

Cite as: S. Daniotti *et al.*, *Science*
10.1126/science.adz9311 (2026).

Who is using AI to code? Global diffusion and impact of generative AI

Simone Daniotti^{1,2*}, Johannes Wachs^{1,3,4}, Xiangnan Feng¹, Frank Neffke^{1,5}

¹Complexity Science Hub, Vienna, Austria. ²Copernicus Institute of Sustainable Development, Utrecht University, Utrecht, Netherlands. ³Department of Network Science, Corvinus University of Budapest, Budapest, Hungary. ⁴Agglomeration, Networks, and Innovation Research Group, ELTE Centre for Economic and Regional Studies, Budapest, Hungary. ⁵Transforming Economies Lab, IT:U Interdisciplinary Transformation University, Linz, Austria.

*Corresponding author. Email: daniotti@csh.ac.at

Generative coding tools promise big productivity gains, but uneven uptake could widen skill and income gaps. We train a neural classifier to spot AI-generated Python functions in over 30 million GitHub commits by 160,097 software developers, tracking how fast, and where, these tools take hold. Currently AI writes an estimated 29% of Python functions in the US, a shrinking lead over other countries. We estimate quarterly output, measured in online code contributions, consequently increased by 3.6%. AI seems to benefit experienced, senior-level developers: they increased productivity and more readily expanded into new domains of software development. In contrast, early-career developers showed no significant benefits from AI adoption. This may widen skill gaps and reshape future career ladders in software development.

According to proponents, artificial intelligence (AI)—in particular generative AI (genAI)—will drastically increase our productivity and revolutionize the way we work. For instance, genAI is expected to complement or substitute humans in an increasing set of tasks (1). This forces individuals, firms, and policymakers to make important decisions about the use and regulation of genAI under major uncertainty. The stakes are high: genAI has become widely accessible through tools such as ChatGPT or Claude, directly complements human thinking (2), and holds the potential of becoming a general-purpose technology that can solve a wide variety of problems (3).

Experimental and quasi-experimental evidence so far supports the notion that genAI has transformative potential, showing that the use of genAI leads to increases in productivity and output of individual workers in a variety of jobs (1, 4–6). Surveys and data reported by large language model (LLM) owners suggest that these technologies are diffusing rapidly (7–9). Yet, estimates of the aggregate impact of AI on gross domestic product (GDP) and employment are often modest (10, 11), suggesting that we are far from having a clear picture of the overall impacts of AI.

We do know that there is significant heterogeneity of adoption, which could lead to economic divergence. Although genAI use is widespread in the working age population, self-reported adoption rates differ markedly across demographics, seniority, work experience, and sectors (8, 9). Evidence from job ads and firm websites suggests that adoption of genAI varies across geography (12, 13). If genAI indeed substantially raises productivity, any implied barriers to adoption will have significant consequences for inequality within and across countries (14). Historically, however, macro-level productivity effects of general-purpose technologies, such as steam engines, dynamos, and computers, have taken long to

materialize (4, 15–17). Together, this leads to substantial uncertainty about the impact of genAI today.

Resolving this uncertainty requires accurately determining adoption rates, intensity of use, and productivity effects at a global level. Surveys demonstrating demographic and sectoral heterogeneities in genAI adoption often focus on single countries (8, 9). Previous work comparing AI adoption in different countries using survey data finds evidence of differences within and between countries (18), but differences in sample weighting and analysis periods of the surveys limit our ability to directly compare observed rates. In the context of genAI, respondents may under-report usage, especially at work, to avoid judgment (19, 20). Nevertheless, surveys provide a valuable resource for understanding adoption patterns. Similarly, randomized controlled trials (RCTs) (1, 4–6, 21) and natural experiments (22–24) are indispensable because they measure causal effects of genAI adoption by design. However, they typically consider individuals as “treated” whenever they have access to genAI tools, without quantifying the extent to which treated individuals used genAI during the experiments. Moreover, surveys and experiments tend to observe individuals over short time periods, which limits our ability to know the dynamics of adoption and to observe effects of adoption that materialize more slowly.

To begin to address these gaps, we ask if we can directly measure the adoption and intensity of use of genAI by individuals over time using machine learning instead of from self-reported information. If so, what do such measures tell us about the rate of adoption of genAI? Does this differ across countries and demographics? How does genAI impact the output individuals produce, and how do individual characteristics such as experience moderate such effects?

To answer these questions, we study genAI use at a fine-grained level in one of its main domains of application: software development, an important and high-value sector (25, 26) that is uniquely exposed to genAI (21, 22, 27, 28). To do so, we design and implement a machine learning classifier to identify code written with substantial AI assistance in over 30 million software developer contributions, also known as commits, to open-source Python projects on GitHub. To train this classifier, we assemble a custom training set, combining existing sources with a procedure that generates synthetic training data. This allows us to analyze shifting patterns of AI-generated code at a granular level. We leverage this novel source of micro-data to study how quickly the use of genAI in coding diffuses in six major countries, how this diffusion relates to demographic characteristics, and how it affects programming activity in a sample of 100,097 US software developers.

Detecting AI generated code

To collect a large dataset of coding activity, we gather all commits by 100,097 US GitHub users to Python-based open-source projects, recursively cloning all GitHub directories related to each project. Next, we add commits from a random sample of 2,000 developers per year (60,000 developers in total) for each of five other major countries in software development: China, France, Germany, India and Russia. We then analyze these commits to assess the prevalence of AI-generated code (see materials and methods).

Figure 1 describes how we classify these code contributions as either human or AI-generated. We limit this analysis to blocks of code that represent functions to focus on a fine-grained, self-contained, yet substantive unit of code. We first construct a ground truth dataset (Fig. 1A), collecting Python functions of which we are certain they were written by a human programmer/developer. To do so, we take functions written in 2018, as they predate the release of capable genAI models. Because programming styles evolve over time, we add functions created in later years from the HumanEval datasets for the years 2022 and 2024. To add a dataset of similar functions but written by genAI we apply a two-step procedure. First, for each human-written function we ask one LLM to describe the function in English, specifying the type of input and output of the function. Second, we feed this text to a second LLM and request the model to generate a function based on this description. Our use of two different LLMs — unlike previous approaches (29)— avoids creating unnecessarily strong correlations between human code and its transcription, while ensuring that the (synthetic) AI-generated functions in our training data are close in functionality to the original human-written functions.

We then train a machine learning classifier on this dataset. Following (30), we transform each function using

GraphCodeBert, a pre-trained language model for code that embeds a function into a high-dimensional vector space using its tokens, comments, and the dataflow graph of its variables (31). The resulting vectors are fed into a classifier to determine whether a given function was written by a human or by genAI (Fig. 1, B and C).

Results

The classifier performs remarkably well, achieving an out-of-sample ROC AUC Score of 0.96 (Fig. 1D) and Average Rate of True positives of 0.95. We apply this classifier to 5 million functions extracted from 31 million contributions to Python projects from the beginning of 2019 to the end of 2024 for the full population of US-based users and the sampled users in the five other countries (Fig. 1D). In the supplementary materials (see supplementary materials, section S2), we show that the classifier also correctly identifies code generated by more recent LLMs introduced after our data collection ended, as well as code produced in real-world interactions with LLMs, albeit with somewhat lower accuracy. Retraining the classifier on code produced by these newer LLMs further improves performance.

Figure 2A plots the AI adoption trajectory for US developers. Adoption rates sharply increase following major AI advancements, including the launches of Copilot, ChatGPT, and second generation LLMs. Figure 2B compares the US to the five other major countries we cover in the global race toward AI adoption. This shows that the US took an early lead, which it has managed to maintain ever since. About 29% of Python functions in the US were generated by AI by the end of 2024, closely followed by 23/24% for Germany and France. India draws close at 20%, after having initially lagged in adoption. In contrast, Russia and China have so far remained late adopters.

Focusing on the full population of US developers, we find that AI adoption rates drop with the number of years that developers have been active on GitHub. Figure 3B shows that whereas the most experienced developers use genAI in 27% of their code, developers who have just joined the GitHub platform use these tools for 37% of code. In contrast, using (self-reported) first-name-based gender inference algorithms, we find no difference between men and women (Fig. 3A).

To assess how genAI impacts the quantity and nature of code that developers produce, we rely on regression models with user and quarter-of-year fixed effects. This compares the output — in terms of quarterly number of commits — of the same developer at different points in time, controlling for economy-wide trends. These models, summarized in Fig. 3C, suggest a substantial impact of genAI on developer productivity. We find consistent effects across different sets of commits: all commits, commits that modify multiple files (which typically require navigating dependencies across scripts), and

commits that add new libraries (which typically introduce new functionality) to scripts. Moving from 0 to 29% genAI usage—the estimated US adoption rate by the end of 2024—is associated with a 3.6% increase in commit rates across all these commit types. However, these associations with user productivity are fully driven by senior-level users, for whom a 29% adoption rate would imply a 6.2% increase in commit rates (Fig. 3D). In contrast, we observe no statistically significant effects among early-career users.

Apart from increasing activity rates, AI adoption is also associated with increased experimentation with new libraries and combinations of libraries, which past research (32) interpreted as signs of innovation. Because libraries often focus on specific types of functionality — such as visualization, natural language processing, web interactions, or database operations — these findings suggest that genAI helps developers expand their capabilities to new domains of software development. At average end-of-2024 AI use rates for US developers, our models predict that developers who use genAI at the average US end-of-2024 rate of 29% developers will implement 2.7% more extra new combinations of libraries. Results are robust to variations in how we identify new library introductions. It is unlikely that the observed effects are due to reflect the addition of esoteric libraries (“AI slop”): findings do not change much if we only use the 5,000 most common libraries or if we first group libraries into 124 coarse categories first. Moreover, fig. S6 of the supplementary materials (see section S4.5) shows that these effects, as well as the earlier productivity effects, are likely lower bounds because errors in the measurement of users’ AI adoption rates bias each of these estimates downwards.

Discussion

We set out to measure the use of genAI among software developers at the micro-level and on a global scale. Focusing on the software development labor force, we demonstrated how genAI has diffused and how this has affected the quantity and nature of code that developers produce. To do so, we developed a new genAI classifier to identify AI-generated functions in GitHub commits. Applied to a large dataset covering software development activity across six major countries, we document noticeable growth spikes in genAI-generated code soon after key genAI releases. Yet, we also observe significant differences between countries: the United States leads but its advantage is narrowing, with Germany and France close behind, India catching up fast, and China and Russia still lagging at the end of 2024. Corroborating existing studies (8, 9), our estimated adoption rates are higher among early-career developers. However, unlike most previous work, we find no significant differences between men and women.

We also find that genAI reshapes both the volume and nature of programming work. Using within-developer

variation—comparing the same developer before and after adopting genAI—we show that AI adoption substantially increases output. Developers using genAI are also more likely to incorporate novel combinations of software libraries into their code, suggesting they venture into new technical domains (32) using unfamiliar building blocks (33). However, both productivity and exploration gains concentrate almost exclusively among senior-level developers. In contrast, although early-career developers used genAI more, they do not realize the same benefits.

Our observations that early-career developers use AI more but get less out of it than their more experienced colleagues, may reflect differences in how well developers utilize genAI across a broader set of tasks. Senior-level developers will, for instance, be quicker to interpret, and spot mistakes in, AI-generated code. Such an interpretation is supported by (22), who show that access to genAI allows senior developers to spend less time on coordination activities and more time on coding.

Our findings both replicate and extend findings from other investigations. The current study estimates the most recent adoption rates in the US at around 29%, which is remarkably similar to adoption rates claimed for coding work at Microsoft (34) and Amazon (35). This shows that, despite our focus on code from open-source Python libraries, our results closely align with estimates of adoption rates from other contexts and may generalize beyond the specific setting of this study.

Unlike most other studies, our methodological advances and design allow us to compare early adoption rates across countries. Here, we find a clear and sustained lead by US developers. Use of LLMs may be lower in countries like China and Russia because of differences in their supply of (providers such as OpenAI and Anthropic block access) and differences in demand (censorship limits local use, even though many users connect using VPNs (36)). However, other major countries are quickly catching up, eroding the US’ first-mover advantage. Another feature that sets our study apart is that existing literature typically focuses on access to genAI — yielding reduced-form estimates of the causal effect of the so-called intention-to-treat, not of genAI itself — or usage in controlled experimental settings. In contrast, our approach allows us to quantify the intensity with which this new technology is used in real-world work activities. Finally, we note that our cross-country evidence on genAI use complements firm-level survey work on broader AI adoption which extends back to before the genAI era (18); while levels are not directly comparable, both perspectives document persistent cross-country gaps in AI use.

With respect to the productivity effects of genAI, our estimates are generally smaller than those found in RCTs (6, 37) and studies exploiting natural experiments (22, 24). In

robustness checks (supplementary materials, section S4) we study whether nonlinearities or threshold effects in the benefits of genAI adoption can explain such discrepancies, but find little evidence for this hypothesis. A more promising explanation is measurement error, which is likely to bias effect estimates downwards. In line with this, fig. S6 (supplementary materials, section S4.5) shows that our effect estimates increase substantially if we correct them for measurement error. Moreover, we show that beneficial effects concentrate among senior developers, while early-career developers do not appear to benefit much from genAI. The higher effect estimates reported in prior studies may, therefore, also reflect differences in the populations and complier samples they analyzed. At the same time, the gap in benefits from genAI between senior and junior developers creates large uncertainties about the availability and nature of future career and learning paths for early-career developers.

This study has several limitations. First, our analysis focuses on software development. Although this limits its scope, work in this sector is uniquely amenable to quantitative analysis at a level of granularity that is required to study how AI affects workers and their jobs. Within software, we focus only on Python-based open-source contributions. While Python is a widely used language, adoption patterns may differ in other programming ecosystems. We argue that estimates derived from open-source Python code on GitHub are economically meaningful, because open-source software (OSS) underpins most commercial stacks and generates significant value (33, 38). GitHub's central role in collaboration, networking, and signaling further ties our evidence to professional activity (39, 40). Finally, that our estimates of AI use in the US align closely with reported AI use at leading US firms increases our confidence in the external validity of our findings.

More generally, we also do not account for potential externalities between co-workers or heterogeneity in productivity across firms, all of which may be relevant factors in how genAI affects programming activity. Beyond firms, our geographic analysis is limited to a subset of countries and it would be important to widen the analysis to include countries at different income levels. In the specific case of China — where the programming community also relies on an alternative collaboration platform, Gitee (41) — there is some additional risk that our focus on GitHub projects distorts estimates. Finally, revisiting the effects of genAI, there are many other ways to evaluate the productivity of developers that heed more attention to code quality, from tracking how issues get resolved and code merges to implemented test coverage. While feasible in principle, such analysis requires new data collection and careful design of metrics. We therefore leave questions about the effect of genAI on code quality for future research.

How much value has genAI created in coding? While hard to answer definitively, our study offers some important pieces of this puzzle. Based on an analysis of detailed task surveys and wage statistics for about 900 different occupations, we estimate that the US spends between 637 and 1,063B USD on labor costs related to coding activities per year (supplementary materials, section S6). Assuming our estimated diffusion rates of 29% by the end of 2024 (based on open-source Python contributions) are representative of code in general, the annual value generated by genAI coding assistants in the US would depend on how much they increase productivity. Using our own, conservative, baseline estimates, genAI would have increased the volume of commits by 3.6%. Assuming these commits reflect valuable code contributions, our calculation implies that genAI generates 23 – 38 billion USD of additional code per year. This estimate assumes that productivity gains are similar across programming languages. In a more conservative scenario, where productivity effects outside Python are negligible, the value of genAI would drop to about 17% of this figure (approximately 4 – 6 billion USD), using estimates of Python's share of GitHub code (42).

By contrast, various lab experiments (21, 37) and field experiments (6) in software development all yield substantially larger causal effects of genAI on task completion times — arguably a more relevant quantity to track than commit volumes. Averaging across such studies (see materials and methods for details) yields an estimated 6.0%-15.7% increase in productivity at a 29% adoption rate. This translates into a range of 38-167 billion USD for the direct impact of genAI on US coding activities. However, these estimates ignore that genAI may also lead to a reduction in the market price of code. Because this yields cost savings for consumers of code, while reducing profits for suppliers (i.e., programmers/developers), factoring in such general equilibrium effects further widens the range of possible outcomes (supplementary materials, section S7). In the materials and methods, we show that this would mostly affect the upper bound of our estimates, with lower bounds all but unaffected. The upshot of these back-of-the-envelope calculations is that, although the total value of genAI to the US economy is uncertain, it is most likely substantial, on the order of at least tens of billions of USD.

Given that genAI has diffused quickly beyond the US, global cost savings would be larger still, even if we confine ourselves to the software sector. Moreover, we are currently still in the early phases of the diffusion curve of what looks to be a new general purpose technology (3). Historically, early-stage productivity effects of general purpose technologies have been hard to identify because it takes time to integrate them into firm level workflows and procedures, train workers and amass the complementary assets needed to fully exploit their potential. Based on this, we find ourselves on the

bullish side of the debate when it comes to the productivity effects of genAI.

Our results on such effects and the heterogeneous diffusion of genAI raise important questions for policymakers and researchers. We need to understand the barriers to AI adoption: are these barriers similar to prior radical innovations (43) or is this era different? Additionally, these barriers need to be understood not only at the individual level, but also at the firm, regional, and national levels. Our study takes a first step toward answering such questions.

Moreover, given the wide dispersion in productivity across developers (44–46) and our finding that benefits accrue to more experienced coders only, future research should explore how AI adoption affects developer activity at the upper tail of elite developers, where the most significant breakthroughs and innovations are likely to occur (47). Finally, our study exclusively focused on programming tasks. Yet, one study of elite software developers suggests that access to genAI leads to a shift from managerial tasks to coding (22), suggesting that an important margin along which productivity effects materialize is shifts in the task composition of software developer jobs.

The nature of work often changes with the introduction of new technologies. Understanding these changes is especially difficult when the innovation in question is radical (43), such as the spinning jenny, transistors, or robots in the past, and at the same time pervasive (48). The uncertainty of the effects of genAI on work and labor markets is reflected in the wide range of attitudes researchers and policymakers take toward it, ranging from utopian to skeptical and outright apocalyptic. These attitudes are formed in a fast-moving context, and are based on incomplete evidence on the adoption and effects of AI. The findings in this study provide better evidence of how genAI is used in a large, important, and highly exposed sector of the economy, as well as a way to monitor this in real-time going forward. Applying our AI detection classifier to millions of code contributions made over a six-year period, we can confirm that AI adoption is fast, but heterogeneous across countries and individuals. Moreover, AI adoption is associated with increased activity rates in online software development collaborations.

However, one of the most surprising findings was genAI increased experimentation with new libraries. This suggests genAI allowed users to advance faster to new areas of programming, embedding new types of functionality in their code. This corroborates prior findings (49) that genAI increases individual innovation, pushing individuals' capabilities in terms of the use of new combinations of libraries. However, again only experienced, senior-level users seem able to leverage genAI in this way, with important consequences for career development and learning in the presence of genAI.

REFERENCES AND NOTES

1. F. Dell'Acqua, E. McFowland, E. R. Mollick, H. Lifshitz-Assaf, K. Kellogg, S. Rajendran, L. Kraymer, F. Candelon, K. R. Lakhani, Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality (SSRN, Harvard Business School Technology and Operations Management Unit, working paper no. 24-013, 2023); <https://ssrn.com/abstract=4573321>.
2. E. Mollick, *Co-Intelligence: Living and Working with AI* (Penguin, 2024).
3. T. Eloundou, S. Manning, P. Mishkin, D. Rock, GPTs are GPTs: Labor market impact potential of LLMs. *Science* **384**, 1306–1308 (2024). [doi:10.1126/science.adh0998](https://doi.org/10.1126/science.adh0998) [Medline](#)
4. E. Brynjolfsson, The productivity paradox of information technology. *Commun. ACM* **36**, 66–77 (1993). [doi:10.1145/163298.163309](https://doi.org/10.1145/163298.163309)
5. S. Noy, W. Zhang, Experimental evidence on the productivity effects of generative artificial intelligence. *Science* **381**, 187–192 (2023). [doi:10.1126/science.adh2586](https://doi.org/10.1126/science.adh2586) [Medline](#)
6. Z. Cui, M. Demirel, S. Jaffe, L. Musolff, S. Peng, T. Salz, The Effects of Generative AI on High-Skilled Work: Evidence from Three Field Experiments with Software Developers (SSRN, 2025); <https://ssrn.com/abstract=4945566>.
7. T. Teubner, C. M. Flath, C. Weinhardt, W. van der Aalst, O. Hinz, Welcome to the era of ChatGPT et al.: The prospects of large language models. *Bus. Inf. Syst. Eng.* **65**, 95–101 (2023). [doi:10.1007/s12599-023-00795-x](https://doi.org/10.1007/s12599-023-00795-x)
8. A. Bick, A. Blandin, D. J. Deming, "The rapid adoption of generative AI" (National Bureau of Economic Research, working paper 32966, 2025); https://www.nber.org/system/files/working_papers/w32966/w32966.pdf.
9. A. Humlum, E. Vestergaard, The unequal adoption of ChatGPT exacerbates existing inequalities among workers. *Proc. Natl. Acad. Sci. U.S.A.* **122**, e2414972121 (2025). [doi:10.1073/pnas.2414972121](https://doi.org/10.1073/pnas.2414972121)
10. D. Acemoglu, The simple macroeconomics of AI. *Econ. Policy* **40**, 13–58 (2025). [doi:10.1093/epolic/eia042](https://doi.org/10.1093/epolic/eia042)
11. A. Humlum, E. Vestergaard, Large Language Models, Small Labor Market Effects (SSRN, University of Chicago, Becker Friedman Institute for Economics, working paper no. 2025-56, 2025); <https://ssrn.com/abstract=5219933>.
12. E. Andreadis, E. Chatzikonstantinou, E. Kalotychou, C. Louca, C. Makridis, Local Heterogeneity in Artificial Intelligence Jobs Over Time and Space (SSRN, 2024); <https://ssrn.com/abstract=5078805>.
13. D. Bearson, N. Wright, Strategic Targeting and Unequal Global Adoption of Artificial Intelligence (SSRN, Columbia Business School, research paper no. 5187851, 2025); <https://ssrn.com/abstract=5187851>.
14. D. Comin, B. Hobijn, An exploration of technology diffusion. *Am. Econ. Rev.* **100**, 2031–2059 (2010). [doi:10.1257/aer.100.5.2031](https://doi.org/10.1257/aer.100.5.2031)
15. P. A. David, The dynamo and the computer: An historical perspective on the modern productivity paradox. *Am. Econ. Rev.* **80**, 355–361 (1990).
16. E. Brynjolfsson, D. Rock, C. Syverson, The productivity J-curve: How intangibles complement general purpose technologies. *Am. Econ. J. Macroecon.* **13**, 333–372 (2021). [doi:10.1257/mac.20180386](https://doi.org/10.1257/mac.20180386)
17. R. Juhász, M. P. Squicciarini, N. Voigtländer, Technology adoption and productivity growth: Evidence from industrialization in France. *J. Polit. Econ.* **132**, 3215–3259 (2024). [doi:10.1086/730205](https://doi.org/10.1086/730205)
18. F. Calvino, L. Fontanelli, "A portrait of AI adopters across countries: Firm characteristics, assets' complementarities and productivity" (OECD Science, Technology and Industry Working Papers, no. 2023/02, 2023); <https://doi.org/10.1787/0fb79bb9-en>.
19. Y. Ling, A. Kale, A. Imas, Underreporting of AI Use: The Role of Social Desirability Bias (SSRN, 2025); <https://ssrn.com/abstract=5464215>.
20. D. Almog, Barriers to AI adoption: Image concerns at work. *arXiv:2511.18582* [econ.GN] (2025).
21. S. Peng, E. Kalliamvakou, P. Cihon, M. Demirel, The impact of AI on developer productivity: Evidence from GitHub Copilot. *arXiv:2302.06590* [cs.SE] (2023).
22. M. Hoffmann, S. Boysel, F. Nagle, S. Peng, K. Xu, Generative AI and the Nature of Work (SSRN, Harvard Business School Strategy Unit, working paper no. 25-021, 2025); <https://ssrn.com/abstract=5007084>.
23. D. Yevercheyahu, R. Mayya, G. Oestreicher-Singer, The impact of large language models on open-source innovation: Evidence from GitHub Copilot. *arXiv:2409.08379* [cs.SE] (2024).

24. F. Song, A. Agarwal, W. Wen, The impact of generative AI on collaborative open-source software development: Evidence from GitHub Copilot. [arXiv:2410.02091 \[cs.SE\]](#) (2024).
25. S. Aum, Y. Shin, "Is software eating the world?" (National Bureau of Economic Research, working paper 32591, 2024); https://www.nber.org/system/files/working_papers/w32591/w32591.pdf.
26. S. Juhász, J. Wachs, J. Kaminski, C. A. Hidalgo, The software complexity of nations. [arXiv:2407.13880 \[econ.GN\]](#) (2024).
27. K. Handa, A. Tamkin, M. McCain, S. Huang, E. Durmus, S. Heck, J. Mueller, J. Hong, S. Ritchie, T. Belonax, K. K. Troy, D. Amodei, J. Kaplan, J. Clark, D. Ganguli, Which economic tasks are performed with AI? Evidence from millions of Claude conversations. [arXiv:2503.04761 \[cs.CY\]](#) (2025).
28. R. M. Del Rio-Chanona, N. Laurentsyeva, J. Wachs, Large language models reduce public knowledge sharing on online Q&A platforms. *PNAS Nexus* **3**, pgae400 (2024). [doi:10.1093/pnasnexus/pgae400 Medline](#)
29. T. Ye, Y. Du, T. Ma, L. Wu, X. Zhang, S. Ji, W. Wang, Uncovering LLM-Generated Code: A Zero-Shot Synthetic Code Detector via Code Rewriting. [arXiv:2405.16133 \[cs.SE\]](#) (2024).
30. P. T. Nguyen, J. Di Rocco, C. Di Sipio, R. Rubei, D. Di Ruscio, M. Di Penta, GPTSniffer: A CodeBERT-based classifier to detect source code written by ChatGPT. *J. Syst. Softw.* **214**, 112059 (2024). [doi:10.1016/j.jss.2024.112059](#)
31. D. Guo, S. Ren, S. Lu, Z. Feng, D. Tang, S. Liu, L. Zhou, N. Duan, A. Svyatkovskiy, S. Fu, M. Tufano, S. K. Deng, C. Clement, D. Drain, N. Sundaresan, J. Yin, D. Jiang, M. Zhou, GraphCodeBERT: Pre-training Code Representations with Data Flow. [arXiv:2009.08366 \[cs.SE\]](#) (2021).
32. H. Fang, J. Herbsleb, B. Vasilescu, "Novelty begets popularity, but curbs participation - A macroscopic view of the Python open-source ecosystem" in *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering* (Association for Computing Machinery, 2024), pp. 1–11.
33. N. Eghbal, "Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure" (Ford Foundation, 2016); <https://www.fordfoundation.org/learning/library/research-reports/roads-and-bridges-the-unseen-labor-behind-our-digital-infrastructure/>.
34. J. Novet, "AI now writes a big chunk of code at Microsoft and Google — And it could be coming for even more at Meta," *Business Insider*, 30 April 2025; <https://www.businessinsider.com/ai-code-meta-microsoft-google-llama-on-engineers-2025-4>.
35. N. Scheiber, "At Amazon, some coders say their jobs have begun to resemble warehouse work," *New York Times*, 27 May 2025; <https://www.nytimes.com/2025/05/25/business/amazon-ai-coders.html>.
36. H. Bao, M. Sun, M. Tepitskiy, Where there's a will there's a way: ChatGPT is used more for science in countries where it is prohibited. *Quant. Sci. Stud.* **6**, 716–731 (2025). [doi:10.1162/qss_a_00368](#)
37. E. Paradis, K. Grey, Q. Madison, D. Nam, A. Macvean, V. Meimand, N. Zhang, B. Ferrari-Church, S. Chandra, "How much does AI impact development speed? An enterprise-based randomized controlled trial" in *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (IEEE, 2025), pp. 618–629.
38. M. Hoffmann, F. Nagle, Y. Zhou, The Value of Open Source Software (SSRN, Harvard Business School Strategy Unit, working paper no. 24-038, 2024); <https://ssrn.com/abstract=4693148>.
39. L. Dabbish, C. Stuart, J. Tsay, J. Herbsleb, "Social coding in GitHub: Transparency and collaboration in an open software repository" in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work* (Association for Computing Machinery, 2012), pp. 1277–1286.
40. L. Abou El-Kompoz, M. Goldbeck, Career concerns as public good: The role of signaling for open source software development. *Labour Econ.* **97**, 102800 (2025). [doi:10.1016/j.labeco.2025.102800](#)
41. J. Gortmaker, "Open source software policy in industry equilibrium" (2024); https://jeffgortmaker.com/files/Open_Source_Software_Policy_in_Industry_Equilibrium.pdf.
42. GitHub 2.0, GitHub language stats (2025); https://madnight.github.io/github/#/pull_requests/2024/1.
43. K. Frenken, M. B. Punt, A New View on Radical Innovation, SocArXiv (OSF, 2023); https://osf.io/preprints/socarxiv/6cr5t_v1.
44. H. Sackman, W. J. Erikson, E. E. Grant, Exploratory experimental studies comparing online and offline programming performance. *Commun. ACM* **11**, 3–11 (1968). [doi:10.1145/362851.362858](#)
45. G. E. Bryan, "Not all programmers are created equal" in *1994 IEEE Aerospace Applications Conference Proceedings* (IEEE, 1994), pp. 55–62.
46. L. Betti, L. Gallo, J. Wachs, F. Battiston, The dynamics of leadership and success in software development teams. *Nat. Commun.* **16**, 3956 (2025). [doi:10.1038/s41467-025-59031-7 Medline](#)
47. H. Fang, P. Park, J. Evans, J. Herbsleb, B. Vasilescu, The Strength of Weak Ties Between Open-Source Developers. [arXiv:2411.05646 \[cs.SE\]](#) (2024).
48. C. Freeman, C. Perez, "Structural Crises of Adjustment, Business Cycles and Investment Behavior" in *Technical Change and Economic Theory*, G. Dosi, C. Freeman, R. Nelson, G. Silverberg, L. Soete, Eds. (Pinter, 1988), pp. 38–66.
49. A. R. Doshi, O. P. Hauser, Generative AI enhances individual creativity but reduces the collective diversity of novel content. *Sci. Adv.* **10**, eadn5290 (2024). [doi:10.1126/sciadv.adn5290 Medline](#)
50. GitHub, GitHub GraphQL API Documentation (2024); <https://docs.github.com/en/graphql>.
51. D. Spadini, M. Aniche, A. Bacchelli, "PyDriller: Python framework for mining software repositories" in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Association for Computing Machinery, 2018), pp. 908–911.
52. M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, W. Zaremba, Evaluating Large Language Models Trained on Code. [arXiv:2107.03374 \[cs.LG\]](#) (2021).
53. Q. Zheng, X. Xia, X. Zou, Y. Dong, S. Wang, Y. Xue, L. Shen, Z. Wang, A. Wang, Y. Li, T. Su, Z. Yang, J. Tang, "CodeGeeX: A pre-trained model for code generation with multilingual benchmarking on HumanEval-X" in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Association for Computing Machinery, 2023), pp. 5673–5684.
54. Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, M. Zhou, "CodeBERT: A pre-trained model for programming and natural languages" in *Findings of the Association for Computational Linguistics: EMNLP 2020*, T. Cohn, Y. He, Y. Liu, Eds. (Association for Computational Linguistics, 2020), pp. 1536–1547.
55. Scikit-learn Developers, Precision, recall and F-measures (2025); https://scikit-learn.org/stable/modules/model_evaluation.html#precision-recall-f-measure-metrics.
56. R. P. Buse, W. R. Weimer, Learning a metric for code readability. *IEEE Trans. Softw. Eng.* **36**, 546–558 (2010). [doi:10.1109/TSE.2009.70](#)
57. R. J. Miera, J. A. Musselman, J. A. Navarro, B. Shneiderman, Program indentation and comprehensibility. *Commun. ACM* **26**, 861–867 (1983). [doi:10.1145/182.358437](#)
58. A. Hindle, E. T. Barr, M. Gabel, Z. Su, P. Devanbu, On the naturalness of software. *Commun. ACM* **59**, 122–131 (2016). [doi:10.1145/2902362](#)
59. W. Zhao, X. Ren, J. Hessel, C. Cardie, Y. Choi, Y. Deng, WildChat: 1M ChatGPT interaction logs in the wild. [arXiv:2405.01470 \[cs.CL\]](#) (2024).
60. L. Santamaria, H. Mihaljević, Comparison and benchmark of name-to-gender inference services. *PeerJ Comput. Sci.* **4**, e156 (2018). [doi:10.7717/peerj-cs.156 Medline](#)
61. A. D. VanHelene, I. Khatri, C. B. Hilton, S. Mishra, E. D. Gamsiz Uzun, J. L. Warner, Inferring gender from first names: Comparing the accuracy of Genderize, Gender API, and the gender R package on authors of diverse nationality. *PLOS Digit. Health* **3**, e0000456 (2024). [doi:10.1371/journal.pdig.0000456 Medline](#)
62. B. Vasilescu, D. Posnett, B. Ray, M. G. J. van den Brand, A. Serebrenik, P. Devanbu, V. Filkov, "Gender and tenure diversity in GitHub teams" in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Association for Computing Machinery, 2015), pp. 3789–3798.

63. J. Terrell, A. Kofink, J. Middleton, C. Rainear, E. Murphy-Hill, C. Parnin, J. Stallings, Gender differences and bias in open source: Pull request acceptance of women versus men. *PeerJ Comput. Sci.* **3**, e111 (2017). [doi:10.7717/peerj-cs.111](https://doi.org/10.7717/peerj-cs.111)
64. Z. Griliches, J. A. Hausman, Errors in variables in panel data. *J. Econom.* **31**, 93–118 (1986). [doi:10.1016/0304-4076\(86\)90058-8](https://doi.org/10.1016/0304-4076(86)90058-8)
65. A. van Dam, A. Gomez-Lievano, F. Neffke, K. Frenken, An information-theoretic approach to the analysis of location and colocation patterns. *J. Reg. Sci.* **63**, 173–213 (2023). [doi:10.1111/jors.12621](https://doi.org/10.1111/jors.12621)
66. M. J. Handel, The O*NET content model: Strengths and limitations. *J. Labour Mark. Res.* **49**, 157–176 (2016). [doi:10.1007/s12651-016-0199-8](https://doi.org/10.1007/s12651-016-0199-8)
67. US Bureau of Labor Statistics, Employer costs for employee compensation – September 2022 (2022); https://www.bls.gov/news.release/archives/eccec_12152022.pdf.
68. M. Svanberg, W. Li, M. Fleming, B. Goehring, N. Thompson, Beyond AI Exposure: Which Tasks are Cost-Effective to Automate with Computer Vision? (SSRN, 2024); <https://ssrn.com/abstract=4700751>.
69. D. H. Autor, D. Dorn, The growth of low-skill service jobs and the polarization of the U.S. labor market. *Am. Econ. Rev.* **103**, 1553–1597 (2013). [doi:10.1257/aer.103.5.1553](https://doi.org/10.1257/aer.103.5.1553)
70. G. de Rassenfosse, B. van Pottelsberghe de la Potterie, On the price elasticity of demand for patents. *Oxf. Bull. Econ. Stat.* **74**, 58–77 (2012). [doi:10.1111/j.1468-0084.2011.00638.x](https://doi.org/10.1111/j.1468-0084.2011.00638.x)
71. G. de Rassenfosse, “On the price elasticity of demand for trademarks” in *Trademarks and Their Role in Innovation, Entrepreneurship and Industrial Organization*, C. Castaldi, J. Block, M. J. Flikkema, Eds. (Routledge, 2021), pp. 83–96.
72. S. Daniotti, J. Wachs, F. Neffke, X. Feng, Who is using AI to code? Global diffusion and impact of generative AI [Dataset], Dryad (2026). [doi:10.5061/dryad.3r2280gy0](https://doi.org/10.5061/dryad.3r2280gy0)

ACKNOWLEDGMENTS

We thank Ulrich Schetter, Hillary Vipond, Andrea Musso, and participants of the ANETI Brownbag seminar for helpful comments. We thank Márton Salamon for valuable research assistance. **Funding:** Austrian Research Promotion Agency (FFG) grant ESSENCSE (873927) (SD, JW, XF, FN); Hungarian National Scientific Fund grant OTKA FK-145960 (JW). **Author contributions:** Conceptualization: SD, FN; Data Analysis: SD, JW, XF, FN; Data Collection: SD, JW; Data Collection and Estimation, US wage sums: XF; Primary Method: SD; Writing – original draft: FN, JW; Writing – review & editing: SD, JW, XF, FN. **Competing interests:** There are no competing interests to declare. **Data, code, and materials availability:** Data to replicate our analyses are available on Dryad (72). **License information:** Copyright © 2026 the authors, some rights reserved; exclusive licensee American Association for the Advancement of Science. No claim to original US government works. <https://www.science.org/about/science-licenses-journal-article-reuse>

SUPPLEMENTARY MATERIALS

science.org/doi/10.1126/science.adz9311

Materials and Methods

Supplementary Text

Figs. S1 to S9

Tables S1 to S18

References (50–71)

Submitted 19 June 2025; accepted 29 December 2025

Published online 22 January 2026

10.1126/science.adz9311

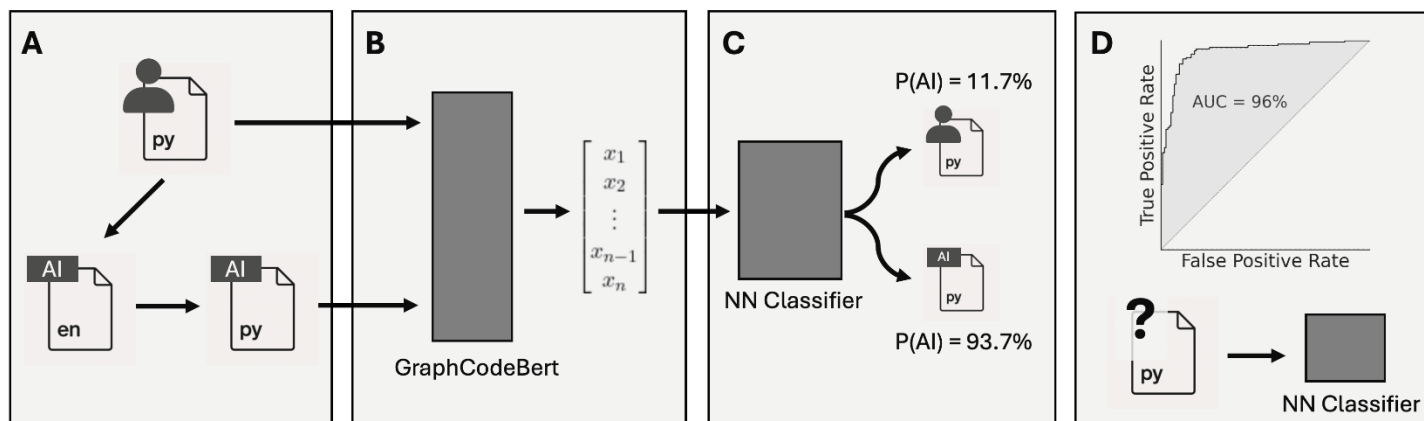


Fig. 1. Classifying code from functions written in the Python programming language as human or AI generated. (A) Using a collection of human generated code, we ask one LLM to describe the code in English, then another to implement that description as a Python function. (B) We vectorize the resulting code using GraphCodeBert, an embedding method that uses a code's tokens, comments, and variable graph flow. (C) We train a neural network classifier combining GraphCodeBert with a classification head to predict the human/AI labels. (D) We evaluate the classifier on out-of-sample data and apply it to a large database of unlabeled Python functions.

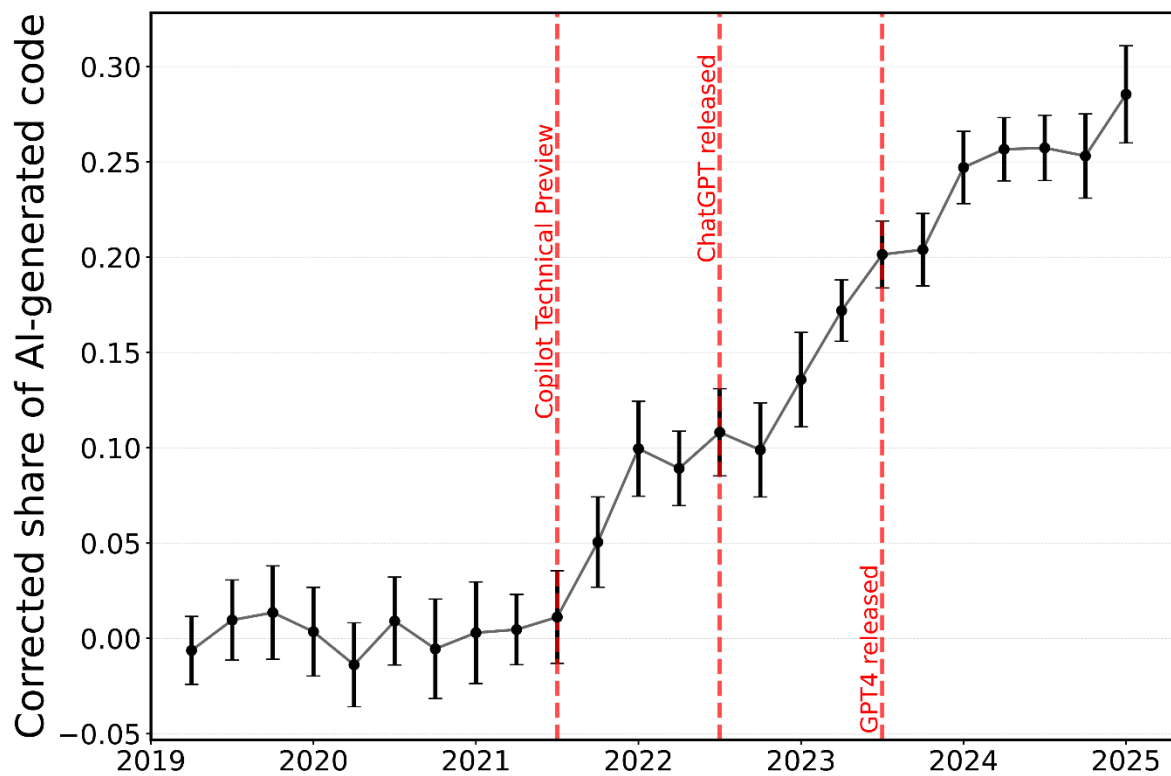
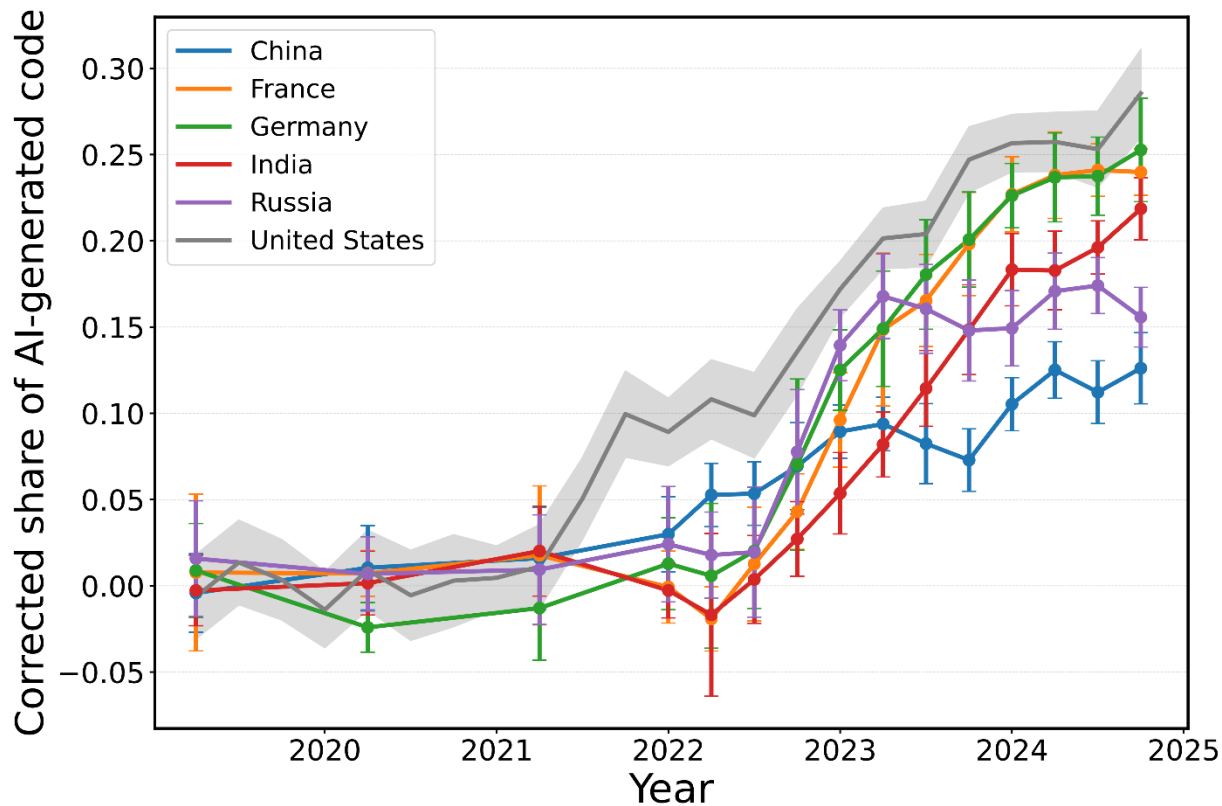
A**B**

Fig. 2. Share of AI-generated Python functions over time. (A) Share of Python functions that were created or substantially changed by GitHub users in the US. Vertical lines depict 95% confidence intervals. The plot reveals abrupt shifts in adoption coinciding with key AI-related events: the release of GitHub Copilot Preview, the public launch of ChatGPT, and the second wave of LLM releases (GPT4 and related models). (B) Adoption in China, France, Germany, India and Russia for which we sampled 2,000 random developers per country-year (note that in China, GitHub competes with the alternative collaboration platform, Gitee (39)). The US curve is replicated from (A) as a point of reference. The US led the early adoption of genAI, followed by European nations such as France and Germany. From 2023 onward, India rapidly catches up, whereas adoption in China and Russia progresses more slowly.

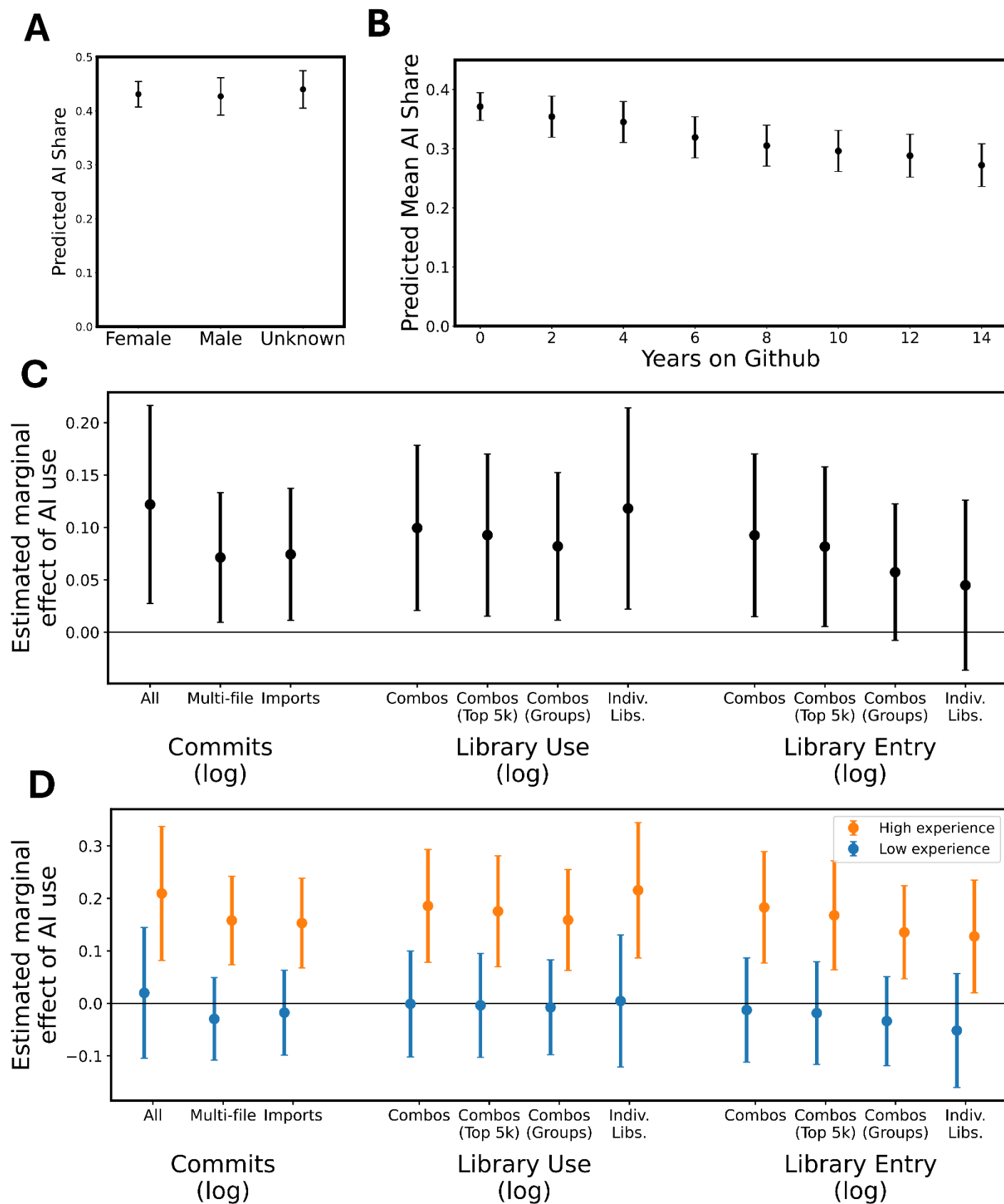


Fig. 3. Heterogeneity in adoption and effects. (A) Intensity of genAI use by gender inferred from GitHub user display names (US, 2024). (B) Intensity of genAI use by users' GitHub tenure (US, 2024). (C) Estimated effect of genAI use on user activity from a user-quarter panel regression with user and quarter fixed-effects. GenAI use is associated with increased commit activity across all commits, multi-file commits ("Multi-File") that navigate project interdependencies, and commits adding library imports ("Imports"), which we interpret as adding new features. GenAI is also associated with using wider ranges of libraries ("Indiv. Libs") and of library combinations ("Combos"), and increased experimentation with new libraries or combinations. Results are similar when subsetting on the 5,000 most common library combinations ("Combos (Top 5k)") and using coarsened library categories instead of libraries themselves ("Combos (Groups)"). (D) However, these benefits accrue entirely to experienced developers, with no measurable gains for inexperienced developers. Error bars: 95% confidence intervals (standard errors clustered by user).



Who is using AI to code? Global diffusion and impact of generative AI

Simone Daniotti, Johannes Wachs, Xiangnan Feng, and Frank Neffke

*Science***Ahead of Print** DOI: 10.1126/science.adz9311

View the article online

<https://www.science.org/doi/10.1126/science.adz9311>

Permissions

<https://www.science.org/help/reprints-and-permissions>

Use of this article is subject to the [Terms of service](#)

Science (ISSN 1095-9203) is published by the American Association for the Advancement of Science. 1200 New York Avenue NW, Washington, DC 20005. The title *Science* is a registered trademark of AAAS.

Copyright © 2026 The Authors, some rights reserved; exclusive licensee American Association for the Advancement of Science. No claim to original U.S. Government Works