LIFE AFTER BENCHMARK SATURATION: A CASE STUDY OF CORE-BENCH

Anonymous authors
Paper under double-blind review

ABSTRACT

When a benchmark's accuracy saturates, it is often retired and replaced with a more challenging version. We show that this approach privileges accuracy and misses the opportunity to study other key dimensions of agent performance: construct validity issues such as shortcuts, out-of-distribution generalizability, efficiency, reliability, the relative importance of the model versus the scaffold, and uplift in duration from human-agent collaboration. We use CORE-Bench Hard, a benchmark for computational reproducibility of scientific code, as a case study to demonstrate that measuring agents along these dimensions yields meaningful insights into agent performance even after accuracy saturates. First, we surface threats to construct validity in CORE-Bench Hard that are difficult to anticipate with less capable agents. We introduce an improved benchmark, CORE-Bench v1.1, and an out-of-distribution task suite, CORE-Bench Extended. Second, we find that despite accuracy saturation, CORE-Bench v1.1 remains useful for measuring efficiency, reliability, model performance, and scaffold performance. Finally, we conduct a small-scale randomized study to measure uplift in duration from human-agent collaboration on real-world computational reproducibility tasks. We find a statistically significant speedup by about a factor of two — likely underestimated due to one-fifth of human-only reproductions reaching the time limit — and describe various other findings. Together, our contributions present a more rigorous alternative to the dominant accuracy-centric evaluation paradigm.

1 INTRODUCTION

AI agents are increasingly deployed across a wide range of domains, including customer service (Yao et al., 2025), software engineering (Anthropic, 2025), legal services (Harvey, 2022), financial analysis (Endex, 2026), and scientific discovery (Lu et al., 2026). As these systems proliferate, benchmarks have become the standard tool for comparing performance across vendors and over time. Most benchmarks distill performance into a single headline metric: overall accuracy, defined as the proportion of tasks an agent solves correctly. This metric has shown steady improvement for years, but on many widely used benchmarks, progress has begun to plateau. Top agents now cluster near ceiling-level scores and are often statistically indistinguishable from one another (Akhtar et al., 2026; Jimenez et al., 2024; Chollet, 2019; Chen et al., 2021; Hendrycks et al., 2021). Many in the field interpret this as evidence that such benchmarks have lost their discriminative power. The prevailing response has been to retire these benchmarks in favor of more difficult successors; for example, ARC-AGI 1 progressing to ARC-AGI 2 and 3, MMLU to MMLU-Pro, HumanEval to HumanEval+, and SWE-bench to SWE-bench Pro (Chollet et al., 2026; Foundation, 2026; Wang et al., 2024; Liu et al., 2023; Deng et al., 2025). We refer to this recurring pattern as *retire-and-replace*.

In this paper, we argue that although this strategy may be useful for model developers who focus on optimizing relative accuracy, it is fundamentally inadequate for helping researchers and downstream developers understand how well an agent solves a real-world task. A central thesis of our work is that *accuracy saturation*, i.e., the state in which top-performing agents achieve statistically indistinguishable accuracies, does not imply that there exist no further insights into performance across all meaningful dimensions of agent behavior. We demonstrate that even when a benchmark's accuracy metrics have plateaued, we can obtain useful information on agent performance along other critical axes. These include (i) *benchmark validity*, i.e., whether high scores reflect genuine task mastery rather than exploited shortcuts or overfitting; (ii) *evaluation completeness*, capturing reliability,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

computational efficiency, and the relative performance of the model versus its scaffolding; and (iii) the *practical impact on human workflows*. Although prior work has advocated for evaluating these multifaceted dimensions in principle (Kapoor et al., 2025; Rabanser et al., 2026; Wang et al., 2025; Liang et al., 2023; Kapoor et al., 2026), the field has largely defaulted to developing increasingly difficult benchmark successors that continue to optimize solely for accuracy. By challenging the *retire-and-replace* paradigm, we argue for extracting the rich signals that persist beyond a benchmark's accuracy ceiling and emphasize that the limits of accuracy-centric evaluation are present throughout a benchmark's lifecycle, not just at saturation.

We study this claim through CORE-Bench Hard (Siegel et al., 2024), a benchmark for computational reproducibility. CORE-Bench Hard is a useful case study because reproducibility is a high-value, real-world task with a direct human counterpart (enabling a concrete human uplift study), out-of-distribution axes (e.g., changing the research fields), and multiple relevant performance dimensions (e.g., correctness, cost, latency, and reliability). We make the following three contributions.

1. **New CORE-Bench variants to improve benchmark validity (Section 2).** We use log analysis to uncover 15 task-level errors and 20 tasks with exploitable shortcuts in CORE-Bench Hard that would have been difficult to surface before accuracy saturation. We correct these and add ten new tasks to produce CORE-Bench v1.1, a 39-task suite that preserves CORE-Bench Hard's original disciplines, languages, and construction pipeline. We also test whether saturated accuracy transfers under field distribution shift by introducing CORE-Bench Extended, with 19 tasks covering different disciplines from CORE-Bench Hard: physics, engineering, economics, and computer science. We provide a description of each CORE-Bench variant in Table 7.
2. **Results from multidimensional evaluation (Section 3).** Even after a benchmark loses discriminative power w.r.t. agent accuracy, it remains useful for differentiating performance across other dimensions. Across 20 agent runs, we show that agents with statistically indistinguishable accuracies differ in efficiency, reliability, and model–scaffold behavior.
3. **Observations from measuring the uplift in duration of agent collaboration (Section 4).** While benchmarks are useful proxies for agent capability in task automation, they are insufficient indicators of practical utility for human-agent collaboration. We run a small randomized study on real-world computational reproducibility tasks comprising 20 machine learning and social science papers, and find that agent collaboration more than *halves* completion time. This is likely a conservative estimate, since one-fifth of human-only sessions never completed before reaching the three-hour time limit while all human-agent collaborative sessions did.

2 Accuracy saturation surfaces threats to benchmark validity

Benchmark validity is threatened along two axes. The first is *task-level threats*, where headline metrics do not faithfully measure the intended capability. Recent work has shown that this affects many widely used benchmarks, including SWE-Bench tasks that are impossible to solve (Jimenez et al., 2024; Chowdhury et al., 2024), a τ-Bench Airline scaffold bug (Yao et al., 2025; Kapoor et al., 2026), and WebArena tasks with incorrect grading (Zhou et al., 2024; Zhu et al., 2025). These surface once more capable agents are able to exploit alternative solution paths, uncover subtle shortcuts, or succeed end-to-end but are graded incorrectly. Table 10 illustrates examples of task-level threats in CORE-Bench Hard that surfaced only once accuracy saturated. Log analysis, the tracking of an agent's inputs, outputs, and environment, has emerged as a method for identifying them (UK AISI, 2026), and prior work has used it to uncover benchmark bugs, shortcuts, environmental barriers, and scaffold-level errors (Hamin & Edelman, 2025; Parikh & Wijk, 2025; Kapoor et al., 2026). However, log analysis is not exhaustive: it requires specifying target behaviors, some threats to construct validity surface only in particular agent runs, and LLM-based classifiers require manual validation.

The second axis is *benchmark-specific adaptation*, which arises when benchmarks are used as development targets: as developers iterate on agents against a fixed benchmark, they may adjust prompts, scaffolds, tool-use, dependency handling, timeout settings, or recovery heuristics based on observed failures. These changes improve benchmark performance but can also tailor the agent to the benchmark's idiosyncrasies (e.g., task distributions or output formats). Hence, strong performance may partly reflect adaptation rather than general capability (Kapoor et al., 2025). Motivated by this, we introduce two new task suites: CORE-Bench v1.1, which improves construct validity relative to CORE-Bench Hard, and CORE-Bench Extended, which evaluates out-of-distribution generalization.

Table 2: We analyzed the logs of our top-performing agents using Docent (Meng et al., 2025). Our rubrics were designed to surface threats to construct validity that could either lead to underestimation or overestimation of agent capabilities. We supplemented automated log analysis with manual log inspection of all incorrect tasks and all tasks flagged by the rubric across runs. We conducted log analysis using GPT-5 (medium) and GPT-5.4 (low).

For tasks graded as:	We inspect logs to see whether:
Incorrect	The agent solves the intended task end-to-end and gives a logically or procedurally correct answer based on the environment or reasoning in the transcript.
Correct	The agent either doesn't reproduce the paper correctly (process incorrectness) or doesn't perform the correct final computation (computation incorrectness).
All tasks	The agent is able to obtain the correct answer to a task by reading a value that already exists (pre-run) inside static artifacts or rendered documents, or applying only extremely trivial operations over values in the pre-existing artifacts (for example, very simple filtering-plus-counting or literal pattern-counting in text).

2.1 CORE-BENCH V1.1: A MORE ROBUST MEASURE OF COMPUTATIONAL REPRODUCIBILITY

We introduce *CORE-Bench v1.1*, a corrected benchmark developed by identifying task-level threats to construct validity in CORE-Bench Hard via log analysis.

We construct CORE-Bench v1.1 by applying automated and manual log analysis to the 45 original CORE-Bench Hard tasks and 27 new candidate tasks. Rather than serving as a novel benchmark, CORE-Bench v1.1 repurposes CORE-Bench Hard. We inspect trajectories using Docent (Meng et al., 2025) for process correctness, computation correctness, pre-existing artifact contamination, and grading errors using the rubrics in Table 2. This process removes or leads to edits for tasks with threats to construct validity; these were difficult to surface prior to accuracy saturation, since less capable agents were not progressing far enough into tasks. It yields a final 39-task benchmark: 13 computer science, 10 social science, and 16 medical science tasks. Full construction details are in Section A.1. We provide a visual overview of the benchmark construction process in Figure 4a.

Table 1: **CORE-Bench v1.1 accuracies.** For Claude models, "thinking" denotes reasoning (10K token budget for Opus 4.5; "adaptive" has no budget parameter). `max_thr` controls the maximum number of concurrent Codex CLI subagents (omitting it disables subagents). Accuracies shown as $\text{value}^{\text{upper}}_{\text{lower}}$ with 95% Wilson CI bounds.

Agent scaffold	Model (reasoning effort)	Accuracy
Codex CLI (default)	GPT-5 (`medium`)	$84.6\%^{92.8}_{70.3}$
	GPT-5.1 (`medium`)	$87.2\%^{94.4}_{73.3}$
	GPT-5.2 (`medium`)	$94.9\%^{98.6}_{83.1}$
	GPT-5.3-Codex (`medium`)	$97.4\%^{99.5}_{86.8}$
	GPT-5.4 (`low`)	$92.3\%^{97.3}_{79.7}$
	GPT-5.4 (`medium`)	$94.9\%^{98.6}_{83.1}$
	GPT-5.4 (`high`)	$97.4\%^{99.5}_{86.8}$
	GPT-5.4 (`xhigh`)	$97.4\%^{99.5}_{86.8}$
Codex CLI (`max_thr=1`)	GPT-5.4 (`medium`)	$94.9\%^{98.6}_{83.1}$
Codex CLI (`max_thr=3`)	GPT-5.4 (`medium`)	$97.4\%^{99.5}_{86.8}$
Codex CLI (`max_thr=6`)	GPT-5.4 (`medium`)	$92.3\%^{97.3}_{79.7}$
Codex CLI (`max_thr=9`)	GPT-5.4 (`medium`)	$97.4\%^{99.5}_{86.8}$
Claude Code	Opus 4.5 (thinking)	$89.7\%^{95.9}_{76.4}$
	Opus 4.6 (adaptive)	$92.3\%^{97.3}_{79.7}$
OpenCode	Opus 4.5 (thinking)	$82.1\%^{91.0}_{67.3}$
	Opus 4.6 (`none`)	$82.1\%^{91.0}_{67.3}$
	GPT-5.4 (`high`)	$84.6\%^{92.8}_{70.3}$
CORE-Agent	Opus 4.5 (`none`)	$82.1\%^{91.0}_{67.3}$
	Opus 4.6 (`none`)	$100\%^{100}_{91.0}$
	GPT-5.4 (`medium`)	$51.3\%^{66.1}_{36.2}$

Results. Despite the construct validity improvements introduced in CORE-Bench v1.1, *accuracy saturation persists*: the top agent obtains an accuracy of 100% and the next four agents tie at 97.4% (see Table 1), so accuracy alone no longer distinguishes leading agents. At the same time, our results also highlight the importance of the scaffold, a finding we discuss in more detail in Section 3.3. For example, with GPT-5.4 (medium), Codex CLI outperforms the CORE-Agent scaffold by ≈ 44 pp.

2.2 CORE-BENCH EXTENDED: AN OUT-OF-DISTRIBUTION TASK SUITE

CORE-Bench Extended tests whether performance on CORE-Bench v1.1 transfers under a field distribution shift. This shift is critical, as disciplines vary significantly in repository organization, software ecosystems, manuscript conventions, and computational workflows. While preserving the underlying task structure of v1.1, CORE-Bench Extended changes the disciplinary composition as follows: two economics, ten engineering, five physics, and two computer science tasks (one of which has a runtime of around 50 minutes). Following the same log analysis procedures used for v1.1 (see Section 2.1), we evaluated an initial pool of 30 task-distribution shifted tasks using CORE-Agent

(Opus 4.5 and 4.6) and OpenCode (GPT-5.2). This initial round of removing 12 tasks, editing 8, and adding 6 yielded a 24-task subset. Subsequent log analysis of incorrect tasks across 12 Codex CLI runs identified further errors, prompting the removal of 5 additional tasks and the regrading of one to establish the final 19-task benchmark (see Figure 4b and Section A.1 for details).

Results. We evaluate 12 agents on CORE-Bench Extended. We present results in Table 3, and find that the top five agents obtain statistically indistinguishable accuracies (see Section A.3). *Accuracy saturation on CORE-Bench v1.1 translates across a discipline distribution shift*.

We further note that due to the limitations of log analysis, we treat CORE-Bench v1.1 and CORE-Bench Extended as active benchmarks that we plan to update as new validity threats are found.

3 MULTIDIMENSIONAL AGENT EVALUATION

Accuracy saturation limits the usefulness of a benchmark for distinguishing between agents by accuracy. Consequently, we propose decoupling *accuracy saturation* from *benchmark saturation*: we show that extending a benchmark's lifecycle beyond accuracy to measuring additional dimensions of agent performance retains their utility even after accuracy saturates. While accuracy-centric evaluations are insufficient measurement tools even before accuracy saturates, saturation highlights the immediate necessity of moving beyond them.

3.1 RELIABILITY

Two agents with identical mean accuracy can differ in how consistent their outputs are across repeated runs and in how well their stated confidence anticipates success. We adopt the reliability framework of Rabanser et al. (2026) and measure four metrics: *outcome consistency* (rate at which repeated runs of a task yield the same verdict), *resource consistency* (variability in tokens), *calibration* (gap between stated confidence and empirical success), and *discrimination* (whether confidence rankings separate successes from failures).[1] We run five additional trials on each of five Codex CLI agents (GPT-5, GPT-5.1, GPT-5.2, GPT-5.3-Codex, and GPT-5.4, all at medium reasoning), eliciting post-hoc confidence via an additional prompt at the end of the agent interaction (see Section A.8).[2] Based on Figure 1, we draw two conclusions:

Table 3: **CORE-Bench Extended accuracies.** Accuracies shown as $\text{value}^{\text{upper}}_{\text{lower}}$ with 95% Wilson CI bounds.

Agent scaffold	Model (reasoning effort)	Accuracy
Codex CLI (default)	GPT-5 (`medium`)	$89.5\%^{97.1}_{68.6}$
	GPT-5.1 (`medium`)	$94.7\%^{99.1}_{75.4}$
	GPT-5.2 (`medium`)	$100.0\%^{100.0}_{83.2}$
	GPT-5.3-Codex (`medium`)	$89.5\%^{97.1}_{68.6}$
	GPT-5.4 (`low`)	$84.2\%^{94.5}_{62.4}$
	GPT-5.4 (`medium`)	$89.5\%^{97.1}_{68.6}$
	GPT-5.4 (`high`)	$89.5\%^{97.1}_{68.6}$
	GPT-5.4 (`xhigh`)	$100.0\%^{100.0}_{83.2}$
Codex CLI (`max_thr=1`)	GPT-5.4 (`medium`)	$94.7\%^{99.1}_{75.4}$
Codex CLI (`max_thr=3`)	GPT-5.4 (`medium`)	$89.5\%^{97.1}_{68.6}$
Codex CLI (`max_thr=6`)	GPT-5.4 (`medium`)	$84.2\%^{94.5}_{62.4}$
Codex CLI (`max_thr=9`)	GPT-5.4 (`medium`)	$84.2\%^{94.5}_{62.4}$

In a small sample, agents that are more accurate on average are also more consistent. The most accurate agent based on the average score across five runs also has the most consistent outputs (dependably correct or incorrect) and the most consistent token usage.

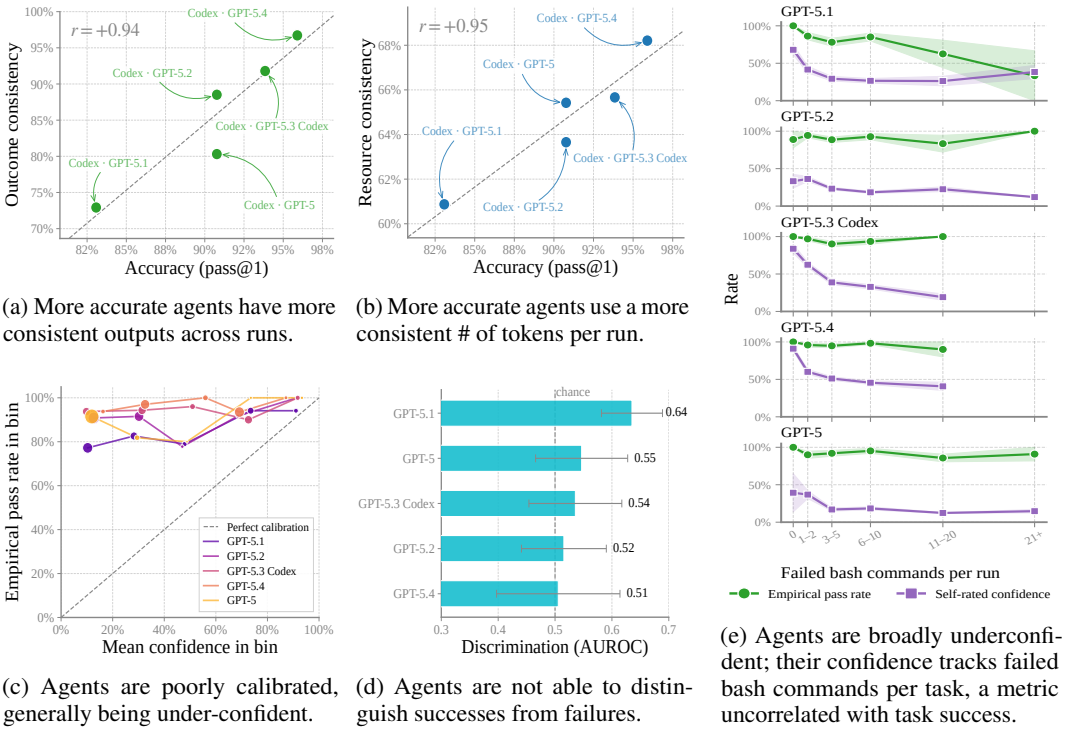
Agents are massively under-confident and struggle to separate correct from incorrect responses. While the mean empirical pass rate across all runs is 93%, the mean reported confidence is only 32.1%. Reported confidence tracks the number of bash tool errors, but this metric is uncorrelated with task success. In fact, no agent appears to be outperforming a simple random guessing baseline at distinguishing correct and incorrect tasks based on confidence.

3.2 EFFICIENCY

The well-documented returns from inference scaling (Brown et al., 2024; Guo et al., 2025) show that agents can often achieve high accuracy simply by using more compute. For researchers, this "brute force" capability is useful for identifying the upper bounds of a model's potential. However, for most

[1] The framework also covers robustness and safety, which we address via the Extended analysis (Section 2.2) and benchmark validity analysis (Section 2).

[2] We use Codex CLI v0.122 for GPT-5.1 and v0.130.0 for all other models. See Section A.4.1 for details.



(a) More accurate agents have more consistent outputs across runs.

(b) More accurate agents use a more consistent # of tokens per run.

(c) Agents are poorly calibrated, generally being under-confident.

(d) Agents are not able to distinguish successes from failures.

(e) Agents are broadly underconfident; their confidence tracks failed bash commands per task, a metric uncorrelated with task success.

Figure 1: Reliability analyses, with estimates and standard errors in Table 25. **(a)** Outcome consistency and **(b)** resource consistency both increase with reliability-sample accuracy, indicating that more accurate agents are also more repeatable across runs. **(c)** Agents are systematically underconfident and **(d)** frequently do not exhibit discrimination better than random chance. **(e)** Per-agent predictability curves: empirical pass rates remain high across tool-error bins, while self-rated confidence declines with failed bash commands.

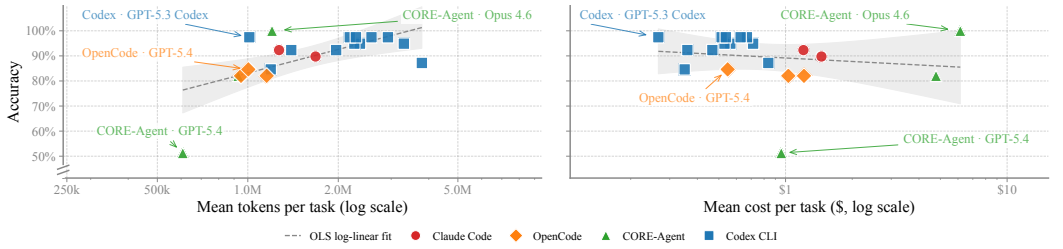


Figure 2: **Accuracy versus total token usage and cost.** Token usage and cost have different relationships with accuracy and similarly accurate agents can differ significantly in their cost.

practitioners, the cost of reaching an answer is just as important as the answer itself. To address this, we analyze efficiency by measuring both token usage and total dollar cost.[3] Token usage includes the sum of all input, cached, and output tokens, while the dollar cost is calculated based on prices at the time of the run. In Figure 2, we plot accuracy against these two metrics. From this data, we find:

Some high-scoring agents are much more efficient than others. Cost-aware analysis allows us to differentiate between our top scoring agents. GPT-5.3-Codex (medium) is most efficient by both token usage and cost. Compared to GPT-5.4 (high), which achieved equal accuracy (97.4%), GPT-5.3-Codex (medium) costs roughly 60% less.

Token usage and cost tell different stories of efficiency, driven by model provider pricing, some Codex CLI model-scaffold pairs caching aggressively, and CORE-Agent not caching at all.

[3]For CORE-Agent with Opus 4.5, we drop two tasks from the mean resource usage calculation. These tasks timed out, so resource usage was not logged.

3.3 DECOUPLING MODEL AND SCAFFOLD

Agent benchmark leaderboards typically report only accuracy, collapsing the contributions of the model and the scaffold that orchestrates it. When accuracy improves from one leaderboard entry to the next, it is therefore unclear whether the gain is attributable to a more capable model, a better-engineered scaffold, or a better match between the two. Our evaluation design provides model-scaffold comparisons that allow us to probe these effects. We evaluate Opus 4.5, Opus 4.6, and GPT-5.4, on three of four scaffolds each. Claude Code is a proprietary, vendor-developed scaffold. CORE-Agent (built on HuggingFace smolagents), OpenCode, and Codex CLI are open-source scaffolds. We provide scaffold configurations in Section A.4 and reasoning configurations vary as per Table 1. We inspected the trajectories of tasks where outcomes varied across models and scaffolds, classified all 56 failures by root cause using Docent (GPT-5.5 and high reasoning), and applied a Docent rubric to all 390 logs to surface trajectory differences. Our analysis reveals three findings:

Similar accuracies can mask different failures. We provide representative examples of these disagreements in Table 4 and Table 5. This persists even when comparing different scaffolds paired with the same model. For example, Opus 4.5 achieves 82.1% accuracy on both CORE-Agent and OpenCode, yet the two scaffolds' outcomes disagree on 12 of 39, or 31% of capsules (see Figure 6). An oracle router that selects the best scaffold per task achieves 100% accuracy for both Opus 4.5 and GPT-5.4, implying that every task in CORE-Bench v1.1 is solvable by at least one scaffold for these models. This complementarity suggests that scaffolds are altering which tasks models can solve and how they solve them.

Table 4: **Root-causes of 56 failures.** Failure modes are unevenly distributed across scaffolds. "Spiraling" timeouts reflect repeated failed fix attempts; "environment" timeouts reflect slow-running processes. CC: Claude Code; Cx: Codex CLI; OC: OpenCode; CA: CORE-Agent.

Failure root cause	**CC**	**Cx**	**OC**	**CA**	**Total**
Wrong metric / computation	2	2	0	14	18
Timeout (spiraling on fixes)	3	0	8	3	14
Gave up (no answer)	0	0	5	2	7
Dependency failure	0	0	6	0	6
Vision / web fallback	0	0	0	5	5
Precision / rounding	0	0	0	2	2
Timeout (environment)	2	0	1	0	3
Format mismatch	0	1	0	0	1
Total failures	7	3	20	26	56

Scaffolds induce distinct solution strategies. Holding the model constant and swapping the scaffold highlights the scaffold-induced differences. Some vision tasks can be solved using code output without rendering the figure. With Opus 4.6, Claude Code derives 41% of answers from the text output of unmodified code (no vision-read) and only 3% from a vision-reading of a rendered figure; the same model with CORE-Agent derives the answer from the text output of unmodified code in just 21% of runs and a vision-read 31% of the time. Vision-read rates jump from 3% (Claude Code) to 62% (CORE-Agent) on Opus 4.5 and from 1% (Codex CLI) to 56% (CORE-Agent) on GPT-5.4. Vision-reads following a clean run pass 93% of the time, but those used as a fallback after the agent abandons the original code (47%) or gives up entirely (60%) roughly pass only 50% of the time. CORE-Agent's accuracy gap is largely the accumulation of these fallback failures.

Direct fixes strongly outperform rewrites. Scaffolds that diagnose a root cause and apply a targeted fix succeed 95.2% of the time ($n = 269$), whereas scaffolds that abandon the original implementation and rewrite succeed only 67.8% of the time ($n = 59$). The 26 capsules where both strategies were attempted show a similar pattern: 96% success for direct fixes versus 68% for rewrites, despite small per-capsule sample sizes. A scaffold's tendency toward direct fixes tracks its overall accuracy: Codex CLI uses direct fixes 82% of the time, while CORE-Agent does so only 49% of the time.

4 MEASURING UPLIFT IN DURATION FROM HUMAN-AGENT COLLABORATION

Computational reproducibility is grounded in scientific workflows where humans interpret, validate, and build on results. Once top-performing agents converge at near-ceiling accuracy, the question shifts from whether agents can complete a task to whether they provide value alongside humans. High benchmark accuracy may not cleanly translate to uplift in duration: agent failures may be more time-consuming for a human (or the agent itself) to resolve than human failures or agents may take more time to effectively respond to human redirection. Prior work shows productivity gains from coding agents are highly context-dependent, often emerging only in human-in-the-loop settings (Wang et al., 2025; Becker et al., 2025). To measure this directly, we ran a randomized study in

Table 5: **Representative trajectory-level disagreements across scaffolds.** The "Model provider scaffold" column reports Codex CLI for GPT-5.4 runs and Claude Code for Opus runs. We provide specific details on how failure modes differed by task in Section A.6.

Capsule	Model	CORE-Agent	Model provider scaffold	OpenCode
1175539	GPT-5.4	**Fail** (8 msgs). Stale environment symlinks and a shallow filesystem search miss the script one directory deeper. Falls back to an unrelated notebook render using a different dataset and extracts the wrong group name.	**Pass** (56 msgs). Permission restrictions block the script's hard-coded absolute paths; after exhausting filesystem workarounds, redirects both input and output paths to the working directory, then runs the original script.	**Pass** (35 msgs). Installs dependencies with sudo, runs the script, and computes group medians with a targeted R command to extract the answer.
4252248	GPT-5.4	**Fail.** (84 msgs). Computes all four benchmark AUCs but selects the sensitivity-layer value instead of the integration-layer value.	**Fail.** (110; 138 msgs). High reasoning selects the sensitivity-layer value; medium reasoning skips preprocessing and reports the wrong value.	**Pass.** (117 msgs). Bioconductor version conflicts block a required package; rather than resolving the full dependency chain, creates a slim local stand-in that defines only the single class needed to load the data, patches R 4.x bugs, and runs the full pipeline.
	Opus 4.5	**Fail.** (128 msgs). rJava fails to load despite installing the JDK and reconfiguring Java; falls back to a simplified computation that skips the preprocessing, producing an incorrect value.	**Fail.** (180 msgs). Cannot compile R's curl package (missing dev headers for installed libcurl4t64); reimplements the benchmarking pipeline standalone with different preprocessing.	**Pass.** (195 msgs). Iteratively resolves Bioconductor version conflicts including a BH downgrade, patches R 4.x compatibility bugs in the paper's code, and runs the full pipeline.
5136217	Opus 4.6	**Pass.** (114 msgs). Discovers `bsts` is unavailable; installs R from scratch and runs the scripts needed for the figure. Reads the answer from the generated plot via a vision model, which returns the wrong group; catches the error by cross-checking against a self-authored Python replication, then verifies via direct R computation.	**Pass** (63 msgs). Traces the figure target to two upstream scripts and runs them, and extracts the answer by computing group means directly in R, never rendering or reading the generated plot.	**Fail** (32 msgs). Attempts to compile `Boom` the `bsts` dependency that isn't needed from source, hitting two bash timeouts, and then times out.
	Opus 4.5	**Pass** (76 msgs). Creates a modified copy of the relevant script with unavailable packages commented out and runs only what is needed to generate the figure.	**Fail** (262 msgs). Largely consumed by dependency installation failures and package workarounds. Derives the correct answer from intermediate data but does not finish before the task timeout elapses.	**Fail.** (46 msgs). Compiles `bsts` successfully, but times out during data processing on the large dataset.
0851068	GPT-5.4	**Fail.** (38 msgs). Data symlinks point to a nonexistent agent run directory, leaving the input folder empty. Rather than repairing the symlinks, searches the web for the paper's reported results and submits an AUC from a different experimental condition.	**Pass.** (69 msgs). Runs the demo script; PyTorch's `DataLoader` crashes because the deep workspace path exceeds the 108-byte `AF_UNIX` socket limit at `num_workers=16`. Diagnoses the socket-path constraint, patches `num_workers=0`, and reruns to completion.	**Pass.** (47 msgs). Hits the same `AF_UNIX` socket-path crash, reaches the same diagnosis independently, and applies the same `num_workers=0` patch to complete.
	Opus 4.6	**Pass.** (66 msgs). Discovers that data symlinks point to a different agent run's directory, deletes and recreates them against the correct path. After installing PyTorch, proactively reduces `num_workers` to 0 before encountering the socket-path error, avoiding the crash entirely.	**Fail** (72 msgs). Diagnoses the `AF_UNIX` socket-path bug, patches `num_workers=0`, and computes the correct AUC, but the 2,700 s timeout elapses before answer collection.	**Pass.** (35 msgs). Most efficient run across both models. Hits the `AF_UNIX` error, patches `num_workers=0`, and completes in 35 messages.

which five evaluators reproduced results from 20 machine learning and social science papers, with and without agent collaboration, to estimate uplift in task duration.

4.1 METHODOLOGY

Paper selection. We selected 20 papers across machine learning and the social sciences. The machine learning papers were drawn from a list of award-winning papers at major machine learning conferences since 2011 (Eppner, 2026). The social science papers were drawn from a dataset published by the Institute for Replication (I4R) (Institute for Replication, 2024). To enhance representativeness, each selector was given a random subset of papers from each dataset in randomized order and evaluated them sequentially for inclusion according to our paper selection criteria (see Appendix A.5.1 for details) until reaching the required number of selections. Unlike CORE-Bench, the purpose of this study was to gain process-level insights into uplift in duration from human-agent collaboration, rather than validate final answer correctness. Accordingly, the papers were not limited to those that are confirmed to be computationally reproducible. We deliberately included two social science papers that I4R had assessed as not achieving a "perfect reproduction" to better reflect real-world computational reproducibility work. A single result from each selected paper was specified as the replication target (see A.5.4 for a full list).

Participants. Five of the authors joined the experiment as evaluators, all of whom have a master's degree in data science and experience with computational reproducibility tasks. Each of the papers (i.e. reproduction tasks) was independently attempted by two or three of the five evaluators. The same five authors who conducted the replication attempts also carried out paper and reproduction target selection. To ensure blinding, no author was assigned as an evaluator for a paper they had encountered during the selection process. For the social science papers, the selector was aware of whether the paper had been assessed as "perfectly reproducible" by I4R replicators, but the evaluator was not. For all the machine learning papers, the reproducibility status was not known beforehand.

Agent configuration. The human-agent collaboration condition used Codex CLI with GPT-5.4 (xhigh). Participants used a standardized interface but were otherwise free to interact with the agent. We constructed one Docker-based evaluation environment for machine learning papers and one for social science papers. Each had tailored Python and R support. We applied a standardized prompt (see Section A.5.3) using a fully autonomous execution setting in which the agent iteratively generates

and runs code without intermediate human approval. The prompt instructed the agent to stop and escalate to the human when encountering blockers it could not resolve after 2-3 attempts.

Paper replication. We randomly assigned these 20 papers across the five evaluators (see Section A.5.7). Each participant attempted 10 papers total, 5 with agent assistance and 5 without, and 5 from each source dataset. Each paper was attempted by two or three participants, with at least one manual and one human-agent collaboration attempt, yielding 50 replication experiments across the 20 papers. To mitigate learning effects, participants were asked to complete tasks in a pre-specified randomized order. In the manual condition (see Section A.5.5), participants were allowed to use traditional web search tools (e.g., Google, StackOverflow) but were prohibited from using generative AI systems or AI search summaries (e.g., ChatGPT, Copilot, or AI overviews in search engines), consistent with prior experimental protocols (Becker et al., 2025; Hong et al., 2026). In the human-agent collaboration condition, participants applied a shared prompt template (see Section A.5.3) uniformly across tasks. We set a maximum time limit of 3 hours per run for both conditions.

Task questionnaire. We adopted design approaches used in prior work to design a questionnaire for documenting agent failure modes and instances of human intervention as structured feedback (Paradis et al., 2025). Our questionnaire is provided in Section A.5.9.

4.2 Results

Our results show that human-agent collaboration provides substantial uplift in duration on computational reproducibility tasks compared to humans alone. Specifically, we find:

1. **Human-agent collaboration provides uplift in reproduction time.** Our fixed effects model with CR2 standard errors (see Section A.5.8) estimates that manual reproduction sessions lasted 2.11 times as long as human-agent collaborative sessions, with a 95% confidence interval of [1.62, 2.75]. The (two-sided) p-value is 0.00176, indicating a statistically significant positive result. The three-hour time limit was reached for five out of 25 manual runs and none of the human-agent runs, suggesting that without this constraint, the estimated uplift in duration would likely be larger (see Figure 3). We also note that the uplift estimate primarily reflects minimal human intervention.
2. **Most human-agent collaborative runs required only minimal or no human assistance.** Across 25 human-agent collaborative runs, evaluators reported that the agent was able to complete 19 fully autonomously (aside from two setup steps explicitly assigned to humans: starting the instance and Docker image, and starting the agent). In the remaining six runs, humans intervened mainly during setup, code execution, result comparison, and discrepancy investigation. These interventions ranged from minimal human input to complete redirection (see Table 6 for a full list).
3. **Agents were perceived to be the most useful in environment setup and running code.** After each human-agent collaborative reproduction session, the human evaluators were asked to assess "Where Agent [had] added value". The most frequent responses (see Table 19) were environment setup (25 of 25 sessions), running code (23), identifying main scripts (20), and navigating the README and related files quickly (19). While not every reproduction required fixing errors, agents were perceived as adding value in such situations as well (e.g. "Debugging errors from running code as is" in 14 sessions). See Section A.5.10 for a few concrete examples where agents were able to resolve such operational blockers without human intervention.
4. **The agent logged blockers more often than humans on the agent-only runs, but recovered more reliably.** Agents recorded at least one blocker category humans did not in 18 of the 19 papers where the agent was able to complete reproduction on its own. Across 34 paper-blocker category pairs, half fell in tooling or environment, which agents resolved without human intervention. On 39 occasions, the agent and the human encountered the same blocker category in a particular paper pair. Out of these, there were 11 instances where the agent fully recovered while the human only partially recovered or did not recover at all, six where the agent partially recovered while the human fully recovered (there were no instances where the agent completely failed to recover), and 22 where recovery (either full, partial, or none) was tied. The agent fully or partially resolved all but 2 of the 114 individual blockers it encountered, while humans left 11 of 60 unresolved.

Evaluators answered questions including which steps of the process had been performed solely by the agent and its success (see Table 16), types of struggles the agent encountered (see Table 20). Complementing the AI-assisted analysis of the session logs reported in the fourth finding above (see

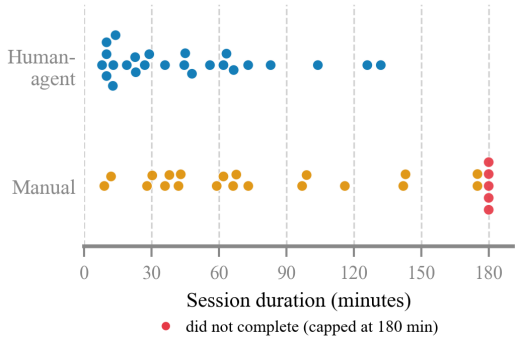


Figure 3: **Durations of reproduction sessions**. Evaluators were instructed to abandon runs if no result had been produced after three hours, a limit that was only reached during manual sessions.

Table 6: **Observed collaboration patterns across 25 human-agent collaborative reproduction runs.**

* Originated from the same run.

Collaboration pattern observed	Runs
Agent did all the work on its own	19
Minor human suggestions or redirection*	3
Agent asked for human input less than 5 times*	1
Agent made major error(s), requiring human redirection	1
Agent completed task but required significant scope clarification upfront	1
Agent wasted a lot of time going down the wrong path, but eventually stopped to check in with the human (as requested in the prompt), to suggest an alternative approach, which worked after human approval	1

Section A.5.6), evaluators independently flagged session-level blockers they saw and whether each required human intervention (see Tables 17 and 18): 11 of 25 human-agent collaborative sessions involved at least one substantive blocker, and 10 of 30 blocker events required human intervention.

4.3 Limitations

Sample size limits generalizability. The uplift study involves 20 papers and 5 participants, which limits the generalizability of our findings. While the estimated positive effect is statistically significant, the small sample size did not permit a serious investigation of potential heterogeneous effects.

Lack of ground truth results. We chose a mix of reproducible and non-reproducible social science papers, verified by I4R. For the machine learning papers, we did not know which were reproducible and which were not. This reflects real-world tasks where researchers may encounter non-reproducible papers, but the lack of ground truth of reproduction for 12 papers prevents us from assessing outcome correctness. We provide results from a post-hoc outcome correctness audit in Section A.5.11.

Reproducers' backgrounds are non-representative. The backgrounds of the reproducers may not reflect the broader population of researchers using agents for computational reproducibility.

The construct misses some benefits of manual reproduction. The results of our randomized study show uplift of human-agent collaboration in completion time and recovery from blockers. However, these miss some benefits of manual code reproduction such as gaining an understanding of the codebase, data, or paper itself that may be important for certain types of reproduction tasks.

Reproducers may be biased. AI uplift study results are often vulnerable to participant biases due to the difficulty of fully blinding participants to AI treatment (Paskov et al., 2026). Since the reproducers are also coauthors of this paper, demand effects could also be possible. We attempt to partially address this by recording detailed terminal logs of manual and human-agent sessions using Docent.

Papers have a skewed distribution. The machine learning papers in the study were drawn from award-winning conference papers, which is not representative of the broader literature. These papers may be better documented or more reproducible on average. In general, our paper selection criteria are not representative of all computational reproducibility tasks (see Section A.5.1 for the full criteria).

5 Conclusion

The dominant *retire-and-replace* paradigm falls short of extracting robust information about agent performance beyond benchmark accuracy. Our premise is that this convention misses underlying dimensions of agent behavior that are crucial for informing deployment decisions. We propose essential steps towards measurement beyond accuracy saturation: investigating benchmark validity, evaluating agents in multiple dimensions (efficiency, reliability, and the relative importance of the model versus the scaffold), and measuring uplift in duration from human-agent collaboration. Our aim is for these contributions to serve as a basis for moving past accuracy-centric evaluation.

AI USE STATEMENT

In this work, we used generative AI tools for copyediting, polishing writing, coding assistance, table and figure formatting, and LLM-assisted log analysis. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide information on our process for updating CORE-Bench in Section 2 and the task, prompt, and grading script changes from the original benchmark in Section A.1. Section A.4 contains further details on running the benchmark, along with agent timeout and retry settings used. The prompt to elicit post-hoc confidence for the reliability analysis is given in Section A.8. For the uplift study, in addition to the methodology in Section 4, we include more details in Section A.5. These include which papers were selected and the paper selection criteria, agent configuration, prompts, instructions evaluators are given, and the questionnaire.

REFERENCES

James Adams, David Bracken, Noam Gidron, Will Horne, Diana Z. O'Brien, and Kaitlin Senk. Can't we all just get along? how women MPs can ameliorate affective polarization in western publics. *American Political Science Review*, 2023. doi: 10.1017/S0003055422000491. URL https://doi.org/10.1017/S0003055422000491.

Mubashara Akhtar, Anka Reuel, Prajna Soni, Sanchit Ahuja, Pawan Sasanka Ammanamanchi, Ruchit Rawal, Vilém Zouhar, Srishti Yadav, Chenxi Whitehouse, Dayeon Ki, Jennifer Mickel, Leshem Choshen, Marek Šuppa, Jan Batzner, Jenny Chim, Jeba Sania, Yanan Long, Hossein A. Rahmani, Christina Knight, Yiyang Nan, Jyoutir Raj, Yu Fan, Shubham Singh, Subramanyam Sahoo, Eliya Habba, Usman Gohar, Siddhesh Pawar, Robert Scholz, Arjun Subramonian, Jingwei Ni, Mykel Kochenderfer, Sanmi Koyejo, Mrinmaya Sachan, Stella Biderman, Zeerak Talat, Avijit Ghosh, and Irene Solaiman. When AI Benchmarks Plateau: A Systematic Study of Benchmark Saturation, February 2026. URL http://arxiv.org/abs/2602.16763. arXiv:2602.16763 [cs].

Anthropic. Claude Code, 2025. URL https://www.claude.com/product/claude-code.

Sabrina B. Arias and Christopher W. Blair. Changing tides: Public attitudes on climate migration. *Journal of Politics*, 2022. doi: 10.1086/715163. URL https://doi.org/10.1086/715163.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 274–283. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/athalye18a.html.

Joel Becker, Nate Rush, Elizabeth Barnes, and David Rein. Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity, July 2025. URL http://arxiv.org/abs/2507.09089. arXiv:2507.09089 [cs].

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024. URL https://arxiv.org/abs/2407.21787.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 5016–5026, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1547. URL https://aclanthology.org/D18-1547/.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating Large Language Models Trained on Code, July 2021. URL http://arxiv.org/abs/2107.03374. arXiv:2107.03374 [cs].

Francois Chollet. ARC-AGI-1, 2019. URL https://arcprize.org/arc-agi/1.

Francois Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. ARC-AGI-2: A New Challenge for Frontier AI Reasoning Systems, January 2026. URL http://arxiv.org/abs/2505.11831. arXiv:2505.11831 [cs.AI].

Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Alijubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Kevin Liu, and Aleksander Madry. Introducing SWE-bench Verified, August 2024. URL https://openai.com/index/introducing-swe-bench-verified/.

Lauren D. Davenport, Annie Franco, and Shanto Iyengar. Multiracial identity and political preferences. *Journal of Politics*, 2022. doi: 10.1086/714760. URL https://doi.org/10.1086/714760.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean Hendryx, Zifan Wang, Vijay Bharadwaj, Jeff Holm, Raja Aluri, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. SWE-Bench Pro: Can AI Agents Solve Long-Horizon Software Engineering Tasks?, 2025. URL https://arxiv.org/abs/2509.16941. Version Number: 2.

Thomas Douenne and Adrien Fabre. Yellow vests, pessimistic beliefs, and carbon tax aversion. *American Economic Journal: Economic Policy*, 2022. doi: 10.1257/pol.20200092. URL https://doi.org/10.1257/pol.20200092.

Endex. AI Built For Excel, 2026. URL https://endex.ai.

Clemens Eppner. AI best papers: Top research papers in AI, ML, CV, and NLP. https://aibestpape.rs/?sub=AI,ML,CV,NLP, 2026.

Julen Etxaniz, Oscar Sainz, Naiara Perez, Itziar Aldabe, German Rigau, Eneko Agirre, Aitor Ormazabal, Mikel Artetxe, and Aitor Soroa. Latxa: An open language model and evaluation suite for Basque. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14952–14972, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.799. URL https://aclanthology.org/2024.acl-long.799/.

Taoran Fang, Zhiqing Xiao, Chunping Wang, Jiarong Xu, Xuan Yang, and Yang Yang. DropMessage: Unifying random dropping for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023. URL https://doi.org/10.1609/aaai.v37i4.25545.

ARC Prize Foundation. ARC-AGI-3: A New Challenge for Frontier Agentic Intelligence, March 2026. URL http://arxiv.org/abs/2603.24621. arXiv:2603.24621 [cs].

Yvette Graham. Improving evaluation of machine translation quality estimation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015. URL https://www.aclweb.org/anthology/P15-1174/.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, et al. Deepseek-r1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z.

Maia Hamin and Benjamin Edelman. Cheating On AI Agent Evaluations, November 2025. URL https://www.nist.gov/caisi/cheating-ai-agent-evaluations. Last Modified: 2025-12-02T12:20-05:00.

Harvey. Building the Business Case for Legal AI | In-House Guide from Harvey, 2022. URL https://www.harvey.ai/.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Stephen Herzog, Jonathon Baron, and Rebecca Davis Gibbons. Antinormative messaging, group cues, and the nuclear ban treaty. *Journal of Politics*, 2022. doi: 10.1086/714924. URL https://doi.org/10.1086/714924.

Shen Zhou Hong, Alex Kleinman, Alyssa Mathiowetz, Adam Howes, Julian Cohen, Suveer Ganta, Alex Letizia, Dora Liao, Deepika Pahari, Xavier Roberts-Gaal, Luca Righetti, and Joe Torres. Measuring mid-2025 llm-assistance on novice performance in biology, 2026. URL https://arxiv.org/abs/2602.16703.

Chuxuan Hu, Liyun Zhang, Yeji Lim, Aum Wadhwani, Austin Peters, and Daniel Kang. REPRO-Bench: Can Agentic AI Systems Assess the Reproducibility of Social Science Research?, July 2025. URL http://arxiv.org/abs/2507.18901. arXiv:2507.18901 [cs.CL].

Institute for Replication. Meta database, version 1. https://i4replication.org/reports/?cpt=metadata, 2024.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI Agents That Matter. *Transactions on Machine Learning Research*, February 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=Zy4uFzMviZ.

Sayash Kapoor, Benedikt Stroebl, Peter Kirgis, Nitya Nadgir, Zachary S Siegel, Boyi Wei, Tianci Xue, Ziru Chen, Felix Chen, Saiteja Utpala, Franck Ndzomga, Dheeraj Oruganty, Sophie Luskin, Kangheng Liu, Botao Yu, Amit Arora, Dongyoon Hahm, Harsh Trivedi, Huan Sun, Juyong Lee, Tengjun Jin, Yifan Mai, Yifei Zhou, Yuxuan Zhu, Rishi Bommasani, Daniel Kang, Dawn Song, Peter Henderson, Yu Su, Percy Liang, and Arvind Narayanan. Holistic agent leaderboard: The missing infrastructure for AI agent evaluation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=vUaY1t64ZZ.

Eunji Kim. Entertaining beliefs in economic mobility. *American Journal of Political Science*, 2023. doi: 10.1111/ajps.12702. URL https://doi.org/10.1111/ajps.12702.

Shanda Li, Qiuhong Anna Wei, Jingwu Tang, Valerie Chen, Nihar B. Shah, Tim Dettmers, Yiming Yang, and Ameet Talwalkar. ReproRepo: Scaling Reproducibility Audits with GitHub Repository Issues, June 2026. URL http://arxiv.org/abs/2606.18237. arXiv:2606.18237 [cs.CL].

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue WANG, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun,

Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Andrew Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=iO4LZibEqW. Featured Certification, Expert Certification, Outstanding Certification.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 21558–21572. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/43e9d647ccd3e4b7b5baab53f0368686-Paper-Conference.pdf.

Gabriel López-Moctezuma, Leonard Wantchekon, Daniel Rubenson, Thomas Fujiwara, and Cecilia Pe Lero. Policy deliberation and voter persuasion: Experimental evidence from an election in the Philippines. *American Journal of Political Science*, 2022. doi: 10.1111/ajps.12566. URL https://doi.org/10.1111/ajps.12566.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of AI research. *Nature*, 651(8107):914–919, March 2026. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-026-10265-5. URL https://www.nature.com/articles/s41586-026-10265-5.

Liang Lu, Peirong Xie, and David Mortensen. Semisupervised neural proto-language reconstruction. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14715–14759, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.788. URL https://aclanthology.org/2024.acl-long.788/.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022. URL https://aclanthology.org/2022.acl-long.556/.

Kevin Meng, Vincent Huang, Jacob Steinhardt, and Sarah Schwettmann. Introducing docent, March 2025. URL https://transluce.org/introducing-docent.

Adriana Molina-Garzón, Tara Grillos, Alan Zarychta, and Krister P. Andersson. Decentralization can increase cooperation among public officials. *American Journal of Political Science*, 2022. doi: 10.1111/ajps.12606. URL https://doi.org/10.1111/ajps.12606.

Elise Paradis, Kate Grey, Quinn Madison, Daye Nam, Andrew Macvean, Vahid Meimand, Nan Zhang, Ben Ferrari-Church, and Satish Chandra. How much does ai impact development speed? an enterprise-based randomized controlled trial. In *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 618–629, 2025. doi: 10.1109/ICSE-SEIP66354.2025.00060.

Neev Parikh and Hjalmar Wijk. MALT: A Dataset of Natural and Prompted Behaviors That Threaten Eval Integrity, October 2025. URL https://metr.org/blog/2025-10-14-malt-dataset-of-natural-and-prompted-behaviors/.

Patricia Paskov, Kevin Wei, Shen Zhou Hong, Dan Bateyko, Xavier Roberts-Gaal, Carson Ezell, Gailius Praninskas, Valerie Chen, Umang Bhatt, and Ella Guest. Rcts & human uplift studies: Methodological challenges and practical solutions for frontier ai evaluation, 2026. URL https://arxiv.org/abs/2603.11001.

James E. Pustejovsky. *clubSandwich: Cluster-Robust (Sandwich) Variance Estimators with Small-Sample Corrections*, 2026. URL https://CRAN.R-project.org/package=clubSandwich. R package version 0.7.0.

James E. Pustejovsky and Elizabeth Tipton. Small-sample methods for cluster-robust variance estimation and hypothesis testing in fixed effects models. *Journal of Business & Economic Statistics*, 36(4):672–683, 2018. doi: 10.1080/07350015.2016.1247004. URL https://doi.org/10.1080/07350015.2016.1247004.

Stephan Rabanser, Sayash Kapoor, Peter Kirgis, Kangheng Liu, Saiteja Utpala, and Arvind Narayanan. Towards a Science of AI Agent Reliability, February 2026. URL http://arxiv.org/abs/2602.16666. arXiv:2602.16666 [cs].

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.442. URL https://aclanthology.org/2020.acl-main.442/.

Zachary S Siegel, Sayash Kapoor, Nitya Nadgir, Benedikt Stroebl, and Arvind Narayanan. CORE-bench: Fostering the credibility of published research through a computational reproducibility agent benchmark. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=BsMMc4MEGS.

Steve Newman. The Black Spatula Project, December 2024. URL https://the-black-spatula-project.github.io/.

David Szakonyi. Indecent disclosures: Anticorruption reforms and political selection. *American Journal of Political Science*, 2023. doi: 10.1111/ajps.12646. URL https://doi.org/10.1111/ajps.12646.

UK AISI. A pipeline for transcript analysis using Inspect Scout, February 2026. URL https://www.aisi.gov.uk/blog/a-pipeline-for-transcript-analysis-using-inspect-scout.

Wandb.ai. Wandb Weave, 2024. URL https://wandb.ai/site/weave.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290. Curran Associates, Inc., 2024. doi: 10.52202/079017-3018. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ad236edc564f3e3156e1b2feafb99a24-Paper-Datasets_and_Benchmarks_Track.pdf.

Zora Zhiruo Wang, John Yang, Kilian Lieret, Alexa Tartaglini, Valerie Chen, Yuxiang Wei, Zijian Wang, Lingming Zhang, Karthik Narasimhan, Ludwig Schmidt, Graham Neubig, Daniel Fried, and Diyi Yang. Position: Humans are Missing from AI Coding Agent Research, 2025. URL https://zorazrw.github.io/files/position-haicode.pdf.

Cai Xu, Jiajun Si, Ziyu Guan, Wei Zhao, Yue Wu, and Xiyue Gao. Reliable conflictive multi-view learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024. URL https://doi.org/10.1609/aaai.v38i14.29546.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. τ-bench: A benchmark for Tool-Agent-User interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

Adam Zelizer. Talking shops: The effects of caucus discussion on policy coalitions. *American Journal of Political Science*, 2021. doi: 10.1111/ajps.12636. URL https://doi.org/10.1111/ajps.12636.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. URL https://doi.org/10.1609/aaai.v35i12.17325.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun, Andy Zhang, Shu Liu, Sasha Cui, Sayash Kapoor, Shayne Longpre, Kevin Meng, Rebecca Weiss, Fazl Barez, Rahul Gupta, Jwala Dhamala, Jacob Merizian, Mario Giulianelli, Harry Coppock, Cozmin Ududec, Jasjeet Sekhon, Jacob Steinhardt, Antony Kellermann, Sarah Schwettmann, Matei Zaharia, Ion Stoica, Percy Liang, and Daniel Kang. Establishing Best Practices for Building Rigorous Agentic Benchmarks. NeurIPS 2025 Datasets and Benchmarks, August 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f316275b44ee2de533102913828a8107-Paper-Datasets_and_Benchmarks_Track.pdf.

A Appendix

A.1 Benchmark update details

We made the following changes to CORE-Bench Hard's grading script when grading agent responses in CORE-Bench v1.1 and CORE-Bench Extended:

1. Expanded CORE-Bench Hard's original 95% prediction interval to accept answers that lie within the default tolerances of `np.isclose` at the upper and lower bounds of the prediction interval.
2. Expanded CORE-Bench Hard's original 95% prediction interval to accept answers where agents reported unrounded results directly from computation when the ground truth was a rounded value.
3. Checked if the ground truth answer was "True" or "False" as a `string`, and if the agent's answer was instead reported as a `boolean`. Converted the agent's answer to a `string` before grading (this only affected task `capsule-2242462`).
4. Accepted multiple answers for the tasks in Table 11.

A.2 CORE-Bench Variants

We provide a description of each CORE-Bench variant (CORE-Bench, CORE-Bench Hard, CORE-Bench v1.1, and CORE-Bench Extended) in Table 7.

A.3 Accuracy saturation of CORE-Bench v1.1 and CORE-Bench Extended

We adopt metrics from Akhtar et al. (2026) that use the standard error of the difference in accuracy between the scores of top and kth agent to determine the similarity of accuracies on CORE-Bench v1.1 and CORE-Bench Extended.

The standard error of the difference between the top and kth agent for n benchmark tasks is:

$$\text{SE}_\Delta \approx \sqrt{\frac{s_1(1-s_1)}{n_{\text{eff}}} + \frac{s_k(1-s_k)}{n_{\text{eff}}}}$$

$$\text{where } n_{\text{eff}} = n^\alpha, \alpha \in [0,1], \text{ default } \alpha = 0.5$$

$$\text{and } s_1 \geq ... \geq s_k \text{ denotes the scores of the top } k \text{ agents.}$$

The top k agents are statistically indistinguishable in accuracy if:

$$s_1 - s_k \leq z \cdot \text{SE}_\Delta$$

Table 7: **CORE-Bench variants.** CORE-Bench v1.1 corrects threats to construct validity in CORE-Bench Hard. CORE-Bench Extended is an out-of-distribution task suite of CORE-Bench v1.1.

CORE-Bench variant	Description
CORE-Bench	Original CORE-Bench variant (Siegel et al., 2024) that evaluates agents on computational reproducibility tasks across three fields (computer science, medical science, and the social sciences) and two languages (Python and R). The test set consists of 45 tasks at each of three difficulty levels: Easy, Medium, and Hard. Each task is selected from a capsule on `codeocean.com` that contains the codebase of a research paper that is verified to be locally reproducible.[a] Each capsule corresponds to a single task, and a task is made up of one or more task questions. While task questions are identical across the three difficulty levels, the agent is provided with less information about solving the task as the difficulty level increases. We refer to Siegel et al. (2024) for the full capsule-selection criteria.
CORE-Bench Hard	Most difficult level of CORE-Bench, where agents must reproduce a paper's code given only the README, the code, and the data (no Dockerfile, runfile, or other instructions).
CORE-Bench v1.1	Updated version of CORE-Bench Hard. Corrects the 15 task-level errors (spanning incorrect ground truths, malformed task questions, grading errors, and unsolvable tasks) and 20 tasks that allow shortcuts in CORE-Bench Hard. Includes 10 new tasks created using the same construction process and task distribution as CORE-Bench Hard, for 39 total tasks.
CORE-Bench Extended	Suite of 19 tasks designed to evaluate agent performance across a field distribution shift from the other CORE-Bench variants, which consist only of tasks from computer science, medical science, and the social sciences. CORE-Bench Extended evaluates generalizability across fields by covering physics, engineering, economics, and computer science tasks.

[a] A capsule is a self-contained, executable research environment that bundles code, data, and software dependencies needed to reproduce a computational experiment.

Table 8: **Updates to CORE-Bench Hard**. For each task, we compared the agent's accuracy to its computation and process correctness. We manually analyzed logs to identify the reason for the discrepancy between the original grade and the process or computation correctness.

Original grade	Process correctness	Computation correctness	Possible explanation of discrepancy	Reason for discrepancy	Update
Correct	Correct	Correct	N/A	Neither	No changes
Correct	Correct	Incorrect	The agent reproduced the paper correctly but ultimately used results from a pre-existing artifact for the answer.	Threat to construct validity	Remove task
Correct	Incorrect	Correct	The agent reproduced only what was necessary to obtain the correct answer or wrote ad-hoc scripts to subvert needing to reproduce the entire paper's code.	Neither	No changes
Correct	Incorrect	Incorrect	The agent was able to guess the answer or used results from a pre-existing artifact for the answer.	Threat to construct validity	Remove task
Incorrect	Correct	Incorrect	The agent's process for reproducing the paper's code was correct, but ultimately made a computation error.	Agent error	No changes
Incorrect	Incorrect	Correct	The agent incorrectly reported the answer.	Agent error	No changes
Incorrect	Correct	Correct	The task prompt, ground truth, or grading contained errors.	Threat to construct validity	Edit the task or grading
Incorrect	Incorrect	Incorrect	N/A	Agent error	No changes
Incorrect	Unsolvable task	Unsolvable Task	The agent must access a dataset, library, or package that is not available.	Neither	Remove task

Using $\alpha = 0.5$ and $z = 1.96$ for a 95% confidence interval, we show that accuracies on both CORE-Bench v1.1 and CORE-Bench Extended for the top $k = 5$ agents are statistically indistinguishable (see Table 12).

Table 9: **Number of tasks affected by threats to construct validity in CORE-Bench Hard.** In total, we found **15** tasks with one or more task-level errors, and 20 tasks (four overlapping with the errors) where the answer can be trivially obtained from a pre-existing artifact. We removed 16 tasks and edited 15 tasks by either removing or editing only the affected task questions, editing the ground truth, or editing the grading script. These threats are difficult to surface prior to saturation, and we provide a few examples of how in Table 10.

Error type	Explanation	Example	Num. tasks affected
Incorrect ground truth	The ground truth answer used for grading was incorrect.	The task requires the agent to report the highest y-axis value. The ground truth answer is the lowest y-axis value.	1
Task question error or underspecification	The task question was unclear or contained an error.	The task requires the agent to report the best accuracy on a test dataset. It's unclear what test dataset the task is referring to.	3
Grading error	The 95% prediction interval didn't capture floating point differences, rounding, or alternate task solutions. The task could have multiple correct answers.	The task requires the agent to report an accuracy. The accuracy value is present in two places in the results: a text output file where the value is not rounded, and a figure where the value is rounded. The agent reports the value from the text output file, but the ground truth answer is from the figure.	7
Unsolvable task	The task relies on data, packages, or libraries that are not available. The results are non-deterministic.	The task requires the agent to download a dataset from a URL that is no longer live.	4

Table 10: **Examples of task-level threats to benchmark validity in CORE-Bench Hard and our initial version of CORE-Bench Extended.** These threats are difficult to surface with less capable agents. Prior to accuracy saturation, one of the most common failure cases on CORE-Bench Hard was agents unable to resolve version dependency conflicts (Siegel et al., 2024). These agents were not progressing far enough into the reproduction pipeline to take shortcuts, report correct answers that were inaccurately marked as incorrect, or encounter environmental barriers. This made anticipating all task-level threats intractable before accuracy saturated.

Capsule ID	Task question	Error
`capsule-9670283`	*From the final result plot, report the label for the blue line.*	The agent is able to guess the label from the color of the plot line using `matplotlib`'s default color order.
`capsule-3262218`	*Report the number of methods counter-arguments provided to defend the original study in light of the contradictory replication results.*	The agent could obtain the correct answer by running a trivial, ad-hoc command to count CSV rows where `methodsCounter == TRUE` without reproducing the paper's results.
`capsule-4299879`	*From the figure measuring bootstrapped predictive distribution of endline trust in police assuming mean regression at rate of mean regression among unexposed citizens, report the p-value from the Heard of Meetings plot.*	If the agent re-runs the bootstrap calculation in isolation without running the full end-to-end reproduction pipeline, the code will produce non-deterministic random samples because the seed is set at the beginning of the script.
`capsule-5801588`	*Report the label of the line from the plot measuring model evaluations at each iteration with the highest Model Evaluations at iteration 10.0.*	During benchmark construction, three code runs yielded the same task answer. However, when multiple agents were marked incorrect for this task with no apparent trajectory-level errors, we ran the script twice more and found the answer to be non-deterministic.
`capsule-2675546`	*From the ROC curve of UE #74, report the true positive rate when the false positive rate is 0.4.*	The answer to this task question differs when the agent runs the script with Python 3.12 and newer libraries, versus the original paper's runs that use Python 3.6.

A.4 BENCHMARK IMPLEMENTATION

We run all agents on Azure virtual machines. The tasks requiring GPU are run on `Standard_NC4as_T4_v3` and the remainder of the tasks are run on `Standard_D4s_v3`. All runs use the HAL evaluation harness (Kapoor et al., 2026). HAL provides a standard harness for reproducible agent evaluation and uses Weave for automated logging (Wandb.ai, 2024). All agents have full file system access and full web access.

For all Codex CLI, Claude Code, and OpenCode agents, we set per-task timeout to 45 minutes, max retries to 3 at the harness level, and max attempts to 3 at the scaffold level. For CORE-Agent, we set

Table 11: We updated the grading script for six capsules and seven task questions to accept multiple answers. `capsule-2151475` is in CORE-Bench Extended. The rest are in CORE-Bench v1.1.

Capsule ID	Task question	Reason for accepting multiple answers
`capsule-2816027`	*For CTCF Signature Enrichment, report the name of the group with the highest median GSVA score.*	The group name in the capsule's figure label and the actual group name differ. We accept both.
`capsule-3639589`	*Report the color of the line with the highest maximum activation for target memory activation, DM.*	There are two plots in the results that show maximum activation for DM with different plot colors. We accept both.
`capsule-2151475`	*Report the name of the university ranked #1 by impact factor.*	The ground truth is the abbreviation of the university name. We accept both the full university name and the abbreviation as it appears in the result figure.
	Report the name of the journal with the highest 2011 impact factor from the analysis of 30 journals.	The ground truth is the abbreviation of the journal name. We accept both the full journal name and the abbreviation as it appears in the result figure.
`capsule-0152700`	*Given the Kruskal-Wallis for Group 0-2 (Group 1 vs. Group 3), what is the p-value?*	The capsule results contain three deterministic p-values. We accept all three.
`capsule-9477017`	*Pearson correlation coefficients between the estimated proportions of different cell types were calculated, what is the highest Pearson correlation related to? Give the response in a list of strings.*	There are two possible highest correlation coefficients from the result plots. We accept cell types related to both, order-agnostic.
`capsule-4252248`	*Report the overall AUC from the PR curve generated with the CTRPv2 sensitivity dataset, tested against ATC annotations and drug-target information from CHEMBL.*	The AUC in the plot title is not rounded, but the AUC in the plot legend is rounded to three decimal places. We accept both.

Table 12: **Saturation metrics.** We use the operationalization of saturation from Akhtar et al. (2026) to show that the top-five agents on both CORE-Bench v1.1 and CORE-Bench Extended have statistically indistinguishable accuracies.

Benchmark	s_1	s_5	$\Delta = s_1 - s_5$	$z \cdot \mathrm{SE}_\Delta$	$\Delta \leq z \cdot \mathrm{SE}_\Delta$
CORE-Bench v1.1	1	0.9744	0.0256	0.1240	True
CORE-Bench Extended	1	0.8947	0.1053	0.2881	True

per-task timeout to 5 hours, max steps to 200, and max retries to 3 at the harness level. HuggingFace smolagents does not have directly comparable retry logic at the scaffold level.

In particular, for Codex CLI, Claude Code, and OpenCode, if time permits, the scaffold allows the model to produce output for up to 3 attempts for erroneous answers (e.g. if no `answer.json` is produced or wrongly formatted), and for all scaffolds, the HAL evaluation harness allows the agent to be rerun for up to 3 retries if it encounters a transient error such as a timeout or connection error.

We also provide a comparison of the interface and additional tools of the scaffolds used in Table 13.

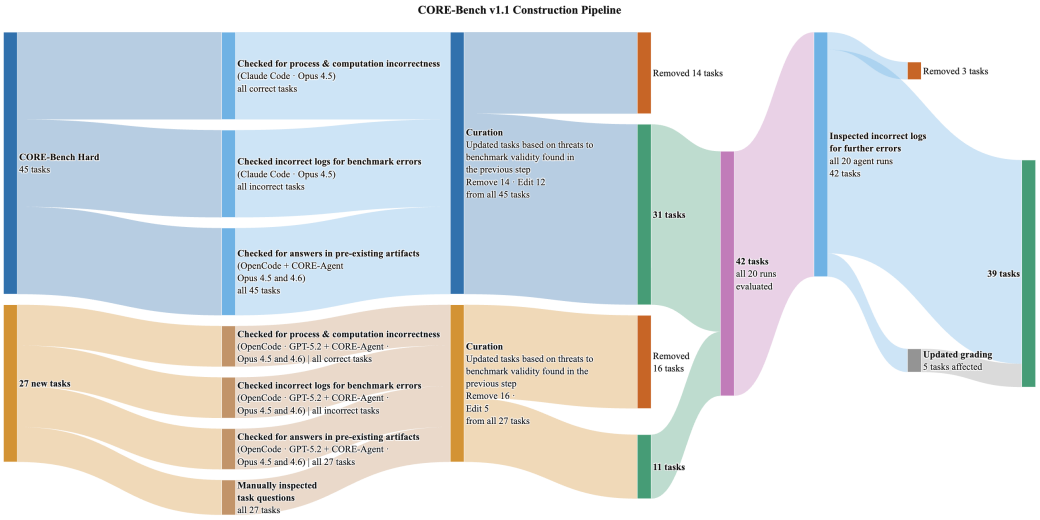
A.4.1 Differences in results from Codex CLI versions

We found that accuracy on CORE-Bench v1.1 with GPT-5.1 differed significantly based on Codex CLI version, with Codex CLI v0.122 obtaining an accuracy about 40% higher than Codex CLI v0.130.0. Despite both versions using GPT-5.1, Codex CLI v0.130.0 had much shorter trajectories than Codex CLI v0.122: about two-thirds the total commands and one-fourth the output tokens.

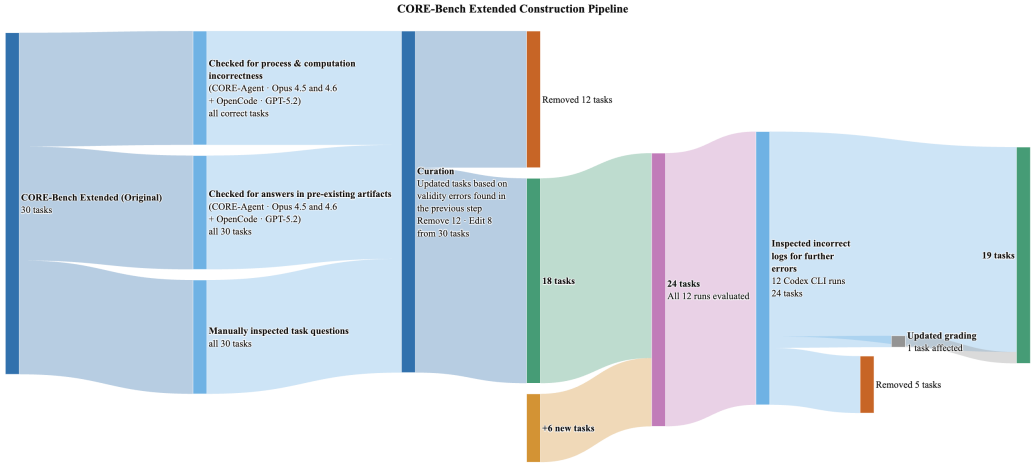
In Section 2 and Section 3.2, we report results using Codex CLI v0.122 for all Codex CLI runs. In Section 3.1, we report results using Codex CLI v0.130.0 for all models except GPT-5.1, where we use Codex CLI v0.122.

A.4.2 Benchmark task breakdowns

We provide a task breakdown of CORE-Bench v1.1 compared to CORE-Bench Hard in Table 14.



(a) **CORE-Bench v1.1 construction pipeline.** We used automated and manual log analysis to identify threats to construct validity affecting the 45 CORE-Bench Hard tasks and 27 newly added tasks that informed updates and grading changes. These threats were difficult to surface with less capable agents that weren't progressing far enough past initial task solution steps to encounter errors or exploit shortcuts. The resulting benchmark, CORE-Bench v1.1, consists of 39 tasks that reflect validity improvements compared to the original dataset. We provide a summary of our rubrics (Table 2) and other details on benchmark construction in Section A.1



(b) **CORE-Bench Extended construction pipeline.** We used a similar method of automated and manual log analysis as Figure 4a to identify threats to benchmark validity affecting our original CORE-Bench Extended test set. The resulting benchmark, CORE-Bench Extended, has 19 tasks.

Figure 4: Construction pipelines for CORE-Bench v1.1 and CORE-Bench Extended.

A.4.3 LIMITATIONS

Unlike Codex CLI, Claude Code and OpenCode, we did not use any scaffold-level retries for CORE-Agent because Hugging Face smolagents does not expose a directly comparable retry mechanism. In contrast, we evaluated the other scaffolds with three retries at both the harness and scaffold levels. Retry behavior was not fully standardized across scaffolds, which may affect comparisons (see Section A.4 for implementation details). Additionally, our analyses are limited to the specific models, scaffolds, reasoning-effort settings, and runtime budgets evaluated. Finally, these dimensions beyond accuracy are not exhaustive; we focus on dimensions that directly inform deployment decisions, while recognizing that saturated benchmarks can support many other important analyses beyond accuracy.

Table 13: Comparison of the interface and additional tools of scaffolds used.

Scaffold	Execution interface	Additional tools	Interface design
Codex CLI	`exec_command`	N/A	A single terminal interface supports all agent actions.
Claude Code	`Bash`	`Read`, `Write`, `Edit`, `Glob`, `Grep`, `Monitor`, `ToolSearch`, `Agent`	Dedicated tools support file and image reading, file modification, search, process monitoring, tool discovery, and subagent delegation.
OpenCode	`Bash`	`read`, `write`, `edit`, `apply_patch`, `glob`, `grep`, `webfetch`, `todowrite`, `task`	Dedicated tools support file reading, writing, editing, and search; URL retrieval; task planning; and subagent delegation.
CORE-Agent	`python_interpreter`	`execute_bash`, `web_search`, `visit_webpage`, `inspect_file_as_text`, `edit_file`, `file_content_search`, `query_vision_language_model`, `final_answer`	Dedicated tools support Python and shell execution, web retrieval, document inspection, file editing and search, vision-language queries, and final-answer submission.

Table 14: Number of tasks by field and language in the test set of CORE-Bench Hard and CORE-Bench v1.1.

	Computer Science	Social Science	Medical Science	Total
CORE-Bench Hard	18	14	13	45
CORE-Bench v1.1	13	10	16	39

(a) Task comparison by field

	Python	R	Total
CORE-Bench Hard	22	23	45
CORE-Bench v1.1	18	21	39

(b) Task comparison by language

A.5 Randomized study details

We provide additional details on methodology and implementation of the uplift study.

A.5.1 Paper Selection Criteria

A paper was included only if all of the following criteria were met:

1. **From sources:**
   a. For ML papers: Won a paper award at one of these conferences, AAAI, ACL, CVPR, ECCV, EMNLP, ICCV, ICLR, ICML, IJCAI-JAIR, NeurIPS, and 3DV from 2011–2025 (sourced from `https://github.com/clemense/ai-bestpapers`)
   b. For non-ML papers: evaluated in the I4R reproducibility study (and found to be "evaluable" there, e.g. data available)
2. **GitHub (or other) repository exists with code**
3. **Can run on single GPU or CPU** (no hosted models)
4. **More specifically:** Reproduction of targets we selected from the paper (see below) looks likely to run in our setup (A40 48GB VRAM, disk space: 40GB+40GB - see evaluator instructions for details)
5. **Data available (link). (where applicable)**
   a. "Available" meaning for direct download without registration or such
   b. For ML papers, this might include pre-existing benchmarks (e.g. for `https://arxiv.org/pdf/2312.12337` this could be the RealEstate10k dataset from an earlier paper)
6. **Pretrained weights available (link)** (where applicable). Notes:
   a. "Available" meaning for direct download without registration or such
7. **Uses Python or R**
8. **Clear success criteria** (specific tables/figures)
9. **Not previously seen** by the evaluator (defined as having read at most the abstract)
10. **Compute time limit**: running the code / inference necessary for the reproduction is anticipated to take less than 45 minutes on our hardware. Notes:

a. "Compute time" refers to the cumulative duration of the agent and/or human evaluator having to wait for VM to complete compute tasks.
b. This represents the compute reproduction time for all replication targets together.
c. Does not include Run 2 & Run 3 for non-deterministic outcomes if floating point tolerance criterion (see evaluator instructions) is not met.
d. Does not include wait times for data or model downloads.
e. Does not include the time the agent spends reasoning or using other tools.
f. Estimates are OK (e.g. concluding that this criterion is not met after a progress bar shows 10% completed after 10 minutes).
g. Does not include the environment set up and dependencies

A.5.2 Uplift study implementation

For the uplift study, both human-only (manual) and human-agent reproduction attempts are run inside standardized Docker environments to ensure consistency across participants and conditions. ML papers (from AI conferences) are run on cloud GPU instances using A40 GPUs on RunPod with a dedicated ML Docker image, while non-ML papers (from the I4R source) are run using a separate non-ML Docker image; both templates are configured with 40 GB container disk (plus 40 GB volume for the ML template). Using a uniform GPU and VM configuration ensures that runs are comparable in compute resources.

In the human–agent (AI-allowed) condition, participants use Codex CLI with the GPT-5.4 model at the "extra high" reasoning setting. Codex is launched inside a Docker container which automates session logging and uploads traces to Docent for later inspection. Each participant provides the paper PDF, the default reproduction prompt (Appendix A.5.3), and the replication target.

In the manual (AI-disallowed) condition, participants may use only traditional web resources such as documentation, forums and StackOverflow; no generative AI tools (e.g., ChatGPT, GitHub Copilot, Claude) and no AI-generated search summaries (e.g., Google AI Overviews, Bing Copilot) are permitted. To suppress AI Overviews, participants append the `-ai` flag to every Google query. Non-generative IDE autocomplete is allowed.

To preserve independence across attempts, a fresh pod is launched for each reproduction, or the current pod is fully reset before reuse. The maximum time limit for a single reproduction attempt is 3 hours, after which the attempt is recorded as unsuccessful. Following each run, participants complete a structured post-run questionnaire (Appendix A.5.9) capturing their experience, blockers encountered and self-reported confidence in the result.

A.5.3 Default Prompt

The following prompt is provided to participants (and, in the agent condition, to the agent) at the start of each reproduction attempt. The placeholders `[PAPER_NAME]`, `[REPO_URL]`, "`replication target`", and `{replication target}` are filled in per task.

> Below is a result ("replication target") selected from the research paper present in this directory, titled "[PAPER_NAME]". Reproduce this replication target exclusively by running the paper's code. All we care about is getting there through genuine reproduction.
>
> Obtain the code here: [REPO_URL]
>
> Read the README (if present). Set up the environment, install dependencies, download any required data, then run the code to reproduce the following result ("replication target") reported in the paper:
>
> {replication target}
>
> Rules:

- Do not modify any script's scientific logic. Limit changes to environment compatibility only (e.g. dependency versions, paths, deprecated APIs, config variables, runtime arguments/variables such as model type or dataset).

- If stuck after 2-3 attempts on the same error, stop and tell me what's wrong so we can figure it out together.

- Save all generated outputs and report back the results. For numeric values, report the exact value of the output - do not round or truncate

- If the reproduction value of the replication target is not within the floating point tolerance of `1e5 * sys.float_info.epsilon` of the paper's reported value after rounding the reproduction value to the same number of decimal places, then run 2 more times and determine if the reproduction value falls within the 95% prediction interval using the `_compute_prediction_intervals` function below.

```
def _compute_prediction_intervals(
    reproduction_values: list[dict],
    numeric_keys: list[str]
) -> dict[str, dict]:
    """Compute 95% prediction intervals for numeric
    values."""
    intervals = {}
    sample_size = len(reproduction_values)
    if sample_size < 2:
        for key in numeric_keys:
            value = reproduction_values[0].get(key, 0)
            intervals[key] = {"lower": value, "upper":
    value, "mean": value}
        return intervals
    t_value = t.ppf(0.975, sample_size - 1)
    for key in numeric_keys:
        values = [rv.get(key, 0) for rv in
    reproduction_values]
        mean = np.mean(values)
        std = np.std(values, ddof=1)
        margin = t_value * std * math.sqrt(1 + 1 /
    sample_size)
        intervals[key] = {
            "lower": mean - margin,
            "upper": mean + margin,
            "mean": mean,
        }
```

A.5.4 PAPERS SELECTED FOR REPRODUCTION

Table 15: **Papers selected for reproduction, with field and reproduction target.** Targets were chosen by selectors by picking a specified value from the published paper. The final column records the observed outcome from our study: "Matched" indicates that at least one reproduction attempt achieved the target metric within the specified tolerance; "Result, no match" indicates that at least one attempt produced a result but none matched within tolerance; and "No results produced" indicates that no attempt produced a usable result. All social science papers are confirmed to be reproducible unless indicated otherwise.
† indicates the paper is not reproducible but we selected a target not validated by I4R.
‡ indicates the paper is not reproducible and we also selected the non-reproducible target.

Paper	Field	Reproduction Target	Paper Code	Observed outcome
Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examplesAthalye et al. (2018)	Machine Learning	Accuracy under the defense from Buckman et al. (2018) on CIFAR (Table 1): 0%	Code	No results produced
Latxa: An open language model and evaluation suite for basqueEtxaniz et al. (2024)	Machine Learning	Performance of Latxa 7B on EusProf (Table 1): 30.26	Code	Result, no match
Beyond accuracy: Behavioral testing of NLP models with CheckListRibeiro et al. (2020)	Machine Learning	Failure rate of BERT-base on Sentiment Analysis "Negated neutral should still be neutral" MFT (Table 1): 98.4%	Code	Matched
Semisupervised neural protolanguage reconstructionLu et al. (2024)	Machine Learning	TED of Transformer DPD-ΠM-BST on 10% labeled WikiHan, averaged across all runs in four groups (Table 2): 1.0075	Code	Matched
Improving evaluation of machine translation quality estimationGraham (2015)	Machine Learning	Williams test outcome for HTER prediction in EN→ES WMT-14 Task 1.2: significant increase in Pearson correlation for HTER-DCU-rtm-svr over HTER-DCU-rtm-tree ($p < 0.05$)	Code	Matched
Reliable conflictive multi view learningXu et al. (2024)	Machine Learning	Conflictive test-set accuracy of ECML on Scene15 (Table 3): $56.97 \pm 0.52\%$	Code	Matched
Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivityLu et al. (2022)	Machine Learning	Performance of GPT-2 0.1B GlobalE on Template 1 (Table 3): 63.8	Code	Matched
Informer: Beyond efficient transformer for long sequence time-series forecastingZhou et al. (2021)	Machine Learning	MSE of Informer on ETTh1 with 24 counts (Table 2): 0.577	Code	Matched
DropMessage: Unifying random dropping for graph neural networksFang et al. (2023)	Machine Learning	Accuracy of GCN-DropMessage on PubMed (Table 2): 79.20	Code	Matched
MultiWOZ — a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modellingBudzianowski et al. (2018)	Machine Learning	Number of dialogues in the MultiWOZ training split (Table 1): 8,438	Code	Matched
Talking shops: The effects of caucus discussion on policy coalitionsZelizer (2021)	Social Science	Deliberation effect on cosponsorship for attended meetings, non-sponsor's party (Table 4): 5.9 pp	Code	Matched
Decentralization can increase cooperation among public officialsMolina-Garzón et al. (2022)	Social Science	Coefficient for Decentralized in Weighted Poisson Full Model for Strong Ties (Table 3): 1.07	Code	Matched

Continued on next page

Paper	Field	Reproduction Target	Paper Code	Observed outcome
Changing tides: Public attitudes on climate migrationArias & Blair (2022)	Social Science	AMCE for flooding vs. economic opportunity as migration reason, German sample (Table 2): 0.086	Code	Matched
Multiracial identity and political preferencesDavenport et al. (2022) ‡	Social Science	Whether White-Blacks are more conservative or more liberal than Blacks on police perceptions (Figure 1): more conservative	Code	Matched
Entertaining beliefs in economic mobilityKim (2023)	Social Science	Coefficient of Rags-to-Riches TV Treatment on belief in economic mobility, lab-in-the-field sample (Table 1, Col. 5): 0.068	Code	Matched
Antinormative messaging, group cues, and the nuclear ban treatyHerzog et al. (2022)	Social Science	Treatment effect of Institution Cue on support for TPNW (Appendix Table H1, Model 2): -19.2 pp	Code	Matched
Policy deliberation and voter persuasion: Experimental evidence from an election in the PhilippinesLópez-Moctezuma et al. (2022)	Social Science	ITT of Vote (Akbayan) (Table 1): 1.955	Code	Matched
Can't we all just get along? how women MPs can ameliorate affective polarization in western publicsAdams et al. (2023)	Social Science	Coefficient for out-party proportion of women MPs ($t-1$) among women partisans (Table 1): 2.1	Code	Matched
Indecent disclosures: Anticorruption reforms and political selectionSzakonyi (2023)	Social Science	Treatment group $\times$ Second period election coefficient (Table 1): -0.057 (0.015)	Code	Matched
Yellow vests, pessimistic beliefs, and carbon tax aversionDouenne & Fabre (2022) †	Social Science	OLS coefficient for Yellow Vests: supports (Table 2): -0.108 (0.026)	Code	Matched

A.5.5 Instructions for evaluators (for manual and agent-based runs)

- We're using Codex (with OpenAI credits)
- Reproduction should be run in one of two Docker images (for ML/non-ML papers): (when using Runpod, this is already integrated into the template, see below)
- Starting codex in the docker image (choose `gpt-5.4` with extra high thinking)
- Reproductions of ML papers should be run on a cloud GPU environment like Lambda (which we already have credits for) or RunPod (AWS and Google Cloud also work). Currently we have planned to use **A40s on RunPod**. Using the same kind of GPU and VM ensures that runs are comparable in that respect.
- Runpod
  - Create a Runpod account and configure SSH
  - Use ML template/Docker image for the AI conference papers `https://console.runpod.io/deploy?type=GPU&gpu=A40&count=1&template=jo8klw71d0`. Under "Select an instance", pick the A40 GPU option. The template sets disk space at 40GB Container + 40GB Volume.
  - Use Non-ML template/Docker image for the I4R papers `https://console.runpod.io/deploy?type=GPU&gpu=A40&count=1&template=071bthitdg`. Under "Select an instance", pick the A40 GPU option.
    The template sets disk space at 40GB Container.
  - Keep the pod running to prevent data loss through the course of reproduction (Codex logs, etc. not mounted in runpod `/workspace` directory)
  - If you have to download a file from your Runpod instance for inspection etc.:
    Step 1: Install `runpodctl` (on your local machine)

```
mkdir -p ~/.local/bin && \
curl -sL https://github.com/runpod/runpodctl/\
releases/latest/download/\
runpodctl-linux-amd64.tar.gz \
| tar xz -C ~/.local/bin
export PATH="$HOME/.local/bin:$PATH"
runpodctl version
```

Step 2: On your runpod instance run

```
runpodctl send ~/test_image.png
# outputs something like:
# Code is: 3476-quiet-telex-premium-9
# On the other computer run

# runpodctl receive 3476-quiet-telex-premium-9
```

Step 3: On your local computer run

```
runpodctl receive 3476-quiet-telex-premium-9
```

- The Docker image will launch an internal script that automates logging and will ask for your Docent API key to upload traces/logs to Docent, including run metadata, once the user enters `finish-session`. Details in `README.md`.
  - Will be using the final Docent collection for final runs. For the pilot we will be using pilot.
- Upload a PDF of the paper (for Runpod, run the `runpodctl` tool on your local machine: `runpodctl send paper.pdf`)
- Launch Codex in the terminal, per the hint provided by the script.
- Use `/model` to switch to `gpt-5.4` with extra high thinking
- Keep the pod (VM) running during the reproduction attempt
- After the reproduction attempt, use the `finish-session` command in the terminal to log the (raw) duration and upload the session log to Docent (optional when doing testing/pilot runs). Be sure to exit from your virtual environments before running the command.
- After the run, fill out the questionnaire in Google Forms [see Section A.5.9]
- Launch a new instance for each reproduction, OR completely reset the current one (make sure it will be returned to the same state as after a new deployment, so as not to affect consistency)
  - On Runpod, resetting can be achieved by
    - wiping the workspace (i.e. delete all files and folders, using a command like `find /workspace -mindepth 1 -maxdepth 1 -exec rm -rf - {} +`), and then
    - using "edit a pod" (`https://docs.runpod.io/pods/manage-pods#update-a-pod` and saving the configuration again with no changes
- Maximum time limit for a single reproduction attempt (referring to "duration" as defined below): 3 hours

Manual Condition (AI-Disallowed)

- No generative AI tools used (e.g., GitHub Copilot, ChatGPT)
- No AI-generated search summaries used (e.g., Google AI Overviews, Bing Copilot)
- Only traditional web search (links, docs, StackOverflow) used
  - In Google searches, append the "-ai" flag to every search to suppress automatic AI-generated results
- No AI-based code generation or debugging assistance
- All reproduction tasks must be executed within a standardized Docker environment.

Allowed: non-generative IDE autocomplete, documentation, forums

A.5.6 BLOCKERS REVIEW RUBRIC

We used the following rubric to select blockers from our logs of the human-agent collaboration runs, with the assistance of Codex using GPT-5.4. We define operational blockers as any concrete obstacle that delayed progress, forced a workaround or required debugging or rerouting. Agents encountered operational blockers 122 times across 25 runs (4.88 per run), identified using Codex-assisted log analysis. 74 arose during setup, 40 during execution and only 8 during result extraction or reporting. In practice, we saw that the agent's contribution was usually to repair the local reproduction path rather than simply launch a clean released pipeline.

```
# Blocker Review Rubric

Each AI run should be reviewed independently from its exported transcript JSON.

Goal: extract a comprehensive but disciplined list of blockers the AI agent faced.

Definition of blocker:
- Any concrete obstacle that delayed progress, forced a workaround, caused a
  failed attempt, or required debugging/rerouting.
- Include both root-cause blockers and shorter-lived operational blockers if
  they materially interrupted the run.
- Do not include ordinary progress steps that were not obstacles.
- Do not include purely hypothetical risks unless they became an actual
  impediment in the transcript.

Granularity rule:
- Split distinct obstacles into separate blocker entries when they required
  different fixes or occurred in different phases.
- If multiple symptoms clearly stem from one issue, keep them in one blocker
  entry and describe the symptoms in the evidence.

Required output fields per run:
- collection_name
- actual_collection_name
- researcher
- paper
- agent_run_id
- agent_run_name
- model
- overall_outcome
- notes
- blockers

Required fields per blocker:
- label
- category
- phase
- resolved
- description
- evidence

Allowed category values:
- environment
- dependency
- repo_artifact
- path_config
- data_input
- runtime
- tooling
```

```
Allowed phase values:
- setup
- execution
- postprocess

Allowed resolved values:
- yes
- no
- partial

Evidence rule:
- Cite concrete transcript evidence in plain text, ideally with message indices
  or an explicit quoted/paraphrased action/result.
- Keep evidence concise.
```

A.5.7 RANDOMIZED STUDY DESIGN

The randomized study aims to estimate the uplift effect in duration of human-agent collaboration on the task of computationally reproducing a result (replication) target from a given paper. It was designed assuming that both the papers (replication targets) and the researchers (evaluators) carrying out the reproduction task may have unobserved characteristics that affect task duration. This motivates the use of a blocked randomization assignment with blocking on both researchers and papers, where the sampling of paper-evaluator pairs (among all possible combinations) and their random assignment to either the treatment (human-agent collaboration) or control (Manual) condition is restricted by the following balancing requirements:

- Each of the 20 papers was assigned to either 2 or 3 of the 5 evaluators, and to each condition (manual or human-agent collaborative) at least once.
- Each of the evaluators was assigned 10 papers (5 from each source), and to each condition (manual or human-agent collaborative) 5 times.

To mitigate learning effects, evaluators were instructed to carry out the tasks in a specified randomized order.

The same five authors who acted as evaluators also carried out the selection of papers from the aforementioned two sources. This task included vetting of the predefined selection criteria (such as the availability of code and data for the paper, or that the reproduction should be feasible on the hardware used in the experiment), and selection of one specific result to replicate from each paper. This process was designed to ensure blinding (i.e. as an additional constraint on the randomized assignment, no team member was assigned a paper as evaluator that they had encountered during the paper selection process). To achieve a degree of representativeness with respect to the given source and criteria, selectors were assigned a randomly selected and randomly ordered slice from each dataset, to assess for eligibility in the given order.

A.5.8 FIXED EFFECTS MODEL TO ESTIMATE UPLIFT IN TASK DURATION

To estimate the uplift in duration, we use linear regression with log task duration as the outcome variable (see Section A.5.9 for its exact definition as part of the evaluator questionnaire). Our aforementioned assumption (that task duration might be affected by unobserved characteristics of both papers and evaluators) also motivates the use of a fixed effects model here, with fixed effects for both researchers and papers:

$$\log(\text{duration}_i) = \alpha + \beta\,\text{AI}_i + \gamma_p + \delta_r + \varepsilon_i, \tag{1}$$

Here i indexes the individual replication session (identified with a paper-evaluator pair), p denotes the paper being reproduced in session i, and r the evaluator conducting the session. The terms γ_p and δ_r are paper and evaluator fixed effects, respectively, and β is the estimated difference between log task

duration in the Manual condition relative to the Human-agent collaborative condition. We conceive uplift as a speed change here, and speed is reciprocal to duration; so, for simplicity we estimate the reciprocal factor for duration - from Human-agent collaborative to Manual instead of vice versa.

We use CR2 standard errors clustered by researcher. CR2 standard errors are designed for use with small-sample fixed effect models, and represent a conservative choice relative to conventional clustered standard errors.(Pustejovsky & Tipton, 2018) They are the recommended small-sample correction in R's clubSandwich package.(Pustejovsky, 2026)

The model's coefficient estimate for the Manual condition is 0.7485, with a CR2 variance of 0.00844, a CR2 standard error of 0.0919, Satterthwaite degrees of freedom of 3.7, and p-value 0.00176. The 95% CR2 confidence interval for this effect on the log-duration scale is [0.4842, 1.0127]. This implies that Manual sessions last about 2.11 times as long as human-agent collaborative sessions, with a 95% CR2 confidence interval of [1.62, 2.75] for this duration ratio.

In the experiment, session duration was capped at 180 minutes, i.e. the outcome variable is right-censored. We did not attempt to account for this in the model. Because only manual runs hit this limit in our experiment, our uplift estimate is conservative in that regard.

A.5.9 QUESTIONNAIRE FOR HUMAN-AGENT REPRODUCIBILITY EVALUATION

```
============================================================
Part I: Metadata and Environment Setup (Questions 1--48)
============================================================

1  Paper Title
2  Link to the paper's code (GitHub or other)
3  Domain: {AI Conferences, I4R}
4  Email
5  Human Researcher [your name]

6  Hardware (Select "Other" only if you used an environment
   other than A40 on RunPod.):
     {GPU: A40 48GB VRAM CPU: Intel(R) Xeon(R) Gold 6342 CPU
      @ 2.80GH, Other}

7  OS (Select "Other" only if you used an environment other
   than A40 on RunPod.):
     {Ubuntu 22.04, Other}

8  Execution Environment (Select "Other" only if you performed
   a non-standardized step.):
     {Docker instance with pre-installed libraries, Other}

9  Date (PST) {mm, dd, yyyy}
10 Start Time (PST)
11 End Time (PST)
12 Link to Docent log of this session
13 Condition {Manual, AI-assisted}
14 Agent version: {gpt-5.4-codex with extra high thinking,
   Other}

15 Step 1.1: Start the instance and Docker image (Human)
   -- Outcome {Success, Failure, Partial Success}

16 Step 1.1: Start the instance and Docker image (Human)
   -- Notes

17 Step 1.2: Start Agent with logging and prompt replication
   target task (using the generic default prompt) (Human)
```

```
   -- Outcome
     {Success, Failure, Partial Success}

18 Step 1.2: Start Agent with logging and prompt replication
   target task (using the generic default prompt) (Human)
   -- Notes

19 Step 1.3: Obtain the paper's code (e.g. clone repo)
   -- Who did it {Human, Agent, Both}

20 Step 1.3: Obtain the paper's code (e.g. clone repo)
   -- Outcome {Success, Failure, Partial Success}

21 Step 1.3: Obtain the paper's code (e.g. clone repo)
   -- Notes

22 Step 1.4: Read README
   -- Who did it {Human, Agent, Both}

23 Step 1.4: Read README
   -- Outcome {Success, Failure, Partial Success}

24 Step 1.4: Read README -- Notes

25 Step 1.5: Create environment (e.g. using conda/venv)
   -- Who did it {Human, Agent, Both}

26 Step 1.5: Create environment (e.g. using conda/venv)
   -- Outcome {Success, Failure, Partial Success}

27 Step 1.5: Create environment (e.g. using conda/venv)
   -- Notes

28 Step 1.6: Install dependencies
   -- Who did it {Human, Agent, Both}

29 Step 1.6: Install dependencies
   -- Outcome {Success, Failure, Partial Success}

30 Step 1.6: Install dependencies -- Notes

31 Step 1.7: Download/prepare data
   -- Who did it {Human, Agent, Both}

32 Step 1.7: Download/prepare data
   -- Outcome {Success, Failure, Partial Success}

33 Step 1.7: Download/prepare data -- Notes

34 Step 1.8: Verify setup (import test, etc.)
   -- Who did it {Human, Agent, Both}

35 Step 1.8: Verify setup (import test, etc.)
   -- Outcome {Success, Failure, Partial Success}

36 Step 1.8: Verify setup (import test, etc.) -- Notes

37 Phase 1 Blocker 1: What was the blocker
38 Phase 1 Blocker 1: Who got stuck {Human, Agent, Both}
```

```
39 Phase 1 Blocker 1: What was the resolution
40 Phase 1 Blocker 1: Intervention needed? {Yes, No}

41 Phase 1 Blocker 2: What was the blocker
42 Phase 1 Blocker 2: Who got stuck {Human, Agent, Both}
43 Phase 1 Blocker 2: What was the resolution
44 Phase 1 Blocker 2: Intervention needed? {Yes, No}

45 Phase 1 Blocker 3: What was the blocker
46 Phase 1 Blocker 3: Who got stuck {Human, Agent, Both}
47 Phase 1 Blocker 3: What was the resolution
48 Phase 1 Blocker 3: Intervention needed? {Yes, No}

============================================================
Part II: Reproduction Execution and Runtime Debugging
(Questions 49--75)
============================================================

49 Phase 1 Step 2.1: Identify entry point / main script
   -- Who did it {Human, Agent, Both}

50 Phase 1 Step 2.1: Identify entry point / main script
   -- Outcome {Success, Failure, Partial Success}

51 Phase 1 Step 2.1: Identify entry point / main script
   -- Notes

52 Phase 1 Step 2.2: Understand required run parameters
   -- Who did it {Human, Agent, Both}

53 Phase 1 Step 2.2: Understand required run parameters
   -- Outcome {Success, Failure, Partial Success}

54 Phase 1 Step 2.2: Understand required run parameters
   -- Notes

55 Phase 1 Step 2.3: Run code
   -- Who did it {Human, Agent, Both}

56 Phase 1 Step 2.3: Run code
   -- Outcome {Success, Failure, Partial Success}

57 Phase 1 Step 2.3: Run code -- Notes

58 Phase 1 Step 2.4: Monitor / debug runtime errors
   -- Who did it {Human, Agent, Both}

59 Phase 1 Step 2.4: Monitor / debug runtime errors
   -- Outcome {Success, Failure, Partial Success}

60 Phase 1 Step 2.4: Monitor / debug runtime errors
   -- Notes

61 Phase 1 Step 2.5: Locate output files
   -- Who did it {Human, Agent, Both}

62 Phase 1 Step 2.5: Locate output files
   -- Outcome {Success, Failure, Partial Success}
```

```
63 Phase 1 Step 2.5: Locate output files -- Notes

64 Phase 2 Blocker 1: What was the blocker
65 Phase 2 Blocker 1: Who got stuck {Human, Agent, Both}
66 Phase 2 Blocker 1: What was the resolution
67 Phase 2 Blocker 1: Intervention needed? {Yes, No}

68 Phase 2 Blocker 2: What was the blocker
69 Phase 2 Blocker 2: Who got stuck {Human, Agent, Both}
70 Phase 2 Blocker 2: What was the resolution
71 Phase 2 Blocker 2: Intervention needed? {Yes, No}

72 Phase 2 Blocker 3: What was the blocker
73 Phase 2 Blocker 3: Who got stuck {Human, Agent, Both}
74 Phase 2 Blocker 3: What was the resolution
75 Phase 2 Blocker 3: Intervention needed? {Yes, No}

============================================================
Part III: Result Evaluation and Blockers
(Questions 76--96)
============================================================

76 Phase 3 Step 3.1: Parse/extract our results
   -- Who did it {Human, Agent, Both,
   N/A -- no results produced}

77 Phase 3 Step 3.1: Parse/extract our results
   -- Outcome {Success, Failure, Partial Success,
   N/A -- no results produced}

78 Phase 3 Step 3.1: Parse/extract our results -- Notes

79 Phase 3 Step 3.2: Compare to paper values
   -- Who did it {Human, Agent, Both,
   N/A -- no results produced}

80 Phase 3 Step 3.2: Compare to paper values
   -- Outcome {Success, Failure, Partial Success,
   N/A -- no results produced}

81 Phase 3 Step 3.2: Compare to paper values -- Notes

82 Phase 3 Step 3.3: Investigate discrepancies (if any)
   -- Who did it {Human, Agent, Both,
   N/A -- no results produced}

83 Phase 3 Step 3.3: Investigate discrepancies (if any)
   -- Outcome {Success, Failure, Partial Success,
   N/A -- no results produced}

84 Phase 3 Step 3.3: Investigate discrepancies (if any)
   -- Notes

85 Phase 3 Blocker 1: What was the blocker
86 Phase 3 Blocker 1: Who got stuck {Human, Agent, Both}
87 Phase 3 Blocker 1: What was the resolution
88 Phase 3 Blocker 1: Intervention needed? {Yes, No}

89 Phase 3 Blocker 2: What was the blocker
```

```
90 Phase 3 Blocker 2: Who got stuck {Human, Agent, Both}
91 Phase 3 Blocker 2: What was the resolution
92 Phase 3 Blocker 2: Intervention needed? {Yes, No}

93 Phase 3 Blocker 3: What was the blocker
94 Phase 3 Blocker 3: Who got stuck {Human, Agent, Both}
95 Phase 3 Blocker 3: What was the resolution
96 Phase 3 Blocker 3: Intervention needed? {Yes, No}

============================================================
Part IV-A: Collaboration Patterns, Agent Contribution,
Struggle Analysis, and Reproduction Failure Classification
(Questions 97--101)
============================================================

97 Collaboration pattern observed
   *If multiple choices apply, use the "Other" freeform text
   field.

   1. Agent did all the work on its own
   2. Agent asked for human input less than 5 times
   3. Human had to provide a minor suggestion or two to
      redirect agent on the right path
   4. Agent made major error(s), requiring human redirection
   5. Agent stopped before completing full answer(s),
      requiring human prodding to continue
   6. Agent asked for human input/assistance for several
      steps
   7. Agent and human worked back-and-forth as near-equal
      partners
   8. Agent completed task but required significant scope
      clarification upfront
   9. Agent failed completely
   10. Other:

98 Where Agent added value

   1. Navigating readme and necessary associated files
      quickly to understand requirements
   2. Environment setup
   3. Downloading data
   4. Identifying main scripts
   5. Running code
   6. Debugging errors from running code as is
   7. Making the most appropriate choice to adjust code to
      run correctly
   8. Identifying deprecated code/requirements and quickly
      finding fixes
   9. Catching potential issues proactively (e.g., noticing
      a bug in the code before it caused a major error)
   10. Finding an alternative more efficient approach
   11. Interpreting intermediate results intelligently so
       that it could move on quickly to next steps
   12. Other:

99 Where Agent struggled and needed help

   1. Understanding the initial prompt
   2. Following README instructions
```

```
   3. Setting up environment as directed in readme or
      repository
   4. Identifying data source and downloading it correctly
   5. Identifying correct scripts needed for reproducing
   6. Making appropriate adjustments for deprecated code
   7. Making an inappropriate adjustment to the source code
      for compatibility
   8. Providing the final answer
   9. Hallucinating file paths, function names or model
      details that didn't exist
   10. Losing track of context
   11. Not knowing when to stop and continuing past the
       correct solution
   12. Failure to produce final results, or to
    check
    obviously incorrect intermediate results
   13. Getting stuck in a loop of retries
   14. Asking clarifying questions too late
   15. Making assumptions about the environment without
       checking
   16. Failure to follow instructions
   17. Other:

100 Reproduction failure mode classification
    (in case reproduction of the given target failed)

   1. Environment setup failure
   2. Missing dependencies
   3. Data access issues
   4. Ambiguous instructions
   5. Code bugs
   6. Conceptual misunderstanding
   7. Timeout / resource exhaustion
   8. Results do not match within error tolerance
   9. Other:

101 Other Notes (error messages, surprises, observations -
    anything that doesn't fit above)

============================================================
Part IV-B: Reproduction Results and Execution Duration
(Questions 102--103)
============================================================

102 Reproduction results

    Methodology

    Run once, round to same number of digits as paper's
    value and check if it falls within floating point
    derived tolerance (e.g. 2.2e-11 = 0.000000000022 =
    1e5 * sys.float_info.epsilon). If yes, mark as Match

    If not, run twice more and use the CORE-Bench paper's
    method to generate a tolerance interval from the three
    values (plus floating point derived tolerance). You can
    use this Colab notebook for calculating the interval.

    If the target value (from the paper) falls into this
```

Table 16: **Overview of reproduction outcomes by step.** Counts are grouped by executor: Agent (autonomous execution), Both (human–agent collaboration), and Human (human-only execution). Success denotes successful completion; Partial denotes completion with runtime issues or interruptions; Failed denotes unsuccessful completion. N/A denotes steps that were not applicable.

	Agent		Both			Human	
Step	Success	Partial	Success	Partial	Failed	Success	**N/A**
1. Environment setup							
1.1 Start the instance and Docker image*	0	0	0	0	0	25	0
1.2 Start Agent with logging and prompt replication target task*	0	0	0	0	0	25	0
1.3 Obtain the paper's code	25	0	0	0	0	0	0
1.4 Read README	25	0	0	0	0	0	0
1.5 Create environment	25	0	0	0	0	0	0
1.6 Install dependencies	23	0	2	0	0	0	0
1.7 Download/prepare data	25	0	0	0	0	0	0
1.8 Verify setup	25	0	0	0	0	0	0
2. Code execution							
2.1 Identify entry point / main script	25	0	0	0	0	0	0
2.2 Understand required run parameters	24	0	1	0	0	0	0
2.3 Run code	20	2	3	0	0	0	0
2.4 Monitor / debug runtime errors	24	0	1	0	0	0	0
2.5 Locate output files	25	0	0	0	0	0	0
3. Results and comparison							
3.1 Parse/extract our results	24	0	0	0	0	0	1
3.2 Compare to paper values	22	0	2	0	0	0	1
3.3 Investigate discrepancies	17	1	3	0	0	0	4

*These two steps were always executed by the human evaluator, by design.

```
    interval, mark as Within tolerance interval . If not, mark as Fail

    Also refer to the instructions in the default agent
    prompt

    a) Results: Question
    b) Results: Paper Value
    c) Results: Our Value
    d) Results: Match {Match, Within tolerance interval, Fail}

103 Total duration (in minutes)

    Measure by: Start from difference between first and
    last timestamp (as provided by script), manually
    subtract lunch breaks etc. (afk time), add any
    additional time for analysis etc. after the last
    timestamp
```

A.5.10 RANDOMIZED STUDY OBSERVATIONS

We provide a few examples of instances where the agent overcame an operational blocker:

1. In *Beyond Accuracy Behavioral Testing of NLP Models with CheckList*, the run only progressed after rebuilding an older Python stack so the released suite could deserialize.
2. In *Yellow Vests, Pessimistic Beliefs, and Carbon Tax Aversion*, the agent had to move to an older `R 4.0.3` environment after the modern stack repeatedly failed.

Table 17: Evaluator-reported blockers in human-agent collaboration sessions

Metric	Value
Sessions with at least 1 substantive blocker	11 (44%)
Total substantive blocker events	30
Sessions with at least 1 blocker requiring human intervention	5 (20%)
Blocker events requiring human intervention	10 (33%)
Mean blocker events per affected session	2.73

Note. Blocker items were annotated only for human-agent collaborative sessions. Percentages are therefore calculated over human-agent collaborative sessions ($N = 25$) or blocker events ($N = 30$), as appropriate. One missing intervention flag was adjudicated as requiring intervention based on its description.

Table 18: Illustrative evaluator-reported blockers in human-agent collaborative sessions

Example blocker	Resolution	Intervention?
"The first run failed before the code started because this container doesn't have /usr/bin/time." (according to the agent)	"I'm rerunning the same preprocessing command without that wrapper."	No
According to the agent: "nltk==3.9 imports wordnet at module import time in this environment, so the original script stops before preprocessing begins."	According to the agent: "I've hit the same nltk import bug twice now. One final environment-only fix is reasonable here: swap nltk to 3.8.1, which still provides the nltk.util.ngrams API this script uses but avoids the unrelated wordnet import-time failure on this Python 3.11 setup."	No
Agent ran code on wrong dataset sample	Told agent to consult paper for dataset config.	Yes
"package 'oglmx' is not available for this version of R" (and similar for others)	removed as unnecessary for replication target	No
The code hit a difficult looking bug involving exhaustion of the C stack.	The agent stopped to check in with the user (as requested in the prompt), and suggested resorting to the older R version specified in the README, which worked after approval by the analyst	Yes
The agent began with a smoke test and then paused to request guidance on next steps, likely recognizing that training all 40 models from scratch would be computationally intensive.	The agent estimated that completing the full training would take over 10 days, which exceeded available resources. Based on this constraint, the agent and the human researcher shifted the approach to using pretrained checkpoints to assess reproducibility.	Yes
Agent misinterpreted the instructions and ran models with hyperparameters in the repo.	Human researcher advised the agent to follow the original instructions provided in the prompt: reproduce paper results	Yes

Note. Entries are reproduced verbatim from evaluator responses, except for LaTeX escaping and line wrapping. Examples were selected to illustrate the range of blockers and are not exhaustive.

3. In *Multiracial Identity and Political Preferences* and *Informer*, the agent recreated expected filesystem layouts or traced historical code paths before the relevant pipeline could be evaluated.

A.5.11 CORRECTNESS AUDIT

To adjudicate correctness of reproduction attempts, we conducted a retrospective, non-blinded correctness audit of all 50 reproduction attempts using the completed surveys, human and agent

Table 19: **Where the agent was perceived to be useful for human-agent collaborative reproduction runs.** Multiple selections were allowed per run. We consider an agent to be useful at a particular step in the human-agent collaboration runs based on the reproducer's judgement of steps they would have found difficult to fix without agent assistance.

Where Agent added value	Mentions across runs
Environment setup	25
Running code	23
Identifying main scripts	20
Navigating readme and necessary associated files quickly to understand requirements	19
Downloading data	17
Debugging errors from running code as is	14
Making the most appropriate choice to adjust code correctly	10
Catching potential issues proactively (e.g., noticing a bug in the code before it caused a major error)	8
Finding an alternative more efficient approach	8
Identifying deprecated code/requirements and quickly finding fixes	7
Interpreting intermediate results intelligently so that it could move on quickly to next steps	6

Table 20: **Where the agent encountered difficulties across human-agent collaboration reproduction runs.** Multiple selections were allowed per run. Fourteen runs reported none of the following areas.

Where the agent struggled	Mentions across runs
Identifying correct scripts needed for reproduction	2
Providing the final answer	2
Setting up environment as directed in the README/repository	2
Making assumptions about the environment without checking	1
Failure to follow instructions	1
Understanding the initial prompt	1
Making inappropriate compatibility adjustments to source code	1
Spending too much time pursuing an incorrect path	1
Forgetting original instructions and rescoping the task	1
Making a decision for the next step	1
Making appropriate adjustments for deprecated code	1
Losing track of context	1
Minor output formatting issues	1

execution traces, the paper's code, and outputs. We use this method rather than relying on the paper's result for correctness because there may exist papers in which a correct code reproduction does not yield the same results as the paper's.

We assessed correctness of the reproduction attempt in two ways:

1. **Method correctness.** The run followed the procedure prescribed in the paper and provided codebase for reproducing the result (e.g. repeating a training run with 10 different seeds).

2. **Output correctness.** The run produced the same outcome as a run adjudicated as method correct. For the 10 social-science papers we selected from the Institute for Replication (I4R) dataset, I4R had confirmed 8 as reproducible; for these papers, we used I4R validated targets to assess output correctness.

Two social science papers we selected were labeled irreproducible at the paper level by I4R. For one, we selected the target I4R found to be non-reproducible. For the other, I4R's report did not validate the target used in our study. For these and the ML papers, we adjudicated output correctness

Table 21: **Target-reproduction comparison between human-agent collaborative and manual reproduction runs.** Result category shows whether the reproduction attempt resulted in a final value for the reproduction target that matched with selected target from the published paper; either exactly or within a tolerance interval (see calculation in A.5.3). If a manual run or human-agent collaborative run determined that the pipeline to reproduce the target value was not present in the code provided, the result was marked as a fail. Five manual runs were marked as failures solely because they exceeded the 3-hour runtime limit.

Result category	Human-agent Collaborative	Manual
Exact match	15	10
Within tolerance interval	3	4
Fail	7	11

Table 22: **Additional evaluator observations from human-agent collaboration reproduction runs.** Most runs required no notable intervention and were completed successfully by the agent. Reported observations primarily related to execution efficiency, scope interpretation, runtime optimization, and the agent's handling of discrepancies or recovery from initial errors. We reported no additional notes for 20 runs.

Other notes
Although the agent sought human guidance for the next step, it showed reasonable judgment by recognizing the computational cost and avoiding full pretraining, which would have required more than 10 days.
Highly efficient agent run that successfully reproduced the result
It was still somewhat impressive seeing the agent work its way through resolving the problems resulting from its initial wrong choice, and the eventual successful option went smoothly. Still, the [agent] could have saved over half an hour by following the README instructions (on the required R version etc.) more closely from the beginning.
This particular reproduction went over the 45 minute compute time limit that was imposed as a criterion in the paper selection. I haven't investigated whether the agent could have chosen a more performant (e.g. multi-core) way to run the process. For the two additional runs required after the first result mismatch, it intelligently found a way to make them run in parallel so that they only required about the same time together as the first one alone.
As described in more detail in the notes for [reproduction step] 3.3, on human request the agent was also helpful in investigating the discrepancy of the reproduced result and narrow[ing] down the possible cause. (Since this task is not explicitly specified in our prompt, I still rate this run as "Agent did all the work on its own".)
a very smooth run by the agent

ourselves based on whether the run produced the same result as a method correct run. We provide results from our correctness audit in Table 23.

A.5.12 Agents surface errors in the paper's code

Across all attempts, 72% of human-agent runs matched the paper's results, while only 56% of human-only runs did (See Table 21). However, among the eight reproducible social science papers (ground truth was known), human-agent and human-only runs had similar correctness rates: only one reproduction didn't match the paper for both conditions (1/9 for the human-only condition and 1/10 for the human-agent condition). This divergence serves as evidence that agents can help surface and correct errors in paper's code that make code reproduction results differ from reported paper results. Using agents and LLMs for this purpose has also been explored in prior work (Steve Newman, 2024; Li et al., 2026; Hu et al., 2025). We provide examples of agents correcting errors humans didn't from our reproduction attempts in Table 24.

Table 23: **Post-hoc correctness audit results.** Overall, manual runs had lower correctness rates on both measures, though using the same fixed-effects/CR2 specification as the duration analysis, neither difference was statistically significant after Holm correction for the two correctness outcomes. Thus, agent speedup did not come at the cost of correctness. On 10/20 papers (50%), all human-only & human-agent runs on that paper agreed: 8 unanimously matched the paper, while 2 unanimously failed to match it.

Post-hoc Audit	Human-Agent	Human-only	Difference	Holm-adjusted CR2 p-value
Correct output	22/25 (88%)	14/25 (56%)	+32 pp	0.127
Correct output and method	21/25 (84%)	10/25 (40%)	+44 pp	0.091

Table 24: **Examples of errors in paper's code that the agent surfaced but the human did not**. This could explain why a greater proportion of human-agent runs matched the paper's results than human-only runs.

Paper title	Error surfaced by the agent
Improving evaluation of machine translation quality estimation (Graham, 2015)	The authors changed their code to fix a bug, but the historical R script supplied the wrong sample size (`n = length(a$HUMAN) # 2,288 rows across all systems`). The Williams test should use the paired observations for the two systems being compared (`n = length(m1$HUMAN) # 208 segments`)
Informer: Beyond efficient transformer for long sequence time-series forecasting (Zhou et al., 2021)	The code drops the last 9 of 2857 windows during test evaluation, although this did not significantly affect the final result.
Reliable conflictive multi view learning (Xu et al., 2024)	The appendix says each view is processed by a fully connected network containing a ReLU layer. However, the released code uses a dimension list containing only the input dimension. The agent found this by comparing `model.py` against the appendix.
Policy deliberation and voter persuasion: Experimental evidence from an election in the Philippines (López-Moctezuma et al., 2022)	Running the code from the paper's entry point generates Table 1 (the target) and then runs an appendix that reuses the variables that make up Table 1, overwriting them to different values. The human ran the code to generate only Table 1, so didn't find this bug, while the agent ran the full code.
Decentralization can increase cooperation among public officials (Molina-Garzón et al., 2022)	The replication script's appendix analysis overwrites the Table 3 model variable with a new model with a different value. The agent ran the entire pipeline and noticed a mismatch.

A.6 Scaffold- and model-level failure mode decomposition examples by capsule

We decompose accuracy along model and scaffold in Figure 5, Figure 6, and Figure 7. We present the following additional findings:

Changing the scaffold alone can rescue performance. In `capsule-1175539`, CORE-Agent's output format triggers early termination before the intended R analysis runs, while Codex CLI provides enough iteration budget for the same model to adapt to the library-path issue and complete the pipeline. This pattern extends broadly: 18 of GPT-5.4's 19 CORE-Agent failures recover in at least one Codex CLI configuration (17 at matched reasoning effort), with zero regressions. Message counts reinforce this reading: GPT-5.4 averages 36.8 messages on passing CORE-Agent runs and 36.0 on failing ones, suggesting the model does not change effort in response to difficulty. The

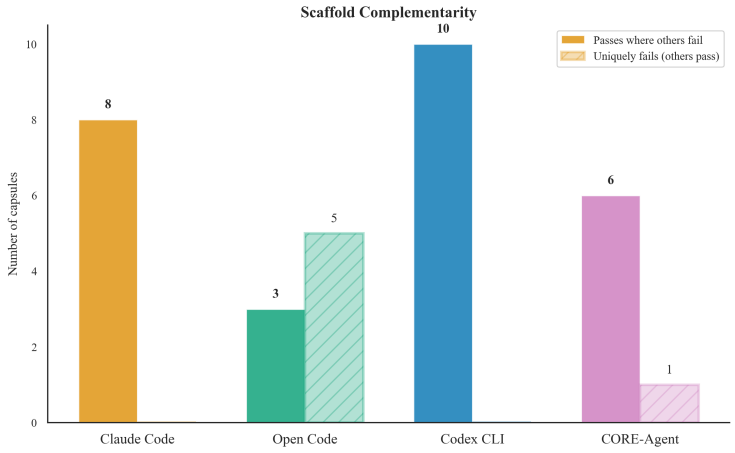


Figure 5: **Scaffold complementarity across capsules.** Solid bars are cases where a scaffold passes while at least one other scaffold fails. Hatched bars are cases where the scaffold uniquely fails while others pass. Codex CLI provides the largest number of rescues with no unique failures in this slice, while CORE-Agent rescues some capsules but also uniquely fails others.

recovered runs also require a message budget comparable to capsules that pass in both scaffolds (75 vs. 70), consistent with scaffold-imposed constraints rather than intrinsic task difficulty. Every one of GPT-5.4's 19 CORE-Agent failures passes under at least one alternative scaffold; none is a universal failure.

Some trajectories follow the model, not the scaffold. For `capsule-4252248`, Opus 4.6 computes the correct value (0.4929241) in three separate scaffolds, then submits the rounded figure-legend value (0.493) each time. GPT-5.4 and Opus 4.5, both with OpenCode, extract the value directly from code output without consulting the figure. The behavior recurs across scaffolds, pointing to a possible model-level tendency.

Some failures depend on the match between agent speed and scaffold constraints. In `capsule-5136217`, Claude Code with Opus 4.6 resolves the task in 63 messages and never encounters the `bsts`-dependent code, while Opus 4.5 in the same scaffold spends most of its 262 messages on package installation and is cut off by the 2,700s timeout before answer collection. In `capsule-0851068`, the pattern reverses: Claude Code with Opus 4.6 correctly diagnoses a PyTorch socket-path bug and computes the right AUC, but the timeout expires before the answer is submitted, while the same model in OpenCode reaches the fix faster and completes within budget. In both cases the model can solve the task; whether it finishes depends on how its working pace aligns with the scaffold's time limit.

A.7 Reliability estimates and uncertainty

Table 25: **Reliability estimates and standard errors for Codex CLI.** Calibration, outcome consistency, and resource consistency estimates for the five models evaluated in Section 3.1. SE denotes standard error; the bootstrap SE corresponds to calibration.

Model	Calibration	Bootstrap SE	Outcome consistency	Outcome SE	Resource consistency	Resource SE
GPT-5	0.296	0.025	0.803	0.056	0.654	0.023
GPT-5.1	0.532	0.031	0.729	0.061	0.609	0.035
GPT-5.2	0.353	0.025	0.885	0.045	0.637	0.032
GPT-5.3-Codex	0.512	0.027	0.918	0.040	0.657	0.026
GPT-5.4	0.557	0.020	0.967	0.023	0.682	0.021

A.8 Prompt for post-hoc confidence

To elicit post-hoc confidence, we used the prompt in Listing 1 at the end of the agent interaction:



Figure 6: **Per-capsule outcomes across scaffolds for the same model.** Each row is a capsule; each column is a scaffold. GPT-5.4 (medium) has the most scaffold-sensitive tasks (17/39), driven largely by CORE-Agent's 19 failures compared to Codex CLI's 2. Claude Opus 4.5 shows 12/39 scaffold-sensitive tasks, indicating that task-level disagreement can be substantial even when aggregate accuracy is similar.

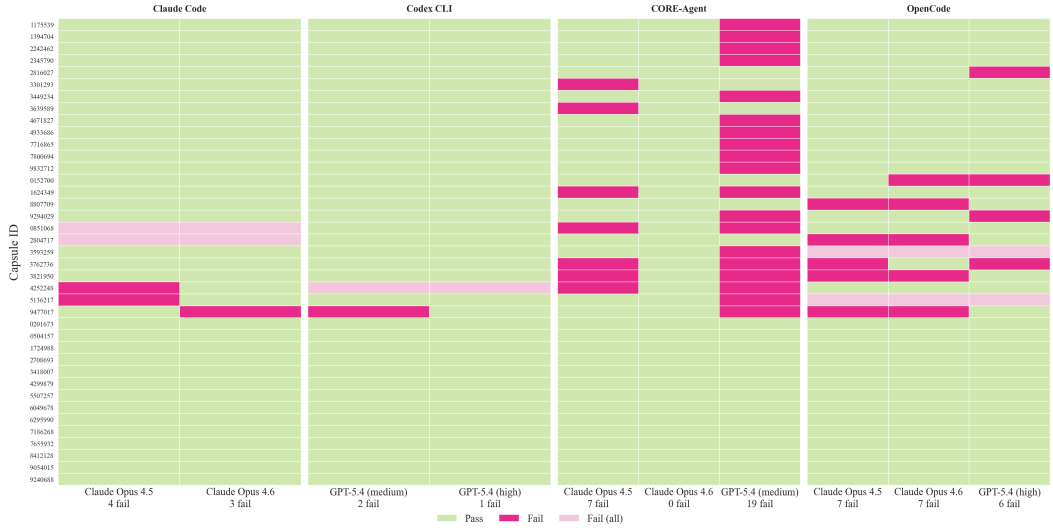


Figure 7: **Per-capsule outcomes across models for the same scaffold.** Each row is a capsule; each column is a model. CORE-Agent shows the widest model sensitivity, with Claude Opus 4.6 passing all 39 tasks compared to 19 failures for GPT-5.4 (medium). Claude Code and Codex CLI show high model agreement, with near-identical failure patterns across their respective model pairs.

```
CONFIDENCE_PROMPT = """You just completed a computational
    reproducibility task. You were asked to reproduce results from a
    scientific paper's codebase and provide answers as a Python
    dictionary. Please assess your confidence in the correctness of
    your submitted answers.

TASK PROMPT:
{task_prompt}

EXECUTION SUMMARY:
{execution_summary}

ANSWER PRODUCED:
{answer_json}

Consider the following when rating your confidence:

1. Did the code execute successfully without errors, or did you
    encounter dependency/environment issues that may have produced
    incorrect outputs?

2. How confident are you that each value in your answer dictionary is
    correct - did you extract values directly from program output, or
    did you have to estimate or interpret ambiguous results?

3. Did the code produce the expected type of output (figures, tables,
    numerical results) that the questions refer to?

4. Were there any discrepancies between what the code produced and
    what the questions asked about (e.g., missing output files,
    incomplete runs, partial results)?

5. Did you verify that your answer dictionary keys exactly match the
    requested field names?

6. For numerical answers, are you confident in the precision and
    format (e.g., rounding, scientific notation, percentage vs decimal
    )?

Please provide a confidence score from 0 to 100, where:

- 0-20: Very uncertain - code failed to run, outputs are missing, or
    answers are guesses

- 21-40: Low confidence - code ran with significant errors, or answers
     were inferred from partial output
- 41-60: Moderate confidence - code ran but some outputs are ambiguous
     or may not match expected format

- 61-80: Good confidence - code ran successfully, most answers
    extracted directly from output

- 81-100: Very confident - code ran cleanly, all answers directly
    extracted and verified against output

Respond with ONLY a number between 0 and 100. No explanation needed.""
    "
```

Listing 1: Confidence assessment prompt