PRECISE RECORDS, UNRESOLVED RANKS: AUDITING THE RESOLUTION OF AGENT LEADERBOARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

We audit the SWE-bench Verified leaderboard and find that its ranking means two different things. Read as a *record* of what happened under the leaderboard's own scoring rule, it is nearly exact: the median compatible-rank width — the number of rank positions still possible for an entry — is 1, and it rises only to 3 when every genuinely unrecoverable outcome is allowed to vary. Read as a *measurement* of relative performance, it resolves little: simultaneous rank intervals have a median width of 56 positions on a 175-entry leaderboard, and no adjacent pair survives multiple-testing correction. The gap matters to anyone who promotes a system because its score rose. When an acceptance rule keeps the best of several seeds, it promotes all 14 harness variants whose rendered prompts are byte-identical to their baseline — every such promotion is noise. At a budget of 500 tasks, the one-sided fixed-look rule requires a gap of roughly 5.2–6.2 percentage points to reach target power under the discordance envelope measured for one worked pair — several times the gap between adjacent rows. We therefore define the *resolution* of an evaluation — the smallest effect a declared comparison rule reliably detects at a target power — and release an executable tool that produces certificates binding each reported resolution to its data, rule, assumptions, and provenance.

1 INTRODUCTION

A leaderboard rank admits two readings. As a *record*, it states which submission resolved which instances under a scoring rule; that is a census, and its precision is a question of what the published files determine. As a *measurement*, it is taken as evidence that one system is better than another; that is an inference, and its precision is a question of sampling: would the gap survive a different draw of tasks, or a rerun? Papers that cite a change in rank — and search procedures that promote a harness because its score rose (Zhang et al., 2025; 2026) — need the measurement reading. Leaderboards publish only the record.

On the same SWE-bench Verified data (Jimenez et al., 2024), the two readings give sharply different answers. The record is nearly exact, with a median compatible-rank width in the low single digits across entries. The measurement is mostly unresolved: task-iid rank intervals (Neuhof & Benjamini, 2026b) span tens of positions, and no adjacent pair is separable after multiplicity correction (Section 2). The gap is invisible in standard reporting because a score table carries no statement of how large a difference the evaluation could have distinguished (Miller, 2024). We call that missing quantity the evaluation's *resolution*: the minimum detectable effect (MDE) of a declared comparison contract, formalized as $R(C; \rho_d)$ in Equation 2, where C is the comparison contract and ρ_d the declared discordance input. Power inversion is standard; our contribution is an executable certificate generator that binds each result to the full comparison contract and provenance. The accompanying ledger records whether the analysis was frozen before the outcomes were observed. We therefore report resolution per comparison instead of assigning one number to SWE-bench.

Three further questions remain separate, and Table 1 assigns each to a distinct certificate output: *identification* (what the published files determine about instances a submission never reported under the grader's semantics), *dependence* (whether task clustering supports the declared inference), and *provenance* (whether the analysis predated its outcomes).

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.

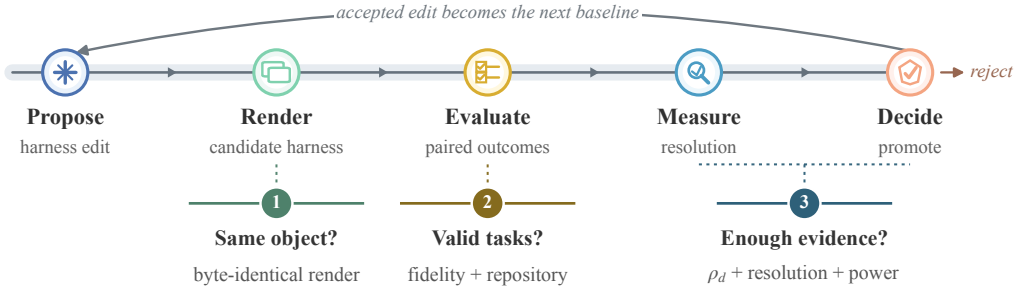

Figure 1: Overview of one candidate–parent decision. Left to right, an edit is rendered, evaluated on paired tasks, measured, and promoted or rejected. Checks 1–3 ask whether the render preserves the intended object, the tasks are evaluable under a justified sampling unit, and the evidence resolves the threshold under the declared sampling and stopping rules. Section 2 audits the public leaderboard; Sections 3.1–3.3 apply the checks to promotion rules and a frozen comparison.

SWE-bench clusters tasks into 12 repositories, too few for exact repository-level inference to reject any member of the declared all-pairs family (Section 2.3); Figure 1 locates these checks within a promotion decision. The certificate records rather than validates assumptions. The accompanying ledger cannot establish temporal priority without an independently timestamped manifest, and the certificate covers one fixed pair rather than run-level error control for an adaptive search.

Contributions. *Finding.* These are two different properties of the same data: a leaderboard record can be nearly exact while the comparisons read off it stay unresolved. *Instrument.* The reusable instrument is a comparison-specific resolution certificate: the conditional estimand $R(C; \rho_d)$ turns a declared pair, error rule, power target, and discordance input into a replayable resolution report. Its executable generator binds the report to its input, contract, and provenance hash; the accompanying ledger records analysis status (Section 3). *Evidence.* Three audits support the finding:

1. A fixed-snapshot audit of 175 published entries separating what the records settle from what sampling leaves open: under the upstream grader's documented semantics the median compatible-rank width across entries is 1, while task-iid rank intervals over that same census have median width 56 of 175 positions (Section 2).
2. A design-level negative result for repository-level inference, computed on the 44 entries whose outcome matrices are complete: with $R = 12$ repository signs, every attainable exact p-value is at least 2^{1-R}, above the initial Holm cutoff α/M of the declared all-pairs family. No member of that family can be rejected on any data, and adding tasks inside the same repositories cannot refine the grid (Section 2.3).
3. An audit of the acceptance rules that consume such scores: on 14 comparisons whose variant and baseline prompts render byte-identically, accepting the best of several seeds promotes all 14 while the confidence-sequence gate promotes 0 (Section 3.2).

Table 1: Certificate workflow. Only the Resolve row defines evaluation resolution; the other rows answer identification, dependence, and provenance questions that a resolution number cannot settle.

Step	Certificate output	Valid inference
Identify	Effect and worst/best completion bounds on a declared census	Statements determined by the reported outcome array; no sampling claim
Resolve	$R(C; \rho_d)$ for a contract C that fixes direction, rule, α, power, task budget, and pilot discordance	Prospective planning only when the pilot and target comparison share the stated conditions
Stress dependence	Repository count and exact sign grid alongside task-iid power	Exact repository inference only under randomized or exchangeable signs; otherwise a design-feasibility sensitivity
Record	Frozen operational rule, analysis status, hashes, and outcome	A later analysis or repeat is a new estimate, not a retroactive change to the decision

Beyond the public leaderboard, a frozen private-endpoint comparison in Section 3.3 exercises the certificate end to end, with its protocol and collection audit in Appendix K.1. Public analyses reproduce offline; private-model results support outcome replay.

2 THE LEADERBOARD RECORD IS PRECISE, BUT ITS RANKS ARE UNRESOLVED

Leaderboards for agentic software tasks publish which instances each submission resolved. Two entries evaluated on the same instance set therefore form a paired outcome array. We separate two questions. Worst/best completion bounds state what the published array determines without a sampling model. Exact McNemar and rank intervals then apply a declared iid discordant-task model as an operational diagnostic; entries differ in base model, scaffold, and budget, so neither output attributes a gap to one component. This section computes both over 175 entries on the 500-instance SWE-bench Verified split (Jimenez et al., 2024). We acquired the per-instance records on 2026-09-11 from the official SWE-bench experiments repository at upstream commit `40f164d5b8f1`; the local manifest verifies every raw file by SHA-256.

For each pair of entries we count discordant instances and apply the two-sided exact McNemar test (McNemar, 1947) at level 0.05. Two-sidedness matters because the higher-rate entry and hence the direction are chosen from the same data. We call a pair *separated* when this declared rule rejects, and also apply Holm correction (Holm, 1979) over all leaderboard pairs before reading off adjacent rows. "Separated" is a deterministic label on the published outcome array under this model, not a calibrated prediction for future tasks.

2.1 WHAT THE RECORDS IDENTIFY

Missing records and the two scoring conventions. The upstream files expose four machine-readable forms of missingness: `no_generation`, `no_logs`, a per-instance record with no API call, or an absent task row; the form does not identify whether the underlying cause was infrastructure, submission policy, or another failure. 104 of 175 entries (59.4%) *name* at least one instance as missing in one of the three cell-level forms, 131 have at least one unknown cell once absent task rows are counted as well, and 44 are complete. Under the two conventions found in circulation — scoring missing as unresolved, or dropping it from the pairing — 1.1% and 2.3% of adjacent pairs separate respectively. Individual rows can turn on the choice completely: one entry leaves 121 instances without an outcome and swings by 21.8 points depending on the convention, from rank 50 to rank 1 of 175 (Appendix D.1). We use the unresolved convention because it is the upstream grader's. The evaluation report of the benchmark's release-pinned harness package sets its total to the full dataset size and files an instance without a prediction as incomplete without removing it from that denominator; the supplementary bundle pins the grader file and its hash. Exclusion is also not a second scoring rule — dropping uncontested instances scores each entry on its own denominator, and comparison-specific denominators do not define one common leaderboard score — so we quote it as a sensitivity only.

Sparse files are summaries, not unknowns. The same semantics settles what an absent task row means. Because an unattempted instance stays in the denominator as not resolved, a submission's file that lists only its resolved instances and a few named failure buckets is a sparse summary of a full-census score, not a record with 26,060 unknown outcomes. The arithmetic rules out the alternative reading: 122 of 175 entries are such sparse summaries, and if their unlisted instances were unknown, their median resolved rate on attempted instances would be 99.1% — against a best full-census rate of 79.2% anywhere in the snapshot. A `no_generation` instance likewise produced no patch, so its outcome is determined. What the record leaves genuinely open are the 257 cells where a run happened but its outcome is unrecoverable (`no_logs`, an empty per-instance record, or an absent row in a file that otherwise enumerates the full task set).

Completion bounds. Completing each genuinely open cell once as zero and once as one bounds every score and rank without choosing a convention for it, in the worst/best-completion tradition of partial identification (Manski, 2003); an entry's *compatible-rank width* is the number of positions attainable across those completions. Table 2 reports what each successively larger freed set leaves

Table 2: One census, four completion regimes. Each row frees a larger set of cells, completes each freed cell once as zero and once as one, and reports the rank ambiguity that remains. All widths are deterministic partial-identification bounds on the fixed census, not confidence intervals; the task-iid rank intervals of Table 10 answer a separate sampling question. Dashes: not computed for this intermediate regime.

	Compatible-rank width		Pair directions
Cells freed	median	maximum	fixed
None (the leaderboard's own rule)	1	4	99.6%
Genuinely open cells (257 cells)	3	74	98.7%
+ every `no_generation` cell	6	–	–
+ every absent task row (26,060 cells)	135	167	32.6%

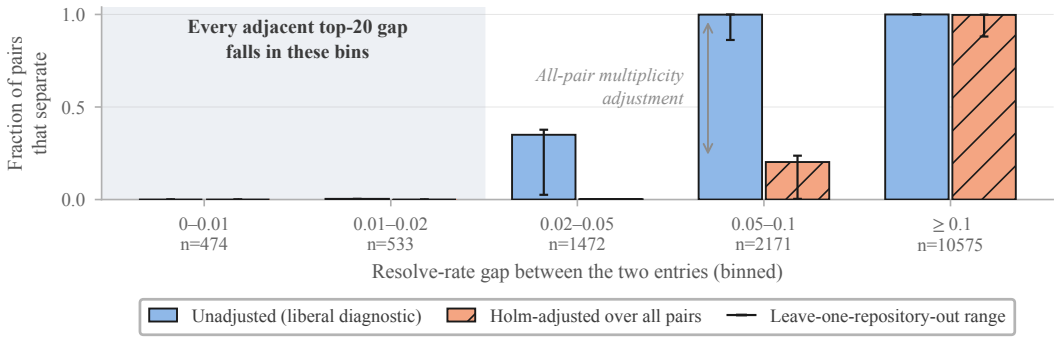

Figure 2: Distinguishability of leaderboard pairs against resolve-rate gap (binned census proportions, not confidence intervals). Blue: unadjusted two-sided exact McNemar, a liberal descriptive diagnostic; coral: Holm-adjusted over all leaderboard pairs before the adjacent subset is read. Capped marks give the range after leaving out each of 12 repositories and recomputing rates, ordering, tests, and bins. Below a gap of 0.01, no pair is distinguishable, and every adjacent top-20 gap falls in the shaded bins.

ambiguous: under the leaderboard's own rule the matrix is complete and only exact ties remain, and freeing the genuinely open cells and then every `no_generation` cell still leaves the median compatible-rank width in the low single digits, so the published records settle the displayed ordering almost completely. The last row is a more hostile analysis that also frees every absent row — treating sparse summaries as if their complements were unknown; we report it as an upper envelope, but what it relaxes is the grader's semantics, not the records.

2.2 What task-level inference resolves

Adjacent displayed rows seldom pass the rule. Adjacency is defined after sorting, so small gaps are expected by construction; we audit adjacent rows because the leaderboard visually orders them, not because SWE-bench is unusually noisy relative to another benchmark. Of the 174 pairs of entries adjacent in the ranking, 1.1% pass the unadjusted exact test and 0.0% survive the all-pair Holm correction. 47 adjacent pairs tie exactly in resolve rate. The systems do not behave identically: every tied pair still disagrees on at least 36 instances. Leaving out each repository cluster and recomputing the full ordering and Holm family puts the unadjusted adjacent fraction between 0.0% and 1.1%; the Holm-adjusted fraction remains 0.0%.

This null is mostly not design-implied in the way the repository grid of Section 2.3 is. There a single floor covers the declared all-pairs family, so rejection is impossible on any data. Here the floors are pair-specific: 170 of 174 adjacent pairs carry enough discordance to reach the initial Holm cutoff, 173 can reach the unadjusted level, and for 1 pair no outcome could reject under any ordering. The unadjusted diagnostic also separates 99.9% of pairs at gaps of 0.05 or more, so the instrument is not

vacuous. A non-rejection therefore labels the remaining rows as unseparated under the declared rule at the discordance they carry, which is weaker than equality.

Observed gaps and diagnostic outcomes in this snapshot. Figure 2 groups the fixed-snapshot diagnostics by their observed resolve-rate gaps. Among the 474 pairs separated by less than 0.01, with a median of 82 discordant instances, exactly 0 are significant. Among pairs separated by 0.05 to 0.10, 99.9% pass the unadjusted test and 20.3% survive Holm. The same outcomes define both the gap bins and the diagnostics, so the display describes where the declared diagnostics reject in this snapshot rather than isolating pool size as the cause.

Among 3 same-model resubmission pairs, 2 separate under the exact test and 1 *regress*; these counts are too small to carry a claim.

For a direct task-iid rank baseline, we use the ordered one-sided tests with global Holm correction in Neuhof & Benjamini (2026b, Algorithm 2). With missing outcomes unresolved, the inclusive widths $W_j = U_j - L_j + 1$ of intervals $[L_j, U_j]$ have median/maximum 56/ 74 across 175 entries; pairwise deletion yields 122 but changes denominators. Appendix I.4 gives the endpoints and repository analysis (Table 10).

Across available leaderboard splits. Other splits with per-instance results differ: Lite has 1.3%, Multilingual 7.7%, and the larger 2294-instance test split 60.9% of adjacent pairs separable (Appendix D). Entries, task counts, and task mix change together, so the contrast is descriptive. Multimodal publishes only aggregates, precluding the paired recomputation used here.

2.3 Repository granularity can make rejection impossible

The task-level diagnostic treats discordant signs as independent. SWE-bench groups tasks by source repository, so a prospective design may instead randomize or sample repositories. Appendix I.1 defines a conditional repository-label sign-flip analysis for that design. Applied to an observational leaderboard, it is a resolution stress test because the required sign exchangeability is not established.

The stress test needs complete binary blocks. It therefore retains 44 of 175 entries, including only 4 of the published top 20. Its central result is analytic. A two-sided test with R repository signs draws its p-values from 2^R sign assignments, so $p_{\min} \geq 2^{1-R}$; zero-valued blocks only coarsen the grid. For a family of M hypotheses tested at level α, the condition $2^{1-R} > \alpha/M$ puts Holm's initial cutoff below every attainable p-value. No member of that family can then be rejected, regardless of the observed outcomes.

The complete Verified census has 44 entries, $M = 946$ pairwise hypotheses, and $R = 12$ repositories. Its exact floor, 0.000488, exceeds the initial cutoff 0.000053. Thus no comparison in the declared all-pairs family can be rejected on any data; the observed 0/946 count is design-implied rather than evidence for equality. Adding tasks inside the same repositories cannot refine the grid.

The obstruction depends jointly on the sign count and the declared family. Holding the same p-values fixed but narrowing the family to published-top-versus-all (43 hypotheses) rejects 15 comparisons. On the same complete entries, task-level rank intervals have median width 14, whereas repository-label intervals span all 44 positions. This contrast records the cost of treating repositories as the independent units; it does not estimate the leaderboard's repository dependence.

Appendix H compares exact sign flipping, CR1, and the wild-cluster bootstrap across frozen repository counts, size profiles, and heterogeneity. Appendix I gives the statistic, orbit distributions, Multilingual comparison, decision-family sensitivities, and the post-outcome aggregation correction; the full-census claims of the two subsections above do not depend on this analysis.

3 Promotion rules must match evaluation resolution

The same distinction applies inside a search loop: a candidate–parent score difference records one evaluation, while promotion treats that difference as evidence of improvement. A defensible promotion needs two guarantees at once, and they are different statistical problems: the rule must control false-promotion risk when the true effect is zero, and the design must have power at the smallest

effect worth promoting. This section treats the two in turn. Section 3.1 prices the power side of the test used in Section 2 — arithmetic once one quantity is measured, and the quantity is not the variance of a score but the rate at which two harnesses *disagree*. Section 3.2 audits the error side on comparisons whose true effect is zero by construction, and Section 3.3 shows how an operational threshold and a resolution audit lead to different readings of one frozen decision.

3.1 Paired discordance determines comparison cost

The paired quantity. Let $X_i(c), X_i(p) \in \{0, 1\}$ be the outcomes of a child and parent harness on task i of a shared pool, and define $\mu(h) = \mathbb{E}[X_i(h)]$ under the task distribution. Write $\rho_d = \Pr[X_i(c) \neq X_i(p)]$ for the discordance rate and $q = \Pr[X_i(c) = 1 \mid X_i(c) \neq X_i(p)]$ for the child's win probability among discordant tasks. Then

$$\mu(c) - \mu(p) = \rho_d\,(2q - 1), \qquad \text{so} \qquad \mu(c) \leq \mu(p) \iff q \leq \tfrac{1}{2} \quad (\rho_d > 0). \tag{1}$$

This is an identity, not an approximation, and it is what makes the hypothesis about means translate exactly into a hypothesis about a Bernoulli parameter. Concordant tasks contribute zero to the difference and are discarded without loss. The identity uses $X_i \in \{0, 1\}$: for graded scores the same construction tests $\Pr(X_i(c) > X_i(p))$ against $\Pr(X_i(p) > X_i(c))$, which is a statement about who wins more often and not about the difference in means. The two coincide only in the binary case, and SWE-bench resolution is binary.

Conditional resolution. A comparison contract C fixes the candidate–parent pair, target task distribution, outcome and missingness rules, direction, test ϕ_C, error level α, target power $1 - \beta$, and task budget n. For a declared discordance input ρ_d, define

$$R(C; \rho_d) = \inf\left\{\epsilon \in (0, \rho_d] : \Pr_C(\phi_C = 1 \mid \mu(c) - \mu(p) = \epsilon, \rho_d) \geq 1 - \beta\right\}. \tag{2}$$

If the set is empty, the contract has no finite resolution in the binary effect domain $(0, \rho_d]$. This R, indexed by pair, rule, budget, dependence model, and discordance input, is the resolution reported in the paper; missingness completion sets and repository-sign grids answer the separate identification and dependence questions of Table 1, and the certificate reports all three without renaming the latter two as R.

Error control. Index the discordant tasks in order of arrival and let W_m be the child's win count among the earliest m of them. Our gate promotes when the lower bound of a two-sided Robbins normal-mixture confidence sequence (Robbins, 1970; Howard et al., 2021) on W_m/m clears $1/2$. For a candidate–parent comparison and task arrival order fixed before the evaluation stream is observed, with discordant wins arriving as independent Bernoulli(q) observations, this gate promotes a non-improving candidate with probability at most δ at any stopping time (Proposition 1; statement and proof in Appendix C). This is a pre-fixed single-comparison result: clustered tasks, reuse of parent outcomes or a task pool across candidates, candidates or arrival orders selected from adaptive search history, and run-level multiplicity require separate designs; the same appendix bounds the paired gates' run-level rate by resampling an earlier audit.

What it costs. For a target improvement $\epsilon = \mu(c) - \mu(p) > 0$, Equation 1 inverts to $q = 1/2 + \epsilon/(2\rho_d)$. The target effect becomes an asymmetry inside the discordant set. We obtain the required task count from unconditional power at level 0.05 and target power 80%. Because the discordant-task count $D \sim \text{Binomial}(n, \rho_d)$ is random, the calculation sums the conditional rejection probability over the full distribution of D. The columns hold the effect estimand fixed and change the decision contract. Fixed-look inference is two-sided when either direction is allowed and one-sided when improvement is prespecified; monitoring uses a wider boundary to remain valid at every stopping time.

Table 3 illustrates the arithmetic with the initial repeat of the completed Qwen3.8-27B, 25-step candidate–incumbent comparison in Appendix K.1. That paired table has 50 discordant tasks among 200 evaluable pairs, giving the cross-arm value $\rho_d = 0.2500$. A repository-cluster bootstrap gives the model-based range $[0.20, 0.29]$ across 6 repositories. Because the same confirmation census supplies this input and the reported effect, the table is a retrospective illustration for this pair, not a prospective sizing of that experiment.

Table 3: Retrospective pair-specific design arithmetic at level 0.05 and target power 80%. Point requirements use the observed cross-arm $\rho_d = 0.2500$; ranges propagate the repository-bootstrap envelope $[0.20, 0.29]$. The dash marks a target not used for the leaderboard analysis.

	Two-sided fixed look	One-sided fixed look	One-sided monitored
n for $\epsilon = 0.10$ (point)	207	166	340
n for $\epsilon = 0.10$ (sensitivity)	171–240	136–193	270–409
n for $\epsilon = 0.05$ (point)	–	651	1571
Resolvable ϵ at $n = 500$ (sensitivity)	0.058–0.069	0.052–0.062	0.075–0.092
Minimum n at maximum expressible effect ($\epsilon = \rho_d$, $q = 1$)	31	26	58

The requirements inherit ρ_d, which is not a benchmark constant. Public-quantile pairs in this snapshot span $\rho_d = 0.140$ to 0.624, and Equation 1 makes the required effect at a fixed budget scale as $\sqrt{\rho_d}$, so that span would move the requirements by more than a factor of two. Those public pairs are leaderboard comparisons rather than candidate–parent pairs, so they bound the range of the input without supplying it, and the envelope above rests on 6 repositories alone.

At effect 0.10, the monitored rule needs 2.0 times as many tasks as the one-sided fixed-look rule under this model; this is the price of optional monitoring in the comparison shown here, and a betting e-process tuned in simulation closes none of it (Appendix B).

Inverting the calculation. The effect domain in Table 3 ends at $\epsilon = \rho_d$, since $q = 1$ is the largest possible discordant-task win probability (here 0.250). The final row's task count is the requirement at that maximum effect, not an effect-size floor. A fixed pool therefore has a pair-specific detectable effect. At $n = 500$, the one-sided fixed-look contract resolves roughly 5.2–6.2 percentage points under the worked pair's discordance envelope and the task-iid model. The largest resolve-rate gap between adjacent top-20 rows is 1.2 percentage points, several times smaller than this contract's resolution.

3.2 Best-of-seeds promotes constructed-null variants

The rules the gate replaces fail on inputs whose correct answer is known: we rendered 14 harness variants byte-identical to their base after parsing (machine-verified), so every observed difference is noise. Table 4 tallies what each rule does with that noise on the Qwen base-versus-variant comparisons, and what the same rules do on additive positive controls. Selecting the best of 3 seeds promotes every variant; the two paired tests promote at most 1 while still detecting controls, so their near-zeros are discrimination and not inertness. The comparisons share base outcomes, so these are correlated counterexamples, not error-rate estimates; Appendix J has the full grid and the DeepSeek replication. A model-conditional simulation of the same rule family from the audit base arm (Figure 8, Appendix J.1) puts accept-if-score-increases at 45.7% false promotion when the true effect is exactly zero.

Table 4: Acceptance rules applied to a constructed null: 14 base-versus-variant comparisons whose rendered prompts are byte-identical to base, 3 seeds per arm, nominal level 0.05 for the rules that take one. Every promotion is false by construction. Detections on additive positive-control arms separate a conservative rule from an inert one, but only for a rule that is not already promoting nearly everything: the top row's detections say nothing about its discrimination.

Acceptance rule	Null promotions	Control detections
Best of 3 seeds	14/14	2/3
Mean of 3 seeds	5/14	2/3
Exact McNemar	1/14	2/3
Anytime confidence sequence	0/14	1/3

3.3 AUDITING A FROZEN DECISION

Operational decision. We selected a prompt-only candidate on a discovery subset and froze the arms, protocol, and a disjoint 200-task evaluation census before collecting outcomes. The task IDs were disjoint, but 5 of the 6 evaluation repositories also appeared among the 9 discovery repositories; selection could therefore adapt to shared repository factors. Our rule promoted only if the task-weighted difference reached +0.080. The observed difference was +0.060, so we did not promote. The author-chosen threshold encoded the gain required to replace the incumbent. No external utility model justified its value, and it carried no inferential guarantee. We retain that decision and separately ask whether this census could reliably detect a threshold-sized effect.

Resolution at the threshold. Under the one-sided fixed-look task-iid model with observed discordance $\rho_d = 0.2500$, an exact test on 200 tasks has 69.5% power against a true effect of +0.080. Reaching 80% power requires 259 tasks. The observed effect fixes the operational verdict, but this power result limits its interpretation: the design falls short of the target power at the smallest effect worth promoting, so non-promotion does not show that the candidate lacks such an effect. More tasks would sharpen that distinction without changing the operational threshold.

Stability and scope. We froze two added repeats after observing the decision census, so their pooled analysis is descriptive. With 3 rollouts per arm and task, the repeat-averaged effect is +0.062, still below the threshold. Individual repeats fall on both sides of it. Repository-equal weighting raises the estimate to +0.147. Under the same frozen protocol on a second model, we measured +0.032 on 200 tasks with 3 repeats. The effects remain positive, but their magnitude depends on repetition, weighting, and model. On disjoint Test tasks, the frozen transport check had RMSE 0.232, above its prespecified 0.150 threshold. It therefore failed the gate for using the Verified inputs as a global sizing multiplier (Appendix E). Within-harness repeat disagreement is distinct from the cross-arm ρ_d in Equation 1 (Appendix G).

The certificate emits the fixed-census effect, missingness bounds, pair-specific iid resolution, and repository sign grid in one provenance-bound report (Appendix A). It records the analysis; the user must justify iid or sign-exchangeability assumptions. The result neither establishes efficacy beyond the frozen pair nor supplies a universal correction for repository dependence. Appendix K reports the full protocol and outcomes, including repeat-specific effects, repository sign grids, and leave-one-repository-out ranges, together with the process audit and frozen harnesses.

4 RELATED WORK

Self-improving agent harnesses search over edits and retain candidates according to scores on a task pool. AFlow (Zhang et al., 2025) is the reference construction for tree search over typed edit operators. Most recent systems still accept point-estimate improvements, using pass-count deltas, Pareto dominance, or parent-relative utility (Zhang et al., 2026; Chen et al., 2026; Wang et al., 2026). Among these systems, Bilevel-MCTS (Huang et al., 2026a) is closest to a variance-aware gate: it ranks recorded refinement outcomes using a lower-confidence-bound score, but specifies neither the comparison baseline nor an error-rate target for an individual refinement or the search run (scope in Appendix L). PACE (Shawn, 2026) uses a paired anytime-valid test, gives a per-candidate guarantee, and measures 72–100% false commits under greedy acceptance when no true gain is on offer. Proposition 1 gives the corresponding single-comparison guarantee for a Robbins-mixture construction. PACE controls a decision on a given pool without sizing the pool; AFlow uses score comparison without a statistical gate, although its held-out validation supports the reported final workflow. Evaluation resolution and attribution to individual promotions remain outside both results (scope in Appendix L).

Repeated evaluations make point-estimate acceptance hard to interpret. Evo-Bench (Huang et al., 2026b) finds a 2.2-point spread across byte-identical snapshots of one model, while several adjacent mid-table pairs are separated by less than a point.

Power analysis is established in NLP evaluation: Card et al. (2020) document widespread underpowering and release analysis notebooks, while Miller (2024) gives formulas for model comparison and prospective planning. We use this standard inversion. Other work motivates variance reporting

(Agarwal et al., 2021; Bouthillier et al., 2021; Madaan et al., 2024), while rank-uncertainty methods invert pairwise tests into rank sets (Ameli et al., 2025; Neuhof & Benjamini, 2026a;b). We implement Algorithm 2 of Neuhof & Benjamini (2026b) with ordered one-sided exact McNemar tests and global Holm correction, then repeat the inversion with repository-label p-values as a conditional sensitivity. This substitution does not inherit the original method's task-iid coverage. Because rank inversion assumes complete outcome matrices, we prepend worst/best-completion bounds from partial identification (Manski, 2003). For repository dependence, cluster-robust t procedures and wild-cluster bootstrap refinements address independent groups, heterogeneity, and unequal cluster sizes (Ibragimov & Müller, 2010; Cameron et al., 2008; MacKinnon & Webb, 2017); we compare their small-cluster behavior with exact sign-flipping rather than select a default correction.

BenchGuard (Tu et al., 2026) applies LLM auditors to benchmark specifications and evaluation scripts and reports author-confirmed defects in published suites. Our dated gold/empty check records a narrower failure of our local pipeline. Our contribution is the executable certificate in Section 3, which binds a pair-specific resolution estimate to its contract, identification audit, dependence sensitivity, analysis status, and provenance for one dated public snapshot and one internally frozen private comparison.

5 Discussion

The audit returns two verdicts about one dataset. Read as a record, the Verified leaderboard is nearly exact: its files settle the displayed ordering under the upstream scoring rule and every completion regime that respects its semantics. Read as a measurement, it is mostly open. Task-iid rank intervals span tens of positions, most adjacent rows remain unseparated under the declared contract, and the declared all-pairs repository-sign family cannot reject any comparison on the complete-case matrix at 12 signs. In the constructed-null audit, selecting the best of 3 seeds promotes 14/14 variants, whereas the two paired tests promote at most 1 (Section 3.2). Our frozen comparison shows the gap at decision scale (Section 3.3; its power figures are retrospective plug-ins, not guarantees).

Scope and limitations. The primary public result is a retrospective audit of one fixed snapshot. Missing-record bounds describe identification in that census; task-iid rank intervals and $R(C; \rho_d)$ require their stated sampling models. Repository identity supplies a grouping variable rather than a validated dependence model, so the cluster analyses remain sensitivities. All public evidence comes from one benchmark family: the four SWE-bench splits that publish per-instance outcomes. That family is auditable because it publishes those records with a documented scoring rule; the workflow applies to any leaderboard doing the same, but we have not verified any finding on another family, and nothing here estimates how common such disclosure is.

Minimum reporting contract. *Before collection.* Fix the contract fields of Table 1, the model, the target task distribution, the smallest worthwhile effect, the missing-outcome rule, and the multiplicity family. Estimate ρ_d on an independent pilot under those conditions, then size the task-level design. Report task and repository counts and justify the dependence model. Repository labels are randomization units only under exchangeable signs; otherwise, treat sign flipping as a sensitivity calibrated against plausible imbalance and heterogeneity.

After collection. Publish per-instance outcomes, the effect, ρ_d, the certificate, and a dated environment screen with its exclusions (ours, with its expiry, appears in Appendix F). For comparisons across budgets, report separately for each arm how often runs hit a step or cost limit and how often they encounter a context-window error. Adaptive search requires separate confirmation evidence and an error budget over decisions. The comparison-specific certificate records the pair, target tasks, rule, sampling unit, analysis status, provenance, and the resolution available to rank readers and search loops.

ARTIFICIAL INTELLIGENCE USE STATEMENT

An artificial intelligence (AI) coding assistant was used for experimental code and manuscript drafting; the authors are responsible for every claim. We checked its three uses separately. *Experiment code* was validated against closed-form and simulated references, benchmark per-instance labels, and constructed inputs with known answers (Appendix J). These tests caught an assertion that passed for every input and a task pool silently recomputed on resume, either of which could have produced plausible results. *Numbers*: no experimental value was typed by a human or model. A generator emits each LaTeX macro from a pinned artifact, and a build check rejects unmarked bare decimals in prose. *Prose* was drafted with assistance and revised by the authors. We resolved every citation against a publisher or preprint record, marked unverifiable works, and limited claims to material we could read.

ETHICS AND DATA STATEMENT

This study uses no human-subject data: it analyzes public leaderboard outcomes, public task identifiers, and binary outcomes from our own model runs. The anonymous supplementary bundle excludes task text, repository snapshots, model transcripts, generated patches, credentials, and endpoint addresses, and grants no new rights over SWE-bench or the repositories its task identifiers name. The main foreseeable harm is over-interpreting a diagnostic tied to one task pool, model, or budget as a population error rate or as evidence about a specific published system; we report the dependence structure and scope each claim to its measured conditions.

REPRODUCIBILITY STATEMENT

The analysis and figure generators read dated, pinned artifacts and fail when a required input is absent or inconsistent. The anonymous supplementary bundle supports offline verification of the four summaries reported in the abstract. It distributes their statistical inputs, replay outputs, analysis code, and a manifest of SHA-256 hashes for the distributed files. Its verifier recomputes those summaries without network access or private endpoints; private-endpoint comparisons support outcome-level statistical replay, not regeneration of model behavior. Appendix A.1 separates pre-outcome freezes from mechanically retrospective analyses.

Mapping. The paper contains no hand-entered experimental values: each is a macro in a generated file, the generator pins artifact paths rather than matching patterns (so a later run cannot silently replace a paper input), for those four summaries the bundle records the hashes of the distributed inputs and the expected summary values, and derived quantities are rounded once for display.

Conditions. Every rate depends on its step and cost budget; rates at different budgets are not comparable, so each is reported with its budget. The direct audits use Qwen3.8-27B on a self-hosted vLLM server and DeepSeek V4 Flash on a separate private endpoint, and the transfer analysis adds Claude Sonnet 5 through a private gateway; the bundle records request parameters and the Qwen server fingerprint but omits endpoint addresses, and frozen artifacts retain the server's raw identifier `Qwen38-27B` for the model this manuscript calls Qwen3.8-27B. The Qwen endpoint replayed deterministically at temperature zero; the DeepSeek endpoint did not under concurrent calls. The frozen repeat analyses use 3 complete rollouts per arm–task cell, aggregated within task; seed labels index repeated complete rollouts. The legacy fidelity screen lacks its collection commit and a complete dependency lock, so its artifact supports replay without environment-level reproduction.

Exclusions. Two kinds of records are excluded from every reported rate, both by name in the analysis code rather than by judgement. Runs that failed inside our pipeline are observations of the pipeline, not of task difficulty, and are re-run rather than scored; a completed run is never re-run because its outcome was inconvenient. Instances whose evaluation environment cannot discriminate any harness (their tests fail, or pass, before any patch is applied) would otherwise read as ordinary negative results and depress every condition equally, so pools are screened at no model cost, the screen is re-checked against direct observation because environments break as upstream dependencies move, and the admitted and rejected counts are recorded per pool.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. arXiv:2108.13264.

Siavash Ameli, Siyuan Zhuang, Ion Stoica, and Michael W. Mahoney. A statistical framework for ranking LLM-based chatbots. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2412.18407.

Ruicheng Ao, Hongyu Chen, Siyang Gao, Hanwei Li, and David Simchi-Levi. Best arm identification with LLM judges and limited human. *arXiv preprint arXiv:2601.21471*, 2026.

Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Naz Sepah, Edward Raff, Kanika Madan, Vikram Voleti, Samira Ebrahimi Kahou, Vincent Michalski, Dmitriy Serdyuk, Tal Arbel, Chris Pal, Gaël Varoquaux, and Pascal Vincent. Accounting for variance in machine learning benchmarks. In *Machine Learning and Systems (MLSys)*, 2021. arXiv:2103.03098.

A. Colin Cameron, Jonah B. Gelbach, and Douglas L. Miller. Bootstrap-based improvements for inference with clustered errors. *The Review of Economics and Statistics*, 90(3):414–427, 2008. doi: 10.1162/rest.90.3.414.

Dallas Card, Peter Henderson, Urvashi Khandelwal, Robin Jia, Kyle Mahowald, and Dan Jurafsky. With little power comes great responsibility. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9263–9274, 2020.

Mingju Chen, Can Lv, Guibin Zhang, Heng Chang, and Shiji Zhou. HarnessForge: Joint harness and policy evolution for adaptive agent systems. *arXiv preprint arXiv:2606.01779*, 2026.

Luis Da Costa, Álvaro Fialho, Marc Schoenauer, and Michèle Sebag. Adaptive operator selection with dynamic multi-armed bandits. In *Genetic and Evolutionary Computation Conference (GECCO)*, 2008.

Carol Hanna, Aymeric Blot, and Justyna Petke. Reinforcement learning for mutation operator selection in automated program repair. *arXiv preprint arXiv:2306.05792*, 2023.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979. JSTOR 4615733.

Steven R. Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Time-uniform, nonparametric, nonasymptotic confidence sequences. *The Annals of Statistics*, 49(2):1055–1080, 2021. doi: 10.1214/20-AOS1991.

Chia-Yu Hsu and Shubhanshu Shekhar. Efficient sequential evaluation of large language models. *arXiv preprint arXiv:2607.17409*, 2026.

Chenyi Huang, Haoting Zhang, Jingxu Xu, Zeyu Zheng, and Yunduan Lin. Bilevel optimization of agent skills via monte carlo tree search. *arXiv preprint arXiv:2604.15709*, 2026a.

Lisheng Huang, Chen Yang, Hao Zhou, Huatong Song, Zongchao Chen, Ran Le, Yang Song, Wayne Xin Zhao, and Tao Zhang. Evo-Bench: Can language models improve agent harness? *arXiv preprint arXiv:2608.09096*, 2026b.

Rustam Ibragimov and Ulrich K. Müller. t-statistic based correlation and heterogeneity robust inference. *Journal of Business & Economic Statistics*, 28(4):453–468, 2010. doi: 10.1198/jbes.2009.08046.

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 *International Conference on Learning Representations (ICLR)*, 2024. URL `https://openreview.net/forum?id=VTF8yNQM66`.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. ShinkaEvolve: Towards open-ended and sample-efficient program evolution. *arXiv preprint arXiv:2509.19349*, 2025.

Ke Li, Álvaro Fialho, Sam Kwong, and Qingfu Zhang. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18(1):114–130, 2014.

James G. MacKinnon and Matthew D. Webb. Wild bootstrap inference for wildly different cluster sizes. *Journal of Applied Econometrics*, 32(2):233–254, 2017. doi: 10.1002/jae.2508.

Lovish Madaan, Aaditya K. Singh, Rylan Schaeffer, Andrew Poulton, Sanmi Koyejo, Pontus Stenetorp, Sharan Narang, and Dieuwke Hupkes. Quantifying variance in evaluation benchmarks. *arXiv preprint arXiv:2406.10229*, 2024.

Charles F. Manski. *Partial Identification of Probability Distributions*. Springer Series in Statistics. Springer, New York, 2003. doi: 10.1007/b97478.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996.

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

Bitya Neuhof and Yuval Benjamini. Rank intervals for leaderboards: A hierarchical framework for model evaluation. *arXiv preprint arXiv:2606.08679*, 2026a.

Bitya Neuhof and Yuval Benjamini. Quantifying ranking uncertainty in LLM benchmarks. *arXiv preprint arXiv:2607.16259*, 2026b.

Zhe Ren, Yimeng Chen, Dandan Guo, Guowei Rong, Tonghui Li, R. B. Xiong, Qingfeng Lan, Wenyi Wang, Li Nanbo, Yibo Yang, Mingchen Zhuge, and Jürgen Schmidhuber. Self-improvements in modern agentic systems: A survey. *arXiv preprint arXiv:2607.13104*, 2026.

Herbert Robbins. Statistical methods related to the law of the iterated logarithm. *The Annals of Mathematical Statistics*, 41(5):1397–1409, 1970. doi: 10.1214/aoms/1177696786.

Zayx Shawn. PACE: Anytime-valid acceptance tests for self-evolving agents. *arXiv preprint arXiv:2606.08106*, 2026.

Xinming Tu, Tianze Wang, Yingzhou Lu, Kexin Huang, Yuanhao Qu, and Sara Mostafavi. BenchGuard: Who guards the benchmarks? automated auditing of LLM agent benchmarks. *arXiv preprint arXiv:2604.24955*, 2026.

Jean Ville. *Etude critique de la notion de collectif*. Gauthier-Villars, Paris, 1939.

Zefeng Wang, Minxi Yan, Jinhe Bi, Sikuan Yan, Volker Tresp, and Yunpu Ma. MetaSkill-Evolve: Recursive self-improvement of LLM agents via two-timescale meta-skill evolution. *arXiv preprint arXiv:2607.05297*, 2026.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves. *arXiv preprint arXiv:2606.09498*, 2026.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations (ICLR)*, 2025. Oral. arXiv:2410.10762.

Appendix contents.

A Certificate interface, failure behavior, and analysis ledger

B Anytime-versus-fixed power, in full

C Paired promotion error: statement, proof, and run-level error of per-candidate gates

D Resolvability across splits, and missing-value conventions

E Transport of the discordance input: cross-split stress test and a local held-out case

F Local evaluator fidelity caveat

G Within-harness repeat instability across models and budgets

H Repository-aware prospective sensitivity (cluster methods)

I Repository sign-flip inference in full: statistic, decision families, orbits, and fixed-geometry controls

J Correlated constructed-null stress test and score-rule simulation

K The worked comparison in full: protocol, provenance, and frozen harnesses

L Extended related work

A CERTIFICATE INTERFACE, FAILURE BEHAVIOR, AND ANALYSIS LEDGER

The supplementary bundle places the certificate script and its deterministic test at the bundle root. Its `README.md` gives the verification order and environment requirements. From that directory, invoke the certificate on a CSV containing one row per task:

```
python3 resolution_certificate.py --input PAIR.csv
```

The required columns are `task_id`, `repository`, `candidate`, and `incumbent`; either outcome may be empty, while observed outcomes must be binary. Optional arguments declare the error level, target power, task budget, and target effect. The output field tree is:

```
schema_version
fixed_census
  missing_as_unresolved
    effect, discordance, task_iid_exact
    task_iid_resolution, repository_signflip
  pairwise_observed
    effect, discordance, task_iid_exact
    task_iid_resolution, repository_signflip
  assumption_free_missingness_bounds
claim_contract
provenance
  input, input_sha256, git, created_at, deterministic
```

The CLI rejects absent columns, duplicate task identifiers, nonbinary outcomes, and invalid probability or budget arguments. A target effect larger than the observed discordance is retained as a labeled, non-expressible request rather than converted into a sample size. The bundled `test_resolution_certificate.py` applies the interface to a four-task, two-repository fixture and checks the missingness bounds, exact sign grid, target-effect design, non-expressible target, and duplicate-task failure.

A.1 ANALYSIS-STATUS AND REPRODUCTION LEDGER

The certificate cannot prove that researchers chose a contract before seeing outcomes. Confirmatory use therefore requires an append-only external record. Table 5 distinguishes that evidence from a retrospective hash: both detect later mutation, but only the former establishes ordering in time. The bundle contains the listed JSON records and verifies their complete hashes; the table shows compact prefixes for inspection.

Table 5: Status of analyses used in the main narrative. "Frozen" means before the named new outcomes, not a public preregistration.

Analysis	Status	Machine-checkable binding	Reproduction level
Public snapshot	Retrospective fixed-census audit	Upstream commit `40f164d5b8f1`	Full offline reanalysis
Worked-pair decision	Internally frozen before its confirmation outcomes; not publicly preregistered	Design `936d03805299`, protocol `d1046cde7239`	Outcome-level replay; private generation
Added repeats	Frozen after the original decision and before the added outcomes	Manifest `9dc6f45ad7d1`	Outcome-level replay; private generation
Cross-split stress test	Mechanically retrospective; public outcomes predate the design record	Design `8090cc5bef92`	Full offline reanalysis

B ANYTIME-VERSUS-FIXED POWER, IN FULL

Exact fixed-look inversion. For the one-sided fixed-look result quoted in the abstract, let $D \sim \mathrm{Binomial}(n, \rho_d)$ be the number of discordant tasks and $W \mid D = m \sim \mathrm{Binomial}(m, q_\epsilon)$ the candidate-win count, where $q_\epsilon = 1/2 + \epsilon/(2\rho_d)$. Define the exact critical count and unconditional power by

$$\begin{aligned} c_\alpha(m) &= \min\left\{c : \Pr\left[\mathrm{Binomial}(m, \tfrac{1}{2}) \geq c\right] < \alpha\right\}, \\ \pi_n^{\mathrm{fix}}(\epsilon, \rho_d) &= \sum_{m=0}^{n} \binom{n}{m} \rho_d^m (1-\rho_d)^{n-m} \Pr[\mathrm{Binomial}(m, q_\epsilon) \geq c_\alpha(m)]. \end{aligned} \tag{3}$$

If the set defining $c_\alpha(m)$ is empty, its conditional rejection probability is zero. The reported fixed-look resolution is the smallest evaluated $\epsilon \in (0, \rho_d]$ for which $\pi_n^{\mathrm{fix}}(\epsilon, \rho_d) \geq 1 - \beta$; the sensitivity endpoints substitute the corresponding ρ_d values from Table 3.

Whether the power gap between the fixed-sample McNemar test and our anytime-valid sequence could be closed by a tighter anytime construction is a question we tested and answer in the negative. A testing-by-betting e-process from the approximate Growth Rate Adapted to the Particular Alternative (aGRAPA) family was tuned over a 9-configuration parameter grid purely in simulation from the measured per-task solve distribution (2,000 trials per cell), with the real replay reserved for a single frozen configuration. At the small realized gap the tuned e-process reaches the same power tier as the Robbins mixture and closes none of the distance to the fixed test; at the large gap all rules saturate. The mechanism is structural: a fixed test spends its entire error budget at one prespecified sample size, while any anytime rule must spread that budget over every stopping time. Simulation also retro-validates both real replay outcomes (McNemar's detection of the smallest-gap arm was a 26.2% event under its simulated power; the sequence missing it, a 97.0% event), so neither replay outcome is surprising under the simulated power curves. The one-shot real replay of the frozen betting configuration was therefore not spent.

C PAIRED PROMOTION ERROR: STATEMENT AND PROOF

Proposition 1 (paired promotion error)**.** *Fix the candidate–parent comparison and the task arrival order before observing its evaluation stream; an externally randomized order is permitted, but the order may not depend on observed outcomes or search history. Assume that discordant task wins arrive independently as Bernoulli(q) observations. For $\delta \in (0, 1)$, let $B_m(\delta)$ be the radius of the two-sided Robbins normal-mixture confidence sequence (Robbins, 1970; Howard et al., 2021) with Bernoulli variance proxy $\sigma^2 = 1/4$ and tuning constant $\rho_{\mathrm{mix}} = 8$, both fixed before evaluation, so that $\Pr[\exists m : |W_m/m - q| > B_m(\delta)] \leq \delta$. If the gate promotes exactly when $W_m/m - B_m(\delta) > 1/2$, then at any stopping time $\Pr[$promote c and $\mu(c) \leq \mu(p)] \leq \delta$ for this comparison.*

Proof. By the paired identity, $\mu(c) \leq \mu(p)$ implies $q \leq 1/2$, and conditional on discordance the wins are i.i.d. Bernoulli(q). Promotion requires $W_m/m - B_m(\delta) > 1/2 \geq q$, hence $W_m/m - q > B_m(\delta)$, so the event $\{\text{promote}\} \cap \{\mu(c) \leq \mu(p)\}$ is contained in the event that the confidence sequence fails at some m. The count m is itself random, but the sequence is valid at all m simultaneously, which is why the optional-stopping argument goes through and why a fixed-sample bound would not suffice. The screening and futility rules only ever *reject*, so adding them shrinks the promotion event and the bound is unaffected. □

The implementation uses the two-sided Robbins normal-mixture boundary

$$B_m(\delta) = \frac{1}{m}\sqrt{2\sigma^2(m+\rho_{\mathrm{mix}})\log\left(\frac{\sqrt{(m+\rho_{\mathrm{mix}})/\rho_{\mathrm{mix}}}}{\delta}\right)}, \qquad \sigma^2 = \tfrac{1}{4}, \quad \rho_{\mathrm{mix}} = 8,$$

where the mixture tuning constant is fixed before evaluation. The construction is a nonnegative supermartingale combined with Ville's inequality (Ville, 1939). Equality at the promotion threshold does not promote: the implementation and proposition both use the strict crossing $W_m/m - B_m(\delta) > 1/2$.

C.1 Run-level error of per-candidate gates

The following run-level sensitivity comes from an auxiliary analysis of the earlier eight-arm audit: draw N comparisons with replacement from its 28 observed null outcomes per rule (20000 resampled runs, seeded) and count runs promoting at least one. The stated approximation is that a loop's candidates are exchangeable draws from these outcomes; the comparisons share arms (and one base), so these are approximations under that assumption, not independent measurements. For the two paired tests the observed promotion count is zero, so their run-level rate is capped, not estimated: the Wilson upper bound on the per-candidate rate gives at most 0.72 over ten candidates. This statement concerns the audit's denominator, not the gate. Both the point estimates and the Wilson bands per rule and per $N \in \{5, 10, 20\}$ are in the artifact.

D Resolvability across splits, and missing-value conventions

We apply the computation of Section 2 unchanged to every other split of the same leaderboard that publishes per-instance results (adjacent-rank pairs, unresolved convention):

split	entries	instances	unadjusted exact diagnostic
Verified	175	500	1.1%
Lite	76	300	1.3%
Multilingual	14	300	7.7%
Test	24	2294	60.9%

These rows differ in entry set, task count, and task mix. The larger Test split has more separated adjacent pairs, but this comparison does not identify pool size as the cause. Multimodal is absent because its entries publish aggregate counts and per-repository tallies without per-instance outcomes, so the paired computation cannot be reproduced from the public record.

D.1 Missing-value conventions

Fixed-census rank bounds. For a completion regime, let $\underline{X}_{ji}$ and $\overline{X}_{ji}$ be the lower and upper binary outcomes for entry j on task i, and define score bounds $\ell_j = n^{-1}\sum_i \underline{X}_{ji}$ and $u_j = n^{-1}\sum_i \overline{X}_{ji}$. The rank interval and its inclusive width are

$$\begin{aligned} L_j &= 1 + \sum_{k\neq j} \mathbf{1}\{\ell_k > u_j\}, \quad U_j = N - \sum_{k\neq j} \mathbf{1}\{\ell_j > u_k\}, \\ W_j &= U_j - L_j + 1. \end{aligned} \tag{4}$$

The strict inequalities leave tied positions compatible. The leaderboard-rule row of Table 2 fixes unreported outcomes at zero; the next row frees only genuinely open cells. The table summarizes W_j across entries.

The survey behind Section 2's missing-value paragraph was motivated by a rank-4 discrepancy against the public site that turned out to hinge on exactly this convention choice; the worked example's numbers — the swing and the rank movement it produces — are reported once in Section 2. Because we recompute every rate from the published per-instance record under a stated convention, individual positions in this paper may differ from the public site's rendering of the same data.

E TRANSPORT OF THE DISCORDANCE INPUT

E.1 METADATA-MATCHED CROSS-SPLIT STRESS TEST

Table 3 uses a pair-specific discordance input. We illustrate its limited transport on public data by changing the task set while holding entry identity and disclosed metadata fixed. Merely matching entry names was insufficient: the official Verified and Test metadata assign different attempt counts or system fields to 7 of the 14 overlap-consistent entries. The strict case therefore uses the remaining 7 entries and all 21 pairs. Model, agent, system, and attempt-count fields agree across splits; the metadata do not expose complete token, time, or monetary budgets. Selection is mechanical and does not use pair outcomes. The pair rows share entries, so the counts enumerate this fixed universe rather than independent replications.

The source is the 500-task Verified census; the target is the 1,794 Test tasks outside Verified. Under the missing-as-unresolved convention, median ρ_d changes from 0.224 to 0.090, with median absolute change 11.0 percentage points and source–target Spearman correlation 0.87 (Figure 3a). All 21/21 target values are lower. Deleting each entry in turn preserves the direction for every retained pair; the median absolute change ranges from 9.6 to 12.6 points.

At a common 500-task iid design point, 18 pairs have finite one-sided minimum detectable effects (MDEs; Figure 3b); all 18 change downward, from median 5.5 to 3.6 percentage points, with median absolute change 1.70 points. Pairwise deletion gives corresponding absolute changes of 11.5 and 1.80 points. The original same-entry universe has 14 entries and 91 pairs; despite its metadata mismatches, all 91 pairwise discordances move in the same direction (paragraph below). The strict subset therefore preserves rather than creates the directional result. These fixed censuses do not identify which feature of task composition caused the change.

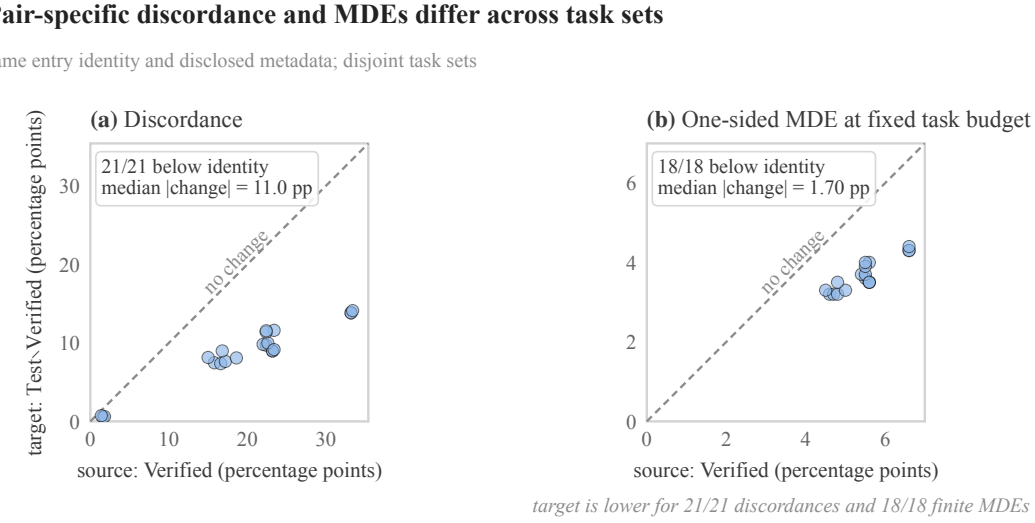

Figure 3: Descriptive cross-split stress test for metadata-matched entry pairs. Each dot compares Verified with disjoint Test tasks after requiring equality of the official model, agent, system, and attempt-count fields across splits. Panel (a) shows discordance; panel (b) recomputes the one-sided iid minimum detectable effect at level 0.05, power 80%, and $n = 500$. Pairs share entries and are not independent; metadata equality does not establish full compute-budget equality.

Full same-entry sensitivity. The parent transport universe contains every overlap-consistent entry appearing in both the Verified and Test manifests: 14 entries and 91 pairs. After that analysis, an audit of the upstream `metadata.yaml` files found that 7 entry identifiers did not preserve the disclosed attempt count or another system field across splits; the strict analysis above therefore uses only the metadata-matched subset. For completeness, the broader same-entry sensitivity gives lower target discordance for 91/91 pairs and a median absolute change of 13.1 percentage points. Among

pairs with finite standardized one-sided MDEs on both splits, 86/86 are lower on Test, with median absolute change 1.85 points. The pairwise-observed convention gives corresponding changes of 13.2 and 1.80 points. Because entry identity alone does not fix the disclosed evaluation protocol, these rows are a robustness check, not evidence that task composition caused the change.

E.2 Local held-out case

The frozen candidate screen contained 16 entries; 4 had complete outcomes on the 500-task Verified split. Before reading outcomes on the disjoint Test complement, the overlap audit excluded 1 because its recorded result disagreed across the two source snapshots. The remaining 3 entries define all 3 primary pairs. This complete-case subset is too small to estimate transport prevalence.

We evaluated fixed two-sided predictions on the 1794 Test tasks outside Verified. For each of 6 frozen target sizes, 200 repository-block subsamples preserved whole repositories. Figure 4 compares the fraction of rejecting subsets in this held-out finite population (not population power) with predictions frozen from Verified.

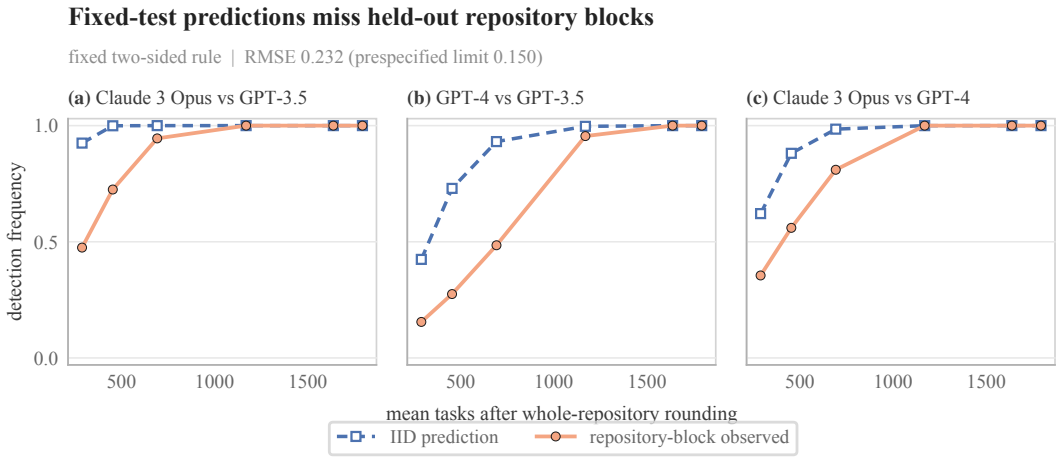

Figure 4: Local transport diagnostic across 18 frozen pair-by-size cells. Predictions condition on each pair's Verified effect and discordance. Observations use disjoint Test tasks sampled in whole-repository blocks; horizontal locations show the mean realized task count after block rounding.

The frozen root mean squared error (RMSE) is 0.232, above the 0.150 engineering threshold used to decide whether to retain a global sizing multiplier. This is not a field-standard calibration criterion. The result records a local counterexample to automatic reuse of the Verified inputs and supports no estimate of behavior across other systems.

F Local evaluator fidelity caveat

We ran a dated operational check on 200 SWE-bench Verified instances: the gold patch must resolve and the empty patch must not. 41 (20.5%) fail this check in our containerless evaluator; 37 failures arise because PASS_TO_PASS tests already fail at the base commit. Such tasks can enter a local aggregate as ordinary `resolved=False` records. Failures concentrate by repository (astropy 0/7, scikit-learn 0/6, matplotlib 6/25; every pydata task in the sample passes), so an iid interval is inappropriate. A repository bootstrap over 11 repositories gives the model-based range [5.6%, 47.8%], while leave-one-repository-out spans 12.6%–25.8%.

The screen ran on 2026-09-03 in per-task Python 3.8, 3.9, 3.10, and 3.11 environments. Its legacy artifact lacks the collection commit and a complete dependency lock. These observations describe our dated containerless pipeline, not the official harness, a benchmark defect rate, or every generated patch.

G WITHIN-HARNESS REPEAT INSTABILITY ACROSS MODELS AND BUDGETS

Define $\rho_{\text{rep}} = \Pr[X_i(h, s) \neq X_i(h, s')]$ for two complete rollouts of the same harness on task i. We estimate it in each collection as the mean over seed-pairs of per-task disagreement. Intervals resample repositories while preserving each task's complete seed vector. They are model-based cluster-bootstrap summaries, not exact coverage statements:

model	budget	ρ_{rep}	cluster interval	source
Qwen3.8-27B	25 steps	0.29	[0.25, 0.34]	audit base arm
Qwen3.8-27B	40 steps	0.22	[0.14, 0.28]	re-audit
DeepSeek V4 Flash	25 steps	0.13	[0.08, 0.18]	direct audit base arm
Claude Sonnet 5	25 steps	0.25	[0.17, 0.35]	transfer base arm
Claude Sonnet 5	40 steps	0.21	[0.12, 0.32]	transfer frozen arm
Claude Sonnet 5	50 steps, dollar-capped	0.29	[0.12, 0.42]	historical estimate

The 6 values span $[0.13, 0.29]$. No common constant is claimed: pools, budgets, and models differ, and the 50-step cell is a historical small-pool measurement retained with its saturation caveat. These values characterize run-to-run instability; substituting them for the candidate–parent ρ_d would change the estimand in Equation 1. The two-seed Claude cells contribute one seed-pair each, so seed-level uncertainty cannot be estimated from those cells. The artifact also reports naive pooled-cell Wilson intervals, repository leave-one-out ranges, and seed-pair leave-one-out ranges; only the cluster intervals appear in the table. Figure 5 reports a companion sensitivity on this pool: how the task-level exact test's null rejection moves as repository-shared signs replace independent ones.

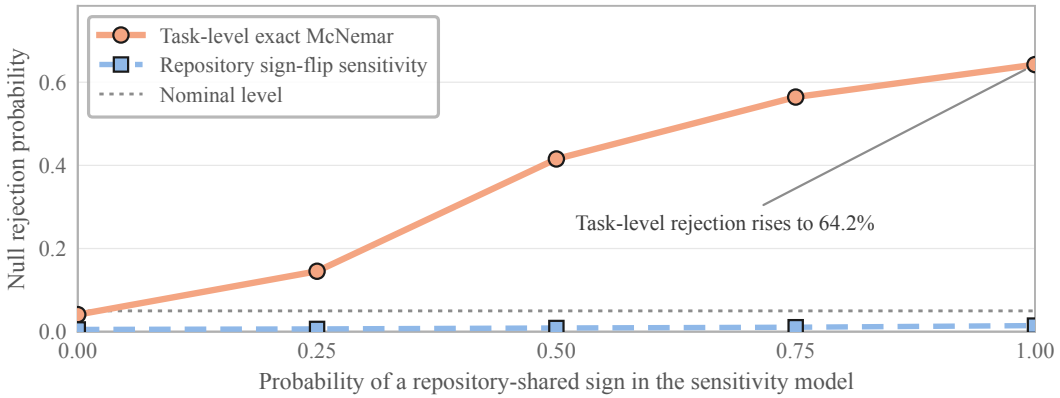

Figure 5: Sensitivity of null rejection to repository-shared signs. For each discordant task, the simulation uses one repository-shared Bernoulli sign with the probability on the horizontal axis and an independent sign otherwise; repository sizes match the measured pool. The task-level exact test is below the nominal level at the independent endpoint but becomes anti-conservative as sharing increases. The repository sign-flip curve is a sensitivity comparator, not a universal replacement: it additionally assumes independent, sign-symmetric repository scores. The horizontal line marks level 0.05.

H REPOSITORY-AWARE PROSPECTIVE SENSITIVITY

The frozen cluster-method benchmark fixes the marginal discordance at the value in Section 3 and samples a discordant count within each repository. Under iid task signs, every repository uses the same q. The symmetric model uses $q_r = q + hU_r$ with an independent Rademacher U_r. The asymmetric model uses a two-point, mean-zero, unit-variance repository effect whose probabilities and magnitudes are unequal. In the latter two models, $h = 0.15$. The null therefore has zero task-weighted mean in expectation in all three models, while only the symmetric model supplies the sign-flip theorem's invariance.

The frozen grid crosses repository count with balanced small repositories, balanced large repositories, and a rescaled Verified-like size profile. Each cell contains 2,000 simulated datasets. Task-level

exact McNemar conditions on the total discordance. The repository sign-flip statistic is Equation 5. CR1 uses the intercept-only sandwich variance with a t_{R-1} reference. The wild-cluster bootstrap multiplies restricted repository score contributions by frozen Rademacher draws and recomputes the CR1 studentizer in every draw. Exact sign enumeration is used when feasible; larger grids use frozen Monte Carlo signs with the plus-one correction.

The benchmark reports every one of its 72 cells in the supplementary CSV. Figure 6 shows the Verified-like slice. The comparison does not identify a universally valid test. It separates an exact result under sign symmetry from model-based evidence that the studentized wild bootstrap can remain calibrated when symmetry is relaxed, and it documents the poor small-cluster behavior of uncorrected CR1 in this design.

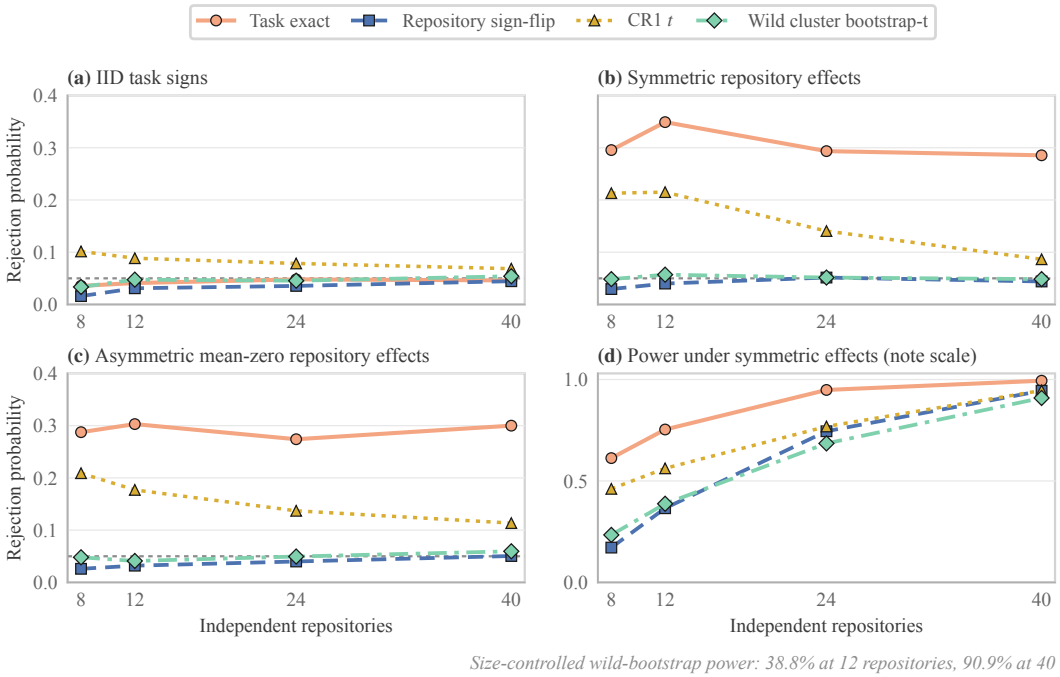

Figure 6: Frozen cluster-method benchmark at the Verified-like size profile. Panels (a)–(c) show null rejection on one shared scale; the horizontal rule is nominal level 0.05. Panel (d) shows power for effect +0.080 under symmetric repository effects, drawn on its own wider scale, so heights are not comparable to (a)–(c). Every panel uses 2,000 simulated datasets per point and the paired-case discordance. Task-exact and CR1 power is not interpretable where panels (b)–(c) show null inflation. The complete grid also varies tasks per repository and balanced sizes.

Public-profile extension. The separately frozen extension uses the same four implementations and 2,000 simulated datasets per cell. It replaces the single candidate–parent discordance and rescaled size profiles with 4 values selected by the frozen public-quantile rule and the observed repository sizes of the complete Verified and Multilingual snapshots. It is empirically anchored in those two inputs, not a fitted model of repository effects. Table 6 shows the symmetric slice; the supplementary artifact retains the iid and asymmetric slices and every Monte Carlo interval.

public size profile	ρ_d	task null	sign null	CR1 null	wild null	sign power	wild power
Verified ($R = 12$)	0.15	22.7	3.5	17.8	4.9	71.5	74.0
Verified ($R = 12$)	0.20	28.4	3.4	19.2	4.5	50.8	52.7
Verified ($R = 12$)	0.25	34.8	4.0	20.6	4.6	38.9	39.9
Verified ($R = 12$)	0.45	54.4	5.1	24.1	6.8	14.6	14.3
Multilingual ($R = 41$)	0.15	4.8	3.8	6.0	5.7	89.9	91.5
Multilingual ($R = 41$)	0.20	5.1	3.0	4.6	4.0	77.6	80.2
Multilingual ($R = 41$)	0.25	5.3	3.1	4.9	3.9	64.9	68.0
Multilingual ($R = 41$)	0.45	7.7	4.2	5.3	4.8	38.9	40.8

Table 6: Separately frozen public-profile robustness, in percent. Repository sizes are the observed complete-case profiles. Discordance values follow the frozen public-quantile rule. Null columns use symmetric repository effects; power uses $\epsilon = 0.08$ under the same model. Task and CR1 power are omitted because their null rejection is inflated in several cells. The artifact reports the asymmetric and iid slices, all Monte Carlo standard errors, and Wilson intervals.

I Repository sign-flip inference in full

I.1 Task-weighted compatible rank sets

For repository r with n_r evaluated tasks, let $S_r = \sum_i [X_{ri}(A) - X_{ri}(B)]$ and let $G_r \in \{-1, 1\}$ swap the complete A/B block. The task-weighted statistic and its conditional p-value are

$$T_w(G) = \left| \sum_{r=1}^{R} G_r S_r \right|, \qquad p_{\text{repo}} = 2^{-R} \sum_{g} \mathbf{1}\{T_w(g) \geq T_w(\mathbf{1})\}. \tag{5}$$

Tables 7 and 8 give the complete all-pair Holm inversion used in Section 2.3: a set contains positions not excluded by rejected task-weighted repository- randomization hypotheses. It is conditional on the fixed sign orbit and does not provide superpopulation rank coverage.

split	entries	pair rejections	rank-width range	singleton sets
Verified	44	0 / 946	44–44	0
Multilingual	12	14 / 66	1–11	1

Table 7: All-pair Holm inversion for task-weighted repository randomization tests; width counts compatible integer positions.

Multilingual entry	observed rank	compatible ranks
`gemini-3-flash`	1	[1, 10]
`claude-4-6-opus`	2	[1, 10]
`claude-4-5-opus`	3	[1, 11]
`glm-5`	4	[1, 10]
`gemini-3-pro`	5	[1, 11]
`kimi-k2-5`	6	[1, 11]
`claude-4-5-sonnet`	7	[1, 11]
`gpt-5-2-high`	8	[1, 11]
`gpt-5-2-codex`	9	[1, 11]
`claude-4-5-haiku`	10	[1, 11]
`deepseek-3-2`	11	[4, 11]
`gpt-5-mini`	12	[12, 12]

Table 8: Complete Multilingual compatible sets; common entry prefixes are omitted.

I.2 Decision-specific multiplicity sensitivity

Table 9 compares three families recorded in the dated design artifact before their results were read. Each reuses the task-weighted repository- randomization p-values from Section 2.3 and changes only the set passed to Holm's procedure. The pinned artifact retains, and independently recomputes, full membership, pairwise decisions, and compatible positions.

split	Holm family	hypotheses	rejections	median width	exact $p_{\min}$	initial cutoff	repositories needed
Verified	all complete pairs	946	0	44.0	4.883×10^{-4}	5.285×10^{-5}	16
Verified	published-order adjacent	43	0	44.0	4.883×10^{-4}	1.163×10^{-3}	11
Verified	published top vs. all	43	15	44.0	4.883×10^{-4}	1.163×10^{-3}	11
Multilingual	all complete pairs	66	14	11.0	9.095×10^{-13}	7.576×10^{-4}	12
Multilingual	published-order adjacent	11	1	12.0	9.095×10^{-13}	4.545×10^{-3}	9
Multilingual	published top vs. all	11	2	12.0	9.095×10^{-13}	4.545×10^{-3}	9

Table 9: Multiplicity sensitivity using unchanged task-weighted repository-randomization p-values. Median width follows inversion of the listed family. The exact p-value floor is theoretical; Multilingual uses frozen Monte Carlo draws. Focused families cover only their listed queries.

I.3 Complete conditional-orbit distributions

Figure 7 reports the complete conditional-orbit distributions behind the analysis of Section 2.3.

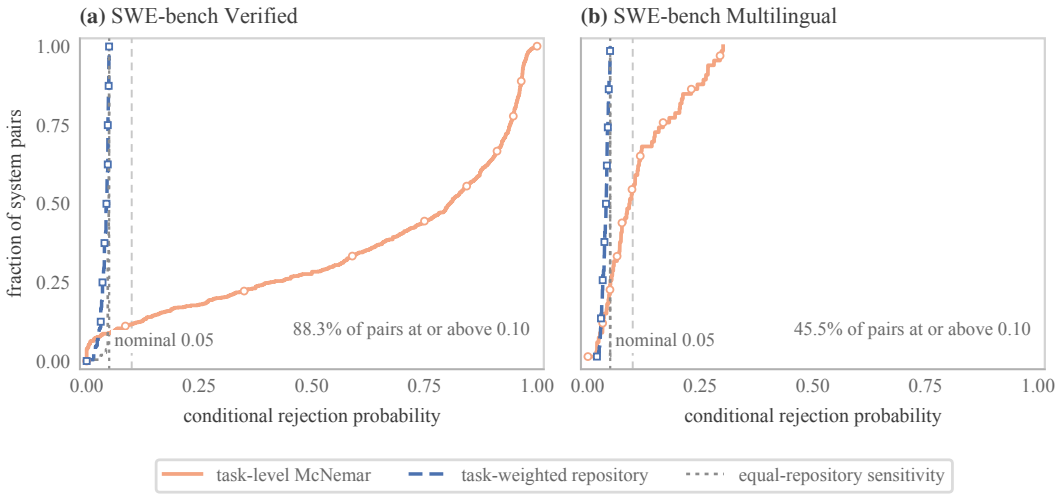

Figure 7: Conditional rejection probabilities over repository-level label assignments for every informative complete-system pair. The task-weighted repository statistic matches the leaderboard aggregate; equal-repository weighting is a sensitivity analysis. Panel (a) enumerates all 4,096 assignments; panel (b) uses 99,999 frozen Monte Carlo draws with the plus-one correction. Pair cells form fixed censuses and do not provide independent samples. The horizontal coordinate is not an unconditional benchmark error rate.

I.4 Literature rank baseline and fixed-geometry controls

Table 10 reports every frozen rank analysis. The task-exact rows implement the published ordered-comparison construction; the repository-sign rows hold the inversion fixed and change only the sampling unit. After global Holm correction, if b_j and w_j entries are found significantly better and worse than entry j, Algorithm 2 sets $L_j = 1 + b_j$ and $U_j = N - w_j$. Thus the inclusive width is $W_j = U_j - L_j + 1$; the reported median and maximum aggregate the N entrywise widths. Table 11 preserves the full repository-by-discordance layout of each selected public pair. The design crosses all supported effects with iid, symmetric, and asymmetric repository-effect generators. The table shows the common symmetric slice; the artifact contains all 105 cells and their Monte Carlo intervals.

universe	ordered comparison	entries	rejections	median width	width range	singletons
Verified, full	task exact; missing unresolved	175	10841	56	5–74	0
Verified, full	task exact; pairwise delete†	175	6504	122	5–164	0
Verified, complete	task exact	44	688	14	2–21	0
Verified, complete	repository sign	44	0	44	44–44	0
Multilingual, complete	task exact	12	16	10	1–11	1
Multilingual, complete	repository sign	12	14	11	1–11	1

Table 10: Simultaneous compatible-rank intervals from ordered one-sided comparisons and global Holm correction. Task-exact rows implement the literature baseline; repository-sign rows change the sampling unit and remain conditional sensitivities. †The pairwise-deletion row has pair-specific denominators and therefore does not target one shared leaderboard score. Width counts integer positions.

public pair geometry	ρ_d	task null	CR1 null	sign null	wild null	sign power	wild power	repositories
Verified q10	0.140	24.1	20.1	4.2	5.3	75.2	80.0	12
Verified q50	0.256	41.0	21.9	4.3	5.3	31.8	31.2	12
Verified q90	0.624	64.0	23.7	3.2	4.3	8.3	9.0	12
Multilingual q10	0.153	4.2	5.0	3.7	4.6	86.5	88.5	41
Multilingual q50	0.200	4.5	5.9	3.6	5.0	78.3	81.6	41
Multilingual q90	0.340	6.6	5.4	3.9	4.5	47.9	51.0	41

Table 11: Fixed-geometry controls (percent). Each row preserves one quantile-selected public pair's repository sizes and discordant-task counts; only discordant winner signs are regenerated. Null and power use symmetric repository effects, with power at the largest common supported effect, $\epsilon = 0.08$. Task and CR1 power are omitted because their null rejection is inflated in several cells. Every cell contains 2,000 simulated datasets; the artifact retains all effects, iid and asymmetric generators, intervals, and selected pair identities.

J CORRELATED CONSTRUCTED-NULL STRESS TEST

This appendix gives the construction and full grid for the secondary byte-identical-variant stress test. Its comparisons share base outcomes within model, so the promotion tallies are correlated counterexamples rather than estimates of a population error rate.

Construction. Table 12 lists the 14 variants. The manifest, checksums, and tier labels were frozen before any rollout was collected, and the analysis refuses to run against a manifest whose checksum has changed since collection. Every variant edits the YAML configuration in a way the parser discards: after rendering, the prompt sent to the model is byte-identical to base, verified at generation time, so its true effect is zero by construction.

Reading the grid. Every cell in Table 13 is promotions over comparisons, printed with its denominator because several cells have small ones: the DeepSeek result is a tally, not a rate estimate. The base-vs-variant columns mirror a search loop judging children against their parent and are primary. The all-pairs columns compare every pair of arms; because those comparisons reuse outcomes, they are correlated stress tests rather than additional independent samples.

Conditions. Both direct audits use a 25-step budget, 100 tasks, 3 seeds per arm, and nominal level 0.05 for every rule that takes one. Qwen contributes 4,500 rollouts and DeepSeek 1,200. Each arm is reduced to a strict majority binary outcome per task before either paired rule is applied. The per-task outcomes, frozen manifest, and replay code are retained as pinned project artifacts.

Table 12: The null variants. Every arm renders to a harness whose true effect against base is zero: all are byte-identical after rendering (machine-verified at generation time). Rates are over the full pool and all seeds; every difference from base is evaluation noise.

arm	edit	rate
`base`	unmodified harness	0.557
`t0_trailing_ws`	trailing spaces on top-level keys	0.557
`t0_yaml_comment`	a YAML comment added	0.547
`t0x_agent_scalar_order`	step_limit and cost_limit swapped inside the agent mapping; unordered.	0.547
`t0x_blank_before_model`	Blank line between top-level blocks; insignificant to YAML structure.	0.557
`t0x_comment_after_model_key`	Indented comment between model: and its first key; outside every scalar.	0.550
`t0x_comment_alt_text`	Leading YAML comment with different wording; comment content never parsed.	0.577
`t0x_comment_before_env`	YAML comment line between two top-level blocks; discarded by the parser.	0.547
`t0x_comment_before_model_kwargs`	Indented comment inside the model mapping, after a block scalar ends.	0.563
`t0x_comment_inside_agent`	Indented comment between two scalar keys inside the agent mapping.	0.540
`t0x_doc_start`	Explicit YAML document-start marker '—'; same single document.	0.560
`t0x_env_key_reversed`	environment.env mapping keys reversed; YAML mappings are unordered.	0.570
`t0x_key_order_env_first`	agent and environment blocks swapped; model stays last (EOF trap), and YAML mappings are unordered (a different permutation from t0_key_order).	0.560
`t0x_trailing_ws_env_key`	Trailing spaces on the nested env: key, outside every block scalar.	0.523
`t0x_trailing_ws_scalar_keys`	Trailing spaces after three plain-scalar values; stripped on parse.	0.550

Table 13: False promotions per acceptance rule on the null audit, nominal $\delta = 0.05$. Every comparison has a true effect of exactly zero, so every promotion is a false positive. Cells are promotions over comparisons. Base columns are the primary audits; all-pairs columns are correlated stress tests because arms share outcomes. Models are not pooled into an independence claim. DS denotes DeepSeek, SEM standard error, and CS confidence sequence. The last column counts detections on the positive-control arms, whose true effect is deliberately not zero (each is a purely additive injection of wrong or budget-wasting instruction into the base prompt, fixed as base-beats-arm before collection). It is reported here because a false-positive rate of zero is also what a rule that never promotes would score. Detection is conditional on the realised gaps, -0.083 and 0.083 and 0.340, and says nothing about smaller ones.

rule	Qwen base	DS base	Qwen all	DS all	detects
best of k seeds	14/14	1/3	101/105	2/6	2/3
mean of k seeds	5/14	1/3	48/105	2/6	2/3
mean + SEM margin	1/14	1/3	16/105	1/6	2/3
single run	1/14	0/3	39/105	1/6	2/3
exact McNemar (paired)	1/14	0/3	1/105	0/6	2/3
anytime-valid CS (paired, ours)	0/14	0/3	0/105	0/6	1/3

Positive-control arms: additive edits; direction pre-registered; gaps realised.

arm	edit	rate	gap vs base
`p_wrong_path`	asserts a repo path that does not exist	0.184	+0.340
`p_full_suite`	mandates the full test suite around every edit	0.441	+0.083
`p_step_ritual`	mandates a 4-command ritual before every edit	0.608	-0.083

J.1 Model-conditional score-rule simulation

Figure 8 reports both error modes.

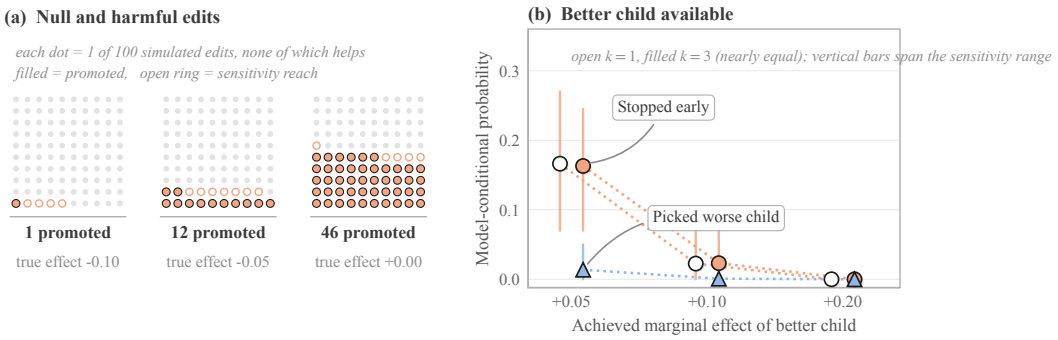

Figure 8: Model-conditional simulation from the audit base arm. Left: accept-if-score-increases promotion probability versus the achieved marginal effect after clipping, one dot per simulated edit. Right: stopping (coral circles) and wrong-selection (blue triangles) probabilities when a better child is available, open markers for branching factor $k = 1$ and filled for $k = 3$. Vertical bars span repository resampling with either within-task seed resampling or a Jeffreys-Beta probability draw. These are finite sensitivity envelopes, not confidence bands. Each point's raw additive shift is numerically calibrated so that the mean clipped per-task probability difference equals the reported horizontal coordinate; every sensitivity draw is recalibrated independently.

K THE WORKED COMPARISON IN FULL

K.1 PROTOCOL AND PROVENANCE

This appendix records the protocol behind the frozen comparison of Section 3.3, which applies the certificate workflow of Table 1 to one prompt comparison: it separates the operational decision from a repeat-averaged estimate, a task-iid sensitivity, and a repository-label sensitivity. Selection over the discovery edits limits the case to the chosen candidate rather than a general harness population.

We used a 100-task discovery subset of the SWE-bench Verified test split to search harness edits and select the prompt-only candidate, then froze the candidate and comparison protocol. The evaluation census contains 200 disjoint task IDs from the same split that passed the prespecified fidelity screen. Task disjointness is not cluster disjointness: 5 of the 6 evaluation repositories also occur among the 9 discovery repositories, so candidate selection can adapt to shared repository factors. The compound edit changes several workflow instructions, so the comparison cannot attribute an effect to one clause.

The prospectively specified Qwen decision used 1 complete rollout per arm–task cell and the author-chosen operational criterion stated in the body; all data-availability and infrastructure gates passed. After sealing that decision, we separately froze a Qwen extension with the same arms, tasks, evaluator, and budget, adding two repeats so each arm–task cell holds 3 complete rollouts (repeat one is the decision census itself), and transferred the frozen comparison to DeepSeek V4 Flash without further search. Because the added repeats were frozen after the decision census's outcome was known, their pooled statistics are descriptive; the extensions do not establish repository-disjoint confirmation or identify a mechanism. Appendix K.2 reports the complete outcomes, repository blocks, weighting sensitivities, process audit, and harnesses.

K.2 FROZEN HARNESSES, OUTCOMES, AND PROCESS AUDIT

Two weightings of the same pooled file. Both grids inherit the outcome-dependent continuation caveat stated in Section 3.3, and each estimate travels with the sign test of its own statistic. The task-weighted statistic, which matches the leaderboard aggregate, gives effect +0.062 with a one-sided repository sign grid of 3/64 $= 0.0469$. Repository-equal weighting gives +0.147 with 2/64 $= 0.0312$; that estimate is more than twice the task-weighted one because it is driven by two blocks of 5 and 3 tasks, while the largest block carries the weakest positive effect and another is negative

(-0.023). Leaving out each repository in turn keeps the task-weighted effect positive (+0.049 to +0.097). What this frozen file supports depends on the reading.

The listings below are read from the frozen confirmation artifacts during the paper build. They reproduce the complete YAML configurations used for both model-specific comparisons. The frozen supplementary bundle contains the constructed-null YAML variants, but not these two comparison files, so we retain their full text here. The task-facing workflow text described in Appendix K.1 is their only difference.

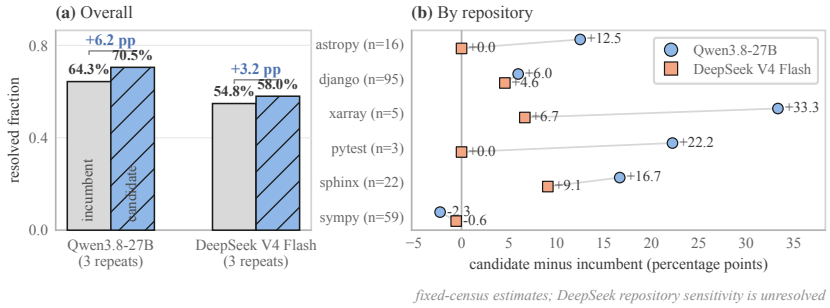

Figure 9: Frozen prompt comparison on the same 200-task census: task-averaged resolved fractions (a) and repository effects (b). Parentheses give task counts; lines aid comparison and do not pool models. The bars are descriptive fixed-census estimates, not iid interval estimates.

Repeat outcomes. All 200 Qwen tasks are evaluable. Across 1200 rollouts, the candidate resolves 423 of 600 arm–task–repeat slots (70.5%), compared with 386 (64.3%) for the incumbent (Figure 9). A one-sided task sign-flip test gives $p = 0.0093$, descriptive under the freeze-order caveat above. All 6 repository sums are nonzero, so the one-sided repository grid is $k/64$. Django and SymPy contribute 154 of 200 tasks (77.0%). The individual-repeat effects are +0.060, +0.020, and +0.105; both weighting readings with their sign grids and the leave-one-repository-out range are in the paragraph above.

All 200 DeepSeek tasks and 600 repeat pairs are evaluable. Across 1200 rollouts, the candidate resolves 348 of 600 arm–task–repeat slots (58.0%), compared with 329 (54.8%) for the incumbent. The repeat-specific differences are +0.055, +0.020, and +0.020. Only 4 repository sums are nonzero, leaving $2^4 = 16$ assignments. The observed numerator 2 gives $p = 0.125$, against the attainable floor 0.0625, and carries no verdict at level 0.05. The repeat-averaged effect is reported in Section 3.3; fixed-census leave-one-repository-out effects range from +0.019 to +0.047.

Descriptive process audit. The trace audit covers the original Qwen repeat and all DeepSeek repeats. In those records, the candidate uses 2.25 fewer calls and 26,114 fewer tokens per Qwen rollout; the DeepSeek reductions are 1.56 calls and 23,329 tokens. Submission rises by 14.5 and 14.7 percentage points. These measures do not identify a causal prompt clause. The retained response hashes do not include command trajectories.

K.2.1 INCUMBENT HARNESS

```
agent:
  system_template: 'You are a helpful assistant that can interact with a computer.

    Your response must contain exactly ONE bash code block with ONE command (or commands
    connected with && or ||).

    Include a THOUGHT section before your command where you explain your reasoning
    process.

    Format your response as shown in <format_example>.

    <format_example>

    Your reasoning and analysis here. Explain why you want to perform the action.

    ```mswea_bash_command

    your_command_here
```
```

```
    ```

    </format_example>

    Failure to follow these rules will cause your response to be rejected.

    ,
  instance_template: "Please solve this issue: {{task}}\n\nYou can execute bash commands\
    \ and edit files to implement the necessary changes.\n\n## Recommended Workflow\n\
    \nThis workflow should be done step-by-step so that you can iterate on your changes\
    \ and any possible problems.\n\n1. Analyze the codebase by finding and reading\
    \ relevant files\n2. Create a script to reproduce the issue\n3. Edit the source\
    \ code to resolve the issue\n4. Verify your fix works by running your script again\n\
    5. Test edge cases to ensure your fix is robust\n6. Submit your changes and finish\
    \ your work by issuing the following command: `echo COMPLETE_TASK_AND_SUBMIT_FINAL_OUTPUT`.\n\
    \   Do not combine it with any other command. <important>After this command, you\
    \ cannot continue working on this task.</important>\n\n## Important Rules\n\n\
    1. Every response must contain exactly one action\n2. The action must be enclosed\
    \ in triple backticks\n3. Directory or environment variable changes are not persistent.\
    \ Every action is executed in a new subshell.\n   However, you can prefix any\
    \ action with `MY_ENV_VAR=MY_VALUE cd /path/to/working/dir && ...` or write/load\
    \ environment variables from files\n\n<system_information>\n{{system}} {{release}}\
    \ {{version}} {{machine}}\n</system_information>\n\n## Formatting your response\n\
    \nHere is an example of a correct response:\n\n<example_response>\nTHOUGHT: I\
    \ need to understand the structure of the repository first. Let me check what\
    \ files are in the current directory to get a better understanding of the codebase.\n\
    \n```mswea_bash_command\nls -la\n```\n</example_response>\n\n## Useful command\
    \ examples\n\n### Create a new file:\n\n```mswea_bash_command\ncat <<'EOF' > newfile.py\n\
    import numpy as np\nhello = \"world\"\nprint(hello)\nEOF\n```\n\n### Edit files\
    \ with sed:\n\n{%- if system == \"Darwin\" -%}\n<important>\nYou are on MacOS.\
    \ For all the below examples, you need to use `sed -i ''` instead of `sed -i`.\n\
    </important>\n{%- endif -%}\n\n```mswea_bash_command\n# Replace all occurrences\n\
    sed -i 's/old_string/new_string/g' filename.py\n\n# Replace only first occurrence\n\
    sed -i 's/old_string/new_string/' filename.py\n\n# Replace first occurrence on\
    \ line 1\nsed -i '1s/old_string/new_string/' filename.py\n\n# Replace all occurrences\
    \ in lines 1-10\nsed -i '1,10s/old_string/new_string/g' filename.py\n```\n\n###\
    \ View file content:\n\n```mswea_bash_command\n# View specific lines with numbers\n\
    nl -ba filename.py | sed -n '10,20p'\n```\n\n### Any other command you want to\
    \ run\n\n```mswea_bash_command\nanything\n```"
  step_limit: 25
  mode: confirm
environment:
  env:
    PAGER: cat
    MANPAGER: cat
    LESS: -R
    PIP_PROGRESS_BAR: 'off'
    TQDM_DISABLE: '1'
model:
  observation_template: '{% if output.exception_info -%}

    <exception>{{output.exception_info}}</exception>

    {% endif -%}

    <returncode>{{output.returncode}}</returncode>

    {% if output.output | length < 10000 -%}

    <output>

    {{ output.output -}}

    </output>

    {%- else -%}

    <warning>

    The output of your last command was too long.

    Please try a different command that produces less output.

    If you''re looking at a file you can try use head, tail or sed to view a smaller
    number of lines selectively.

    If you''re using grep or find and it produced too much output, you can use a more
    selective search pattern.

    If you really need to see something from the full command''s output, you can redirect
    output to a file and then search in that file.

    </warning>

    {%- set elided_chars = output.output | length - 10000 -%}

    <output_head>

    {{ output.output[:5000] }}

    </output_head>

    <elided_chars>

    {{ elided_chars }} characters elided

    </elided_chars>

    <output_tail>

    {{ output.output[-5000:] }}

    </output_tail>

    {%- endif -%}

    ,
  model_kwargs:
    drop_params: true
  format_error_template: '{% if finish_reason is defined and finish_reason in ["length",
```
```

```
    "tool_calls"] -%}

    Your previous response reached the output token limit (finish_reason={{ finish_reason
    }}) before you produced a complete action, so it was cut off. Respond more concisely
    and provide exactly one action in the required format. If you need to think more,
    do so briefly.

    {%- else -%}

    Format error:


    <error>

    {{error}}

    </error>


    Here is general guidance on how to format your response:


    Please always provide EXACTLY ONE action in triple backticks, found {{actions|length}}
    actions.


    Please format your action in triple backticks as shown in <response_example>.


    <response_example>

    Here are some thoughts about why you want to perform the action.


    ```mswea_bash_command

    <action>

    ```

    </response_example>

    If you have completed your assignment, please consult the first message about
    how to

    submit your solution (you will not be able to continue working on this task after
    that).

    {%- endif %}'
```

K.2.2 CANDIDATE HARNESS

```
agent:
  system_template: 'You are a helpful assistant that can interact with a computer.


    Your response must contain exactly ONE bash code block with ONE command (or commands
    connected with && or ||).

    Include a THOUGHT section before your command where you explain your reasoning
    process.

    Format your response as shown in <format_example>.


    <format_example>

    Your reasoning and analysis here. Explain why you want to perform the action.


    ```mswea_bash_command

    your_command_here

    ```

    </format_example>

    Failure to follow these rules will cause your response to be rejected.

    '
  instance_template: "Please solve this issue: {{task}}\n\nYou can execute bash commands\
    \ and edit files to implement the necessary changes.\n\n## Recommended Workflow\n\
    \nYou have a limited number of steps, so work efficiently: combine related shell\
    \ commands with && whenever possible, avoid re-reading files you have already\
    \ inspected, and skip writing a separate reproduction script for simple, localized\
    \ fixes.\n\n1. Locate the relevant code with a single targeted search (grep/find),\
    \ reading only the files that matter.\n2. If the bug is non-trivial, write ONE\
    \ short script to reproduce it; otherwise skip straight to the fix.\n3. Edit the\
    \ source code to resolve the issue.\n4. Verify the fix with the smallest command\
    \ that proves it (rerun the repro script or the relevant test), combining verification\
    \ steps with && where possible.\n5. Once verified, immediately submit: `echo COMPLETE_TASK_AND_SUBMIT_FINAL_OUTPUT`.\n\
    \   Do not combine it with any other command. <important>After this command, you\
    \ cannot continue working on this task.</important>\n\n## Important Rules\n\n\
    1. Every response must contain exactly one action\n2. The action must be enclosed\
    \ in triple backticks\n3. Directory or environment variable changes are not persistent.\
    \ Every action is executed in a new subshell.\n   However, you can prefix any\
    \ action with `MY_ENV_VAR=MY_VALUE cd /path/to/working/dir && ...` or write/load\
    \ environment variables from files\n\n<system_information>\n{{system}} {{release}}\
    \ {{version}} {{machine}}\n</system_information>\n\n## Formatting your response\n\
    \nHere is an example of a correct response:\n\n<example_response>\nTHOUGHT: I\
    \ need to understand the structure of the repository first. Let me check what\
    \ files are in the current directory to get a better understanding of the codebase.\n\
    \n```mswea_bash_command\nls -la\n```\n</example_response>\n\n## Useful command\
    \ examples\n\n### Create a new file:\n\n```mswea_bash_command\ncat <<'EOF' > newfile.py\n\
    import numpy as np\nhello = \"world\"\nprint(hello)\nEOF\n```\n\n### Edit files\
    \ with sed:\n\n{%- if system == \"Darwin\" -%}\n<important>\nYou are on MacOS.\
```

```
    \ For all the below examples, you need to use `sed -i ''` instead of `sed -i`.\n\
    </important>\n{%- endif -%}\n\n```mswea_bash_command\n# Replace all occurrences\n\
    sed -i 's/old_string/new_string/g' filename.py\n\n# Replace only first occurrence\n\
    sed -i 's/old_string/new_string/' filename.py\n\n# Replace first occurrence on\
    \ line 1\nsed -i '1s/old_string/new_string/' filename.py\n\n# Replace all occurrences\
    \ in lines 1-10\nsed -i '1,10s/old_string/new_string/g' filename.py\n```\n\n###\
    \ View file content:\n\n```mswea_bash_command\n# View specific lines with numbers\n\
    nl -ba filename.py | sed -n '10,20p'\n```\n\n### Any other command you want to\
    \ run\n\n```mswea_bash_command\nanything\n```\n"
  step_limit: 25
  mode: confirm
environment:
  env:
    PAGER: cat
    MANPAGER: cat
    LESS: -R
    PIP_PROGRESS_BAR: 'off'
    TQDM_DISABLE: '1'
model:
  observation_template: '{% if output.exception_info -%}

    <exception>{{output.exception_info}}</exception>

    {% endif -%}

    <returncode>{{output.returncode}}</returncode>

    {% if output.output | length < 10000 -%}

    <output>

    {{ output.output -}}

    </output>

    {%- else -%}

    <warning>

    The output of your last command was too long.

    Please try a different command that produces less output.

    If you're looking at a file you can try use head, tail or sed to view a smaller
    number of lines selectively.

    If you're using grep or find and it produced too much output, you can use a more
    selective search pattern.

    If you really need to see something from the full command's output, you can redirect
    output to a file and then search in that file.

    </warning>

    {%- set elided_chars = output.output | length - 10000 -%}

    <output_head>

    {{ output.output[:5000] }}

    </output_head>

    <elided_chars>

    {{ elided_chars }} characters elided

    </elided_chars>

    <output_tail>

    {{ output.output[-5000:] }}

    </output_tail>

    {%- endif -%}

    '
  model_kwargs:
    drop_params: true
  format_error_template: '{% if finish_reason is defined and finish_reason in ["length",
    "tool_calls"] -%}

    Your previous response reached the output token limit (finish_reason={{ finish_reason
    }}) before you produced a complete action, so it was cut off. Respond more concisely
    and provide exactly one action in the required format. If you need to think more,
    do so briefly.

    {%- else -%}

    Format error:


    <error>

    {{error}}

    </error>


    Here is general guidance on how to format your response:


    Please always provide EXACTLY ONE action in triple backticks, found {{actions|length}}
    actions.


    Please format your action in triple backticks as shown in <response_example>.


    <response_example>
```

```
Here are some thoughts about why you want to perform the action.

'''mswea_bash_command
<action>
'''
</response_example>

If you have completed your assignment, please consult the first message about
how to
submit your solution (you will not be able to continue working on this task after
that).
{%- endif %}'
```

L EXTENDED RELATED WORK

Section 4 gives the comparisons needed for the main argument. This appendix records the distinctions that delimit those comparisons.

Acceptance tests. PACE (Shawn, 2026) gives a paired acceptance test for self-evolving agents: McNemar's discordant pairs against $q = 1/2$, made anytime-valid by a testing-by-betting e-process, with a per-candidate false-commit guarantee that survives optional stopping. Our gate uses the same discordant-pair estimand and supermartingale argument with a Robbins normal-mixture confidence sequence; the choice of construction and tuning constant is not a contribution. PACE reports that greedy acceptance commits 30–42% false and 10–33% harmful edits when a genuine improvement is available. With no true gain, it commits 13–21 spurious modifications per run and degrades its most fragile agent by 4.9 points.

PACE's guarantee is per candidate. Its attempted online false-discovery-rate correction across rounds was too conservative, and shared ancestors or reused parent samples further complicate a tree. Proposition 1 also stops at the single-comparison boundary. Related sequential methods study *model* capability (Hsu & Shekhar, 2026) or best-arm identification with LLM judges (Ao et al., 2026), rather than harness promotion.

Search over harnesses and workflows. AFlow (Zhang et al., 2025) performs Monte Carlo tree search over workflow graphs with typed operators. We study a harness around a fixed agent, so our audit cannot estimate how many AFlow promotions were noise. AFlow's held-out validation supports its final workflow; the missing gate instead limits attribution of internal branch comparisons and search cost.

HarnessForge (Chen et al., 2026) promotes joint harness and low-rank adaptation edits by deterministic Pareto dominance. MetaSkill-Evolve (Wang et al., 2026) accepts a directed-acyclic-graph edit on a parent-relative score increase, where one task can determine retention on its small validation split. Bilevel-MCTS (Huang et al., 2026a) is closer to a variance-aware gate: it ranks recorded refinement outcomes using a lower-confidence-bound score and related criteria. The paper does not define the comparison baseline, report the confidence level or critical value, or state how many variants enter each bound. It also gives no run-level correction or time-uniform claim. Our gate instead states an error-rate target for one comparison under optional stopping.

A recent survey (Ren et al., 2026) catalogs several acceptance mechanisms, including holistic scaffolding updates. It therefore does not support a universal claim about threshold acceptance or a workflow-only search space. The narrower common gap is that these systems state neither an error-rate target for promotions nor the effect size their pools can resolve.

Greedy harness evolution, and why its own numbers motivate us. Self-Harness (Zhang et al., 2026) defines in-split and held-out pass-count changes Δ_{in} and Δ_{ho}, then accepts when $\Delta_{\text{in}} \geq 0$, $\Delta_{\text{ho}} \geq 0$, and $\max(\Delta_{\text{in}}, \Delta_{\text{ho}}) > 0$, computed on pass *counts* over pools of 64, 100, and 180 tasks – Terminal-Bench, SWE-bench Verified, and AppWorld – with two repeats. The authors reduce operational noise by excluding unstable resources, mirroring slow ones, and capping bandwidth, but specify no noise model, confidence interval, or seed sweep. One task flip satisfies the rule at every pool size, and the held-out count uses only a subset of its pool. Some promotions are therefore

consistent with noise alone; identifying them would require per-instance outcomes. Self-Harness itself recommends stronger gates for higher-stakes changes.

Evo-Bench (Huang et al., 2026b) runs each model once, without seeds or error bars, and reports a 2.2-point Overall Score spread across byte-identical snapshots. Ranks 3–9 in its Table 2 span 43.5 to 35.9, with several sub-one-point gaps, so its reported precision does not resolve the mid-table ordering. Its ungated evolver can freeze the last restored revision rather than the reported peak; the paper proposes automatic best-revision recovery. We adopt its held-out protocol.

Adaptive operator selection. Evolutionary computation separates adaptive operator selection into credit assignment and selection (Da Costa et al., 2008; Li et al., 2014). ShinkaEvolve (Lange et al., 2025) similarly runs a bandit over an LLM ensemble, where an operator is a model or prompt. A study of 30,080 repair attempts over 353 Defects4J bugs found that learned selection produced more test-passing variants without fixing more bugs than uniform selection (Hanna et al., 2023). This negative result motivates a frozen-proposer control before crediting an operator ledger.

Statistical measurement of noisy benchmarks. The variance-reporting and rank-uncertainty work cited in Section 4 supplies the statistical ingredients. Our setting adds per-instance agent outcomes upstream and a harness acceptance decision downstream. The former requires a declared pairing, task order, dependence model, direction, and multiplicity family; the latter requires run-level control beyond a fixed comparison. Our all-pair Holm inversion gives simultaneous compatibility under a conditional label-randomization family, without superpopulation rank coverage. The adjacent-pair proportions remain descriptive, and Appendix J keeps the constructed-null stress test separate from the main claims.

BenchGuard (Tu et al., 2026) uses LLM auditors to find author-confirmed defects in published benchmark specifications and evaluation scripts. Our gold-patch/empty-patch check records one dated failure mode in a containerless pipeline. Its missing collection commit and dependency lock preclude a claim about the official benchmark.