arXiv:2602.07150v3 [cs.LG] 25 Mar 2026

ON RANDOMNESS IN AGENTIC EVALS

Bjarni Haukur Bjarnason*, **André Silva***, **Martin Monperrus**
KTH Royal Institute of Technology
Stockholm, Sweden
{bhbj, andreans, monperrus}@kth.se

ABSTRACT

Agentic systems are evaluated on benchmarks where agents interact with environments to solve tasks. Most papers report a `pass@1` score computed from a single run per task, assuming this gives a reliable performance estimate. We test this assumption by collecting 60,000 agentic trajectories on `SWE-Bench-Verified`, spanning three models and two scaffolds. We find substantial variance: single-run `pass@1` estimates vary by 2.2 to 6.0 percentage points depending on which run is selected, with standard deviations exceeding 1.5 percentage points even at temperature 0. This variance has critical implications: reported improvements of 2–3 percentage points may reflect evaluation noise rather than genuine algorithmic progress. Through token-level analysis, we show that trajectories diverge early, often within the first few percent of tokens, and that these small differences cascade into different solution strategies. To enable reliable evaluation of agentic systems, we recommend three concrete practices: (1) estimate `pass@1` from multiple independent runs per task, especially when measuring small improvements, (2) use statistical power analysis to determine the number of runs needed to detect expected effect sizes, and (3) consider metrics like `pass@k` (optimistic bound) and `pass^k` (pessimistic bound) with $k > 1$ to better characterize the full performance envelope. While these practices increase evaluation cost, they are essential for distinguishing genuine scientific progress from statistical noise.

Inference Analysis Data

1 INTRODUCTION

Agentic systems that use tools and interact with environments are becoming increasingly capable. Measuring their performance reliably is essential: evaluation scores guide critical decisions about which models to deploy, whether algorithmic changes provide genuine improvements, and leaderboards are commonly used to quantify how much progress the field is making (Kwa et al., 2025). These decisions have substantial engineering and business consequences. A reported 3% improvement might justify adopting a new model, investing in a particular research direction, or making deployment decisions affecting millions of users. But how reliable are the evaluation scores we use to make these decisions?

Today, most agentic evals follow the approach established for code generation (Kulal et al., 2019; Chen, 2021). Agents are tested on benchmark tasks like `SWE-Bench-Verified` (Jimenez et al.) and scored using `pass@1` – the probability that a task is solved in a single attempt. Despite the name suggesting a statistical estimator, in practice, most researchers run the agent exactly once per task and report the fraction that succeeded. This single-run approach has become standard practice across research papers, model releases, and community leaderboards (Yang et al., 2024).

However, single-run evaluation is methodologically unsound for several reasons. First, estimating `pass@1` from a single binary outcome per task provides a high-variance estimate of the true success probability. Second, sampling with temperature > 0 introduces stochasticity that can produce different outcomes across runs of the same agent on the same task. Third, beyond sampling, environment interactions might introduce further non-determinism through tool execution or timing effects.

*Equal contribution.

In this paper, we quantify how randomness affects agentic evals. We conduct ten independent runs (instead of the standard single run) of six agent configurations on `SWE-Bench-Verified`, systematically varying models and scaffolds. In total, we collect 60,000 agent trajectories, generating over 25.58B tokens and 1.88M tool calls, and systematically analyze their outcomes, performance distributions and divergence points. We demonstrate substantial randomness in evaluation outcomes: single-run `pass@1` estimates vary by 2.2 to 6.0 percentage points depending on which run is observed, with standard deviations exceeding 1.5 percentage points even at temperature 0. This variance persists across all configurations, including theoretically deterministic settings (temperature 0), where non-determinism from inference engines and environments still clearly produces measurable variance. Through token-level trajectory analysis, we find that runs diverge early, often within the first few percent of tokens, and these initial differences cascade into fundamentally different solution strategies through the autoregressive conditioning mechanism of agentic loops. Using `pass@k` (the probability that at least one of k attempts succeeds) (Chen, 2021) and `pass^k` (the probability that all k attempts succeed) (Yao et al., 2025), we find gaps up to 24.9 percentage points between best-case and worst-case performance, revealing how much success depends on stochastic exploration rather than deterministic problem-solving capability.

These findings have critical implications for interpreting progress in agentic AI. Many papers claim small improvements based on single-run `pass@1` scores; our results demonstrate that differences of this magnitude often fall within the natural variance of the evaluation process itself. A reported one-run improvement from, say, 31% to 33% could reflect sampling a favorable run from the same underlying distribution rather than genuine algorithmic progress. To enable sound evaluation of agentic systems, we recommend three concrete practices: (1) estimate `pass@1` from multiple independent runs per task, especially when measuring small improvements, (2) using statistical power analysis to determine how many runs to do, and (3) consider multiple metrics like `pass@k` (optimistic bound) and `pass^k` (pessimistic bound) with $k > 1$ to characterize the full performance envelope.

To summarize, our contributions are: (1) a large-scale empirical study quantifying variance in agentic evals across three models, two scaffolds, and two temperature settings (60,000 trajectories total, 25.58B tokens, 1.88M tool calls); (2) token-level divergence analysis revealing when and how agent trajectories split into different solution strategies; (3) characterization of performance bounds using `pass@k` and `pass^k` metrics, demonstrating gaps up to 24.9 percentage points between optimistic and pessimistic scenarios; and (4) concrete, actionable recommendations for reliable evaluation practices that enable sound scientific progress in agentic AI.

2 Characterizing Randomness in Agentic Evals

Our goal is to characterize the randomness happening in evals or agentic systems, and understand its sources. This is essential for interpreting reported scores on leaderboards: total randomness would mean that rankings are unsound and insignificant for decision making. We design and perform systematic experiments where we run several agents (i.e., model-scaffold pairs) ten times each and analyze the distribution of outcomes and the agentic trajectories. We perform these experiments with both theoretically deterministic sampling (temperature=0.0), as well as with the sampling hyperparameters suggested by the authors of each model, which is the temperature typically used in leaderboards.

2.1 Experimental Setup

We consider agentic coding as the domain of choice for our experiments, as it is one of the most popular and active domains for agentic research. Agentic coding tasks are often highlighted in model cards and used to make claims about trends in AI development (Kwa et al., 2025). Particularly, we focus on the software engineering issue resolution tasks from the `SWE-Bench-Verified` benchmark (Jimenez et al.). This is the most widely used benchmark for agentic coding, and is massively used in model cards and research papers. In `SWE-Bench-Verified`, the agents are tasked with resolving a GitHub issue and their success is validated through automated unit tests.

We exhaustively evaluate six different agents, where an agent is defined as a model-scaffold pairs. We consider the following models:

- `Qwen/Qwen3-32B` Yang et al. (2025) is a medium sized model commonly used by researchers in agentic coding experiments with, at the time of writing, over 1.2M downloads over 300 fine-tuned versions available on Hugging Face. This model is a common model used for research on agentic models (Luo et al.; Tang et al., 2025; Cao et al., 2025; Qian et al., 2025).
- `agentica-org/DeepSWE-preview` Luo et al. is a fine-tuned variant of `Qwen/Qwen3-32B` specifically for agentic coding. For us, this model is meant to represent fine-tuned models for agentic coding.
- `mistralai/Devstral-2-123B-Instruct-2512` (Rastogi et al., 2025) is a large open-weights model specifically fine-tuned for agentic coding, achieving state-of-the-art performance amongst open-weights models on `SWE-Bench-Verified`. It enables us to give perspective on the `DeepSWE-preview` results, both models being fine-tuned for agentic coding.

And the following scaffolds:

- `nano-agent` is our own minimal scaffold for agentic coding experiments, providing a simple yet functional environment for agent-task interaction. We use it because it is guaranteed to not have been used in the training of any models we evaluate, allowing us to reason about scaffold independence with guarantees.
- `R2E-Gym` is a code agent scaffold proposed by Jain et al. and used during the training of `DeepSWE-preview`. This scaffold is more feature-rich than `nano-agent` and allows us to inspect the effect of evaluating a model on its specific training scaffold.

Our selection of models and scaffolds is designed to mitigate potential implementation-specific artifacts that could confound our variance measurements. Specifically, we introduce diversity across two dimensions: (1) scaffold implementation, using both our own `nano-agent` and the independently developed `R2E-Gym`, and (2) model deployment infrastructure, where `Qwen3-32B` and `DeepSWE-preview` are deployed locally with vLLM while `Devstral-2` is accessed through Mistral's hosted API. This orthogonal variation ensures that observed variance patterns are not artifacts of bugs or idiosyncrasies in any single implementation, but generalize well.

Both scaffolds use append-only conversation contexts without any context truncation, compaction, or summarization strategies. This property is important for our trajectory divergence analysis and token accounting methodology (see Section 2.3), as these measurements assume the complete conversation history is preserved throughout the interaction. Studying the randomness induced by compaction is left to future work.

2.2 Agent Trajectories

To understand the mechanisms underlying randomness in evals, we need to analyze agent trajectories at the token level. We formalize the concepts of trajectories and token usage below. These definitions apply to scaffolds with append-only conversation contexts.

An agentic run consists of K interaction steps. At each step k, the model receives a context C_k containing all prior messages and generates a response G_k (which may include reasoning tokens, text, and tool calls). The environment then provides a response R_k (e.g., tool execution results).

Trajectory. We define the trajectory τ_j for run j as the complete linearized sequence of all messages in chronological order, including both model-generated tokens and environment-generated tokens (tool responses):

$$\tau_j = C_1 \oplus G_1 \oplus R_1 \oplus G_2 \oplus R_2 \oplus \cdots \oplus G_K \oplus R_K \tag{1}$$

where $\oplus$ denotes concatenation and C_1 includes the initial system and user prompt. The trajectory includes both model-generated tokens and environment-generated tokens, as both influence subsequent model behavior through autoregressive conditioning.

2.3 Metrics

We employ several complementary metrics to characterize agent performance, each capturing different aspects of agent behavior. We consider a benchmark with N tasks and m independent evaluation

runs on each task (in our case, $N = 500$ and $m = 10$). Let c_i denote the number of successful attempts for task i across all m runs.

Single-run resolution rate: Let r_j denote the resolution rate from run j, computed as the fraction of tasks solved in that run:

$$r_j = \frac{|\{i : \text{task } i \text{ solved in run } j\}|}{N} \tag{2}$$

When we perform m independent runs, we obtain m different values $r_1, r_2, \ldots, r_m$. The mean $\bar{r} = \frac{1}{m}\sum_{j=1}^{m} r_j$ and standard deviation of these values quantify the expected performance and run-to-run variability. In Table 1, we report these statistics to characterize the distribution of outcomes across runs.

`pass@k` and `pass^k`: With multiple evaluation runs on each task, we can compute two complementary metrics to characterize agent capabilities. The `pass@k` metric (Chen, 2021) estimates the probability that at least one of k randomly selected attempts succeeds. The `pass^k` metric (Yao et al., 2025) estimates the probability that all k attempts succeed:

$$\text{pass@}k = \frac{1}{N}\sum_{i=1}^{N}\left[1 - \frac{\binom{m-c_i}{k}}{\binom{m}{k}}\right] \quad \text{and} \quad \text{pass}^{\wedge}k = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{\binom{c_i}{k}}{\binom{m}{k}}\right] \tag{3}$$

where $\binom{a}{b}$ denotes the binomial coefficient.

The `pass@k` metric answers: "If we randomly select k of our m attempts for each task, what fraction of tasks would be solved at least once?" For $k = 1$, this estimator reduces to $\frac{1}{N}\sum_{i=1}^{N}\frac{c_i}{m} = \bar{r}$, which is exactly the mean of single-run resolution rates. Thus, `pass@1` $= \bar{r}$ represents the pooled estimate of first-attempt success probability across multiple runs. For $k > 1$, the `pass@k` estimator properly accounts for the combinatorics of sampling and differs from simply averaging empirical success rates. It also represents an optimistic bound of model capabilities. Complementarily, `pass^k` measures consistency and robustness, also representing a pessimistic bound of model capabilities. A high `pass^k` indicates that the agent reliably solves tasks across multiple attempts, while a low `pass^k` relative to `pass@k` suggests success depends heavily on stochastic exploration.

In many related works: 1) `pass@1` is reported as the only metric; 2) too often based on a single run; 3) too rarely, the number of runs used to estimate it is reported.

First token divergence (τ_{div}): To understand when and how agent runs diverge, we measure the first position at which two trajectories differ. Using the trajectory formalism from Section 2.2, for two runs i and j on the same task with tokenized trajectories $\tau_i = [t_1^i, t_2^i, \ldots]$ and $\tau_j = [t_1^j, t_2^j, \ldots]$, we define:

$$\tau_{\text{div}}(i, j) = \min\{k : t_k^i \neq t_k^j\} \tag{4}$$

This metric captures when trajectories begin to explore different solution paths, which is critical for understanding variance propagation in agentic evals.

2.4 Experimental Results

Our experiment yields 60,000 agent trajectories in total, from 120 experimental runs (6 configurations $\times$ 10 runs each $\times$ 2 scaffolds). These runs consumed 25.58B tokens, generating 1.88M tool calls. In this section, we quantify the randomness in evaluation outcomes, as well as try to understand the mechanisms behind it.

2.4.1 Quantifying Randomness in Evaluation Outcomes

Table 1 presents the single-run resolution rates r_j (percentage of tasks successfully resolved) for each model-scaffold-temperature combination, aggregated across 10 independent evals per agent under test. We report the mean ($\bar{r}$ = `pass@1`), standard deviation, minimum, and maximum values to characterize the distribution of outcomes. Across all conditions, we observe substantial run-to-run variability. For example, `DeepSWE-preview` on `nano-agent` with temperature 1.0 achieves $\bar{r} = 31.4 \pm 1.0\%$ (`pass@1`), with individual runs ranging from 28.8% to 32.4% (a 3.6 percentage point spread. Similarly, `Qwen3-32B` on `R2E-Gym` with temperature 0.6 shows a mean of $23.9 \pm 1.4\%$, ranging from 21.4% to 26.4% (a 5.0 percentage point spread). Across all twelve

Table 1: Resolution rates across 10 independent evals on `SWE-Bench-Verified`. Each row shows statistics of r_j (single-run resolution rates) computed over 10 separate runs with identical configuration. Mean values ($\bar{r}$) are equivalent to `pass@1` estimated by pooling all runs (see Section 2.3), while standard deviation quantifies run-to-run variability. The substantial ranges (min to max) demonstrate the presence of randomness in evaluation outcomes, even with identical settings and temperature 0.

Model	Temp	`nano-agent`			`r2e-gym`		
		$\bar{r}\pm$Std	Min	Max	$\bar{r}\pm$Std	Min	Max
`Qwen3-32B`	0.6	$16.4\% \pm 0.7\%$	15.0%	17.2%	$23.9\% \pm 1.4\%$	21.4%	26.4%
`DeepSWE-preview`	1.0	$31.4\% \pm 1.0\%$	28.8%	32.4%	$34.4\% \pm 1.5\%$	31.6%	37.0%
`Devstral-2`	0.2	$63.5\% \pm 1.1\%$	61.8%	65.0%	$34.9\% \pm 1.5\%$	32.2%	37.0%
`Qwen3-32B`	0.0	$16.4\% \pm 1.2\%$	14.4%	18.0%	$22.3\% \pm 1.8\%$	19.8%	25.2%
`DeepSWE-preview`	0.0	$20.4\% \pm 1.0\%$	18.2%	21.4%	$19.2\% \pm 1.5\%$	17.0%	21.6%
`Devstral-2`	0.0	$63.8\% \pm 1.6\%$	60.6%	66.6%	$35.4\% \pm 1.7\%$	32.0%	37.8%

configurations in the table, the ranges span 2.2 to 6.0 percentage points, representing substantial variability where a single run could report performance anywhere within this window. This variability is significant enough that improvements measured with a single run might be purely due to randomness in the evaluation process rather than a genuine improvement.

"Deterministic" sampling. In theory, evals can be run deterministically with temperature 0. In practice, determinism is not achievable (Yuan et al., 2025; He & Lab, 2025): modern LLM inference engines introduce various sources of non-determinism including floating-point precision, parallelization, hardware-specific optimizations, and batching strategies. We study the extent of the problem at temperature 0 (bottom half of the table). Clearly, this variance persists even with theoretically deterministic sampling (temperature 0.0). For instance, `DeepSWE-preview` on `nano-agent` (temp zero) achieves $20.4 \pm 1.0\%$ (range: 18.2%–21.4%), and `Qwen3-32B` on `R2E-Gym` achieves $22.3 \pm 1.8\%$ (range: 19.8%–25.2%). Counter-intuitively, the variance never decreases and sometimes increases with temperature zero (eg 0.7% variance for Qwen3-32B x nano-agent at temperature 0.6 to 1.2% variance at temperature 0). To sum up, temperature 0 does not result in determinism.

Statistical significance of temperature effects. The impact of temperature on performance varies significantly across models, highlighting the importance of statistical testing rather than relying solely on single-runs or mean differences. For `DeepSWE-preview`, temperature 1.0 achieves $31.4\% \pm 1.0\%$ on `nano-agent` versus $20.4\% \pm 1.0\%$ at temperature 0.0 (11.0 percentage point difference). Given the standard deviations, this difference is statistically significant, indicating that stochastic exploration genuinely improves the problem-solving capability of this model. In contrast, `Devstral-2` shows no statistically significant difference: on `nano-agent`, temperature 0.2 achieves $63.5\% \pm 1.1\%$ versus $63.8\% \pm 1.6\%$ at temperature 0.0. Despite the slightly higher mean at temperature 0.0, the 0.3 percentage point difference is well within the noise given the large standard deviations (1.1% and 1.6%). These examples demonstrate that statistical testing is essential to distinguish genuine effects from random variation, as we further develop in Section 3.2 and Section A.

Implications. Our results demonstrate substantial randomness in agentic evaluation outcomes, even under identical configurations. The practical implications are significant: single-run scores in technical reports and articles can be misleading without statistical bounds. Consider evaluating whether a model improves performance. Observing a 3 percentage point increase on a single run could be purely due to randomness in the evaluation process rather than a genuine improvement. This affects how we interpret progress in the field: a newly released model claiming a 2-3 point improvement over its predecessor might simply be reporting a favorable outcome from the same underlying distribution. To enable sound decision-making, for research, development, or deployment, evaluation results should be reported with variance estimates over multiple runs rather than point estimates from single runs.

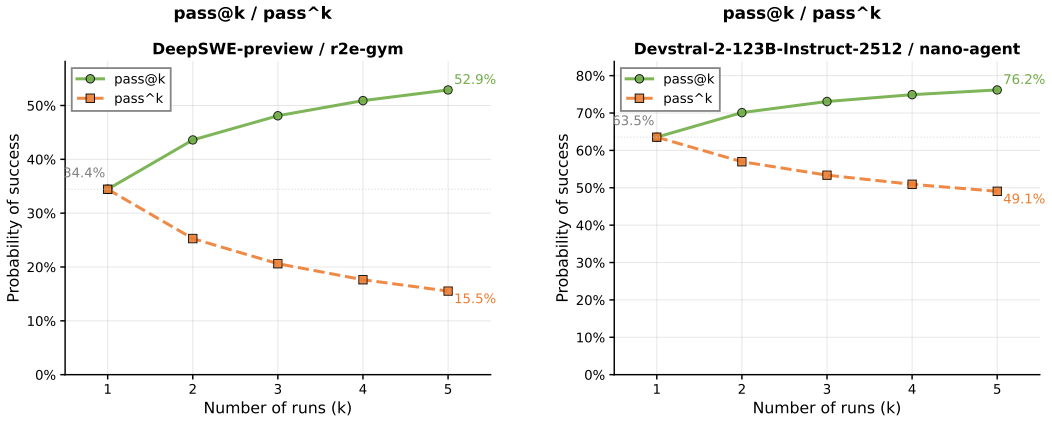


Figure 1: Performance bounds revealed by `pass@k` and `pass^k` for `DeepSWE-preview` on `r2e-gym` and `Devstral-2` on `nano-agent`. The vertical distance between curves quantifies how much performance depends on random choices. `DeepSWE-preview` exhibits wider gaps (high sensitivity to randomness), while `Devstral-2` shows narrower gaps (more consistent solutions), though both demonstrate substantial dependence on stochastic exploration as k increases.

2.4.2 What do pass@1, pass@k, and pass^k reveal about agent randomness?

Randomness in agentic trajectories has a good aspect: it can be leveraged to increase performance via retrying. We now examine the extremes. What is the best performance we could achieve if we exploit this randomness via retries? What is the worst performance we can expect when randomness is maximally unfavorable?

We analyze three metrics that capture different aspects of this question, see Section 2.3. A large (`pass@k - pass@1`) gap indicates the agent has the potential to solve many tasks, but needs multiple attempts to find the right path. A large (`pass@1 - pass^k`) gap indicates the agent's success is highly sensitive to which random choices are made and might not be able to reliably produce consistent solutions. Together, these metrics bound the agent's capabilities.

Figure 1 shows those three metrics for two modelxscafold pairs. It reveals substantial gaps between these bounds, exposing how much agent performance depends on randomness. For `DeepSWE-preview` on `r2e-gym`, the first-attempt success probability (`pass@1`) is 34.4%, but with five retries, performance reaches 52.9% (`pass@5`), an 18.5 percentage point improvement representing the optimistic potential. At the other extreme, only 15.5% of tasks (`pass^5`) are solved consistently across five attempts, which is less than half of the `pass@1` rate. This 18.9 percentage point gap between `pass@1` (34.4%) and `pass^5` (15.5%) reveals that part of the agent's capability depends on favorable random choices.

This pattern generalizes across configurations, though the magnitude varies. Consider Figure 1, where we show the `pass@k` and `pass^k` curves for `DeepSWE-preview` on `r2e-gym` and `Devstral-2` on `nano-agent` (other model-scaffold pairs are provided in the appendix)). `Devstral-2` on `nano-agent` shows a narrower range: `pass@1` is 63.5%, `pass@5` reaches 76.2% (12.7 point gap to optimistic bound), and `pass^5` is 49.1% (14.4 point gap to pessimistic bound). These narrower gaps indicate that higher-performing models exhibit more consistent solution strategies, yet they still benefit significantly from stochastic exploration. Across all twelve configurations, the maximum improvement from `pass@1` to `pass@5` is 24.9 percentage points (`Devstral-2` on `r2e-gym` at temperature 0), demonstrating that scaffold choice and model-scaffold interaction significantly impact the degree of stochastic dependence.

In summary, the gap between `pass@k` (optimistic bound) and `pass^k` (pessimistic bound) quantifies how much agent performance depends on favorable stochastic exploration. This dependence is substantial across all configurations and demonstrate that randomness is not a minor perturbation but a fundamental component of agent performance, with implications for evaluation methodologies and interpretation of results.

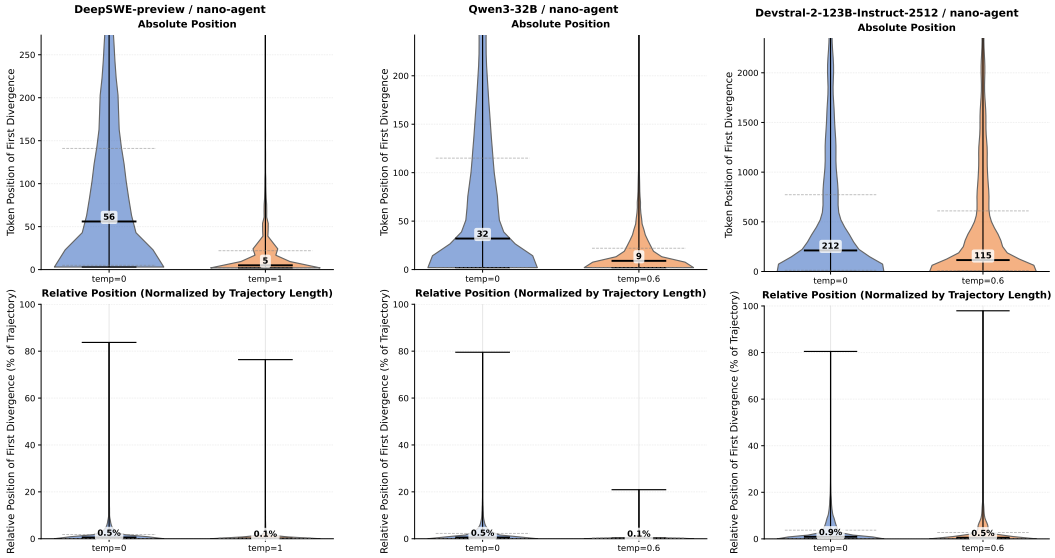


Figure 2: Distribution of first token divergence across different models under `nano-agent`. In blue, we show the distributions with temperature 0, while in orange we show the distributions with the suggested temperatures. On top, we plot by absolute token position, while on bottom, we plot by relative position (percentage through the trajectory). The distributions are shown for all pairs of divergent runs, one per model-scaffold pair.

2.4.3 Understanding the Mechanism: When Do Trajectories Diverge?

To qualitatively understand the underlying randomness in agentic outcomes, we analyze when and how runs diverge at the trajectory level. Once a single token differs between two runs, the probability distribution (logits) computed by the LLM for subsequent tokens also changes, since the model conditions on the full context including the divergent token. This creates a butterfly effect: a single early difference can propagate through the trajectory, affecting ever more tokens, tool calls, observations, and ultimately the final outcome.

Figure 2 shows the distribution of first token divergence position with (blue) and without (red) deterministic sampling, across all pairs of runs, one per model-scaffold pair. The top plots show the distribution of divergence by absolute token position, while the bottom plot shows the distribution by relative position (percentage through the trajectory), revealing whether divergence is consistently early regardless of trajectory length.

Early divergence. Figure 2 shows that the distribution of first token divergence reveals that trajectories typically diverge very early, within the first tokens (top figure) immediately after the common system and user prompts. Relative to the size of the trace, the first divergence always happen in the 1% of the total trajectory (bottom of the figure). For example, for `DeepSWE-preview` on `nano-agent`, the median first token divergence occurs at token position 5 with default temperature (1.0), that is at 0.5% of the total trajectory length.

Temperature effect on trajectories. Deterministic sampling (temperature 0.0) shifts divergence substantially later, as expected. For `DeepSWE-preview` on `nano-agent`, the median first token divergence position increases from 5 (default temperature 1.0) to 56 when using temperature 0.0. Similarly, `Qwen3-32B` on `nano-agent` exhibits median divergence at token 9 with default temperature (0.6), increasing to token 32 at temperature 0.0. The same phenomenon is observed for `Devstral-2`. So, temperature does have a little positive impact.

Yet, confirming the results of Section 2.4.1, deterministic sampling only delays divergence, but does not suppress it. Since trajectory divergence happens early, we expect that the longer the trajectories, the more likely they are to also diverge semantically, because of cascading effects in the next token probability distributions. As the community moves towards more complex and long-horizon tasks, the importance of measuring variance will only increase.



Figure 3: A subtle reasoning divergence at token 94 cascades into opposite outcomes. Both runs share identical reasoning through the first paragraph, understanding the task of adding an `__iter__` method to Django's `Paginator` class. At token 94, the reasoning diverges: run 1 reasons "Let me search..." while run 2 reasons "Let me check...Using the shell tool...". This difference leads to a different first tool call, which propagates through subsequent steps, with only run 2 succeeding. Even at temperature 0, non-determinism causes trajectory divergence that compounds into fundamentally different problem-solving strategies.

Case study. Figure 3 illustrates a concrete example of how early divergence in reasoning cascades into drastically different outcomes. On task `django__django-9296`, two independent runs of `Qwen3-32B` on `nano-agent` at temperature 0 generated identical reasoning for the first 93 tokens. At token 94, within the model's internal reasoning trace, a subtle difference emerged: run 1 reasoned "Let me search for the Paginator class" while run 2 reasoned "Let me check the Django source code." This seemingly minor phrasing difference propagated through the autoregressive conditioning to lead to different first tool calls, where the first searches for the `Paginator` class in a specific file while the other searches in an entire directory, leading to different tool call outputs. At the end, this cascades into fundamentally different problem-solving strategies, and with opposite outcomes. Run 1 found the correct file but applied an incorrect patch that inserted the new method in the wrong location breaking Python syntax, failing the task. Run 2's broader search led to more careful analysis of the code structure, ultimately finding the correct insertion point and successfully resolving the task. Even at temperature 0, randomness causes divergent reasoning that compounds through autoregressive conditioning into fundamentally different problem-solving strategies with opposite outcomes.

To sum up, early divergences have important implications for long-horizon agentic tasks. Since divergence occurs early and propagates through the remainder of the trajectory via the autoregressive conditioning mechanism, longer trajectories exhibit amplified variance. In agentic evals, more than zero-shot prompting, small initial perturbations lead to increasingly divergent outcomes as the trajectory lengthens.

Table 2: Required runs per agent to detect improvements at different variance and significance levels (power = 80%).

Improvement	**Std Dev** (σ)	n ($p < 0.05$)	n ($p < 0.01$)	n ($p < 0.001$)
1%	0.7%	8	12	17
1%	1.5%	36	53	77
1%	1.8%	51	76	111
2%	0.7%	2	3	5
2%	1.5%	9	14	20
2%	1.8%	13	19	28
5%	0.7%	1	1	1
5%	1.5%	2	3	4
5%	1.8%	3	4	5
10%	0.7%	1	1	1
10%	1.5%	1	1	1
10%	1.8%	1	1	2

3 Implications and Mitigation Strategies

3.1 False sense of progress

The variance documented in this paper has immediate consequences for how we interpret progress in agentic systems. Single-run evaluations can lead to researchers not being able to determine whether observed differences represent genuine capability gaps or merely different samples from overlapping performance distributions. This problem extends beyond individual papers to affect the broader scientific ecosystem. Leaderboards that rank systems based on single-run scores may reflect evaluation noise rather than true capability ordering. Research directions may be chosen based on apparent improvements that are not statistically distinguishable from noise. Organizations making deployment decisions, deciding whether to adopt a new model or agentic tool, or allocating engineering resources, face similar challenges. The scores guiding these decisions may not reliably reflect underlying performance differences.

The problem is particularly acute because evaluation practices have not kept pace with the evolution of agentic systems. While the `pass@1` metric originated in code generation settings with relatively short, independent generations like HumanEval (Chen, 2021), agentic tasks involve long-horizon, multi-step trajectories where early divergence cascades through subsequent actions. Our trajectory analysis (Section 2.2) demonstrates that this cascading effect might amplify variance. As the field moves toward longer-horizon tasks with more complex tool use, this amplification effect is likely to intensify further.

3.2 Recommendations

To enable reliable evaluation, we recommend running multiple runs per agent under test, and estimating the performance metrics from them. The required number of runs depends on the magnitude of improvement to detect and the desired statistical power (the probability of correctly identifying a real improvement when it exists). Table 2 shows the required number of runs per agent under test for detecting different improvement magnitudes at various significance levels, assuming a normal distribution of randomness.[1] The table presents three variance scenarios corresponding to the minimum ($\sigma = 0.7\%$), median ($\sigma = 1.5\%$), and maximum ($\sigma = 1.8\%$) standard deviations observed across our experiments.

Detecting a 2% improvement at $p < 0.05$ with 80% power requires approximately 9 runs per agent under test, while detecting a 1% improvement requires 36 runs. A study like ours, in which 10 runs are made per agent under test, can reliably detect improvements ≥ 2 percentage points but not smaller effects.

[1]We verified this assumption via Shapiro-Wilk tests (Shapiro & Wilk, 1965) across all 12 configuration–scaffold groups ($n = 10$ runs each); 11 of 12 passed at $p \geq 0.05$. See Section B for details.

Detecting a 1% improvement at median variance levels requires 36 runs, while the same detection at the lowest observed variance ($\sigma = 0.7\%$) would require only 8 runs. On the other hand, detecting large improvements (e.g., 10%) can be done with a much smaller number of runs and, depending on the desired statistical power and significance threshold, might even be possible with single-runs. Further analysis can be found in Section A.

We also suggest characterizing the performance envelope: by always reporting `pass@1` (expected performance), `pass@k` (optimistic bound with retries), and `pass^k` (pessimistic consistency bound). The `pass@k-pass^k` gaps reveal how much stochasticity might be detrimental or beneficial for the agent under test.

We acknowledge that multiple runs increase cost, which is a valid concern for GPU-poor organizations, in particular in academic settings like ours. However, this investment is necessary to avoid long term costs due to poorly informed decisions, at local and systemic levels.

4 Related Work

4.1 Randomness in Large Language Models

Recent work has identified multiple sources of non-determinism in large language models. At the infrastructure level, Yuan et al. (2025) and He & Lab (2025) demonstrate that non-associative floating point operations, rounding errors, hardware configuration, and batch size variations impact reproducibility at temperature 0. For code generation specifically, Ouyang et al. (2025) find that repeated queries yield different implementations even with greedy sampling. Prompt sensitivity represents another major source of variance. Zhuo et al. (2024); Sclar et al. (2024); Andersson et al. (2025) show that meaning-preserving changes (spacing, punctuation, example ordering, case) cause substantial performance shifts. All sources of non-determinism impact agentic evaluation scores.

Most directly related to our work, Mustahsan et al. (2025) proposes using intraclass correlation to quantify evaluation stability in agentic systems, showing that stability varies with task complexity and model capability. Biderman et al. (2024) document broader reproducibility challenges in few-shot language model evals, and propose a harness for standardized assessment. Madaan et al. (2024) propose methods for quantifying and understanding variance in evaluation benchmarks. Pimentel et al. (2024) conduct an analysis exposing how different evaluation frameworks introduce variability in LLM evals. Heineman et al. propose a framework for distinguishing between meaningful signal and noise in evals using a signal-to-noise ratio. Shen et al. (2026) apply the same signal-to-noise ratio to assess the reliability of their agentic training findings, and find a median standard deviation of 1.2% in their experiments with `SWE-Bench-Verified`.

Work on agent diversity (Audran-Reiss et al., 2025) demonstrates that behavioral diversity is an important factor in achieving higher performance by enabling creative search of diverse solutions to the same problem. Wang et al. (2023) propose self-consistency to exploit variance beneficially by sampling multiple reasoning paths. These works recognize that variance exists in large language model inference, and leverage it to improve performance on a certain task. Complementing these, our work highlights the importance of accounting for this variance when interpreting agentic evaluation scores. We provide the first extensive analysis of when and why multi-step agent trajectories diverge.

4.2 Reproducibility in Machine Learning

Reproducibility challenges are pervasive in science (Collaboration, 2015; Baker, 2016). Machine learning research is no exception. Henderson et al. (2018) report difficulties in reproducing baselines and Agarwal et al. (2021) show that the shift to computationally expensive benchmarks led to the detrimental practice of evaluating on a small number of runs per task. In the large language model domain, similar concerns have led to proposals for standardized reporting with multiple runs and confidence intervals (Dodge et al., 2019; Biderman et al., 2024; Miller, 2024). Our work extends these methodological insights for LLM prompting to multi-step agentic evals, which also suffer from single or too small number of runs, misleading researchers and practitioners alike.

5 CONCLUSION

We have demonstrated that randomness fundamentally affects the reliability of agentic evals. Through 60,000 trajectories and 25.58B tokens across six agentic systems (three models and two scaffolds), we quantified substantial variance in single-run `pass@1` estimates (2.2–6.0 pp ranges, persisting at temperature 0). We traced the problem to early trajectory divergence (median within first 1% of tokens) that cascades through autoregressive conditioning. We characterized performance envelopes showing gaps up to 24.9 pp between optimistic and pessimistic bounds. Future work should investigate how dynamic context strategies (e.g., context compactation), widely used in production systems but excluded from our study, affect evaluation variance, and extend this analysis to longer-horizon tasks, where cascading effects may amplify our findings.

ACKNOWLEDGMENTS

This work was partially supported by the WASP program funded by Knut and Alice Wallenberg Foundation. Some computation was enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS).

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Vivi Andersson, Benoit Baudry, Sofia Bobadilla, Ludvig Christensen, Serena Cofano, Khashayar Etemadi, Raphina Liu, Martin Monperrus, Frank Reyes García, Javier Ron Arteaga, et al. Uppercase is all you need. *SIGBOVIK*, pp. 24–35, 2025.

Alexis Audran-Reiss, Jordi Armengol-EstapÃŠ, Karen Hambardzumyan, Amar Budhiraja, Martin Josifoski, Edan Toledo, Rishi Hazra, Despoina Magka, Michael Shvartsman, Parth Pathak, et al. What does it take to be a good ai research agent? studying the role of ideation diversity. *arXiv preprint arXiv:2511.15593*, 2025.

Monya Baker. 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604):452–454, 2016.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Lessons from the trenches on reproducible evaluation of language models. *arXiv preprint arXiv:2405.14782*, 2024.

Shiyi Cao, Dacheng Li, Fangzhou Zhao, Shuo Yuan, Sumanth R Hegde, Connor Chen, Charlie Ruan, Tyler Griggs, Shu Liu, Eric Tang, et al. Skyrl-agent: Efficient rl training for multi-turn llm agent. *arXiv preprint arXiv:2511.16108*, 2025.

Mark Chen. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Open Science Collaboration. Estimating the reproducibility of psychological science. *Science*, 349 (6251):aac4716, 2015.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A Smith. Show your work: Improved reporting of experimental results. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2185–2194, 2019.

Horace He and Thinking Machines Lab. Defeating nondeterminism in llm inference. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250910. https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/.

David Heineman, Valentin Hofmann, Ian Magnusson, Yuling Gu, Noah A Smith, Hannaneh Hajishirzi, Kyle Lo, and Jesse Dodge. Signal and noise: A framework for reducing uncertainty in language model evaluation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Naman Jain, Jaskirat Singh, Manish Shetty, Tianjun Zhang, Liang Zheng, Koushik Sen, and Ion Stoica. R2e-gym: Procedural environments and hybrid verifiers for scaling open-weights swe agents. In *Second Conference on Language Modeling*, 2025.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*.

Sumith Kulal, Panupong Pasupat, Kartik Chandra, Mina Lee, Oded Padon, Alex Aiken, and Percy S Liang. Spoc: Search-based pseudocode to code. *Advances in Neural Information Processing Systems*, 32, 2019.

Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, et al. Measuring ai ability to complete long tasks. *arXiv preprint arXiv:2503.14499*, 2025.

Michael Luo, Naman Jain, Jaskirat Singh, Sijun Tan, Ameen Patel, Qingyang Wu, Alpay Ariyak, Colin Cai, Tarun Venkat, Shang Zhu, et al. Deepswe: Training a state-of-the-art coding agent from scratch by scaling rl, 2025. *Notion Blog*.

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.

Zairah Mustahsan, Abel Lim, Megna Anand, Saahil Jain, and Bryan McCann. Stochasticity in agentic evaluations: Quantifying inconsistency with intraclass correlation. *arXiv preprint arXiv:2512.06710*, 2025.

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

Marco AF Pimentel, Clément Christophe, Tathagata Raha, Prateek Munjal, Praveen K Kanithi, and Shadab Khan. Beyond metrics: A critical analysis of the variability in large language model evaluation frameworks. *arXiv preprint arXiv:2407.21072*, 2024.

Cheng Qian, Zuxin Liu, Akshara Prabhakar, Jielin Qiu, Zhiwei Liu, Haolin Chen, Shirley Kokane, Heng Ji, Weiran Yao, Shelby Heinecke, et al. Userrl: Training interactive user-centric agent via reinforcement learning. *arXiv preprint arXiv:2509.19736*, 2025.

Abhinav Rastogi, Adam Yang, Albert Q Jiang, Alexander H Liu, Alexandre Sablayrolles, Amélie Héliou, Amélie Martin, Anmol Agarwal, Andy Ehrenberg, Andy Lo, et al. Devstral: Fine-tuning language models for coding agent applications. *arXiv preprint arXiv:2509.25193*, 2025.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. International Conference on Learning Representations, 2024.

S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965. ISSN 00063444, 14643510. URL `http://www.jstor.org/stable/2333709`.

Ethan Shen, Danny Tormoen, Saurabh Shah, Ali Farhadi, and Tim Dettmers. Sera: Soft-verified efficient repository agents. *arXiv preprint arXiv:2601.20789*, 2026.

Qiaoyu Tang, Hao Xiang, Le Yu, Bowen Yu, Yaojie Lu, Xianpei Han, Le Sun, WenJuan Zhang, Pengbo Wang, Shixuan Liu, et al. Beyond turn limits: Training deep search agents with dynamic context window. *arXiv preprint arXiv:2510.08276*, 2025.

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

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*, 2025.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.15793.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *The Thirteenth International Conference on Learning Representations*, 2025.

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 *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. Prosa: Assessing and understanding the prompt sensitivity of llms. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1950–1976, 2024.

A Statistical Power Analysis for Determining the Number of Runs

This appendix details the mathematical framework for determining the required number of runs to reliably detect differences in `pass@1` scores.

Consider two experimental conditions (e.g., two models, two temperatures, or two scaffolds) with true `pass@1` values μ_1 and μ_2. When we run each condition n times on a benchmark, we obtain sample means $\bar{x}_1$ and $\bar{x}_2$ with standard deviations σ_1 and σ_2.

Given a desired significance level α (probability of rejecting H_0 when it is true) and statistical power $1-\beta$ (probability of rejecting H_0 when it is false), we aim to determine the required number of runs n to reliably detect a difference of magnitude $\Delta = |\mu_1 - \mu_2|$.

We frame this as a two-sample hypothesis test:

$$H_0 : \mu_1 = \mu_2 \quad \text{(no difference)} \tag{5}$$

$$H_a : \mu_1 \neq \mu_2 \quad \text{(difference)} \tag{6}$$

For a two-sample t-test of means with equal sample sizes, assuming known and equal standard deviation σ, the required number of runs per agent under test ($n_1 = n_2 = n$) can be computed as follows:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sigma\sqrt{\frac{2}{n}}} \tag{7}$$

Under the null hypothesis $H_0 : \mu_1 = \mu_2$, we reject H_0 if $|t| > Z_{\alpha/2}$. Under the alternative hypothesis $H_a : \mu_1 \neq \mu_2$, the expected value of the test statistic is:

$$\mathbb{E}[t] = \frac{\Delta}{\sigma\sqrt{2/n}} \tag{8}$$

For the test to achieve power $1-\beta$, the expected test statistic must exceed the critical value by at least Z_β standard errors:

$$\frac{\Delta}{\sigma\sqrt{2/n}} \geq Z_{\alpha/2} + Z_\beta \tag{9}$$

$$\sqrt{\frac{n}{2}} \geq \frac{(Z_{\alpha/2} + Z_\beta)\sigma}{\Delta} \tag{10}$$

$$n \geq 2\left(\frac{Z_{\alpha/2} + Z_\beta}{\Delta/\sigma}\right)^2 \tag{11}$$

Where $Z_{\alpha/2}$ is the critical value from the standard normal distribution for a two-tailed test at significance level α, and Z_β is the critical value corresponding to the desired power $1-\beta$.

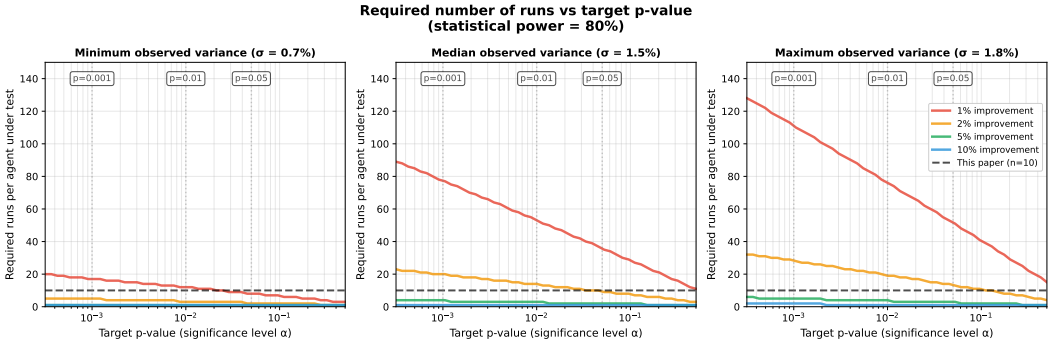


Figure 4: Required number of runs per agent under test for detecting improvements of different magnitudes (1%, 2%, 5%, 10%) under three variance scenarios observed in our experiments, at significance level $p < 0.05$ and 80% statistical power. The minimum variance scenario ($\sigma = 0.7\%$) represents the most favorable case, while the maximum variance ($\sigma = 1.8\%$) represents the most challenging evaluation conditions. The exponential increase in required runs for smaller improvements, particularly at higher variance levels, demonstrates that single-run evals cannot reliably distinguish small performance differences from random variations.

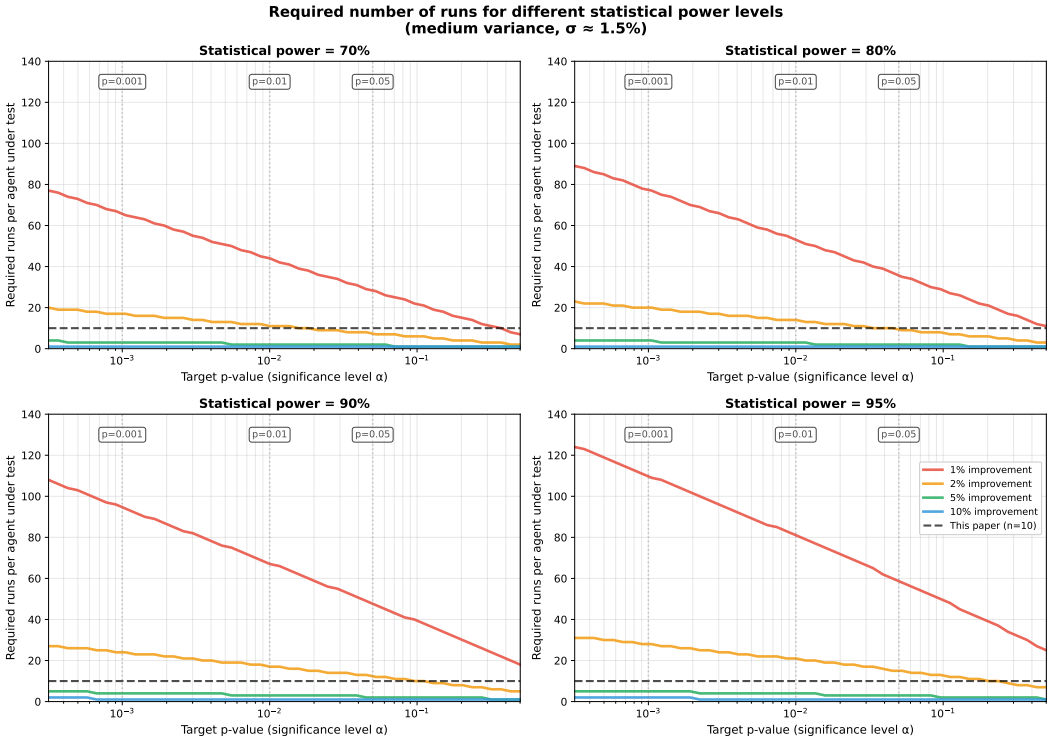


Figure 5: Required number of runs per agent under test for detecting improvements of different magnitudes (1%, 2%, 5%, 10%) at varying statistical power levels (70%, 80%, 90%, 95%), assuming median observed variance ($\sigma = 1.5\%$) and significance level $p < 0.05$. Higher desired statistical power requires substantially more runs, particularly for detecting small improvements. For example, detecting a 2% improvement with 80% power requires 9 runs per agent, while achieving 95% power for the same effect size requires 15 runs. The exponential growth in required sample size for smaller effect sizes demonstrates why single-run evals are insufficient for reliably detecting small improvements.

B NORMALITY OF PASS@1 SCORES

The power analysis in Section A assumes that pass@1 scores are approximately normally distributed across runs. We verified this assumption empirically using the Shapiro-Wilk test (Shapiro & Wilk, 1965), which tests the null hypothesis that the data are drawn from a normal distribution.

Table 3 reports the results for all 12 configuration–scaffold groups ($n = 10$ runs each). Eleven of the 12 groups pass the test at $p \geq 0.05$. The one exception, DeepSWE-preview on nano-agent ($W = 0.80$, $p = 0.016$), is also the group with the smallest observed variance ($\sigma = 1.0\%$). With $n = 10$, the test has limited power to detect departures from normality, so passing is not a confirmation of exact normality; nonetheless, the results are consistent with the normal approximation being adequate for the purpose of power analysis.

Table 3: Shapiro-Wilk normality test results for pass@1 scores across 10 runs per configuration–scaffold group. W is the test statistic; a p-value ≥ 0.05 indicates no significant departure from normality at the 5% level.

Model	Scaffold	Mean	Std	W	p-value
DeepSWE-preview	nano-agent	31.4%	1.0%	0.803	0.016
DeepSWE-preview	r2e-gym	34.4%	1.5%	0.989	0.996
DeepSWE-preview-temp0	nano-agent	20.4%	1.0%	0.881	0.134
DeepSWE-preview-temp0	r2e-gym	19.2%	1.5%	0.923	0.383
Devstral-2	nano-agent	63.5%	1.1%	0.894	0.190
Devstral-2	r2e-gym	34.9%	1.5%	0.956	0.733
Devstral-2-temp0	nano-agent	63.8%	1.6%	0.970	0.893
Devstral-2-temp0	r2e-gym	35.4%	1.7%	0.952	0.695
Qwen3-32B	nano-agent	16.4%	0.7%	0.885	0.149
Qwen3-32B	r2e-gym	23.9%	1.4%	0.964	0.834
Qwen3-32B-temp0	nano-agent	16.4%	1.2%	0.962	0.809
Qwen3-32B-temp0	r2e-gym	22.3%	1.8%	0.964	0.830

C Inference Hyper-Parameters

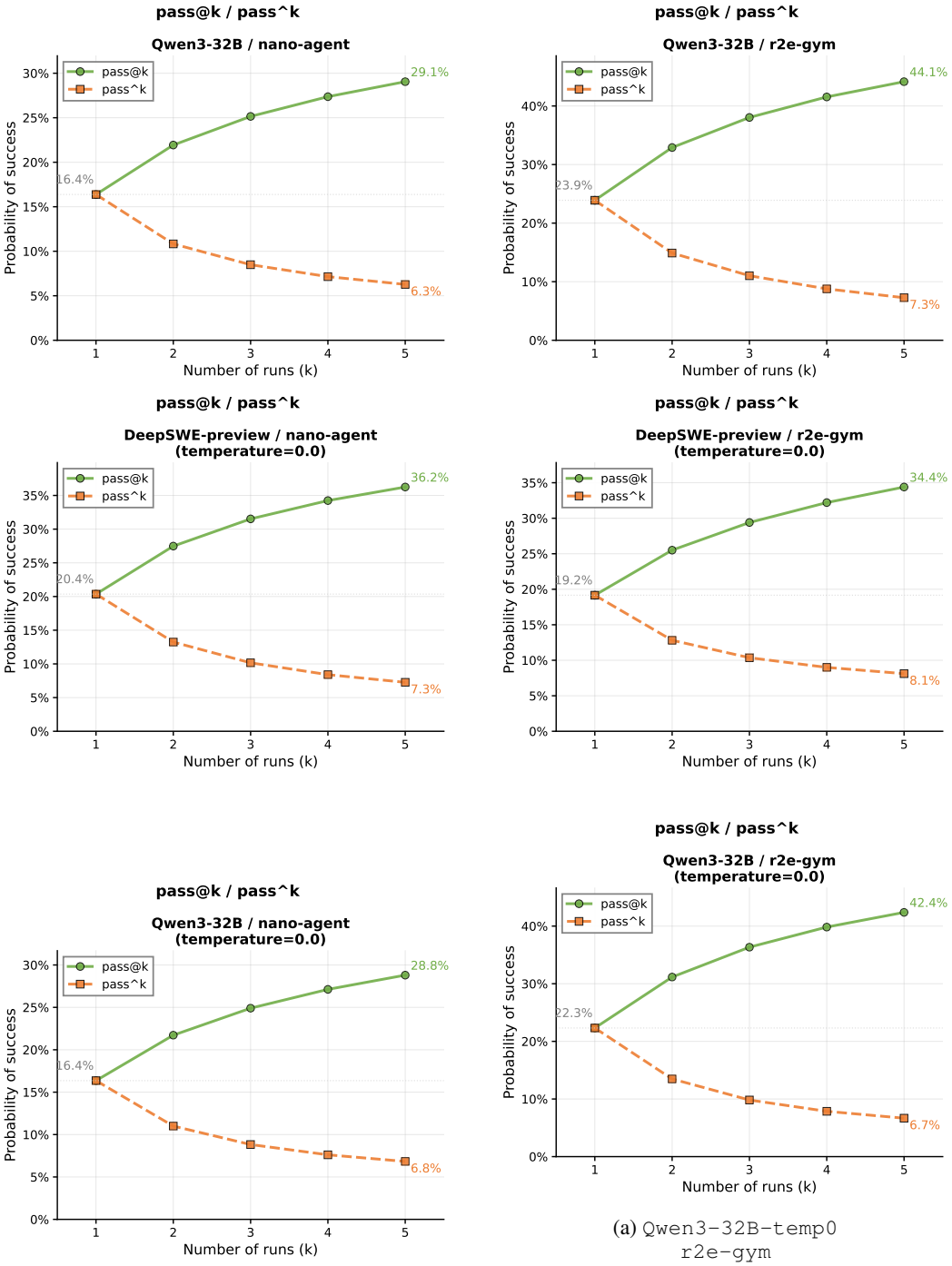
This section details the hyperparameters used for each model in our experiments. All locally deployed models (`Qwen3-32B` and `DeepSWE-preview`) were hosted using vLLM on NVIDIA A100 80GB GPUs, using a total of approx. 3,500 GPU hours. `Devstral-2` was accessed through Mistral's API.

Both scaffolds use their default configurations. For `nano-agent`, we set a maximum of 500 tool calls per run, while `r2e-gym` allows up to 100.

Table 4: Model inference configuration.

Model	Deployment	Context Limit	Temperature	top_p	top_k
`Qwen3-32B`	vLLM (8 x A100 80GB)	65,536	0.6 / 0.0	0.95	20
`DeepSWE-preview`	vLLM (8 x A100 80GB)	65,536	1.0 / 0.0	—	—
`Devstral-2`	Mistral API	256,000	0.2 / 0.0	—	—

Table 4 presents the inference hyperparameters for each model. For `Qwen3-32B`, we use the sampling parameters used by the authors (Yang et al., 2025) when evaluating the model in thinking mode, with the exception of the context limit, which we increase to 65,536 tokens. For `DeepSWE-preview`, we use the temperature suggested in the model card. For `Devstral-2`, we follow the recommendations in the release post.Temperature 0 experiments use greedy decoding for all models.



(a) `Qwen3-32B-temp0 r2e-gym`

Figure 6: Additional `pass@k` and `pass^k` curves for all model-scaffold pairs (part 1/2).

D PASS@K PLOTS

Figure 1 in the main paper shows `pass@k` and `pass^k` curves for `DeepSWE-preview` on both `nano-agent` and `r2e-gym`. For completeness, we provide additional `pass@k` plots for all other model-scaffold pairs evaluated in this study.

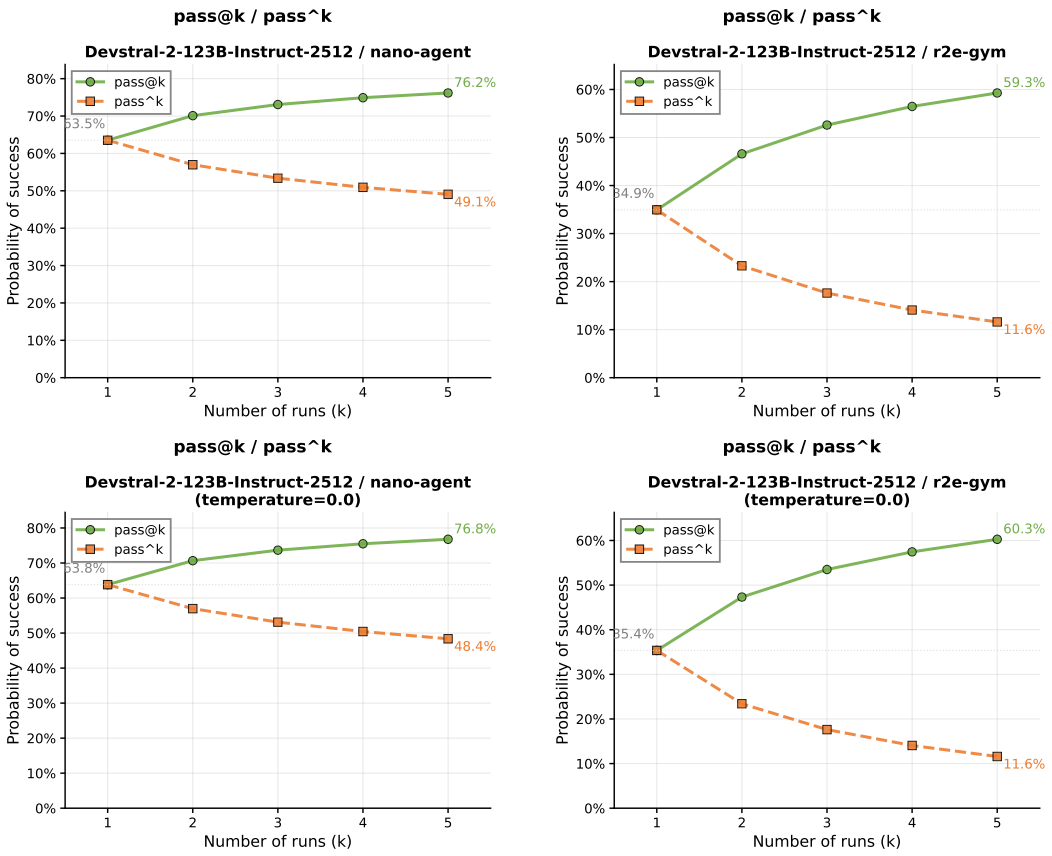


Figure 7: Additional `pass@k` and `pass^k` curves for all model-scaffold pairs (part 2/2).