BIS Working Papers
No 1208

Generative AI and labour productivity: a field experiment on coding

by Leonardo Gambacorta, Han Qiu, Shuo Shan and Daniel M Rees

Monetary and Economic Department

September 2024

JEL classification: D22, G31, R30

Keywords: Artificial intelligence, productivity, field experiment, big tech

This publication is available on the BIS website (www.bis.org).

ISSN 1020-0959 (print)
ISSN 1682-7678 (online)

Generative AI and labour productivity: A field experiment on coding

Leonardo Gambacorta, Han Qiu, Shuo Shan and Daniel M. Rees*

Abstract

In this paper we examine the effects of generative artificial intelligence (gen AI) on labour productivity. In September 2023, Ant Group introduced CodeFuse, a large language model (LLM) designed to assist programmer teams with coding. While one group of programmers used it, other programmer teams were not informed about this LLM. Leveraging this event, we conducted a field experiment on these two groups of programmers. We identified employees who used CodeFuse as the treatment group and paired them with comparable employees in the control group, to assess the impact of AI on their productivity. Our findings indicate that the use of gen AI increased code output by more than 50%. However, productivity gains are statistically significant only among entry-level or junior staff, while the impact on more senior employees is less pronounced.

JEL Codes: D22, G31, R30.
Keywords: artificial intelligence, productivity, field experiment, big tech.

[*] Leonardo Gambacorta (email: leonardo.gambacorta@bis.org) is with the Bank for International Settlements (BIS) and research fellows of CEPR. Han Qiu (email: han.qiu@bis.org) is with the BIS. Shuo Shan (email: shanshuo.ss@antgroup.com) is with Ant Group. Daniel M. Rees (daniel.rees@bis.org) is with the BIS. We thank Giulio Cornelli, Sebastian Doerr, participants at seminars held at the Bank for International Settlements and European Central Bank for useful comments and suggestions. The views in this paper are those of the authors only and do not necessarily reflect those of the Bank for International Settlements or Ant Group. The authors highlight that the data and analysis reported in this paper may contain errors and are not suited for the purpose of company valuation or to deduce conclusions about the business success and/or commercial strategy of Ant Group or other firms. All statements made reflect the private opinions of the authors and do not express any official position of Ant Group and its management. The authors declare that they have no relevant or material financial interests related to the research described in this paper. Shuo Shan discloses an employment relationship in Ant Group. Ant Group did not exercise any influence on the content of this paper. The authors acknowledge and thank the support from the Digital Economy Open Research Platform (www.deor.org.cn). All data were sampled and desensitized, and were analysed remotely on the Ant Open Research Laboratory in an Ant Group Environment, which is only remotely accessible for empirical analysis.

1. Introduction

Generative artificial intelligence (gen AI)[1] tools have the potential to enhance workers' productivity. A variety of gen AI models have demonstrated human-level capabilities in fields such as clinical care, education, language modelling, art, music and design (Gupta et al, 2024). While there is a growing literature studying commercial and non-commercial applications (Luo et al., 2022; Shaji George and Gabrio Martin, 2023), ethical considerations (Nassar and Camal, 2021; Shet and Baker, 2024), regulatory frameworks (Bradford, 2023; Aldasoro et al., 2024a), implication for security ((Sandoval et al., 2022; Aldasoro et al., 2024b) and education (Kasneci et al., 2023; Bahroun et al., 2023; Imran and Almusharraf, 2023), there has been little empirical research on productivity impacts of AI in tasks that requires cognitive abilities (Brynjolfsson and Raymond, 2023; Noy and Zhang, 2023; Peng et al, 2024).

To study the impact of gen AI on productivity, in this paper we leverage data from Ant Group, one relevant Chinese big tech. In September 2023, Ant Group unveiled CodeFuse, a large language model (LLM) designed to assist software programmer teams in coding. Prior to its widespread release, for an initial six-week trial period this LLM was accessible only to a select group of programmers. Leveraging on this event, we considered a field experiment on these two groups of programmers. We identified employees who used CodeFuse as the treatment group and paired them with comparable employees in the control group to assess the impact of AI on their productivity. [2]

The results suggests that LLMs can boost productivity for coders. Comparing programmers with similar productivity levels and work experience with and without access to the LLM shows a 55% increase in productivity (measured by the number of lines of code produced) on average. Roughly one third of this increase can be attributed to the code lines generated by the LLM directly, with the rest resulting from improved programmers' code efficiency elsewhere (likely reflecting additional time available for other programming tasks).

Dividing programmers by their experience levels revealed significant differences: productivity increased only among junior programmers. Comparing the number of requests and acceptance rate of LLM suggestions by workers with different experience sheds light on these differences as

[1] Gen AI encompasses a wide range of artificial intelligence technologies that specialize in synthesizing, or generating, content or data frequently indistinguishable from that produced by humans (Ebert and Louridas, 2023).

[2] This study was remotely conducted in the Ant Open Research Laboratory (https://www.deor.org.cn/labstore/laboratory). All data were sampled, desensitized, and analysed in this laboratory that is a sandbox environment where the authors can only remotely conduct empirical analysis, and individual observations are not visible.

senior programmers used the LLM by less. At the same time, the acceptance rate (the frequency at which programmers used the LLM's suggestions) did not vary with the level of experience. These findings suggest that the lower impact of the LLM on senior programmers' productivity stems from their lower engagement with the LLM rather than a lack of usefulness.

2. Structure of the field experiment

We conducted a controlled experiment to measure the productivity impact of using CodeFuse, an LLM introduced to assist with coding specific programmer teams. The experiment was conducted for a total of 12 weeks. It began on Monday September 4th, 2023, and ended on Friday November 25th 2023, right before CodeFuse became generally available to all programmers. For the experiment, we considered a *random sample* of 1,219 programmers. A treatment group of 335 programmers, belonging to two specific departments, was informed and received precise instructions about the new LLM from September 4th, 2023 to October14th and started to use the CodeFuse during this period. At the same time, we also analysed a control group of 884 programmers working in other departments that were not informed about the introduction of CodeFuse.

The *treatment group* and the *control group* were composed of programmers with a similar level of ex-ante experience and productivity. Table 1 presents the summary statistics of the treatment group (panel A), the control group (panel B), and the whole sample (panel C). Programmer characteristics that are time-invariant were recorded at the beginning of September, before the introduction of CodeFuse, and are very similar when comparing the two groups. We consider four level of professionals, from P5, at the first stage of their career, to P8, who are the most experienced programmers in our sample. The two groups are quite similar also in terms of years of experience. For example, senior economists - those with more than one year of experience - represented 40% of the treatment group and 44% of the control group.

At the bottom of the table, we also report the productivity levels of the two groups before and after the introduction of the LLM. Data were collected every two weeks, for a total of *7 observations:* the three two-week periods before the LLM became available, the two-week period of its introduction, and the three two-week periods after its release to the treated group. Our key dependent variable is the *volume of lines of code*, which is an important indicator of work output for programmers in a tech company. Considering the six weeks before the introduction of CodeFuse, the treatment and the control groups registered a very similar number of lines of output code (6.46 vs 6.44, in logs, with standard errors of 2.61 and 2.56, respectively).[3] However, when comparing the eight weeks after the introduction of the CodeFuse, we detected a 33% increase in productivity of the control group, from 6.46 to 6.79, and a 20% reduction in the treated group, from 6.44 to 6.24. This reduction was largely expected because of a one-week public holiday in China at the beginning of October. In relative terms, the overall increase in productivity of the treated group versus the control group was around 53%.

We also had access to task-based data for each staff request to the LLM. Specifically, we could track whether each programmer in the treated group used CodeFuse, when they started using it, and qualitative indicators of their use. This aspect is crucial because the generative AI application proposes lines of code that a programmer can either accept or decline. Therefore, only if a programmer accepts the lines does it indicate satisfaction with the AI's suggestions. We tracked whether the LLM's suggestions were accepted or rejected by using an AI application that compared the suggested lines with the final output. We employed two different acceptance rate metrics: the number of lines accepted out of the total suggested and, on a more granular level, the number of words accepted out of the total.

3. Results

To rule out the possibility that selection bias in the treatment and control groups may influence our results, we use propensity score matching combined with a difference-in-differences analysis. First, we average selected programmers' characteristics in the period before the introduction of CodeFuse (pre-treatment period) and use the log of the total lines of code, professional level, and years of experience to predict the probability of being treated. Finally, we match each programmer in the treated group with one or more programmers in the control group who have the closest propensity score, which reflects the same probability of being treated.[4]

The main results of the analysis are represented in the left panel of Graph 1. As we express the variable in logs what is reported in the vertical axis of this graph can be interpreted as a growth rate in the number of code lines produced. As discussed above we report on the x axis the 7 two weeks periods: three two-weeks periods prior to the LLM introduction, the two weeks of the

[3] We express the number of lines of output code in logs so that their first difference, or the value of the coefficients in the regression, can approximate growth rates.

[4] Matching is done using a Nearest Neighbor approach with a conservative Caliper equal to 0.0001. Finally, the matching is done with replacement, so that there is more than one match between a firm in the treatment with a firm in the control group.

introduction (represented by 0) and the three subsequent two-week periods (for a total of another 6 weeks) after the introduction of the CodeFuse.

Our findings show that the use of the *LLM increased code output by 55%* on average, and that this positive effect remains quite stable over time. This result is qualitatively similar to other findings in the literature. For example, Peng et al. (2024) found that the introduction of GitHub Copilot, an AI pair programmer, made the treatment groups 56% faster in completing tasks than the control group.

One important question to answer is also *how does generative AI impact productivity?* There is an obvious, mechanical, and direct impact due to the extra lines of code that programmers obtain from the LLM upon specific requests. But there is also an indirect effect because generative AI can help programmers save time, organize their work better, and have more free time to dedicate to more complex and creative tasks. We have therefore examined the lines of codes (and words therein) produced by programmers to determine the number of lines/words that have been directly derived from the LLM and "approved" by the programmer. The results reported in the right panel of Graph 1 indicate that direct LLM output contributes only to a 11%-18% increase in productivity (depending on if we consider code lines or words). This means that the remaining 36-43% increase in productivity is probably due to the time saving, as AI can assist programmers in overcoming bottlenecks during coding.

In a second step we also analysed the potential *heterogeneous effects of gen AI among programmers with different levels of experience*. To this end, we developed a complementary test by separating junior and senior programmers. Specifically, junior programmers have one year or less of work experience, while senior programmers have more than one year of experience. The results of this test, reported in Graph 2, show that the positive effect of gen AI on productivity is especially marked among entry-level or junior staff, with a 67% increase in code volume. In contrast, the impact on more senior employees is positive but not statistically significant.

Comparing the *number of requests and acceptance rate of LLM suggestions* by workers with different levels of experience sheds light on these differences. The red line in the left panel of Graph 3 indicates a negative correlation between the volume of requests following the LLM's introduction and the programmers' years of experience. Specifically, the x-axis represents the level of work experience, ranging from less than one year to five or more years. The y-axis (left scale) shows the number of requests per user after the introduction of CodeFuse. The red line indicates that the total number of requests per users dramatically declines with years of experience. It is around 900 for programmers with less than two years of experience over the 8-week period, which averages to approximately 110 per week and 19 per day. For more experienced programmers (with more than 5 years), the number is roughly half, around 450, which averages to about 55 per week and 9 per day. Therefore, senior programmers used the LLM less.

4. Why do senior programmers benefit less from CodeFuse?

To better investigate why senior programmers benefit less from CodeFuse, we have moved from a diff-in-diff approach to a more standard OLS analysis on the treatment group of 335 programmers. We also extended the analysis to a total of 10 observations, spanning therefore a period of 20 weeks after the introduction of CodeFuse. The results presented in Table 2 confirm the diff-in-diff analysis. We use both the senior programmer dummy (column I) and a more granular measure of seniority given by the number of working years (column II). In both cases, we detect a negative correlation, between the number of lines of code requested via the CodeFuse gen AI application and the programmer’s experience. From the first column, we can see that senior programmers solicit AI advice approximately 70% less frequently than junior programmers. From the second column, we observe that help requests decline with years of experience. With each additional year of programming experience, the reliance on AI-generated code advice decreases by 13%.

At the same time, the acceptance rate (the frequency at which programmers agreed with the LLM’s suggestions) did not vary with the level of experience. The blue line indicates that between 15 to 20% of the time (right scale) , programmers, regardless of their seniority, used the suggestions of CodeFuse. These findings suggest that the lower impact of the LLM on senior programmers’ productivity stems from their lower engagement with the LLM rather than a lack of usefulness.

In Table 3, we consider the evolution of the acceptance rates for different tasks, resulting in a larger number of observations compared to the previous regression. In columns (I) and (III), we consider all tasks separately, while in columns (II) and (IV), we include a specific dummy variable that takes the value of 1 for complex tasks and 0 otherwise. We define complex tasks as those requiring the writing of a complete program paragraph rather than a one-line command. As done in Table 2, we regress the acceptance rate on a senior programmer dummy in columns (I)-(II) and the number of working years in columns (III)-(IV).

We also include two additional controls: (i) reading time and (ii) generated text. Reading time is the time needed for the programmer to read the CodeFuse suggestion, normalized by the number of rows in the specific programme. Specifically, it is the time from when the programmer receives the code suggestion to when they decide to accept or reject it. Generated text measures how many words the AI helped to write in the program code.

Acceptance rate analysis reveals minimal differences between junior and senior programmers. We detect only a 2% lower acceptance rate for senior programmers. Moreover, when we consider a more granular measure of work experience (number of working years), the difference is not statistically significant and the acceptance rate is similar, consistent with what we observed in the graphical analysis.

The right-hand panel of Graph 3 reports the acceptance rates for simple and more creative tasks. Simple tasks (red histograms) refer to those that can be instructed with a one-line command, while complex tasks (blue histograms) require several lines of prompt commands and necessitate more creativity. Interestingly, we find that users' acceptance rates for more creative tasks are only slightly lower than for simple tasks, suggesting that GenAI performs quite similarly in both types of tasks. This result is also confirmed by the regression in Table 3: the acceptance rate for complex tasks is only 2.1-3.1% lower than for simpler ones.

In a final test, we check senior programmers' level of attention to CodeFuse suggestions. Specifically, in Table 4, we examine how reading time (per unit of line of suggestions) correlates with the senior programmer dummy (column I) and the number of working years (column II). Each observation corresponds to a task. The results show no substantial differences in attention levels among programmers of varying experience.

5. Conclusions

In this paper, we have examined the effects of gen AI on labour productivity through a field experiment. In September 2023, Ant Group introduced CodeFuse, a LLM designed to assist programmer teams with coding. We compared a treatment group of programmers using CodeFuse with a control group that was not informed about the LLM. Our findings indicate that gen AI significantly increased code output by over 50%. However, these productivity gains are statistically significant only among entry-level or junior staff. The impact on more senior employees was not statistically significant, primarily because senior programmers used gen AI less frequently than their junior counterparts. Despite this, when senior programmers did use gen AI, they paid close attention to the suggestions and found them useful. This suggests that while gen AI can enhance productivity, its adoption and usage patterns vary significantly with experience levels, highlighting the need for targeted strategies to maximize its benefits across different seniority levels.

References

Aldasoro, I., L. Gambacorta, A. Korinek, V. Shreeti and M. Stein (2024a), "Intelligent financial system: how AI is transforming finance", BIS Working Papers, 1194.

Aldasoro, I., S. Doerr, L. Gambacorta, S. Notra, T. Oliviero and D. Whyte (2024b), "Generative artificial intelligence and cybersecurity in central banking", BIS Papers, 145.

Bahroun, Z., C. Anane, V. Ahmed, and A. Zacca, (2023), "Transforming education: A comprehensive review of generative artificial intelligence in educational settings through bibliometric and content analysis", Sustainability, 15(17).

Bradford, A. (2023), Digital empires: The global battle to regulate technology, Oxford University Press, 2023.

Brynjolfsson, E, D Li, and L Raymond (2023), "Generative AI at work", NBER Working Paper, 31161.

Ebert, C., and P. Louridas (2023), "Generative AI for software practitioners". IEEE Software, 40(4), 30–38.

Gupta, P., B. Ding, C. Guan, and D. Ding (2024), "Generative AI: A systematic review using topic modelling techniques", Data and Information Management, February.

Imran, M., and N. Almusharraf (2023), "Analyzing the role of ChatGPT as a writing assistant at higher education level: A systematic review of the literature", Contemporary Educational Technology, 15(4).

Kasneci, E., et al. (2023), "ChatGPT for good? On opportunities and challenges of large language models for education", *Learning and Individual Differences*, 103.

Luo, B., R. Y. K. Lau, C. Li, , and Y.-W. Si, (2022), "A critical review of state-of-the-art chatbot designs and applications", WIREs Data Mining and Knowledge Discovery, 12(1).

Nassar, A. and M. Kamal (2021), "Ethical Dilemmas in AI-Powered Decision-Making: A Deep Dive into Big Data-Driven Ethical Considerations", International Journal of Responsible Artificial Intelligence, 11(8), 1–11.

Noy, S, and W Zhang (2023): "Experimental evidence on the productivity effects of generative artificial intelligence." Science, 187-192.

Peng, S., W Swiatek, A Gao, P Cullivan, and H Chang, H (2024): "AI Revolution on Chat Bot: Evidence from a Randomized Controlled Experiment", arXiv preprint arXiv:2401.10956.

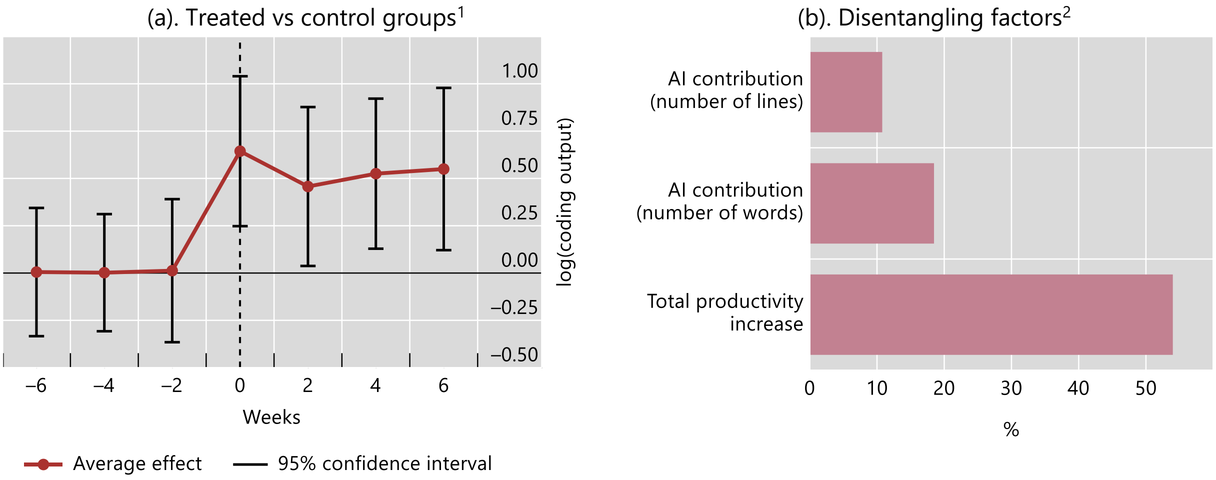
Sandoval, G., H. Pearce, , T. Nys, , R. Karri, , B. Dolan-Gavitt, , and Garg S. (2022), "Security implications of large language model code assistants: A user study", arXiv:2208.09727.

Shaji George, A. S. H. G. A., and A. S. Gabrio Martin (2023), "A review of ChatGPT AI's impact on several business sectors", Partners Universal International Innovation Journal, 1(1).

Sheth, S. and H.P. Baker (2024), "Ethical Considerations of Artificial Intelligence in Health Care: Examining the Role of Generative Pretrained Transformer-4", Journal of the AAOS, 32(5), 205-210.

Impact of gen AI on labour productivity

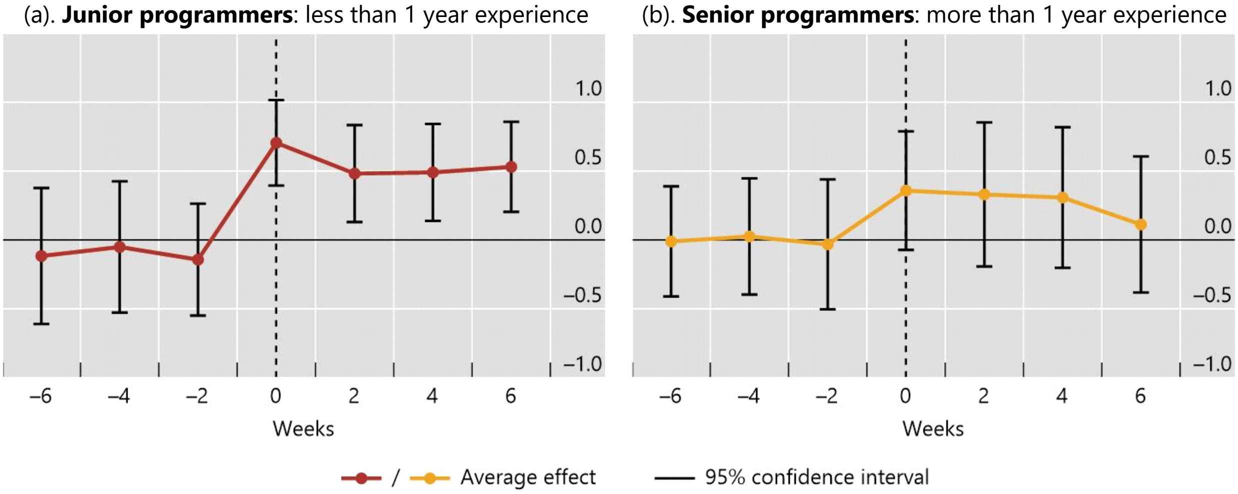
Graph 1



Note: [1] Based on a difference-in-differences analysis to evaluate the effects on labour productivity between two groups of programmers: those with access to CodeFuse (treatment group) and those without (control group). The comparison spans 6 weeks prior to the introduction of CodeFuse (with time 0 marking the introduction) and 6 weeks afterwards. The y-axis approximates the growth rate in the number of lines of code produced (in logarithm). [2] This panel reports the productivity increase in terms of: (i) the number of lines of code directly suggested by CodeFuse, (ii) the number of words suggested by CodeFuse and used in code, and (iii) the total productivity in terms of lines of code.

Source: Authors' calculations.

Effect of generative AI on productivity for different levels of work experience Graph 2

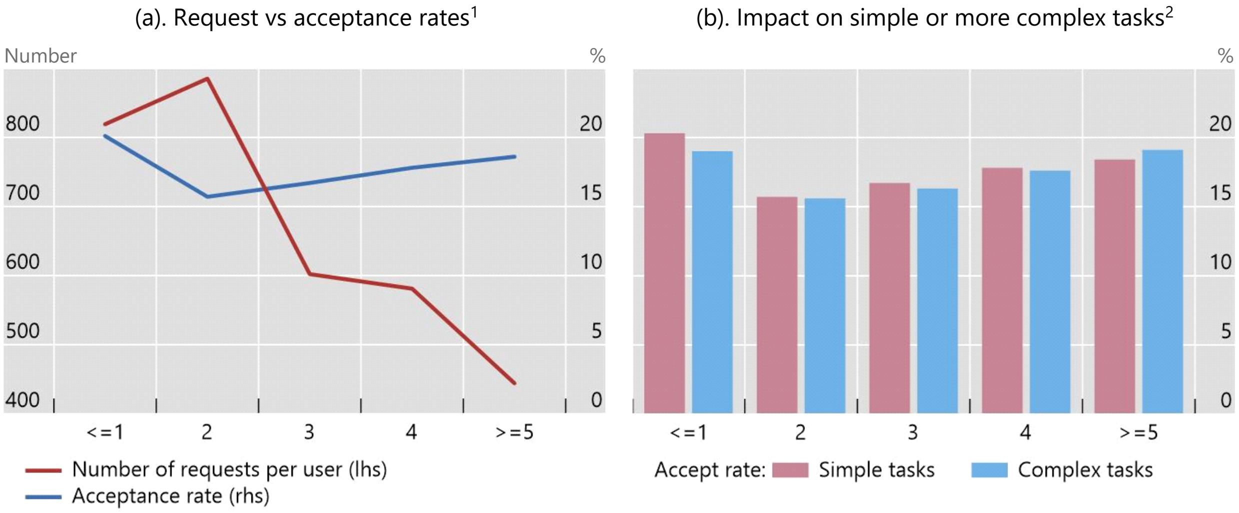


Note: Based on a difference-in-differences analysis to evaluate the effects on labour productivity between two groups of programmers: those with access to CodeFuse (treatment group) and those without (control group). The comparison spans 6 weeks prior to the introduction of CodeFuse (with time 0 marking the introduction) and 6 weeks afterwards. The y-axis approximates the growth rate in the number of lines of code produced (in logarithm). Junior programmers (left panel) are defined as those with up to one year of experience. Senior programmers (right panel) have more than one year of experience.

Source: Authors' calculations.

Why gen AI is not useful for senior programmers?

Graph 3



(a). Request vs acceptance rates[1]

(b). Impact on simple or more complex tasks[2]

Note: [1] The number of requests per user is determined by the average number of times a programmer, categorised by years of work experience, has requested assistance from the LLM in the weeks of the introduction of CodeFuse (marked by 0) and 6 weeks afterwards. The acceptance rate represents the proportion of these requests for which a programmer has accepted the suggestions offered by the LLM application with less than 50% human modification. [2] Simple tasks refer to those tasks that can be instructed with a one line command. Complex tasks need several lines of prompt command and require more creativity.

Source: Authors' calculations.

Table 1: Summary statistics

Variables	**A. Treatment group**					**B. Control group**					**C. All sample**				
	Obs	Mean	Std. Dev.	Min	Max	Obs	Mean	Std. Dev.	Min	Max	Obs	Mean	Std. Dev.	Min	Max
Lines of output code (log)	2,345	6.65	2.36	0	11.31	6,188	6.33	2.55	0	12.01	8,533	6.42	2.51	0	12.01
Working years	2,345	1.57	1.99	0	12	6,188	1.69	1.85	0	12	8,533	1.65	1.89	0	12
Job level:															
P5	2,345	0.39	0.49	0	1	6,188	0.35	0.48	0	1	8,533	0.36	0.48	0	1
P6	2,345	0.19	0.39	0	1	6,188	0.19	0.39	0	1	8,533	0.19	0.39	0	1
P7	2,345	0.20	0.40	0	1	6,188	0.22	0.42	0	1	8,533	0.22	0.41	0	1
P8	2,345	0.21	0.41	0	1	6,188	0.24	0.42	0	1	8,533	0.23	0.42	0	1
Senior programmer	2,345	0.40	0.49	0	1	6,188	0.44	0.50	0	1	8,533	0.43	0.50	0	1
Number of programmers	335					884					1219				
%	*27.5*					*72.5*					*100.0*				
	Six weeks before the introduction of CodeFuse														
Lines of output code (log) (a)	1,005	6.46	2.61	0	11.31	2,652	6.44	2.56	0	11.89	3,657	6.45	2.58	0	11.89
	Eight weeks after the introduction of CodeFuse														
Lines of output code (log) (b)	1,340	6.79	2.15	0	11.24	3,536	6.24	2.54	0	12.01	4,876	6.39	2.45	0	12.01
Change (b)-(a)		0.33					-0.20					-0.06			

Table 2 Senior programmers use less gen AI

Explanatory variables:	Dependent variable: Log (AI code lines requested)	
	(I)	(II)
Log (Total code line programme)	0.471***	0.466***
	(0.028)	(0.029)
Senior programmer (0/1 dummy)	**–0.643*****	
	(0.181)	
Number of working years		**–0.093*****
		(0.044)
Constant	–1.639***	–1.545***
	(0.228)	(0.257)
Adj-R2	0.259	0.259
Number of observations	3,350	3,350

Notes: Standard errors in brackets. Significance level: *p<0.1; ** p<0.05; *** p<0.01.

Table 3: When experienced programmers use gen AI, they find it useful

Explanatory variables:	Dependent variable: Acceptance rate			
	(I)	(II)	(III)	(IV)
Complex tasks[1]		–0.021***	–0.021**	–0.031***
		(0.004)	(0.004)	(0.003)
Reading time[2]				–0.000001***
				(0.000)
Generated text[3]				0.00014***
				(0.000)
Senior programmer (0/1)	–0.019**	–0.019**		
	(0.007)	(0.007)		
Number of working years			–0.002	–0.003
			(0.002)	(0.002)
Time* fixed effects[4]	Yes	Yes	Yes	Yes
Programming language fixed effects	Yes	Yes	Yes	Yes
Adj-R2	0.03	0.03	0.03	0.03
Number of observations	561,320	561,320	561,320	481,763

Notes: [1] Dummy variable for complex tasks that need several lines of prompt command and require more creativity. [2] Reading time is the time needed for the programmer to read the CodeFuse suggestion, normalized by the number of rows in the specific programme. Specifically, it is the time from when the programmer receives the code suggestion to when he decides to accept or reject it. [3]Generated text measures how many words the AI helped to write in the program code. [4] Time fixed effect for the first time the user starts to use CodeFuse, which can control for the user's experience with the Gen AI application. Standard errors in brackets. Significance level: *p<0.1; ** p<0.05; *** p<0.01.

Table 4 When seniors use gen AI they pay attention to suggestions

Explanatory variables	Dependent variable: Reading time[1]	
	(I)	(II)
Complex tasks[2]	-50.60	-49.30
	(61.30)	(61.36)
Senior programmer (0/1)	38.82	
	(180.59)	
Number of working years		-39.85
		(63.06)
Time*first time use fixed effects[3]	Yes	Yes
Programming language fixed effects	Yes	Yes
Adj-R2	0.11	0.11
Observations	561,320	561,320

Notes: [1] Reading time is the time needed for the programmer to read the CodeFuse suggestion, normalized by the number of rows in the specific programme. Specifically, it is the time from when the programmer receives the code suggestion to when he decides to accept or reject it. [2] Dummy variable for complex tasks that need several lines of prompt command and require more creativity. [3] Time fixed effect for the first time the user starts to use CodeFuse, which can control for the user's experience with the Gen AI application. Standard errors in brackets. Significance level: *p<0.1; ** p<0.05; *** p<0.01.

Previous volumes in this series

1207 September 2024	The rise of generative AI: modelling exposure, substitution and inequality effects on the US labour market	Raphael Auer, David Köpfer, Josef Švéda
1206 August 2024	Covered interest parity: a forecasting approach to estimate the neutral band	Juan R. Hernández
1205 August 2024	The Measure Matters: Differences in the Passthrough of Inflation Expectations in Colombia	Andres Sanchez-Jabba and Erick Villabon-Hinestroza
1204 August 2024	Climate Policies, Labor Markets, and Macroeconomic Outcomes in Emerging Economies	Alan Finkelstein Shapiro and Victoria Nuguer
1203 August 2024	Strike while the Iron is Hot: Optimal Monetary Policy with a Nonlinear Phillips Curve	Peter Karadi, Anton Nakov, Galo Nuno, Ernesto Pasten, and Dominik Thaler
1202 August 2024	Are low interest rates firing back? Interest rate risk in the banking book and bank lending in a rising interest rate environment	Lara Coulier, Cosimo Pancaro and Alessio Reghezza
1201 July 2024	Crypto Exchange Tokens	Rodney Garratt, Maarten R.C. van Oordt
1200 July 2024	Financial inclusion transitions in Peru: does labor informality play a role?	Jose Aurazo and Farid Gasmi
1199 July 2024	New spare tires: local currency credit as a global shock absorber	Stefan Avdjiev, John Burger and Bryan Hardy
1198 July 2024	Sovereign green bonds: a catalyst for sustainable debt market development?	Gong Cheng, Torsten Ehlers, Frank Packer and Yanzhe Xiao
1197 July 2024	The gen AI gender gap	Iñaki Aldasoro, Olivier Armantier, Sebastian Doerr, Leonardo Gambacorta and Tommaso Oliviero
1196 July 2024	Digital payments, informality and economic growth	Ana Aguilar, Jon Frost, Rafael Guerra, Steven Kamin and Alexandre Tombini
1195 July 2024	The asymmetric and persistent effects of Fed policy on global bond yields	Tobias Adrian, Gaston Gelos, Nora Lamersdorf, Emanuel Moench
1194 June 2024	Intelligent financial system: how AI is transforming finance	Iñaki Aldasoro, Leonardo Gambacorta, Anton Korinek, Vatsala Shreeti and Merlin Stein

All volumes are available on our website www.bis.org.