THE EMPEROR'S NEW SCAFFOLD: SEMANTICALLY EQUIVALENT HARNESSES RESHUFFLE THE LEADERBOARD

Anonymous authors
Paper under double-blind review

ABSTRACT

Every new language model arrives with a leaderboard that crowns it, and the community reads these tables as measurements of the models themselves. But does a leaderboard for agentic tasks measure the model, or the model together with the harness of prompts and protocol it was run under? We show that a one-word, semantically equivalent edit to the harness drops a flagship model's accuracy from 93.3 to 14.0 on code generation, below all five sub-2B models we tested. To test whether this is an isolated failure, we evaluate 24 models under 32 equivalent harness variants on three multi-turn agentic tasks (text-to-SQL, math, code). Nearly every pair of closely ranked models can be flipped by some equivalent rewrite, with swings far beyond resampling noise, and equivalent harnesses disagree on the top-3 in 47% to 84% of configurations across the three tasks. This instability lands exactly where it costs the most. Nobody consults a leaderboard to learn that a frontier model beats a sub-1B one; they consult it to choose between close competitors. Tracing every deviation through an exact decomposition, we find three failure modes, Delivery, Format, and Content; which one fails depends on which model meets which rewrite, and no harness in our grid treats all models impartially. Harness rewrites cannot move mountains, but they reshuffle the close races. We therefore release ScaffoldSpread, an eight-configuration audit that reproduces full-grid sensitivity at $\rho = 0.981$, together with the parameterized harness.

1 INTRODUCTION

Every new language model ships with a leaderboard. Releases report their models on agentic benchmarks (Merrill et al., 2026; Patwardhan et al., 2026; Center for AI Safety et al., 2026; Xie et al., 2024), and aggregators keep running tables (Liang et al., 2023; Chiang et al., 2024), so the community reads these tables as measurements of the models themselves. But a leaderboard never measures a model alone, since it measures the model together with a harness, the system prompt and instructions that carry each task. The harness is assumed to be neutral, but prior work on prompt and format sensitivity says otherwise (Lu et al., 2022; Zhao et al., 2021; Sclar et al., 2024; Voronov et al., 2024). We show how far it can go on multi-turn agentic tasks. GPT-5.6 scores 93.3 on code generation when the harness asks for its final answer in `<solution>` tags, but drops to 14.0 when it asks for `<answer>` instead. The two instructions differ by only one word and give the model no different information about the task. Its rank falls from #6 to #24, below all five sub-2B models we tested (Figure 1). The code in this example is correct, but the model violates the submission format the instruction specifies. Executing the extracted submission with the extra tags intact raises a SyntaxError.

A single case cannot establish a general claim, no matter how large the swing. We therefore built an instrument covering seven dimensions of presentation and instantiated it as 32 configurations that are semantically equivalent by construction. We evaluate 24 models on three multi-turn agentic tasks drawn from established benchmarks: text-to-SQL (Li et al., 2023), competition mathematics with a Python interpreter (Hendrycks et al., 2021; Mathematical Association of America, 2024; 2025), and code generation (Chen et al., 2021). Averaged over the 24 models, the 32 configurations span at most 8 points on SQL, 11 on math, and 20 on code. That is real but modest, since the models themselves are separated by 61 to 86 points on the same scale and no configuration helps or hurts every model

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.

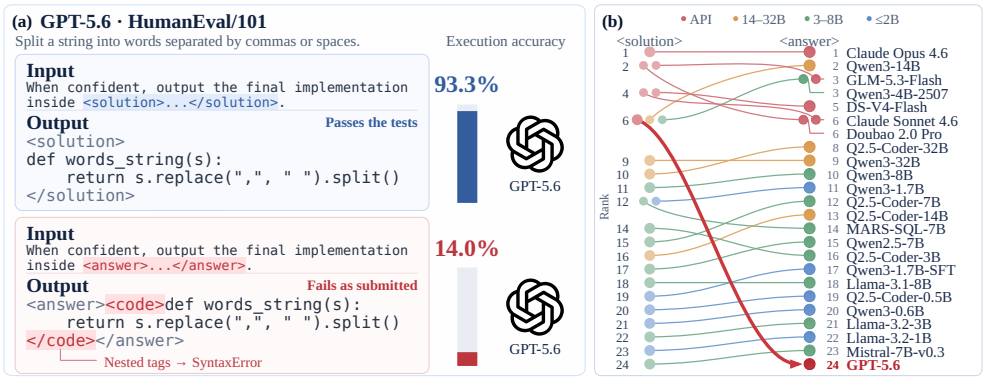

Figure 1: Changing one word in the harness. (a) The same model and the same code task, with the final answer asked for in `<solution>` tags above and `<answer>` tags below; the model still writes correct code, and only the surrounding tags differ. (b) The two configurations are semantically equivalent, yet GPT-5.6 falls from rank 6 to rank 24, below all five sub-2B models, while the mid-field reorders around it. Ranks follow the minimum-rank convention: tied models share the best of their ranks, which leaves some rank numbers unused.

alike. The average conceals the damage, which lands in some specific model–configuration cells among different models and is cancelled out by pooling. A leaderboard, however, is not an average but a series of pairwise comparisons. Nearly every pair of closely ranked models can be reversed by some equivalent rewrite. A natural objection is that this churn is noise. Rerunning the suspicious cells changes a model's score by only 1.4 points at the median, while most large drops, like the one in Figure 1, persist.

Those reversals trace back to changes in individual scores. Within a single model–configuration cell, the change in accuracy decomposes into three terms, one for each way a run can fail. Delivery covers the loss when no answer is submitted through the protocol our prompt defines, Format when a correct answer is not submitted in the format the instructions specify, and Content when the submitted answer is wrong. Which gates carry the loss depends on the domain, counting only the cells where a rewrite costs a model more than seven points: all three on code, Delivery and Content alone on math and text-to-SQL, whose judges execute the query or normalize the expression rather than parse the wrapper.

We make four contributions.

- An instrument: a parameterized harness engine, instantiated as 32 semantically equivalent configurations and run over 24 models on three multi-turn agentic tasks, released with the paper.
- A phenomenon: equivalent rewrites are modest on average and decisive in the tails. A ranking is therefore a property of the model together with its harness rather than of the model alone, and the damage concentrates in the close comparisons that leaderboards exist to settle.
- A mechanism: every lost point traces to one of three causes, Delivery, Format, or Content, and which one dominates depends on which model meets which rewrite, not on the rewrite alone.
- A prescription: ScaffoldSpread, an eight-configuration audit that recovers the full grid's ranking of model sensitivity at $\rho = 0.981$, and the practice of reporting harness sensitivity alongside benchmark scores.

Section 2 defines what makes a harness rewrite equivalent, and Sections 3 and 4 present the phenomenon and its anatomy. Section 5 turns the anatomy into the audit, and Sections 6 and 7 take up prior work, the limits of our claims, and what we recommend.

2 Setup: Semantically Equivalent Harness Perturbations

We compare model scores across harnesses that differ only in presentation, such as the two in Figure 1, and this section defines what "only in presentation" means and how we enforce it.

2.1 THE GRID

Every run gives the model a system prompt, then the task, then up to five action–observation turns. Seven dimensions rewrite this text and nothing else: *form* (layout of the task input), *think* (reasoning format; the `react` level follows ReAct (Yao et al., 2023)), *action* (the tag or tool used to submit), *reminder* (a turn countdown in each observation), *wording* (synonyms in the prompt), *case* (capitalization of tag and tool names), and *order* (permutation of the prompt's rules). Appendix A.1 shows every level rendered, with the changed span highlighted.

No level adds information, budget, or a scoring advantage: the task content, the five turns, the token limit, the tools, the judge, and the reference answer are identical under every configuration. That is what we mean by equivalent. *Form* and *case* are byte-level changes, while *think* and *action* name a convention the model can follow badly, and that is the effect we measure.

The grid has 32 configurations: one anchor, its twelve single-dimension neighbours, and nineteen random combinations drawn once with a fixed seed. The anchor is the configuration closest in Hamming distance to all others, so the deviations Section 4 measures from it are measured from the centre of the grid. The grid was fixed before any model was run.

2.2 TASKS, MODELS, BANDS

The three tasks are text-to-SQL on BIRD (Li et al., 2023), competition mathematics (Hendrycks et al., 2021; Mathematical Association of America, 2024; 2025) with a Python interpreter, and code generation on HumanEval (Chen et al., 2021). Each has an executable judge and a natural probe action (run a query, run code, run the visible tests). Twenty-four models run all 32 configurations on all three, giving 72 model–task units and 2,304 cells.

For the reliability analysis the models fall into four bands: at most 2B (5 models), 3–8B (9), 14–32B (4), and frontier API (6). The cut points sit at the natural gaps in the pool and were fixed before any experiment was run.

If a model never submits a final answer, the engine scores its last probe instead of a zero, which separates a model that could not follow the protocol from one that had nothing to submit. The parser strips exactly the wrappers the harness asked for and nothing else, so a model is never penalized for obeying its instruction and never rescued from ignoring it.

3 THE PHENOMENON AND ITS CONSEQUENCES

Equivalent harnesses move scores by a small amount on average and by a large amount in a few specific cells.

3.1 THE VARIANCE HAS STRUCTURE

Table 1 orders the models by mean accuracy, and the range column does not follow that order. GPT-5.6 sits in the frontier band on every task, yet its code range is 85 points, against 2 for Claude Opus 4.6 and 11 for Claude Sonnet 4.6. The bold and underlined cells are a minority and they cluster in the code column.

To put numbers on that shape, we ask how much of the total variance in the 2,304 cell scores is due to the harness. Each cell is one model on one task under one configuration, and we write its distance from the grand mean as a sum of parts: a model effect for each model, a task effect for each task, a configuration effect for each configuration, and the interaction between them. We report each term as the percentage of the total variance it accounts for. This is a three-factor analysis of variance, and Appendix A.2 gives the formulas, including the two remaining interaction terms.

The model effect accounts for 61.8% of the variance and the task effect for 22.4%; these two are the leaderboard's signal, and they are the ordering of the means in Table 1. The harness effect is the other two terms. The configuration effect, the average effect of a rewrite across all models, is 0.8%. The model–configuration interaction, the effect of a rewrite that differs from model to model, is 3.0%, 3.75 times the configuration effect, and it decides whether a given rewrite helps or hurts a given model.

		Text-to-SQL				Math				Code			
	Model	median	mean	std	range	median	mean	std	range	median	mean	std	range
A	Claude Opus 4.6	63.7	63.9	1.2	5.0	96.3	95.9	2.4	9.0	100.0	99.6	0.5	1.8
	Claude Sonnet 4.6	59.2	59.3	0.9	3.3	92.6	92.5	3.0	13.6	95.7	95.6	3.1	11.0
	GLM-5.3 Flash	59.8	59.4	2.2	10.3	92.0	92.0	2.1	8.5	96.3	93.2	8.3	**43.9**
	DeepSeek V4 Flash	52.7	52.5	1.9	7.3	91.5	91.4	2.0	9.3	94.5	95.2	2.7	10.4
	Doubao Seed 2.0 Pro	53.0	52.9	1.2	6.7	86.3	87.2	3.2	11.2	95.1	93.5	6.6	26.8
	GPT-5.6	56.7	56.1	2.6	13.3	95.9	96.0	1.4	6.6	92.7	79.7	24.6	**85.4**
B	Qwen3-14B	52.0	50.2	4.9	18.3	90.4	90.3	1.1	5.1	93.3	93.2	2.2	10.4
	Qwen3-32B	51.2	51.1	1.9	10.0	88.7	88.4	1.8	6.9	92.7	90.0	7.8	**43.3**
	Qwen2.5-Coder-32B	54.3	54.5	1.8	6.3	65.2	65.0	5.9	19.7	88.7	79.0	19.7	**90.2**
	Qwen2.5-Coder-14B	54.2	53.7	2.5	12.0	63.0	62.9	2.8	12.8	79.0	71.0	21.9	**80.5**
C	Qwen3-4B-2507	57.5	56.0	5.1	23.3	88.8	85.2	7.3	29.8	92.7	90.3	5.7	21.3
	Qwen3-8B	49.7	49.4	2.5	11.0	88.6	88.4	1.0	4.0	85.1	83.6	8.7	**42.7**
	MARS-SQL-7B	50.8	48.2	9.7	**45.7**	55.3	55.0	6.8	**35.9**	73.2	67.0	17.9	**65.9**
	Qwen2.5-Coder-7B	46.2	41.2	11.3	**41.0**	56.0	55.2	7.4	**30.3**	71.3	70.7	14.1	**56.1**
	Qwen2.5-7B	44.5	40.3	11.5	**50.0**	56.9	48.4	16.3	**48.9**	73.2	69.8	15.4	**51.8**
	Qwen2.5-Coder-3B	36.3	35.1	5.2	24.7	46.8	46.2	7.1	**31.9**	76.5	70.7	12.4	**42.7**
	Llama-3.1-8B	30.3	26.9	10.6	**34.7**	28.6	28.1	7.5	28.5	58.2	59.2	7.9	**36.0**
	Llama-3.2-3B	17.8	15.1	5.9	20.0	26.1	25.1	5.9	29.0	47.6	42.8	13.4	**50.6**
	Mistral-7B-v0.3	15.2	14.2	5.7	21.3	9.8	9.8	1.9	9.3	20.7	21.1	4.7	20.1
D	Qwen3-1.7B	39.2	38.4	4.4	19.3	81.0	80.9	1.6	6.9	77.7	76.2	6.3	25.0
	Qwen3-1.7B-SFT	28.5	28.5	8.2	**31.0**	74.2	66.7	16.2	**62.5**	64.0	61.3	11.1	**48.8**
	Qwen3-0.6B	22.0	20.3	5.4	24.3	62.2	61.4	3.4	15.4	44.5	43.8	5.5	24.4
	Qwen2.5-Coder-0.5B	2.8	4.2	3.8	12.0	13.2	12.4	2.0	7.4	51.5	44.4	17.6	**57.9**
	Llama-3.2-1B	3.2	3.1	2.9	9.0	15.8	14.9	6.1	20.5	28.4	26.2	10.2	**36.0**

Table 1: Accuracy of every model on every task over the 32 equivalent configurations, in percentage points. A–D are the four pre-registered bands, as in Figure 1: A frontier API, B 14–32B, C 3–8B, D at most 2B; within a band, models are ordered by mean accuracy. A range of 30 points or more is set in bold (25 of 72 units) and a standard deviation of 10 points or more is underlined (16 of 72). The standard deviation is the sensitivity that Section 5 estimates with eight configurations.

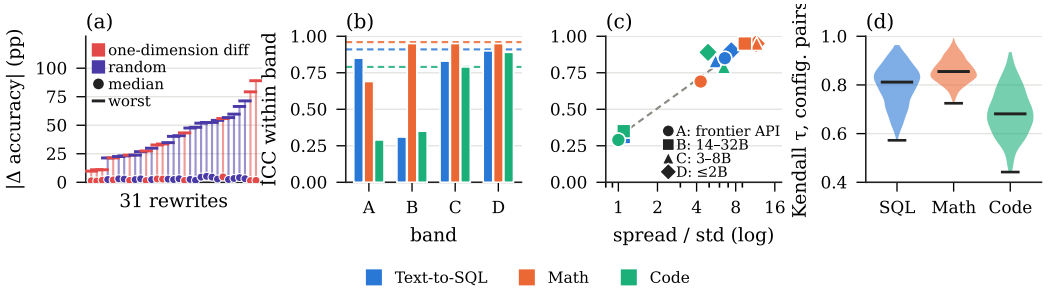

Figure 2: The harness effect is small on average and large in specific places. (a) For each of the 31 non-anchor configurations, the median (dot) and the largest (bar) absolute change in accuracy relative to the anchor across the 72 model–task units; red marks change one dimension of the anchor, violet marks are random combinations. (b) Score consistency within each band, ICC(3,1), by task; dashed lines are the same statistic over the full pool of 24 models. (c) The same twelve within-band ICC values against spread/std on a log axis; the dashed line is a least-squares fit, and Spearman $\rho = 0.93$. (d) Kendall τ between the model rankings induced by two equivalent configurations, over all 496 configuration pairs per task; the bar is the median and the tick the minimum. Color is the task; in (c), marker shape is the band. Appendix A.7 lists the numbers behind (a) and (c).

The variance is also concentrated in a few units rather than spread evenly: the Gini coefficient of per-unit variance across the 72 units is 0.68, and the five most variable units account for 40% of the total (Appendix A.4).

Changing one dimension at a time shows the same pattern (Figure 2a). For every dimension, the median absolute change across the 72 units is small: at most 4.8 points (switching the action protocol to native function calling) and as little as 1.1 (removing the turn countdown). The maximum change is not small for any dimension. Reordering the three rules changes one unit by 89.0 points; renaming the answer tag changes one by 79.3. The median does not predict the maximum.

3.2 These swings are not noise

A 79-point swing invites the initial question of whether the measurement itself is that unstable, and it is not, on any of the three ways we checked. Resampling the items of a task, so that a model is scored on a different draw of questions, moves its score by 1.2 to 3.5 points. Rerunning the same cell five times, holding both the items and the configuration fixed, gives a per-cell standard deviation of 1.4 points at the median and 2.0 pooled RMS; we ran this on the cells that deviate most, where run-to-run variation would be largest if it were the explanation. Retesting the six API models a day apart moves any configuration's score by at most two points, so provider drift over the study period does not account for the effects either. The single-dimension effects reach 89 points (Figure 2a) and the example of Figure 1 is 79, so the reshuffling of the next section rests on swings more than an order of magnitude above the noise these controls establish.

3.3 Ranking consequences

A leaderboard reports pairwise comparisons, not averages, so the interaction term affects it directly. For each pair of equivalent configurations we order the models by score, breaking ties by model name, and count how many of the 276 model pairs keep their order under both and how many reverse; Kendall's τ is the difference as a fraction of 276. Figure 2d shows all 496 configuration pairs per task. The median is 0.86 on math, 0.81 on text-to-SQL, and 0.68 on code, and code has the long lower tail: one in five of its configuration pairs is below 0.6, and the worst pair, at 0.44, reverses 77 of the 276 model pairs.

Adjacent ranks are the least stable. The reference board, ordered by cross-configuration median score with tied medians excluded, has 23 pairs of adjacent ranks. Some equivalent configuration reverses 95.7% of them on text-to-SQL, 90.0% on code, and 82.6% on math. If we count only reversals larger than 3.5 points, the shares are 91.3%, 90.0%, and 69.6%. That cutoff is a filter, not an estimate of noise. The reversals it keeps still lie above the noise floor of Section 3.2.

The top of the board is not stable either. We take the top-3 set under each configuration and compare it to the top-3 set of the board built from cross-configuration median scores. The two disagree for 84% of configurations on code, 50% on text-to-SQL, and 47% on math. The Friedman and Nemenyi analysis of these rankings, and the analysis of reversals by band distance, are in Appendix A.3.

Individual models move a long way. On code, GPT-5.6 occupies every rank from 2 to 24 across the grid, Coder-32B from 5 to 24, and GLM-Flash from 2 to 19. Twelve of the 24 models move ten or more places. Text-to-SQL and math move less: the median rank span is 7 and 4 places, against 10 on code.

3.4 Reliability by band

The instability is not evenly spread across the pool. We measure score consistency with the intraclass correlation, treating configurations as raters and models as subjects; because the grid is fixed, we use the two-way mixed-effects, single-measure, consistency form, ICC(3,1) (Shrout & Fleiss, 1979; Koo & Li, 2016). Over all 24 models the ICC is high: 0.79 on code, 0.91 on text-to-SQL, 0.96 on math. Within a single band it is not (Figure 2b). On code, the frontier band has ICC 0.29 and the 14–32B band 0.35, while the smallest band has 0.89. Three of the four bands are below 0.8. The high pooled figure hides this: scores can be consistent across the whole pool while the order inside the frontier band depends on the configuration.

The low within-band ICC has a simple explanation. For each band and domain we compute two numbers: the spread of the members' median scores, and the average per-model standard deviation across configurations. Their ratio tracks ICC across the twelve band–domain cells, with Spearman correlation 0.93 (Figure 2c). In the frontier code band the spread is 7.3 points and the harness

Model	median	mean	std	range	acted	Model	median	mean	std	range	acted
DeepSeek V4 Flash	25.0	26.7	10.5	28.0	0.98	Claude Opus 4.6	24.0	22.0	7.7	**30.0**	1.00
Doubao Seed 2.0 Pro	24.0	26.0	11.1	**30.0**	1.00	Claude Sonnet 4.6	21.0	21.2	8.9	**30.0**	1.00
GLM-5.3 Flash	22.0	23.5	8.0	24.0	0.95	GPT-5.6	12.0	14.9	14.2	**40.0**	0.73

Table 2: Out-of-house validation on BFCL v3 multi-turn: 50 items, the official state checker, six API models under 20 equivalent configurations each, sorted by mean accuracy. Accuracy in percent, with bold range and underlined std marked as in Table 1. Acted is the mean fraction of episodes in which the model issued at least one tool call rather than only announcing one; every model does so in nearly every episode except GPT-5.6, whose acted rate collapses under the text-carried protocols.

deviation is 7.6. When the gaps between models are smaller than the movement a rewrite can cause, the order inside the band depends on the configuration, which is what an ICC of 0.29 reports.

Reversals are also more common between nearby models, and we report the rates by band distance in Appendix A.3. Either way, rewrites do not scramble the whole board, only the close races.

3.5 Out-of-house validation

Everything above uses our grid, our tasks, and our judges, so the effect could be an artifact of our setup. To check, we repeated the core comparison on a benchmark we did not build with a checker we did not write. We used BFCL v3 multi-turn base (Patil et al., 2025): 50 items, scored by the official state checker. Six API models ran 20 equivalent configurations each, and all 120 cells completed.

The within-band instability appears here too, and more strongly (Table 2). The range between a model's best and worst equivalent configuration is between 24 and 40 points across the six models, and GPT-5.6 scores zero under one configuration. We ran the same decomposition here. In 57 of the 63 cells that fall, the largest term is Content: models issue at least one tool call in almost every episode, but when the protocol carries the call as text rather than as a native call, what they do with it is more often wrong. This is a different balance from code, where Delivery and Format dominate.

4 Anatomy of the Variance

This section explains where the variance of Section 3 comes from. It has two parts, and we keep them separate. The first is an accounting identity, which splits each cell's deviation exactly into the three terms from the introduction and holds by construction, so on its own it explains nothing. The second is evidence: for each term, an intervention or a trace analysis that ties it to a specific failure the model commits.

4.1 Three gates, one identity

A run scores through one of two routes: an answer submitted through the protocol the prompt defines, or, failing that, the fallback of Section 2, which scores the last probe. The three gates sort what can go wrong on the way. Delivery is coverage: which route the run ends on. Format is the parsing tax: the score lost between the strict parse and a parse that strips one wrapper. Content is precision: whether what was submitted passes the judge, on either route.

Taking the anchor configuration as reference, each cell's accuracy change decomposes as

$$\Delta\mathrm{Acc} \;=\; -\Delta\mathrm{Tax} \;+\; \Delta\mathrm{Coverage} \;+\; \Delta\mathrm{Precision},$$

where a coverage change is weighted by the anchor's precision and a precision change by the cell's own coverage, the standard split. The identity holds with zero residual in all 2,304 cells, and that exactness comes from the construction rather than from measurement.

Every signed term is relative to the anchor, so the anchor has to be a reasonable reference (Appendix A.8). In 68 of the 72 units, the anchor score is within seven points of the unit's median score. In three units it is well below: the Qwen2.5-Coder 3B, 7B, and 14B code units, at 21.0, 25.0, and

11.3 points under their medians. Their signed terms are measured against a weak reference, and we flag them wherever they appear. In one unit it is above: Qwen2.5-7B on SQL, at 7.5 points over its median.

4.2 FORMAT: THE PARSING TAX IS REAL, CODE-SPECIFIC, AND CONFIGURATION-PRICED

To measure the Format term we hold the model output fixed and change the parser. We replayed 289,871 submission records through three parsers. L0 is the engine's strict parse. L1 is the same parse after stripping one wrapper layer, an outer fence or one enclosing tag. L3 is an oracle that accepts any extractable candidate.

On code, moving from L0 to L1 raises the pooled score by 7.7 points. The per-configuration picture is different. Across the 32 configurations the gain ranges from 0.0 to 33.2 points. Configurations that ask for fenced blocks gain nothing, because their output already parses cleanly. The `fenced-*-sql_solution` family gains 17 to 33 points, because that much was lost to wrapping. L1 recovers almost everything L3 recovers, so one wrapper layer accounts for the tax.

The one-word case of Figure 1 is an example. Under the `<answer>` configuration, GPT-5.6 submits on 98.8% of items and its code is correct. It nests the probe tag inside the final tag. The strict parse rejects the nesting, and 79 points are lost at the Format gate.

On math and text-to-SQL the tax is zero or close to it. Across the full 290k-record replay it is zero in every math cell and at most 0.43 points in any text-to-SQL cell; on the seal-aligned refill sample it is exactly zero in every cell of both domains. This follows from how those judges work: the math judge normalizes the answer expression and the SQL judge executes the query, so a wrapper around the answer does not change the verdict. Code is the one domain where the wrapper becomes part of the artifact the judge runs, which is why it is the domain that reorders most.

4.3 DELIVERY AND CONTENT: WHO FAILS WHICH GATE

After the tax, what remains splits between Delivery and Content, and we use the traces to see which failure each cell commits. For this analysis we score under L1, which removes the Format contribution, and we select cells that deviate from their unit median rather than from the anchor. This gives 146 cells. In 99% of them the dominant failure is one of three: the model writes code it never submits, submits the wrong artifact, or submits before it is done. The failure labels change sharply between a deviating cell and its best single-dimension neighbour. In one pair, 99% of runs never submit in one cell and 99% submit cleanly in the other, and the two cells differ in one dimension.

A run that never submits may still have written correct code. We took the last code block from each of 4,867 never-submitted runs and ran the official tests; 23.3% pass. A more aggressive extractor that tries seven routes, on a per-cell sample, raises this to 38.9%. The two numbers measure different things (one route versus seven), and we cite the lower one as the conservative figure. Either way, a large share of what the Delivery gate discards was correct.

We have no intervention for the Content term, so we only classify. The falling cells fall into three groups: models the fallback rescues (the engine submits their last probe, and the probe is often right), models that produce no answer anywhere in the trace, and models that submit a wrong answer with no sign of uncertainty.

4.4 THE SIGNED PICTURE

Figure 3 shows the full accounting. It has seven panels, one per domain and term, and each cell shows that term's signed contribution relative to the anchor.

Among cells that fall more than seven points below the anchor, the three domains fail differently. On code, 45% of the loss is at Delivery, 40% at Format, and 15% at Content. On math and text-to-SQL, where Format is zero, the split between Delivery and Content is 38% to 62% and 84% to 16%. The 106 code cells that rise more than seven points do not mirror this: 79% of the gain is at Content and only 3% at Format. Losses and gains come through different gates.

The figure also shows why no configuration is safe for every model. In the Format panel, whole columns are red because the tax depends on the configuration. In the Delivery panel, whole rows



Figure 3: Signed contribution of each term to a cell's accuracy change relative to the anchor, in percentage points, under strict parsing. Rows are the 24 models (key at lower right, order fixed across panels) and columns are the 32 configurations, grouped as anchor, single-dimension, and random. Red is a loss, blue a gain, on a shared ± 40 scale. Content in (a–c) and Delivery in (e–g); Format in (d), Code only, since it is zero for Math and SQL. Outlined cells: GPT-5.6 under `<answer>` in (d), Qwen2.5-Coder-32B under `action_first` in (g), and Qwen3-14B's five `codeblock` cells in (b). † marks the three code anchors that sit 11–25 points below their unit median (Appendix A.8).

are red because submission failure depends on the model. A configuration that avoids one model's failure mode runs into another's.

The decomposition uses each run's original correctness flag while the main table uses the flag resolved after reruns; the two disagree on 2,258 raw records, which changes 232 of the 768 code cells at two decimal places (Appendix A.6). The seven-point threshold comes from the pilot study (Appendix A.9), and the shares it produces are threshold-sensitive in math and text-to-SQL, so we state it wherever we report them.

5 An Eight-Configuration Audit

Running 32 configurations per model is too expensive for routine use. This section gives a subset of eight that serves one purpose: ranking models by how sensitive they are to the harness. FormatSpread (Sclar et al., 2024) estimates the range of a model's performance across prompt formats on a limited budget; our target is different, the ranking of sensitivity across models rather than the range for one model.

We define the target and then the subset. Each of the 72 units has a sensitivity score: the standard deviation of its accuracy across configurations. ScaffoldSpread is a fixed set of eight configurations. Estimating the sensitivity score from those eight instead of all 32 ranks the 72 units with Spearman correlation $\rho = 0.981$ against the full-grid ranking. That is what it certifies, which units are fragile and which are robust. It does not recover the full grid's leaderboard and it does not estimate accuracy, since a mean over eight configurations depends on the harness as much as any other mean.

We found the subset by exhaustive search over the same grid, so the correlation is in-sample. Two facts make it more than an artifact of the search. The subset is one fixed set of eight for all 72 units, not a different set per unit, and it covers all seven dimensions, including both action protocols that produce the code tax. Selecting the subset on two domains and evaluating it on the held-out third yields correlations of $\rho = 0.919$–0.968 (Appendix A.10).

A related question, how many configurations a benchmark needs to average over to reach a given reliability, is addressed in Appendix A.5.

6 Related Work

This paper draws on two lines of work. The first shows that prompt form affects single-turn scores: few-shot example order (Lu et al., 2022), prompt-induced label biases (Zhao et al., 2021), formatting choices that move scores by tens of points (Sclar et al., 2024), benchmark-specific perturbations (Alzahrani et al., 2024), and template-format sensitivity (Voronov et al., 2024). We move the object of study from the prompt to the harness of a multi-turn run, where the protocol for acting, observing, and submitting is part of the presentation.

The second line treats evaluation as measurement. Benchmark scores change when a dataset is rebuilt (Recht et al., 2019). Leaderboard targets can diverge from what users need (Ethayarajh & Jurafsky, 2020). A model that does well on a static benchmark can fail on adversarial examples (Kiela et al., 2021) or under small perturbations (Alzahrani et al., 2024). The choice of metric can create or remove an apparent qualitative effect (Schaeffer et al., 2023). Variance accounting (Bouthillier et al., 2021) and error bars (Miller, 2024) are two responses to evaluation uncertainty, and these concerns apply to aggregators such as HELM and Chatbot Arena (Liang et al., 2023; Chiang et al., 2024). We use two tools from this line. The intraclass correlation measures score consistency across raters (Shrout & Fleiss, 1979); here the raters are harness configurations. Our coverage and precision terms have the same quantity-and-price structure as the wage decomposition of Oaxaca (1973) and Blinder (1973).

7 Discussion

Is format failure a capability failure? One objection is that models should be robust to our perturbations, so sensitivity to them is a capability signal and a harness that exposes it is fair. We agree, and that is the problem. If format robustness is a capability, current leaderboards measure it by accident, under one arbitrary harness, without saying which. Whether format robustness counts as a capability or as a nuisance, a within-band ranking with ICC 0.29 does not tell us which model is better.

No gold harness. The fix is not to find the fair configuration, because our grid does not contain one. We define a configuration's effect on a model as the model's score under that configuration minus its median score across all configurations; this uses no configuration as a reference. A fair configuration would have about the same effect on all 24 models. The narrowest effect profile in our grid (Appendix A.11) still spans 6.2 points on text-to-SQL, 8.5 on math, and 12.2 on code; the widest span 48.7, 61.4, and 89.6. No configuration in any domain moves even half the models in the same direction with none moving the other way. Choosing a harness is choosing which models to favor. The only neutral option is to report the choice and the spread.

Practice. Report the harness, run a small set of equivalent rewrites, and publish the spread. For developing models, most of the loss is at the Delivery gate, in submission routing, so a model that submits reliably under whatever convention it is given loses nothing to a judge that tolerates wrappers. Day-apart retests of the API models point the same way, with scores moving by at most two points and the movement concentrated in the same cells the grid flags as volatile.

The limits of these claims are listed in Appendix A.12.

8 Conclusion

Equivalent harness rewrites leave the average score almost where it was and move the comparisons a leaderboard is read for. Across 24 models, 32 equivalent configurations, and three agentic tasks, the effect on the pooled mean is small and the effect on adjacent ranks is not: the loss decomposes into a submission, a format, and a content term, and which one carries it depends on which model meets which rewrite. ScaffoldSpread estimates a model's sensitivity to that choice from eight configurations, and we release it with the parameterized harness.

AI USE STATEMENT

We used generative AI tools for drafting and revising the text of this paper under detailed author direction, for writing analysis scripts, and for literature search. All experimental design decisions, all statistical rulings, and all claims were made and verified by the authors, and every number in the paper was recomputed from the sealed result files before use. We have reviewed all AI-assisted work.

ETHICS STATEMENT

This work evaluates publicly released and commercially served language models on established benchmarks; it involves no human subjects, no personal data, and no deployment. The main ethical consideration is downstream: our results show that leaderboard ranks among closely matched models are sensitive to harness choices, so we caution against using single-harness rankings as grounds for procurement, safety, or capability claims about one model over a close competitor. We release our harness and audit protocol to make such claims checkable.

REPRODUCIBILITY STATEMENT

We release the parameterized harness engine, the frozen 32-configuration grid with its generation seed, the frozen scale of decoding and budget parameters, the per-episode result files with per-row provenance fields, the analysis scripts behind every number in the paper, and the ScaffoldSpread subset with its selection procedure. Section 2 specifies the equivalence constraints and Section 3.2 the noise controls; the appendix records the anchor audit, threshold sensitivity, and the audit pipeline. API models are identified by serving alias and access date, and day-apart retest drift is reported in Section 7.

REFERENCES

Norah Alzahrani, Hisham Alyahya, Yazeed Alnumay, Sultan AlRashed, Shaykhah Alsubaie, Yousef Almushayqih, Faisal Mirza, Nouf Alotaibi, Nora Al-Twairesh, Areeb Alowisheq, M Saiful Bari, and Haidar Khan. When benchmarks are targets: Revealing the sensitivity of large language model leaderboards. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13787–13805. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.744. URL https://aclanthology.org/2024.acl-long.744/.

Alan S. Blinder. Wage discrimination: Reduced form and structural estimates. *The Journal of Human Resources*, 8(4):436–455, 1973. doi: 10.2307/144855. URL https://doi.org/10.2307/144855.

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 *Proceedings of Machine Learning and Systems*, volume 3, pp. 747–769, 2021. URL https://proceedings.mlsys.org/paper_files/paper/2021/hash/0184b0cd3cfb185989f858a1d9f5c1eb-Abstract.html.

Center for AI Safety, Scale AI, and HLE Contributors Consortium. A benchmark of expert-level academic questions to assess AI capabilities. *Nature*, 649(8099):1139–1146, 2026. doi: 10.1038/s41586-025-09962-4.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders,

Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374, 2021. URL https://arxiv.org/abs/2107.03374.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael I. Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating LLMs by human preference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 8359–8388. PMLR, 2024. URL https://proceedings.mlr.press/v235/chiang24b.html.

Lee J. Cronbach, Goldine C. Gleser, Harinder Nanda, and Nageswari Rajaratnam. *The Dependability of Behavioral Measurements: Theory of Generalizability for Scores and Profiles*. John Wiley & Sons, New York, 1972. ISBN 0-471-18850-6.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006. URL https://www.jmlr.org/papers/v7/demsar06a.html.

Kawin Ethayarajh and Dan Jurafsky. Utility is in the eye of the user: A critique of NLP leaderboards. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4846–4853. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.393. URL https://aclanthology.org/2020.emnlp-main.393/.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4110–4124. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.324. URL https://aclanthology.org/2021.naacl-main.324/.

Terry K. Koo and Mae Y. Li. A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of Chiropractic Medicine*, 15(2):155–163, 2016. doi: 10.1016/j.jcm.2016.02.012. URL https://doi.org/10.1016/j.jcm.2016.02.012.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems*, volume 36, pp. 42330–42357. Curran Associates, Inc., 2023. doi: 10.52202/075280-1835. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/83fc8fab1710363050bbd1d4b8cc0021-Abstract-Datasets_and_Benchmarks.html.

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

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

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-long.556. URL https://aclanthology.org/2022.acl-long.556/.

Mathematical Association of America. AIME problems and solutions, 2024. URL https://maa.org/.

Mathematical Association of America. AIME problems and solutions, 2025. URL https://maa.org/.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong (Ryan) Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Christopher Rytting, Ryan Marten, Yixin Wang, Jenia Jitsev, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *The Fourteenth International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026. URL https://openreview.net/forum?id=a7Qa4CcHak.

Evan Miller. Adding error bars to evals: A statistical approach to language model evaluations. arXiv preprint arXiv:2411.00640, 2024. URL https://arxiv.org/abs/2411.00640.

Ronald Oaxaca. Male-female wage differentials in urban labor markets. *International Economic Review*, 14(3):693–709, 1973. doi: 10.2307/2525981. URL https://doi.org/10.2307/2525981.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simon Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, Natalie Kim, Samuel Miserendino, Gildas Chabot, David Li, Patrick Chao, Michael Sharman, Alexandra Barr, Amelia Glaese, and Jerry Tworek. GDPval: Evaluating AI model performance on real-world economically valuable tasks. In *The Fourteenth International Conference on Learning Representations*, volume 2026, pp. 24005–24040, 2026. URL https://openreview.net/forum?id=hcuEdq6eKD.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5389–5400. PMLR, 2019. URL https://proceedings.mlr.press/v97/recht19a.html.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? In *Advances in Neural Information Processing Systems*, volume 36, pp. 55565–55581. Curran Associates, Inc., 2023. doi: 10.52202/075280-2425. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/adc98a266f45005c403b8311ca7e8bd7-Abstract-Conference.html.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations*, volume 2024, pp. 25055–25083, 2024. URL https://openreview.net/forum?id=RIu5lyNXjT.

Patrick E. Shrout and Joseph L. Fleiss. Intraclass correlations: Uses in assessing rater reliability. *Psychological Bulletin*, 86(2):420–428, 1979. doi: 10.1037/0033-2909.86.2.420. URL https://doi.org/10.1037/0033-2909.86.2.420.

Anton Voronov, Lena Wolf, and Max Ryabinin. Mind your format: Towards consistent evaluation of in-context learning improvements. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 6287–6310. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.375. URL https://aclanthology.org/2024.findings-acl.375/.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094, 2024. doi: 10.52202/079017-1650. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12697–12706. PMLR, 2021. URL https://proceedings.mlr.press/v139/zhao21c.html.

A Appendix

A.1 The seven dimensions, with rendered examples

Table 3 lists the seven dimensions and their levels. The rest of this section shows what each level actually changes. Every example is the text the engine renders for the code domain; the anchor is shown first, and each variant is shown as the lines that differ from the anchor, with the changed span highlighted. Lines not shown are identical to the anchor.

The anchor. The system prompt every variant below is compared to:

```
You are a Python programmer solving a function-implementation task over
     multiple turns with a unit-test harness.

- Conduct your reasoning inside <think>...</think> blocks before each
    action.
- To submit candidate implementation and see the failing assertions,
    wrap the complete function in <code>...</code> and stop.
- When confident, output the final implementation inside <solution
    >...</solution>.
```

Dimension	Levels	What the rewrite does
form	3	renders the task input verbatim, inside a markdown fence, or with whitespace collapsed
think	3	asks for reasoning in `<think>` tags, in plain step-by-step text, or in Thought/Action lines
action	4	names the submission channel: two synonymous tag pairs (`<solution>` vs. `<answer>`), a fenced code block, or a native tool call
reminder	2	shows or omits the remaining-turn countdown
wording	2	swaps a synonym set in the prompt ("SQL expert" vs. "SQL specialist")
case	2	upper- or lower-cases the tag and tool names
order	3	permutes the three rule clauses

Table 3: The seven dimensions. Each rewrite changes the presentation of the same instruction: no level adds information, examples, or turns, and none changes what a correct answer is.

```
- You have at most 5 test submissions before you must give the final
    answer.
```

The task input under the anchor (`form=canonical`), for a toy item:

```
Complete this function:

def add(a, b):
    """Return a + b."""
```

After each action, the model receives an observation. Under the anchor (`reminder=on`) the observation ends with a countdown:

```
<observation>All visible tests passed.
<reminder>You have 3 turns left.</reminder></observation>
```

form. Only the task input changes. Under `fenced`:

```
Complete this function:

```python
def add(a, b):
    """Return a + b."""
```
```

Under `flat`, runs of blanks collapse to one space:

```
Complete this function:

def add(a, b):
 """Return a + b."""
```

think. The first rule changes. Under `cot`:

```
- Reason step by step in plain text before each action.
```

Under `react`:

```
- Use the format: Thought: <your reasoning>
Action: <your action> for each step.
```

action. The second and third rules change together, because they name the probe channel and the submit channel. Under `sql_answer` (the one-word rewrite of Figure 1), only the submit tag changes:

```
- When confident, output the final implementation inside
    <answer>...</answer> .
```

Under `codeblock`, both channels become fences:

```
- To submit candidate implementation and see the failing assertions,
   wrap the complete function in ```python...``` and stop.
- When confident, output the final implementation inside
    ```answer...``` .
```

Under `native_fc`, both rules become tool calls and the tools are passed as a schema rather than described in text:

```
- Call the `run_tests` tool to submit candidate
  implementation and see the failing assertions.
- Call the `submit_solution` tool to output the
  final implementation.
```

reminder. Under `off`, the observation has no countdown:

```
<observation>All visible tests passed.</observation>
```

The turn budget is unchanged; the model is simply not told the count.

wording. Five noun phrases swap to synonyms. Under `v2`:

```
You are a Python developer solving a function-implementation task over
    multiple turns with a unit-check harness.

- Conduct your reasoning inside <think>...</think> blocks before each
   action.
- To submit candidate routine and see the failing assertions, wrap the
    complete function in <code>...</code> and stop.
- When confident, output the concluding routine inside <solution>...</
   solution>.
- You have at most 5 check submissions before you must give the final
   answer.
```

case. Tag names upper-case. Under `upper`:

```
- To submit candidate implementation and see the failing assertions,
   wrap the complete function in <CODE>...</CODE> and stop.
- When confident, output the final implementation inside
    <SOLUTION>...</SOLUTION> .
```

For `native_fc` the tool names upper-case instead (`RUN_TESTS`, `SUBMIT_SOLUTION`). Fences have no letter case and are unaffected.

order. The three rules permute; no rule is reworded. Under `submit_first`:

```
- When confident, output the final implementation
  inside <solution>...</solution>.
```
```

```
- Conduct your reasoning inside <think>...</think> blocks before each
    action.
- To submit candidate implementation and see the failing assertions,
    wrap the complete function in <code>...</code> and stop.
```

Under `action_first` the order is probe, submit, think.

A.2 THE VARIANCE DECOMPOSITION

Let y_{mtc} be the accuracy of model m on task t under configuration c, with $M = 24$ models, $T = 3$ tasks, and $C = 32$ configurations, every combination present. Write $\bar{y}$ for the grand mean and $\bar{y}_m$, $\bar{y}_t$, $\bar{y}_c$ for the means over all cells sharing that model, task, or configuration; $\bar{y}_{mt}$, $\bar{y}_{mc}$, $\bar{y}_{tc}$ for the means over cells sharing a pair. Each cell's deviation from the grand mean splits as

$$\begin{aligned} y_{mtc} - \bar{y} = {} & (\bar{y}_m - \bar{y}) + (\bar{y}_t - \bar{y}) + (\bar{y}_c - \bar{y}) \\ & + (\bar{y}_{mt} - \bar{y}_m - \bar{y}_t + \bar{y}) + (\bar{y}_{mc} - \bar{y}_m - \bar{y}_c + \bar{y}) \\ & + (\bar{y}_{tc} - \bar{y}_t - \bar{y}_c + \bar{y}) + r_{mtc}, \end{aligned}$$

where r_{mtc} is the residual. Because the design is fully crossed, the seven terms are orthogonal, and summing squares over all MTC cells gives

$$\begin{aligned} \mathrm{SS}_{\text{total}} = {} & \mathrm{SS}_M + \mathrm{SS}_T + \mathrm{SS}_C \\ & + \mathrm{SS}_{MT} + \mathrm{SS}_{MC} + \mathrm{SS}_{TC} + \mathrm{SS}_{\text{res}} \end{aligned}$$

with, for example, $\mathrm{SS}_M = TC \sum_m (\bar{y}_m - \bar{y})^2$ and $\mathrm{SS}_{MC} = T \sum_{m,c} (\bar{y}_{mc} - \bar{y}_m - \bar{y}_c + \bar{y})^2$. The share reported for each term is its sum of squares divided by $\mathrm{SS}_{\text{total}}$. On the sealed grid the shares are: model 61.8%, task 22.4%, configuration 0.8%, model–task 6.6%, model–configuration 3.0%, task–configuration 0.6%, residual 4.8%. To check that no single model drives these shares, we recompute them 24 times, each time leaving one model out. Across those 24 recomputations the model effect ranges from 57.4% to 63.2%, the task effect from 21.3% to 26.0%, the configuration effect from 0.7% to 1.0%, and the model–configuration interaction from 2.7% to 3.3%; the ratio of interaction to configuration effect stays between 3.2 and 4.0.

The claim in Section 3.1 that the five most variable units are all code units is tested in Appendix A.4.

A.3 ADDITIONAL RANKING STATISTICS

Friedman and Nemenyi. A Friedman test with Nemenyi post-hoc comparisons (Demšar, 2006), with configurations as blocks, finds no adjacent pair in the mean-rank order that exceeds the critical difference at $\alpha = 0.05$, in any domain, and about 47% of all model pairs are likewise below the threshold. We treat the test as exploratory, because the configurations are a designed grid on a shared item set rather than independent samples.

Reversals by band distance. We group model pairs by how many bands apart they are. On code, the per-instance reversal rate is 19.2% for pairs in the same band, 18.5% one band apart, 10.9% two apart, and 5.3% three apart. The difference between near pairs (zero or one band apart) and far pairs (two or three) is $+10.1$ points on text-to-SQL, $+5.7$ on math, and $+9.5$ on code, with 95% confidence lower bounds of $+9.3$, $+5.0$, and $+9.0$. If we restrict to pairs within three rank positions of each other, the same-band and cross-band rates become similar. This suggests that what drives reversals is closeness in score rather than band membership as such, which we state as a hypothesis, since we have not yet run a gap-controlled analysis.

A.4 THE CONCENTRATION OF VARIANCE

Section 3.1 reports that the five most variable units account for 40% of the total variance across the 72 units. All five are code units. That claim is tested against the null that variance is unrelated to task, and there are 24 code units among 72, so under that null the probability that the five largest-variance units are all code is $\binom{24}{5} / \binom{72}{5} = 0.003$; a permutation of task labels over the 72 units (100,000 draws) gives the same value. The largest single unit is GPT-5.6 on code, with a standard deviation of 25 points.

A.5 How many configurations a benchmark needs

Generalizability theory (Cronbach et al., 1972) asks how many configurations a benchmark needs to average over to reach a given reliability. A D-study on the code variance components, under the assumption that configurations are sampled independently, predicts a relative generalizability coefficient of $G(3) = 0.92$ for a mean over three configurations, against $G(1) = 0.79$ for one. This projection is for the full 24-model pool; within-band reliability would need its own D-study.

A.6 Correctness flags and the signed decomposition

The decomposition in Section 4.4 uses each run's original correctness flag, while the main table uses the flag resolved after reruns. The two disagree on 2,258 raw records, which changes 232 of the 768 code cells at two decimal places. The identity still holds within the decomposition, and every score reported in the paper is a main-table score.

A.7 Numbers behind Figure 2a and 2c

Table 4 gives the effect of each single-dimension rewrite, the numbers plotted in Figure 2a, and Table 5 gives the two quantities behind Figure 2c.

Dimension	Rewrite applied to the anchor	median $\lvert\Delta\rvert$	max $\lvert\Delta\rvert$	unit at the max
action	submit via a native tool call instead of tags	4.8	55.9	Qwen3-1.7B-SFT, math
	submit inside a fenced code block instead of tags	2.1	21.3	Qwen2.5-Coder-14B, code
	rename the final tag `<solution>` to `<answer>`	1.5	**79.3**	GPT-5.6, code
think	reason in Thought/Action lines instead of `<think>` tags	3.0	32.7	Qwen2.5-7B, math
	reason in plain text instead of `<think>` tags	2.8	33.5	Qwen2.5-Coder-32B, code
order	list the rules as probe, submit, think	1.7	**89.0**	Qwen2.5-Coder-32B, code
	list the rules as submit, think, probe	1.5	27.4	Qwen2.5-Coder-32B, code
form	wrap the task input in a markdown fence	1.6	**43.3**	Qwen2.5-Coder-14B, code
	collapse runs of blanks in the task input	1.3	9.8	Qwen3-0.6B, code
wording	swap five nouns for synonyms	1.8	11.0	Qwen3-1.7B, code
case	upper-case the tag names	1.6	23.8	Qwen2.5-Coder-7B, code
reminder	drop the turn countdown	1.1	11.0	Qwen2.5-Coder-14B, code

Table 4: Effect of changing one dimension of the anchor configuration. For each of the twelve single-dimension rewrites, the median and the maximum absolute change in accuracy across the 72 model–task units, in percentage points, and the unit where the maximum occurs. Every median is below 5 points; three maxima above 40 are in bold. Appendix A.1 shows the rendered text of each rewrite; Figure 2a plots these alongside the 19 multi-dimension rewrites.

A.8 Anchor audit

Every signed term in Section 4 is measured relative to the anchor configuration, so Table 6 reports, for every unit, how far the anchor sits from the unit's own median. Sixty-eight of 72 units are within seven points.

A.9 Threshold sensitivity of the signed shares

The shares in Section 4.4 are computed over cells that fall more than seven points below the anchor. Table 7 recomputes them as the threshold moves from 4 to 15 points. The code shares do not move; the math and text-to-SQL splits do, which is why the main text always states the threshold when it reports them.

Band	Domain	spread (pp)	harness std	ratio	ICC
≤2B	Text-to-SQL	36.3	4.9	7.4	0.9
≤2B	Math	67.8	5.9	11.5	0.95
≤2B	Code	49.4	10.1	4.9	0.89
3–8B	Text-to-SQL	42.3	7.5	5.6	0.83
3–8B	Math	79.0	6.8	11.6	0.95
3–8B	Code	72.0	11.1	6.5	0.79
14–32B	Text-to-SQL	3.2	2.8	1.1	0.31
14–32B	Math	27.4	2.9	9.4	0.95
14–32B	Code	14.3	12.9	1.1	0.35
frontier API	Text-to-SQL	11.0	1.7	6.6	0.85
frontier API	Math	10.0	2.3	4.3	0.69
frontier API	Code	7.3	7.6	1.0	0.29

Table 5: The two quantities behind Figure 2c. For each band and task: the spread between the members' median accuracies, the mean per-model standard deviation across configurations, their ratio, and the within-band ICC(3,1).

	Model	Text-to-SQL	Math	Code
A	Claude Opus 4.6	+1.3	+0.3	+0.0
	Claude Sonnet 4.6	-0.5	-1.1	-1.2
	GPT-5.6	+0.3	+1.2	+0.6
	DeepSeek V4 Flash	+1.0	+1.1	+0.0
	GLM-5.3 Flash	+1.5	+0.3	+0.0
	Doubao Seed 2.0 Pro	+0.7	-1.5	+1.2
B	Qwen3-32B	+1.5	+1.5	+0.3
	Qwen2.5-Coder-32B	+1.7	-6.9	+3.4
	Qwen3-14B	+0.3	+0.5	+0.0
	Qwen2.5-Coder-14B	+1.5	-1.6	**-11.3**
C	Qwen3-8B	-2.7	-0.5	+3.0
	Llama-3.1-8B	+1.3	+0.4	+4.9
	Qwen2.5-7B	**+7.5**	-0.3	-3.7
	Qwen2.5-Coder-7B	-0.8	-2.5	**-25.0**
	Mistral-7B-v0.3	+0.2	-1.3	-3.7
	MARS-SQL-7B	+2.8	-0.8	+2.1
	Qwen3-4B-2507	+1.5	+1.1	+0.0
	Qwen2.5-Coder-3B	+3.7	+0.5	**-21.0**
	Llama-3.2-3B	+0.2	-1.6	-0.6
D	Qwen3-1.7B	-0.8	+0.4	+1.8
	Qwen3-1.7B-SFT	+0.8	+0.8	+1.8
	Llama-3.2-1B	+2.2	-6.0	-1.5
	Qwen3-0.6B	+1.0	+4.0	+4.9
	Qwen2.5-Coder-0.5B	+1.5	+1.5	+0.9

Table 6: Anchor health. For every model–task unit, the anchor configurations accuracy minus the median accuracy of that units 32 configurations, in percentage points. Bands A–D as in Table 1. Entries more than 7 points from zero are in bold; there are 4, three below (the Qwen2.5-Coder 3B, 7B, and 14B code units) and one above (Qwen2.5-7B on text-to-SQL). Signed terms in Section 4 for the three below-median units are measured against a weak reference.

A.10 Cross-domain transfer of the eight-configuration audit

Eight configurations selected on two domains closely recover full-grid sensitivity rankings on a third. We test this with three leave-one-domain-out folds on the frozen 32-configuration grid. In each fold, we search for the eight configurations that maximize the Spearman correlation between subset-based and full-grid sensitivity across the 48 model–domain units in the two selection domains. Sensitivity is the sample standard deviation of accuracy across configurations. Each candidate includes the anchor and covers every level of all seven scaffold dimensions, giving 299,473 feasible subsets to

Threshold	Code cells	Code D/F/C	Math cells	Math D/C	Text-to-SQL cells	Text-to-SQL D/C
$>$ 4pp	222	45/40/16	102	29/71	147	86/14
$>$ 5pp	184	45/40/15	83	32/68	122	85/15
$>$ 7pp	154	45/40/15	57	38/62	86	84/16
$>$ 10pp	114	45/40/15	48	39/61	54	85/15
$>$ 15pp	83	46/40/14	28	45/55	29	93/7

Table 7: Signed loss shares as the deviation threshold varies. D/F/C = Delivery/Format/Content share of the loss in percent; math and text-to-SQL have no Format term. Code shares are threshold-stable; the math and SQL splits move with the threshold and are always reported with it.

Selection domains	Held-out domain	Selected ρ	Coverage baseline Median	Coverage baseline 2.5–97.5% range
Code + Math	SQL	0.9609	0.9157	[0.7887, 0.9652]
Code + SQL	Math	0.9678	0.9496	[0.8991, 0.9765]
Math + SQL	Code	0.9191	0.8948	[0.7513, 0.9661]

Table 8: Cross-domain transfer of the eight-configuration selection procedure. Each subset is selected using two domains and evaluated on the third. Spearman correlations compare the selected subset's sensitivity ranking with that of all 32 configurations across the 24 models in the held-out domain. SQL denotes text-to-SQL. The coverage baseline is the uniform distribution over all 299,473 subsets satisfying the same anchor and dimension-coverage constraints. Its range describes variation across subsets, not a confidence interval over models.

search exhaustively. We freeze the selected subset before evaluating its sensitivity ranking across the 24 models in the held-out domain. All correlations use average ranks for ties.

Table 8 reports held-out correlations of 0.961 on SQL, 0.968 on math, and 0.919 on code. For comparison, we evaluate every feasible subset on the held-out domain, giving the exact performance distribution of a uniformly sampled subset under the same anchor and coverage constraints. The selected subsets exceed this baseline's median in all three folds. Coverage alone also yields high agreement, with median correlations ranging from 0.895 to 0.950.

These results support cross-domain transfer of the eight-configuration selection procedure within the evaluated model pool. Each fold selects a different subset, and all folds share the same 24 models; generalization to unseen models or model families remains to be evaluated.

A.11 No gold harness

The claim in Section 7 that no configuration treats all models alike rests on the effect profiles in Table 9. A configuration's effect on a model is that model's accuracy under the configuration minus its median accuracy across all 32; the profile is the vector of those 24 effects, and its width is the largest minus the smallest.

Domain	narrowest profile (pp)	widest (pp)	uniform-sign configs
Text-to-SQL	6.2	48.7	0 of 32
Math	8.5	61.4	0 of 32
Code	12.2	89.6	0 of 32

Table 9: No gold harness: per-configuration effect profiles across the 24 models. Effect of a configuration on a model is its score minus that model's cross-configuration median; a profile's width is its cross-model spread. No configuration moves twelve or more models in one direction with none opposite.

A.12 LIMITATIONS

We ran one seed at temperature zero, except for GPT-5.6, which rejects temperature zero and runs at the provider default (Appendix A.13). Each domain uses one fixed item set. The L3 oracle is a conservative upper bound. The multi-turn counterfactual replay is incomplete, so in the BFCL domain the tax is folded into coverage, as noted there. The SQL and Python extraction paths differ in strictness, and the zero tax on math and SQL comes from wrapper-tolerant judges, not from model behavior. The coverage term does not distinguish code that was never written from code that was written and not submitted; the substantive- correctness probes bound this but do not resolve it. The code corpus is 27% claimed and 73% rerun episodes, with the source of each row released. The band cut points were fixed in advance, but they are one particular choice, and the frontier band is defined by serving, not by parameter count.

A.13 PER-MODEL DISCLOSURES

Three models depart from the ideal in ways that bear on individual dimensions. We record them here rather than in the main text because each affects one model, but each one qualifies a claim the main text makes about a dimension.

GLM-5.3-Flash returns uncontrollable `reasoning_content` alongside its answers, so its `think` dimension is partially confounded by provider-side reasoning that the harness does not set.

The Llama-3.1-8B chat template has no tools branch, so its `native_fc` configurations are carried by the text-level protocol rather than by true native tool calling. Its cells under that dimension therefore measure the fallback path, and the `action` dimension is not fully crossed for this model.

GPT-5.6 rejects temperature zero and runs at the provider default instead, recorded per row in `temperature_note`. It is the only model in the pool for which the fixed-temperature statement of Appendix A.12 does not hold.