

# Comparing AI Coding Agents: A Task-Stratified Analysis of Pull Request Acceptance

Giovanni Pinna  
University of Trieste  
Trieste, Italy  
giovanni.pinna@phd.units.it

Jingzhi Gong  
King's College London  
London, UK  
jingzhi.gong@kcl.ac.uk

David Williams  
University College London  
London, United Kingdom  
david.williams.22@ucl.ac.uk

Federica Sarro  
University College London  
London, United Kingdom  
f.sarro@ucl.ac.uk

## Abstract

The rapid adoption of AI-powered coding assistants is transforming software development practices, yet systematic comparisons of their effectiveness across different task types and over time remain limited. This paper presents an empirical study comparing five popular agents (OpenAI Codex, GitHub Copilot, Devin, Cursor, and Claude Code), analyzing 7,156 pull requests (PRs) from the AIDev dataset. Temporal trend analysis reveals heterogeneous evolution patterns: Devin exhibits the only consistent positive trend in acceptance rate (+0.77% per week over 32 weeks), whereas other agents remain largely stable. Our analysis suggests that the PR task type is a dominant factor influencing acceptance rates: documentation tasks achieve 82.1% acceptance compared to 66.1% for new features—a 16 percentage point gap that exceeds typical inter-agent variance for most tasks. OpenAI Codex achieves consistently high acceptance rates across all nine task categories (59.6%–88.6%), with stratified Chi-square tests confirming statistically significant advantages over other agents in several task categories. However, no single agent performs best across all task types: Claude Code leads in documentation (92.3%) and features (72.6%), while Cursor excels in fix tasks (80.4%).

## CCS Concepts

- Software and its engineering → Collaboration in software development.

## Keywords

GenAI, LLM4Code, Code Review, Performance Evolution, AI4SE

### ACM Reference Format:

Giovanni Pinna, Jingzhi Gong, David Williams, and Federica Sarro. 2026. Comparing AI Coding Agents: A Task-Stratified Analysis of Pull Request Acceptance. In *23rd International Conference on Mining Software Repositories (MSR '26)*, April 13–14, 2026, Rio de Janeiro, Brazil. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3793302.3793583>

## 1 Introduction

The Software Engineering (SE) landscape has undergone a significant transformation with the emergence of AI-powered coding assistants [7, 11, 13]. Tools such as GitHub Copilot [26], OpenAI Codex [3], and autonomous agents like Devin have moved beyond simple code completion to generating entire functions, fixing bugs,



This work is licensed under a Creative Commons Attribution 4.0 International License.  
MSR '26, Rio de Janeiro, Brazil  
© 2026 Copyright held by the owner/author(s).  
ACM ISBN 979-8-4007-2474-9/2026/04  
<https://doi.org/10.1145/3793302.3793583>

and creating pull requests autonomously [5]. Li et al. [16] characterize this transition as “Software Engineering 3.0,” defined by autonomous AI teammates capable of completing complex development activities with minimal human supervision.

This phenomenon raises several fundamental questions: *How do different AI coding agents compare when deployed in real-world software development workflows? What factors influence their effectiveness? Do outcome metrics vary across agents, task types, and time?* These questions have implications for three key stakeholder groups: practitioners selecting tools for their teams; tool developers prioritizing capability improvements; and researchers designing robust evaluation methodologies for AI-assisted software engineering.

Moreover, open challenges remain regarding methodological shortcomings in LLM performance evaluations [24], including the fact that global performance metrics may fail to distinguish between agent capability and task assignment. For instance, an agent that handles mostly documentation tasks (which tend to have higher acceptance rates according to our results) may appear superior to one that handles complex feature implementations, even if the latter has stronger underlying capabilities. This confounding effect has been documented in past observational studies of SE practices [14], where uncontrolled variables can lead to misleading conclusions.

In this paper, we tackle some of the aforementioned questions by examining temporal trends to understand how AI coding agents perform over extended deployment periods, remaining agnostic about underlying mechanisms such as model updates, user behavior changes, or task distribution shifts. Additionally, we address workload heterogeneity through *task-stratified comparisons*, evaluating agents within each task category (e.g., feature, fix, documentation) to control for confounding factors in different task distributions. Specifically, we answer the following three research questions by analyzing the AIDev dataset [15, 17]:

**RQ1: Does AI coding agent performance evolve over time?** We investigate whether agents demonstrate measurable changes in acceptance rates over extended observation periods.

**RQ2: What factors are associated with performance?** We examine how task type, review frequency, and other factors correlate with acceptance rates, and whether differences in task distribution across agents confound global comparisons.

**RQ3: How do different agents compare?** We perform task-stratified statistical comparisons of five major AI coding agents to identify differences in performance.

Our contributions include: (i) temporal analysis revealing heterogeneous evolution patterns across agents, with Devin showing the only consistent improvement; (ii) a task-stratified comparison methodology that controls for workload heterogeneity when evaluating AI agents; (iii) empirical evidence that task type is a dominant

performance factor, with a 29 percentage point (pp) gap between task types; and (iv) analysis of agent-specific strengths across task types, showing that no single agent outperforms others in all tasks.

## 2 Background and Related Work

The technological evolution of LLM-based development tools has been rapid. OpenAI Codex launched in August 2021 [3], powering GitHub Copilot, which reached general availability in June 2022 [9]. Since then, we have witnessed the advent of autonomous agents: Devin by Cognition AI (March 2024) achieved 13.86% on SWE-bench unassisted, a substantial improvement over the prior 1.96% state-of-the-art [5]; Cursor grew to over one million users by 2025 [2]; and Claude Code debuted in February 2025 as Anthropic's terminal-based agentic coding tool [1]. Research on productivity impacts presents mixed findings: Peng et al. [21] found Copilot users completed tasks 55.8% faster, yet METR [18] observed experienced developers were 19% slower, despite believing otherwise. Murillo et al. [19] reported transient velocity gains (281% increase collapsing to baseline by month three) with persistent quality degradation (30% more static analysis warnings). Yetistiren et al. [25] found that only 28.7% of Copilot suggestions were directly usable, with quality varying by language and task complexity. Security vulnerabilities in AI-assisted code have also been documented [22]. Finally, He et al. [12] conducted a longitudinal study of Cursor adoption, finding that velocity gains were concentrated in the first two months before returning to baseline, while technical debt—measured through static analysis warnings and code complexity—accumulated persistently. These findings reinforce concerns about using acceptance rate as a sole quality proxy, as short-term acceptance may mask long-term maintenance burdens.

## 3 Methodology

### 3.1 Dataset and Preprocessing

We utilize the AIDev dataset [15], which contains AI-generated pull requests from GitHub repositories with at least 100+ stars (the AIDev-POP subset). The raw dataset contains 33,596 PRs, along with corresponding file-level changes, reviews, comments, and LLM-classified task categories. From this data, we derive a filtered dataset of 7,156 PRs by applying the following quality criteria: We retain ① *closed* PRs from ② repositories with *permissive licenses* (MIT or Apache-2.0) and ③ required that each PR receive *at least one review or comment from someone other than the creator before closure* (ensuring meaningful evaluation). Before analysis, we aggregate commit- and review-level data to the PR level, computing additions, deletions, files changed, commits, reviews, and comments per PR. For temporal analysis, we group PRs by the week of their creation date; an *active week* is any week containing at least one PR from a given agent. Table 1 summarizes the resulting agent distribution. The “Start” column reflects dataset coverage rather than official releases; “Weeks” reports the number of active weeks containing at least one PR. Cursor’s 13 active weeks span May–August 2025, representing a concentrated adoption period compared to Devin’s extended 32-week observation window. These temporal differences have two implications: agents cannot be compared across equivalent maturity periods, and observed patterns may reflect deployment contexts

**Table 1: Agent PR distributions, active weeks, and overall acceptance rates in the filtered dataset.**

Agent	Start	PRs	Weeks	PRs/Wk	Acc. Rate
Devin	12/24/24	2,252	32	70.4	61.6%
OpenAI Codex	05/16/25	2,002	12	166.8	77.9%
GitHub Copilot	05/19/25	2,194	11	199.5	68.0%
Cursor	05/01/25	569	13	43.8	74.5%
Claude Code	02/24/25	139	19	7.3	71.9%
<b>Total</b>		<b>7,156</b>	<b>87</b>	—	<b>69.3%</b>

rather than intrinsic capabilities. We address this limitation through sensitivity analysis using aligned time windows in Section 5.

### 3.2 Metrics and Statistical Methods

We characterize success outcomes through *acceptance rate*, defined as the proportion of merged PRs (those with a non-null `merged_at` timestamp). Since our filtering retains only closed PRs (Section 3.1), the acceptance and rejection rates sum to 100% in our analysis. We define *task-stratified observations* as unique (agent, task type, week) combinations where at least one PR was submitted. Summing across all agents and the 12 task types yields 552 total task-stratified observations (Table 2). To address RQ1, we measure *temporal trends* in acceptance rate per agent by fitting linear regression models defined as  $y = \beta_0 + \beta_1 \cdot t + \epsilon$ , where  $\beta_1$  represents the weekly rate of change. A slope of +0.01 would indicate a one-percentage-point weekly increase. We report  $R^2$  values to characterize model fit, with values closer to 1 indicating stronger linear trends and values near 0 suggesting no consistent temporal pattern. To capture non-linear patterns, we apply LOESS (Locally Estimated Scatterplot Smoothing) with a fraction parameter  $frac = 0.5$  [4], balancing smoothness against sensitivity to local trends. For RQ2, we examine task-level acceptance rates and *review frequency*, defined as the number of reviews per PR (indicating the extent of evaluation each agentic contribution undergoes). For RQ3, we conduct task-stratified pairwise agent comparisons using Pearson’s Chi-square tests of independence [20]. To avoid over-interpreting sparse observations, we exclude comparisons where at least one agent has fewer than 10 total PRs for a given task. We additionally exclude three task types (`style`, `revert`, and `other`) because multiple agents have zero observations for these categories in our filtered dataset. With these criteria, comparisons across the 10 possible agent pairs and 9 remaining task types yield 64 task-stratified tests. For comparisons where expected cell frequencies fall below 5, we apply Fisher’s exact test [8] (17 of 64 comparisons). To control for multiple comparisons, we apply the Bonferroni correction across the executed test set, yielding an adjusted significance threshold of  $\alpha = 0.05/64 \approx 0.00078$ . In addition, we report the phi coefficient ( $\phi$ ) [6] to measure effect size, following standard interpretation thresholds:  $\phi < 0.1$  (negligible),  $0.1 \leq \phi < 0.3$  (small),  $0.3 \leq \phi < 0.5$  (medium), and  $\phi \geq 0.5$  (large).

Finally, we emphasize that this study is observational: we document patterns without claiming causality, acknowledging potential confounders such as repository concentration, user population dynamics, model updates, and shifts in task distributions.

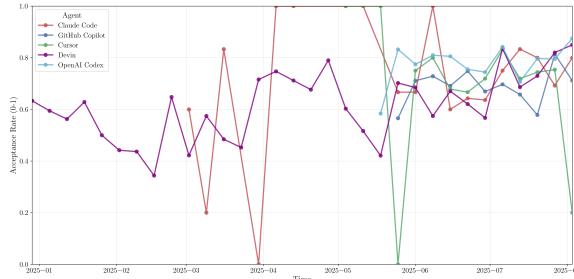


Figure 1: RQ1. Acceptance rate over time per agent.

Table 2: RQ2. Acceptance rate by task type. MAR: Mean acceptance rate; SD: standard deviation; Obs: # weekly agent-task observations.

	chore	docs	style	ci	build	refactor	feat	fix	test	revert	perf	other
MAR	84.0%	82.1%	78.1%	75.0%	72.5%	71.2%	66.3%	66.0%	61.5%	60.0%	55.4%	0.0%
SD	36.7%	38.4%	42.0%	43.5%	44.8%	45.3%	47.3%	47.4%	48.8%	54.8%	50.2%	0.0%
Obs	58	71	25	41	40	69	81	77	53	4	29	4

## 4 Results

### 4.1 RQ1: Performance Over Time

Figure 1 illustrates heterogeneous performance patterns across agents. Devin exhibits a positive trend in acceptance rate (+0.77% per week,  $R^2 = 0.34$ ), increasing from approximately 60% to 80%. Devin’s LOESS curve suggests three phases: an initial phase (~60%), an increasing phase (to ~80%), and a stabilization phase—though we cannot isolate underlying causal mechanisms. Notably, while Devin shows improvement, its weekly variance remains high, suggesting inconsistent performance despite the positive trend. In contrast, OpenAI Codex and GitHub Copilot exhibit plateau behavior, with consistently high acceptance rates from their first observed week.

**RQ1:** Devin is the only agent showing consistent performance improvement over time, while other agents maintain stable acceptance rates.

### 4.2 RQ2: Factors Associated with Performance

Table 2 reports acceptance rates calculated at the task level (i.e., merged PRs divided by total PRs for each task type). From these results, task type emerges as a dominant factor in agentic PR outcomes. Table 2 shows a 29pp gap between the highest (chore: 84.0%) and lowest (perf: 55.4%) acceptance rates. Among high-volume task types, documentation achieves 82.1% versus 66.1% for features—a 16 percentage-point gap. This pattern suggests that AI agents achieve higher acceptance on well-structured tasks, though alternative explanations exist (e.g., less stringent review standards for documentation PRs). Beyond task type, we observe variation in review frequency across agents. GitHub Copilot PRs receive substantially more reviews (4.94/PR) than OpenAI Codex PRs (1.39/PR), coinciding with lower acceptance rates (68.0% vs. 77.9%). However, this does not imply a causal link: varying review counts may reflect confounders such as task complexity or repository reviewing policies.

**RQ2:** PR task type is a dominant factor—the 29% acceptance rate gap between task categories substantially exceeds inter-agent variance within categories.

Table 3: Task distribution by agent (% of each agent’s PRs). † indicates &lt;20 PRs.

Agent	feat	fix	docs	refactor
OpenAI Codex	28.9%	28.3%	11.3%	7.0%
GitHub Copilot	29.2%	41.6%	8.6%	6.7%
Devin	41.5%	25.7%	11.4%	8.3%
Cursor	33.2%	25.1%	16.9%	6.7%
Claude Code	52.5%	22.3%	9.4%†	9.4%†



Figure 2: RQ3. Acceptance rates (%) by agent and task type. “†” indicates &lt;20 PRs; “—” indicates zero PRs.

### 4.3 RQ3: Task-Stratified Agent Comparison

Global comparisons may confound agent capability with task distribution. Table 3 reveals substantial differences in task distributions across agents: GitHub Copilot handles proportionally more fix tasks (41.6%) than other agents, while Claude Code focuses heavily on feat tasks (52.5%). It should be noted that any results for Claude Code must be interpreted with caution due to the limited sample size (139 PRs total). If one agent handles mostly documentation (82% acceptance rate) while another handles fixes (66% acceptance), global metrics are misleading. Therefore, we stratify by task type. Figure 2 visualizes acceptance rates for each agent-task combination across nine categories (style, revert, other were excluded due to insufficient data). Cells marked “†” contain <20 PRs; the cell marked “—” indicates zero PRs (Claude Code has no ci PRs in our filtered dataset). OpenAI Codex achieves consistently high acceptance rates across the nine task categories, ranging from 59.6% (test) to 88.6% (chore), and leads in fix (83.0%) and refactor (74.3%) tasks. However, other agents achieve higher rates in specific categories: Claude Code leads in docs (92.3%, albeit with few samples) and feat (72.6%), while Cursor excels in test (77.8%, also with few samples). This task-dependent variation suggests that no single agent outperforms others across all categories, though unobserved confounders such as user expertise or repository characteristics remain uncontrolled. This pattern is illustrated in test tasks, where Devin achieves a higher acceptance rate (64.7%) than OpenAI Codex (59.6%), despite exhibiting substantially lower global performance (61.6% vs. 77.9%, shown in Table 1). We further examine whether task distributions shift over time within agents in Section 5. Table 4 presents task-stratified Chi-square tests of independence; only results passing the Bonferroni-adjusted significance threshold ( $\alpha = 0.05/64 \approx 0.00078$ ) are shown, with full results available in the replication package [23]. Of the 64 stratified comparisons across all 10 agent pairs, 6 are significant, with the strongest effect observed in fix tasks (Codex vs. Devin:  $\phi = 0.39$ , medium

**Table 4: RQ3. Pairwise agent comparisons. Only Bonferroni-significant results are shown ( $\alpha = 0.05/64 \approx 0.00078$ ).**

Comparison	Task	p-value	Winner	$\phi$	Effect
Codex vs Devin	fix	<0.0007	Codex	0.39	medium
Copilot vs Devin	fix	<0.0007	Copilot	0.20	small
Devin vs Cursor	fix	<0.0007	Cursor	0.28	small
Codex vs Copilot	fix	<0.0007	Codex	0.19	small
Codex vs Devin	feat	<0.0007	Codex	0.11	small
Cursor vs Copilot	fix	<0.0007	Cursor	0.11	small

effect). All significant comparisons involve `fix` (5 of 6) or `feat` (1 of 6) tasks, suggesting that performance differences between agents are most detectable in these core development activities. Notably, comparisons involving Devin on `fix` tasks show consistent patterns: Devin underperforms relative to Codex ( $\phi = 0.39$ ), Copilot ( $\phi = 0.20$ ), and Cursor ( $\phi = 0.28$ ), indicating a potential weakness in bug-fixing capabilities (which can also be observed in terms of raw acceptance rates in Table 2). However, it must be noted that without controlling for other possible confounders, such as repository-level effects, these findings remain observational.

**RQ3:** No single agent consistently outperforms all others; OpenAI Codex shows consistent strength across categories, but Claude Code and Cursor lead in specific task types.

## 5 Discussion

Our results reveal that AI agents exhibit heterogeneous temporal trends: Devin’s acceptance rate increases over time, while others remain relatively stable. OpenAI Codex’s consistently high acceptance rates (60–88% across all tasks), leading in `fix` and `refactor` tasks, suggest strong underlying capabilities. However, Claude Code and Cursor outperform Codex in specific tasks. The 44.4 percentage-point gap on test tasks (Cursor 77.8% vs. Claude Code 33.3%) represents the largest inter-agent disparity across task categories, suggesting that more complex tasks exhibit greater differences.

**Sensitivity Analysis: Aligned Time Windows.** To address concerns about temporal comparability, we repeated our analysis using only the 11-week window common to all agents (May 19–July 30, 2025). Results remain consistent: OpenAI Codex has the highest acceptance rate (79.9%), followed by Cursor (74.4%), Claude Code (72.6%), Devin (68.0%), and GitHub Copilot (68.0%). Claude Code shows moderate variance (std=11.7%) with 113 PRs in this window.

**Task Distribution Heterogeneity.** As shown in Table 3, task distributions vary substantially across agents, underscoring the importance of task-stratified comparisons. We further examined whether task type distributions shift over time within agents, as such compositional changes could confound the interpretation of performance trends. Comparing early versus late observation periods, we find varying patterns: Devin and GitHub Copilot shifted toward `feat` tasks (+9.8pp and +8.1pp), while Codex shifted away (-5.7pp), and Claude Code shows the largest compositional change (+18.4pp `feat`, -23.0pp `docs`). These shifts have competing implications for trend interpretation: Devin’s acceptance rate increased despite handling a progressively larger proportion of more complex tasks (+9.8pp shift toward `feat` tasks). This shift toward harder tasks suggests the observed improvement may underestimate actual capability gains.

**Implications for Practitioners.** Our results indicate that agent choice matters most for bug fixes (Cursor 80.4%, Codex 83.0% vs. Devin 45.6%) and testing tasks (33–78% range); documentation shows minimal differentiation (>79%). Depending on the use case, practitioners may want to adopt a combination of agents to address different workflows.

**Implications for Researchers.** Task stratification should become standard practice: future studies should report task distributions alongside global metrics, stratify comparisons by task type, and flag underrepresented types. Given the limitations of acceptance rates, complementary metrics (static analysis, complexity, maintenance burden) are needed [12]. Extended discussion of practical implications and actionable recommendations is available in the [supplementary material](#) [23].

**Threats to Validity.** We cannot establish causality (internal validity)—observed trends could result from model updates, user learning, or task distribution shifts. Our data covers only 8 months from 100+ star repositories (external validity), limiting generalization to smaller projects. PR acceptance does not equal code quality (construct validity), as merged PRs may contain bugs [22]. Additionally, repository-level clustering—where multiple PRs originate from the same repository—is not explicitly modeled in our Chi-square tests. If certain repositories systematically accept or reject PRs regardless of quality, this could inflate statistical significance; future work should consider mixed-effects models to account for repository-level variance. Observation windows vary substantially (11–32 active weeks), and some agents show discontinuous activity, requiring cautious interpretation of observed trends. Finally, agent representation ranges from 139 to 2,252 PRs, with Claude Code’s sparse data limiting reliability.

**Ethical Considerations.** This study uses only publicly available data from repositories with permissive licenses (Apache-2.0 and MIT) [10]. Through our analysis, no attempt is made to re-identify any individuals present in the dataset, and only aggregate findings are reported. The study received ethics approval from University College London (LREC no.: 2568).

## 6 Conclusion

Analyzing 7,156 pull requests from five AI coding agents, we find that **task type is a dominant factor over** acceptance rates (29pp gap between types), motivating our **task-stratified comparison** methodology. Our stratified analysis reveals that no single agent performs best across all task types. OpenAI Codex achieves consistently high acceptance rates (59.6%–88.6%), but Claude Code leads in documentation (92.3%) and features (72.6%), while Cursor excels in `fix` tasks (80.4%). These task-dependent variations provide evidence against simple “best agent” rankings. Temporal analysis reveals heterogeneous evolution patterns: Devin exhibits the only consistent positive trend (+0.77%/week over 32 weeks), while other agents maintain stable acceptance rates. These findings underscore the importance of considering both task context and temporal dynamics when evaluating AI coding agents. Future work should incorporate code quality metrics and static analysis warnings to complement acceptance rate analysis.

**Data Availability.** All data, scripts, results, and other supplementary materials are available on [GitHub](#) [23].

## References

- [1] Anthropic. 2025. Claude Code Documentation. <https://docs.anthropic.com/clause-code>
- [2] Anysphere. 2025. Cursor: The AI-First Code Editor. <https://cursor.sh>
- [3] Mark Chen et al. 2021. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374* (2021).
- [4] William S Cleveland. 1979. Robust Locally Weighted Regression and Smoothing Scatterplots. *J. Amer. Statist. Assoc.* 74, 368 (1979), 829–836.
- [5] Cognition AI. 2024. Introducing Devin, the First AI Software Engineer. <https://www.cognition.ai/blog/introducing-devin>
- [6] Jacob Cohen. 1988. *Statistical Power Analysis for the Behavioral Sciences* (2nd ed.). Lawrence Erlbaum Associates.
- [7] Angela Fan, Beliz Gokkaya, Mark Harman, Mitessh Lytvyn, Shubho Sengupta, Shin Yoo, and Jie M. Zhang. 2024. Large Language Models for Software Engineering: A Systematic Literature Review. *ACM Transactions on Software Engineering and Methodology* 33, 7 (2024), 1–79.
- [8] Ronald Aylmer Fisher. 1922. On the interpretation of  $\chi^2$  from contingency tables, and the calculation of P. *Journal of the Royal Statistical Society* 85, 1 (1922), 87–94.
- [9] GitHub. 2022. GitHub Copilot Research Recitation. <https://docs.github.com/copilot>
- [10] Nicolas E. Gold and Jens Krinke. 2020. Ethical Mining: A Case Study on MSR Mining Challenges. In *Proceedings of the 17th International Conference on Mining Software Repositories, MSR 2020*. ACM, Seoul, Korea, 265–276. doi:10.1145/3379597.3387462
- [11] Jingzhi Gong, Yixin Bian, Luis de la Cal, Giovanni Pinna, Anisha Uteem, David Williams, Mar Zamorano, Karine Even-Mendoza, William B Langdon, Hector D Menendez, et al. 2025. GA4GC: Greener Agent for Greener Code via Multi-Objective. In *SSBSE 2025 Challenge Case: Green SBSE*.
- [12] Hao He, Courtney Miller, Shyam Agarwal, Christian Kästner, and Bogdan Vasilescu. 2025. Does AI-Assisted Coding Deliver? A Difference-in-Differences Study of Cursor's Impact on Software Projects. *arXiv e-prints* (2025), arXiv–2511.
- [13] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2023. Large Language Models for Software Engineering: A Systematic Literature Review. *arXiv preprint arXiv:2308.10620* (2023).
- [14] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. 2014. The Promises and Perils of Mining GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR)*. ACM, 92–101.
- [15] Hao Li et al. 2025. AIDev: A Large-Scale Dataset of AI-Generated Code Contributions. HuggingFace Datasets. <https://huggingface.co/datasets/hao-li/AIDev>
- [16] Hao Li et al. 2025. The Rise of AI Software Engineers: A Quantitative Analysis of AI-Authored Contributions on GitHub. *arXiv preprint* (2025).
- [17] Hao Li, Haoxiang Zhang, and Ahmed E. Hassan. 2025. The Rise of AI Teammates in Software Engineering (SE) 3.0: How Autonomous Coding Agents Are Reshaping Software Engineering. *arXiv preprint arXiv:2507.15003* (2025).
- [18] METR. 2025. Measuring the Impact of AI Coding Assistants on Developer Productivity.
- [19] David Murillo et al. 2025. The Speed-Quality Tradeoff in AI-Assisted Software Development. *Proceedings of ICSE* (2025).
- [20] Karl Pearson. 1900. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50, 302 (1900), 157–175.
- [21] Sida Peng, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. 2023. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. In *arXiv preprint arXiv:2302.06590*.
- [22] Neil Perry, Megha Srivastava, Deepak Kumar, and Dan Boneh. 2023. Do Users Write More Insecure Code with AI Assistants?. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2785–2799.
- [23] Giovanni Pinna. 2025. *Comparing AI Coding Agents*. [https://github.com/giovannipinna96/Comparing\\_AI\\_Coding\\_Agents](https://github.com/giovannipinna96/Comparing_AI_Coding_Agents)
- [24] David Williams, Max Hort, Maria Kechagia, Aldeida Aleti, Justyna Petke, and Federica Sarro. 2026. Empirical and Sustainability Aspects of Software Engineering Research in the Era of Large Language Models: A Reflection. In *Proceedings of the 2026 IEEE/ACM 48th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER'26)* (Rio de Janeiro, Brazil). ACM, New York, NY, USA. doi:10.1145/3786582.3786827
- [25] Burak Yetistiren, Isik Ozsoy, and Eray Tuzun. 2022. Assessing the Quality of GitHub Copilot's Code Generation. In *Proceedings of the 18th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE)*. ACM, 62–71.
- [26] Albert Ziegler, Eirini Kalliamvakou, X Alice Li, Andrew Rice, Devon Rifkin, Shawn Simister, Ganesh Sittampalam, and Edward Aftandilian. 2022. Productivity assessment of neural code completion. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, 21–29.