How good is your harness? Statistical evaluation of coding harnesses

Jiwoo Han [1] **Yuekai Sun** [1]

Abstract

AI agent benchmarks conflate the underlying language model with the harness that wraps the model in tools, prompts, and control flow. We develop a simple statistical method to disentangle the effects of the LLM and its harness on the agent's score; *i.e.*, attribute the variation in agents' scores to their LLMs and harnesses. We use the method to evaluate harnesses and LLMs on Terminal-Bench 2.0, and our results show that

1. the harness matters as much as the LLM: the gains from simply picking the best harness are comparable to those from picking the best LLM.
2. the harness effects can be heterogeneous; *i.e.* some harnesses work better with some LLMs than with others.

Our results confirm and, more importantly, quantify practitioners' intuition on the importance of the harness and validate efforts to tailor harnesses to specific applications.

1. Introduction

Terminology AI agents consist of an LLM and a *harness*: a supporting framework of prompts, tools, and control logic that enables the agent to interact with its *in silico* environment. The terms agent and harness are sometimes used interchangeably, but we shall distinguish between them in the rest of this paper; *i.e.*, we always use the term agent to refer to a combination of an LLM with a harness.

When an agent fails/succeeds at a task, the outcome does not directly inform us whether the failure/success is due to the harness or LLM (or some interaction between the two). There is a growing body of work that examines the agent's traces/trajectories (in addition to the outcome) to obtain more detailed diagnoses of the agent's failures/successes (see Cemri et al. (2025); Zhang et al. (2025); Barke et al. (2026); Ma et al. (2024) and follow up works). Such methods typically rely on LLMs to annotate/label the agent's traces with failure analyses before clustering/taxonomizing the labels.

In this paper, we develop an alternative purely statistical approach that relies on the variation in agents' scores across different harness-LLM pairs to attribute the variation to the harness and the LLM. As we shall see, the method is simple and general, and can be applied to any benchmark with a sufficiently crossed design of agents; *i.e.* the scores of enough harness-LLM pairs are observed. Using the method, we study the agents on the Terminal-Bench 2.0 leaderboard (Merrill et al., 2026) and find that

1. the harness matters as much as the LLM: the largest harness effects and the largest LLM effects are comparable in magnitude.

2. the harness effects are sometimes heterogeneous: for example, OpenHands seems to work better with LLMs from OpenAI than LLMs from Anthropic or Google.

The first finding confirms practitioners' intuition that the harness is not a minor implementation detail, but rather a critical component of the agent's performance. That said, the sizes of the harness effects are surprising: changing the baseline harness to the best harness and changing the baseline LLM to the best LLM leads to similar score jumps!

The second finding suggests that the harness and LLM are not perfectly exchangeable components: some harnesses may be better suited for some LLMs than others. The sizes of the interaction effects are also surprising: they're comparable in size to the harness and LLM (main) effects!

2. Additive harness-LLM decomposition

Model To decompose the variation in agent scores into harness effects and LLM effects, we consider the leaderboard as a table with rows and columns corresponding to harnesses and LLMs respectively: each cell contains the score of an agent consisting of the harness and the LLM corresponding to the cell's row and column respectively.

[1]Department of Statistics, University of Michigan, Ann Arbor, MI, USA. Correspondence to: Yuekai Sun <umich.fatml@gmail.com>.

Accepted to the 1st Workshop on Combining Theory and Benchmarks, CTB@ICML 2026, Seoul, South Korea. Copyright 2026 by the author(s).

We fit an additive model to the scores, which posits that the score of each agent can be expressed as the sum of a baseline effect, a harness effect, and an LLM effect. More concretely, if $Y_{i,j}$ is the score of the agent formed by harness i and LLM j, we model its mean as

$$g(\mathbf{E}[Y_{i,j}]) = \mu + \alpha_i + \beta_j, \tag{2.1}$$

where g is an invertible link function, μ is the baseline score, α_i is the additive effect of harness i, β_j is the (additive) effect of LLM j. The choice of the link function g depends on the nature of the scores. For example, if the scores are pass rates (proportions), a common choice is the logit link, which models the log-odds of passing as a linear function of the harness and LLM effects. In the rest of this paper, we use the logit link for the pass rates on Terminal-Bench 2.0 (but the method can be adapted to leaderboards with different score types by changing the link function) (McCullagh & Nelder, 1989).

Methods For the logit link, we interpret μ as the baseline log-odds of success for the baseline agent, α_i as the average (over LLMs) log-odds change from changing the baseline harness to harness i, and β_j as the average (over harnesses) log-odds change from changing the baseline LLM to LLM j. In terms of odds ratios, we interpret $\exp(\alpha_i)$ and $\exp(\beta_j)$ are multiplicative odds ratios relative to the corresponding baseline harness/LLM. We fit (2.1) to data with off-the-shelf logistic regression software.

Data Terminal-Bench 2.0 evaluates coding agents on 89 terminal-based tasks, and the agents are given 5 trials per task. Each leaderboard entry is the average (over 89 tasks and 5 trials per task) accuracy $Y_{i,j} \in [0, 1]$ of an agent. At the time of writing, the cleaned leaderboard contains 118 entries after excluding entries listing "Multiple" as the model. After normalizing duplicate LLM names, this corresponds to 39 LLM levels, but the design is sparse and unbalanced: many harnesses are evaluated on only one or two LLMs, and many LLMs are evaluated under only one or two harnesses. We drop two harnesses whose effects are completely confounded with singleton LLMs, leaving 105 harness–LLM pairs across 28 harnesses and 26 LLMs. The resulting design remains far from fully crossed, but the Terminal-Bench 2.0 leaderboard has enough entries to identify (2.1).

Overall results We fit the additive model (2.1) to the Terminal-Bench 2.0 leaderboard. Tables 1 and 2 report the five largest fitted harness coefficients and the five largest fitted LLM coefficients from the additive GLM, and Table 5 reports the five highest-scoring LLMs when evaluated with the Terminus 2 harness. All fitted coefficients are log-odds shifts relative to the Mini-SWE-Agent (Yang et al., 2024) reference harness and Claude Sonnet 4.5 reference LLM, with standard errors shown after $\pm$. Bold coefficients have 95% confidence intervals that exclude zero.

Table 1. Largest fitted harness effects. Coefficients are log-odds shifts relative to the Mini-SWE-Agent reference harness. Bold coefficients are statistically significant (*i.e.* have 95% confidence intervals excluding 0).

Harness	Coef $\pm$ std err	Entries
Ante	$\mathbf{0.926} \pm 0.129$	1
ForgeCode	$\mathbf{0.890} \pm 0.114$	2
Capy	$\mathbf{0.656} \pm 0.127$	1
TongAgents	$\mathbf{0.653} \pm 0.110$	2
Terminus-KIRA	$\mathbf{0.588} \pm 0.109$	2

Table 2. Largest fitted LLM effects. Coefficients are log-odds shifts relative to the Claude Sonnet 4.5 reference LLM. Bold coefficients are statistically significant (*i.e.* have 95% confidence intervals excluding 0).

LLM	Coef $\pm$ std err	Entries
GPT-5.3-Codex	$\mathbf{1.071} \pm 0.076$	7
Gemini 3.1 Pro	$\mathbf{0.928} \pm 0.111$	3
Claude Opus 4.6	$\mathbf{0.835} \pm 0.062$	10
GPT-5.2	$\mathbf{0.561} \pm 0.075$	4
Claude Opus 4.5	$\mathbf{0.506} \pm 0.059$	8

The spread of the top model effects and the top harness effects is similar. The estimated model effect from Claude Sonnet 4.5 to GPT-5.3-Codex is about 1.07 logits, while the estimated harness effect from Mini-SWE-Agent to ForgeCode is about 0.89 logits. This comparison is the central empirical result: on this leaderboard, changing the harness can move accuracy by an amount comparable to changing the model.

Ante has the largest estimated harness effect, while ForgeCode is the highest-ranked harness with more than one leaderboard entry, making it the most robustly strong harness among the top point estimates. Several other high-ranking harnesses have only one observed model pairing, so their coefficients are especially sensitive to the unbalanced design.

Per-category results The Terminal-Bench 2.0 tasks are broken down into seven task categories (data science, debugging, file operations, scientific computing, security, software engineering, and system administration). For each task category, imagine a leaderboard in the same shape as the overall leaderboard but with the entries computed as the average accuracy across tasks in that category. We can fit the same additive model (2.1) to each category-specific leaderboard to obtain category-specific harness and model effects.

We fit within each of the 7 task categories containing at least five tasks by computing each pair's average accuracy across

tasks in that category. Table 3 shows the top three harness coefficients by category, with standard errors shown after $\pm$. ForgeCode ranks first in three of the six non-debugging categories. The exceptions are data science and scientific computing, where Ante has the largest estimated effect, and security, where SageAgent leads. A plausible explanation for SageAgent's strong security performance is that it includes security-oriented toolkits and workflows (Li et al., 2026). This suggests that many strong harnesses are broadly useful, but harness design can also confer category-specific strengths.

3. Saturated harness-LLM decomposition

Implicitly, (2.1) assumes that the harness and LLM effects are separable on the log-odds scale; *i.e.* there is no interaction between the harness and LLM effects. We test this (implicit) assumption by fitting an interaction model.

Model To probe whether certain harness–LLM pairs exhibit synergy beyond their additive main effects, we extend (2.1) with an interaction term:

$$g(\mathbf{E}[Y_{i,j}]) = \mu + \alpha_i + \beta_j + \gamma_{i,j}, \tag{3.1}$$

where γ_{ij} is the interaction effect of harness i and LLM j. As in section 2, we take g to be the logit link. The coefficient γ_{ij} measures the pair-specific deviation from the additive main-effects prediction: a nonzero value indicates that harness i pairs well or poorly with LLM j (depending on its sign). A distinction between (2.1) and (3.1) is that (3.1) is only identifiable when the leaderboard is completely observed; this is not the case with (2.1) (we leveraged this fact to fit (2.1) in section 2 even though the leaderboard is missing many entries).

Data For the interaction model, we use the largest fully observed submatrix of the leaderboard of Terminal-Bench 2.0: the harnesses Mini-SWE-Agent, OpenHands, and Terminus 2 crossed with nine LLMs evaluated under all three harnesses, yielding 27 agents. We restrict the analysis to a fully observed block because the interaction parameters are identified through comparison among cells in the crossed harness–LLM design. As before, each agent is evaluated on 89 tasks with 5 trials per task. Rather than collapsing the trials into a single averaged accuracy per agent, we compute the per-trial overall accuracies $Y_{i,j} \in [0, 1]$ from the task-level trial data, yielding 135 samples. As in additive model analysis, Mini-SWE-Agent and Claude Sonnet 4.5 serve as references, interaction terms involving either reference level are zero by construction.

Overall results We fit the interaction model (3.1) to the largest fully observed subset of the Terminal-Bench 2.0 Leaderboard. Table 4 reports the fitted interaction coefficients γ_{ij} for OpenHands and Terminus 2 across nine LLMs, with standard errors shown after $\pm$. All coefficients are deviations from the main effects prediction on the log-odds scale. Mini-SWE-Agent is omitted because it is the reference harness, and Claude Sonnet 4.5 has zero interaction by construction as the reference model.

OpenHands (Wang et al., 2025) shows the largest interaction pattern. Its GPT-5 interaction is significantly positive, while its Claude Haiku 4.5 and Gemini 2.5 Pro interactions are significantly negative. Its positive interactions on GPT-5-Mini and GPT-5-Nano point in the same direction, but their intervals include zero. These interaction terms are comparable in magnitude to several main effects in Tables 1 and 2, indicating that additive decomposition is a useful first-order summary but does not fully describe every harness-model pairing. We speculate that OpenHands interacts positively with OpenAI models because it is built on the OpenAI Agents SDK, which uses JSON (instead of XML) structured outputs, and OpenAI's models anecdotally prefer JSON structured outputs (while Anthropic's models prefer XML structured outputs).

On the other hand, Terminus 2, the default Terminal-Bench 2.0 harness, shows smaller deviations overall, and all of its non-reference interaction intervals include zero. Terminus 2's bash-only design likely explains this: unlike OpenHands, it does not rely on model-specific tool-calling conventions, so its performance is less sensitive to which LLM is paired with (though this explanation is only suggestive).

4. Summary and Discussion

The additive model (2.1) provides a practical way to quantify the value-add of a harness to LLM performance on agentic tasks. On Terminal-Bench 2.0, we see that the estimated variation attributable to harness choice is of the same order as the variation attributable to model choice. Our findings have consequences for both benchmark interpretation and harness development/engineering. For example, it is common in model release reports to compare the new LLM's scores on agentic benchmarks with those of competitor LLMs (*e.g.* DeepSeek-AI (2024); Kimi Team (2025)). Such comparisons are misleading because they fail to adjust for harness effects, and they risk overstating/understating a model's ability depending on the harness used for evaluation. On the other hand, our findings confirm reports that suggest it is possible to gain substantial accuracy by improving the harness while keeping the underlying LLM fixed (Lee et al., 2026; Pryzant et al., 2023; Romera-Paredes et al., 2024; Novikov et al., 2025; Lee et al., 2025; Agrawal et al., 2025). On Terminal-Bench 2.0, the gains from changing to the best harness are comparable to those from changing to the best LLM.

Table 3. Top fitted harness effects by task category. The debugging category is omitted from this table because debugging tasks exhibit complete or near-complete separation (some agents solve all debugging tasks). Bold coefficients are statistically significant (*i.e.* have 95% confidence intervals excluding 0).

Category	Rank 1 (coef $\pm$ std err)	Rank 2	Rank 3
Data science (8)	Ante ($\mathbf{1.395} \pm 0.425$)	ForgeCode ($\mathbf{1.330} \pm 0.367$)	Simple Codex ($\mathbf{1.113} \pm 0.420$)
File operations (5)	ForgeCode ($\mathbf{2.121} \pm 0.494$)	TongAgents ($\mathbf{1.679} \pm 0.467$)	Capy ($\mathbf{1.140} \pm 0.502$)
Scientific computing (8)	Ante ($\mathbf{1.310} \pm 0.425$)	Simple Codex (0.848 ± 0.448)	OpenCode (0.828 ± 0.757)
Security (8)	SageAgent ($\mathbf{1.161} \pm 0.383$)	Terminus-KIRA ($\mathbf{1.083} \pm 0.401$)	Codex CLI ($\mathbf{1.062} \pm 0.213$)
Software engineering (26)	ForgeCode ($\mathbf{1.101} \pm 0.226$)	Ante ($\mathbf{1.065} \pm 0.238$)	Terminus-KIRA ($\mathbf{0.793} \pm 0.215$)
System administration (9)	ForgeCode ($\mathbf{0.903} \pm 0.374$)	CAMEL-AI ($\mathbf{0.775} \pm 0.385$)	Capy (0.723 ± 0.422)

Table 4. Interaction coefficients in the complete 3 by 9 block. Values are deviations from the additive prediction on the log-odds scale. Bold coefficients have 95% confidence intervals excluding zero.

Model	OpenHands	Terminus 2
Claude Haiku 4.5	$\mathbf{-0.954} \pm 0.219$	-0.087 ± 0.201
Claude Opus 4.1	0.079 ± 0.195	0.103 ± 0.195
Claude Sonnet 4.5	0.000 ± 0.000	0.000 ± 0.000
GPT-5	$\mathbf{0.422} \pm 0.194$	0.048 ± 0.196
GPT-5-Mini	0.370 ± 0.206	0.096 ± 0.209
GPT-5-Nano	0.389 ± 0.280	0.127 ± 0.290
Gemini 2.5 Flash	-0.044 ± 0.225	-0.026 ± 0.224
Gemini 2.5 Pro	$\mathbf{-0.585} \pm 0.216$	0.296 ± 0.201
Grok 4	0.100 ± 0.204	-0.122 ± 0.208

That said, the interaction analysis shows that the simple additive model can be a useful first-order approximate, but harnesses and models are not perfectly exchangeable components. The benefits of more full-featured harnesses may (unintentionally) depend on the underlying LLM due to seemingly unimportant design choices (*e.g.* using JSON instead of XML structured outputs), but the benefits of basic bash-only harnesses seem to be more model agnostic. This justifies emerging harness engineering efforts that develop harnesses to make up for deficiencies in the tool calling capabilities of LLMs. For example, ForgeCode implements model-specific fixes—schema field ordering, flatter schemas, explicit truncation notices, and enforced reviewer-mode verification—brought both GPT-5.4 and Opus 4.6 to 81.8% on Terminal-Bench 2.0 (ForgeCode, 2026).

Last (but not least), this paper focuses on harness evaluation, so we spent most of the paper interpreting the harness effects, but the LLM effects are also revealing. Terminal-Bench 2.0 evaluates LLMs together with the Terminus 2 harness (see Table 5) but Terminus 2 is a very basic harness, so the scores may not reflect their abilities to take advantage of the features of more advanced harnesses. On the other hand, the LLM effects in Table 2 do depend on the LLMs' scores on more advanced harnesses, and we see some changes in LLM rankings between Table 2 and Table 5.

Table 5. Top LLMs evaluated with the Terminus 2 harness on Terminal-Bench 2.0. The right-most column is the rank change relative to the additive LLM-effect ranking; positive values mean the LLM ranks higher under Terminus 2 than by additive LLM effect.

Rank	LLM	Score	Δ
1	GPT-5.3-Codex	$64.7\% \pm 2.7\%$	0
2	Claude Opus 4.6	$62.9\% \pm 2.7\%$	+1
3	Claude Opus 4.5	$57.8\% \pm 2.5\%$	+2
4	Gemini 3 Pro	$56.9\% \pm 2.5\%$	+2
5	GPT-5.2	$54.0\% \pm 2.9\%$	-1

This suggests that the scores with the Terminus 2 harness in Table 5 may not reflect the more advanced capabilities of LLMs.

References

Agrawal, L. A., Tan, S., Soylu, D., Ziems, N., Khare, R., Opsahl-Ong, K., Singhvi, A., Shandilya, H., Ryan, M. J., Jiang, M., Potts, C., Sen, K., Dimakis, A. G., Stoica, I., Klein, D., Zaharia, M., and Khattab, O. GEPA: Reflective prompt evolution can outperform reinforcement learning, 2025. URL https://arxiv.org/abs/2507.19457.

Barke, S., Goyal, A., Khare, A., Singh, A., Nath, S., and Bansal, C. Agentrx: Diagnosing ai agent failures from execution trajectories, 2026. URL https://arxiv.org/abs/2602.02475.

Cemri, M., Pan, M. Z., Yang, S., Agrawal, L. A., Chopra, B., Tiwari, R., Keutzer, K., Parameswaran, A., Klein, D., Ramchandran, K., Zaharia, M., Gonzalez, J. E., and Stoica, I. Why do multi-agent llm systems fail?, 2025. URL https://arxiv.org/abs/2503.13657.

DeepSeek-AI. Deepseek-v3 technical report, 2024. URL https://arxiv.org/abs/2412.19437.

ForgeCode. Benchmarks don't matter – until they do (part 2). https://forgecode.dev/blog/

gpt-5-4-agent-improvements/, March 2026. Accessed 2026-05-02.

Kimi Team. Kimi k2: Open agentic intelligence, 2025. URL https://arxiv.org/abs/2507.20534.

Lee, Y., Boen, J., and Finn, C. Feedback descent: Open-ended text optimization via pairwise comparison, 2025. URL https://arxiv.org/abs/2511.07919.

Lee, Y., Nair, R., Zhang, Q., Lee, K., Khattab, O., and Finn, C. Meta-harness: End-to-end optimization of model harnesses, 2026. URL https://arxiv.org/abs/2603.28052.

Li, H., Wang, Z., Dai, Q., Nie, Y., Peng, J., Liu, R., Zhang, J., Zhu, K., He, J., Wang, L., et al. Opensage: Self-programming agent generation engine. *arXiv preprint arXiv:2602.16891*, 2026.

Ma, C., Zhang, J., Zhu, Z., Yang, C., Yang, Y., Jin, Y., Lan, Z., Kong, L., and He, J. Agentboard: An analytical evaluation board of multi-turn llm agents. *Advances in neural information processing systems*, 37:74325–74362, 2024.

McCullagh, P. and Nelder, J. A. *Generalized Linear Models*. Chapman and Hall, London, 2 edition, 1989.

Merrill, M. A., Shaw, A. G., Carlini, N., et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026. URL https://arxiv.org/abs/2601.11868.

Novikov, A., Vu, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., Shirobokov, S., Kozlovskii, B., Ruiz, F. J. R., Mehrabian, A., Kumar, M. P., See, A., Chaudhuri, S., Holland, G., Davies, A., Nowozin, S., Kohli, P., and Balog, M. Alphaevolve: A coding agent for scientific and algorithmic discovery, 2025. URL https://arxiv.org/abs/2506.13131.

Pryzant, R., Iter, D., Li, J., Lee, Y. T., Zhu, C., and Zeng, M. Automatic prompt optimization with "gradient descent" and beam search, 2023. URL https://arxiv.org/abs/2305.03495.

Romera-Paredes, B., Barekatain, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., Ruiz, F. J. R., Ellenberg, J. S., Wang, P., Fawzi, O., Kohli, P., and Fawzi, A. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://doi.org/10.1038/s41586-023-06924-6.

Wang, X., Li, B., Song, Y., Xu, F. F., Tang, X., Zhuge, M., Pan, J., Song, Y., Li, B., Singh, J., et al. Openhands: An open platform for ai software developers as generalist agents. In *International Conference on Learning Representations*, volume 2025, pp. 65882–65919, 2025.

Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K. R., and Press, O. 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.

Zhang, S., Yin, M., Zhang, J., Liu, J., Han, Z., Zhang, J., Li, B., Wang, C., Wang, H., Chen, Y., and Wu, Q. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems, 2025. URL https://arxiv.org/abs/2505.00212.