Artificial Intelligence in the Firm

Fiona Chen
Harvard University

James Stratton
Harvard University

January 7, 2026
Preliminary

Abstract

How do generative AI technologies affect productivity, task assignment, and employment? We study the effects of GitHub Copilot and Cursor, two generative AI software development tools, using a novel proprietary dataset covering approximately 200 million work events of 100,000 engineers at 500 firms. We utilize a staggered difference-in-differences strategy, exploiting variation in timing of firm-level adoption of the technology. AI adoption leads to moderate increases in productivity, measured by code activity and task completion speed, without measurable declines in quality. However, these productivity gains do not pass through to effects on output, task composition, or employment.

1 Introduction

A significant body of research has examined the effects of new technologies on labor market outcomes. In the canonical model of automation, new technologies automate tasks that are more routinized and codifiable, and they augment labor in tasks that are less routine.[1] Generative AI tools fundamentally differ from previous technologies in their capacity to perform a wide range of non-routine, cognitive tasks. As a result, many researchers and policymakers have expressed significant excitement as well as concern about how these tools will shift the organization of work.[2]

In ongoing work, we examine the short-run effects of AI technologies on productivity, task assignment, and employment among software engineers. Specifically, we study the firm-level adoption of GitHub Copilot and Cursor, two generative AI software engineering tools. These AI tools provide engineers with real-time code suggestions, with the stated goal of increasing coders' productivity and freeing up time for creative and complex tasks (Perkel, 2025). Both are industry-standard tools: for instance, Microsoft announced in mid-2025 that Copilot had more than 20 million users, including more than 90% of Fortune 100 companies (Zeff, 2025).

Emails: `fionachen@g.harvard.edu` and `jstratton@g.harvard.edu`. The conclusions of this study are those of the authors and do not represent the views of Jellyfish. The results displayed are based on data that has been aggregated and anonymized and do not identify individual Jellyfish customers. This research was supported by funding from the Stripe Economics of AI Fellowship and Harvard Kennedy School's James M. and Cathleen D. Stone Program in Wealth Distribution, Inequality, and Social Policy. We are grateful to have received comments from many individuals, including Lawrence Katz, Jesse Shapiro, Amanda Pallais, Raj Chetty, and Sydnee Caldwell; as well as Andrew Lau, David Gourley, Nicholas Arcolano, Lena Chretien, and the Jellyfish team; and seminar participants at Harvard University, Opportunity Insights, the Machine Learning in Economics Summer Conference at the University of Chicago, MIT Computer Science and AI Laboratory (CSAIL), and the Berkeley Artificial Intelligence Research (BAIR) Lab.

[1]See, for instance, Zeira (1998); Autor et al. (2003, 2006); Acemoglu & Autor (2011); Goos et al. (2014); Acemoglu & Restrepo (2018b, 2019, 2022b).

[2]See, for instance, Brynjolfsson & Li (2024); Korinek & Suh (2024); Brynjolfsson, Korinek, & Agrawal (2025); Agrawal et al. (2025); Stevenson (2025); Chatterji et al. (2025).

A survey in 2025 of software engineers showed that 67.9% of software developers used GitHub Copilot (the second most commonly used, behind ChatGPT) (Overflow, 2025).

Software engineering is a natural context to study the impacts of AI tools for two reasons. First, software engineers represent a substantial share of the U.S. workforce: around 2.1 million workers were employed as software or web developers, programmers, or testers in 2024, an increase of almost 50% over the prior decade (U.S. Bureau of Labor Statistics, 2024). Second, prior work indicates that an especially large share of tasks completed by software engineers can be completed by current AI tools[3] (Kinder et al., 2024; Eloundou et al., 2024), and software engineering has been among the occupations with the most rapid diffusion of generative AI tools (Bick et al., 2024).

We use a novel proprietary dataset from Jellyfish ("JF"), a software firm whose main product is an analytics platform for companies to understand the activity of their engineering teams. Our data provides granular information on roughly 200 million work events of 100,000 workers across 500 firms, from January 2022 to June 2025. JF integrates data from five sources: (1) HR information; (2) version control systems such as GitHub, which provide information on engineers' coding activity; (3) task management systems such as Jira, which indicate engineers' assignment and completion of tasks; (4) scheduling tools such as Google Calendar, which provide information on meeting activity and time allocation; and (5) generative AI usage, as JF's integration with the APIs for GitHub Copilot and Cursor allows us to observe firm- and individual-level adoption of the two tools.

The scope of our data confers several advantages. First, we observe the full production pipeline — from intermediate measures of coding productivity to final task completion and employment. This allows us to measure both short-run coding activity — where prior studies document sizable productivity effects — and downstream outcomes, where findings are more mixed. Second, we directly observe AI tool usage, which avoids concerns over measurement error or misattribution that arise when inferring AI usage.[4] Finally, our dataset spans a wide range of engineering activity, providing us a more representative aggregate measure of AI's effects and allowing us to examine heterogeneity across tasks, workers, and firms.

We estimate the effects of AI adoption using a staggered difference-in-differences design that exploits variation in the timing of firms' purchases of Copilot and Cursor business licenses. That is, we compare outcomes before and after adoption to those at firms which have not yet adopted Copilot or Cursor. We show that firms differ in their AI adoption dates, generating variation that we can utilize. Additionally, individual-level usage rises sharply at adoption, consistent with the view that firm-level adoption meaningfully shifts workers' usage of AI tools.

Our identification strategy relies on the assumption that, absent treatment, early-adopting firms and late- or non-adopting firms would have followed parallel trajectories in outcomes. This assumption could be violated if the timing of adoption is systematically related to firms' underlying growth dynamics — for instance, if faster-growing firms are more likely to adopt AI tools earlier, or if firms adopt in anticipation of upcoming coding-heavy projects. In practice, however, firms report that the timing of enterprise AI adoption is shaped by a range of logistical and organizational constraints that are plausibly orthogonal to productivity trends. These include data privacy and security reviews, budget cycles, IT setup, piloting processes, and internal training

[3]Code generation is a domain in which large, high-quality corpora of training data are publicly available and clear benchmarks with verifiable outputs have been developed. As a result, recent AI models are able to successfully complete 70-80% of tasks on standard coding benchmarks (OpenAI, 2025; Anthropic, 2025; Pichai et al., 2025).

[4]Direct observation of AI usage is rare. Most papers impute usage, utilize self-reported adoption dates from surveys, or rely on tool release dates rather than tool take-up dates for identification. Ling et al. (2025) find some evidence that students under-report AI usage because of social desirability bias. We avoid any concerns about misreporting by directly observing usage of AI tools.

and change-management processes, all of which can introduce idiosyncratic delays in rollout timing even among otherwise similar firms (Rapoport et al., 2025; IBM, 2024). To assess the plausibility of this identification assumption, we test for differential pre-trends in outcomes and find no evidence that early- and late-adopting firms were on diverging trajectories prior to adoption.

This paper summarizes our preliminary findings on the effects of AI adoption. We begin by studying the effects on worker productivity. Using two industry-standard measures of coding output, we estimate an 8.5% increase in coding activity per worker. We then examine speed and find that AI reduces time to task completion by 8.7%. Finally, we detect no significant changes in measures of code quality. Overall, the productivity effects are noticeable but not transformative: our 95% confidence intervals allow us to rule out increases in coding activity larger than 19.6%. Our estimates are consistent with, but on the lower end of, existing estimates of the productivity gains from generative AI for software engineers.

We next ask whether these productivity gains translate into increased output. Several previous studies define software engineers' output using measures of *intermediate coding activity* such as GitHub commits or pull requests (see, for instance, Peng et al., 2023; Cui et al., 2024; Hoffmann et al., 2024; Yeverechyahu et al., 2024; Song et al., 2024; Sarkar, 2025). In addition to observing these measures of intermediate coding activity, we also observe software engineers' final *output* of work: completion of tasks assigned to them. Holding fixed software engineers' time on coding tasks, an increase in productivity translates one-for-one to an increase in task completion. However, because engineers can change their allocation between coding and non-coding tasks — and because firms can change their total employment of software engineers — an increase in productivity may not result in any increase in output. We find limited effects on task output: on our main specification, the effect on output is insignificant, and we can rule out an effect larger than 11% at the 95% level.

Although task counts remain constant, the composition of tasks could still shift. To evaluate this possibility, we use machine learning methods to classify tasks based on their text descriptions. We label tasks along two dimensions: (1) task length — the time required for an experienced engineer to complete the task, without the assistance of AI tools; and (2) whether the task is coding or non-coding, where "coding" refers to a task requiring any writing or modification of code.

First, we find no evidence that task complexity changes following AI adoption. We then examine whether workers reallocate effort between coding and non-coding tasks. One common hypothesis is that productivity improvements in coding free up time for workers to shift into complementary, non-coding activities such as communication or planning. If the two types of tasks are not substitutable, however, task composition should remain unchanged. Consistent with the latter view, we observe no changes in the number of coding or non-coding tasks completed.

Finally, we study the effects of AI adoption on employment.[5] We begin by assessing the effect of AI on overall employment, as measured by the number of workers indicating employment at a firm on LinkedIn. Then, we estimate the effect of employment on software engineers overall, as well as separately for junior versus senior software engineers, measured by the number of workers active in each firm in the Jellyfish data. On all four outcomes, we estimate reasonably precise zero effects.

Taken together, these preliminary results indicate that while AI tools improve individual-level productivity, we are yet to observe transformational impacts on any firm outcomes. In ongoing work, we are analyzing the barriers to this transformational change.

[5]Many companies have stated that AI tools are already affecting demand for software engineers, with some firms announcing layoffs due to AI (*e.g.*, Kessler, 2023; Roose, 2025), and others announcing increases in hiring (*e.g.*, Stanley, 2025).

One natural possibility is that firms' ability to increase output in response to generative AI depends on the nature of their products and services. In many firms, software engineering functions as an internal input supporting non-software activities. For these firms, the demand for software as an intermediate input is relatively fixed — for example, the retail firm that maintains a website and inventory system may respond to a productivity shock by reducing engineering employment rather than expanding task output. By contrast, firms whose core products are software-based have more expandable output capacity. For these firms, productivity gains can translate into greater task output and, in some cases, higher employment. Our estimates are consistent with this distinction, with software-product firms exhibiting significant increases in task output and non-significant increases in employment, and software-input firms exhibiting no change in either.

We emphasize four limitations when interpreting our results. First, we show that the *short-run* effects of generative AI coding tools are limited. It is plausible that the long-run effects are larger, either because firms will take time to adjust in response to the arrival of generative AI tools, or because generative AI tools will improve over time.[6] Second, our estimates are limited to the context of AI *coding* tools and firms that employ software engineers, though one might anticipate *ex ante* larger effects on these workers. Third, we estimate partial equilibrium effects of adoption of generative AI tools; we do not attempt to assess the general equilibrium effects of these tools, which may include broader labor market effects. Finally, we emphasize that our analysis is preliminary and ongoing.

Our project contributes to a large literature on the effects of technological change on labor market outcomes. This work documents substantial impacts of new technologies on employment, wage dispersion, and task composition (Harrigan et al., 2016; Hoffman et al., 2018; Acemoglu & Restrepo, 2018a, 2019, 2022a,b; Autor et al., 2022; Autor, 2022). Some papers specifically examine the effects of technology adoption on workplace organization, documenting effects on worker and manager autonomy (Bloom et al., 2014) as well as specific skill demands and production processes (Bresnahan et al., 2002).

A newer and rapidly growing literature examines the effects of AI technologies. AI differs from previous technologies in its ability to deduce tacit relationships which are difficult to explicitly encode, and thus its ability to perform non-routine cognitive work. Research has shown rapid growth in AI investments and increased labor demand for AI skills (at the expense of non-AI skills) (Acemoglu et al., 2022; Babina et al., 2024). Our paper is most closely related to a subset of research examining the effects of AI on worker productivity and firm-level outcomes across a wide range of settings, including writing (Noy & Zhang, 2023), customer support (Brynjolfsson, Li, & Raymond, 2025), and more (Chen & Chan, 2024; Choi & Schwarcz, 2023; Dell'Acqua et al., 2023, 2025; Kim et al., 2024; Roldan-Mones, 2024; Otis et al., 2024; Dillon et al., 2025). Most recently, a number of papers measured the employment effects of AI technologies, finding mixed results thus far (Brynjolfsson, Chandar, & Chen, 2025; de Souza, 2025; Humlum & Vestergaard, 2025; Lichtinger & Hosseini Maasoum, 2025), and to embed estimates into macroeconomic frameworks (Acemoglu, 2024).

Within this literature, several papers study AI coding tools. Studies generally find that these tools increase coding output and reduce task completion times, though estimates vary widely in magnitude (Peng et al., 2023; Cui et al., 2024; Song et al., 2024), and some even report negative effects (Becker et al., 2025). A subset of studies have examined the impacts on work patterns, showing shifts in developers' workflows towards project management rather than code writing in GitHub data (Hoffmann et al., 2024), increased contributions to open source projects (Yeverechyahu et al., 2024), and reallocation of effort away from direct code writing and toward

[6]Indeed, an existing literature shows substantial improvement in generative AI tools for coding even over their short lifespan (Achiam et al., 2023; Jimenez et al., 2023; Yang et al., 2024). Brynjolfsson & Milgrom (2012) argue that the effects of general purpose technologies are often small on impact because of the time required to accumulate complementary capital and labor.

prompting and supervising AI agents (Sarkar, 2025). Notably, however, this body has focused on changes in individual workers' behavior, rather than on firm-level organizational responses to AI adoption.

Our paper builds on this literature in several ways. First, the breadth of our data allows us to estimate productivity effects that are more representative of real-world usage, rather than effects among selected users or experimental tasks. Second, our firm-level identification strategy, combined with comprehensive data coverage, allows us to move beyond individual-level productivity effects and behavioral responses. In particular, we go beyond intermediate measures of coding activity to examine downstream outcomes that are contingent on firm-level organizational responses, including final output, task allocation, and employment. By doing so, we shed light on how productivity gains from AI coding tools propagate within firms and reshape the organization of work, complementing existing evidence that focuses primarily on individual-level effects.

The rest of the paper proceeds as follows. In Section 2, we describe our data and institutional details about AI, including details about the data classification and AI take-up. In Section 3, we describe the effects of AI on productivity. In Section 4, we describe the effects of AI on task output and composition. In Section 5, we describe the effects of AI on employment. In Section 6, we describe heterogeneity in effects by worker and firm characteristics. Section 7 concludes.

2 Data and institutional context

This section summarizes the data used in our analysis, and provides background on technology firms and generative AI coding tools.

2.1 Data sources

Jellyfish data. Our primary source of data is Jellyfish ("JF"), a software engineering management platform used by firms to understand the activity of their engineering teams. Firms use JF to monitor teams' progress, and to evaluate the performance of both teams and individual employees. JF integrates data on engineers' activity from several sources: we make use of five sources throughout our analysis.

Analysis sample. We analyze a subsample of JF's data, restricted to customers who consent to their data being used for research. We construct an analysis sample of engineers who are active at a client firm of JF between January 2022 and June 2025. There are 114,634 workers from 518 firms (Appendix Table 1).

Code version control data. Software engineering teams employ "version control" systems to record, organize, and coordinate changes to a shared codebase. Industry-standard version control systems share a common structure, visualized in Figure 1A.[7] A "repository" stores the complete history of edits to code; individual engineers work on a specific "branch" of the repository. Engineers write code on their local machines. As they update their code, they "commit" changes to their branch. When work is ready to be incorporated into the collective codebase, the engineer creates a "pull request", which generates a log of differences between the engineer's "branch" and the current state of the repository. This pull request is then reviewed; depending on the firm's version control system, an engineer may be able to review their own pull request, or a manager or other co-worker may need to do so. If the pull request is accepted, their code is then incorporated into the shared repository ("merged"); otherwise, the code is not incorporated into the shared repository.

Version control systems are ubiquitous in software engineering teams: for instance, the world's largest

[7]Some of this terminology depends on the precise version control system used. We adopt the terminology of GitHub, the world's largest platform for version control.

version control platform, GitHub, is used by around 90% of Fortune 100 firms (GitHub, 2025b). JF integrates data from several version control platforms, including GitHub, GitLab, and Bitbucket.

Our analysis sample includes 112,629,616 commits, for an average of 52.99 per worker-month. We do not observe the content of each commit, but we do observe the metadata associated with each commit, including the time of the commit, the identity of the engineer making the commit, and the text associated with the commit. We also observe 20,861,946 pull requests, for an average of 11.38 per worker-month. 85% of pull requests are merged into a shared repository. Each observation of a pull request includes information on the names of files modified, as well as lines of code added versus deleted by the person creating the pull request. It also includes information on the number of comments by the reviewer of the pull request.

We use commits and pull requests as related but distinct measures of coding activity. Pushing a commit corresponds to making an update to code; creating a pull request represents the completion of a discrete unit of work. In practice, the correlation between these two units of work is around 0.72 at the worker-month level, indicating that workers tend to make more pull requests in time periods in which they are pushing more commits, but the frequency of the two updates is very different: in our analysis sample, on average, workers make five commits for every pull request.

We use the other content of pull requests — number of lines of code deleted, percent of pull requests merged, and number of comments by the reviewer — as indicators of code quality.

Task management data. Engineering teams also employ "task management" systems to track completion of broader work items. Industry-standard task management systems also share a common structure, visualized in Figure 1B.[8] Engineers or managers create "tasks", which are then assigned to specific engineers. Each task can include a title, text description, and links to overarching projects and related tasks. Engineers then write code for the task; this code is tracked by the version control systems discussed above. After updating the code, the engineer marks the task as complete, in which case the task is "resolved".

We view completion of a Jira task as marking the end of a fixed unit of work. This interpretation is consistent with prior academic literature in computer science (*e.g.*, Ortu et al., 2015; Lenarduzzi et al., 2019; Lüders et al., 2022), and the industry-standard use of Jira; indeed, Jira's product description states that tasks "typically represent individual work items such as big features, user requirements, and software bugs" (Jira, 2025).

Task management systems are also industry-standard: Atlassian, Jira's owner, claims that four-fifths of Fortune 500 firms use Jira (Farquhar, 2019). JF integrates data from several task management systems, including Jira and Microsoft Azure DevOps. Our analysis sample includes 13,132,828 tasks, for an average of 5.75 per worker-month.

Figure 1C presents an example of a task in Jira. The task includes a title, description, and links to the worker assigning the task, and the worker to whom the task has been assigned. The text description of each task is typically rich: in our analysis sample, the median length of an issue description is 580 characters.

We also measure the time taken to resolve each task, which we define as the difference between the date at which the task is begun and the date at which the task is marked as completed. This difference is known by software engineers as "cycle time". Teams of engineers organized using the "agile" project methodology usually aim to have a cycle time of under one week. Appendix Figure 1 shows the distribution of cycle times for tasks within our sample. Around two-thirds of tasks are resolved within a week. We construct a binary

[8]As before, for simplicity, we adopt the language of the most common task management system globally, Jira. We use the word "task" instead of "issue".

indicator of a task being resolved quickly, equal to 1 if a task is resolved within one week, and zero otherwise.

Google Calendar data. For a subset of 211 firms, JF receives data from engineers' Google Calendar accounts. Google Calendar is a scheduling tool: users create calendar events and share them with one another. Our analysis sample covers 58,267,744 events. Each calendar event includes information on the individuals who are invited, individuals who accept vs decline, text description, and meeting length.

HR information. Most firms also input HR information into JF platform, including information on the job titles of workers.

Coverage across datasets. Appendix Table 1 summarizes the coverage of the JF data. Essentially every firm in our analysis sample uses both Jira and GitHub.Collectively, the activities tracked by JF — issue assignment and completion, code changes, and calendar events — allow for a wide-ranging view on work activity. To illustrate this point, Figure 1D shows JF's records of the work activity of a single worker on a single day in January 2024. The worker makes several commits to their firm's codebase, and some pull requests, as measured by their version control activity. At the end of this work activity, the worker marks the task on which they have worked as complete. The detailed data made available through JF allow us to construct work schedules of this form for the full analysis sample.

LinkedIn. We obtain data on workers' and firms' characteristics by merging the JF data to data from LinkedIn collected by Revelio Labs, a data provider which scrapes worker profiles and updates the database weekly. We first hand-match the analysis sample of JF firms with the list of firms whose data are collected from LinkedIn; we are able to match essentially every firm in the sample. Revelio provides information on firms' industries and LinkedIn descriptions. Next, we merge worker names from JF data to profile names in the LinkedIn data, restricting attention to profiles that have, at some point, worked at a JF firm. Revelio also provides information on workers' employment and education histories.

We complement this LinkedIn measure of employment with a measure based on the number of active engineers within the JF platform.[9] The advantage of this measure is that it may respond more quickly to changes in employment, to the extent that workers are slow to update their LinkedIn profiles. The disadvantage is that some employees at the firm may not be tracked on JF: for instance, non-engineers are by construction excluded by JF. For this reason, we make use of both employment measures and compare results.

2.2 AI coding tools

Background to AI coding tools. Finally, we describe AI coding tools. GitHub Copilot is a generative AI coding tool developed by GitHub and OpenAI. Users can choose to run it with a number of recent AI models from OpenAI, Anthropic, and Google GitHub (2025a). As of June 2025, it by default uses the GPT-4o model, which was trained on a wide range of public GitHub repositories, and provides coverage of over 30 programming languages (GitHub, 2025c). Cursor is a similar tool developed by Anysphere, which was first released in March 2023.

Engineers use AI in several ways. First, AI can be used as a code completion tool: engineers receive in-line "autocomplete" suggestions while writing code. Second, AI can be used to create initial drafts of code. Third, engineers can use AI to facilitate reviewing others' code. Fourth, engineers can use AI's "chat" feature to iterate on code; they can ask questions and engage in conversation about code implementation. Finally, engineers can use AI to debug code: AI provides suggested fixes in response to error messages. In each application, engineers

[9]We define a worker's employment duration at a firm as the time between their first recorded activity in JF and their last recorded activity.

can accept, revise, or ignore AI's suggestions. Figure 2 provides examples of the visual interface observed by users when interacting with AI.

Measurement. We measure firm- and individual-level adoption of AI using the AI coding tools' APIs, which are integrated with JF's platform. The API collects information on usage by all firms with a business license. At the firm-level, the API collects information on the first date of business license usage as well as daily information on number of active users, number of suggestions, lines of code suggested, lines of code accepted, number of chats, and chat suggestions accepted. At the individual-level, the API collects information on the first date of individual usage. JF built its integration with the API in May 2024 — since then, it has also collected information on whether or not an individual uses AI each day (GitHub, 2024).[10] We define a firm's AI adoption date as the date at which the firm activates its business license.

Take-up rate. Figure 3 shows the rate of take-up of AI within the analysis sample. GitHub Copilot for Business was released in February 2023, and Cursor was initially released in March 2023. At the firm level, adoption gradually increased over the following 18 months; by June 2025, 64% of firms have taken up the AI tools (Figure 3A).

Figure 3B shows the number of users who have ever used AI, as a share of the total number of engineers employed at the firm. After a firm first purchases a business license, workers begin adopting fairly quickly: after 3 months, 28% of workers have begun using AI. After 1.5 years, half of workers have begun using AI. Figure 3C shows the distribution of worker-level adoption rates across firms using AI — while roughly 10% of firms have low individual-level take-up rates of 20% or fewer workers; the overwhelming majority have rather significant take-up rates.

Finally, Figure 4 shows the *extent* of individual-level usage of AI, as measured by the number of days in which individuals submit at least one AI query. Panel 4A displays the percent of workers who continue using AI over time — 6 months after adoption, 62% of workers are still using the tool. Panel 4B displays the distribution of share of working days that workers use AI — on average, workers use the tool on 48% of their working days.

2.3 Jira task classification

Classification method. Much of our analysis relies on understanding the content of Jira work. We use machine learning (ML) tools to classify Jira tasks according to their text descriptions.

We classify tasks along two dimensions. First, we label tasks as "coding" vs "non-coding." We define a task as "coding" if it requires any writing, editing, or deleting lines of code; non-coding if not. For example, bug fix, user interface, refactoring, architecture, or code review tasks would be considered "coding"; planning, documentation, and communication tasks would be considered "non-coding." Second, we label tasks according to their length. We group tasks into four categories: short tasks (< 4 hours), medium tasks (4 to 8 hours), long tasks (8 to 12 hours), and extra long tasks (≥ 12 hours). We provide several illustrative examples of Jira tasks and their labels in Table 2. Hereafter, we refer to these labels as the "ML-assigned labels."

We perform the classification in two steps. First, we use a large language model (GPT-4o) to label a 0.5% sample of tasks (approx. 100,000 tasks), stratified by company. We take these labels as our "ground truth." Second, because the large language model is costly, we use a supervised learning approach to extrapolate the labels to the full set of task descriptions. Specifically, this step involves converting the text descriptions into a 384-dimension embeddings vector using Sentence Transformers (SBERT), then training a neural network

[10]For firms adopting prior to May 2024, we observe the date of adoption so long as the firm turns on the API.

to predict the GPT-assigned labels from the embeddings. The full description of the method is provided in Appendix Section C.1.

Validation of supervised learning step. We begin by validating the supervised learning step in Appendix Table 3. We take the GPT-assigned labels as the true labels and the outputs of the supervised learning step as the predicted labels. The table displays a comparison of these labels for a holdout sample of tasks. Panel A displays the confusion matrix and validation metrics for the coding label. The F1 score is a common metric for assessing the performance of a model with a binary output. It ranges from 0 (worst) to 1 (best); an F1 score above 0.8 is generally considered good. We obtain an F1 score of 0.876, which indicates that our model performs very well.

Panel B displays the validation metrics for the length model. Because the length metric is continuous, we instead use the mean squared error (MSE) as our main validation metric. We obtain an MSE of 0.410 (on a 0-3 scale). We interpret this to indicate that our model performs reasonably well.

Comparison to Jira metadata. We also validate the ML-assigned labels by comparing them to information contained directly in the Jira data, displayed in Appendix Figure 3.

Panel A displays the validation of the "coding" label. We compare the ML-assigned labels to engineer-inputted labels of the task categories. Using a keyword search, we group the engineer-inputted task categories into "coding" and "non-coding," using the same definition provided to the ML algorithm. We find that among tasks categorized as "coding" by the engineer, 80% are labeled as "coding" by the ML algorithm. In contrast, among tasks categorized as "non-coding" by the engineer, 50% are labeled as "coding" by the ML algorithm. The difference is highly significant.

Panel B displays the validation of the "length" label. We compare the ML-assigned labels to the task completion time recorded in Jira. Tasks recorded as being completed within 1 day in Jira are labeled as having an average length of 4.4 hours by our machine learning algorithm. In contrast, tasks recorded as being completed in over 10 days in Jira are labeled as having an average length of 6.0 hours.

3 Effects on productivity

We begin by documenting the effects of AI adoption on productivity. When studying knowledge workers, defining productivity is a central question. Establishing such a definition is far more straightforward in more manual or routinized occupations, such as production or sales, which have more standardized outputs, such as number of items produced or number of customers helped. We measure effects of AI on three definitions of productivity: code activity, task completion speed, and code quality. In each, we use industry-standard outcomes to evaluate engineers' activity. While each outcome individually is imperfect, we view them as useful correlates of productivity, and collectively painting a consistent picture.

3.1 Effects on code activity

First, we document the effects of AI adoption on total coding activity. Our dataset records two main forms of coding activity: (1) GitHub commits, which represent incremental updates to code, and (2) GitHub pull requests, which represent larger updates that must be reviewed. We view each of these as proxies for coding activity. Following Kling et al. (2007), we construct an index to increase precision and avoid concerns around multiple hypothesis testing, by standardizing each of these two measures, then taking the equally-weighted mean of the standardized scores. We report effects on the index, and on the disaggregated measures of coding activity.

We estimate effects on these measures of coding activity using the following difference-in-differences regression for individuals i in firms f and months m:

$$\text{Activity}_{i,f,m} = \gamma_i + \delta_m + \beta_m \text{AI}_{f,m} + \varepsilon_{i,f,m},$$

where $\text{Activity}_{i,f,m}$ is a measure of total coding activity, γ_i is an individual fixed effect, δ_m is a month fixed effect, and $\text{AI}_{f,m}$ is an indicator equal to 1 if the firm has purchased a GitHub Copilot or Cursor business license at month m. That is, access to AI is an absorbing state, in the terminology of the literature on staggered difference-in-difference designs (Callaway et al., 2021; Roth et al., 2023; Dube et al., 2023). Due to the staggered nature of treatment timing and the plausibility of heterogeneous treatment effects, we implement this regression using the imputation estimator of Borusyak et al. (2024). We cluster standard errors at the firm level, since treatment is defined at the firm level.

Our coefficients of interest are the causal effects of AI adoption on total coding activity, β_m. We interpret these as the effects of firm-level access to generative AI tools. We focus on this firm-level effect for three reasons. First, these effects are the economically relevant margin for firms making decisions about whether to adopt generative AI tools. Second, many of our downstream outcomes are defined at the firm level. Finally, worker-level take-up is likely endogenously determined by factors including workers' own ability, the composition of tasks they face, and the encouragement of their managers. By contrast, we argue that variation in timing of firm-level take-up is more likely to be unrelated to trends in productivity.

Figure 5 displays our estimated effects on coding activity. Panel A displays an event study showing the estimated effects on the index of coding activity. The pooled estimate for the index indicates that AI access increases coding activity by 0.063 (s.e. = 0.042) standard deviations. The point estimate corresponds to an effect of an increase in productivity of 8.5%.[11] We are able to rule out the possibility of transformative effects on coding activity: the top of the 95% confidence interval for the pooled estimate is 0.14 standard deviations (or 19.6% of the average worker's productivity).

Panels B-C report event study estimates for each of the components of the index. The point estimates for the effects are generally positive, but noisy. Panel B displays the effect on number of commits. The pooled estimate suggests AI access increased commits by 4.6 commits per worker-month (12.5% of the baseline average number of monthly commits). Panel C displays the effect on number of pull requests. The pooled estimate indicates an increase by 0.44 per worker-month (5.5%), but this estimate is not statistically significant. The smaller magnitude of this pooled estimate is driven by a non-monotonic time trend — though we observe an initial significant increase in pull requests, the effect declines after 10 months.

Our identification strategy relies on the assumption that, absent treatment, early-adopting firms and late- or non-adopting firms would have followed parallel trajectories in outcomes. To support this assumption, we test for differential pre-trends in outcomes. The event study figures in Figure 5 do not show clear patterns in pre-trends. Formally, we test the null hypothesis that the lead coefficients are jointly equal to zero, using a χ^2 test. Using the estimates for commits shown in Panel B, we get a p-value of 0.50. Using the estimates for pull requests shown in Panel C, we get a p-value of 0.61. In both cases, we do not find evidence of differential

[11]To convert from the effect in standard deviations, we take the following steps. For each of the three variables that are components of the index, we multiply the pooled point estimate and the baseline standard deviation of the variable, to express the effect in levels, and then divide by the baseline average of the variable to express the effect in percentage points. We then take the equally-weighted average of the three variables in the index. In practice, the ratio of the mean to the standard deviation is similar for all the variables.

pre-trends.

We examine the sensitivity of our estimates to alternative specifications in Appendix Figure 2. Panel A compares the estimates for our main worker-level and imbalanced specification to specifications with a balanced panel and with outcomes aggregated to the firm-level. Panel B compares our results across different estimators used in the difference-in-differences literature. We do not observe significant differences across any of these specifications. Thus, for the rest of the paper, we continue with the same individual-level specification using the imputation estimator of Borusyak et al. (2024).

3.2 Effects on time to complete work

Next, we are interested in examining how increases in coding output translate to changes in time taken to complete discrete units of work. Thus, we document the effects of AI adoption on time to complete each Jira task. As before, we construct an index of coding speed. We use two measures of coding speed: First, we use the total time to completion (which enters the index negatively). Second, we use a binary indicator for a task being completed within one week. Here, a task that is marked as "incomplete" and discarded would enter as a "0"; however, it would not have a completion time associated with it.

We estimate effects on coding speed using the same difference-in-difference design. As before, we first report results on an index which aggregates our two measures of coding speed, before turning to disaggregated results.

Panel A of Figure 6 shows estimated effects on the index. The estimated effect of AI access is substantial: on the pooled estimate, coding speed increases by 0.027 standard deviations (s.e. = 0.021). This increase in coding speed is reflected in both of our individual measures: average time to complete a task falls by 1.8 days (8.7% improvement, relative to a mean of 20.5 days); the share of tasks resolved within a week increases by 0.2 p.p. (relative to a mean of 45.9%).

3.3 Effects on code quality

It is possible that AI tools help workers to write more code and to do so more quickly, but that this code is of worse quality. Thus, we finally turn to effects on code quality. Since we do not observe the code produced by developers, we are not able to directly measure code quality. Instead, we measure three indirect indicators of code quality, each of which is used as an industry-standard tool to assess developer performance.

First, we measure the total number of comments on each pull request. Comments are typically left to request improvements on code; for this reason, a smaller number of comments indicates an improvement in quality of code. Second, we measure the number of lines of code deleted in the pre-pull commits. We view the replacement of previous blocks of code, as measured by lines deleted, as a negative indicator of quality. Third, we measure the share of GitHub pull requests which are merged. The code from a pull request is manually reviewed by a senior or managerial engineer prior to deployment — thus, we view an increase in the share of pull requests that are merged as evidence of an increase in quality of code. Successfully being merged enters our index positively; comments and lines deleted enter our index negatively.

We estimate effects on coding quality using the same difference-in-difference design. Panel A of Figure 7 shows our estimated effects on our index of code quality. We are able to estimate a reasonably small effect on coding quality: the pooled point estimate is 0.032 standard deviations (s.e. = 0.016). Panels B-D report estimated effects on each variable included in the index. We observe a decrease in the number of comments per pull request by 0.30 (relative to a baseline mean of 3.0), which is significant. However, we do not observe much change in the number of lines of code deleted per month, or the fraction of pull requests that are successfully merged.

In all, we interpret these findings to indicate that there is not much change in code quality. This result comes in contrast to several existing studies that have found a quantity-quality tradeoff in the effects of generative AI tools on open source software projects (see, *e.g.*, He et al., 2025; Fu et al., 2025) or code written in lab settings (see, *e.g.*, Perry et al., 2023). One possible explanation for the discrepancy is that users in our setting are subject to stricter quality control processes than users coding as part of controlled experiments or contributing to open source software projects. As a result, access to generative AI coding tools appears to increase the speed with which engineers write code, without corresponding deteriorations in code quality.

3.4 Discussion

We estimate a moderate productivity effect, with a 8.5% increase in coding activity and 8.7% decrease in time to task completion. How do these estimates compare to prior research on AI coding tools? To date, seven papers have examined productivity effects, using outcomes such as number of commits, number of pull requests, and task completion speed (Peng et al., 2023; Cui et al., 2024; Hoffmann et al., 2024; Song et al., 2024; Becker et al., 2025; Sarkar, 2025). Figure 8 benchmarks our estimates against these studies. Across all outcome measures, our estimates fall near the lower end to midpoint of existing findings.

4 Effects on task output and composition

Our preliminary evidence suggests that AI adoption increases coding activity and task completion speed, without any change in measured code quality. How do these productivity improvements translate to changes in task completion? While the coding measures from the previous section represent intermediate outputs, individual tasks represent discrete units of work, and hence final outputs. We begin by documenting effects on overall task output. Then, we turn to the effects on task composition, measured by the amount of coding versus non-coding work.

4.1 Effects on task output

The effects on total task output may go in two possible directions. In one view, the amount of coding work that a firm needs to complete is fixed, in which case productivity improvements do not actually translate to an increase in the number of tasks completed. In another view, firms can expand the amount of work they complete, in which case productivity improvements translate to an increase in the number of tasks completed.

Figure 9 displays our event study estimates for task output. Panel A displays the results with number of tasks completed per worker-month as the outcome variable. We find no effect. Our pooled estimate is -0.07 (s.e. = 0.25, baseline mean = 5.0). This gives us a confidence interval of to -0.416 to 0.564, ruling out a decrease in task completion larger than 8.3% as well as an increase larger than 11.3%.

A concern might be that the definition of a "task" may change post-AI adoption. For instance, each task may become more complex, so the task count itself may not capture an increase in the amount of work being completed. We test for this possibility using our measure of task length, labeled using the machine learning methods described in Section 2.3. Panel B displays the average length of each task in hours. We again find no effect.

4.2 Effects on task composition

One possibility is that AI adoption has no effect on *total* output, but does affect the *composition* of work. We assess the effects of generative AI tools on the composition of coding *vs.* non-coding work. Figure 10 displays our event study estimates. Panel A displays the effects on the number of coding tasks completed; Panel B displays the effects on the number of non-coding tasks completed. In both, we do not observe an apparent change.

5 Effects on employment

Finally, we turn to assessing the effects of AI adoption on employment. AI's employment effects have been the focus of significant public discourse as well as academic work, with many voicing concerns about job displacement. We adopt the same dynamic difference-in-difference design described in Section 3.

5.1 Effects on firm-level employment

We begin by assessing the effects of AI adoption on a firm's total employment, as measured by the total number of LinkedIn profiles attached to a firm at a given point in time. Figure 11 Panel A displays our estimates. The event study displays no time trend in terms of employment loss or gain. The pooled estimate is 1.60 p.p. with a standard error of 1.75 p.p., which allows us to rule out at the 95% level the possibility of an employment decline greater than 1.83 p.p. or gain greater than 5.03 p.p. The estimates in the later months of the event study are noisier but still zero. For instance, 1 year after adoption, we get a point estimate of 2.95 p.p. (s.e. = 3.36 p.p.).

5.2 Effects on software engineers

Effects on overall software engineers. We next assess the effects of AI adoption on a firm's number of software engineers, as measured by the number of workers who are active in Jellyfish data at a given point in time. We do this for two reasons. First, conceptually, it is possible that the shock of AI adoption is sufficiently localized to coding teams that there is no broader impact on firm-wide employment. Second, from a measurement point of view, people may have concerns about measuring employment using LinkedIn data — for instance, LinkedIn data may be slow to update layoffs.

Figure 11 Panel B displays our event study estimates. Again, the event study displays no time trend in terms of employment loss or gain, and the pooled estimates provide a reasonably precise zero effect. Specifically, we get a pooled estimate of -1.50 p.p. with a standard error of 3.32 p.p, which allows us to rule out at the 95% level the possibility of an employment decline greater than 8.00 p.p. or gain greater than 5.00 p.p.

Effects by worker seniority. Finally, it is possible that the aggregate effect masks heterogeneity in the employment effects by worker seniority. In particular, some papers have found evidence of employment declines among junior workers and gains among senior workers (Brynjolfsson, Chandar, & Chen, 2025; Lichtinger & Hosseini Maasoum, 2025). We test for this possibility using a combination of the Jellyfish and LinkedIn data. We use the Jellyfish data to assess workers' employment dates at the firm, due to the higher frequency nature of this data. Then, we use the LinkedIn data to assess workers' seniority levels, as LinkedIn provides more comprehensive coverage of job titles. Using this method, we do not find employment effects for either junior or senior engineers.

5.3 Discussion

These null effects on employment are consistent with some empirical work documenting small employment effects of generative AI (Humlum & Vestergaard, 2025), though some papers find evidence of much larger effects (Brynjolfsson, Chandar, & Chen, 2025; Lichtinger & Hosseini Maasoum, 2025). We display a comparison of these estimates in Figure 12.

One advantage of our setting is our ability to directly observe AI take-up. This makes us more confident in our capacity to attribute any employment changes to AI adoption, rather than to broader changes in the economy.

However, we also emphasize several caveats when interpreting our results. First, we present a reasonably short-run estimate of the effect of generative AI on employment; it is plausible that employment responds slowly to technological changes. Second, our analysis is restricted the impacts of AI *coding* tools on firms that employ software engineers, though one may anticipate that this is a domain in which the impacts of AI are more likely to materialize quickly. Finally, our event study design identifies the partial equilibrium effect of generative AI adoption. It is plausible that this null partial equilibrium effect masks a negative general equilibrium effect. To see this possibility, note that giving a single firm access to a labor-augmenting technology will have two effects in partial equilibrium: first, there will be a *displacement effect*, since the number of workers per unit of output will fall; and, second, there will be a *scale effect*, as the output of the firm may increase (Acemoglu & Restrepo, 2018a). This scale effect may be primarily business-stealing from other firms. If so, we would expect reductions in those firms' employment.

6 Heterogeneity by worker and firm characteristics

We now turn from these aggregate effects to heterogeneity by worker and firm characteristics.

We examine two dimensions of worker-level heterogeneity — seniority and baseline coding activity. Prior work has suggested that generative AI has larger effects for inexperienced workers in some settings (*e.g.*, Brynjolfsson, Li, & Raymond, 2025) and for more experienced workers in others (*e.g.*, Otis et al., 2024). We classify seniority according to workers' job titles on LinkedIn 2022. We perform the classification using a keyword-based search, where a worker is classified as "senior" if their job title includes any words indicating a senior or managerial position (e.g., "senior", "lead", "director"). Then, we classify a worker as having high baseline coding activity if they have an above-median value on the coding activity index in the year 2022 (prior to any firm adopting Copilot or Cursor), and below otherwise.

Then, we turn to two dimensions of firm-level heterogeneity — firm size and whether the firm creates a software product versus uses software as an input. Though these two dimensions have not been examined thus far in the literature, they may moderate the speed and extent of organizational response to AI adoption. First, anecdotally, smaller firms are often more flexible and capable of responding organizationally. We measure firm size by the total number of workers connected to the firm on LinkedIn during our timeframe sample. We label a firm as "large" if it is above median size, and "small" if below. Second, firms may differ in their ability to scale software output. "Software-as-product" firms create and sell software or digital platforms — such as SaaS providers, developer tool companies, and fintech platforms — and can expand production when engineering productivity rises. By contrast, "software-as-input" firms employ engineering teams primarily to support non-software operations (e.g., real estate, retail, logistics, healthcare) and therefore face relatively fixed internal software demand. To distinguish these groups, we use a keyword-based classification applied to firms' LinkedIn industry tags and text descriptions. Technology-specific keywords such as "software," "artificial intelligence," "cloud," or "fintech" indicate that software is a primary output, while terms for other industries like "real estate," "healthcare," or "retail" signal that software is used mainly as an internal input. The full description of the classification method is provided in Appendix Section C.2.

We estimate these heterogeneous effects by separately estimating our staggered difference-in-difference specification by different categories of firm and worker. We maintain the same identifying variation as in the baseline specification and continue to cluster standard errors at the firm level.

6.1 Heterogeneity in effects on productivity

We begin by assessing heterogeneity in the effects on our three productivity indices — output, speed, and quality. The results are displayed in Figure 13.

In Panel A, we display differences in effects by worker seniority. In each of our three measures of productivity, the point estimates suggest marginally larger effects for *senior* workers, though the estimated differences are insignificant at the 5% level.

In Panel B, we display differences in effects by baseline activity. We generally find larger point estimates for engineers with low baseline coding activity, though the differences again are insignificant at the 5% level. This is consistent with Panel B, since more senior engineers tend to have lower coding activity (perhaps because they complete more non-coding tasks).

In Panel C, we display differences in effects by firm size. We do not find consistent differences across our three indices.

Finally, in Panel D, we display differences by whether firms create software products versus use software as an input in their production process. Accordingly, we find slightly larger point estimates for the firms that create software products than the firms that use software as an input, though the differences still are insignificant.

6.2 Heterogeneity in effects on task output and composition

Next, we turn to heterogeneity in task output and composition. Figure 14 displays the estimates for number of tasks completed overall, count of coding tasks completed, and number of non-coding tasks completed. In Panels A-C, we do not observe pronounced differences across worker seniority, baseline coding activity, or firm size.

However, in Panel D, we observe suggestive evidence of pronounced differences by whether firms create software products versus use software as an input. Firms that use software as an input do not exhibit any change in number of overall, coding, or non-coding tasks completed. In contrast, firms that create software products exhibit significant increases in overall and coding tasks completed, without any change in non-coding tasks completed. The difference in point estimates for coding tasks is marginally insignificant. These results are consistent with the notion that some firms have fixed software needs, whereas others have expandable software needs.

6.3 Heterogeneity in effects on employment

Finally, we test for heterogeneity in employment effects across our two firm characteristics in Figure 15. The figure displays the pooled estimates for employment of total software engineers, junior software engineers, and senior software engineers, measured through the merge of the Jellyfish and LinkedIn data.

Panel A displays the results for firm size, for which there is not much variation in the employment estimates. Panel B provides suggestive evidence for the importance of firm type. For overall, junior, and senior employment, the point estimates are greater for firms that create software products than those that use software as an input. The point estimates are highly noisy; however, they are directionally consistent with the notion that some firms have expandable software needs — in turn, they may be more inclined to expand employment in response to a productivity shock. In contrast, firms that have fixed software needs will cut employment.

7 Conclusion

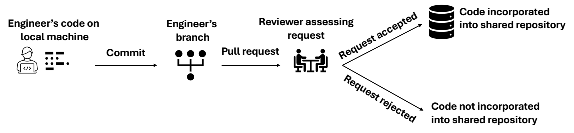
These preliminary results suggest that generative AI tools like Copilot and Cursor lead to moderate, but not transformational, improvements in engineers' productivity. Productivity rises through faster task completion and slightly higher coding activity, without observable deterioration in quality. These effects are economically meaningful yet modest in scale, and they do not translate into changes in organizational structure or employment.

In ongoing work, we are examining why the productivity shock generated by these AI tools does not translate to effects on downstream outcomes. We provide suggestive evidence that these downstream outcomes vary by firm type. Some firms create software products and have the capacity to expand software outputs — for these, we observe significant increases in task output, and slight but non-significant increases in employment. In contrast, other firms only use software as an input and have more fixed software needs — for these, we do not observe any change in task output or employment.

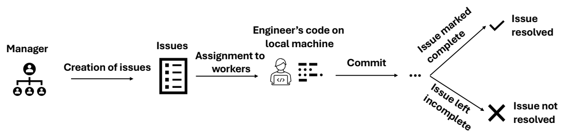
Figures

FIGURE 1. Examples of work activity monitored by JF

A. Example pattern of usage of version control system

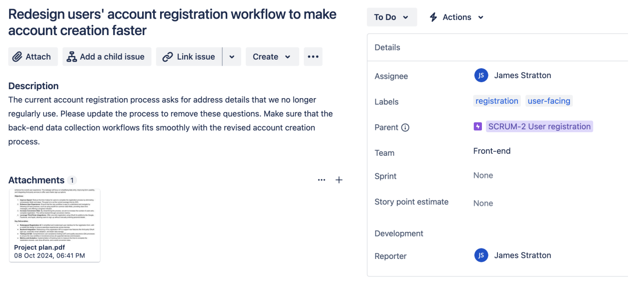

Engineer's code on local machine
Commit
Engineer's branch
Pull request
Reviewer assessing request
Request accepted
Code incorporated into shared repository
Request rejected
Code not incorporated into shared repository

B. Example pattern of usage of task management system



D. Example of different categories of work activity monitored by JF for a single worker on a single day

C. Example of a task issued via a task management system



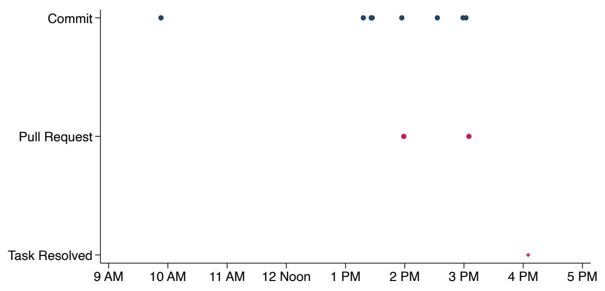
Redesign users' account registration workflow to make account creation faster

Description
The current account registration process asks for address details that we no longer regularly use. Please update the process to remove these questions. Make sure that the back-end data collection workflows fits smoothly with the revised account creation process.

Attachments 1
Project plan.pdf
08 Oct 2024, 06:41 PM

Details

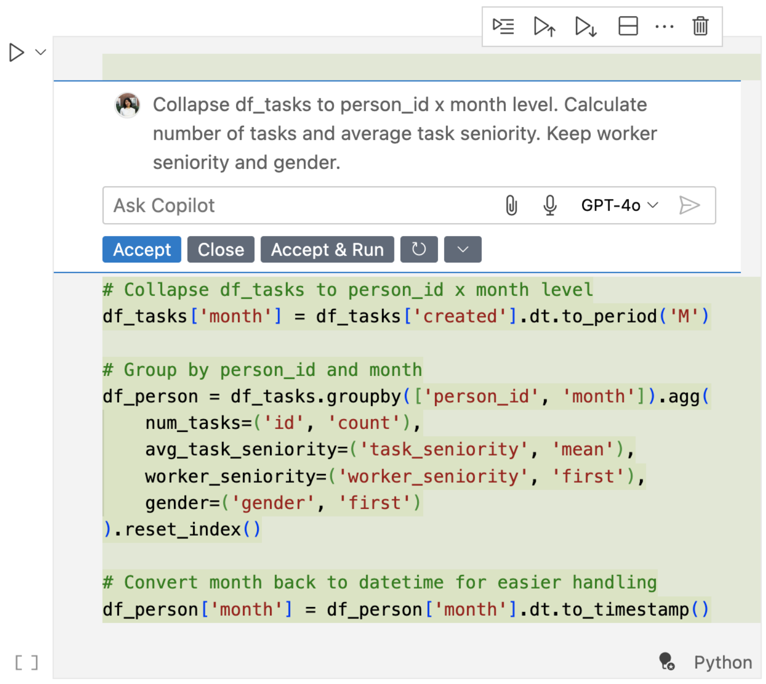
Assignee	JS James Stratton
Labels	registration user-facing
Parent	SCRUM-2 User registration
Team	Front-end
Sprint	None
Story point estimate	None
Development	
Reporter	JS James Stratton



Notes: This figures displays illustrative examples of work activity monitored by JF. Panel A provides an example of the workflow of a team using a version control system. Panel B provides an example of the workflow of a team using a task management system. Panel C provides an example of a task in Jira. (The proprietary language for these tasks is an "issue".) Panel D shows an example of various sources of work activity observed for one worker on one day.

FIGURE 2. Examples of AI coding tool interface

A. AI for drafting code

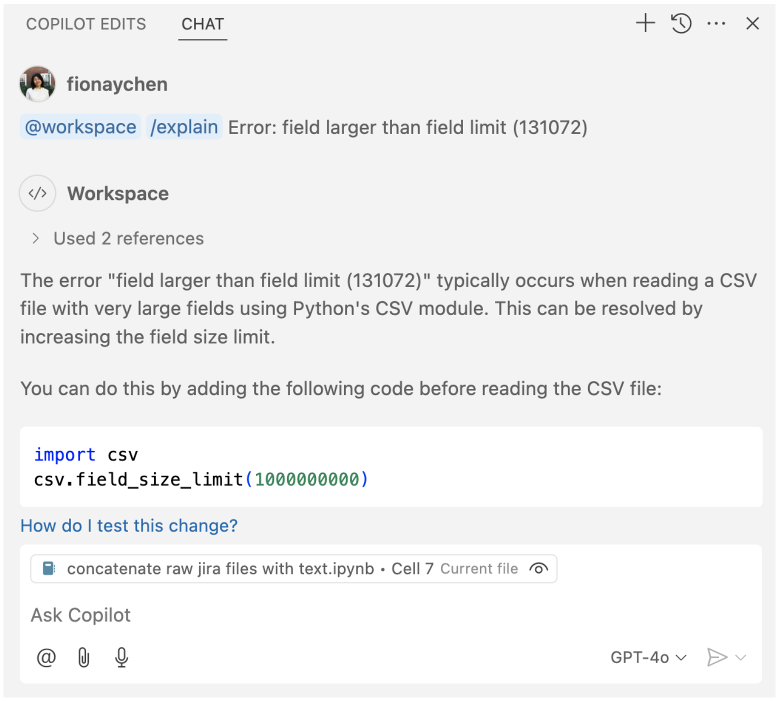

Collapse df_tasks to person_id x month level. Calculate number of tasks and average task seniority. Keep worker seniority and gender.

```
# Collapse df_tasks to person_id x month level
df_tasks['month'] = df_tasks['created'].dt.to_period('M')

# Group by person_id and month
df_person = df_tasks.groupby(['person_id', 'month']).agg(
    num_tasks=('id', 'count'),
    avg_task_seniority=('task_seniority', 'mean'),
    worker_seniority=('worker_seniority', 'first'),
    gender=('gender', 'first')
).reset_index()

# Convert month back to datetime for easier handling
df_person['month'] = df_person['month'].dt.to_timestamp()
```

B. AI for code iteration



fionaychen

@workspace /explain Error: field larger than field limit (131072)

Workspace

Used 2 references

The error "field larger than field limit (131072)" typically occurs when reading a CSV file with very large fields using Python's CSV module. This can be resolved by increasing the field size limit.

You can do this by adding the following code before reading the CSV file:

```
import csv
csv.field_size_limit(1000000000)
```

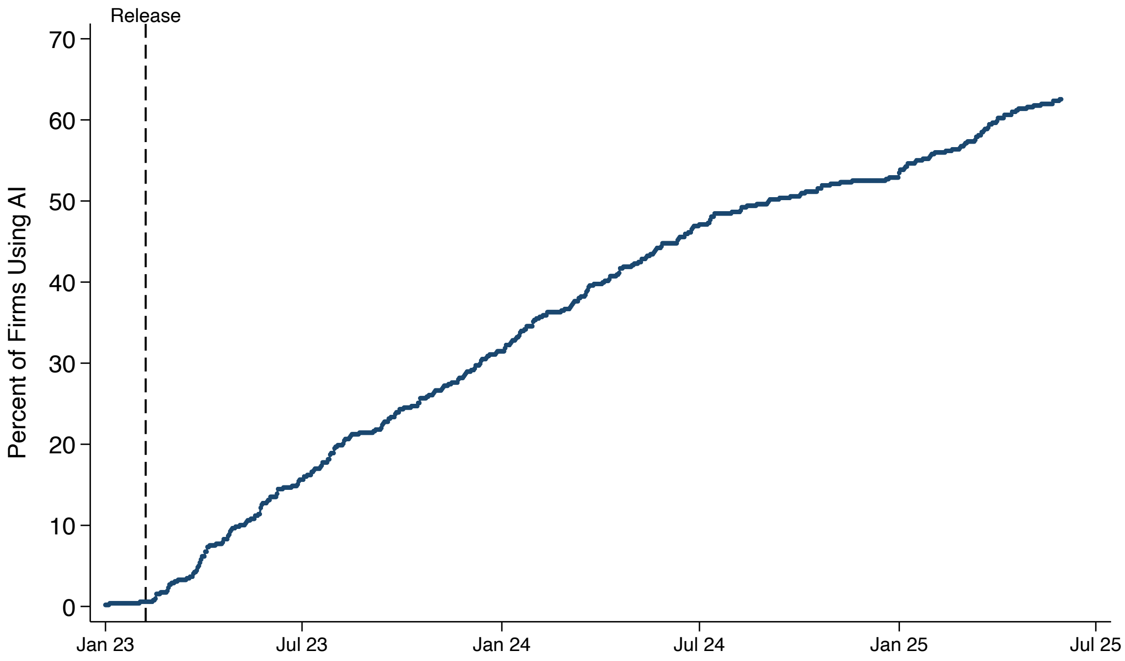
How do I test this change?

concatenate raw jira files with text.ipynb • Cell 7 Current file

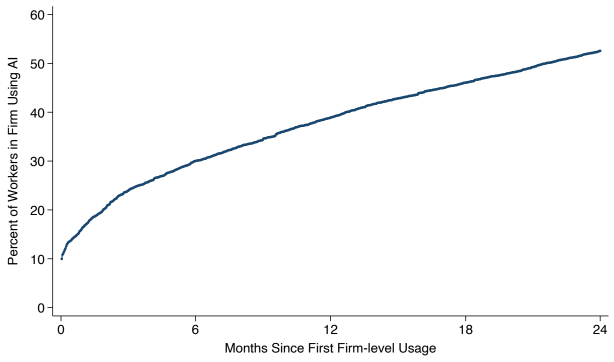
Notes: This figure displays two examples of the user interface for GitHub Copilot. Panel A displays an example of Copilot's code generation feature: engineers can provide Copilot with a prompt, and Copilot will generate code in-line. Panel B displays an example of Copilot's chat feature: engineers can ask Copilot questions, *e.g.*, about bugs or inefficiencies in code.

FIGURE 3. Adoption of generative AI tools

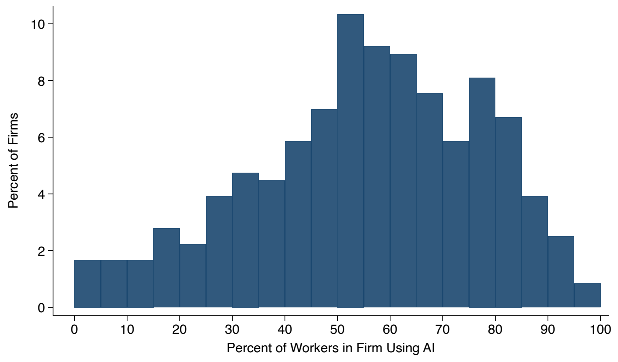
A. Firm adoption of Copilot and Cursor



B. Average worker adoption rates of Copilot/Cursor, within firms using Copilot/Cursor



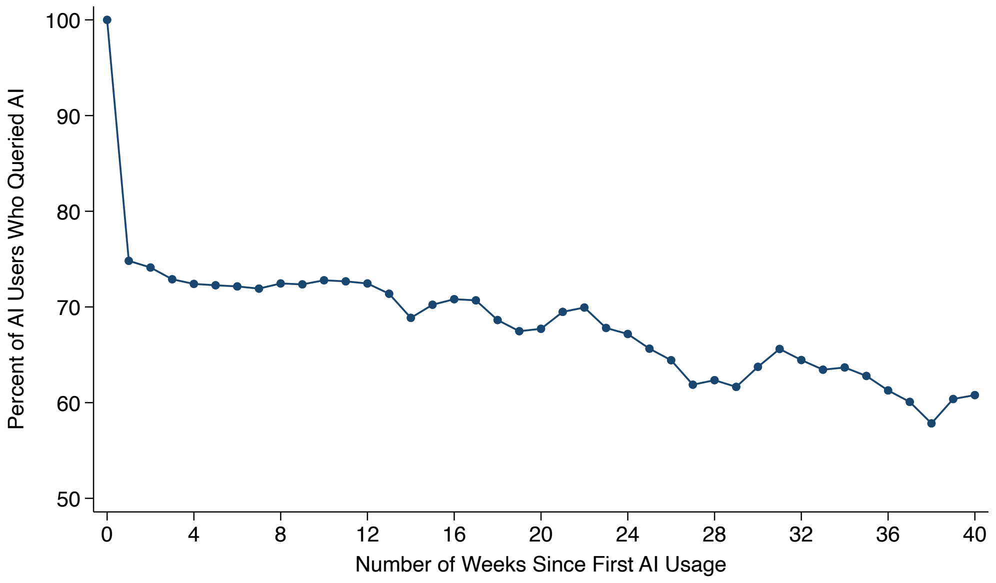
C. Distribution of worker adoption rates, across firms using Copilot/Cursor



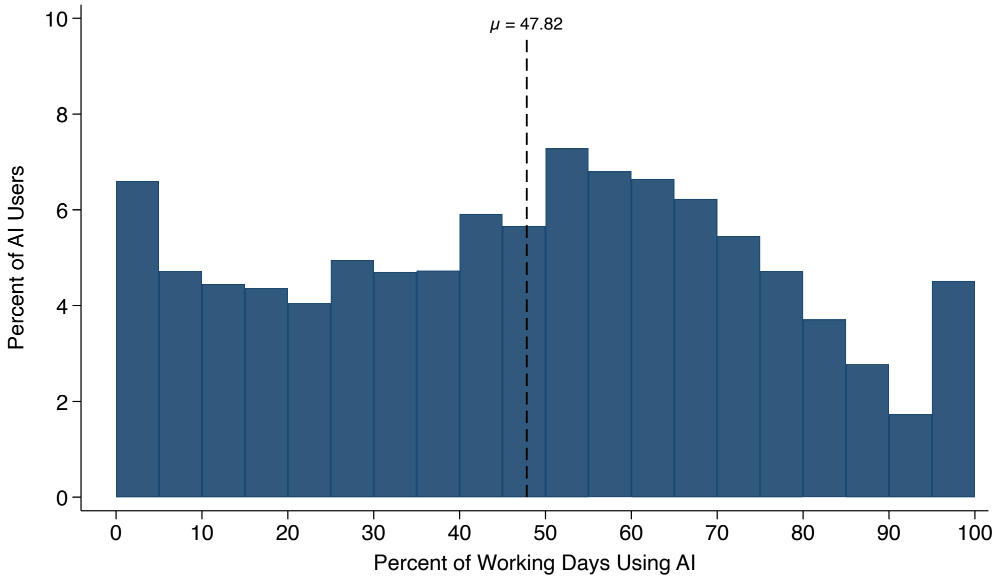
Notes: This figure displays information on level of Copilot and Cursor adoption over time. Panel A displays the share of firms that have adopted Copilot or Cursor in our sample since the release of Copilot's business license in February 2023. Firm-level adoption is defined as business license purchase, and adoption date is measured as the firm's first observation through the API. Panel B displays the average share of workers who have adopted Copilot or Cursor in each firm, relative to the first instance of firm-level adoption. Panel C displays the distribution of worker-level adoption rates across firms. Worker-level adoption is defined as any usage, and adoption date is measured as the worker's first observation through the Copilot or Cursor API.

FIGURE 4. Extent of AI usage among AI adopters

A. Individual-level AI usage over time



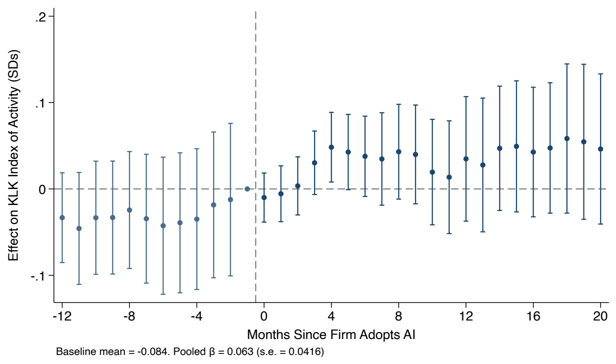
B. Percent of working days using AI since first usage



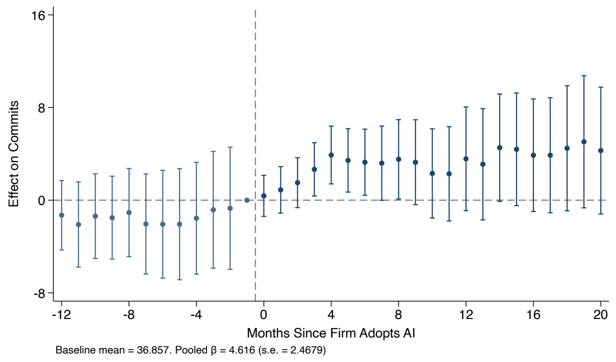
Notes: This figure displays information on the extent of AI usage, among individuals who have adopted AI, using data on AI queries that is available from May 27, 2024 onwards. Panel A shows the continued usage rate in the weeks since first usage — i.e., it calculates the share of individuals who used AI in a given relative week, among individuals who ever used Copilot. Panel B shows the distribution of the share of working days that an individual has used AI, since the first date of usage.

FIGURE 5. Effects of AI on coding activity

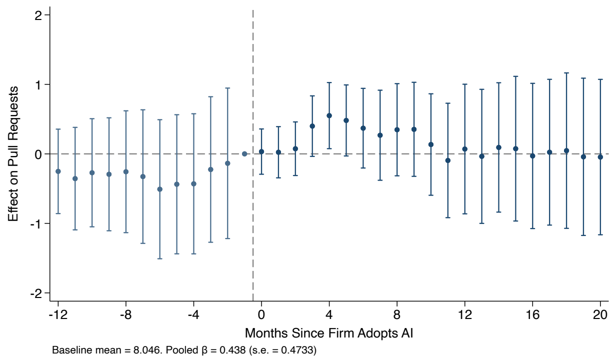
A. Index of coding activity



B. Number of GitHub commits



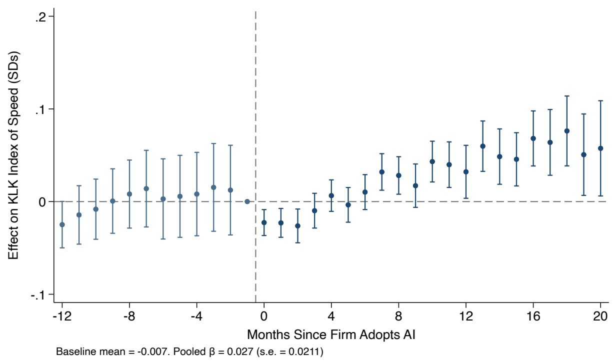
C. Number of GitHub pull requests



Notes: This figure displays event studies of the effects of AI adoption on total coding activity, using the imputation estimator of Borusyak et al. (2024). The pooled coefficient is calculated by taking the average of the post-adoption coefficients and subtracting the average of the pre-adoption coefficients. In Panel A, the outcome variable is a Kling et al. (2007) index of coding activity, consisting of the outcomes in the other three panels. In Panel B, the number of GitHub commits per worker-month is the outcome variable; in Panel C, the number of GitHub pull requests per worker-month.

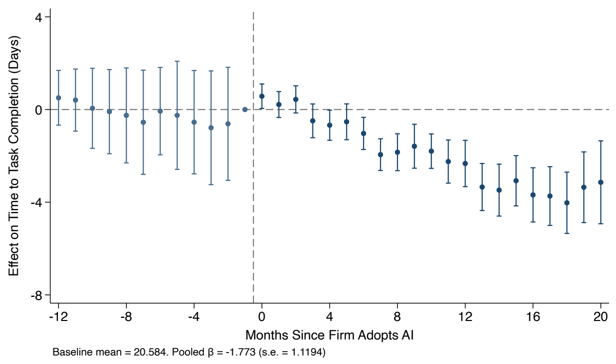
FIGURE 6. Effects of AI on completion speed

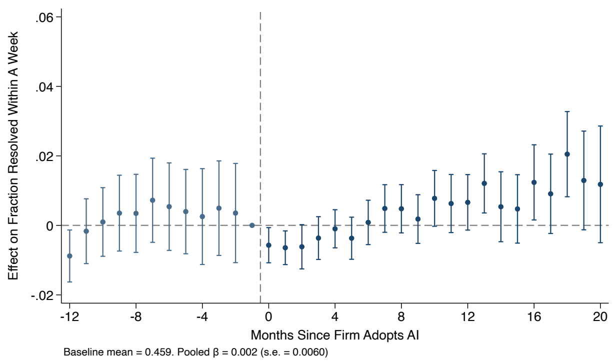
A. Index of completion speed



B. Days to task completion

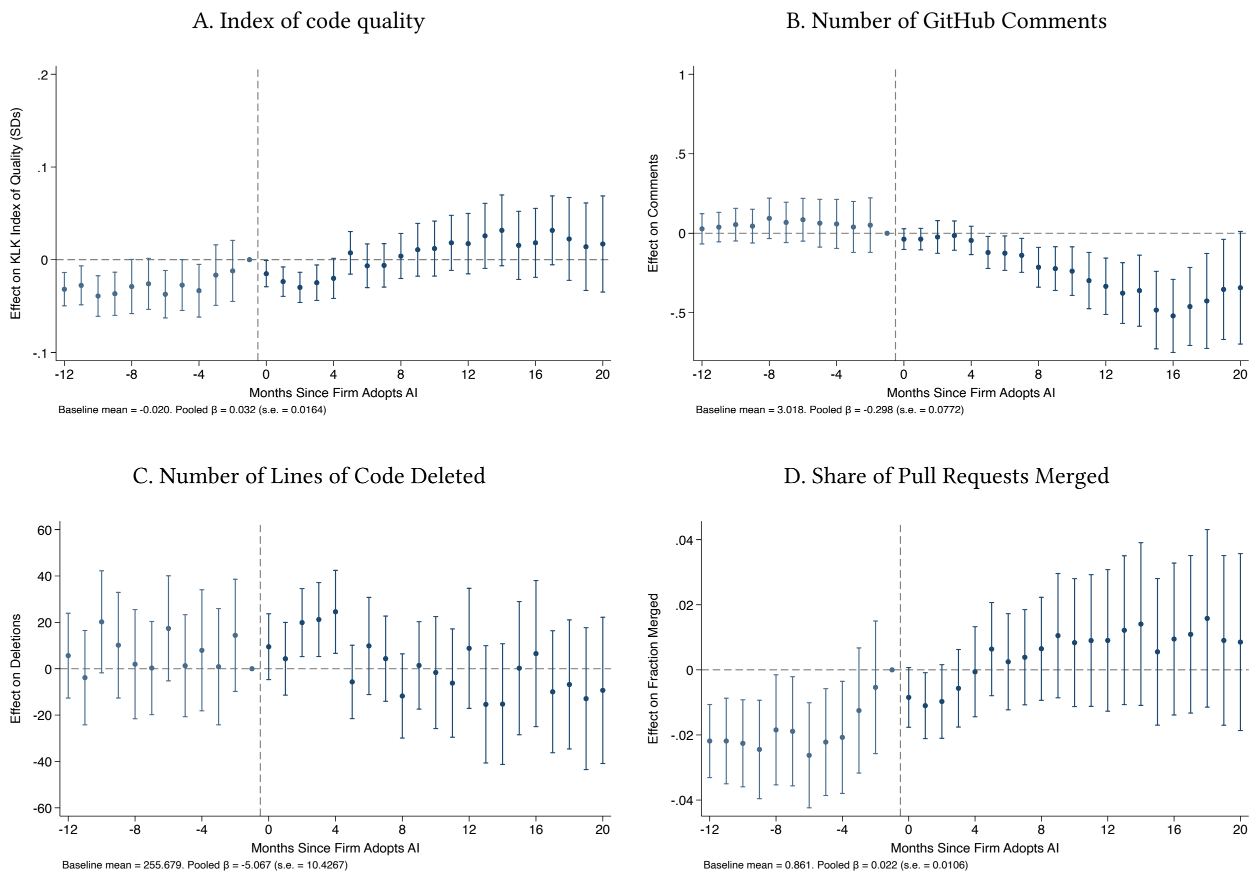
C. Share of tasks completed within 1 week





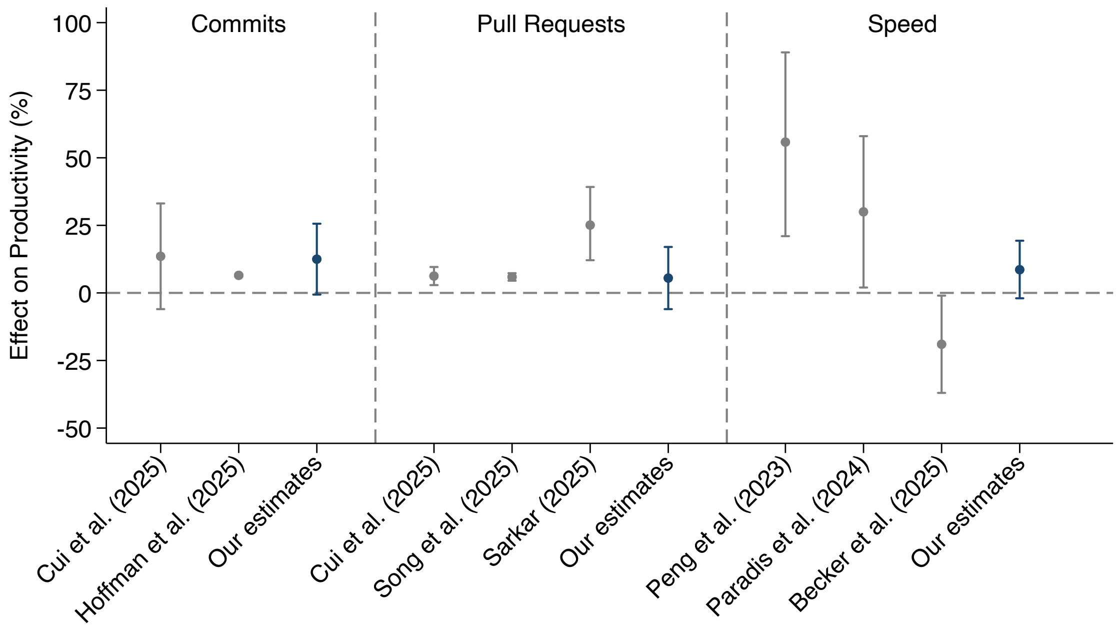
Notes: This figure displays event studies of the effects of AI adoption on code quality, using the imputation estimator of Borusyak et al. (2024). The pooled coefficient is calculated by taking the average of the post-adoption coefficients and subtracting the average of the pre-adoption coefficients. In Panel A, the outcome variable is a Kling et al. (2007) index of coding speed, consisting of the outcomes in the other two panels. In Panel B, the average number of days to task completion is the outcome variable; in Panel C, the share of tasks per worker-month resolved within one week; in Panel D, the share of tasks per worker-month resolved within one week, among coding tasks; in Panel E, the share of tasks per worker-month resolved within one week, among non-coding tasks.

FIGURE 7. Effects of AI on code quality



Notes: This figure displays event studies of the effects of AI adoption on code quality, using the imputation estimator of Borusyak et al. (2024). The pooled coefficient is calculated by taking the average of the post-adoption coefficients and subtracting the average of the pre-adoption coefficients. In Panel A, the outcome variable is a Kling et al. (2007) index of code quality, consisting of the outcomes in the other three panels. In Panel B, the number of GitHub comments per pull request is the outcome variable; in Panel C, the number of lines of code deleted per pull request; in Panel D, the share of pull requests that are successfully merged.

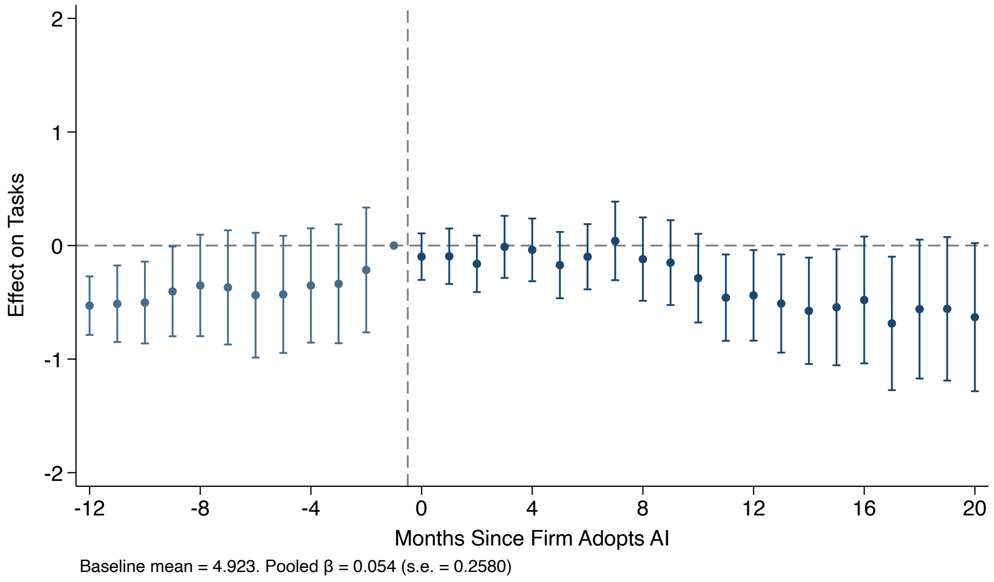
FIGURE 8. Comparison of productivity estimates



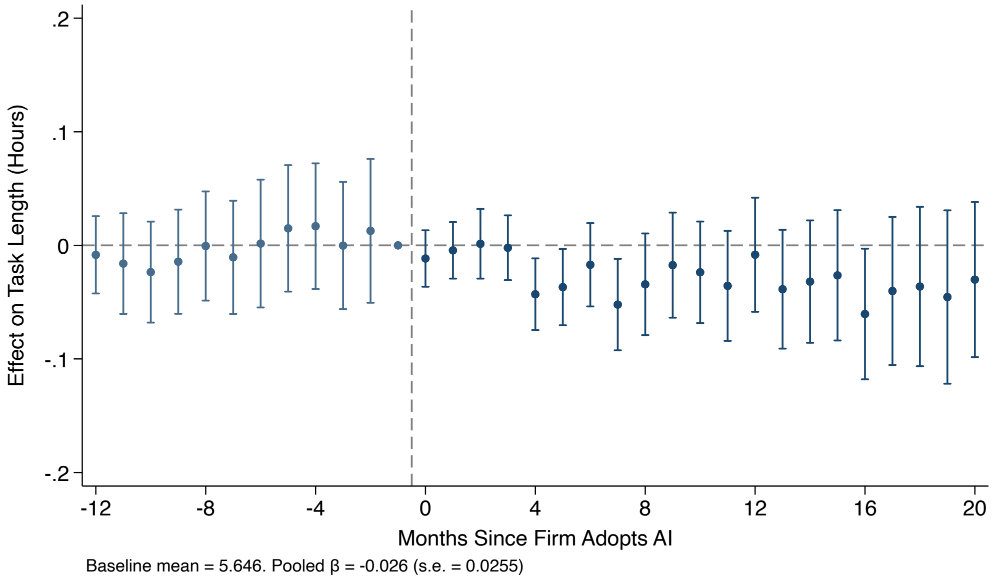
Notes: This figure displays a comparison of estimates of AI coding tools' productivity impacts, from our paper with those of Peng et al. (2023); Cui et al. (2024); Hoffmann et al. (2024); Song et al. (2024); Becker et al. (2025), and Sarkar (2025).

FIGURE 9. Effects of AI on task output

A. Number of tasks



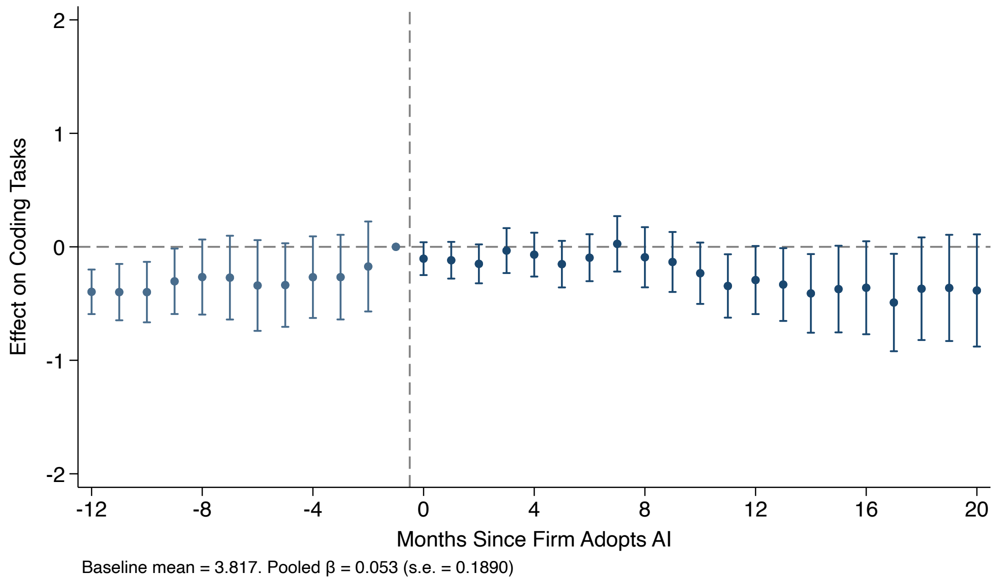
B. Average task size



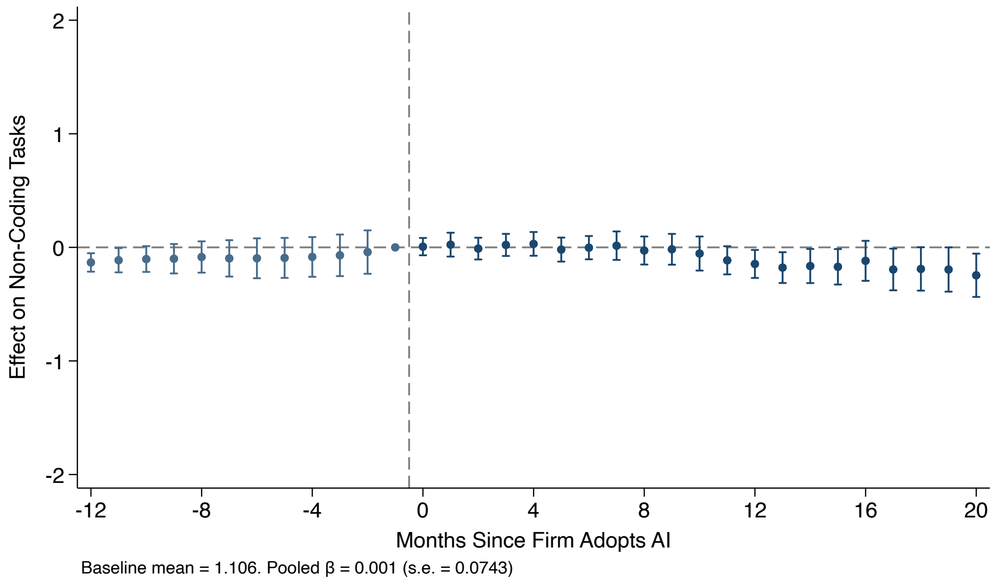
Notes: This figure displays event studies of the effects of AI adoption on task output, using the imputation estimator of Borusyak et al. (2024). The pooled coefficient is calculated by taking the average of the post-adoption coefficients and subtracting the average of the pre-adoption coefficients. In Panel A, the outcome variable is the number of Jira tasks resolved per person per month; in panel B, the outcome variable is the average task length in hours. To calculate task length, we first use the method described in Section 2.3 to label each task with a category. Then, we convert each category to a numerical length by taking the midpoint of the range (i.e., if a task was labeled with the 0-4 hour category, we say it takes 2 hours).

FIGURE 10. Effects of AI on task composition

A. Number of coding tasks



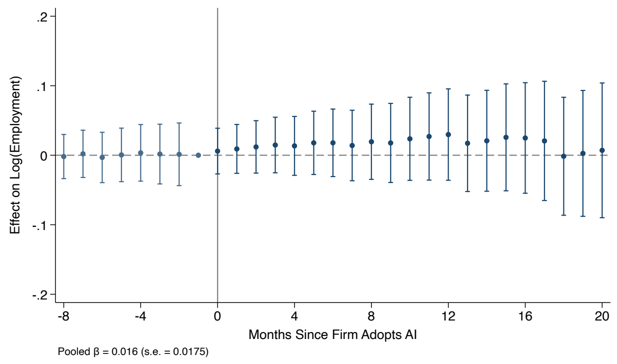
B. Number of non-coding tasks



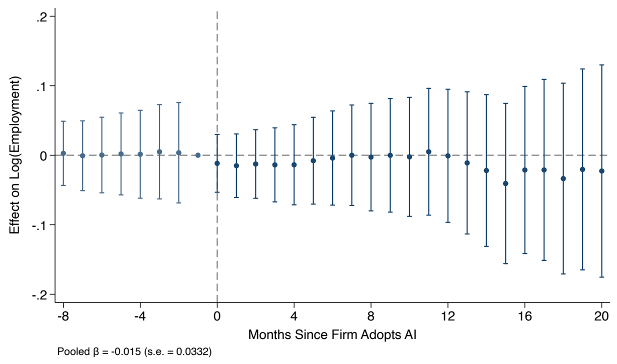
Notes: This figure displays event studies of the effects of AI adoption on task composition, using the imputation estimator of Borusyak et al. (2024). The pooled coefficient is calculated by taking the average of the post-adoption coefficients and subtracting the average of the pre-adoption coefficients. In Panel A, the outcome variable is the number of coding Jira tasks resolved per person per month; in panel B, the outcome variable is the number of non-coding Jira tasks resolved per person per month. To label tasks as coding versus non-coding, we first use the method described in Section 2.3.

FIGURE 11. Effects of AI on employment

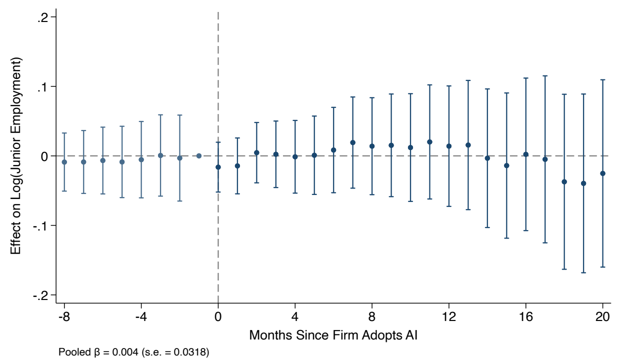
A. Total firm employment



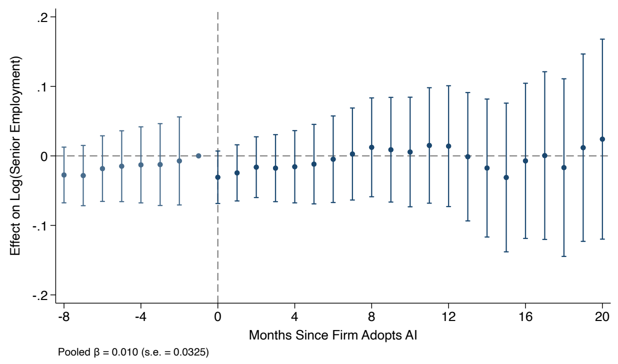
B. Employment of engineers



C. Employment of junior engineers

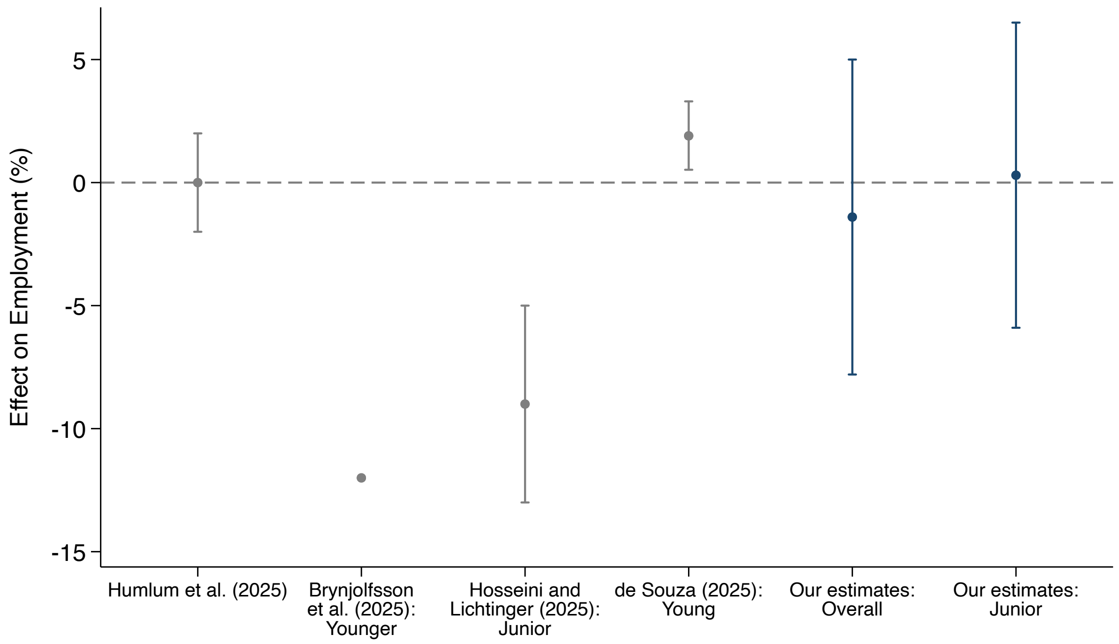


D. Employment of senior engineers



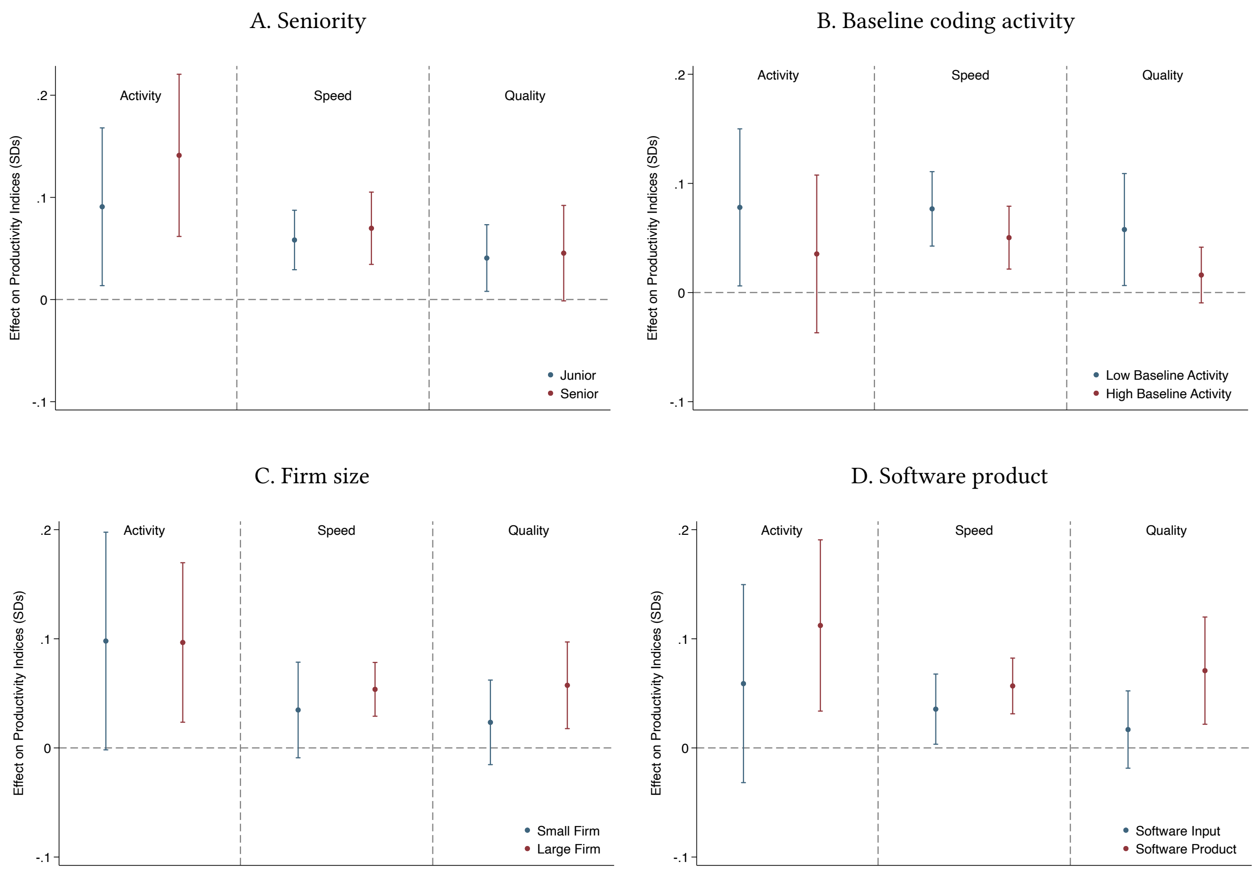
Notes: This figure displays the event study of the effects of AI adoption on employment, using the imputation estimator of Borusyak et al. (2024). Panel A displays the effect for the log of overall employment at each firm, where employment is measured as the number of LinkedIn profiles attached to a firm at a given point in time. Panel B displays the effect for the log of employment for all software engineers, where a worker's employment duration at the firm is defined as the first and last instance they display any activity in the JF data. Panel C displays the effect for the log of employment of junior software engineers, defining employment equivalently to Panel B. Seniority is defined using a keyword search in the job title from the merge with LinkedIn. Panel D displays the effect for the log of employment of senior software engineers, defined equivalently.

FIGURE 12. Comparison of employment estimates



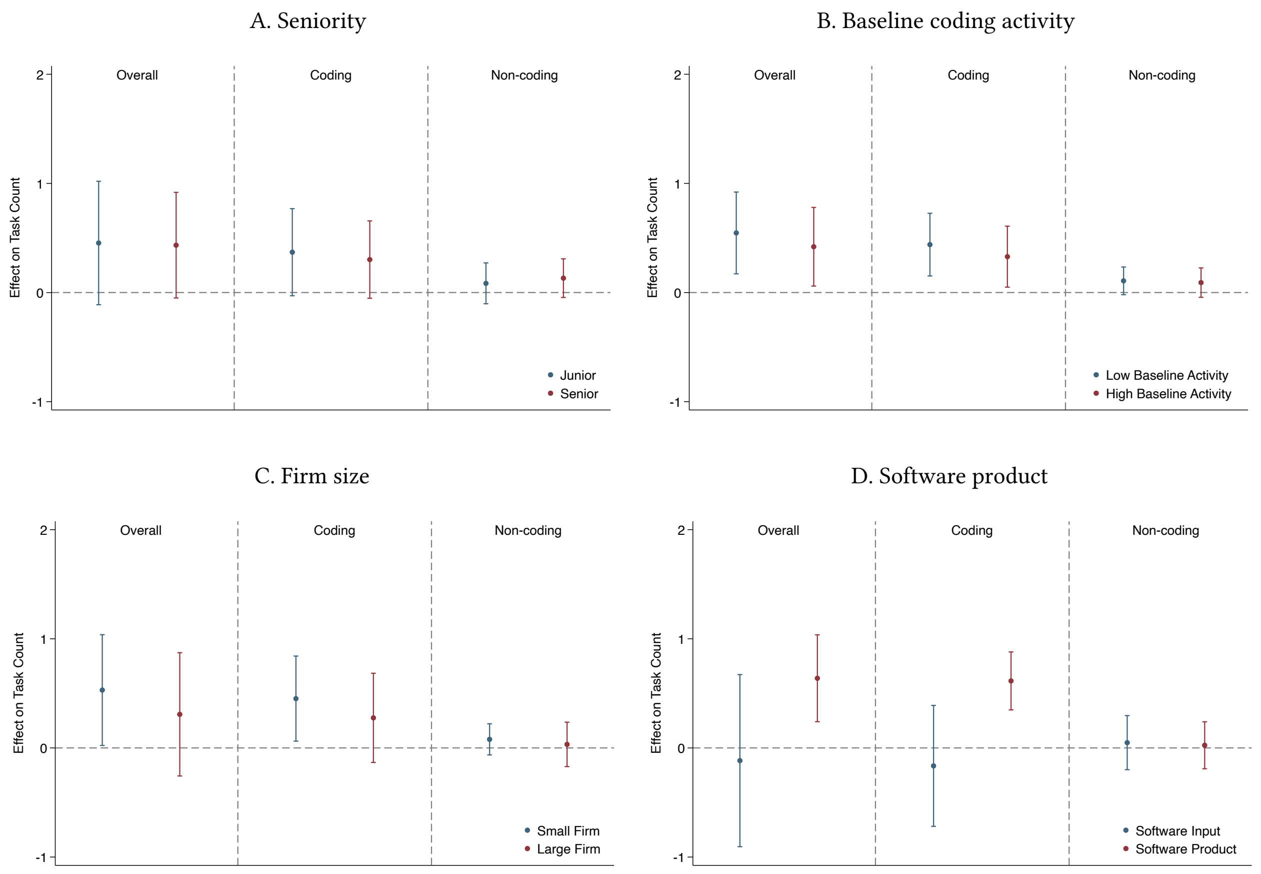
Notes: This figure displays a comparison of estimates of AI tools' employment impacts, from our paper with those of Humlum & Vestergaard (2025); Brynjolfsson, Chandar, & Chen (2025), and Lichtinger & Hosseini Maasoum (2025).

FIGURE 13. Heterogeneity in productivity effects



Notes: This figure displays pooled estimates of the effects of AI adoption on indices for coding activity, quality, and speed, separately by worker and firm characteristics. The indices are calculated using the method of Kling et al. (2007), as described in Section 3. The pooled coefficients are calculated by taking the average of 20 post-adoption coefficients and subtracting the average of 8 pre-adoption coefficients, using estimates produced by the Borusyak et al. (2024). In Panel A, the leftmost point and confidence interval are the point estimate and 95% confidence interval for the estimated effect of AI access on coding activity, for junior-level engineers, classified by job titles. We perform the classification using a keyword-based search, where a worker is classified as "senior" if their job title includes any words indicating a senior or managerial position (e.g., "senior", "lead", "director"). The second-leftmost point and confidence interval instead restricts attention to senior-level engineers. The third and fourth points replicate the first and second, changing the outcome variable to the index for coding quality; the fifth and sixth change the outcome variable to the index for coding speed. Panel B replicates Panel A, instead using an indicator for the worker having above-median coding activity in 2022. Panel C replicates Panel A, instead using an indicator for the firm being above-median size. Panel D replicates Panel A, instead using an indicator or the firm creating software products rather than using software as an input. Firms are classified using the method described in Section C.2.

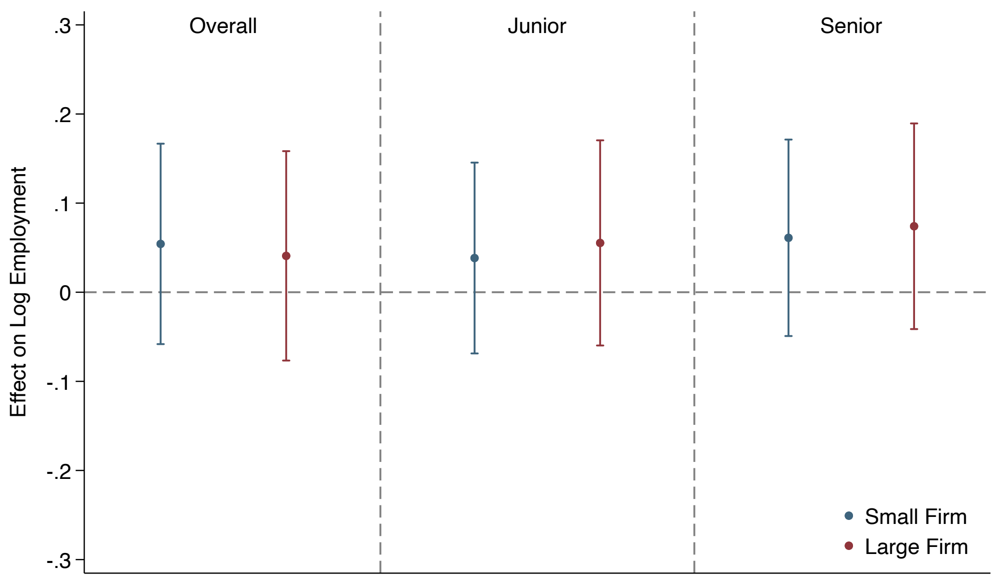
FIGURE 14. Heterogeneity in task output effects



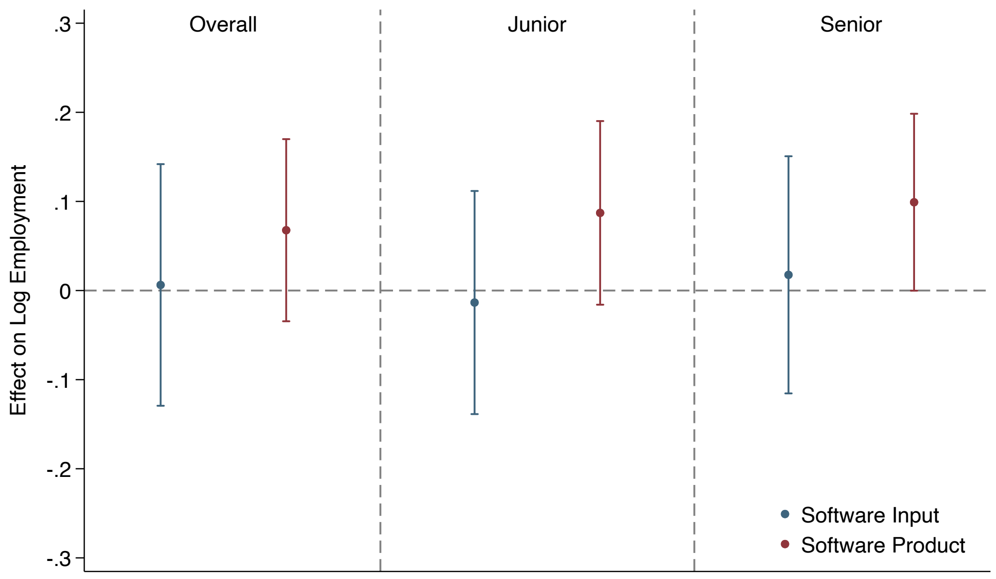
Notes: This figure displays pooled estimates of the effects of AI adoption on overall number of Jira tasks resolved, number of coding Jira tasks resolved, and number of non-coding Jira tasks resolved, separately by worker and firm characteristics. The pooled coefficients are calculated by taking the average of 20 post-adoption coefficients and subtracting the average of 8 pre-adoption coefficients, using estimates produced by the Borusyak et al. (2024). In Panel A, the leftmost point and confidence interval are the point estimate and 95% confidence interval for the estimated effect of AI access on number of Jira tasks completed, for junior-level engineers, classified by job titles. We perform the classification using a keyword-based search, where a worker is classified as "senior" if their job title includes any words indicating a senior or managerial position (e.g., "senior", "lead", "director"). The second-leftmost point and confidence interval instead restricts attention to senior-level engineers. The third and fourth points replicate the first and second, changing the outcome variable to number of coding Jira tasks completed; the fifth and sixth change the outcome variable to number of non-coding Jira tasks completed. Panel B replicates Panel A, instead using an indicator for the worker having above-median coding activity in 2022. Panel C replicates Panel A, instead using an indicator for the firm being above-median size. Panel D replicates Panel A, instead using an indicator or the firm creating software products rather than using software as an input. Firms are classified using the method described in Section C.2.

FIGURE 15. Heterogeneity in employment effects

A. Firm size



B. Software product



Notes: This figure displays pooled estimates of the effects of AI adoption on employment of software engineers overall, employment of junior software engineers, and employment of senior software engineers. Employment is calculated using the merge of the Jellyfish with LinkedIn data. A worker's employment duration at the firm is defined as the first and last instance they display any activity in the JF data. The employment count is the number of workers at the firm in each month, among the workers in the merged sample. Seniority is classified using the job title from LinkedIn. The pooled coefficients are calculated by taking the average of 20 post-adoption coefficients and subtracting the average of 8 pre-adoption coefficients, using estimates produced by the Borusyak et al. (2024). For all In Panel A, the leftmost point and confidence interval are the point estimate and 95% confidence interval for the estimated effect of AI access on the log of number of engineers, for small firms of below-median size. The second-leftmost point and confidence interval instead restricts attention to firms of above-median size. The third and fourth points replicate the first and second, changing the outcome variable to the log of number of junior engineers; the fifth and sixth change the outcome variable to the index for the log of number of senior engineers. Panel B replicates Panel A, instead using an indicator or the firm creating software products rather than using software as an input. Firms are classified using the method described in Section C.2.

A Appendix Tables

TABLE 1. Analysis sample of JF data

Panel A: Size of analysis sample			
	Number of firms	Number of workers	
Analysis sample	518	114,634	
Panel B: Sample observed in various data sources			
	Unit of work	Number of units	Number per worker-month
Jira data	Issues	13,132,828	5.75
GitHub data	Commits	105,821,864	47.79
	Pull Requests	19,641,764	9.80
Calendar data	Events	58,267,744	52.77

Notes: This table displays summary information about the analysis sample. Panel A displays the total number of firms and workers in the sample. Panel B displays the amount of coverage across the various sources of data. It also displays the unit of work associated with each data source, the total number of units of each work activity, and the average units per active worker-month in the analysis sample.

TABLE 2. Example Jira tasks and labels

Task Description	Coding	Length
TASK-101: Implement OAuth 2.0 authentication system As a user, I want to log in using my Google or GitHub account so that I don't need to create a new password. Integrate OAuth 2.0 flow for Google and GitHub providers, create authentication middleware for token validation, implement session management and refresh token logic, add user account linking functionality for existing users, write unit and integration tests for auth flows, and update API endpoints to use new authentication system.	Yes	Short (<4H)
TASK-102: Fix date format display in user profile The user's date of birth is displaying in MM/DD/YYYY format, but should display in DD/MM/YYYY format for European users. Update date formatting function to check user's locale settings and apply correct format in profile view component.	Yes	Short (<4H)
TASK-103: Conduct security audit and document remediation plan Perform a comprehensive security audit of the application to identify vulnerabilities and create a detailed remediation roadmap. Review all API endpoints for security vulnerabilities, analyze data storage and transmission practices, evaluate third-party dependencies for known CVEs, document all findings with severity ratings, create prioritized remediation plan with effort estimates, and present findings to engineering leadership.	No	Long (8-12H)
TASK-104: Update API documentation for new endpoints Three new endpoints were added in the last sprint but are not yet documented in our API docs. Document `/api/v2/notifications/preferences` (GET, PUT) and `/api/v2/notifications/history` (GET). Include request/response examples and parameter descriptions, and update Postman collection with new endpoints.	No	Long (8-12H)

TABLE 3. Validation of supervised learning step

Panel A: Coding model

TN	FP	FN	TP	Agreement	Precision	Recall	F1
167	87	52	491	82.56%	84.95%	90.42%	0.876

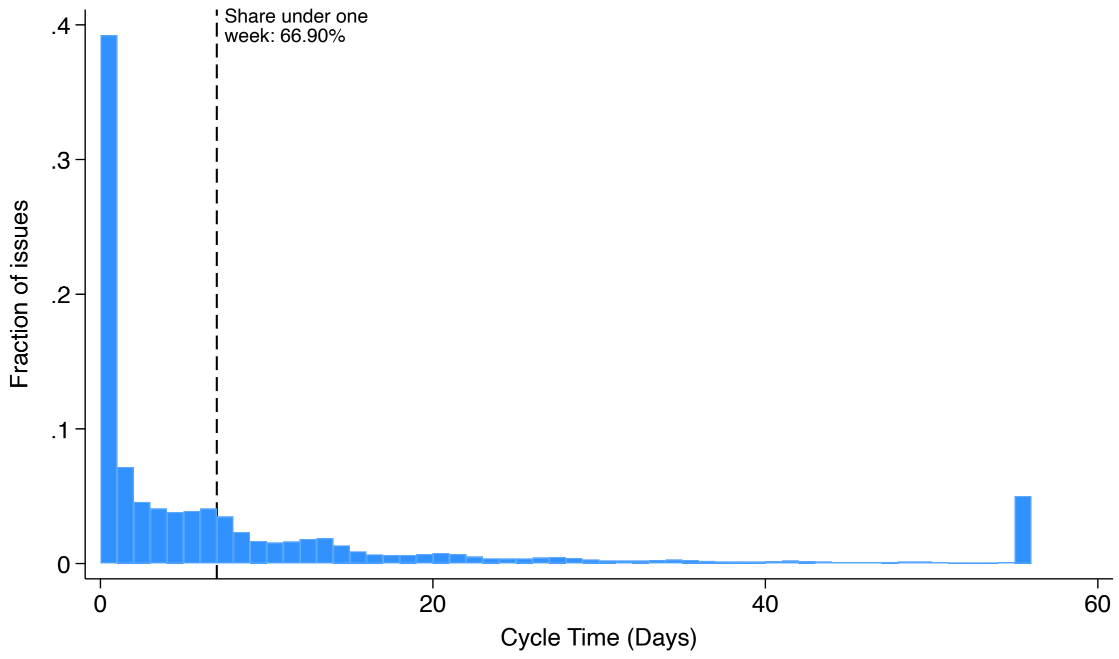
Panel B: Length model

MSE	MAE
0.4105	0.5019

Notes: This table shows the validation of the supervised learning step for assigning the "coding" and "length" labels. Here, the true labels are the GPT-assigned labels, and the predicted labels are the labels assigned by the supervised learning model. Panel A shows the validation of the "coding" label. We report the rate of true negatives, false positives, false negatives, true positives, agreement, precision, recall, and F1 score. Panel B shows the validation of the "length" label. We report the mean squared error and mean absolute error.

B Appendix Figures

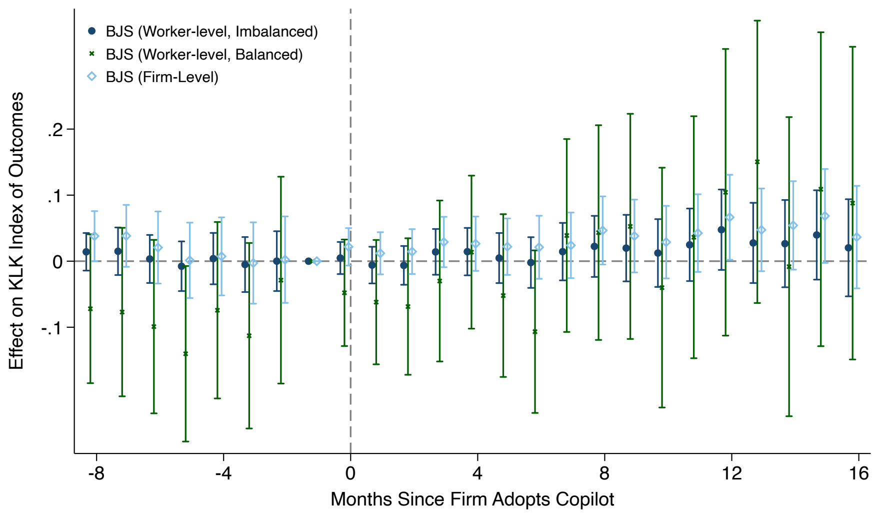
FIGURE 1. Distribution of cycle times



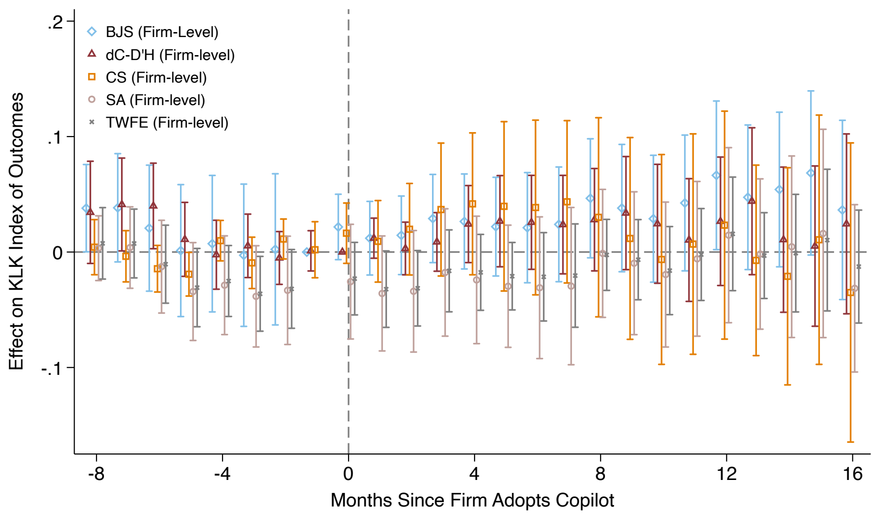
Notes: This figure shows the distribution of cycle times among tasks in the analysis sample.

FIGURE 2. Effect of AI on coding activity under various sample restrictions

A. Estimation at worker-month *vs.* firm-month level



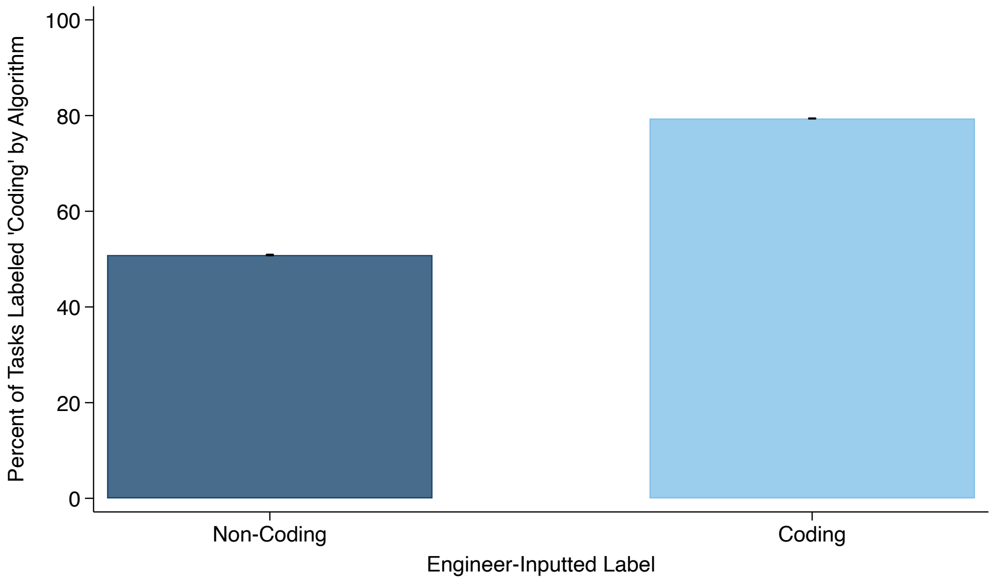
B. Estimation at firm-month level, varying estimator



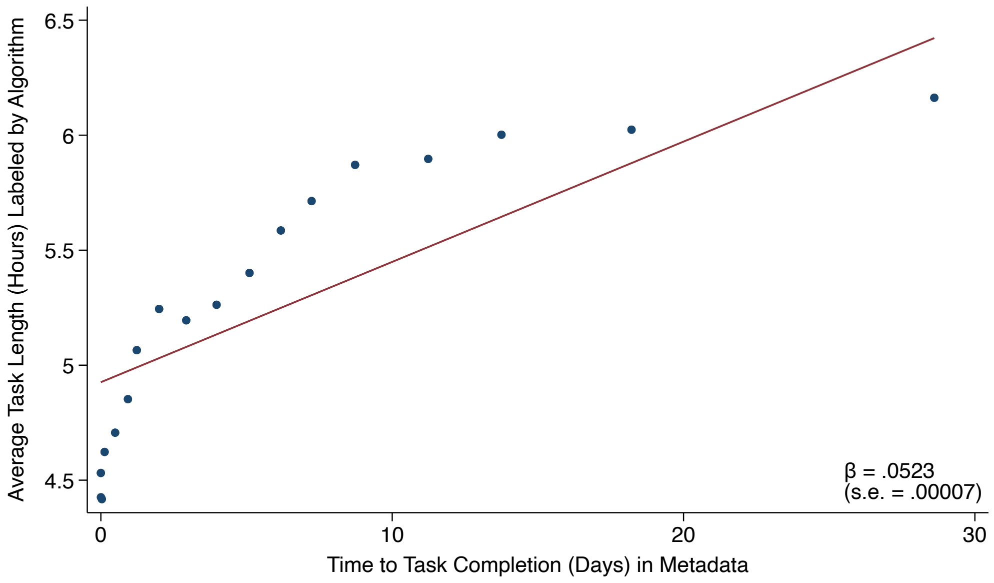
Notes: This figure displays the event study of the effects of AI adoption on an index of coding activity, under various sample restrictions. Panel A displays the event studies at the worker-level, with imbalanced (main specification) and balanced panels. It also displays the event study at the firm-level, with an imbalanced panel. Panel B displays the event studies at the firm-level, with imbalanced panels, using the estimators of Borusyak et al. (2024), de Chaisemartin & D'Haultfœuille (2024), Callaway et al. (2021), and Sun & Abraham (2021), and two-way fixed effects.

FIGURE 3. Validation of coding and length labels

A. Comparison of algorithm-assigned "coding" label to engineer-assigned label



B. Comparison of algorithm-assigned "length" label to task completion time



Notes: This figure shows the validation of "coding" and "length" labels assigned by our machine learning algorithm. Panel A shows the validation of the "coding" label against the "task type" labels inputted by engineers into Jira. We restrict the data to tasks with an associated task type. Then, we group task types into "coding" or "non-coding" using a keyword search. We consider a task "non-coding" if it contains keywords associated with documentation, planning, communication, clients, HR, finances, or legal work. We consider a task "coding" if it contains keywords associated with bug fixes, new features, refactoring, testing, architecture, security, or code review. We classify tasks that contain both forms of keywords as "coding". Panel B shows the validation of the "length" label against days to task completion, as measured in Jira. We winsorize task completion time at 30 days.

C Data Appendix

C.1 Measuring task content

In this section, we describe our method for classifying Jira tasks based on their text descriptions. Each Jira task contains a “summary”, a “description”, a user-inputted “issue type”, linked “project name”, and linked “parent task”. This task information can be manually entered by a worker (the “task creator”) or automatically generated (e.g., an automatic bug report). On average, the text descriptions are 580 characters long, amounting to roughly 1 paragraph.

LLM labels. We begin by using the OpenAI API to classify tasks along two dimensions. Specifically, we use the GPT-4o model, with the following prompt. We label a 0.5% sample of tasks, stratified by company.

```
frame = f"""You are a Jira task classification AI.
        For each provided software engineering task, classify it according to two labels: ‘‘coding’’
             and ‘‘length’’,
        according to the definitions provided below.
        Assume the perspective of a software engineer completing the task, inferring implicit steps
            and context.

        Coding: Whether the task involves writing or modifying code.
        - 0 = No: Task does not involve any code changes - e.g., writing documentation; planning;
        communication; investigating (but not fixing) issues; designing (but not implementing in
            code)
        UI changes.
        - 1 = Yes: Task involves any writing, editing, or deleting lines of code - e.g., writing new
             application
        logic, APIs, services, UI components; fixing bugs; writing or designing test cases;
            infrastructure as code
        or configuration changes; data and database code changes; security fixes requiring code;
            refactoring,
        cleaning up, or otherwise modifying existing code; code review tasks that require edits to
            code;
        any task that ends with opening a pull request or committing code; tasks that are ambiguous
            but
        plausibly require a code change.

        Length: The number of hours that an experienced software engineer (i.e., 5+ years of work
            experience)
        would require to complete the task, without the assistance of generative AI tools.
        - 0 = Short: < 4 hours. Low complexity, routine, single-component change.
        - 1 = Medium: >= 4 and < 8 hours. Medium complexity, multi-component change.
        - 2 = Long: >= 8 hours and < 12 hours. High complexity, cross-team change.
        - 3 = Very Long: >= 12 hours. Very high complexity, large scope change.

        As output, provide the labels in the following JSON format. Strip additional text or
            formatting (e.g., the word "json").
        {{
            "coding":<int>,
            "length":<int>
        }}

        Below is information about the task, including context about its parent task and project.
        Classify the task itself, but use the context to inform your classification.
        Project name: {project_name}
        Parent task summary: {parent_task}
        Task type: {issue_type}
        Task summary: {summary}
        Task description: {description}
"""
```

Supervised extrapolation. We use a supervised learning approach to label the full set of tasks.

First, to represent task descriptions numerically, we extract semantic embeddings using the Sentence-BERT (SBERT) model all-MiniLM-L6-v2, a transformer-based architecture designed for sentence-level encoding. This model maps each input into a vector of 384 fixed dimensions.

Second, we train two feedforward neural networks in PyTorch to predict the GPT-assigned labels from the SBERT embeddings. The first network is a binary classifier that predicts whether a task is a coding task. It takes the 384-dimensional embeddings as input and passes them through two hidden layers of sizes 256 and 128, each followed by batch normalization, a ReLU activation, and dropout. The final layer outputs a single logit, and we train this model using the Binary Cross Entropy with Logits Loss (BCEWithLogitsLoss) function, an Adam optimizer with learning rate 0.001, a batch size of 512, and 10 training epochs. The second network has the same architecture but outputs a single continuous prediction for task length; we train it using a Smooth L1 loss (Huber loss) with the same optimizer, batch size, and number of epochs.

Finally, we use these models to predict coding and length labels for our entire dataset of Jira tasks. For the coding model, we pass the logits through a sigmoid function to obtain predicted probabilities and then binarize them using a cutoff of $p = 0.5$. For the length model, we interpret the continuous output as an underlying latent length and map it to discrete length categories by rounding to the nearest integer and truncating to lie in $\{0, 1, 2, 3\}$.

C.2 Classifying firms as software input vs output

To distinguish whether a firm produces software as a core product or instead uses software as an internal input, we construct a text-based classification using information from firms' industry descriptions, RICS industry codes, slogans, and company descriptions. We compile keyword lists associated with software-producing businesses (e.g., "software," "SaaS," "platform," "API," "developer tools," "cloud," "cybersecurity," "analytics," "fintech," "AI") and with firms whose primary business lies outside software (e.g., "retail," "logistics," "manufacturing," "healthcare," "insurance," "banking," "transportation," "real estate," "consumer goods"). For each firm, we count the frequency of matches to these software-related and industry-related keywords and construct a software centrality number, defined as the ratio of software-related matches to the total number of matches across both lists.

Firms with a high software centrality number (0.7) are labeled software as product, meaning their revenue derives from building and selling software-based products or digital services. Firms with a low index (0.3) are labeled software as input, meaning software development primarily supports non-software operations and does not constitute the core commercial offering. Firms with intermediate values (0.3–0.7) are classified as hybrid or ambiguous, reflecting business models that incorporate both software production and substantial non-software activity (e.g., fintech lenders, IT services firms, or industry-specific digital platforms).

References

Acemoglu, D. (2024, May). *The Simple Macroeconomics of AI* [Working Paper]. National Bureau of Economic Research. Retrieved 2024-11-14, from https://www.nber.org/papers/w32487 doi: 10.3386/w32487

Acemoglu, D., & Autor, D. (2011). Skills, tasks and technologies: Implications for employment and earnings. In *Handbook of labor economics* (Vol. 4, pp. 1043–1171). Elsevier.

Acemoglu, D., Autor, D., Hazell, J., & Restrepo, P. (2022, April). Artificial Intelligence and Jobs: Evidence from Online Vacancies. *Journal of Labor Economics*, *40*(S1), S293–S340. Retrieved 2024-10-10, from https://www.journals.uchicago.edu/doi/10.1086/718327 doi: 10.1086/718327

Acemoglu, D., & Restrepo, P. (2018a, January). *Artificial Intelligence, Automation and Work* [Working Paper]. National Bureau of Economic Research. Retrieved 2022-12-13, from https://www.nber.org/papers/w24196 doi: 10.3386/w24196

Acemoglu, D., & Restrepo, P. (2018b). The race between man and machine: Implications of technology for growth, factor shares, and employment. *American economic review*, *108*(6), 1488–1542.

Acemoglu, D., & Restrepo, P. (2019, May). Automation and New Tasks: How Technology Displaces and Reinstates Labor. *Journal of Economic Perspectives*, *33*(2), 3–30. Retrieved 2022-10-05, from https://www.aeaweb.org/articles?id=10.1257/jep.33.2.3 doi: 10.1257/jep.33.2.3

Acemoglu, D., & Restrepo, P. (2022a, January). Demographics and Automation. *The Review of Economic Studies*, *89*(1). Retrieved 2022-12-13, from https://doi.org/10.1093/restud/rdab031 doi: 10.1093/restud/rdab031

Acemoglu, D., & Restrepo, P. (2022b). Tasks, Automation, and the Rise in U.S. Wage Inequality. *Econometrica*, *90*(5). Retrieved 2022-12-13, from https://onlinelibrary.wiley.com/doi/abs/10.3982/ECTA19815 (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.3982/ECTA19815) doi: 10.3982/ECTA19815

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., … others (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Agrawal, A. K., Korinek, A., & Brynjolfsson, E. (2025). *The economics of transformative ai* (Tech. Rep.). National Bureau of Economic Research.

Anthropic. (2025, May). *Introducing Claude 4.* Retrieved 2025-12-19, from https://www.anthropic.com/news/claude-4

Autor, D. (2022, May). *The Labor Market Impacts of Technological Change: From Unbridled Enthusiasm to Qualified Optimism to Vast Uncertainty* [Working Paper]. National Bureau of Economic Research. Retrieved 2024-10-09, from https://www.nber.org/papers/w30074 doi: 10.3386/w30074

Autor, D., Chin, C., Salomons, A. M., & Seegmiller, B. (2022, August). *New Frontiers: The Origins and Content of New Work, 1940–2018* [Working Paper]. National Bureau of Economic Research. Retrieved 2022-10-05, from https://www.nber.org/papers/w30389 doi: 10.3386/w30389

Autor, D., Katz, L., & Kearney, M. (2006). The polarization of the us labor market. *American economic review*, *96*(2), 189–194.

Autor, D., Levy, F., & Murnane, R. (2003). The skill content of recent technological change: An empirical exploration. *The Quarterly journal of economics*, *118*(4), 1279–1333.

Babina, T., Fedyk, A., He, A., & Hodson, J. (2024, January). Artificial intelligence, firm growth, and product innovation. *Journal of Financial Economics*, *151*, 103745. Retrieved 2024-10-09, from https://www.sciencedirect.com/science/article/pii/S0304405X2300185X doi: 10.1016/j.jfineco.2023.103745

Becker, J., Rush, N., Barnes, E., & Rein, D. (2025, July). *Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity.* arXiv. Retrieved 2025-12-15, from http://arxiv.org/abs/2507.09089 (arXiv:2507.09089 [cs]) doi: 10.48550/arXiv.2507.09089

Bick, A., Blandin, A., & Deming, D. J. (2024, September). *The Rapid Adoption of Generative AI* (NBER Working Paper No. 32966). National Bureau of Economic Research. Retrieved from https://www.nber.org/system/files/working_papers/w32966/w32966.pdf doi: 10.3386/w32966

Bloom, N., Garicano, L., Sadun, R., & Van Reenen, J. (2014, December). The Distinct Effects of Information Technology and Communication Technology on Firm Organization. *Management Science*, *60*(12), 2859–2885. Retrieved 2024-10-10, from https://pubsonline.informs.org/doi/10.1287/mnsc.2014.2013 (Publisher: INFORMS) doi: 10.1287/mnsc.2014.2013

Borusyak, K., Jaravel, X., & Spiess, J. (2024, November). Revisiting Event-Study Designs: Robust and Efficient Estimation. *The Review of Economic Studies*, *91*(6), 3253–3285. Retrieved 2024-11-14, from https://doi.org/10.1093/restud/rdae007 doi: 10.1093/restud/rdae007

Bresnahan, T. F., Brynjolfsson, E., & Hitt, L. M. (2002, February). Information Technology, Workplace Organization, and the Demand for Skilled Labor: Firm-Level Evidence*. *The Quarterly Journal of Economics*, *117*(1), 339–376. Retrieved 2024-10-10, from https://doi.org/10.1162/003355302753399526 doi: 10.1162/003355302753399526

Brynjolfsson, E., Chandar, B., & Chen, R. (2025). Canaries in the coal mine? six facts about the recent employment effects of artificial intelligence. *Stanford Digital Economy Lab. Published August*.

Brynjolfsson, E., Korinek, A., & Agrawal, A. K. (2025). A research agenda for the economics of transformative ai.

Brynjolfsson, E., & Li, D. (2024). The economics of generative ai. *NBER Reporter*(1), 16–19.

Brynjolfsson, E., Li, D., & Raymond, L. (2025, May). Generative AI at Work. *The Quarterly Journal of Economics*, *140*(2), 889–942. Retrieved 2025-07-09, from https://doi.org/10.1093/qje/qjae044 doi: 10.1093/qje/qjae044

Brynjolfsson, E., & Milgrom, P. (2012, December). 1. Complementarity in Organizations. In R. Gibbons & J. Roberts (Eds.), *The Handbook of Organizational Economics* (pp. 11–55). Princeton University Press. Retrieved 2025-03-18, from https://www.degruyter.com/document/doi/10.1515/9781400845354-003/pdf?licenseType=restricted doi: 10.1515/9781400845354-003

Callaway, B., Goodman-Bacon, A., & Sant'Anna, P. H. (2021). Difference-in-differences with a continuous treatment. *arXiv preprint arXiv:2107.02637*.

Chatterji, A., Rock, D., & Talamas, E. (2025). Transformative ai and firms. *NBER Chapters*.

Chen, Z., & Chan, J. (2024, December). Large Language Model in Creative Work: The Role of Collaboration Modality and User Expertise. *Management Science*, *70*(12), 9101–9117. Retrieved 2025-05-06, from https://pubsonline.informs.org/doi/10.1287/mnsc.2023.03014 (Publisher: INFORMS) doi: 10.1287/mnsc.2023.03014

Choi, J. H., & Schwarcz, D. (2023, August). *AI Assistance in Legal Analysis: An Empirical Study* [SSRN Scholarly Paper]. Rochester, NY: Social Science Research Network. Retrieved 2025-05-09, from https://papers.ssrn.com/abstract=4539836 doi: 10.2139/ssrn.4539836

Cui, K. Z., Demirer, M., Jaffe, S., Musolff, L., Peng, S., & Salz, T. (2024, March). The Productivity Effects of Generative AI: Evidence from a Field Experiment with GitHub Copilot. *An MIT Exploration of Generative AI*. Retrieved 2024-07-23, from https://mit-genai.pubpub.org/pub/v5iixksv/release/2 (Publisher: MIT) doi: 10.21428/e4baedd9.3ad85f1c

de Chaisemartin, C., & D'Haultfœuille, X. (2024, February). Difference-in-Differences Estimators of Intertemporal Treatment Effects. *The Review of Economics and Statistics*, *1*(45). Retrieved 2025-11-07, from https://direct.mit.edu/rest/article-abstract/doi/10.1162/rest_a_01414/119488/Difference-in-Differences-Estimators-of?redirectedFrom=fulltext

Dell'Acqua, F., Ayoubi, C., Lifshitz-Assaf, H., Sadun, R., Mollick, E. R., Mollick, L., … Lakhani, K. R. (2025, March). *The Cybernetic Teammate: A Field Experiment on Generative AI Reshaping Teamwork and Expertise* [SSRN Scholarly Paper]. Rochester, NY: Social Science Research Network. Retrieved 2025-05-01, from https://papers.ssrn.com/abstract=5188231 doi: 10.2139/ssrn.5188231

Dell'Acqua, F., McFowland III, E., Mollick, E. R., Lifshitz-Assaf, H., Kellogg, K., Rajendran, S., … Lakhani, K. R. (2023, September). *Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality* [SSRN Scholarly Paper]. Rochester, NY: Social Science Research Network. Retrieved 2025-05-01, from https://papers.ssrn.com/abstract=4573321 doi: 10.2139/ssrn.4573321

de Souza, G. (2025, July). *Artificial Intelligence in the Office and the Factory: Evidence from Administrative Software Registry Data* [SSRN Scholarly Paper]. Rochester, NY: Social Science Research Network. Retrieved 2025-10-21, from https://papers.ssrn.com/abstract=5375463 doi: 10.2139/ssrn.5375463

Dillon, E. W., Jaffe, S., Immorlica, N., & Stanton, C. T. (2025). *Shifting work patterns with generative ai* (Tech. Rep.). National Bureau of Economic Research.

Dube, A., Girardi, D., Jorda, O., & Taylor, A. M. (2023). *A local projections approach to difference-in-differences event studies* (Tech. Rep.). National Bureau of Economic Research.

Eloundou, T., Manning, S., Mishkin, P., & Rock, D. (2024). Gpts are gpts: Labor market impact potential of llms. *Science*, *384*(6702), 1306–1308.

Farquhar, S. (2019, September). *Reaching new heights in the cloud.* Retrieved 2025-07-01, from https://www.atlassian.com/blog/platform/cloud-premium

Fu, Y., Liang, P., Tahir, A., Li, Z., Shahin, M., Yu, J., & Chen, J. (2025). Security weaknesses of copilot-generated code in github projects: An empirical study. *ACM Transactions on Software Engineering and Methodology*, *34*(8), 1–34.

GitHub. (2024, April). *GitHub Copilot Metrics API now available in public beta · GitHub Changelog.* Retrieved 2024-10-09, from https://github.blog/changelog/2024-04-23-github-copilot-metrics-api-now-available-in-public-beta/

GitHub. (2025a, July). *Changing the AI model for Copilot code completion.* Retrieved 2025-07-01, from https://docs-internal.github.com/_next/data/E0dI0-FlZEZ1stG2Qsn5N/en/free-pro-team%40latest/copilot/how-tos/ai-models/changing-the-ai-model-for-copilot-code-completion.json?versionId=free-pro-team%40latest&productId=copilot&restPage=how-tos&restPage=ai-models&restPage=changing-the-ai-model-for-copilot-code-completion

GitHub. (2025b). *Compare GitHub to the competition.* Retrieved 2025-07-01, from https://resources.github.com/devops/tools/compare/

GitHub. (2025c, July). *GitHub language support - GitHub Enterprise Cloud Docs.* Retrieved 2025-07-01, from https://docs-internal.github.com/en/enterprise-cloud@latest/get-started/learning-about-github/github-language-support

Goos, M., Manning, A., & Salomons, A. (2014). Explaining job polarization: Routine-biased technological change and offshoring. *American economic review*, *104*(8), 2509–2526.

Harrigan, J., Reshef, A., & Toubal, F. (2016, March). *The March of the Techies: Technology, Trade, and Job Polarization in France, 1994-2007* [Working Paper]. National Bureau of Economic Research. Retrieved 2022-10-05, from https://www.nber.org/papers/w22110 doi: 10.3386/w22110

He, H., Miller, C., Agarwal, S., Kästner, C., & Vasilescu, B. (2025). Speed at the cost of quality? the impact of llm agent assistance on software development. *arXiv preprint arXiv:2511.04427*.

Hoffman, M., Kahn, L. B., & Li, D. (2018, May). Discretion in Hiring*. *The Quarterly Journal of Economics*, *133*(2), 765–800. Retrieved 2024-10-10, from https://doi.org/10.1093/qje/qjx042 doi: 10.1093/qje/qjx042

Hoffmann, M., Boysel, S., Nagle, F., Peng, S., & Xu, K. (2024). *Generative AI and the Nature of Work.* Retrieved 2024-11-07, from https://www.ssrn.com/abstract=5007084 doi: 10.2139/ssrn.5007084

Humlum, A., & Vestergaard, E. (2025). *Large language models, small labor market effects* (Tech. Rep.). National Bureau of Economic Research.

IBM. (2024, January). *Data Suggests Growth in Enterprise Adoption of AI is Due to Widespread Deployment by Early Adopters.* Retrieved 2025-12-22, from https://newsroom.ibm.com/2024-01-10-Data-Suggests-Growth-in-Enterprise-Adoption-of-AI-is-Due-to-Widespread-Deployment-by-Early-Adopters

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., & Narasimhan, K. (2023). Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770.*

Jira. (2025). *Jira work items.* Retrieved 2025-07-03, from https://www.atlassian.com/software/jira/guides/issues/overview#what-is-an-work%20item

Kessler, S. (2023). As artificial intelligence proliferates, the u.s. job market keeps up—but not for everyone. *The New York Times.* Retrieved from https://www.nytimes.com/2023/06/10/business/ai-jobs-work.html (Business section)

Kim, A. G., Muhn, M., & Nikolaev, V. V. (2024, October). *From Transcripts to Insights: Uncovering Corporate Risks Using Generative AI* [SSRN Scholarly Paper]. Rochester, NY: Social Science Research Network. Retrieved 2025-05-09, from https://papers.ssrn.com/abstract=4593660 doi: 10.2139/ssrn.4593660

Kinder, M., de Souza Briggs, X., Muro, M., & Liu, S. (2024, October). *Generative AI, the American worker, and the future of work.* Retrieved 2025-04-27, from https://www.brookings.edu/articles/generative-ai-the-american-worker-and-the-future-of-work/

Kling, J. R., Liebman, J. B., & Katz, L. F. (2007). Experimental analysis of neighborhood effects. *Econometrica*, *75*(1), 83–119.

Korinek, A., & Suh, D. (2024). *Scenarios for the transition to agi* (Tech. Rep.). National Bureau of Economic Research.

Lenarduzzi, V., Saarimäki, N., & Taibi, D. (2019). The technical debt dataset. In *Proceedings of the fifteenth international conference on predictive models and data analytics in software engineering* (pp. 2–11).

Lichtinger, G., & Hosseini Maasoum, S. M. (2025, August). *Generative AI as Seniority-Biased Technological Change: Evidence from U.S. Résumé and Job Posting Data* [SSRN Scholarly Paper]. Rochester, NY: Social Science Research Network. Retrieved 2025-10-21, from https://papers.ssrn.com/abstract=5425555 doi: 10.2139/ssrn.5425555

Ling, Y., Kale, A., & Imas, A. (2025). Underreporting of ai use: The role of social desirability bias. *Available at SSRN 5464215.*

Lüders, C. M., Bouraffa, A., & Maalej, W. (2022). Beyond duplicates: Towards understanding and predicting link types in issue tracking systems. In *Proceedings of the 19th international conference on mining software repositories* (pp. 48–60).

Noy, S., & Zhang, W. (2023, July). Experimental evidence on the productivity effects of generative artificial intelligence. *Science*, *381*(6654), 187–192. Retrieved 2024-10-03, from https://www.science.org/doi/10.1126/science.adh2586 (Publisher: American Association for the Advancement of Science) doi: 10.1126/science.adh2586

OpenAI. (2025, December). *Introducing GPT-5.* Retrieved 2025-12-19, from https://openai.com/index/introducing-gpt-5/

Ortu, M., Destefanis, G., Adams, B., Murgia, A., Marchesi, M., & Tonelli, R. (2015). The jira repository dataset: Understanding social aspects of software development. In *Proceedings of the 11th international conference on* *predictive models and data analytics in software engineering* (pp. 1–4).

Otis, N., Clarke, R., Delecourt, S., Holtz, D., & Koning, R. (2024, February). *The Uneven Impact of Generative AI on Entrepreneurial Performance* [SSRN Scholarly Paper]. Rochester, NY: Social Science Research Network. Retrieved 2025-05-06, from https://papers.ssrn.com/abstract=4671369 doi: 10.2139/ssrn.4671369

Overflow, S. (2025). *2025 Stack Overflow Developer Survey.* Retrieved 2025-12-22, from https://survey.stackoverflow.co/2025/

Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023, February). *The Impact of AI on Developer Productivity: Evidence from GitHub Copilot.* arXiv. Retrieved 2024-07-23, from http://arxiv.org/abs/2302.06590 (arXiv:2302.06590 [cs]) doi: 10.48550/arXiv.2302.06590

Perkel, S. (2025). *Ai will allow software engineers to be more creative and reach the 'magical flow state' easier, github ceo says.* Insider Inc. Retrieved 2025-12-17, from https://www.businessinsider.com/github-ceo-ai-lets-engineers-be-more-creative-2025-5

Perry, N., Srivastava, M., Kumar, D., & Boneh, D. (2023). Do users write more insecure code with ai assistants? In *Proceedings of the 2023 acm sigsac conference on computer and communications security* (pp. 2785–2799).

Pichai, S., Hassabis, D., & Kavukcuoglu, K. (2025, November). *A new era of intelligence with Gemini 3.* Retrieved 2025-12-19, from https://blog.google/products/gemini/gemini-3/

Rapoport, G., Bicanic, S., & Talabi, M. (2025, May). *Survey: Generative AI's Uptake Is Unprecedented Despite Roadblocks.* Retrieved 2025-12-22, from https://www.bain.com/insights/survey-generative-ai-uptake-is-unprecedented-despite-roadblocks/ (Section: Brief)

Roldan-Mones, A. (2024, August). When GenAI increases inequality: evidence from a university debating competition.

Roose, K. (2025). For some recent graduates, the A.I. job apocalypse may already be here. *The New York Times.* Retrieved from https://www.nytimes.com/2025/05/30/technology/ai-jobs-college-graduates.html (Technology column)

Roth, J., Sant'Anna, P. H., Bilinski, A., & Poe, J. (2023). What's trending in difference-in-differences? a synthesis of the recent econometrics literature. *Journal of Econometrics.*

Sarkar, S. K. (2025, November). *AI Agents, Productivity, and Higher-Order Thinking: Early Evidence From Software Development* [SSRN Scholarly Paper]. Rochester, NY: Social Science Research Network. Retrieved 2025-11-20, from https://papers.ssrn.com/abstract=5713646 doi: 10.2139/ssrn.5713646

Song, F., Agarwal, A., & Wen, W. (2024). The impact of generative ai on collaborative open-source software development: Evidence from github copilot. *arXiv preprint arXiv:2410.02091.*

Stanley, M. (2025, October). *More Software, More Developer Jobs.* Retrieved 2025-12-17, from https://www.morganstanley.com/insights/articles/ai-software-development-industry-growth

Stevenson, B. (2025). What is there to fear in a post agi world. *NBER Chapters.*

Sun, L., & Abraham, S. (2021, December). Estimating dynamic treatment effects in event studies with heterogeneous treatment effects. *Journal of Econometrics*, *225*(2), 175–199. Retrieved 2025-11-07, from https://www.sciencedirect.com/science/article/pii/S030440762030378X doi: 10.1016/j.jeconom.2020.09.006

U.S. Bureau of Labor Statistics. (2024, May). *Occupational Employment and Wage Statistics (OEWS) Tables.* https://www.bls.gov/oes/tables.htm.

Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., & Press, O. (2024). Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, *37*, 50528–50652.

Yeverechyahu, D., Mayya, R., & Oestreicher-Singer, G. (2024). The impact of large language models on open-source innovation: Evidence from github copilot. *arXiv preprint arXiv:2409.08379*.

Zeff, M. (2025, July). *GitHub Copilot crosses 20M all-time users.* Retrieved 2025-12-14, from https://techcrunch.com/2025/07/30/github-copilot-crosses-20-million-all-time-users/

Zeira, J. (1998). Workers, machines, and economic growth. *The Quarterly Journal of Economics*, *113*(4), 1091–1117.