Measuring Learning on the Job

By

Jiageng Liu

B.S. Mathematics, University of California, Los Angeles, 2017
M.S. Statistics, University of Chicago, 2019

SUBMITTED TO THE DEPARTMENT OF MANAGEMENT IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MANAGEMENT RESEARCH

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024

Authored by: Jiageng Liu
Department of Management
May 2nd, 2024

Certified by: Antoinette Schoar
Department of Management
Thesis supervisor

Accepted by: Eric So
Professor, Global Economics and Finance
Faculty Chair, MIT Sloan PhD Program

Measuring Learing on the Job

By

Jiageng Liu

Submitted to the Department of Management on May 2nd, 2024
in Partial Fulfillment of the Requirements for the Degree of Master of Science in Management Research

ABSTRACT

I study on-the-job learning at IT firms. Using detailed online activity data of 144,000 employees matched with 25,000 firms across the globe, I measure the intensity and the direction of technology acquisition, a key input to innovation. A standardized measure shows that employee-entrepreneurs who join small, young firms spend more time learning about new software than similar employees who join large incumbent firms. They engage with more diverse and rarely combined topics, behaviors that are found to be associated with more radical innovations. Within firms, more actively learning employees work on more projects and start reviewing others' code sooner. The results are consistent with channels of firm-employee matching and job security at incumbent firms. They complement Akcigit and Goldschlag (2023) that finds inventors apply for fewer patents and receive higher wages after joining incumbent firms. A heterogeneous supply of unobserved learning cannot explain all results. I also document life-cycle patterns of learning behavior that are consistent with predictions of the standard labor theory. Such predictions had been challenging to test past formal education.

Thesis supervisor: Antoinette Schoar

Title: Stewart C. Myers-Horn Family Professor of Finance

List of Figures

List of Tables

1. Introduction

How does high-skilled labor manage and invest in human capital on their job? To firms, knowledge retained by high-tech labor constitutes a core competitive edge over their competitors. To macroeconomists, high-skilled human capital is a key input in producing new ideas and the driver of economic growth. To financial economists, the question matters to workers' consumption and portfolio choices as human capital is the main source of income and wealth.

Economists dating back to Mincer (1962) have recognized on-the-job human capital accumulation as a central determinant of wage and productivity. A burgeoning literature has studied skill acquisition among high-skilled labor, but most focus on early phases such as family background (Bell et al., 2019), education (Hampole, 2022), and initial job choices (Deming and Noray, 2020). In contrast, on-the-job human capital investment receives relatively little empirical treatment in the literature. The main challenge, as Mincer puts it, is that "such data are rarely available." In an ideal setting, I would need to (1) measure individual-level investment in various skills and quantify their labor market outcome that maps to those skills; (2) make the observations comparable across firms because both reported training efforts and outcome depend on the firm's management and data collection practices; and (3) obtain relatively high-frequency data because while learning-by-doing can be inferred from working experiences, skill investment such as learning how to use a tool or understanding a competitor's business model is rather short-term on the job.

This paper attempts to measure on-the-job human capital investment by leveraging a dataset that meets these conditions. Our primary data source is GitHub, the largest provider of software development and management services and the largest host of software source code in the world. I gain access to the minute-by-minute activity log of public online interactions between IT professionals and software development projects from 2011 to 2023. By identifying patterns of employed working, I find 144,000 employees matched with 317,000 working projects across 25,000 employees around the globe. Around 1/3 of the employees are in the U.S., 1/3 are in Europe, and 1/3 are from the rest of the world. They left 435 million records of interactions with their working projects with their employer, 88 million interactions with outside projects (projects not owned by themselves or their employers) while they were working, and another 48 million interactions with outside projects before the inferred period of employment. The employees interacted with projects such as OpenAI's *GPT-3*, the core technology behind ChatGPT; *pandas*, the most widely used data science toolkit in Python; and *Kubernetes*, the most popular system for automated software deployment[1].

[1]The three projects can be found at https://github.com/openai/gpt-3, https://github.com/panda

Since these projects are published and developed on GitHub, the platform represents frontier knowledge and the primary source of skill development for IT professionals.

Because the dataset is comprised of unprocessed event logs, I devise novel methods to credibly and efficiently identify work-related versus learning-related activities, as well as demographic information. I identify employee-employer matches by leveraging GitHub's organization account feature that gives firms detailed control over editing and reviewing privileges on different projects. I convert the number of interaction logs to a lower-bound estimate of the time spent on GitHub because different actions cost different amounts of effort to perform. For example, one of the most common forms of learning is "starring," i.e., bookmarking a project, which only takes a click of a button. In contrast, drafting a question on an unfamiliar project is more difficult. Our time-based measure adapts to these differences and captures the effort of learning.

For each month an employee is active on GitHub, I measure the scope of learning by the interactions with projects that they have never interacted with before. The effort of learning is measured as the hours spent on these new projects. I assume each software package embodies a skill or an idea that merits an interaction from the employee. I exclude the projects they are working on for their employer. Interactions with an external project after the first month are disregarded to avoid situations where learning turns into a hobby or a pet project. Finally, I define working projects as the employer's projects where the employee accumulated at least 20 hours of interactions.

To establish the relationship between skill investment and outcomes on the job, I also measure the content of learning and working by natural language processing (NLP) techniques. I collect short descriptions and read-me files of 920,000 software projects that the employees either worked on or interacted with. I use the TF-IDF method to extract the keywords from the descriptions and manually classify the 800 most common keywords into 10 topics, such as front-end development, mobile apps, and physical sciences. I assign each software project to a category by finding the most prominent keyword.

Several empirical regularities confirm that our measures capture the working and learning efforts of individual employees. For example, in a random sample of 100 employees who report their real names in their GitHub profile, 92 are also seen on LinkedIn, and 85 of them report their employment experiences that match our data. For over 95% of the employees who left the timezone information in their activity log, their most intensive work-related activities fall in the 8 AM-6 PM range in that timezone. I also observe that picking up a new topic on the work correlates with an increased interaction with outside projects related to the new topic, but not the old one.

s-dev/pandas, and https://github.com/kubernetes/kubernetes on GitHub, respectively.

In the second part of the paper, I use the metrics of working and exploration to establish novel stylized facts about on-the-job skill investment. I begin by documenting a large variation in employees' propensity to explore software projects outside their work. An employee at the 75th percentile interacts with six times as many external projects as an employee at the 25th percentile, who explores around 4 external projects per year in the first two years on the job. The number of hours spent on working projects has a much smaller difference.

On the firm level, I find that employees in young, small organizations spend significantly more time to explore outside projects, when there is no significant difference between their hours of work per month. I propose three candidate hypotheses to explain this fact: (1) a matching channel, where more exploratory employees self-select into employment with younger and smaller firms because their personally acquired skills are more likely to valued in the firm; (2) a job-security channel, where employees who join larger and older firms are offered more job security and have less incentive to increase their outside options; (3) a substitution channel, where employees at large firms substitute learning on GitHub for unobserved investment, such as proprietary technology infrastructure and colleagues with more diverse skills. Our results are consistent with the existence of both matching and treatment, and show that learning substitution cannot explain all effects.

I first show that employees who join smaller, younger firms are more inquisitive two years before they start working, when they know neither their future employer's compensation structure nor internal training programs. The effect is the strongest among "entrepreneurs," or those who are the first in their firm to work on GitHub, and presumably the most powerful in driving firm-level innovations. Also, I show that employees in smaller, younger firms explore topics more different from their work than their counterparts in large firms. To the extent that learning indeed contributes to better technology applied at work, employees in larger firms seem to focus more on (firm-specific) adoption than their (general) skill investment, although I cannot rule out the alternative that large firms provide more unobserved general training.

Since technology and human capital are highly complementary in the software industry, firms that hire employees with high learning capacity are often better adapting to new technology. Therefore, I should see a treatment effect conditional on joining the firm. I show it is indeed the case. Both the firm size and the firm age measured at the time when an employee joins the firm are negatively correlated with their on-the-job exploration, a result that is robust even when I control for firm fixed effects and individuals' pre-job exploration. I also show that conditional on continued employment, employees explore fewer new projects as they age. Using the layoff shocks of 2022-2023 that impact 60 firms in our sample, I show that staying employees maintain their working effort, but start to explore more topics

outside of their workplace.

Are employers aware of the learning activities of their employees? I provide some evidence that they do. I find that more pre-job exploration predicts faster promotion. In a regression with the firm fixed effect, I find that having explored one more project per month *before* starting the job predicts 2 days sooner promotion, defined by the first review of a code submission, and predicts work experiences with 0.04 more projects in the first two years. Using an event-study design, I show that employees increase their learning of new topics that they are about to work on, when the learning of the old topics do not change.

Our main contribution in this paper is to devise a new method to measure learning and working effort from log data commonly found in computer-based management systems. Our method starts from parsimonious assumptions on learning behavior and extracts comparable measures across 25,000 different firms around the globe. Armed with this measure, I present a set of stylized facts that are consistent with predictions from standard human capital theories and highlight the features of the high-skilled labor market.

First, I speak to the long strand of literature on human capital accumulation. Ben-Porath (1967) first considered the trade-off between the time spent on supplying labor and building skills. I verify one of its central predictions, namely the decline in the share of time invested in learning over the life-cycle. Empirical treatment of on-the-job skill building is relatively sparse. One strand of literature including Autor (2001) and Acemoglu and Pischke (1999) emphasizes the role of labor market frictions in firm-sponsored training, where training acts as a screening or selection device on the worker's ability. The asymmetric information channel arguably plays a lesser role in our high-skilled labor setting. In more recent work, Jarosch et al. (2021) uses establishment-employee level administrative data to show strong complementarity in workers' earnings growth, where it's assumed that employees acquire high-value skills from working with high-growth co-workers.

Our work speaks to the innovation economics literature. A long strand of literature has made use of patent data. However, in the U.S., software patents are relatively rare, especially after a U.S. Supreme Court rule[2]. For example, OpenAI holds no patents in the USPTO database as of January 2024. Our study attempts to fill in the gap. A recent study, Akcigit and Goldschlag (2023b), shows that fewer inventors are found in small, young firms. Our results complement theirs by showing how job security offered by large, old firms could quench the exploration incentive and thus lower the innovation (Fleming and Sorenson (2004), Kelly et al. (2021)). In terms of data innovation, this paper is similar to Biasi et al. (2021) which uses course syllabus data to show a large difference in the gap between education and frontier knowledge in higher-education institutions. On GitHub, frontier knowledge is developed and

[2]See Alice Corp. v. CLS Bank Int'l, 573 U.S. 208 (2014)

picked up by engineers, and I contribute to the study by highlighting the difference in the labor market outcome.

Our work also joins the literature that considers the economics of open-source software development. The pioneering work of Lerner and Tirole (2002) theorizes open-source as a signaling tool for aspiring developers and a bundling strategy for software firms. I take for granted that firms develop open-source software to increase adoption but focus on the incentives of individual developers who adopt new skills. I are also careful in removing interactions that could be signaling in the labor market–for example, I disregard any interaction where the employee works on their own project unless the project is part of the work that the employee does for an organization. Our work is close to Hann et al. (2004), which finds that lines of code written does not predict higher wages, but peer-reviewing results do, and Nagle (2018) that finds positive return to learning on the firm level where participation is interpreted as learning. I argue that interacting with external projects constitutes a better measure of human capital.

The rest of the paper is organized as the follows. Section 2 discusses how I acquire and process the data. Section 3 shows our results about on-the-job learning from four perspectives. Section 4 concludes.

2. Data and measurement

Our primary data source is the collection of all publicly available event logs on GitHub since February 2011.

The core of GitHub is Git, the most widely management system that tracks software development, accounting for 94% of the market share by 2022. Git helps software engineers collaborate on a project by tracking each author's detailed contributions, also known as "commits" and keeping a history of the software from its first line of code. The design philosophy can be found many other business management systems. Loeliger and McCullough (2012) provides a detailed introduction. GitHub is a hosting site for Git-managed software and adds extra features such as task management, feature requests, and bug tracking. The website is compared to a discussion forum built around software projects, also known as "repositories." Aside from writing code, software developers can use their accounts to bookmark, download, or raise questions about others' software. GitHub discourages users from creating multiple accounts and suggests users merge existing ones for both open-source projects and paid employment[3]. As such, I observe jointly both the learning and the working

[3]See https://docs.github.com/en/get-started/learning-about-github/types-of-github-accounts

records of open-source developers.

I note that Git is commonly considered the third-generation version control system. Software engineers before 2010, such as those studied in Lerner et al. (2006), did not use Git but earlier systems like SVN. To the extent that researchers can access data on mailing lists that offer similar functionalities as GitHub, our study can be replicated on those data.

2.1. Measuring topics of learning

Unlike patents where the patent office assigns classifications and inventors clarify their claims, software projects do not have any systematic classifications according to our knowledge. I thus use its name, its short description, the keywords, and the ReadMe file content to infer its topics. See Appendix Figure 8 for an exhibit of these items in a typical project. I use natural language processing (NLP) methods on the project ReadMe files to identify the underlying ideas of the project.

To achieve this, I collect these terms for around 920,000 software projects. I strip out links and special characters. Next, I collect 800 common keywords found in the descriptions of top repositories. I manually classify them into 10 categories: Frontend Development, DevOps & Cloud, Backend & Databases, Data Science, AI & Computing, Blockchain & Cryptocurrency, Media & Gaming, Mobile Apps, Internet-of-Things, Quality Assurance, and Physical Sciences. I apply the TF-IDF method and assign each project a weighted vector of the ten topics. Table 1 lists the top 10 most common keywords for in each category.

2.2. Identifying firms, employees, and projects

I start by considering the hours spent on GitHub by employees. I assign the hour when the interaction happened between a user and a project to be one hour of effort paid by the user on the project. If there are multiple interactions in one hour, I evenly split that hour by the total number of interactions from that user. For example, if a user answers two questions about their working project A, and bookmarks another firm's project B in 9 AM of a day, then I count 40 minutes spent on A and 20 minutes spent on B. The method likely underestimates editing-related activities because the user can edit many lines of code before posting a change[4], but it should have a lower bias on activities that happen during work, such as bookmarking another firm's project in the middle of the work.

[4]I observe that users tend to post more lines of code in the early stage of a project, and fewer lines in multiple postings in the later stage of the project. I interpret it as evidence for staged innovation, where the project matures and requires less overhaul and more tinkering. I ignore such differences in this paper and focus on the lower-bound estimate of time spent.

Table 1: *Most common keywords associated with each topic.*

Topic	Top 10 keywords
Front-end Appearance	user, react, javascript, ui, client, html, npm, js, frontend, browser
Back-end Databases	api, server, io, c, java, database, go, php, security, backend
Operations & Cloud	ci, git, docker, workflow, cloud, kubernetes, linux, plugin, network, distributed
Quality Assurance	sentry, zephyr, cypress, selenium, jest, playwright, quality assurance, mocha, chai, puppeteer
Data Science	data, python, r, analytics, visualization, scala, algorithm, julia, py, conda
AI & Computing	model, machine learning, ai, hardware, matrix, intelligence, deep learning, pytorch, hpc, optimization
Mobile Apps	building, android, testing, mobile, ios, swift, kotlin, gradle, objective, xamarin
Internet of Things	iot, firmware, driver, emulator, embedded, robotics, robot, raspberry, arduino, raspberry pi
Blockchain & Crypto	blockchain, protocol, fork, ethereum, bitcoin, decentralized, chain, cryptocurrency
Media & Gaming	svg, game, discord, product, format, video, png, minecraft, 3d, ss13
Physical Sciences	simulation, science, lab, scientific, physic, quantum, mathematics, math, bioinformatics, statistics

I identify firms on GitHub via organization accounts because setting up these accounts implies anticipation of growth. Organization accounts are accounts that can assign members to different authorization levels for different projects. Organizations cannot edit or discuss code. They can only invite users to be their members, create new projects, or make a project publicly available, so their founders and employees must work from their personal accounts. Personal accounts can edit code but do not have the flexibility and lack access to software management features.[5] The method is similar to Guzman and Stern (2020) which uses company registry information to infer high-growth entrepreneurship.

One limitation of our method is that large firms often control multiple organization accounts. For example, YouTube, Google Chrome, and Tensorflow (a Google-developed deep learning platform) are distinct organizations on GitHub. The ownership information of these accounts, also known as Enterprise accounts, is not public on GitHub. I cluster these organizations by the link of their website pages[6] and by manual lookup. I drop education-oriented firms such as FreeCodeCamp and Learn.co because their users may not be employees.

I identify employees by their activity at identified firms. In particular, I add multiple restrictions to remove short-term projects or class projects. I first find users who have

[5]Personal accounts can be upgraded and re-branded as organizations. Successful firms that started as personal projects often rebrand. I have not covered such cases.

[6]For example, the link of YouTube's website on GitHub is https://developers.google.com/youtube/

visited GitHub for more than 90 days and interacted with an organization's projects for over 45 hours. Next, I find the longest consecutive period that I identify as working: during this period, the employee interacts with one of the employer's projects on at least 10 different days per month. The period must last for more than 3 months and contain at least 3 months in which the employee worked on the projects for more than 15 days. For example, if I regard Alice as being employed at Microsoft from June to August, she must interact with one of Microsoft's projects in patterns such as 16 days in June, 20 days in July, and 18 days in August. If the patterns are 30 days in June, 5 days in July, and 7 days in August, then I do not count it as employment because the "employment" period is only one month. In the final step, I manually checked a random sample of 100 employee-employer matches against LinkedIn data.

I observe 15 types of interactions between users and projects. I classify them into four groups: (1) exploration, such as bookmarking, (2) editing, (3) discussion, and (4) management. Nevertheless, I do not place restrictions on how the user interacts with the projects, because I would like to capture all sorts of work-related activities from answering user questions to managing code.

2.3. Data summary

Taken together, our data contains 6.8 billion interactions between GitHub's registered users and its source code repositories (hereafter, projects) from February 2011 to November 2023. Our data cover over 60 million users, and 14.4 million of them have used GitHub once in the first half of 2023. If all of them are IT professionals, and I take GitHub's claim that 1/3 of its users are from the U.S., that would imply all of the 4.6 million IT professionals in the U.S., according to the Bureau of Labor Statistics, have used GitHub at least once during that period.

Table 2 provides summary statistics of the sample of identified employees.

2.4. Limitations

Our method does not rely on data sources other than GitHub and it comes with several limitations. First, our data are not necessarily a representative sample of the software industry or software engineers because I only cover open-sourced projects. For example, I do not capture proprietary commercial software such as Microsoft Windows or Oracle Database, but I do observe the development of their open-source competitors, Linux and MySQL.

Our employment filter is rather strict and misses out on employment in several cases: (1) employees whose work is only partially available on GitHub, (2) employees after being

Table 2: *Summary statistics of identified employees*

Panel A: all identified employees

	Count	Mean	SD	Min	25%	50%	75%	Max
Number of employers	144080	1.16	0.50	1	1	1	1	17
Months before the job	144080	32.89	32.89	0	3	24	54	149
Months on the job	144080	34.80	27.73	2	14	26	47	153
Firm age upon joining	144080	37.52	37.60	0	3	26	63	149
Firm size upon joining	144080	80.32	259.11	1	2	6	21	1876
# of work projects	144080	4.09	5.53	0	1	2	5	266
Hours working per year	144080	421	352	0	243	345	508	8500
# of projects explored	144080	171.13	451.82	0	19	56	156	28228
Hours learning per year	144080	37	58	0	7.1	18.9	43.1	3560

Panel B: sample employers identified on GitHub

	Microsoft	Google	Facebook	OpenAI	Astron	PHPMailer
Employee count	3951	1893	554	7	1	1

promoted to higher management positions, and (3) work in the early stages of a project, when only one employee works on it and only posts change on GitHub at infrequent intervals.

I also risk the false identification of employees. GitHub is free to use and organizations are free to set up, so I do not necessarily observe paid employment. I have made an effort to eliminate personal projects via various filters, but I do not observe contracts or wages.

For some projects, not the entire development history was public. Our data only covers activities after the public release of a project. I am collecting the development activities before the release.

Finally, I have yet to identify name changes to users, organizations, or projects. These often happen at the time of merger and acquisition or a change of project status (for example, from an incubator to a mature project). I have not covered such cases and assume that each username or project name is uniquely associated with a user or project.

3. Determinants of on-the-job learning

I first describe how I identify on-the-job learning and measure outcomes that correlate with learning. To measure learning, I find the new external projects that an employee had interacted with throughout the time they are on GitHub. I define external projects as those not owned or controlled by the observed employee. I start from the first time I observe a user on GitHub and count the number of distinct project names that the user has interacted with by the end of each month. Self-owned projects are excluded because I assume that the knowledge created by any individual is infinitesimal. Re-visiting a project after the first

month does not count toward learning to exclude cases where learning turns into between-firm collaboration or hobby projects. I assume each project name captures a unique idea, and in particular, I disregard cases where two different projects belong to different owners but have the same name. Project name duplicates largely fall in two cases: the name is generic such as "test" or "blog", or the project is a copy of a collaborated project.[7] Finally, I excluded any project that the employee had worked on. Importantly, I take into account projects that belong to the firm, but the employee never accumulated more than 20 hours of work time. This ensures that I also capture "internal learning" that may play a more important role at large firms.

The measure is designed to capture several forms of learning, including self-driven, work-driven, and employer-sponsored. The employee may be learning a new programming language and bookmark a project that uses that language (self-driven). The employee may run into a bug in a third-party library and ask a question (work-driven)[8]. Finally, the employer may sponsor the employee to incorporate their product in an open-source framework (employer-sponsored).

3.1. Learning over the life-cycle

One central challenge with the our empirical method is that I observe little biographic information on GitHub. Although employees often report their names and occasionally their email and current location, they usually don't report their age or tenure at the firm. This poses a challenge because the standard theory predicts a decline in human capital investment over the life-cycle. I use a crude measure to distinguish experienced developers: whether they immediately start working as they join GitHub. I assume that young employees are more likely to create their accounts as a student and thus only start working until 18 to 24 months after account creation. However, I caution that the.

Figure 1 plots the number of new projects that employees interacted with every month over the time they are observed on GitHub. I separate the employees into two groups: those who started at least 18 months after account creation, whom I call "students," and those who start working within 3 months of account creation, whom I call "existing employees."

Two facts are consistent with the decreasing investment in learning over the life-cycle as proposed by, e.g., Ben-Porath (1967). Employees in both samples gradually spend less

[7] One of the most common forms of collaboration is the "fork & pull request workflow", where a developer creates a personal copy of an existing project (fork), edit and test the idea, request to update the code (pull request), and delete the personal copy. As a result, I often observe thousands of projects with the same name. In those cases, I only count once.

[8] Such user-driven product development is a common motivation that drives firms to make their software open-source.

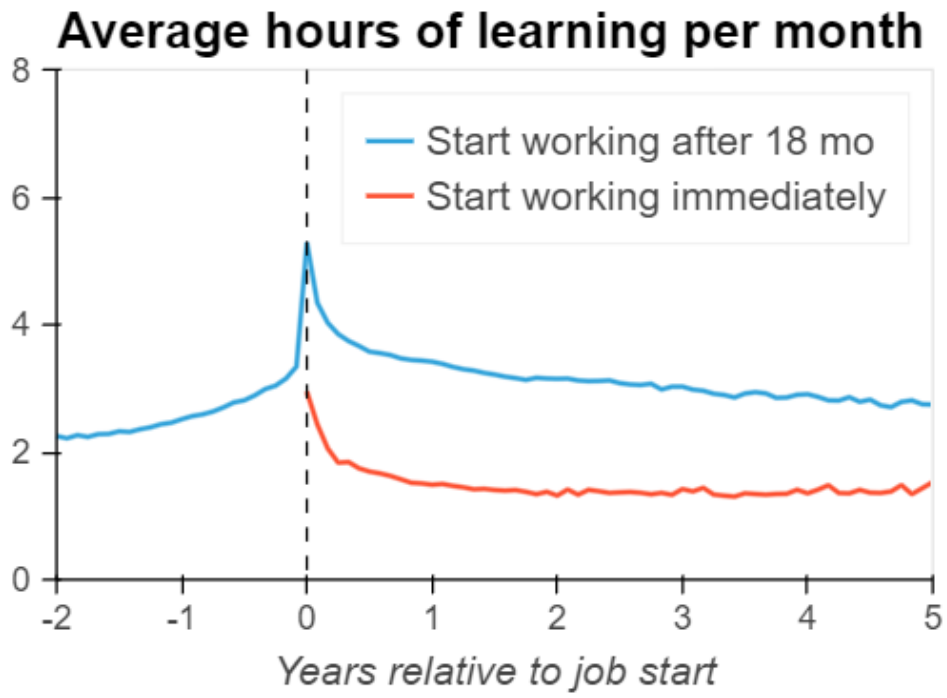

Figure 1: *Average hours spent on new projects by inferred status at account creation.*

time exploring new projects on the job[9], and existing employees interact significantly more projects throughout their work than students. I also see that employees interacted significantly more with external projects in the month when they started working, which confirms the validity of our identification of employment. The spike is not mechanical because I exclude projects that the employee spends on the working project.

In the rest of the paper, I will focus on the behavior of the subsample of employees who started at least 24 months after registering their GitHub account.

3.2. Learning by firm sizes and ages

Figure 2 splits the sample of students (those who started working at least 18 months after registration) by the quintile size of the firm. The size of the firm is defined as the number of employees who are also working for the firm upon joining. I split the sample into five quintiles, where each quintile roughly contains the same number of employees. In particular, Q1 contains 20,086 "entrepreneurs" who joined the firm when there is no one else. Q5 contains 29,106 employees who joined the firm when there are 36 (e.g. Uber) to 1952 (e.g. Microsoft) employees observed in the data. I see that the general trend established

[9]I drop employees from our data once they stop working so that the result is not simply driven by lower participation on GitHub.

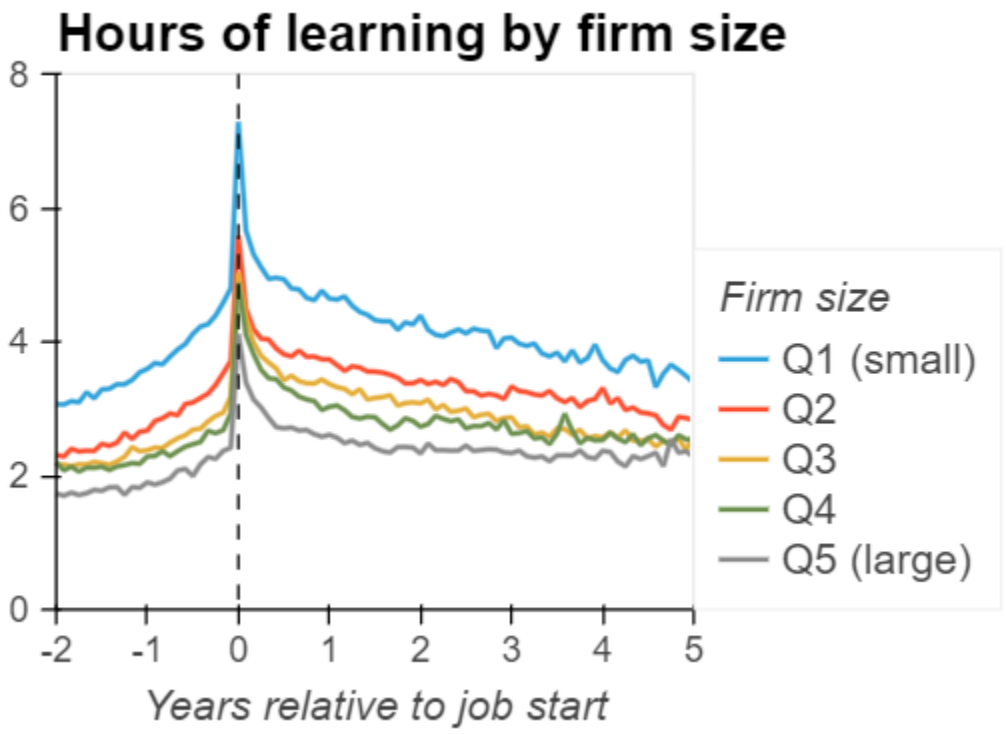

Figure 2: *Average hours spent on new projects by the firm's size when the employee joins the firm.*

above continues to hold across the sub-samples.

I also find that employees who join smaller firms are already significantly more active in exploring external projects before joining the firm. Two years before joining the firm, when the employee is unlikely to know about the future employer, those who would start a firm (or be the first one in the firm to work on GitHub) is already exploring twice as much as those who would join a large firm. I interpret this as evidence of a matching mechanism where more curious employees seek employment at small firms. I also see the gap *increases* until they started working. The more significant anticipation effect suggests that employees who join smaller firms are more likely to learn about outside technology that would contribute to the future employer. Large firms with more mature but proprietary infrastructure may not merit such exploration.

After joining the firm and conditional on continued employment, I see that those at small firms continue to explore more, but the gap shrinks. I interpret this both as evidence of better "job security" as the employee accumulates tenure and as the (small) firms continue to survive. Interestingly, I see that Q2-Q5 converges after around two to three years and Q5 employees overtakes Q2-Q4 in their interaction with new outside projects. One possible explanation is that Q5 employers, such as Uber and Microsoft, are more likely to be platform firms as discussed in Rochet and Tirole (2003), and require their employees to continue

Table 3: *On-the-job Learning Effort.*

	(1)	(2)	(3)	(4)	(5)
	Hours of learning on the job				
Log(joining firm size)	-0.067	-0.036	-0.025	-0.026	-0.058
	(0.002)	(0.002)	(0.002)	(0.002)	(0.006)
Log(joining firm age)		-0.054	-0.027	-0.016	-0.017
		(0.002)	(0.002)	(0.002)	(0.003)
Log(pre-job learning hours)			0.080	0.066	0.017
			(0.010)	(0.010)	(0.011)
Log(pre-job learning scope)			0.546	0.556	0.521
			(0.011)	(0.011)	(0.011)
Intercept	1.451	1.522	1.003	0.979	1.124
	(0.006)	(0.006)	(0.007)	(0.007)	(0.013)
Registration Cohort FE				Yes	Yes
Job Start Cohort FE				Yes	
Firm FE					Yes
N	55278	55278	55278	55278	55278
R^2	0.023	0.033	0.342	0.357	0.647

interacting with outside employees to on-board them to the platform. Detailed discussions are outside the scope of this paper.

I run regressions on the on-the-job exploration to describe the relationship more precisely. Table 3 reports the results. The dependent variable is the average number of hours employees spend on external projects in the first two years on the job. The firm size and firm age when the employee joins a firm predict lower on-the-job exploration throughout various specifications. In Column (3), I add the individual-level controls of the scope (number of new external projects) and effort (number of hours spent on new external projects) designed to capture individual effects. I add fixed effects in Columns (4) to (5). The cohort effect is the years between account registration and an employee joining the firm. The firm effect applies to every employee who has joined a firm. Column (5) reports the most complete results: within employees who started their account on GitHub in the same year and joined the same firm, and controlling for their pre-job behavior on GitHub, those who joined when the firm was young and small put significantly more effort in interacting with external projects while they are working.

Is the result driven by a heterogeneous supply of unobserved learning at small firms? Large firms supply more internal tools and knowledge that employees learn in a firm, even if their work activities are published on GitHub. While I do not observe internal activities, I can find the composition of the topics employees interact with.

3.2.1. Matching employees in small versus large firms

To disentangle the selection and treatment effect of joining a startup versus an incumbent firm, I perform a matching exercise between employees who join small firms and large firms. In the subsample of employees for whom I observe their experiences for at least two years before they join a firm, I match each employee who joined a large firm (such as Facebook or Google) with an employee who joined a small firm (such as OpenAI). The matching variables are the time of joining, the time of starting the job, and their behavior on GitHub in each of the quarters from the 8th to the 5th quarter before they start their job. For each quarter in year $T-2$, I measure the number of hours and topics that the employee interacted with on GitHub.

I run a DiD-like design on the matched sample, as the following

$$Y_{it} = \alpha + \beta \text{Small}_i + \sum_{q=-8}^{8} \lambda_q \mathbf{1}\{t=q\} + \sum_{q=-8}^{8} \mu_q \mathbf{1}\{t=q\} \times \text{Small}_i + \delta \text{Sector}_i, \tag{1}$$

where $\mathbf{1}\{t=q\}$ is the indicator for the q-th quarter relative to the start of the job (when $t=0$). Small_i is the indicator if the employee joins a small firm[10]. Sector_i is the indicator of the topic that employees at the firm worked on the most, such as cloud or front-end.

Figures 3 through 5 present the coefficient μ_q on the interaction term $\mathbf{1}\{t=q\} \times \text{Small}_i$, for each of the three outcome variables: number of hours spent on learning, number of topics interacted, and novelty of the topics interacted. The novelty is measured by the number of topics weighted by the inverse of the frequency of the topic or bi-gram topic combinations.

The figures reveal a set of interesting facts about the on-the-job learning of employees who join small firms. First, Figure 3 shows that employees who join or start small firms spend more time learning than similar employees who join incumbent firms. The effect begins to pick up right after the matching period and stays at a high level until the end of the sample, year $T+2$. To the extent that the need for innovation drives learning or that innate curiosity breeds innovation, those at small firms seem more innovative. The result is consistent with Akcigit and Goldschlag (2023b) who show that employees who join large firms seem less innovative but earn a higher income.

I inspect the learning scope and find a distinct result, shown in Figure 4. Small-firm employees interact with more topics before starting their job and the pace is similar to their time spent on exploration. However, once they start their career, the scope measure drops precipitously when the effort measure remains high. This suggests a "brainstorming" period before they initiate their venture. After they have started, the scope becomes narrow,

[10] In my sample, 92% of the observed employees joined one firm. If I observe multiple experiences, I use the one with the longest tenure and disregard the remaining experiences.

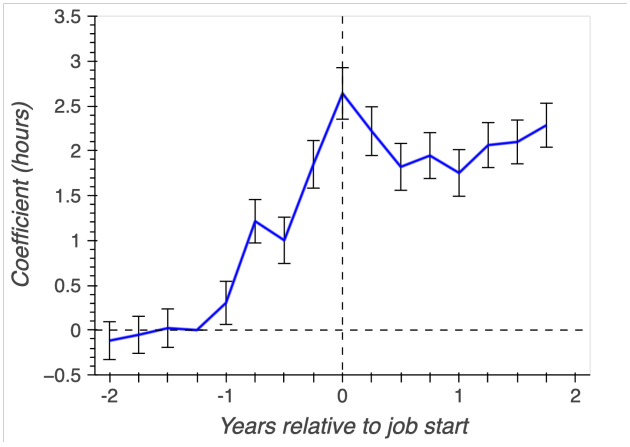

Figure 3: *Event study: number of hours spent on learning*

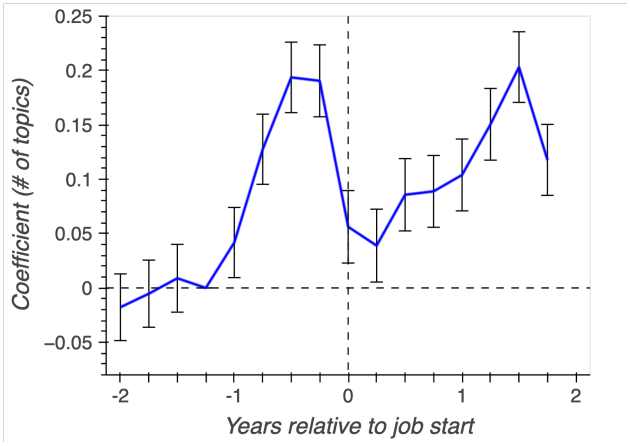

Figure 4: *Event study: number of topics interacted*

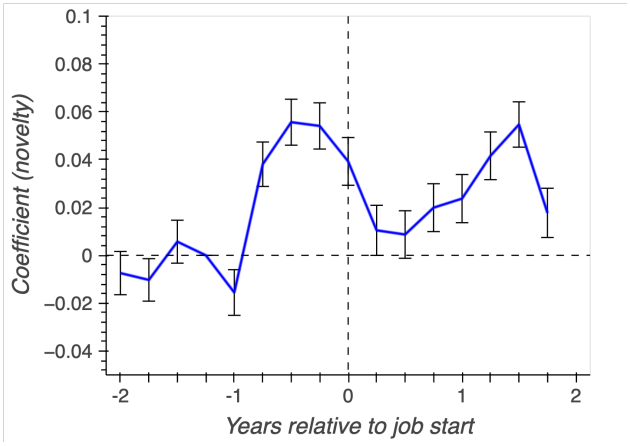

Figure 5: *Event study: novelty-adjusted number of topics interacted*

similar to a "market research" phase where the entrepreneur intensely studies competitors who are working on related margins, though their effort remains high relative to counterparts at incumbent firms. The scope then goes up again as the entrepreneur stays on the job, suggesting a gradual expansion of the research scope as the product matures and the tasks become more multi-faceted.

Figure 5 presents a similar set of results as the scope, but also considers if there is a supply of similar-minded engineers who also interacted with these topics. To that end, I calculate the cumulative frequency of people-project interactions until the quarter of observation and assign weights to a topic or a two-topic combination if fewer people have explored them. For example, the combination of cloud service and database operations has been common since GitHub was online, but the combination of artificial intelligence (AI) and physical sciences has been relatively rare until recently. The result shows similar dynamics as the scope measure, but there's a relatively smaller variance in the novelty than the scope, which is consistent with a model where the novelty of the idea is relatively fixed, conditional on the entrepreneur joining the firm.

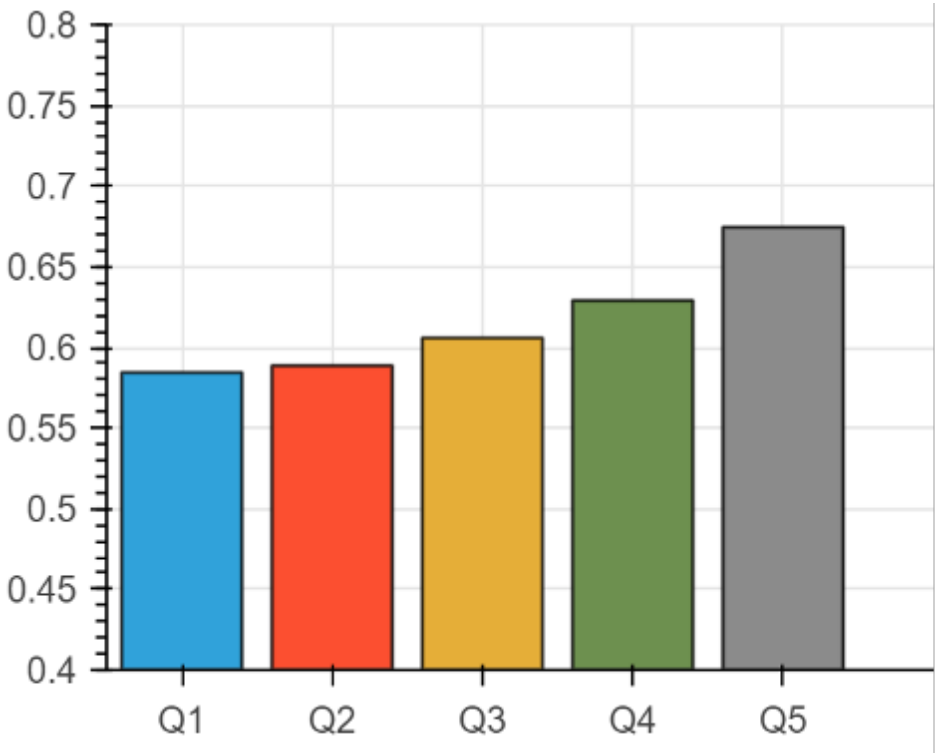

Figure 6: *Text-based similarity between the topics of learning on the job and the topics of the working projects, by firm sizes.*

Table 4: *Pre-job exploration predicts better on-the-job outcome*

	(1)	(2)	(3)	(4)	(5)	(6)
	# of projects		# of topics		Time to promotion (days)	
Pre-job exploration	0.054***	0.046***	0.0071***	0.0078***	−1.65***	−1.28***
	(0.0069)	(0.0067)	(0.0010)	(0.0012)	(0.3357)	(0.2504)
Year FE	✓	✓	✓	✓	✓	✓
Firm FE		✓		✓		✓
N	39118	39118	39118	39118	37166	37166
R^2	0.106	0.545	0.003	0.532	0.217	0.605

3.3. Learning and on-the-job outcome

The natural question is what learning has brought to employees and the firm. GitHub is well known among practitioners as a signaling device. Various indicators have been developed to evaluate employees' performance on the job[11]. I hypothesize that the market is best described by symmetric learning as described in Gibbons and Waldman (1999), where both the firm and employee know the human capital accumulation, including the acquisition of technology from outside firms.

I indeed find evidence consistent with that employers are aware of the technology adoption (learning) of their employees. First, I show that more exploratory employees are rewarded with a higher status in the firm, where I proxy the promotion with the first time the employee starts reviewing code. Indeed, I find that they begin to review code submissions to their working projects sooner, work with more projects, and work on more topics in the first two years, conditional on staying with the same employer. The regression coefficients are presented in Table 4.

In terms of learning topics, I show that on-the-job learning correlates with new working projects. Figure 7 plots the event study where an employee starts working on a new topic. The blue curve shows the time spent on external projects with the new topic, and the red curve shows that on the old topic. I normalized the time spent on either topic to zero three months before they started working on the new topic. I define the old topic as the topic of the employee's first working project and the new one that the employee starts working on at least six months after starting their job. The time gap ensures the result is not simply driven by "full-stack" employees. The result shows a clear jump in the interaction with the new topic around the time of the new project, which confirms that such learning indeed correlates with labor supply.

[11]For example, firms like LinearB and Swarmia provide performance analytics on Git-managed software projects.

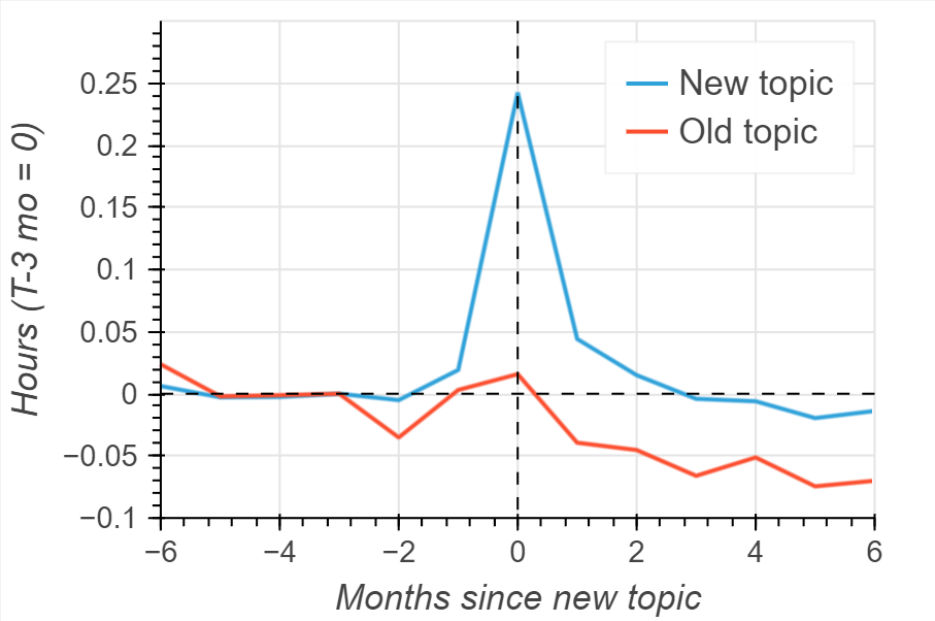

Figure 7: *Effort paid to learning new versus old topics when the employee picks up a a project with the new topic on the job.*

4. Conclusion

Leveraging detailed activity data collected from 144,000 employees across the globe, I document novel stylized facts about high-skilled labor's human capital investment behavior. Our measurement captures learning that are self-driven, work-driven, or employer-sponsored. Our findings are consistent with three channels: (1) more inquisitive employees match themselves with smaller, younger firms, where their skills play a more important role; (2) employees with less tenure and employees at younger, smaller firms have less job security and thus seek to diversify their human capital; (3) large firms offer more proprietary materials of learning that complement observed learning, but are not transferable to future employers. By directly measure the interaction between employees and knowledge, I shed new lights on the measurement problems that have long challenged empirical studies of on-the-job human investment.

Appendix

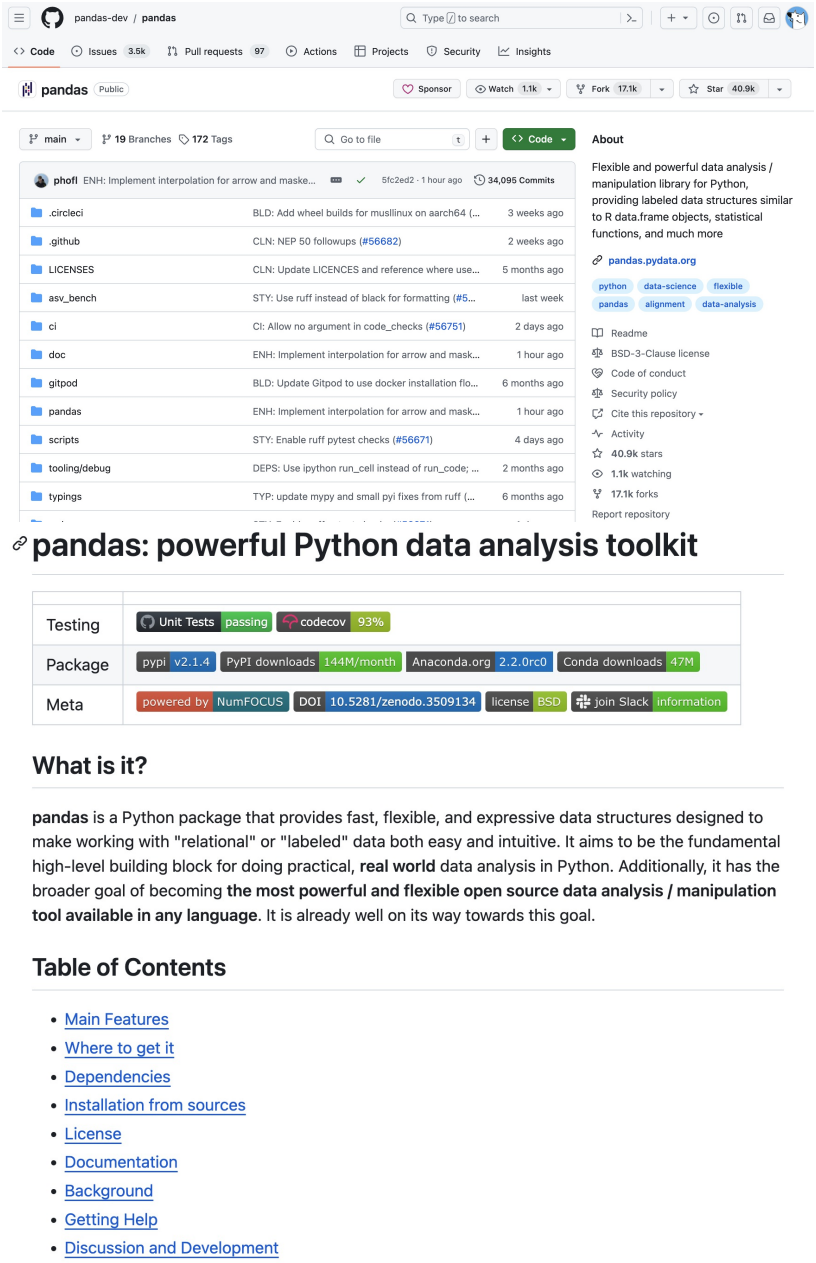

Figure 8: *A typical GitHub project, showing short description, topics, and Readme file. The short description is found in the upper right side under the "About" section. The topics are blue word bubbles under the short description. The lower half shows the Readme file.*

References

Acemoglu, D. and Pischke, J.-S. (1998). Why Do Firms Train? Theory and Evidence*. *The Quarterly Journal of Economics*, 113(1):79–119.

Acemoglu, D. and Pischke, J.-S. (1999). The Structure of Wages and Investment in General Training. *Journal of Political Economy*, 107(3):539–572.

Ahmadpoor, M. and Jones, B. F. (2017). The dual frontier: Patented inventions and prior scientific advance. *Science*, 357(6351):583–587.

Akcigit, U. and Goldschlag, N. (2023a). Measuring the characteristics and employment dynamics of US inventors. Technical report, National Bureau of Economic Research.

Akcigit, U. and Goldschlag, N. (2023b). Where have all the" creative talents" gone? Employment dynamics of us inventors. Technical report, National Bureau of Economic Research.

Autor, D. H. (2001). Why Do Temporary Help Firms Provide Free General Skills Training?*. *The Quarterly Journal of Economics*, 116(4):1409–1448.

Bell, A., Chetty, R., Jaravel, X., Petkova, N., and Van Reenen, J. (2019). Who Becomes an Inventor in America? The Importance of Exposure to Innovation*. *The Quarterly Journal of Economics*, 134(2):647–713.

Ben-Porath, Y. (1967). The Production of Human Capital and the Life Cycle of Earnings. *Journal of Political Economy*, 75(4, Part 1):352–365.

Biasi, B., Deming, D. J., and Moser, P. (2021). Education and Innovation. Working Paper 28544.

Bloesch, J. and Weber, J. (2023). Congestion in Onboarding Workers and Sticky R&D.

Deming, D. J. and Noray, K. (2020). Earnings Dynamics, Changing Job Skills, and STEM Careers*. *The Quarterly Journal of Economics*, 135(4):1965–2005.

Fleming, L. and Sorenson, O. (2004). Science as a map in technological search. *Strategic Management Journal*, 25(8-9):909–928.

Gibbons, R. and Waldman, M. (1999). A Theory of Wage and Promotion Dynamics inside Firms. *The Quarterly Journal of Economics*, 114(4):1321–1358.

Goldfarb, A. and Tucker, C. (2019). Digital Economics. *Journal of Economic Literature*, 57(1):3–43.

Guzman, J. and Stern, S. (2020). The State of American Entrepreneurship: New Estimates of the Quantity and Quality of Entrepreneurship for 32 US States, 1988–2014. *American Economic Journal: Economic Policy*, 12(4):212–243.

Hampole, M. V. (2022). Financial frictions and human capital investments. Technical report, Working Paper.

Hann, I.-H., Roberts, J., Slaughter, S., and Fielding, R. (2004). An empirical analysis of economic returns to open source participation.

Jarosch, G., Oberfield, E., and Rossi-Hansberg, E. (2021). Learning From Coworkers. *Econometrica*, 89(2):647–676.

Kelly, B., Papanikolaou, D., Seru, A., and Taddy, M. (2021). Measuring Technological Innovation over the Long Run. *American Economic Review: Insights*, 3(3):303–320.

Kerr, W. R., Nanda, R., and Rhodes-Kropf, M. (2014). Entrepreneurship as Experimentation. *Journal of Economic Perspectives*, 28(3):25–48.

Kogan, L., Papanikolaou, D., Schmidt, L. D. W., and Song, J. (2020). Technological Innovation and Labor Income Risk.

Lerner, J., Pathak, P. A., and Tirole, J. (2006). The Dynamics of Open-Source Contributors. *American Economic Review*, 96(2):114–118.

Lerner, J. and Tirole, J. (2002). Some Simple Economics of Open Source. *The Journal of Industrial Economics*, 50(2):197–234.

Loeliger, J. and McCullough, M. (2012). *Version Control with Git: Powerful Tools and Techniques for Collaborative Software Development*. "O'Reilly Media, Inc.".

Mincer, J. (1962). On-the-Job Training: Costs, Returns, and Some Implications. *Journal of Political Economy*, 70(5, Part 2):50–79.

Mincer, J. A. (1974). *Schooling, Experience, and Earnings*. NBER.

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

Nagle, F. (2019). Open Source Software and Firm Productivity. *Management Science*, 65(3):1191–1215.

Rochet, J.-C. and Tirole, J. (2003). Platform Competition in Two-Sided Markets. *Journal of the European Economic Association*, 1(4):990–1029.

Smith, M., Yagan, D., Zidar, O., and Zwick, E. (2022). The Rise of Pass-Throughs and the Decline of the Labor Share. *American Economic Review: Insights*, 4(3):323–340.

Tambe, P., Hitt, L., Rock, D., and Brynjolfsson, E. (2020). Digital Capital and Superstar Firms.