QUANTITATIVEFINANCE-BENCH: BENCHMARKING AI AGENTS ON REAL-WORLD QUANTITATIVE FINANCE TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Evaluating language models on quantitative finance tasks is challenging because such tasks require simultaneous mathematical rigor, domain expertise, and data engineering in a single executable artifact. We introduce **QuantitativeFinance-Bench (QF-Bench)**, an execution-grounded benchmark for the computational core of professional quantitative finance. We have four main contributions. First, we curate 86 practitioner-authored tasks spanning derivatives pricing, fixed income, credit, risk management, factor research, backtesting, microstructure, and financial NLP. Each task is executed in a network-isolated Docker container with real financial data. Second, we introduce a hierarchical verification framework with more than 3,300 assertions across 86 tasks. These assertions encode domain-specific checks, including no-arbitrage bounds, put-call parity, and convergence diagnostics that are not captured by generic code-level tests. Third, we define **Finance-Zero**, a non-agentic baseline for difficulty calibration, and introduce automated error attribution for *Computation*, *Convention*, and *Mislabeling* errors. Fourth, we evaluate 38 Finance-Zero base-model configurations and 9 production CLI-agent configurations, finding that the best system achieves 61.7% pass@1 while the single-call baseline reaches 42.6%. A run passes only when all required deliverables and verifier checkpoints succeed. The benchmark and evaluation resources are available at `https://anonymous.4open.science/r/QuantitativeFinance-Bench-07BB`.

1 INTRODUCTION

Quantitative finance poses challenges not captured by existing AI benchmarks. Quantitative-finance practice extends well beyond forecasting next-day stock returns. Common tasks include pricing derivatives, calibrating stochastic models to market data, decomposing portfolio risk, bootstrapping discount curves, and checking outputs against no-arbitrage constraints (Hull & White, 1990; Bollerslev, 1986). Many of these tasks admit precise numerical checks. For example, a Black-Scholes implementation can be compared with analytical Greeks to a tolerance of 10^{-6}, while a Hull-White calibration can be evaluated against the target market caplet volatility surface. This setting differs from financial reasoning benchmarks such as FinQA (Chen et al., 2021b) and FinSheet-Bench (Ravnik et al., 2026), and from stock-prediction benchmarks such as StockBench (Chen et al., 2025b) and InvestorBench (Li et al., 2025). These benchmarks primarily evaluate document reasoning or predictive performance rather than executable numerical correctness.

Coding agents for quantitative finance. Recent coding agents achieve strong performance on software-engineering and terminal-interaction benchmarks. For example, Terminal-Bench (Merrill et al., 2026) reports that frontier models resolve 63% of its general-purpose terminal tasks. Yet a model that can write and debug Python is not the same as one that can correctly price an exotic derivative or detect look-ahead bias in a backtest. A model that performs well on software-engineering tasks can still produce financially incorrect outputs that pass generic code-quality checks because there is no general-purpose oracle for whether an option price is correct. Evaluating such systems therefore requires domain-specific numerical references and financial constraints in addition to generic code-execution checks.

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.

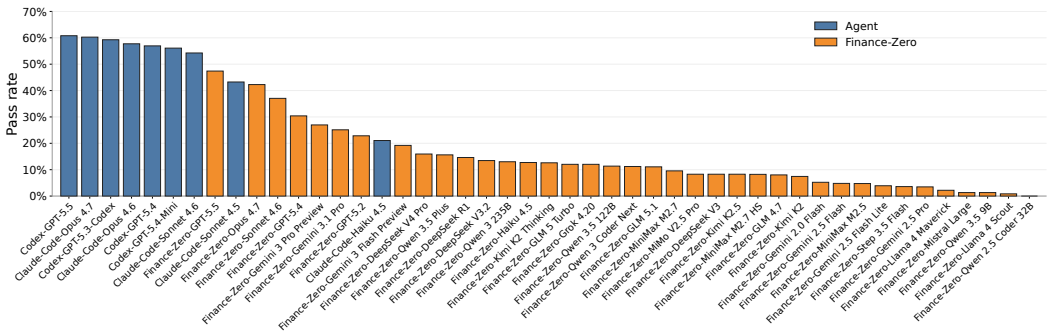


Figure 1: Results of pass@1 of all evaluated configurations on QF-Bench. Blue bars are CLI agents; orange bars are Finance-Zero single-call baselines.

What QuantitativeFinance-Bench (QF-Bench) evaluates. QF-Bench is an execution-grounded benchmark of 86 containerized quantitative-finance tasks, each authored by practitioners with quant or risk-management backgrounds. Each task is deployed in a hardened Docker sandbox with real market data (CRSP, SEC EDGAR, Binance) to simulate the "messiness" of real desk work (including missing values and holiday calendars). We employ a **hierarchical verification system** that enforces domain-specific invariants such as no-arbitrage bounds, put-call parity, and variance ordering. We evaluate 38 base-model configurations under Finance-Zero and 9 production CLI-agent configurations in single-call and agentic configurations. Figure 1 summarizes pass@1 for all 47 configurations. Our results show that strong performance on general coding tasks does not imply high performance on QF-Bench. Even the highest-performing agent configurations solve fewer than half of the Hard tasks at pass@1.

Our primary technical contributions are:

1. **Scale and Authenticity.** 86 practitioner-authored tasks span derivatives, fixed income, credit, risk management, factor research, backtesting, and microstructure. The tasks are based on practitioner-authored problems, use real rather than synthetic data, and undergo practitioner review. All tasks are executed in sandboxed Docker environments without internet access.
2. **Hierarchical Verification.** We use a five-stage pytest-based verifier that checks deliverable existence, structural validity, numerical accuracy, financial invariants, and convergence diagnostics. Task-specific numerical tolerances range from 10^{-6} to 0.05, and failed checks are attributed to *Computation*, *Convention*, and *Mislabeling* errors.
3. **Large-Scale Multi-Regime Evaluation.** We evaluate 38 Finance-Zero base-model configurations and 9 production CLI-agent configurations and report both accuracy and evaluation cost. For GPT-5.5, pass@1 increases from 42.6% in Finance-Zero to 61.7% with Codex CLI (Table 1).
4. **Difficulty Calibration.** The Finance-Zero baseline provides a reference for identifying tasks that are already solvable in a single model call and may therefore require further difficulty calibration as model performance improves.

All tasks, verifiers, and evaluation code are included in the supplementary material and will be publicly released upon publication.

2 Related Work

Coding and agent benchmarks. AI evaluation has progressed from function-level code generation (Chen et al., 2021) to repository-scale software engineering (Jimenez et al., 2024; Yang et al., 2024), terminal interaction (Merrill et al., 2026; Liu et al., 2024b), tool use (Yao et al., 2024; Patil et al., 2025), and computer-use environments (Zhou et al., 2024; Xie et al., 2024). Domain-specific benchmarks extend this paradigm to ML engineering (Chan et al., 2025) and scientific computing (Chen et al., 2025c; Tian et al., 2024). However, these benchmarks primarily evaluate code

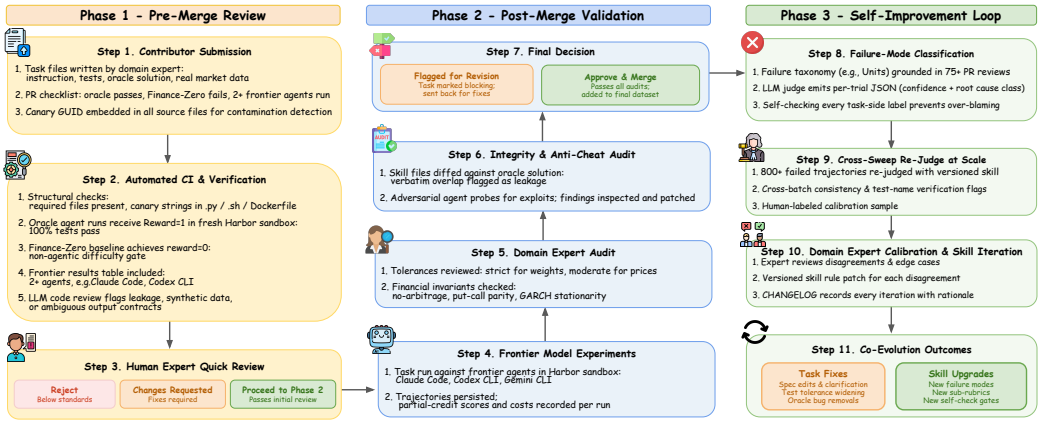


Figure 2: The QF-Bench task quality assurance pipeline. Phase 1 (left, from top to bottom) gates every task before merging via automated CI checks (oracle must pass, Finance-Zero must fail) and expert human review. Phase 2 (middle, from bottom to top) audits merged tasks through frontier model experiments, domain expert review of tolerances and financial invariants, and adversarial integrity probing, culminating in a final approve-and-merge decision. Phase 3 (right, from top to bottom) refines merged tasks through domain expert calibration and versioned task analysis skill iterations to drive co-evolution outcomes across both task fixes and skill upgrades.

execution, tool use, or task completion. They do not require agents to satisfy domain-specific financial constraints such as no-arbitrage consistency, pricing identities, or calibration accuracy. QF-Bench instead treats financial mathematics as the oracle.

Financial intelligence benchmarks. Existing financial AI benchmarks mainly evaluate document reasoning, financial NLP, trading, or portfolio management. FinQA (Chen et al., 2021b), FinanceBench (Islam et al., 2023), and FinSheet-Bench (Ravnik et al., 2026) focus on reasoning over financial documents and spreadsheets. BloombergGPT (Wu et al., 2023) and FinGPT (Yang et al., 2023) evaluate finance-domain language understanding. StockBench (Chen et al., 2025b), InvestorBench (Li et al., 2025), QuantBench (Wang et al., 2025), and LiveTradeBench (Yu et al., 2025) evaluate trading, portfolio management, or live market decision making. Related agentic benchmarks include FinToolBench (Lu et al., 2026) and FinSearchComp (Hu et al., 2025). Closest to our setting, QuantEval (Kang et al., 2026) executes model-written trading strategies in a backtest and scores them by financial performance. QF-Bench instead evaluates the correctness of the computation itself: agents work in a sandbox over multiple steps, and their outputs are checked against reference values and financial invariants.

3 QUANTITATIVEFINANCE-BENCH

3.1 TASK FORMULATION

Each QF-Bench task models a real quant desk assignment. An agent operates in a fresh Docker sandbox containing real market data, such as CRSP returns, SEC EDGAR filings, option-chain files, or yield-curve series. A practitioner-written instruction specifies what to compute, where to write the output, the required schema, and the expected numerical precision. The agent has access to a standard quantitative Python environment but not to the grader or reference solution. It must solve the task from the instruction and data alone.

After execution, outputs are evaluated using hierarchical verification covering file validity, numerical accuracy, and financial consistency. Section 3.3 provides the full evaluation procedure. Financial consistency checks include no-arbitrage bounds, put-call parity, and calibration constraints that rarely appear in general coding benchmarks. Each task includes a human-written reference solution used only to verify solvability and calibrate tolerances. The reference solution is never exposed to the

agent during execution. We build on the Harbor (Harbor, 2026) evaluation system, which isolates grading by exposing tests and reference solutions only after execution completes.

QF-Bench contains 86 tasks spanning derivatives pricing, fixed income, credit, factor research, risk management, backtesting, microstructure and execution, FX/crypto, and NLP-on-finance. Tasks are grouped into three empirical difficulty tiers based on frontier-model performance: 18 Easy, 33 Medium, and 35 Hard. The full task list appears in Appendix F.

3.2 Benchmark design

The design of QF-Bench emphasizes authenticity, verifiability, and evaluation integrity. Tasks reflect real quant workflows through real market data and practitioner-authored problems. Inputs preserve practical challenges such as missing values, corporate actions, mixed-frequency series, and tick-level noise. Tasks require applied financial analysis rather than formula recall, including option pricing, curve calibration, portfolio attribution, and risk modeling.

To ensure reproducible evaluation, every task defines an explicit output contract specifying file paths, schemas, and numerical precision. Grading is fully programmatic using deterministic Python tests, with heterogeneous tolerances calibrated to the financial context and anchored against human reference solutions.

Task quality is maintained through expert review and network-isolated evaluation. Each task is submitted through a version-controlled review process and must pass automated continuous-integration checks before human review. A primary reviewer reproduces the reference solution and audits task design and tolerances, while a secondary reviewer checks convention-sensitive choices such as sign, annualization, and day count. Reviewers additionally verify the absence of look-ahead bias and data leakage. Grading remains isolated during execution, and all source files contain unique canary strings for contamination detection (Srivastava et al., 2023). Figure 2 summarizes the full quality-assurance pipeline.

3.3 Evaluation framework

QF-Bench evaluates tasks using hierarchical verification rather than binary pass/fail alone. Each task contains progressive checkpoints covering deliverable existence, structural validity, numerical accuracy, financial consistency, and convergence diagnostics. This structure distinguishes superficial failures from substantive financial errors while providing fine-grained diagnostic signals for model analysis. The verifier additionally attributes failures to computation errors, convention mismatches, or output mislabeling. Full implementation details and verification examples are provided in Appendix B.

QF-Bench separates diagnostic scoring from headline evaluation. Hierarchical tests produce partial-credit signals, including pass rates for file validity, numerical accuracy, and financial consistency, which we use to analyze model behavior and failure modes. However, the headline metric is binary. A task is considered solved only if every checkpoint and deliverable passes.

This design reflects quantitative-finance practice, where a pricing report, risk decomposition, or trading strategy is useful only when fully correct. A sign error in a Greek or a VaR estimate computed at the wrong confidence level can be as damaging as a completely incorrect calculation. Partial-credit scores help identify near-misses and guide model development, but they do not replace the all-or-nothing binary criterion used for the headline benchmark score.

4 Experimental setup

We evaluate models under two distinct settings. *Finance-Zero* measures single-call performance, while the agentic setting allows a CLI scaffold to interact with the sandbox over multiple execution and revision steps.

4.1 Finance-Zero: a single-call difficulty floor

Finance-Zero is a non-agentic baseline. For each (model, task) pair we issue one API call with a minimal system prompt and execute the first generated Python code block once in the Docker sandbox

Table 1: Model leaderboards: pass@1/pass@3 for CLI agents (3 runs) and pass@1 for Finance-Zero (single call). All scores are binary at the round level.

Agent evaluation (CLI scaffolds, 3 runs)			**Finance-Zero (single call)**	
Agent (scaffold + model)	pass@1	pass@3	Model	pass@1
Codex CLI + GPT-5.5	**61.7%**	66.2%	GPT-5.5	**42.6%**
Claude Code + Opus 4.7	61.2%	67.1%	Claude Opus 4.7	38.5%
Codex CLI + GPT-5.3-codex	60.8%	67.5%	Claude Sonnet 4.6	32.7%
Claude Code + Opus 4.6	59.2%	65.0%	GPT-5.4	27.4%
Codex CLI + GPT-5.4	57.5%	63.8%	Gemini 3 Pro Prev.	22.1%
Codex CLI + GPT-5.4-mini	57.1%	68.8%	Gemini 3.1 Pro	20.2%
Claude Code + Sonnet 4.6	56.3%	67.1%	GPT-5.2	19.8%
Claude Code + Sonnet 4.5	46.2%	60.0%	DeepSeek V4-Pro	15.5%
Claude Code + Haiku 4.5	20.8%	31.2%	Gemini 3 Flash Prev.	14.4%

with a 120-second timeout. The model receives no retries, debugging steps, or multi-turn interaction. Tasks solved reliably by strong single-call models are flagged for review because they may provide limited discrimination among agent configurations (Appendix A). For the remaining tasks, Finance-Zero provides a reference for measuring the the gap between the two regimes, which reflects the combined effect of interaction budget, execution feedback, time limit, and scaffold in Section 4.2. We run it on 38 base models drawn from Google, OpenAI, Anthropic, Alibaba, DeepSeek, Moonshot, Xiaomi, Zhipu, MiniMax, Meta, Mistral, xAI and StepFun, with up to three rounds per model and task.

4.2 Agent evaluation: multi-turn CLI scaffolds

The agent-evaluation regime measures performance when a strong base model is allowed to behave as a coding agent. The agent can read `instruction.md`, inspect data files, execute code, observe errors, and iterate. We evaluate Claude Code and Codex CLI, each paired with several supported base models. Claude Code is evaluated with Opus 4.6, Opus 4.7, Sonnet 4.5, Sonnet 4.6, and Haiku 4.5. Codex CLI is evaluated with GPT 5.3-codex, GPT 5.4, GPT 5.4-mini, and GPT 5.5. The scaffold runs inside the same Docker sandbox as Finance-Zero, has access to the file system and shell, and can execute and re-execute Python.

Each (scaffold, model, task) configuration is run three times to account for nondeterminism in agentic execution. A small number of runs are excluded due to provider-side rate limits or environment errors.

5 Results

We evaluate 38 base models under *Finance-Zero* and 9 CLI-based agents under the multi-turn agent regime, both on the full 86-task suite. We report **pass@**k as the headline metric. It is the fraction of tasks solved in at least one of k independent rounds, where a round is counted as a pass only if every required checkpoint and deliverable succeeds (Section 3.3). This section reports headline performance, breaks results down by category and difficulty, and analyzes failure modes.

5.1 Overall performance

Table 1 reports agentic and Finance-Zero leaderboards on QF-Bench. For models appearing in both leaderboards, the CLI configuration has a higher pass@1. Because our headline metric is binary, this gain reflects additional *fully-correct* deliverables rather than partial credit. Performance varies across tasks.

We observe three main patterns. (i) Base-model choice remains a major determinant of performance within a fixed scaffold, although agentic execution can alter local rankings relative to Finance-Zero. (ii) Retries improve reliability but do not eliminate a substantial unsolved core, even for the highest-performing configuration (Section 5.3). (iii) Retry gains vary considerably across systems. Claude

Table 2: Agent pass@1 by difficulty tier (18 Easy / 33 Medium / 35 Hard).

Agent	Easy (18)	Medium (33)	Hard (35)
Codex CLI + GPT-5.5	91%	70%	35%
Claude Code + Opus 4.7	92%	68%	33%
Codex CLI + GPT-5.3-codex	91%	67%	34%
Claude Code + Opus 4.6	88%	64%	32%
Claude Code + Sonnet 4.6	86%	62%	28%
Claude Code + Haiku 4.5	56%	18%	5%

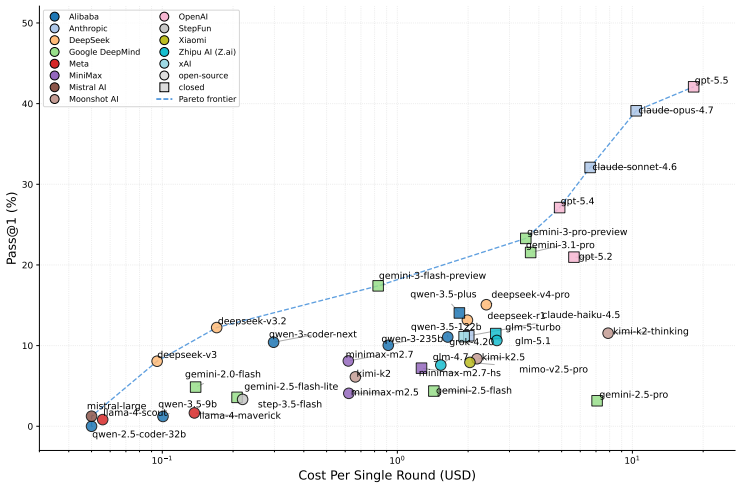


Figure 3: Cost-vs-accuracy Pareto frontier under Finance-Zero (38 base models, log cost axis). Open circles are open-weight models; filled squares are closed. The dominant configurations and their per-round cost are tabulated in Appendix E (Table 7).

Code + Sonnet 4.5 shows the largest increase from pass@1 to pass@3, while Codex CLI + GPT-5.4-mini achieves the highest pass@3. These differences suggest that independent retries can recover stochastic failures and offer an attractive cost–reliability trade-off.

Table 2 shows clear performance differences across difficulty tiers. Easy tasks have substantially higher pass rates than Hard tasks across the reported agents, while Medium tasks discriminate top from mid-tier. On Hard tasks, the highest observed pass@1 ranges from 30% to 35% among the highest-performing configurations. These are predominantly long pipelines combining numerical calibration with multi-step state construction (Hull-White swaption pricing, Dupire local-volatility calibration, OIS bootstrapping, IPCA latent factors, DCC-GARCH portfolio VaR); failure modes are attributed in Section 5.3.

5.2 Cost–accuracy trade-off

Figure 3 plots cost-vs-accuracy across the 38-model Finance-Zero pool.

The model ordering differs from that reported on general-purpose coding benchmarks. Models that perform strongly on Terminal-Bench, SWE-Bench, and HumanEval do not necessarily achieve the highest scores on QF-Bench. The benchmark additionally requires numerical robustness, convention consistency, long-horizon reasoning over multi-step calibration pipelines, and execution reliability against messy real-world data. This result is consistent with the motivation in Section 1 that quantitative-finance tasks require capabilities not fully captured by general-purpose coding benchmarks.

Closed-source frontier systems dominate the Pareto frontier. Claude and GPT configurations occupy the high-accuracy region of the plot. GPT-5.5 achieves 42.6% pass@1 under Finance-Zero and 61.7% with Codex CLI. This indicates that the benchmark remains far from saturated. The strongest open-weight entry, DeepSeek-V4-Pro, scores 15.5%, representing a $\sim$27 pp gap to GPT-5.5 under Finance-Zero (Table 1).

5.3 Failure-mode analysis

Failures on QF-Bench split into two different layers. **Layer 1** contains software-engineering failures, including runtime crashes, schema mismatches, missing output files, incorrect column names, unstable sorting, and malformed JSON. **Layer 2** contains quantitative-finance errors that are not detected by generic code-quality checks. Layer 2 further splits into two sub-layers: **Layer 2a (Theory / Academic)**, where the mathematics itself is wrong (a missing Jacobian, a wrong sign in a drift term, the wrong variant of a cited paper's formula); and **Layer 2b (Empirical / Best-Practice)**, where the mathematics is defensible in isolation but the chosen implementation path violates an industry convention that every working quant follows (e.g., PCA sign anchoring, stable sort for time-series operations).

This decomposition separates generic software-engineering failures from finance-specific theoretical and convention-related errors. The reported failure-attribution distributions differ substantially: engineering-bug attributions account for 41.0% and 6.7%, while Layer-2 attributions account for 44.7% and 76.9%. Appendix I describes differences in judging protocols and sample coverage, so these distributions should not be interpreted as a controlled estimate of the effect of agentic iteration.

5.3.1 Methodology

The taxonomy, iterative calibration, and cross-agent bias-reduction step of our labeling protocol are summarised in Figure 7.

Two-axis judgment. We evaluate every failed trial along two independent axes. The **process axis** (9 modes) classifies how the agent behaved at trajectory level, adapted from prior agent-benchmark work (Merrill et al., 2026): Execution (DisobeyTaskSpecification DTS, StepRepetition SR, UnawareOfTermination UnaT), Coherence (ReasoningActionMismatch RAM, ContextLoss CL, TaskDerailment TD), and Verification (NoOrIrrelevantVerification NoV, WeakVerification WV, PrematureTermination PT). The **content axis** (10 modes) classifies what quantitative-finance mistake the generated code or output actually contains: Model & Formula (A1 Wrong Model, A2 Missing/Extra Term, A3 Wrong Parameterization), Convention & Units (B1 Scale, B2 Sign, B3 Time/Calendar), Data & Numerics (C1 Data Fabrication, C2 Optimization Failure, C3 Statistical Variant), and Spec & Output Compliance (D1 Schema/Format). Each failed trial is labeled independently along both axes. We derive the L1/L2a/L2b label from the content and process annotations using the following mapping. D1 and the schema-subtype slice of DTS map to L1; A1/A2 and the mathematical slice of B2 map to L2a; A3.f (non-industry default), C3, and convention defaults map to L2b.

LLM-as-judge pipeline. For every failed trial we assemble a judging bundle containing the task specification, unit tests, agent trajectory, generated code, output files, and failing-test values. A constrained-JSON LLM judge then evaluates each of the 19 failure modes independently. We require each positive label to be supported by evidence from both the verifier trace and the task instruction. Specifically, the judge must identify a verifier line associated with the failure and an instruction line relevant to the violated requirement. Preflight summaries are used only as auxiliary context. A deterministic pre-flight gate short-circuits empty-stub trials ($<$500 chars of code, no output files, no numeric comparisons) with nine `match=false` labels at zero cost, reducing sweep spend by $\sim$50% with 0 false positives on the 50-trial validation set.

Bias reduction via cross-agent falsification. Single-pass LLM judging can misattribute an agent error to ambiguity in the task specification. This is particularly problematic when other agents successfully solve the same task under the same specification. We therefore apply a cross-agent verification step to trials initially classified as containing a task-level issue. The pipeline retrieves passing trials for the same task from other agent sweeps and provides their generated code to an independent judge for comparison. Passing trials provide comparison evidence for reassessing a

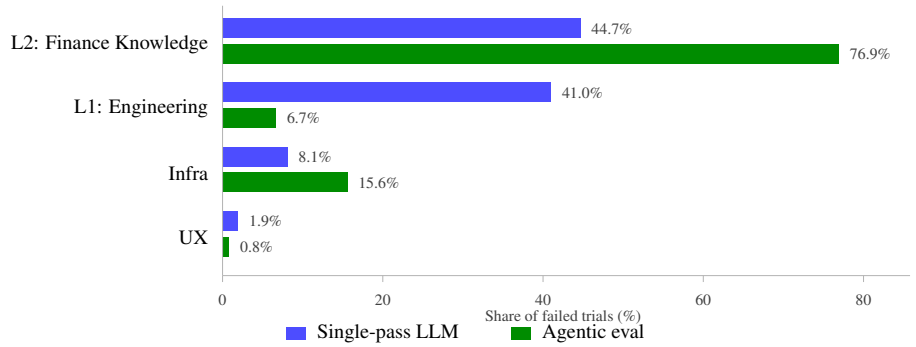


Figure 4: Knowledge-versus-execution: distribution of failure attributions across the ∼1,100-trial-per-regime failed corpus, comparing the single-pass LLM run (blue) with the agentic run (green). Layer-1 engineering-bug failures drop sixfold (41.0% → 6.7%) while Layer-2 finance-knowledge gaps grow in share (44.7% → 76.9%). Infrastructure failures (8.1% → 15.6%) and UX failures (1.9% → 0.8%) are orthogonal to both layers.

task-side verdict, while possible ambiguity and oracle-visible shortcuts still require review. Each task-level label is additionally checked against six criteria covering practitioner convention, citation specificity, mathematical equivalence, literal specification requirements, consistency across trials, and judge confidence. Labels with confidence below 0.8 do not pass this check.

Validation. We calibrated the content-axis taxonomy against independent human labels on a 20-trial stratified sample across 13 (harness × model) combinations. The overall Cohen's κ is 0.89. Six of the nine content modes shown in Figure 7 reached perfect agreement, while the remaining modes have κ values between 0.64 and 0.77. The taxonomy was refined over seven calibration rounds grounded in 75+ task reviews; Figure 7 shows the five rounds following adoption of the strict-JSON protocol.

5.3.2 Software-engineering failures

Layer 1 failures are highly concentrated on the process axis: DTS, RAM, and NoV account for 91.2% of all failed trajectories, with DTS alone contributing 50.0%. Many DTS cases are *not* failures of quantitative reasoning: agents often complete the core computation but violate the verifier-facing contract through wrong field names, incorrect output paths, missing intermediates, or unit convention errors. Schema and path violations account for roughly 40% of named DTS subtypes. These failures are often amenable to iterative correction because stack traces, deprecation warnings, and missing-column errors provide direct execution feedback that the agent can use to revise its implementation. The engineering-attribution comparison in Fig. 4 is descriptive; differences in judging protocols and sample coverage prevent attributing the full shift to iteration alone.

Scaffold differences. The two agent frameworks exhibit distinct Layer-1 signatures. `Claude Code` produces false-success claims at 2.7× the rate of `Codex CLI` (RAM: 15.2% vs. 5.6%), while `Codex CLI` fails to verify outputs at 1.8× the rate (NoV: 10.9% vs. 6.1%). The combined DTS+RAM burden is also higher for `Claude Code` (40.5% vs. 26.2%). The two scaffolds differ by approximately 11 percentage points in pass rate. These results indicate different failure profiles for the two scaffolds. `Claude Code` exhibits more coherence-related failures, whereas `Codex CLI` exhibits more verification-related failures.

Model-family differences. Within the Claude family, RAM declines sharply from 31.5% (Haiku 4.5) to 4.6% (Opus 4.7) as overall task performance increases. DTS narrows only modestly (30.7% → 20.7%), remaining the dominant failure mode even for the strongest Claude. The GPT family shows a qualitatively different profile: RAM is low and roughly flat across all four models (3.1–7.2%), comparable to the best Claude models, but NoV is consistently elevated (8.8–12.0%) and does not improve with scale (GPT-5.5 achieves the highest pass rate in the GPT family yet carries the highest NoV at 12.0%).

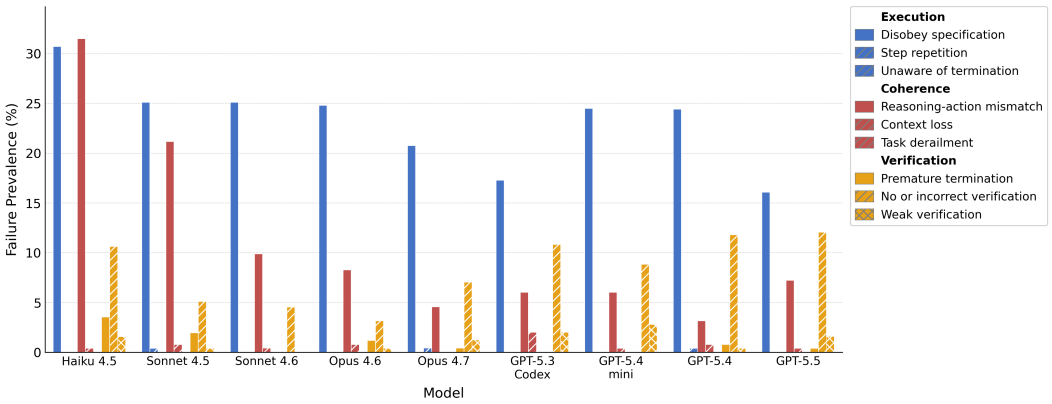


Figure 5: Layer-1 (process-axis) failure prevalence by model, expressed as a percentage of all runs. Bars are grouped by model and colored by failure class: Execution (blue), Coherence (red), and Verification (gold). Claude models show coherence failures (RAM) that decay sharply with capability (31.5% Haiku $\rightarrow$ 4.6% Opus 4.7), while GPT models carry persistently elevated verification failures (NoV 8.8–12.0%) that do *not* improve with scale.

5.3.3 Quantitative-finance failures

Layer 2 is where QF-Bench becomes diagnostic in a way general coding benchmarks cannot be. The content axis separates Layer 2 into two sub-layers: **Layer 2a (Theory / Academic)**, where the math itself is wrong, and **Layer 2b (Empirical / Best-Practice)**, where the math is locally defensible but violates practitioner convention. The content-attribution comparison is shown in Fig. 4; judging protocols and sample coverage are described in Appendix I.

The key Layer-2 observation is that *finance errors can persist through iteration*. The within-trial analysis in Appendix I includes both successful corrections and persistent errors; iteration does not reliably resolve every finance mistake. Every trial of the regime-risk-parity CVaR task adopts the wrong absorption-ratio formula; every non-crashing trial of the regime-CTA volatility-target task picks the wrong composite-volatility aggregator; the `lookback-options` task uniformly bifurcates a unified canonical formula (one trial spent 302 KB of reasoning across 74 "reconsider" iterations before committing to the wrong bifurcation, while its Finance-Zero counterpart picked the correct unified formula in a single pass).

Why finance errors can persist through iteration. When an agent makes an engineering mistake (such as a wrong column name, a deprecated API, a syntax error), the sandbox produces an error message pointing to the fault, and the agent converges by iterating on that signal. Finance-specific errors may not produce an execution failure. An incorrect formula variant, omitted Jacobian, or inappropriate day-count convention can still yield numerically plausible outputs. Without an explicit financial correctness signal, iterative execution does not necessarily correct these errors and may preserve an incorrect modeling assumption (as the lookback-options example demonstrates). Domain-specific references and verification may help address these errors.

6 Conclusions and Future Work

QF-Bench evaluates agentic coding on quantitative-finance tasks that combine executable implementation with domain-specific numerical and financial constraints. Across the evaluated systems, the highest pass@3 remains below 70%. Our results suggest three directions for future work: including finance-aware retrieval of canonical formulas, industry conventions, and textbook references during execution; using the structured failure taxonomy as a fine-grained training signal; and incorporating curated quantitative-finance best practices, such as PCA sign anchoring, stable sorting for time series, and Newey–West bandwidth conventions, into pre-training or fine-tuning.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and internal consistency checks of the manuscript. The research also evaluates language-model agents and uses LLM-based error attribution, as described in Sections 4.2 and 5.3.1.

REFERENCES

Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327, 1986.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, et al. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *ICLR*, 2025.

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

Zhiyu Chen, Wenhu Chen, Charese Smiley, et al. FinQA: A dataset of numerical reasoning over financial data. In *EMNLP*, 2021.

Yanxu Chen, et al. StockBench: Can LLM agents trade stocks profitably in real-world markets? 2025.

Ziru Chen, Shijie Chen, et al. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *ICLR*, 2025.

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments. 2026. URL `https://github.com/harbor-framework/harbor`.

Liang Hu, et al. FinSearchComp: Towards a realistic, expert-level evaluation of financial search and reasoning. *arXiv preprint arXiv:2509.13160*, 2025.

John Hull and Alan White. Pricing interest-rate-derivative securities. *The Review of Financial Studies*, 3(4):573–592, 1990.

Pranab Islam, et al. FinanceBench: A new benchmark for financial question answering. *arXiv preprint arXiv:2311.11944*, 2023.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-Bench: Can language models resolve real-world GitHub issues? In *ICLR*, 2024.

Zhaolu Kang, Junhao Gong, Wenqing Hu, Shuo Yin, et al. QuantEval: A benchmark for financial quantitative tasks in LLMs. *arXiv preprint arXiv:2601.08689*, 2026.

John L. Kelly. A new interpretation of information rate. *Bell System Technical Journal*, 35(4):917–926, 1956.

Haohang Li, et al. InvestorBench: A benchmark for financial decision-making tasks with LLM-based agents. In *ACL*, 2025.

Qianqian Xie, et al. FinBen: A holistic financial benchmark for large language models. In *NeurIPS Datasets & Benchmarks*, 2024.

Xiao Liu, et al. AgentBench: Evaluating LLMs as agents. In *ICLR*, 2024.

Jiaxuan Lu, Kong Wang, Yemin Wang, Qingmei Tang, et al. FinToolBench: Evaluating LLM agents for real-world financial tool use. *arXiv preprint arXiv:2603.08262*, 2026.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, et al. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *arXiv preprint arXiv:2601.11868*, 2026.

Shishir G. Patil, et al. Berkeley Function Calling Leaderboard. 2025.

Jan Ravnik, Matjaž Ličen, Felix Bührmann, Bithiah Yuan, et al. FinSheet-Bench: From Simple Lookups to Complex Reasoning, Where LLMs Break on Financial Spreadsheets. *arXiv preprint arXiv:2603.07316*, 2026.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023.

Minyang Tian, et al. SciCode: A research coding benchmark curated by scientists. *arXiv preprint arXiv:2407.13168*, 2024.

Saizhuo Wang, et al. QuantBench: Benchmarking AI methods for quantitative investment. *arXiv preprint arXiv:2504.18600*, 2025.

Shijie Wu, Ozan Irsoy, Steven Lu, et al. BloombergGPT: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023.

Tianbao Xie, et al. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *NeurIPS*, 2024.

Hongyang Yang, Xiao-Yang Liu, and Christina D. Wang. FinGPT: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*, 2023.

John Yang, Carlos E. Jimenez, Alexander Wettig, et al. SWE-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*, 2024.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Haofei Yu, Fenghai Li, and Jiaxuan You. LiveTradeBench: Seeking real-world alpha with large language models. *arXiv preprint arXiv:2511.03628*, 2025.

Shuyan Zhou, Frank F. Xu, Hao Zhu, et al. WebArena: A realistic web environment for building autonomous agents. In *ICLR*, 2024.

A HARBOR TASK LIFECYCLE

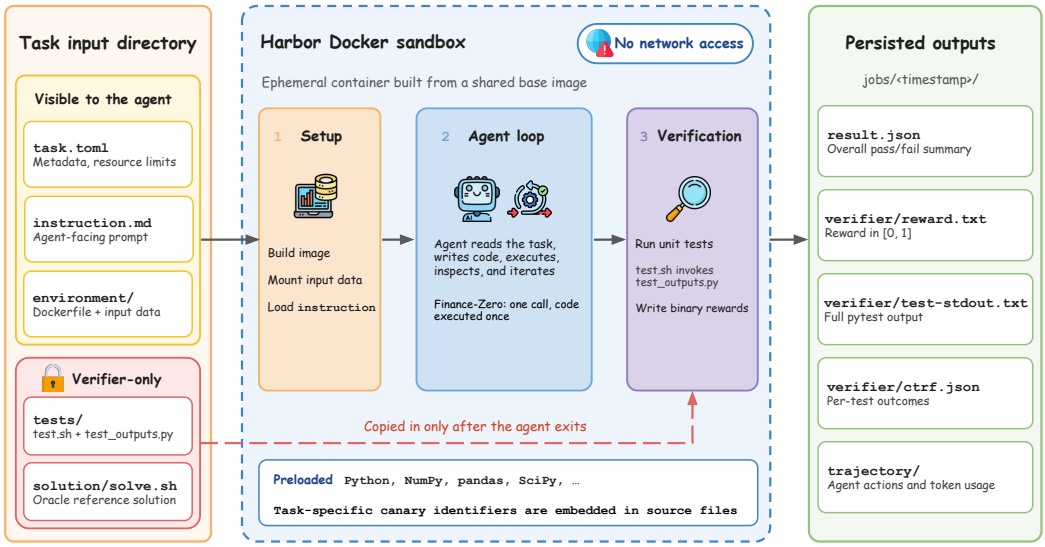


Figure 6: Task lifecycle in Harbor. A task directory (left) is loaded into an air-gapped Docker container where the agent reads `instruction.md`, writes code, and iterates. After the agent finishes, the verifier runs `test.sh` and persists structured outputs (right) including per-test results, reward, and the full agent trajectory.

B HIERARCHICAL VERIFICATION DETAILS

QF-Bench evaluates tasks using hierarchical verification pipelines rather than binary pass/fail checks alone. Each task contains 20 to 60 assertions organized into progressive checkpoints and complementary scoring tracks.

B.1 PROGRESSIVE CHECKPOINT TESTING

Test suites are organized into sequential validation stages:

1. **Deliverable existence**: verify that all required output files exist.
2. **Structural validity**: check file formats, column names, row counts, and schema compliance.
3. **Numerical accuracy**: compare outputs against oracle reference values using task-specific tolerances. Tolerances vary by metric, ranging from strict thresholds for normalization constraints to looser thresholds for noisy statistical estimates.
4. **Financial consistency**: enforce domain-specific invariants, including no-arbitrage bounds, put-call parity relationships, variance ordering constraints, and factor-loading consistency.
5. **Convergence and diagnostics**: validate numerical properties such as calibration convergence, stationarity conditions, and discretization behavior.

This layered structure distinguishes different failure modes. An agent producing valid files but incorrect numerical outputs scores higher than one that crashes entirely, while an agent satisfying core financial constraints scores higher than one violating basic financial principles.

B.2 PARTIAL-CREDIT SCORING

QF-Bench supports two complementary scoring tracks.

Track A: pytest-based tasks. For standard pytest tasks, the framework reports:

$$\texttt{tests_passed}/\texttt{tests_total},$$

yielding a natural partial-credit score.

Track B: verifier-based tasks. For verifier-based tasks, the framework computes:

$$\text{reward} = 0.5 \times \text{CC}_{\text{frac}} + 0.5 \times \text{CP}_{\text{frac}},$$

where CC_{frac} is the fraction of deliverables passing correctness checks and CP_{frac} is the fraction of checkpoints matching oracle values.

B.3 Error attribution

The verifier automatically classifies failed checkpoints into three categories:

- **Computation error**: the output is mathematically incorrect.
- **Convention error**: the output is correct up to a sign, scaling, or reporting convention.
- **Mislabeling error**: the numerical value is correct but assigned to the wrong output field.

Error attribution combines label matching, value matching, and structural matching to distinguish substantive computational failures from superficial formatting issues.

B.4 Financial verification examples

QF-Bench includes domain-specific validation rules uncommon in general-purpose benchmarks. Examples include:

- American option price $\geq$ European option price.
- Put-call parity consistency for Greeks.
- Asian option price $\leq$ corresponding European option price.
- GARCH stationarity condition:
$$\alpha + \beta < 1.$$
- Calibration RMSE thresholds for fitted models.

These checks ensure that outputs satisfy economically meaningful constraints rather than merely matching numerical targets.

C Illustrative task specifications

This appendix walks through several representative tasks in detail. These are not exhaustive (available per-task pass@1, attempt counts, and cost statistics are in Appendix F), but the examples below illustrate the kind of methodological depth, financial-invariant checking, and multi-step pipeline structure that QF-Bench tasks are written around. Table 3 summarizes the time estimates and output complexity of the early-cohort tasks profiled in the rest of this appendix.

C.1 Derivatives pricing tasks

american-option-fd-new (60/180 min). The agent implements a Crank-Nicolson finite difference scheme to price American and European puts and calls with discrete cash dividends. The grid uses $N_S = 300$ stock steps and $N_T = 600$ time steps over a domain $S \in [0, 300]$. Handling American early exercise requires solving a non-linear complementarity problem via the Projected Successive Over-Relaxation (PSOR) algorithm with convergence tolerance 10^{-8}. The agent must compute the early exercise boundary $S^*(t)$ for American puts, verify that American prices weakly dominate European prices, confirm that American calls without dividends equal European calls (within 10^{-3}), compute Greeks via central finite differences, and perform Richardson extrapolation from coarse/fine grids to estimate convergence order. Outputs include the full option value grid, exercise boundary time series, Greeks, and convergence diagnostics.

Table 3: Human-effort estimates and output complexity for selected tasks. *Expert* and *Junior* report estimated implementation time (minutes) for a domain expert and a junior quantitative practitioner, respectively; *Output files* is the number of required deliverables. These human-effort estimates are distinct from the empirical model-difficulty tiers reported in Table 8.

Task	Expert (min)	Junior (min)	Output files
fama-french-factor-model	30	90	8
momentum-backtest	30	90	4
bollinger-backtest-aapl	45	120	5
sentiment-factor-alpha	60	120	2
cta-basel-capital	60	120	2
kelly-var-sizing	55	110	2
regime-cta-vol-target	55	110	2
barrier-garch-var	55	110	2
structured-note-risk	50	100	2
american-option-fd-new	60	180	7
mc-greeks-surface (Legacy Task)	75	240	7
regime-riskparity-cvar	60	120	2
stochvol-implied-surface	60	180	10
hull-white-swaption	75	240	7
mc-greek-surface-1	75	240	7

mc-greek-surface-1 (75/240 min). The agent computes a full surface of option Greeks (Δ, Γ, $\mathcal{V}$, Θ, ρ) for European and Asian options using three independent Monte Carlo methods: finite differences (central bumping with specified step sizes), pathwise derivatives (Broadie & Glasserman IPA, where Γ is unavailable due to the non-differentiable indicator function), and the likelihood ratio method (score-function weighting, where Θ is unavailable). The agent must use 500,000 paths with seed 42, validate against Black-Scholes analytical Greeks, verify put-call parity relationships ($\Delta_C - \Delta_P \approx e^{-qT}$, $\rho_C - \rho_P \approx KTe^{-rT}$), confirm that pathwise estimators have lower variance than likelihood ratio for delta, and verify that Asian option prices are strictly less than European. Outputs include 13×9 Greeks surfaces over spot and volatility grids.

mc-greeks-surface (75/240 min, Legacy Task). A variant of mc-greek-surface-1 with identical methodology. Both tasks test the same three Monte Carlo Greeks methods but with different parameterizations, verifying the agent's ability to generalize rather than memorize solutions.

hull-white-swaption (75/240 min). The agent calibrates a one-factor Hull-White short-rate model (Hull & White, 1990) ($dr = [\theta(t) - ar]dt + \sigma\, dW$) to 5 market caplet prices by minimizing sum-of-squared errors via L-BFGS-B optimization. Using the calibrated parameters, the agent constructs a trinomial tree with monthly time steps ($\Delta t = 1/12$) out to 10.5 years, calibrating Arrow-Debreu state prices at each node to match the market discount curve (verification: $\sum_j Q_n(j) = P(0, t_n)$ within 10^{-6}). European swaptions are priced analytically via Jamshidian's decomposition into zero-coupon bond options. Bermudan swaptions are priced via backward induction with optimal stopping on the tree, tracking the exercise boundary. DV01 risk is computed by bumping the yield curve $+1$bp and repricing. Calibration RMSE must be below 1bp. The task requires implementing 7 output files including calibration diagnostics, tree verification, and exercise boundary tracking.

stochvol-implied-surface (60/180 min). The agent prices European calls under a two-factor Heston stochastic volatility model using the Chiarella-Ziveyi semi-analytical characteristic function approach. Prices are computed on a 12×12 grid of strikes ($K \in [79, 165]$) and maturities ($T \in [1/12, 1]$). Two numerical integration methods are compared (`scipy.integrate.quad` and 2000-point Gauss-Legendre quadrature) with the agent selecting the method achieving max absolute price difference below 10^{-5}. The agent then extracts the implied volatility surface via Brent's root-finding, verifies put-call parity (all differences $< 10^{-6}$), and computes the Dupire local volatility surface via finite differences of the call price surface. Outputs span 10 files including 3D surface plots, integration comparison diagnostics, and parity verification.

structured-note-risk (50/100 min). The agent decomposes a structured note into a zero-coupon bond component (PV $= N \cdot e^{-rT}$) plus participation in a down-and-out barrier call option, priced analytically under Black-Scholes. A GARCH(1,1) model (Bollerslev, 1986) is fitted to 750 daily log-returns to estimate conditional volatility. The agent computes the note's fair value, breakeven volatility (the σ at which note value equals par), Greeks (Δ, Γ), and position risk metrics (VaR and Expected Shortfall at 99%) via historical simulation.

C.2 Risk management and portfolio tasks

barrier-garch-var (55/110 min). The agent fits a GARCH(1,1) model to 750 daily log-returns, prices a down-and-out barrier call analytically with Greeks (Δ, Γ, $\mathcal{V}$), and computes parametric VaR for a multi-contract position. The key deliverable is a VaR decomposition showing the percentage contribution of delta, gamma, and vega exposures to total portfolio risk, testing whether the agent understands the distinct risk channels in options portfolios.

kelly-var-sizing (55/110 min). The agent estimates Kelly optimal fractions (Kelly, 1956) for a 3-asset portfolio ($f^* = \Sigma^{-1}\mu_{\text{excess}}$), imposes a parametric VaR constraint (scaling fractions down if portfolio VaR exceeds the maximum), and simulates four sizing schemes (full Kelly, half Kelly, VaR-constrained Kelly, and equal-weight) via Monte Carlo with geometric compounding. For each scheme, the agent reports median terminal wealth, Sharpe ratio ($\sqrt{252} \times \bar{r}/s$), and median maximum drawdown across simulation paths.

regime-riskparity-cvar (60/120 min). The agent classifies market regimes via rolling eigenvalue decomposition of the correlation matrix for 15 stocks (5 sectors $\times$ 3), using the Marchenko-Pastur random matrix theory threshold ($\lambda_{\text{MP}} = (1 + \sqrt{q})^2$) to separate signal from noise eigenvalues. An absorption ratio (fraction of variance from above-threshold eigenvalues) is computed rolling, with quantile thresholds defining CRISIS, RISK_OFF, and RISK_ON regimes. The agent constructs monthly-rebalanced inverse-volatility risk-parity portfolios with regime-conditional risk budgets and computes historical CVaR at 99%.

cta-basel-capital (60/120 min). The agent builds an EMA-crossover trend-following CTA strategy across 5 assets with EWMA-based volatility targeting and per-asset leverage caps. A GARCH(1,1) model is fitted to portfolio returns for multi-step VaR forecasting. The final deliverable is a Basel Internal Models Approach (IMA) capital charge combining standard and stressed VaR components: Capital $= \max(\text{VaR}_{10d},\, k \times \overline{\text{VaR}}_{60d}) + \text{sVaR component}$.

C.3 Systematic trading tasks

momentum-backtest (30/90 min). The agent implements an EMA crossover momentum strategy for SPY (Jan 2000–Dec 2012). Golden crosses (EMA-50 above EMA-200) trigger buys at next day's open; death crosses trigger sells. The agent must handle trade execution mechanics (floor-based share calculation, one position at a time, forced close at series end) and compute standard performance metrics (annualized return, Sharpe ratio, max drawdown, Calmar ratio, win rate). An interactive Plotly chart with signal overlays is required.

bollinger-backtest-aapl (45/120 min). The agent implements a Bollinger Band mean reversion strategy for AAPL (Jan 2019–Jan 2023) with three execution refinements absent in the simpler momentum task: (1) transaction costs (flat \$9.99 + 0.1% proportional per trade), (2) risk-based position sizing (2% of portfolio value divided by stop distance), and (3) trailing stop-loss at 5% from peak price. Exit priority rules apply when both Bollinger and trailing-stop signals trigger simultaneously. The agent must produce per-trade analysis including entry/exit dates, exit reason classification, and intra-trade maximum drawdown.

regime-cta-vol-target (55/110 min). The agent fits per-asset GARCH(1,1) models to derive a composite volatility measure, classifies volatility into three regimes (LOW/MID/HIGH) using percentile thresholds, and compares a static vol-target CTA against a regime-adjusted variant where position sizes are scaled by regime-specific factors. The key metric is Sharpe ratio improvement from dynamic regime adaptation.

C.4 FACTOR MODEL TASKS

fama-french-factor-model (30/90 min). The agent decomposes daily returns of 10 stocks across 3 sectors (IT, Services, Finance) into market, size, and value factors using Fama-French 3-factor OLS regression over 2,515 trading days. Beyond basic regression, the agent must compute Newey-West HAC standard errors (Bartlett kernel), the GRS joint alpha test ($F(10, 2502)$ statistic for whether all alphas are jointly zero), Variance Inflation Factors for multicollinearity diagnostics, rolling 252-day betas, and sector-level aggregations. The task produces 8 output files including CSV summaries, a JSON with nested per-stock results, and 4 visualization PNGs.

sentiment-factor-alpha (60/120 min). The agent constructs a sentiment momentum factor from social media posts for 8 stocks. The pipeline involves engagement-weighted sentiment scoring ($1 + \ln(1 + \text{likes} + 2 \times \text{retweets} + 3 \times \text{replies})$), momentum signal construction (current aggregate minus lookback mean), and long-short portfolio formation (top-N long, bottom-N short with equal weights and transaction costs). Signal quality is evaluated via Spearman Information Coefficient (rank correlation between day-t signal and day-$(t+1)$ returns) and IC information ratio. Risk-adjusted performance is measured through market regression yielding Jensen's alpha, beta, and R^2.

C.4.1 ILLUSTRATIVE FAILURE MODE CASE STUDIES

Figure 7 summarises the five-stage iterative pipeline by which we built the taxonomy referenced in §5.3.1: 50 seed trials manually reviewed, bottom-up taxonomy construction, per-mode Cohen's κ calibration on a 20-trial stratified sample, five rounds of rubric refinement (κ improves monotonically from 0.69 to 0.89), and a cross-agent check that compares the failed trial against a passing solution to reduce attribution bias. Figure 8 anchors the methodology in a concrete real trial from our leaderboard (`fb-pr-s45-ipca-latent-factors`, Sonnet 4.5, reward 0.286): the spec does not pin a sign-anchoring rule for the extracted factors; the resulting deliverables show a near-zero Factor-1 Sharpe where the oracle expects $+0.307$, an ALS optimiser that takes $6\times$ more iterations than the reference, and additional deviations that warrant checking sign conventions, normalization, and numerical convergence. These diagnostics alone do not establish a unique root cause. All numbers in the figure are verbatim from `verifier/diagnostic.json` for that trial.

Case (a) below is a canonical Layer-2a failure (the math variant itself is wrong because the cited textbook formula was not followed); case (b) is a canonical Layer-2b failure (the math is internally correct but violates a practitioner convention the training corpus rarely surfaces).

(a) Layer 2a — cited methods that the agent ignores. In `implied-vol-approximations`, instruction L17 reads *"Reference: Chin, Ólafsson, Nel: Vol II §7.2.1"*; failing agents nonetheless implement *some* Li-ATM formula (the Minqiang Li 2005 cubic resolvent, the Hallerbach 2004 CMH variant, or a Cardano cubic that happens to be more accurate than the oracle) but never the variant in the cited textbook. Similarly in `event-study-earnings`, the spec composes BMP (1991) $\rightarrow$ Patell (1976) $\rightarrow$ Kolari–Pynnönen (2010); a Sonnet 4.6 trial computes `mean(SAR)` $\cdot\sqrt{N}$ (the literal Patell 1976 statistic) instead of BMP's 1991 cross-sectional standardisation. The agent treats the citation as a topic label rather than a pointer to be dereferenced—a Theory-layer failure where the agent's chosen formula is mathematically wrong relative to the named reference.

(b) Layer 2b — practitioner conventions the agent lacks. On `lob-pc-signal`, the spec asks that *"consecutive changes in the PCA component should be minimized,"* which any practicing quant reads as an instruction to anchor PC1 sign to a dominant economic signal (here, $\text{corr}(\text{pc1}, \text{weighted_ofi}_0) > 0$) and then enforce sign continuity across rolling windows. 21 of 23 failing trials implement only the consecutive-window stabilization, inheriting `sklearn`'s sign-arbitrary first-window loadings and producing exact -1 flips on every pinned numeric test. The implementation is one line away from passing; what is missing is not reasoning capacity but the inline convention a practitioner supplies automatically because it is rarely surfaced in public training data—an Empirical-layer failure where the math is internally consistent but the industry default was not chosen.

D MODELS EVALUATED

Finance-Zero (38 base models from 13 providers). Google (7): Gemini 2.0 Flash, Gemini 2.5 Flash, Gemini 2.5 Flash Lite, Gemini 2.5 Pro, Gemini 3 Flash Preview, Gemini 3 Pro Preview, Gemini

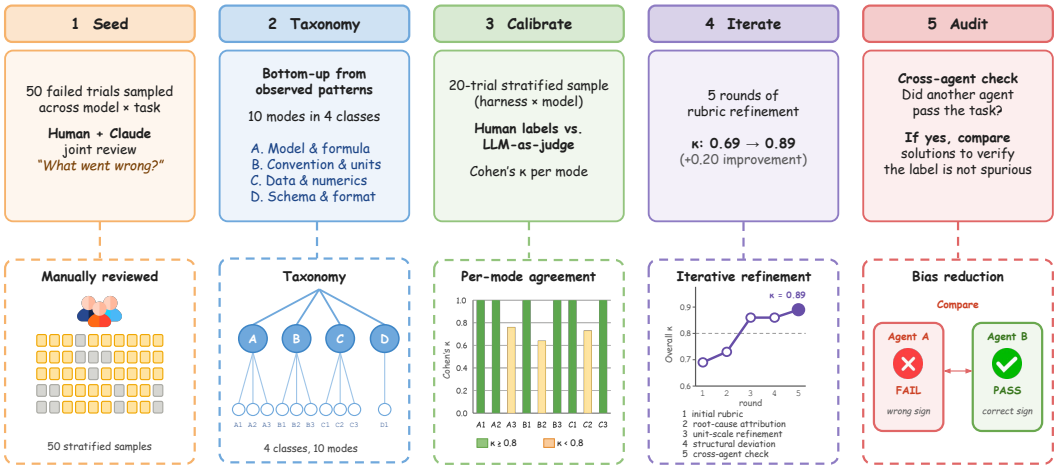


Figure 7: Five-stage iterative pipeline for constructing the QF-Bench failure taxonomy. Top row: stage summaries (Seed → Taxonomy → Calibrate → Iterate → Audit). Bottom row, from left to right: 50 stratified seed samples; a taxonomy tree comprising 4 classes and 10 content failure modes; per-mode Cohen's κ on the 20-trial calibration sample (dashed line: $\kappa = 0.8$), reported for the nine quantitative-content modes A1–C3. D1 (Schema or Format) is excluded because it is evaluated by a deterministic schema validator and therefore does not require human-annotation agreement; the overall κ trajectory across the five refinement rounds shown in the figure ($0.69 \rightarrow 0.89$). Seven calibration rounds were conducted in total, but rounds 1–2 predated strict-JSON parsing and are excluded because parse failures could mask genuine disagreement in paired judgments; and a cross-agent check that compares a failed solution of agent with a passing solution for the same task to reduce attribution bias.

3.1 Pro. OpenAI (3): GPT-5.2, GPT-5.4, GPT-5.5. Anthropic (3): Claude Haiku 4.5, Claude Sonnet 4.6, Claude Opus 4.7. Alibaba (6): Qwen 2.5 Coder 32B, Qwen 3 235B, Qwen 3 Coder Next, Qwen 3.5 9B, Qwen 3.5 122B, Qwen 3.5 Plus. DeepSeek (4): DeepSeek V3, DeepSeek V3.2, DeepSeek V4 Pro, DeepSeek R1. Moonshot AI (3): Kimi K2, Kimi K2 Thinking, Kimi K2.5. Zhipu AI (3): GLM 4.7, GLM 5.1, GLM 5 Turbo. MiniMax (3): MiniMax M2.5, MiniMax M2.7, MiniMax M2.7 HS. Meta (2): Llama 4 Scout, Llama 4 Maverick. Mistral AI (1): Mistral Large. xAI (1): Grok 4.20. StepFun (1): Step 3.5 Flash. Xiaomi (1): MiMo V2.5 Pro.

Agent regime (9 CLI agent configurations). Claude Code paired with Anthropic Opus 4.6, Opus 4.7, Sonnet 4.5, Sonnet 4.6, and Haiku 4.5; and Codex CLI paired with OpenAI GPT-5.3-codex, GPT-5.4, GPT-5.4-mini, and GPT-5.5. Gemini CLI agents are reported in the supplementary material.

Operational protocol. All runs use temperature 0.0 where supported, with up to 8 k output tokens, up to 8 parallel workers, and a per-task wall-clock budget of 30 minutes for the agent regime (120 seconds for Finance-Zero, since execution is single-shot). Results are persisted as structured JSON containing per-task reward, per-test outcomes, token usage, USD cost, and wall-clock time; the pipeline supports resumption so completed (model, task) pairs are skipped on re-execution.

E Cost Analysis

Methodology. For every trial we read `n_input_tokens` and `n_output_tokens` from the provider response and convert to USD using each model's published list price (OpenRouter, May 2026 snapshot). When a vendor's `cost_usd` field is missing in the API response we substitute the published price; reported numbers therefore reflect what the caller would pay rather than any internal billing field, and use a uniform basis across all models. Coverage is reported in Table 4: 6,936 cost-tracked trials spanning CLI agents and Finance-Zero. Throughout this appendix, Trials denotes the number of cost-tracked trial records included for each configuration; trial counts are not balanced across configurations.

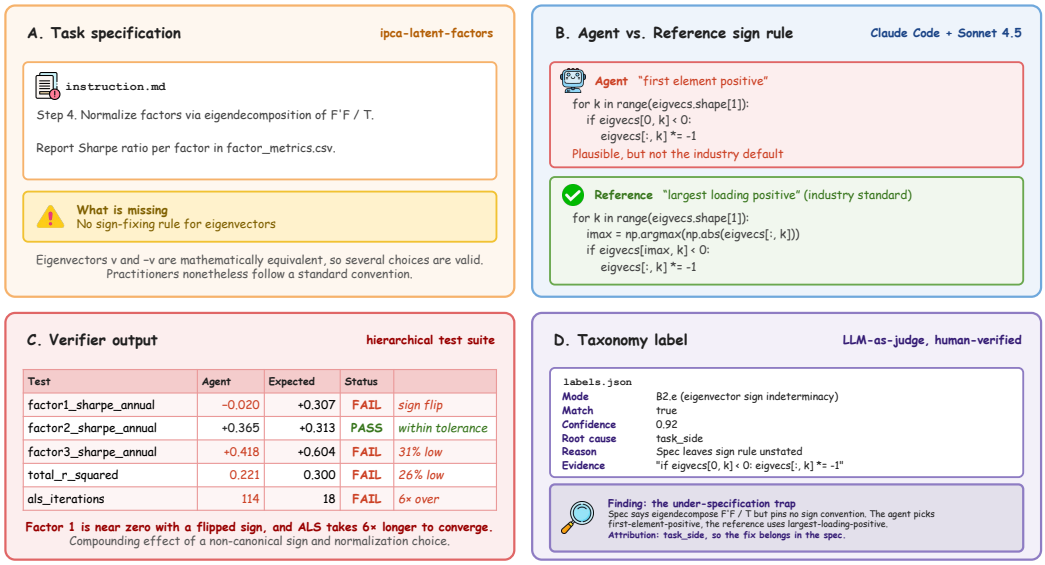


Figure 8: Failure case study on a real trial of `ipca-latent-factors`. (a) The task specification asks the agent to normalize factors via eigendecomposition and report Sharpe ratios, but it does not state the sign-anchoring rule for the eigenvectors. The reference solution applies this Layer-2b convention implicitly. (b) The agent anchors each eigenvector so that its first element is positive, whereas the reference makes the largest loading positive, which is the industry standard. (c) Verifier output from Claude Code with Sonnet 4.5 (reward 0.286, all values verbatim from `verifier/diagnostic.json`). The Factor-1 Sharpe collapses to -0.020 against an expected $+0.307$, the Factor-3 Sharpe is 31% low, and the ALS optimiser consumes $6\times$ the reference iteration budget. (d) The resulting taxonomy label is mode B2.e (eigenvector sign indeterminacy) with root cause `task_side`. The fix therefore belongs in the specification rather than in the agent's finance knowledge, which illustrates the under-specification trap.

Table 4: Total token usage and cost across all trials.

	Trials	Total tokens	Cost ($)
CLI Agent	4,840	2.24 B	$11,749
Finance-Zero	2,096	13.9 M	$396
Grand total	6,936	2.25 B	$12,145

Aggregate cost.

Per-model efficiency: agent regime. Table 5 ranks CLI agents by USD per fully-resolved task ($/pass), a deployment-relevant efficiency metric. Codex CLI with GPT-5.4-mini is the most cost-efficient configuration at $0.97 per pass, approximately $14\times$ lower than Claude Code with Opus 4.6 ($13.29). GPT-5.3-codex consumes the fewest tokens per trial (217K), and Codex CLI with GPT-5.5 achieves the highest leaderboard pass rate (61.7%) at $2.38 per pass, placing it on the accuracy–cost Pareto frontier.

Per-model efficiency: Finance-Zero. Finance-Zero consumes substantially fewer tokens per trial than the reported CLI agents (Tab. 6). The lowest cost-per-pass we observed is Sonnet 4.6 under Finance-Zero at $0.22, more than $4\times$ below the cheapest agent configuration; Opus 4.7 under Finance-Zero costs $0.87 per pass. These Finance-Zero configurations have lower pass rates than their corresponding CLI configurations.

Table 5: CLI Agent efficiency, sorted by $/pass. Pass rate is binary pass@1 over all trials of that agent across the 86-task suite. Trials denotes the number of cost-tracked records included for that configuration.

Agent	Trials	Tok/trial	Pass rate	$/pass	Total ($)
Codex CLI + GPT-5.4-mini	610	495K	57.1%	**$0.97**	$337
Codex CLI + GPT-5.3-codex	603	217K	60.8%	$1.17	$430
Claude Code + Sonnet 4.6	266	374K	56.3%	$2.21	$331
Codex CLI + GPT-5.5	605	272K	61.7%	$2.38	$890
Codex CLI + GPT-5.4	745	352K	57.5%	$3.23	$1,385
Claude Code + Sonnet 4.5	580	591K	46.2%	$4.15	$1,113
Claude Code + Haiku 4.5	584	978K	20.8%	$5.00	$607
Claude Code + Opus 4.7	262	440K	61.2%	$12.80	$2,053
Claude Code + Opus 4.6	585	422K	59.2%	$13.29	$4,602

Table 6: Finance-Zero efficiency for the agent base models.

Model	Trials	Tok/trial	Pass rate	$/pass
Claude Sonnet 4.6	596	6.5K	32.7%	**$0.22**
Claude Haiku 4.5	303	6.6K	8.9%	$0.27
Claude Opus 4.7	610	6.6K	38.5%	$0.87

Pareto-dominant Finance-Zero models. Table 7 enumerates the configurations on the cost–accuracy Pareto frontier visualized in Figure 3 (Section 5.2). Across the listed configurations, higher pass@1 is accompanied by higher per-round cost, with varying cost increments.

Table 7: Pareto-dominant Finance-Zero models (cost vs. pass@1).

#	Model	$/round	pass@1
1	qwen-2.5-coder-32b	$0.01	0.0%
2	mistral-large	$0.06	0.7%
3	deepseek-v3	$0.09	9.1%
4	deepseek-v3.2	$0.18	12.8%
5	gemini-3-flash-preview	$0.82	17.5%
6	gemini-3-pro-preview	$3.49	22.2%
7	gpt-5.4	$4.90	26.5%
8	claude-sonnet-4.6	$6.61	32.1%
9	claude-opus-4.7	$10.18	39.3%
10	gpt-5.5	$18.05	42.1%

Practical takeaways. We highlight three observations. First, *tokens-per-trial vary by orders of magnitude across scaffolds and base models*: Claude Code with Haiku 4.5 averages nearly 1 M tokens per trial, whereas GPT-5.3-codex remains below 220K and Finance-Zero below 10K. Second, *cost-per-pass is non-monotonic in headline accuracy*: GPT-5.4-mini at $0.97 per pass dominates Opus 4.7 at $12.80 per pass on cost while resolving only 4.1 pp fewer tasks. Third, *Finance-Zero remains competitive on cost-per-pass when paired with a strong base model*: Sonnet 4.6 under Finance-Zero at $0.22 per pass is the lowest-cost configuration we observed, and is appropriate when the deployment can tolerate a reduced pass rate or employs Finance-Zero as an upstream filter that promotes failed cases to an agentic scaffold.

F PER-TASK SUMMARY

Table 8 reports the available per-task results on QF-Bench, summarizing pass@1, total attempts across our 38-model evaluation pool, token consumption, and cost in Finance-Zero regime. Rows are sorted by tier and then by pass@1 descending. This table is the per-task companion to the aggregate Pareto analysis in Appendix E.

Table 8: Per-task summary across 38 base models in Finance-Zero regime. *Tokens* are summed across all models in thousands. *Cost/round* is the total USD spend across all models for one round of this task; *Cost/run* is the average per-(model, round) cost.

Task	Tier	Pass@1	Runs	Tokens in/out (k)	Cost/round ($)	Cost/run ($)
interest-rate-cap-floor	easy	88.6%	114	32k / 65k	0.29	0.008
variance-swap-replication	easy	67.5%	114	76k / 137k	0.65	0.017
smith-tail-index	easy	50.4%	131	37k / 167k	0.89	0.023
delta-hedging-pnl-simulation	easy	49.2%	122	62k / 175k	0.76	0.020
cir-bond-pricing	easy	46.5%	114	74k / 182k	1.01	0.026
sma-crossover-spy	easy	46.5%	114	61k / 173k	0.83	0.022
momentum-backtest	easy	44.4%	81	62k / 185k	0.88	0.023
historical-var-data-prep	easy	39.5%	114	52k / 99k	0.47	0.012
merton-jump-diffusion	easy	39.3%	84	63k / 166k	0.90	0.024
zero-coupon-bootstrapping	easy	36.0%	111	31k / 168k	0.78	0.020
fama-french-factor-model-new	easy	34.2%	114	156k / 203k	1.20	0.032
pca-factor-portfolio	easy	33.3%	114	39k / 88k	0.41	0.011
bs-greeks-pde	easy	27.7%	119	70k / 159k	0.82	0.022
bollinger-backtest-aapl	easy	27.2%	125	96k / 222k	1.14	0.030
ou-jump-commodity	easy	27.2%	81	60k / 178k	0.92	0.024
cross-sectional-momentum	easy	25.4%	114	38k / 134k	0.61	0.016
option-put-call-parity-forward-audit	easy	24.6%	114	65k / 197k	1.04	0.027
kelly-var-sizing	easy	0.9%	106	36k / 189k	0.93	0.024
realized-vol-estimators	medium	25.0%	80	67k / 165k	0.70	0.019
geometric-mean-reverting-jd	medium	23.0%	113	69k / 183k	0.97	0.025
yield-curve-pca-dynamics	medium	22.8%	92	74k / 163k	0.74	0.020
spread-option-kirk-margrabe	medium	20.0%	35	58k / 167k	0.90	0.026
double-sort	medium	18.4%	114	31k / 159k	0.73	0.019
credit-spread-decomposition	medium	18.3%	120	87k / 188k	0.96	0.025
copula-equity-fitting	medium	17.2%	122	36k / 211k	1.03	0.027
asian-option-levy-curran	medium	16.3%	123	68k / 234k	1.14	0.030
var-es-estimation	medium	14.8%	128	56k / 205k	0.95	0.025
digital-barrier-options	medium	14.7%	116	53k / 219k	1.10	0.029
creditmetrics-portfolio-var	medium	14.0%	121	59k / 202k	0.97	0.025

Task	Tier	Pass@1	Runs	Tokens in/out (k)	Cost/round ($)	Cost/run ($)
fft-compound-poisson	medium	13.2%	114	44k / 221k	1.07	0.028
intraday-volume-fitting-/-and-execution-scheduling	medium	12.5%	112	112k / 220k	1.07	0.028
first-passage-time	medium	11.4%	114	73k / 236k	1.15	0.030
copula-sampling-rank-correlation	medium	11.1%	117	49k / 225k	1.02	0.027
cliquet-ratchet-pricing	medium	10.5%	114	54k / 163k	0.79	0.021
standard-var-methods	medium	9.6%	135	40k / 215k	1.07	0.028
yield-curve-bootstrap-immunization	medium	9.4%	32	71k / 222k	1.07	0.033
credit-migration-matrix	medium	8.3%	120	144k / 224k	1.29	0.034
residual-momentum	medium	8.0%	138	50k / 239k	1.20	0.032
brinson-sector-attribution	medium	7.4%	121	51k / 219k	1.06	0.028
compound-option-geske	medium	5.7%	122	76k / 256k	1.29	0.034
barone-adesi-whaley	medium	5.3%	113	71k / 252k	1.31	0.035
stochvol-implied-surface-new	medium	5.2%	97	92k / 275k	1.48	0.039
etf-cross-asset-lead-lag	medium	5.1%	118	138k / 309k	1.66	0.044
fomc-tone-event-study	medium	4.5%	112	134k / 286k	1.60	0.042
barrier-garch-var	medium	0.0%	36	44k / 190k	0.99	0.028
mc-greek-surface-1	medium	0.0%	23	73k / 166k	1.03	0.045
prediction-markets-cross--/venue-dislocation	medium	0.0%	114	156k / 283k	1.60	0.042
regime-cta-vol-target	medium	0.0%	36	66k / 190k	1.05	0.029
regime-riskparity-cvar	medium	0.0%	34	44k / 173k	0.86	0.025
sentiment-factor-alpha	medium	0.0%	36	69k / 212k	1.15	0.032
structured-note-risk	medium	0.0%	114	29k / 202k	1.03	0.027
ohlc-realized-vol-estimators	hard	27.8%	36	61k / 190k	1.03	0.029
lob-pc-signal	hard	6.5%	31	39k / 137k	0.80	0.026
earnings-surprise-calculator	hard	4.2%	120	25k / 78k	0.36	0.009
american-option-fd-new	hard	1.7%	115	65k / 255k	1.38	0.037
lookback-options	hard	1.0%	104	99k / 232k	1.26	0.033
credit-portfolio-var-cvar	hard	0.9%	115	182k / 279k	1.60	0.042
13f-amendment-aware-crowding	hard	0.0%	123	60k / 281k	1.58	0.042
alpha-hedge-strategy	hard	0.0%	123	48k / 184k	0.99	0.026
binance-btc-participation-tca	hard	0.0%	114	84k / 241k	1.29	0.034
bl-regime-hmm	hard	0.0%	114	68k / 212k	1.06	0.028

Task	Tier	Pass@1	Runs	Tokens in/out (k)	Cost/round ($)	Cost/run ($)
cme-hdd-option-pricing	hard	0.0%	119	61k / 244k	1.30	0.034
corporate-action-adjustment	hard	0.0%	114	30k / 150k	0.71	0.019
crypto-funding-rate-basis-carry	hard	0.0%	32	113k / 217k	1.26	0.039
cta-basel-capital	hard	0.0%	35	64k / 189k	0.97	0.028
dcc-garch-portfolio-var	hard	0.0%	121	60k / 226k	1.13	0.030
dupire-local-vol	hard	0.0%	27	43k / 172k	1.26	0.047
etf-overlap-redemption-pressure	hard	0.0%	117	84k / 301k	1.66	0.044
event-study-earnings	hard	0.0%	114	67k / 243k	1.28	0.034
evt-pot-var	hard	0.0%	113	42k / 263k	1.26	0.033
ewma-portfolio-risk-decomposition	hard	0.0%	114	43k / 131k	0.64	0.017
form4-cross-sectional-sale-pressure	hard	0.0%	113	71k / 272k	1.47	0.039
fx-carry-forward-hedge	hard	0.0%	110	149k / 286k	1.63	0.043
fx-forward-cross-rate	hard	0.0%	35	57k / 226k	1.21	0.035
hull-white-swaption	hard	0.0%	114	126k / 281k	1.62	0.043
implied-vol-approximations	hard	0.0%	111	64k / 225k	1.16	0.031
ipca-latent-factors	hard	0.0%	114	80k / 237k	1.27	0.033
localvol-barrier	hard	0.0%	28	76k / 180k	1.02	0.036
mtm-xccy-basis-desk	hard	0.0%	79	126k / 266k	1.54	0.042
multimodal-alpha-fusion-/edgar-cot-gdelt	hard	0.0%	111	168k / 310k	1.72	0.045
polars-api-migration	hard	0.0%	114	23k / 120k	0.76	0.020
sec-10k-report-long	hard	0.0%	114	106k / 251k	1.42	0.037
sec-8k-event-alpha	hard	0.0%	114	50k / 217k	1.21	0.032
stable-residual	hard	0.0%	152	183k / 269k	1.57	0.041
swap-curve-bootstrap-ois	hard	0.0%	97	70k / 270k	1.48	0.039
yield-curve-bond-immunization	hard	0.0%	96	120k / 303k	1.67	0.044

G BINARY TASK–MODEL COVERAGE

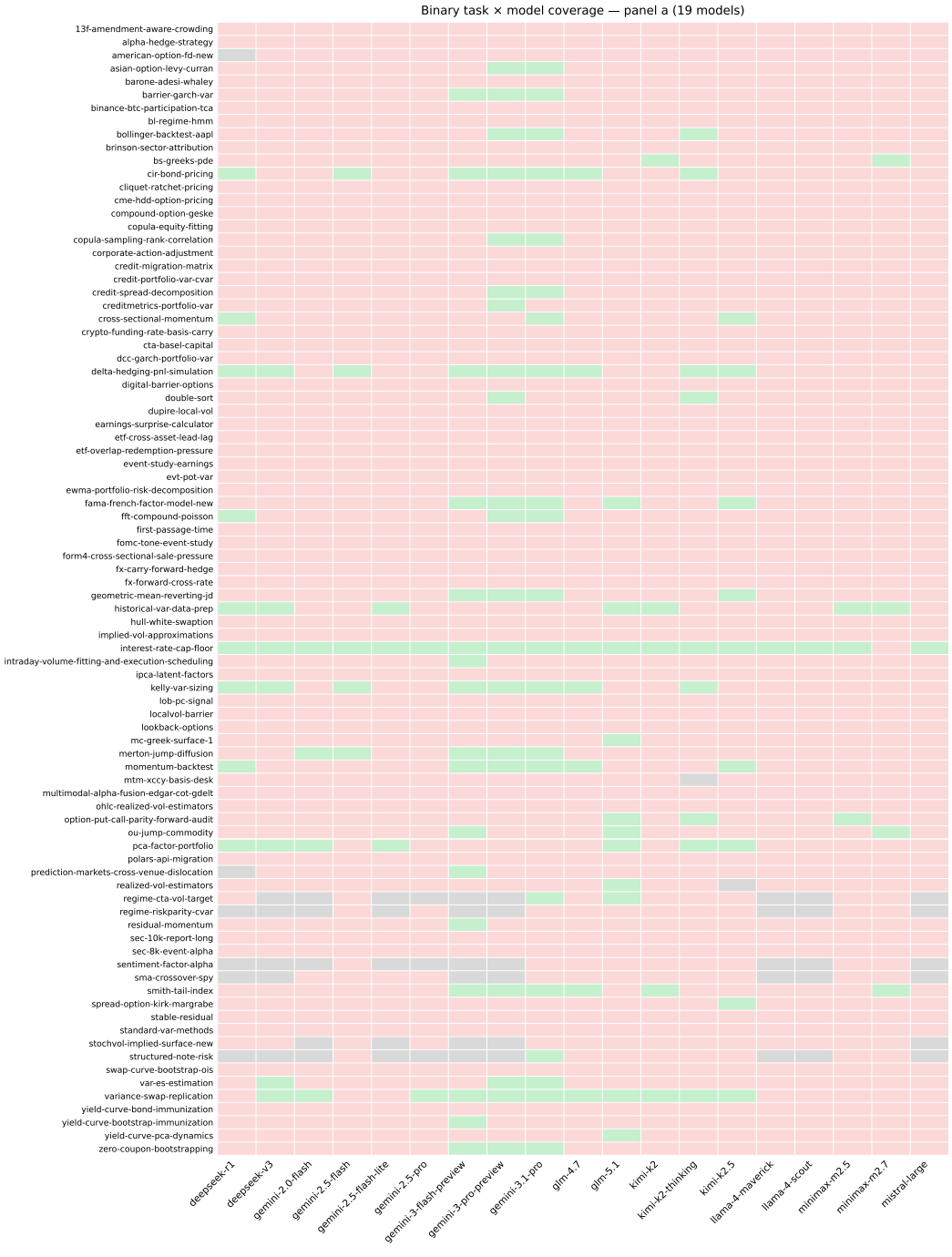


Figure 9: Binary task × model coverage, panel (a): first 19 models. Light green: pass@1 > 0.5; light pink: pass@1 ≤ 0.5; gray: not run.

Figures 9–10 render the binary task-by-model resolution map under Finance-Zero (single-call regime). Each cell encodes the per-(task, model) pass@1 *thresholded at 0.5*: a green cell indicates the model passed strictly more than half the rounds on that task, a pink cell indicates it passed at most half, and a gray cell indicates the (task, model) pair was not run (typically due to provider rate limits or model

unavailability during the evaluation window). Models are split into two panels for readability; tasks are listed alphabetically and identical between panels so the two figures can be read as a left-right continuation of one matrix.

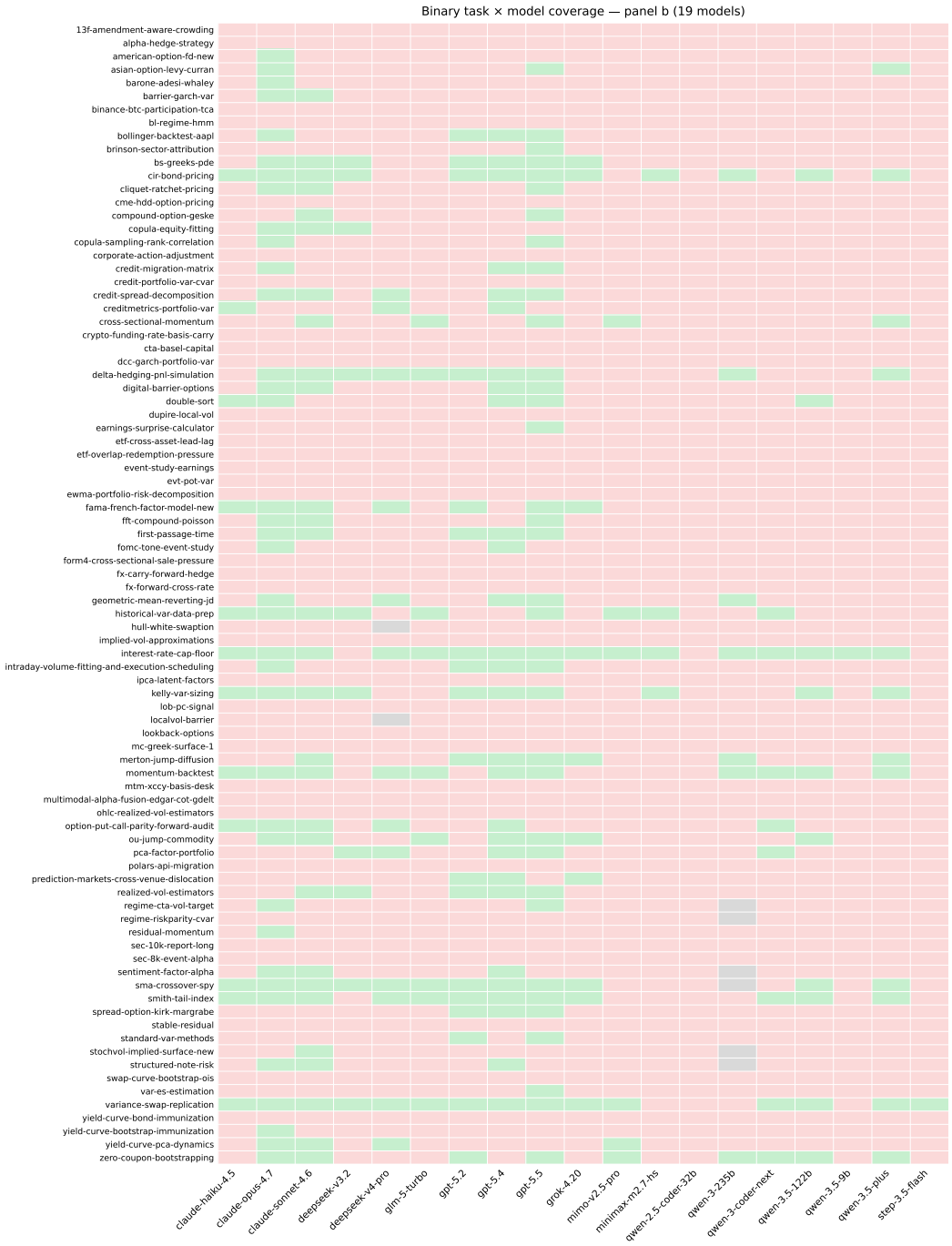


Figure 10: Binary task × model coverage, panel (b): remaining 19 models (same task ordering as panel (a)).

H SANDBOX BASE IMAGE

All tasks execute inside Docker containers built from a shared base image providing Python 3.11, core scientific libraries (NumPy, Pandas, SciPy, Matplotlib), financial libraries (TA-Lib compiled from source, Backtrader, statsmodels, scikit-learn, yfinance), and data format support (OpenPyXL, PyArrow). Per-task Dockerfiles inherit from this base and add task-specific datasets to `/app/data/`. This provides a realistic and reproducible environment while avoiding library installation as a confounding factor.

The shared Docker base image specification:

```
FROM python:3.11-slim
RUN apt-get install build-essential gcc g++ wget curl
# TA-Lib compiled from source (v0.6.4)
RUN wget [ta-lib] && ./configure && make install
pip install numpy pandas scipy ta-lib backtrader \
    statsmodels scikit-learn matplotlib seaborn \
    openpyxl pyarrow tqdm requests yfinance
RUN mkdir -p /app/data /app/output /app/tests
WORKDIR /app
```

I SINGLE-PASS VS AGENTIC: DETAILED COMPARISON

This appendix provides the detailed breakdown supporting the summary in §5.3.

I.1 SETUP AND HEADLINE

Two judging protocols were applied to QF-Bench failed trials, with the coverage described below. The single-pass LLM run does one deep-read per trial across approximately 900 trials in the master deep-read document plus another 250 in the recent finance-zero rerun on 17 difficult tasks ($\sim$1,100 trials total of mixed agent kinds), covering 75 task families. The agentic run layers cross-batch consistency, test-name verification, and post-cross-check falsification on top of the same per-trial deep-read, judging just over 1,000 multi-step trials across 78 task families. The big shifts are on engineering attribution, not on finance reasoning. The engineering-bug class drops by roughly sixfold (41% of failures to under 7%); the systematic finance-knowledge-gap class shifts in the opposite direction, nearly doubling, because the protocol re-routes engineering crashes whose underlying quant reasoning is also wrong toward deeper attribution. The underlying quant errors themselves do not disappear: missing-formula and wrong-parameterization sub-rubrics appear in both runs at similar per-trial rates. The headline class distributions, computed over each side's full corpus, are shown below.

Failure attribution	**Single-pass LLM**	**Agentic**	**Shift (pp)**
Systematic finance-knowledge gap	44.7% (395)	76.9% (784)	+32.2
Engineering / coding bug	41.0% (362)	6.7% (68)	−34.3
Infrastructure / harness issue	8.1% (72)	15.6% (159)	+7.5
Agent / harness UX trap	1.9% (17)	0.8% (8)	−1.1

I.2 ENGINEERING FAILURES AMENABLE TO ITERATIVE CORRECTION

The examples below illustrate three engineering failure modes that multi-step execution can help address. First, output-token-budget truncation kills single-shot finance-zero agents on tasks whose canonical solutions exceed the model's emission cap. Second, schema-introspection failures hit single-shot agents that commit to a default column-name prior without inspecting input files. Third, iterative-debugging recovery closes the rest: one-keyword fixes on date-parsing or deprecated pandas APIs that a second pass on the test trace would correct trivially.

Task	Single-shot fail share	Failure trigger
Monte Carlo Greek-surface (`mc-greek-surface-1`)	every trial	8 output files, 60 Greeks entries, surface grids, convergence study, plot: exceeds emission cap
Local-volatility barrier (`localvol-barrier`)	majority of trials	∼25–28K-character Dupire FD task; mid-statement truncation
ETF cross-asset lead-lag	most trials	∼500–600 line implementation; smaller models truncate earlier
Fama–French factor model	every trial	Markdown fence not stripped; SyntaxError on opening fenced line
FX cross-rate	roughly four of every five	KeyError on guessed column names without input introspection
Spread-option Kirk–Margrabe	most trials	Case-mismatched column names
Dupire local-volatility	a third of trials	Hallucinated column headers
Regime-risk-parity CVaR	nearly half of trials	Deprecated pandas API (`fillna(method='ffill')`)
Historical-VaR data prep	single trial	One-keyword fix: `format='mixed'` on multi-format dates

I.3 Convention defaults and aggregation-axis errors (finance gap, theme one)

The dominant remaining finance failure mode is wrong choice of an under-specified convention or aggregation axis, observed at near-identical rates in single-pass and agentic. Concrete instances from the per-task deep reads:

Task	Convention error (vs. specification)
Regime-CTA vol target	Composite vol uses linear mean of per-asset annualized GARCH conditional vols (≈ 0.213) instead of the spec-mandated root-mean-square / portfolio-variance aggregator (≈ 0.234, ratio $\sqrt{0.83}$). Industry has no canonical aggregator to default to. Affects every non-crashing trial.
Regime-risk-parity CVaR	Absorption ratio uses the Marchenko–Pastur noise-filter form (sum of eigenvalues above MP threshold over total) when the specification explicitly mandates the top-three eigenvalues over the sum and explicitly forbids the MP form. Affects every trial.
CreditMetrics portfolio VaR	Mean-loss denominator computed per-bond-per-simulation instead of per-rating-category-per-simulation, yielding values smaller than reference by a factor of twenty (the bonds-per-rating count). Latent in four other rating buckets that the test does not assert.
Sentiment-factor alpha	Vol-target estimator uses realized standard deviation of past portfolio returns instead of the spec-mandated ex-ante asset-covariance times position vector. Plus missing leverage-cap clip.
Put-call-parity forward audit	Even-n median computed as upper-index (`sorted(xs)[n//2]` $=$ 100.420) instead of canonical midpoint average ($\frac{100.330+100.420}{2} =$ 100.375). Majority of non-codex trials.

I.4 Canonical-formula misreadings, plus practical implication

The second-largest finance gap is misreading well-known canonical formulas, also persistent in both methodologies. The protocol's deeper-analysis stages re-route engineering crashes toward systematic finance attribution and surface more canonical-coefficient errors than single-pass (because multi-step agents reach the formula step before crashing), but they do not close the underlying gaps.

Task	Canonical-formula misreading
Lookback options	Conze–Viswanathan fixed-strike call bifurcated into in-the-money / out-of-the-money branches when the canonical formula at the no-history boundary is unified. Opus's M-correction sub-block exactly cancels the K-correction sub-block (gives zero); Sonnet's out-of-the-money branch uses a sign-flipped normal-CDF argument.
Yield-curve bootstrap immunization	Svensson third coefficient keyed on the second decay rate instead of the first, leaving the long-rate constant off by roughly 36%. Diebold–Rudebusch attribution.
OHLC realized-volatility estimators	Rogers–Satchell formula substitutes log-high-over-low for log-high-over-open and log-low-over-open in the cross-product. Cascades into clamped-to-zero negative variance.
Zero-coupon bootstrapping	Every single-shot trial divides the one-year forward rate by the gap between maturities "to annualize," although the specification literally pins the divisor to one.
Barrier-GARCH VaR	Every non-crashing trial invents some forbidden vol-of-vol heuristic; the spec's exact realized-standard-deviation-of-daily-first-differences procedure is implemented by no trials.
Residual momentum	Fama–MacBeth t-statistic reported as the per-month Newey–West standard error (magnitudes 60–70) instead of the canonical mean-of-coefficients divided by their across-month standard error (magnitudes well under 5). Wrong measure entirely.

Practical implication. Switching to agentic does not fix the regime-CTA, regime-risk-parity, CreditMetrics, lookback-options, barrier-GARCH, or residual-momentum failures; the fix is on the finance-knowledge side: better priors or industry-standard retrieval (Conze–Viswanathan unified formula; Diebold–Rudebusch Svensson keying; MSCI / AQR composite-volatility aggregator; Rogers–Satchell 1991; Smith heavy-tail prior; Fama–MacBeth cross-sectional convention) before code emission.

I.5 Why does the agentic run actually surface more finance failures?

The 32-pp share growth on the finance side has two distinct empirical signatures. The first is dominant; the second is a real but smaller second-order effect. Both are visible in the trial data.

Reach-further (survival bias). Single-shot agents on most tasks die early: truncation, KeyError on the data load, or a runtime crash before they ever write a finance formula. The agent never had to commit to an aggregator, a sign convention, a day-count, or a canonical-formula coefficient, so no finance-mode label could fire. Agentic agents iterate past the engineering layers and *must* write the formulas, exposing the same finance-knowledge gap the single-shot trials had implicitly. The empirical fingerprint is unambiguous: deep finance sub-rubrics inflate while the engineering one-keyword-fix bin stays roughly flat.

Sub-rubric	Single-pass	Agentic	Mult.
Wrong canonical ODE / characteristic-fn coefficient (Heston, Hull–White, Svensson)	0	75	∞
Missing formula term (Jacobian, $/S_0$ chain rule, normalization)	27	118	4.4×
Non-industry-standard convention default	18	206	11.4×
Missing or extra formula term (generic)	16	82	5.1×
Library API default (engineering one-keyword fix)	*19*	*22*	*flat*

The control row tells the story: the engineering library-default class is essentially unchanged. Only finance-content categories inflate. This is exactly the survival-bias signature: agentic agents reach the finance step at all, and the finance-knowledge gap that was always there now has a place to land.

First-attempt lock-in (the dominant within-trial mechanism). Within a single test, a multi-turn agent makes many LLM calls and can in principle revise its finance commitments at any turn. To

test whether iteration moves the agent across the correct/wrong boundary, we ran a structured before-and-after analysis on 114 multi-edit CLI trials across 20 tasks: for each trial, the first concrete code attempt (v01) and the last-version commit (v_{last}) were classified against the task's canonical formula. The transition matrix is overwhelmingly diagonal in the wrong-and-stayed-wrong cell.

Transition (v01 → v_{last})	**Count**	**Share**
LOCK-IN (wrong → wrong)	**74**	**64.9%**
SUCCESS (correct → correct)	21	18.4%
RECOVERY (wrong → correct)	7	6.1%
DRIFT (correct → wrong)	4	3.5%
MIXED (oscillation) / UNCLEAR	8	7.0%
Total multi-edit trials	114	100%

Two-thirds of trials lock in their v01 finance commitment and never escape it across all later edits; recovery (iteration successfully fixing v01) is more common than drift (iteration making v01 worse) by 7:4. The dominant in-trial mechanism is therefore not drift but *first-attempt lock-in*: the agent's prior commits in v01, multi-turn iteration often fails to revise it, and the trial ends where it began. Concretely, in the lookback-options Opus 4.6 trial, the agent's only Write call commits a bifurcated `if S≥K / else` framing that contradicts the canonical Conze–Viswanathan unified formula. Furthermore, 119 "actually" / 74 "reconsider" instances of internal reasoning afterward never cross the boundary back to canonical-correct. In the credit-portfolio-var-cvar Opus 4.6 trial, by contrast, v01 was largely correct (per-obligor Vasicek with + sign, correct Gordy α-derivatives), but v04 reformulated the Gordy chain rule from the spec's $dh/d\alpha$ to a non-canonical dh/dq parameterization that drops a load-bearing term: this is a clean DRIFT case, but a rare one (4 of 114 trials).

Why finance errors can persist through iteration. The asymmetry comes from where the oracle lives: the sandbox *contains* the engineering ground truth (column names from `df.columns.tolist()`, deprecation warnings, the line where SyntaxError fired). Multi-turn iteration converges to truth on engineering because the sandbox itself is the oracle. The sandbox does *not* contain the finance-canonical-convention ground truth (no Conze–Viswanathan textbook, no Diebold–Rudebusch keying note, no MSCI/AQR composite-volatility documentation). For finance conventions, a wrong initial assumption can persist across later edits. In this sample, persistent errors are more common than successful corrections, although corrections do occur.

J LIMITATIONS

Task count and statistical power: At 86 tasks QF-Bench is comparable in size to Terminal-Bench (89) but substantially smaller than SWE-Bench (2,294). Each task is substantially more domain-specific, and the difficulty stratification (18/33/35) means some tier-level statistics are based on small denominators.

Agentic coverage: The headline evaluation covers Claude Code and Codex CLI under three independent runs per model and task. This is enough to compute pass@1 / pass@3 but not enough to characterize the full distribution of agentic execution; runtime API rate limits also forced a small number of (model, task) cells to be excluded from the headline pool.

Data contamination: Despite canary strings, models may have seen similar problems during training. We mitigate by using real market data for non-trivial date ranges and unconventional instruments, but cannot guarantee zero leakage.

Domain coverage: While the 86 tasks span derivatives, fixed income, credit, factors, risk, microstructure, FX/crypto, and NLP-on-finance, several practitioner-relevant areas remain under-represented, including high-frequency market-making, regulatory capital under Basel IV, climate-financial-risk stress tests, and structured-credit waterfall mechanics. We plan to expand in future releases.

External dependencies: Some tasks rely on library behavior (TA-Lib, Backtrader) that may change across versions. We pin all versions in the base Docker image for reproducibility.

K BROADER IMPACTS

Intended use. QF-Bench is an evaluation artifact: it measures how reliably language-model agents implement the computational core of professional quantitative finance. Its purpose is to expose capability gaps that general-purpose coding benchmarks cannot detect—sign conventions, no-arbitrage bounds, calibration convergence, day-count accuracy—so that researchers and practitioners can make informed decisions about where current systems are and are not deployment-ready.

Positive impacts. By grounding evaluation in domain-specific invariants rather than free-form correctness, QF-Bench discourages over-reliance on headline coding-benchmark scores when assessing finance-AI deployments. The hierarchical verifier and L1/L2 failure taxonomy turn opaque benchmark numbers into actionable diagnostic signals (*which* class of error a model commits, on *which* task family), which is the form practitioners need to design human-in-the-loop workflows or model-routing policies.

Risks of misuse. Quantitative finance is a high-stakes domain: a sign error in a Greek, a wrong VaR confidence level, or a calibration that silently fails to converge can translate into material monetary losses or mis-managed risk. Strong QF-Bench scores should not be read as a license to deploy models autonomously on production desks: even our top systems resolve only $\sim$62% of tasks at pass@1. We therefore caution against using aggregate pass rates as a proxy for production-readiness, and recommend continuous re-evaluation as models update, with mandatory human review on financial-invariant violations and convention errors (Section 5.3).

Data and dual-use considerations. All market data used by QF-Bench tasks is sourced from public or licensed sources and contains no personally identifiable information or proprietary trading data. Tasks, evaluation harness, and per-task scores are released under Apache 2.0; oracle reference implementations are released alongside the benchmark to enable reproducibility, with the trade-off that future training-data contamination is possible. We mitigate via canary strings and by privileging real-data tasks over closed-form recall, but cannot guarantee zero leakage—a limitation also noted in the preceding section. The benchmark itself does not enable any attack capability beyond what is already publicly available in finance textbooks and open-source libraries.

L GLOSSARY OF FINANCIAL TERMS

Task specifications in this benchmark are written in the register a quant would use on a desk. That is deliberate, decoding professional language into an exact computation is part of what the tasks test, so the specs are not simplified. This page is the on-ramp for everyone else.

It defines every financial term the benchmark relies on. Each entry gives a **Definition** (one precise sentence) and an **Intuition** (what it amounts to, and why it is checkable), with a machine-learning analogy wherever one genuinely clarifies rather than merely decorates.

A recurring theme is worth stating up front, because it explains what many of the verification checks are doing. A large share of these terms name **conventions** rather than mathematics, which day-count basis, which annualization constant, which sign an eigenvector carries. They are arbitrary but binding: two implementations can be mathematically equivalent and only one of them correct, and picking the wrong one raises no error and returns a plausible number. Another share name **invariants**, put-call parity, no-arbitrage bounds, variance ordering, relations that must hold however a number was computed, which is what makes them usable as checks when the true answer is unknown.

L.1 OPTIONS AND PRICING BASICS

Option (call / put). Definition: the right, not obligation, to buy (call) or sell (put) an asset at strike price K by expiry T; European = exercisable only at T, American = any time, Bermudan = on a fixed date grid.

Intuition: insurance contracts on price moves; the payoff $\max(S - K, 0)$ makes valuation an expectation over future scenarios, and Bermudan/American add an optimal-stopping problem.

Greeks (Δ, Γ, ν, Θ, ρ). Definition: partial derivatives of an option's value with respect to spot (delta), spot twice (gamma), volatility (vega), time (theta), and interest rate (rho).

Intuition: gradients of the pricing function, the outputs hedging actually consumes; "analytical Greeks" means the closed-form gradient, the reference against which numerical estimates are checked.

Black–Scholes model. Definition: the canonical closed-form pricing model for European options under geometric Brownian motion with constant volatility.

Intuition: the linear-regression of derivatives pricing, the baseline every desk and every textbook shares, with exact formulas for prices and Greeks.

Implied volatility / volatility surface. Definition: the volatility that makes the Black–Scholes price match an observed market price; collected across strikes and maturities it forms the vol surface.

Intuition: market prices re-expressed in a normalized coordinate system; fitting a model to the surface is constrained function approximation with no-arbitrage side conditions.

Put–call parity. Definition: the exact identity $C - P = S - Ke^{-rT}$ for European options on the same strike and expiry.

Intuition: a metamorphic test, it must hold no matter how prices were computed, so violations prove an implementation wrong without knowing the right answer.

No-arbitrage bounds. Definition: inequalities any internally consistent price set must satisfy (e.g., American $\geq$ European; Asian $\leq$ European for the same terms; calls decreasing in strike).

Intuition: feasibility constraints, like conservation laws, model-free invariants the verifier can check independently of the oracle's modeling choices.

Variance ordering. Definition: theory-mandated inequalities between variances rather than between prices: a more efficient estimator must show lower variance than a less efficient one (pathwise vs. likelihood-ratio Greeks; range-based vs. close-to-close volatility estimators), and a process's dispersion must grow with horizon in the prescribed way ($\mathrm{Var}(X_\tau)$ non-decreasing in τ, bounded above by the stationary variance for a mean-reverting process).

Intuition: an ordering constraint checkable without the right answer, like asserting that a variance-reduced Monte Carlo estimator actually has lower variance than the plain one. The ranking must hold whatever the numbers are, so a violation localizes an implementation error even when every individual value looks plausible.

Exotic / path-dependent options (barrier, Asian, lookback, compound, spread, digital, cliquet). Definition: payoffs depending on the price path or several assets: knock-in/out at a barrier level; averages (Asian); running maxima/minima (lookback); options on options (compound, Geske); differences of two assets (spread, Kirk/Margrabe); binary payouts (digital); periodically resetting caps (cliquet).

Intuition: the payoff depends on trajectory state, not just the endpoint, more state to track, fewer closed forms, more convention traps.

Risk-neutral measure / martingale property. Definition: the probability measure under which discounted asset prices have zero drift, making price = expected discounted payoff.

Intuition: a reweighting of scenarios that turns pricing into computing an expectation; "discounted price is a martingale" is a testable zero-drift invariant.

L.2 Models, Calibration, and Numerical Methods

Calibration. Definition: choosing model parameters so model prices reproduce quoted market instruments, typically by minimizing squared pricing error subject to constraints, with convergence and RMSE diagnostics.

Intuition: constrained curve-fitting where the loss is market fit and the convergence diagnostics are part of the deliverable, not an afterthought.

Hull–White model. **Definition:** a one-factor short-rate model $dr = [\theta(t) - ar]dt + \sigma\, dW$ whose time-varying drift $\theta(t)$ is fitted so the model reproduces today's discount curve exactly.

Intuition: a mean-reverting Gaussian model of the interest rate with enough flexibility to match the observed yield curve before pricing anything else.

Heston model / characteristic function. **Definition:** a stochastic-volatility model where variance follows its own mean-reverting square-root process; European prices come semi-analytically by Fourier inversion of the model's characteristic function.

Intuition: heteroskedasticity promoted to a latent state variable; pricing via integral transforms instead of simulation.

Dupire local volatility. **Definition:** the unique state-dependent diffusion coefficient $\sigma(K, T)$ consistent with an entire observed call-price surface, extracted via Dupire's formula from surface derivatives.

Intuition: a nonparametric inverse problem, differentiate a fitted price surface to recover the volatility field that would generate it.

Jump-diffusion (Merton) / OU / CIR processes. **Definition:** extensions of Brownian dynamics: Poisson jumps superimposed on diffusion (Merton); mean-reverting Gaussian (Ornstein–Uhlenbeck); mean-reverting square-root, nonnegative (Cox–Ingersoll–Ross).

Intuition: the standard SDE vocabulary, fat tails via jumps, mean reversion for rates/spreads/commodities, positivity where the quantity can't go negative.

Monte Carlo pricing; pathwise vs likelihood-ratio Greeks. **Definition:** pricing by simulating many paths; Greeks estimated either by differentiating the payoff along paths (pathwise / IPA) or by differentiating the sampling density (likelihood-ratio / score function).

Intuition: the same estimator dichotomy as ML, pathwise is the reparameterization trick, likelihood-ratio is REINFORCE; the pathwise estimator has lower variance where it applies but fails for kinked payoffs (e.g., digital indicators), exactly the trade-offs the tasks verify.

Finite differences: Crank–Nicolson, PSOR, early-exercise boundary. **Definition:** solving the pricing PDE on a grid with an implicit second-order scheme (Crank–Nicolson); American early exercise turns it into a linear complementarity problem solved by projected successive over-relaxation (PSOR); the early-exercise boundary $S^*(t)$ separates hold from exercise regions.

Intuition: a PDE solver plus an obstacle constraint, the option value may never fall below immediate exercise value, and the algorithm must track where that constraint binds.

Trinomial tree / Arrow–Debreu prices. **Definition:** a lattice discretization of the rate or price process; the Arrow–Debreu price of a node is today's value of receiving one unit in that node and nothing elsewhere; their per-date sums must reproduce the discount curve.

Intuition: a discrete state-space model where AD prices are discounted state-occupancy weights, "$\sum Q(\text{node}) = \text{bond price}$" is a sharp internal consistency check.

Richardson extrapolation / convergence order. **Definition:** combining solutions at two grid resolutions to cancel the leading error term and estimate the scheme's empirical convergence rate.

Intuition: standard numerical-analysis practice; the benchmark checks that refinement behaves as theory predicts, not just that one grid "looks right."

L.3 Rates and Curves

Discount curve / zero-coupon bootstrapping. Definition: the function P(0, t) giving today's value of one unit paid at t; bootstrapping recovers it sequentially from quoted instruments of increasing maturity.

Intuition: recursive curve-fitting where each instrument pins down the next segment, order and day-count conventions matter at every step.

OIS / swap curve. Definition: the discount curve built from overnight-indexed swaps, the post-2008 market standard for collateralized discounting.

Intuition: the "risk-free curve" desks actually use; building it is a multi-instrument bootstrap with its own conventions.

Caplet / cap / floor. Definition: a caplet is a call option on a floating interest rate over one accrual period; caps/floors are portfolios of caplets/floorlets; the "caplet vol surface" is their implied-volatility grid.

Intuition: per-period insurance against rate moves, the market data Hull–White is calibrated to.

Swaption (European / Bermudan). Definition: an option to enter an interest-rate swap at a future date (European) or on a schedule of dates (Bermudan, priced by backward induction with optimal stopping).

Intuition: the workhorse rates derivative; Bermudan pricing is dynamic programming on a tree.

DV01 / duration / immunization. Definition: DV01 is the price change for a one-basis-point parallel shift in rates; duration is the corresponding relative sensitivity; immunization constructs a portfolio whose net rate sensitivity is zero.

Intuition: first-order Taylor sensitivities and gradient-matching, computed by bump-and-reprice, which is finite-difference differentiation of the whole pipeline.

Day-count conventions (ACT/360, 30/360, ACT/365). Definition: market rules for converting calendar spans into year fractions for interest accrual.

Intuition: unit conventions, arbitrary but binding; choosing 365 where the market uses 360 misstates accruals by $\sim 1.4\%$ everywhere, silently.

L.4 Risk Measurement

VaR / CVaR (expected shortfall). Definition: Value-at-Risk is a quantile of the loss distribution at confidence α; CVaR is the expected loss beyond that quantile.

Intuition: a distributional quantile and its conditional tail mean; CVaR is the coherent (subadditive) one, and confidence-level and sign conventions are classic silent-error territory.

GARCH(1,1) / stationarity $\alpha + \beta < 1$. Definition: a recursion where today's conditional variance is a weighted combination of yesterday's squared shock (α) and yesterday's variance (β); $\alpha + \beta < 1$ guarantees a finite long-run variance.

Intuition: exponential smoothing of variance with mean reversion; the stationarity condition is a checkable invariant of any fitted model.

DCC-GARCH. Definition: Dynamic Conditional Correlation GARCH, univariate GARCH per asset plus a time-varying correlation matrix layer, yielding a full dynamic covariance.

Intuition: a two-stage covariance model: volatilities first, correlation dynamics second, long pipelines with convention choices at each stage.

EWMA covariance. Definition: exponentially weighted moving-average covariance (RiskMetrics-style, decay λ).

Intuition: momentum-style decay weighting of past outer products, the simplest dynamic covariance baseline.

EVT-POT (peaks over threshold). Definition: extreme-value method fitting a Generalized Pareto Distribution to exceedances above a high threshold to estimate tail risk.

Intuition: fit only the tail, with the distribution family asymptotic theory says tails must follow; threshold choice is the convention-laden hyperparameter.

Realized / OHLC volatility estimators. Definition: volatility estimated from observed price paths; open-high-low-close estimators (Parkinson, Garman–Klass, Rogers–Satchell) use intraday ranges with known relative efficiencies.

Intuition: several unbiased-ish estimators of the same latent quantity with a known ordering and efficiency hierarchy, the ordering itself is a verifiable invariant.

Volatility targeting / risk parity / CTA. Definition: scaling exposure so realized portfolio volatility tracks a target; allocating so each asset contributes equal risk; "CTA" denotes systematic trend-following futures strategies.

Intuition: normalization schemes over the covariance structure, the traps are in *which* volatility estimate feeds the scaler (the composite-volatility aggregator convention).

Absorption ratio / Marchenko–Pastur. Definition: the fraction of total variance captured by the top-k covariance eigenvectors; Marchenko–Pastur gives the eigenvalue distribution of a pure-noise covariance, used to separate signal from noise.

Intuition: explained-variance ratio of top principal components, judged against the random-matrix null, using the wrong "absorption" formula is a pure convention error that changes capital numbers.

L.5 Factor Research and Econometrics

Fama–French factors. Definition: canonical return factors (market, size, value, plus extensions) built from characteristic-sorted portfolios; attribution regresses asset returns on them.

Intuition: the standard linear feature set for equity returns; the details (breakpoints, rebalancing calendar) are conventions, not mathematics.

IPCA (instrumented PCA). Definition: a latent-factor model in which factor loadings are linear functions of observable characteristics, estimated jointly.

Intuition: PCA where the loadings are parameterized by features, a bilinear model bridging latent factors and interpretable characteristics.

PCA sign anchoring. Definition: the industry rule fixing each eigenvector's sign (e.g., positive loading on a designated reference series or on the first estimation window).

Intuition: eigenvectors are sign-indeterminate; the industry pins the gauge so factor series are comparable across windows, a pure convention no optimizer can discover.

Newey–West standard errors. Definition: heteroskedasticity-and-autocorrelation-consistent standard errors with a bandwidth (lag) parameter chosen by convention.

Intuition: autocorrelation-robust error bars; the bandwidth choice is exactly the kind of "defensible-but-nonstandard" knob the benchmark pins.

Momentum / residual momentum / double sort. Definition: return-continuation strategies ranked on past returns (momentum) or on residuals from a factor regression (residual momentum); double sorts form portfolios by two characteristics simultaneously.

Intuition: feature engineering plus stratified backtesting; lookback windows, skip months, and rebalance dates are all conventions.

Annualization ($\sqrt{252}$). Definition: scaling daily statistics to annual by trading-day count (252), with $\sqrt{\cdot}$-scaling for volatility.

Intuition: unit conversion with a convention constant, 252 vs 365 vs 260 silently rescales every reported number.

Look-ahead bias. Definition: using information not yet available at the simulated decision time (future prices, restated data, delayed filings).

Intuition: label leakage, time-series edition; task review audits for it specifically).

Brinson attribution. Definition: decomposing a portfolio's active return versus a benchmark into allocation and selection effects by sector.

Intuition: an additive ablation accounting of where performance came from.

L.6 Credit

Credit spread / z-spread. Definition: the yield premium of a defaultable bond over the risk-free curve; the z-spread is the constant shift to the discount curve that reprices the bond exactly.

Intuition: default risk expressed as a curve shift, which "spread" convention is meant is itself a frequent silent error.

Rating migration matrix / CreditMetrics. Definition: the transition-probability matrix over rating classes per horizon; CreditMetrics simulates correlated rating migrations to obtain a portfolio credit-loss distribution.

Intuition: a Markov chain over ratings plus a correlated sampler, credit VaR is the tail of the resulting simulated loss distribution.

Copulas / rank correlation. Definition: a copula couples fixed marginal distributions into a joint distribution, separating dependence from marginals; rank correlations (Kendall, Spearman) are the invariant dependence measures used to calibrate them.

Intuition: dependence modeling factored out from marginal modeling, sampling and fitting them correctly is mostly about respecting which correlation is which.

L.7 Market Structure and Data

CRSP / SEC EDGAR / 13F / Form 4 / 8-K / 10-K. Definition: CRSP is the standard US equity returns database; EDGAR is the SEC's filing repository; 13F = quarterly institutional holdings, Form 4 = insider transactions, 8-K = material events, 10-K = annual reports.

Intuition: the canonical public data plumbing of US markets, each filing type has its own timing, amendment, and restatement quirks the tasks exploit.

Limit order book (LOB). Definition: the standing set of buy and sell orders by price level; its shape and imbalance drive short-horizon price dynamics.

Intuition: the queue state of the market, microstructure signals are features computed from it.

TCA / participation rate. Definition: transaction-cost analysis benchmarks realized execution prices against references (arrival, VWAP); participation rate is the executed fraction of market volume per interval.

Intuition: grading an execution schedule against counterfactual benchmarks, conventions decide the denominator.

Funding rate / basis / carry. **Definition:** the periodic payment tying perpetual futures to spot (crypto funding); the price gap between related instruments (basis); the return earned from holding a position absent price moves (carry).

Intuition: arbitrage-linked spreads whose sign and accrual conventions determine strategy PnL.

Cross-currency (xccy) basis. **Definition:** the spread added to one leg of a cross-currency swap so that FX-hedged funding in two currencies prices consistently.

Intuition: the market's measured deviation from textbook covered-interest parity, mark-to-market conventions on the notional resets are the trap.

Corporate actions. **Definition:** splits, dividends, spin-offs and similar events requiring adjustment of historical prices and share counts.

Intuition: data-normalization events; mishandling one silently corrupts every downstream return.