THE TURN PARTITION IS A FREE PARAMETER IN MULTI-TURN LLM EVALUATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models often score lower when a complete request is split across several conversation turns. Measuring this gap requires choosing how to turn the request into a conversation. We study how that choice affects the conclusions of a multi-turn evaluation. Using four tasks from the sharded benchmark of Laban et al. (2026), we evaluate five open-weight models across 24 conversation designs. These designs vary the number of turns, requirement order, the amount of information provided initially, and whether an earlier requirement is repeated, while preserving the final requirements and evaluation checker. We observe that conversation design substantially changes measured performance: the best and worst designs differ by a median of 0.123 on a 0-1 scale. Gaps between models can be small relative to their variation across designs, and the gains from four mitigation methods vary across designs, sometimes changing sign. When the same content is given in a single message, performance still varies across designs, indicating that design sensitivity is not exclusive to interactive conversations. These results show that conclusions about multi-turn performance and mitigation effectiveness are conditional on the conversation design used to evaluate them. We recommend evaluating across multiple explicitly specified designs and reporting variation alongside average performance.

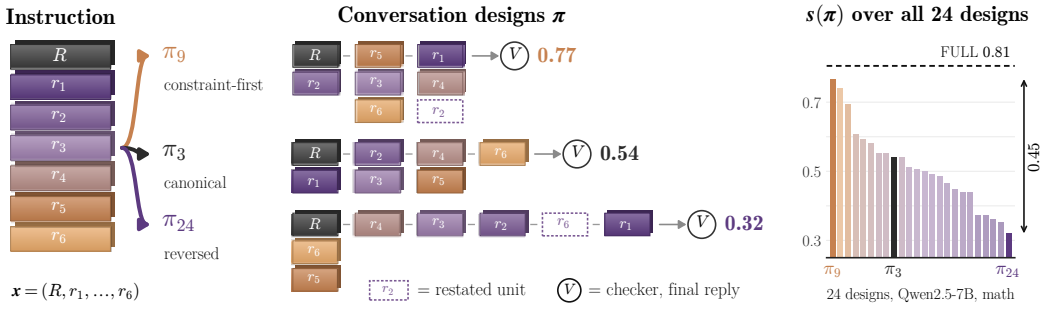

Figure 1: **A multi-turn score depends on how the request is split.** One instruction x (left) is delivered to Qwen2.5-7B under three of the 24 conversation designs (middle); every design uses the same requirements, instances, and checker V, and only the turn split changes. Across all 24 designs on math (right), the score ranges from 0.32 to 0.77, a spread of 0.45, and every design scores below sending the full request in one message (FULL, 0.81). π_3 is the reference design π_0.

1 INTRODUCTION

People rarely give a complete specification in one message. They add details, correct earlier statements, and repeat important constraints. A useful language model should be able to follow this process. Yet most standard evaluations give the model one complete prompt.

Laban et al. (2026) study this mismatch by splitting a complete instruction into smaller pieces, called *shards*. The shards are revealed over several turns. They find that models perform much worse in this setting than when they receive the full instruction at once. Their analysis shows a common failure

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.

pattern: a model answers too early, makes an incorrect assumption, and then does not fully recover when new information arrives.

This setup makes multi-turn evaluation more controlled, but it also introduces a new choice. The evaluator must decide how to turn one instruction into a conversation. How many turns should there be? Which requirements should appear together? Which one should come first? Should an important requirement be repeated? We call one answer to these questions a *conversation design*.

A score measured under one conversation design need not generalize to other designs. The same applies to model comparisons and mitigation methods such as recapitulation and snowballing. Evaluating a single design leaves this dependence unmeasured.

We test this distinction directly. We keep the task instances, requirement text, model settings, and programmatic checkers fixed. We change only the way the same requirements are delivered. Figure 1 summarizes the idea.

Our experiments show that **conversation design is an evaluation variable, not merely an implementation detail**. Even when the task, requirements, model, and checker are fixed, changing how the same requirements are ordered and distributed across turns substantially changes measured performance. Across the 11 settings with reliable answer extraction, the difference between better- and worse-performing designs has a median of 0.123.

Model rankings can depend on the conversation design. In four of eight model comparisons, the direction of the performance difference changes across designs, such that the model with lower average performance scores higher under some designs. Although these reversals are small and do not consistently replicate on new instances, they indicate that model rankings can be sensitive to the conversation design under which they are measured. **Mitigation gains also depend on the design**: the same method can appear substantially more or less effective depending on how the conversation is constructed, and in 10 of 16 task–method comparisons its gain changes systematically across designs.

Finally, **design sensitivity is not limited to multi-turn interaction**. When all requirements are presented in a single message, while preserving the same order and grouping used in the corresponding conversations, scores still differ across designs, and designs that perform poorly in multi-turn conversations also tend to perform poorly in the single-message condition. Presenting those same requirements across multiple turns lowers performance further in every reliable setting, and the size of this additional drop depends on the design in six of the eleven settings.

Together, these results show that **the measured multi-turn gap is conditional on how the evaluator constructs the conversation**. A benchmark based on one conversation design therefore provides evidence about that design, not automatically about multi-turn performance more broadly. We recommend making conversation design explicit and, when broader claims are intended, reporting results across several specified and explicitly weighted designs.

2 Related Work

Early multi-turn benchmarks often use related but separable questions at each turn (Zheng et al., 2023; Bai et al., 2024; Kwan et al., 2024). The sharded setup of Laban et al. (2026) instead starts from one complete task and gradually reveals the information needed to solve it, enabling a direct comparison between single-turn and multi-turn performance. Later work examines similar failures in specialized settings and proposes new training or prompting methods (Guo et al., 2026; Li et al., 2026; Zheng et al., 2026; Luo et al., 2026; Chen et al., 2026).

Several methods aim to prevent a model from becoming anchored to an early answer. Recapitulation repeats all requirements at the end, whereas snowballing repeats all requirements seen so far at every turn (Laban et al., 2026). Other approaches reset the conversation when uncertainty is high (Khalid et al., 2025), remove or reduce the influence of the model's earlier responses (Huang et al., 2026), or teach the model to wait for more information (Li et al., 2025; Zhang et al., 2025). These methods are typically evaluated under a single conversation design. We instead ask whether their measured gains remain stable as the design changes.

A closely related line of work shows that a model's answer depends on the order in which evidence appears *within a single prompt*: information placed in the middle of a long context is used less

FULL: the original instruction

A print shop charges $0.08 a page for the first 200 pages and $0.05 a page after that. Mia prints a 350-page manuscript and buys a $12 binder. A coupon takes 15% off the printing only. What does she pay in total?

π_{coarse}: canonical, $T = 2$

1. What does Mia pay in total? It's a 350-page manuscript, and the first 200 pages are $0.08 each.
→ first answer attempt, r_1, r_2, r_3

2. After 200 pages it drops to $0.05. She also buys a $12 binder, and the coupon is 15% off the printing only, not the binder.

π_{fine}: canonical, $T = 6$

1. What does Mia pay in total?
→ first answer attempt, r_1 only
2. It's a 350-page manuscript.
3. The first 200 pages are $0.08 each.
4. After 200 pages it drops to $0.05.
5. She also buys a $12 binder.
6. The coupon is 15% off the printing only, not the binder.

Figure 2: **Two request-first conversations made from the same instruction.** Both contain all six requirements by the final turn. The two-turn design gives the model half of the specification before its first answer. The six-turn design gives it only the question, followed by five chances to revise.

reliably (Liu et al., 2024), and audits of order sensitivity find that the same evidence can yield different answers (Paruchuri et al., 2026; Lin et al., 2026). Our question differs in two ways. Those studies permute items inside one message and report the change in accuracy; we hold the requirement set, the task instance and the checker fixed and vary how the same request is *partitioned across turns*, a choice an evaluator must make and that a single-prompt study cannot expose. And the outcome we track is not accuracy alone but the conclusions a benchmark draws from it: which model appears better, and how much a mitigation appears to help. Order is one of our four factors, and Section 5.3 finds it the dominant one on math and database; the claim here is that the turn partition is a free parameter of the evaluation, not that order matters.

This question connects multi-turn evaluation to a broader literature on evaluation sensitivity. Single-turn scores can change with paraphrasing, formatting, and answer order (Sclar et al., 2024; Mizrahi et al., 2024; Alzahrani et al., 2024; Salinas & Morstatter, 2024). More generally, evaluation is itself a measurement process, and benchmark-design choices can introduce additional variation (Cronbach et al., 1972; Miller, 2024; Bean et al., 2025). We apply this perspective to a choice specific to multi-turn evaluation: how a complete request is ordered and divided across turns.

3 Conversation Design as an Evaluation Variable

3.1 From one request to many conversations

Each task instance contains a full instruction and a list of atomic requirements. Every design uses the same unique requirements and the same final checker; only the delivery changes. In a two-turn design half of the specification arrives before the first answer, while in a six-turn design the first answer rests on the question alone and five later turns ask the model to revise it (Figure 2).

Formally, a task instance x carries a requirement set $\mathcal{R}(x) = \{r_1, \ldots, r_K\}$ of atomic units. A programmatic checker V maps a final answer y to a score $V(y) \in [0, 1]$. A conversation design π first orders the requirements and then partitions them into T consecutive blocks,

$$\pi(\mathcal{R}(x)) = (B_1^\pi, \ldots, B_T^\pi), \qquad \bigcup_{t=1}^{T} B_t^\pi = \mathcal{R}(x). \tag{1}$$

Turn t delivers block B_t^π. Four choices fix a design. **Granularity T:** the instruction is spread over 2, 4, or 6 turns. **Front-loading f:** 15%, 30%, or 45% of the requirements appear in the first turn. **Order:** $\mathcal{R}(x)$ uses the canonical order, reverse order, constraint-first order, or a random order. *Constraint-first* places every unit that states a constraint ahead of every unit that does not, keeping source order inside each group; because the overall request is the only non-constraint unit, it moves that unit to the end of the ordered sequence. **Restatement:** one earlier requirement is either repeated later or not repeated. All four are choices an evaluator must make, and none of them changes the set of requirements the model must satisfy by the final turn.

Following Laban et al. (2026), every design opens with the overall request in the first turn. This keeps comparisons between designs about how the remaining requirements are distributed, rather than about whether the model knows the goal of the task from the start. The order factor therefore permutes only the requirements that follow the request.

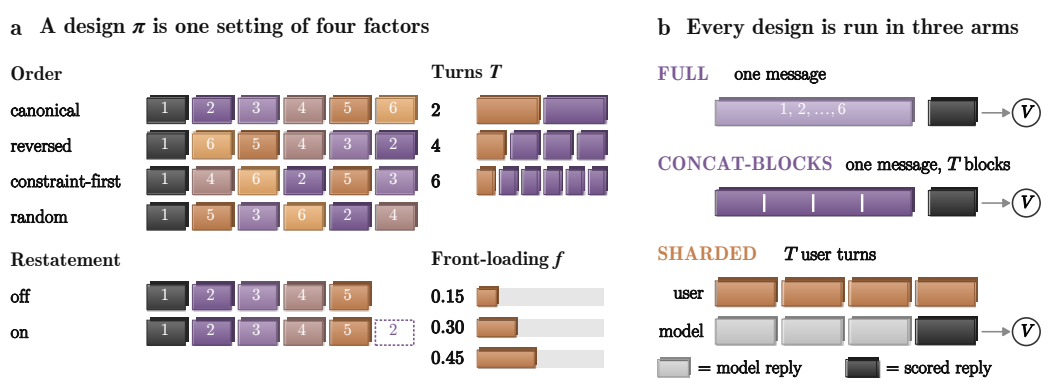

Figure 3: **Design factors and evaluation conditions.** The matched CONCAT-BLOCKS control keeps the same order and block boundaries as the interactive SHARDED condition.

3.2 THREE EVALUATION CONDITIONS

Figure 3 shows the three conditions used in the study. FULL gives the original complete instruction in one message. CONCAT-BLOCKS puts the designed blocks into separate paragraphs of a single message, preserving their order and grouping while removing the interactive loop. SHARDED instead sends the blocks over several user turns, so the model sees its own earlier responses before producing the final answer.

This matched control shows whether design sensitivity also appears when all requirements arrive in one message. The paired difference between CONCAT-BLOCKS and SHARDED measures the additional effect of delivering the same blocks over turns.

4 EXPERIMENTAL SETUP

4.1 TASKS AND MODELS

We use four tasks from the public release of Laban et al. (2026), each scored by a programmatic checker: math (exact numeric match), database (query execution), tool selection (structured call match), and code (unit tests). For math, tool selection and code we take the first 50 instances with at least six requirement units and use $T \in \{2, 4, 6\}$. Database instances are shorter, so we lower the threshold to five, which admits all 33 database instances, and use $T \in \{2, 4\}$. Unlike the adaptive user simulator in Laban et al. (2026), our harness delivers requirement text literally and scores the final turn. Our results describe this controlled protocol rather than a numerical reproduction of their simulation.

Crossing the four factors (granularity, order, front-loading and restatement) gives $3 \times 3 \times 4 \times 2 = 72$ possible designs, but not all of them are usable on these task instances. Some require more turns than an instance has requirements, and on short instances different settings can produce the same conversation. Consequently, only 48–53 designs are distinct on math, tool selection, and code, and only 20 are distinct on database. We shuffle the 72 designs with a fixed seed, discard designs that cannot be built or duplicate another design, and retain the first 24. The resulting family $\mathcal{D}$ covers every factor level. Database, drawn by the same procedure under its own constraints, has a separate family of 20 designs that shares nine with the 24. Appendix P gives both lists.

Let P denote the uniform distribution over $\mathcal{D}$ and report averages for $\pi \sim P$. Because this design-marginal estimand averages over the entire family, it does not require a single design to serve as a baseline. For interpretability, we nevertheless report one fixed reference design, π_0, beside the family-wide results. It uses the canonical order, $T = 4$, $f = 0.15$, and no restatement. By *canonical order*, we mean the order in which the atomic requirements are listed in the public release, after keeping the overall request first.

Five open-weight models are evaluated: Qwen2.5-7B-Instruct, Qwen2.5-14B-Instruct, Qwen2.5-32B-Instruct, Mistral-7B-Instruct-v0.3, and Gemma-2-9B-it. The 7B and 14B Qwen models and Mistral run on all four tasks, and Qwen2.5-32B on math and code, giving the 14 model–task settings behind our main estimates. Gemma-2-9B, from a different model family, runs the same grid on all four tasks as a replication and is discussed alongside the main result and in Appendix M. All decoding uses temperature 1.0 and three random seeds.

4.2 Mitigation methods

We implement four inference-time methods in one common harness: Recapitulation, which restates every requirement in the final turn, and snowballing, which repeats all earlier requirements in each new turn (Laban et al., 2026); entropy-triggered resetting, which restarts from a consolidated prompt when the model's uncertainty spikes (Khalid et al., 2025); and user-turn-only prompting, which drops the model's own earlier replies from its context (Huang et al., 2026).

4.3 Measures and statistical tests

For model m under conversation design π, let $S(m, \pi)$ denote the mean final-answer score. We measure design sensitivity primarily by the standard deviation of $S(m, \pi)$ across designs. We correct this quantity for variation caused by the finite number of instances and generations, so that it better reflects variation attributable to design. We also report a held-out comparison between two selected designs and, for descriptive purposes, the best-to-worst range.

For mitigation method I, we define its gain under design π as

$$G(I, \pi) = S(m + I, \pi) - S(m, \pi). \tag{2}$$

We summarize this gain across the design family using

$$\bar{G}(I) = \mathbb{E}_{\pi \sim P}[G(I, \pi)], \qquad \sigma_G(I) = \sqrt{\mathrm{Var}_{\pi \sim P}[G(I, \pi)]}, \tag{3}$$

the average gain and its standard deviation across designs. A method is both effective and stable when its average gain is large and its variation across designs is small. We also show the gain at the reference design π_0 for comparison, but our conclusions are based on the average over $\mathcal{D}$.

We test for variation across conversation designs using paired permutation tests that preserve the matching of instances and random seeds. For each (instance, seed) pair, the null treats the observed design labels as exchangeable, allowing us to test whether performance differs across designs while preserving the structure of the observed generations. We construct bootstrap intervals by jointly resampling designs and task instances, so the resulting intervals capture variation from both sources. Appendix J provides further details on this distinction, and Appendix O shows that the main design contrasts persist under re-generation.

To account for multiple comparisons, we control the false discovery rate using the Benjamini–Hochberg procedure at $q = 0.05$. We apply the correction separately to the 16 mitigation tests and the 14 tests comparing evaluation conditions reported in Section 5.7.

5 Results

5.1 The same model receives different scores under different designs

Changing the conversation design can change the measured score even when the task and requirements remain fixed (Figure 4). Across the 11 model–task settings with answer extraction of at least 0.5, the median bias-corrected standard deviation across designs is 0.067. To check whether the difference between high- and low-scoring designs persists beyond the instances used to identify them, we select the best and worst designs on half of the instances and compare them on the other half. Their median difference is 0.123, and the difference is positive in 10 of the 11 settings.

The raw best-to-worst range is larger, with a median of 0.273, because the same instances are used both to identify and evaluate the extremes. If designs differed only because of sampling error, the expected median range would still be 0.169. The observed range is at least 1.30 times this noise

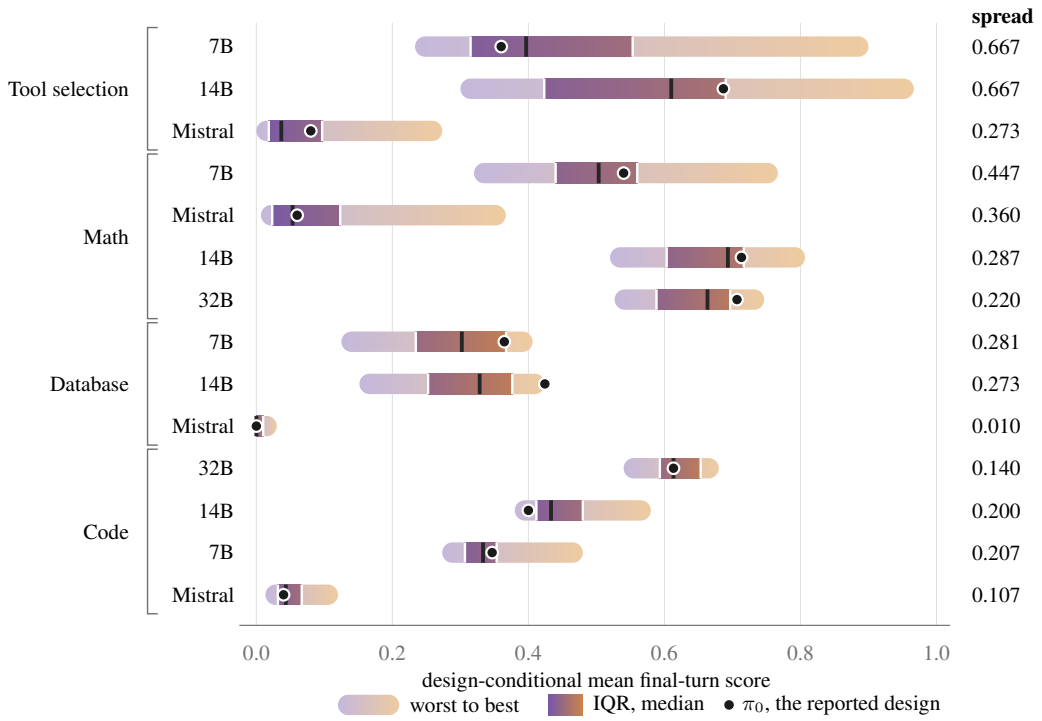

Figure 4: **Scores across conversation designs.** Database uses 20 designs; all other tasks use 24. Each bar runs from the worst design (purple) to the best (orange); *spread* is best minus worst.

floor in 10 of the 11 reliable settings; Qwen2.5-32B on code is the exception, with a ratio of 1.01 (Appendix N). Paired permutation tests give nominal $p < 0.05$ in 12 of the 14 settings, including nine with $p \leq 0.001$. The exceptions are Mistral-7B on database, where the mean score is near zero, and Qwen2.5-32B on code, whose bias-corrected design standard deviation is 0.011 and whose held-out contrast is negative. We therefore do not establish an overall design effect for Qwen2.5-32B on code.

Across all 14 settings, the reference design scores above the median design in nine and below it in five. A result from a single design can therefore overstate or understate performance relative to the family of designs we evaluate.

5.2 How much of the multi-turn gap is the design?

Averaged over designs, FULL scores above SHARDED in all 11 reliable settings, with a median gap of 0.296. The largest gap occurs on database, where Qwen2.5-7B falls from 0.869 under FULL to 0.289 under SHARDED. None of the 256 observed design–setting pairs among the reliable settings closes the gap, although for Qwen2.5-14B on tool selection the best-performing design reduces it to 0.007.

The size of the gap nevertheless varies substantially across designs. At each setting's best-performing design, the median observed gap is $+0.106$, compared with $+0.401$ at its worst-performing design. Because these designs are selected as extremes on the same instances, this comparison exaggerates the difference expected on new instances; the held-out difference in Section 5.1 is the more conservative estimate. Table 9 reports the gaps for every setting, and Appendix E decomposes the gap into three descriptive contrasts.

5.3 Different factors matter for different tasks

We fit a joint linear model with all four design factors and report each factor's unique share of between-design variance, the part no other factor explains as well (Table 8). Order dominates math at every model size, with unique shares of 0.425, 0.675 and 0.689 at 7B, 14B and 32B, and it dominates database as well. For Qwen2.5-7B, turn count matters most on tool selection (0.552) and code (0.157),

and the four factors together explain 66–90% of the variation. Restatement has smaller unique shares (0.002–0.053). Because the family is unbalanced the factors are correlated and part of the explained variance is shared rather than belonging to any one, so these shares do not by themselves establish the effect of any one choice; Section 5.4 makes the comparison, by holding two choices fixed and varying the other two.

5.4 Order and restatement alone change the score

More turns and less information in the first turn may deliberately make a conversation harder. We therefore ask whether scores still change when both choices are held fixed. Within each (T, f) cell, every design has the same turn count, front-loaded share, and set of requirements; only their order and whether an earlier requirement is repeated differ.

Scores still vary within these cells. A permutation test that shuffles order and restatement labels within each cell gives nominal $p < 0.05$ in nine of the 11 settings with reliable answer extraction (Table 18; Appendix K). The median within-cell design standard deviation is 0.76 times the standard deviation across all designs. This compares two measures of score spread; it does not mean that order and restatement explain 76% of the variance.

The conditional test also gives $p = 0.027$ for Qwen2.5-32B on code, where we cannot establish an overall design effect. Because the tests ask different questions, we do not treat this isolated result as evidence for an overall effect in that setting.

5.5 Model gaps can be small relative to design variation

We ask whether conversation design can change which model appears better. We compare Qwen model sizes on the same designs across eight model-pair and task comparisons. A design reverses the ranking when it favors the model that loses on average across designs. Four comparisons have at least one point-estimate reversal (Table 1).

A reversal found on the observed instances may be a chance extreme, so we check it on held-out instances: in each of 400 random half-splits we select the reversing designs on one half and test them on the other, comparing the reproduce rate with the *base rate* of all designs (Table 13). Math 32B versus 7B reproduces at 0.55 against a base rate of 0.12, and tool selection 14B versus 7B at 0.31 against 0.12; the other five comparisons sit at their base rates. No comparison's excess excludes zero, and the two largest have intervals $[-0.05, +0.64]$ and $[-0.09, +0.55]$. In every comparison the held-out *score* difference of the selected designs still favours the model that wins on average, by $+0.030$ on math 32B versus 7B and $+0.073$ on tool selection. Four comparisons therefore show point-estimate reversals and none of them is shown to carry to new instances at our instance count. What the data support is that these gaps are small relative to their variation across designs, not that the rankings are reversed. The database comparison is an extractor artifact outright: under the deterministic extractor of Appendix I no design reverses at all (Table 1, ‡).

The average model gap relative to its variation across designs helps explain which rankings are vulnerable. Let Δ be the mean score difference between two models and σ_D the standard deviation of that difference across designs. For the true distribution over designs, Cantelli's inequality bounds the probability of a reversal by $\sigma_D^2/(\sigma_D^2 + \Delta^2)$. We report $|\Delta|/\sigma_D$ to summarize this relationship: all four comparisons with a ratio of at least 2.4 keep the same ranking, while all four with a ratio of 1.6 or less reverse under some designs. Four comparisons also have descriptive point-estimate reversals when turn count and front-loading are held fixed.

5.6 Mitigation gains depend on the design

This section tests whether a mitigation that helps under one conversation design also helps across other designs. On Qwen2.5-7B, averaged across tasks and designs, user-turn-only has the largest gain ($+0.178$), followed by snowballing ($+0.103$), recap ($+0.066$), and entropy reset ($+0.006$; Table 2). Nine of the 16 task–method average gains pass the reported multiple-testing correction under joint resampling.

Those averages can differ substantially from a result at one design. On tool selection, user-turn-only gains $+0.540$ at the reference design but $+0.353$ averaged across all 24 designs, and recap gains

Table 1: **Model comparisons that reverse under some designs.** Δ is the average score difference and σ_D is its standard deviation across designs. A reversal is a design at which the sign of the difference disagrees with the average; designs at which the two models score exactly the same are ties and are not counted as reversals. *Bound* is $\sigma_D^2/(\sigma_D^2 + \Delta^2)$, the largest fraction of designs that can disagree with the average ordering. Database runs on 20 designs. ‡ an artifact of the source benchmark's LLM extractor: under the deterministic extractor of Appendix I, $\Delta = +0.127$, $|\Delta|/\sigma_D = 2.85$, and no design reverses.

Task	Model pair	Δ	$\lvert\Delta\rvert/\sigma_D$	Reversals	Bound
Math	32B vs. 7B	+0.135	1.52	2/24	0.30
Tool selection	14B vs. 7B	+0.126	1.34	2/24	0.36
Math	14B vs. 32B	+0.028	0.55	6/24	0.77
Database‡	14B vs. 7B	+0.014	0.28	7/20	0.93

Table 2: **Mitigation gains averaged over tasks and designs.** A ratio near zero means the result changes greatly across designs. *Corrected* σ_G subtracts the sampling variance of a single design's gain (Appendix A). Range gives the min/max of $\bar{G}/\sigma_G$ across tasks; the negative lower end belongs to entropy reset (Table 4). Task-level results are in Appendix A.

Method	Mean gain $\bar{G}$	Design SD σ_G	corrected σ_G	$\bar{G}/\sigma_G$	Range across tasks
Entropy reset	+0.006	0.031	0.006	0.20	-0.30 to $+0.93$
Recap	+0.066	0.069	0.047	0.95	$+0.10$ to $+1.66$
Snowball	+0.103	0.075	0.051	1.37	$+0.04$ to $+2.25$
User-turn-only	+0.178	0.088	0.062	2.02	$+1.60$ to $+2.42$

$+0.273$ there against $+0.151$ on average, a factor of 1.8. On database, recap is -0.104 at the reference design and $+0.015$ on average, with an interval that includes zero: an evaluation at the reference design alone would have reported a harm where the design family shows no reliable effect in either direction (Appendix A).

A significant average gain does not show that the gain *varies* by design; that is a different null, $G(I, \pi) = g_I$ for every π. Testing it directly, by permuting design labels of the gain within each (instance, seed) pair, 11 of the 16 task–method pairs reject at a nominal $p < 0.05$ and 10 survive Benjamini–Hochberg across the 16 tests, the same correction we apply to the average gains; database snowballing ($p = 0.045$) is the one that does not. The *placebo* is a semantically null edit: it rewrites each requirement's surface form without changing what the checker requires, so a method-like intervention is applied while the task content is held fixed. It rejects in none of its four (Table 5). Part of this heterogeneity is mechanical: entropy reset cannot fire in two-turn designs, and methods that restate requirements have more to restate when more turns are used. Consistent with this, after bias correction 13 of the 16 task–method pairs retain a positive estimated spread across designs, while the placebo's spread falls to at most 0.004 (Appendix A).

The strongest method invites a direct comparison. On 7B math the SHARDED score is 0.510 and user-turn-only adds 0.156, which is 0.666 against a matched one-message score of 0.660. Paired on shared cells, user-turn-only and CONCAT-BLOCKS differ by at most 0.022 on every task, each interval covering zero (Table 6). Dropping the model's earlier replies therefore recovers the matched non-interactive score rather than repairing an interactive history; it removes the interaction penalty by removing the interaction.

5.7 How much of the variation needs the conversation

The one-message control also changes across designs and tends to find the same designs difficult. The correlation between design means in SHARDED and CONCAT-BLOCKS is positive in all 14 settings, with a median of $+0.57$ among the 11 reliable settings; five individual intervals exclude zero (Table 11). This shows that design sensitivity is present without earlier assistant replies. The correlation does not measure what fraction of the score difference is caused by arrangement.

The paired penalty for delivering the same blocks over turns, $I_\pi = C_\pi - S_\pi$, is positive on average in all 11 reliable settings (median $+0.078$). Whether it *varies* by design is a separate question, tested by the same within-(instance, seed) permutation on I_π: six of the 11 reliable settings reject at $p < 0.05$, both tool-selection settings and 7B math at $p < 0.001$ (Table 19). In the other five, including code at 7B and 32B and database at 7B, the penalty is not shown to depend on the design. A different comparison asks whether SHARDED has *more marginal design variance* than CONCAT-BLOCKS. None of the 14 corresponding tests survives multiple-testing correction (Appendix G).

6 RECOMMENDATIONS

A multi-turn score is defined only once the conversation design is specified. We recommend that benchmarks (i) disclose the turn count, front-loading, requirement order and restatement they use, together with the procedure that generates them; (ii) vary the choices they are not studying, since more instances under one arrangement cannot detect arrangement effects; (iii) report the design-marginal mean, the design standard deviation, the number of designs and the weighting distribution, and for mitigations the mean and standard deviation of the gain; (iv) include a matched one-message control and examine the paired penalty from delivering the same blocks over turns; and (v) report model gaps relative to the design standard deviation, $|\Delta|/\sigma_D$, as a warning that a ranking may not hold under another design.

7 LIMITATIONS

Our conclusions are conditional on the design family we study. The 24 designs are a random draw from the factorial rather than a balanced array, so factor effects are partly confounded. The instance filter retains 183 of 627 instances and may favor longer requests. Most evidence comes from Qwen2.5 models; a replication on Gemma-2-9B supports the main findings (Appendix M), but frontier API models are untested. Our literal-delivery protocol also differs from the adaptive simulation of Laban et al. (2026), and the mitigations are evaluated on one model. Future studies could sample natural human partitions of requests, add more model families, and extend to other languages and tool-using agents.

8 CONCLUSION

A multi-turn gap is not a fixed property of a model and task; it also depends on how the request is turned into a conversation. Across our experiments, conversation design changes measured performance, mitigation gains, and the size of model differences, even when the underlying requirements are unchanged. The matched one-message control shows that some of this sensitivity comes from requirement arrangement, while distributing the same requirements across turns adds a further penalty. These results suggest that conclusions from a single conversation design should not be assumed to generalize to other designs. Multi-turn evaluations should therefore specify the conversation design and report results across multiple plausible designs.

REPRODUCIBILITY STATEMENT

Sections 3–4 and Appendix Q describe the design factors, task filters, models, decoding settings, statistical tests, and exclusion criteria. All scores are produced by programmatic checkers. The supplementary material includes per-conversation records, the design and evaluation code, and the analysis scripts.

Design construction is deterministic, but model generation is not. Batched inference introduces run-to-run variation even with a fixed seed, and conversations were originally submitted in grid order, so a design's position in the request stream was not randomized. We test both sources of variation directly: five reruns of the Qwen2.5-7B math grid and a randomized-submission-order rerun of five settings preserve the main design contrasts and σ_D. Appendix O reports these results in full.

USE OF AI ASSISTANCE

Large language models were used in four ways: to edit the manuscript for clarity and organization; to search for related work, with every cited paper read and verified by the authors; as a coding assistant for the evaluation harness and analysis scripts, and as a discussion partner during the design of the study; and to check mathematical derivations. All generated code was reviewed, tested, and run by the authors, every derivation was verified by the authors, and every reported number was checked against the raw conversation outputs. The authors selected the research question, designed and ran the experiments, interpreted the results, and take responsibility for all claims and citations.

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 (ACL)*, pp. 13787–13805, 2024.

Ge Bai, Jie Liu, Xingyuan Bu, Yancheng He, Jiaheng Liu, Zhanhui Zhou, Zhuoran Lin, Wenbo Su, Tiezheng Ge, Bo Zheng, and Wanli Ouyang. MT-bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 7421–7454, 2024.

Andrew M. Bean, Ryan Othniel Kearns, Angelika Romanou, Franziska Sofia Hafner, Harry Mayne, Jan Batzner, Negar Foroutan, et al. Measuring what matters: Construct validity in large language model benchmarks. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025.

Tianlang Chen, Shirley Wu, and Jure Leskovec. Found in conversation: LLMs teach themselves to close the multi-turn gap. *arXiv preprint arXiv:2605.24432*, 2026.

Lee J. Cronbach, Goldine C. Gleser, Harinder Nanda, and Nageswari Rajaratnam. *The Dependability of Behavioral Measurements: Theory of Generalizability for Scores and Profiles*. Wiley, New York, 1972.

Kevin H. Guo, Chao Yan, Avinash Baidya, Katherine Brown, Xiang Gao, Juming Xiong, Zhijun Yin, and Bradley A. Malin. Stop listening to me! how multi-turn conversations can degrade LLM reliability. *arXiv preprint arXiv:2603.11394*, 2026.

Jenny Y. Huang, Leshem Choshen, Wei Sun, Omar Khattab, Ramón Fernandez Astudillo, Mehul Damani, Tamara Broderick, and Jacob Andreas. Do LLMs benefit from their own words? *arXiv preprint arXiv:2602.24287*, 2026.

Haziq Mohammad Khalid, Athikash Jeyaganthan, Timothy Do, Yicheng Fu, Sean O'Brien, Vasu Sharma, and Kevin Zhu. ERGO: Entropy-guided resetting for generation optimization in multi-turn language models. *arXiv preprint arXiv:2510.14077*, 2025.

Wai-Chung Kwan, Xingshan Zeng, Yuxin Jiang, Yufei Wang, Liangyou Li, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. MT-eval: A multi-turn capabilities evaluation benchmark for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=VKGTGGcwl6.

Belinda Z. Li, Been Kim, and Zi Wang. QuestBench: Can LLMs ask the right question to acquire information in reasoning tasks? In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2025. arXiv:2503.22674.

Ming Li, Pei Chen, Zhenhao Zhang, Tao Yang, Xinyang Zhang, Han Li, Tianyu Cao, Ming Zeng, Zhuofeng Wu, Meng Jiang, Huasheng Li, Lihong Li, and Bing Yin. Mitigating lost in multi-turn conversation via curriculum RL with verifiable accuracy and abstention rewards. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026. arXiv:2510.18731.

Zizhuo Lin, Quanling Liu, Jinsheng Quan, Chao Zhang, Yifan Zhu, Xing Shi, Jingtao Xu, Zhihui Li, and Yawei Luo. Same evidence, different answers: Canonical-context on-policy distillation for multi-turn language models. *arXiv preprint arXiv:2605.30251*, 2026.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638.

Shu Tong Luo, Wenqin Liu, Rui Liu, Mingming Gong, and Jiaxian Guo. Multi-turn reasoning when context arrives in pieces: Scalable sharding and memory-augmented RL. *arXiv preprint arXiv:2606.12941*, 2026.

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

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. State of what art? A call for multi-prompt LLM evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949, 2024. doi: 10.1162/tacl_a_00681.

Akshay Paruchuri, Sanmi Koyejo, and Ehsan Adeli. Same evidence, different answer: Auditing order sensitivity in multimodal large language models, 2026. arXiv:2606.26079.

Abel Salinas and Fred Morstatter. The butterfly effect of altering prompts: How small changes and jailbreaks affect large language model performance. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4629–4651, 2024.

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 *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.11324.

Michael J.Q. Zhang, W. Bradley Knox, and Eunsol Choi. Modeling future conversation turns to teach LLMs to ask clarifying questions. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.13788.

Haoyu Zheng, Yun Zhu, Shu Yuan, Shangming Chen, Qing Wang, Wenqiao Zhang, Jun Xiao, and Yueting Zhuang. MAIGO: Mitigating lost-in-conversation with history-cleaned on-policy self-distillation. *arXiv preprint arXiv:2605.27186*, 2026.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2023.

A Full Mitigation Results

Table 3 gives the task-level results behind Table 2. $G(\pi_0)$ is the gain at the one reference design. $\bar{G}$ is the mean over all designs, 24 for every task except database, which has 20. "Non-positive" counts designs where the method does not improve on the baseline. The placebo rows are the semantically null edit of Section 5.6; they are shown for comparison and are not part of the multiplicity correction.

Bias-corrected gain SD. The σ_G column of Table 3 is a raw standard deviation across designs and inherits the sampling error of each design's gain. Correcting it as we correct σ_D sends the placebo to 0.000 on math, code and database and to 0.004 on tool selection, and sends entropy reset to 0.000 on the three tasks where it seldom fires. Recap, snowballing and user-turn-only retain a corrected σ_G above zero in all 12 of their task–method pairs, with a median of 0.035 against a raw median of 0.057 over all 16; entropy reset keeps a positive corrected spread only on tool selection.

Table 3: **Mitigation results by task, with the placebo.** A dagger marks the nine gains that pass the reported joint-resampling Benjamini–Hochberg procedure at $q = 0.05$; the placebo is not part of that correction. Database has 20 designs; every other task has 24. All gains are computed on the cells present in both the baseline and the intervention arm, which on database is 32 of the 33 instances; this is the same paired set the heterogeneity test of Table 5 permutes, so the two tables report the same σ_G.

Task	Method	$G(\pi_0)$	$\bar{G}$	σ_G	Non-positive
Math	Entropy reset	+0.007	-0.010	0.034	15/24
	Recap†	+0.087	+0.092	0.056	2/24
	Snowball†	+0.120	+0.114	0.069	2/24
	User-turn-only†	+0.240	+0.156	0.092	2/24
	Placebo (null edit)	-0.040	+0.003	0.031	11/24
Tool selection	Entropy reset†	+0.013	+0.032	0.034	8/24
	Recap†	+0.273	+0.151	0.109	3/24
	Snowball†	+0.440	+0.276	0.122	1/24
	User-turn-only†	+0.540	+0.353	0.146	0/24
	Placebo (null edit)	+0.007	-0.002	0.010	17/24
Code	Entropy reset	+0.020	+0.005	0.026	9/24
	Recap	+0.007	+0.005	0.045	7/24
	Snowball	-0.007	+0.017	0.044	10/24
	User-turn-only†	+0.127	+0.109	0.056	1/24
	Placebo (null edit)	-0.047	-0.010	0.033	15/24
Database	Entropy reset	-0.042	-0.002	0.030	13/20
	Recap	-0.104	+0.015	0.066	9/20
	Snowball	-0.021	+0.003	0.062	10/20
	User-turn-only†	+0.042	+0.094	0.058	1/20
	Placebo (null edit)	-0.051	-0.013	0.033	14/20

Table 4: **Entropy-reset firing rate for Qwen2.5-7B.** The method cannot fire before turn four, so it is inactive in every two-turn design. Rates are fractions of conversations in each cell.

Task	$T = 2$	$T = 4$	$T = 6$	overall
Math	0.0%	1.0%	2.7%	1.0%
Tool selection	0.0%	5.9%	9.5%	4.7%
Code	0.0%	3.4%	5.6%	2.7%
Database	0.0%	1.6%	–	0.9%

B THE ORIGINAL, FREE-ORDER FAMILY

Every design in our main family opens with the request. A natural worry is that pinning the request into turn 1 is itself what makes the scores move. To check, we generated a second family that permutes every unit freely, so that 11 of its 24 designs place a constraint before the request, and ran the same grid on it. Table 7 compares the design standard deviations of the two families over the settings with answer extraction of at least 0.5.

The two families agree closely: their design standard deviations differ by at most 0.029, and by 0.010 at the median. The largest disagreement is Qwen2.5-32B on code, which is significant in the free-order family ($\sigma = 0.040$) but not in ours ($\sigma = 0.011$); we report both and do not read either as a design effect that holds across families. Among the other ten settings, the request-first SD is smaller in eight and larger in two, Qwen2.5-7B on code (0.029 versus 0.022) and Qwen2.5-14B on database (0.067 versus 0.057), so the stricter protocol does not uniformly reduce sensitivity.

C FACTOR SHARES FOR EVERY MODEL

Table 8 gives the unique factor shares for all four models, and each setting's design standard deviation beside the value from the free-order family of Appendix B. Shares are *Type-II unique shares*: each factor's contribution to between-design variance once the other three factors are already accounted for. Design variances are bias-corrected, since sampling noise inflates the naive between-design variance estimate; the reported SD is the square root of the bias-corrected variance component, truncated at

Table 5: **Do mitigation gains vary by design? A direct test.** The null is $G(I, \pi) = g_I$ for every design π, tested by permuting the design labels of the paired gain within each (instance, seed). This is the heterogeneity null, distinct from the benefit null $\mathbb{E}_\pi[G] = 0$ that the sixteen-test correction addresses. The semantically null placebo edit never rejects.

Task	Method	SD of design gains	perm. p
Math	Entropy reset	0.034	0.737
	Recap	0.056	0.015
	Snowball	0.069	< 0.001
	User-turn-only	0.092	< 0.001
	Placebo (null edit)	0.031	0.873
Tool selection	Entropy reset	0.034	< 0.001
	Recap	0.109	< 0.001
	Snowball	0.122	< 0.001
	User-turn-only	0.146	< 0.001
	Placebo (null edit)	0.010	0.180
Code	Entropy reset	0.026	0.834
	Recap	0.045	0.017
	Snowball	0.044	0.135
	User-turn-only	0.056	0.004
	Placebo (null edit)	0.033	0.503
Database	Entropy reset	0.030	0.915
	Recap	0.066	0.027
	Snowball	0.062	0.045
	User-turn-only	0.058	0.280
	Placebo (null edit)	0.033	0.877

Table 6: **User-turn-only recovers the matched one-message score.** Paired on shared (design, instance, seed) cells for Qwen2.5-7B; the interval is an instance bootstrap; values are on cells shared by both arms, so the database column differs from Table 10's 0.386, which uses all cells. Every difference is small and every interval covers zero.

Task	user-turn-only	CONCAT-BLOCKS	difference	95% CI
Math	0.666	0.660	+0.006	$[-0.011, +0.023]$
Tool selection	0.807	0.829	-0.022	$[-0.049, +0.004]$
Code	0.449	0.440	+0.009	$[-0.007, +0.026]$
Database	0.392	0.372	+0.020	$[-0.010, +0.048]$

zero. Significance uses the same permutation test as the main text (labels shuffled within instance and seed); code at 32B does not beat this null ($p = 0.20$). Extraction is the fraction of final turns producing a parseable answer; three Mistral-7B rows fall below our 0.5 threshold ($*$) and are reported but not interpreted. Qwen2.5-32B is absent from tool selection and database because it does not run those tasks.

Database uses its own family of 20 designs (Appendix P). A design standard deviation has no mechanical increase with the number of designs, but it still depends on the family and its weights. Best-to-worst ranges are sensitive to the number of designs tested, so we compare SDs rather than ranges here.

D BEST AND WORST DESIGNS

Table 9 gives the score range for all model–task settings.

Table 7: **Design standard deviation in the request-first family and in the original family, which lets some designs delay the overall request.** Settings with answer extraction of at least 0.5. The two agree closely, so the effect is not produced only by designs that withhold the goal.

Model	Task	Request-first	Original family
Qwen2.5-7B	Code	0.029	0.022
	Database	0.075	0.085
	Math	0.109	0.121
	Tool selection	0.193	0.202
Qwen2.5-14B	Code	0.031	0.034
	Database	0.067	0.057
	Math	0.067	0.090
	Tool selection	0.185	0.204
Qwen2.5-32B	Code	0.011	0.040
	Math	0.052	0.059
Mistral-7B	Code	0.021	0.024

Table 8: **Order carries the largest variance share on math and database even when the request opens turn 1.** Shares are Type-II unique shares and do not sum to R^2; the difference is variance shared between correlated factors and is given in the *shared* column. Bold marks the factor with the largest unique share. $*$ marks extraction below 0.5: reported, not interpreted. A zero SD is the square root of a variance estimate truncated at zero, not evidence of exactly zero variance.

		design SD		unique variance share, compliant						
Task	Model	orig.	compl.	order	turn count	front-load.	restate.	shared	R^2	extr.
Math	7B	0.121	0.109	**0.425**	0.242	0.007	0.004	0.226	0.90	0.91
	14B	0.090	0.067	**0.675**	0.073	0.008	0.004	0.143	0.90	0.94
	32B	0.059	0.052	**0.689**	0.012	0.007	0.007	0.017	0.73	0.85
	Mistral*	0.099	0.102	0.102	**0.621**	0.028	0.016	0.125	0.89	0.45
Tool selection	7B	0.202	0.193	0.122	**0.552**	0.063	0.019	0.125	0.88	0.91
	14B	0.204	0.185	**0.279**	0.232	0.115	0.034	0.099	0.76	0.90
	Mistral*	0.052	0.072	0.076	**0.408**	0.157	0.012	0.225	0.88	0.25
Code	7B	0.022	0.029	0.145	**0.157**	0.096	0.053	0.207	0.66	0.98
	14B	0.034	0.031	0.105	**0.169**	0.076	0.014	0.190	0.55	0.99
	32B	0.040	0.011	**0.267**	0.008	0.020	0.018	0.070	0.38	0.99
	Mistral	0.024	0.021	0.097	**0.460**	0.028	0.044	0.227	0.86	0.90
Database	7B	0.085	0.075	**0.524**	0.113	0.061	0.014	0.167	0.88	0.69
	14B	0.057	0.067	**0.614**	0.079	0.000	0.002	0.153	0.85	0.58
	Mistral*	0.002	0.000	0.098	0.045	**0.142**	0.047	-0.006	0.33	0.08

E THREE CONTRASTS WITHIN THE GAP

The gap between FULL and SHARDED combines several changes to the input. To describe them separately, we define A as a one-message prompt containing the requirement units in canonical order, without restatement or block boundaries. Plain CONCAT creates this prompt by joining the units with spaces. The contrast between FULL and A reflects the change from original prose to requirement units; the contrast between A and CONCAT-BLOCKS reflects how those units are arranged in a particular design; and the contrast between CONCAT-BLOCKS and SHARDED reflects delivering the same blocks over turns. These three contrasts sum to the observed gap (Table 10). They are descriptive and depend on the choice of A. The split differs by task. For Qwen2.5-7B, arrangement accounts for a larger pre-conversation score difference than conversion on math ($+0.100$ versus $+0.045$) and tool selection ($+0.117$ versus $+0.014$). Conversion is larger on code ($+0.239$ versus -0.004) and database ($+0.462$ versus $+0.026$). These are contrasts relative to one canonical rendering A, not a unique causal allocation of the gap. The database row uses a separate free-order run, as noted in the table caption, and the source benchmark's LLM extractor, which Appendix I shows lowers database scores; part of its large conversion contrast may therefore reflect extraction rather than the model.

Table 9: **The one-message score, and the lowest and highest mean scores among the designs.** FULL delivers the original unsplit instruction and does not depend on the design. *Range* is Best − Worst. *Gap* is FULL minus the design-marginal SHARDED score; the two gap columns give it at the best and the worst design, and their difference is the range. The gap is positive in every reliable setting and no design closes it, but its measured size depends on which design a benchmark happens to use. Database uses its own 20-design family (Appendix P). The three Mistral rows with extraction below 0.5 (∗) are included for completeness but are not used in the headline median.

Model	Task	Designs	FULL	Worst	Best	Range	Gap@best	Gap@worst
Qwen2.5-7B	Tool selection	24	0.961	0.233	0.900	0.667	+0.061	+0.728
Qwen2.5-14B	Tool selection	24	0.973	0.300	0.967	0.667	+0.007	+0.673
Qwen2.5-7B	Math	24	0.805	0.320	0.767	0.447	+0.039	+0.485
Mistral-7B*	Math	24	0.382	0.007	0.367	0.360	+0.015	+0.375
Qwen2.5-14B	Math	24	0.913	0.520	0.807	0.287	+0.106	+0.393
Qwen2.5-7B	Database	20	0.869	0.125	0.406	0.281	+0.463	+0.744
Mistral-7B*	Tool selection	24	0.149	0.000	0.273	0.273	-0.125	+0.149
Qwen2.5-14B	Database	20	0.843	0.152	0.424	0.273	+0.419	+0.692
Qwen2.5-32B	Math	24	0.783	0.527	0.747	0.220	+0.036	+0.256
Qwen2.5-7B	Code	24	0.674	0.273	0.480	0.207	+0.194	+0.401
Qwen2.5-14B	Code	24	0.743	0.380	0.580	0.200	+0.163	+0.363
Qwen2.5-32B	Code	24	0.802	0.540	0.680	0.140	+0.122	+0.262
Mistral-7B	Code	24	0.139	0.013	0.120	0.107	+0.019	+0.126
Mistral-7B*	Database	20	0.436	0.000	0.010	0.010	+0.426	+0.436

Table 10: **Three score contrasts for Qwen2.5-7B.** Atomic units are the individual requirements extracted from the original instruction. A joins these units in one message, in canonical order and without restatement or block boundaries. The three contrasts compare FULL with A, A with CONCAT-BLOCKS, and CONCAT-BLOCKS with SHARDED. Scores use instances present in all four arms. The database row (†) uses a separate free-order run because the request-first grid lacks a plain CONCAT arm.

Task	FULL	A	CONCAT-BLOCKS	SHARDED	atomis.	arrang.	interact.
Math	0.805	0.760	0.660	0.510	+0.045	+0.100	+0.150
Tool selection	0.961	0.947	0.829	0.454	+0.014	+0.117	+0.375
Code	0.674	0.436	0.440	0.341	+0.239	-0.004	+0.099
Database†	0.873	0.412	0.386	0.309	+0.462	+0.026	+0.077

F Do the Two Arms Agree About Which Designs Are Hard?

Table 11 gives, for every setting, the correlation between the design means of the two arms that Section 5.7 summarizes.

G Cross-Arm Variance Tests

A pre-declared check used separate six-turn grids on math (25 instances, 19 designs) and code (58 instances, 18 designs) to allow several revisions following the first answer. The bias-corrected design-variance difference between SHARDED and CONCAT-BLOCKS was -0.0010 on math (95% joint-resampling interval $[-0.0133, +0.0112]$) and $+0.0005$ on code ($[-0.0009, +0.0029]$). Neither interval excludes zero; no equivalence margin was set.

In the main grid, the untruncated bias-corrected variance difference is positive in all 14 settings. Table 12 gives the test results. The two smallest raw p-values are 0.006 and 0.013 on tool selection. After Benjamini–Hochberg correction across 14 tests at $q = 0.05$, neither survives. These marginal-variance tests differ from variation in the paired penalty $C_\pi - S_\pi$ (Appendix L).

H Held-Out Validation of the Ranking Reversals

Table 13 gives the held-out check of Section 5.5 for seven of the eight comparisons, including the three with no point-estimate reversal. Code 32B versus 7B has no reversing design on any split, so the procedure never selects one and the row is undefined; it is omitted.

Table 11: **The one-message and interactive conditions tend to find the same designs difficult.** Correlation between design means in SHARDED and CONCAT-BLOCKS on matched instances, with joint resampling of designs and instances. Every point estimate is positive; five intervals among the 11 reliable settings exclude zero. Correlation does not allocate the observed score difference between arrangement and interaction. $*$ marks extraction below 0.5.

Task	Model	σ_D SHARDED	σ_D control	Pearson r	95% CI
Tool selection	Qwen2.5-7B	0.193	0.092	+0.57	[+0.09, +0.81]
	Qwen2.5-14B	0.185	0.091	+0.62	[+0.24, +0.82]
	Mistral-7B*	0.072	0.068	+0.64	[+0.05, +0.86]
Math	Qwen2.5-7B	0.109	0.065	+0.67	[+0.20, +0.83]
	Qwen2.5-14B	0.067	0.046	+0.77	[+0.18, +0.89]
	Qwen2.5-32B	0.052	0.033	+0.61	[-0.08, +0.82]
	Mistral-7B*	0.102	0.062	+0.35	[-0.08, +0.66]
Database	Qwen2.5-7B	0.075	0.042	+0.74	[+0.02, +0.87]
	Qwen2.5-14B	0.067	0.000	+0.46	[-0.43, +0.77]
	Mistral-7B*	0.000	0.000	+0.59	[-0.20, +0.97]
Code	Qwen2.5-7B	0.029	0.022	+0.54	[-0.12, +0.73]
	Qwen2.5-14B	0.031	0.027	+0.23	[-0.30, +0.62]
	Qwen2.5-32B	0.011	0.000	+0.49	[-0.24, +0.72]
	Mistral-7B	0.021	0.000	+0.13	[-0.41, +0.59]

Table 12: **No cross-arm excess survives correction for multiple testing.** One test of the interactive variance excess per setting, two-sided bootstrap p-values from 20,000 joint resamples, then Benjamini–Hochberg at $q = 0.05$. The two tool-selection settings have the smallest raw p-values and are the two that an uncorrected reading would call significant; neither clears the corrected threshold. $*$ marks extraction below 0.5.

Task	Model	excess	raw p	survives BH
Tool selection	Qwen2.5-7B	+0.02865	0.013	no
	Qwen2.5-14B	+0.02588	0.006	no
	Mistral-7B*	+0.00058	0.912	no
Math	Qwen2.5-7B	+0.00761	0.104	no
	Qwen2.5-14B	+0.00239	0.271	no
	Qwen2.5-32B	+0.00161	0.427	no
	Mistral-7B*	+0.00658	0.194	no
Database	Qwen2.5-7B	+0.00385	0.325	no
	Qwen2.5-14B	+0.00562	0.125	no
	Mistral-7B*	+0.00000	1.000	no
Code	Qwen2.5-7B	+0.00033	0.882	no
	Qwen2.5-14B	+0.00024	0.852	no
	Qwen2.5-32B	+0.00020	0.837	no
	Mistral-7B	+0.00053	0.656	no

I EXTRACTION BY DESIGN

Answer extraction varies across designs and is part of the measured outcome: unparsed final turns receive zero. Table 14 compares raw score spread across all conversations with spread among parsed replies. The latter remains substantial, but it conditions on a design-dependent subset of instances, so on its own it does not settle the question.

What the unparsed database replies are. Database is the setting where extraction is lowest (0.69 at 7B, 0.58 at 14B) and most variable across designs, so we audited it. The database answer is lifted by a 7B LLM extractor in the source benchmark's prefix/suffix mode, and its own per-turn labels are recorded. Of the 614 final turns it declined at 7B, 608 are labelled *answer attempt, unextractable* and every one contains a SQL statement; at 14B the corresponding figures are 825 declined, 558 so labelled, again all with a statement present (Table 15). Several are clean fenced queries. The decline rate across designs correlates -0.64 (7B) and -0.57 (14B) with the design's score. So the extractor, not the model, was producing a large part of the zeros, and it did so unevenly across designs.

Table 13: **Held-out checks for model-ranking reversals, with uncertainty.** Reversing designs are chosen on one random half of the instances and tested on the other, over 400 splits, at a fixed seed. *Reproduce* is the fraction of chosen designs that also reverse on the held-out half; *base* is the fraction of *all* designs that reverse on that same held-out half, which is what choosing designs at random would give. Because both are measured on half the instances, a design can reverse on a half without reversing on the full data. *Excess* is reproduce minus base, bootstrapped as a paired difference within each replicate; no comparison's excess excludes zero. *Held-out* Δ is the mean held-out score difference of the selected designs, positive when the model that wins on average still wins on them. The database row uses the source benchmark's extractor; under the deterministic extractor it reproduces at 0.00 against a held-out base rate of 0.03, and no design reverses on the full instance set.

Task	Model pair	reproduce	base	excess	95% CI on excess	held-out Δ
Math	14B vs. 7B	0.00	0.02	-0.02	$[-0.06, +0.57]$	+0.141
Code	14B vs. 7B	0.00	0.06	-0.06	$[-0.11, +0.37]$	+0.153
Code	32B vs. 14B	0.00	0.01	-0.01	$[-0.06, +0.52]$	+0.168
Math	32B vs. 7B	0.55	0.12	$+0.44$	$[-0.05, +0.64]$	+0.030
Math	14B vs. 32B	0.32	0.31	$+0.01$	$[-0.17, +0.35]$	+0.034
Tool selection	14B vs. 7B	0.31	0.12	$+0.19$	$[-0.09, +0.55]$	+0.073
Database	14B vs. 7B	0.41	0.35	$+0.06$	$[-0.17, +0.23]$	+0.025

Table 14: **Score variation remains among parsed replies.** *ext* is final-turn parse rate; r correlates parse rate with score across designs. The last two columns are raw, uncorrected standard deviations across designs, using all conversations or only parsed replies. The parsed subset changes with the design, so this comparison cannot rule out an extraction contribution. $*$ marks mean extraction below 0.5.

Task	Model	ext mean	ext min	ext max	r(ext, score)	σ_D all	σ_D parsed
Tool selection	Qwen2.5-7B	0.906	0.840	0.993	+0.67	0.200	0.200
	Qwen2.5-14B	0.897	0.773	0.993	+0.79	0.192	0.180
	Mistral-7B*	0.250	0.133	0.407	+0.73	0.077	0.213
Math	Qwen2.5-7B	0.912	0.860	0.933	+0.46	0.117	0.123
	Qwen2.5-14B	0.941	0.913	0.967	-0.04	0.079	0.085
	Qwen2.5-32B	0.850	0.807	0.893	+0.31	0.068	0.076
	Mistral-7B*	0.454	0.213	0.760	+0.90	0.105	0.136
Database	Qwen2.5-7B	0.682	0.542	0.781	+0.73	0.089	0.110
	Qwen2.5-14B	0.583	0.455	0.677	+0.80	0.085	0.112
	Mistral-7B*	0.078	0.020	0.162	-0.13	0.005	0.033
Code	Qwen2.5-7B	0.984	0.960	1.000	-0.04	0.045	0.046
	Qwen2.5-14B	0.988	0.953	1.000	+0.54	0.048	0.045
	Qwen2.5-32B	0.991	0.967	1.000	+0.16	0.037	0.037
	Mistral-7B	0.899	0.780	0.980	+0.79	0.026	0.026

We therefore re-scored every database conversation with a deterministic extractor, the last fenced SQL block or else the last statement, through the identical execution checker (Table 16). The extractor had been suppressing the *level*: the 7B mean rises from 0.289 to 0.391 and the 14B mean from 0.309 to 0.523. It had not been producing the *design effect*: σ_D is 0.067 against 0.075 at 7B and 0.098 against 0.067 at 14B, the 20 design means correlate $+0.92$ and $+0.95$ across the two extractors, and the permutation test rejects at $p < 0.001$ under both. The reported database scores in the main tables keep the source benchmark's extractor for comparability with it, but the design sensitivity on database is not an artifact of that extractor, and a benchmark using this task should not rely on an LLM extractor for it.

J What the Estimand Is, and What the Intervals Cover

The reported mean, its uncertainty for new instances, and the risk of using only a subset of designs are distinct quantities.

Table 15: **What the unparsed database replies are.** Each unparsed final turn is classified from its text. *Query present* means a complete SQL statement is there and the parser did not lift it; *truncated* means the reply hit the token budget mid-statement, so it contains a partial one; *no query* means neither. Every unparsed reply contains at least a partial statement and 66–70% a complete one. The last column correlates the per-design parser-miss rate with the design's score.

Model	unparsed	query present	truncated	no query	r(miss, score)
Qwen2.5-7B	614/1979	66%	34%	0%	-0.64
Qwen2.5-14B	825/1980	70%	30%	0%	-0.57

Table 16: **Database re-scored with a deterministic extractor.** *rule* lifts the last fenced SQL block (else the last statement) and scores it with the identical checker. Same conversations, same checker; only the extractor changes. *corr* correlates the 20 design means under the two extractors.

Model	extr. LLM	extr. rule	mean LLM	mean rule	σ_D LLM	σ_D rule	corr	perm. p rule
Qwen2.5-7B	0.69	1.00	0.289	0.391	0.075	0.067	+0.92	< 0.001
Qwen2.5-14B	0.58	1.00	0.309	0.523	0.067	0.098	+0.95	< 0.001

Fixed-family target. P is uniform over the realized family $\mathcal{D}$ of 24 designs (20 on database). The reported design-marginal mean is the average over exactly these designs, with their specified weights. We do not claim that $\mathcal{D}$ is representative of all possible conversations. Because every design in $\mathcal{D}$ was evaluated, there is no design-sampling uncertainty in this fixed-family average. An interval for its performance on new instances would keep all designs and resample the shared instances, while accounting for variation across generations.

Reported intervals. The existing bootstrap intervals instead resample designs and shared instances jointly. They describe how an estimate can move when both the instance sample and the composition of a design subset change. They should be read as joint-resampling sensitivity intervals, not as confidence intervals for the fixed-family mean or for unseen conversation designs. The reported bootstrap tests and their multiple-testing corrections use this same joint-resampling convention. Table 17 gives the instance-only intervals for the fixed-family target beside the joint ones: resampling designs as well widens the interval by a median factor of 1.12 over the 11 reliable settings (1.16 over all 14), so the two agree closely here, but they answer different questions and are labeled accordingly.

Table 17: **Uncertainty for the fixed-family mean, and separately for benchmark construction.** μ_D is the mean over all designs at their fixed weights. *Instances only* keeps every design and resamples the shared instances: this is the interval for μ_D on new instances, and there is no design-sampling term in it because every design was evaluated. *+designs* additionally resamples designs and answers a different question, how far a benchmark built from another draw of designs would land; it is wider by construction and is not an interval for μ_D. $*$ marks extraction below 0.5.

Task	Model	μ_D	95% CI, instances only	95% CI, +designs	width ratio
Tool selection	Qwen2.5-7B	0.454	[0.396, 0.511]	[0.355, 0.559]	1.77
	Qwen2.5-14B	0.579	[0.520, 0.636]	[0.476, 0.673]	1.70
	Mistral-7B*	0.068	[0.046, 0.091]	[0.035, 0.113]	1.73
Math	Qwen2.5-7B	0.510	[0.436, 0.581]	[0.428, 0.599]	1.17
	Qwen2.5-14B	0.673	[0.601, 0.742]	[0.592, 0.751]	1.12
	Qwen2.5-32B	0.645	[0.581, 0.708]	[0.571, 0.717]	1.15
	Mistral-7B*	0.097	[0.071, 0.128]	[0.052, 0.153]	1.77
Database	Qwen2.5-7B	0.289	[0.211, 0.377]	[0.207, 0.376]	1.02
	Qwen2.5-14B	0.309	[0.252, 0.368]	[0.240, 0.386]	1.26
	Mistral-7B*	0.004	[0.001, 0.009]	[0.000, 0.011]	1.38
Code	Qwen2.5-7B	0.341	[0.263, 0.422]	[0.259, 0.428]	1.06
	Qwen2.5-14B	0.447	[0.359, 0.536]	[0.357, 0.539]	1.03
	Qwen2.5-32B	0.619	[0.531, 0.700]	[0.529, 0.703]	1.03
	Mistral-7B	0.051	[0.023, 0.085]	[0.021, 0.088]	1.09

Table 18: **The score still moves when the turn count and the front-loaded share are held fixed.** Within a fixed (T, f) cell only the requirement order and the restatement flag differ, so these columns isolate those two choices. The permutation p shuffles order and restatement labels inside each (T, f) cell and inside each (instance, seed) pair, conditional on the observed conversations; it does not measure additional run-to-run variation. The bootstrap interval resamples designs and instances jointly; it is wide because a variance component truncated at zero is poorly determined from 24 designs, and the permutation test is the sharper instrument here. $*$ marks extraction below 0.5: reported, not interpreted.

Task	Model	σ_D all designs	σ_D within (T, f)	95% CI	perm. p
Tool selection	Qwen2.5-7B	0.193	0.081	[0.000, 0.118]	< 0.001
	Qwen2.5-14B	0.185	0.128	[0.051, 0.162]	< 0.001
	Mistral-7B*	0.072	0.019	[0.000, 0.044]	< 0.001
Math	Qwen2.5-7B	0.109	0.082	[0.021, 0.118]	< 0.001
	Qwen2.5-14B	0.067	0.066	[0.000, 0.103]	< 0.001
	Qwen2.5-32B	0.052	0.063	[0.000, 0.100]	< 0.001
	Mistral-7B*	0.102	0.049	[0.000, 0.081]	< 0.001
Database	Qwen2.5-7B	0.075	0.069	[0.000, 0.121]	< 0.001
	Qwen2.5-14B	0.067	0.065	[0.000, 0.121]	< 0.001
	Mistral-7B*	0.000	0.000	[0.000, 0.009]	1.000
Code	Qwen2.5-7B	0.029	0.020	[0.000, 0.054]	0.018
	Qwen2.5-14B	0.031	0.014	[0.000, 0.057]	0.109
	Qwen2.5-32B	0.011	0.021	[0.000, 0.055]	0.027
	Mistral-7B	0.021	0.004	[0.000, 0.024]	0.158

The variance estimator. The naive spread of design means is biased upward, because each design mean is itself measured with error. For a balanced layout of s designs, n instances and r seeds with cell means $\bar{y}_d$, we report

$$\hat{\sigma}_D^2 \;=\; \frac{1}{s-1}\sum_{d=1}^{s}\big(\bar{y}_d - \bar{y}\big)^2 \;-\; \frac{\mathrm{MS}_{AB}}{nr}, \tag{4}$$

truncated at zero, where MS_{AB} is the design-by-instance interaction mean square. The subtracted term is the sampling variance of a single design mean. The dependence between design means is exactly what makes MS_{AB} the right correction rather than a pooled within-cell variance: because every design is evaluated on the same instances, a hard instance raises or lowers all 24 design means together, and that shared component belongs to neither the design effect nor the noise floor. Using the interaction mean square removes the instance main effect from the correction and leaves only the part that varies design by design. Truncation at zero makes the estimator conservative in one direction, which matters when we compare two truncated estimates, and where we do so we report the untruncated difference as well.

K Design Variance Within Fixed Turn-Count and Front-Loading Cells

Table 18 gives the per-setting detail behind Section 5.4. The reported ratio of 0.76 is the median of the per-setting ratios of standard deviations.

L The Matched Interaction Penalty

Table 19 gives, for every setting, the paired penalty $I_\pi = C_\pi - S_\pi$ behind Section 5.7: its mean, its raw spread across designs, and the bias-corrected spread.

M Replication on a Different Model Family

We also tested Gemma-2-9B, a different model family. Its design effect clears the permutation test on three tasks but not code (Table 20). Gemma used 24 designs on math, tool selection, and code, and

Table 19: **Paired penalty for delivering the same blocks over turns.** $I_\pi = C_\pi - S_\pi$ compares paired conversations. *SD corr.* is a bias-corrected estimate of its spread across designs; it is positive in eight of the 11 reliable settings. A permutation test on I_π within (instance, seed) rejects a design-invariant penalty in six of the 11 reliable settings (Section 5.7). $*$ marks extraction below 0.5.

Task	Model	mean I	SD raw	SD corr.
Tool selection	Qwen2.5-7B	+0.375	0.165	0.153
	Qwen2.5-14B	+0.312	0.152	0.140
	Mistral-7B*	+0.042	0.064	0.052
Math	Qwen2.5-7B	+0.150	0.087	0.068
	Qwen2.5-14B	+0.116	0.051	0.021
	Qwen2.5-32B	+0.044	0.054	0.019
	Mistral-7B*	+0.225	0.104	0.096
Database	Qwen2.5-7B	+0.076	0.060	0.000
	Qwen2.5-14B	+0.078	0.076	0.034
	Mistral-7B*	+0.001	0.005	0.000
Code	Qwen2.5-7B	+0.099	0.041	0.000
	Qwen2.5-14B	+0.023	0.057	0.032
	Qwen2.5-32B	+0.016	0.036	0.000
	Mistral-7B	+0.060	0.032	0.015

20 on database. Its median design standard deviation across the four tasks is 0.134, and its median multi-turn gap is $+0.404$. Gemma-2's chat template rejects a system turn, so we placed the same text at the start of the first user message.

Table 20: **Gemma-2-9B replication.** Three tasks show a design effect above the permutation null; code does not. The grid has 24 designs except on database, which has 20. *null* is the range expected from sampling noise alone and *obs/null* the factor above it; *corr* is the correlation between the design means of SHARDED and CONCAT-BLOCKS. $*$ marks extraction below 0.5: reported, not interpreted.

Task	Model	σ_D	Range	Null	Obs/null	perm. p	corr	extr.
Tool selection	Gemma-2-9B	0.156	0.613	0.191	3.21	< 0.001	+0.65	0.79
Math	Gemma-2-9B	0.150	0.553	0.175	3.15	< 0.001	+0.63	0.76
Database	Gemma-2-9B	0.118	0.414	0.215	1.93	< 0.001	+0.73	0.77
Code	Gemma-2-9B	0.000	0.107	0.116	0.92	0.272	+0.17	0.94

N THE RANGE AGAINST ITS NOISE FLOOR

The best-to-worst range is the distance between two *selected* design means, so it inherits the sampling error of both. Table 21 bounds how much of it that accounts for. The floor is what the range would be if the design means differed only by noise; the cross-validated column removes the selection by choosing the extremes on half the instances and scoring them on the other half.

Two readings follow. The design effect is real: 10 of the 11 reliable settings exceed their own noise floor by at least 1.30 times and keep a positive cross-validated range. But the raw range overstates the effect by roughly a factor of two, and a reader comparing ranges across settings should use σ_D, which is bias-corrected and does not select on extremes. Qwen2.5-32B on code is near the range floor, has a negative held-out contrast, and fails the overall permutation test, so we cannot establish a design effect in that setting.

O RUN-TO-RUN AND SUBMISSION-ORDER ROBUSTNESS

Two things in the pipeline are not deterministic or not randomised, and this appendix tests both. Batched inference varies run to run even at a fixed seed. Rerunning the Qwen2.5-7B math grid five times changed 42% of final answers and 20% of (design, instance) score cells between any two runs, yet the contrasts built from those scores hold: design means correlate $+0.83$ to $+0.91$ between runs, and choosing the extreme designs on one run and scoring them on another gives a range of 0.394 on

Table 21: **The best-to-worst range is inflated by selection, but not explained by it.** *null* is the expected range when 24 (or 20) design means differ only by sampling noise at that setting's own standard error $\mathrm{SE} = \sqrt{\mathrm{MS}_{AB}/nr}$, simulated 20,000 times. *cross-val* removes the selection bias by choosing the extreme designs on half the instances and scoring them on the other half. The observed range exceeds the noise floor and survives cross-validation in ten of the 11 reliable settings; Qwen2.5-32B on code sits at the floor, has a negative held-out contrast, and is the setting whose permutation test also fails. σ_D, which is bias-corrected and does not select on extremes, is the safer headline. $*$ marks extraction below 0.5.

Task	Model	σ_D	Range	SE	Null	Obs/null	Cross-val
Tool selection	Qwen2.5-7B	0.193	0.667	0.053	0.205	3.25	0.570
	Qwen2.5-14B	0.185	0.667	0.052	0.204	3.26	0.580
	Mistral-7B*	0.072	0.273	0.026	0.103	2.66	0.229
Math	Qwen2.5-7B	0.109	0.447	0.044	0.169	2.64	0.351
	Qwen2.5-14B	0.067	0.287	0.042	0.163	1.76	0.151
	Qwen2.5-32B	0.052	0.220	0.043	0.169	1.30	0.064
	Mistral-7B*	0.102	0.360	0.026	0.101	3.58	0.322
Database	Qwen2.5-7B	0.075	0.281	0.048	0.178	1.58	0.151
	Qwen2.5-14B	0.067	0.273	0.053	0.196	1.39	0.123
	Mistral-7B*	0.000	0.010	0.006	0.023	0.44	0.000
Code	Qwen2.5-7B	0.029	0.207	0.034	0.133	1.56	0.104
	Qwen2.5-14B	0.031	0.200	0.036	0.140	1.43	0.090
	Qwen2.5-32B	0.011	0.140	0.036	0.139	1.01	-0.005
	Mistral-7B	0.021	0.107	0.016	0.061	1.75	0.061

average against 0.457 within a run. This is the evidence the permutation null cannot supply, since that null is conditional on the observed generations.

The main grid submitted conversations to the inference server in grid order, instance-major then design, at 64 concurrent workers. The batch in flight at any moment therefore held adjacent designs of neighbouring instances, and a design's position in the request stream was not random. Batch composition changes model outputs, so a design's mean could in principle carry a position component that pairing within instance and seed cannot remove. To test this we reran the Qwen2.5-7B grid on all four tasks and the Qwen2.5-32B math grid with a seeded shuffle of submission order and no other change. Table 22 compares the two runs. The comparison uses the original run's extreme designs scored in the shuffled run, so it involves no selection on the shuffled data.

Randomising submission order changes nothing that matters. Across the five settings the per-design means correlate a median $+0.94$ between the two runs, $+1.00$ on tool selection; σ_D changes by at most 0.007; the original run's best-to-worst contrast retains a median 89% of its value when scored in the shuffled run; and the shuffled run rejects the permutation null in all five. The math 32B-versus-7B reversal, the one that reproduces most often on held-out instances, also persists: both designs that reverse in the original run reverse in the shuffled run, with the marginal gap essentially unchanged ($+0.134$ against $+0.135$). Batch position did not produce the design effect.

Table 22: **The design effect survives randomised submission order.** *shuf* reruns the same designs, instances and seeds with a seeded shuffle of submission order and nothing else changed. *corr* correlates per-design means across the two runs. *best−worst* is the contrast between the original run's extreme designs, scored in the shuffled run, so it involves no selection on the shuffled data.

Task	Model	σ_D orig	σ_D shuf	corr	contrast orig	in shuf	perm. p shuf
Math	Qwen2.5-7B	0.109	0.113	+0.97	0.447	0.453	< 0.001
Tool selection	Qwen2.5-7B	0.193	0.196	+1.00	0.667	0.687	< 0.001
Code	Qwen2.5-7B	0.029	0.029	+0.79	0.207	0.160	0.002
Database	Qwen2.5-7B	0.075	0.076	+0.94	0.281	0.250	< 0.001
Math	Qwen2.5-32B	0.052	0.059	+0.83	0.220	0.180	< 0.001

P THE DESIGN FAMILIES

Table 23 lists the 24-design family used for math, tool selection and code, and Table 24 the 20-design family used for database. The two are drawn by the same seeded procedure under different constraints: database instances carry five to seven requirement units, so $T \in \{2, 4\}$ and $K \geq 5$, and the feasibility and duplicate filters remove different cells from the shuffled sequence. The two families are therefore different draws, not a subset relation, and share nine designs. The score columns show that neither family is ordered by score.

Realized front-loading. The first block holds $\max\big(1, \min(K - (T - 1), \mathrm{round}(fK))\big)$ units, where round is round-half-to-even, as in the implementation; the only instances on which this differs from rounding halves upward are the nine with $K = 10$ at $f = 0.45$, seven on code and two on tool selection, where the first block holds four units rather than five. The remaining $K -$ first are split as evenly as possible over the other $T - 1$ blocks, earliest blocks taking the remainder. The lower clamp keeps the first block non-empty and the upper clamp keeps every later block non-empty. The realized share never equals the nominal f: at $K = 6$ the levels 0.15, 0.30 and 0.45 realize as 0.167, 0.333 and 0.500, and at $K = 6$, $T = 6$ the upper clamp forces the first block to one unit for all three levels, so front-loading is inert in that cell. A restatement repeats one unit drawn uniformly from those already delivered, at a turn drawn uniformly from $2, \ldots, T$, both from the per-instance seed.

Duplicate and collapsed cells. Two designs are duplicates when they render the identical conversation on any instance; the later cell in the shuffled order is then dropped for every instance, keeping the crossing complete. Two mechanisms produce duplicates. At $K = 5$ the levels $f = 0.30$ and $f = 0.45$ both put two units in the first block. And when the opening block holds a single unit, that unit is the request, the only non-constraint unit, so the constraint-first ordering becomes the request followed by the constraints in source order, exactly the canonical ordering; reverse and random orderings stay distinct, so four order levels become three in those cells. Counting distinct conversations, the attainable crossing is 48 of 72 cells on math, 49 on tool selection, 53 on code, and 20 of 48 on database. Order is applied to the full sequence of units before the blocks are cut from it, so it also rearranges units *within* every block; we do not canonicalize within-block order, and CONCAT-BLOCKS concatenates the same blocks in the same internal order.

Q ADDITIONAL PROTOCOL DETAILS

At the six-unit threshold the released pool holds 51 math, 61 tool-selection and 79 code instances, of which we use the first 50 of each; at five units it holds 33 database instances, which we use in full. Only 41 unused instances therefore exist across all four tasks, and math has exactly one.

Every design is checked to contain the same set of unique requirements by its final turn. Database uses $T \in \{2, 4\}$ because fewer instances support six valid shards. Each cell uses the same instances and three seeds, which allows paired tests. Answer extraction uses the released task parser. We exclude a setting from the headline summary when fewer than half of final turns produce a parseable answer. This rule removes Mistral-7B on tool selection, math and database from the 11-setting headline, but the paper still reports all three rows.

The method study uses Qwen2.5-7B on all four tasks. Its grid is 24 designs, 50 instances, three seeds, and baseline/intervention pairs; database uses 20 designs and 33 instances. Bootstrap intervals resample designs and instances jointly, for the reasons given in Appendix J.

Table 23: **The 24-design family.** Every design is a cell (order, T, f, restatement) of the full factorial, drawn as described in Section 3, and is used for math, tool selection and code. The last column is the Qwen2.5-7B math score. Order levels appear 7, 7, 6 and 4 times and constraint-first never occurs at six turns: the family is a reproducible random draw, not a balanced array.

#	Order	T	f	Restate	7B math
1	canonical	2	0.45	no	0.593
2	canonical	2	0.45	yes	0.607
3	canonical	4	0.15	no	0.540
4	canonical	4	0.30	yes	0.507
5	canonical	4	0.45	yes	0.500
6	canonical	6	0.30	no	0.487
7	canonical	6	0.30	yes	0.467
8	constraint-first	2	0.15	no	0.693
9	constraint-first	2	0.15	yes	0.767
10	constraint-first	2	0.30	no	0.740
11	constraint-first	4	0.15	yes	0.493
12	constraint-first	4	0.30	no	0.580
13	constraint-first	4	0.30	yes	0.540
14	constraint-first	4	0.45	yes	0.553
15	random	2	0.30	yes	0.513
16	random	2	0.45	no	0.447
17	random	4	0.30	yes	0.360
18	random	4	0.45	yes	0.440
19	random	6	0.30	no	0.440
20	random	6	0.30	yes	0.353
21	reversed	2	0.15	no	0.553
22	reversed	4	0.15	no	0.373
23	reversed	4	0.30	no	0.373
24	reversed	6	0.30	yes	0.320

Table 24: **The 20-design database family.** Every factor level appears. *7B db* is the Qwen2.5-7B database score, and *In 24* marks the nine designs shared with Table 23.

#	Order	T	f	Restate	7B db	In 24
1	canonical	2	0.15	yes	0.396	no
2	canonical	2	0.45	no	0.375	yes
3	canonical	2	0.45	yes	0.312	yes
4	canonical	4	0.15	no	0.365	yes
5	canonical	4	0.15	yes	0.385	no
6	canonical	4	0.30	yes	0.312	yes
7	constraint-first	2	0.15	no	0.406	yes
8	constraint-first	2	0.30	no	0.385	yes
9	constraint-first	2	0.45	yes	0.302	no
10	constraint-first	4	0.30	yes	0.302	yes
11	constraint-first	4	0.45	no	0.365	no
12	random	2	0.45	yes	0.271	no
13	random	4	0.15	yes	0.250	no
14	random	4	0.30	no	0.177	no
15	random	4	0.45	yes	0.146	yes
16	reversed	2	0.30	yes	0.281	no
17	reversed	2	0.45	no	0.250	no
18	reversed	4	0.15	no	0.188	yes
19	reversed	4	0.15	yes	0.177	no
20	reversed	4	0.45	yes	0.125	no