Detecting AI Coding Tool Adoption and Its Behavioural Effects on Developer Commit Activity

Working paper

Andreas Varotsis

2026-04-07

Abstract

We study how adopting AI coding tools changes developer behaviour on GitHub, using a behavioural classifier and two empirical designs. First, we build a classifier that identifies AI coding tool users from observable commit history, without self-reported adoption data or proprietary telemetry. Trained on 235 GitHub accounts (33 confirmed Claude Code adopters and 202 controls), it reaches a cross-validated AUC of 0.94, and 0.93 when the training set is expanded to 276 accounts (74 adopters). It also scores users of a tool it never saw in training (Aider) at a mean predicted probability of 0.73, against 0.03 for controls, which suggests it detects AI-assisted coding behaviour in general rather than the style of one tool.

Second, we use the labelled accounts and the classifier in two designs. An account-level difference-in-differences finds large, statistically significant changes for confirmed adopters relative to controls: more commits per active week, much shorter gaps between commits, and fewer active weeks. Adopters also open and merge substantially more pull requests (+46 opened and +43 merged over the post-period, $p < 0.001$), an effect that survives false-discovery-rate correction and five sensitivity checks. A country-level panel across 34 countries (2022–2024) uses the mean classifier score in each country as an adoption measure. Its estimates for commits per developer are negative in every specification but at best borderline significant (weighted coefficient −7.56, $p = 0.051$; unweighted −5.14, $p = 0.25$), while pull requests per developer show no detectable association (+1.33, $p = 0.77$).

We read the gap between the account-level PR gain and the country-level null as a **scale mismatch**: the effect is concentrated among a small share of developers and is diluted in national aggregates. The split between commits and pull requests is consistent with AI tools changing how work is packaged into commits without reducing accepted output, but the country-level design is underpowered and we cannot rule out other explanations. Independent of these results, the classifier shows that AI tool adoption can be measured at scale from public commit behaviour, opening the way to non-survey measurement of adoption across the developer population.

1. Introduction

The rapid diffusion of AI coding assistants since late 2022 — including GitHub Copilot, Anthropic's Claude Code, and Aider — has prompted widespread speculation about their

effects on software development behaviour. Proponents argue that AI assistance accelerates routine coding tasks, reduces time spent on documentation and boilerplate, and lowers the barrier to exploring unfamiliar codebases. Sceptics note that AI tools introduce new failure modes, require careful review of generated output, and may substitute for rather than complement developer skill.

Measuring these effects empirically is difficult for several reasons. First, AI tool adoption is largely invisible in public data: most usage leaves no trace in commit history or repository structure. Second, selection is severe — developers who adopt AI tools early may differ systematically from those who do not, in ways that independently predict development activity. Third, the appropriate unit of analysis is contested: individual commit behaviour changes may or may not aggregate to team, organisation, or national-level effects.

This paper addresses the measurement problem directly. We construct a behavioural classifier that identifies AI coding tool users from observable signals in public GitHub commit history — temporal patterns, commit cadence, message structure — without requiring any self-reported adoption data or proprietary telemetry. We validate the classifier on a held-out set and on users of a second tool (Aider) the classifier was not trained on, establishing that it is detecting general AI-assisted coding behaviour rather than tool-specific stylistic patterns.

We then use the labelled accounts and the classifier in two designs. An account-level difference-in-differences compares changes in confirmed AI adopters with changes in randomly sampled controls over the same period, for both commit behaviour and authored pull requests. A country-level panel regression uses the mean classifier score in each country as an adoption measure and relates it to commit and pull-request activity in national aggregates.

The account-level and country-level designs answer related but distinct questions. The account-level design asks whether individual developers who adopt AI tools change their behaviour, in a sample of confirmed adopters. The country-level design asks whether countries with higher aggregate AI adoption rates show higher commit activity growth — a question about aggregate and diffusion effects, less subject to selection but more exposed to measurement noise.

We find strong evidence for account-level effects. Confirmed adopters concentrate their work into fewer, more intense weeks, and they open and merge substantially more pull requests than controls. The country-level results are weaker and split by outcome: estimates for commits per developer are consistently negative but at best borderline significant, while pull requests per developer show no association. Our preferred reading is that AI tools change how work is packaged into commits without reducing accepted output, and that the account-level gains are too concentrated to show up in national aggregates. The country-level design is underpowered, so we cannot rule out alternative explanations from these data alone.

The remainder of the paper is structured as follows. Section 2 reviews the relevant literature. Section 3 describes the data. Section 4 presents the classifier methodology and validation. Section 5 describes the two empirical designs. Section 6 presents results: commit behaviour and pull-request outcomes at the account level, then the country-level panel. Section 7 discusses the findings and their limitations. Section 8 concludes.

2. Literature Review

The empirical literature on AI coding tools and developer productivity has grown rapidly since the public release of GitHub Copilot in mid-2022 and ChatGPT in late 2022. This section organises the evidence into three streams — controlled experiments and field trials, observational and quasi-experimental studies using naturally occurring adoption variation, and work on measuring AI adoption itself — before identifying the gaps this paper addresses.

2.1 Controlled Experiments and Field Trials

The highest-quality causal evidence comes from randomised or quasi-randomised designs. Peng et al. (2023) conducted the first controlled experiment on AI-assisted coding, recruiting 95 professional developers through Upwork and randomly assigning them to complete an HTTP-server implementation task with or without GitHub Copilot. The treatment group completed the task 55.8% faster (95% CI: 21–89%, $p = 0.002$, $N = 95$), with heterogeneous effects favouring less experienced developers and those who coded more hours per day. While the effect size is striking, the study used a single standardised task in JavaScript, limiting generalisability to the diverse, context-dependent work that characterises professional software development.

Ziegler et al. (2024) surveyed 2,047 users of the Copilot technical preview and linked their responses to usage telemetry. The share of suggestions a developer accepted predicted perceived productivity better than measures of how much suggested code persisted. The outcome is self-reported, but the scale of the study and its link to telemetry complement the smaller experiments.

Cui et al. (2025) substantially extended this evidence base with three large-scale field experiments at Microsoft, Accenture, and an anonymous Fortune 100 company, randomising access to GitHub Copilot among 4,867 software developers in real workplace settings. Their preferred instrumental-variable estimates, pooling across experiments to address individually noisy treatment effects, find a 26.08% increase (SE: 10.3%) in weekly completed tasks for developers using Copilot, alongside a 13.55% increase (SE: 10.0%) in commits and a 38.38% increase (SE: 12.55%) in code compilations. Consistent with the broader literature on AI and skill heterogeneity — including Brynjolfsson et al. (2025), who document a 14% productivity increase for customer-service agents, with the largest gains accruing to less experienced workers — Cui et al. (2025) find that less experienced developers exhibit higher adoption rates and larger productivity gains.

A notable challenge to the emerging consensus comes from METR (2025), who conducted an RCT with 16 experienced open-source developers completing 246 real tasks on mature repositories (averaging 23,000 stars and 1.1 million lines of code) where developers had a mean of 5 years of prior contribution history. Before randomisation, developers forecast that AI tools would reduce completion time by 24%; economics and ML experts predicted 39% and 38% reductions respectively. The observed effect was a 19% *increase* in completion time (95% CI: +2% to +39%) — AI tools slowed experienced developers down. Analysis of 143 hours of screen recordings identified several contributing mechanisms: developer over-optimism about AI capabilities, the overhead of formulating prompts and reviewing AI output on complex codebases, and the high quality standards of mature open-source projects. While the authors caution that results are specific to their setting — highly experienced developers on large, well-established codebases — the finding sharply illustrates that the relationship between

AI assistance and productivity is not uniformly positive and may depend critically on task complexity, codebase familiarity, and developer expertise.

2.2 Observational and Quasi-Experimental Studies

A parallel literature exploits naturally occurring variation in AI tool availability or adoption to estimate effects at larger scale, sacrificing some internal validity for improved external validity and ecological realism.

Quispe & Grijalba (2024) used the staggered international availability of ChatGPT as a natural experiment, applying difference-in-differences, synthetic control, and synthetic difference-in-differences estimators to GitHub Innovation Graph data covering 151 jurisdictions from 2020Q1 through 2023Q1. Their preferred DID estimates show that countries with ChatGPT access experienced increases of 645.6 git pushes per 100,000 population (baseline mean: 741.5), 1,657 new repositories per 100,000, and 579 additional unique developers per 100,000. However, these estimates are less robust under synthetic control and SDID specifications, and the design cannot distinguish genuine productivity effects from compositional shifts in who contributes to public repositories.

He et al. (2025) provide the most detailed study of a modern agentic coding tool. Using the appearance of `.cursorrules` configuration files in GitHub repositories to identify Cursor adoption, they construct a staggered difference-in-differences design comparing 806 adopting repositories against 1,380 propensity-score-matched controls, using an imputation estimator designed for staggered treatment timing. Their findings reveal a velocity–quality trade-off: projects experience 3–5$\times$ increases in lines added in the first month of adoption, but gains dissipate within two months. Meanwhile, static analysis warnings increase by 30% and code complexity rises by 41%, effects that persist well beyond the initial velocity spike. Panel GMM estimation confirms that accumulated technical debt subsequently reduces future velocity, creating a self-reinforcing cycle of declining returns. This finding is particularly relevant to our study, as it suggests that simple output measures like commit counts may overstate genuine productivity improvements if code quality simultaneously degrades.

2.3 Measuring AI Adoption

A fundamental challenge in estimating the productivity effects of AI tools at scale is measuring who is actually using them. Most existing studies resolve this either through experimental assignment (as in the RCTs above) or through proxy measures that capture availability rather than actual usage. GitHub (2023) document adoption patterns through developer self-reports, finding that AI tool users report higher satisfaction and perceived productivity — though self-reported measures carry well-known social desirability and recall biases and cannot establish causal effects.

Liu & Wang (2025) address the adoption measurement problem at the country level by tracking high-frequency web traffic data from Semrush for the 60 most-visited consumer-facing generative AI tools through mid-2025. Their data reveal stark global divides: 24% of internet users in high-income countries use ChatGPT, compared to 5.8% in upper-middle-income, 4.7% in lower-middle-income, and just 0.7% in low-income countries. Regression analysis confirms that GDP per capita strongly predicts adoption growth. While web traffic captures real usage rather than policy readiness or infrastructure capacity, it cannot distinguish between casual

exploration and deep workflow integration, nor does it identify which specific professional activities (such as software development) the usage supports.

2.4 Gaps and Contributions of This Paper

Three gaps emerge from this literature. First, there is a **measurement gap**: studies with strong causal identification — RCTs and firm-level field experiments — typically have narrow samples (a single task, a single firm, or a small group of developers), while studies using naturally occurring variation rely on coarse proxies for adoption such as country-level ChatGPT availability (Quispe & Grijalba, 2024) or the presence of configuration files (He et al., 2025). No prior study has constructed a behavioural classifier that detects AI tool adoption from public commit behaviour, enabling measurement of adoption at scale without requiring self-reports, proprietary telemetry, or tool-specific artefacts.

Second, there is a **cross-tool generalisation gap**. Each existing study is specific to a single tool — Copilot (Cui et al., 2025; Peng et al., 2023; Ziegler et al., 2024), ChatGPT (Quispe & Grijalba, 2024), or Cursor (He et al., 2025). Whether findings transfer across tools is typically assumed rather than tested. Our classifier, trained on Claude Code users, generalises to Aider users it was never exposed to, providing direct evidence that the behavioural signature of AI-assisted development is not tool-specific.

Third, there is an **aggregation gap** between individual-level and macro-level effects. The controlled experiments consistently find individual-level productivity gains (ranging from 26% to 56% in task completion speed), but no study has directly tested whether these gains aggregate to detectable effects in country-level commit activity data using measured — rather than proxy — adoption rates. Our country-level panel regression attempts precisely this test. Its results (Sections 6.5–6.8) are informative in their own right: they are consistent with real individual-level effects that are too concentrated, relative to the noise in country-year aggregates, or an adoption window too short, to produce statistically detectable country-level shifts. That disciplines expectations about how quickly micro-level AI productivity gains translate into macro-level outcomes.

3. Data

3.1 Data Sources

Our pipeline draws on two complementary sources: GitHub Archive as a sampling frame for active developer accounts, and the GitHub REST API for collecting the behavioural features used in the analysis.

GitHub Archive is a public record of GitHub activity events (pushes, pull requests, issues, releases) available from 2011 onward. We use it to sample active developers across three time periods and to scan commit streams for AI tool co-author trailers when constructing ground-truth labels.

GitHub REST API provides the behavioural data. Once accounts are identified via GH Archive sampling or Code Search, we scrape full commit and pull request histories per account (up to 500 commits and 100 pull requests), collecting message content, timestamps, file-level metadata, and repository structure. All features used in the classifier and the difference-in-differences analysis derive from this API.

We use three samples:

Classifier training sample. We sample 12 hourly windows spanning November 2024, January 2025, and March 2025 from GH Archive, yielding approximately 380,000 unique active developer accounts. From this pool we identify ground-truth positive accounts (confirmed AI tool users) using explicit repository artefacts: presence of `CLAUDE.md`, `.claude/` directories, or `Co-Authored-By: Claude` commit trailers. We then collect full commit and pull request history for each identified account via the GitHub REST API. A later expansion scrape of co-author trailers added 41 further confirmed adopters, which we use for the expanded classifier and the pull-request analysis.

Commit activity panel. We collect 9 quarterly hourly windows from Q4 2022 through Q4 2024 from GH Archive, sampling 500 active developers per window. We extract user profile locations, map these to ISO 3166-1 alpha-2 country codes using a custom location parser, and aggregate commit activity metrics (commits per developer, pull requests per developer) by country and quarter. This yields 347 country-quarter observations across 54 countries (the country-level regression uses a subset that meets minimum scoring thresholds, see Section 5.2).

Population scoring sample. To construct per-country AI adoption rates, we scrape GitHub accounts with parseable location fields mapping to panel countries via the GitHub REST API across three rounds (v1: 2,048 accounts, v2: 312 accounts, v3: 2,999 accounts) for a combined 4,824 unique scored accounts. Each account is scored by the classifier trained on the original 235 accounts to yield a predicted probability of AI tool adoption. Countries with at least 15 scored accounts (46 countries) are eligible for the country-level regression; 34 of these also pass the panel's minimum-developer threshold.

3.2 Ground Truth Labels

Positive accounts (AI tool users). Confirmed via two routes: 1. *GitHub Code Search*: repositories containing `CLAUDE.md` in the root, resolved to account logins. 2. *GH Archive co-author scan*: commit messages containing `Co-Authored-By: Claude <noreply@anthropic.com>` or equivalent Aider trailers.

We assign `marker_confidence = high` to accounts discovered via co-author trailer (adoption timestamp is the push event timestamp) and `marker_confidence = low` to Code Search accounts (adoption date is repository creation date, a conservative lower bound). The original training set has 33 positive accounts, 25 of them high-confidence. The expansion scrape added 41 co-author positives, giving 74 positives in the expanded set, 66 of them high-confidence.

Negative accounts (non-adopters). Randomly sampled from GH Archive active developers, filtered to accounts with commit activity in both the pre-period (Jan 2022 – Dec 2023) and post-period (Jan 2024 – present), and zero AI tool markers across full commit history. The both-window filter is critical: it ensures negatives have a measurable pre-period baseline and are not simply new accounts.

3.3 Pre/Post Windows

For the account-level analysis, all accounts are split at a global cutoff: - **Pre-period**: January 2022 – December 2023 - **Post-period**: January 2024 – present

This global cutoff captures the period after widespread AI coding tool availability (ChatGPT: November 2022; Claude Code and Aider: 2023–2024). All analyses use this global cutoff. The high-confidence robustness check (Section 6.4) changes the treated group, not the windows.

3.4 Summary Statistics

Table 1 summarises the 235 accounts used in the commit-level analysis. Adopters and controls made a similar number of commits in the pre-period, but adopters' mean commit count fell in the post-period while controls' rose. Adopters were active in fewer weeks and made more commits in each active week, and the gap between them widened after adoption. Commit histories are capped at 500 commits per account, which binds for 2% of controls and no adopters.

For the country-level commit activity panel, the median country-quarter cell contains 2 located developers, and the median country-year cell contains 5. That thinness substantially limits the power of the country-level regression (Section 6.8).

Table 1. Summary statistics, commit-level sample (N=235: 33 adopters, 202 controls)

Variable	AI mean	Ctrl mean	AI median	Ctrl median	AI SD	Ctrl SD
Pre-period commits	81.88	79.42	54.00	50.50	85.35	94.29
Post-period commits	54.03	92.81	31.00	46.00	53.43	132.03
Pre-period active weeks	7.82	13.22	4.00	10.00	8.13	11.05
Post-period active weeks	3.79	17.66	2.00	11.00	5.20	16.73
Pre commits / active week	13.73	6.58	8.25	5.16	13.63	4.67
Post commits / active week	23.55	5.26	14.00	4.20	24.55	4.13
Pre inter-commit hours	281.06	180.49	74.48	107.84	664.60	206.25
Post inter-commit hours	57.67	324.89	5.82	171.37	125.90	372.98

4. Behavioural Classifier

4.1 Design Rationale

The central methodological challenge is identifying AI tool users without relying on explicit markers (which are rare and may be biased toward power users) or survey data (which is expensive and subject to recall and social desirability bias).

Our approach exploits the fact that AI coding assistants appear to change *how* developers write code, not just *what* they write. Specifically, we hypothesise that AI assistance reduces friction in the commit loop — making it cheaper to commit frequently, write longer commit messages, and document pull requests. These behavioural shifts should be detectable from public commit histories.

Critical design constraint. The explicit artefacts used to identify ground truth (CLAUDE.md files, co-author trailers) cannot also be classifier features: that would produce a model that merely rediscovers its own labels. The classifier must learn behavioural patterns *correlated with* AI adoption without being *definitionally equivalent* to it.

4.2 Features

We compute 15 behavioural measures for each account in each window, plus their change between windows (Δ = post − pre). Raw commit counts are used only as a change, which leaves 43 features in two groups:

Message and documentation (21 features): mean commit message length, fraction of multiline messages, fraction using bullet points, fraction using conventional commit format, fraction mentioning tests, mean PR body length, and fraction of PRs with a body.

Activity and tempo (22 features): active weeks, commits per active week, mean inter-commit hours, fraction of burst commits (multiple commits within one hour), repositories touched, test co-write rate, sampled file count, and the change in commit count.

The change features carry the strongest signal for a difference-in-differences framing.

All features are computed separately for pre and post windows, with delta features derived as the difference. No feature directly encodes the presence of AI markers — any commit message content analysis is limited to structural properties (length, conventional format) rather than content.

4.3 Model and Performance

Table 2. Classifier Performance (N=235, 5-fold CV)

Model	CV AUC (mean)	CV AUC (±SD)	Ablation AUC	AUC drop
Logistic Regression	0.906	0.060	0.896	0.010
Random Forest	0.940	0.054	0.909	0.031
Gradient Boosting	0.898	0.097	0.890	0.008

Ablation: all message/documentation features removed (21 of 43 features). Random Forest selected as primary model.

The Random Forest achieves CV AUC of 0.940 ± 0.054, the highest of the three models tested. The top features by importance are post-period inter-commit hours (0.130), pre-period message length (0.120), and post-period active weeks (0.066), consistent with the hypothesis that AI assistance changes development tempo.

Retraining the Random Forest on the expanded set of 276 accounts (74 positives) gives a cross-validated AUC of 0.93 (±0.04), so the result is not an artefact of the small original positive class. The country-level adoption scores in Section 6.1 use the original model.

4.4 Writing-Style Ablation

A key validity concern is whether the classifier is detecting genuine behavioural change or merely learning Claude's distinctive verbose commit message style. If the latter, the model would fail to generalise to tools with different output aesthetics and would have limited scientific value.

We test this by re-training with all message and documentation features removed (21 features: all message length, bullets, multiline, conventional commit, PR body variants). The activity-only model achieves AUC 0.909 — a drop of only 3.1 points. Inter-commit hours and active weeks carry the model independently.

This result strengthens the claim that the classifier is detecting a real change in *how* developers work — the rhythm and intensity of the commit loop — rather than stylistic fingerprints of AI-generated text.

4.5 Cross-Tool Generalisation

Figure 1. Classifier score distribution: Claude Code, Aider, and controls

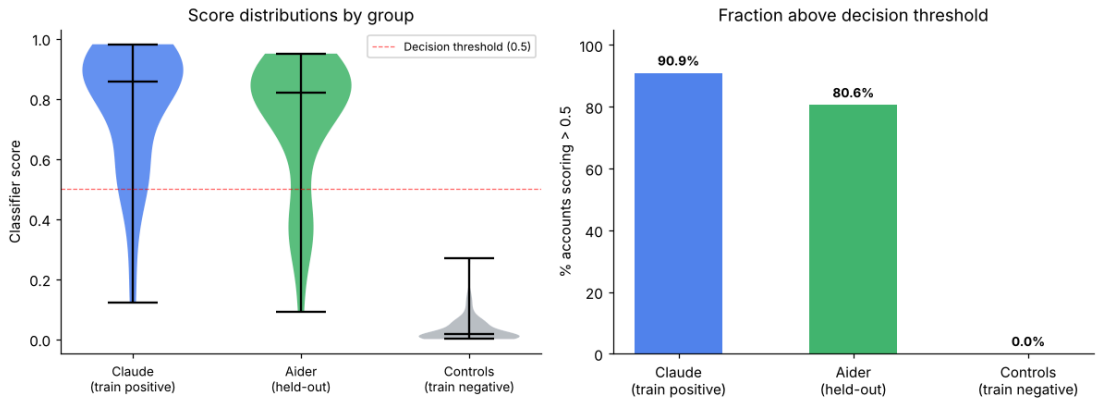


Table 3. Three-way validation results

Group	N	Mean	Median	SD	>0.5
Claude (train positive)	33	0.776	0.856	0.214	90.9%
Aider (held-out)	36	0.727	0.820	0.222	80.6%
Controls (train negative)	202	0.033	0.016	0.045	0.0%

Training-sample scores are out-of-fold cross-validated predictions. One-sided Mann-Whitney tests: Aider above controls, p < 0.0001; Aider below Claude Code positives, p = 0.065 (not significant at 5%).

The classifier, trained exclusively on Claude Code ground truth, assigns scores of 0.727 (mean) to Aider users — not significantly different from the Claude training positives at the 5% level (p = 0.065) and far above the negative controls (p < 0.0001). 80.6% of Aider accounts score above the 0.5 decision threshold, compared to 90.9% of Claude positives and 0% of controls.

This cross-tool generalisation is the key validity result. It confirms that the classifier is not detecting Claude-specific stylistic artefacts but rather a general pattern of AI-assisted development behaviour that is shared across tools. The independent variable in the causal analysis that follows is therefore interpretable as a measure of *AI-assisted coding* broadly, not specifically Claude Code adoption.

5. Empirical Designs

5.1 Account-Level Difference-in-Differences

Setup. For commit behaviour, we treat the 33 confirmed AI tool adopters in the original training set as the treatment group and the 202 controls as the comparison group. For pull-request outcomes, we use the expanded cohort (74 adopters, 202 controls), because PR histories were collected after the expansion scrape. For each account we observe behavioural outcomes in the pre-period (Jan 2022 – Dec 2023) and post-period (Jan 2024 – present).

Estimator. For each outcome Y, we estimate:

$$\Delta Y_i = \alpha + \beta \cdot \text{Treatment}_i + \gamma \cdot Y_i^{\text{pre}} + \varepsilon_i$$

where $\Delta Y_i = Y_i^{\text{post}} - Y_i^{\text{pre}}$ is the within-account change, $\text{Treatment}_i = 1$ for AI adopters, and Y_i^{pre} controls for baseline differences between groups [regression adjustment; Angrist & Pischke (2009)]. Standard errors are HC3 heteroskedasticity-robust.

The coefficient β estimates the average treatment effect on the treated: the additional change in the outcome for AI adopters relative to controls, conditional on their pre-period level.

Identifying assumption. Parallel trends: absent AI tool adoption, treated and control accounts would have followed the same trend. We assess this by comparing pre-period levels between groups (Table 1). Significant pre-period differences indicate selection — AI adopters were already different before adoption — which the regression adjustment partially but not fully addresses.

Outcomes. For commit behaviour: commits per active week (primary commit activity measure), inter-commit hours (development tempo), active weeks, commit message length, fraction of conventional commits, fraction of PRs with a body, and test co-write rate. For pull requests: PRs opened and merged (in total and per month), merge rate, hours to merge, PRs closed without merging, lines changed, review comments, and commits per PR. We apply a Benjamini-Hochberg false-discovery-rate correction within each family of outcomes.

5.2 Country-Level Panel Regression

Setup. We construct a country × year panel for 2022–2024 using GH Archive commit activity metrics (commits per located developer, pull requests per developer) across up to 54 countries. The 2022 observations cover Q4 only. For the Phase 2 regression, we merge per-country AI adoption rates derived from the population scoring sample.

Adoption rate construction. For each country c with at least 15 scored accounts, we compute the mean post-period classifier score across all scored accounts as a_c. This is an adoption *intensity* index, not the share of developers using AI tools: very few population accounts score above the 0.5 decision threshold (Section 6.1). The AI adoption variable is:

$$\text{pct_ai_users}_{ct} = \begin{cases} 0 & \text{if } t < 2024 \\ a_c & \text{if } t = 2024 \end{cases}$$

This gives cross-country variation in the 2024 treatment intensity while holding pre-treatment at zero for all countries — a standard staggered-adoption design collapsed to two periods.

Estimator. PanelOLS with country and time fixed effects, clustered standard errors at the country level (linearmodels):

$$\log(\text{commits_per_dev}_{ct} + 1) = \mu_c + \lambda_t + \delta \cdot \text{pct_ai_users}_{ct} + \varepsilon_{ct}$$

We report four specifications:

- **Regression A**: Oxford Insights AI Readiness Index as the adoption regressor (Phase 1 baseline)
- **Regression C**: per-country classifier scores from the population sample (primary)
- **Regression C-W**: Regression C weighted by the number of developers in each country-year
- **Regression D**: a parallel-trends diagnostic that regresses the pre-period (2022⟶2023) change in commits per developer on the 2024 adoption score

An earlier specification (Regression B) used a single global 2024 adoption value. It is perfectly collinear with the year fixed effect, so we do not report it.

6. Results

6.1 Per-Country AI Adoption Rates

Table 4. Top 15 countries by mean classifier score

Country	Mean score	% above 0.5	N accounts
NO	0.113	0.0	40
FI	0.111	0.0	32
AR	0.106	0.0	34
BE	0.102	0.0	22
HU	0.099	0.0	16
CL	0.095	0.0	17
CA	0.094	0.0	128

Country	Mean score	% above 0.5	N accounts
TW	0.093	0.0	31
TH	0.093	0.0	15
CH	0.092	0.0	70
GB	0.091	0.5	212
AT	0.088	0.0	48
DE	0.086	0.3	310
ID	0.085	1.1	90
SG	0.085	0.0	46

Total scored accounts (deduplicated): 4,824. Countries with at least 15 accounts: 46. Mean score range: 0.061 to 0.113. Accounts scoring above 0.5 across the whole sample: 0.25%.

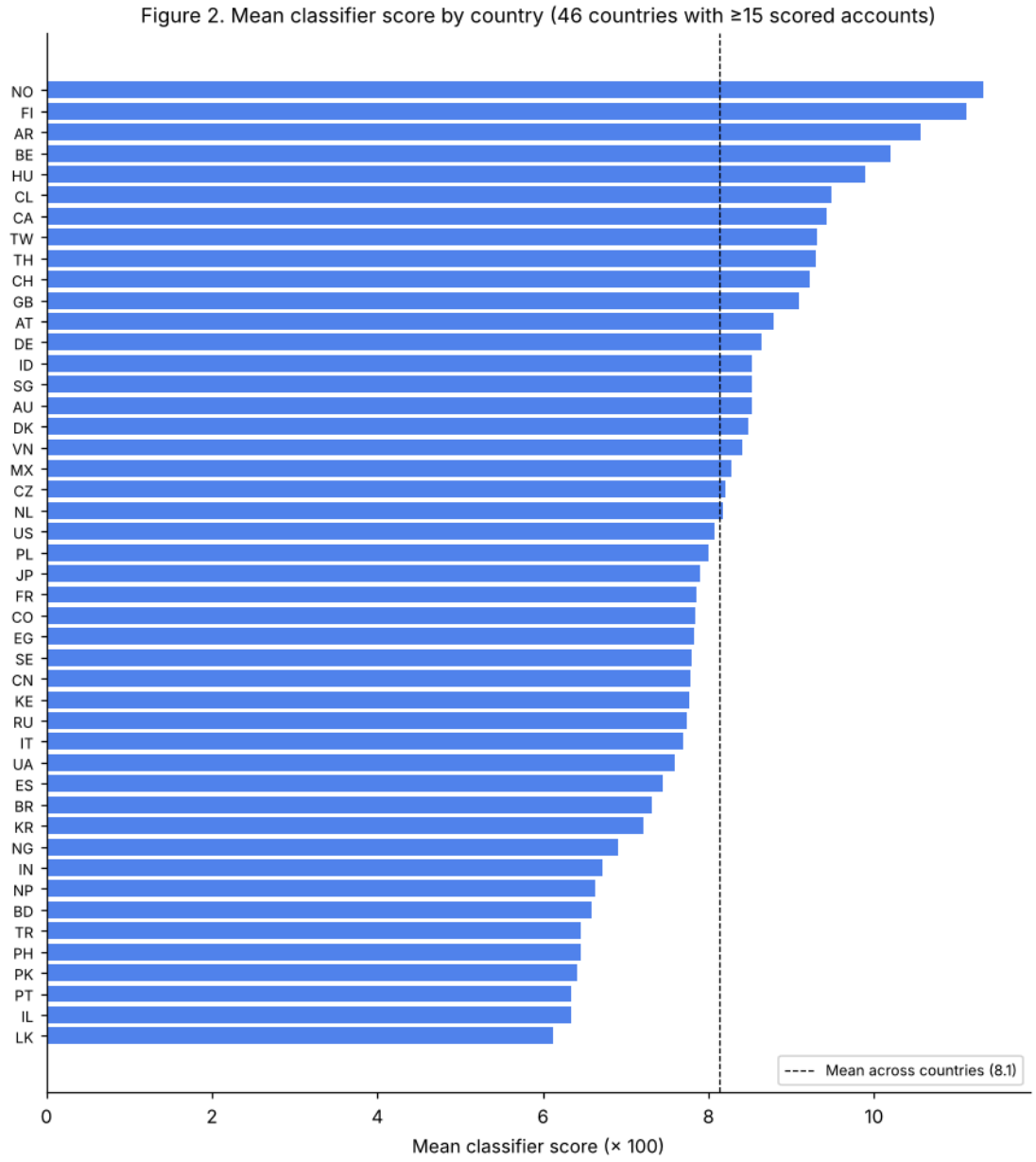


Figure 2 shows the mean classifier score for each of the 46 countries with at least 15 scored accounts. The cross-country range is narrow: from 0.061 (Sri Lanka) to 0.113 (Norway), with a standard deviation of 0.013. Almost no population account crosses the classifier's decision threshold: 0.25% of all scored accounts score above 0.5, and no country exceeds 5%. The mean score therefore measures how AI-assisted a country's typical commit behaviour looks, not the share of its developers using AI tools.

The narrow cross-country variation is a key feature of the data. It limits the statistical power of the country-level regression (Section 6.8).

6.2 Account-Level Diff-in-Diff: Commit Behaviour

Table 5. Account-level diff-in-diff, commit behaviour (N=235)

Outcome	AI Δ	Ctrl Δ	Coef	SE	p	FDR q
Commits / active week	9.820	−1.324	13.073	3.073	<0.001	<0.001
Inter-commit hours	−223.395	144.408	−275.258	37.647	<0.001	<0.001
Active weeks	−4.030	4.441	−11.253	1.714	<0.001	<0.001
Message length (chars)	46.853	6.639	54.259	26.125	0.038	0.044
Conventional commits	0.108	0.033	0.076	0.049	0.123	0.123
PR has body	0.094	0.015	0.322	0.086	<0.001	<0.001
Test co-write rate	0.010	−0.037	0.144	0.062	0.020	0.027

Estimator: OLS of the post − pre change on treatment and the pre-period level, HC3 standard errors. FDR q-values use the Benjamini-Hochberg correction across the 7 outcomes.

Primary outcomes. Two outcomes are particularly striking. *Commits per active week* increases by 13.1 for AI adopters relative to controls (SE = 3.07, $p < 0.001$). *Inter-commit hours* decreases by 275 hours (SE = 37.6, $p < 0.001$): the mean gap between an adopter's commits falls from about 281 hours in the pre-period to 58 hours in the post-period, roughly a fifth of what it was. Controls show the opposite pattern, with the mean gap rising from 180 to 325 hours.

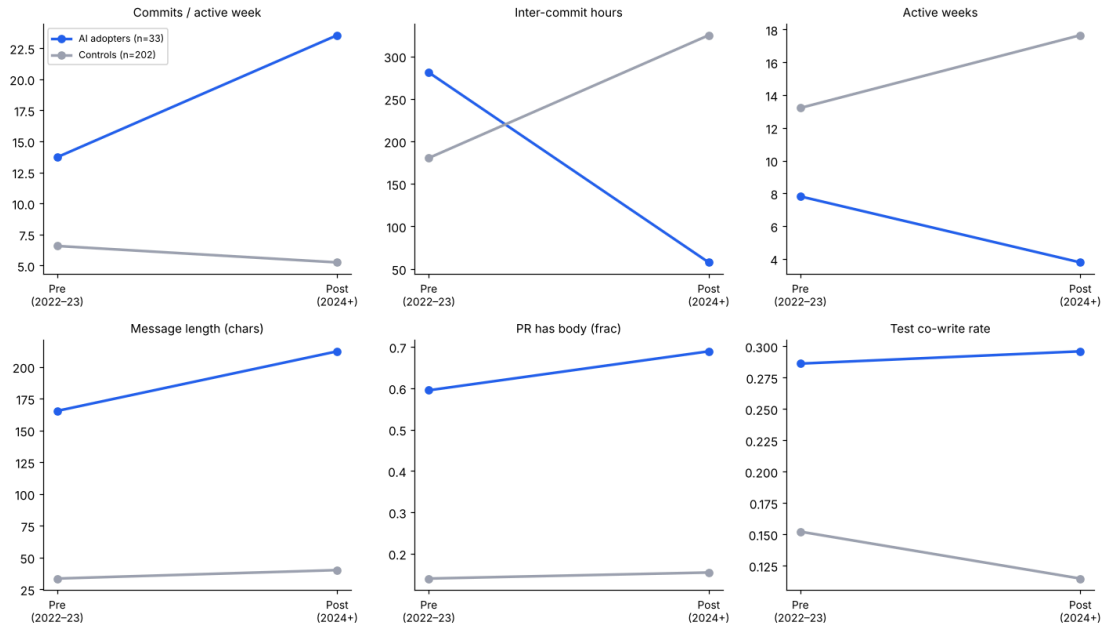
Adopters are also active in fewer weeks after adoption (−11.3 relative to controls, $p < 0.001$), and their total commit count falls (from a mean of 82 to 54) while controls' rises (79 to 93; Table 1). The combination points to AI adopters shifting toward more concentrated, high-intensity coding sessions: fewer weeks active, but many more commits in each active week.

Secondary outcomes. The fraction of pull requests with a body increases by 0.32 ($p < 0.001$), indicating substantially improved PR documentation. Message length (+54 chars, $p = 0.04$) and test co-write rate (+0.14, $p = 0.02$) are also significant. Conventional commit adoption is not significant ($p = 0.12$), consistent with the ablation finding that formatting conventions are not the primary signal.

Multiple testing correction. With 7 outcomes tested simultaneously, we report Benjamini-Hochberg FDR-corrected q-values alongside raw p-values (Table 5). Every outcome except conventional commits remains significant at $q < 0.05$. The message-length effect is the most fragile: it is close to the threshold here and loses significance under winsorisation (Section 6.4).

Pre-period differences. Adopters already differed from controls before adoption: they made more commits per active week (13.7 vs 6.6) across fewer active weeks (7.8 vs 13.2). This indicates selection: early AI adopters already worked in more concentrated bursts. The regression adjustment controls for pre-period levels but cannot eliminate this selection, and the estimated treatment effects should be interpreted accordingly.

Figure 3. Before/after: AI adopters vs controls (means across pre and post periods)



6.3 Account-Level Diff-in-Diff: Pull-Request Outcomes

The commit results measure workflow tempo and commit-message style, not accepted shipped work. To complement them, we collected authored pull requests via GitHub issue search for the expanded cohort over the same pre and post windows, then reran the difference-in-differences on PR outcomes.

Table 6. Account-level diff-in-diff, pull-request outcomes (N=275: 73 adopters, 202 controls)

Outcome	Coef	SE	p	FDR q
PRs opened	+46.41	12.12	<0.001	<0.001
PRs merged	+42.64	11.25	<0.001	<0.001
Opened PRs / month	+1.64	0.43	<0.001	<0.001
Merged PRs / month	+1.51	0.40	<0.001	<0.001
Merge rate	+0.16	0.06	0.004	0.008
Median hours to merge	−18.52	8.76	0.034	0.049
PRs closed unmerged	+3.66	1.49	0.014	0.023
Mean lines changed / PR	−4080.18	3783.84	0.281	0.312

Outcome	Coef	SE	p	FDR q
Mean review comments / PR	−0.21	0.47	0.653	0.653
Mean commits / PR	−2.42	1.81	0.180	0.226

Same estimator as Table 5. FDR q-values use the Benjamini-Hochberg correction across the 10 PR outcomes. Accounts with no PRs in a window have merge rate and hours to merge set to zero for that window.

Table 7. Pull-request outcomes: sensitivity checks

Specification	N (treated)	PRs opened	PRs merged	Merge rate
Drop accounts at the 300-PR cap	252 (60)	+28.0 (p = 0.001)	+25.3 (p = 0.002)	+0.17 (p = 0.007)
Drop accounts with no PRs	218 (58)	+58.1 (p < 0.001)	+53.5 (p < 0.001)	+0.15 (p = 0.004)
PRs in both windows	104 (18)	+83.3 (p = 0.002)	+74.9 (p = 0.003)	+0.03 (p = 0.386)
Drop accounts with no pre-period PRs	131 (20)	+83.5 (p = 0.002)	+74.9 (p = 0.002)	+0.11 (p = 0.163)
High-confidence adopters only	267 (65)	+50.8 (p < 0.001)	+46.7 (p < 0.001)	+0.18 (p = 0.003)

Coverage. Of the 276 accounts in the expanded cohort, 275 had retrievable PR data (73 adopters, 202 controls). The treated group is more PR-active: 58 of 73 adopters had at least one authored PR, compared with 160 of 202 controls. Twenty-three accounts hit the 300-PR retrieval cap (13 adopters, 10 controls). Fifty-seven accounts had no PRs at all (15 adopters, 42 controls), and only 104 had PRs in both windows (18 adopters, 86 controls).

Primary outcomes. The PR-volume results are strong and consistent (Table 6). Adopters open 46.4 more PRs over the post-period than controls (SE = 12.1, p < 0.001) and merge 42.6 more (SE = 11.2, p < 0.001). Per month, that is 1.64 more opened and 1.51 more merged. All four volume measures survive the FDR correction across the ten PR outcomes.

Secondary outcomes. Merge rate also rises (+0.16, p = 0.004), median hours to merge falls by about 18.5 hours (p = 0.034), and adopters close slightly more PRs without merging them (+3.7, p = 0.014). There is no detectable change in lines changed, review comments or commits per PR. The merge-rate effect collapses when we restrict to accounts with PRs in both windows (+0.03, p = 0.39), while the volume effects remain significant (Table 7). This suggests the rate and latency findings are partly mechanical: when an account had no PRs in the pre-period, its merge rate and hours to merge are recorded as zero, which inflates the post-period gain. We therefore treat PR volume as the primary accepted-output check and rate and latency as secondary.

Robustness. The PR-volume effects survive all five sensitivity checks in Table 7: dropping accounts at the 300-PR cap, dropping accounts with no PRs, keeping only accounts with PRs in both windows, dropping accounts with no pre-period PRs, and restricting the treated group to the 65 high-confidence adopters. In every specification, opened and merged PRs remain significant at $p < 0.005$.

Interpretation. The PR results support the view that AI adopters increase accepted, packaged work, not merely commit tempo or message style. They do not, however, contradict the country-level PR null (Section 6.7). The country panel aggregates across all developers in a country-year, most of whom show no sign of AI-assisted behaviour: fewer than 1% of scored population accounts cross the classifier's decision threshold, and mean scores vary only between 0.06 and 0.11 across countries. An effect of this size among a small minority of developers is invisible in that aggregate. The discrepancy is a scale mismatch, not a contradiction.

6.4 Account-Level Robustness

Table 8. Commit behaviour, high-confidence adopters only (25 treated, 202 controls)

Outcome	N treated	Coef	SE	p
Commits / active week	25	15.686	3.749	<0.001
Inter-commit hours	25	−334.577	31.447	<0.001
Active weeks	25	−13.534	1.527	<0.001
Message length (chars)	25	104.122	39.864	0.009
Conventional commits	25	0.053	0.051	0.299
PR has body	25	0.319	0.103	0.002
Test co-write rate	25	0.189	0.077	0.015

The first robustness specification restricts the treated group to the 25 high-confidence adopters (those identified via co-author trailer, with a known adoption date). All primary results hold, and point estimates are larger for commits per active week (+15.7 vs +13.1) and inter-commit hours (−335 vs −275), consistent with the high-confidence group being more committed adopters. Every outcome that was significant in the main specification remains significant at $p < 0.05$, and the message-length effect roughly doubles (+104 characters).

Table 9. Commit behaviour, winsorised at the 5th and 95th percentiles

Outcome	Main coef	Wins. coef	Change	Wins. p	Significance
Commits / active week	13.073	8.042	−38%	<0.001	Unchanged
Inter-commit hours	−275.258	−224.850	−18%	<0.001	Unchanged
Active weeks	−11.253	−9.345	−17%	<0.001	Unchanged
Message length (chars)	54.259	23.714	−56%	0.090	Lost
Conventional commits	0.076	0.049	−36%	0.094	Unchanged
PR has body	0.322	0.279	−13%	<0.001	Unchanged
Test co-write rate	0.144	0.113	−21%	0.024	Unchanged

Outcome and pre-period control variables are clipped at the 5th and 95th percentiles. Significance compares $p < 0.05$ before and after winsorisation.

To assess sensitivity to outliers, a particular concern given only 33 treated accounts, we re-estimate the commit-level difference-in-differences after winsorising all outcome and pre-period control variables at the 5th and 95th percentiles (Table 9).

The headline outcome attenuates the most: commits per active week falls from 13.1 to 8.0 (−38%). Inter-commit hours (−275 to −225), active weeks (−11.3 to −9.3), PR has body (+0.32 to +0.28) and test co-write rate (+0.14 to +0.11) are more stable, shrinking by roughly 13–21%. Message length shrinks by more than half (+54 to +24 characters) and is no longer significant ($p = 0.09$). No outcome changes sign.

This pattern confirms that a minority of high-activity adopters contribute disproportionately to the headline effect sizes, and strengthens the case for treating the main estimates as upper bounds. Even under winsorisation, the primary outcomes remain practically large: roughly 8 additional commits per active week, and a mean gap between commits about 9 days shorter.

6.5 Country-Level Panel Regression

The country-level panel uses 34 countries (those with at least 15 scored accounts in the population sample and at least 5 developers per country-year in the GH Archive panel) across 2022, 2023 and 2024. Table 10 reports the four specifications described in Section 5.2.

Table 10. Country-level panel regression results

Spec	DV	N	Coef	SE	p	R^2
A — Oxford IV (Phase 1)	log_commits	88	0.0667	0.0896	0.4618	0.0345
C — per-country IV	log_commits	72	−5.1431	4.4082	0.2512	0.0281
C-W — per-country IV (weighted)	log_commits	72	−7.5598	3.7473	0.0514	0.0639
D — parallel trends	Δ log_commits (22→23)	9	−5.0403	9.7473	0.6210	0.0368

Countries in Regression C: 34. Country-year observations: 72. Country and year fixed effects; standard errors clustered by country. Regression D is a cross-sectional OLS on countries observed in both 2022 and 2023.

Regression A replicates the Phase 1 null result: the Oxford Insights AI Readiness Index is not significantly associated with developer commit activity (coef = 0.067, p = 0.46). This is expected, since the index measures government AI policy readiness, a distal proxy for developer tool adoption.

Regression C, using per-country classifier scores from the 4,824-account population sample, is the primary specification. The coefficient is −5.14 (SE = 4.41, p = 0.25), not statistically

significant. Weighting by the number of developers in each country-year (Regression C-W) gives a larger negative coefficient of −7.56 (SE = 3.75), which is borderline (p = 0.051). Countries with higher adoption scores do not show higher commit activity; if anything, the association is negative.

The parallel-trends diagnostic (Regression D) finds no relationship between the 2022⟶2023 change in commits per developer and the 2024 adoption score (p = 0.62). This is a weak test: only 9 countries meet the minimum-developer threshold in both 2022 (Q4 only) and 2023.

6.6 Country-Level Robustness Checks

We stress-test the Regression C / C-W result with 10 specifications that vary how the adoption measure is built, the dependent variable, weighting, and whether we use the panel or a 2024 cross-section. Results are summarised in Table 11.

The coefficient on commits per developer is negative in every specification, but only borderline significant at best. It stays negative when:

- Switching from the mean classifier score per country to a threshold-based IV (`pct_above_0.5`) or a median classifier score per country.
- Estimating an honest 2024-only OLS cross-section with no panel machinery (coef = −5.06, SE = 3.62, p = 0.17) and a weighted variant (coef = −6.64, p = 0.06).

None of the unweighted specifications is significant at the 5% level, and the two weighted commit specifications are borderline (p = 0.05 and 0.06).

However, the dependent variable matters more than any specification choice. When we replace `log(commits_per_dev + 1)` with `log(prs_per_dev + 1)`, the coefficient flips sign and is statistically indistinguishable from zero (coef = +1.33, SE = 4.46, p = 0.77). This difference between dependent variables is the central robustness finding and is discussed in Section 6.7.

Permutation placebo. Using the 2024 cross-section, we randomly reassign the adoption score across countries 1,000 times and estimate the coefficient in each shuffled sample. The two-sided permutation p-value is 0.15: a coefficient as negative as the one we observe is not unusual under random assignment. With only 31 countries and a narrow range of adoption scores, the test has little power either way.

Parallel trends diagnostic. The pre-period (2022⟶2023) change in commits per developer does not predict 2024 adoption (Regression D, p = 0.62). This is consistent with parallel trends, but with 9 countries it cannot rule out a confounder that drives both the pre-trend and adoption.

Table 11. Country-level robustness checks

Specification	DV	N	Coef	SE	p
Mean IV (commits)	log_commits	72	−5.143	4.408	0.251

Specification	DV	N	Coef	SE	p
Mean IV, weighted (commits)	log_commits	72	−7.560	3.747	0.051
Threshold IV (commits)	log_commits	72	−21.718	14.012	0.130
Median IV (commits)	log_commits	72	−4.542	8.215	0.584
Mean IV (PRs)	log_prs	72	+1.326	4.458	0.768
Mean IV, weighted (PRs)	log_prs	72	+1.573	5.279	0.767
Mean IV (total events)	log_events	72	−7.588	4.586	0.107
Mean IV, weighted (total events)	log_events	72	−8.693	5.874	0.148
2024 cross-section OLS (commits)	log_commits	31	−5.060	3.616	0.172
2024 cross-section WLS (commits)	log_commits	31	−6.640	3.401	0.061

Permutation placebo (2024 cross-section, 1,000 permutations): observed coefficient −5.06, two-sided p = 0.153.

6.7 Dependent Variable Heterogeneity

The single most informative robustness check is the dependent variable. Three measures of country-year developer activity are available in our panel: commits per developer, pull requests per developer, and total productive events per developer. Under any account of AI adoption that affects developer *productivity*, all three should move in the same direction. They do not (Table 12, Figure 4).

Table 12. Regression C by dependent variable

Dependent variable	N	Coef	SE	p
log(commits per developer + 1)	72	−5.143	4.408	0.251
log(PRs per developer + 1)	72	+1.326	4.458	0.768

Dependent variable	N	Coef	SE	p
log(total events per developer + 1)	72	−7.588	4.586	0.107

Unweighted panel specification with country and year fixed effects, as in Table 11.

Pattern by outcome:

- Commits: negative in every specification, borderline significant only when weighted.
- PRs: near zero, slightly positive.
- Total events: negative, because commits make up most recorded events (62% in the panel).

The PR result is a null: pull requests per developer show no detectable association with AI adoption in this panel, although its confidence interval is wide. Total events follows commits. The negative sign on commits per developer is therefore *commits-specific*, not a general productivity effect.

Plausible interpretations. We list these in order of how strongly the data support them; we cannot identify between them with this design alone.

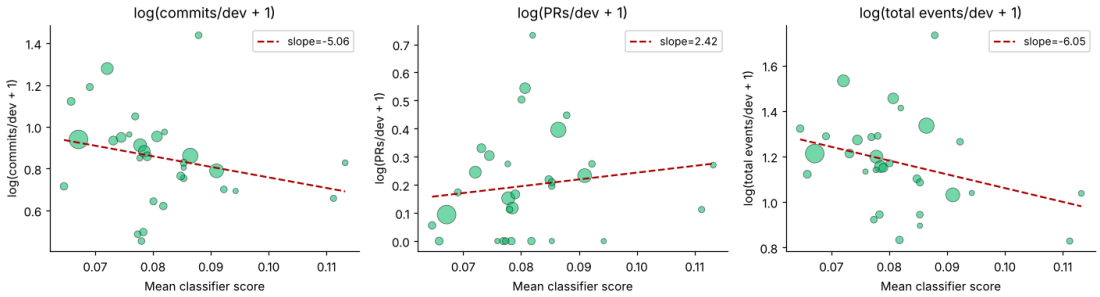
1. **Commit packaging shift.** AI tools, and the workflows they encourage (conversational sessions, longer reviews, squash-merge habits), may change how work is packaged into commits without reducing output. At the account level, adopters' total commits fall relative to controls while their PR output rises (Sections 6.2 and 6.3), which is the same split the country panel shows. PRs per developer would be less affected because PR cadence is driven by feature scope, not commit cadence.
2. **Selection on developer experience or style.** Higher-adoption countries may have a higher proportion of developers who commit less often for reasons unrelated to AI. The classifier partly captures pre-existing style (Section 7.4), so its country averages may too.
3. **Genuine negative effect on commit cadence.** AI tools may slow individual commits (more time per commit reviewing AI output) without changing output as measured by PRs. This is consistent with METR (2025)'s finding of a 19% slowdown for experienced developers on mature codebases.
4. **Statistical artefact.** With 31 countries in the 2024 cross-section and a narrow range of adoption scores, the design is underpowered. The permutation p-value of 0.15 means the negative coefficient is not unusual under random assignment.

We are not in a position to pick between these from country-level data alone. The account-level evidence lends some weight to interpretation 1, but does not establish that this mechanism explains the country-level coefficient.

The country-level null on PRs per developer sits alongside a large account-level PR effect. The country panel finds `log(prs_per_dev + 1)` effectively flat (coef = +1.33, p = 0.77), while the account-level difference-in-differences finds large increases in opened and merged PRs (+46 and +43, $p < 0.001$). The two differ in unit of analysis and treatment intensity. At the country level, the treatment is a small adoption score spread across all developers; at the account level, it is confirmed adoption compared with confirmed non-adoption. Neither result is wrong. The country panel tells us that aggregate PR rates do not rise measurably with aggregate adoption; the account panel tells us that the adopters themselves produce more PRs.

Both are interpretable once one distinguishes *average* effects from *average treatment effects on the treated*.

Figure 4. 2024 cross-section by activity measure (point size ∝ number of developers)



6.8 Statistical Power

Regression C: coefficient −5.14, 95% CI [−14.1, 3.8]. Minimum detectable effect at 80% power: 12.3 log points per unit of the adoption score. Standard deviation of the 2024 adoption score across countries: 0.0110. A one-SD difference in adoption would need to shift commits per developer by about 15% to be reliably detected.

The country-level design has limited power for three reasons. First, although the panel has 72 country-year observations, the adoption score is zero for every country before 2024, so the variation that identifies the coefficient is essentially the 2024 cross-section of 31 countries (Section 7.4). Second, the adoption score varies little across countries: its standard deviation in the 2024 cross-section is about 0.01. Third, the dependent variable is noisy, with a median of 5 located developers per country-year before the minimum-developer filter.

The result is a wide confidence interval. The primary coefficient of −5.14 has a 95% confidence interval running from about −14.1 to +3.8, and an effect would need to be roughly 2.8 standard errors (about 12 log points per unit of the adoption score) to be detected 80% of the time. In practical terms, a one-standard-deviation difference in adoption would need to move commits per developer by around 15% to be reliably detected. Effects of the size the account-level analysis implies, diluted across a population in which few developers are adopters, are well below that threshold.

Future work should expand the GH Archive sample per country and lengthen the post-period, so that adoption varies over time within countries as well as across them.

6.9 Heterogeneity by Developer Experience

Whether AI coding tools affect developers differently depending on their experience is important for interpreting the aggregate results. We cannot support a fully identified heterogeneity analysis, but we can split the commit-level sample by a simple proxy: pre-period commit count, in terciles. Table 13 re-estimates the commits-per-active-week effect within each tercile.

Table 13. Commits per active week by pre-period experience tercile

Experience tercile	Pre-period commits	Adopters	Controls	Coef	SE	p
Low	10–30	11	69	15.33	6.75	0.023
Medium	31–78	10	69	6.86	2.15	0.001
High	80–652	12	64	16.03	4.42	<0.001

Same estimator as Table 5, run separately within each tercile of pre-period commit count.

The increase in commits per active week appears in every tercile: +15.3 for the least active third, +6.9 for the middle third, and +16.0 for the most active third. There is no monotone pattern, so the data favour neither the hypothesis that AI tools mainly help less experienced developers nor the hypothesis that gains concentrate among experienced ones. With 10 to 12 adopters per tercile, these estimates are imprecise, and because adoption is not randomly assigned across experience levels, subgroup differences could reflect selection rather than different treatment effects.

Our data do not include a direct measure of primary programming language, so we do not examine heterogeneity by language. Properly studying heterogeneity would need either individual-level random assignment (for example, A/B tests at firms) or sources of variation in adoption that are unrelated to developer type.

---

7. Discussion

7.1 Reconciling the Account-Level and Country-Level Results

The account-level results show large, statistically significant behavioural changes in confirmed AI adopters: more commits per active week, much shorter gaps between commits, fewer active weeks, more documented PRs, and substantially more opened and merged pull requests. The country-level results are weaker. Pull requests per developer show no detectable association with adoption ($p = 0.77$). Commits per developer show a negative association in every specification, though it is at best borderline significant (Section 6.6).

We argue these results are most plausibly reconciled by a **commit packaging hypothesis**: AI tools change how developers package work into commits, concentrating activity into fewer, more intense sessions, without reducing how much accepted work they produce. Three strands of evidence are consistent with this:

1. Within adopters' own behaviour, total commits fall relative to controls while commits per active week rise and pull-request output grows. Commit counts and shipped work move in different directions.
2. At the country level, PR cadence (which is driven by feature scope) shows no association with adoption, while commit cadence shows a negative one.
3. The total-events measure follows commits because commits dominate the count; removing commits would eliminate most of the effect.

This is not the only explanation. Alternatives include selection on developer experience or style, a genuine slowdown in commit cadence (consistent with METR (2025)), and a statistical

artefact from a low-power 2024 cross-section. Our data do not let us discriminate between these; we present the packaging hypothesis as a *plausible reading* of the data, not an established finding.

The PR results also explain why the country-level PR null does not contradict the account-level gain. The PR-volume effect is concentrated among a small share of developers and is diluted below detectability in country-level aggregates. Future work with larger treated samples or finer-grained adoption measures could sharpen this distinction.

A separate measurement issue compounds the difficulty. The country-level commit activity measure is constructed from GH Archive hourly samples, yielding a median of 2 located developers per country-quarter and 5 per country-year (Section 3.4). With this level of noise in the dependent variable, even a substantial true effect on PRs per developer might not be detectable; the null should be read as bounded by this measurement constraint, not as a strong claim of zero effect.

7.2 The Classifier as a Measurement Tool

A finding independent of the commit behaviour question is the demonstration that AI coding tool adoption is detectable from public commit behaviour at AUC 0.94, and that this detection generalises across tools. This has implications for future measurement.

Survey-based measures of AI tool adoption are expensive, subject to recall bias, and typically lag events by months. Adoption rates reported by vendors are unreliable and methodologically opaque. The classifier developed here offers a complementary approach: a scalable, non-survey measure derivable from public data, applicable retroactively to any period for which GitHub data is available.

The ablation result — that the classifier achieves AUC 0.91 using only activity features, without any message content — is particularly relevant for privacy considerations. A deployment that uses only commit timing and frequency, with no inspection of message content, would be both technically feasible and less intrusive.

7.3 What the Behavioural Signature Means

The dominant features in the classifier — reduced inter-commit hours, increased commits per active week, and longer commit messages — paint a consistent picture of AI adopters committing more frequently within active sessions and writing more structured commit messages. Whether this corresponds to higher productivity or merely a different commit style is the central interpretive ambiguity of this paper, and we have argued in Section 7.1 that the country-level data are insufficient to discriminate the two. The account-level PR results tilt the balance toward productivity: adopters are not only committing differently but also opening and merging more pull requests.

7.4 Limitations

Several limitations bear on the interpretation of our results.

Classifier confound. A pre-period placebo test (computing classifier scores using only pre-2024 features) shows that 6 of 8 features differ significantly between positives and negatives in the pre-period. The classifier therefore captures some combination of (a) AI adoption and (b) pre-existing developer conscientiousness or style. We attempted to address this with a `baseline_log_commits` control in the country-level regression, but the control is fully

absorbed by the country fixed effects and contributes nothing. The country-level IV therefore retains a conscientiousness component, and the negative coefficient on commits per developer may partially reflect cross-country variation in the prevalence of careful developers rather than AI adoption per se.

IV is structurally close to a 2024 cross-section. Because pct_ai_users is set to 0 for 2022 and 2023 and to a country-specific value for 2024, with country and time fixed effects the variation that identifies the coefficient is essentially the 2024-only cross-section across 31 countries. We report the 2024 OLS specification (Section 6.6) alongside the panel results to make this transparent. The panel framing improves precision (and tests parallel trends) but does not add new identifying variation beyond the 2024 cross-section.

Country-level commit activity panel is thin. The median country-quarter cell has 2 located developers (5 per country-year), severely limiting power for the country-level regression. Larger samples per quarter would substantially improve precision.

Different cohorts for commits and PRs. The commit-level analysis uses the original 33 adopters, while the PR analysis uses the expanded cohort of 73 adopters with PR data. The two sets of results are consistent in direction, but they are not estimated on identical samples. Post-period windows also run from January 2024 to the date of each scrape, and commit histories are capped at 500 commits per account.

Account-level effects are upper bounds. The account-level treatment group consists of confirmed adopters discovered through explicit markers (CLAUDE.md files, co-author trailers). These are likely the most engaged AI adopters; effects on the marginal adopter may be smaller. The estimates should be read as approximate upper bounds on the treatment effect at the population level.

Single-tool training, partial cross-tool validation. The classifier is trained on Claude Code adopters and validated on Aider users (mean predicted probability 0.73, substantially above the negative-class mean of 0.03). This supports cross-tool generalisation but does not establish that the classifier identifies users of every AI coding tool. Tools that leave less of a behavioural fingerprint (e.g. inline autocomplete used sparingly) may be undercounted.

Commit packaging hypothesis is interpretation, not evidence. Section 7.1 proposes that the split between commits and PRs most plausibly reflects a change in how work is packaged into commits. This is consistent with the data but not established by it. The account-level PR data include lines changed per PR, but that measure is too noisy to be informative here (Table 6). Distinguishing packaging from output effects more directly would require task-completion, CI or issue-resolution measures.

8. Conclusion

We make two contributions. First, we develop a behavioural classifier for AI coding tool adoption that achieves a cross-validated AUC of 0.94 on public GitHub commit data (0.93 on an expanded training set) and generalises across tools (mean predicted probability 0.73 on Aider users). The classifier requires no survey data and no proprietary telemetry, and it works without direct inspection of commit message content (an activity-only ablation achieves AUC 0.91). It is a measurement tool that can be applied at scale to estimate AI adoption in any developer population visible on GitHub.

Second, we use it in two empirical designs. The account-level difference-in-differences finds large, statistically significant behavioural changes in confirmed AI adopters relative to controls: they concentrate their work into fewer, more intense weeks, and they open and merge substantially more pull requests (+46 opened and +43 merged, $p < 0.001$). The country-level panel regression shows divergent results across dependent variables: commits per developer is negatively associated with adoption in every specification we tested, though at best borderline significant (weighted coefficient −7.56, $p = 0.051$), while pull requests per developer shows no detectable effect (coefficient +1.33, $p = 0.77$).

We interpret these results as most plausibly reflecting a change in how work is packaged into commits rather than a fall in output: adopters commit in fewer, more intense bursts while shipping more pull requests. The country-level PR null is best read as a scale mismatch, with a concentrated effect among a small share of developers diluted in national aggregates. We emphasise that this interpretation is consistent with the data but not established by it; alternatives (selection on experience, a genuine slowdown in commit cadence, a statistical artefact of low cross-country variation) cannot be ruled out from these data alone.

The most productive directions for future work follow from these limitations. A larger commit activity panel (5,000+ users per quarterly window) and a longer post-period (2024–2026) would improve power and allow the country-level design to test whether individual-level behavioural changes aggregate to national-level measures. Distinguishing packaging from output effects needs measures GH Archive does not provide, such as task completion, issue resolution or CI signals. The classifier provides a ready-made adoption measure for those studies.

Acknowledgements

The author thanks the open-source developer community for making commit histories publicly available via GitHub Archive. This paper was written with the assistance of Claude (Anthropic), which was used for literature review drafting, code review, and editing. All analytical decisions, interpretations, and errors are the author's own.

References

9. Code and Data Availability

A PDF version of this paper is available.

All analysis code and derived data are in the public repository github.com/AndreasThinks/ai-productivity-analysis. It includes:

- `notebooks/research_paper_v3.ipynb`: the notebook version of this paper
- `scripts/`: the scraping, classifier training (`train_classifier.py`, `retrain_classifier_expanded.py`), Aider validation (`validate_aider.py`), panel construction (`build_panel.py`, `build_panel_v2.py`), robustness (`robustness_checks.py`) and pull-request outcome (`pr_outcome_metrics.py`) code
- `data/`: the derived datasets every table and figure here is computed from, including the per-account classifier features and predictions, the population classifier scores, the pull-request outcomes, and the country panels (`github_panel_flat.csv` and `panel_dataset.csv`)

The repository does not pin dependencies. The paper notebook needs Python 3.11 with `pandas`, `numpy`, `scipy`, `statsmodels`, `linearmodels` and `matplotlib`. To reproduce the tables and figures:

```
git clone https://github.com/AndreasThinks/ai-productivity-analysis.git
cd ai-productivity-analysis
pip install pandas numpy scipy statsmodels linearmodels matplotlib jupyter
jupyter nbconvert --to notebook --execute notebooks/research_paper_v3.ipynb
```

Re-running the scrapes requires a GitHub API token and access to GH Archive, and the results will differ as accounts change over time.

Bibliography

Angrist, J. D., & Pischke, J.-S. (2009). *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton University Press.

Brynjolfsson, E., Li, D., & Raymond, L. R. (2025). Generative AI at Work. *Quarterly Journal of Economics*, *140*, 889–942.

Cui, Z., Demirer, M., Jaffe, S., Musolff, L., Peng, S., & Salz, T. (2025). The Effects of Generative AI on High-Skilled Work: Evidence from Three Field Experiments with Software Developers. *Management Science*. https://demirermert.github.io/Papers/Demirer_AI_productivity.pdf

GitHub. (2023,). *Survey reveals AI's impact on the developer experience*.

He, H., Miller, C., Agarwal, S., Bogart, C., & Herbsleb, J. D. (2025). Speed at the Cost of Quality: How Cursor AI Increases Short-Term Velocity and Long-Term Complexity in Open-Source Projects. *Arxiv Preprint Arxiv:2511.04427*. https://arxiv.org/abs/2511.04427

Liu, Y., & Wang, H. (2025). *Who on Earth Is Using Generative AI? Global Trends and Shifts in 2025* (Number 11231). https://doi.org/10.1596/1813-9450-11231

METR. (2025). Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity. *Arxiv Preprint Arxiv:2507.09089*. https://arxiv.org/abs/2507.09089

Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. *Arxiv Preprint Arxiv:2302.06590*. https://arxiv.org/abs/2302.06590

Quispe, A., & Grijalba, R. (2024). Impact of the Availability of ChatGPT on Software Development: A Synthetic Difference in Differences Estimation using GitHub Data. *Arxiv Preprint Arxiv:2406.11046*. https://arxiv.org/abs/2406.11046

Ziegler, A., Kalliamvakou, E., Li, X. A., Rice, A., Rifkin, D., Simister, S., Sittampalam, G., & Aftandilian, E. (2024). Measuring GitHub Copilot's Impact on Productivity. *Communications of the ACM*, *67*(3), 54–63. https://doi.org/10.1145/3633453