

EarnBench: A Controlled Paper-Trading Benchmark for Language-Model Agents on Live On-Chain Markets

EarnBench

Green Wick · earnbench.wick.pics

Working draft v0.2 · 24 September 2026 · results pending

Abstract

We describe the method of EarnBench, a benchmark that asks which combinations of language model, agent harness, prompt and tools make profitable trading decisions under realistic execution. Agents trade a fixed paper stake on a live blockchain market; every order is filled by an independent referee at a live multi-source quote and, since this draft, is re-checked by simulating the route's own transaction so that token taxes and failing transactions are charged as they would be on chain. Before a model trades it must pass an entrance exam with no errors. Results are scored as the liquidation value of the book in the starting currency and are only compared across agents that traded the same window, against a random-trading baseline and a measured noise floor. This draft fixes the design, the variables and the known limitations *before* results are reported, so that later changes to the method are visible as dated revisions rather than silent re-interpretation.

Keywords: LLM agents, trading, benchmark methodology, paper trading, execution realism, pre-registration.

1 Introduction

Public claims that an AI system “can trade” are rarely comparable: they differ in market, capital, costs, fill assumptions and evaluation window, and they are usually reported after the fact. Benchmarks in other domains became useful once the task, the scoring and the environment were fixed and published in advance [2, 3]. EarnBench applies that discipline to a narrow, measurable question: *given the same market, the same starting stake and the same execution rules, does this agent stack end with more than it started with, by more than chance?* Trading is done on paper so that many stacks can be tried at no financial risk, but every price is live, so the market cannot be replayed or memorised.

2 Design

2.1 The unit under test

The unit is a *stack*: model $\times$ harness $\times$ prompt $\times$ tools. A harness is a versioned manifest that fixes the tool list, the step and output limits and the provider; a prompt is stored verbatim. Both are identified by SHA-256 hashes recorded with every run, so any result can be re-run on an identical stack and any change produces a new identity rather than an edited one.

2.2 Market and referee

Agents trade on Robinhood Chain (chain id 4663), an EVM network with Uniswap-style pools [1]. Each run starts with 0.036 ETH ($\approx$ US\$100). An agent is woken every 10 minutes, receives a data bundle and its own book, and may submit orders (buy, sell, open or close a concentrated-liquidity position). It never holds balances: a separate referee process does. For each order the referee requests a live board of routes from several aggregators and fills at the best row net of gas, refusing boards older than 30 s, buys whose honeypot check is not clear, and trades whose gas exceeds their value. The chosen route's own transaction is then simulated from the paper wallet on the latest block (`eth_simulatev1` with transfer tracing [4]; the wallet's balance slot is found through the node's access list [5]), and the agent is credited with what actually arrives, not with what the route reports. A transaction that reverts is not filled. Two ledgers are kept from the same boards: the *headline* ledger counts only rows proven by simulation; a second ledger counts any row. Holdings are marked every 30 minutes by a full-size live sell quote, and the run ends with a full live liquidation.

2.3 Protocol

1. **Entrance exam.** Three fixed trading turns captured from a real run, played under the exact limits of a live turn. A model passes only if every check holds on every turn: the provider answers; the answer arrives within the time, step and output limits; it is exactly one valid decision object; the model uses its tools at least once and every tool call is well-formed; it names the inputs it relied on; every address it orders came from its data; and every order is affordable against the book it was shown. A pass belongs to one model on one harness identity.

2. **Plan turn.** Before its window opens, the stack gets one long research turn (up to 20 minutes, same tools, a higher quote allowance, no book yet) and writes a trading plan. The plan is frozen into the run's recorded prompt and published with it, so every later trade can be judged against it. The window clock starts only after the plan is written, so preparation never uses trading time.
3. **Solo shakedown.** One 1-hour run per passing stack, to show it can trade end-to-end. Not used for ranking.
4. **Noise floor.** Four copies of one stack trade the same window. The spread of their returns estimates how much chance alone moves a score.
5. **Comparison.** Stacks trade the *same* window side by side, with a random trader on the same cadence and order sizes as a baseline.
6. **Evolution arena.** A later format in which the weakest agent is periodically removed and the strongest copied with one deliberate change; described separately.

3 Variables

Table 1: What is varied, what is measured, and what is held fixed.

Varied (the question)	Measured (the answer)	Held fixed (the control)
Model and its reasoning effort	Liquidation return, headline ledger, in ETH	Chain, starting stake (0.036 ETH)
Harness: tools, step and output limits	Entrance-exam checks passed	Wake cadence (10 min), window per format
Prompt and prompting framework	Fills, refusals and their reasons	Fill rule, quote sources, simulation of every fill
Tool and data access	Costs paid: gas, pool fees, tax, slippage	Marking rule (30 min, full-size sell)
(Later) evolution settings	Tokens used and wall time per turn	Starting data bundle layout per harness

4 Scoring and inference

The score of a run is the percentage change of the headline ledger's liquidation value, measured in ETH. Because every agent in a window experiences the same ETH price move, rankings within a window are identical in ETH or in dollars; the currency is a presentation choice. Holding ETH scores exactly zero and is therefore the natural floor. A difference between two stacks is reported as a finding only when (a) both traded the same window, (b) it exceeds the

measured noise floor for that window length (two standard errors of the copies' spread), and (c) both beat the random trader. Every score is published with its n ; a single 1-hour run is treated as an anecdote, not a result. With many stacks tested, some will look good by chance [6, 7]; the number of stacks tried is therefore published beside every leaderboard.

Every run is also reported *after compute*: the trading result minus what the model's own calls would cost at the paid list price for that model (nothing is charged during the benchmark, which uses free models or a subscription). Because profit and compute are in the same unit, this yields a single, sharper question than raw return: at what stake does a model's trading cover the cost of running it? Since compute per decision is roughly fixed while profit scales with the stake until liquidity binds, each stack has a break-even range of stakes, possibly empty.

5 Threats to validity

1. **Execution delay.** Fills occur at the quote taken when the order arrives; a real transaction lands seconds later. In an earlier live system of ours, live trades lagged their paper twins materially on identical signals, and the gap was execution. Not yet modelled; a delayed re-quote with a slippage bound is planned.
2. **Own price impact.** A paper buy does not move the pool, so repeated buys into one thin pool look cheaper than they are. A single round trip is approximately unaffected.
3. **No orders between turns.** Agents cannot place stop or limit orders; they act only when woken.
4. **Honeypot gate.** The quote engine's honeypot check refuses some tokens as unknown; those tokens are untradeable here even where a real trader could buy them.
5. **Simulation coverage.** A fill that cannot be simulated (lane outage, unusual token storage) is refused on the headline ledger for buys and tagged for sells, so that an exit is never trapped by our own infrastructure.
6. **Model drift and availability.** Free hosted models change behind a stable name and are sometimes unavailable; exam passes are dated, and an outage is recorded as “not sat”, never as a failure.
7. **Regime and sample size.** Short windows in one market regime generalise poorly; sequential solo runs face different markets and are not compared.

8. **Exam representativeness.** The three exam turns come from one early run; a model can pass the exam and still trade poorly. The exam tests ability to play, not skill.
9. **Decision latency and turn order.** Fills are priced when an order arrives, so a slower model trades at later prices. We count that as part of the model's result, as it would be in real trading, and publish each model's seconds per turn. What is not the model's doing is controlled: when several agents share a window their turn order rotates each tick, provider rate-limit waits are recorded, and a turn a provider refuses is labelled unavailable rather than counted against the model.
10. **Liquidity provision.** Paper LP fees are the position's true share of real swaps in range and impermanent loss follows the real price path, but a paper position does not attract or deter flow.

6 Status

At this draft, ten hosted chat models have sat the entrance exam. Two have passed, and both only on the harness whose output limit is 16,000 tokens rather than 6,000: reasoning models spend hidden reasoning inside that limit, and three models ran out of it before answering. The other common failures are answers that are not a single decision object and provider outages, which are recorded as not sat. Live counts are on the entrance page, the parameters on /param and the rules with their enforcement status on /rules. No performance result is reported in this version.

References

- [1] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, D. Robinson. *Uniswap v3 Core*. Whitepaper, 2021.
- [2] P. Liang et al. *Holistic Evaluation of Language Models*. arXiv:2211.09110, 2022.
- [3] C. E. Jimenez et al. *SWE-bench: Can Language Models Resolve Real-World GitHub Issues?* arXiv:2310.06770, 2023.
- [4] Ethereum Execution APIs. *eth_simulatev1 specification*. github.com/ethereum/execution-apis.
- [5] V. Buterin, M. Swende. *EIP-2930: Optional access lists*. Ethereum Improvement Proposals, 2020.
- [6] D. H. Bailey, J. Borwein, M. López de Prado, Q. J. Zhu. *The Probability of Backtest Overfitting*. Journal of Computational Finance, 2017.
- [7] C. R. Harvey, Y. Liu, H. Zhu. *...and the Cross-Section of Expected Returns*. Review of Financial Studies 29(1), 2016.

Revision log. v0.2 (2026-09-24): the plan turn before each window; results after compute; decision latency and turn order as a threat to validity. v0.1 (2026-09-24): first draft; receipt simulation of every fill introduced the same day. Changes to the method are appended here with their date and reason; earlier versions are kept.