control variables include seniority and the natural log of total compensation of the developer's previous position. Origin-Destination-Time fixed effects control for time-varying shocks within origin-destination country pairs. Standard errors are clustered at the developer level. Significance: *, p < 0.1; **, p < 0.05; ***, p < 0.01.

(a) Job Change After Copilot Launch

	Job Change		Internal Job Change		Across-Firm Job Change	
	(1)	(2)	(3)	(4)	(5)	(6)
Post×AI Exposure	0.0045	0.0152***	-0.0027	0.0008	0.0071***	0.0144***
	(1.29)	(2.61)	(-1.14)	(0.19)	(2.71)	(3.28)
Post×Senior		0.0029		-0.0007		0.0036
		(0.77)		(-0.29)		(1.27)
Post×AI×Senior		-0.0170**		-0.0055		-0.0115**
		(-2.41)		(-1.14)		(-2.14)
Total Effect (Senior)		-0.0019		-0.0047		0.0029
		(-0.44)		(-1.64)		(0.90)
N	113,501	113,501	113,501	113,501	113,501	113,501
Adj. R2	0.1382	0.1382	0.0447	0.0447	0.1871	0.1871
Individual FE	Y	Y	Y	Y	Y	Y
Firm-Time FE	Y	Y	Y	Y	Y	Y
Origin-Destination-Time FE	Y	Y	Y	Y	Y	Y
Controls	Y	Y	Y	Y	Y	Y

(b) Across-Firm Promotion and Demotion of Junior Developers

To Firms	Promotion			Demotion		
	All	Public	Private	All	Public	Private
	(1)	(2)	(3)	(4)	(5)	(6)
Post×AI Exposure	0.0038	0.0034	0.0005	0.0059**	0.0054**	0.0005
	(0.68)	(0.81)	(0.12)	(2.14)	(2.37)	(0.31)
Post×Innovator	-0.0045	-0.0033	-0.0012	-0.0006	0.0007	-0.0013
	(-1.28)	(-1.26)	(-0.48)	(-0.30)	(0.51)	(-0.86)
Post×AI×Innovator	0.0062	-0.0003	0.0065	-0.0041	-0.0060**	0.0019
	(0.86)	(-0.06)	(1.27)	(-1.05)	(-2.16)	(0.71)
Total Effect (Innovator)	0.0100**	0.0031	0.0069**	0.0018	-0.0006	0.0024
	(2.12)	(0.92)	(2.04)	(0.63)	(-0.38)	(1.07)
N	38,534	38,534	38,534	38,534	38,534	38,534
Adj. R2	0.2150	0.1970	0.2112	0.1339	0.0801	0.1657
Individual FE	Y	Y	Y	Y	Y	Y
Firm-Time FE	Y	Y	Y	Y	Y	Y
Origin-Destination-Time FE	Y	Y	Y	Y	Y	Y
Controls	Y	Y	Y	Y	Y	Y

Table 10. Cumulative Abnormal Returns After Copilot Launch

This table reports event study results after Copilot launch. Panel (a) uses the sample of firms in the information technology industry (with 3-digit SIC code as 737) that are active on the GitHub platform before the event day (with cumulative number of pushes higher than 100). Panel (b) splits the sample based on firms' innovativeness. Specifically, firms with R&D expenditure as a share of total assets higher than the median are classified as innovative firms. *AI Exposure* is calculated based on languages used in firm-owned projects. *Aligned* is a dummy that equals to one if an innovative firm has an average employees' tenure in the first quartile or or if a non-innovative firm has an average employees' tenure in the fourth quartile. Control variables include the natural logarithms of market capitalization and cumulative number of pushes, revenue growth, profitability, R&D expenditure as a share of assets, and whether R&D expenditure is missing. All firm-year control variables are one-year lagged and winsorized at 1% and 99% levels. See Table IA1 for variable descriptions and sources. Robust standard errors are used. Significance: *, $p < 0.1$; **, $p < 0.05$; ***, $p < 0.01$.

(a) Information Technology Firms Active on GitHub

	Day 0 AR	10-Day CAR	20-Day CAR	30-Day CAR
	(1)	(2)	(3)	(4)
AI Exposure	0.0064	-4.9531*	-8.9922***	-8.5442*
	(0.01)	(-1.94)	(-2.98)	(-1.94)
Aligned	1.2194	-7.2286**	-9.0036**	-8.7373*
	(1.38)	(-2.53)	(-2.28)	(-1.82)
AI Exposure×Aligned	-2.6533*	11.3800**	16.3546**	16.2955**
	(-1.67)	(2.27)	(2.50)	(1.97)
N	191	191	191	191
Adj. R2	0.0540	0.0646	0.1655	0.1522

(b) Non-Innovative vs. Innovative Tech Firms

	Non-Innovative Firms			Innovative Firms		
	10-Day CAR	20-Day CAR	30-Day CAR	10-Day CAR	20-Day CAR	30-Day CAR
	(1)	(2)	(3)	(4)	(5)	(6)
AI Exposure	-5.8487	-9.4358**	-8.2365	-4.3185	-8.4402	-8.1433
	(-1.58)	(-2.62)	(-1.24)	(-1.10)	(-1.64)	(-1.30)
Aligned	-10.1184**	-13.3066**	-11.4851	-3.5804	-2.5088	-4.5615
	(-2.09)	(-2.22)	(-1.64)	(-1.03)	(-0.47)	(-0.58)
AI Exposure×Aligned	16.6087**	21.9545**	19.9154	5.3011	7.8709	9.9377
	(2.06)	(2.17)	(1.58)	(0.87)	(0.94)	(0.80)
N	96	96	96	95	95	95
Adj. R2	0.0869	0.1948	0.1596	0.0288	0.0613	0.0463

Appendix A Theoretical Appendix

Combining the multitask signaling framework in Holmström and Milgrom (1991) and career concern dynamics in Holmström (1999), I develop a multitask signaling model to conceptualize how GenAI affects a developer's incentives to engage in different tasks. The model features a career-concerned developer who allocates effort between two tasks: an AI-assisted task (e.g., coding) and a non-assisted human-centric one (e.g., creativity). While GenAI improves the developer's immediate productivity, it also adds noise to the signal of the developer's underlying talent sent to the labor market. The developer, mindful of their long-term reputation, needs to balance the productivity gains from using GenAI against the potential dilution of their reputational capital.

Appendix A.1 Setup

The developer, tasks, and effort. Consider a developer endowed with a persistent but unobservable talent θ, which is drawn from a normal distribution with a mean of zero and variance σ_θ^2, i.e., $\theta \sim \mathcal{N}(0, \sigma_\theta^2)$. The precision of this prior belief is denoted by $\tau_\theta = 1/\sigma_\theta^2$. This precision captures career concerns as in Holmström (1999): when the prior is noisy (low τ_θ), the market updates heavily on observed signals, creating strong reputational incentives for junior developers; when the prior is precise (high τ_θ), new signals barely move beliefs, so seniors face weak reputational incentives.

The developer chooses an effort vector $(e_1, e_2) \in \mathbb{R}^2$ to allocate between two productive tasks. Task 1 represents an AI-assisted activity like coding, while task 2 represents a purely human-driven activity like creative problem-solving or innovation. Exerting effort is costly, described by the convex cost function:

$$C(e_1, e_2) = \frac{1}{2}(e_1^2 + e_2^2 + 2\gamma e_1 e_2).$$

I take the parameter $\gamma \in (0, 1)$ which implies efforts in the two tasks are substitutes.

Output signals. The two tasks yield distinct, observable outputs, (y_1, y_2), which serve as public signals of the developer's performance. The output from the AI-assisted task, y_1, is a composite of the developer's contribution and GenAI's input. I model this as a combination governed by an exogenous parameter $\lambda \in [0, 1]$, which represents the intensity of GenAI exposure:

$$y_1 = \underbrace{\big((1-\lambda)e_1 + \theta\big)}_{\text{Human contribution}} + \underbrace{\lambda(b_g e_1 + g)}_{\text{GenAI contribution}} + \varepsilon_1, \quad b_g > 1.$$

Here, $b_g > 1$ represents the productivity multiplier from GenAI. $(1-\lambda)$ is the share of pure human effort in the final output. The AI's contribution increases the marginal return to effort but also introduces AI-specific noise, g.

The output from the second task, y_2, depends solely on the developer's effort and talent:

$$y_2 = e_2 + \theta + \varepsilon_2.$$

I assume g, ε_1, and ε_2 are drawn independently from normal distributions with zero means and respective variances σ_g^2, σ_1^2, and σ_2^2. Furthermore, I assume θ, g, ε_1, and ε_2 are mutually independent.

Information structure and compensation. The model features two periods, representing the short-term compensation and long-term labor market outcomes. In the current period, the developer works for an employer who has full information, observing the developer's talent θ and effort choices (e_1, e_2). The labor market is competitive, so the current wage, B, is equal to the developer's expected total outputs. Let $A(\lambda) = 1 + \lambda(b_g - 1) > 1$:

$$B(e_1, e_2; \theta) = \mathbb{E}[y_1 + y_2 | \theta, e_1, e_2] = A(\lambda)e_1 + e_2 + 2\theta.$$

In the future period, however, the wider market cannot observe θ or (e_1, e_2). Future employers must form beliefs about the developer's talent based only on the observable output signals (y_1, y_2). Let $\hat{\theta} = \mathbb{E}[\theta | y_1, y_2]$ denote the posterior belief about the developer's talent held by future employers. The developer's future compensation will be based on this market perception of their talent $W(\hat{\theta}) = \hat{\theta}$.

Developer's objective. The developer is risk-neutral and career-concerned. They choose their effort levels (e_1, e_2) to maximize their total expected utility, which is the sum of their current wage and the value of their future reputation, net of effort costs. The developer's optimization problem is:

$$\max_{e_1, e_2} \quad \mathbb{E}[U(e_1, e_2) | \theta] = B(e_1, e_2; \theta) + \mathbb{E}[\hat{\theta} | \theta, e_1, e_2] - C(e_1, e_2).$$

Here, the developer places equal weight on current and future income. The intensity of career concerns is determined endogenously by the precision of the prior τ_θ, which dictates how responsive the market's expectation $\hat{\theta}$ is to performance.

Appendix A.2 Analysis

Market's inference process. To solve the developer's problem, I start with the market's inference process. The market observes outputs (y_1, y_2) but not talent θ or efforts (e_1, e_2). In a rational expectations equilibrium, the market correctly anticipates the developer's equilibrium effort choices, denoted (e_1^*, e_2^*). Hence, observing outputs is equivalent to observing purged signals as following:

$$\begin{aligned} x_1 &:= y_1 - A(\lambda)e_1^* = \theta + \lambda g + \varepsilon_1, \\ x_2 &:= y_2 - e_2^* = \theta + \varepsilon_2. \end{aligned}$$

The informational content of the signals about θ is captured by the precision (inverse variance) of their noise components. Let τ_θ be the precision of the prior belief about talent, and let the signal precisions be:

$$\begin{aligned} \tau_1(\lambda) &= \frac{1}{\tilde{\sigma}_1^2} \\ \tau_2 &= \frac{1}{\sigma_2^2} \end{aligned}$$

Where $\tilde{\sigma}_1^2 := \mathrm{Var}(\lambda g + \varepsilon_1) = \lambda^2 \sigma_g^2 + \sigma_1^2$.

The market's posterior belief $\hat{\theta}$ is a linear combination of the prior and the signals for Bayes-updating process with normal random variables. Let $\alpha_1(\lambda)$ and $\alpha_2(\lambda)$ be the information weights the market places on each signal for inferring talent:

$$\begin{aligned} \alpha_1(\lambda) &= \frac{\tau_1(\lambda)}{\tau_\theta + \tau_1(\lambda) + \tau_2} \\ \alpha_2(\lambda) &= \frac{\tau_2}{\tau_\theta + \tau_1(\lambda) + \tau_2} \end{aligned}$$

The posterior mean is therefore $\hat{\theta} = \alpha_1(\lambda)x_1 + \alpha_2(\lambda)x_2$. Because the prior expectation of θ is zero, it does not appear in the posterior mean. The magnitude of these weights depends critically on τ_θ: for juniors (low τ_θ), the weights α_1, α_2 are large; for seniors (high τ_θ), the weights approach zero.

Equilibrium effort allocation. The developer anticipates this updating rule. Their expected future reputation, conditional on their talent and effort choices, is:

$$\begin{aligned}\mathbb{E}[\hat{\theta}|\theta, e_1, e_2] &= \mathbb{E}\left[\alpha_1(\lambda)(y_1 - A(\lambda)e_1^*) + \alpha_2(\lambda)(y_2 - e_2^*)\middle|\theta, e_1, e_2\right] \\ &= \alpha_1(\lambda)(A(\lambda)e_1 + \theta - A(\lambda)e_1^*) + \alpha_2(\lambda)(e_2 + \theta - e_2^*) \\ &= \alpha_1(\lambda)A(\lambda)(e_1 - e_1^*) + \alpha_2(\lambda)(e_2 - e_2^*) + (\alpha_1(\lambda) + \alpha_2(\lambda))\theta.\end{aligned}$$

The developer's objective function can now be written as:

$$\begin{aligned}\max_{e_1,e_2} \quad U(e_1, e_2) = A(\lambda)e_1 + e_2 + 2\theta + [\alpha_1(\lambda)A(\lambda)(e_1 - e_1^*) + \alpha_2(\lambda)(e_2 - e_2^*)] \\ + [(\alpha_1(\lambda) + \alpha_2(\lambda))\theta] - \frac{1}{2}(e_1^2 + e_2^2 + 2\gamma e_1 e_2). \end{aligned} \tag{1}$$

The first-order conditions (FOCs), setting $e_i = e_i^*$ in equilibrium, form a system of two linear equations:

$$\begin{aligned}\frac{\partial U}{\partial e_1}: &\quad A(\lambda) + \alpha_1(\lambda)A(\lambda) - e_1 - \gamma e_2 = 0, \\ \frac{\partial U}{\partial e_2}: &\quad 1 + \alpha_2(\lambda) - e_2 - \gamma e_1 = 0.\end{aligned} \tag{2}$$

Solving the system of linear equations (2) yields the unique equilibrium effort levels (e_1^*, e_2^*).

$$\begin{aligned}e_1^* &= \frac{1}{(1-\gamma^2)}\{A(\lambda)[1 + \alpha_1(\lambda)] - \gamma(1 + \alpha_2(\lambda))\}, \\ e_2^* &= \frac{1}{(1-\gamma^2)}\{1 + \alpha_2(\lambda) - \gamma A(\lambda)[1 + \alpha_1(\lambda)]\}.\end{aligned}$$

The adoption of GenAI, parameterized by λ, influences effort allocation through two primary channels:

1. Productivity gain: An increase in λ raises the marginal return to effort in task 1, $A(\lambda) = 1 + \lambda(b_g - 1)$, since $b_g > 1$. This effect, ceteris paribus, incentivizes the developer to exert more effort in the AI-assisted task.
2. Signal dilution: An increase in λ introduces more AI-specific noise $(\lambda^2\sigma_g^2)$ into the output y_1. This reduces the precision of the signal from task 1, $\tau_1(\lambda)$, which in turn lowers the informational weight $\alpha_1(\lambda)$ the market places on that signal. The market attributes a smaller portion of y_1 to the developer's talent. This lowers the marginal reputational incentive, $\alpha_1(\lambda)$, to exert effort in task 1.

The net effect of GenAI on effort in task 1 (e_1^*) is therefore ambiguous. The productivity effect encourages more e_1, while the signal-diluting effect discourages it. The developer faces a trade-off between boosting current output and maintaining the clarity of their talent signal for future career prospects.

For task 2, an increase in λ makes it a relatively more attractive signaling instrument. As $\alpha_1(\lambda)$ falls, the relative information weight of the signal from task 2 ($\alpha_2(\lambda)$) increases. This can lead the developer to shift effort away from the AI-assisted task and towards the non-assisted task. However, the total effect also depends on the multitasking spillover between the two tasks. I derive the comparative statics in more detail in the next section.

Appendix A.3 Comparative Statics

I now analyze how the developer's equilibrium effort allocation changes with the intensity of GenAI adoption, λ, and how it depends on the developer's career stage (prior precision τ_θ). This is the central question of the model: does the availability of GenAI tools incentivize developers to work more on AI-assisted tasks, or does it cause them to shift focus to purely human-centric tasks to better signal their talent? To answer this, I differentiate the equilibrium efforts (e_1^*, e_2^*) with respect to λ.

Appendix A.3.1 The Impact of GenAI Adoption on Effort Allocation

Effect on signal precision and posterior weights. One of the mechanisms through which λ affects effort is by blurring the informational content of the developer's output. An increase in AI adoption λ introduces additional AI-specific noise, $\lambda^2\sigma_g^2$, into the coding output y_1. This directly reduces the precision of the signal from task 1, $\tau_1(\lambda)$:

$$\frac{\partial \tau_1}{\partial \lambda} = -2\lambda\sigma_g^2(\tau_1(\lambda))^2 < 0.$$

This change in precision, in turn, affects the market's posterior weights. Let the total precision $T(\lambda) := \tau_1(\lambda) + \tau_2 + \tau_\theta$, which also decreases with λ (as $T'(\lambda) = \tau_1'(\lambda)$). The derivatives of the weights are:

$$\begin{aligned}
\alpha_1'(\lambda) &= \frac{\partial}{\partial \lambda}\left(\frac{\tau_1}{T}\right) = \frac{\tau_1' T - \tau_1 T'}{T^2} = \frac{\tau_1'(T-\tau_1)}{T^2} = \frac{\tau_1'(\tau_2+\tau_\theta)}{T^2} < 0, \\
\alpha_2'(\lambda) &= \frac{\partial}{\partial \lambda}\left(\frac{\tau_2}{T}\right) = -\frac{\tau_2 T'}{T^2} = -\frac{\tau_2 \tau_1'}{T^2} > 0.
\end{aligned}$$

The intuition is simple. As GenAI makes the coding signal noisier, the market rationally reduces the weight it places on this signal for inferring talent ($\alpha_1'(\lambda) < 0$) and increases the relative weight on the stable creativity signal ($\alpha_2'(\lambda) > 0$).

The effect on coding effort (e_1^*). With these intermediate results, I differentiate the equilibrium coding effort e_1^* with respect to λ:

$$\frac{\partial e_1^*}{\partial \lambda} = \frac{1}{1-\gamma^2}\left\{ \underbrace{(b_g - 1)[1 + \alpha_1(\lambda)]}_{\substack{\text{(i) Direct Productivity Effect} \\ (+)}} + \underbrace{A(\lambda)\alpha_1'(\lambda)}_{\substack{\text{(ii) Direct Signaling Effect} \\ (-)}} \underbrace{-\gamma\alpha_2'(\lambda)}_{\substack{\text{(iii) Multitasking Spillover} \\ (-) \text{ if } \gamma > 0}} \right\}.$$

The first-order differentiation suggests three distinct channels through which GenAI influences incentives.

1. Direct productivity effect: Since the AI tool is beneficial ($b_g > 1$), an increase in λ directly raises the marginal product of coding effort, as $A'(\lambda) = b_g - 1 > 0$. This provides a direct, positive incentive to increase e_1, an effect that is amplified by the existing career-concern motive ($1 + \alpha_1(\lambda)$).
2. Direct signaling effect: As λ increases, the coding output y_1 becomes a noisier signal of the developer's talent. The market responds by placing less informational weight on it when forming beliefs about ability ($\alpha_1'(\lambda) < 0$). This lowers the marginal reputational return to coding effort, discouraging investment in e_1.
3. Multitasking spillover: As the market relies more heavily on the creativity signal y_2 ($\alpha_2'(\lambda) > 0$), the marginal reputational incentive to perform task 2 increases. Since the efforts in these two tasks are substitutes ($\gamma > 0$), the increased incentive for task 2 makes task 1 relatively more costly from an opportunity cost perspective, thereby reducing effort e_1. This spillover effect is negative.

The overall effect of GenAI adoption on coding effort is ambiguous. The productivity effect encourages more e_1, while the signal-diluting effect discourages it. The developer faces a trade-off between boosting current output and maintaining the clarity of their talent signal for future career prospects. The positive productivity effect is more likely to dominate when the productivity gain from AI ($b_g - 1$) is large or when career concerns are weak (high τ_θ). On the other hand, the negative signaling and spillover effects are more likely to dominate when career concerns are strong (low τ_θ), when the coding signal's precision degrades quickly with GenAI use (i.e., $|\alpha_1'(\lambda)|$ is large), and when the tasks are strong substitutes (γ is large).

The effect on creativity effort (e_2^*). I now turn to the non-assisted creativity task, e_2^*. Differentiating its equilibrium level with respect to λ mirrors the analysis for e_1^*, resulting in a combination of a direct re-weighting effect and two spillover effects from the coding task:

$$\frac{\partial e_2^*}{\partial \lambda} = \frac{1}{1-\gamma^2}\left\{ \underbrace{\alpha_2'(\lambda)}_{\substack{\text{(i) Signal Re-weighting} \\ (+)}} \quad \underbrace{-\gamma(b_g - 1)[1+\alpha_1(\lambda)]}_{\substack{\text{(ii) Productivity Spillover} \\ \text{(-) if } \gamma > 0}} \quad \underbrace{-\gamma A(\lambda)\alpha_1'(\lambda)}_{\substack{\text{(iii) Signaling Spillover} \\ \text{(+) if } \gamma > 0}} \right\}.$$

These three effects are as follows:

1. Signal re-weighting: As AI makes the coding signal y_1 noisier, the market rationally increases the weight on the creativity signal y_2 to infer ability ($\alpha_2'(\lambda) > 0$). This directly strengthens the marginal reputational return to creativity effort, providing a direct, positive incentive to increase e_2.
2. Productivity spillover: The increased productivity of coding effort due to AI affects the decision for e_2 through the structure of the cost function. As efforts are substitutes ($\gamma > 0$), the stronger incentive to perform task 1 (because of its higher marginal product) raises the opportunity cost of performing task 2, thus pulling effort away from it. This effect is negative.
3. Signaling spillover: The dilution of the signaling value of e_1 also creates a spillover through the cost function structure. The reduced reputational incentive for task 1 (driven by $\alpha_1' < 0$) makes task 2 a relatively more attractive channel for signaling talent. With substituting efforts ($\gamma > 0$), this incentivizes a reallocation of effort from task 1 to task 2, creating a positive effect on e_2^*.

The net effect on creativity effort e_2^* is generally ambiguous, but the analysis points towards a strategic reallocation of effort. It is likely that developers with strong signaling motives will increase their effort in the non-assisted creative task. This effort substitution represents a strategic shift towards tasks which remain clear and valuable signals.

Appendix A.3.2 The Role of Career Stage (Prior Precision)

The comparative statics in the previous section show that the developer's response to increased GenAI adoption is ambiguous. In this section, I show that the precision of the prior belief about talent, τ_θ, is a key factor that resolves this ambiguity. By analyzing how the sensitivity of effort to λ (i.e., $\frac{\partial e_i^*}{\partial \lambda}$) changes with τ_θ, one can identify thresholds that determine the direction of the developer's effort reallocation.

Impact on coding effort (e_1^*). To analyze the net effect on the AI-assisted task, I first express the derivative $\frac{\partial e_1^*}{\partial \lambda}$ as a function of τ_θ:

$$\frac{\partial e_1^*}{\partial \lambda} = \frac{1}{1-\gamma^2}\left((b_g - 1) + \left[(b_g - 1)\alpha_1(\lambda) + A(\lambda)\alpha_1'(\lambda) - \gamma\alpha_2'(\lambda) \right] \right).$$

The constant term, $(b_g - 1)$, is always positive, representing the direct productivity gain. The term in the brackets captures all reputational incentives, which depend on τ_θ. Its sign can be assumed to be negative when the signal dilution effect is strong, as the negative direct signaling effect and multitasking spillover are typically stronger than the reputational component of the productivity effect.

Crucially, the magnitude of this reputational term is strictly decreasing in τ_θ. As the prior becomes more precise (higher τ_θ), the market places less weight on current signals ($\alpha \to 0$ and $\alpha' \to 0$), causing the reputational motives to vanish.

While the exact threshold τ_θ^* is defined by a non-linear condition (since τ_θ appears in the denominator of the updating weights), the existence and uniqueness of this threshold are guaranteed by the monotonicity of the reputational incentives. By the Intermediate Value Theorem, there exists a unique level of seniority τ_θ^* such that for any $\tau_\theta > \tau_\theta^*$ (seniors), the productivity gain dominates ($\frac{\partial e_1^*}{\partial \lambda} > 0$), and for any $\tau_\theta < \tau_\theta^*$ (juniors), the signal dilution effect dominates ($\frac{\partial e_1^*}{\partial \lambda} < 0$).

Impact on creativity effort (e_2^*). A similar analysis applies to the creativity effort. The derivative is:

$$\frac{\partial e_2^*}{\partial \lambda} = \frac{1}{1-\gamma^2}\left(-\gamma(b_g - 1) + \left[\alpha_2'(\lambda) - \gamma(b_g - 1)\alpha_1(\lambda) - \gamma A(\lambda)\alpha_1'(\lambda) \right] \right).$$

Here, the constant term is negative, indicating that the productivity spillover pulls effort away from the creative task. The reputational term in the brackets, however, is generally positive. It is driven by the signal re-weighting effect and the positive signaling spillover, which together tend to outweigh the negative component from the productivity spillover.

Since the positive reputational incentives vanish as $\tau_\theta \to \infty$, the derivative is a strictly decreasing function of τ_θ. This implies a unique threshold τ_θ^{**} exists. When the prior precision is high ($\tau_\theta > \tau_\theta^{**}$), the negative productivity spillover dominates, causing developers to reduce their creativity effort ($\frac{\partial e_2^*}{\partial \lambda} < 0$). However, when the prior precision is low ($\tau_\theta < \tau_\theta^{**}$), the signaling effects dominate. Developers strategically increase their effort in the creative task ($\frac{\partial e_2^*}{\partial \lambda} > 0$), as it remains a clear signal of their talent.

In summary, the precision of the prior is the critical factor governing the developer's strategy. GenAI induces a polarization in behavior based on career stage: seniors adopt it to boost output, while juniors shy away from it (or shift focus to human-centric tasks) to preserve their reputation.

Internet Appendix A

Internet Appendix A.1 Skill-Based GitHub Activity Classification

Internet Appendix A.1.1 Prompt Used With GPT-4 Model in April 2024

Suppose you are a programmer who is active on GitHub platform. Define what may be job-specific core skills and what may be transferable general skills.

For the following GitHub events, classify them into three categories: job-specific core skills, transferable general skills, mixture of core and general skills, and others. Each event should be uniquely assigned to only one category that is the most relevant.

List of GitHub events: CommitCommentEvent, CreateEvent, DeleteEvent, ForkEvent, GollumEvent, IssueCommentEvent, IssuesEvent, MemberEvent, PublicEvent, PullRequestEvent, PullRequestReviewEvent, PullRequestReviewCommentEvent, PullRequestReviewThreadEvent, PushEvent, ReleaseEvent, SponsorshipEvent, WatchEvent

Internet Appendix A.1.2 Classification Details

Job-specific core skills

- PushEvent: Relates to pushing code to a repository, a basic GitHub operation.
- PullRequestEvent: Central to managing code contributions and integrations.
- PullRequestReviewEvent: Linked to the code review process within pull requests.

General skills

- IssueCommentEvent: Involves communication and discussion over issues.
- IssuesEvent: Engages problem-solving, managing bug reports, and feature requests

Mixture of core and general skills

- CommitCommentEvent: Tied to code reviews, requiring technical insights as well as communication skills.
- PullRequestReviewCommentEvent: Specific to commenting on code reviews in pull requests, requiring technical understanding and collaborative feedback.
- PullRequestReviewThreadEvent: Involves discussions around specific parts of a pull request, blending code-specific knowledge with teamwork and communication.

Nonskill related

- ForkEvent: Represents a user's engagement with and branching off from an existing repository to potentially contribute or alter separately.
- GollumEvent: Pertains to the management of Wiki pages on a GitHub repository.
- SponsorshipEvent: Linked to the GitHub Sponsors program, reflecting community support and funding mechanisms.

- WatchEvent: Involves starring a repository, indicating interest or following updates, more about user engagement than a direct skill.

Others

There are other related events I define as core/general in a broader sense. But they are not used in the analysis.

- Broader core activities
  - CreateEvent: Involves creating branches or tags, fundamental to version control.
  - DeleteEvent: Involves deleting branches or tags, another version control aspect.
  - ReleaseEvent: Pertains to the release of new software versions, important in software lifecycle management.
- Broader general activities
  - PublicEvent: While more of an administrative function, it also involves decision-making and policy setting regarding project visibility. (Initiate project)
  - MemberEvent: Related to teamwork and the management of repository collaborators.

Internet Appendix A.2 Name-Based Gender Inference

Internet Appendix A.2.1 Parameters for GPT Model Interaction via OpenAI's API

- `model`: gpt-3.5-turbo
- `temperature`: 0
- `system_text`: Process a list of names, extracting identifiable components and infer demographic information. Return the findings in JSON format with fields for original_str, first_name, last_name, company, type (with an inf_type among "user", "organization" and "bot", and score), gender (with an inf_gender either "female" or "male", and score), race (with an inf_race and score), ethnicity (with an inf_ethnicity and score), and country_of_origin (with an inf_origin and score). Put 'NA' for string subfields with no findings, and 0 for scores with no findings. Scores are for the confidence level of the inference and range from 0 to 1 rounded to two decimals. Score closer to 1 means the inference is certain while score closer to 0 means the inference is uncertain. The output is with 'results' as the key.
- `user_text`: ['name1', 'name2', 'name3',...]

Internet Appendix A.2.2 Example: Name-Based Inference Response

The JSON response example for a person named Bob Chen is:

```
{
  "results": [
    {
      "original_str": "Bob Chen",
      "first_name": "Bob",
      "last_name": "Chen",
      "company": "NA",
      "type": {
        "inf_type": "user",
        "score": 0.95
      },
      "gender": {
        "inf_gender": "male",
        "score": 0.85
      },
      "race": {
        "inf_race": "Asian",
        "score": 0.80
      },
      "ethnicity": {
        "inf_ethnicity": "NA",
        "score": 0
      },
      "country_of_origin": {
        "inf_origin": "United States",
        "score": 0.75
      }
    }
  ]
}
```

Internet Appendix A.3 Prompt for Language AI Usefulness Score With GPT-4 in April 2024

For the following programming languages, assign a score between 0 and 1 for its exposure to LLMs such as GitHub Copilot. Exposure is defined as to what extent are the GenAI tools helpful for programmers using these languages to complete their daily tasks. If it is not a programming language, return 'NA' for the score. Return your result in JSON format (language:score).

Language list: ['language1', 'language2', ...]

Internet Appendix A.4 Text-Similarity-Based Novelty Measure

This section describes the implementation of the text-similarity-based novelty measure following Kelly et al. (2021). The measure quantifies how distinct a repository's textual representation is from prior repositories.

Internet Appendix A.4.1 Text Preprocessing

For each repository, I construct a textual representation by combining three sources: (1) README file content, (2) repository description, and (3) topic tags. README content is collected via the GitHub API and BigQuery public dataset. The combined text undergoes the following preprocessing:

- Remove markdown formatting, HTML tags, code blocks, and URLs
- Tokenize using NLTK's word tokenizer
- Convert to lowercase and remove English stopwords
- Build vocabulary from terms appearing in at least 5 documents

The final corpus contains 133,347 repositories with 49,338 unique vocabulary terms. The mean token count per repository is 187 (median: 91).

Internet Appendix A.4.2 TF-BIDF Computation

Following Kelly et al. (2021), I compute term frequency–backward inverse document frequency (TF-BIDF) vectors. For a document d created at time t, the weight for term w is:

$$\text{TF-BIDF}_{w,d} = \text{TF}_{w,d} \times \text{BIDF}_{w,t}$$

where $\text{TF}_{w,d}$ is the normalized term frequency (count of w in d divided by document length) and the backward IDF is:

$$\text{BIDF}_{w,t} = \log\left(\frac{N_t}{1 + \text{DF}_{w,t}}\right)$$

Here, N_t is the total number of documents created before period t, and $\text{DF}_{w,t}$ is the number of prior documents containing term w. The backward specification ensures that IDF weights are computed using only information available at the time of repository creation.

Internet Appendix A.4.3 Novelty Score Calculation

For each repository, I compute backward similarity as the average cosine similarity to the ten most similar prior repositories:

$$\text{Backward Similarity}_d = \frac{1}{10} \sum_{j \in \text{Top-10}} \cos(\mathbf{v}_d, \mathbf{v}_j)$$

where $\mathbf{v}_d$ is the TF-BIDF vector for the target repository and $\mathbf{v}_j$ are vectors for prior repositories. The comparison pool consists of repositories created in the prior 12 months with the same primary programming language. When fewer than 10 same-language prior repositories exist, I fall back to comparing against all prior repositories regardless of language. This language-based comparison parallels Kelly et al.'s within-technology-class approach for patents.

The novelty score is defined as:

$$\text{Novelty}_d = 1 - \text{Backward Similarity}_d$$

Higher values indicate greater textual distinctiveness from prior work.

Internet Appendix A.4.4 Discussion on Methodology Choices

Several design choices adapt the Kelly et al. (2021)'s framework from patents to open-source repositories:

Top-10 aggregation versus all prior documents. Kelly et al. (2021) compute backward similarity as the mean cosine similarity to all prior patents within the same technology class. For GitHub repositories, most prior projects are entirely unrelated. They address different domains, use cases, and technical problems, resulting in near-zero similarity. Computing the mean across all priors yields novelty scores clustered near one with minimal variation. Restricting to the ten most similar prior repositories focuses on the relevant comparison set and produces a more discriminating measure (standard deviation of 0.19 versus 0.01 when using all priors).

12-month backward window. Kelly et al. (2021) employ various backward windows depending on the analysis horizon. I use 12 months to reflect the faster pace of software development relative to patenting, where the relevant "prior art" consists of recent rather than historical projects.

Same-language comparison pool. Comparing repositories within the same primary programming language parallels Kelly et al. (2021)'s within-technology-class approach for patents. Programming languages define distinct technical ecosystems with different conventions, libraries, and typical project structures. Cross-language comparisons would conflate genuine novelty with differences in language-specific boilerplate (e.g., Python versus Java package structures).

Text sources. README files, repository descriptions, and topic tags serve as the natural analog to patent abstracts and claims. These fields describe what the project does and its intended purpose, capturing the conceptual content relevant for novelty assessment.

Internet Appendix A.4.5 Summary Statistics

The resulting novelty measure has mean 0.76 and standard deviation 0.19. Approximately 85% of repositories are compared against same-language prior work, with the remaining 15% using the all-language fallback (primarily repositories in rare programming languages).

The correlation between this text-similarity measure and the LLM-based originality score is $\rho = 0.23$ ($p < 0.0001$), indicating that the two measures capture related but distinct aspects of novelty: the text-similarity approach emphasizes lexical distinctiveness of repository descriptions, while the LLM-based measure captures conceptual originality as assessed by a language model.

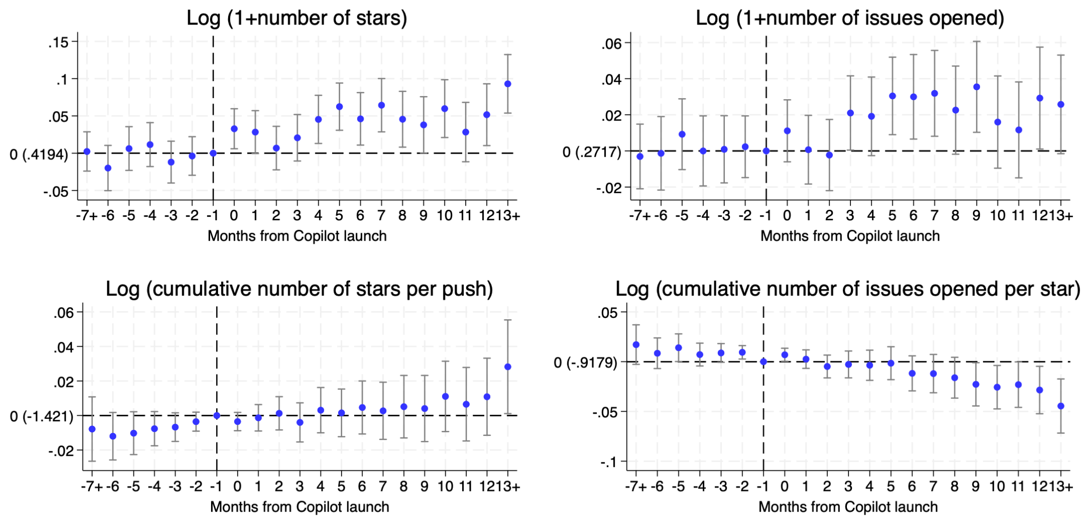

Figure IA1. Quality Change After Copilot Launch

This figure plots coefficients of the event study specification described in equation 3 with 95% confidence intervals. The outcome variables are: the $ln(1+x)$ transformations of the number of stars and issues opened that are associated with developers' work each month; the cumulative number of stars, scaled by the number of pushes, and the cumulative number of issues opened, scaled by the number of stars. Standard errors are clustered at developer level.

Table IA1. Variable Definitions

Variable	Description	Source
10/20/30-day Cumulative Abnormal Return	Cumulative abnormal return over 10/20/30 days	CRSP
AI Exposure	Dummy that equals one if the AI exposure score of the developer (0-1) is in the fourth quartile	GPT-4 and author's calculation
AI Exposure Score	The AI exposure score of a programming language	GPT-4
Abnormal Return	The difference between the actual return and the expected return, as estimated by the Fama-French 3-factor model using the Nasdaq 100 index for market return	CRSP
Across-Firm Demotion	Dummy that equals one if a developer is demoted, either by compensation or by seniority, across firms in a given quarter	Revelio
Across-Firm Job Change	Dummy that equals one if a developer changes jobs across firms in a given quarter	Revelio
Across-Firm Promotion	Dummy that equals one if a developer is promoted, either by compensation or by seniority, across firms in a given quarter	Revelio
Aligned	Dummy that equals to one if an innovative rm has an average employees' tenure in the rst quartile or or if a non-innovative rm has an average employees' tenure in the fourth quartile	Compustat
Core Event	Number of core-skill related events in a given month/quarter	GHArchive
Cumulative Nrepo	Cumulative number of repositories released by a firm prior to month t	GHArchive
Employees	Number of employees in the firm	Compustat
Firm AI Exposure	Dummy that equals one if the AI exposure score of the firm is in the fourth quartile	GPT-4 and author's calculation
Foreign Revenue Share	Share of revenue from foreign operations	Compustat
Forks	Number of forks of a repository as of February 2024	GitHub API
General Event	Number of general-skill related events in a given month/quarter	GHArchive
Has Core Event	Dummy that equals one if a developer has at least one core-skill related event in a given month	GHArchive
Has General Event	Dummy that equals one if a developer has at least one general-skill related event in a given month	GHArchive
Has Mixed Event	Dummy that equals one if a developer has at least one mixed-skill related event in a given month	GHArchive
Initiated Project	Indicator if a developer is among the innovator team of a new project in a given quarter	GHArchive
Initiator N	Number of developers in the innovator team of a new project	GHArchive
Innovative Firm	Dummy that equals one if the firm's R&D expenditure as a share of total assets is higher than the median	Compustat
Innovator	Dummy that equals one if the developer has initiated at least one project prior to 2022Q3	GHArchive

Continued

Variable	Description	Source
Innovator AI Exposure	Dummy that equals to one if the AI exposure score of at least one member of the innovator team is in the fourth quartile	GPT-4 and author's calculation
Interest Expense / Total Assets	Interest expense divided by total assets	Compustat
Internal Job Change	Dummy that equals one if a developer changes jobs within the same firm in a given quarter	Revelio
Issues Opened	Number of issues opened that are associated with a developer's work in a given month	GHArchive
Issues Opened Per Star	Cumulative number of issues opened divided by the cumulative number of stars received	GHArchive
Job Change	Dummy that equals one if a developer changes jobs in a given quarter	Revelio
Language Share	The share of a given programming language used by a developer before July 2022	GHArchive, GitHub API
Leverage	Total debt divided by total assets	Compustat
Main Language	The main programming language used by a develope before July 2022	GHArchive, GitHub API
Market Capitalization	Share price times the number of shares outstanding	CRSP
Mixed Event	Number of mixed-skill related events in a given month/quarter	GHArchive
Novelty (LLM)	An LLM-based novelty score of the repository between 0 and 1 inferred from repository information. The score measures how novel or groundbreaking a repository is compared to existing solutions, focusing on whether it introduces new ideas, techniques, or approaches	GPT-4o
Novelty (Kelly)	A text-similarity-based novelty score following Kelly et al. (2021). Computed as one minus the average cosine similarity of a repository's TF-BIDF vector to the ten most similar repositories created in the prior 12 months within the same programming language. Higher values indicate more novel repositories	GitHub API, BigQuery
Number of Cumulative Pushes	Cumulative number of pushes by a firm before July 2022	GHArchive
Number of Initiated Projects	Number of projects initiated by a developer in a given quarter	GHArchive
Patent Portfolio Value	The total estimated economic value of the patents owned by the firm using stock market returns around the patent grant date	Kogan et al. (2017)
Post	Dummy that equals one if the time period is or after July 2022 or the third quarter of 2022	
Profitability	Pre-tax income divided by total assets	Compustat
R&D Expenditure as a Share of Assets	R&D expenses divided by lagged total assets	Compustat
R&D Missing	Dummy that equals one if R&D expense is missing	Compustat

Continued

Variable	Description	Source
Repo Value	The estimated private value of the repository in 2023 USD estimated by using stock market returns around the release date of the repository	Author's calculation based on Emery et al. (2024)
Repos With Core Event	Number of repositories with core-skill related events in a given month	GHArchive
Repos With General Event	Number of repositories with general-skill related events in a given month	GHArchive
Repos With Mixed Event	Number of repositories with mixed-skill related events in a given month	GHArchive
Return on Assets	Net income divided by lagged total assets	Compustat
Revenue Growth	The growth rate of revenue	Compustat
Senior	Dummy that equals one if the developer's tenure is in the fourth quartile	GitHub API
Seniority	Seniority rank of the job position assigned by Revelio (1-7)	Revelio
Stars	Number of stars received by a repository as of February 2024	GitHub API
Stars Per Push	Cumulative number of stars received by a repository divided by the cumulative number of pushes	GHArchive
Total Compensation	Total yearly compensation in USD of a job position predicted by Revelio	Revelio
Work Completed During Weekends	Cumulative ratio of core events occurring during weekends ($\frac{\text{cumulative number of core events during weekends}_{i,t}}{\text{cumulative total number of core events}_{i,t}}$)	GHArchive
Work Completed Outside Common Hours	Cumulative ratio of core events occurring outside common hours ($\frac{\text{cumulative number of core events outside common hours}_{i,t}}{\text{cumulative total number of core events}_{i,t}}$)	GHArchive
Work Completed Per Hour	Cumulative number of core events per hour ($\frac{\text{cumulative total number of core events}_{i,t}}{\text{cumulative total number of hours}_{i,t}}$)	GHArchive

Table IA2. Summary Statistics of GitHub Firms by GenAI Exposure (Jan2021-Jun2022)

This table reports summary statistics of GitHub firms' characteristics before GitHub Copilot's official launch, from January 2021 to June 2022. The analysis includes only firms with more than 10 developer-months during this period. *High AI Exposure* is a dummy variable that equals one if the average AI exposure score of developers affiliated with a given firm falls in the fourth quartile. A developer has high AI exposure if their AI exposure score ranks in the fourth quartile among all developers. Senior developers are individuals whose GitHub platform tenure is above the median. See Table IA1 for variable descriptions.

	High AI Exposure				Low AI Exposure			
	Mean	Median	SD	N	Mean	Median	SD	N
Number of developer-months	100.58	36.00	209.96	168	691.60	79.00	3,771.73	504
AI exposure score	0.91	0.90	0.04	168	0.77	0.80	0.09	504
Developer with high AI exposure	0.62	0.66	0.36	168	0.14	0.07	0.18	504
Senior developer	0.63	0.70	0.36	168	0.61	0.65	0.30	504
Market capitalization	24985.43	5137.73	57,771.04	136	61955.95	6559.70	222426.87	393
Percentage revenue growth	19.86	16.86	20.27	137	18.58	17.87	21.05	410
Profitability	-2.42	0.91	18.44	144	-2.22	1.17	23.24	434
Foreign revenue share	0.41	0.37	0.32	124	0.45	0.43	0.31	379
Leverage	0.25	0.21	0.21	144	0.26	0.23	0.22	432
Interest expense / total assets	0.01	0.01	0.01	138	0.01	0.01	0.01	419
R&D / total assets	0.06	0.04	0.08	147	0.07	0.06	0.09	442

Table IA3. Coding Activity And Pre-Treatment Language Share Exposure

This table reports regression results of equation 2 at the individual-language level. The outcome variable is the $ln(1 + x)$ transformations of the number of pushes that are associated with developers' work in a given language each month. Columns (1) and (3) represent results for the main language a developer uses, while columns (2) and (4) include all other languages a developer uses prior to GenAI. *Post* is a dummy that equals one if the time period is after July 2022. *AI Exposure* is a dummy variable that equals one if the language's AI exposure score is in the fourth quartile. *AI Score* is the raw AI exposure score of a language. *Share* is the pre-treatment share of a language used within a developer. Main effects and other cross-interactions are included. Standard errors are clustered at developer level. Significance: *, $p < 0.1$; **, $p < 0.05$; ***, $p < 0.01$.

	Main Language	Other Languages	Main Language	Other Languages
	(1)	(2)	(3)	(4)
Post×AI Exposure	0.0203	0.0072**		
	(0.42)	(2.01)		
Post×AI Exposure×Share	0.0147	0.1959***		
	(0.21)	(3.08)		
Post×AI Score			-0.0362	0.0361***
			(-0.20)	(4.83)
Post×AI Score×Share			0.0603	0.4602***
			(0.25)	(3.30)
N	517,480	3,881,057	517,480	3,881,057
Adj. R2	0.5410	0.3517	0.5410	0.3523
Language FE	Y	Y	Y	Y
Individual-Year-Month FE	N	Y	N	Y
Individual FE	Y	N	Y	N
Year-Month FE	Y	N	Y	N

Table IA4. Coding Activity After Copilot Launch at the Individual-Language Level

This table reports regression results of equation 1 and equation 2 at the individual-language level. In Columns (1)-(4), the outcome variable is a dummy that equals one if a developer contributes code in a given language in a given quarter. In Columns (5)-(8), the outcome variable is the $ln(1+x)$ transformations of the number of coding events that are associated with developers' work in a given language each quarter. Columns (3) and (7) restrict the sample to languages a developer worked with prior to GenAI introduction, while Columns (4) and (8) include only languages new to the developer. For computational efficiency, the analysis is conducted at the quarterly frequency and includes only languages with over 10,000 coding events across the sample period. These 42 languages represent 98.8% of total coding activities. *Post* is a dummy that equals one if the time period is after July 2022. *AI Score* is the raw AI exposure score of a language. Standard errors are clustered at the developer level. Significance: *, p < 0.1; **, p < 0.05; ***, p < 0.01.

	Has Coding Events				Ln(1 + Coding Events)			
Language Type	All	All	Familiar	New	All	All	Familiar	New
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Post×AI Score	0.0555^{***}	0.0462^{***}	0.0030	0.0510^{***}	0.0439^{***}	0.0374^{***}	-0.0272^{**}	0.0318^{***}
	(25.29)	(13.52)	(0.44)	(28.58)	(22.15)	(11.93)	(-2.38)	(24.50)
Post×AI×Senior		0.0158^{***}	0.0284^{***}	0.0050^{**}		0.0113^{***}	0.0233	0.0019
		(3.55)	(3.19)	(2.07)		(2.79)	(1.64)	(1.12)
N	9,745,218	9,731,274	1,560,209	8,154,030	9,745,218	9,731,274	1,560,209	8,154,030
Adj. R2	0.2793	0.2793	0.6303	0.1431	0.1199	0.1199	0.2924	0.0428
Language FE	Y	Y	Y	Y	Y	Y	Y	Y
Individual-Year-Quarter FE	Y	Y	Y	Y	Y	Y	Y	Y

Table IA5. Coding Activity by Pre-Treatment Project Team Size and Popularity

This table reports regression results of equation equation 2. Outcome variables in Columns (1)-(4) are monthly indicators for any public coding activity in firm-owned repositories by project type. Columns (5)-(8) use the log of one plus monthly coding activities by project type. *Team* projects have on average at least two contributors monthly from January 2021 to June 2022; *Solo* projects average fewer. *High Popularity* projects have cumulative stars above the 90th percentile during January 2021 to June 2022. *Post* is a dummy that equals one if the time period is after July 2022 (or the third quarter of 2022). *AI Exposure* or *AI* are dummy variables that equal one if the developer's AI exposure score is in the fourth quartile. *Senior* is a dummy that equals one if the developer's tenure is above median. The total effects for senior developers (sum of the coefficients of the post treatment indicator and the interaction term) are reported underneath. Standard errors are clustered at developer level. Significance: *, p < 0.1; **, p < 0.05; ***, p < 0.01.

	Has Coding Events				Ln(1+Coding Events)			
Project Type	Team Size		Popularity		Team Size		Popularity	
	Solo (1)	Team (2)	Low (3)	High (4)	Solo (5)	Team (6)	Low (7)	High (8)
Post×AI Exposure	-0.0096	-0.0067	0.0044	-0.0064	-0.0325**	-0.0124	-0.0080	-0.0089
	(-1.47)	(-0.91)	(0.57)	(-0.91)	(-2.13)	(-0.47)	(-0.35)	(-0.36)
Post×Senior	-0.0167***	-0.0138***	-0.0081*	-0.0173***	-0.0359***	-0.0577***	-0.0372***	-0.0572***
	(-4.20)	(-3.15)	(-1.73)	(-4.10)	(-3.84)	(-3.51)	(-2.78)	(-3.74)
Post×AI×Senior	0.0043	0.0157*	0.0021	0.0205**	0.0262	0.0596*	0.0320	0.0761**
	(0.52)	(1.71)	(0.21)	(2.27)	(1.37)	(1.82)	(1.14)	(2.46)
Total Effect (Senior)	-0.0053	0.0090	0.0065	0.0141**	-0.0063	0.0472**	0.0240	0.0672***
	(-1.04)	(1.62)	(1.12)	(2.50)	(-0.55)	(2.44)	(1.48)	(3.55)
N	563,582	563,582	563,582	563,582	563,582	563,582	563,582	563,582
Adj. R2	0.3714	0.6025	0.4454	0.6097	0.4690	0.7280	0.5749	0.7381
Individual FE	Y	Y	Y	Y	Y	Y	Y	Y
Time FE	Y	Y	Y	Y	Y	Y	Y	Y

Table IA6. Summary Statistics of Job Changes of Firm Developers on GitHub

This table presents summary statistics on employee mobility at the individual-year-quarter level from January 2021 to December 2023. Panel (a) summarizes job changes and position characteristics of firms' developers active on GitHub. Panel (b) compares developers based on whether their tenure is above or below the median and whether they initiated a GitHub project owned by their employers before the official launch of GitHub Copilot. *Promotion* is a binary variable equal to one if the new job position offers higher compensation or a higher seniority rank.

(a) Full Sample at the Individual-Quarter Level

	Mean	SD	Min	P25	Median	P75	Max	Obs
Job Change	0.069	0.254	0.000	0.000	0.000	0.000	1.000	151,568
Across-Firm Job Change	0.043	0.204	0.000	0.000	0.000	0.000	1.000	151,568
Across-Firm Promotion	0.028	0.165	0.000	0.000	0.000	0.000	1.000	151,568
Across-Firm Demotion	0.008	0.090	0.000	0.000	0.000	0.000	1.000	151,568
Senior GitHub Developer	0.646	0.478	0.000	0.000	1.000	1.000	1.000	151,568
GitHub Project Initiator	0.554	0.497	0.000	0.000	1.000	1.000	1.000	151,568
Job Position Seniority	3.207	1.358	1.000	2.000	3.000	4.000	7.000	151,568
Total Compensation USD (000)	192.113	136.958	2.927	108.685	173.676	245.186	3,317.018	151,565

(b) By Developer Characteristics

	Senior Developer				Project Initiator			
	Yes		No		Yes		No	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Job Change	0.064	0.245	0.079	0.270	0.067	0.250	0.072	0.259
Across-Firm Job Change	0.041	0.199	0.047	0.212	0.042	0.201	0.045	0.207
Across-Firm Promotion	0.026	0.160	0.031	0.173	0.027	0.162	0.029	0.169
Across-Firm Demotion	0.008	0.091	0.008	0.088	0.008	0.091	0.008	0.088
Job Position Seniority	3.287	1.382	3.060	1.300	3.336	1.402	3.047	1.283
Total Compensation USD (000)	204.795	147.548	168.970	111.489	198.119	148.059	184.650	121.349

Internet Appendix B Economic Model on Programming Language Specialization And Productivity Gain from AI Exposure

This appendix section provides a simple economic model that explains the intuition behind the AI exposure aggregation methodology at the developer level, where a developer specialized in an AI-exposed language before the introduction of GitHub Copilot has a higher level of AI exposure than a developer who only occasionally work with the language.

Internet Appendix B.1 The Pre-AI Environment

Utility and constraints. Consider a single programmer who allocates a budget, $w > 0$, across coding activities in two distinct programming languages. Let l denote the activity in a focal language and l_2 be the activity in a second language. The programmer's preferences are represented by a Cobb-Douglas utility function:

$$U_0(l, l_2) = l^{\rho} l_2^{1-\rho}, \tag{1}$$

where $\rho \in (0, 1)$ is the parameter determining the expenditure share allocated to the focal language. The unit costs for human-written code are $c_h \geq 1$ for language l and $c_2 > 0$ for language l_2. The programmer's allocation problem is constrained by the budget:

$$c_h l + c_2 l_2 \leq w. \tag{2}$$

Initial coding quantities. Maximizing utility (1) subject to the budget constraint (2) yields the standard Marshallian demands:

$$l^0 = \frac{\rho w}{c_h} \quad \text{and} \quad l_2^0 = \frac{(1-\rho)w}{c_2}. \tag{3}$$

From these initial quantities, define the pre-AI quantity share of the focal language, s^0, as a measure of its initial prominence:

$$s^0 := \frac{l^0}{l^0 + l_2^0} = \left(1 + \frac{1-\rho}{\rho}\frac{c_h}{c_2}\right)^{-1}. \tag{4}$$

This share, $s^0 \in (0,1)$, will be a key state variable determining the effectiveness of AI adoption.

Internet Appendix B.2 The Introduction of Generative AI

I now introduce a generative AI tool (e.g., GitHub Copilot) that changes the production of code in the focal language l.

Production of effective code. After the AI introduction, services in language l become a composite good, L, produced by combining two inputs: human-written code, $l_h \geq 0$, and AI-assisted code, $l_{AI} \geq 0$. I model the aggregation of these inputs using a Constant Elasticity of Substitution (CES) production function:

$$L = \left(a_{AI}(s^0) l_{AI}^{\frac{\sigma-1}{\sigma}} + a_h(s^0) l_h^{\frac{\sigma-1}{\sigma}} \right)^{\frac{\sigma}{\sigma-1}}, \tag{5}$$

where $\sigma > 0$ ($\sigma \neq 1$) is the elasticity of substitution between human and AI inputs. The terms $a_{AI}(s^0)$ and $a_h(s^0)$ are technology weights that depend on the language's pre-AI share, s^0.

Adoption complementarity. A core feature of this model is to set the AI's effectiveness as endogenous to the prior revealed workflow intensity of the developer in the language. Formally, I assume the AI technology weight is an increasing function of the pre-AI share, $a'_{AI}(s^0) > 0$. This assumption captures the intuition that a higher initial activity share (s^0) implies greater developer specialization and a more established coding context, both of which allow for a more efficient and synergistic use of the AI assistant. I further impose a normalization on the weights:

$$a_{AI}(s^0) + a_h(s^0) \equiv 1 \quad \text{for all } s^0 \in (0,1). \tag{6}$$

This ensures that the model focuses on changes on relative effectiveness rather than arbitrary scale effects.

Post-AI optimization. The programmer's preferences across the composite good L and the other language l_2 remain Cobb-Douglas:

$$U(L, l_2) = L^\rho l_2^{1-\rho}. \tag{7}$$

The unit price of AI assistance is normalized to one ($c_{AI} = 1$), while the prices for human-written code, c_h, and the other language, c_2, remain the same. The new budget constraint is:

$$l_{AI} + c_h l_h + c_2 l_2 \leq w. \tag{8}$$

Internet Appendix B.3 Analysis

To quantify the impact of the AI tool, I first derive the effective price of the composite good L and then define the resulting productivity gain.

Effective price index. The minimal cost to produce one unit of the composite good L is given by its dual price index, P_L. Solving the cost-minimization problem for the aggregator in (5) yields:

$$P_L(s^0, c_h) = \left(a_{AI}(s^0) + (1 - a_{AI}(s^0))c_h^{1-\sigma}\right)^{\frac{1}{1-\sigma}}, \tag{9}$$

P_L represents the post-AI effective price of coding in language l.

Productivity gain. Define the productivity gain, $G(s^0)$, as the ratio of the new equilibrium quantity of effective language services, L^*, to the pre-AI quantity, l^0. Given the Cobb-Douglas structure, the total expenditure on the focal language remains constant at ρw. Therefore, the change in output is driven entirely by the change in the effective price:

$$L^* = \frac{\rho w}{P_L(s^0, c_h)} \quad \text{and} \quad l^0 = \frac{\rho w}{c_h}. \tag{10}$$

The productivity gain is thus the inverse ratio of the effective prices:

$$G(s^0) \equiv \frac{L^*}{l^0} = \frac{c_h}{P_L(s^0, c_h)}. \tag{11}$$

Substituting the expression for the price index (9) gives the final measure:

$$G(s^0) = c_h \left(a_{AI}(s^0) + (1 - a_{AI}(s^0))c_h^{1-\sigma}\right)^{-\frac{1}{1-\sigma}}. \tag{12}$$

This expression captures how the productivity gain from AI depends on the language's initial share s^0, the cost of human-written code c_h, and the elasticity of substitution σ.

Internet Appendix B.4 Results

The key question for this model is whether the productivity benefits of AI favor a language a developer is specialized in or a language a developer is less familiar with. To investigate this, I analyze how the productivity gain, $G(s^0)$, changes with the initial activity share, s^0. The main finding is that the sign of this relationship is always positive unless human cost is close to the cost of AI ($c_h \to 1$).

Proposition 1. *The productivity gain $G(s^0)$ is increasing in the initial specialization s^0 if and only if human-written code is more expensive than AI-assisted code ($c_h > 1$). Formally, for any elasticity of substitution $\sigma > 0$:*

$$\operatorname{sign}\left(\frac{dG}{ds^0}\right) = \begin{cases} > 0, & \text{if } c_h > 1, \\ = 0, & \text{if } c_h = 1. \end{cases} \tag{13}$$

Proof. I first derive the derivative of $G(s^0)$ for the general CES case ($\sigma \neq 1$) and then confirm the result for the Cobb-Douglas limit ($\sigma \to 1$).

Case 1: General CES ($\sigma \neq 1$). Recall that the productivity gain is $G(s^0) = c_h B(s^0)^{-1/(1-\sigma)}$, where the base term is $B(s^0) \equiv a_{AI}(s^0) + \big(1 - a_{AI}(s^0)\big)c_h^{1-\sigma}$. The derivative of the base term with respect to s^0 is:

$$\frac{dB}{ds^0} = a'_{AI}(s^0) - a'_{AI}(s^0)c_h^{1-\sigma} = a'_{AI}(s^0)\big(1 - c_h^{1-\sigma}\big). \tag{14}$$

Use the chain rule to differentiate $G(s^0)$:

$$\begin{aligned} \frac{dG}{ds^0} &= c_h \cdot \left(-\frac{1}{1-\sigma}\right) B(s^0)^{-\frac{1}{1-\sigma}-1} \cdot \frac{dB}{ds^0} \\ &= -\frac{c_h}{1-\sigma} B(s^0)^{-\frac{2-\sigma}{1-\sigma}} \cdot a'_{AI}(s^0)\big(1 - c_h^{1-\sigma}\big). \end{aligned} \tag{15}$$

Since $c_h > 0$, $B(s^0)$ is a positive base raised to a power, and the adoption complementarity assumption states $a'_{AI}(s^0) > 0$, the sign of the derivative is determined entirely by the product of two terms:

$$\operatorname{sign}\left(\frac{dG}{ds^0}\right) = \operatorname{sign}\left(-\frac{1}{1-\sigma}\right) \cdot \operatorname{sign}\big(1 - c_h^{1-\sigma}\big). \tag{16}$$

If $1 - \sigma > 0$ (i.e., $\sigma < 1$), the first term is negative, and the second term is positive if $c_h < 1$ and negative if $c_h > 1$. If $1 - \sigma < 0$ (i.e., $\sigma > 1$), the first term is positive, and the second term is positive if $c_h > 1$ and negative if $c_h < 1$. In both instances, the product is positive if and only if $c_h > 1$.

Case 2: Cobb-Douglas limit ($\sigma \to 1$). In the limiting case where the production aggregator is Cobb-Douglas, the productivity gain is $G(s^0) = c_h^{a_{AI}(s^0)}$. Its derivative is:

$$\frac{dG}{ds^0} = c_h^{a_{AI}(s^0)} \cdot a'_{AI}(s^0) \cdot \ln c_h, \tag{17}$$

which is always nonnegative under the assumption $c_h \geq 1$, confirming the result. $\square$

The intuition is as follows. The primary source of productivity gain in this model is the opportunity to substitute away from an expensive input (human labor, c_h) toward a cheap one (AI assistance, $c_{AI} = 1$). The core assumption of adoption complementarity ($a'_{AI}(s^0) > 0$) means that developers who are already specialized in a language are more effective at using the AI tool and thus better at making this substitution than other languages they use.

However, if human labor is almost as cheap as AI ($c_h \to 1$), the economic incentive is to use humans, not the AI substitute. Thus, even if the adoption complementarity is very high, the benefit from substituting away is limited.