WHAT SWE-BENCH CANNOT TELL APART: THE RESOLUTION LIMIT OF AGENTIC LEADERBOARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Public leaderboards rank coding agents by resolve rate on a few hundred tasks, and the small gaps at the top drive model selection. We assemble the per-task outcomes of 175 public SWE-bench Verified submissions into a 175×500 matrix that reproduces every official score. The top of the board is compressed: the leading ten systems span 3.4 points, adjacent ranks differ by a median of a single task, and among the top thirty, 243 of the 500 tasks are solved by all of them or by none. Under task subsampling, the 0–1 and 1–2 point gap bands need 497 and 421 of the 500 tasks to reach 95% mean ordering preservation. We then examine sensitivity to run-to-run variation: fifteen pairs of same-agent, same-model resubmissions put the per-run flip rate at 5.6% of tasks (95% CI 3.9 to 7.2). Carrying that rate through the resampling, under an independent equal-probability flip model, leaves the bands within two points below 95% mean ordering accuracy at any subset size, the full benchmark included; at the point estimate 293 of 435 top-thirty pairs (67%) fall short. That result is conditional on the noise model, and a band average does not determine the reliability of any individual pair inside it. A comparison with HELM MMLU shows similar task-sampling behaviour once pool size and score range are matched. Tasks chosen by a held-out panel of comparable systems are worth two to three times their number, but the panel has to be contemporaneous: freezing the chosen set and reusing it on later submissions gives the advantage back. Finally, among the 39 submissions publishing per-task dollars, resolve-rate order and cost-per-resolved-issue order are close to unrelated here (Spearman $\rho = -0.08$).

1 INTRODUCTION

Every measuring instrument has a resolution below which its readings carry no information, and a leaderboard is no exception. SWE-bench Verified (Jimenez et al., 2024) has become the reference benchmark for coding agents. Its public leaderboard orders systems by the fraction of 500 GitHub issues they resolve, and the differences it reports at the top are small: as we write, the tenth-ranked system sits 3.4 points below the first, and consecutive entries are typically separated by a single task. These differences are routinely read as results, and they appear in model cards, launch posts and procurement decisions.

We ask whether the benchmark actually supports them. The question is answerable because SWE-bench publishes more than a score: every submission ships its per-task outcomes, so the leaderboard can be reconstructed as a system-by-task matrix and analysed directly, without running a single agent.

Yauney et al. (2026) recently asked a version of this question for multiple-choice benchmarks and found that no method for picking a small evaluation subset ranks models 3.5 points apart reliably, with random sampling competitive once the subset is large enough to work at all. Agentic evaluation differs from that setting in ways that plausibly matter: tasks are long, execution is stochastic, and the benchmark is an order of magnitude smaller than MMLU-Pro.

We find that the per-task statistics are almost identical across the two settings, and that the informative difference lies elsewhere. Our contributions are as follows.

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.

- We reconstruct the public SWE-bench Verified leaderboard as a 175×500 outcome matrix and verify it against every submission's official score (§3). Among the top thirty systems, 243 tasks separate nobody, leaving an effective benchmark of 257 (§4.1).
- We move the subset meta-evaluation of Yauney et al. (2026) onto agentic data. The two closest gap bands need 497 and 421 of the 500 tasks for their mean ordering preservation to reach 95% (§4.2).
- We estimate what re-running costs, from 15 pairs of near-identical resubmissions in the public record, at 5.6% of tasks flipping per run. Folding that in, close pairs stop being resolvable at any subset size, the full benchmark included (§4.3).
- Using HELM (Liang et al., 2023) we show this is not specific to agents: matched for pool size and score range, a single-turn multiple-choice board compresses comparably, and which board is tighter depends on how model families are counted (§4.4).
- Two practical consequences: tasks selected on a held-out panel of comparable systems are worth two to three times their number, and the accuracy ordering is close to unrelated to the cost ordering a deployer actually needs (§5).

The analysis is inexpensive to reproduce: it uses only files that can be downloaded without an account, and ships as a small library that regenerates every number reported here.

2 Related Work

Small-subset evaluation has been pursued for efficiency, through anchor points (Vivek et al., 2023) and item-response models (Maia Polo et al., 2024). Yauney et al. (2026) turned the question around and asked whether such subsets rank models correctly, finding that they largely do not on multiple-choice benchmarks; we move that meta-evaluation to agentic data, add a re-run noise term, and show the behaviour is shared across task types. On uncertainty in reported scores, Miller (2024) formalises evaluation as sampling from a task super-population and Madaan et al. (2024) quantifies seed and configuration variance. Zhang et al. (2026) shows rankings depend on adaptation before testing; the panel selection of §5.1 is a cheaper intervention in the same direction.

Closest to §4.3, Bjarnason et al. (2026) collect 60,000 agent trajectories on SWE-bench Verified and trace run-to-run variation to trajectories diverging within the first few percent of tokens, establishing the size of that variation at a scale we do not attempt; what we add is downstream, carrying that noise through the subset-resolution calculation to a statement about which published gaps a board can resolve. On cost, Kapoor et al. (2024) argue that accuracy-only agent evaluation is misleading and study the joint cost–accuracy frontier, of which §5.2 is the leaderboard-level version. SWE-bench (Jimenez et al., 2024) and its scaffolds (Yang et al., 2024; Wang et al., 2025; Xia et al., 2024) supply our data; work on leaderboard integrity has focused on preference platforms (Chiang et al., 2024) and on whether reported reasoning is faithful (Wang et al., 2026), whereas the failure mode here is neither gaming nor deception but an instrument read past its resolution.

3 Data and Methods

3.1 Reconstructing the Leaderboard

The SWE-bench submission repository holds 182 entries for the Verified split. Each carries either a `results.json` listing the tasks it resolved, or a `per_instance_details.json` giving per-task outcome, API calls and dollar cost, or both.

Two details govern how the matrix is built. First, the schema is inconsistent: some submissions enumerate all 500 tasks, others list only the exceptional categories and leave the rest implicit. Second, the denominator is always 500, which we confirmed by reading the leaderboard generation script rather than inferring it. A task with no record is therefore not counted as resolved in the official score and enters the matrix as a zero; the record does not say whether it was attempted.

Seven submissions carry no per-task record at all, only an aggregate percentage. Writing them as rows of zeros would invent 500 observed failures that nobody measured, so we exclude them and flag them in the metadata. The remaining **175 systems reproduce their official leaderboard score**

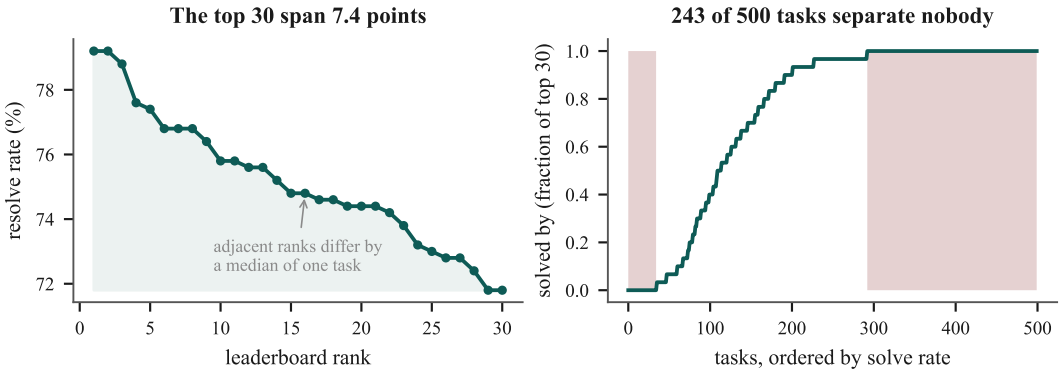

Figure 1: Left: the top thirty systems on SWE-bench Verified, spanning 7.4 points, with consecutive ranks a median of one task apart. Right: among those thirty, 35 tasks are solved by none of them and 208 by all of them. The shaded regions carry no information about the ordering.

exactly, with no tolerance. That checks the totals. The per-task labels have no independent source to check against, so a compensating pair of errors within one submission would pass unnoticed.

Forty submissions publish per-task dollar cost. One of those resolved nothing and has no cost per resolved issue, leaving 39 for the cost analysis in §5.2. We also build the same matrix for SWE-bench Lite (84 systems, 300 tasks, all 84 matching their official scores) and use it as a secondary check throughout.

One confound deserves early mention. The metadata records how many attempts a system makes: 85 declare a single attempt and 81 declare two or more, 7 of them in the top thirty, and the board mixes the two. Repeating the central analysis on single-attempt systems only barely moves the numbers (§4.2), so this is a fairness problem for the leaderboard rather than a driver of our results.

3.2 Pairwise Subset Resolution

For a pair of systems, only tasks where they *disagree* can change the ordering. Write n_+ for tasks the first solved and the second did not, n_- for the reverse, and n_0 for the rest. A random subset of size k is a multivariate hypergeometric draw over the three groups, and it preserves the full-benchmark ordering exactly when the sampled n_+ exceeds the sampled n_-. Every pair thus collapses to a triple, identical triples need simulating once, and the 15,225 pairs reduce to 6,365 distinct cases. We report the fraction of subsets reproducing a pair's observed ordering, averaged within bands of that gap, and define a band's requirement as the smallest subset at which that average reaches 95%; it is a property of the band, not a promise about any pair in it. A pair tied on the full benchmark has no correct ordering, so we score it 0.5 and report bands with and without such pairs where it matters.

3.3 A Model of Re-run Variation

The subset analysis above holds each system's per-task outcome fixed and varies only which tasks are drawn, which is a lower bound on the uncertainty, because an agent that is run again does not reproduce itself exactly. We therefore add a second source of variation. Under an independent-flip model, each run flips a given task's outcome with probability p, so two independent runs of the same system disagree on a task with probability $2p(1-p)$. Section 4.3 estimates p from the public record and propagates it through the subset calculation; the model is deliberately simple, and §4.3 reports what it does and does not capture.

4 Experiments

4.1 Compression at the Top of the Leaderboard

Figure 1 summarises the scale of the distinctions the benchmark is asked to make. The leading ten systems span 3.4 points, the leading twenty span 4.8, and the leading thirty span 7.4. The median gap

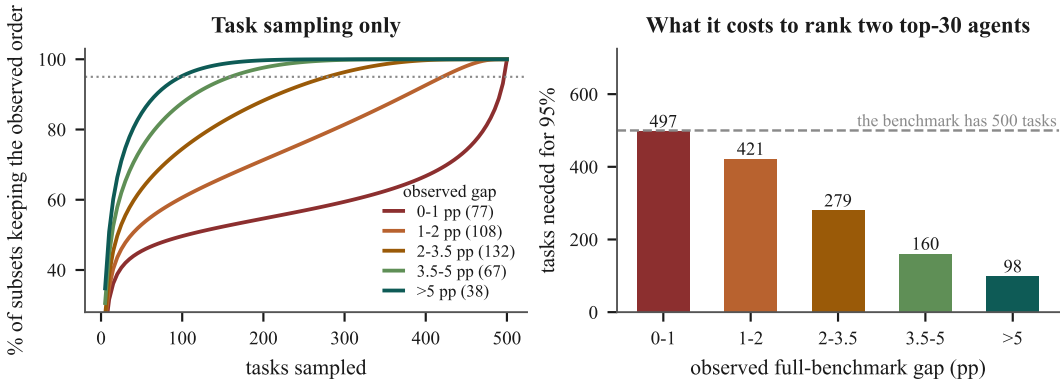

Figure 2: Ranking the top thirty of SWE-bench Verified. Left: the fraction of random subsets that reproduce a pair's observed full-benchmark ordering, by the size of that observed gap. Right: the smallest budget at which each band's mean reaches 95%, bisected exactly. The two closest bands need 497 and 421 of the 500 tasks.

Table 1: What it takes to order two top-thirty systems correctly 95% of the time. The third column is the smallest budget at which the band's *mean* preservation reaches 95%, computed in closed form from the multivariate hypergeometric and bisected to the single task rather than searched over a grid. It is a band summary, not a guarantee for every pair in the band, which is what the last three columns report. The last three columns report, per pair rather than per band, the share that a re-run would mis-order more than 5% of the time at each flip rate; the estimate is $p = 0.056$ with a 95% interval of 0.039 to 0.072, and the interval reflects sampling variation only (§4.3).

		at $p = 0$	share of pairs unresolvable at flip rate p		
observed gap	pairs	tasks needed	0.039	0.056	0.072
0–1 pp	77	497	100%	100%	100%
1–2 pp	108	421	100%	100%	100%
2–3.5 pp	132	279	28%	72%	92%
3.5–5 pp	67	160	0%	0%	0%
$>$ 5 pp	38	98	0%	0%	0%
share of all 435 pairs		—	54%	67%	74%

between consecutive ranks is 0.20 points, which on a 500-task benchmark is one task. Thirteen pairs inside the top thirty hold *identical* scores while occupying different ranks, in nine tied clusters, one of them at the top of the board.

Compression alone would be tolerable if every task contributed evidence, but most do not: among the top thirty, 208 tasks are solved by all and 35 by none, so 243 of 500 cannot distinguish any pair in the group. **The effective benchmark for ranking current systems is 257 tasks**, half of the nominal size.

4.2 Task Budget Required for Reliable Ordering

Figure 2 and Table 1 give the answer. Pairs 3.5 to 5 points apart are ordered reliably by 200 tasks and pairs further apart by 100. Below that the cost rises sharply: 2 to 3.5 points needs 300 tasks, and **anything under two points needs almost the whole benchmark**, 497 and 421 of the 500. Since 76% of top-thirty pairs sit within 3.5 points and 46% within two, most of the comparisons the leaderboard displays consume the entire benchmark with nothing left over.

Restricting to single-attempt systems moves the requirements by at most six tasks (426 against 421 for the 1–2 point band, 274 against 279 for 2–3.5, 163 against 160 for 3.5–5), so the multi-attempt confound of §3 is not what drives the result. What is measured here is preservation of the ordering this particular run reports, not evidence that one system's expected performance exceeds the other's.

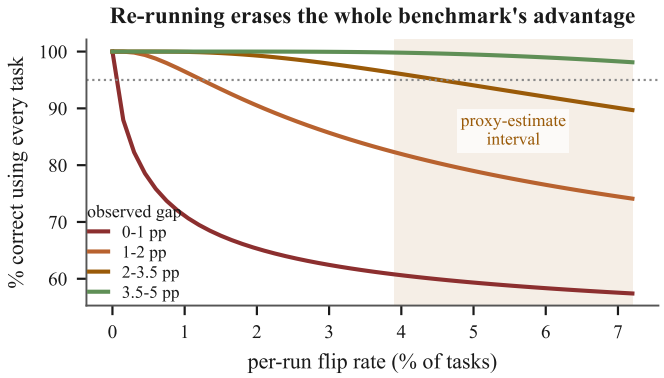

Figure 3: Ordering accuracy using *every* task, as re-run noise increases. The shaded band is the interval on the proxy estimate, which covers sampling variation in the 15 resubmission pairs and not the configuration differences they carry. Beyond a flip rate of about two percent the closest bands never reach 95%, no matter how many tasks are used.

4.3 The Effect of Re-run Variation

We now estimate the flip rate p of §3.3 and propagate it through the subset calculation. The public record already contains evidence that run-to-run variation is not negligible. Twenty-one `(agent, model)` combinations appear more than once. Taking the pairs whose total scores differ by less than two points, two submissions of the same agent with the same model, **0.8 points apart in score, disagree on 64 to 72 individual tasks** out of 500. The aggregate looks stable only because the flips cancel.

Two pairs is too thin to build on. We widen the base by regressing per-task disagreement on score gap within strata of how related two systems are, and reading the intercept, which estimates the disagreement expected between two systems of identical score. This uses 15 same-agent same-model pairs rather than 2, and gives an intercept of 10.5% (95% CI 7.5 to 13.4). Inverting the independent-flip model of §3.3 gives $p = 0.056$**, CI 0.039 to 0.072**.

One side observation is worth recording: at score gaps under one point, two submissions of the same agent and model disagree on 13.6% of tasks, while two *entirely unrelated* systems that happen to tie disagree on 17.6%. Only four points therefore separate re-running a system from replacing it. We cannot decompose the two components, so we report the gap without attributing it.

Folding the flip rate into the resampling (Figure 3) leaves some bands without any sufficient subset size: the requirement is not merely larger, it does not exist within the benchmark (Table 1). Across the whole confidence interval the 0–1 and 1–2 point bands stay below 95% **on average** at any subset size, the full benchmark included. A band average is not a guarantee about every pair in it, because a band spans a range of leads: at $p = 0.015$, 160 of the 198 top-thirty pairs within two points stay below 95% on the full benchmark and 38 clear it, and a pair 1.8 points apart reaches 98.4% at $p = 0.015$, failing only at $p = 0.025$. On the full benchmark the preservation probability depends on the net lead alone, since a separating task and a tied task contribute the same variance $2q(1-q)$. At the point estimate of 0.056 the failure reaches into the 2–3.5 point band, where 95 of 132 pairs fall short, and 293 of 435 top-thirty pairs (67%) fail overall; even at $p = 0.02$ the two closest bands are already unreachable on average.

As a rough summary: once re-running is allowed for, the smallest gap at which most top-thirty pairs are still ordered reliably moves from below one point to around **3.5 points**. It is a summary of a distribution, not a threshold that applies pair by pair.

Limitation. The 15 pairs are resubmissions rather than controlled reruns: they differ in reasoning effort or scaffold variant as well as in random seed, so their disagreement mixes run-to-run variation with systematic differences between configurations, and regressing to zero score gap does not remove that component. We therefore treat 5.6% as a proxy that configuration differences should push upward, not a measurement, and its interval covers sampling variation only. What the conclusion

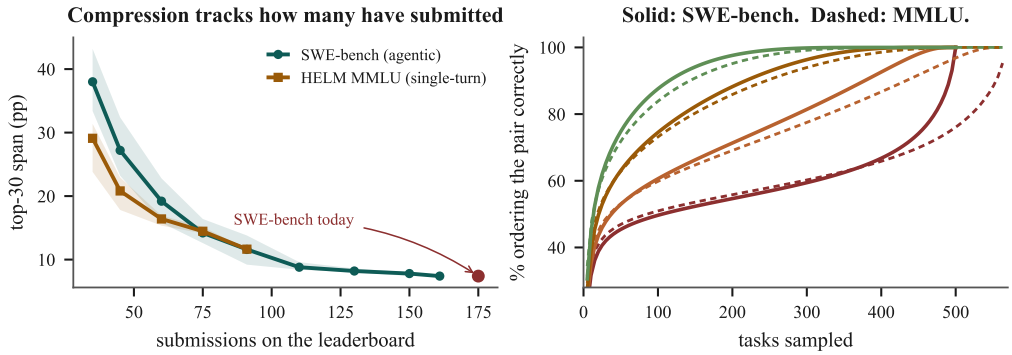

Figure 4: Left: top-thirty span as a function of how many systems the leaderboard holds, with pools matched for score range; bands are the 10th to 90th percentile over subsamples. Right: ordering accuracy against subset size on both benchmarks, by gap band. The two behave the same.

needs is a threshold: the claim that the bands within two points average below 95% holds for any $p \geq 0.015$, narrows to one point at $p \leq 0.010$, and disappears at $p = 0$. Measuring p properly requires repeated runs under a frozen configuration, so we ran them at the scale a small budget allows.

A small controlled check. The measurement this calls for already exists at scale: Bjarnason et al. (2026) run 60,000 trajectories over three models and two scaffolds and see single-run scores move 2.2 to 6.0 points at temperature 0. We ran three of our own only to confirm that a self-hosted configuration reproduces a published submission closely enough for this paper to rest on the public record: Devstral-Small-2512 under mini-SWE-agent, frozen byte-for-byte, resolves 11, 11 and 10 of 13 tasks against 10 for the published one. Exactly one task is non-unanimous, giving $p = 0.026$ (95% interval 0.0006 to 0.140) — a consistency check, not evidence about the rate. Two details survive: the point estimate is half the proxy's, the direction predicted for an estimate inflated by configuration differences, and the one flip was the agent looping until its step budget ran out, the early-divergence behaviour Bjarnason et al. (2026) characterise at token level. Appendix B gives the design.

4.4 Control Experiment: A Single-Turn Benchmark

We expected agentic tasks to be noisier per item and the agentic leaderboard to be more compressed than a single-turn one. A control experiment supported neither expectation.

For the control we needed a single-turn benchmark with per-task outcomes for many models. The Open LLM Leaderboard publishes such files, but they are gated and return HTTP 401 without an approved account, which would undermine a reproducibility claim we make about our own work. HELM (Liang et al., 2023) publishes the same kind of data openly, so we take MMLU (Hendrycks et al., 2021) across its five HELM subjects: **91 models on 567 tasks**, close enough to 500 that the two analyses read side by side.

Task sampling behaves identically. At the same subset size and observed gap the two benchmarks track each other closely, though not to within a point everywhere: of the fifteen cells of Table 2, five differ by under one point and the largest difference is 3.8. At 100 tasks, pairs 2 to 3.5 points apart are ordered correctly 73.2% of the time on MMLU and 74.3% on SWE-bench; at 250 tasks, 90.3% and 92.9% (Figure 4, right). At matched pool size and gap band, the two boards need comparable numbers of tasks, so on this comparison the difficulty of agentic evaluation does not come from individual tasks carrying less signal.

Compression tracks submission count. The raw numbers appear to favour our expectation, since the SWE-bench top thirty spans 7.4 points against MMLU's 11.6, but SWE-bench has 175 submissions and MMLU only 91. Subsampling SWE-bench to 91 systems gives a median top-thirty span of 13.2 points, *wider* than MMLU's, with MMLU's actual value at the 24th percentile of that distribution. Matching score range as well, the two agree closely: at pools of 91 both give 11.6 points, and at 75 they give 14.2 and 14.5 (Figure 4, left). They separate only at the smallest pools, 37.6

Table 2: Ordering accuracy at a matched task budget, agentic against single-turn multiple choice. Each cell is the percentage of random subsets that order the pair correctly. The curves follow the same shape; cell-wise differences reach 3.8 points, with five of the fifteen cells under one.

	50 tasks		100 tasks		250 tasks	
observed gap	MMLU	SWE	MMLU	SWE	MMLU	SWE
0–1 pp	46.9	45.5	50.9	49.7	58.0	56.9
1–2 pp	53.0	53.0	59.8	60.6	73.4	76.3
2–3.5 pp	62.6	63.0	73.2	74.3	90.3	92.9
3.5–5 pp	71.8	74.7	83.8	87.6	97.7	99.2
> 5 pp	86.2	84.6	95.2	95.3	99.9	100.0

against 28.9 at 35 systems, where subsampling variance is largest; we report the whole curve rather than only the matched end.

One objection is that SWE-bench's top is dense because the same model appears under many scaffolds, with 64 of its 175 submissions Anthropic-backed. Capping each family at five systems removes that and leaves SWE-bench 5 to 6 points *wider* than MMLU at matched pool size. The comparison is therefore sensitive to family composition, and what we can claim is that the apparent compression gap is not a robust property of task type, not that family repetition plays no part in it.

The resolution wall therefore tracks leaderboard maturity rather than task type. Two boards of different kinds behave the same way as submissions accumulate, and SWE-bench Verified is the further along of them. Whether every benchmark follows this curve is a conjecture these two cases support but cannot settle, and we have no third board with enough public submissions to test it.

One asymmetry survives and we have not measured it. Our flip rate is agent-specific, whereas greedy-decoded multiple choice is presumably far more deterministic. HELM publishes no repeated runs, so we state this as an expectation rather than a result; it is the one place where agentic evaluation may genuinely be worse off, and the natural next experiment.

5 Analysis

5.1 Selecting Tasks That Discriminate

If 243 tasks separate no pair of systems, the obvious response is to drop them. That response is circular, however, because knowing which tasks are dead requires having already run the systems you want to rank. Selecting on the top thirty themselves buys 13 points of ordering accuracy at 100 tasks, while selecting on the other 145 systems buys 1.0, because a pool spanning 2023 baselines to 2026 agents calls 442 of 500 tasks discriminative and so selects nothing. What works is a *reference panel* of comparable strength, held out from the systems being ranked.

Protocol. We separate exploration from validation by date: submissions before 2025-07-01 form the development set on which the panel ranks and threshold were swept and frozen, the 75 from that date onward form the test set, and a second cutoff at 2025-04-01 repeats the procedure. The split is retrospective, so the guarantee is narrower than a prospective one; Appendix A states what was and was not held out.

What transfers and what does not. The *rule* transfers when the panel is drawn at the same time: hyperparameters chosen on 100 older systems gain 13.7 to 20.6 points on the 75 later ones (Table 3). The chosen task set does not. Freezing the tasks and reusing them later gives up the whole advantage — the 0–1 band falls from $+13.7$ to -1.0 — and at the earlier cutoff drops below random by 6 to 22 points, so we claim only the first. The tasks it keeps explain why: what separated weaker historical systems are easy tasks that every later system solves. The gain is not specific to our rule either. A variance-maximising selector, the instinct behind anchor-point and item-response methods, gains 7.9 to 17.8 points on the same contemporaneous panel, so both simple rules collect most of what the panel offers, ours doing better on the closest gaps by 5.8 points in the 0–1 band. We did not

Table 3: Task selection under a freeze. Each cell is the change in ordering accuracy against the exact expectation of drawing 100 tasks at random from all 500, on systems held out from development. Freezing the *task set* gives up the gain; freezing only the *hyperparameters* and re-selecting on a contemporaneous panel is what works, and a variance-maximising selector on the same panel also improves over random, by less.

what is frozen	panel drawn from	0–1	1–2	2–3.5	3.5–5	> 5
the task set, cutoff 1	earlier systems	-1.0	$+1.1$	$+2.7$	$+4.9$	$+3.1$
the task set, cutoff 2	earlier systems	-1.5	-6.4	-12.9	-22.3	-18.2
hyperparameters	contemporaneous peers	$+13.7$	$+18.4$	$+20.6$	$+12.7$	$+4.7$
variance selector	contemporaneous peers	$+7.9$	$+13.1$	$+17.8$	$+12.8$	$+4.7$
hyperparameters	a different model family	-1.2	-2.1	-10.6	-8.2	-5.1

Table 4: What a held-out reference panel buys. This is the *exploratory* analysis over the whole board: targets are the top thirty of all 175 systems, the panel is ranks 31–60, and no system appears in both. It is not the frozen test, which is Table 3 and uses ranks 31–45 chosen on development data. The last column is how many randomly drawn tasks reach the same ordering accuracy as 100 selected ones. Note the level: 57.9% in the closest band is cheaper than random, not reliable.

observed gap	100 selected tasks	equivalent random tasks	ratio
0–1 pp	57.9%	300	$3.0\times$
1–2 pp	70.8%	200	$2.0\times$
2–3.5 pp	88.6%	250	$2.5\times$
3.5–5 pp	98.2%	250	$2.5\times$
> 5 pp	99.9%	250	$2.5\times$

implement the published methods, so this stands in for them rather than tests against them. Panels drawn from another model family lose most of the benefit, and panel *breadth* matters far more than the threshold.

What it buys, and where it applies. With a contemporaneous held-out panel, **100 selected tasks are worth 200 to 300 random ones**, a ratio between 2.0 and 3.2 that holds across every gap band (Table 4); the panel keeps 134 tasks and the comparison is made at a budget of 100. The attached condition is essential: selection must be redone as the board moves, on systems drawn from the same population as those being ranked. The one use the result does not support is the tempting one of fixing a cheap subset now and reusing it on whatever arrives next.

This replicates on SWE-bench Lite at the one band width where Lite has enough systems; no third split has enough public submissions to test further (24, 22 and 1 for the test, multimodal and multilingual splits).

5.2 Accuracy Ordering versus Cost Ordering

Thirty-nine submissions publish per-task dollar costs. These enable a comparison the leaderboard itself does not make. The figures are self-reported but internally consistent: in all 37 submissions giving both, the per-task costs sum to the stated run total. Failed attempts are billed rather than dropped and cost more than successful ones on average, \$0.51 against \$0.28, as an agent that exhausts its budget without converging should. Prices are whatever each submitter paid at submission time, unnormalised across providers and dates.

Per-task cost is heavy-tailed: the 99th percentile is 12.3 times the median, and the 50 most expensive tasks account for 36% of a run's total cost. Even so, *ranking* by cost is easier than ranking by accuracy, because relative cost differences between systems are enormous while accuracy differences are a few points; at 100 tasks the cost ordering survives 96.7% of the time against 92.0% for accuracy.

The two orderings are close to unrelated in this sample. Ranking these systems by dollars per resolved issue rather than by resolve rate gives **Spearman** $\rho = -0.077$, a median rank movement of 15 places

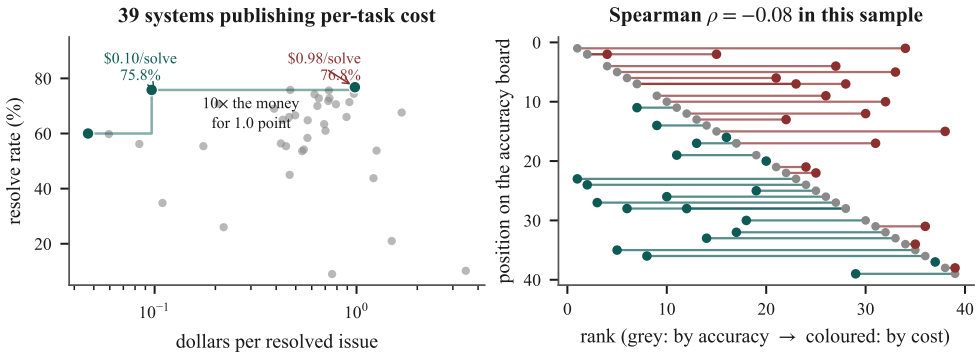

Figure 5: Left: the 39 submissions publishing per-task cost, on the accuracy-cost plane; three are on the frontier. Right: each system's rank by accuracy joined to its rank by dollars per resolved issue.

out of 39 and a maximum of 33: the most accurate system is 34th by cost, and the cheapest resolves issues for $0.05 while sitting 23rd by accuracy. Only **three of 39 systems are Pareto optimal**; the other 36 are beaten on both axes at once.

The top step of the frontier illustrates our argument concretely. Moving from MiniMax M2.5 at 75.8% and $0.10 per resolved issue to Claude 4.5 Opus at 76.8% and $0.98 buys **1.0 point for roughly ten times the money**, and this pair deserves its own numbers rather than an inference from its gap band. The two systems disagree on 55 of the 500 tasks, 25 solved only by MiniMax and 30 only by Claude, so a margin of five tasks decides the ordering. At the estimated flip rate a re-run reproduces the reported ordering 70.6% of the time, reverses it 24.9%, and ties the rest. This comparison is thus settled by five tasks, and at the estimated flip rate would come out the other way about once in four.

Ranking on both axes at once is the expensive operation: preserving a dominance relation needs about 400 tasks for 95% against 200 for accuracy alone, since two orderings must be right simultaneously.

Limitation. Most cost-reporting submissions use the same scaffold, so the twentyfold cost spread largely reflects model pricing rather than scaffold efficiency; the two should be separated before the frontier is read as a statement about agent design. Costs are also self-reported, with no common definition of what is included.

6 Discussion

Four recommendations follow, all inexpensive to adopt and none requiring a new benchmark. **Report a resolution, not just a score,** pair by pair rather than as one board-wide cut-off: at the estimated flip rate 28% of pairs 2 to 3.5 points apart are ordered reliably, while none of the 198 pairs within two points is. **Publish per-task outcomes and costs**; this analysis was possible only because SWE-bench requires the former, and forty submissions reporting the latter sufficed to show that the accuracy ordering is not the one deployers need. **Report re-run variation,** which Bjarnason et al. (2026) show is affordable to measure and large enough to matter at these margins. **Retire tasks that separate nobody, using a held-out contemporaneous panel**: selecting on the systems being ranked is circular, whereas a panel of peers is not, and is worth two to three times its size. None of this is specific to SWE-bench — a multiple-choice board approaches the same wall just as fast.

7 Conclusion

Reconstructed from its public record, SWE-bench Verified resolves far less than the gaps it displays: half its tasks separate no pair of current systems, and the closest gap bands are not reliably ordered at any subset size once run-to-run variation is included. A single-turn board behaves the same way at matched pool size and score range. Two boards cannot establish a general relationship between how crowded a leaderboard is and what it can resolve, but they are enough to show the behaviour is not confined to agentic tasks.

REPRODUCIBILITY STATEMENT

Every input is public and requires no account. The SWE-bench submissions come from the `swe-bench/experiments` repository and the control data from HELM's open storage bucket; we deliberately avoided the gated Open LLM Leaderboard files. Our reconstruction is verified against every submission's official score; the check is per submission on the reported total, not an independent audit of the upstream per-task labels. The analysis ships as a small library whose test suite reproduces the compression, budget and panel figures in this paper from the released matrix; the noise columns of Table 1 are regenerated by their own script. Minimum task budgets are computed in closed form from the multivariate hypergeometric and bisected to the single task, not searched over a grid, so no reported budget is rounded up to a grid point; quantities that need the flip model are computed in closed form by convolution, since each task contributes independently to the net lead; the library also carries a Monte Carlo path for the same quantity, used only where a subset draw and a flip are combined, and it runs with fixed, reported seeds. The interval on the flip rate is a bootstrap over the 15 resubmission pairs and covers sampling variation only, not the configuration differences discussed in §4.3. Every citation was checked against its primary record with the script included in the release. The release pins the upstream data commits, states the simulation seeds alongside each figure and table, and the anonymised repository link is given on the submission page.

AI USE STATEMENT

The authors set the research question and imposed the compute budget that determined the scale of the controlled re-run. Drafts were put through AI-assisted adversarial review; the authors decided how to respond to it and required that every disputed number be recomputed from scratch before any claim was changed. Generative AI tools were used to search and summarise related literature, to give feedback on author's design, to write the analysis the figures, and to polish the manuscript. They were not used to generate synthetic data or to prove mathematical claims; neither applies to this work. The authors reviewed all AI-assisted work: every reported number was verified against the released data and every citation against its primary record. The authors take responsibility for the final manuscript, code and claims.

REFERENCES

Bjarni Haukur Bjarnason, André Silva, and Martin Monperrus. On randomness in agentic evals. *arXiv preprint arXiv:2602.07150*, 2026. arXiv:2602.07150.

Wei-Lin Chiang et al. Chatbot arena: An open platform for evaluating LLMs by human preference. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2403.04132.

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 (ICLR)*, 2021. arXiv:2009.03300.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.06770.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. Ai agents that matter. *arXiv preprint arXiv:2407.01502*, 2024. arXiv:2407.01502.

Percy Liang et al. Holistic evaluation of language models. *Transactions on Machine Learning Research (TMLR)*, 2023. arXiv:2211.09110.

Lovish Madaan et al. Quantifying variance in evaluation benchmarks. *arXiv preprint arXiv:2406.10229*, 2024.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinybenchmarks: Evaluating LLMs with fewer examples. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.14992.

Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations. *arXiv preprint arXiv:2411.00640*, 2024.

Rajan Vivek, Kawin Ethayarajh, Diyi Yang, and Douwe Kiela. Anchor points: Benchmarking models with much fewer examples. *arXiv preprint arXiv:2309.08638*, 2023.

Xingyao Wang et al. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2407.16741.

Xinpeng Wang et al. Is it thinking or cheating? detecting implicit reward hacking by measuring reasoning effort. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.01367.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Agentless: Demystifying LLM-based software engineering agents. *arXiv preprint arXiv:2407.01489*, 2024.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2405.15793.

Gregory Yauney, Shahzaib Saqib Warraich, and Swabha Swayamdipta. How reliable is language model micro-benchmarking? In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.08730.

Guanhua Zhang, Ricardo Dominguez-Olmedo, and Moritz Hardt. Train-before-test harmonizes language model rankings. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2507.05195.

A Task Selection Protocol

A selection method tuned while its targets are visible proves nothing, so we separate exploration from validation by date. Submissions before 2025-07-01 are the development set: 100 systems, on which we swept panel ranks and solve-rate thresholds and chose one setting, panel ranks 31–45 with the threshold $[0.15, 0.85]$. That setting is then frozen. The 75 submissions from 2025-07-01 onward are the test set. The split is retrospective: the full board had been examined before this protocol was written, so the guarantee we can offer is narrower than a prospective one, namely that the panel ranks and threshold were selected using development systems only and applied once, with no re-tuning, to the test systems. A second cutoff at 2025-04-01 repeats the whole procedure.

B Controlled Re-run: Design and Results

What the measurement requires. One system, re-run with everything fixed except sampling: the same weights and decoding parameters, the same scaffold and prompts, the same step budget, and the same evaluation image and commit. A run that changes reasoning effort or scaffold variant reproduces the confound we are trying to remove.

What we ran. Devstral-Small-2512 under mini-SWE-agent, self-hosted on a single H100 through vLLM at temperature 0, configuration frozen byte-for-byte. Three runs over 13 tasks, resolving 11, 11 and 10.

The task count is a compromise forced by cost, and the way it was forced is worth stating because it will recur for anyone repeating this. A first run over 25 randomly drawn tasks returned no patch on 11 of the 24 that started: 10 exhausted the 64k context window and 1 reached the step limit. Those 11 were also the expensive ones. Every task that terminated normally did so within about 104 agent steps and the whole group finished inside 56 minutes, while the 11 that did not ran a further 67 minutes and produced nothing. Repeating the full draw therefore cost roughly four times what repeating the 13 cost, for outcomes that are mostly deterministic zeros.

We did not lower the step limit to make those failures cheaper. At 250 it never binds on these 13 tasks, so leaving it alone costs nothing and keeps the later runs configured identically to the first; lowering it would have made any observed difference uninterpretable.

How this departs from the design. In two ways. The subset is selected on an outcome of the first run, namely that the agent terminated normally. The criterion is not the one that inflates p mechanically — choosing tasks that already looked unstable — but it is still outcome-dependent, and its direction is not known in advance, because excluding tasks the agent cannot finish removes both a source of deterministic zeros and a source of genuine instability. The subset is also easier than the draw it came from: the published submission resolves 76.9% of these 13 against 60.0% of the original 25.

Run 1 also executed alongside the 11 context-exhausting tasks and so under much heavier KV-cache pressure than runs 2 and 3. At temperature 0 the variation between runs comes from continuous batching rather than from sampling, so load is not a neutral background condition. In the event it made no difference: runs 1 and 2 agree on all 13 tasks despite the difference in load.

What it found. Exactly one task is ever non-unanimous, giving $p = 0.026$ with an exact 95% interval of 0.0006 to 0.140; §4.3 reads that result. Two tasks are never resolved and ten always are.

What it does not settle, and why more runs will not fix it. Thirteen tasks cannot resolve a rate this small. Clearing 0.015 with three runs would need three or more of the 13 to come out unstable. Raising the run count does not help: the count required for the interval to clear the threshold rises faster than the expected count does, so at the measured rate the probability of clearing it is about 7% at three runs and no better at five or seven. More tasks is the only lever, and here that means paying for the tasks where the agent exhausts its context and returns nothing.

We also caution against one check that looks stronger than it is. With a single non-unanimous task, the spread of the run scores and the p fitted from the disagreement are both arithmetic consequences of that one flip, so the observed and predicted standard deviations agree exactly and the agreement means nothing. Validating the independent-flip model needs several unstable tasks, and therefore more tasks rather than more runs.

What to report. Three quantities, not one: per-task disagreement between runs, the standard deviation of the total score, and the observed frequency of rank flips between runs. The third is what validates the model behind $2p(1-p)$; the first alone only calibrates its parameter. Percentages should not be carried across denominators: one task is 7.7 points on 13 and 0.2 points on 500, and the re-run here moved one task.

Scale, for anyone repeating this. The public submissions that report per-task call counts average 40 to 60 model calls per task, so one pass over all 500 tasks is roughly 30,000 calls. The binding constraint in practice was not the call count but KV-cache capacity: at 64k tokens per request the server held about 4.6 concurrent full-length conversations, and raising the worker count past that collapsed the prefix cache hit rate from 94% to 1% and cut generation throughput to a fraction of its former value. Agent concurrency should be set from cache capacity, not from the task count. Separately, the Modal evaluation path in `swebench` 5.0.2 does not run: its image builder reads two `TestSpec` fields that were removed when the dataset moved to pre-built images, and its file writer uses a Modal API that has since been retired. Both failures surface as instances with no result rather than as errors in the summary, which is easy to mistake for a model that solved nothing. The release includes the patch.