RETHINKING OVERTHINKING: EFFECTS OF RUN-TO-RUN INSTABILITY IN LARGE REASONING MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Reasoning Models solve complex tasks by generating intermediate steps before outputting a final answer. However, they often exhibit a failure mode known as *overthinking*, intuitively defined as generating excessively long reasoning chains for a problem the model could otherwise solve concisely. The literature traditionally diagnoses this by checking whether a single reasoning trace exceeds a fixed token budget. This single-trajectory view ignores the complexity of stochastic sampling. To overcome this limitation, we generate 100 independent runs per problem, revealing high variance in trace length. We reframe overthinking as *instability* and introduce a difficulty-aware, threshold-free framework to measure it. We define a model-relative measure of *empirical difficulty*, a new overthinking-quantifying metric, and we use a variance decomposition to isolate within-problem instability from between-problem difficulty. Applying this framework across three datasets (MATH500, GSM8K, DUMB500 Math) and seven models (Gemma4 and Qwen3, from 1.7B to 14B parameters), we find that models think $1.2-2.3\times$ more than they *could* and *should* in comparison to their ability. We show that runs longer than usual for their problem are systematically less likely to be correct, and for some models the relationship between accuracy and relative length takes an inverted-U shape. This instability does not scale monotonically with parameter count, and it concentrates on easy-to-medium problems before collapsing on the hardest ones. We suggest that future evaluation protocols report this run-to-run instability alongside mean length and accuracy.

1 INTRODUCTION

Large Reasoning Models (LRMs) solve math and logic problems by writing out intermediate reasoning steps before giving a final answer (Wei et al., 2022). While this helps on hard problems, it also causes a well-known failure mode called *overthinking* (Chen et al., 2024). The model sometimes writes an excessively long reasoning chain for a problem it could otherwise solve in a few lines. Traditionally, this is diagnosed by sampling a single trace and flagging whether it exceeds a fixed token budget (Huang et al., 2025; Aggarwal et al., 2026).

This diagnosis misses two things. First, LRMs are stochastic. Sampling the same problem twice under identical settings can produce traces of vastly different lengths, meaning a single long trace may just be an unlucky run rather than genuine overthinking. Second, a fixed token budget cannot separate "long because the problem is hard" from "long because the model is unstable". A budget of 2,000 tokens is generous for basic arithmetic but insufficient for a competition-level proof, as length only means something when compared to how much reasoning a problem actually requires.

We argue that overthinking is best understood not as a property of a single trace, but as instability– a decoding policy whose reasoning length is uncertain and occasionally runs away. For all seven models, within-problem trace length correlates negatively with correctness (significantly for five). A model's longer runs on a fixed problem are, on average, less likely to be correct than its shorter ones Hassid et al. (2025); Wu et al. (2025); Hochlehnert et al. (2025). This empirical pattern forms the starting point for our framework.

This distinction matters beyond diagnosis. A growing line of work proposes methods to mitigate overthinking, from length-conditioned training to early-exit decoding to activation steering. These

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.

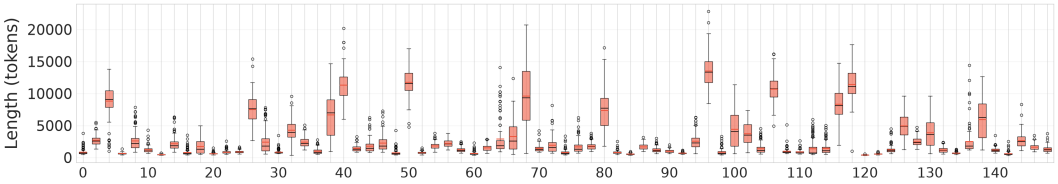


Figure 1: Per-problem distribution of trace length (in tokens) across $\mathcal{N}$=100 runs, for Gemma-4 12B on GSM8K. Each box shows the interquartile range and median for one problem; whiskers extend to 1.5× the interquartile range; circles mark individual runs beyond that range.

methods are typically evaluated on a single, near-deterministic generation per problem ($\mathtt{T}\approx 0$) by reporting the mean length and accuracy. However, instability only manifests across repeated samples. Because a seemingly mitigated model can appear stable under greedy decoding while remaining prone to runaway generations using stochastic sampling, we introduce the repeated-sampling protocol and instability metrics. These tools provide the essential diagnostic foundation required to accurately evaluate and guide future mitigation efforts.

We list below our main contributions:

1. **A difficulty-aware, threshold-free framework** for measuring reasoning-length instability, including a variance decomposition that separates within-problem instability from between-problem difficulty.
2. **A model-relative measure of empirical difficulty**, defined from a model's shortest correct runs and accuracy rather than a fixed label, and validated against official difficulty labels.
3. **An empirical study across seven open-weight models and three datasets**. We find that successful runs use 1.2–2.3× more tokens than the model's own best case, unusually long runs fail more often, and instability does not scale monotonically with model size and concentrates on easy-to-medium problems (second-guessing), not on hard ones.
4. **A sampling-budget analysis** showing that 50 generations per problem recover absolute metric values with negligible bias, while 25 suffice for difficulty scores, mean overthinking ratio, and detecting a $\geq 10\%$ change between two evaluations.

The code for reproducing our experiments is included in the supplementary material.

2 Related Work

We contextualize our variance-based approach by reviewing single-trace diagnostics, length-accuracy tradeoffs, self-verification, and overthinking mitigations.

Overthinking in reasoning models. While long reasoning traces improve complex task performance (Wei et al., 2022; Guo et al., 2025; Kimi Team, 2025), models often *overthink*, generating excessive tokens on simple problems (Chen et al., 2024; Cuadron et al., 2025; Fan et al., 2025). Existing diagnostics typically evaluate single traces against fixed budgets (Han et al., 2024), reference lengths, or early-answer positions (Chen et al., 2024; Kang et al., 2025). To shorten redundant reasoning, methods rely on RL penalties (Arora & Zanette, 2025; Kimi Team, 2025), compressed fine-tuning (Xia et al., 2025; Shen et al., 2025), prompt optimization (Yu et al., 2025), or inference-time early stopping (Han et al., 2024), including step-tagging (Belkhiter et al., 2026), bandit discriminators (Sun et al., 2025), and reward prediction (Zabounidis et al., 2026). Although some works (Huang et al., 2025; Hu et al., 2026) acknowledge single-trace limitations, most mitigations are still evaluated on individual runs. Our framework provides the multi-sample protocol needed to evaluate whether such methods genuinely reduce reasoning instability across generations.

Reasoning length and correctness. At the population level, accuracy and trace length generally follow an inverted U-shape (Wu et al., 2025; Su et al., 2025). Forcing extra computation improves accuracy initially before degrading it (Muennighoff et al., 2025; Ghosal et al., 2025; Gema et al., 2025; Shojaee et al., 2025). Studies using multiple samples (Hassid et al., 2025; Wu et al., 2025; Hochlehnert et al., 2025) show that shorter chains are more frequently correct. We show this inverted U-shape is not just a global trend, but occurs at the individual prompt level. By defining length

relative to each prompt's own distribution, we eliminate between-problem confounds and isolate genuine within-problem effects.

Stochasticity and variance in LLM evaluation. Repeated sampling has long served to improve or characterize model accuracy. Pass@k (Chen et al., 2021), self-consistency (Wang et al., 2023), and related approaches (Brown et al., 2024) already exploit multiple generations per problem. Song et al. (2024) show greedy and sampled decoding can yield conflicting conclusions, Madaan et al. (2024) quantify benchmark seed variance, and Miller (2024) urge reporting confidence intervals. For reasoning models specifically, Hochlehnert et al. (2025) find benchmark scores shift substantially across seeds and hardware, while Yuan et al. (2025) show greedy decoding diverges by thousands of tokens across precision settings. While these works observe or correct for sampling effects, we formalize a variance decomposition isolating within-problem instability from between-problem difficulty, treating variance as crucial to the definition of the overthinking phenomenon itself.

Self-verification in long traces. Chen et al. (2024) and Kang et al. (2025) observe that reasoning after an initial candidate answer rarely changes the outcome, acting mostly as confirmatory reflections, and is more frequent on easier tasks. Hidden-state analyses likewise show models keep verifying after internally encoding the correct answer (Zhang et al., 2025). We add distributional evidence that run-to-run instability peaks on easy-to-medium problems and falls on the hardest ones.

3 Evaluation Framework

This section defines the metrics used throughout the paper to characterize reasoning-length instability from repeated sampling. All are difficulty-aware and threshold-free by construction. We build them in three layers: (1) quantities recorded for a single generation, (2) quantities that summarize one problem across its $\mathcal{N}$ generations, and (3) quantities that summarize a full evaluation set.

3.1 Per-run quantities

For each problem p we draw $\mathcal{N}$=100 independent generations $\{x_1, \ldots, x_{\mathcal{N}}\}$ from a fixed decoding policy at fixed sampling settings (§4). For each generation we record its trace length $\ell(x_i)$, measured in tokens, and its correctness $c(x_i) \in \{0, 1\}$.

Mean and relative length. We write the mean length of a problem's $\mathcal{N}$ runs and each run's length relative to that mean as

$$\bar{\ell}(p) = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \ell(x_i), \qquad \tilde{\ell}(x_i) = \frac{\ell(x_i)}{\bar{\ell}(p)}, \tag{1}$$

where $\tilde{\ell}(x_i)$ expresses how long a single run is compared to the model's own typical behaviour on that specific problem, so runs can be compared across problems of very different absolute length without picking a token threshold.

3.2 Per-problem quantities

Win rate. $\mathrm{WR}(p) = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} c(x_i)$ is the fraction of a problem's $\mathcal{N}$ runs that are correct.

Empirical difficulty. We define a problem's *difficulty* as

$$d(p) = \frac{L_{10}(p)}{\mathrm{WR}_s(p)}, \tag{2}$$

where $L_{10}(p)$[1] is the mean length of the shortest decile of a problem's correct generations, and

$$\mathrm{WR}_s(p) = \frac{k(p) + 1}{\mathcal{N} + 2} \tag{3}$$

[1] $L_{10}(p)$ averages the shortest decile of a problem's $k(p)$ correct runs, a more stable reference than the single shortest run, to which it reduces when $k(p)$<10. Problems with $k(p)$=0 have no defined $d(p)$ and are excluded. Across all model-dataset pairs, only 0.3% of problems have $1 \leq k(p) < 10$ and 0.3% have $k(p)$=0, so the fallback has negligible influence on the reported statistics.

is an add-one-smoothed win rate and $k(p)$ the number of correct runs. At near-ceiling accuracy, $\mathrm{WR}_s(p)\approx 1$ and $d(p)\approx L_{10}(p)$, length alone. As accuracy falls, $d(p)$ grows past $L_{10}(p)$ by exactly the factor $1/\mathrm{WR}_s(p)$, so a model that succeeds half as often is scored twice as difficult at the same length. Concretely, two problems with the same $L_{10}(p)=1000$ tokens but win rates of 100% and 50% score $d(p)\approx 1010$ and $d(p)= 2000$ respectively.

We leave $d(p)$ unnormalized so it stays directly interpretable in raw tokens and comparable across datasets, models, and studies outside this one. We use it rather than a dataset's official difficulty label (e.g. MATH500's Level 1–5) since it is measured from each model's own behaviour rather than assigned once by a human, and it is not redundant with accuracy alone. Appendix C.1 details these design choices. Appendix C.2 validates $d(p)$ against MATH500's official Level and accuracy.

Overthinking ratio. For each problem we define the *overthinking ratio* as

$$\mathrm{OR}(p) = \frac{\mathrm{mean}_{i\,:\,c(x_i)=1}\,\ell(x_i)}{L_{10}(p)}. \tag{4}$$

We divide by $L_{10}(p)$,[1] the shortest-decile mean, since a single shortest correct run is not a reliable reference point and would make the ratio inherit that fragility. $\mathrm{OR}(p)\approx 1$ means a model's successful runs use about as much reasoning as its own best-case (shortest-decile) length on that problem. $\mathrm{OR}(p)>1$ means we are paying more than we should in terms of token-dollars for the same outcome.

Instability. We measure how a problem's trace length varies across its $\mathcal{N}$ runs with the *coefficient of variation*

$$\mathrm{CV}(p) = \frac{\mathrm{std}_i\,\ell(x_i)}{\bar{\ell}(p)}. \tag{5}$$

Because trace lengths are heavy-tailed, a handful of extreme runs can dominate $\mathrm{CV}(p)$. Hence, we also report a robust variant based on the median absolute deviation (MAD),

$$\mathrm{rCV}(p) = \frac{1.4826 \cdot \mathrm{MAD}_i\,\ell(x_i)}{\mathrm{median}_i\,\ell(x_i)}. \tag{6}$$

The constant 1.4826 scales the MAD to the standard deviation of normally distributed lengths, so $\mathrm{rCV}(p) \approx \mathrm{CV}(p)$ when lengths have no heavy tail. A $\mathrm{CV}(p)$ well above $\mathrm{rCV}(p)$ therefore flags a problem whose instability is driven by a few long runs rather than broadly spread.

Runaway rate. We isolate a single run trapped in an unusually long trace as the *runaway rate*, for a threshold multiplier κ (default $\kappa = 2$, sensitivity swept in Appendix C.4),

$$\mathrm{RR}(p,\kappa) = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \mathbb{1}[\ell(x_i) > \kappa \cdot \mathrm{median}_i\,\ell(x_i)]\,, \tag{7}$$

the fraction of runs on p that blow past κ times the problem's own median length. This is kept separate from general instability since $\mathrm{CV}(p)$ is symmetric and could also be inflated by ordinary two-sided spread.

3.3 Per-evaluation-set quantities

Median instability. We report the *median instability* over the evaluation set

$$\widehat{\mathrm{CV}}=\mathrm{median}_p\ \mathrm{CV}(p);\ \widehat{\mathrm{rCV}}=\mathrm{median}_p\ \mathrm{rCV}(p).$$

Variance decomposition and instability share. By the law of total variance, trace-length variance across an evaluation set splits into a between-problem term, reflecting that harder problems need more tokens on average, and a within-problem term, which is instability, as we use the word throughout this paper. We report the *instability share*

$$\rho = \frac{\mathbb{E}_p[\mathrm{Var}_i[\ell(x_i) \mid p]]}{\mathrm{Var}[\ell]} \in [0,1], \tag{8}$$

a single number summarizing how much of the length variation observed for a given model and dataset is instability rather than difficulty.

Length-correctness correlation. We correlate a generation's length against its correctness two ways. First, the pooled correlation is the ordinary Pearson correlation over every generation, ignoring problem identity,

$$\text{P-corr} = \text{corr}_i\big(\ell(x_i),\, c(x_i)\big), \tag{9}$$

which mixes a between-problem effect (harder problems need more tokens and have lower accuracy) with the within-problem effect that constitutes genuine overthinking. We isolate the latter by first demeaning each generation's length and correctness by its own problem's mean, then correlating these centered values pooled across all problems,

$$\text{W-corr} = \text{corr}_{p,i}\big(\ell(x_i) - \bar{\ell}(p),\; c(x_i) - \text{WR}(p)\big). \tag{10}$$

Pooling before correlating, rather than averaging a per-problem correlation, means a problem whose $\mathcal{N}$ generations all share the same outcome is not dropped.

4 Experimental Setup

Datasets. Our evaluation relies on three maths reasoning benchmarks spanning a broad difficulty spectrum, DUMB500 Math (Pu et al., 2025), GSM8K (Cobbe et al., 2021), and MATH500 (Lightman et al., 2023), detailed in Appendix A.1 (other domains are left to future work, §G).

Models. The open-weight reasoning models evaluated (listed in Table 2) span from 1.7B to 14B parameters (restricted by single-GPU hardware, Appendix G) and belong to two model families, Gemma4 (Gemma Team, 2026) and Qwen3 (Yang et al., 2025a).

Generation Hyperparameters. For each problem and dataset, we draw $\mathcal{N}{=}100$ independent generations per model. We set the maximum token limit to $30{,}000$ and fix the random seed to 42 across all runs to ensure reproducibility. Decoding parameters follow the official recommendations of each model family, are kept fixed throughout (Appendix G), and are listed in Appendix A.2.

Correctness Judging. The final answers are first evaluated via exact matching against the ground truth. For responses that do not yield an exact match, correctness is evaluated using an LLM judge (`Qwen2.5-7B-Instruct`). To reduce judge variance and keep the evaluation robust, each response is graded across 3 independent judge runs, with the final label assigned by majority vote.

5 Results

We organize our findings around five questions. We first measure the instability of reasoning length (§5.1). We then examine whether this instability affects correctness (§5.2) and how many tokens are wasted during successful runs (§5.3). Next, we identify where instability concentrates along the difficulty axis (§5.4). Finally, we discuss the implications of these findings for evaluating overthinking mitigation methods (§5.5). All results include a bootstrap confidence interval and an FDR-corrected significance test (Appendix F.2, F.3). Appendix B.1 confirms every model is competent enough on every dataset for this instability analysis to be meaningful.

5.1 How Unstable Is Reasoning Length?

We analyse the stability of reasoning trace lengths by generating $\mathcal{N}{=}100$ rollouts per problem using identical prompts and decoding settings. Figure 1 illustrates the resulting length distributions for Gemma-4-12B on GSM8K. Some problems produce tightly clustered traces while others spread over thousands of tokens with isolated runs extending far beyond the main distribution. This pattern holds across all other evaluated models and datasets (Appendix E). When aggregated across problems, the median coefficient of variation $\widehat{\text{CV}}$ indicates that a typical run deviates from the mean length of its specific problem by 12 to 30% (Table 1).

Instability can manifest in two ways: lengths might scatter evenly around the mean, or most runs might remain consistent while a few unusually long outliers occur. To distinguish between these scenarios, we compare the standard coefficient of variation (CV) with its robust counterpart (rCV), which downweights extreme runs and matches CV when lengths are normally distributed. If CV is larger, part of the instability stems from a few exceptionally long runs. CV exceeds rCV for 64–84%

Model	Dataset	Mean length	d mean	ρ	$\widehat{\mathrm{CV}}$	rCV	$\overline{OR}$	P-corr	W-corr
Gemma-4-E2B	MATH500	2377	1642	0.126	0.151	0.130	1.288	-0.509	-0.151
	GSM8K	974	863	0.454	0.102	0.089	1.212	-0.293	-0.182
	Dumb500M	586	426	0.178	0.131	0.121	1.326	-0.195	-0.188
Gemma-4-E4B	MATH500	1789	824	0.187	0.282	0.194	1.827	-0.418	-0.078
	GSM8K	737	346	0.474	0.304	0.141	2.225	-0.153	-0.112
	Dumb500M	519	287	0.245	0.229	0.194	1.713	-0.086	-0.108
Gemma-4-12B	MATH500	3839	1644	0.119	0.277	0.260	1.651	-0.404	-0.117
	GSM8K	2777	969	0.261	0.415	0.308	2.332	-0.284	-0.070
	Dumb500M	770	421	0.311	0.211	0.170	1.721	-0.463	-0.102
Qwen3-1.7B	MATH500	4922	2746	0.227	0.237	0.175	1.466	-0.523	-0.333
	GSM8K	2337	1501	0.330	0.245	0.204	1.547	-0.438	-0.203
	Dumb500M	1673	990	0.653	0.214	0.193	1.480	-0.534	-0.621
Qwen3-4B	MATH500	4412	2618	0.138	0.181	0.160	1.387	-0.432	-0.102
	GSM8K	2327	1485	0.242	0.205	0.203	1.506	-0.377	-0.148
	Dumb500M	1486	975	0.187	0.191	0.177	1.419	-0.099	-0.041
Qwen3-8B	MATH500	4671	2739	0.169	0.183	0.170	1.394	-0.500	-0.425
	GSM8K	2373	1455	0.379	0.224	0.198	1.521	-0.391	-0.290
	Dumb500M	1745	1107	0.500	0.199	0.188	1.457	-0.274	-0.313
Qwen3-14B	MATH500	4111	2416	0.185	0.174	0.160	1.378	-0.347	-0.095
	GSM8K	1971	1179	0.283	0.203	0.190	1.494	-0.330	-0.195
	Dumb500M	1155	719	0.239	0.222	0.208	1.520	-0.005	0.047

Table 1: Instability and failure-prediction statistics by model and dataset. $\boldsymbol{d}$ mean is the geometric mean of $\boldsymbol{d(p)}$ (§3.2); $\boldsymbol{\rho}$ is the instability share (§3.3); $\widehat{\mathbf{CV}}$ is the aggregate stability index, while **rCV** is its robust counterpart (§3.3); $\overline{\boldsymbol{OR}}$ is the average overthinking ratio; **P-corr** and **W-corr** are the pooled and within-problem length-correctness correlations (§3.3). Confidence intervals and significance tests for every entry are provided in Appendices F.2 and F.3. All statistics except OR are computed across all generations, including truncated ones (which account for only 0.15% of the total).

of problems, against 50% expected without a heavy tail, and by more than $2\times$ for up to a quarter of them (Appendix C.3). Indeed, the runaway rate (RR) shows that 2.4 to 10.5% of runs exceed 1.5 times their problem's median length, and only 0.4 to 3.9% exceed twice the median (Appendix C.4).

Instability is also independent of both absolute trace length and model size. Figure 2 plots $\widehat{\mathrm{CV}}$ against mean length with one point per model-dataset pair. The two variables do not correlate. For example, on MATH500, Gemma-4-12B produces shorter traces than every Qwen3 model (3,839 tokens versus 4,111 to 4,922), but has the highest overall $\widehat{\mathrm{CV}}$ (0.30 versus 0.19 to 0.22 for Qwen3). Meanwhile, Gemma-4-E2B produces similarly short traces but is the most stable model of all. Model scale does not determine stability either. The smallest Gemma-4 model is the most stable and the largest is the least stable. Statistical tests show that 18 out of 21 pairwise $\widehat{\mathrm{CV}}$ comparisons are significant, and the exceptions occur entirely among Qwen3 models (Appendix F.3). Instability appears to depend on the training recipe of a model rather than its parameter count.

5.2 Does Instability Cost Accuracy?

Instability costs accuracy, not only compute, since on the same problem a run that goes unusually long is more likely to be wrong. Showing this requires separating length variation caused by instability from that caused by problem difficulty. A naive correlation between trace length and correctness across all runs (P-corr) yields a negative value for every model-dataset pair, reaching down to -0.53 (Table 1). However, this overstates the effect because harder problems naturally require longer traces and have higher failure rates. For instance, Gemma-4-12B shows a P-corr of -0.46 but exhibits no significant correlation once problem difficulty is factored out. To isolate instability, we use W-corr (§3.3), which measures the correlation exclusively among different runs of the same problem. W-corr remains negative, with a 95% confidence interval excluding zero for five of the seven evaluated models (Table 11).

Binning runs by relative length $\tilde{\ell}(x_i)$, Figure 3 shows where this penalty occurs. Accuracy changes little across typical lengths and is lowest in the longest decile for six of seven models, where the error rate is up to $3.4\times$ higher than in the other nine deciles (from 1.5% to 5.1% for Qwen3-8B; from 4.5% to 9.6% for Qwen3-1.7B). Failures thus concentrate in the runaway tail rather than in

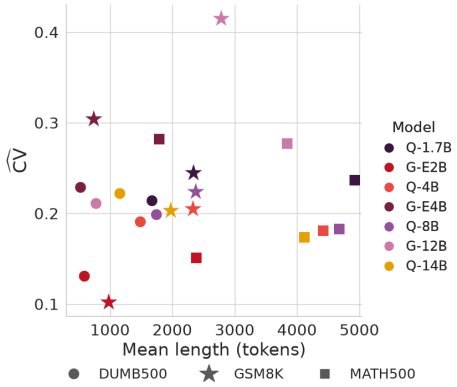


Figure 2: Mean length vs. $\widehat{\mathrm{CV}}$, one point per model-dataset pair. It shows that level and instability are two distinct axes.

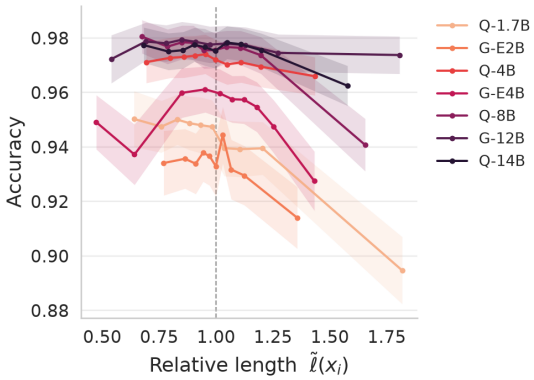


Figure 3: Accuracy as a function of relative length, pooled across each model's datasets. Accuracy declines at the longest relative length bin.

moderately long reasoning, which is exactly what the instability metrics capture. The effect is not universal (Gemma-4-12B stays flat, and Gemma-4-E4B's interval touches zero) and it is moderate, since most unusually long runs still reach the correct answer. Accuracy cannot see these inefficient successes (Wu et al., 2025), so length instability must be measured directly. Conditioning on the problem also separates our analysis from prior work reporting an inverted-U relationship between length and accuracy across whole datasets (Wu et al., 2025; Xia et al., 2026), where instability and problem difficulty are mixed.

5.3 How Much Do Models Overthink When They Succeed?

Existing overthinking metrics compare a reasoning trace against external references such as a fixed token budget (Huang et al., 2025), a baseline model length, or the position of the first correct answer (Chen et al., 2024). These references quantify how much longer a trace is than a hypothetical standard *would* require. Our Overthinking Ratio (OR) instead compares successful runs against the demonstrated capability of the model on that specific problem (§3.2). This isolates how much more the model thinks than it demonstrably *could* and *should*.

OR exceeds 1 for every tested model and dataset and ranges from 1.21 to 2.33 (Table 1). Qwen3 models cluster tightly between 1.38 and 1.55. In contrast, Gemma4 E4B and 12B consume almost $2\times$ as many tokens on average as their shortest correct ones. At the problem level, OR correlates with $\mathrm{CV}(p)$, yielding Spearman coefficients between 0.62 and 0.90 (Appendix C.5). This monotonic and saturating relationship suggests both metrics stem from outlier-heavy runs. However, the metrics remain distinct because CV evaluates every run while OR isolates only correct outputs.

OR is higher on GSM8K than on MATH500 for six of the seven models, with Gemma-4-E2B being the sole exception. Models proportionally inflate successful runs more on the easier benchmark. OR also shows no reliable correlation with the official difficulty levels in MATH500 (Spearman correlation in $[-0.14, +0.22]$). Wasted effort on a successful run does not strictly depend on the designated difficulty label of a problem. This implies that instability might not scale with difficulty in the expected direction. We explore this dynamic directly in the next section.

5.4 Where Does Instability Concentrate?

Trace lengths vary across an evaluation set for two main reasons: harder problems demand longer reasoning, and independent runs on the same problem naturally diverge. The instability share ρ isolates these two factors. Table 1 shows that ρ ranges between 0.12 and 0.65. While inherent difficulty explains the majority of length variance, pure run-to-run instability accounts for a substantial portion. To analyse how this instability share shifts alongside problem difficulty, we divide the problems for each model into terciles of $d(p)$ pooled across the three datasets. As shown in Figure 5, these terciles reflect perceived model difficulty rather than rigid dataset boundaries, though they naturally

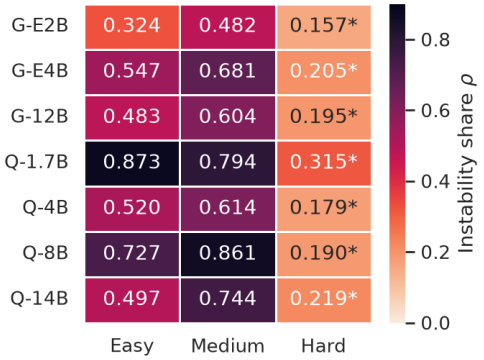


Figure 4: Instability share ρ by empirical-difficulty tercile. Starred cells mark a significant Easy-to-Hard drop ($q < 0.05$, Appendix F.3).

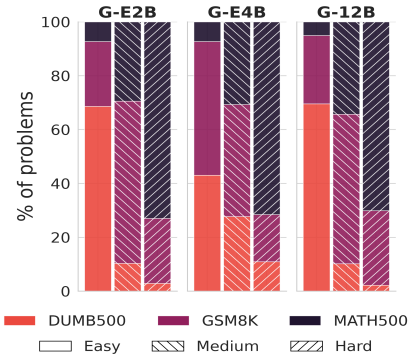


Figure 5: Dataset composition per difficulty tercile (Gemma4 family). Terciles reflect perceived model difficulty rather than rigid dataset boundaries.

group problems by their source dataset. We then compute ρ within each specific tercile (Figure 4). Every model exhibits its lowest instability share in the Hard tercile, landing between 0.16 and 0.31. A paired bootstrap test shows that the drop from Easy to Hard is statistically significant for all seven models with $q<0.05$ (Appendix F.3). ρ also peaks in the Medium tercile for six of the seven models.

If this instability originated from genuine model struggle, the variance should scale up with problem difficulty. Alternatively, a second-guessing theory posits that instability arises when a model repeatedly double-checks a solution it has already found. Just as a person might doubt a straightforward answer and search for hidden complexity, the model explores alternative paths because the initial solution appears too easy, leading it to second-guess its own correct intuition and generate unnecessary computational effort. Under this theory, instability should peak on problems the model can mostly solve and drop off on extremely hard problems where the model must fully commit to a single solution path. Our empirical data support the second-guessing theory. Instability maximizes where models are competent but uncertain and then collapses on the hardest tasks. The behaviour of OR in §5.3 corroborates this conclusion since successful runs inflate considerably more on GSM8K than on MATH500. These results align with existing trace-level observations that reflection steps are mostly confirmatory and appear more often in simpler datasets (Kang et al., 2025). This offers a testable prediction for future research. Appendix D shows that long correct runs typically state their answer early and spend the rest of the trace re-checking it.

5.5 Implications for Evaluating Overthinking Mitigation

Instability Is Not Length. Current evaluations of overthinking mitigation typically compare mean trace length and accuracy before and after an intervention. However, mean length and its instability are two independent axes (§5.1), so judging a model by mean length alone ignores run-to-run variance. Two models can produce the same average trace length while differing sharply in predictability, and a mitigation method that merely lowers the mean while leaving the volatile upper tail intact does not address the actual instability.

Greedy Decoding vs. Sampling. Many existing mitigation studies rely on evaluating a single greedy generation ($\texttt{T}=0$) per problem (Yang et al., 2025b; Xia et al., 2025; Guan et al., 2026). This single-generation approach is flawed because it is impossible to measure instability without repeated sampling. A greedy trace also does not represent a normal draw from a model's true length distribution, but is instead heavily skewed towards degenerate outputs. Our data shows that greedy decoding hits the 30K-token limit $23\times$ more often than stochastic sampling, and 95% of these truncated greedy traces contain repetition loops (Appendix B.2). While some recent literature (Huang et al., 2025; Hu et al., 2026) has recognized these issues and moved towards stochastic sampling, studies often still summarize just a single run or average a small handful of runs.

How Many Samples Are Needed. To determine the necessary sample size, we run a subsampling sweep over budgets k. While aggregate dataset-level metrics stabilize reasonably well, per-problem means are highly volatile at low k. At $k=5$, a sample mean can land anywhere from roughly 0.87 to

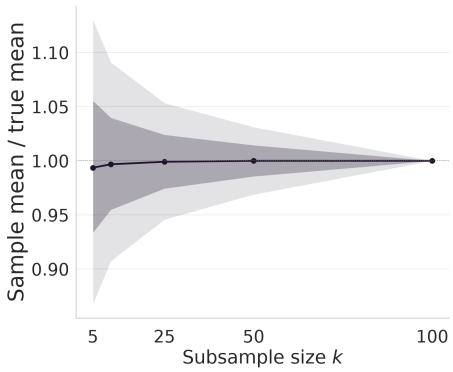


Figure 6: Per-problem sample-mean length relative to that problem's own true ($\mathcal{N}$=100) mean, pooled across all models and datasets. Band: IQR (dark) and 10th–90th percentile (light).

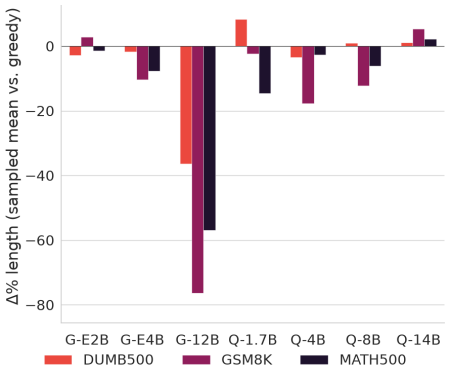


Figure 7: $\Delta\%$ length under sampling vs. greedy decoding ($\Delta\%$=(sampled−greedy)/greedy), one bar per model-dataset pair (Table 5).

1.13× the true mean of its problem (Figure 6). Variance-based metrics such as $\widehat{\mathrm{CV}}$ and ρ converge much more slowly than mean length and still carry substantial bias at k=10. Comparing two evaluations is cheaper than estimating absolute values, because the small-sample bias pushes both in the same direction. When two evaluations are subsampled independently, k=25 recovers the direction of a $\geq$10% difference in d, $\overline{OR}$ or $\widehat{\mathrm{CV}}$ in over 99.5% of subsamples, although differences below 10% remain unreliable even at k=50 (Table 9).

Suggestions for Future Evaluation. Box 1 condenses the framework into a minimal protocol, listing for each question the metric to report, how to read it, and the minimum number of generations per problem $\mathcal{N}$ needed to estimate it reliably.

Box 1. A recipe for detecting overthinking: reasoning length instability evaluation in LRMs.

Sampling. $\mathcal{N}$=50 generations per problem at the recommended T>0. $\mathcal{N}$=25 suffices to compare two evaluations, e.g. before/after mitigation (Table 9). We suggest reporting accuracy, mean length, AND:

Question	Metric	How to read it	$\mathcal{N}$
How hard is each problem?	$d(p)$	Tokens the model needs, inflated by $1/\mathrm{WR}_s$.	25
How much extra reasoning?	$\overline{OR}$	If >1: more tokens than the model *could* use.	25
How unstable is reasoning length on a problem? ($\widehat{\mathrm{CV}} \gg \widehat{\mathrm{rCV}}$ and RR>0 ⇒ a few runaway runs, not broad scatter)			
- How much?	$\widehat{\mathrm{CV}}$	Spread of all runs; a few long runs can inflate it.	25
- Typical runs?	$\widehat{\mathrm{rCV}}$	Spread of the bulk; ≈CV if no tail.	25
- How often it blows up?	RR	Share of runs >2× the problem's median.	25
Instability or difficulty?	ρ	Share of length variance that is run-to-run.	50
Does instability cost accuracy?	W-corr	If <0: on the same problem, longer runs fail more.	50

Reading a mitigation. It works if accuracy holds while $\overline{OR}$, $\widehat{\mathrm{CV}}$, $\widehat{\mathrm{rCV}}$ and RR drop and W-corr moves →0. Lower mean length with these metrics unchanged only shifts the distribution, leaving its tail intact.

6 Conclusion and Future Work

This paper reframes overthinking not as a single-trace failure but as run-to-run instability. Using 100 generations per problem across seven models and three datasets, our threshold-free metrics show that a model's unusually long runs are systematically less likely to succeed, and that instability concentrates on problems the model can already solve, pointing to second-guessing rather than genuine struggle. Since mean length and accuracy cannot distinguish a stable model from a volatile one, we recommend reporting run-to-run instability under stochastic decoding (T>0) following Box 1, with 50 generations per problem for absolute values and 25 to detect a $\geq$10% change between two evaluations. Future work should test whether proposed mitigations suppress this variance, and extend the analysis beyond maths, e.g. to code generation.

AI USE STATEMENT

AI tools were utilized during the research and writing of this paper in the following ways:

- We used Gemini 3.1 Pro during the writing phase to identify grammatical errors and refine the prose for better clarity and flow.
- We employed Claude Code (Sonnet 5) to assist with code debugging, script execution, and monitoring of our experimental pipelines.
- We used `Qwen2.5-7B-Instruct` as an automated judge for correctness grading of model responses that did not yield an exact or symbolic match with the ground truth. Details of this setup are provided in §4.

ETHICS STATEMENT

This work analyses reasoning-length instability in open-weight language models on public, widely-used math benchmarks (DUMB500 Math, GSM8K, MATH500), involving no human subjects or sensitive/personal data. All models and datasets are publicly released and used in accordance with their respective licenses and original citations (§4).

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our empirical findings, Section 4 details our experimental setup, including generation hyperparameters and random seeding. The seed (42) initializes the random number generator of the vLLM engine once per run and also fixes the subsampling of problems. It is not reset between generations. The $\mathcal{N}{=}100$ samples of each problem are drawn in a single request, and every sequence receives its own random draw at each decoding step. The seed therefore makes each run reproducible without making its generations identical or correlated. Appendix A.1 describes the datasets, Appendix A.2 the models and decoding hyperparameters, and Appendix A.3 the correctness-judging protocol, including the exact judge prompt template. The supplementary material contains the complete codebase, with all scripts for data generation and experiment execution, and a `README.md` describing the purpose of each file.

REFERENCES

Pranjal Aggarwal, Seungone Kim, Jack Lanchantin, Sean Welleck, Jason E Weston, Ilia Kulikov, and Swarnadeep Saha. Optimalthinkingbench: Evaluating over and underthinking in llms. In *International Conference on Learning Representations*, volume 2026, pp. 8886–8918, 2026.

Daman Arora and Andrea Zanette. Training language models to reason efficiently. *arXiv preprint arXiv:2502.04463*, 2025.

Yannis Belkhiter, Seshu Tirupathi, Giulio Zizzo, and John D. Kelleher. TRACES: Tagging reasoning steps for adaptive cost-efficient early-stopping. In *Proceedings of the Third Conference on Language Modeling (COLM)*, 2026. URL `https://arxiv.org/abs/2604.21057`.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do NOT think that much for 2+3=?: On the overthinking of o1-like LLMs, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, Nicholas Thumiger, Aditya Desai, Ion Stoica, Ana Klimovic, Graham Neubig, and Joseph E. Gonzalez. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks, 2025.

Chenrui Fan, Ming Li, Lichao Sun, and Tianyi Zhou. Missing premise exacerbates overthinking: Are reasoning models losing critical thinking skill? *arXiv preprint arXiv:2504.06514*, 2025.

Aryo Pradipta Gema, Alexander Hägele, Runjin Chen, Andy Arditi, et al. Inverse scaling in test-time compute. *arXiv preprint arXiv:2507.14417*, 2025.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026. URL `https://arxiv.org/abs/2607.02770%7D`.

Soumya Suvra Ghosal, Souradip Chakraborty, Avinash Reddy, Yifu Lu, Mengdi Wang, Dinesh Manocha, Furong Huang, Mohammad Ghavamzadeh, and Amrit Singh Bedi. Does thinking more always help? mirage of test-time scaling in reasoning models. In *Advances in Neural Information Processing Systems*, 2025.

Weixin Guan, Liang Li, Jiapeng Liu, Bing Li, Peng Fu, Chengyang Fang, Xiaoshuai Hao, Can Ma, and Weiping Wang. Mitigating overthinking in large reasoning language models via reasoning path deviation monitoring. *arXiv preprint arXiv:2603.14251*, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z.

Tingxu Han, Zhenting Wang, Chunrong Fang, Shiyu Zhao, Shiqing Ma, and Zhenyu Chen. Token-budget-aware LLM reasoning. *arXiv preprint arXiv:2412.18547*, 2024.

Michael Hassid, Gabriel Synnaeve, Yossi Adi, and Roy Schwartz. Don't overthink it. preferring shorter thinking chains for improved LLM reasoning. *arXiv preprint arXiv:2505.17813*, 2025.

Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. *arXiv preprint arXiv:2504.07086*, 2025.

Chenzhi Hu, Qinzhe Hu, Yuhang Xu, Junyi Chen, Ruijie Wang, Shengzhong Liu, Jianxin Li, Fan Wu, and Guihai Chen. Smartthinker: Progressive chain-of-thought length calibration for efficient large language model reasoning. *arXiv preprint arXiv:2603.08000*, 2026.

Yao Huang, Huanran Chen, Shouwei Ruan, Yichi Zhang, Xingxing Wei, and Yinpeng Dong. Mitigating overthinking in large reasoning models via manifold steering. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Liwei Kang, Yue Deng, Yao Xiao, Zhanfeng Mo, Wee Sun Lee, and Lidong Bing. First try matters: Revisiting the role of reflection in reasoning models. *arXiv preprint arXiv:2510.08308*, 2025.

Kimi Team. Kimi k1.5: Scaling reinforcement learning with LLMs. *arXiv preprint arXiv:2501.12599*, 2025.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023.

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

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

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Xiao Pu, Michael Saxon, Wenyue Hua, and William Yang Wang. Thoughtterminator: Benchmarking, calibrating, and mitigating overthinking in reasoning models, 2025.

Yi Shen, Jian Zhang, Jieyun Huang, Shuming Shi, et al. DAST: Difficulty-adaptive slow-thinking for large reasoning models. *arXiv preprint arXiv:2503.04472*, 2025.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *arXiv preprint arXiv:2506.06941*, 2025.

Yifan Song, Guoyin Wang, Sujian Li, and Bill Yuchen Lin. The good, the bad, and the greedy: Evaluation of LLMs should not ignore non-determinism. *arXiv preprint arXiv:2407.10457*, 2024.

Jinyan Su, Jennifer Healey, Preslav Nakov, and Claire Cardie. Between underthinking and overthinking: An empirical study of reasoning length and correctness in LLMs, 2025.

Renliang Sun, Wei Cheng, Dawei Li, Haifeng Chen, and Wei Wang. Stop when enough: Adaptive early-stopping for chain-of-thought reasoning. In *arXiv preprint arXiv:2510.10103*, 2025. URL https://arxiv.org/abs/2510.10103.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed H. Chi, Quoc V. Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Yuyang Wu, Yifei Wang, Ziyu Ye, Tianqi Du, Stefanie Jegelka, and Yisen Wang. When more is less: Understanding chain-of-thought length in LLMs, 2025.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. TokenSkip: Controllable chain-of-thought compression in LLMs. *arXiv preprint arXiv:2502.12067*, 2025.

Yinan Xia, Haotian Zhang, and Huiming Wang. Balancing the reasoning load: Difficulty-differentiated policy optimization with length redistribution for efficient and robust reinforcement learning, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Chenxu Yang, Qingyi Si, Yongjie Duan, Zheliang Zhu, Chenyu Zhu, Qiaowei Li, Minghui Chen, Zheng Lin, and Weiping Wang. Dynamic early exit in reasoning models, 2025b.

Ye Yu, Yaoning Yu, and Haohan Wang. PREMISE: Scalable and strategic prompt optimization for efficient mathematical reasoning in large models. In *arXiv preprint arXiv:2506.10716*, 2025. URL https://arxiv.org/abs/2506.10716.

Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wentian Zhao, Kun Wan, Jing Shi, Xia Hu, and Zirui Liu. Understanding and mitigating numerical sources of nondeterminism in LLM inference. In *Advances in Neural Information Processing Systems*, volume 38, 2025. arXiv:2506.09501.

Renos Zabounidis, Aditya Golatkar, Michael Kleinman, Alessandro Achille, Wei Xia, and Stefano Soatto. Re-FORC: Adaptive reward prediction for efficient chain-of-thought reasoning. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2511.02130.

Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification. *arXiv preprint arXiv:2504.05419*, 2025.

A Additional experimental details

A.1 Datasets

1. *DUMB500 (Math)* serves as our trivial baseline. The original DUMB500 dataset (Pu et al., 2025) consists of 500 instances divided into multiple domain splits (easy math, reasoning, code, and task problems). We isolate only the math split, which comprises 113 problems specifically designed to test overthinking rather than complex reasoning capacity. Because these problems are intentionally straightforward, an ideal model should respond concisely rather than deploy lengthy reasoning chains.
2. *GSM8K* Cobbe et al. (2021) represents our easy benchmark, consisting of grade-school word problems that require basic multi-step quantitative reasoning. We evaluate on a random sample of 150 problems.
3. *MATH500* Lightman et al. (2023) represents our medium-to-hard benchmark, covering challenging high-school and competition mathematics. To ensure balanced coverage across the difficulty spectrum, we randomly sample 150 problems stratified by official difficulty, exactly 30 problems per level, from Level 1 to Level 5. This official stratification is used only to build a balanced evaluation set; every difficulty-conditioned analysis in the paper itself uses $d(p)$ (§3), not these labels.

A.2 Models

Qwen3 models are sampled with a temperature of 0.6 and top-p of 0.95 (Guo et al., 2025), while Gemma4 models use a default temperature of 1.0 alongside their default top-p and top-k settings **?**. Table 2 shows the sizes of the evaluated models.

Family	Sizes evaluated
Gemma4	E2B, E4B, 12B
Qwen3	1.7B, 4B, 8B, 14B

Table 2: Open-weight models evaluated across parameter scales.

A.3 Correctness Evaluation Judge Prompt

Here we detail the exact prompt template used for evaluating response correctness with `Qwen2.5-7B-Instruct` as the judge model. The judge is provided with only the final answer text (i.e., the text generated after the end-of-reasoning token) rather than the full reasoning trace. This approach avoids context-window limitations and potential judge bias regarding the length or verbosity of the final answer itself.

Correctness Evaluation Judge Prompt

You are grading whether a model's answer to a problem is CORRECT, compared against a ground truth answer. Judge the underlying claim/conclusion, not surface form.

Ground truth answer: {gt}

{boxed_hint}

Model's full response: {pred}

How to grade:
1. If the response contains a \boxed{...} answer, treat that as the model's final answer. Otherwise, find the model's final stated conclusion anywhere in the response.

2. Decide if that final answer is EQUIVALENT to the ground truth. Equivalent includes:

- Different but equal numeric forms.
- Equivalent expressions or reordered lists/sets.
- Numerals vs. spelled-out numbers.
- Equivalent yes/no statements.
- Minor formatting differences.

3. Ignore the correctness of intermediate reasoning-only judge the final answer.

4. If there are multiple conflicting final answers, use the last one.

5. If no determinable final answer exists, grade False.

First, briefly reason step by step (1-3 sentences) about what the model's final answer is and whether it matches. Then, on the LAST line, output exactly:

Verdict: True
or
Verdict: False
Nothing after that line.

B BASELINES

B.1 BASELINE ACCURACY

Table 3 reports baseline accuracy for every model-dataset pair, under both exact-match and judge-assisted scoring. This establishes that every model in our study is competent on every dataset, so the instability we characterize in §5 is not an artefact of models that mostly fail.

Ground-truth corrections. DUMB500 Math has a less standardized answer format and a small number of incorrect ground-truth labels, which is why that dataset shows the largest exact-match-to-judged accuracy gap in Table 3. A manual audit of the DUMB500 Math ground truths, prompted by cases where every model in our study answered identically and consistently disagreed with the labelled ground truth, surfaced four unambiguous labelling errors, listed in Table 4. We corrected these four labels for all seven models before computing the accuracy figures reported here and throughout §5–§5.5. An incorrect label penalizes every model equally, so it does not by itself bias model-vs-model comparisons, but it does distort the absolute accuracy numbers in Table 3 and, more importantly, can distort a problem's measured overthinking/instability metrics (§3) for reasons that have nothing to do with the model's actual behaviour on it.

B.2 GREEDY VERSUS SAMPLED DECODING: LENGTH, TRUNCATION AND REPETITION LOOPS

§5.5 argues that a single greedy generation is not a representative draw from a model's length distribution. This appendix gives the full comparison (Table 5), describes the repetition-loop detector behind its loop-rate columns, and shows representative degenerate traces.

Loop detector. We flag a trace as containing a *repetition loop* if either of two conditions holds. (i) An n-gram sweep finds a span of at least n words repeating verbatim three or more times consecutively, for some n in a sweep from 6 upward; we record the word index where the first such span begins (*loop start*) and its period. (ii) Duplicate-line coverage, the fraction of non-empty lines that exactly repeat an earlier line in the same trace, exceeds 0.4. The two are complementary: condition (i) misses *rumination* loops, where the model re-derives the same quantity with slightly different wording on each pass so that no long span repeats verbatim, while condition (ii) misses collapses written as one unbroken paragraph, where there are too few line breaks for line-level duplication to register. We report the union. Loops typically begin early: in the verbatim cases we inspected, the first repeating span starts between word 700 and word $2{,}800$ of a trace running to roughly $10{,}000$–$26{,}000$ words, so most generated tokens are the loop itself.

Model	Dataset	Acc. (%) exact match	Acc. (%) w/ judge
Gemma-4-E2B	MATH500	90.55	93.64
	GSM8K	90.98	91.63
	DUMB500M	89.27	97.64
Gemma-4-E4B	MATH500	91.59	96.32
	GSM8K	93.15	94.14
	DUMB500M	89.29	97.81
Gemma-4-12B	MATH500	98.97	99.30
	GSM8K	96.92	97.24
	DUMB500M	92.71	98.58
Qwen3-1.7B	MATH500	93.59	95.22
	GSM8K	91.33	91.81
	DUMB500M	82.63	98.63
Qwen3-4B	MATH500	97.93	98.70
	GSM8K	95.71	95.87
	DUMB500M	83.99	99.55
Qwen3-8B	MATH500	98.19	98.63
	GSM8K	96.25	97.04
	DUMB500M	78.70	99.26
Qwen3-14B	MATH500	98.20	98.75
	GSM8K	96.38	96.93
	DUMB500M	83.06	99.35

Table 3: Baseline accuracy summary across models and datasets. Gemma4 models are the instruction-tuned (`-it`) checkpoints and Qwen3 models the released hybrid checkpoints run in thinking mode, all evaluated at the settings specified in §4. DUMB500M denotes the DUMB500 Math dataset split, after the ground-truth corrections in Table 4. Judge-assisted accuracy uses majority-vote aggregation over three independent runs of the LLM judge (`Qwen2.5-7B-Instruct`).

Prompt	Old GT	New GT
"You have 10 dollars and buy 3 cookies for 2 dollars each. How much money do you have left?"	2 dollars	4 dollars
"1, 1, 2, 3, 5, (?), 13 — What number is missing?"	7	8
"Can I divide by zero if I ask nicely?"	yes	no
"How many meters per second are in 60 kilometers per hour?"	1000	16.67

Table 4: The four DUMB500 Math ground-truth corrections applied uniformly across all seven models. All four are unambiguous arithmetic/factual errors in the original dataset release rather than judgement calls, verified independently of any model's output.

Examples. The clearest case is a verbatim arithmetic re-check repeating unchanged until the token limit (Gemma-4-12B, GSM8K; loop start at word $2{,}467$, period 20, duplicate-line coverage 0.90):

```
340 * 194 = 65,960.
Wait, let me do 340 * 194 again.
340 * 194 = 340 * 194
340 * 194 = 65,960.
Wait, let me do 340 * 194 again.
```
[...repeats to the token limit]

A second Gemma-4-12B GSM8K trace shows the rumination variant: it re-reads the same clause of the problem and re-derives the same numbers, but the surrounding phrasing changes on each pass, so no 20-word span repeats verbatim and only the line-level condition fires ($1{,}464$ of $1{,}611$ non-empty lines, 91%, duplicate an earlier line). A Qwen3-8B GSM8K trace shows the opposite failure: it runs on as a single unbroken paragraph, with duplicate-line coverage 0.000, yet repeats a verbatim 35-word span about the same buy-one-get-one-free ambiguity, and only the n-gram sweep catches

		Mean length			Truncated		Loops		Excl. truncated		
Model	**Dataset**	**Greedy**	**Sampled**	**Δ%**	**G**	**S**	**G**	**S**	**Greedy**	**Sampled**	**Δ%**
Gemma-4-E2B	MATH500	2411	2377	-1.4	0.67	0	100 (1)	–	2226	2377	+6.8
	GSM8K	947	974	+2.8	0	0.01	–	100 (1)	947	972	+2.6
	Dumb500M	604	586	-2.9	0	0	–	–	604	586	-2.9
Gemma-4-E4B	MATH500	1936	1789	-7.6	0	0	–	–	1936	1789	-7.6
	GSM8K	823	737	-10.4	0	0	–	–	823	737	-10.4
	Dumb500M	528	519	-1.7	0	0	–	–	528	519	-1.7
Gemma-4-12B	MATH500	8921	3839	-57.0	19.3	0.27	97 (29)	0 (40)	3869	3769	-2.6
	GSM8K	11752	2777	-76.4	34.7	0.02	100 (52)	0 (3)	2069	2772	+33.9
	Dumb500M	1211	770	-36.4	1.8	0.01	100 (2)	100 (1)	692	767	+10.9
Qwen3-1.7B	MATH500	5768	4922	-14.7	4.7	0.83	57 (7)	48 (124)	4582	4713	+2.9
	GSM8K	2394	2337	-2.4	0.67	0.01	100 (1)	100 (2)	2209	2333	+5.6
	Dumb500M	1546	1673	+8.2	0	0.44	–	34 (50)	1546	1547	+0.1
Qwen3-4B	MATH500	4533	4412	-2.7	1.3	0.40	100 (2)	18 (60)	4189	4309	+2.9
	GSM8K	2830	2327	-17.8	2.0	0	100 (3)	–	2276	2327	+2.3
	Dumb500M	1539	1486	-3.4	0	0	–	–	1539	1486	-3.4
Qwen3-8B	MATH500	4976	4671	-6.1	2.0	0.57	67 (3)	50 (86)	4465	4525	+1.3
	GSM8K	2705	2373	-12.3	1.3	0.10	100 (2)	100 (15)	2336	2345	+0.4
	Dumb500M	1730	1745	+0.9	0	0.07	–	100 (8)	1730	1725	-0.3
Qwen3-14B	MATH500	4021	4111	+2.2	0.67	0.35	100 (1)	58 (53)	3847	4019	+4.5
	GSM8K	1872	1971	+5.3	0	0.03	–	100 (4)	1872	1964	+4.9
	Dumb500M	1142	1155	+1.1	0	0	–	–	1142	1155	+1.1
Pooled					**3.6**	**0.15**	**95 (103)**	**43 (447)**			

Table 5: Greedy decoding ($T{=}0$, one generation per problem) versus stochastic sampling ($T{>}0$, $\mathcal{N}{=}100$ generations per problem). **Mean length** gives both means in tokens and $\Delta\%{=}(\text{sampled} - \text{greedy})/\text{greedy}$, so a negative value means the greedy trace is longer. **Truncated** is the percentage of traces reaching the $30{,}000$-token limit under each policy. **Loops** is the percentage of those truncated traces containing a repetition loop, with the number of truncated traces in parentheses; "–" marks cells with none, and cells based on one or two traces carry no statistical weight. **Excl. truncated** recomputes both means with truncated traces removed, as $(\bar{\ell} - f \cdot 30{,}000)/(1 - f)$ with f the truncation rate of that policy, since every truncated trace contributes exactly the token limit. The raw greedy excess reaches 76%, but most of it comes from truncation: greedy is longer than sampled in 15 of 21 pairs before the correction and in 7 after, and Gemma-4-12B's GSM8K gap reverses. The largest remaining gaps are Gemma-4-E4B's (-7.6% and -10.4%), which has no truncated traces. Per-problem run-to-run variability for the sampled runs (rCV) is in Table 1.

it. The same collapse appears in the smallest Gemma model (Gemma-4-E2B, MATH500; loop start at word 736, period 25), so the failure mode is not specific to model size.

C Metric Definitions and Sensitivities

C.1 Design of the Empirical Difficulty Score

$d(p)$ combines a length component and an accuracy component, each chosen for a specific reason.

The length component, $L_{10}(p)$, is the mean length of the shortest decile of a problem's correct generations: we find the 10^{th}-percentile length threshold among correct runs, then average the actual generations at or below it. This is a more robust relative of the literal shortest correct run,

$$\ell^\star(p) = \min_{i\,:\,c(x_i)=1} \ell(x_i), \tag{11}$$

used in earlier length-based difficulty measures. A minimum over $\mathcal{N}$ samples is one lucky short generation away from understating a problem's true difficulty: it is the single most sample-size-sensitive statistic one could build here, since fewer draws systematically undersample the extreme short tail a minimum depends on.

The accuracy component is a smoothed win rate,

$$\mathrm{WR}_s(p) = \frac{k(p) + 1}{\mathcal{N} + 2}, \tag{12}$$

where $k(p)$ is the number of correct runs, used in place of the raw $\mathrm{WR}(p) = k(p)/\mathcal{N}$. Add-one smoothing keeps every win rate strictly between 0 and 1, so a problem a model solves only once or

twice still gets a large but finite, comparable difficulty score instead of an unstable one. A problem a model never solves in $\mathcal{N}$ tries has no correct run to compute $L_{10}(p)$ from, so $d(p)$ is undefined for it and it is excluded from difficulty-conditioned analyses (0.3% of model-problem pairs, §3.2).

$d(p) = L_{10}(p)/\mathrm{WR}_s(p)$ reads as *expected tokens spent before a first correct answer*: at near-ceiling accuracy, $\mathrm{WR}_s(p) \approx 1$ and $d(p) \approx L_{10}(p)$, length alone; as accuracy falls, $d(p)$ grows past $L_{10}(p)$ in direct proportion to how often a fresh attempt is needed. Concretely, two problems with the same $L_{10}(p) = 1000$ tokens but win rates of 100% and 50% score $d(p) \approx 1010$ and $d(p) = 2000$ respectively: halving accuracy at fixed length roughly doubles difficulty, the same "$+1$ means twice as costly" reading we use for $\log_2 d(p)$ throughout this paper.

We deliberately do not normalize $d(p)$ to $[0, 1]$. Left in raw tokens, a $d(p)$ value stays directly interpretable (2000 literally means "2000 expected tokens"), comparable across datasets and models without committing to a normalization scheme, and comparable to any other study that reports length in tokens, including outside this one. $d(p)$ is, however, heavy-tailed by construction: a handful of low-accuracy problems, where $\mathrm{WR}_s(p)$ shrinks towards its floor, can inflate it 30–70$\times$ over $L_{10}(p)$ alone (the maximum for $\mathcal{N}$=100, reached at $k(p)$=1). Every table, plot, and cross-model comparison in this paper involving $d(p)$ therefore displays it on a $\log_2$ scale, since a linear one squashes almost every problem into a sliver near d=0 against a long right tail.

C.2 Empirical Difficulty versus Official Difficulty Labels and Accuracy

§3 argues that $d(p)$, not a dataset's official difficulty label or accuracy alone, is the right conditioning variable for this paper's difficulty-aware analyses, since it is measured per model rather than assigned once by a human grader, and adds information a pure accuracy-based measure cannot. Table 6 substantiates this two ways: against the one dataset with an official difficulty label available, MATH500's *level* 1–5 annotation, and against raw accuracy $\mathrm{WR}(p)$, pooled across all three datasets. $d(p)$ correlates positively with *level* for every one of the seven models, by Pearson (0.23–0.41) and, the more informative statistic for a rank-based label, by Spearman (0.52–0.57), confirming it tracks real difficulty rather than an arbitrary per-model artefact. It also correlates negatively with accuracy, as it must by construction, but far from perfectly (Pearson -0.58 to -0.74; Spearman, the more honest summary given $d(p)$'s heavy right tail, only -0.25 to -0.50): a large share of $d(p)$'s variation is not explained by accuracy alone, direct evidence that length is contributing real, independent information rather than $d(p)$ simply repackaging $1-\mathrm{WR}(p)$ under a different name.

	vs. MATH500 Level		vs. Accuracy $\mathrm{WR}(p)$	
Model	**Pearson**	**Spearman**	**Pearson**	**Spearman**
Gemma-4-E2B	0.248	0.542	-0.577	-0.424
Gemma-4-E4B	0.241	0.523	-0.595	-0.382
Gemma-4-12B	0.407	0.566	-0.700	-0.253
Qwen3-1.7B	0.230	0.544	-0.713	-0.501
Qwen3-4B	0.271	0.547	-0.720	-0.325
Qwen3-8B	0.387	0.531	-0.735	-0.355
Qwen3-14B	0.286	0.537	-0.612	-0.269

Table 6: Correlation between $d(p)$ and (left) MATH500's official level 1–5 label, per model (150 problems each), and (right) accuracy $\mathrm{WR}(p)$, per model, pooled across all three datasets (410–413 problems each).

The level correlation is positive but far from perfect ($r^2 \approx 0.05$–0.17), and the disagreements are informative in their own right, not just noise. For every one of the seven models, we rank problems both by their own empirical percentile of $d(p)$ and by *level*'s percentile, and take the problems with the largest absolute gap between the two ranks. A fixed external label, assigned once, has no mechanism to catch this; $d(p)$, measured directly from a model's own generations on every dataset, does not need one.

Why accuracy alone is not enough. Figure 8 makes the accuracy-correlation numbers above concrete. Each point is one of our 413 problems, difficulty and accuracy each averaged across all seven models. 228 of the 413 problems already sit at 100% accuracy for every model, the flat band along the top of the plot; a difficulty measure built only from $1 - \mathrm{WR}(p)$ would assign every one of them the exact same score of 0, unable to say anything further about which of these "already solved"

problems is nonetheless more costly to solve. $d(p)$ does not have this ceiling: restricted to just this 100%-accuracy subset, it still ranges from 278 to 3538 tokens, a $12.7\times$ spread (more than 3.5 doublings on the $\log_2$ scale we use throughout this paper), showing that even among problems every model gets right every time, some demand a systematically longer reasoning chain than others. This is exactly the information a pure accuracy-based difficulty measure discards.

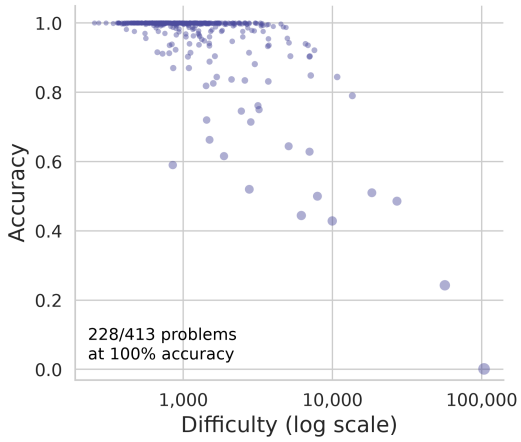


Figure 8: Difficulty $d(p)$ (log scale) vs. accuracy, one point per problem (n=413), both averaged across all seven models. The flat band at accuracy $= 1$ (225 problems) still spans a $12.7\times$ range in $d(p)$, evidence that length carries real signal accuracy alone cannot.

C.3 CV(P) VERSUS RCV(P): A PER-PROBLEM VIEW

§5.1 reports that the aggregate stability index $\widehat{\mathrm{CV}}$ exceeds the median rCV by roughly 1.1 to 1.6 times across the seven models (Table 1). Comparing two aggregate statistics like this cannot rule out that the gap is a quirk of the medians themselves, so Figure 9 shows the same comparison at the per-problem level instead: for every model, a scatter of $\mathrm{CV}(p)$ against $\mathrm{rCV}(p)$ with the $y = x$ reference line, annotated with the fraction of problems above it and the median per-problem ratio $\mathrm{CV}(p)/\mathrm{rCV}(p)$.

Two things are visible that the two aggregate numbers alone do not show. First, most problems sit above the diagonal: $\mathrm{CV}(p) > \mathrm{rCV}(p)$ for 64–84% of problems in every model, against the 50% expected if lengths were normally distributed, since rCV is scaled to equal CV in that case. Second, the ratio $\mathrm{CV}(p)/\mathrm{rCV}(p)$ is right-skewed: its median is only 1.05–1.40, but it exceeds 2 for 3–25% of problems and reaches 5–$16\times$ in every model. A broad, symmetric spread would place problems evenly on both sides of the diagonal with no such tail; instead, the gap is concentrated in a minority of problems whose standard deviation is inflated by a few runaway generations. This is the per-problem evidence for treating $\mathrm{CV}(p)$ and $\mathrm{rCV}(p)$ as reporting different things, and for attributing the gap to a minority of long runs rather than to broadly elevated variance.

Model	$\kappa = 1.5$	$\kappa = 2.0$	$\kappa = 3.0$	$\kappa = 5.0$
Gemma4-12B	0.105	0.039	0.012	0.003
Gemma4-E2B	0.024	0.006	0.001	0.000
Gemma4-E4B	0.028	0.004	0.000	0.000
Qwen3-1.7B	0.060	0.022	0.007	0.002
Qwen3-4B	0.034	0.006	0.001	0.000
Qwen3-8B	0.038	0.007	0.002	0.001
Qwen3-14B	0.036	0.006	0.001	0.001

Table 7: Mean runaway rate $\mathrm{RR}(p, \kappa)$, Overall rows, at four threshold values. Per-dataset values and the fraction of problems with at least one qualifying run are in the released CSVs.

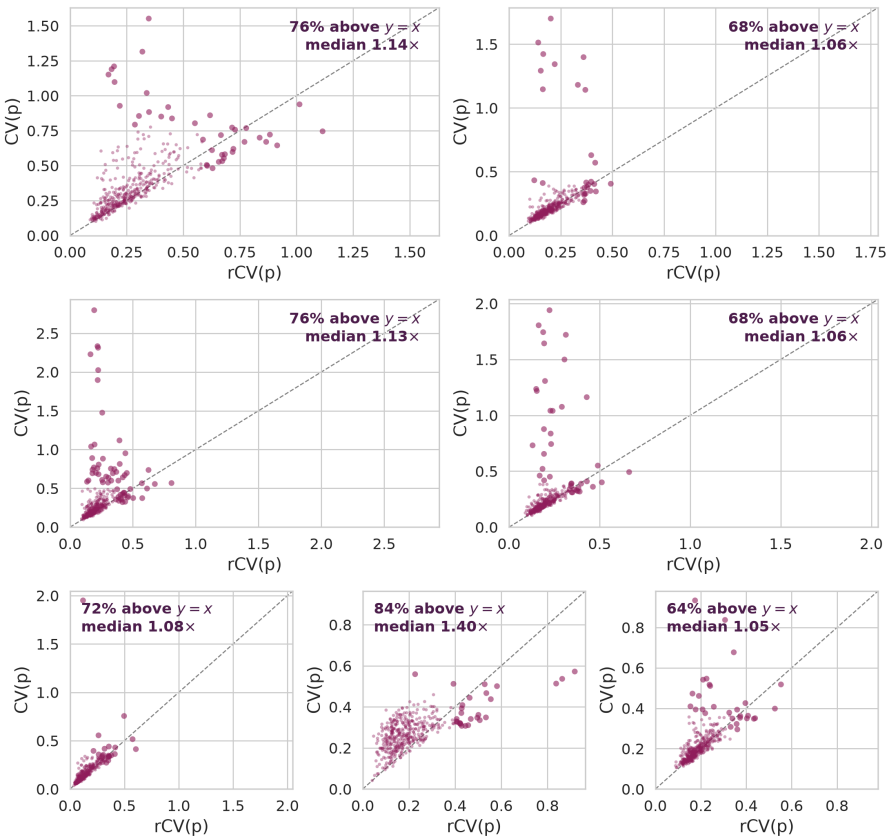


Figure 9: $\mathrm{CV}(p)$ vs. $\mathrm{rCV}(p)$ per problem (Overall rows), dashed line at $y = x$; larger markers are Tukey-fence outliers on either axis. Panels, left to right then top to bottom: Gemma4-12B, Qwen3-14B, Qwen3-1.7B, Qwen3-8B, Gemma4-E2B, Gemma4-E4B, Qwen3-4B.

C.4 Runaway-Rate Sensitivity to the Threshold κ

We chose $\kappa = 2$ for the runaway rate $\mathrm{RR}(p, \kappa)$ used in the main text as a middle ground rather than a value fit to any target rate. Table 7 shows: at $\kappa = 1.5$, 43–89% of problems have at least one qualifying run, at $\kappa = 3$ the mean rate is at most 1.2%, and $\kappa = 2$ sits between the two while still registering a non-trivial event rate.

C.5 Overthinking Ratio versus CV(p)

§5.3 reports that $\mathrm{OR}(p)$ correlates with $\mathrm{CV}(p)$ (Spearman 0.62–0.90) but not with MATH500's official Level (-0.14 to $+0.22$, sign-flipping across models, not reported as a table since there is no consistent relationship to summarize). Figure 10 shows the OR-vs-CV relationship per model (overall), with a linear trend and bootstrap 95% band for the three models whose ranges are compact enough for a line to be a fair summary. The other four have a long outlier tail in one or both statistics, where the Spearman coefficient alone is the honest summary of a relationship that is monotonic but visibly saturating rather than linear.

D Second-Guessing in Long Runs

§5.4 attributes the concentration of instability on easy-to-medium problems to second-guessing, i.e. when a model reaches the answer early and then keeps re-checking it. This appendix shows what that looks like in our traces, first in aggregate and then through two representative runs.

Long runs re-check an answer they already have. In every correct run with a numeric answer, we locate the first point where the reasoning states that answer (e.g. "$= 79$" or "the answer is 79"),

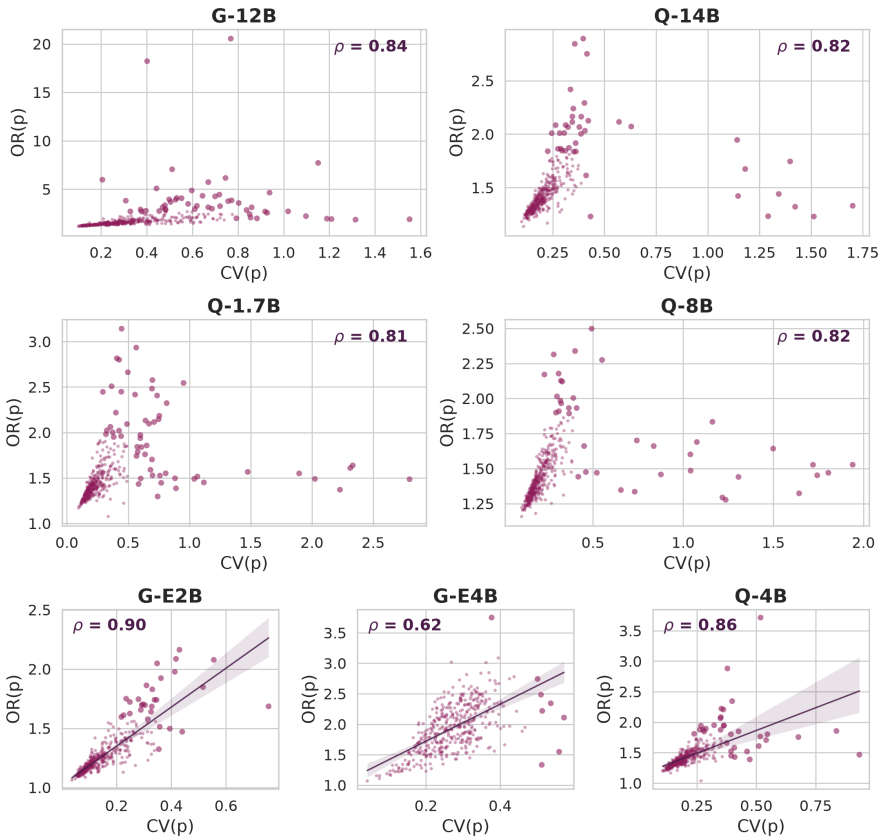


Figure 10: $\mathrm{OR}(p)$ vs. $\mathrm{CV}(p)$ per problem (Overall rows), one panel per model. Larger markers are Tukey-fence outliers on either axis. ρ is the Spearman correlation; the three models with the most compact ranges also get a linear OLS trend with a bootstrap 95% CI band.

and compare runs at least twice as long as their problem's median with runs of typical length (0.8–1.25× the median). Long runs reach their answer early: pooled over models, 81–90% of them state it within the first quarter of the trace, against 19–37% of typical-length runs, and the median long run spends about 90% of its trace after the answer, against 56–67%. The comparison also holds within problems. When a problem has both a long and a typical correct run, the long one spends a larger share of its trace after the answer in 92–96% of problems. The extra length of a long correct run is therefore not spent searching for the answer, but re-checking one the model has already found.

Example 1: doubting a correct premise. Qwen3-14B, DUMB500 Math, "If $10-3=7$, what is $7-3$?" This run uses 3,238 tokens, against a median of 1,210 over the problem's 100 runs (which is still incredibly high for such a simple question). Both reach the correct answer, 4. The excerpt starts after the first 72 words of the trace, where the model has just restated the problem:

```
So, subtracting 3 from 7.  If I have 7 objects and I take
away 3, how many are left?  Let me count.  Starting with 7,
taking away 1 leaves 6, taking away another leaves 5, and
then another one leaves 4.  So, 7 - 3 should be 4.
Wait, but maybe I should check if there's some trick
here.  The problem starts with 10 - 3 = 7, which is
straightforward.  Maybe they want me to use that equation
to find 7 - 3?  Like, maybe there's a pattern or something?
Let me see.
If 10 - 3 = 7, then perhaps they want me to think of 7 as
the result of 10 - 3.  So then, if I take that result, which
is 7, and subtract 3 from it, that would be (10 - 3) - 3.
Let me compute that.  10 - 3 is 7, then 7 - 3 is 4.  So that
still gives me 4.
```

> Alternatively, maybe they want me to do something else? Like, if 10 - 3 is 7, then maybe 7 - 3 is 10? But that doesn't make sense because 7 - 3 is definitely 4. Unless there's some modular arithmetic or something going on here?
> *[... 1,358 words omitted ...]*
> Wait, but if we take the problem at face value, even though 10 - 3 is actually 7 in decimal? Wait, no, 10 - 3 is 7? Wait, 10 - 3 is 7? Wait, no! 10 - 3 is 7? Wait, that's not correct. Wait, 10 minus 3 is 7? Wait, 10 - 3 is 7? Wait, no! 10 - 3 is 7? Wait, 10 minus 3 is 7? Wait, that's not correct. Wait, 10 - 3 is 7? Wait, 10 minus 3 is 7? Wait, no! Wait, 10 - 3 is 7? Wait, that's only true if you're subtracting 3 from 10 and getting 7? Wait, but 10 minus 3 is 7? Wait, that's not correct. Wait, 10 minus 3 is 7? Wait, 10 - 3 is 7? Wait, no! Wait, 10 - 3 is 7? Wait, I think I'm going in circles here. Let me actually do the math.
> 10 minus 3: 10 - 3. 10 minus 1 is 9, minus 1 is 8, minus 1 is 7. So that's three subtractions of 1, totaling 3. So yes, 10 - 3 = 7. Wait, that's correct! Oh my goodness, I was confused for a moment. So 10 - 3 is indeed 7. Then, 7 - 3 is 4. So the answer is 4.
> Wait, why did I get confused earlier? Maybe I thought there was a trick, but no, it's straightforward. So the problem is just testing basic subtraction. Given that 10 - 3 = 7, which is correct, then 7 - 3 is 4. So the answer is 4.
> I think I overcomplicated it by thinking there was a trick or different base or something, but no, it's just simple subtraction. So the answer is 4.

Example 2: a premise that conflicts with world knowledge. Gemma-4-12B, GSM8K, "India has 4 more than half the number of states in the USA. What's the total number of states in both countries together?" This run uses 10,998 tokens, 5× the problem's median of 2,185. Both reach the correct answer, 79. The answer is fixed within the first lines of the trace, and the rest is spent reconciling it with the fact that India has 28 states. The model re-counts India's states many times and eventually starts listing the 50 US states as well:

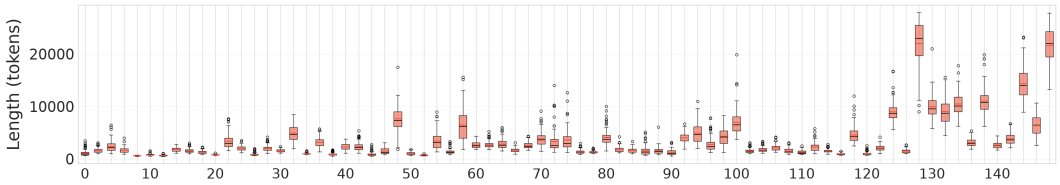
> * Number of states in the USA = 50. (This is a well-known fact).
> * Number of states in India = (1/2 * Number of states in USA) + 4.
> * Number of states in India = (1/2 * 50) + 4.
> * Number of states in India = 25 + 4.
> * Number of states in India = 29.
> * Wait, let me double-check the number of states in India.
> * Actually, India has 28 states and 8 union territories. The question refers to "states".
> * Let's re-read: "India has 4 more than half the number of states in the USA."
> * If USA has 50 states, half of 50 is 25.
> * 25 + 4 = 29.
> * Wait, is the number of states in India 29? No, it's 28. Let me check the wording again.
> * "India has 4 more than half the number of states in the USA."
> * Wait, the question might be a math problem *based* on the information provided, not necessarily a fact-check of real-world data. Let's re-examine.
> * If I use the *real-world* number of states in India (28), then:
> 28 = (1/2 * USA_states) + 4
> 24 = 1/2 * USA_states

```
USA_states = 48.
This is incorrect (the USA has 50 states).
* If I use the *real-world* number of states in the USA
(50), then:
India_states = (1/2 * 50) + 4 = 25 + 4 = 29.
This contradicts the real-world fact that India has 28
states.
```
[... 4,049 words omitted ...]
```
Wait, I only count 28.  Where is the 29th?
Let me search "How many states in India?".
"India has 28 states."
Okay, so the number is 28.
Then my original calculation stands:
If $U = 50$, $I = 29$.
Total = 79.
If the question had $I = 28$, $U = 48$.
Total = 76.
Wait, let me think.  Is there any other way?
Could the USA have 50 states and India have 28, and the
question is just slightly incorrect?
If I answer 78, I'm assuming the question is wrong about "4
more than half".
If I answer 79, I'm assuming the question is wrong about the
number of states in India.
In most competitive exams, the provided rule is what you
follow.
Rule:  $I = U/2 + 4$
Given:  $U = 50$
Result:  $I = 29$
Total:  $I + U = 79$.
Let's re-calculate one more time.
$U = 50$
$I = (50/2) + 4 = 25 + 4 = 29$
$I + U = 50 + 29 = 79$
Wait, let me think if I missed any state in the USA.
1.  Alabama
2.  Alaska
3.  Arizona
```
[...]

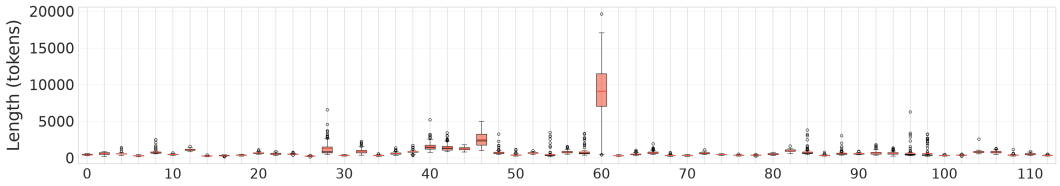
In both cases the extra length does not move the model towards a different answer. Instead, it re-derives, doubts and re-confirms a solution it already had, which is the pattern the second-guessing account predicts.

E Per-Problem Length Distributions for All Models and Datasets

Figure 1 (§1) shows the per-problem distribution of trace length for Gemma-4-12B on GSM8K. Figures 11–17 give the same view for every other model–dataset pair: Figure 11 completes Gemma-4-12B with its two remaining datasets, and Figures 12–17 cover the remaining six models on all three datasets. In every panel, each box shows the interquartile range and median for one problem across $\mathcal{N}{=}100$ runs, whiskers extend to $1.5\times$ the interquartile range, and circles mark individual runs beyond that range. Note the differing y-axis scales across panels.

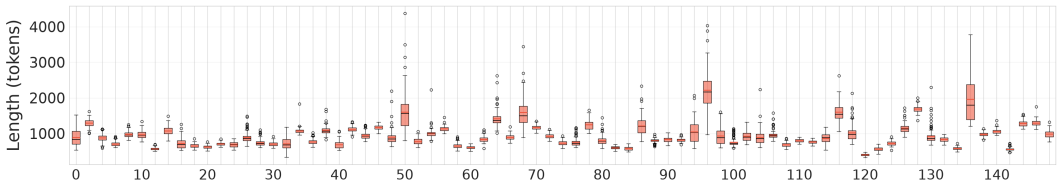


(a) MATH500

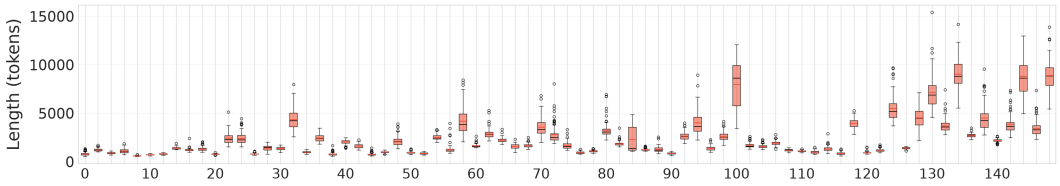


(b) DUMB500 Math

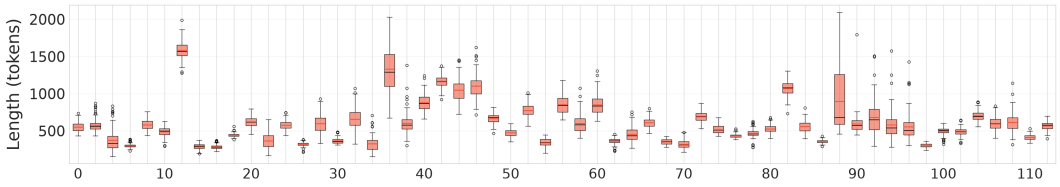
Figure 11: Per-problem distribution of trace length (in tokens) across $\mathcal{N}$=100 runs, for Gemma-4-12B on the two datasets not shown in Figure 1.



(a) GSM8K

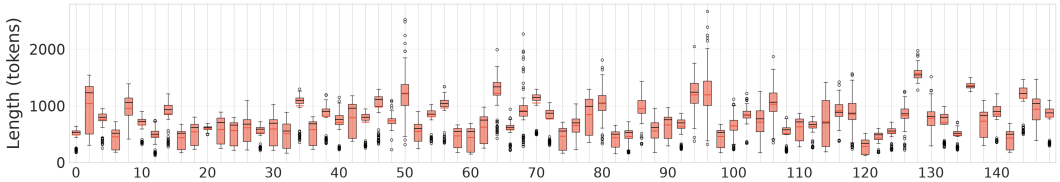


(b) MATH500

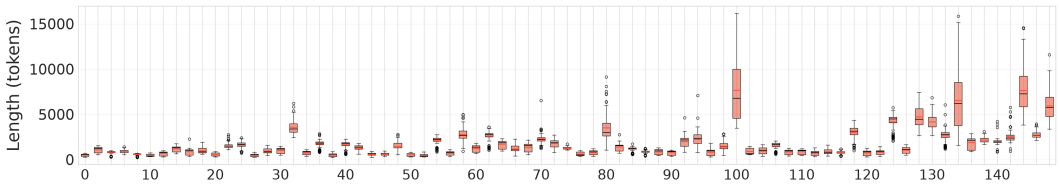


(c) DUMB500 Math

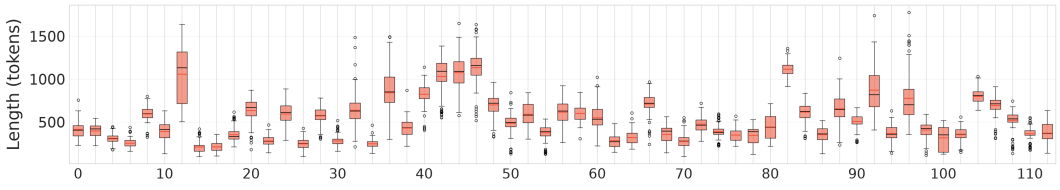
Figure 12: Per-problem distribution of trace length (in tokens) across $\mathcal{N}$=100 runs, for Gemma-4-E2B.



(a) GSM8K

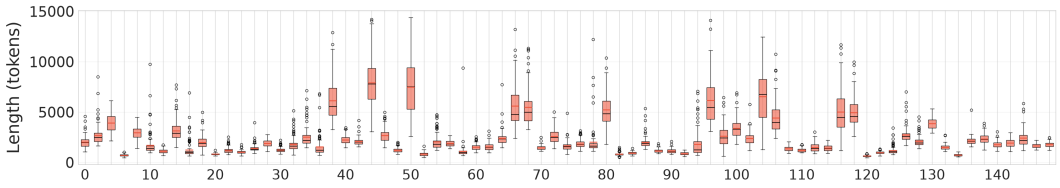


(b) MATH500

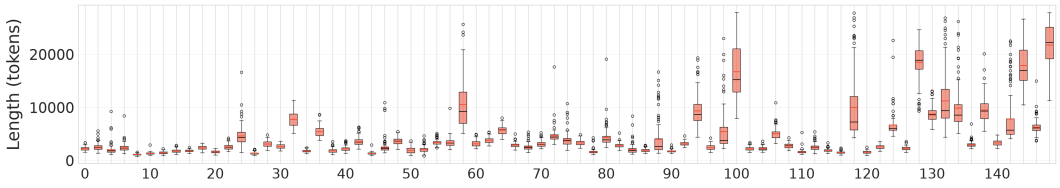


(c) DUMB500 Math

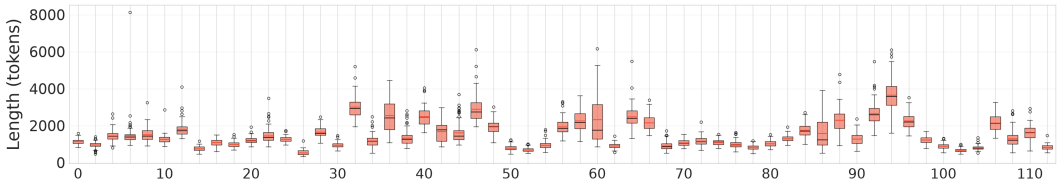
Figure 13: Per-problem distribution of trace length (in tokens) across $\mathcal{N}$=100 runs, for Gemma-4-E4B.



(a) GSM8K

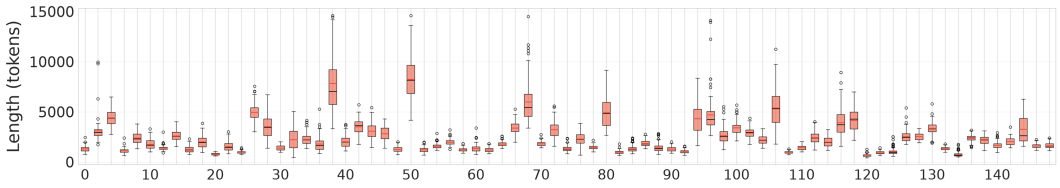


(b) MATH500

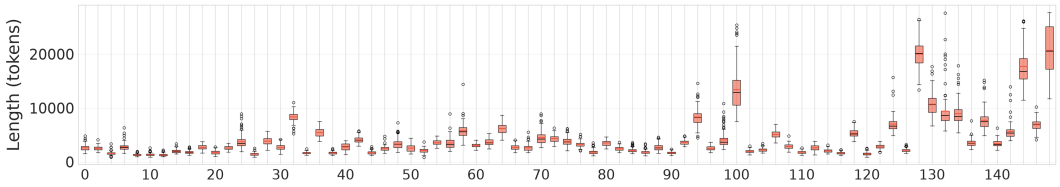


(c) DUMB500 Math

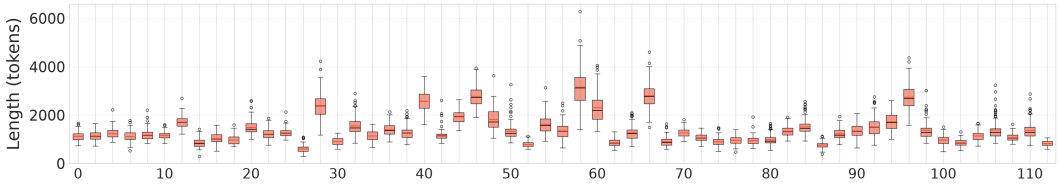
Figure 14: Per-problem distribution of trace length (in tokens) across $\mathcal{N}$=100 runs, for Qwen3-1.7B.



(a) GSM8K

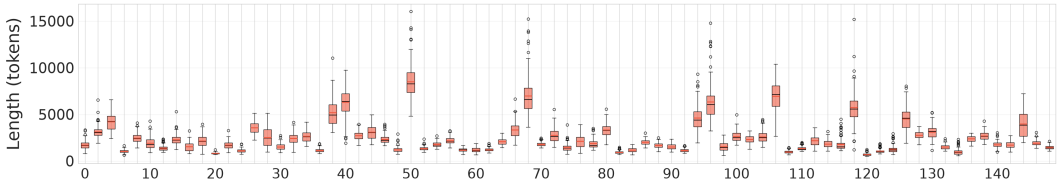


(b) MATH500

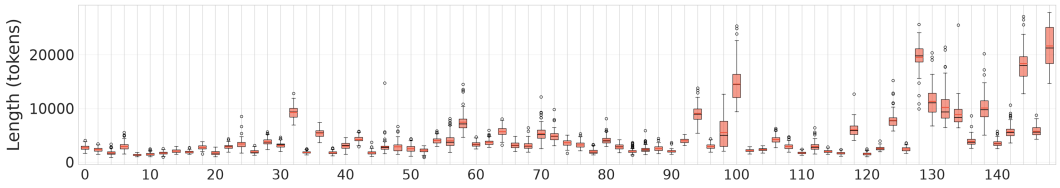


(c) DUMB500 Math

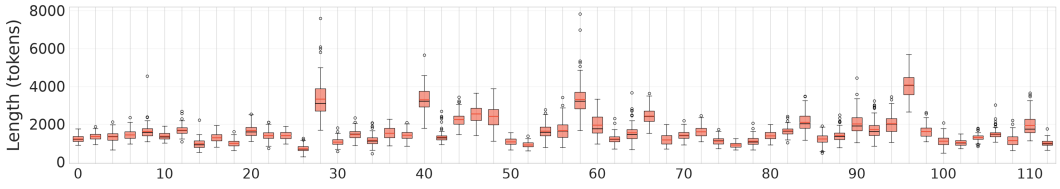
Figure 15: Per-problem distribution of trace length (in tokens) across $\mathcal{N}$=100 runs, for Qwen3-4B.



(a) GSM8K

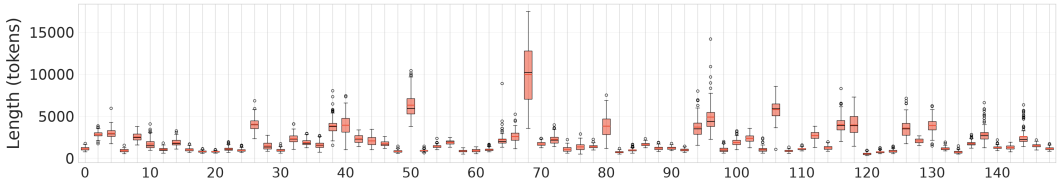


(b) MATH500

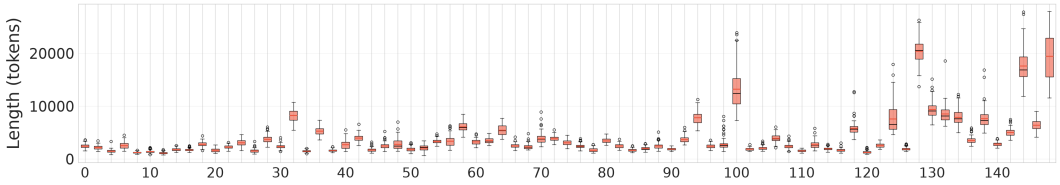


(c) DUMB500 Math

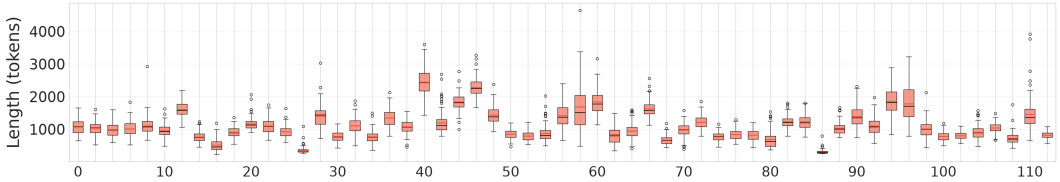
Figure 16: Per-problem distribution of trace length (in tokens) across $\mathcal{N}$=100 runs, for Qwen3-8B.



(a) GSM8K



(b) MATH500



(c) DUMB500 Math

Figure 17: Per-problem distribution of trace length (in tokens) across $\mathcal{N}$=100 runs, for Qwen3-14B.

F Statistical Rigor

F.1 Estimator Stability: A Subsampling Sweep

Every headline statistic in this paper is computed from $\mathcal{N}$=100 generations per problem, a budget chosen to characterise instability well, not to minimise compute. A study that wants to apply this framework needs to know the smallest budget that still gives a trustworthy answer, and whether $\mathcal{N}$=100 itself is even enough. Different quantities in this paper answer different questions, though, and there is no reason to expect them to converge at the same rate: a difficulty score describes a single problem's typical behaviour, while an instability statistic describes how much that behaviour varies run to run, a strictly harder quantity to pin down from few samples. This appendix checks all eight statistics reported in Table 1 for which a subsample-based estimate is meaningful, mean length, d (difficulty), ρ, $\widehat{\mathrm{CV}}$, rCV, $\overline{OR}$, and the two length–correctness correlations (P-corr, W-corr), and is the basis for the $\mathcal{N} = 50$ recommendation in §6 and the abstract.

We use one procedure for all eight: for $k \in \{5, 10, 25, 50, 100\}$, we repeatedly draw, without replacement, a subsample of size k from each problem's 100 generations, recompute every statistic on that subsample (Overall rows, pooled across a model's three datasets), and repeat 1000 times per k; k=100 uses the full sample, with no resampling variance by construction. Figures 18–21 show the median and interquartile range at each k, for every model, for the four statistics we feature in the main text: d, $\widehat{\mathrm{CV}}$, ρ, and $\overline{OR}$. Mean length, rCV, and the two correlations converge with the same procedure; we report their bias and spread in Table 8 and release the corresponding plots with the analysis code rather than reproduce all eight figures here.

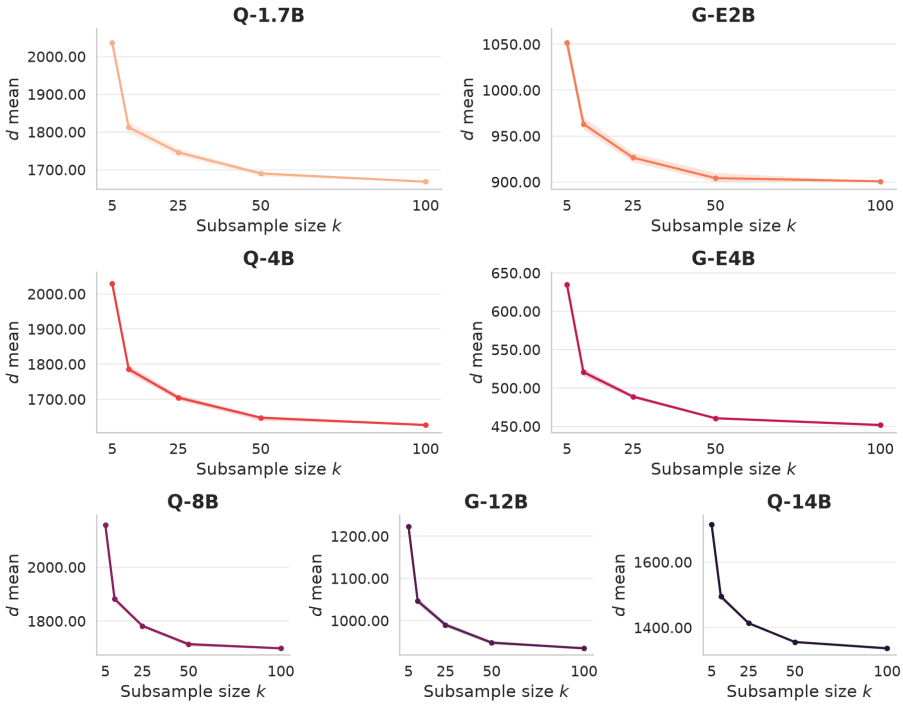


Figure 18: Median d and interquartile range (shaded band) over 1000 subsamples of size $k \in \{5, 10, 25, 50, 100\}$, one panel per model (Overall rows, own y-scale per panel), ordered and coloured by parameter size as in Figure 3. Each line collapses to a single point at $k = 100$ (the full sample, no resampling variance by construction).

Difficulty converges fast, and that is the point. d describes how hard a single problem is for a model under repeated sampling, a property of that problem–model pair alone. It stabilises early: bias is already down to 7–15% (worst case across models) at $k = 10$, 3–8% at k=25, and under 2% at k=50, with spread never exceeding 2–3% at any k we test. This is the practical reading an overthinking-mitigation study needs. Because d is measured from a model's own behaviour, evaluating a mitigation method's effect on it, e.g. checking whether the method reduces d on average

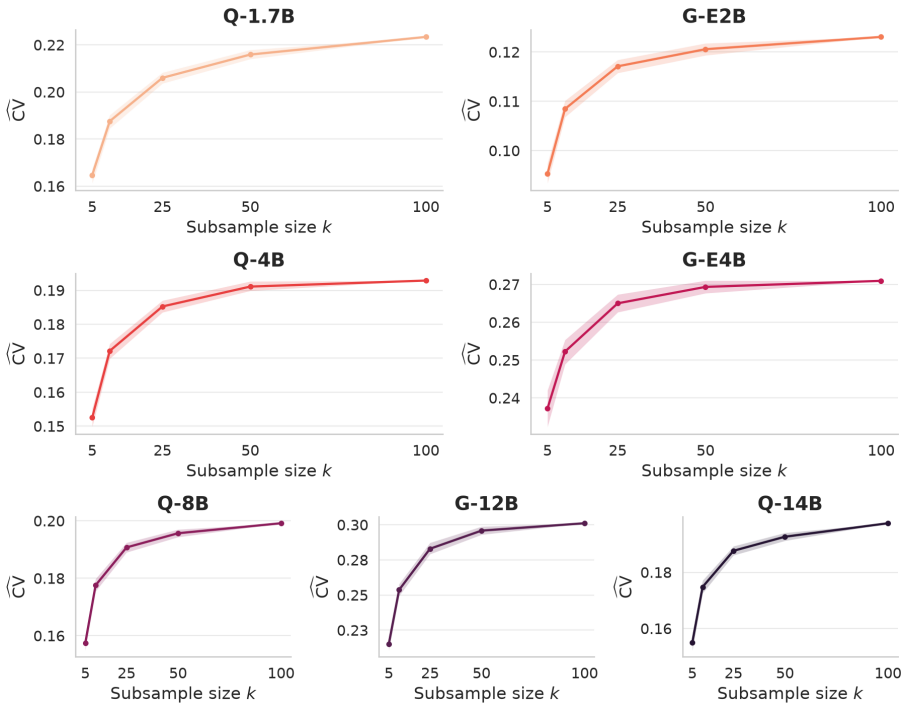


Figure 19: The same subsampling sweep applied to $\widehat{\mathrm{CV}}$.

while holding accuracy at least constant, does not require the full $\mathcal{N}$=100 budget; k = 10–25 already gives a reliable read on inherent difficulty, since a mitigation method that works should lower the tokens a model needs without needing many resampled runs per problem to see that shift.

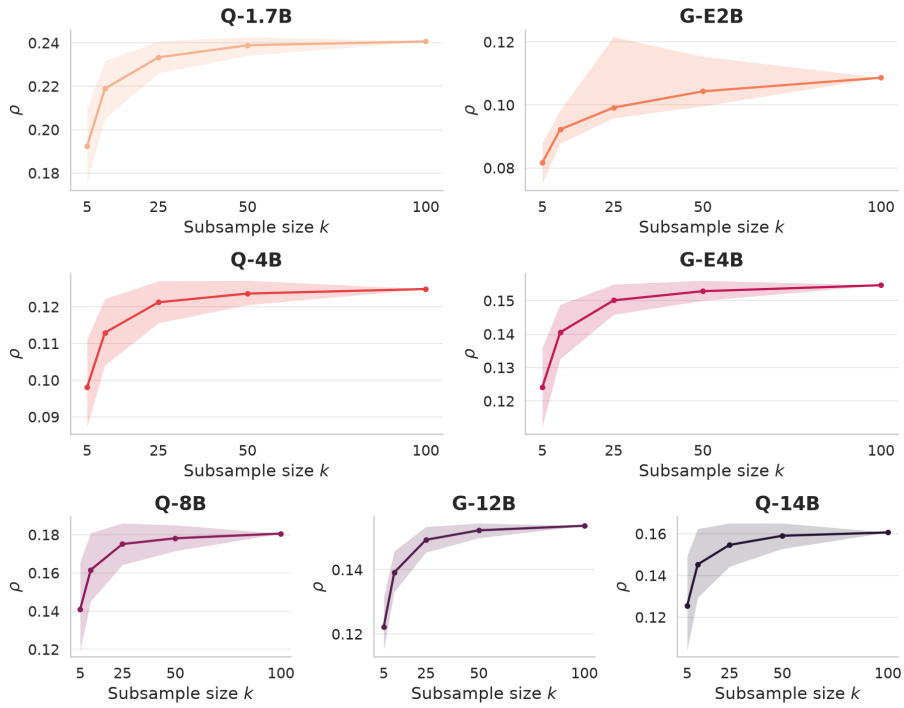


Figure 20: The same subsampling sweep applied to ρ.

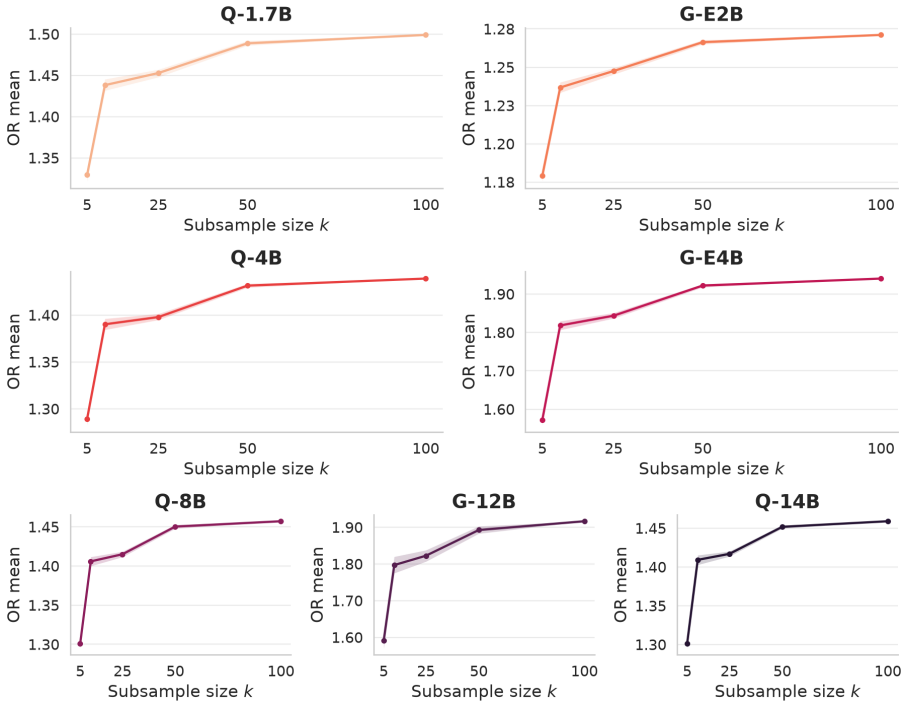


Figure 21: The same subsampling sweep applied to $\overline{OR}$. Bands are visibly tighter than $\widehat{\mathrm{CV}}$'s or ρ's (Figures 19, 20) at every k, since $\overline{OR}$'s denominator, $L_{10}(p)$, is itself a shortest-decile mean rather than a raw minimum (§3, Appendix C.1) and so is far less exposed to undersampling the short tail.

Difficulty alone is not enough: instability needs larger k, but not uniformly. A difficulty score by construction says nothing about how much a problem's behaviour varies run to run, which is exactly what ρ, $\widehat{\mathrm{CV}}$, and rCV measure. All three are spread statistics and converge more slowly than d in the sense that matters for a reduced budget. Their bias at k=10 is of similar size to d's ($\widehat{\mathrm{CV}}$: -16% to -7%; ρ: -15% to -9%; rCV: -12% to -9%, worst case, against d's $+7\%$ to $+15\%$), but d's bias falls under 2% by k= 50 while $\widehat{\mathrm{CV}}$'s is still up to 3.4% and ρ's up to 4%, and ρ's spread stays wide as k grows (21–24% at k = 10–25, still up to 14.5% at k = 50, against at most 3.2% for d). This is the expected behaviour of a standard-deviation-based statistic on few samples, and it means a study that cares about the *magnitude* of instability, not just whether a problem is hard, should budget for $k\geq$50: at that point $\widehat{\mathrm{CV}}$'s bias is under 3.4% and rCV's under 2.1% for every model, small enough that a mitigation effect of the size reported anywhere in this paper would still be detectable.

$\overline{OR}$ is the one instability-adjacent statistic that does *not* need as large a k, and this is a direct consequence of the $L_{10}(p)$–not–$\ell^\star(p)$ design choice in Appendix C.1. Because $\mathrm{OR}(p)$ divides by a shortest-decile mean rather than a raw minimum, it is far less sensitive to undersampling the short tail than $\widehat{\mathrm{CV}}$ or ρ: its bias is -19% to -7% at k=5 and -6% to -3% at k=10, already smaller than $\widehat{\mathrm{CV}}$'s or ρ's at k=10, and it closes to -1.2% to -0.4% by k=50, with spread under 2.3% throughout. A study that only needs the average overthinking ratio, and can tolerate being "more lenient" about precisely capturing rare run-away cases, can therefore treat k=25–50 as sufficient for $\overline{OR}$ specifically, without paying the larger budget a raw-minimum-based statistic would have demanded.

The two length–correctness correlations, P-corr and W-corr, are the noisiest quantities we test, and behave differently from the other six. Their bias stays small and unsystematic at every k (under 1% for P-corr, under 12% but non-monotonic in sign for W-corr), but their spread is very large at small k: up to 24% for P-corr and up to 130% for W-corr at k=5, still 5–26% at k=50. This is expected for a correlation coefficient estimated near zero from a small number of within-problem pairs, and it means these two should not be trusted below k=50, and are best read as full-$\mathcal{N}$=100 diagnostics rather than quantities a reduced-budget study can rely on.

Statistic	$k = 5$	$k = 10$	$k = 25$	$k = 50$
Mean length	0.0% / 1.8%	0.0% / 1.2%	0.0% / 0.7%	0.0% / 0.4%
d (difficulty)	17–41% / 3.2%	7–15% / 2.1%	3–8% / 1.2%	0.4–2.0% / 1.1%
ρ	20–25% / 28%	9–15% / 21%	3–9% / 24%	1–4% / 15%
$\widehat{\mathrm{CV}}$	13–29% / 3.5%	7–16% / 2.8%	2–8% / 2.7%	1–3.4% / 2.0%
rCV	22–26% / 5.4%	9–12% / 4.2%	2–4% / 3.2%	0–2.1% / 2.3%
$\overline{OR}$	7–19% / 2.3%	3–6% / 2.3%	2–5% / 1.6%	0–1.2% / 1.0%
P-corr	0–1% / 24%	0–1% / 16%	0–1% / 9.3%	0–0.2% / 5.1%
W-corr	0–12% / 130%	0–2% / 84%	0–2% / 49%	0–0.6% / 26%

Table 8: Bias (median subsample estimate relative to the $\mathcal{N}$=100 value, magnitude, worst case min–max across the seven models) / spread (interquartile range as a percentage of the $\mathcal{N}$=100 value, worst case across the seven models) at each k. Signs: d is biased upward at small k (a shortest-decile mean is a minimum-adjacent statistic); ρ, $\widehat{\mathrm{CV}}$, rCV, and OR mean are all biased downward, consistent with the well-known negative bias of variance- and minimum-based estimators on small samples.

Comparisons need fewer samples than absolute values. The bias in Table 8 matters when a statistic is read as an absolute value. A mitigation study instead compares two evaluations, and a bias that pushes both in the same direction largely cancels in their difference. To test this without a mitigation method at hand, we treat every pair of our seven models on the same dataset (21 model pairs $\times$ 3 datasets plus the pooled overall rows, 84 pairs) as a stand-in for a baseline/mitigated pair, whose true difference is known from the full $\mathcal{N}$=100 sample. For each k we subsample the two arms independently, as two separate evaluation runs would be, and record how often the subsampled difference has the same sign as the true one. Table 9 splits pairs by the size of the true relative difference. For differences of at least 10%, k=25 recovers the direction in 99.5–100% of subsamples f$\overline{OR}$ length, d, $\overline{OR}$, $\widehat{\mathrm{CV}}$ and rCV, and in 95% for ρ. For smaller differences no budget below $\mathcal{N}$=100 is reliable for the instability statistics: at k=50, $\widehat{\mathrm{CV}}$ still gets the sign of a <10% difference wrong in 12% of subsamples. RR and the two correlations were not included in this test.

	True difference <10%			**True difference** ≥10%		
Statistic	k=10	k=25	k=50	k=10	k=25	k=50
Mean length	94.3	97.2	98.8	100	100	100
d (difficulty)	76.0	88.1	92.8	100	100	100
$\overline{OR}$	85.8	93.2	96.0	99.5	100	100
$\widehat{\mathrm{CV}}$	77.4	83.6	87.6	98.7	99.9	100
rCV	74.2	80.7	85.9	98.3	99.5	100
ρ	62.7	70.9	78.5	88.0	95.1	98.2

Table 9: Sign agreement (%) between the difference of two subsampled evaluations and the true $\mathcal{N}$=100 difference, over 1000 subsamples per k and averaged over model pairs (each pair of our seven models on one dataset, or on the Overall rows, stands in for a baseline/mitigated pair). Pairs are split by the true relative difference $|A - B|/\mathrm{mean}(A, B)$; each cell pools 13–32 pairs (<10%) or 52–71 pairs (≥10%) depending on the statistic.

How to choose k in practice. The right budget depends on which axis a study needs. A study that only wants difficulty scores, e.g. to stratify problems or check that a mitigation method's effect concentrates where the paper predicts (§5.4), can use k=10–25: d is already within 3–15% of its full-sample value there, at negligible spread. A study that wants to measure a mitigation method's effect on the *magnitude* of instability, via $\widehat{\mathrm{CV}}$, ρ, or rCV, should budget for k=50, half of what this paper uses; going lower risks conflating a real reduction in instability with residual small-sample bias, since all three still carry 7–16% bias at k=10. If only the *direction* of a change of at least 10% is needed, k=25 suffices (Table 9). $\overline{OR}$ is the exception among the instability statistics: because it inherits $L_{10}(p)$'s robustness rather than a raw minimum's fragility, k=25–50 is already sufficient, and a study willing to be more lenient about rare run-away cases specifically can go as low as k=25 without the bias one would expect from a minimum-based statistic. The two correlation diagnostics, P-corr and W-corr, are the odd ones out. Both are structurally checked, not stratified, in this paper (Appendix C.3, C.5), and both need k≥50, ideally the full $\mathcal{N}$=100, before their spread is small enough to trust a specific sign or magnitude.

Model	ρ	$\widehat{\mathrm{CV}}$	$\overline{OR}$
Gemma4-E2B	0.109 [0.084, 0.143]	0.123 [0.113, 0.134]	1.271 [1.254, 1.289]
Gemma4-E4B	0.155 [0.124, 0.190]	0.271 [0.260, 0.283]	1.940 [1.903, 1.978]
Gemma4-12B	0.154 [0.123, 0.200]	0.301 [0.282, 0.321]	1.916 [1.793, 2.074]
Qwen3-1.7B	0.241 [0.183, 0.320]	0.223 [0.215, 0.243]	1.499 [1.470, 1.525]
Qwen3-4B	0.125 [0.100, 0.164]	0.193 [0.187, 0.199]	1.439 [1.418, 1.460]
Qwen3-8B	0.181 [0.130, 0.265]	0.199 [0.188, 0.208]	1.457 [1.437, 1.477]
Qwen3-14B	0.161 [0.109, 0.236]	0.197 [0.190, 0.203]	1.459 [1.435, 1.482]

Table 10: 95% bootstrap confidence intervals for ρ, $\widehat{\mathrm{CV}}$, and $\overline{OR}$, Overall rows.

Model	P-corr	W-corr
Gemma4-E2B	-0.327 [-0.431, -0.211]	-0.151 [-0.244, -0.066]
Gemma4-E4B	-0.227 [-0.337, -0.106]	-0.072 [-0.143, 0.005]
Gemma4-12B	-0.272 [-0.411, -0.126]	-0.086 [-0.173, 0.009]
Qwen3-1.7B	-0.400 [-0.486, -0.310]	-0.320 [-0.418, -0.232]
Qwen3-4B	-0.271 [-0.404, -0.142]	-0.101 [-0.176, -0.035]
Qwen3-8B	-0.338 [-0.460, -0.217]	-0.346 [-0.458, -0.247]
Qwen3-14B	-0.229 [-0.372, -0.115]	-0.104 [-0.178, -0.042]

Table 11: 95% bootstrap confidence intervals for the pooled and within-problem length to correctness correlations, Overall rows.

F.2 Confidence Intervals for Instability Statistics

Every per-problem statistic ($\mathrm{CV}(p)$, $\mathrm{rCV}(p)$, $\mathrm{RR}(p,\kappa)$, $\mathrm{OR}(p)$) gets a bootstrap confidence interval by resampling its $\mathcal{N}$ generations. Every aggregate statistic gets a second, cluster bootstrap over which problems are included in the evaluation set, which also underlies the FDR-corrected significance tests in Appendix F.3.

Table 10 and Table 11 report the aggregate-level (cluster bootstrap) intervals for every *overall* point estimate shown in Table 1 in the main text. Per-dataset confidence intervals follow the same procedure and are available in the released analysis code, they are omitted here purely for length.

Table 12 gives the same treatment to the tercile breakdown in Figure 4.

Model	Easy	Medium	Hard
Gemma4-E2B	0.324 [0.269, 0.390]	0.482 [0.425, 0.545]	0.157 [0.122, 0.212]
Gemma4-E4B	0.547 [0.502, 0.599]	0.681 [0.637, 0.728]	0.205 [0.159, 0.260]
Gemma4-12B	0.483 [0.437, 0.689]	0.604 [0.531, 0.679]	0.195 [0.151, 0.265]
Qwen3-1.7B	0.873 [0.763, 0.919]	0.794 [0.715, 0.847]	0.315 [0.229, 0.432]
Qwen3-4B	0.520 [0.472, 0.575]	0.614 [0.538, 0.680]	0.179 [0.137, 0.264]
Qwen3-8B	0.727 [0.518, 0.837]	0.861 [0.790, 0.891]	0.190 [0.137, 0.289]
Qwen3-14B	0.497 [0.435, 0.564]	0.744 [0.596, 0.847]	0.219 [0.146, 0.347]

Table 12: 95% bootstrap confidence intervals for ρ by d-difficulty tercile, Overall rows.

F.3 Statistical Significance Tests

Table 13 reports all $\binom{7}{2} = 21$ pairwise comparisons of $\widehat{\mathrm{CV}}$ at the Overall level, each an independent two-sample bootstrap test, Benjamini-Hochberg corrected across all 21 comparisons. Table 14 reports the Easy versus Hard ρ comparison within each model, corrected separately across those 7 comparisons.

Model A	Model B	$\widehat{\mathrm{CV}}_A$	$\widehat{\mathrm{CV}}_B$	q	Sig.
Gemma4-12B	Gemma4-E2B	0.301	0.123	0.000	✓
Gemma4-12B	Gemma4-E4B	0.301	0.271	0.007	✓
Gemma4-12B	Qwen3-14B	0.301	0.197	0.000	✓
Gemma4-12B	Qwen3-1.7B	0.301	0.223	0.000	✓
Gemma4-12B	Qwen3-4B	0.301	0.193	0.000	✓
Gemma4-12B	Qwen3-8B	0.301	0.199	0.000	✓
Gemma4-E2B	Gemma4-E4B	0.123	0.271	0.000	✓
Gemma4-E2B	Qwen3-14B	0.123	0.197	0.000	✓
Gemma4-E2B	Qwen3-1.7B	0.123	0.223	0.000	✓
Gemma4-E2B	Qwen3-4B	0.123	0.193	0.000	✓
Gemma4-E2B	Qwen3-8B	0.123	0.199	0.000	✓
Gemma4-E4B	Qwen3-14B	0.271	0.197	0.000	✓
Gemma4-E4B	Qwen3-1.7B	0.271	0.223	0.000	✓
Gemma4-E4B	Qwen3-4B	0.271	0.193	0.000	✓
Gemma4-E4B	Qwen3-8B	0.271	0.199	0.000	✓
Qwen3-14B	Qwen3-1.7B	0.197	0.223	0.000	✓
Qwen3-14B	Qwen3-4B	0.197	0.193	0.466	✗
Qwen3-14B	Qwen3-8B	0.197	0.199	0.682	✗
Qwen3-1.7B	Qwen3-4B	0.223	0.193	0.000	✓
Qwen3-1.7B	Qwen3-8B	0.223	0.199	0.000	✓
Qwen3-4B	Qwen3-8B	0.193	0.199	0.276	✗

Table 13: All 21 pairwise comparisons of the aggregate stability index $\widehat{\mathrm{CV}}$ at the Overall level, independent two-sample bootstrap test, Benjamini-Hochberg corrected. 18 of 21 pairs are significant at $q < 0.05$.

Model	ρ_{Easy}	ρ_{Hard}	q	Sig.
Gemma4-12B	0.483	0.195	0.000	✓
Gemma4-E2B	0.324	0.157	0.000	✓
Gemma4-E4B	0.547	0.205	0.000	✓
Qwen3-14B	0.497	0.219	0.002	✓
Qwen3-1.7B	0.873	0.315	0.000	✓
Qwen3-4B	0.520	0.179	0.000	✓
Qwen3-8B	0.727	0.190	0.000	✓

Table 14: Easy versus Hard ρ, paired bootstrap test per model, Benjamini-Hochberg corrected across the 7 comparisons. All 7 are significant at $q < 0.05$.

G Limitations

G.1 Hardware Constraints and Model Scale Selection

All experiments were conducted on single-GPU workstations equipped with either an NVIDIA RTX A6000 (48GB VRAM) or an NVIDIA GeForce RTX 5090 (32GB VRAM). The scope of model scales (up to 14B parameters) was deliberately selected to balance dense statistical sampling with computational tractability: evaluating each problem across $\mathcal{N}$=100 independent generations, for seven models and three datasets, already represents a substantial compute budget, and scaling to ultra-large models ($\geq$30B) at the same sampling density would have been prohibitive on this hardware.

Nevertheless, the 1.7B–14B parameter regime selected here represents a well-established standard in contemporary reasoning literature. This range offers a testbed for observing scaling dynamics across multiple size steps within two model families while remaining computationally tractable for dense, sample-heavy empirical investigation. We cannot say from this data alone whether the non-monotonic relationship between model size and instability (§5.1) continues, saturates, or reverses at frontier scale.

G.2 Fixed Decoding Settings

Every result in this paper uses one fixed decoding-hyperparameter setting per model family (temperature, top-p, and the rest listed in Appendix A.2), following each family's official recommendation. We therefore cannot say whether the instability we measure is itself sensitive to decoding temperature, or whether it is a property of the model that persists across any reasonable sampling setting. A

sweep over decoding hyperparameters, repeating the same $\mathcal{N}=100$-generations-per-problem protocol at each setting, would be needed to answer this and was outside this paper's compute budget.

G.3 Domain Scope

All three evaluation datasets, DUMB500 Math (Pu et al., 2025), GSM8K (Cobbe et al., 2021), and MATH500 (Lightman et al., 2023), are mathematical reasoning tasks. We chose this domain because it gives clean, symbolically verifiable ground truth for the correctness signal our metrics depend on, but it also means we cannot say whether the instability patterns characterised in §5.1–§5.5 extend to other reasoning domains, such as code generation, agentic tool use, or open-ended natural-language tasks, where correctness is harder to judge and reasoning traces may follow different length dynamics. Our three benchmarks are also largely solved by the evaluated models (91.6–99.6% accuracy), which limits how strongly failure can vary with instability. Harder benchmarks such as AIME would test the length–correctness relationship with more headroom.