# IS IT A REGRESSION, OR JUST NOISE? CALIBRATED BEHAVIORAL REGRESSION TESTING FOR STOCHASTIC LLM BEHAVIOR

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

An agent can fail a previously successful task even when its model and prompt are unchanged. Regression testing must separate this rerun variation from evidence that an update reduced reliability. We examine a repeated count test that compares each item's success counts with a one sided test and corrects across the suite for multiple comparisons. Our evaluation distinguishes two questions: unchanged version splits measure spurious alarms, while update runs measure agreement with labels marking large observed success rate drops. The test raised no observed alarms in unchanged version controls and selected many of the positive labels after updates. A structured function response study, repetition budget sweep and correction ablation show how the results depend on workload and sampling. Because the update labels share the tested runs, their agreement characterizes an empirical screen; independent reference runs remain necessary to measure detection of latent probability decreases.

## 1 INTRODUCTION

After a prompt or model update, a developer reruns a fixed set of tasks and finds that some previously successful cases now fail. These failures matter even when average accuracy improves: users may depend on the behaviors that were lost (Xie et al., 2021; Dixit et al., 2024). With stochastic language model agents, however, an unchanged version can also fail a task that it passed before. A regression test must therefore establish how much evidence a failed rerun provides about the update.

The missing information is the reliability of each behavior across runs. Repeating a task reveals whether success is usual or occasional, and whether the updated version succeeds less often. A second comparison holds the version fixed and treats two groups of its runs as if they came from different versions. Since this control contains no update, its alarms show how the decision rule responds to ordinary execution variation. Together, the comparisons separate a change in observed reliability from the variation that a rerun can produce on its own.

We study a simple implementation of this idea: a one sided test compares success counts for each item, and a procedure for correcting multiple comparisons is applied across items in a suite. These are established statistical tools. Our contribution is an empirical examination of their behavior across prompt edit and model swap settings in reasoning and code generation. The test produced no observed discoveries in unchanged version comparisons. Under updates, it selected many of the items labeled by a nominal drop threshold. This second quantity is agreement with labels computed from the same finite runs, so it measures consistency of the screening rule with the observed drops.

The evaluation also shows what the screen costs and where its interpretation changes. We compare count based testing with majority voting, examine a threshold estimated from unchanged runs, and trace the effect of the repetition budget and multiplicity correction. A separate function call generation study checks the same procedure on structured responses. The resulting picture combines low observed unchanged version alarm rates with a clear dependence on how many runs are collected and how reference drops are defined.



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.

## 2 Related Work

Regression measurement begins with the behaviors an update loses. Xie et al. (2021) use negative flips, from correct to incorrect, to measure NLP model regressions and study ways to reduce them. MUSCLE extends compatibility evaluation to generative tasks and trains an adapter to reduce instance regressions (Echterhoff et al., 2024). RETAIN supports migration decisions through interactive comparison and error discovery (Dixit et al., 2024). These approaches make lost behavior visible and actionable. Our question concerns the interpretation of a loss when the behavior itself varies across repeated executions.

Repeated evaluation addresses that ambiguity. Studies of LLM evaluation and code generation document variation across calls, including under greedy or zero temperature configurations (Song et al., 2025; Ouyang et al., 2025). Ma et al. (2024) identify nondeterminism as a reason to rethink regression testing for evolving LLM APIs. Closest to our setting, Xu & Wu (2026) use repeated calls, itemwise Fisher tests, false discovery control and a permutation null to study commercial model migrations. Their results show how aggregate gains can coexist with item level regressions. We use the same established statistical ingredients to examine unchanged version splits, alternative decision rules and repetition budgets in an open weight benchmark battery. This is complementary empirical evidence about calibration and sensitivity, rather than a new multiple testing procedure.

A statistical decision also depends on what the test considers correct. CheckList organizes behavioral tests around capabilities and expected transformations (Ribeiro et al., 2020); Testora uses the stated intent of a software change to distinguish intended from unintended behavioral differences in software (Pradel, 2026). AgentBench evaluates reasoning and decisions through interactions with environments (Liu et al., 2024). These works address test content, intended behavior and workload coverage. Here we hold automatic correctness checks fixed and study the decision made from repeated binary outcomes. Our workloads comprise answer generation, code generation and a single structured function response, which give a controlled starting point for this statistical question.

## 3 Testing Repeated Behavior

A behavioral regression makes a particular task less likely to succeed after an update. For version $V$ and item $x$, let $Y(V, x) \in \{0, 1\}$ be the automatic check's outcome, with 1 denoting success. Its success probability is $p_V(x) = \mathbb{E}[Y(V, x)]$. Updating from $A$ to $B$ gives the latent regression set $R_0 = \{x : p_A(x) > p_B(x)\}$. Repeated runs provide evidence about these unknown probabilities; one run supplies only one binary observation.

For a comparison setting $c$, the suite $X_c = \{x_{c1}, \ldots, x_{cm_c}\}$ and its correctness checks stay fixed. The version specifies the model and prompt; decoding and execution settings are held constant across the comparison. The statistical model treats repeated outcomes for one item and version as independent Bernoulli draws with a stable probability. Under this model, we test $H_{0,ci} : p_A(x_{ci}) \leq p_B(x_{ci})$ against the directional alternative $H_{1,ci} : p_A(x_{ci}) > p_B(x_{ci})$. Figure 1 connects this decision rule to its two empirical checks.

### 3.1 Evidence from success counts

Fisher's test asks whether successes are unusually concentrated in the reference version. Run each version $K$ times on an item and count its successes, $S_A = \sum_{j=1}^{K} Y_{Aj}(x)$ and $S_B = \sum_{j=1}^{K} Y_{Bj}(x)$. The corresponding failure counts are $K - S_A$ and $K - S_B$. If both versions have the same success probability, the total $T = S_A + S_B$ can be allocated between the two groups without favoring either version. The directional tail adds the probabilities of allocations with at least the observed $S_A$ successes in the reference group:

$$P_{\mathrm{F}}(x) = \sum_{u=S_A}^{\min(K,T)} \frac{\binom{K}{u}\binom{K}{T-u}}{\binom{2K}{T}}.$$

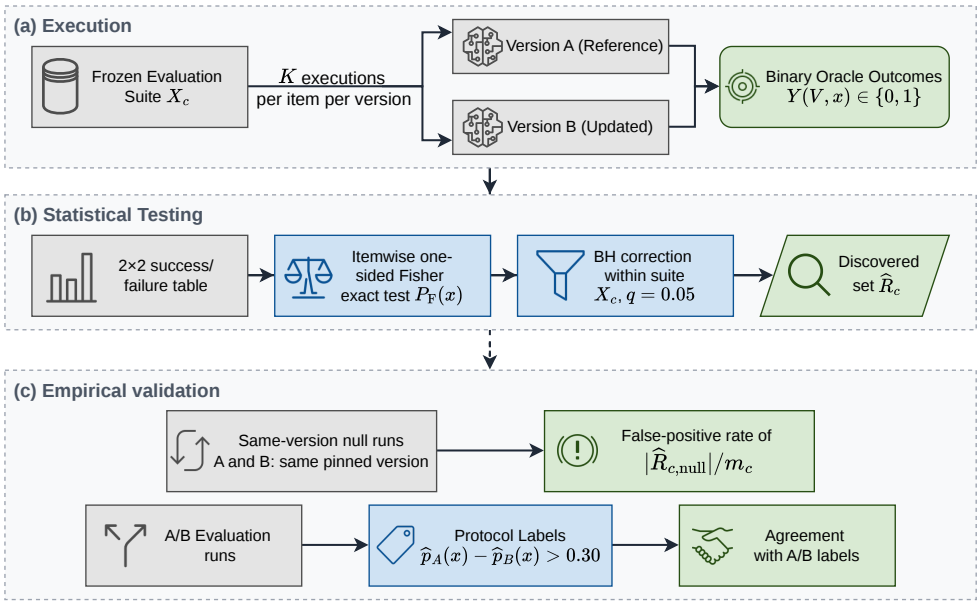


Figure 1: Repeated-count regression testing. Each version produces binary outcomes on a fixed suite; itemwise Fisher tests and BH yield a discovered set. Unchanged-version runs measure spurious alarms, while A/B runs supply observed-drop labels for agreement. In the main battery, the unchanged-version split has 10 runs per side and the update comparison has 20 per side.

A small $P_F(x)$ therefore indicates that the observed concentration of successes in $A$ is hard to explain by equal probabilities. Including the observed allocation in the tail makes the test conservative for these discrete counts. The direction matters: improved success under $B$ does not produce evidence for a regression.

### 3.2 Decisions across a suite

A suite contains many opportunities to flag an item by chance. BH limits the expected fraction of false discoveries among the selected items, using the ordered evidence across the suite (Benjamini & Hochberg, 1995). Sort its $p$ values as $P_{(1)} \leq \cdots \leq P_{(m_c)}$, with corresponding items $x_{c(i)}$. At the target false discovery rate $q = 0.05$, find the largest rank that passes its threshold, then include the entire prefix:

$$\widehat{R}_c = \{x_{c(1)}, \ldots, x_{c(k^\star)}\}, \qquad k^\star = \max\left\{i : P_{(i)} \leq \frac{iq}{m_c}\right\}.$$

If no rank passes, the discovered set is empty. The FDR guarantee requires valid itemwise $p$ values and independence or the positive regression dependence condition for which BH is valid (Benjamini & Yekutieli, 2001). Each setting is a separate family; settings and function call comparisons are never pooled. This guarantee concerns the expected fraction of false discoveries. The empirical unchanged version false positive rate instead counts flagged item comparisons, and directly describes the runs used in the control.

For example, suppose $A$ succeeds on 17 of 20 runs and $B$ on 10 of 20. The observed drop is 0.35, or 35 percentage points, and the one-sided Fisher $p$ value is about 0.020. That is evidence about this item, but it is not by itself a discovery in a 200-item suite: the BH decision also depends on the other items' $p$ values. Repetition supplies evidence; the suite correction determines which evidence is strong enough to report together.

# 4 EXPERIMENTS

## 4.1 TASKS, COMPARISONS AND MEASUREMENTS

Each setting starts with a fixed set of tasks, two specified versions and 20 binary outcomes per item from each version. We make two comparisons from these outcomes. For the unchanged-version control, the first 10 reference runs are compared with the remaining 10; any discovery is a false alarm because the version is identical. For the update comparison, all 20 reference runs are compared with all 20 updated runs. The latter asks which observed drops the rule selects. The control and update thus use 20 and 40 executions per item, respectively; the control describes a smaller per-side sample size than the update.

The main battery covers five datasets and four models in three families (Table 1). A mathematical answer, multiple-choice answer or generated program is scored by a fixed automatic check. Fifteen settings change the task prompt from its reference form to a terse form while holding the model fixed; nine replace the reference model with Qwen2.5-Instruct 1.5B while retaining the reference prompt. The selected items, scoring rules and sampling configuration stay fixed. Generation uses temperature 0.7 and top-$p$ 0.95. Twenty-two settings contain 200 items and the two HumanEval settings contain 164, giving $22 \times 200 + 2 \times 164 = 4{,}728$ item–setting comparisons in the control. Reused items count once in each setting in which they appear. The pooled zero count uses the item–setting comparison as its reporting unit, whereas means and bootstrap intervals use settings as the unit; reused tasks and reference outcomes can therefore recur across settings without being treated as distinct tasks.

Table 1: Dataset coverage and correctness checks.

| Tasks | Models / changes | Check / size |
|---|---|---|
| GSM8K; MMLU; ARC-Challenge | Qwen2.5-Instruct 1.5B/7B, Phi-3.5-mini-instruct, Mistral-7B-Instruct; prompt and model changes | Answer or choice match; 200 per setting |
| HumanEval; MBPP | Qwen; prompt edits | Unit tests; 164 / 200 items |
| Glaive function calls | Qwen; one prompt edit, one model swap | Function name and arguments; 200 each |

The experiment implementation follows the saved analysis. The generation harness reads a fixed dataset order, shuffling MMLU and ARC rows with seed 0 before selecting the requested prefix. GSM8K, HumanEval and MBPP use the selected prefix of their stored data. Prompt edits replace reasoning-oriented math and multiple-choice instructions with answer-only instructions; code prompts change the programmer instruction and request terser output. Numerical scoring extracts the final matching number when the answer separator is absent; multiple-choice scoring uses the answer marker or the final standalone letter from A to H. Code is scored by executing the task's tests. These scoring rules remain identical on both sides of an update. The analysis uses seed 20260624 for sampled ordered pairs and seed 7 for run subsets. The comparison manifest binds each comparison to its reference and updated correctness matrices, with rows aligned by the selected task order. The A/B label is implemented as floating-point `(pA - pB) > 0.30`; it does not explicitly enforce an exact rational tie rule, so $18/20 - 12/20$ evaluates slightly above 0.30 in binary floating arithmetic. The reported labels and results retain that executed comparison.

For the update comparison we need an observable reference against which to describe the discovered set. Let $\widehat{p}_V(x) = K^{-1} \sum_{j=1}^{K} Y_{Vj}(x)$ be an item's observed success rate. Its A/B label is positive when $\widehat{p}_A(x) - \widehat{p}_B(x) > \delta$, with $\delta = 0.30$, using the floating-point comparison described above. *Agreement* is the fraction of these positive labels selected by a method, calculated within each setting and then averaged over settings. A setting with no positive labels has undefined agreement. The labels and tests use the same outcomes, so this measure describes selection of large observed drops. The margin defines the labels, while Fisher/BH tests for any directional probability decrease.

## 4.2 COMPARISON RULES AND SUMMARIES

Majority voting provides the direct comparison for how repeated outcomes are used. It reduces each version's runs to a pass/fail decision and flags the item when at least half of $A$'s runs succeed

and fewer than half of $B$'s succeed. It uses the same 20+20 update pool and 10+10 unchanged-version split as Fisher/BH, but it discards the magnitude of the count difference and applies no suite correction.

The negative-flip baseline starts from the familiar event that $A$ passes and $B$ fails (Xie et al., 2021). The stored analysis summarizes that event differently on the two comparisons. Its unchanged-version rate averages the event over 200 sampled ordered pairs of distinct reference-run indices and then over items. Its update flag is positive when at least half of the corresponding cross-version pairs show a negative flip, using the full 20+20 pool. Thus two executions define an individual flip, whereas the reported update agreement uses a pool-based frequency rule. The paired summaries should be read with this distinction in mind.

The empirical null-quantile baseline estimates one common drop threshold per setting. For every item, subtract the mean of the first reference half from the mean of the second half. Take the $q$ quantile of this vector *across items*, and flag an update when $\widehat{p}_B - \widehat{p}_A$ is at or below that threshold. Its default quantile is 0.05. This shares unchanged-version information across items, whereas Fisher compares each item's own counts. Ties in the binary-rate differences can make the observed flagged fraction exceed the chosen quantile.

Alongside default rules, we retain a retrospective matched summary. For Fisher/BH and the null-quantile baseline, the analysis sweeps $\{0.01, 0.02, 0.05, 0.10, 0.20, 0.30\}$ and reports the largest update agreement whose observed unchanged-version rate is at most 0.05. Selection therefore uses both control rates and update labels. A fixed rule contributes its agreement if its control rate meets the cap, and zero otherwise; the per-setting table marks the latter cases N/A. Bootstrap intervals resample settings. These summaries describe the recorded settings and threshold choices; a prospective operating point would require separate calibration data.

The main execution and reporting conventions are summarized in Table 2.

Table 2: Execution and reporting conventions.

| Convention | Definition | Unit |
|---|---|---|
| Unchanged-version control | First 10 versus last 10 of 20 reference runs. | Item / suite |
| Negative-flip null | 200 sampled pairs of distinct reference-run indices, averaged over items. | Item |
| Negative-flip update | At least half of pairs are reference success and updated failure. | Item |
| Calibrated update | Fisher on 20 runs per version; BH across the suite. | Item / suite |
| A/B label | $\widehat{p}_A - \widehat{p}_B > 0.30$. | Item |
| Multiplicity family | BH at $q = 0.05$ within a suite. | Setting |
| Confidence interval | Bootstrap over settings. | Setting |
| Budget sweep | One subset per setting at $K \in \{4, 8, 12, 16, 20\}$, seed 7; labels recomputed. | Setting |

### 4.3 Unchanged-version behavior

Before examining alarms, we measure how much behavior can vary without an update. An item is variable if its $K$ runs contain both success and failure. The fraction of such items, called the noise floor here, is

$$F_c = \frac{1}{m_c} \sum_{i=1}^{m_c} \mathbb{1}\{\exists j, j' \in \{1, \ldots, K\} : Y_j(V, x_{ci}) \neq Y_{j'}(V, x_{ci})\}\,.$$

For independent runs with success probability $p_x$, one ordered pair has negative-flip probability $p_x(1 - p_x)$. Repetition exposes more items with both outcomes as $K$ grows. The noise floor therefore describes the opportunity for a single pair to look worse, rather than a minimum attainable testing error.

Across the 24 settings, the noise floor ranged from 0.14 to 0.81. Pairwise negative-flip rates were larger in more variable settings, with Pearson $r = 0.994$ and a setting-bootstrap interval of

$[0.985, 0.999]$ (Figure 2). The mean rate was 0.064 and the maximum was about 14%. Both axes summarize the same outcomes, so the association is a descriptive relationship within the battery.

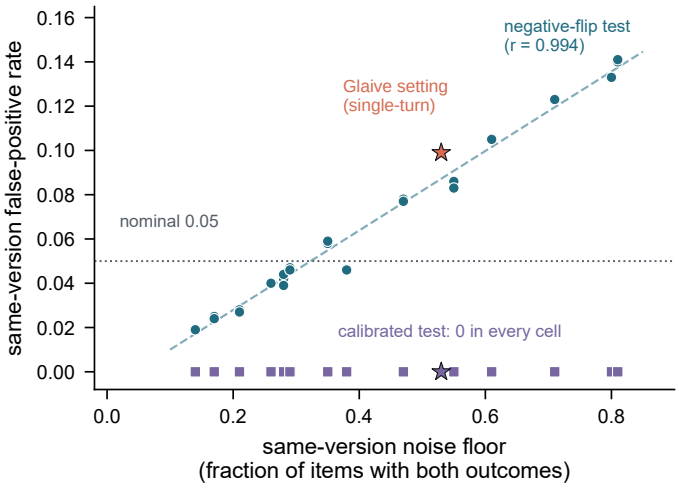


Figure 2: Unchanged-version variation and alarms. Circles show pairwise negative-flip rates in the 24 benchmark settings; squares show Fisher/BH, which flagged no items. The star is the separate Qwen Glaive check. The dotted 0.05 line is a reference level. The correlation summarizes the benchmark settings only.

Fisher/BH flagged 0 of 4,728 item comparisons, with no discoveries in any of the 24 unchanged-version suites. Majority voting had a mean flagged fraction of 0.022, and the default null-quantile rule had 0.060 (Table 4). These counts establish the observed behavior of the 10+10 control. For comparison with the observed zero count, an independent-Bernoulli model gives the one-sided 95% binomial upper bound

$$1 - 0.05^{1/4,728} \approx 0.00063.$$

This calculation indicates the finite-count scale associated with observing zero events under that model. Repeated tasks, shared reference runs and suite-level decisions mean that the comparisons need not be independent, so the value is not a dependence-adjusted population bound for this battery. The empirical statement remains the observed zero count.

At the suite level, the same issue appears in a separate GSM8K pilot with Qwen2.5-Instruct 1.5B, 300 items and 20 runs, rather than the 24-setting battery. Its noise floor was 0.793, mean per-item correctness variance was 0.126, and pairwise disagreement probability was 0.265. The negative-flip rate was 0.131. To ask whether averaging across tasks removes rerun variation, the pilot took one run from each side and computed the suite's mean success difference:

$$D_c = \frac{1}{m_c} \sum_{i=1}^{m_c} [Y(A, x_{ci}) - Y(B, x_{ci})] .$$

A suite alarm occurs when $D_c > \tau$. With the version unchanged, the alarm rate was 0.465 (about 0.47) at $\tau = 0$, 0.365 at $\tau = 0.01$, and 0.260 at $\tau = 0.02$. Fisher/BH's unchanged-version item rate was zero. For the pilot prompt edit, average success changed from 0.574 to 0.081 and agreement with its observed-drop labels was 1.000. The pilot illustrates why a zero-tolerance aggregate difference can alarm frequently even when no update occurred.

The pilot also separates item-level calibration from aggregate monitoring. The item-level Fisher/BH rate remains zero in the unchanged control, while the suite-level alarm rate changes with the tolerance $\tau$. This difference is expected: the aggregate rule averages one run from each side before applying a threshold, whereas Fisher/BH retains item-level counts and corrects within a suite. A small aggregate difference can therefore trigger an alarm even when no individual item meets the calibrated discovery rule. Conversely, an item-level drop can be diluted in a suite average. The pilot is therefore a check on the reporting unit, not a second estimate of Fisher/BH's item-level false-positive rate.

### 4.4 AGREEMENT UNDER PROMPT AND MODEL CHANGES

At its default $q = 0.05$, Fisher/BH selected 85.7% of the positive A/B labels, compared with 79.1% for majority voting using the same update-run budget. The stored negative-flip frequency rule selected 72.0%. The comparison with voting shows the benefit, in these settings, of using the count difference and suite correction together instead of reducing each version to a majority outcome. All three agreement values refer to the large observed drops defined above.

Table 3: All 24 benchmark settings. P changes the prompt; S changes the listed reference model to Qwen2.5-Instruct 1.5B. Noise floor is the fraction of items with both outcomes in 20 reference runs. Matched agreement is the stored retrospective summary at an observed same-version rate no greater than 0.05. N/A means the negative-flip rule exceeds that cap; these entries contribute zero to its aggregate summary.

| Dataset | Model | Update | Noise floor | Version unchanged false positive rate | | Matched agreement at false positive rate $\leq 0.05$ | | |
|---|---|---|---|---|---|---|---|---|
| | | | | Naive | Calibrated | Calibrated | Null quantile | Naive negative flip |
| ARC-Challenge | Mistral-7B | P | 0.21 | 0.028 | 0.000 | 1.00 | 1.00 | 0.75 |
| GSM8K | Mistral-7B | P | 0.81 | 0.140 | 0.000 | 1.00 | 0.93 | N/A |
| MMLU | Mistral-7B | P | 0.28 | 0.042 | 0.000 | 1.00 | 1.00 | 0.76 |
| ARC-Challenge | Phi-3.5-mini | P | 0.14 | 0.019 | 0.000 | 0.75 | 1.00 | 0.75 |
| GSM8K | Phi-3.5-mini | P | 0.47 | 0.078 | 0.000 | 1.00 | 1.00 | N/A |
| MMLU | Phi-3.5-mini | P | 0.29 | 0.047 | 0.000 | 0.95 | 1.00 | 0.64 |
| ARC-Challenge | Qwen2.5-Instruct 1.5B | P | 0.28 | 0.039 | 0.000 | 1.00 | 1.00 | 0.90 |
| GSM8K | Qwen2.5-Instruct 1.5B | P | 0.80 | 0.133 | 0.000 | 1.00 | 1.00 | N/A |
| HumanEval | Qwen2.5-Instruct 1.5B | P | 0.71 | 0.123 | 0.000 | 1.00 | 0.71 | N/A |
| MBPP | Qwen2.5-Instruct 1.5B | P | 0.61 | 0.105 | 0.000 | 0.88 | 1.00 | N/A |
| MMLU | Qwen2.5-Instruct 1.5B | P | 0.38 | 0.046 | 0.000 | 1.00 | 1.00 | 0.73 |
| ARC-Challenge | Qwen2.5-7B | P | 0.17 | 0.025 | 0.000 | 1.00 | 0.00 | 0.90 |
| GSM8K | Qwen2.5-7B | P | 0.55 | 0.086 | 0.000 | 1.00 | 1.00 | N/A |
| HumanEval | Qwen2.5-7B | P | 0.26 | 0.040 | 0.000 | 1.00 | 1.00 | 0.52 |
| MMLU | Qwen2.5-7B | P | 0.35 | 0.058 | 0.000 | 1.00 | 1.00 | N/A |
| GSM8K | Qwen2.5-7B | S | 0.55 | 0.083 | 0.000 | 1.00 | 1.00 | N/A |
| MMLU | Qwen2.5-7B | S | 0.35 | 0.059 | 0.000 | 1.00 | 1.00 | N/A |
| ARC-Challenge | Qwen2.5-7B | S | 0.17 | 0.024 | 0.000 | 1.00 | 0.00 | 0.83 |
| GSM8K | Mistral-7B | S | 0.81 | 0.141 | 0.000 | 0.71 | 0.65 | N/A |
| MMLU | Mistral-7B | S | 0.28 | 0.044 | 0.000 | 1.00 | 1.00 | 0.79 |
| ARC-Challenge | Mistral-7B | S | 0.21 | 0.027 | 0.000 | 1.00 | 1.00 | 0.81 |
| GSM8K | Phi-3.5-mini | S | 0.47 | 0.077 | 0.000 | 1.00 | 1.00 | N/A |
| MMLU | Phi-3.5-mini | S | 0.29 | 0.046 | 0.000 | 1.00 | 1.00 | 0.80 |
| ARC-Challenge | Phi-3.5-mini | S | 0.14 | 0.019 | 0.000 | 1.00 | 1.00 | 0.87 |

Table 4: Mean results over 24 benchmark settings. Agreement is the fraction of positive finite-run A/B labels found. Matched summaries maximize agreement subject to the observed unchanged-version cap; intervals resample settings. The negative-flip row uses pairwise null rates and pool-frequency update flags (Section 4.2).

| Method | Update runs / item | Same-version rate | Default agreement | Matched agreement (95% interval) |
|---|---|---|---|---|
| Fisher + BH, $q = 0.05$ | 40 | 0.000 | 0.857 | 0.970 [0.94, 1.00] |
| Majority voting | 40 | 0.022 | 0.791 | 0.725 [0.62, 0.81] |
| Negative-flip frequency | 40 | 0.064 | 0.720 | 0.419 [0.26, 0.57] |
| Empirical null quantile | 40 | 0.060 | — | 0.887 [0.76, 0.98] |

The matched summaries reach 0.970 agreement for Fisher/BH and 0.887 for the empirical null threshold. Their intervals overlap, and the threshold baseline is stronger in some settings: for the Phi prompt edit on ARC-Challenge it reaches 1.00 agreement while Fisher/BH reaches 0.75. Table 3 lists all 24 settings and preserves these exceptions. For negative flips, only 13 of 24 settings meet the control-rate cap; zero-coding the others lowers its matched mean to 0.419. These retrospective aggregates are useful for comparing the recorded operating choices, but the default Fisher/BH and majority-vote rules provide the clearer fixed-rule comparison.

### 4.5 STRUCTURED FUNCTION RESPONSES

The separate Glaive evaluation tests whether the procedure also handles a structured output. Given a request and a function schema, the model returns one function call; success requires the function name and arguments to match the target. No tool is executed. Across the two Qwen comparisons,

the reference noise floor was 53%, the naive same-version false-positive rate was about 9.9%, and Fisher/BH produced no unchanged-version discoveries. For the model swap, 41 items received positive A/B labels; agreement was 92.7% at the default rule and 100% in the matched summary.

The prompt edit produced no observed drop above the 0.30 margin, so its agreement is undefined. This is an informative limit of the comparison: it supplies a structured-output control but no positive update labels with which to measure sensitivity. Both comparisons remain separate from the 24-setting benchmark averages.

### 4.6 Repetition budget and multiplicity

With few runs, even the largest possible count difference gives limited statistical evidence. At $K = 4$, Fisher's smallest directional $p$ value is $1/\binom{8}{4} = 0.0143$. That exceeds the first BH threshold, $0.05/200 = 0.00025$, in a 200-item suite. A group can still pass: at rank 58 the threshold is $58(0.05/200) = 0.0145$. This discrete resolution explains why a short test can miss isolated drops even when the observed difference is large.

The budget sweep selects one subset of run columns at each $K$, using seed 7, and repeats the analysis. At each budget, the update uses $K$ runs per version and the unchanged-version control splits the $K$ reference runs into two halves. Mean agreement at $K = 4, 8, 12, 16, 20$ was $2.5\%, 42.5\%, 59.8\%, 72.5\%, 85.7\%$, with zero observed unchanged-version flags at every budget (Figure 3). The positive labels are recomputed from the selected runs at each $K$, and settings without positives are excluded from that agreement mean. The curve therefore combines increased test resolution with changes in the finite-run reference. For 200 items, the update comparison costs $2K \times 200$ executions: 1,600 at $K = 4$, 3,200 at $K = 8$, and 8,000 at $K = 20$.

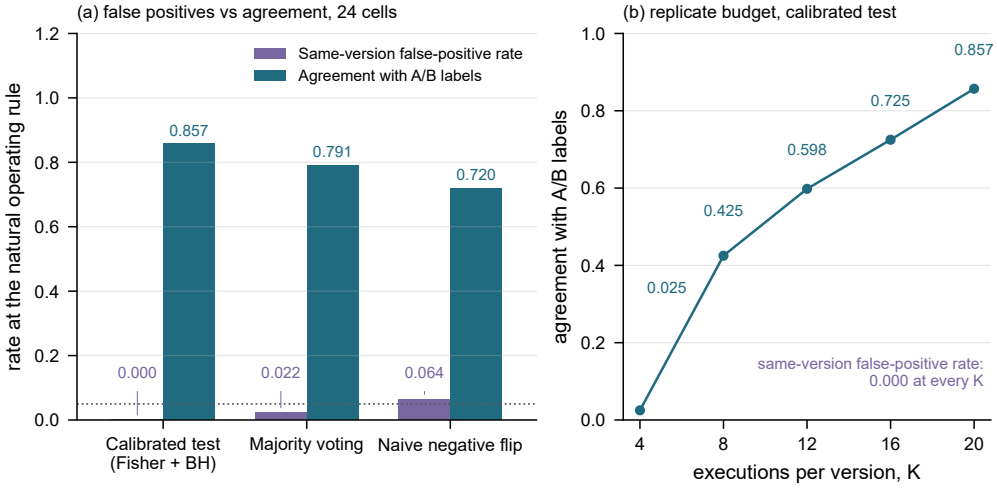


Figure 3: Default-rule summaries and repetition budget. Panel (a) reports the rates in Table 4; the negative-flip null and update bars use the different summaries specified in Section 4.2. Panel (b) uses one subset per budget and recomputes its A/B labels, so it is a descriptive budget curve rather than recall against one fixed reference set.

Removing BH isolates the effect of suite correction while retaining Fisher's itemwise evidence. At the uncorrected threshold $P_F < 0.05$, mean agreement rises to 0.999 and mean unchanged-version flags rise to 0.005 (Table 5). The correction sacrifices agreement with some large observed drops to obtain the more conservative discovered set. This comparison separates the cost of multiplicity correction from the cost of collecting repeated runs.

Table 5: Effect of suite correction, averaged over the 24 benchmark settings. Both rows use the same repeated outcomes and A/B labels.

| Configuration | Same-version false-positive rate | A/B agreement |
|---|---|---|
| Fisher per item + BH ($q = 0.05$) | 0.000 | 0.857 |
| Fisher per item without BH | 0.005 | 0.999 |

Taken together, the comparisons separate calibration from sensitivity. The unchanged-version control asks whether a rule raises alarms when the version is held fixed; the update comparison asks whether it selects items with a large observed drop. These questions are related but not interchangeable. Majority voting and the empirical null threshold can agree with many observed drops while retaining nonzero control rates, whereas Fisher/BH combines itemwise evidence with suite correction and is consequently more conservative. The budget sweep adds a further constraint: at small $K$, the discrete count test cannot resolve an isolated drop even when the observed difference is large. The procedure is therefore best read through its discovered set, control behavior and run budget together.

The per-setting results also show why the aggregate summaries should not be treated as a universal ranking. The calibrated rule has zero observed same-version discoveries in every listed setting, but its matched agreement depends on the task and the update label. The threshold baseline is stronger in some settings, while the negative-flip rule fails the retrospective control cap in several settings and is assigned zero in that summary. These differences reflect the operating point and reporting convention, not only the evidence in the update comparison. The same boundary appears in the structured-output check: the prompt comparison supplies a control but no positive labels, so its agreement is undefined rather than zero. Applying the procedure to a new serving system would therefore require unchanged-version calibration before fixing a threshold and interpreting an observed drop.

## 5 LIMITATIONS

The empirical sensitivity target is a large drop in the observed runs. Since the same outcomes produce both its label and its test statistic, agreement can be optimistic about recovery of latent probability decreases. Retrospective threshold selection adds a second source of reuse. Independent reference runs and separately chosen operating points are needed to estimate detection power against an external target.

The unchanged-version split uses 10 runs per side, compared with 20 per side under updates. Its observed zero rate consequently characterizes that control budget. Dependence among executions from shared infrastructure can affect itemwise test validity, and reuse of reference items and runs across settings can affect uncertainty across settings. The reported bootstrap treats settings as its resampling units and does not model all cross-setting dependence. Formal FDR control and generalization to another serving system require the corresponding assumptions to hold there.

The tasks use fixed binary checks: answer matching, executable unit tests and function-call matching. Continuous scores and LLM-judge ratings introduce additional measurement assumptions, while multi-turn agents add environment state and tool interactions. The current function-call evidence covers one model family and one response per task. The constructed prompt edits and model swaps also represent only part of the changes encountered in production.

## 6 CONCLUSION

A failed rerun becomes more informative when it is interpreted alongside repeated outcomes and an unchanged-version control. Across this benchmark battery, Fisher/BH produced no discoveries in the recorded unchanged-version splits while selecting most large finite-run drops after prompt and model changes. The voting comparison, repetition sweep and BH ablation explain how the discovered set depends on counts, budget and suite correction. These results support reporting each discovered set alongside its repeated success counts, unchanged-version alarm rate and execution budget.

## Reproducibility Statement

The accompanying project contains the generation harness, comparison manifest, analysis and ablation scripts, and per-setting result summaries. The original per-item correctness matrices are identified in the experiment reproduction handoff; the available local result summaries support the values reported here.

## AI Use Statement

Generative AI tools assisted with drafting and revising the manuscript. The authors are responsible for reviewing the text, experimental design, implementation, data, numerical results and methodological claims.

## References

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.

Yoav Benjamini and Daniel Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of statistics*, pp. 1165–1188, 2001.

Tanay Dixit, Daniel Lee, Sally Fang, Sai Sree Harsha, Anirudh Sureshan, Akash V Maharaj, and Yunyao Li. Retain: Interactive tool for regression testing guided llm migration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 301–310, 2024.

Jessica Maria Echterhoff, Fartash Faghri, Raviteja Vemulapalli, Ting-Yao Hu, Chun-Liang Li, Oncel Tuzel, and Hadi Pouransari. Muscle: A model update strategy for compatible llm evolution. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 7320–7332, 2024.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Wanqin Ma, Chenyang Yang, and Christian Kästner. (why) is my prompt getting worse? rethinking regression testing for evolving llm apis. In *Proceedings of the IEEE/ACM 3rd international conference on AI engineering-software engineering for AI*, pp. 166–171, 2024.

Shuyin Ouyang, Jie M Zhang, Mark Harman, and Meng Wang. An empirical study of the non-determinism of chatgpt in code generation. *ACM Transactions on Software Engineering and Methodology*, 34(2):1–28, 2025.

Michael Pradel. Testora: Using natural language intent to detect behavioral regressions. In *Proceedings of the 2026 IEEE/ACM 48th International Conference on Software Engineering*, pp. 3169–3181, 2026.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 4902–4912, 2020.

Yifan Song, Guoyin Wang, Sujian Li, and Bill Yuchen Lin. The good, the bad, and the greedy: Evaluation of llms should not ignore non-determinism. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4195–4206, 2025.

Yuqing Xie, Yi-An Lai, Yuanjun Xiong, Yi Zhang, and Stefano Soatto. Regression bugs are in your model! measuring, reducing and analyzing regressions in nlp model updates. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 6589–6602, 2021.

Xiaonan Xu and Wenjing Wu. What aggregate scores miss: Measuring item-level regressions in commercial llm api migrations. *arXiv preprint arXiv:2608.17719*, 2026.