THE POWER OF PROXIMITY TO COWORKERS [1]

Natalia Emanuel[2] · Emma Harrington[3] · Amanda Pallais[4]

May 2, 2025

Abstract

How does proximity to coworkers affect training and productivity? We study software engineers at a Fortune 500 firm from 2019 to 2024. Our difference-in-differences designs leverage the fact that both the office closures and return-to-office mandates affected co-located teams more than distributed ones. We find that sitting near teammates increases coding feedback by 18.3% and improves code quality. Gains are concentrated among less-tenured and younger employees, who are building human capital. However, there is a tradeoff: experienced engineers write less code when sitting near colleagues. National trends are consistent with remote work scarring young workers.

Keywords: Remote work, on-the-job training, firm-specific human capital, general human capital, peer effects, return to office, telecommunication, gender
JEL: J24, M15, M53, M54, J16, O33, R23

[1]This project would not have been possible without the curiosity and commitment to research of our colleagues at the firm who shared data: Lauren, Susan, and Ebube. We thank Alex Albright, Jenna Anders, Eric Anderson, Garrett Anstreicher, Christopher Campos, Raji Chakrabarti, Kirsten Clinton, Ed Glaeser, Ben Hyman, Bill Johnson, Ezra Karger, Lawrence Katz, Gizem Kosar, Sitian Liu, Daniel Mangrum, Ishan Nath, Dev Patel, Soum Shukla, Jason Somerville, Mattie Toma, Tianyi Wang, and Melanie Wasserman for thoughtful comments. We also thank numerous seminar participants. We thank Jared Fang for research assistance. The findings and conclusions expressed are solely those of the authors and do not reflect the opinions or policy of our employers, the organizations that supported this work, the Federal Reserve Bank of New York or the Federal Reserve System.

[2]The Federal Reserve Bank of New York (natalia@nataliaemanuel.com)
[3]University of Virginia (emma.k.harrington4@gmail.com)
[4]Harvard University and NBER (apallais@fas.harvard.edu)

Coworkers are more distant than ever before. In 2024, workers lived an average of 27 miles from the office, nearly three times farther than in 2019 (Akan et al., 2024). They worked from home one to two days per week (Bureau of Labor Statistics, 2024) and while in the office, many still had colleagues at home or in distant offices (Goldberg, 2021). It is possible that this distance makes it harder to learn from colleagues, who traditionally have been the source of about a sixth of lifetime human capital (Herkenhoff et al., 2024). Yet workers are extremely digitally connected, spending 11−13 hours per day on screens (Belden, 2022). Perhaps, workers can learn just as much from colleagues even when physically distant. In an increasingly digital world, how does proximity to coworkers impact workers' productivity today and human-capital development for tomorrow?

We study these questions among software engineers at a Fortune 500 online retailer from 2019 to 2024. Before the pandemic, some teams of software engineers were all co-located in the same building; others were split between the two office buildings on the main campus (a ten-minute walk apart); and still others were fully distributed, with teammates who were fully remote or in satellite campuses. When the offices closed for the pandemic, these differences became immaterial. By comparing the differences between these teams before the office closures of 2020 — when they varied in proximity — to after the closures — when they did not — we can identify the causal effects of proximity on engineers' training and productivity. We can further investigate whether differentials re-materialize after the return-to-office (RTO) mandates in 2022 and 2023.

We find that sitting alongside teammates increases the digital feedback that engineers receive on their code. Before an engineer can finalize their code, it must be reviewed by other engineers.[1] Like someone commenting on a paper draft, the reviewer highlights problematic sections of the document and suggests improvements. When the offices were open, engineers on co-located teams received 23.9% (or 1.92) more comments on their code than did engineers on distributed teams. Once the offices closed, this advantage largely disappeared, narrowing by 18.3% (or 1.47 comments per program, p-value

[1]Code reviews are crucial for improving code and helping engineers learn. This importance is evident in their frequent emphasis in software-engineer job posts (e.g., these Microsoft or Google postings).

$= 0.0026$), with declines entirely concentrated in feedback from teammates.[2] To the extent that co-located teammates also provide verbal feedback, this is a lower bound on proximity's total effect on feedback.

Proximity increases digital feedback by making asking for it easier. Engineers sitting near their teammates ask more follow-up questions. They also tap a wider range of people for feedback, echoing studies showing that remote work silos communication networks (Yang et al., 2022; Gibbs et al., 2023). These results mirror classic studies showing that people converse more over the phone or email when they can also interact in-person (Gaspar and Glaeser, 1998; Agrawal and Goldfarb, 2008). Even with modern technology, face-to-face interaction is a complement rather than a substitute for digital communication.

We find that small frictions to face-to-face contact can have out-sized effects. Even a 10-minute walk between two buildings on the same campus has the same detrimental effect on feedback as being multiple states away. This finding accords with research on academics, which shows that being in a different building — or even on different floors — reduces coauthorship.[3] This paper contributes by showing that proximity affects less serendipitous collaborations: for coworkers on the same team who already work on common projects, small physical distances still hamper the exchange of feedback.

Having even one distant colleague creates negative externalities for the rest of the team. We see this effect through new hires. When a new hire changes a team from being all co-located to being distributed, there is a dramatic drop off in feedback exchanged between pre-existing teammates who remain together. By contrast, we see no such effect of a new teammate who sits with the rest of the team. Efforts to incorporate the distant colleague — by, for example, moving short meetings online — appear to degrade the connections

[2]There were no differential declines in feedback from non-teammates (about half of all feedback), suggesting there were no differential trends in engineers' need for feedback around the closures.

[3]In Bell Labs, scientists are more likely to collaborate if they work on the same floors (Kraut et al., 1988), and at MIT, professors are more likely to coauthor together if they are in the same building (Catalini, 2018), even when only reassigned to the same building during renovations (Salazar Miranda and Claudel, 2021). Relatedly, in Silicon Valley, software engineers at different firms are more likely to cite one another's patents if they come into physical contact (Atkin et al., 2022). However, others have found that physical distance is less important than intellectual distance in determining spillovers in academia (Azoulay et al., 2010; Waldinger, 2012), especially as technologies improve (Chen et al., 2022).

between co-located colleagues.

The additional feedback co-located engineers receive is substantive. We had external engineers rate a sample of comments, then used a supervised machine-learning algorithm to characterize all comments. We find that proximity to teammates especially increases helpful, actionable, impactful, and well-reasoned comments, which could help engineers learn how to write better code. Indeed, engineers trained on co-located teams write higher-quality code in the long run, introducing fewer bugs and writing fewer programs that get deleted because they are poorly-written or misdirected.[4]

Engineers who have the most to learn tend to receive more feedback when in-person and lose more mentorship when remote. The effects are most potent among engineers who are new to the firm — and need help building firm-specific human capital — and engineers who are young — and need help building general human capital. Female engineers are also more affected, partly because women are more hesitant to ask for feedback when they cannot do so in-person.

Engineers who have the most to learn also see the largest improvement in their code quality when coming back together with coworkers. Around the return-to-office calls, engineers on co-located teams see larger increases in face-to-face time with teammates. These engineers also see greater improvements in the quality of their code. Once engineers were required to back in the office three days per week, those on co-located teams were 2.2 pp less likely to add files that got deleted (p-value $= 0.041$) and 1.4 pp less likely to introduce bugs (p-value $= 0.022$) than engineers on distributed teams. These differences are concentrated among less-tenured and younger engineers, who see twice as large an improvement as the aggregate.

However, providing feedback and improving others' code requires effort. Experienced engineers, who provide the most feedback, write less code when sitting near their colleagues. We see this both around the office closures and the office re-openings. When the

[4]Even comparing engineers on the same team, an engineer previously trained on a co-located team writes less disposable and buggy code than one trained on a distributed team.

offices close, senior engineers who lose proximity to their colleagues see relative increases in the code they write. When the offices re-open, they write relatively less code. This creates a tradeoff for the firm between boosting junior engineers' skills and allowing senior engineers more time to program.[5]

Our results indicate that it is harder to build engineers' talent when teams are distributed. An alternative to building talent is buying it, by hiring more experienced workers. Consistent with this, the firm shifts toward hiring older, more experienced engineers when the offices are closed — and shifts back to hiring younger workers when the offices re-open.

During the office closures, we also see the flipside of this behavior: other firms poach engineers whose skills were built at this firm. Engineers trained on co-located teams — whose skills have been built up more — are more likely to leave for better jobs. Thus, the firm does not fully capture the returns to investing in these engineers' human capital. The firm might under-incentivize coworkers' investments in each other's human capital, even if it could perfectly observe them. This classic problem in general human capital formation (Becker, 1964) makes social connections between coworkers — forged by face-to-face contact — critical in facilitating investment.

We also observe attendance patterns consistent with the returns to proximity being higher for young people. Using badge data on who comes into the office in the RTO period, we find that younger engineers are more likely to come into the office than older engineers.[6] Young people are particularly likely to show up when their teammates are based in the same office: this is driven by both the presence of the boss and the presence of other teammates.[7] This suggests that, for young people, a substantial draw to the office is the possibility of being near colleagues. Nationally, we see that younger people are more likely to be back in the office, both within tech and among all college-educated workers.

[5]Other researchers found tradeoffs from communication across workers in a very different setting: in the Manchester police department, when workers handling 911 calls are together, the police arrive to the scene sooner but the call handler is slower to answer the next call. This improves the measured productivity of one worker at the expense of the other (Battiston et al., 2021).

[6]Young people go into the office more, even conditional on the commute time and parental status.

[7]This finding accords with attendance patterns at Microsoft, where workers were more likely to go into the office when their managers and teammates are present (Charpignon et al., 2023).

We see suggestive evidence that the rise of remote work has had scarring effects on young people's employment prospects. Young engineers have seen relative increases in unemployment between 2017−2019 and 2022−2024, amidst a seismic increase in remote work (Barrero et al., 2021). In fact, all young college-educated people have seen relative increases in unemployment, with more persistent increases in more remotable jobs.

We contribute by providing evidence that proximity to coworkers increases on-the-job training, which is critical given how much workers learn from coworkers.[8] With less co-location — and thus less training — productivity tomorrow may suffer even if productivity today does not. This finding helps resolve the puzzle of remote work's rarity before the pandemic (Mas and Pallais, 2020), which seemed at odds with workers' strong preference for remote work (Mas and Pallais, 2017; Maestas et al., 2023; He et al., 2021; Lewandowski et al., 2023; Cullen et al., 2025) and its often positive productivity effects today.[9] The draw of the office may have been in training for tomorrow, and this pull may have been stronger given the externalities of distant colleagues on the rest of the team.

Our paper also contributes to the debate over how the rise of remote work will impact inequality. Many have highlighted remote work's potential to perpetuate inequality by education (e.g., Adams-Prassl et al., 2020; Bonacini et al., 2021; Maestas et al., 2023), while curbing other forms of inequality like disability status (Bloom et al., 2024). We highlight how remote work could increase intergenerational inequality, as young people struggle to build their human capital away from their coworkers, while older workers can focus more on their own output.

The rest of the paper is organized as follows. Section I provides context on software engineering. Section II introduces the data. Section III describes the empirical design.

[8] Working with higher-paid or more educated coworkers is strongly associated with higher subsequent wage growth (Herkenhoff et al., 2024; Jarosch et al., 2021; Nix, 2020). Working with more prolific patenters (Akcigit et al., 2018) and better teachers (Jackson and Bruegmann, 2009) improves subsequent outcomes, as does seeking advice from better sales workers (Sandvik et al., 2020). Better peers may not always have immediate positive effects if competing with them reduces current wages (Johnsen et al., 2024).

[9] Bloom et al. (2015); Choudhury et al. (2021); Fenizia and Kirchmaier (2024) find positive impacts of remote work on productivity, Dutcher (2012); Emanuel and Harrington (2024) find modestly negative impacts, and Atkin et al. (2022); Jalota and Ho (2024) find large negative impacts in developing countries.

Section IV discusses how proximity impacts feedback around the office closures, while Section V turns to coding output. Section VI investigates changes in code quantity and quality around the return to office. Sections VII−VIII consider implications for hiring and office attendance in the firm and nationally. Section IX concludes.

I Background on Software Development

Our data comes from a Fortune 500 online retailer, whose software engineers handle typical tasks for the industry. Some engineers build the front-end interface of the retailer's website, others maintain the back-end of the digital catalog, still others create internal tools for other people at the firm (e.g., for the finance team), and some engineers work on AI features (e.g., product recommendation services).

The engineers follow a standard process for developing business-critical code (using the Github system). When making changes, engineers create a "branch" from the primary code-base, which separates their rough drafts from live code. Before merging their changes back into live code, their code must be reviewed by at least one other engineer.

The code review process is a key step in software development, allowing other engineers to ensure the new code is well-tested, error-free, and understandable to future engineers (including the author's future self). Typically, two reviewers weigh in on the code, one from the engineer's immediate team and one who brings relevant external experience, such as having worked on that part of the code in the past. While there are strong norms (and managerial expectations) to give at least some feedback when asked, there are no explicit incentives to give extensive feedback since such a system would be easy to game with short or pointless comments.

In addition to vetting the current code, reviewers can build the author's skills. One manager explained: "We ask senior, technical folks...to make their code reviews a learning opportunity by, for example, including the reasoning behind suggested changes." In line with this goal, reviewers are typically more experienced than the code's author. In 70% of reviews, the reviewer has been at the company longer, suggesting deeper firm-specific

knowledge. In 58%, the reviewer is older, indicative of greater general coding experience.

Teams follow an Agile management system that includes daily "stand-up" meetings where everyone literally stands to keep the meeting brief. In these ten-to-fifteen-minute meetings, engineers share what they have been working on and discuss anything blocking their progress. This is also a chance for engineers to ask others to review their code, which many prefer doing face-to-face rather than digitally (e.g., on Slack or Github).

The format of these daily meetings depends on teammates' physical proximity. As one engineer noted, "[my team] would almost never book a room and held all of our meetings [online] since we had a remote team member." Even within the firm's main campus, a short walk was sometimes enough to move the meeting online: it was hard to justify a ten-minute walk between buildings for an equally short meeting. Beyond these daily meetings, co-located engineers also had more chances for informal face-to-face interactions, aided by the firm's open office plan. Our paper investigates the ramifications of this close contact on engineers' training and productivity.

II Data

We collate data to (i) characterize workers' backgrounds, (ii) identify workers' physical proximity to their teammates, and (iii) measure different facets of workers' productivity and on-the-job learning. Together, our different data sources span from mid-2019 to mid-2024, covering the office closures of 2020 and two return-to-office waves in 2022 and 2023.

II.A Workers' Backgrounds and Demographics

Many of the firm's engineers are young and new to the company. Before the closures, engineers were 29 years old on average, suggesting many were still building general human capital (row 1 of Table 1). With an average firm tenure of only 1.4 years (16 months in row 2), many were still building firm-specific human capital. As is common in software engineering, the workforce is male-dominated: men compose 81% of our sample (and 75%

of programmers nationally).[10] Given the youth of the engineers, only 22% had children before the closures, with even lower rates of 17% among female engineers.[11]

II.B Identifying Teammates' Physical Proximity

We use personnel data to distinguish between co-located and distributed teams. We identify each employee's teammates as all the people who report to the same manager.[12] Using each teammate's assigned office, we then classify teams as (i) co-located in one building, (ii) split across multiple buildings on the firm's main campus (a roughly ten-minute walk apart), or (iii) fully distributed, with some members working remotely or in satellite offices. Before the COVID-19 office closures, 49% of engineers on the main campus were on teams co-located in a single building, 34% were on teams split across buildings, and 17% were on fully distributed teams.

During the office closures, the firm consolidated to one engineering building on its main campus while opening new satellite offices and expanding its fully remote workforce. As a result, when the offices re-opened, 25% of engineers on the main campus were on fully co-located teams, while the remaining 75% were on fully-distributed teams.

Badge Data around RTOs: We use badge data from 2022 onward to observe how often teammates are in the office together after the offices re-opened. Employees use their company badge to swipe into the office buildings, yielding a record of daily office attendance.[13] During the first RTO call — which encouraged two in-office days per week — engineers came back on average of 0.6 days per week. During the second RTO — which called for three in-office days per week — office attendance rose to 1.9 days per week. Throughout the RTOs, the firm did not mandate certain in-office days of the week, but

[10]The national statistic uses US Census data (U.S. Census Bureau, 2019), where we define software engineers as (1) Computer Scientists and Systems Analysts, Network systems Analysts, and Web Developers (occupation 2010 code = 1000), (2) Computer Programmers (1010), or (3) Software Developers, Applications and Systems Software (1020).

[11]Starting in June of 2020, the firm began collecting caregiving information, including childcare responsibilities. These data cover 70% of software engineers in our sample.

[12]This strategy identifies the teammates for the plurality of workers since all but the highest-level managers oversee a single team. We exclude the minority of people under high-level managers from our analysis (1.2% of engineers). The results are similar without this restriction.

[13]One of us was embedded in the firm for a summer: Figure A.1 shows Emma's badges.

most engineers went into the office on Tuesday through Thursday (Figure A.2).

II.C Measuring Productivity & Learning

We have multiple extracts of data from the firm's code development system. Around the closures, we observe feedback that engineers received on their code and the quantity of code they wrote in the firm's main code-base. Around the RTOs, we observe the quantity of code engineers wrote in all code-bases and proxies for the quality of this code.

Coding Data around the Office Closures: We observe activity in the firm's main code-base from August 2019 until December 2020. The dataset tracks all the changes to this code-base and, crucially, the feedback that engineers receive on their code. In our analysis sample, the 1,055 engineers made 29,809 distinct changes to the code-base and received 174,014 comments on this code from their coworkers.

For every change in the main code-base, we see who made the change, when it occurred, how many lines of code were added, and how many files were affected. On average, each modification — or "program" — adds 346 lines and affects 7 files. Engineers typically write 2 programs in the main code-base each month. This data on code quantity gives us measures of engineers' immediate productivity.

The data also detail every comment exchanged in the code review process, including its text, timestamp, and identity of the commenter. These comments are a bit like feedback on a paper draft: the commenter often highlights portions of the document and comments about its structure, functionality, or readability. On average, engineers receive 6.5 comments on each program. Each comment averages 16 words (100 characters). As detailed in Section IV.B, the feedback is usually substantive, not only telling the engineer how to change the code but also explaining the underlying reasoning. Thus, this dataset lets us measure investments in engineers' human capital.

Coding Data Around the RTOs: We have data on code quantity and quality for all the firm's code-bases from December 2020 to mid-2024. As above, these data track the frequency with which engineers make changes to the code and the extent of these changes,

but now for all code-bases rather than only the main one.[14]

This dataset also includes code-quality metrics. Engineers often gauge code quality by measuring churn — i.e., are engineers ineffectively spinning their wheels? — and bugs — i.e., are engineers introducing vulnerabilities or errors? We measure churn by tracking the frequency with which engineers introduce files that later get deleted (which occurs in 15% of programs).[15] We measure bugs by looking for instances where the engineer made a set of changes that get fully reverted, which usually indicates that the engineer's changes precipitated a minor emergency that caused their team to revert back to an earlier version of the code. These more serious problems occur 3.5% of the time.

III Empirical Design

Our goal is to identify how proximity to teammates affects engineers' investments in each other's human capital, as well as the quantity and quality of their code. To do this, we exploit two key shocks to co-location: (i) the office closures of COVID-19 and (ii) the subsequent return-to-office periods. In both cases, co-located teams saw bigger changes in proximity than did distributed ones, facilitating difference-in-differences designs.

Office Closure Design: When the offices closed, engineers on co-located teams saw bigger losses in proximity to their teammates than did engineers on distributed ones. We leverage this in the following design:

$$Y_{it} = \beta \text{Post Closure}_t \cdot \text{One-Building Team}_i + \mu_i + \mu_t + X_{it}'\Gamma + \epsilon_{it}, \quad (1)$$

where Y_{it} represents the feedback that engineer i received on her programs in month t (or, in Section V, the number of programs she wrote). We include fixed effects for engineer (μ_i) and month (μ_t). We cluster standard errors by team, the unit of treatment assignment.[16]

We define $\text{One-Building Team}_i = 1$ if engineers were consistently on a one-building team

[14] Our results around the RTOs are directionally similar but noisier when focusing on the main code-base.

[15] We focus on deletions within six months, but results are similar with no restrictions or shorter windows.

[16] This design considers a single focal event — the pandemic-related office closures in March 2020 — so does not run into the problems that can arise with staggered treatment timing (e.g., Goodman-Bacon, 2021).

throughout the pre-period. This gives us a consistent definition to analyze both proximity's immediate and long-run consequences (e.g., for code quality in Section IV.B.1). Our difference-in-differences results are similar if we define team-type contemporaneously because switching is rare: before the closures, only 3.7% of engineers switch across one-and multi-building teams, and another 6.0% see changes in teammate proximity due to others' switching, hiring, or departing. In supplementary analyses, we use these pre-closure changes as alternative identifying variation.

Under a parallel-trends assumption, β reflects the causal effect of the greater loss in proximity for previously co-located teams in the closures. This assumption requires that, without the closures, feedback to engineers on co-located and distributed teams would have evolved similarly. Two pieces of evidence give us confidence in this assumption. Visually, trends in feedback are similar before the office closures (Figure 1). And crucially, around the closures, we see parallel trends in feedback from non-teammates, whose proximity did not change differentially for engineers on co-located versus distributed teams.

Engineers on co-located and distributed teams are broadly similar in their observables (Column 4 of Table 1). Our primary design focuses on engineers whose teammates were all on the main campus, and thus whose team was either all in one building or split across two buildings. For these engineers, co-location sometimes came down to chance: when a new hire came on board, was there an open seat near the rest of the team? These seating logistics were more challenging for internal-tool engineers, who aimed to sit near both end-users and fellow engineers (so only 35% were co-located). Accounting for engineering groups, engineers on one- and multi-building teams look observably similar on all dimensions except tenure (Column 5).[17]

Our preferred controls, X_{it}, include month-specific controls for the engineer's group, tenure, and age. These controls absorb residual variation and address imbalances across one- and multi-building team engineers. By interacting these controls with the month, we capture how they may interact with the pandemic. For instance, if the pandemic affected

[17] In supplementary designs, we compare one-building and fully-distributed teams, but these team structures are less random: fully-distributed engineers are older, more tenured, and higher-level (Table A.1).

website engineers differently from those working on internal tools, that difference would be captured by month-by-group fixed effects. Likewise, if the pandemic posed particular challenges for engineers who were new to the firm, that difference would be absorbed by month-by-tenure fixed effects (where tenure is measured in months). When analyzing feedback, we also control for program scope (quartics in the number of lines added, lines deleted, and files changed), which may mediate the extent of feedback engineers receive. Our full set of controls also includes month-specific controls for team size, engineer gender, race, home zipcode, job level, and initial building.

Office Re-opening Design: When the offices re-opened, differences in proximity between co-located and distributed teams re-emerge. We estimate dynamic differences between co-located and distributed teams:

$$Y_{it} = \sum_{\tau \in \{\text{Closed,1st RTO,2nd RTO}\}} \gamma_\tau \, \text{Co-Located Team}_{it} \, \mathbb{1}[t \in \tau] + \mu_i + \mu_t + X'_{it}\Theta + u_{it}, \quad (2)$$

where Y_{it} is either the number of programs or proxies for their quality. In this analysis, we allow Co-Located Team_{it} to change over time because, over the multiple years of the RTO analysis, nearly half of engineers (48.6%) spent time on both co-located and distributed teams. In each period, hundreds of engineers switched team types, allowing us to identify differences across team types in every period even with individual effects (μ_i).

We focus on three distinct periods: the office closures, the first RTO which required two days per week, and the second RTO which required three days per week. We predict that $\gamma_{closed} \approx 0$ since being assigned to the same office as one's teammates should not matter if that office is closed. We further predict that $\gamma_{2nd\ RTO}$ will be stronger than $\gamma_{1st\ RTO}$, since the second RTO entailed more days in the office (three versus two) and more of those days overlapped (since Tuesday−Thursday are all heavily trafficked days as shown in Figure A.2). We test whether $\gamma_{2nd\ RTO}$ is different from both $\gamma_{1st\ RTO}$ and γ_{closed}.

With only one main-campus building in the RTOs, this design compares one-building to fully-distributed teams. Because these team structures are less random than one- vs.

multi-building setups (Table A.2 shows observable differences), we treat this design as supplementary, though it does have the strength that there were fewer simultaneous disruptions to engineers' work and personal lives around the RTOs than the closures.

IV Proximity & Feedback: Evidence from Office Closures

If physical proximity causally boosts online feedback, we would expect three patterns to emerge around the office closures: (i) co-located engineers should receive more feedback when the offices are open, (ii) feedback should decline for everyone once the offices close, and (iii) the gap in feedback between previously co-located teams and already distributed teams should narrow. All three predictions borne out in Figure 1a. While the offices were open (to the left of the vertical line), engineers who sat in one building with their entire team received 23.9% (or 1.92) more comments on their programs than did engineers on distributed teams, even after controlling for program length, engineer tenure, age, and engineering group (p-value $= 0.0005$). Once all engineers were remote (to the right of the vertical line), this difference shrank to 7.1% (or 0.57 comments per program).

Our difference-in-differences (DiD) design compares the large pre-closure gap to the small post-closure gap. This approach indicates that the greater loss of proximity among one-building-team engineers reduced the feedback they received by 16.8%. When we include individual engineer fixed effects, the DiD coefficient is 18.3% (or 1.47 comments per program, p-value $= 0.0026$, Column 6 of Table 2). This estimate remains similar across alternative specifications, ranging from using no controls ($R^2 = 1.5\%$) to using the full set of controls ($R^2 = 61\%$, Table 2a).[18] The result is similar when measuring feedback as total characters or words (Table 3a, Columns $1-2$), and as we revisit in Section IV.B, the effects are even larger when focusing on substantive, technical feedback. Furthermore, when teammates are proximate, they give feedback more quickly (Table 3a, Column 3).

Notably, the differential decline in feedback for one-building-team engineers is driven solely by reduced feedback from teammates, with no detectable effect on feedback from other engineers (Table 2b). This null contradicts some alternative explanations: if, for

[18] Our full set of controls also include team size, demographics, zipcode, job level, and initial building.

example, the projects pursued by one-building team engineers simply necessitated more feedback before the offices closed but not afterwards, the resulting decline in feedback would have occurred across all sources, not just within their own small teams.[19]

Using engineers who switch team-types, we find that, when the offices were open, the same engineer received more feedback on one-building teams than she did on a multi-building one (Column 1 of Table A.3). Reassuringly, there is no such relationship once the offices are closed and formal assignments to offices are not meaningful. Suggestively, these patterns persist even when we include worker-by-manager fixed effects: when the same worker is working under the same manager, they receive more feedback when their team is co-located than when it becomes distributed (e.g., because a new hire cannot find a seat with the rest of the team, Columns $3-4$ of Table A.3).[20]

Interpretation: We find evidence that there is more digital communication when there is also face-to-face communication. One could have imagined the opposite pattern: when engineers lost in-person modes of communication, they could have compensated by exchanging more feedback online. Instead, we see that when engineers are no longer physically together, they also exchange less online feedback. This suggests that being face-to-face is a complement rather than a substitute for online interaction. We see the same pattern for mentions of other modes of online communication (e.g., Slack) in the code reviews (Table 3a, Column 4). To the extent that proximate teammates also talk more in-person, our estimates of proximity's impact on online mentorship are lower bounds of proximity's total effect on mentorship.

Drivers of the Complementarity: One explanation for the complementarity between physical proximity and online feedback is that engineers feel more comfortable asking for on-

[19]Proximity to non-teammates matters for their feedback. When the offices closed, feedback from non-teammates declines, particularly for engineers who had been in the firm's main building (Figure A.4).

[20]We consider a number of additional robustness checks. First, while our main analysis focuses on main-campus engineers (to ensure more apples-to-apples comparisons), our findings are similar if we instead analyze all engineers regardless of their location (Figure A.3b). Second, our results are also similar if we limit to internal-tool engineers, who are more likely to be distributed so as to work near the users of their tools (Figure A.3c). Finally, our results are comparable if we consider teams that were more or less co-located than average rather than whether all teammates were co-located (Figure A.3d).

line feedback when they are face-to-face with teammates. Engineers may ask more people to comment on their work, and they may also ask more follow-up questions to their commenters. We see evidence that proximity matters for both of these dimensions.

On the extensive margin, we do not see much movement on the number of commenters per program (Table 3b, Column 2), but proximity affects the size of engineers' networks (Column 3). Engineers on co-located teams are less likely to return to the same commenter repeatedly and instead receive feedback from more distinct commenters when the offices were open but not once they close. This result echoes existing research that digital networks shrink when people are no longer physically proximate to their collaborators (DeFilippis et al., 2020; Gibbs et al., 2023).

On the intensive margin, losing proximity to teammates reduces the depth of the back-and-forth conversation about the code (Table 3c, Column 1). This finding resonates with existing experimental findings from Brucks and Levav (2022): when pairs perform a collaborative task on Zoom rather than in-person they have many fewer back-and-forths.

These richer back-and-forths are driven, at least in part, by programmers' follow-up questions. Upon receiving feedback, one-building team engineers ask fully 48.4% (0.12) more follow-up questions when the offices are open, a differential that vanishes when the offices close (Table 3c, Column 4, p-value $= 0.0083$). As a result, about half of proximity's effect on feedback comes from commenters' follow-up comments and the other half comes from their initial feedback about the code (Column 2).

IV.A What Features of Proximity Matter?

Small Frictions vs. Far Distances: We find that small barriers to face-to-face contact can undermine feedback just as much as greater distances can. Figure 1b shows that engineers on teams that are distributed across small distances — just a ten-minute walk on the same campus — have the same level of feedback as those who are distributed across large distances — spanning the firm's campuses that are spread across the nation. Both receive less feedback than those on one-building teams. Even for coworkers working

on common project and meeting regularly, small frictions to face-to-face contact reduce digital exchange.

Externalities from Distant Teammates: Having a distant teammate reduces the feedback exchanged by teammates who themselves sit together in the same building. While the offices are open, co-located engineers exchange 14.5% less feedback when they have even a single teammate in another building rather than being fully co-located (p-value $= 0.037$, Figure A.5). One explanation for this pattern is that distant teammates impose externalities on the rest of their team: the team's efforts to incorporate them into meetings and discussions undermine the collaboration between co-located teammates.

We can also see evidence of these externalities using new hires before the pandemic. When there is not an available desk near the rest of the team, a new hire can transform a team from a one- to a multi-building team. To hold all else constant, we focus on the weeks surrounding the new hire and the feedback exchanged between pre-existing teammates who continue to sit together. When the team transitions from a one- to multi-building team, co-located teammates start to exchange less feedback (the blue line in Figure 1c). By contrast, there is no systematic change when the team gains a new hire, but this hire does not change the co-location status of the team (the orange line in Figure 1c). Taking the difference in these differences indicates that switching from a one- to multi-building team reduces feedback exchanged by 1.71 comments per program (p-value $= 0.095$).[21] In short, when one teammate is added in another building, co-located teammates start to exchange less feedback on their work.

[21] 16 teams (with 46 engineers) switch from one- to multi-building teams around a new hire, and 117 teams (with 369 engineers) hire someone new but do not change their co-location. Our DiD estimates:

$$\left(\frac{\text{Comments}}{\text{Review}}\right)_{ijt} = \alpha \text{Post Hire}_t \times \text{From One to Multi-Building Team}_{ij} + \psi \text{Post Hire}_t + \mu_{ij} + \epsilon_{ijt} \qquad (3)$$

where μ_{ij} represents programmer (i) by commenter (j) pair fixed effects, ψ captures the impact of any new hire, and α captures the additional effect of a new hire who makes the team distributed.

IV.B Substantive Feedback

During code reviews, engineers exchange substantive feedback about their code, and, when engineers are no longer co-located, this substantive feedback diminishes.

Figure 2a shows how losing proximity to teammates affects the frequency of each of the hundred most common words in the comments, excluding stop words like pronouns and prepositions (Bird et al., 2009). The x-axis shows the frequency of the word in comments written while offices were open, while the y-axis shows the DiD coefficient, reflecting how the word's frequency differentially changes around the office closures for engineers on one- versus multi-building teams (Equation 1). The figure shows that when engineers lose proximity to their teammates, the frequency of almost all of these words declines. Notably, many of these words are directly tied to how the code works — e.g., about its **functions**, **methods**, **models**, and what these objects **return**. Additionally, words that explicitly refer to **checking** programs — ensuring they do not **throw exceptions** or create other bugs — also decline in usage. These results suggest that losing proximity to teammates does not merely trim the fat of nitpicky comments, but instead reduces substantive feedback that may be critical for improving code quality.

We see similar results for other measures of substantive feedback, based on a supervised machine learning approach. We employed a set of software engineers outside the firm to label a random sample of 5,377 comments as (i) helpful, (ii) explaining the underlying reasoning, (iii) actionable, and (iv) likely to change the code (see Appendix I.C.1 for details). These evaluators tended to view the comments positively, judging 76% to be helpful, 51% as explaining the reasoning, 60% to be actionable, and 52% as likely to change the code.[22] We used a supervised machine-learning algorithm to scale up from this sample of labeled comments to predict the likely ratings for the full 174,014 comments. Specifically, we use a gradient-boosted decision tree (Chen and Guestrin, 2016) to predict the ratings based

[22]Evaluators sometimes said they had too little context to rate a comment. We conservatively view unrated comments as *not* helpful, well-reasoned, actionable, or impactful. Of the rated comments, 87% were viewed as helpful, 58% as well-reasoned, 68% as actionable, and 70% as impactful for the code.

While the ratings are positively correlated with one another, they capture distinct facets of comments, with correlations of at most 0.56 (for being actionable and likely to change the code).

on the words in the comments (see Appendix I.C.2 for details). In our holdout sample, these predictions are accurate 64−78% of the time.

Figure 2b shows that losing proximity to teammates reduces comments that would likely be rated positively. Prior to the office closures, engineers on one-building teams received more comments that would likely be rated as helpful, well-reasoned, actionable, and likely to change the code. Once the offices close, these gaps all disappear — and high-quality comments decline across the board.

In percentage terms, losing proximity to teammates has bigger effects on the frequency of high-quality comments — which decline by 21−23% (Figure 2b) — than the frequency of all comments — which decline by 18.3% (Figure 1a). Thus, the comments that remain are of lower predicted quality (Table A.4b): 2.9 pp fewer comments are helpful (p-value $=$ 0.039); 1.7 pp fewer explain their reasoning (p-value $=$ 0.094); 1.7 pp fewer are actionable (p-value $=$ 0.17), and 1.9 pp fewer likely change the code (p-value $=$ 0.072).

IV.B.1 Downstream Consequences for Code Quality

Since proximity to teammates increases substantive feedback, it increases engineers' opportunities to learn, leading to lasting improvements in the quality of engineers' code.

Starting in December 2020, the company began recording metrics of code quality. Figure 3 analyzes these metrics from December 2020 to when the offices first reopened in early 2022. We show differences between engineers who had been on one- and multi-building teams, adjusting for engineer age, tenure, and engineering group.[23]

Engineers who had been on one-building teams before the closures were 2.37 pp less likely to add files that got deleted than engineers who had been on distributed teams (p-value $=$ 0.013, Figure 3a). A file may be deleted because it was easier to start from scratch than sort through a tangled mess of logic (what engineers call "spaghetti code"). Alternatively, the file might be deleted because it introduced a feature that the firm later deemed unnecessary. Either way, adding files that get deleted is not a good sign about

[23]One-building team engineers were consistently on co-located teams before the closures (see Section III).

the quality or utility of an engineer's code.

Engineers who had been on one-building teams were 3.09 pp less likely to introduce bugs (p-value $= 0.0012$, Figure 3b). We define bugs as writing programs that get fully reverted. Typically, this arises because an engineer changed some code, a problem then emerged, and to remedy this quickly the team reverted to the earlier version of the code.

Suggestively, much of these gaps persist when we include fixed effects for the current team: when two engineers work on the same team, the one who had been on a one-building team before the closures tends to write less buggy code (Table A.5). These differences in code quality eventually fade, but it takes years for engineers who had been on multi-building teams to catch up to the code quality of engineers who had been on one-building teams (Figure A.6).

Together these results suggest that engineers who sit with their teammates receive more substantive feedback, which builds their skills and enables them to write higher-quality code in the long run.

IV.C Heterogeneous Effects of Proximity on Feedback

Feedback can help build both firm-specific and general human capital. Commenters can build programmers' firm-specific human capital by, for example, sharing information about the firm's proprietary tools. As such, we would expect less-tenured workers to receive more comments. Commenters can also build programmers' general human capital by, for example, sharing advice on how to structure code. As such, we would expect younger workers to receive more comments.

When engineers are proximate to their teammates, this is exactly what we see: engineers who are less-tenured or younger receive appreciably more feedback. By contrast, when engineers are distributed, less-tenured and younger engineers receive hardly more feedback than their more experienced colleagues.

Figure 4a illustrates the heterogeneity by firm tenure. Comparing horizontally across the

two panels, we see that engineers who are newer to the firm — and so have the most to learn about the firm's coding practices and tools — receive more feedback about their code when the offices are open. Among these junior engineers, the ones who receive the most feedback are those on one-building teams. When the offices close, feedback declines and converges to a uniform lower level, regardless of engineers' tenure. The decline for junior engineers is particularly pronounced for those who had been on one-building teams, who lose 2.03 more comments per program around the office closures than juniors whose teams were already distributed (p-value $= 0.001$). When engineers are distributed, junior engineers have less opportunity to learn about the firm.

We see similar patterns by engineer age in Figure 4b. After the offices close, younger engineers — who likely have the most to learn about general coding practices — cease to receive more feedback than older engineers. Young engineers who had been co-located with their teammates are particularly affected by the closures, losing 2.47 more comments per program around the office closures than young engineers on already-distributed teams (p-value 0.0001). The heterogeneity by age persists when accounting for tenure at the firm (in months) and its interaction with proximity (Column 2 of Table A.6).

When we examine the individual words in the comments, the patterns are consistent with less-tenured engineers losing more firm-specific tips and younger engineers losing more generally-relevant advice when no longer proximate to teammates. Both young and junior engineers lose out more on most words in the feedback (Figure A.7). But junior engineers lose out more on words that suggest the commenter is telling the programmer about the code's business purpose, specifically **customer** and **product**. Younger engineers disproportionately lose out on words that may be broadly relevant to coding, such as discussing what **functions return** or asking them to **add comments** to their code.

By Gender: Women receive more feedback on their code than men do, but only when they are sitting in the same building as all their teammates (Figure 4c). When the offices are open, women on one-building teams receive more feedback than their male counterparts, while on multi-building teams, women receive *less* feedback than their male

counterparts. When the offices close, feedback declines to a lower level regardless of gender. And female engineers on one-building teams lose fully 3.71 (38.9%) more comments per program than do female engineers whose teams were already distributed (p-value< 0.0001). These large differential losses persist when controlling for both engineer tenure and age (Columns $5-6$ in Table A.6).

Men are not inured from the effects of losing proximity to teammates, however. Men lose 1.01 comments per program (13.1%, p-value $= 0.047$). For men, these effects are concentrated among young people (the gray squares in the first and third columns of Figure 4d). For women, the effects extend to older engineers who are new to the firm (the blue triangle in the second column). For engineers who are neither young nor new to the firm, women and men both see null effects of proximity on feedback (the fourth column).

Women appear to be more reluctant to ask other people to invest in their development, especially online. On average, engineers ask two people for feedback on each program. Figure A.8a shows that when engineers lose proximity to their teammates, women receive feedback from 0.26 (14.7%) fewer people per program (p-value $= 0.0078$). By contrast, the change for men is negligible at 0.05 (2.6%) commenters per program (p-value of gender difference $= 0.0056$). When women lose proximity to teammates, they stop asking as many people for feedback, while men continue much as before.

One might worry that, when sitting together, male teammates "mansplain" to women, and the additional feedback that women receive is a burden rather than a benefit. Several patterns push against this interpretation. First, we see that when women lose proximity to their teammates, feedback from other women as well as from men declines (Figure A.9a). Second, when women sit near their teammates, they receive fewer rude comments on their code (Figure A.9b).[24] Finally, we see that when engineers lose proximity to their teammates, women lose out more on comments that are helpful, well-reasoned, actionable, and impactful (Figure A.10), just as do younger and less-tenured engineers. This suggests the additional feedback women receive is beneficial, not burdensome.

[24]To evaluate whether comments were rude, we used the same process as for evaluating comments' substance (see Appendix I.C for details).

In sum, these patterns indicate that losing proximity likely makes it particularly hard for less tenured, younger, and female engineers to learn about firm-specific coding practices and general techniques for writing better code.

V Proximity's Tradeoffs: Evidence from Office Closures

Feedback can build engineers' skills, but at what cost? And who bears this burden?

Figure 5a turns the lens away from who *receives* feedback — our focus thus far — and towards who *gives* feedback. When engineers sit near their teammates, they receive more feedback *from* experienced engineers (in the right plot of Figure 5a), with no significant effect on feedback from less experienced engineers (in the left plot). Providing feedback likely takes time for senior engineers, who must read and understand the code, diagnose any potential issues, suggest changes and explain their rationale.

If providing more or better feedback takes time, we would expect thoroughly reviewing code to come at the cost of senior workers' own coding output. The right plot of Figure 5b supports this hypothesis. While the offices were open, senior engineers on one-building teams wrote 0.76 fewer programs per month for the firm's main code-base (p-value $=$ 0.0005). Once the offices close, this difference disappears. Output declines across the board, potentially due to other pandemic stressors. But compared to engineers whose teammates are already distributed, senior engineers who lose proximity to their teammates see relative increases in output of 0.58 programs per month (p-value $=$ 0.0014).

We can also evaluate the relationship between proximity and programming output for junior engineers. For these engineers, receiving and responding to feedback may both (i) take time in the short run and (ii) build their human capital in programming in the long run. The left plot of Figure 5b supports both points. When the offices close, junior engineers on one-building teams see a relative *increase* of 0.3 programs per month (p-value $=$ 0.0092), consistent with spending less time on feedback increasing short-run output. Once the offices close, engineers trained on one-building teams have a relatively higher *level* of programming output than engineers trained on multi-building teams, suggest-

ing that prior feedback increased their human capital and long-run output. This echoes Section IV.B.1, where we find that engineers who had been on one-building teams write higher-quality code in the long run.

For all engineers, our difference-in-differences design indicates that losing proximity to teammates increases immediate output by 0.48 programs per month (p-value $= 0.0002$, Column 1 of Table A.7). We see similar effects for other metrics of output, including total lines of code written and total number of files changed, both overall and by seniority (Columns $3-6$).[25] Results are similar for alternative specifications in Table A.8.

This empirical finding aligns with anecdotal accounts at the firm. When we presented our findings internally, one senior engineer described the results as "a punch in the gut," explaining that she felt more productive at home but had worried that it was due to mentoring less. To her — and the other nodding heads in the Zoom room — our results confirmed this concern.

The tradeoffs from proximity are particularly acute for women. For junior women, being near teammates is more important for receiving feedback than it is for junior men (the first blue triangle versus gray square in the left plot of Figure A.11, p-value of difference $= 0.094$). Yet for senior women, sitting together is more costly for their output. When senior women are no longer sitting near their junior colleagues, they write 1.17 more programs per month, which is economically though not statistically significantly bigger than the effect for senior men of 0.55 programs per month (the right plot of Figure A.11).

Taken together, these results indicate that being near teammates boosts the human capital formation of junior engineers. But this learning isn't free: instead, the price is paid in senior engineers' time. When senior engineers are not sitting near their junior colleagues, they get more done, with particularly acute tradeoffs for women.

[25] Our preferred metric of output is programs written since engineers encourage shorter, more modular programs, which are easier to test and debug.

VI Code Quantity vs. Quality: Evidence from Office-Reopening

We next use the return-to-office mandates as another source of shocks to proximity to investigate how being near teammates affects code quantity and quality.

VI.A Differential Changes in Proximity

The RTOs led to larger upticks in engineers' proximity to their teammates for those on co-located teams — which were all on the main campus — than for those on distributed teams — which had at least one member on a different campus or remote (because they lived far from any campus).[26] Throughout our analysis, we limit attention to engineers who themselves all worked on the main campus, but whose teammates' location varied.

Figure 6a shows that the second RTO had a bigger impact on proximity than the first. During the first RTO, an engineer on a co-located team typically worked in the same place as less than 10% of her teammates. Engineers came into the office relatively rarely, and when they did, few of their teammates were also in the office. During the second RTO, engineers on co-located teams spent their workdays near substantially more of their teammates than did engineers on distributed teams (27% versus 13%).[27] The gap is particularly pronounced for more intensive forms of togetherness: 58% of co-located teams spent at least one day per month with the whole team in the same office compared to only 1% of distributed teams. Since distant teammates can pose externalities for the rest of the team (Section IV.A), these fully in-office days may be particularly important for establishing team cohesion, which may encourage engineers to invest in each other's development.

VI.B Implications for Code Quantity & Quality

We test whether our earlier findings replicate: does proximity to teammates reduce programming quantity — especially for experienced engineers? Does it improve coding quality — especially for inexperienced engineers? We find support for both predictions.

[26] Remote workers lived an average of 123 miles from the office, compared to 25 for on-site workers.

[27] In the second RTO, engineers on co-located teams went to the office 41% of weekdays. On those days, 64% of their teammates were in, meaning they were with 27% of their teammates on average ($= 100\% \times 0.41 \times 0.64$). When distributed engineers went to the office on 39% of weekdays, only 33% of their teammates were in, so they were with 13% of their teammates on average ($= 100\% \times 0.39 \times 0.33$).

Code Quantity: Figure 6b presents our analysis of code quantity. During the fully remote period (to the left of the first dashed line), there is little difference in engineers' productivity regardless of whether they are on co-located versus distributed teams. When everyone is remote, office assignments do not matter. This continues to be true during the first RTO that engendered a limited increase in proximity to teammates, even for co-located teams. In the second RTO (beyond the second dashed line), this dynamic shifts: co-located engineers start writing less code as they start spending more time with their teammates. Compared to engineers on distributed teams, they write 0.91 (11.7%) fewer programs per month during the second RTO (p-value $= 0.044$).

Spending more time with teammates reduces programming output only for more experienced engineers, as measured by either tenure or age. Among engineers with more than sixteen months of tenure (the blue triangles), engineers co-located with their teams write fewer programs per month during the second RTO, with no significant difference for engineers with less tenure (the orange triangles). Analogously, for engineers who are at least 29 years old (the blue squares), those on co-located teams write 1.4 fewer programs per month during the second RTO (p-value $= 0.0064$), a significantly bigger gap than in the first RTO or the office closure (Columns $4-5$ of Table A.9a). By contrast, there is no detectable difference for younger engineers (the orange squares).

The patterns are similar when we include the full suite of controls (Figure A.12a) and for alternative measures of programming output like lines added (Table A.10).

Code Quality: Proximity to teammates improves the *quality* of code written by inexperienced engineers. Figure 6c illustrates our analysis of "disposable" code, which gets deleted either because it was poorly written or misdirected. While everyone is fully remote, there are minimal differences in disposable code. During the first RTO, relatively little changed. But during the second RTO, co-located engineers added 2.2 pp fewer files that got deleted than engineers on distributed teams (p-value $= 0.041$). This pattern is driven by less-tenured engineers, where the differential was 5.5 pp (p-value $= 0.097$), and young engineers, where the differential was 4.6 pp (p-value $= 0.016$). By contrast, we see

minimal differences among experienced engineers, as measured by either age or tenure.

We see the same patterns echoed in the more acute measure of bugs (Figure 6d). Until the second RTO, there are limited differences in bugs across team types. During the second RTO, however, engineers on co-located teams became 1.4 pp less likely to introduce bugs than those on distributed teams (p-value $= 0.022$), which is significantly different than the gap in both the first-RTO and closure period (Columns $1-2$ of Table A.9c). This difference is particularly pronounced among engineers who are new to the firm who are 2.7 pp less likely to introduce bugs on co-located teams than on distributed ones (p-value $= 0.019$).

We probe the robustness of these code-quality results to alternative controls. The results on deletions are nearly identical with the full suite of controls (Figure A.12b). For the rarer outcome of introducing bugs, adding the full suite of controls does not appreciably change the point estimates but reduces their precision (Figure A.12c).

VII Downstream Implications

When sitting together, inexperienced engineers receive more feedback on their work and write higher-quality code. Thus, human-capital concerns may influence firm hiring decisions and individual office-attendance decisions.

VII.A Who is Hired?

Our results indicate that without proximity to coworkers, it is harder to build engineers' human capital. Thus, one possible response to remote work would be for the firm to shy away from hiring inexperienced engineers and to, instead, buy talent built at other firms. Figure 7a shows the firm's hiring is consistent with this response. When the offices are closed (in orange), the firm hires older engineers, effectively buying talent built at other firms. By contrast, when the offices are open — both pre-closure (in light blue) and post-RTO (in dark blue) — the firm hires younger engineers, who may need to build their human capital at the firm. Indeed, Figure 7b shows that the whole age distribution of hires shifts: over half of hires are under the age of 29 both before the offices closed and after they re-opened, compared to less than a third when the offices were closed.

We can further investigate this phenomenon by considering *where* — not just *when* — people are hired. For main-campus jobs, most new hires' teammates are also on the main campus, allowing for proximity if the offices are open. For other jobs — which are fully remote or in satellite campuses — new hires almost always have distant teammates.[28] For these always-distributed jobs, the firm systematically hires older workers than it does on the main campus (Figure 7c). The age gap is 7-10 years when offices are open. By contrast, during the office closures — when *everyone* was remote from their teammates — the firm hired older workers everywhere, so the age gap significantly narrowed to just a few years.[29] This pattern is not unique to the firm's software engineers but instead is broadly replicated in the rest of the firm's corporate population (Figure A.14).

While labor-supply-side factors could contribute to these patterns, these results are consistent with proximity shifting the relative labor demand for younger versus older engineers. When the firm cannot ensure physical proximity — because of either the timing or location of the job — it hires older workers who have already picked up skills at other firms.

VII.B Who is Poached?

We can also investigate other firms' hiring decisions by looking at who gets poached from our retailer. We define engineers as being poached if they voluntarily leave and say that they are going to a better job.[30]

When the offices are closed — and face-to-face mentorship is limited — we find that engineers who have built more human capital at the retailer are more likely to be poached by other firms. Figure 8a shows that 1.2% of engineers who had been on one-building teams when the offices were open leave for better jobs each month, compared to 0.9% of engineers who had been on multi-building-teams of the same tenure, age, and engineering group (p-value of difference $= 0.044$).[31] These differences added up over the course

[28] Indeed, 98% of engineers hired outside the main campus have teammates assigned to other offices.

[29] The patterns are qualitatively similar when excluding remote workers and only focusing on those hired in the main versus satellite campuses (Figure A.13).

[30] Using data from Glassdoor, we see that poached engineers do typically go to positions that pay more.

[31] The differences in quits overall and quits for better jobs specifically are concentrated in the heart of the

of the closures: by the end of the closures, almost a quarter of engineers on one-building teams had been poached compared to a sixth of those on multi-building teams.[32] In contrast to the difference in poaching, we do not see significant differences in firings, layoffs, or other types of quits, which are principally for personal reasons (Figure A.16).

The poaching differences are concentrated among engineers who are building more general human capital, not firm-specific skills. Less-tenured engineers on one-building teams receive more feedback, but to the extent it builds firm-specific skills (e.g., about the firm's proprietary tools), we would not expect poaching differences to vary by engineer tenure (Column 1 of Figure 8b). By contrast, younger engineers on one-building teams receive more feedback that likely builds generalizable skills, making them more attractive to other firms, and indeed they are disproportionately poached (Column 2). We see a similar pattern for female engineers who receive much more feedback on one-building teams, which appears to disproportionately help them land better jobs elsewhere. These patterns hold when we account for all dimensions of heterogeneity jointly (in gray triangles).

For engineers on one-building teams to be poached, it must be that they can garner higher returns to their accumulated human capital elsewhere. One contributing factor may be that the returns to human capital at this retailer are not as high as at "Big Tech" firms, like Google and Microsoft. Another contributing factor may be that the firm's pay-raise system is poorly structured to reward differences in human capital *across* teams, as it primarily rewards relative performance within teams. Consistent with this, we see insignificant differences in total compensation between engineers on one- and multi-building teams both before and after the office closures (Figure A.17).

Poaching reduces the firm's returns on investing in engineers' human capital, as engineers who receive greater investments are more likely to be poached. Thus, even if the firm could perfectly observe coworkers' investments in each other's human capital (which it cannot), it would still under-incentivize them (Becker, 1964). Social bonds be-

closures, when other firms would not be confident that they could build talent internally (Figure A.15).

[32] This attrition suggests that our earlier analysis of differences in code quality in Section IV.B may have offered lower bounds. If the best engineers from one-building teams are being poached by more productive firms, it's all the more impressive that one-building-team engineers who remain write higher quality code.

tween coworkers may help to fill this void and mitigate underinvestment in general human capital when engineers are proximate to one another.

VII.C Who Comes into the Office?

Since young engineers gain more from proximity to their coworkers, we would expect them to go into the office more often. Indeed, young people do come into the office more than other main-campus engineers under the same RTO (Figure 9a), even when adjusting for commute time and parental status (Table A.11). However, young people may be coming in for other reasons, like a desire to escape cramped apartments or enjoy workplace amenities. To net out these factors and determine whether proximity to teammates leads young workers to come into the office, we utilize variation in whether engineers expect their teammates to be in the office: engineers on co-located teams can expect more of their teammates to be in than engineers on distributed teams.

The potential for proximity to coworkers drives young engineers' attendance: engineers under the age of 29 are 5.4 pp (23.2%) more likely to come into the office if their teammates are all in the same office (p-value < 0.0001). For most older engineers, co-location is less of a driver. However, for the oldest engineers, who are likely doing the most mentoring, office attendance again rises on co-located teams, creating a U-shaped pattern in office attendance by age in colocated teams compared to a flatter pattern in distributed ones.[33] These differences are concentrated in the firm's primary in-office days and largely absent from Mondays and Fridays, when relatively few engineers are on-site (Figure A.18).

Young engineers on co-located teams go into the office partly to be near their teammates and partly to have face-time with their managers. As shown in Table A.12, young engineers are 2.6 pp more likely to go in if their manager is co-located with them (p-value $=$ 0.0085). But conditional on their manager being with them, they are fully 5.1 pp more likely to go in if the rest of their teammates are headquarters-based (p-value < 0.00001). We see similar patterns by engineers' experience within the firm (Columns $1-2$ of Table A.13). Like young workers, new hires are not only sensitive to whether their manager is

[33] For engineers over forty, the gap between co-located and distributed teams is 5.7 pp (p-value $=$ 0.0004).

in headquarters but also to whether their teammates are (Columns 3−4).

VIII Generalizability

This section investigates downstream implications in large-scale survey data. We, first, ask: are young people more likely to be back in the office? We then ask: in a world where work is more geographically distributed, do more young people struggle to find a job?

VIII.A Who Comes into the Office?

Young engineers are more likely to be in the office throughout the tech sector, not just at our firm. Figure 9b shows this using survey data from StackOverflow, the top Q&A site for software engineers (akin to Wikipedia for tech). Each year, StackOverflow surveys about a hundred thousand engineers from around the world. In Figure 9b, the x-axis plots the age groups elicited by the survey, and the y-axis plots the percent of work done at the office in 2022 and 2023. For US engineers under age 25, 45% are in the office each day, compared to 26% of older engineers (p-value < 0.00001). For other countries, both the youngest and oldest workers are more likely to be back in the office, creating a U-shaped pattern in office attendance (see Figure A.19 for the top countries separately).

At each age, engineers with less coding experience go into the office more (Figure 9b), suggesting that those with the most to learn have the most incentive to be on-site. On average, engineers who started coding in the last five years are 5.8 pp more likely to be in the office than their same-age peers with more coding experience (p-value < 0.00001).

Higher office attendance among young people is not unique to software engineers. We see the same pattern when focusing on all college-educated workers in Figure 9c, which draws on data from the Census's Household Pulse Surveys (U.S. Census Bureau, 2023).[34] The U-shaped pattern persists when limiting to non-parents, indicating it is not simply a product of parents being more likely to work from home (Figure A.21).

Two mechanisms could potentially produce this pattern. First, in any given job, young

[34]We also see similar patterns using alternative data from the Current Population Survey (Figure A.20).

people may choose to go into the office more to build their skills (as we see at the retailer in Section VII.C). Second, young people may be more attractive candidates for jobs where they will be face-to-face with their coworkers (consistent with the retailer's hiring patterns in Section VII.A).

VIII.B Who Can't Find a Job?

Our findings suggest that the rise of remote work may make it relatively harder for young people to secure employment, as employers may question young candidates' ability to learn on the job and their accumulated human capital from previous roles. Consistent with this hypothesis, Figure 10 shows relative increases in young people's unemployment from 2017−2019 to 2022−2024, a period with a dramatic uptick in remote work (Barrero et al., 2021).[35] For software engineers specifically, Panel a shows increases in unemployment for young engineers (of 1.05 pp for those under 29, p-value = 0.045), with no significant change for older engineers.

For all college-educated workers in Panel b, the unemployment rate rose among young workers, while falling for older workers between 2017−2019 and 2022−2024.[36] We investigate whether this aggregate pattern is concentrated in remotable jobs, like software engineering. Figure 10c plots the evolution of age differences in unemployment in remotable and non-remotable jobs as defined by Dingel and Neiman (2020). To classify job type, respondents must either have a current job or a previous one, whose occupation they can report; thus, this analysis excludes newly graduated students, who may also struggle to find a foothold in the labor market.[37] Prior to the pandemic, age differences in unemployment were stable. During the pandemic, young people's unemployment spiked in both remotable and non-remotable jobs relative to that of older people. How-

[35] We exclude the height of the pandemic (2020−2021) from Figure 10a−b, but Figure A.22a−b shows qualitatively similar patterns for 2020−2021 as 2022−2024.

[36] Specifically, for workers under 29, unemployment rose by 0.56 pp (p-value < 0.0001), while for workers over age 29, unemployment rates fell by -0.07 pp (p-value = 0.061).

[37] We separately estimate the following specification for remotable and non-remotable jobs:

$$\text{Unemployed}_{it} = \sum_{\tau \neq 2019} \beta_\tau \text{Age} < 29_{it} 1[t = \tau] + \mu_{a,o} + \mu_{t,o} + \epsilon_{it}, \quad (4)$$

where $\mu_{a,o}$ denotes age by occupation fixed effects and $\mu_{t,o}$ denotes year by occupation fixed effects.

ever, in non-remotable occupations, age differences in unemployment quickly returned to baseline. By contrast, in remotable jobs, unemployment rates of young people remained stubbornly elevated relative to those of older people.[38] The upticks in young people's unemployment in remotable jobs are driven by involuntary and persistent forms of unemployment and absent from voluntary job leaving and temporary layoffs (Figure A.23).

Remote work is not the only candidate explanation for these empirical patterns. Generative AI could also play a role: remotable jobs are disproportionately exposed to AI (Schubert, 2025), which could increasingly automate younger workers out of jobs. This effect may be especially pronounced in 2023−2024, as AI use grew exponentially after ChatGPT's late-2022 release (Patel, 2025). Given these and other possible factors, the unemployment patterns should be interpreted with caution. Nonetheless, they provide suggestive evidence that the rise of remote work has scarring effects on young workers, who struggle to learn on the job in an increasingly distributed world.

IX Conclusion

This paper demonstrates that physical proximity between coworkers meaningfully increases mentorship and skill development, even for software engineers – the ultimate digital natives with an array of virtual communication tools. This face-to-face interaction is crucial for the development of young and less-experienced employees, which explains why younger and newer workers are more likely to be in the office and why their attendance patterns are more sensitive to their teammates' presence.

However, investments in colleagues' skill development are not free; they impede the output of experienced workers who do the mentoring. This presents firms with a tradeoff: remote work enhances productivity today as experienced workers focus on their own output, while imperiling productivity tomorrow, as junior workers receive less mentorship and develop fewer skills.

[38]These findings resonate with Morales-Arilla and Daboín (2021)'s analysis of job postings: while employment was more resilient in remotable jobs initially, job postings in remotable jobs fell more precipitously, particularly in jobs with high returns to experience.

Remote work stands to shift who is hired, with adverse effects for young workers. Our retailer hired fewer young workers when employees were distant from their teammates — both during the office closures and when hiring outside the main campus. This mirrors hiring patterns nationally, where young workers' unemployment rates rose alongside the rise of remote work, particularly in remotable occupations. Adverse impacts may grow over time as remote workers gain fewer skills: we find that workers who were trained apart from teammates were less likely to be poached by other firms.

Our results indicate that remote work may have nuanced impacts on gender equity. While it may help experienced women excel on their own tasks and enable working mothers to remain in the workforce (Harrington and Kahn, 2023; Ho et al., 2024; Jalota and Ho, 2024), it may disadvantage young women, whose professional development appears to be especially sensitive to physical proximity to colleges.

Our research suggests that doses of coordinated work can pay dividends for mentorship. However, one worker's decision to go into the office does not by itself generate the mentorship benefits of team proximity. Distant teammates degrade the connections between teammates who are in the office and teammates separated by even short distances may not reap the gains from proximity. The key it seems is for coworkers to all be in close proximity.

References

Adams-Prassl, Abi, Teodora Boneva, Marta Golin, and Christopher Rauh, "Inequality in the Impact of the Coronavirus Shock: Evidence From Real Time Surveys," *Journal of Public Economics*, 2020, *189*, 104245.

Agrawal, Ajay and Avi Goldfarb, "Restructuring Research: Communication Costs and the Democratization of University Innovation," *American Economic Review*, 2008, *98* (4), 1578–90.

Akan, Mert, Jose Maria Barrero, Nicholas Bloom, Tom Bowen, Shelby Buckman, Steven J Davis, Luke Pardue, and Liz Wilke, "Americans Now Live Farther From Their Employers," Daily Report, Hoover Institution February 2024.

Akcigit, Ufuk, Santiago Caicedo, Ernest Miguelez, Stefanie Stantcheva, and Valerio Sterzi, "Dancing with the Stars: Innovation Through Interactions," Working Paper 24466, National Bureau of Economic Research March 2018.

Atkin, David, M. Keith Chen, and Anton Popov, "The Returns to Face-to-Face Interactions: Knowledge Spillovers in Silicon Valley," Working Paper 30147, National Bureau of Economic Research June 2022.

Azoulay, Pierre, Joshua S Graff Zivin, and Jialan Wang, "Superstar Extinction," *The Quarterly Journal of Economics*, 2010, *125* (2), 549–589.

Barrero, Jose Maria, Nicholas Bloom, and Steven J Davis, "Why Working from Home Will Stick," Working Paper 28731, National Bureau of Economic Research April 2021.

Battiston, Diego, Jordi Blanes i Vidal, and Tom Kirchmaier, "Face-to-face Communication in Organizations," *The Review of Economic Studies*, 2021, *88* (2), 574–609.

Becker, Gary S, *Human capital: A Theoretical and Empirical Analysis, with Special References to Education*, The University of Chicago Press, 1964.

Belden, Martha, "Has WFH changed the way we see?," 2022. All About Vision, https://www.allaboutvision.com/conditions/computer-vision-syndrome/have-screens-changed-vision/ (visited April 10, 2025).

Bird, Steven, Ewan Klein, and Edward Loper, *Natural Language Processing With Python: Analyzing Text With the Natural Language Toolkit*, O'Reilly Media, Inc., 2009.

Bloom, Nicholas, Gordon B Dahl, and Dan-Olof Rooth, "Work from Home and Disability Employment," Working Paper 32943, National Bureau of Economic Research September 2024.

_ **, James Liang, John Roberts, and Zhichun Jenny Ying**, "Does Working from Home Work? Evidence from a Chinese Experiment," *The Quarterly Journal of Economics*, 2015, *130* (1), 165–218.

Bonacini, Luca, Giovanni Gallo, and Sergio Scicchitano, "Working From Home and Income Inequality: Risks of a 'New Normal' With COVID-19," *Journal of Population Economics*, 2021, *34* (1), 303–360.

Brucks, Melanie S and Jonathan Levav, "Virtual Communication Curbs Creative Idea Generation," *Nature*, 2022, pp. 1–5.

Bureau of Labor Statistics, "2024 Current Population Survey Public Use Microdata Samples," 2024. Data retrieved from IPUMS CPS, https://cps.ipums.org/cps/ (visited March 20, 2024).

Catalini, Christian, "Microgeography and the Direction of Inventive Activity," *Management Science*, 2018, *64* (9), 4348–4364.

Charpignon, Marie-Laure, Yuan Yuan, Dehao Zhang, Fereshteh Amini, Longqi Yang, Sonia Jaffe, and Siddharth Suri, "Navigating the New Normal: Examining Coattendance in a Hybrid Work Environment," *Proceedings of the National Academy of Sciences*, 2023, *120* (51), e2310431120.

Chen, Chinchih, Carl Benedikt Frey, and Giorgio Presidente, "Disrupting Science," Working Paper 2022-4, The Oxford Martin Working Paper Series on Technological and Economic Change 2022.

Chen, Tianqi and Carlos Guestrin, "XGBoost: A Scalable Tree Boosting System," in "Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining" 2016, pp. 785–794.

Choudhury, Prithwiraj, Cirrus Foroughi, and Barbara Larson, "Work-from-anywhere: The Productivity Effects of Geographic Flexibility," *Strategic Management Journal*, 2021, *42* (4), 655–683.

Cullen, Zoe B, Bobak Pakzad-Hurson, and Ricardo Perez-Truglia, "Home Sweet Home: How Much Do Employees Value Remote Work?," Working Paper 33383, National Bureau of Economic Research January 2025.

DeFilippis, Evan, Stephen Michael Impink, Madison Singell, Jeffrey T Polzer, and Raffaella Sadun, "Collaborating During Coronavirus: The Impact of COVID-19 on the Nature of Work," Working Paper 27612, National Bureau of Economic Research July 2020.

Dingel, Jonathan I and Brent Neiman, "How Many Jobs Can Be Done at Home?," *Journal of Public Economics*, 2020, *189*, 104235.

Dutcher, E Glenn, "The Effects of Telecommuting on Productivity: An Experimental Examination. The Role of Dull and Creative Tasks," *Journal of Economic Behavior & Organization*, 2012, *84* (1), 355–363.

Emanuel, Natalia and Emma Harrington, "Working Remotely? Selection, Treatment, and the Market for Remote Work," *American Economic Journal: Applied Economics*, 2024, *16* (4), 528–559.

Fenizia, Alessandra and Thomas Kirchmaier, "Not Incentivized Yet Efficient: Working From Home in the Public Sector," Discussion Paper 2036, Centre for Economic Performance, London School of Economics and Political Science 2024.

Gaspar, Jess and Edward L Glaeser, "Information Technology and the Future of Cities," *Journal of Urban Economics*, 1998, *43* (1), 136–156.

Gibbs, Michael, Friederike Mengel, and Christoph Siemroth, "Work from Home and Productivity: Evidence from Personnel and Analytics Data on Information Technology Professionals," *Journal of Political Economy Microeconomics*, 2023, *1* (1), 7–41.

Goldberg, Emma, "The Worst of Both Worlds: Zooming from the Office," November 2021.

Goodman-Bacon, Andrew, "Difference-in-Differences with Variation in Treatment Timing," *Journal of Econometrics*, 2021, *225* (2), 254–277.

Harrington, Emma and Matthew E Kahn, "Has the Rise of Work-From-Home Reduced the Motherhood Penalty in the Labor Market," Working Paper 2023.

He, Haoran, David Neumark, and Qian Weng, "Do Workers Value Flexible Jobs? A Field Experiment," *Journal of Labor Economics*, 2021, *39* (3), 709–738.

Herkenhoff, Kyle, Jeremy Lise, Guido Menzio, and Gordon M Phillips, "Production and Learning in Teams," *Econometrica*, 2024, *92* (2), 467–504.

Ho, Lisa, Suhani Jalota, and Anahita Karandikar, "Bringing Work Home: Flexible Arrangements as Gateway Jobs for Women in West Bengal," Working Paper, Structural Transformation and Economic Growth 2024.

Jackson, C Kirabo and Elias Bruegmann, "Teaching Students and Teaching Each Other: The Importance of Peer Learning for Teachers," *American Economic Journal: Applied Economics*, 2009, *1* (4), 85–108.

Jalota, Suhani and Lisa Ho, "What Works for Her? How Work-from-Home Jobs Affect Female Labor Force Participation in Urban India," Working Paper, SSRN 2024.

Jarosch, Gregor, Ezra Oberfield, and Esteban Rossi-Hansberg, "Learning from Coworkers," *Econometrica*, 2021, *89* (2), 647–676.

Johnsen, Julian V, Hyejin Ku, and Kjell G Salvanes, "Competition and Career Advancement," *Review of Economic Studies*, 2024, *91* (5), 2954–2980.

Kraut, Robert, Carmen Egido, and Jolene Galegher, "Patterns of Contact and Communication in Scientific Research Collaboration," in "Proceedings of the 1988 ACM conference on Computer-supported cooperative work" 1988, pp. 1–12.

Lewandowski, Piotr, Katarzyna Lipowska, and Mateusz Smoter, "Mismatch in Preferences for Working From Home: Evidence From Discrete Choice Experiments With Workers and Employers," Discussion Paper 16041, IZA March 2023.

Maestas, Nicole, Kathleen J Mullen, David Powell, Till Von Wachter, and Jeffrey B Wenger, "The value of working conditions in the United States and implications for the structure of wages," *American Economic Review*, 2023, *113* (7), 2007–2047.

Mas, Alexandre and Amanda Pallais, "Valuing Alternative Work Arrangements," *American Economic Review*, 2017, *107* (12), 3722–59.

_ **and** _ , "Alternative Work Arrangements," *Annual Review of Economics*, 2020, *12* (1), 631–658.

Miranda, Arianna Salazar and Matthew Claudel, "Spatial Proximity Matters: A Study on Collaboration," *PLOS One*, 2021, *16* (12), e0259965.

Morales-Arilla, José and Carlos Daboín, "Remote Work Wanted? Evidence from Job Postings During COVID-19," Global Working Paper 159, Global Economy and Development at Brookings July 2021.

Nix, Emily, "Learning Spillovers in the Firm," Working Paper 2020:14, Institute for Evaluation of Labour Market and Education Policy 2020.

Patel, Dwarkesh, *The Scaling Era: An Oral History of AI, 2019–2025*, Stripe Press, 2025.

Sandvik, Jason J, Richard E Saouma, Nathan T Seegert, and Christopher T Stanton, "Workplace Knowledge Flows," *The Quarterly Journal of Economics*, 2020, *135* (3), 1635–1680.

Schubert, Gregor, "Organizational Technology Ladders: Remote Work and Generative AI Adoption," Working Paper, SSRN April 2025.

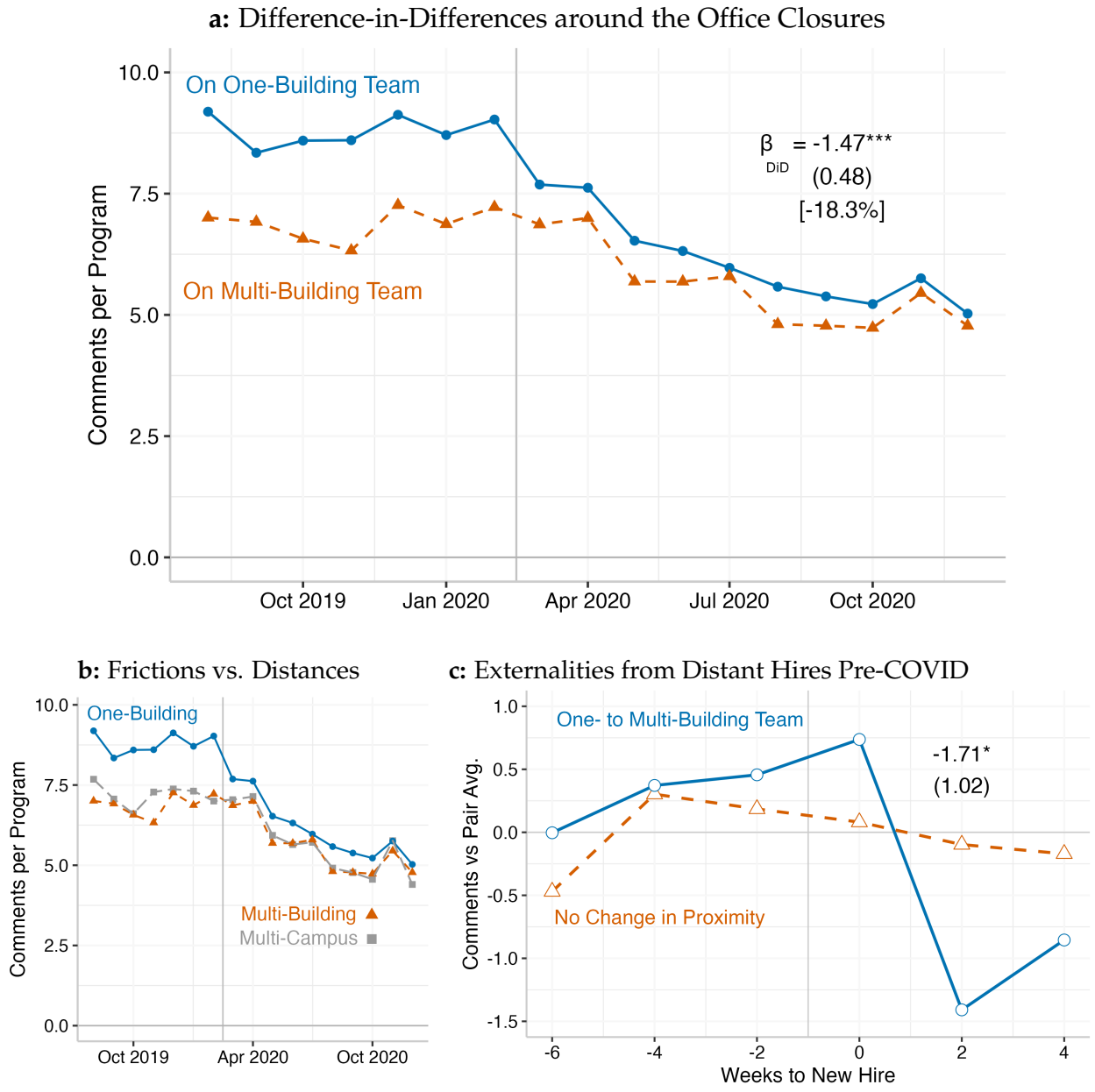
U.S. Census Bureau, "2019 American Community Survey Public Use Microdata Samples," 2019. Data retrieved from IPUMS USA, https://usa.ipums.org/usa/ (visited January 19, 2023).

_ , "Household Pulse Survey," 2023. Data retrieved from Cenus, https://www.census.gov/programs-surveys/household-pulse-survey/datasets.html (visited January 19, 2023).

Waldinger, Fabian, "Peer Effects in Science: Evidence from the Dismissal of Scientists in Nazi Germany," *The Review of Economic Studies*, 2012, *79* (2), 838–861.

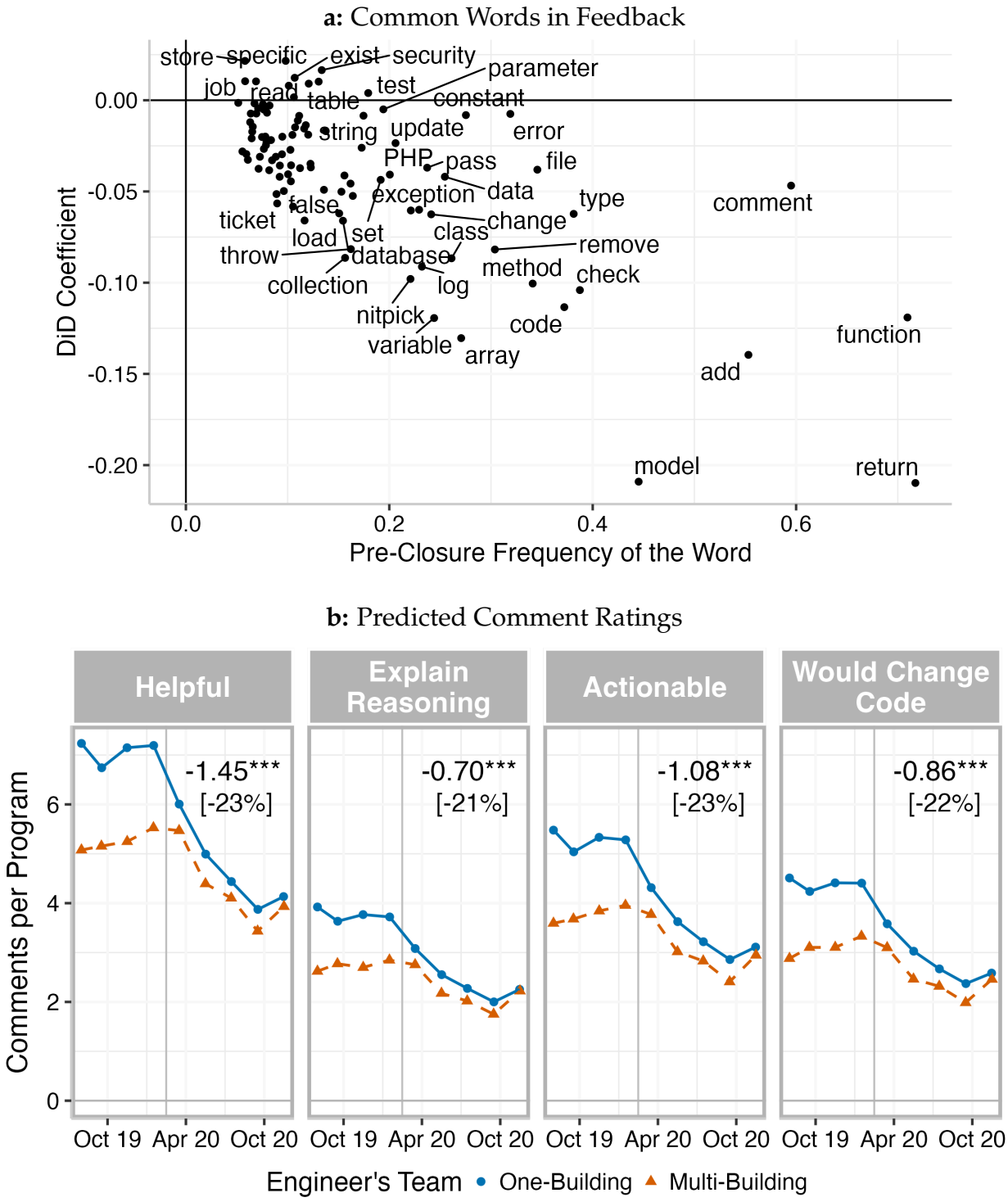
Yang, Longqi, David Holtz, Sonia Jaffe, Siddharth Suri, Shilpi Sinha, Jeffrey Weston, Connor Joyce, Neha Shah, Kevin Sherman, Brent Hecht et al., "The Effects of Remote Work on Collaboration Among Information Workers," *Nature: Human Behaviour*, 2022, *6* (1), 43–54.

Figure 1: Proximity to Teammates and Online Feedback



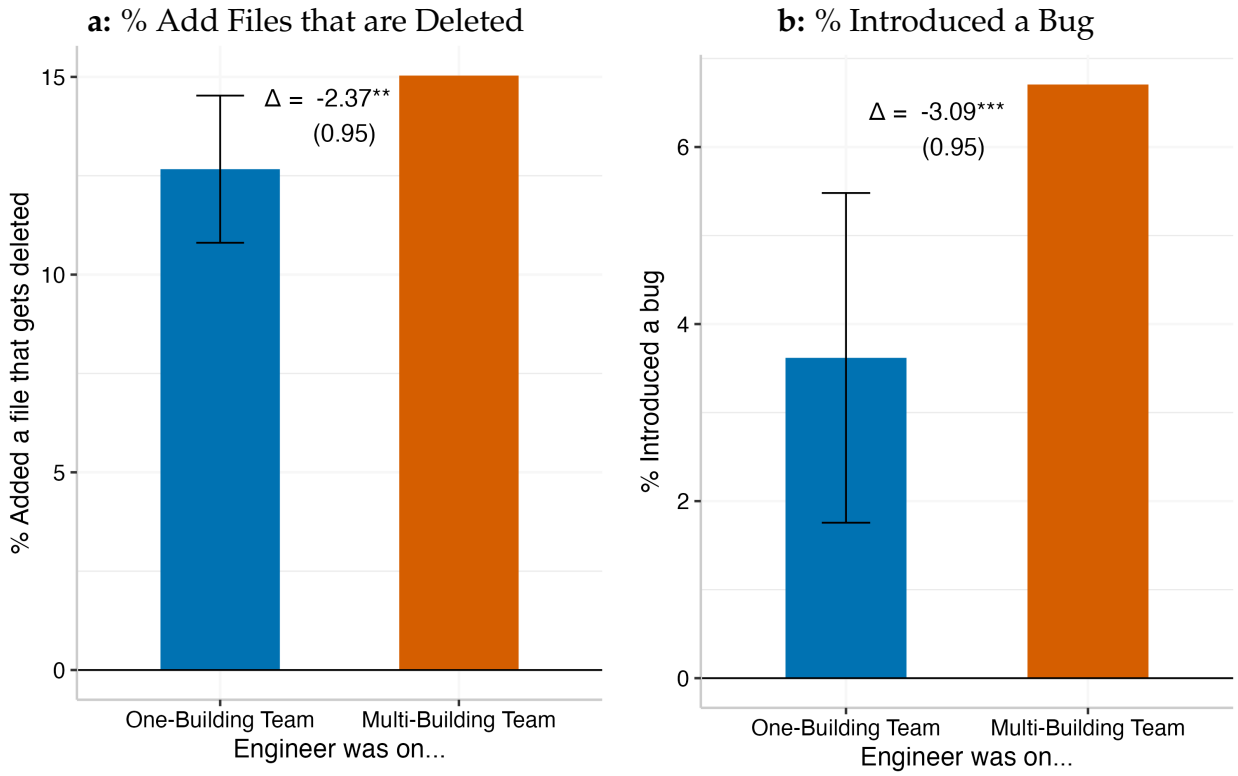
Notes: This figure analyzes whether engineers who are physically proximate to their teammate receive more feedback online. Panel a illustrates our difference-in-differences (DiD) design that compares engineers on one-building, co-located teams (N=637) to engineers on multi-building, distributed teams (N=418) around the COVID-19 office closures (the gray line). The plot residualizes by our preferred controls for program scope, engineer age, tenure, and engineering group. The reported coefficient is the pooled DiD with engineer fixed effects (Table 2, Column 6), with the percent change in brackets. Panel b extends the analysis to include multi-campus teams. Panel c shows a complementary, pre-COVID design, which compares new hires that convert their team from one- to multi-building teams (N=16 teams) versus new hires that do not change the collocation of their team (N=119 teams). This analysis focuses on co-located teammates who both pre-date the 6-week pre-period. The plot shows comments per program relative to the average in the coder-commenter pair, with the annotated DiD coefficient from Equation 3. Standard errors are clustered by team. $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$.

Figure 2: Effects of Losing Proximity on Substantive Feedback



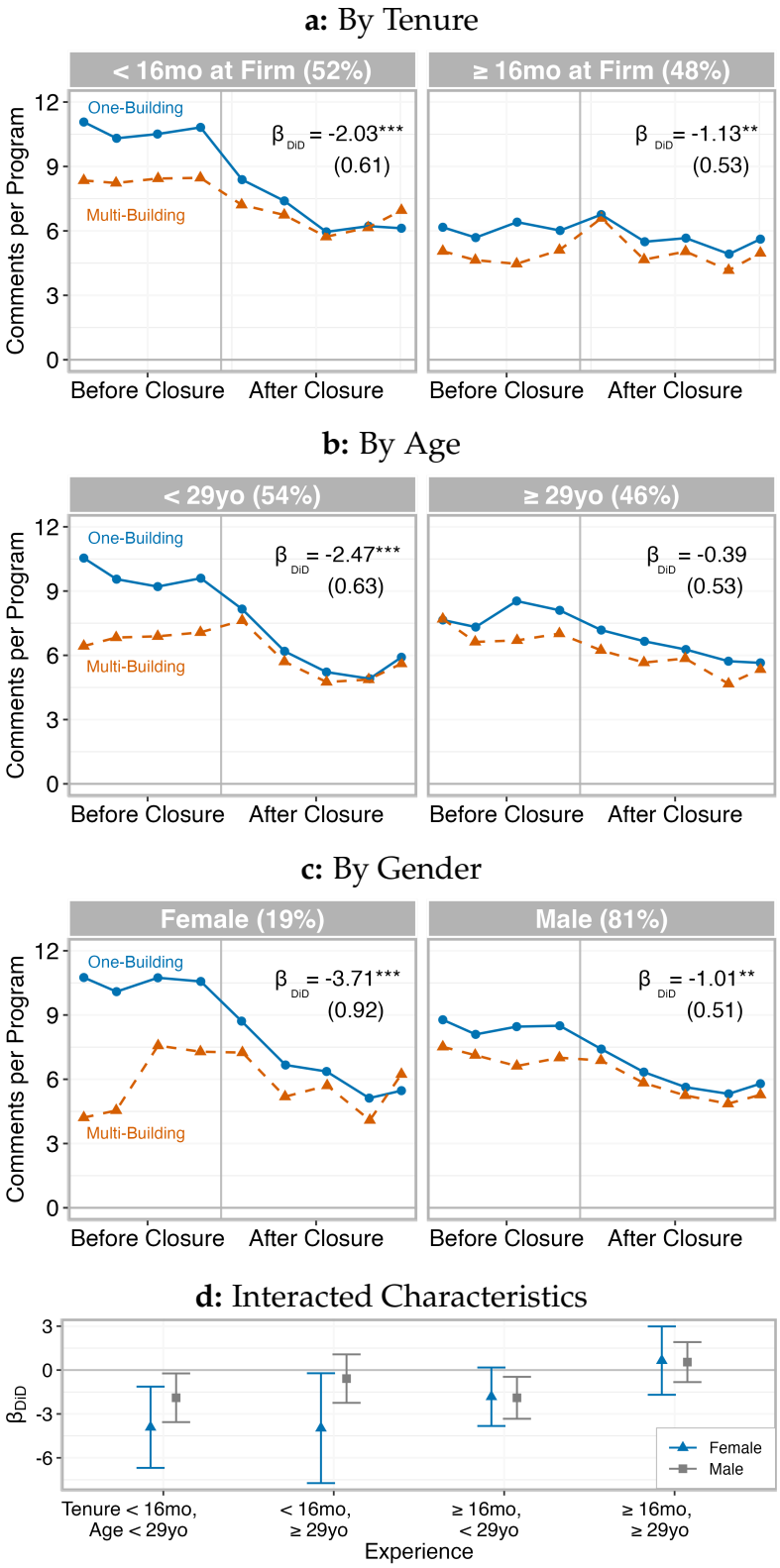
Notes: This figure examines feedback content. Panel a illustrates how losing proximity to teammates affects individual word usage, with each word's pre-closure frequency on the x-axis and the difference-in-differences (DiD) coefficient on the y-axis. Panel b shows the impact of losing proximity on predicted comment quality. Predictions come from a supervised machine learning algorithm applied to the human ratings of external software engineers (see Appendix I.C). The annotated coefficient reflects the DiD, with percentage effects in brackets. All DiD specifications include engineer fixed effects and our preferred month-specific controls for engineer age, tenure, and engineering group. Standard errors are clustered by team. $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$.

Figure 3: Downstream Differences in Code Quality



Notes: This figure examines long-run differences in code quality, using two different quality metrics. The period covers December 2020 (when these metrics started to be recorded) until the office re-opening in 2022 (see Figure A.6 for the full time series). Panel a shows the percent of programs where the engineer added a file that later got deleted. Files may be deleted because the code was fully rewritten or because the firm decided the feature was a dead-end. Panel b focuses on introducing a bug, as defined by all the changes that the engineer makes getting reverted by a subsequent program. The annotated coefficient compares the two sets of engineers, with controls for engineer age, tenure, and engineering group. An engineer is defined as being on a one-building team if they were consistently on a co-located team in the pre-closure period (fall of 2019 and winter of 2020), and as being on a multi-building team otherwise. Standard errors are clustered by team. *p<0.1; **p<0.05; ***p<0.01.

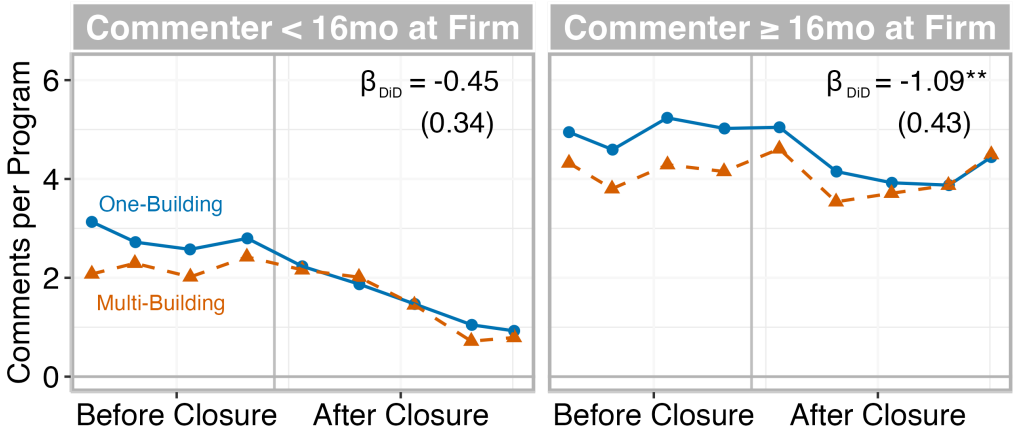
Figure 4: Heterogeneous Effects of Proximity



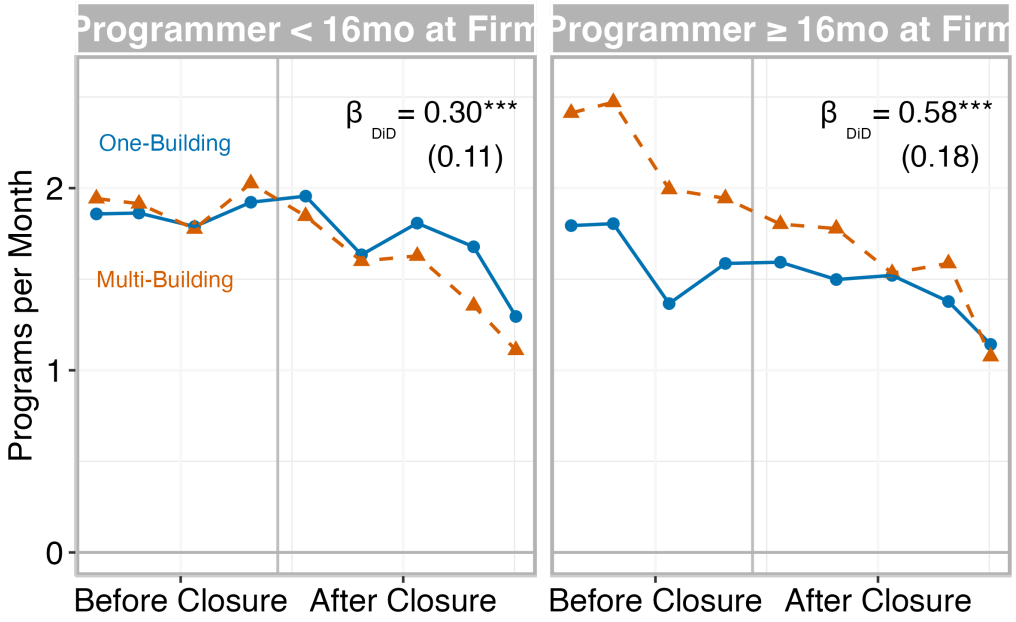
Notes: This figure replicates Figure 1a by (a) tenure (versus the mean of 16 months), (b) age (versus the mean of 29), and (c) gender. Tenure and age are measured at the beginning of each month. Panel d shows the DiD coefficients from a regression with all the interactions. All specifications use our preferred controls and cluster by engineering team. $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$.

Figure 5: Tradeoffs of Proximity

a: Comments Given by Tenure

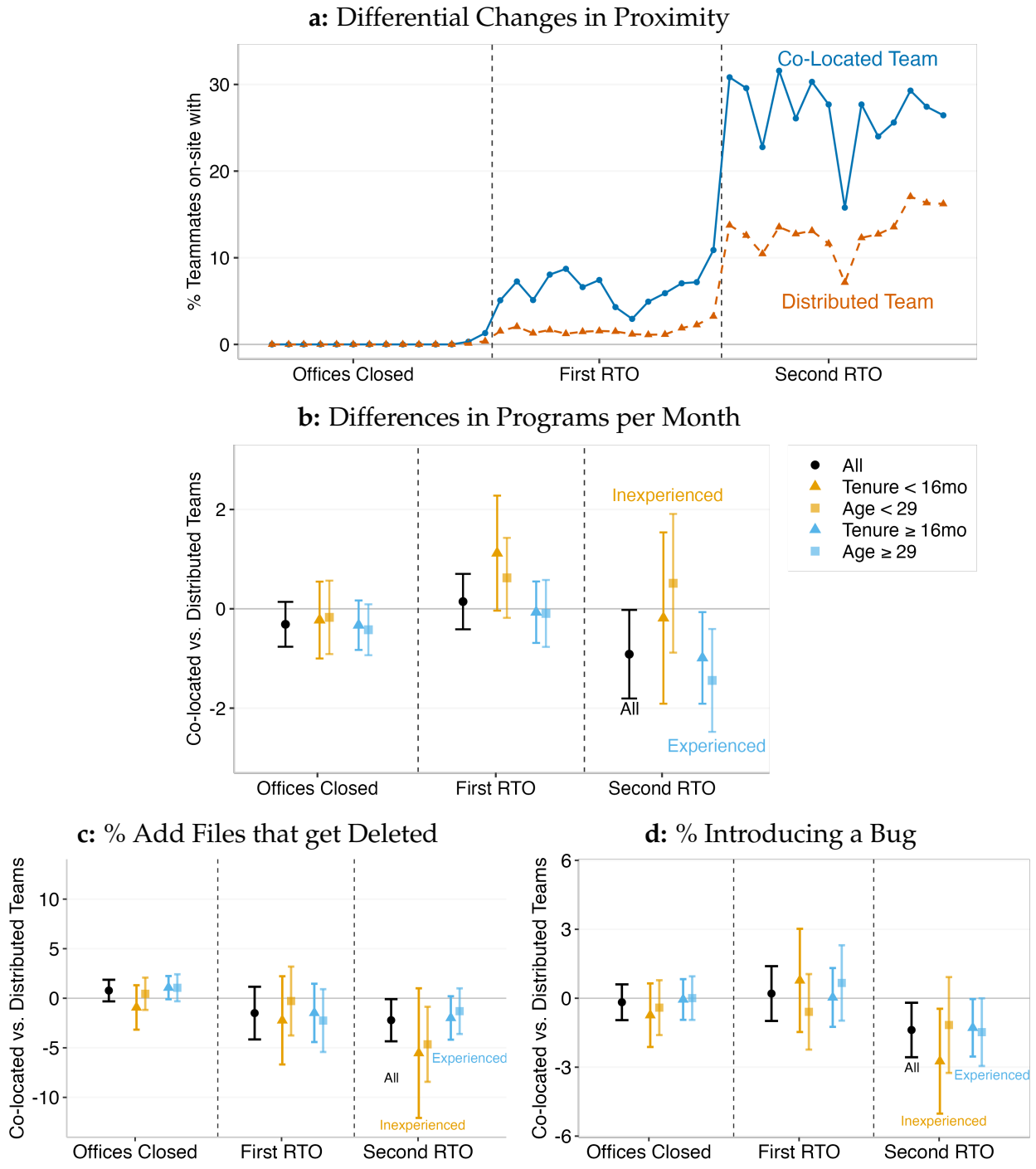


b: Programs Written by Tenure



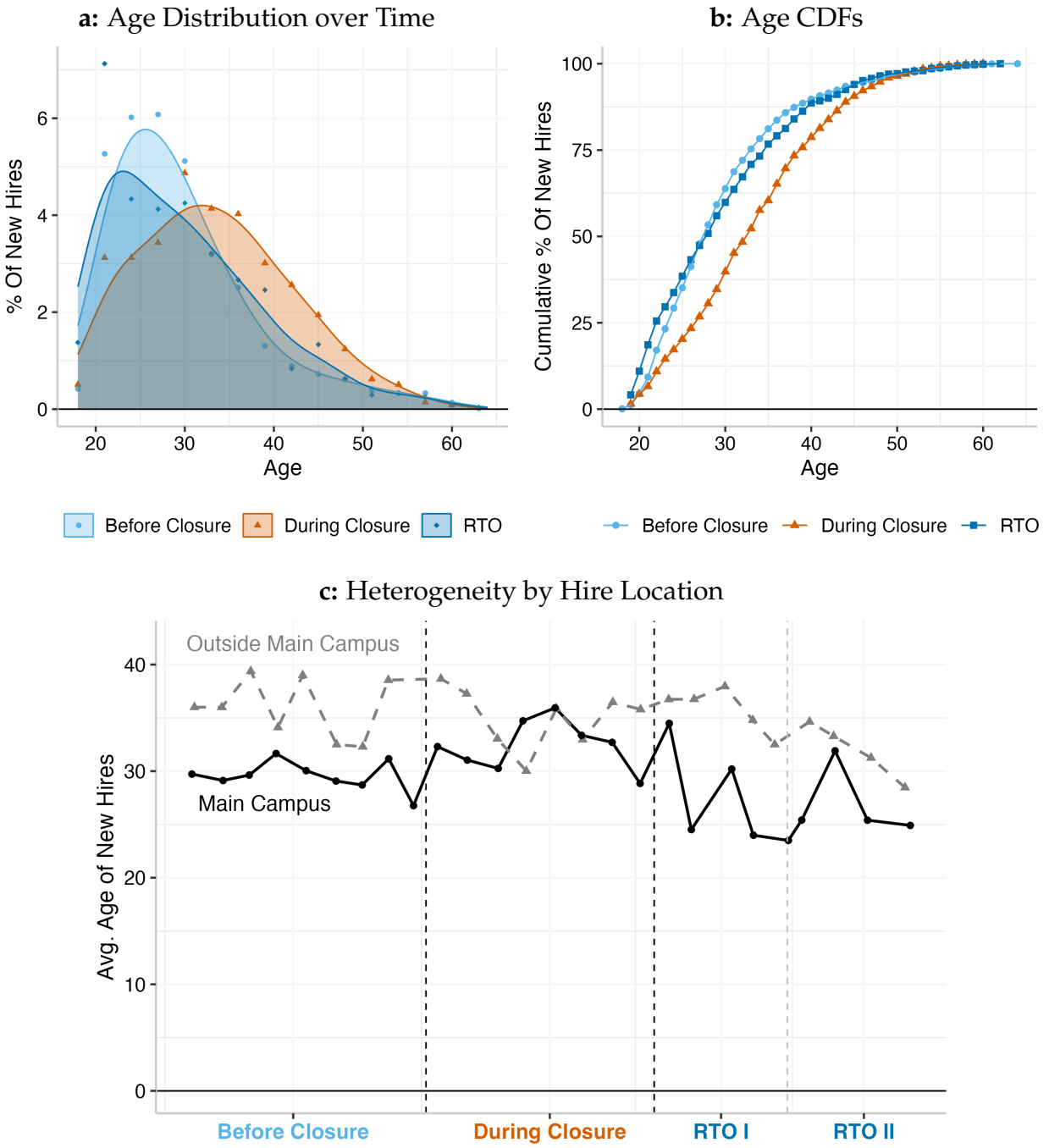
Notes: This figure illustrates the tradeoffs of proximity for engineers of different tenures. Panel a shows comments per program broken down by the seniority of the commenter rather than the program writer. Panel b shows programs written per month. Both plots residualize by our preferred controls for engineer age, tenure, and engineering group, as well as program scope in Panel a. In each plot, tenure is measured at the beginning of each month. The annotated coefficients report the pooled DiD estimate (Equation 1) for the respective subsample, with standard errors clustered by team. $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$.

Figure 6: Return to Office (RTO) and Code Quantity & Quality



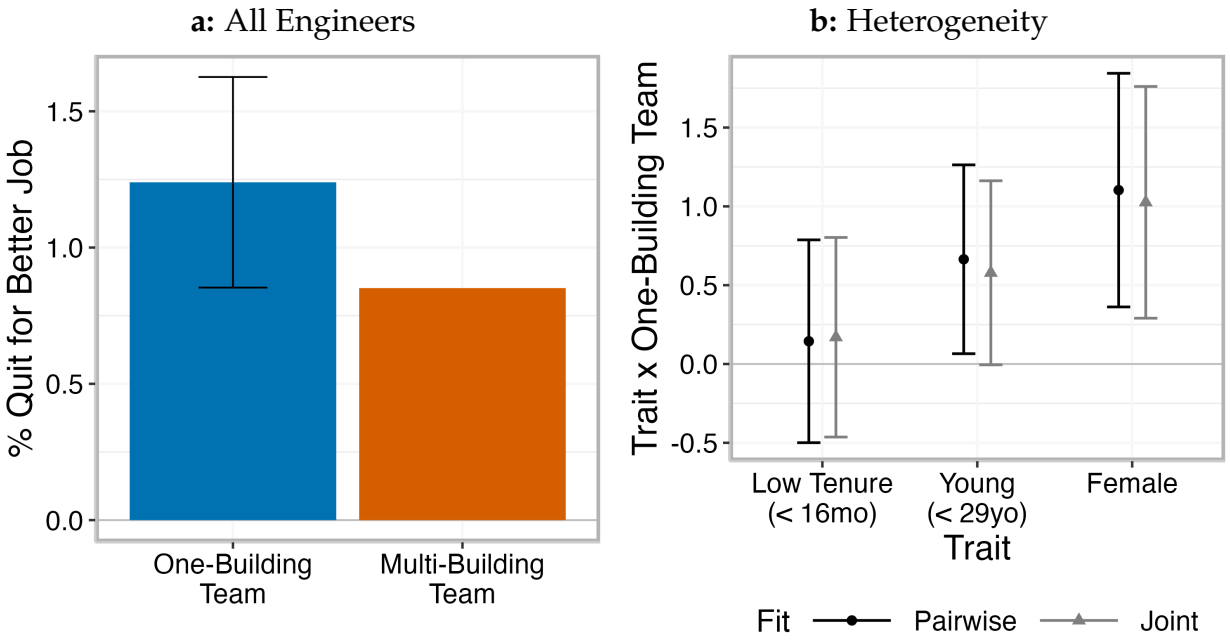
Notes: This figure illustrates changes around the firm's return-to-office (RTO) phases, comparing engineers in the headquarters (HQ) who are on teams that are fully HQ-based versus those that are distributed. Panel a plots face-to-face time with teammates: for each weekday, the measure is zero if the worker is out of the office and otherwise is the share of her teammates also in the office. Panels b-d show differences between engineers on HQ-based and distributed teams with controls for engineer age, tenure, engineering group, and engineer fixed effects. Experience is measured both by tenure at the firm (using a 16-month cutoff) and age (using a 29 year-old cutoff). Tenure and age are both measured at the beginning of the month for each engineer. Error bars represent 95% confidence intervals, with clustering by team.

Figure 7: Younger Hires when Proximity is Feasible



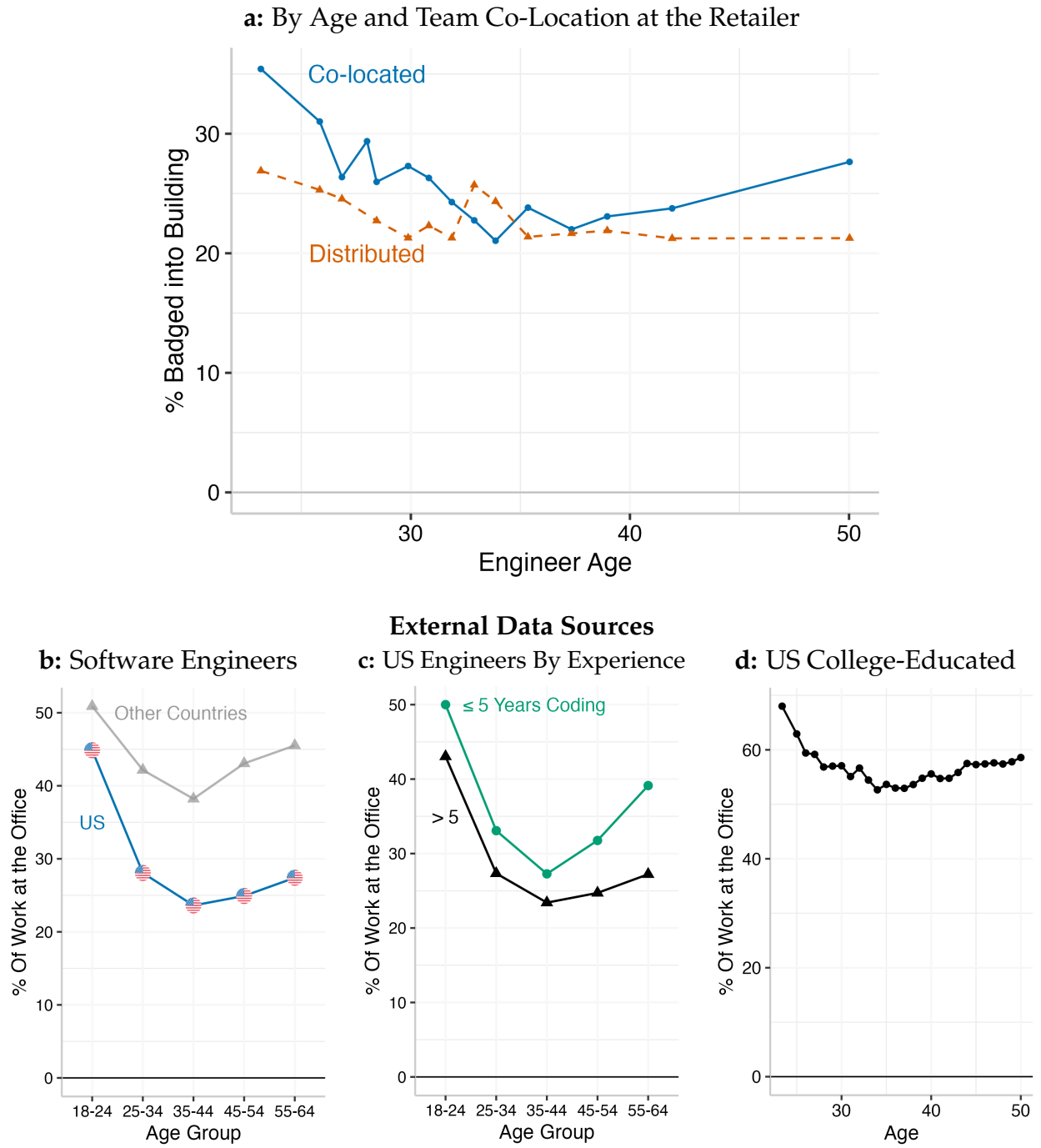
Notes: This figure illustrates the shifting age distribution of new hires among the tech workers at the firm. Panel a shows the distribution of new hires, the kernel densities are Gaussian densities with bandwidths of three years and the points show average densities in three-year age-groups. Panel b shows the cumulative distribution over every age. Both plots differentiate between the pre-closure period (in light blue), the closure period (in orange), and the re-opening and return-to-office (RTO) period (in dark blue). Panel c focuses on heterogeneous trends by the location of the new hire. The solid black line shows the pattern for those hired in the main campus, who would likely sit with their teammates when the offices were open. The dashed gray line shows the pattern for those hired outside the main campus, who would likely be distant from at least some teammates regardless of whether the offices were open (98% of these engineers have teammates assigned to other offices).

Figure 8: Poaching from the Firm during the Office Closures



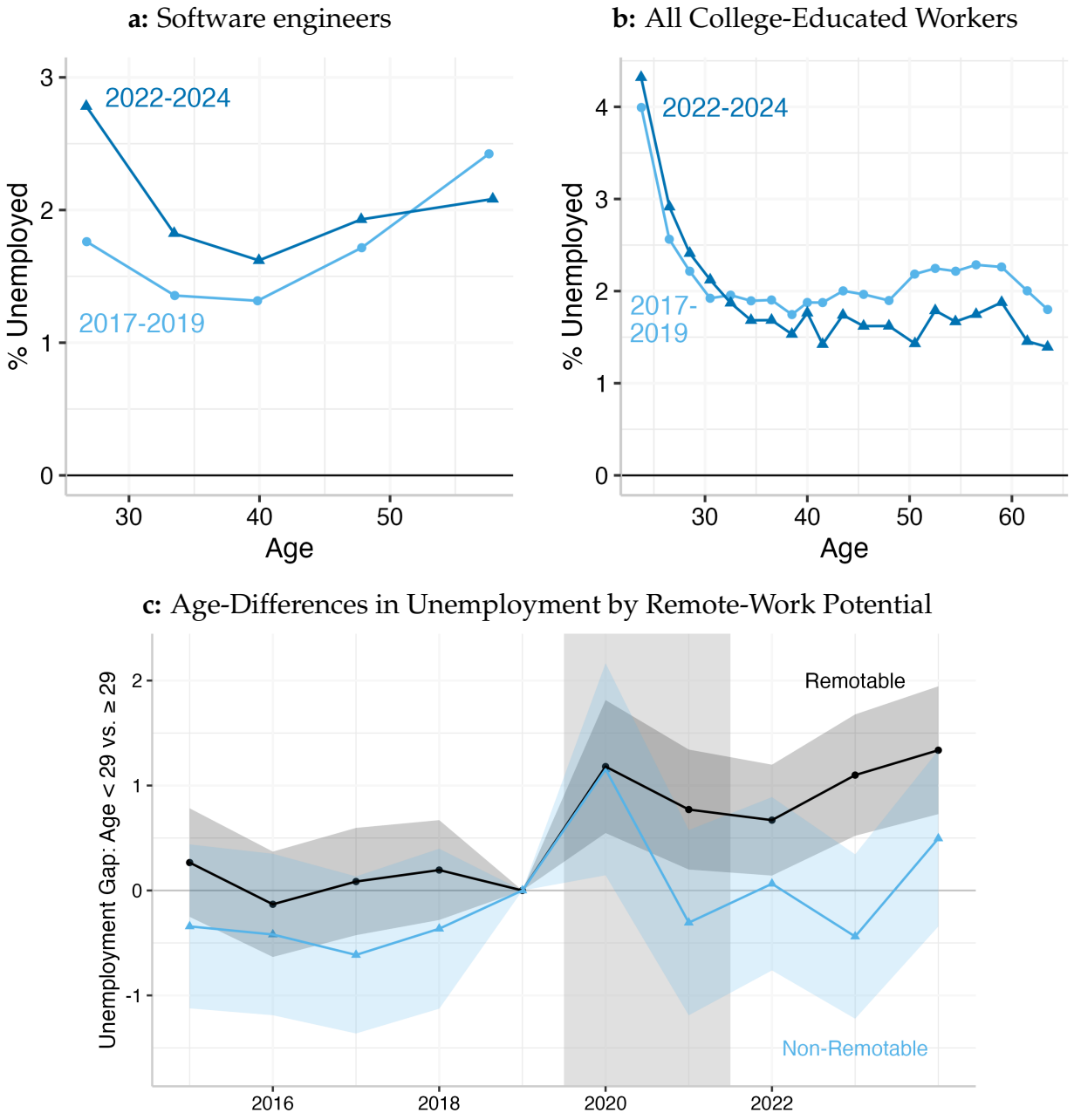
Notes: This figure investigates whether engineers who receive more human capital investments are more likely to be poached by other firms, especially when those investments are in general human capital. The analysis focuses on when offices were closed, proximity is impossible, and so other firms may be more inclined to hire those with more accumulated human capital. Panel a shows the differences between engineers on one- versus multi-building teams. Engineers on one-building teams are those who were consistently on a co-located team in the pre-closure period. Panel b investigates the interactions between being on a one-building team and engineer tenure, age, gender. Black points reflect separate specifications with each interaction term; gray triangles reflect a joint equation with all interactions. The omitted group is more tenured, older, and/or male. All specifications include our preferred time-varying controls for engineer group, age, and tenure, as well as gender (where applicable). Standard errors are clustered by team.

Figure 9: Who Returns to the Office



Notes: This figure illustrates return-to-office (RTO) patterns by age. Panel (a) plots office attendance by engineer age at the firm. Data comes from badge swipes in the headquarters' security system, and the sample focuses on engineers based in the headquarters during both RTOs from spring of 2022 through spring of 2024. The y-axis represents the percent of weekdays people badge in. The x-axis plots age, split into fifteen quantiles. The blue series includes engineers whose teammates are all headquarters-based; the orange series includes those with some teammates who are fully remote or in satellite locations. Panels b and c use data from a 2022 and 2023 survey conducted by StackOverflow, the top online Q&A site for software engineers. There are 95,503 respondents and 22,233 from the US. Figure A.19 shows each of the top countries separately. Panel d plots data from the Census's Household pulse Surveys from 2022 and 2023 (U.S. Census Bureau, 2023), limiting to college-educated workers. The y-axis plots the share of reported in-office time.

Figure 10: Young People's Unemployment and the Rise of Remote Work



Notes: This figure analyzes changes in age-specific unemployment rates around the rise of remote work. Panel a focuses on software engineers, defined as (1) Computer Scientists and Systems Analysts, Network systems Analysts, and Web Developers (occupation 2010 code = 1000), (2) Computer Programmers (1010), or (3) Software Developers, Applications and Systems Software (1020). Panel b includes all college-educated workers (with at least a bachelor's degree). In Panel a, age groups are in quintiles; in Panel b, they are vigintiles, due to the larger sample size. Figure A.22 shows these graphs with 2020−2021 included, where levels differ but the age gradient is similar to 2022−2024. Panel c shows age differences in unemployment vis-á-vis the reference year of 2019 for teleworkable occupations (in black) and non-teleworkable occupations (in blue), where teleworkability is defined by Dingel and Neiman (2020). Estimates come from Equation 4. The shaded area covers the peak years of the pandemic, 2020−2021. All analyses limit to college-educated workers between the ages of 22 and 54 in the Current Population Survey (Bureau of Labor Statistics, 2024). Standard errors are clustered by survey respondent. *p<0.1; **p<0.05; ***p<0.01.

Table 1: Summary Statistics: One- and Multi-Building Teams

	Mean	One-Building	Multi-Building	Δ_0	
Age (Years)	28.76	28.51	29.09	-0.58	0.36
				(0.42)	(0.56)
Firm Tenure (Years)	1.36	1.21	1.56	-0.34***	-0.42***
				(0.11)	(0.12)
% Female	18.58	19.53	17.33	2.19	-2.75
				(2.78)	(3.26)
Job Traits					
Job Level	1.71	1.62	1.82	-0.20***	-0.06
				(0.06)	(0.07)
Salary + Stocks	121,262	119,075	124,161	-5,086***	-894
				(1,810)	(2,210)
Team Traits					
# Teammates	6.09	5.72	6.57	-0.85**	-0.42
				(0.42)	(0.47)
Manager Tenure	2.87	2.84	2.92	-0.08	-0.41
				(0.32)	(0.36)
Manager Job Level	3.30	3.21	3.42	-0.21**	-0.08
				(0.09)	(0.10)
Engineer Group					
Back-End	12.73	21.23	1.48	19.75***	–
				(3.46)	
Front-End	19.83	30.47	5.76	24.71***	–
				(4.61)	
Internal Tools	60.00	37.23	90.13	-52.90***	–
				(5.26)	
AI Features	7.44	11.07	2.63	8.44***	–
				(3.13)	
Engineer Group Controls					✓
# Software Engineers	1,055	637	418		
# Teams	304	206	121		

Notes: This table shows traits of the engineers, their job, and their team before the offices closed for COVID-19. The sample includes engineers whose teams are all in the main campus. "Job level" refers to the engineer's position within the firm's hierarchy from zero (an intern) to six (senior staff). Columns 4−5 present the pre-closure gap (Δ_0) between engineers on one- and multi-building teams. Column 5 includes engineering group controls: indicators for whether the engineer works on front-end website design, back-end catalog management, internal tools for others in the company, or AI features. Standard errors in parentheses are clustered by engineering team. $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$.

Table 2: Proximity to Teammates and Online Feedback

Panel A: Alternative Specifications

	Comments per Program							
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Post x One-Building	−1.29***	−1.58***	−1.52***	−1.16***	−1.35***	−1.47***	−1.73***	−1.25**
	(0.48)	(0.50)	(0.45)	(0.43)	(0.49)	(0.48)	(0.49)	(0.49)
One-Building Team	1.16**	1.45**	2.35***	1.80***	1.92***			
	(0.52)	(0.61)	(0.53)	(0.49)	(0.55)			
Post	−1.22***							
	(0.36)							
Pre-Mean, One-Building	8.04	8.04	8.04	8.04	8.04	8.04	8.04	8.04
Percentage Effects								
Post x One-Building	-16.1%	-19.7%	-18.9%	-14.4%	-16.8%	-18.3%	-21.5%	-15.6%
One-Building	14.5%	18.1%	29.2%	22.4%	23.9%			
Group x Month FE		✓	✓	✓	✓	✓	✓	✓
Program Scope Quartics			✓	✓	✓	✓	✓	✓
Tenure x Month FE				✓	✓	✓	✓	✓
Age x Month FE					✓	✓	✓	✓
Engineer FE						✓	✓	✓
Other Traits x Month FE							✓	✓
Building x Month FE								✓
% One-Building Team	58.3	58.3	58.3	58.3	58.3	58.3	58.3	58.3
# Teams	304	304	304	304	304	304	304	304
# Engineers	1,055	1,055	1,055	1,055	1,055	1,055	1,055	1,055
# Engineer-Months	9,304	9,304	9,304	9,304	9,304	9,304	9,304	9,304
R^2	0.02	0.02	0.29	0.36	0.43	0.56	0.61	0.61

Panel B: Placebo Check

	Comments per Program		
	Overall	From Teammates	From Non-Teammates
Post x One-Building Team	−1.25**	−1.33***	0.12
	(0.49)	(0.38)	(0.31)
Pre-Mean, One-Building Team	8.04	4.28	3.73
All Controls	✓	✓	✓
# Engineers	1,055	1,055	1,055

Notes: This table examines the relationship between physical proximity to teammates and the online feedback that engineers receive on their code. Each column estimates the difference-in-differences model in Equation 1, which is based on the differential loss of proximity among one-building team engineers around the office closures. Panel A tests robustness to different controls. Panel B presents a placebo check: proximity to teammates should impact feedback from teammates but not from non-teammates. Each observation is an engineer-month, with the dependent variable being the average number of comments per program. Program scope controls include quartics for the number of lines added, lines deleted, and files changed. Column 7 adds engineer traits: gender, being a person of color, home zipcode, and job level. Column 8 includes building-by-month fixed effects to allow differential changes in feedback for programmers who sat in the main and auxiliary buildings. The sample includes engineers who submit programs to the firm's main code-base and whose teams are all in the firm's main campus. Standard errors are clustered by team. *p<0.1; **p<0.05; ***p<0.01.

Table 3: Proximity and Other Dimensions of Online Feedback

a: Extensiveness and Timeliness of Online Conversations

	Total Characters	Total Words	Hours to Comment	% Other Online Convo
	(1)	(2)	(3)	(4)
Post x One-Building Team	−164.10**	−26.87**	1.12*	−1.58*
	(67.78)	(11.09)	(0.64)	(0.95)
Pre-Mean, One-Building Team	833.24	136.92	16.02	4.06
Post x One-Building Team as %	-19.7%	-19.6%	7%	-38.9%

b: Intensive and Extensive Margins

	Intensive	Extensive	
	Comments per Commenter	Commenters per Program	Distinct Commenters per Program
	(1)	(2)	(3)
Post x One-Building Team	−0.63***	−0.05	−0.12**
	(0.23)	(0.06)	(0.06)
Pre-Mean, One-Building	4.36	1.77	1.37
Post x One-Building as %	-14.3%	-2.6%	-9%

c: Back-and-Forth Conversations

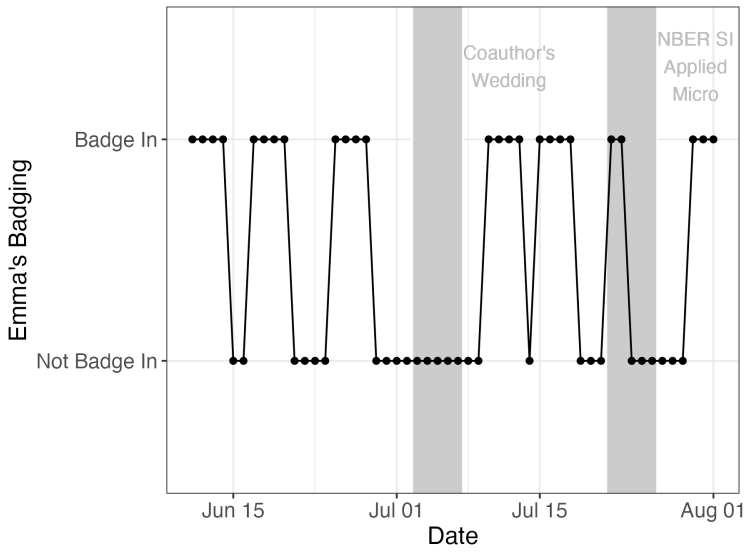
		Commenter's	Program Writer's		Commenter's
	Back and Forths	Initial Comments	Replies	Questions	Follow-up Comments
Post x One-Building Team	−0.39***	−0.71**	−0.72**	−0.15***	−0.76**
	(0.13)	(0.28)	(0.29)	(0.06)	(0.37)
Pre-Mean, One-Building	1.95	4.91	2.14	0.24	3.13
Post x One-Building as %	-19.9%	-14.4%	-33.7%	-62.7%	-24.3%

Notes: This table considers alternative metrics of online feedback: (a) the extent and timeliness of feedback, (b) the extensive and intensive margins of feedback, and (c) the back-and-forth conversation between the commenter and program writer. In panel a, the frequency of other online conversations is measured as the percent of reviews with references to Slack discussions. In panel b, if a programmer wrote five programs in a month and had the same person comment on all of them, she would have one commenter per program but only 0.2 distinct commenters per program. Distinct comments per program aims to approximate the size of the engineer's network. In panel c, the total number of back and forths in Column 1 measures each time there is a author switch in the comments: for example, a commenter giving a set of comments, an author asking a set of questions, and then a commenter clarifying their initial feedback would count as three. Each specification replicates Column 5 of Table 2. *p<0.1; **p<0.05; ***p<0.01.

A Online Appendix

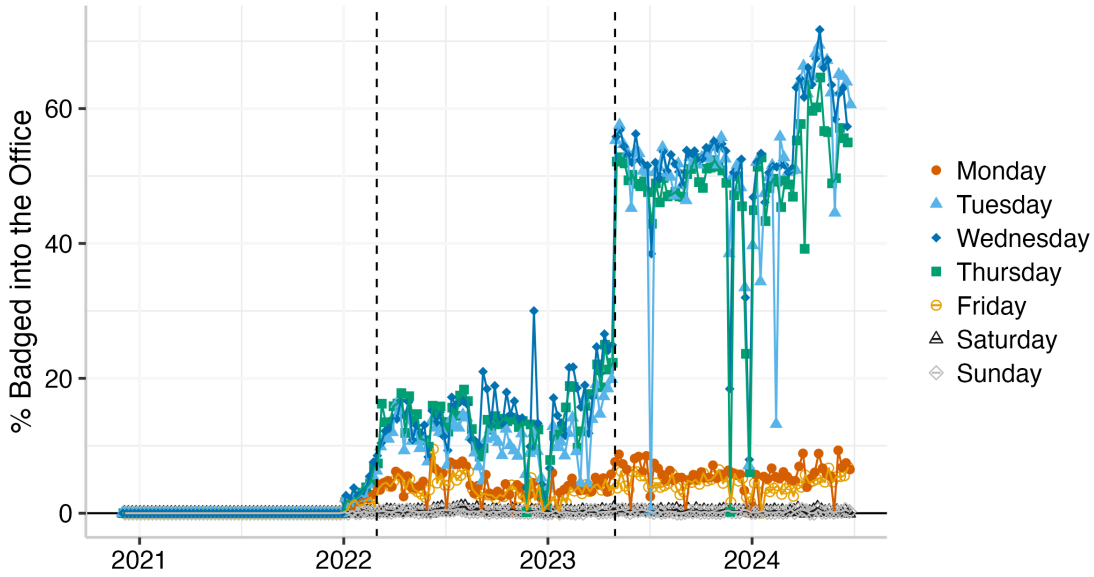
I.A Figures

Figure A.1: Badge Data of One of the Coauthors



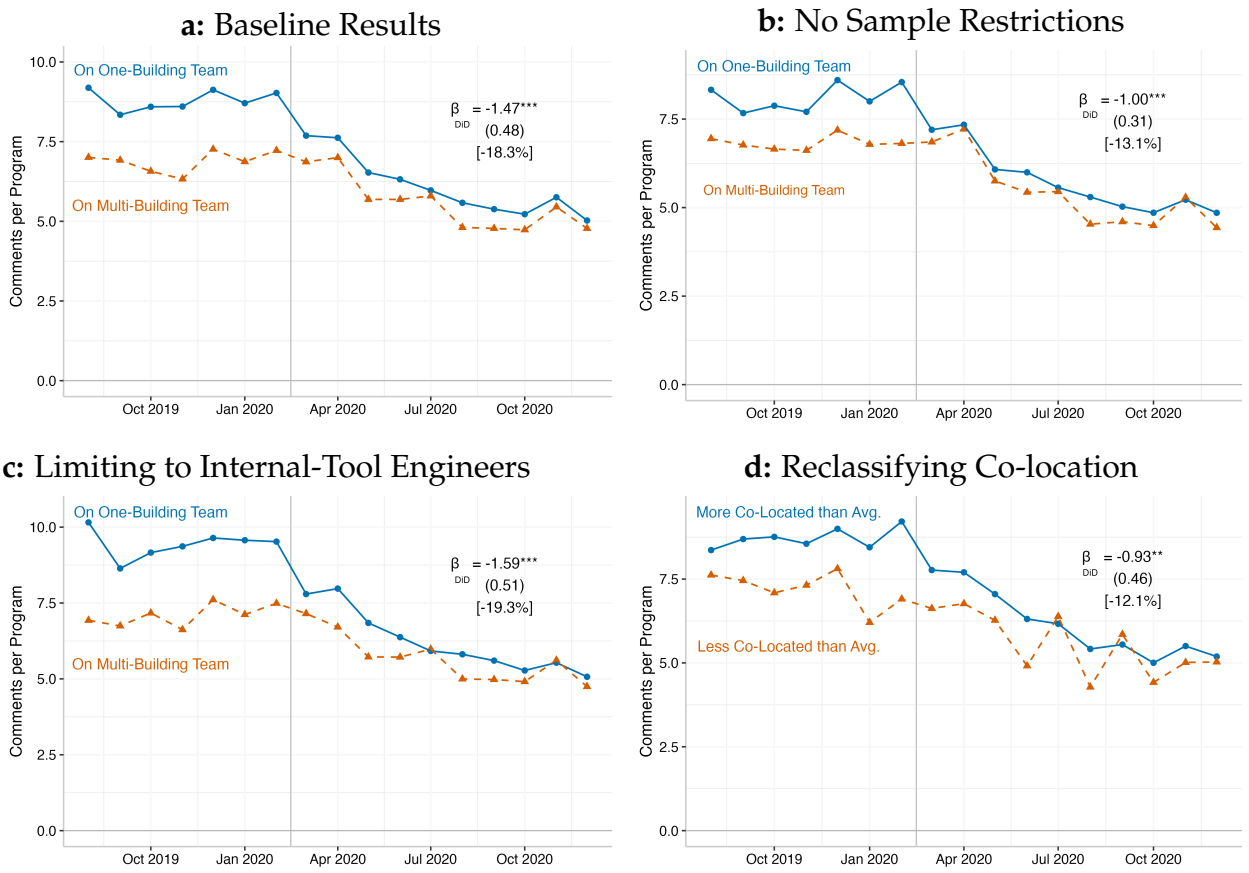
Notes: This figure shows the badge data of one of the coauthors who was embedded at the retailer for a summer.

Figure A.2: Badge Data of Engineers by Day



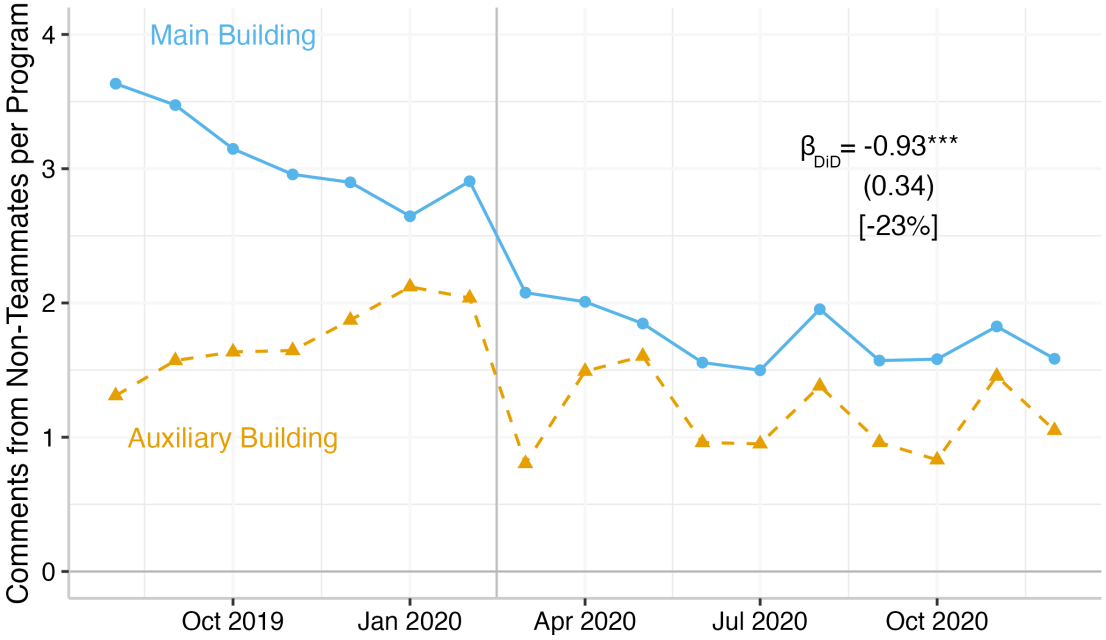
Notes: This figure illustrates the share of engineers who badged into the office each day for those working in the main campus. The vertical lines highlight the firm's two return-to-office mandates: the first required two days per week, while the second required three.

Figure A.3: Robustness of Proximity to Teammates and Online Feedback



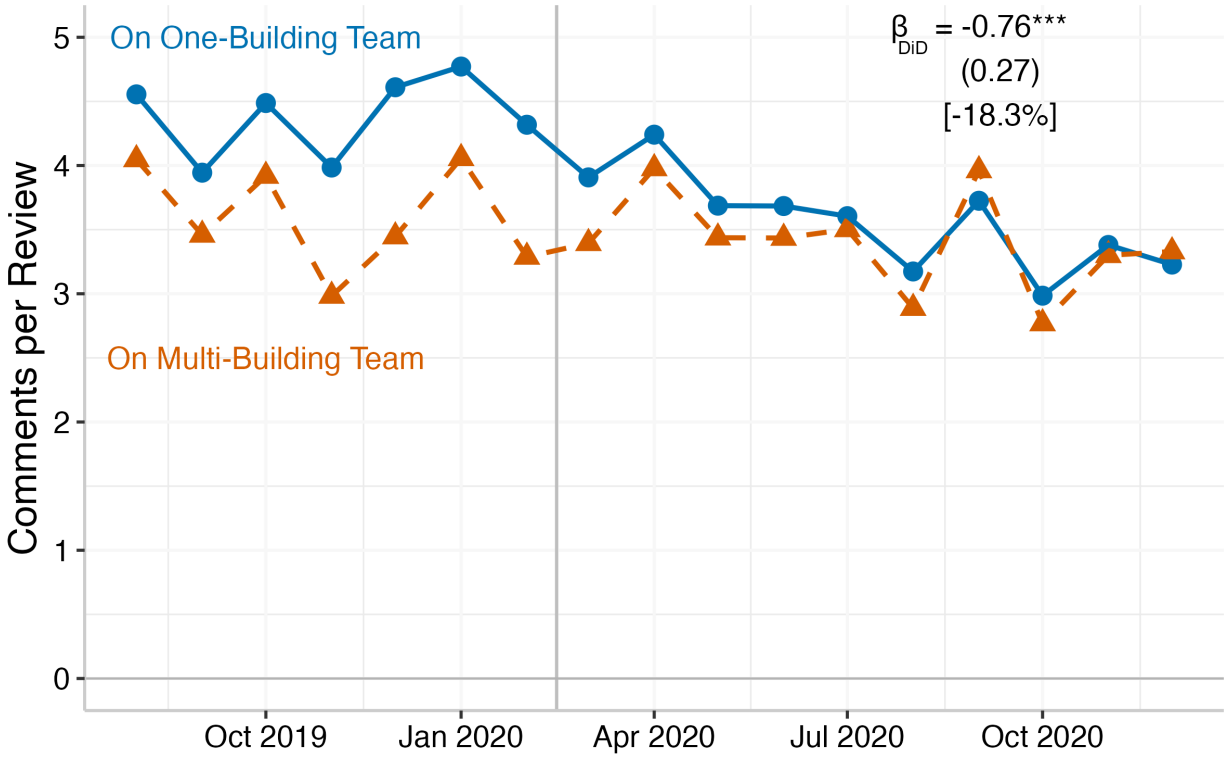
Notes: This figure probes robustness of Figure 1a to alternative ways of implementing the design. Panel a repeats the baseline design for reference. Panel b does not apply any sample restrictions: this panel includes engineers who are outside the main campus and engineers under high-level managers, who may manage multiple teams. Panel c limits to internal-tool engineers who are more likely to be on distributed teams to sit near those who use their tools. Panel d differentiates teams that are more versus less co-located than average (87%) rather than fully co-located or not. Each plot residualizes by our preferred month-specific controls for engineer tenure, age, and engineering group, as well as quartics for program length. Standard errors are clustered by team. $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$.

Figure A.4: Proximity to Non-Teammates & Feedback



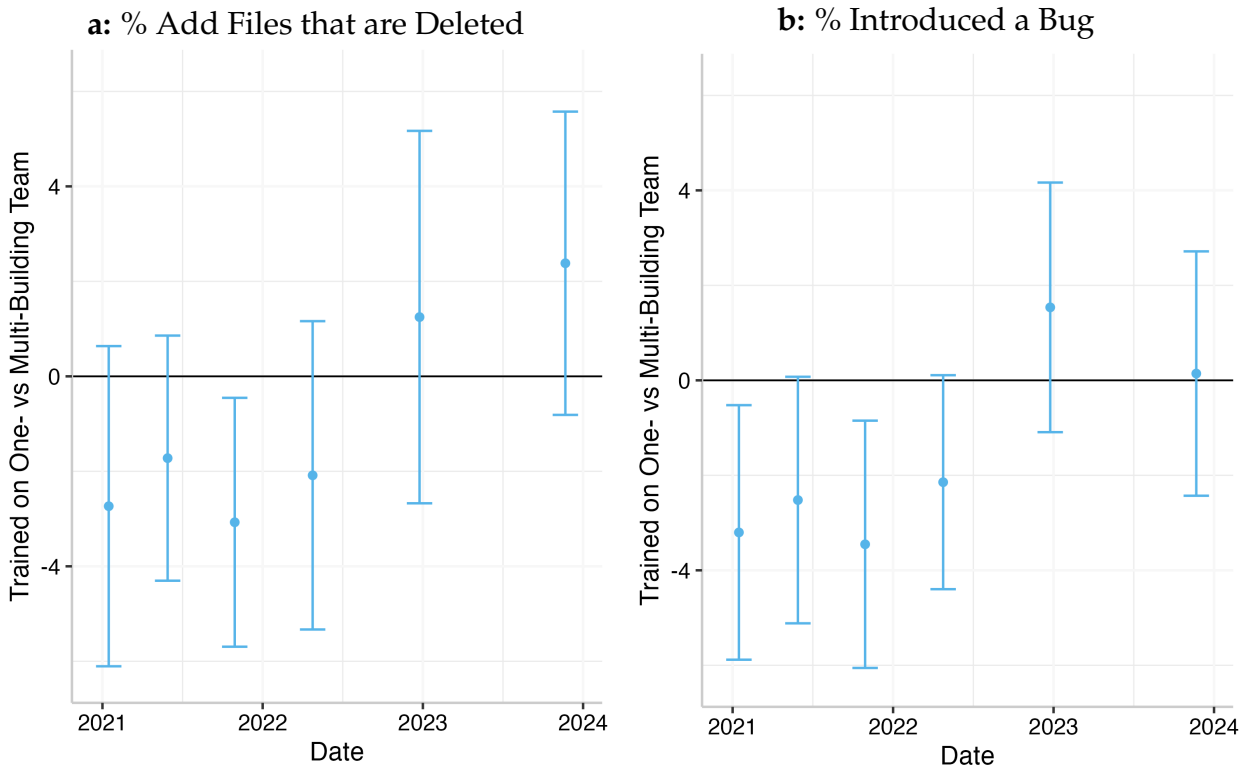
Notes: This figure analyzes whether engineers assigned to the primary building on the main campus — who are proximate to more engineers outside their teams — also receive more feedback from non-teammates. The blue series shows those in the main building with 70% of the engineers, and the orange series shows those in the auxiliary building with 30%. The y-axis plots the comments that engineers receive per program from non-teammates, residualized by our preferred controls for month-specific controls for tenure, age, and engineering group as well as quartics for program length. The annotated coefficient also includes engineer fixed effects. Standard errors are clustered by team. *p<0.1; **p<0.05; ***p<0.01.

Figure A.5: Externalities from Distant Teammates on Interactions between Co-located Teammates



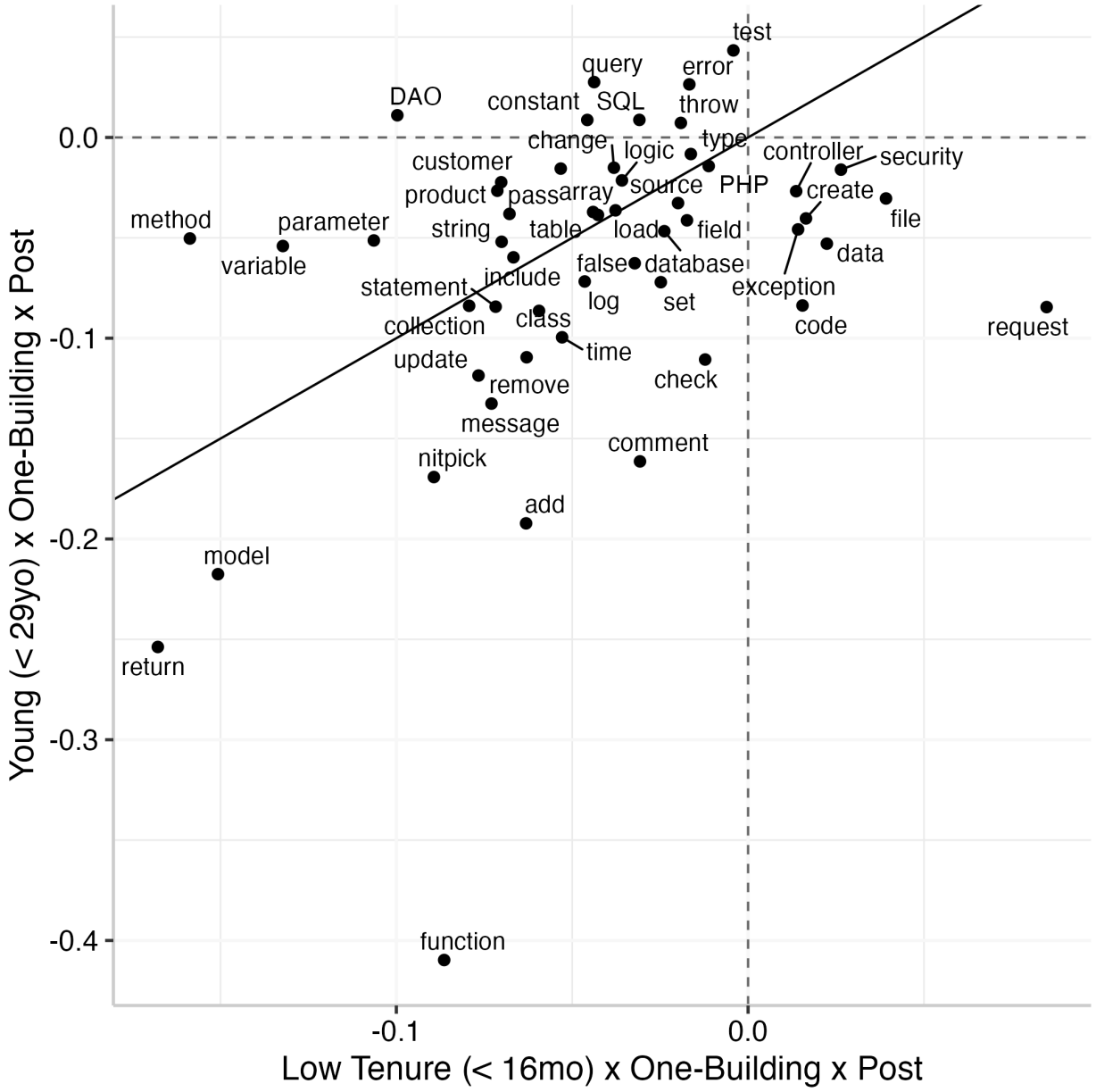
Notes: This figure investigates how the interactions between co-located teammates differ depending on whether the rest of the teammates are also co-located. The figure replicates the analysis in Figure 1a, but the dependent variable is the average number of comments exchanged when co-located teammates review each other's work. The annotated coefficient shows the difference-in-differences coefficient with the percentage effect in brackets. Standard errors are clustered by team. $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$.

Figure A.6: Dynamics of Code Quality for those Trained on One- Versus Multi-Building Teams



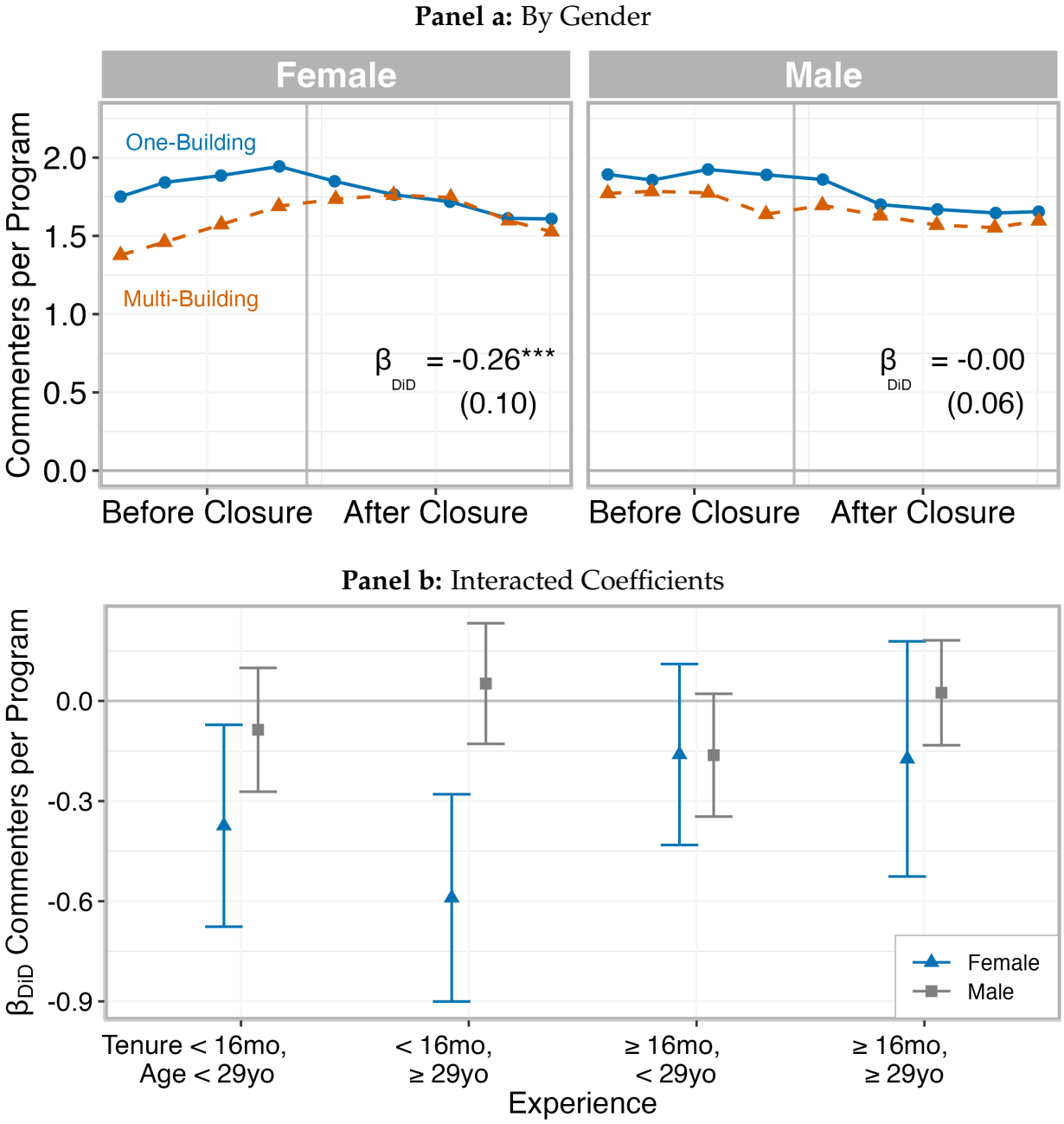
Notes: This figure complements Figure 3 by showing the full time-series of the code quality differences between engineers trained on one-building teams versus those trained on multi-building ones. The period starts in December 2020 when quality metrics started to be recorded. The x-axis is grouped into six quantiles, which are imbalanced because there is attrition out of the sample. Panel a investigates the percent of programs where the engineer adds a file that later gets deleted. Files may be deleted because the code was fully rewritten or because the firm decided the feature was a dead-end. Panel b investigates introducing a bug, as defined by all the changes that the engineer made getting reverted by a subsequent program. The coefficients compare the two sets of engineers, with controls for engineer age, tenure, and engineering group. Standard errors are clustered by team.

Figure A.7: Differential Effect of Losing Proximity on the Words Received by Young and Junior Engineers



Notes: This figure illustrates the differential effects of losing proximity to teammates on the words received by younger workers (the x-axis) and newer hires (the y-axis). The specification considers each interaction term separately as in Column 1 of Table A.6. The diagonal line is the 45°line. The top 50 most frequent words are displayed, as measured before the closures for one-building teams.

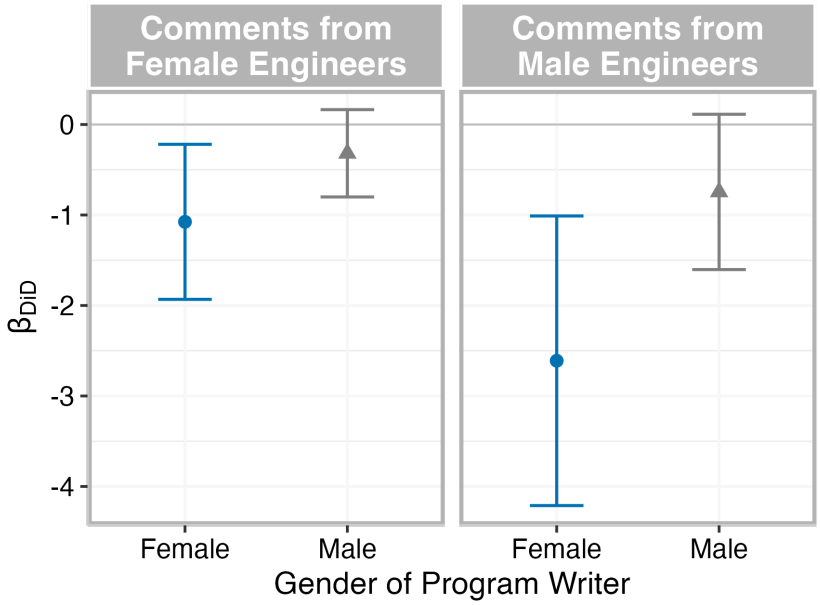
Figure A.8: Gendered Effects of Proximity on Number of Commenters



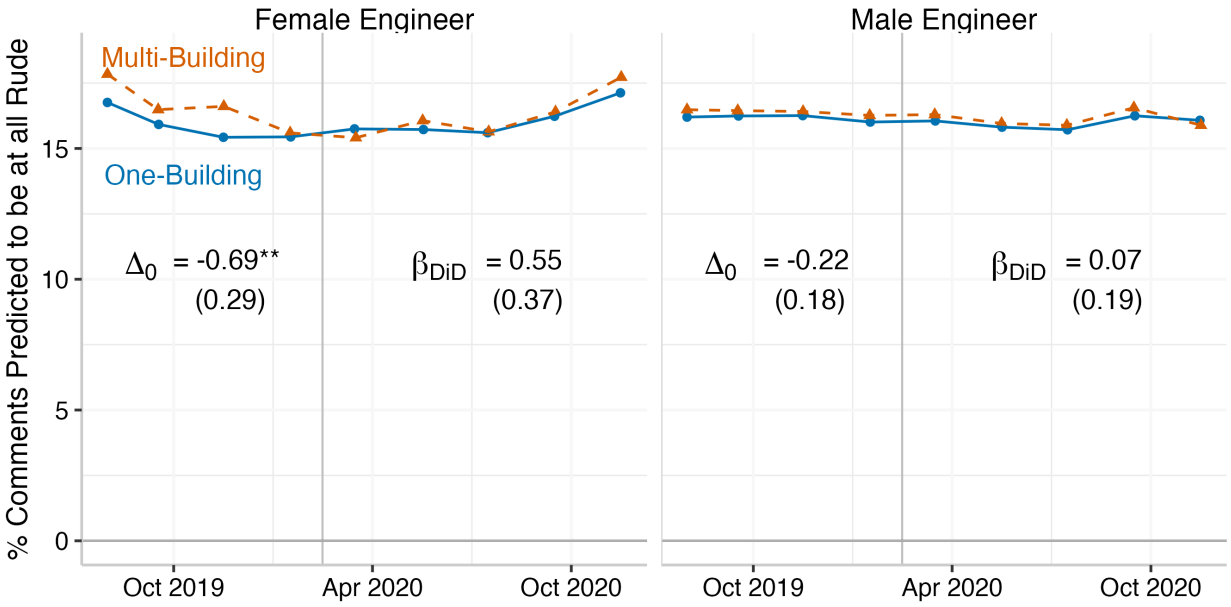
Notes: This figure explores gendered effects of proximity on the total number of people who comment on programmers' work. Panel a replicates Figure 4c with the outcome of number of commenters rather than comments per program. Panel b is the parallel analogue of 4d. All DiD specifications use our preferred controls for month-specific effects of engineering group, engineer age, and engineer tenure. Standard errors are clustered by engineering team. *p<0.1; **p<0.05; ***p<0.01.

Figure A.9: Mansplaining?

Panel a: By Commenter Gender

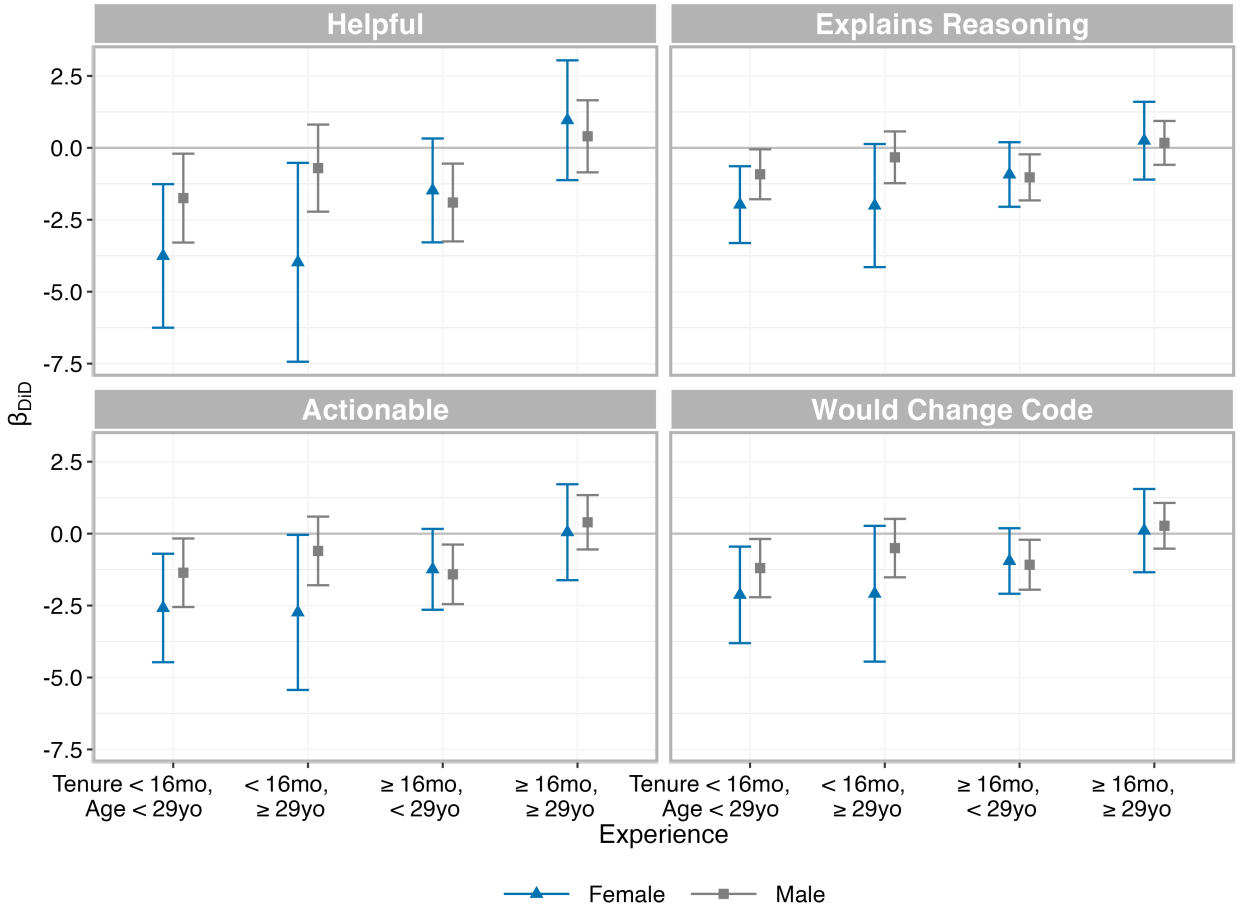


Panel b: Rudeness of Feedback



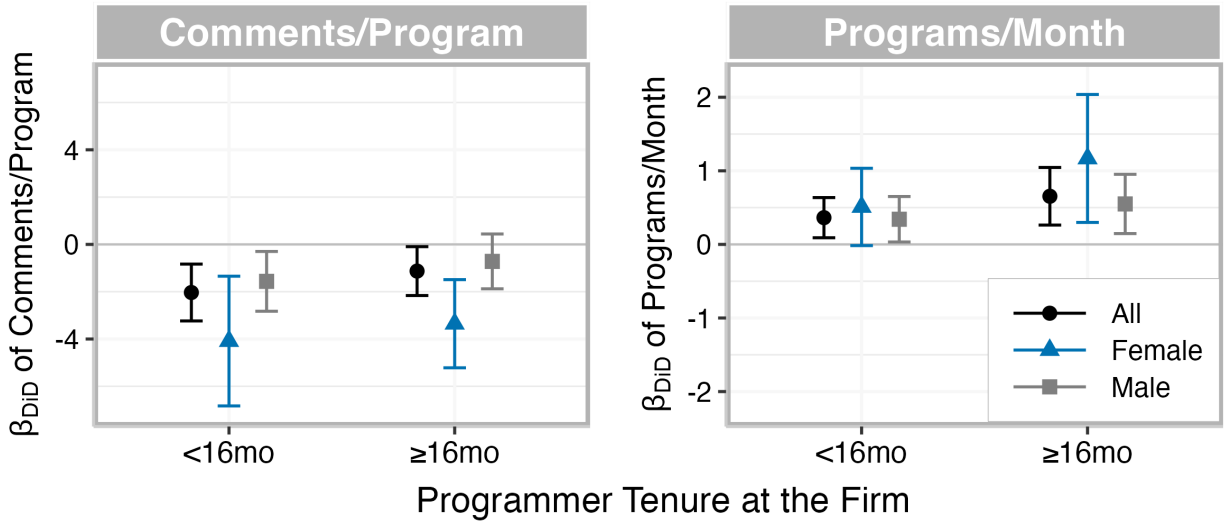
Notes: This figure investigates whether the gendered effect of proximity could be due to "mansplaining." Panel a shows our DiD estimates for comments from both female commenters (on the left) and male commenters (on the right). Panel b investigates comments that are predicted to be very, moderately, or a little bit rude. To form these predictions, we use the ratings of independent evaluators to train a supervised machine learning algorithm (see Appendix I.C for details). In these plots, the annotated coefficient on the left focuses on the pre-period difference and those on the right report the DiD estimates. Throughout, the specifications control for program scope and month-specific effects of engineer gender, age, tenure, and engineering group. Standard errors are clustered by team. $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$.

Figure A.10: Heterogeneity in Proximity's Effects on High-Quality Comments



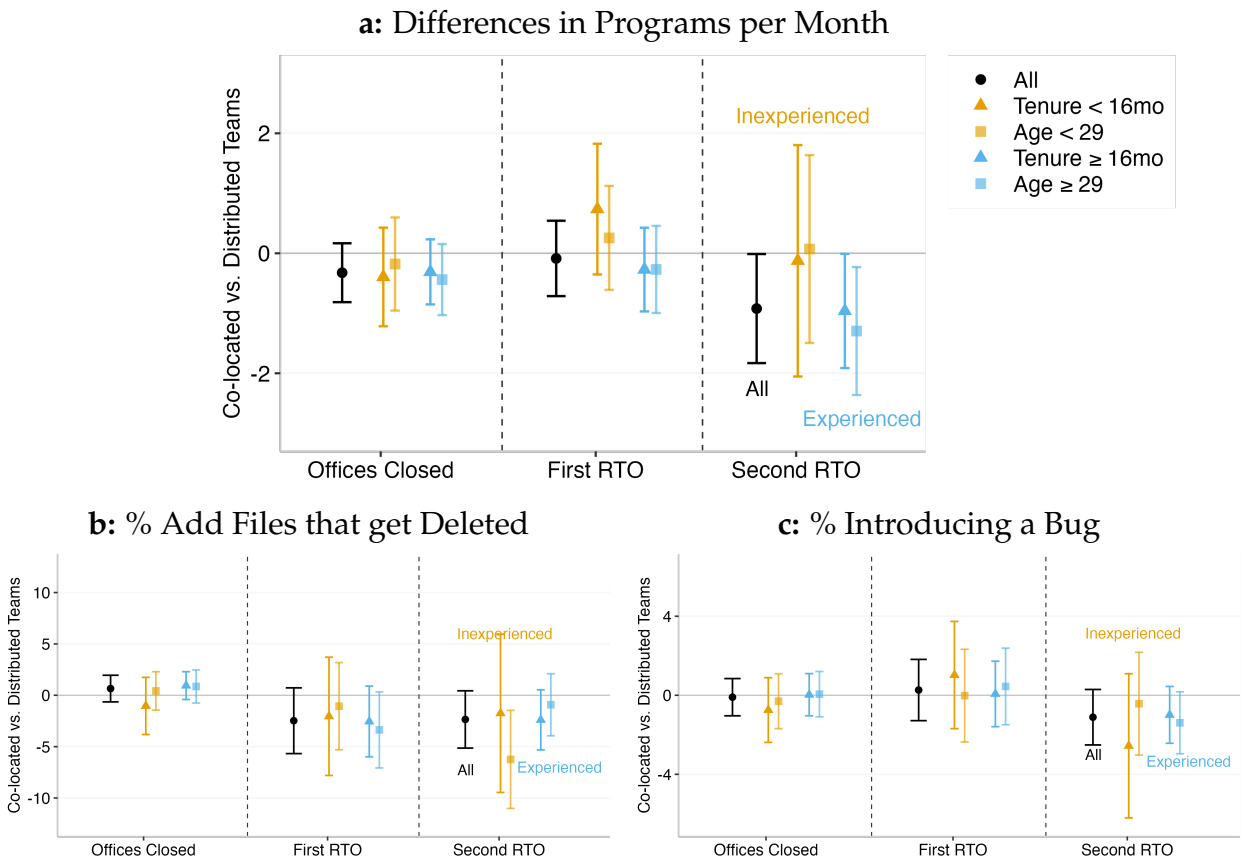
Notes: This figure investigates the heterogeneous effects of proximity on the total number of comments per program that are predicted to be helpful, well-reasoned, actionable, and impactful for the code. To generate these predictions, we employed independent raters to evaluate a random sample of comments. We then used a supervised machine learning algorithm to scale up to the whole dataset (see Appendix I.C for details). For each outcome, the coefficients come from a single difference-in-differences (DiD) design where the estimated effect of losing proximity is allowed to depend on engineer gender, age, and tenure. This is an interacted version of Equation 1, with controls for program scope and month-specific effects of engineer gender, age, tenure, and engineering group. Standard errors are clustered by team. $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$.

Figure A.11: Tradeoffs by Programmer Tenure and Gender



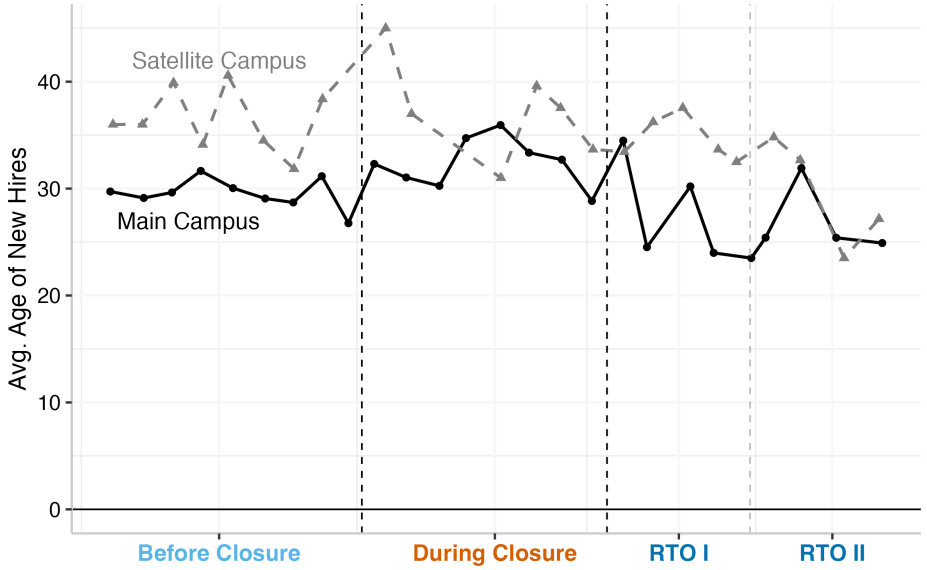
Notes: This figure investigates the heterogeneous effects of proximity on comments received and programs written by gender. Each coefficient comes from the difference-in-differences (DiD) design in Equation 1, with controls for program scope and month-specific effects of engineer gender, age, tenure, and engineering group. Standard errors are clustered by team. *p<0.1; **p<0.05; ***p<0.01.

Figure A.12: RTOs with the Full Set of Controls



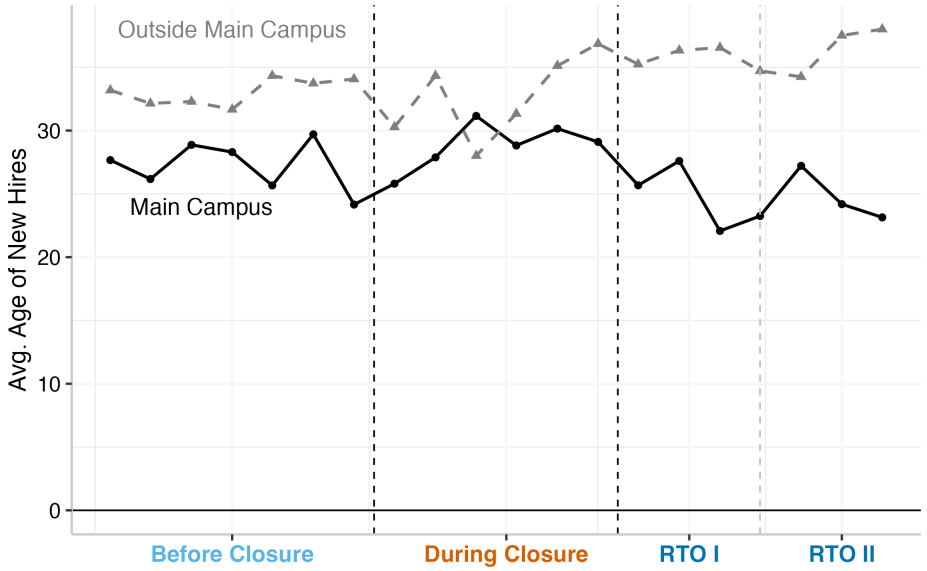
Notes: This figure replicates Figure 6 but includes the full set of controls. In addition to our preferred month-specific controls for engineer age, tenure, and engineering group, this adds month-specific controls for gender, race, home zipcode, and job level. Experience is measured both by tenure at the firm (using a 16-month cutoff) and age (using a 29 year-old cutoff). Error bars represent 95% confidence intervals, with clustering by team.

Figure A.13: Hiring Patterns in Main vs. Satellite Campuses



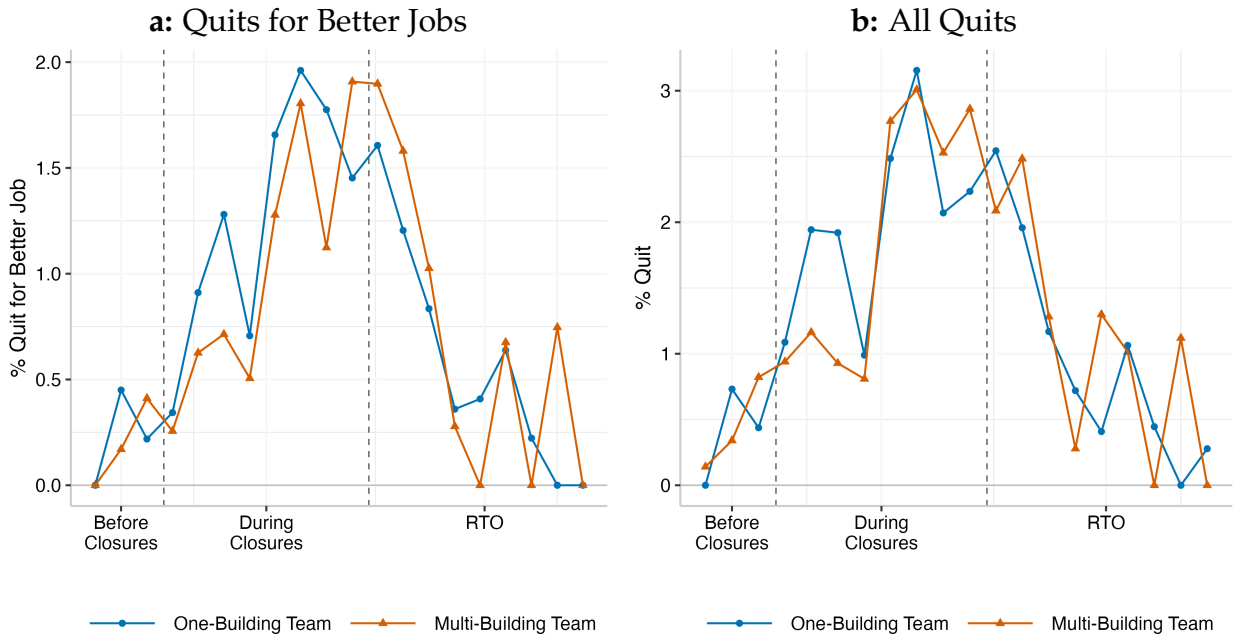
Notes: This figure replicates Figure 7c but excludes fully remote hires. Satellite-campus engineers were much more likely to be on distributed teams with some members who were in the headquarters. At the end of the period, the firm was trying to change by actively engineering "atomic" teams that could operate independently in each of the campuses.

Figure A.14: Hiring Patterns in the Rest of the Firm's Corporate Roles



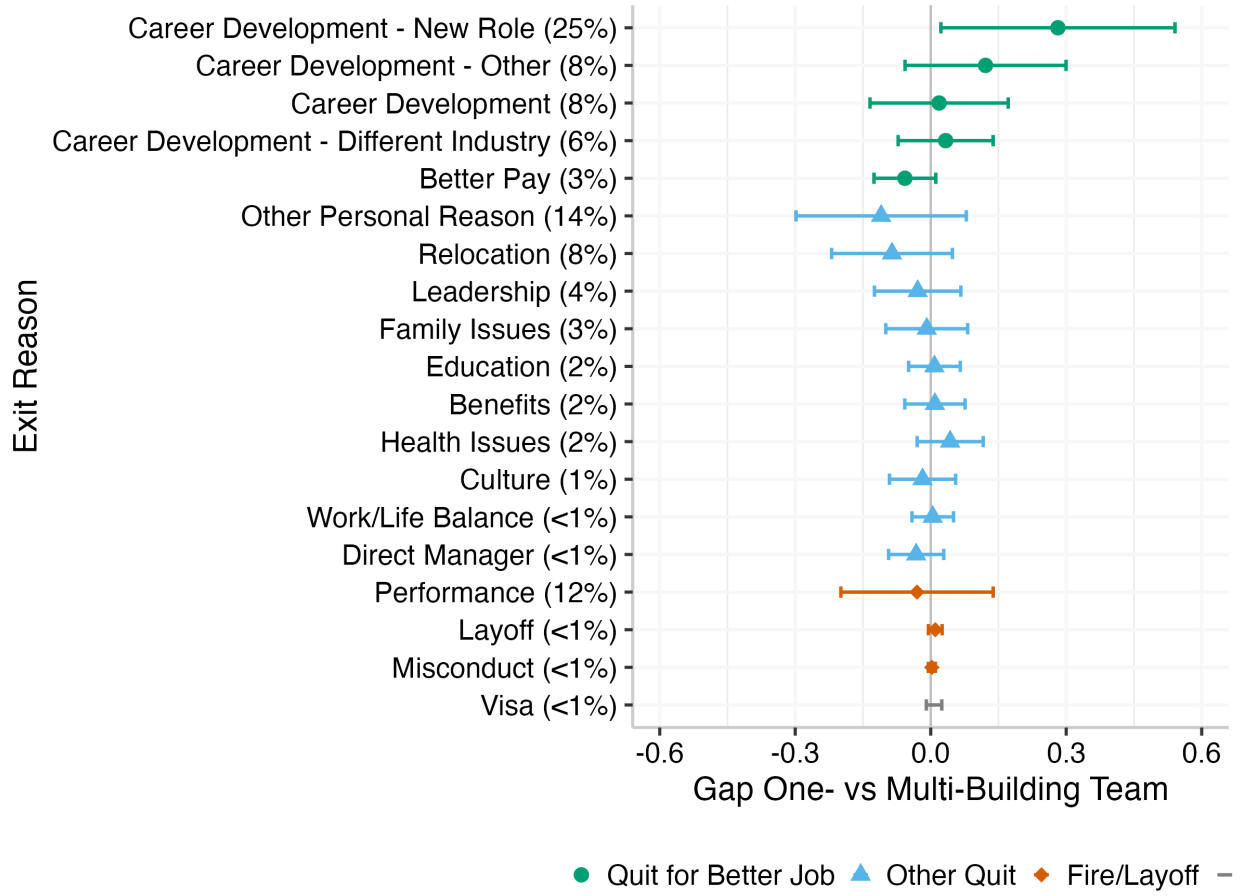
Notes: This figure replicates Figure 7c but focuses on the rest of the firm's corporate population excluding software engineering roles.

Figure A.15: Poaching from the Firm



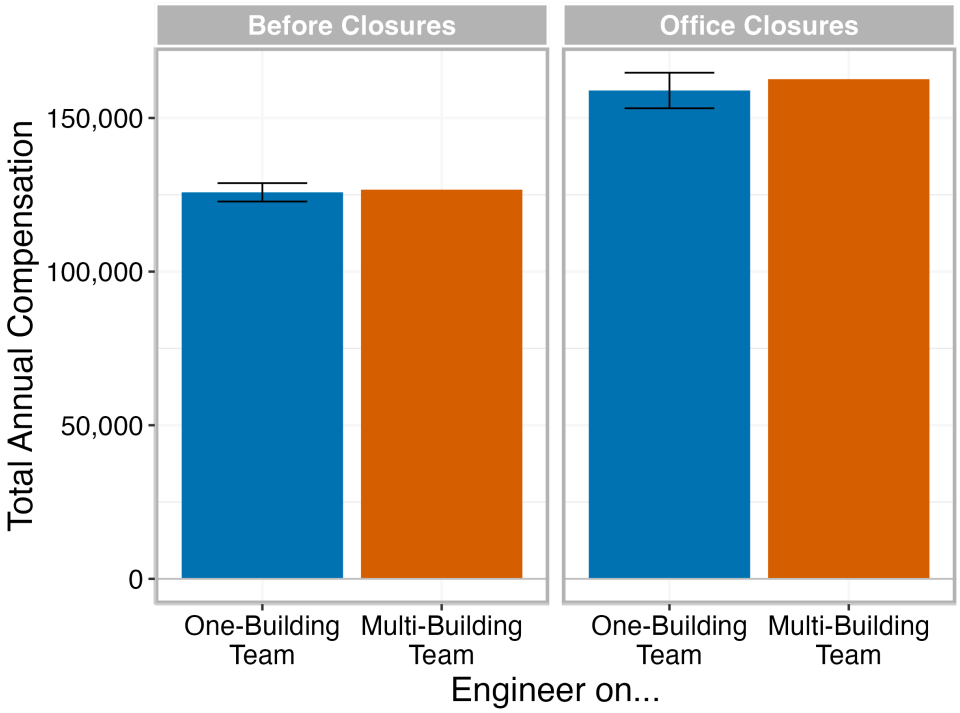
Notes: This figure shows the time series of quits from the firm by engineers on one- and multi-building teams. Panel a focuses on quits for better pay or career development elsewhere. Panel b focuses on all quits.

Figure A.16: Types of Exits for One- versus Multi-Building Teams during the Office Closures



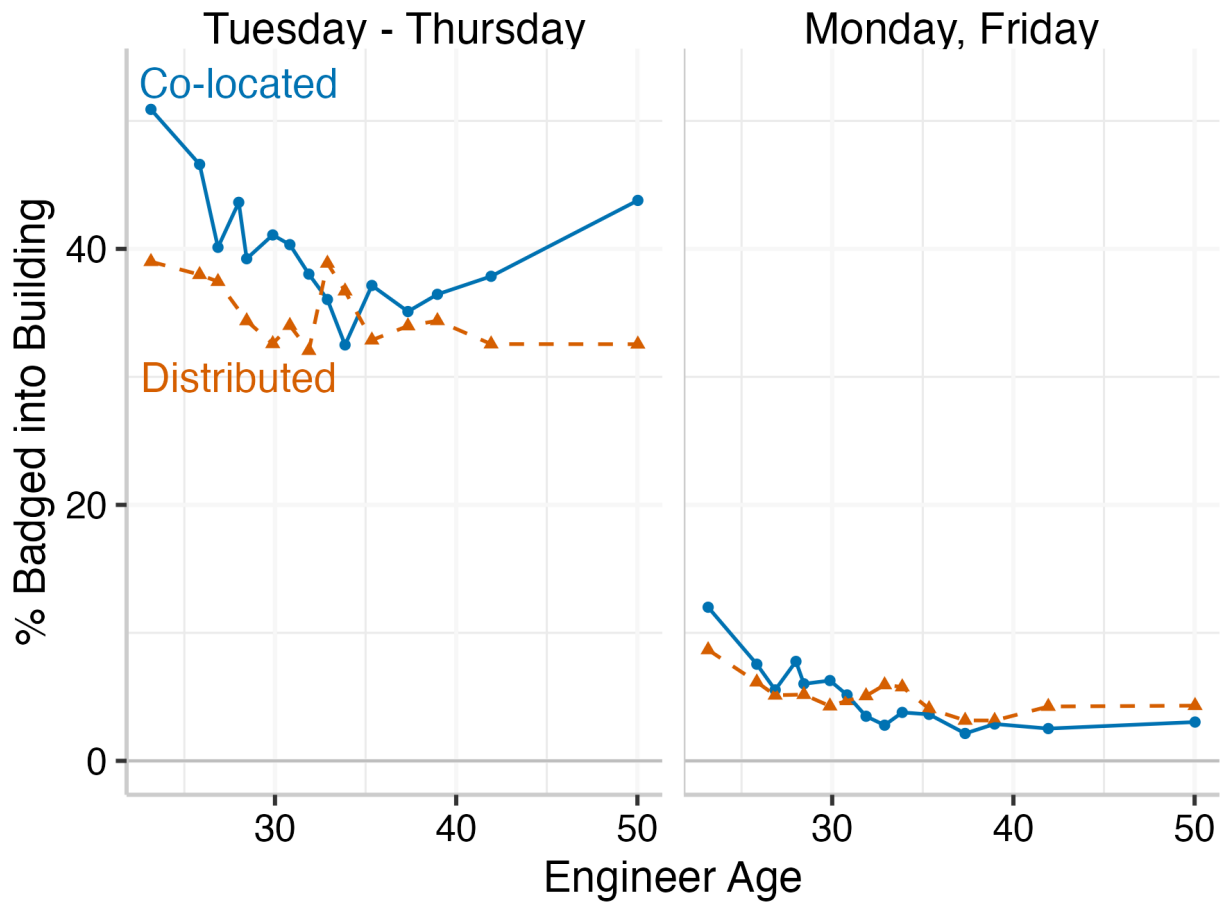
Notes: This figure unpacks Figure 8 by differentiating between all the possible exit reasons. The y-axis plots specific categories for leaving that people give in exit interviews. The firm groups the many reasons that people give into these categories. The label reports the firm-defined category followed by the percentage of exits that fall into that category. The x-axis plots the differences in that type of exit between engineers on one- and multi-building teams during the office closures. All specifications include our preferred time-varying controls for engineer group, age, and tenure. Standard errors are clustered by team and error bars show 95% confidence intervals.

Figure A.17: Proximity and Pay Around Office Closures



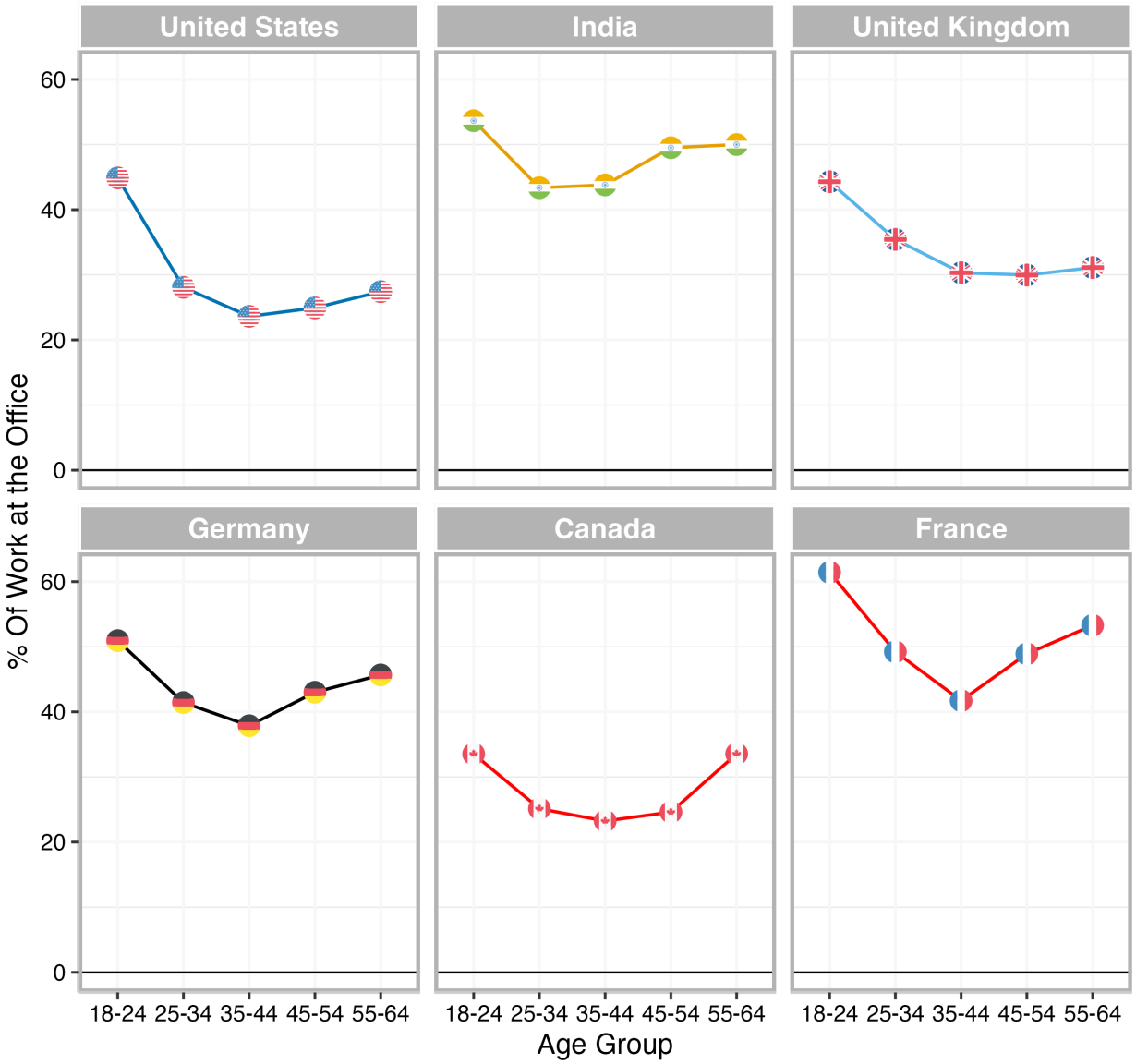
Notes: This figure illustrates differences in total compensation between one- and multi-building teams around the office closures. Each panel show differences between engineers on one- and multi-building teams with controls for engineer age, tenure, engineering group, and engineer fixed effects. Error bars represent 95% confidence intervals, with clustering by team.

Figure A.18: Engineer Age and Office Attendance: By Team Co-Location & Day of Week



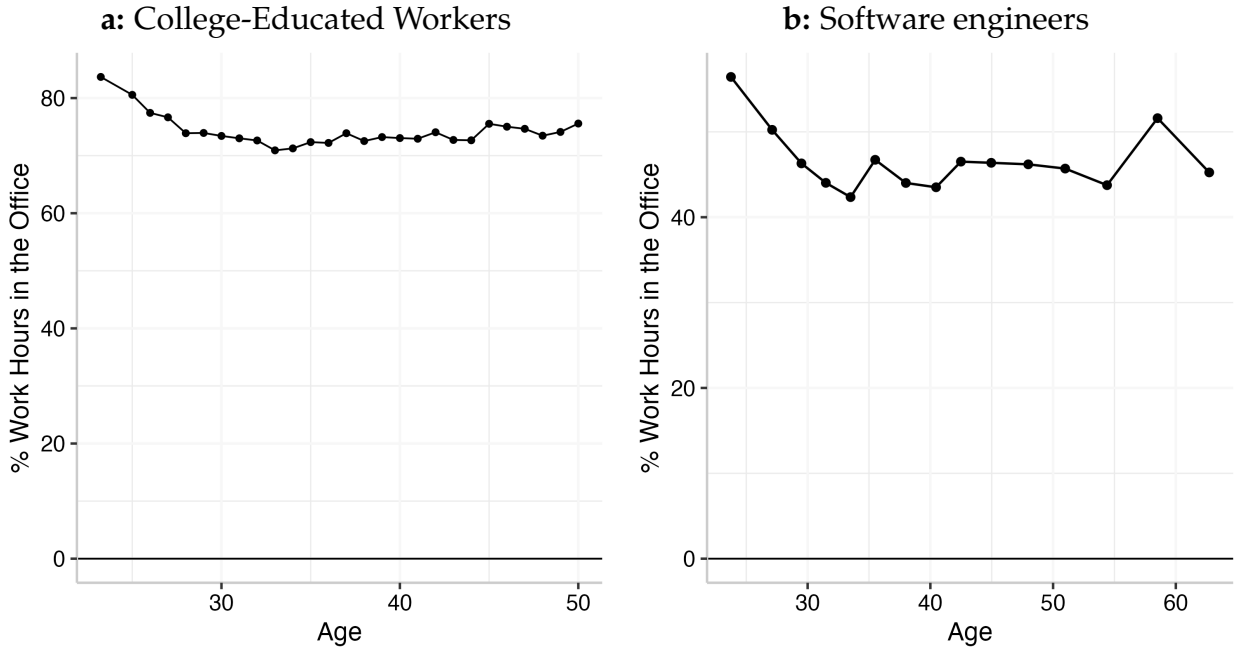
Notes: This figure illustrates variation in office attendance by engineer age depending on the opportunities to be with teammates. As in Figure 9a, engineers are grouped by whether their team is co-located in headquarters (solid blue) or geographically distributed (dashed orange). This figure also differentiates between the firm's core in-office days of Tuesday through Thursday — when engineers might expect their coworkers to be present — and non-core days of Monday and Friday — when engineers could not expect their coworkers to be in the office.

Figure A.19: Software Engineers' Return to the Office by Age & Country



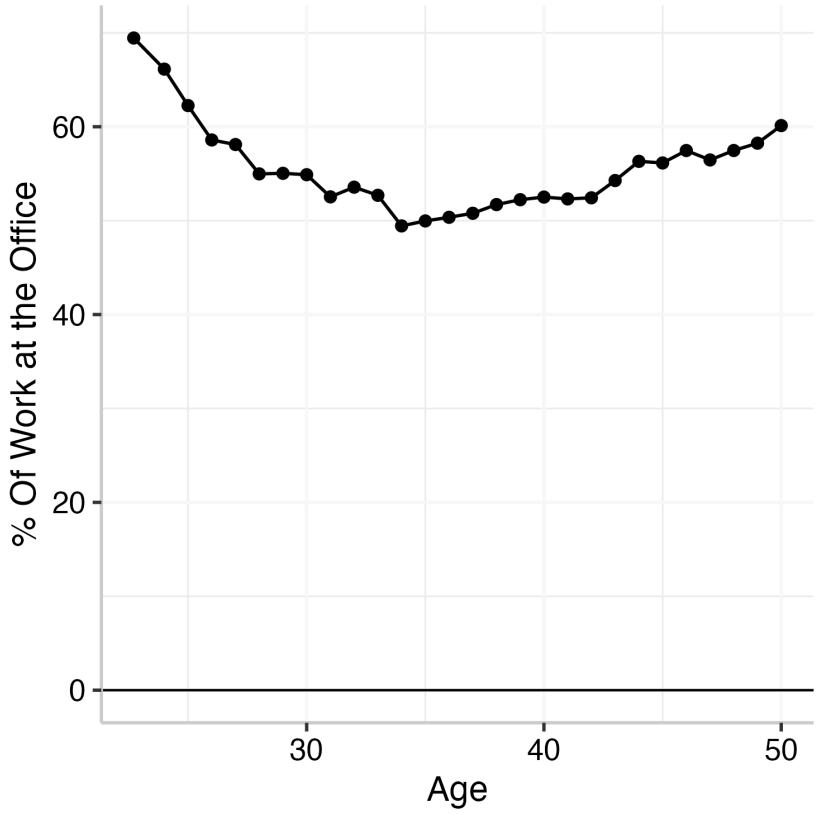
Notes: This figure illustrates return-to-office patterns by age and country among software engineers. Data come from 2022 and 2023 surveys conducted by StackOverflow, the top online Q&A site for software engineers. There are 95,503 respondents and 22,233 from the US.

Figure A.20: Return to the Office by Age in the Current Population Survey



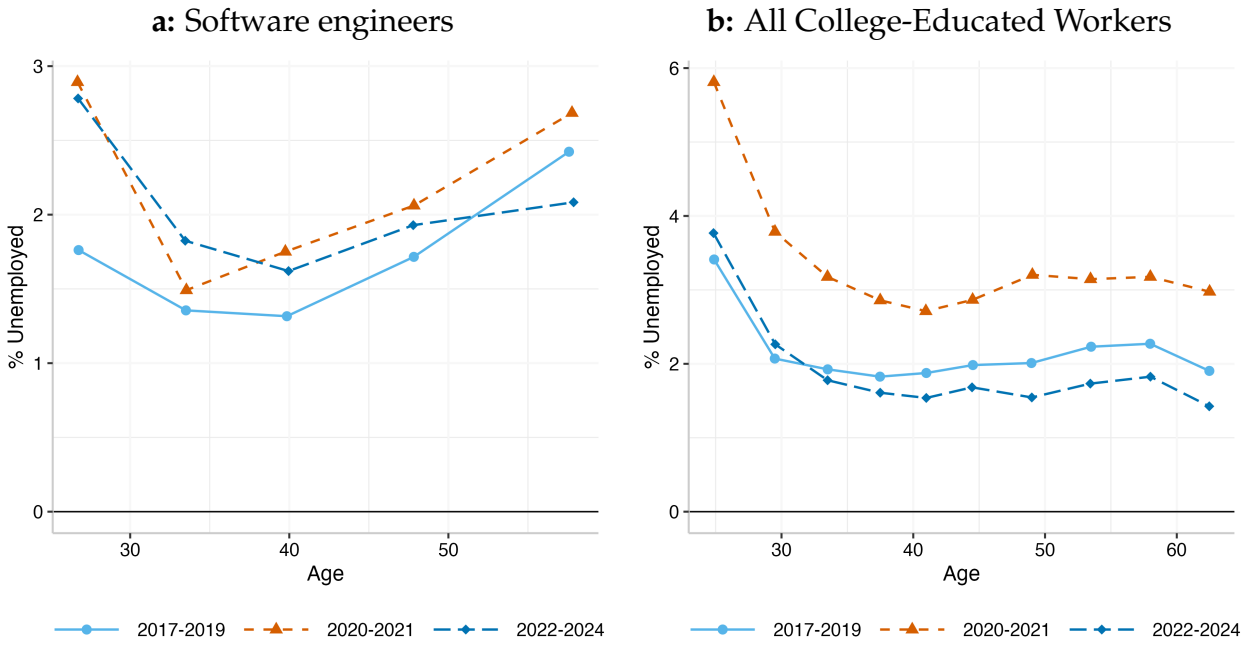
Notes: This figure illustrates return-to-office patterns by age among (a) all college-educated workers with at least a bachelor's degree and (b) software engineers. Data come from 2022, 2023, and 2024 of the Current Population Survey (Bureau of Labor Statistics, 2024). The question of interest asks employed respondents what share of their paid work hours were done from home. We define software engineers as (1) Computer Scientists and Systems Analysts, Network systems Analysts, and Web Developers (occupation 2010 code = 1000), (2) Computer Programmers (1010), or (3) Software Developers, Applications and Systems Software (1020).

Figure A.21: Return to the Office by Age Excluding Parents



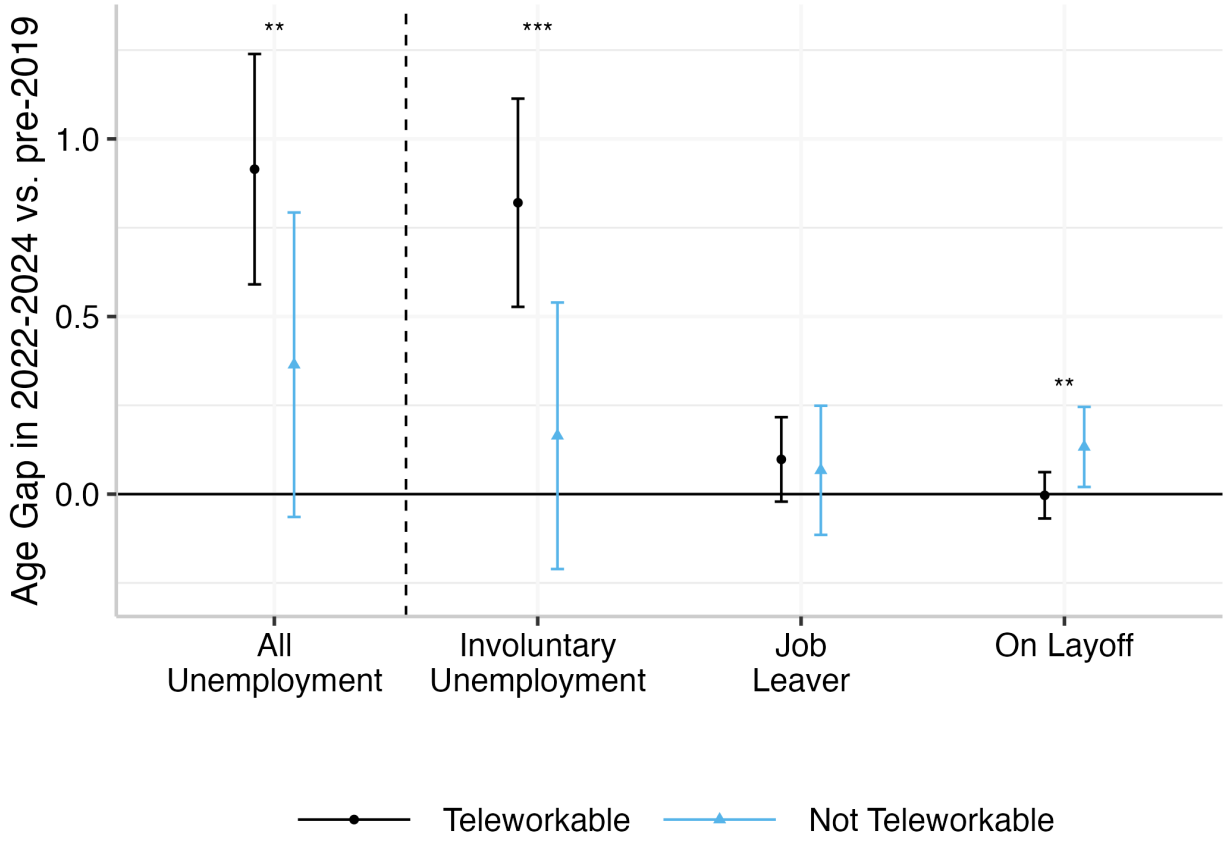
Notes: This figure replicates Figure 9d but excludes parents from the sample.

Figure A.22: Young People's Unemployment and the Rise of Remote Work



Notes: This figure replicates Figure 10a-b but does not exclude data from 2020−2021.

Figure A.23: Relative Increases in Young People's Unemployment by Unemployment Reason



Notes: This figure illustrates the changes in unemployment rates for young people (age < 29 years old) versus older people between 2022−2024 versus 2015−2019, controlling for year by occupation fixed effects and age by occupation fixed effects, separately for remotable and non-remotable occupations (as in Equation 4). The stars denote the significant of the difference-in-differences in the change in the age gap in unemployment between remotable and non-remotable occupations. Standard errors are clustered by respondent. *p<0.1; **p<0.05; ***p<0.01.

I.B Tables

Table A.1: Summary Statistics: One-Building versus Fully-Distributed Teams Before the Office Closure

	Mean	One-Building	Fully-Distributed	Δ_0	
Age (Years)	28.81	28.54	29.32	-0.77*	-0.77*
				(0.42)	(0.40)
Firm Tenure (Years)	17.95	14.54	24.28	-9.74***	-9.51***
				(1.49)	(1.53)
% Female	19.55	19.53	19.58	-0.06	-0.14
				(3.08)	(3.33)
Job Traits					
Job Level	1.76	1.62	2.00	-0.38***	-0.38***
				(0.08)	(0.07)
Salary + Stocks	121,947	119,075	128,429	-9,354***	-9,688***
				(2,151)	(2,147)
Team Traits					
# Teammates	6.53	5.72	8.05	-2.33***	-2.17***
				(0.45)	(0.41)
Manager Tenure	1131.58	1035.76	1330.80	-295.04**	-287.86**
				(116.28)	(112.91)
Manager Job Level	3.29	3.21	3.42	-0.21**	-0.21**
				(0.09)	(0.09)
Engineer Group					
Back-End	15.88	21.23	5.94	15.29***	–
				(4.37)	
Front-End	39.93	30.47	57.52	-27.04***	–
				(7.79)	
Internal Tools	34.43	37.23	29.22	8.01	–
				(7.34)	
AI Features	9.76	11.07	7.32	3.75	–
				(3.25)	
Engineer Group Controls					✓
# Software Engineers	931	637	294		
# Teams	292	206	108		

Notes: This table shows traits of the engineers, their job, and their team before the offices closed. The sample includes engineers in the main campus whose teammates were either all in one-building or distributed across campuses. "Job level" refers to the engineer's position within the firm's hierarchy from zero (an intern) to six (senior staff). Columns 4 and 5 compare the attributes of one-building and fully distributed teams. Column 4 does not include controls, while Column 5 includes engineer group fixed effects (for back-end, front-end, internal tools, and AI). Standard errors in parentheses are clustered by engineering team. $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$.

Table A.2: Summary Statistics: One-Building versus Fully-Distributed Teams around the Office Re-Openings

	Mean	One-Building	Fully-Distributed	Δ_0	
Age (Years)	31.49	30.13	31.82	-1.69***	-1.50***
				(0.23)	(0.25)
Firm Tenure (Years)	3.37	2.69	3.53	-0.84***	-0.60***
				(0.09)	(0.08)
% Female	20.10	22.54	19.51	3.03*	2.42
				(1.72)	(1.77)
Job Traits					
Job Level	2.56	2.27	2.63	-0.36***	-0.30***
				(0.04)	(0.04)
Salary + Stocks	181,466	166,102	185,181	-19,079***	-16,408***
				(2,343)	(2,376)
Team Traits					
# Teammates	5.33	3.72	5.72	-2.00***	-1.94***
				(0.20)	(0.23)
Manager Tenure	3.73	3.86	3.70	0.17	0.40***
				(0.15)	(0.15)
Manager Job Level	3.95	3.71	4.00	-0.29***	-0.27***
				(0.04)	(0.04)
Engineer Group					
Back-End	18.44	27.69	16.21	11.48***	–
				(3.11)	
Front-End	19.13	20.59	18.78	1.82	–
				(2.79)	
Internal Tools	34.66	22.15	37.69	-15.53***	–
				(3.28)	
AI Features	22.14	24.10	21.66	2.44	–
				(2.83)	
Engineer Group Controls					✓
# Software Engineers	2,380	1,298	2,105		
# Teams	915	414	802		

Notes: This table shows traits of the engineers, their job, and their team around the return-to-office mandates. The sample includes engineers in the main campus. "Job level" refers to the engineer's position within the firm's hierarchy from zero (an intern) to six (senior staff). Columns 4 and 5 compare the attributes of one-building and fully distributed teams. Column 4 does not include controls, while Column 5 includes engineer group fixed effects (for back-end, front-end, internal tools, and AI). Standard errors in parentheses are clustered by engineering team. $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$.

Table A.3: Proximity & Feedback Identified from Team-Type Switchers

	Comments per Program			
	(1)	(2)	(3)	(4)
One-Building$_{it}$ Before Closure	0.96**		0.85	
	(0.45)		(0.54)	
One-Building$_{it}$ After Closure	0.09		−0.29	
	(0.36)		(0.52)	
One-Building$_{it}$ x Post		−0.87*		−1.14**
		(0.51)		(0.56)
Pre-Mean, One-Building	8.27	8.27	8.27	8.27
Percentage Effects:				
One-Building$_{it}$ Before Closure	11.6%	11.6%	10.2%	10.2%
One-Building$_{it}$ After Closure	1.1%		-3.5%	
One-Building$_{it}$ x Post		-10.5%		-13.8%
Preferred Controls	✓	✓	✓	✓
Engineer FE	✓	✓	✓	✓
Engineer x Manager FE			✓	✓
# Pre-Closure Switcher Engineers	102	102	102	102
# All Engineers	1,055	1,055	1,055	1,055
# Pre-Closure Switcher Teams	51	51	51	51
# All Teams	304	304	304	304
# Engineer-Months	9,304	9,304	9,304	9,304

Notes: This table uses a complementary design to evaluate how proximity to teammates affects feedback, using engineers who switch between team-types. Each column includes engineer fixed effects, as well as our preferred set of time-varying controls for engineer group, tenure, and age and program scope quartics (as in Column 6 of Table 2). In this table, we define One-Building Team$_{it}$ at the monthly level for each engineer, which allows us to identify the relationship between being on a One-Building Team$_{it}$ and the feedback the engineer receives with engineer fixed effects, both before and after the office closures (in Column 1) as well as the difference in these coefficients (in Column 2). Columns 3−4 include engineer by manager fixed effects to compare the same engineer under the same manager as a function of whether the team is all co-located. Standard errors are clustered by team. *p<0.1; **p<0.05; ***p<0.01.

Table A.4: Proximity and Predicted High-Quality Comments

Panel a: Total Substantive Feedback

	Comments per Program Predicted to be...			
	Helpful	Explain Reasoning	Actionable	Would Change Code
	(1)	(2)	(3)	(4)
One-Building x Post	−1.45***	−0.70***	−1.08***	−0.86***
	(0.46)	(0.27)	(0.36)	(0.31)
Pre-Mean, One-Building	6.37	3.35	4.67	3.91
Post x One-Building as %	-22.8%	-21%	-23.1%	-22%

Panel b: Percent of Substantive Feedback

	% Comments Predicted to be...			
	Helpful	Explain Reasoning	Actionable	Would Change Code
	(1)	(2)	(3)	(4)
One-Building x Post	−2.86**	−1.75*	−1.66	−1.86*
	(1.38)	(1.04)	(1.21)	(1.03)
Pre-Mean, One-Building	50.79	33.12	39.29	33.94
Post x One-Building as %	-5.6%	-5.3%	-4.2%	-5.5%
# Teams	304	304	304	304
# Engineers	1,055	1,055	1,055	1,055
# Engineer-Months	9,304	9,304	9,304	9,304

Notes: This table evaluates how proximity to teammates relates to the predicted quality of comments. Each specification replicates Column 6 of Table 2. Panel a focuses on the total number of comments per program predicted to be (1) helpful, (2) well-reasoned, (3) actionable, and (4) likely to cause the programmer to change the code. These patterns are also illustrated in Figure 2b. Panel b focuses on the share of comments that fall into these categories. To form these predictions, we employ external software engineers to rates 5,337 comments on each of these dimensions. We then use a supervised machine learning algorithm to scale up from this sample to generate predictions for the entire dataset. Appendix I.C provides more details. Standard errors are clustered by team. *p<0.1; **p<0.05; ***p<0.01.

Table A.5: Downstream Differences in Code Quality

	% Added a File that gets Deleted		% Introduced a Bug	
Was on a One Building Team	-2.37^{**}	-2.90^{**}	-3.09^{***}	-1.83
	(0.95)	(1.37)	(0.95)	(1.20)
Dependent Mean	5.32	5.32	14.45	14.45
Percentage Effects				
Team in One Building	-16.4%	-20.1%	-58%	-34.4%
Current Team FE		✓		✓
# Teams	285	285	285	285
# Engineers	828	828	828	828
# Engineer-Months	10,258	10,258	10,258	10,258

Notes: This table examines long-run differences in code quality, using two different quality metrics. The period covers December 2020 (when these metrics started to be recorded) until the office re-opening in 2022. The first two columns focus on the percent of programs where the engineer adds a file that later gets deleted. Files may be deleted because the code was fully rewritten or because the firm decided the feature was a dead-end. The last two columns focus on introducing a bug, as defined by all the changes that an engineer made getting reverted by a subsequent program. Every specification includes our preferred controls for engineer age, tenure, and engineering group, with each interacted with the month. The even columns also include fixed effects for the engineer's current team. Standard errors are clustered by team. $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$.

Table A.6: Robustness of Heterogeneity by Age & Gender

	Comments per Program					
	(1)	(2)	(3)	(4)	(5)	(6)
Young (<29yo) x Post x One-Building	−2.09***	−1.84**	−1.72**			
	(0.69)	(0.71)	(0.71)			
Female x Post x One-Building			−1.41*	−2.70***	−2.41**	−2.33**
			(0.72)	(0.94)	(0.94)	(0.96)
Pre-Mean, One-Building	8.04	8.04	8.04	8.04	8.04	8.04
One Building x Pre-Post Closure x...						
Months at Firm Indicators		✓	✓		✓	✓
Age in Years Indicators						✓
% One-Building Team	58.3	58.3	58.3	58.3	58.3	58.3
# Teams	304	304	304	304	304	304
# Engineers	1,055	1,055	1,055	1,055	1,055	1,055
# Engineer-Months	9,304	9,304	9,304	9,304	9,304	9,304

Notes: This table probes the robustness of heterogeneity in proximity's effects on feedback by age and gender to the inclusion of other interaction terms. Columns 1 and 4 show the baseline heterogeneity by (1) age and (4) gender for reference. These designs mirror those in Panels b and c of Figure 4. Columns 2 and 5 include detailed interactions of engineer tenure (with different indicators for every month of tenure) with being on a one-building team and the pre- and post-closure period. These specifications investigate whether there remains significant differentials by engineer age or gender when accounting for tenure. Column 6 adds analogous interactions for engineer age (with different indicators for every year). All specifications include our preferred controls for program scope, engineer fixed effects, and month-specific controls for engineer age, tenure, and engineering group. Standard errors are clustered by engineering team. *p<0.1; **p<0.05; ***p<0.01.

Table A.7: Proximity to Teammates and Engineer Output

	Monthly Contributions to the Main Codebase					
	Programs		Lines Added		Files Changed	
	(1)	(2)	(3)	(4)	(5)	(6)
Post x One-Building	0.39***		101.00***		1.75*	
	(0.10)		(36.76)		(0.94)	
Senior ($\geq$ 16mo) x Post x One-Building		0.58***		133.90**		3.50**
		(0.18)		(58.74)		(1.53)
Junior ($<$ 16mo) x Post x One-Building		0.30***		84.64**		0.88
		(0.11)		(41.95)		(1.10)
Pre-Mean, One-Building Team	1.71	1.71	320.52	320.52	9.64	9.64
# Engineers	1,055	1,055	1,055	1,055	1,055	1,055
# Engineer-Months	16,058	16,058	16,058	16,058	16,058	16,058

Notes: This table investigates the relationship between sitting near teammates and monthly output of (Columns 1−2) programs submitted to the main code-base, (Columns 3−4) lines of code added, and (Columns 5−6) files changed. The odd columns estimate the aggregate effects, while the even columns differentiate by engineer tenure (split by the median tenure). Each specification estimates Equation 1, with our preferred controls of engineer fixed effects and month-specific controls for engineer age, tenure, and engineering group. The sample includes engineers who ever submitted a program to the firm's main code-base and whose teammates are all in the firm's main campus. *p<0.1; **p<0.05; ***p<0.01.

Table A.8: Robustness of Proximity to Teammates and Engineer Output

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
	Panel (a): Programs per Month						
Post x One-Building Team	0.47***	0.45***	0.48***	0.48***	0.40***	0.37***	0.32***
	(0.11)	(0.13)	(0.13)	(0.13)	(0.11)	(0.11)	(0.12)
One-Building Team	−0.48***	−0.33**	−0.37**	−0.36**			
	(0.16)	(0.16)	(0.16)	(0.16)			
Pre-Mean, One-Building Team	1.71	1.71	1.71	1.71	1.71	1.71	1.71
Post x One-Building Team as %	27.5%	26.3%	27.7%	27.9%	23.4%	21.4%	18.9%
One-Building Team as %	-28.1%	-19.1%	-21.8%	-21.1%			
R^2	0.01	0.07	0.15	0.18	0.52	0.55	0.55
	Panel (b): Lines Added per Month						
Post x One-Building Team	105***	92**	108***	106**	102***	102***	123***
	(36)	(39)	(42)	(41)	(37)	(36)	(39)
One-Building Team	−193***	−158***	−186***	−181***			
	(43)	(44)	(46)	(45)			
Pre-Mean, One-Building Team	321	321	321	321	321	321	321
Post x One-Building Team as %	32.9%	28.6%	33.7%	33.2%	31.7%	31.8%	38.5%
One-Building Team as %	-60.2%	-49.3%	-58%	-56.6%			
R^2	0.02	0.04	0.12	0.14	0.41	0.44	0.44
	Panel (c): Files Changed per Month						
Post x One-Building Team	1.93**	1.66	2.06*	2.01*	1.80*	1.79*	2.12**
	(0.97)	(1.06)	(1.11)	(1.09)	(0.94)	(0.97)	(1.08)
One-Building Team	−3.97***	−3.62***	−4.24***	−4.12***			
	(1.12)	(1.15)	(1.18)	(1.16)			
Pre-Mean, One-Building Team	9.64	9.64	9.64	9.64	9.64	9.64	9.64
Post x One-Building Team as %	20%	17.2%	21.3%	20.9%	18.7%	18.6%	21.9%
One-Building Team as %	-41.2%	-37.6%	-44%	-42.7%			
R^2	0.01	0.03	0.10	0.13	0.38	0.41	0.41
Engineer Group x Month FE		✓	✓	✓	✓	✓	✓
Months at Firm x Month FE			✓	✓	✓	✓	✓
Age x Month FE				✓	✓	✓	✓
Engineer FE					✓	✓	✓
Engineer Traits x Month FE						✓	✓
Main Building x Month FE							✓
# Teams	304	304	304	304	304	304	304
# Engineers	1,055	1,055	1,055	1,055	1,055	1,055	1,055
# Engineer-Months	9,304	9,304	9,304	9,304	9,304	9,304	9,304

Notes: This table investigates the relationship between sitting near teammates and monthly output of (a) programs submitted to the main code-base, (b) lines of code added, and (c) files changed. Each specification estimates Equation 1, with controls defined in Table 2. The sample includes engineers who ever submitted a program to the firm's main code-base and whose teammates are all in the firm's main campus. *p<0.1; **p<0.05; ***p<0.01.

Table A.9: RTO Difference-in-Differences

	Panel (a): Programs per month					
2nd RTO x Co-Located Team	−0.60	−1.06**				
	(0.52)	(0.51)				
Tenure < 16mo: 2nd RTO x Co-Located Team			0.04	−1.28		
			(0.95)	(1.02)		
Tenure ≥ 16mo: 2nd RTO x Co-Located Team			−0.62	−0.88		
			(0.55)	(0.54)		
Age < 29: 2nd RTO x Co-Located Team					0.69	−0.11
					(0.80)	(0.77)
Age ≥ 29: 2nd RTO x Co-Located Team					−1.02*	−1.35**
					(0.60)	(0.63)
Dependent Mean	7.83	7.83	7.83	7.83	7.83	7.83
	Panel (b): % Add file that gets deleted					
2nd RTO x Co-Located Team	−2.99**	−0.72				
	(1.23)	(1.70)				
Tenure < 16mo: 2nd RTO x Co-Located Team			−4.61	−3.30		
			(3.53)	(3.72)		
Tenure ≥ 16mo: 2nd RTO x Co-Located Team			−3.06**	−0.51		
			(1.28)	(1.84)		
Age < 29: 2nd RTO x Co-Located Team					−5.10**	−4.36*
					(2.15)	(2.45)
Age ≥ 29: 2nd RTO x Co-Located Team					−2.36*	0.95
					(1.33)	(1.97)
Dependent Mean	13.17	13.17	13.17	13.17	13.17	13.17
	Panel (c): % Introduce a Bug					
2nd RTO x Co-Located Team	−1.21*	−1.59*				
	(0.73)	(0.81)				
Tenure < 16mo: 2nd RTO x Co-Located Team			−2.00	−3.52**		
			(1.30)	(1.38)		
Tenure ≥ 16mo: 2nd RTO x Co-Located Team			−1.23	−1.32		
			(0.78)	(0.90)		
Age < 29: 2nd RTO x Co-Located Team					−0.76	−0.57
					(1.27)	(1.20)
Age ≥ 29: 2nd RTO x Co-Located Team					−1.48	−2.14*
					(0.91)	(1.12)
Dependent Mean	3.46	3.46	3.46	3.46	3.46	3.46
DiD versus Office Closure	✓		✓		✓	
DiD versus 1st RTO		✓		✓		✓
# Teams	995	995	995	995	995	995

Notes: This table investigates the difference-in-differences between co-located teams and distributed teams in the second RTO versus in either the office closure period (odd columns) or in the first RTO (even columns). This table tests whether the differences in Figure 6 are significantly different from each other. Each specification uses the controls defined in Figure 6. The sample includes engineers who are in the firm's main campus. *p<0.1; **p<0.05; ***p<0.01.

Table A.10: RTO and Alternative Measures of Coding Quantity

	Monthly Programming Output			
	Programs	Added Lines	Changed Files	Main Code-Base Programs
Second RTO x Co-located Team	−0.91**	−133.60	−2.58	−0.17
	(0.45)	(101.10)	(2.44)	(0.14)
First RTO x Co-located Team	0.15	17.72	0.26	−0.04
	(0.28)	(70.69)	(1.56)	(0.06)
Fully Remote x Co-located Team	−0.31	51.60	1.59	−0.03
	(0.23)	(35.08)	(0.99)	(0.06)
Dependent Mean	7.83	7.83	711.41	711.41
Percentage Effect				
Second RTO x Co-located Team	-11.67%	-18.78%	-10.99%	-25.16%
	(5.81)	(14.21)	(10.38)	(20.85)
# Teams	995	995	995	995
Observations	47,806	47,806	47,806	47,806
R^2	0.56	0.43	0.54	0.48

Notes: This table investigates the relationship between sitting near teammates and different dimensions of programming output. Column 1 show number of programs pr month (as in Figure 6b). Column 2 shows lines of code added; Column 3, files changed; and Column 7, programs per month in the main code-base. Each specification estimates Equation 2, with engineer fixed effects and our preferred month-specific controls for engineer age, tenure, and engineering group. The sample includes engineers who ever submitted a program and who are themselves in the firm's main campus. *p<0.1; **p<0.05; ***p<0.01.

Table A.11: Office Attendance by Age with Controls

	Badged into Building		
Young (<29) x Co-Located Team	4.09***	4.87***	4.43**
	(1.51)	(1.51)	(1.81)
Young (<29)	4.71***	3.72***	2.09***
	(0.59)	(0.61)	(0.78)
Commute Time (in Hours) x Co-Located Team		3.79	6.71**
		(2.55)	(3.40)
Commute Time		−8.98***	−9.78***
		(1.15)	(1.28)
Father x Co-Located Team			−3.16
			(2.05)
Father			−1.04
			(0.69)
Mother x Co-Located Team			−1.43
			(2.26)
Mother			−3.24***
			(1.09)
Co-Located Team	1.33	−0.71	−1.57
	(0.89)	(1.47)	(1.66)
Date FE	✓	✓	✓
Dependent Mean	23.52	23.51	23.68
# Teams	790	790	683
Observations	527,186	526,665	369,731
R^2	0.32	0.32	0.33

Notes: This table analyzes the age differences in going into the office and how those differences interact with whether the engineer's team is co-located or distributed. The sample is limited to engineers in the main campus and includes weekdays after the first return-to-office mandate in 2022. The dependent variable is whether a worker badged into the office on that day. Commute time comes from using the Google API to calculate travel time from the engineer's home address (collected for nearly all engineers) to the main headquarters. Parenthood information comes from firm-conducted survey, where non-response reduces the sample size more appreciably. Standard errors are clustered by team. *p<0.1; **p<0.05; ***p<0.01.

Table A.12: Office Attendance by Age Interacted with Team and Manager Proximity

	Badged into Building			
	(1)	(2)	(3)	(4)
Young (<29) x Co-Located Team (Everyone in HQ)	5.44***	3.98***		
	(1.39)	(1.51)		
Young (<29) x All Teammates in HQ			5.13***	3.65***
			(1.14)	(1.27)
Young (<29) x Manager in HQ			2.63***	2.28**
			(0.99)	(1.12)
Older (≥29) x Co-Located Team (Everyone in HQ)	1.47			
	(0.90)			
Older (≥29) x All Teammates in HQ			1.48**	
			(0.74)	
Older (≥29) x Manager in HQ			0.35	
			(0.70)	
Co-Located Team (Everyone in HQ)		1.47		
		(0.90)		
All Teammates in HQ				1.48**
				(0.74)
Manager in HQ				0.35
				(0.70)
Young (<29)	4.84***	4.84***	3.29***	3.29***
	(0.59)	(0.59)	(0.93)	(0.93)
Dependent Mean	23.45	23.45	23.45	23.45
# Teams	782	782	782	782
Observations	519,016	519,016	519,016	519,016
R^2	0.32	0.32	0.32	0.32

Notes: This table analyzes young people's revealed preference to be with their coworkers and their managers. The sample is limited to engineers in the main campus and includes weekdays after the first return-to-office mandate in 2022. The dependent variable is whether a worker badged into the office on that day. The first two columns focus on whether the whole team is co-located in the headquarters as in Figure 9a. The second two columns differentiate between (i) all the teammates being in the headquarters and (ii) the manager being in the headquarters with the engineer. Both of these things must hold for the team to be co-located. Standard errors are clustered by team. *p<0.1; **p<0.05; ***p<0.01.

Table A.13: Office Attendance by Tenure Interacted with Team and Manager Proximity

	Badged into Building			
	(1)	(2)	(3)	(4)
Low Tenure (<16 mo) x Co-Located Team	5.50***	3.97**		
	(1.61)	(1.69)		
Low Tenure (<16 mo) x All Teammates in HQ			4.92***	3.50**
			(1.45)	(1.55)
Low Tenure (<16 mo) x manager in HQ			3.16**	2.53*
			(1.38)	(1.44)
High Tenure (≥16 mo) x Co-Located Team	1.53*			
	(0.83)			
High Tenure (≥16 mo) x All Teammates in HQ			1.42**	
			(0.69)	
High Tenure (≥16 mo) x manager in HQ			0.63	
			(0.62)	
Co-Located Team		1.53*		
		(0.83)		
All Teammates in HQ				1.42**
				(0.69)
manager in HQ				0.63
				(0.62)
Low Tenure (<16 mo)	8.45***	8.45***	6.60***	6.60***
	(0.83)	(0.83)	(1.08)	(1.08)
Dependent Mean	23.57	23.57	23.57	23.57
# Teams	791	791	791	791
Observations	528,346	528,346	528,346	528,346
R^2	0.32	0.32	0.32	0.32

Notes: This table analyzes new hires' revealed preference to be with their coworkers and their managers. The sample is limited to engineers in the main campus and includes weekdays after the first return-to-office mandate in 2022. The dependent variable is whether a worker badged into the office on that day. The first two columns focus on whether the whole team is co-located as in Figure 9a. The second two columns differentiate between the manager being in the headquarters with the engineer versus the rest of the team being in the headquarters. Standard errors are clustered by team. *p<0.1; **p<0.05; ***p<0.01.

I.C Predicting Comment Quality: Crowdsourced Comment Evaluation & Supervised Machine Learning Predictions

We predict the quality of each comment along multiple dimensions. To do this, we first crowdsource labels of about five thousand comments by employing a set of external evaluators to rate a random subset of comments. We then use a supervised machine learning algorithm to scale this approach and generate predicted labels for the nearly two hundred thousand comments in our dataset.

I.C.1 Crowdsourcing Comment Evaluation

We asked external evaluators to rate the quality of a random subset of comments along several dimensions. We recruited the evaluators through Upwork, selecting workers whom Upwork flagged as being top in the programming languages used by the firm. All the evaluators worked as software engineers, knew the programming languages used by the firm (e.g., PHP or Java), and had both written and received code reviews. For each comment, the engineers were asked to imagine that they had received the comment on a piece of code that they had written. They were then asked to respond to the following questions:

- Would you find this comment helpful?
- Do you think you would change your code because of this comment?
- Does this comment suggest actionable steps to change your code?
- Does this comment explain the reason for changing your code?
- Is the tone of this comment rude?

For the first four questions, the crowd-sourced engineers could answer "yes," "no," or "not enough information." For the question about tone, they could answer "No," "A little bit," "Moderately," "Very," or "Not enough information."

A total of 5,377 comments were evaluated by 22 software engineers. Comments were selected at random, stratifying by pre-post period, one- versus multi-building teams, and engineer gender. Each comment was stripped of any firm-specific content (e.g., the name of the firm) or code that may contain sensitive information.

For any particular dimension, engineers said they did not have enough information to rate between 4 to 26 percent of the comments. Of the comments that could be evaluated without additional information, 87 percent were considered helpful, 68 percent were deemed to be actionable, 70 percent were seen as likely to result in changing code, 58 percent gave a reasoning for the change, and 85 percent were considered to not be even a little bit rude.

The crowdsourced evaluations were provided by experienced engineers. Sixty-eight percent worked as software engineers for 5 or more years. All of them had some college

experience, and 86 percent had a college degree. These engineers had all written and received code reviews in the past, having received approximately 600 reviews and written approximately 560 reviews on average. Additionally, to verify that the engineers were sufficiently competent to provide meaningful evaluations of the comments, we conditioned their participation upon successfully answering the following technical questions.

- What is the time complexity of the following Python function that finds the maximum element in a list?

```
def find_max_element(lst):
    max_element = lst[0]
    for element in lst:
        if element > max_element:
            max_element = element

    return max_element
```

  - O(1)
  - O(n)
  - O(log n)
  - O(n^2)

- Suppose you have an array of integers in ascending order. You need to find a target element in the array and return its index. If the target element is not present in the array, you should return -1. Which of the following algorithms would be most appropriate for this task?
  - Linear Search
  - Binary Search
  - Depth-First Search (DFS)
  - Breadth-First Search (BFS)

- Which of the following data structures is typically used to implement a Last-In-First-Out (LIFO) behavior?
  - Linked-List
  - Queue
  - Hash Table
  - Stack

We included five overlapping comments to calculate measures of inter-rater reliability.

I.C.2 Supervised Machine Learning Prediction

To scale up from the labeled comments, we use a supervised machine learning algorithm, specifically a gradient-boosted decision-tree algorithm (Chen and Guestrin, 2016). In our setting, the predictors are the total character length of the comment, its number of words, and a vector of all the distinct words that appear in the comment, after dropping words like prepositions and pronouns (aka stop words) and those that appear in less than 1% of the training comments. These omissions help to reduce dimensionality.

Gradient-boosted decision-trees start with a simple decision tree and then iteratively refine it. For example, this approach could start with a simple decision tree that says that a comment with the word "nitpick" is unlikely to be helpful. The algorithm will then iteratively build on itself, taking the residuals from the original tree as the new object of the prediction. This iteration could note that, for example, when "nitpick" occurs with a substantive word like "model" or "data," it often is helpful. In this way, gradient boosting can iteratively arrive at relatively strong predictors from simple building blocks. We specifically limit the initial trees to a depth of three and iterate the model a hundred times.

We evaluate the accuracy of the model, using a hold-out sample of 20% of the labeled comments. Table A.14 summarizes the results of these validation exercises.

Table A.14: Summarizing Prediction Accuracy

Attribute	Accuracy	Uninformed Benchmark	Inter-rater Reliability
Helpful	77.7%	76.1%	70.1%
	[75.1%, 80.1%]		
Explains reasoning	68.2%	51.1%	59.9%
	[65.4%, 71%]		
Actionable	69.4%	60.3%	60.5%
	[66.6%, 72.1%]		
Likely to change code	63.9%	51.9%	51.7%
	[60.9%, 66.7%]		

Notes: This table examines the evaluates the accuracy of our prediction models, using the 20% of the labeled data that is held out as a test sample. To generate the labeled data, we employed software engineers outside our firm to evaluate a random sample of 5,377 comments on multiple dimensions (see Appendix I.C.1 for details on recruitment and sample validation). These raters assessed each comment along multiple dimensions, including whether the comment (i) was helpful, (ii) explained the underlying reasoning, (iii) was actionable, and (iv) was likely to change the code. To scale this approach, we used a supervised machine learning algorithm to generate predictions on the likely label of all 174,014 comments in our main sample. Specifically, we used the XGBoost algorithm (Chen and Guestrin, 2016). To evaluate the accuracy of this approach, this table compares the accuracy of the predictions in the test sample to an uninformative benchmark that always predicts the most common label and an alternative benchmark based on the inter-rater reliability of the raters themselves.