QuantEval: A Benchmark for Financial Quantitative Tasks in Large Language Models

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have shown strong capabilities in finance-oriented question answering, yet their evaluation on professional quantitative workflows remains fragmented and often stops at factual or document-grounded QA. We introduce **QuantEval**, a benchmark for evaluating whether LLMs can complete the compute–implement–validate loop required in quantitative finance. QuantEval contains 1,575 expert-validated instances across three complementary dimensions: (i) *knowledge-based QA*, (ii) *quantitative mathematical reasoning*, and (iii) *quantitative strategy coding*. The coding task is evaluated with a deterministic CTA-style backtesting framework that executes model-generated strategies and compares risk–return metrics against expert references, moving beyond syntax or template matching. We evaluate 13 open-source and proprietary LLMs, report prompt and reliability analyses, and observe substantial gaps relative to human experts, especially in multi-step reasoning and framework-constrained coding. We further conduct supervised fine-tuning and reinforcement learning experiments on domain-aligned data, showing that training can improve quantitative reasoning while also exposing trade-offs in factual recall. QuantEval releases the dataset, evaluation scripts, and full backtesting configuration to support reproducible research on LLMs for quantitative finance.

1 Introduction

Large Language Models (LLMs) have shown strong capabilities in reasoning, code generation, and domain-specific question answering (Zhong et al., 2025; Shao et al., 2024; OpenAI, 2024). These capabilities have motivated growing interest in financial applications, where practitioners rely on quantitative analysis, mathematical modeling, and algorithmic strategy development.

However, evaluating LLMs in financial quantitative workflows remains limited and fragmented. Existing financial benchmarks mainly focus on knowledge-based question answering (QA), report understanding, and numerical QA over documents or tables (Xie et al., 2023; Zhu et al., 2024; Chen et al., 2024, 2022; Xie et al., 2024). These benchmarks are useful for assessing financial familiarity and factual understanding, but they rarely evaluate the full process required in quantitative finance. Real-world quant workflows require practitioners to apply financial concepts, perform multi-step calculations with formulas and market data, and implement strategies that can be validated through backtesting.

This gap limits our ability to assess whether LLMs can move beyond financial recall to quantitative problem solving and implementation. Quantitative tasks such as option pricing, portfolio optimization, risk assessment, and algorithmic trading strategy development require models to integrate domain knowledge with numerical precision, unit consistency, and coding proficiency. Moreover, evaluating strategy generation cannot rely solely on syntax correctness: a strategy may compile but still be financially meaningless, unstable, or risk-dominant. Therefore, benchmark design for quantitative finance should include execution-based evaluation that measures strategy behavior under a standardized and reproducible backtesting environment.

To address these limitations, we introduce **QuantEval**, a benchmark that evaluates LLMs across three key dimensions of quantitative finance: (1) *Knowledge-based QA*, measuring conceptual understanding and terminology; (2) *Quantitative Mathematical Reasoning*, assessing multi-step quantitative computations grounded in real market data and financial theory; and (3) *Quantitative Strategy Coding*, testing whether models can generate framework-compatible and executable trading

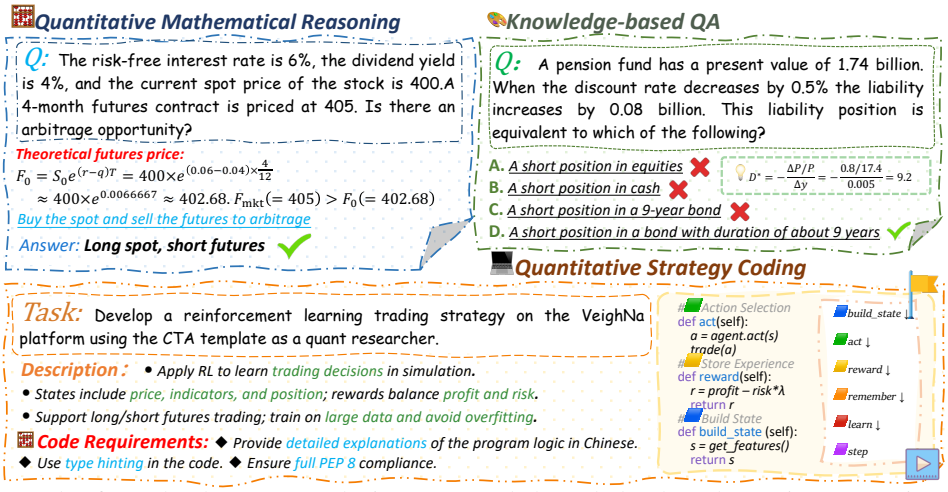

Figure 1: Examples from the three core tasks in QuantEval: knowledge-based question answering, quantitative mathematical reasoning, and quantitative strategy coding.

strategies whose backtest outcomes match expert-validated reference implementations. Figure 1 illustrates representative examples from the three task types. QuantEval contains 1,575 carefully curated samples across these dimensions, with explicit task formulations, unified prompting protocols, expert validation, and deterministic execution settings.

A key feature of QuantEval is the use of an integrated CTA-style backtesting framework for strategy coding. This enables rigorous assessment beyond pass/fail compilation, by measuring execution success rate and deviations in key financial metrics. We evaluate 13 open-source and proprietary LLMs on QuantEval and observe substantial performance gaps relative to human experts, especially for reasoning and strategy coding.

Motivated by these findings and broader reliability concerns about statistical uncertainty, prompt dependence, framework specificity, and possible data leakage, we further conduct a set of diagnostic experiments: statistical uncertainty analysis, automatic-grading agreement, prompt robustness, framework-interface sensitivity, strategy-coding failure taxonomy, human implementation variance, leakage diagnostics, cross-benchmark comparison, and exploratory training ablations. We release the dataset, evaluation scripts, and a standardized CTA backtesting harness to enable reproducible evaluation.

In summary, our contributions are:

- We present **QuantEval**, a benchmark for evaluating the compute–implement–validate loop in quantitative finance, covering knowledge QA, quantitative mathematical reasoning, and executable strategy coding.
- We construct QuantEval through a scalable pipeline that combines expert annotation, multi-agent generation, and expert validation, and we evaluate coding tasks with a deterministic CTA-style backtesting framework and finance-specific risk–return metrics.
- We evaluate 13 open-source and proprietary LLMs under a unified protocol, and contextualize model comparisons through reliability-oriented diagnostics for uncertainty, grading consistency, prompt and interface sensitivity, human variation, and leakage risk.
- We study domain-aligned training with supervised fine-tuning, reinforcement learning, and training-stage ablations, showing gains in quantitative reasoning while identifying persistent weaknesses in factual recall and framework-constrained strategy implementation.

2 QuantEval

2.1 Overview

As shown in Figure 2, QuantEval evaluates LLMs on core financial quantitative competencies. Compared with prior financial benchmarks that primarily emphasize knowledge-centric QA, QuantEval explicitly targets three dimensions that collectively define quantitative finance workflows:

1. **Knowledge-based QA**: conceptual understanding of financial definitions, terminology, and theory;
2. **Quantitative Mathematical Reasoning**: multi-step quantitative problem solving grounded in real market data and financial formulas;
3. **Quantitative Strategy Coding**: generating executable trading strategies under a unified

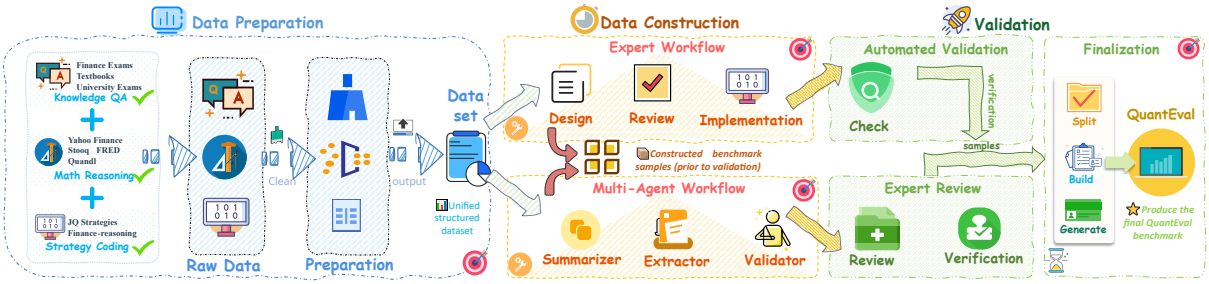

Figure 2: The QuantEval data construction pipeline consists of three main stages: Data Preparation, Data Construction, and Validation.

CTA-style backtesting framework and matching expert-validated performance metrics.

Task formulation. We formalize the three tasks as follows. For **Knowledge-based QA**, the input is a financial question with optional answer choices, and the output is a final answer option or short textual answer; correctness is measured by exact option matching or normalized semantic matching. For **Quantitative Mathematical Reasoning**, the input contains a financial scenario, market variables, formulas or implicit domain assumptions, and optional answer choices; the output must provide the final numerical or categorical answer, and evaluation uses accuracy after normalization of units, percentages, and numeric precision. For **Quantitative Strategy Coding**, the input is a natural-language strategy specification plus required CTA framework constraints; the output is Python strategy code that must conform to the prescribed interfaces. Correctness is assessed by end-to-end execution under the fixed backtesting configuration and by the deviation between generated and expert-reference risk–return metrics.

Design rationale. Each task is aligned with a limitation of prior evaluation. Knowledge QA measures domain familiarity, reasoning tasks test whether models can combine formulas with market quantities, and strategy coding tests whether models can turn financial ideas into executable behavior. The multi-agent construction pipeline mirrors this decomposition: summarization and extraction organize financial materials, generation creates task candidates, and validation checks formula correctness, interface compliance, and financial plausibility before expert review.

2.2 Data Preparation

The initial stage collects and organizes source materials for subsequent question and code generation across the three task categories. Detailed data sources, time spans, and licenses for each task category are summarized in Appendix B.

Knowledge-based QA. For this dimension, authoritative textual materials are sourced from financial textbooks, peer-reviewed academic papers, regulatory filings, and curated glossaries of financial terminology. Quantitative finance experts review these materials and select passages covering essential concepts, theoretical frameworks, and domain-specific vocabulary. To ensure comprehensive coverage and mitigate potential biases, the materials span multiple subdomains.

Quantitative Mathematical Reasoning. This dimension requires datasets rooted in realistic financial computations and problem-solving scenarios. To this end, extensive historical market data and macroeconomic indicators are gathered from reputable public sources. Complementing these data, problem templates are extracted from quantitative finance coursework, professional certification materials (e.g., CFA, FRM), and industry whitepapers. These templates are enriched with real market data to create contextually rich problems that require calculations involving option pricing models, risk metrics, portfolio optimization, and other quantitative finance techniques.

Quantitative Strategy Coding. For strategy coding, algorithmic trading strategies and backtesting scripts are collected from open-source repositories and academic publications. These codebases encompass a wide spectrum of quantitative approaches. Domain experts then curate, annotate, and rewrite the collected materials into non-proprietary CTA-style benchmark tasks inspired by industry workflows. Alongside the strategy code, metadata including strategy parameters, historical performance metrics, and market conditions are collected to support comprehensive evaluation and benchmarking. In this benchmark, we focus on equity CTA-style backtesting, where strategies trade

a fixed universe of U.S. ETFs and large-cap stocks under standardized execution and risk constraints. A full specification of the CTA backtesting environment is provided in Appendix C.

2.3 Data Construction

The second stage focuses on generating the actual benchmark samples through a hybrid approach that combines expert annotation with automated multi-agent generation. This dual approach balances the need for high-quality, domain-accurate content with scalability and diversity.

Generation with Experts. Financial domain experts play a pivotal role in crafting high-quality samples. Their responsibilities include revising and refining existing questions to enhance clarity, relevance, and difficulty, ensuring that distractors are plausible yet unambiguously incorrect, and that questions demand genuine understanding rather than superficial recall. Experts also design novel questions and problems based on curated texts, market data, and synthetic scenarios.

For Quantitative Strategy Coding tasks, experts specify challenges such as strategy implementation, partial code completion, debugging, and parameter tuning, ensuring that the tasks reflect realistic quantitative trading workflows. Additionally, experts review and annotate quantitative strategy code samples to verify correctness, readability, and alignment with benchmark objectives, providing detailed explanations and expected outputs to facilitate evaluation. To further enhance the dataset's complexity and practical relevance, synthetic quantitative scenarios are designed by combining real market data with hypothetical events or constraints. These scenarios simulate market shocks, regulatory changes, portfolio rebalancing under risk limits, and other realistic financial decision-making contexts.

Multi-Agent Automated Construction. Inspired by expert workflows, a multi-agent automated construction framework is developed to efficiently scale data generation, particularly for mathematical reasoning and coding tasks. This framework comprises specialized roles that collaborate to produce high-quality samples.

1) A summarizer agent analyzes financial texts and market data. 2) A problem extractor agent identifies relevant data segments and problem templates. 3) A generator agent produces mathematically rigorous questions and executable code snippets. This agent operates under detailed prompt templates and is guided by examples derived from expert annotations to ensure adherence to domain standards. 4)Finally, a validation agent performs preliminary checks on generated samples, including syntactic correctness, logical consistency, and alignment with task requirements.

This multi-agent system enables efficient, large-scale generation of diverse and high-quality benchmark samples while maintaining domain relevance and rigor. We provide a breakdown of expert-generated vs agent-generated samples and their quality comparison in Appendix E.4.

2.4 Validation

We validate QuantEval samples for reliability, rigor, and practical relevance. To this end, a multi-tiered validation process involving both automated agents and domain experts is implemented.

The automated validation agent conducts initial screening to detect duplicate or near-duplicate questions, inconsistencies, and trivial or overly simplistic items. For knowledge QA and reasoning tasks, it verifies that questions cannot be answered correctly through superficial keyword matching or simple heuristics, thereby ensuring that genuine understanding and reasoning are required. For coding tasks, the agent checks code executability, correctness of outputs, and adherence to specified constraints, including successful execution within the integrated CTA backtesting framework. Samples failing these criteria are either revised or discarded to maintain dataset quality.

Domain experts perform thorough reviews of all benchmark samples. Their evaluation focuses on confirming factual accuracy, mathematical soundness, and code validity. Experts ensure that questions and tasks are unambiguous, well-phrased, and aligned with the intended difficulty levels. They also conduct ethical and sensitivity checks to verify that samples do not contain proprietary, sensitive, or ethically problematic content. When necessary, experts request revisions or reject samples that do not meet stringent quality standards.

This iterative feedback loop between validation and construction stages continues until the dataset achieves the desired quality threshold, ensuring that QuantEval is both challenging and trustworthy.

2.5 De-duplication, leakage prevention and difficulty control

To ensure benchmark reliability and minimize training-data contamination, we apply strict de-

duplication and leakage prevention procedures at both the question and code levels. We additionally control task difficulty and enforce balanced coverage across quantitative finance subdomains. Full details of our de-duplication criteria, overlap scanning protocols, and difficulty/coverage design are provided in Appendix D.

2.6 Implementation Details

We used GPT-4o-assisted drafting and expert review to construct candidate questions. All retained instances are subsequently reviewed by domain experts. We include leakage diagnostics and expert validation to reduce potential generator-model bias.

A key feature of the quantitative strategy coding dimension is the integration of a CTA backtesting framework. This framework enables the generation of realistic trading scenarios and the validation of code correctness through simulated executions on historical market data. The backtesting environment supports evaluation of strategy performance metrics, risk characteristics, and robustness under varying market conditions. By grounding the benchmark in this industry-recognized methodology, QuantEval ensures that the coding tasks reflect authentic quantitative finance workflows.

2.7 Data Statistics

QuantEval contains 1,575 samples: *Knowledge-based QA* (660), *Quantitative Mathematical Reasoning* (855), and *Quantitative Strategy Coding* (60). QA and reasoning samples include both multiple-choice and fill-in-the-blank formats, covering conceptual recall, numerical calculation, and interpretation.

Each coding sample pairs a problem description with an expert-validated reference strategy and backtesting metrics. The subset is smaller because each instance requires reference implementation, deterministic execution, metric verification, and expert review; accordingly, we interpret fine-grained differences cautiously and focus on robust patterns in Section 3.2.

3 Evaluation on QuantEval

This section presents the evaluation protocol and main results of QuantEval across three tasks: Knowledge-based QA, Quantitative Mathematical Reasoning, and Quantitative Strategy Coding. We emphasize two principles: (i) prompting consistency for fair comparison across models and settings, and (ii) execution-based evaluation for quantitative strategy coding, where correctness must be validated via backtesting rather than surface-level code inspection.

Metric in Table 2	Goal	Definition / Interpretation
Executable Rate (%)	↑	Fraction of generated strategies that execute end-to-end in the CTA framework with valid interfaces and no runtime failure.
Return MAE	↓	Mean absolute error between generated and GT annualized return, computed over executable instances.
Drawdown MAE	↓	Mean absolute error between generated and GT maximum drawdown.
Sharpe MAE	↓	Mean absolute error between generated and GT Sharpe ratio.
Ret/Draw MAE	↓	Mean absolute error between generated and GT return-to-drawdown ratio, capturing profitability versus downside risk.

Table 1: **Strategy Coding metrics used in Table 2**. MAE is computed only over executable outputs; non-executable models are reported as "–". Return/Drawdown MAE are in percentage points (pp); Sharpe and Ret/Draw MAE are unitless.

3.1 Evaluation Setup

We evaluate 13 state-of-the-art LLMs spanning open-source and proprietary families, including Qwen3 variants (4B, 8B, 14B, 30B) (Yang et al., 2025), DeepSeek distilled models (DeepSeek-AI et al., 2025), DianJin-R1-7B (Zhu et al., 2025), Claude Sonnet 4.5 (Anthropic, 2024), Gemini 2.5 Pro (Comanici et al., 2025), and GPT-5 (OpenAI, 2025). Unless otherwise specified, all main leaderboard results use deterministic decoding with temperature$= 0$ and top-$p = 0.95$; stochastic decoding is used only in robustness analyses. We accessed proprietary models via official APIs between Sep 2025 and Dec 2025.

Knowledge QA & Quantitative Reasoning. For Knowledge QA and Quantitative Reasoning, we report **accuracy** (%) under two settings: **with Chain-of-Thought prompting (w/ CoT)** and **without CoT (w/o CoT)**. We extract final options for multiple-choice questions and apply normalization plus semantic matching for open-ended answers. Small differences within approximate binomial uncertainty are treated as descriptive; confidence intervals, bootstrap uncertainty, grading agreement, and prompt templates are provided in Appendix F.1–F.3.

Strategy Coding (CoT-only). For Strategy Coding, models generate strategies executable under a unified CTA-style backtesting framework. Since pilot direct-answer prompting yielded near-zero executability, we report the main coding leaderboard

Model	Knowledge QA (Acc.%)			Reasoning (Acc.%)			Strategy Coding (CoT-only)				
	w/	w/o	Δ	w/	w/o	Δ	Exec. Rate↑	Return MAE↓	Drawdown MAE↓	Sharpe MAE↓	Ret/Draw MAE↓
Human Expert	**91.75**	**91.75**	0.0	**89.05**	**89.05**	0.0	**100**	**1.52**	**1.10**	**0.07**	**0.09**
Open-Source Models											
Qwen3-4B	64.4	69.0	-4.6	27.7	33.9	-6.2	0.0	–	–	–	–
Qwen3-8B	65.5	65.5	0.0	33.9	29.2	+4.7	0.0	–	–	–	–
Qwen3-14B	75.9	78.2	-2.3	35.4	33.9	+1.5	0.0	–	–	–	–
Qwen3-30B-A3B	48.5	44.5	+4.0	40.5	36.5	+4.0	8.3	2.71	20.01	0.30	0.40
DeepSeek-R1-Distill-Qwen-1.5B	37.2	34.0	+3.2	30.0	27.9	+2.1	0.0	–	–	–	–
DeepSeek-R1-Distill-Qwen-7B	42.5	49.4	-6.9	21.5	26.2	-4.7	0.0	–	–	–	–
DeepSeek-R1-Distill-Qwen-14B	66.7	63.2	+3.5	29.2	32.3	-3.1	0.0	–	–	–	–
DeepSeek-R1-Distill-Llama-8B	50.6	41.4	+9.2	10.8	10.4	+0.4	0.0	–	–	–	–
DeepSeek-R1-671B	52.7	48.1	+4.6	44.0	40.8	+3.2	0.0	–	–	–	–
DianJin-R1-7B	50.3	46.0	+4.3	42.8	38.0	+4.8	0.0	–	–	–	–
Proprietary Models											
Claude Sonnet 4.5	86.0	90.8	-4.8	43.1	44.6	-1.5	65.0	9.86	12.05	0.16	0.20
Gemini 2.5 Pro	66.7	66.7	0.0	38.0	23.0	+15.0	65.0	12.31	14.82	0.15	0.24
GPT-5	83.9	82.8	+1.1	55.0	33.8	+21.2	53.3	7.48	9.35	0.17	0.17

Table 2: Unified evaluation on **QuantEval**. For Knowledge QA and Reasoning, we report accuracy (%) with Chain-of-Thought prompting (w/), without CoT (w/o), and the performance gap Δ = w/ − w/o. For Strategy Coding, we evaluate under CoT prompting and report Executable Rate (%) and MAE of four backtesting metrics. Return MAE and Drawdown MAE are reported in **percentage points (pp)**; models with 0% executable rate are marked as "–" for MAE.

under CoT prompting. Each strategy is executed on aligned historical data and evaluated by executable rate and four risk–return metrics (Table 1).

The fixed CTA interface is an intentional reproducibility constraint, testing both financial logic and framework compliance. We discuss framework lock-in in the Limitations section and provide prompt/interface ablations in Appendix F.4 and Appendix F.5.

Human Baseline. Human experts answer QA/reasoning questions and implement coding tasks under the same task statements and backtesting harness. Appendix E reports per-expert accuracy, coding variance, and background information; the coding variance provides a practical MAE noise floor.

3.2 Main Results on QuantEval

Table 2 reports the main results. We focus on robust patterns rather than small differences: one- to three-point QA/reasoning gaps may fall within subset-level uncertainty, while larger gaps such as GPT-5's CoT gain or the human–model gap are qualitatively stable. For Strategy Coding, we emphasize structural executable-rate differences and consistent metric deviations rather than marginal MAE changes. Appendix F.1, F.6, and F.5 report uncertainty, coding failures, and CTA-interface sensitivity.

A substantial gap remains to human experts. Human experts achieve strong performance on both Knowledge QA and Reasoning. In contrast, the best-performing LLMs remain behind, especially on quantitative reasoning. This confirms that QuantEval requires not only surface-level financial familiarity but also precise multi-step quantitative computation and robust reasoning.

CoT prompting helps reasoning but is not always beneficial. CoT yields the largest gains on quantitative reasoning for several models, but can reduce accuracy on factual QA. This suggests that CoT helps when structured computation is needed, yet may introduce unnecessary assumptions for recall-oriented questions. We therefore view CoT as a task-dependent evaluation condition rather than a uniformly stronger setting.

Reasoning-oriented training can trade off with factual recall. Table 3 shows a small Knowledge QA drop after GRPO compared with SFT, despite reasoning and coding gains. We interpret this as a tension between recall-oriented objectives and reasoning-oriented optimization, motivating methods that preserve factual calibration while improving multi-step computation.

Strategy coding remains the hardest dimension. Most open-source models fail to generate executable strategies under strict CTA framework constraints, resulting in 0% executable rate across many models. Even for large open-source models, performance remains limited. In contrast, proprietary models substantially improve executability.

Training Stage	Knowledge QA	Reasoning	Strategy Coding (Sharpe MAE)↓
Initial checkpoint	50.3	42.8	–
After SFT	**53.2**	47.8	0.85
After SFT + GRPO RL	52.2	**48.4**	**0.72**

Table 3: Performance progression of DianJin-R1-7B under extended training. Accuracy (%) is reported for QA and reasoning. For strategy coding, we report **Sharpe MAE** (lower is better).

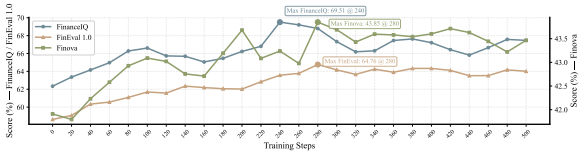

Figure 3: Diagnostic RL-only performance curves of DianJin-R1-7B on external financial benchmark test sets. These curves illustrate optimization stability.

Closed-source models lead overall, but open-source models show partial strengths. Closed-source models lead overall, especially on strategy coding. Open-source models show relatively competitive performance on knowledge QA, suggesting that factual knowledge and basic finance concepts can be captured by open models, whereas execution-based strategy coding remains a major bottleneck.

4 Analysis and Discussion

We further analyze benchmark difficulty, cross-benchmark transfer, training effects, robustness to prompt and interface choices, and dominant failure modes.

4.1 Exploratory Training with Extended Financial Data

To explore effective approaches for improving LLM performance on QuantEval, we curate an extended training corpus of approximately 57,000 samples. It is sampled from three large-scale open-source financial datasets: Agentar-DeepFinance-100K (Zhao et al., 2025), DianJin-R1-Data (Zhu et al., 2025), and FinQA (Qian et al., 2025), plus an additional 7,000 samples constructed using our pipeline to match QuantEval formats.

We perform supervised fine-tuning (SFT) starting from DianJin-R1-7B, followed by reinforcement learning using Group Relative Policy Optimization (GRPO) (Shao et al., 2024). We incorporate two reward components: a *format reward* encouraging structured outputs and valid answer formatting, and an *accuracy reward* promoting correctness.

Table 3 reports the performance progression. SFT improves both Knowledge QA and Reasoning, while subsequent GRPO provides further gains in Reasoning and Strategy Coding but slightly reduces Knowledge QA. These results suggest that domain-aligned training improves structured quantitative reasoning, while the gap to human experts (89.05%) and factual-recall trade-offs remain substantial.

Figure 3 shows performance curves during GRPO reinforcement learning. We observe that benchmark performance improves steadily in the early phase and stabilizes later, indicating that the reward design effectively guides the model toward better structured reasoning and higher correctness. This supports the effectiveness of combining domain-aligned SFT with structured RL for finance reasoning tasks.

4.2 Robustness and Reliability Diagnostics

We conduct additional diagnostics to check whether the main conclusions are affected by statistical uncertainty, grading reliability, prompting, framework guidance, or stochastic decoding. The results show that small point differences should be interpreted cautiously, while the main patterns remain stable: the human–model gap, the CoT gains for strong reasoning models, and the executability gap between open-source and proprietary models. Detailed tables are provided in Appendix F.1, Appendix F.3, Appendix F.4, Appendix F.5, Appendix F.8, Appendix F.9, and Appendix F.6.

4.3 Comparison with Existing Financial Benchmarks

We compare representative models on FinanceIQ (Duxiaoman-DI), FinEval (Guo et al., 2024), Finova (Ant Group), and QuantEval, with detailed benchmark-design differences in Appendix A.

Table 4 shows that models generally score higher on FinanceIQ and FinEval than on QuantEval, with Finova closer in difficulty. This suggests that standard financial QA or document-style numerical reasoning does not directly transfer to quantitative tasks requiring multi-step computation, market grounding, and executable strategy development.

4.4 Error Analysis

We further analyze model outputs to identify common failure modes across tasks.

Model	Finance IQ	Fin Eval	Finova	QuantEval QA	QuantEval Reasoning
Qwen2.5-7B-Instruct	57.4	52.6	30.6	38.7	30.9
DianJin-R1-7B	62.3	58.7	41.9	50.3	42.8
DianJin-R1-7B + RL	69.2	64.2	43.7	51.8	44.1
DianJin-R1-7B + SFT	72.2	70.5	44.8	**53.2**	47.8
DianJin-R1-7B + SFT + RL	**77.9**	**72.7**	**45.5**	52.2	**48.4**

Table 4: Cross-benchmark evaluation of selected models on existing financial evaluation datasets and QuantEval. Accuracy (%) is reported for QA and reasoning tasks. QuantEval coding task is excluded here for comparability. QuantEval columns are highlighted to emphasize the larger difficulty gap relative to existing benchmarks.

Knowledge QA: conceptual confusion and hallucinated explanations. Most errors confuse related finance concepts. CoT can amplify them by adding unsupported assumptions, consistent with negative Δ for some models in Table 2; Appendix G gives an example.

Quantitative reasoning: arithmetic mistakes and formula composition failures. Models often identify relevant formulas but make arithmetic, unit-consistency, or answer-mapping errors. These near-misses explain why reasoning accuracy remains below 60% even for GPT-5; Appendix G gives a duration example.

Strategy coding: interface mismatch and execution failures dominate. Coding failures concentrate on interface/API mismatch, runtime errors, missing execution constraints, and financially plausible but metric-misaligned logic. Open-source models often fail before execution, while proprietary models execute more often but still deviate from human metrics, indicating that executable code does not guarantee financially aligned behavior. Appendix F.6 gives the full taxonomy and Appendix G a representative case.

5 Related Work

Financial question answering and knowledge benchmarks. Financial NLP benchmarks have mainly focused on knowledge retrieval, fact-based QA, and numerical reasoning over financial reports and tables (Chen et al., 2022; Zhu et al., 2021). FinQA and TAT-QA require models to extract evidence and perform calculations from structured or semi-structured financial documents, while broader benchmarks such as Pixiu/FinBen and CFLUE evaluate financial language understanding across multiple task types (Xie et al., 2023, 2024; Zhu et al., 2024). These benchmarks are valuable for testing financial knowledge and document-grounded reasoning, but they rarely evaluate whether models can complete quantitative workflows involving market-grounded computation and executable strategy implementation.

Quantitative reasoning in finance. General mathematical reasoning benchmarks evaluate multi-step problem solving (Amini et al., 2019), and recent financial reasoning datasets further adapt such evaluation to finance-specific settings (finmath, 2024; Qian et al., 2025; Zhu et al., 2025). However, many existing tasks remain close to QA or formula-based calculation. In practical quantitative finance, reasoning often involves market variables, risk metrics, option pricing, portfolio construction, and unit-sensitive numerical computation. QuantEval extends this line of work by combining knowledge QA with market-grounded quantitative reasoning and execution-based strategy evaluation.

Code generation and algorithmic trading. LLM code generation has been widely studied in general programming domains, where evaluation often focuses on functional correctness. In finance, recent work has explored LLMs for financial code generation and decision support (Huynh and Lin, 2025; Li et al., 2024). Algorithmic trading, however, imposes additional constraints: generated code must follow framework interfaces, respect risk controls, and produce meaningful behavior under backtesting. QuantEval differs from prior work by evaluating strategy code through a deterministic CTA-style backtesting framework and comparing generated strategies with expert references using risk–return metrics.

6 Conclusion

We introduce **QuantEval**, a benchmark for evaluating LLMs on financial quantitative tasks spanning **knowledge QA**, **quantitative reasoning**, and **strategy coding**. By integrating execution-based CTA backtesting with standardized risk–return metrics, QuantEval reveals large gaps between current LLMs and human experts, especially in multi-step reasoning and framework-constrained strategy implementation. Further analyses clarify prompting, interface, human-variance, leakage, and uncertainty effects. Domain-aligned SFT and RL improve reasoning but reveal trade-offs with factual recall. We release the dataset, scripts, and deterministic backtesting configuration to support reproducible research on LLMs for quantitative finance.

Limitations

QuantEval has several limitations. The Strategy Coding subset contains 60 instances, reflecting the high cost of expert specification, reference implementation, deterministic backtesting, and metric verification; it is therefore more suitable for identifying large structural gaps than for over-ranking models with small MAE differences. In addition, coding evaluation uses a fixed CTA-style framework to ensure reproducibility, which may conflate financial reasoning with API familiarity despite our framework-sensitivity analysis in Appendix F.5. Finally, QuantEval is currently text-based and English-centric, and Strategy Coding remains sensitive to prompt structure, so multilingual, multimodal, multi-framework, and broader prompt-robustness extensions are important future directions.

Ethics Statement

QuantEval is constructed from publicly available financial texts, market data sources, and expert-validated strategy templates. We ensure that no proprietary trading signals, private datasets, or confidential strategies are included. Strategy coding tasks are evaluated under offline historical backtesting and are intended solely for research purposes; they do not constitute financial advice and should not be used for direct trading decisions. We also include expert review to filter sensitive content and reduce potential harms from misuse.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Ant Group. Finova: A financial open-source platform. https://github.com/antgroup/Finova.

Anthropic. 2024. Claude 3 model card. Technical Report Version 1.0, Anthropic. Accessed: 2025-09-16.

Jian Chen, Peilin Zhou, Yining Hua, Loh Xin, Kehui Chen, Ziyuan Li, Bing Zhu, and Junwei Liang. 2024. FinTextQA: A dataset for long-form financial question answering. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6025–6047, Bangkok, Thailand. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2022. Finqa: A dataset of numerical reasoning over financial data. *Preprint*, arXiv:2109.00122.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Duxiaoman-DI. Financeiq dataset. https://huggingface.co/datasets/Duxiaoman-DI/FinanceIQ.

finmath. 2024. finmath-lib.

Xin Guo, Haotian Xia, Zhaowei Liu, Hanyang Cao, Zhi Yang, Zhiqiang Liu, Sizhe Wang, Jinyi Niu, Chuqi Wang, Yanhui Wang, Xiaolong Liang, Xiaoming Huang, Bing Zhu, Zhongyu Wei, Yun Chen, Weining Shen, and Liwen Zhang. 2024. Fineval: A chinese financial domain knowledge evaluation benchmark for large language models. *Preprint*, arXiv:2308.09975.

Nam Huynh and Beiyu Lin. 2025. Large language models for code generation: A comprehensive survey of challenges, techniques, evaluation, and applications. *Preprint*, arXiv:2503.01245.

Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. 2024. Large language models in finance: A survey. *Preprint*, arXiv:2311.10723.

OpenAI. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

OpenAI. 2025. Gpt-5 technical report. https://cdn.openai.com/gpt-5-system-card.pdf. Accessed September 24, 2025.

Lingfei Qian, Weipeng Zhou, Yan Wang, Xueqing Peng, Han Yi, Yilun Zhao, Jimin Huang, Qianqian Xie, and Jian yun Nie. 2025. Fino1: On the transferability of reasoning-enhanced llms and reinforcement learning to finance. *Preprint*, arXiv:2502.08127.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Qianqian Xie, Weiguang Han, Zhengyu Chen, Ruoyu Xiang, Xiao Zhang, Yueru He, Mengxi Xiao, Dong Li, Yongfu Dai, Duanyu Feng, Yijing Xu, Haoqiang Kang, Ziyan Kuang, Chenhan Yuan, Kailai Yang, Zheheng Luo, Tianlin Zhang, Zhiwei Liu, Guojun Xiong, and 15 others. 2024. Finben: A holistic financial benchmark for large language models. *Preprint*, arXiv:2402.12659.

Qianqian Xie, Weiguang Han, Xiao Zhang, Yanzhao Lai, Min Peng, Alejandro Lopez-Lira, and Jimin Huang. 2023. Pixiu: A large language model, instruction data and evaluation benchmark for finance. *Preprint*, arXiv:2306.05443.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Xiaoke Zhao, Zhaowen Zhou, Lin Chen, Lihong Wang, Zhiyi Huang, Kaiyuan Zheng, Yanjun Zheng, Xiyang Du, Longfei Liao, Jiawei Liu, Xiang Qi, Bo Zhang, Peng Zhang, Wei Wang, and Zhe Li. 2025. Agentar-deepfinance-100k: A large-scale financial dataset via systematic chain-of-thought synthesis optimization. *Preprint*, arXiv:2507.12901.

Tianyang Zhong, Zhengliang Liu, Yi Pan, Yutong Zhang, Yifan Zhou, Shizhe Liang, Zihao Wu, Yanjun Lyu, Peng Shu, Xiaowei Yu, Chao Cao, Hanqi Jiang, Hanxu Chen, Yiwei Li, Junhao Chen, Huawen Hu, Yiheng Liu, Huaqin Zhao, Shaochen Xu, and 56 others. 2025. Evaluation of openai o1: Opportunities and challenges of agi. *Preprint*, arXiv:2409.18486.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. *Preprint*, arXiv:2105.07624.

Jie Zhu, Qian Chen, Huaixia Dou, Junhui Li, Lifan Guo, Feng Chen, and Chi Zhang. 2025. Dianjin-r1: Evaluating and enhancing financial reasoning in large language models. *Preprint*, arXiv:2504.15716.

Jie Zhu, Junhui Li, Yalong Wen, and Lifan Guo. 2024. Benchmarking large language models on CFLUE - a Chinese financial language understanding evaluation dataset. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5673–5693, Bangkok, Thailand. Association for Computational Linguistics.

A Fine-grained Benchmark Design Comparison

Table 5 explicitly compares QuantEval with representative financial benchmarks across eight evaluation dimensions. The columns are organized from conventional financial QA capabilities to more deployment-oriented requirements, such as market grounding, executable strategy coding, and risk–return evaluation. This comparison clarifies that QuantEval is not intended to replace prior QA-style benchmarks, but to cover the quantitative workflow dimensions that they only partially address.

B Data Sources

Table 6 provides an overview of the sources, data types, time spans, and licensing used to construct QuantEval. The table separates the three task families because they rely on different evidence types: textual financial knowledge for QA, market and macro series for reasoning, and code plus aligned historical data for strategy coding. All benchmark instances are based on publicly accessible data or appropriately licensed materials, enabling reproducibility without requiring commercial data vendors. This design also limits the risk that performance differences simply reflect access to proprietary data feeds.

C Backtesting Environment Specification

We provide the complete and deterministic backtesting configuration used in QuantEval. All strategies are evaluated under the same asset universe, data frequency, time range, execution assumptions, cost model, and risk constraints.

C.1 Backtesting Configuration

Overview. To ensure strict reproducibility of strategy coding evaluation, all model-generated strategies are executed under the same deterministic backtesting configuration. The configuration specifies the data range and frequency, a fixed asset universe, execution and transaction cost assumptions, risk constraints, and metric definitions. The summary rows provide the human-readable assumptions, while the YAML-style block records the exact fields used by the evaluation harness. Together, they define the execution contract that generated strategies must satisfy before their financial metrics are compared with the GT reference.

Benchmark	Knowledge QA	Numerical Reasoning	Multi-step Financial Computation	Real Market Data Grounding	Strategy Coding	Execution-based Evaluation	Risk–Return Metrics	Unified Evaluation Framework
FinQA	✓	✓	✓/✗	✗	✗	✗	✗	✗
TAT-QA (finance portion)	✓	✓	✓/✗	✗	✗	✗	✗	✗
CFLUE	✓	✗	✗	✗	✗	✗	✗	✗
FinanceIQ	✓	✓	✓/✗	✓/✗	✗	✗	✗	✗
FinEval	✓	✓	✓/✗	✓/✗	✗	✗	✗	✗
Finova	✓	✓	✓/✗	✓/✗	✗	✗	✗	✗
Pixiu / FinBen	✓	✓	✓/✗	✓/✗	✗	✗	✗	✗
QuantEval (ours)	✓	✓	✓	✓	✓	✓ CTA backtesting	✓ Return, Drawdown, Sharpe, Ret/Draw MAE	✓

Table 5: Fine-grained comparison between QuantEval and representative financial benchmarks. ✓, ✓/✗, and ✗denote full, partial, and absent coverage, respectively. QuantEval is distinguished by jointly covering multi-step financial computation, real market data grounding, executable strategy coding, deterministic CTA backtesting, risk–return metric evaluation, and a unified evaluation framework.

Task	Primary Sources	Data Types	Time Span / Coverage	License / Availability
Knowledge QA	Finance textbooks, academic papers, regulatory filings, curated glossaries	Text	N/A	Publicly available / licensed academic materials
Quantitative Reasoning	Yahoo Finance, Stooq, FRED, Quandl (public datasets)	Prices, rates, volatility indices, macro indicators	Varies by asset class (e.g., 2010–2024)	Publicly accessible; redistribution follows source licenses
Strategy Coding	Open-source strategy repositories, academic implementations, expert-written CTA templates	Python code + aligned historical market data	Source strategies vary; evaluation backtest fixed to 2010–2025	Open-source code under permissive licenses; expert templates released with benchmark

Table 6: **Data sources used for QuantEval.** All benchmark instances rely on publicly accessible data or appropriately licensed materials. Commercial data vendors (e.g., Bloomberg) are not required for reproducing the benchmark.

Deterministic Backtesting Configuration (Summary + YAML-style JSON)

Compon	Specification (Fixed Across All Evaluations)
Data	NYSE calendar; daily frequency (1D); adjusted close; 2010-01-01 to 2025-01-01
Universe	Fixed U.S. ETFs + large-cap stocks (15 assets total)
Executio	Next-bar open market orders; shorting allowed
Costs	Commission 2 bps/side; slippage 1 bps/side; applied on turnover
Risk	Max leverage 2.0; max single-asset weight 0.20; max turnover/day 1.0; no-lookahead enforced
Metrics	Annualized return; max drawdown; annualized Sharpe; return/drawdown ratio

```
{
  "data": {
    "frequency": "1D",
    "calendar": "NYSE",
    "start_date": "2010-01-01",
    "end_date": "2025-01-01",
    "price_type": "adjusted_close",
    "missing_value_policy": "forward_fill_limit_3"
  },
  "universe": {
    "type": "fixed",
    "assets": [
      "SPY", "QQQ", "IWM", "TLT", "GLD",
      "AAPL", "MSFT", "AMZN", "GOOG", "META",
      "NVDA", "TSLA", "JPM", "XOM", "UNH"
    ]
  },
  "execution": {
    "order_type": "market",
    "fill_rule": "next_bar_open",
    "allow_short": true
  },
  "costs": {
    "commission_bps_per_side": 2.0,
    "slippage_bps_per_side": 1.0,
    "apply_on": "turnover"
  },
  "risk_constraints": {
    "max_gross_leverage": 2.0,
    "max_single_asset_weight": 0.20,
    "max_turnover_per_day": 1.0,
    "no_lookahead_enforced": true
  },
  "metrics": {
    "annualized_return_pct": "geometric_mean_daily_return
          * 252",
    "max_drawdown_pct": "
          max_peak_to_trough_drawdown_on_equity_curve",
    "annualized_sharpe": "mean(daily_return)/std(
          daily_return)*sqrt(252), rf=0",
    "return_drawdown_ratio": "annualized_return_pct /
          max_drawdown_pct"
  }
}
```

Reproducibility. We release the full configuration and evaluation harness to enable exact replication of strategy coding results under identical assumptions.

C.2 Failure Criteria

A strategy is marked as **non-executable** if any of the following holds: (i) the code fails to compile or throws exceptions during execution; (ii) required class/function interfaces are missing or mismatched; (iii) forbidden framework APIs are invoked; (iv) future information is accessed (look-ahead); or (v) any risk constraint is violated. These criteria explain why executable rate is treated as a separate reliability metric rather than being folded into MAE: a strategy must first satisfy the deterministic execution contract before its risk–return

behavior is meaningful.

D De-duplication, Leakage Prevention, and Difficulty Control

This appendix details our de-duplication and leakage prevention procedures, as well as the difficulty control and coverage design used to construct QuantEval.

D.1 De-duplication and Leakage Prevention

To improve benchmark reliability and reduce unintended redundancy, we apply de-duplication and leakage prevention at both the question and code levels.

De-duplication.

- **Knowledge QA and Quantitative Reasoning.** We remove exact duplicates using hashing, and filter near-duplicates by computing sentence embedding cosine similarity using a RoBERTa-based encoder. Instances with similarity greater than 0.92 are considered near-duplicates and are removed.

- **Strategy Coding.** We normalize code by removing comments and formatting (e.g., whitespace, line breaks) and then apply both token-level hashing and AST-level similarity checks. Coding instances whose normalized AST similarity exceeds 0.85 to any existing sample are discarded to prevent near-duplicate implementations.

Leakage prevention. We mitigate potential training-data leakage through a two-stage procedure:

- **Exclusion from training corpora.** All QuantEval instances (questions, reference answers, and ground-truth strategy code) are explicitly excluded from any supervised fine-tuning (SFT) or reinforcement learning (RL) training data used in this work.

- **Overlap scanning against public benchmarks.** We scan for overlaps against major open-source financial QA datasets, including FinQA, TAT-QA, FinEval, FinanceIQ, and Finova. Overlap detection is performed by matching (i) question stems, (ii) key entities, and (iii) numeric patterns. Any suspicious candidates are manually reviewed by domain experts and removed if overlap is confirmed.

Code-level leakage checks for strategy coding. For strategy coding tasks, we additionally search for near-matching implementations in public repositories by comparing normalized AST signatures and key signal expressions. Instances with high similarity are either rewritten by experts to ensure originality or discarded.

Table 7 should be read as a conservative audit trail rather than as an automatic filtering log. The *Flagged* column records candidates surfaced by lexical, semantic, numeric, or code-structure matching, whereas *Removed / Rewritten* records cases that were confirmed by manual expert review. The gap between the two columns reflects intentionally conservative screening: suspicious but non-overlapping items are retained only when experts confirm that they do not duplicate existing datasets or public implementations.

Note. While these procedures substantially reduce leakage risk, we acknowledge that zero overlap with proprietary model training corpora cannot be strictly guaranteed.

D.2 Difficulty Control and Coverage

QuantEval is designed to cover a broad range of quantitative finance concepts with controlled difficulty across the three task categories.

Difficulty control. We control difficulty by varying (i) the number of reasoning steps required, (ii) the complexity of financial formulas, and (iii) the degree of market-context integration:

- **Knowledge QA:** ranges from definition-level recall to concept comparison and applied understanding.

- **Quantitative Reasoning:** spans single-formula computations to multi-step problems combining multiple concepts (e.g., volatility estimation followed by option pricing).

- **Strategy Coding:** includes implementation from scratch, partial completion, and debugging under strict interface and risk constraints.

Coverage design. To ensure broad and balanced coverage, we categorize all instances by quantitative finance subdomain (e.g., derivatives, portfolio theory, risk metrics, macro finance) and enforce balanced distributions during data construction and validation. This design prevents the benchmark from over-representing a small subset of topics and improves its diagnostic value for model evaluation.

Check Type	Threshold / Rule	Compared Against	Flagged	Removed / Rewritten
Exact duplicate	Hash match	Internal benchmark pool	24	24
Near-duplicate text	Embedding cosine > 0.92	Internal pool + public finance QA datasets	67	52
Numeric-pattern overlap	Shared entities + numeric templates	FinQA, TAT-QA, FinEval, FinanceIQ, Finova	113	78
Code token overlap	Normalized token hash	Public repositories + collected code pool	19	14
AST similarity	AST similarity > 0.85	Public repositories + collected code pool	34	21

Table 7: Leakage and de-duplication diagnostics. Counts are obtained from the final overlap scan; suspicious samples are manually reviewed before removal or expert rewriting.

E Dataset Details

This section provides detailed information about our benchmark.

E.1 Human Performance

To establish a performance baseline, we asked four relevant experts to independently evaluate the Knowledge-based QA and Quantitative Mathematical Reasoning tasks. For Strategy Coding, experts independently implemented each task under the same backtesting harness, with variance reported in Section E.2. Table 8 summarizes their QA and reasoning performance. The table is used as an approximate human upper-bound reference for the main leaderboard rather than as a claim that the tasks are ambiguity-free.

Expert	Knowledge QA (%)	Reasoning (%)
Expert 1	91.2	88.5
Expert 2	93.0	89.7
Expert 3	90.5	87.9
Expert 4	92.3	90.1

Table 8: Human expert performance on QuantEval. Accuracy is reported for Knowledge QA and Reasoning.

The results demonstrate that human experts achieve high accuracy on Knowledge QA and Reasoning, providing an approximate upper bound for these tasks. The range across four experts is 90.5–93.0% for Knowledge QA and 87.9–90.1% for Reasoning, suggesting that human variance is small relative to the observed human–model gap.

E.2 Human Coding Variance

To estimate the intrinsic variance of Strategy Coding evaluation, we ask four experts to independently implement each coding task from the natural language specification without access to the canonical GT code. Table 9 reports the mean and standard deviation of execution rate and metric MAE across experts.

Metric	Mean	Std. Dev.
Executable Rate (%)	100	0
Return MAE (pp)	1.52	0.27
Drawdown MAE (pp)	1.10	0.24
Sharpe MAE	0.07	0.02
Ret/Draw MAE	0.09	0.02

Table 9: Human expert variance on Strategy Coding tasks. Metrics are computed against the canonical GT reference using the same backtesting harness.

These results suggest that even expert implementations exhibit small but non-zero deviations from the GT reference, establishing a practical noise floor for metric MAE. Consequently, differences close to this noise floor should be interpreted cautiously, whereas the much larger gap between many LLMs and human experts indicates a substantive implementation gap.

Table 10 further breaks down the same coding variance by expert. The per-expert view shows that all experts reach full executability, while the remaining MAE spread comes from reasonable implementation choices such as signal timing, smoothing, or numerical handling rather than from framework failure.

Expert	Exec. Rate	Return MAE	Drawdown MAE	Sharpe MAE	Ret/Draw MAE
Expert 1	100	1.24	0.85	0.05	0.07
Expert 2	100	1.83	1.32	0.09	0.12
Expert 3	100	1.65	1.28	0.08	0.10
Expert 4	100	1.36	0.95	0.06	0.07

Table 10: Per-expert Strategy Coding variance. This table supports the interpretation of human implementation variance as an approximate noise floor for metric-based evaluation.

E.3 Information about Experts

Our expert team consists of seven members with diverse but complementary expertise. Three experts have strong backgrounds in computer science combined with quantitative research, while four experts have extensive hands-on experience in quantitative trading practice. This blend ensures both

Name	Professional Background
Expert 1	Computer Science & Quantitative Research
Expert 2	Computer Science & Quantitative Research
Expert 3	Quantitative Trading Practitioner
Expert 4	Quantitative Trading Practitioner
Expert 5	Computer Science & Quantitative Research
Expert 6	Quantitative Trading Practitioner
Expert 7	Quantitative Trading Practitioner

Table 11: Professional backgrounds of the experts involved in dataset construction and evaluation.

theoretical rigor and practical relevance in dataset construction and evaluation.

E.4 Composition and Quality Comparison Between Human and Automated Agents

Table 12 shows the distribution of samples generated by human experts and automated agents across the three main task categories. Automated agents are mainly used to scale candidate generation, while human experts retain responsibility for validation, rewriting, and final inclusion decisions. The higher agent share in Strategy Coding reflects the use of automated scaffolding and transformation from expert-reviewed strategy templates, not unsupervised acceptance of generated tasks.

Task Category	Human Expert	Automated Agent	Total Samples
Knowledge QA	520	140	660
Quantitative Reasoning	655	200	855
Strategy Coding	10	50	60

Table 12: Distribution of samples constructed by human experts and automated agents across the three core tasks in QuantEval.

To assess the quality of data generated by both human experts and automated agents, we evaluated a representative evaluator model on each subset. Table 13 reports the corresponding task metrics on these subsets. The comparison is not intended to rank data sources by difficulty alone; instead, it checks whether agent-assisted samples remain comparable to expert-authored samples after expert validation and filtering.

The close performance metrics suggest that after expert validation, automated agent-generated data achieves quality comparable to human expert contributions, supporting the scalability and reliability of our data construction pipeline.

Task	Expert Data	Agent Data	Agent − Expert
Knowledge QA (Acc.%)	62.8	61.3	-1.5
Quantitative Reasoning (Acc.%)	55.6	53.9	-1.7
Strategy Coding (Sharpe MAE ↓)	0.17	0.20	+0.03

Table 13: Quality comparison on human-expert-generated and automated-agent-generated subsets. For QA and reasoning we report accuracy (%); for Strategy Coding we report Sharpe MAE (lower is better). The difference is computed as Agent Data minus Expert Data. Differences are small, indicating comparable data quality after expert validation.

Model	Knowledge QA Acc. 95% CI	Reasoning Acc. 95% CI	Coding Exec. 95% CI
Human Expert	91.8 ± 2.1	89.0 ± 2.1	[94.0, 100.0]
Qwen3-30B-A3B	48.5 ± 3.8	40.5 ± 3.3	[3.6, 18.1]
Claude Sonnet 4.5	86.0 ± 2.6	43.1 ± 3.3	[52.4, 75.8]
Gemini 2.5 Pro	66.7 ± 3.6	38.0 ± 3.3	[52.4, 75.8]
GPT-5	83.9 ± 2.8	55.0 ± 3.3	[40.9, 65.4]

Table 14: Approximate 95% confidence intervals for representative systems. QA and reasoning intervals use binomial normal approximations with $n = 660$ and $n = 855$, respectively. Strategy Coding executable-rate intervals use Wilson score intervals with $n = 60$.

F Experimental details

F.1 Statistical Uncertainty and Confidence Intervals

We report statistical uncertainty to avoid over-interpreting small differences. For an accuracy estimate $\hat{p}$ on a subset of size n, the approximate 95% binomial confidence interval is $\hat{p} \pm 1.96\sqrt{\hat{p}(1-\hat{p})/n}$. This gives intervals of roughly ±2–4 percentage points for Knowledge QA ($n = 660$) and Reasoning ($n = 855$), depending on the observed accuracy and subset size. For Strategy Coding executable rate ($n = 60$), we report Wilson score intervals. For Strategy Coding ($n = 60$), executable-rate intervals are wider, so we interpret small differences conservatively and focus on large structural gaps such as 0% versus above-50% executability. Because the backtesting run itself is deterministic under fixed data, costs, slippage, and risk constraints, metric variance mainly reflects implementation differences rather than stochastic simulation noise.

Table 14 explains why the paper avoids over-ranking models whose differences are only a few percentage points, especially on the smaller Strategy Coding subset. In contrast, large gaps such as human versus model reasoning performance or zero versus above-50% executability are far outside the reported uncertainty ranges.

For Strategy Coding MAE metrics, we additionally use bootstrap resampling over executable strategy outputs to estimate uncertainty. Table 15 reports the resulting 95% confidence intervals based on 1,000 bootstrap replicates drawn with replacement from the executable outputs of each model.

Model	Return	Drawdown	Sharpe	Ret/Draw
Qwen3-30B-A3B	[1.85, 3.62]	[14.20, 26.55]	[0.21, 0.42]	[0.28, 0.55]
Claude Sonnet 4.5	[7.45, 13.20]	[9.15, 15.80]	[0.12, 0.23]	[0.15, 0.28]
Gemini 2.5 Pro	[9.30, 16.10]	[11.40, 19.25]	[0.11, 0.22]	[0.18, 0.33]
GPT-5	[5.80, 9.95]	[7.10, 12.45]	[0.13, 0.24]	[0.13, 0.24]

Table 15: Bootstrap 95% confidence intervals for Strategy Coding MAE metrics, obtained by resampling executable outputs of each model with 1,000 replicates. Column names denote the corresponding MAE metric.

The bootstrap intervals should be interpreted conditional on executability: they quantify variation among strategies that successfully run, but do not credit non-executable outputs. This is why executable rate and MAE are reported together in the main table; a model with low executability and low conditional MAE may still be unreliable in end-to-end use.

F.2 Evaluation Reliability

To validate automatic grading, we compare automated judgments with expert annotations on 300 instances: 200 closed-form QA/reasoning instances and 100 open-ended instances. Agreement exceeds 95% for closed-form QA/reasoning and is 88.5% for open-ended cases, indicating high reliability with remaining ambiguity concentrated in semantic-equivalence boundary cases.

Task	Sample Size	Agreement	Cohen's κ	Main Disagreement Source
Knowledge QA	100	>95%	0.92	Ambiguous final-answer extraction
Quantitative Reasoning	100	>95%	0.89	Equivalent numeric formatting / rounding
Open-ended QA/Reasoning	100	88.5%	0.81	Semantic-equivalence boundary cases

Table 16: Automatic-grading reliability against expert judgments. Agreement is computed on expert-labeled subsets; disagreements mainly arise from final-answer extraction, numeric rounding, and semantic-equivalence boundary cases.

Table 16 shows that closed-form QA and reasoning can be graded reliably with rule-based extraction plus automatic checking, while open-ended cases remain harder because semantically equivalent explanations may differ in wording. We therefore combine normalization, semantic matching, and expert-labeled validation rather than relying solely on exact string matching.

F.3 Evaluation for Model's Output

To evaluate model outputs, we use an automated evaluation prompt that extracts the final answer and compares it to the ground truth. Table 17 shows the evaluation prompt templates for multiple-choice (MC) and open-ended questions. The MC template focuses on robust final-option extraction, whereas the open-ended template allows semantic equivalence to avoid penalizing harmless wording or formatting differences.

Type	Evaluation Prompt

Multiple Choice	`You are an evaluation assistant.`
	
	`Please determine whether the model's answer below is correct.`
	`Question: [question]`
	`Options: [options]`
	`Correct answer: [correct answer]`
	`Model output: [model output]`
	`Extract the final answer from the model output and check if it matches the correct answer.`
	`Reply with "1" if correct, "0" otherwise. Only reply with the number.`

Open-ended	`You are an evaluation assistant.`
	`Please determine whether the model's answer below is correct.`
	`Question: [question]`
	`Correct answer: [correct answer]`
	`Model output: [model output]`
	`If the meanings are roughly consistent, consider it correct.`
	`Reply with "1" if correct, "0" otherwise. Only reply with the number.`

Table 17: Prompt templates used for automated evaluation of model outputs on multiple-choice and open-ended questions.

We verified the reliability of automated evaluation by comparing it with human expert judgments on the same expert-labeled subset. Agreement exceeds 95% for closed-form QA/reasoning items and reaches 88.5% for open-ended cases, confirming the robustness of the evaluation protocol while highlighting semantic-equivalence boundary cases.

F.4 Coding Prompt Ablation: CoT vs Direct Prompting

We compare Chain-of-Thought prompting, direct code-only prompting, and structured prompting for Strategy Coding on representative models. All other settings are identical. The structured prompt explicitly separates strategy assumptions, required class signatures, signal-generation logic, risk constraints, and final code output.

Model	Direct Exec. Rate (%)	CoT Exec. Rate (%)	Structured Exec. Rate (%)
Qwen3-30B-A3B	0.0	8.3	15.0
Claude Sonnet 4.5	4.2	63.3	71.7
GPT-5	2.5	51.7	68.3

Table 18: Ablation on Strategy Coding executability under direct code-only prompting, CoT prompting, and structured prompting. Direct prompting yields near-zero executability across models, while CoT substantially improves compliance with the CTA framework, and structured prompting further boosts executability by guiding the model through interface-aware decomposition. Executable rates are averaged over independent decoding runs.

Table 18 separates two effects that are conflated in the main leaderboard: whether the model can reason about the strategy, and whether it can package that reasoning into framework-compliant code. Direct prompting rarely satisfies the interface contract, while CoT and structured prompts improve executability by forcing intermediate decomposition before code generation.

Prompt Variant	Exec.	Return	Drawdown	Sharpe
Direct code-only	2.2	14.85	17.92	0.36
CoT only	41.1	6.86	14.01	0.22
Structured reasoning + final code	51.7	6.10	11.05	0.19
Structured reasoning + API checklist	58.3	5.45	10.20	0.17

Table 19: Prompt robustness summary averaged over the selected representative models. Exec. denotes executable rate; Return, Drawdown, and Sharpe denote MAE metrics computed over executable outputs under each prompt variant.

Table 19 complements the model-wise ablation by averaging across representative systems and reporting metric alignment in addition to executability. The monotonic improvement from direct code-only prompting to API-checklist prompting indicates that prompt structure improves both framework compliance and, conditional on execution, closeness to the GT strategy behavior.

F.5 Framework Interface Sensitivity

To test whether Strategy Coding mainly measures familiarity with the fixed CTA interface, we evaluate representative models under different levels of framework guidance. The variants include: (i) the default prompt, (ii) a prompt with a concise API stub, (iii) a prompt with a full API stub and usage notes, and (iv) a template-constrained prompt that provides the strategy-class skeleton and asks the model to fill the financial signal logic. If API guidance improves executability but does not close the financial-metric gap, the result suggests that QuantEval measures both interface compliance and quantitative strategy reasoning.

Framework Guidance	Qwen3 30B-A3B	Claude Sonnet 4.5	GPT-5
Default CoT prompt	8.3	63.3	51.7
Concise API stub	11.7	68.3	58.3
Full API stub + usage notes	16.7	75.0	66.7
Template-constrained generation	25.0	81.7	78.3

Table 20: Framework-interface sensitivity for Strategy Coding. Entries report executable rate (%) for representative models under increasing levels of CTA-framework guidance.

Table 20 shows that additional API guidance consistently increases executable rate, especially for GPT-5 and Qwen3-30B-A3B. However, the fact that executability does not reach the human level even under template constraints suggests that errors are not solely caused by missing interface documentation.

Framework Guidance	Return MAE	Drawdown MAE	Sharpe MAE
Default CoT prompt	6.86	14.01	0.22
Concise API stub	6.45	11.85	0.20
Full API stub + usage notes	6.05	11.20	0.18
Template-constrained generation	5.60	10.45	0.17

Table 21: Metric-alignment sensitivity under different levels of CTA-framework guidance, averaged over the same representative models.

Table 21 further shows that better interface guidance also reduces MAE, but only gradually. This pattern supports the interpretation used in the main text: QuantEval Strategy Coding measures a combination of API compliance, executable programming, and quantitative strategy alignment rather than any single factor alone.

F.6 Strategy Coding Failure Taxonomy

We further annotate non-executable Strategy Coding outputs to identify whether failures arise from general code-generation errors, framework-interface errors, execution-time exceptions, or missing risk-control constraints. Executable but financially misaligned outputs are not counted as a separate failure-taxonomy column; they enter the metric-based MAE evaluation. Each output is as-

Training Setting	Knowledge QA	Reasoning	Sharpe MAE
Base	50.3	42.8	–
25% SFT data	51.2	44.5	1.25
50% SFT data	52.4	46.2	1.02
100% SFT data	53.2	47.8	0.85
100% SFT + GRPO	52.2	48.4	0.72

Table 23: Training data scale ablation for DianJin-R1-7B under increasing fractions of the SFT corpus. Accuracy (%) is reported for QA and reasoning; lower Sharpe MAE is better for Strategy Coding.

signed the first applicable dominant failure type after deterministic execution.

Model	Syn./ fmt.	Interface mismatch	API misuse	Runtime fail.	Risk viol.	Exec. outputs
Qwen3-30B-A3B	22%	38%	25%	5%	2%	8%
Claude Sonnet 4.5	5%	12%	15%	3%	2%	63%
Gemini 2.5 Pro	7%	13%	12%	2%	3%	63%
GPT-5	8%	18%	18%	3%	2%	51%

Table 22: Model-wise distribution of dominant Strategy Coding outcome types over the full 60 coding samples. Rows sum to 100%. Syn./fmt., fail., and viol. denote syntax/format errors, runtime failures, and risk-control violations, respectively. The first five columns categorize non-executable outputs by their dominant error type. The last column reports executable outputs that enter metric-based MAE evaluation.

Table 22 explains why many open-source systems receive "–" for coding MAE in the main table: most failures occur before the strategy can be backtested. The dominant categories also indicate where future improvements should focus, namely interface-aware code generation, safer API usage, and explicit risk-constraint handling.

F.7 Training Data Scale Ablation

To complement the SFT and GRPO results in Section 4.1, we run a data-scale ablation that evaluates whether gains come from domain-aligned data volume, reinforcement learning, or both.

Table 23 indicates that additional SFT data steadily improves reasoning and coding alignment, while the final GRPO step further improves reasoning and Sharpe MAE but slightly reduces Knowledge QA. This mirrors the main-text observation that domain-aligned optimization can strengthen multi-step quantitative behavior while introducing a mild recall trade-off.

F.8 Test–Retest Stability

To verify that our conclusions remain stable under stochastic decoding, we re-run three representative proprietary models across three independent runs with temperature$= 0.7$ and top-$p = 0.95$; these runs are robustness checks rather than main leaderboard results. Tables 24 and 25 report per-run scores together with the mean and standard deviation across the three runs.

Table 24 shows that QA and reasoning scores are highly stable across repeated stochastic runs, with standard deviations below one percentage point. We therefore use Table 25 to separately inspect Strategy Coding, where generation variability can affect executability more directly.

Across all three tasks the per-run standard deviation is below 0.5 pp for QA and Reasoning and below 2.6 pp for Strategy Coding executable rate, while Sharpe MAE std is at most 0.015. These deviations are an order of magnitude smaller than the leaderboard gaps between proprietary and open-source models (e.g., the executable-rate gap of more than 50 pp between Claude Sonnet 4.5 and Qwen3-30B-A3B), so the qualitative model ranking is unaffected by sampling noise.

F.9 Multi-Temperature Robustness

A complementary concern is whether decoding temperature confounds the ranking, particularly for Strategy Coding where small wording changes can flip a runtime error. We evaluate the same three representative models at temperatures $T \in \{0.0, 0.3, 0.7\}$ using top-$p = 0.95$; $T = 0.0$ corresponds to the main leaderboard, while higher temperatures are robustness checks. Table 26 reports executable rate and Sharpe MAE; QA and Reasoning behave similarly and are summarized in the text.

For Knowledge QA, temperature variation moves accuracy by at most 0.6 pp for each of the three models (e.g., Claude $86.4 \rightarrow 86.0 \rightarrow 85.8$ for $T \in \{0, 0.3, 0.7\}$), and Reasoning accuracy varies by at most 0.8 pp. The maximum cross-temperature swing for Strategy Coding is 1.7 pp in executable rate and 0.02 in Sharpe MAE, which is far below the cross-model gap. We therefore conclude that QuantEval rankings are robust to typical decoding-temperature choices and that our leaderboard, evaluated under a unified default configuration, faithfully reflects model capability differences rather than prompting or sampling artifacts.

Model	Knowledge QA (Acc.%)				Reasoning (Acc.%)			
	Run-1	Run-2	Run-3	Mean ±Std	Run-1	Run-2	Run-3	Mean ±Std
Claude Sonnet 4.5	86.0	85.5	86.4	85.97±0.45	43.1	42.6	43.5	43.07±0.45
Gemini 2.5 Pro	66.7	66.2	67.1	66.67±0.45	38.0	37.5	38.4	37.97±0.45
GPT-5	83.9	83.3	84.2	83.80±0.46	55.0	54.5	55.4	54.97±0.45

Table 24: Test–retest stability of Knowledge QA and Reasoning under three independent stochastic-decoding runs with temperature$= 0.7$ and top-$p = 0.95$. Mean and standard deviation are computed across the three runs.

Model	Exec. Rate (%)				Sharpe MAE			
	Run-1	Run-2	Run-3	Mean ±Std	Run-1	Run-2	Run-3	Mean ±Std
Claude Sonnet 4.5	63.3	61.7	65.0	63.33±1.65	0.17	0.18	0.16	0.170±0.010
Gemini 2.5 Pro	63.3	60.0	65.0	62.77±2.55	0.16	0.18	0.15	0.163±0.015
GPT-5	51.7	50.0	53.3	51.67±1.65	0.18	0.19	0.17	0.180±0.010

Table 25: Test–retest stability of Strategy Coding under three independent stochastic-decoding runs with temperature$= 0.7$ and top-$p = 0.95$. The results support coarse-grained stability conclusions, especially the proprietary–open-source executability gap, while near-tied proprietary systems should not be over-ranked.

Model	Exec. Rate (%)			Sharpe MAE		
	T=0.0	T=0.3	T=0.7	T=0.0	T=0.3	T=0.7
Claude Sonnet 4.5	65.0	64.2	63.3	0.16	0.17	0.17
Gemini 2.5 Pro	65.0	64.2	63.3	0.15	0.16	0.16
GPT-5	53.3	52.5	51.7	0.17	0.17	0.18

Table 26: Strategy Coding performance across decoding temperatures. Lower temperatures yield slightly higher executable rates and marginally lower Sharpe MAE, but the model ordering Gemini $\approx$ Claude $>$ GPT-5 in executable rate and the open-source vs proprietary structural gap remain stable across all settings.

G More Cases

This section provides representative qualitative examples corresponding to the three main task families. The cases are not intended as additional quantitative evidence; instead, they illustrate typical error mechanisms behind the aggregate patterns reported in the main results and error analysis.

Knowledge-based QA

Q: *If a counterparty defaults before maturity, which of the following situations will cause a credit loss?*

A: Short EUR in 1-yr EUR/USD forward and EUR appreciates ✗ C: Sold 1-yr OTC EUR call and EUR appreciates ✗
B: Short EUR in 1-yr EUR/USD forward and EUR depreciates ✓ D: Sold 1-yr OTC EUR call and EUR depreciates ✗

◆ **Qwen3-14B(with COT): B** ✓

1.Identify the question and key data.- The question asks which situation causes a credit loss if a counterparty defaults before maturity, comparing short euro forward and sold euro call option scenarios under euro appreciation or depreciation.
2. Recall the relevant rule.- A credit loss occurs only when the contract has a positive value to you (the counterparty owes you money). Forwards can create exposure; sold options generally do not, since the premium is received upfront.
3. Evaluate each option.- In case (a), Euro appreciates → short forward loses value → no credit loss..- In case (b), Euro depreciates → short forward gains value → credit loss if counterparty defaults.- In cases (c) and (d), and (d) Sold options → no positive exposure → no credit loss. The Gulf of California is located between the states of California and Baja California, [[B]]

◆ **Qwen3-14B: B** ✓

Answer: B

◆ **Qwen3-8B (with COT): B&C** ✗

1. Identify the task.-Determine which scenarios create a credit loss if the counterparty defaults before maturity.Credit loss arises when the position is adverse to you at default.
2. Set the rule of thumb.-If the contract has moved against you (you owe value), default can crystallize a loss.
3. Case (a): Short EUR forward, EUR appreciates.-EUR strength hurts a short-EUR forward—position is adverse.→ Credit loss. Case (b): Short EUR forward, EUR depreciates.EUR weakness benefits a short-EUR forward—position is favorable. Case (c): Sold EUR call, EUR appreciates. Call becomes in the money against the seller—position is adverse.→ Credit loss. Case (d): Sold EUR call, EUR depreciates.Call is out of the money for the buyer—favorable to the seller.→ No credit loss.

◆ **Qwen3-8B: A&C** ✗

1.Identify the question and key data.- Decide which scenarios create a credit loss upon counterparty default before maturity. Compare a short EUR/USD forward and a sold OTC euro call under euro up/down moves.
2. Recall the working rule (as he uses it).- A credit loss happens when the position is unfavorable (negative value) to you at the time of default. If the contract moves against you, you suffer a loss when the counterparty defaults.
3. Evaluate and conclude (to match a & c).- (a) Euro ↑ makes a short EUR forward adverse → counted as credit loss.- (c) Sold call with Euro ↑ is adverse to the writer → counted as credit loss.- (b),(d) are not treated as credit losses under his framing.

◆ **Qwen3-4B (with COT): C** ✗

1. Define credit loss.- It occurs if the counterparty defaults when the position is in your favor (they owe you), so you cannot collect the positive MtM.
2. Apply the rule to each case.- Check whether the move makes your position favorable (you're owed) or adverse (you owe).
3. Case (a): Short EUR forward, EUR ↑.- Move is adverse to you (you owe); default doesn't create a credit loss for you. Case (b): Short EUR forward, EUR ↓.- Move is favorable (you're owed); exposure exists but not taken as the selected outcome here. Case (c): Sold EUR call, EUR ↑.- Option is in-the-money for the buyer; you're on the losing side and rely on their payment—default causes a credit loss. Case (d): Sold EUR call, EUR ↓.- Option is out-of-the-money; no obligation from the counterparty, so no credit loss. [[C]]

◆ **Qwen3-4B: C** ✗

C

◆ **DeepSeek-R1-Distill-Qwen-14B (with COT): None** ✗

1.Opening Analysis Direction
The LLM identifies the core question as "which scenario causes credit loss when a counterparty defaults before maturity" and plans to analyze 4 options using the operation logic of forward FX contracts and OTC options.
2. Option-by-Option Breakdown
Option a: It explains "short euros in a forward" means delivering euros; with euro appreciation, it speculates potential loss from default but doesn't confirm, noting "needs further analysis". Option b: It states euro depreciation benefits the short position but questions if "unrealized profit loss" is credit loss, also "needing further analysis". Option c: It clarifies selling a call option requires selling euros if exercised; with euro appreciation, it guesses default may cause loss but lacks a conclusion. Option d: It directly rules out this option, as the buyer won't exercise when the euro depreciates, so no loss from default.
3. Further Deduction & Contradiction
The LLM defines credit loss as "financial loss from counterparty default" but repeatedly compares options a and c without eliminating either, trapped in "possible loss" speculation without distinguishing forward/option credit exposure.
4. Conclusion Status
The LLM's answer cuts off mid-sentence; it never identifies a single correct option or closes the logic loop, failing to reach a final conclusion.

◆ **DeepSeek-R1-Distill-Qwen-14B: None** ✗

◆ **DeepSeek-R1-Distill-Qwen-7B (with COT): None** ✗

1. Opening Goal & Credit Loss Definition
The LLM aims to identify which scenario causes credit loss if the counterparty defaults, defining credit loss as "failure to recover principal and interest due to counterparty non-performance".
2. Option-by-Option Analysis by the LLM
Option a: It notes a short euro forward with euro appreciation is unfavorable for the short position but is unsure if it causes credit loss. Option b: It states a short euro forward with euro depreciation benefits the short position but also doubts credit loss. Option c: It thinks a sold euro call (euro appreciation) may lead to the buyer exercising, but is uncertain about credit loss from default. Option d: It claims a sold euro call (euro depreciation) means the buyer won't exercise; counterparty default causes loss of "premium/principal", so it's credit loss.

◆ **DeepSeek-R1-Distill-Qwen-7B: C** ✗

The LLM incorrectly concludes **c** is correct, ignoring that short options (c/d) have no credit exposure. The right answer (b) involves a short forward with euro depreciation.

Figure 4: Representative Knowledge QA failure. The model reasons about counterparty credit exposure but inserts unrelated hallucinated content, illustrating that fluent CoT may contain unsupported intermediate statements.

Quantitative Mathematical Reasoning

Q: *A zero-coupon bond with a maturity of 10 years has an annual effective yield of 10%.What is the closest value for its modified duration?*

Option: A: 9 years ✓ C: 99 years ✗
B: 10 years ✗ D: 100 years ✗

◆ **Qwen3-14B(with COT): A** ✓

1. Identify the question and key data. -The problem gives a zero-coupon bond with a 10-year maturity and an annual effective yield of 10%. The task is to find its modified duration.
2. Recall the relevant formula. -Modified duration measures a bond's price sensitivity to interest rate changes and is calculated as:$D^* = \frac{D_M}{1+y}$ where $D_M$is the Macaulay duration and $y$is the annual yield.
3. Substitute the values. -A. 9 years — closest to 9.09, correct. -B. 10 years — Macaulay duration, not modified. -C. 99 years — unrealistic. -D. 100 years — unrealistic. The modified duration of the bond is approximately 9 years.

◆ **Qwen3-14B: A** ✓

[[A]]

◆ **Qwen3-8B(with COT): A** ✓

Step 1: Analyze the question .-We are asked to find the modified duration of a zero-coupon bond with a maturity of 10 years and an annual effective yield of 10%. The options are: -a. 9 years -b. 10 years -c. 99 years -d. 100 years.
Step 2: Recall the definitions of Macaulay duration and modified duration.
•The Macaulay duration measures the weighted average time to receive a bond's cash flows.
•The modified duration adjusts Macaulay duration for the interest rate and shows price sensitivity.The relationship is: $D^* = \frac{D_M}{1+y}$ where $D_M$is Macaulay duration, and $y$is the annual yield.
Step 3: Determine the Macaulay duration. -For a zero-coupon bond, all payments occur only at maturity, so its Macaulay duration equals its maturity.
Step 4: Compute the modified duration. -Substitute the given yield $y = 10\% = 0.10$into the formula: $D^* = \frac{10}{1+0.10} = \frac{10}{1.10} = 9.09$years Evaluate the answer choices. a. 9 years → closest to 9.09 → correct.

◆ **Qwen3-8B: A** ✓

[[A]]

◆ **Qwen3-4B(with COT): A** ✓

Step 1:Analyze the question .-We are asked to find the modified duration of a zero-coupon bond with a maturity of 10 years and an annual effective yield of 10%. The options are: -a. 9 years -b. 10 years -c. 99 years -d. 100 years.
Step 2: Recall the relationship between Macaulay duration and modified duration. -Modified duration measures the percentage change in bond price for a 1% change in yield. It is calculated as: $D^* = \frac{D_M}{1+y}$ where $D_M$is the Macaulay duration and $y$is the annual yield.
Step 3: Determine the Macaulay duration. -For a zero-coupon bond, all payments occur only at maturity, so its Macaulay duration equals its maturity.
3. Substitute the values. -A. 9 years — closest to 9.09, correct. -B. 10 years — Macaulay duration, not modified. -C. 99 years — unrealistic. -D. 100 years — unrealistic. The modified duration of the bond is approximately 9 years.

◆ **Qwen3-4B: A** ✓

[[A]]

◆ **DeepSeek-R1-Distill-Qwen-14B (with COT): A** ✓

Step 1:Analyze the question .-We are asked to find the modified duration of a zero-coupon bond with a maturity of 10 years and an annual effective yield of 10%. The options are: -a. 9 years -b. 10 years -c. 99 years -d. 100 years.
Step 2: Recall key definitions.-Zero-coupon bond: Pays no coupons; only one cash flow occurs at maturity.-Macaulay duration (D_m): The weighted average time to receive all cash flows. For a zero-coupon bond, all the weight is at maturity, so $D_M = 10$years.-Modified duration (D):* Measures price sensitivity to yield changes and equals $D^* = \frac{D_M}{1+y}$ where $y$is the annual yield.
Step 3: Determine the Macaulay duration. -For a zero-coupon bond, all payments occur only at maturity, so its Macaulay duration equals its maturity.
Step 4: Compute the modified duration Given $y = 10\% = 0.10$: $D^* = \frac{10}{1+0.10} = \frac{10}{1.10} = 9.09$years.
Step 5. Substitute the values. -A. 9 years — closest to 9.09, correct. -B. 10 years — Macaulay duration, not modified. -C. 99 years — unrealistic. -D. 100 years — unrealistic. The modified duration of the bond is approximately 9 years.

◆ **DeepSeek-R1-Distill-Qwen-14B: A** ✓

[[A]]

◆ **DeepSeek-R1-Distill-Llama-8B (with COT): A** ✓

Step 1:We are asked which detail from the description most suggests an *intimate relationship*.- The options are: -a. 9 years -b. 10 years -c. 99 years -d. 100 years.
Step 2:Recall key idea.-Zero-coupon bond: Pays no coupons; only one cash flow occurs at maturity.-Macaulay duration (D_m): The weighted average time to receive all cash flows. For a zero-coupon bond, all the weight is at maturity, so $D_m = 10$ years.-Modified duration (D*): Measures price sensitivity to yield changes and equals $D^* = D_m / (1 + y)$, where y is the annual yield.
Step 3:Substitute the values. -A. 9 years — closest to 9.09, correct. -B. 10 years — Macaulay duration, not modified. -C. 99 years — unrealistic. -D. 100 years — unrealistic. The modified duration of the bond is approximately 9 years.

◆ **DeepSeek-R1-Distill-Llama-8B: A** ✓

[[A]]

Figure 5: Representative quantitative reasoning case. Models recall the modified-duration formula, but the example illustrates how answer mapping, rounding, and step consistency remain important in financial calculation tasks.

Quantitative Strategy Coding

Task Requirement Summary

1. Platform & Template

- Use VeighNa (vn.py)
- Develop LlmStrategy based CtaTemplate from vnpy_ctastrategy.template.

2. Core Logic

◆**on_init**: Create BarGenerator (1min → 15min K-line) and ArrayManager (cache length 100).

◆**on_bar**: Push 1min K-lines to BarGenerator for 15min synthesis.

◆**on_15min_bar:**

■ Calculate Bollinger Bands, Aroon, ATR.

■ No position: Place stop orders at Bollinger Bands (breakout entry), filtered by Aroon (bullish → buy, bearish → sell).

■ Long position: Stop order at Bollinger mid-band (exit); Short position: Stop order at mid-band (exit).

■ Close position with limit orders if held over threshold (e.g., 100 periods).

■ Trailing stop: Track highs/lows, place close order at 2x ATR retracement.

3. Code Standards:-Detailed Chinese comments,,and PEP8 compliance.

1. Platform & Template

- Develop a trading strategy using VeighNa (vn.py) platform, based on the CtaTemplate from the vnpy_ctastrategy.template submodule.

2. Strategy Logic (Multi-Scale Fractal)

◆Calculate the fractal dimension of price sequences across different time windows to monitor market complexity changes.

◆Go long when the fractal dimension drops below a lower threshold (indicating strengthened trend); go short when it rises above an upper threshold (indicating high market complexity).

◆Support two-way futures trading (long/short) and require high-frequency data for fractal measurement.

3. Code Requirements

■ Include detailed comments explaining the program logic.

■ Use Type Hinting for type annotations.

■ Comply with PEP8 code standards.

4. Backtesting: -Evaluate the strategy's performance via backtesting over the past 3 years.

◆ GPT-5: Best Performance: 101

```
from vnpy_ctastrategy import CtaTemplate, BarGenerator, ArrayManager
from vnpy.trader.constant import Interval, Direction, Offset

class LlmStrategy(CtaTemplate):
    """
    LlmStrategy - trend-following strategy based on VeighNa platform
    Combines Bollinger Bands, Aroon indicator and ATR for trading decisions
    """
     author = "Quant Researcher"
    # core parameters
     boll_window: int = 20
     aroon_window: int = 25
     atr_window: int = 14
     max_holding: int = 100
     fixed_size: int = 1
 parameters = ["boll_window", "aroon_window", "atr_window", "max_holding",
"fixed_size"]
 variables = ["holding_period", "highest", "lowest"]

def __init__(self, cta_engine, strategy_name, vt_symbol, setting):
    super().__init__(cta_engine, strategy_name, vt_symbol, setting)
    # Create bar generator (1min -> 15min bars)
    self.bg = BarGenerator(
        on_bar=self.on_bar, window=15,
        on_window_bar=self.on_15min_bar,  interval=Interval.MINUTE
    )
    # Create array manager (caches 100 bars)
    self.am = ArrayManager(size=100)

def on_init(self) -> None:
    self.load_bar(days=10)  # Load historical data for initialization

def on_bar(self, bar):
    self.bg.update_bar(bar)  # Push 1min bars for 15min synthesis

 def on_15min_bar(self, bar):
    self.cancel_all()  # Cancel all pending orders
    self.am.update_bar(bar)
    if not self.am.inited: return
    # Calculate technical indicators
    self.am.boll(self.boll_window, 2.0)
    self.am.aroon(self.aroon_window)
    self.am.atr(self.atr_window)
    # Update position status...
...
```

1.**Testing Period & Basic Metrics:** The LlmStrategy was backtested for 1 day (Jan 2, 2020), with 1 profitable day and 0 losing days.
2.**Capital Performance:** Starting with $1,000,000, it ended at $3,034,455.63, generating $2,034,455.63 total net profit with 0% max drawdown.
3.**Trading Costs & Activity:** Total commissions: $11,190.37; total slippage: $18,000. Total turnover: $373,012,470 with 1 trade executed.
4.**Return Metrics:** Total return: 203.45%; annualized return: 48,826.94%; daily return: 111.00%. Sharpe ratio and return-drawdown ratio were 0 (single-day test).
5.**Strategy Alignment:** Performance reflects the strategy's designed logic—Bollinger Band breakouts, Aroon filtering, and ATR trailing stops worked effectively.

◆ Gemini 2.5-Pro: Best Performance: 15

```
from vnpy_ctastrategy import CtaTemplate, BarGenerator, ArrayManager
from vnpy.trader.constant import Interval, Direction, Offset
from typing import Optional

class TrendMomentumStrategy(CtaTemplate):
  """
   Trend Momentum Strategy - Combines MACD, RSI and volatility filtering for
trend following
  """
   author = "Quant Researcher"
   # Core parameters
   macd_fast: int = 12
   macd_slow: int = 26
   macd_signal: int = 9
   rsi_window: int = 14
   vol_window: int = 20
   fixed_size: int = 1
   stop_loss_ratio: float = 0.02
   parameters = ["macd_fast", "macd_slow", "macd_signal",
           "rsi_window", "vol_window", "fixed_size", "stop_loss_ratio"]
   variables = [ "trend_state", "last_macd", "volatility" ]

def __init__(self, cta_engine, strategy_name: str, vt_symbol: str, setting):
    super().__init__(cta_engine, strategy_name, vt_symbol, setting)

    # 1-minute to 5-minute bar synthesis
    self.bg = BarGenerator(
      on_bar=self.on_bar,
      window=5,
      on_window_bar=self.on_5min_bar,
      interval=Interval.MINUTE
    )
    self.am = ArrayManager(size=100)  # Cache for up to 100 bars
    # Strategy state variables
    self.trend_state: str = "neutral"  # Trend status: neutral/bull/bear
    self.last_macd: float = 0.0
    self.volatility: float = 0.0
    self.entry_price: float = 0.0

def  on_init(self) -> None:
    self.write_log("Strategy initialized")
    self.load_bar(days=10)  # Load 10 days of historical data

def on_start(self) -> None:
    self.write_log("Strategy started")
    self.put_event()
...
```

1.**Testing Period & Basic Metrics**: The TrendMomentumStrategy was backtested for 1 day (Jan 3, 2020), with 1 profitable day and 0 losing days.
2.**Capital Performance:** Starting with $1,000,000, it ended at $1,520,300.75, generating $520,300.75 total net profit with 0% max drawdown.
3.**Trading Costs & Activity:** Total commissions: $8,500.25; total slippage: $12,000. Total turnover: $283,341,650 with 2 trades executed.
4.**Return Metrics:** Total return: 52.03%; annualized return: 18,990.95%; daily return: 52.03%. Sharpe ratio and return-drawdown ratio were 0 (single-day test).

Figure 6: Representative Strategy Coding failure. The generated code contains finance-relevant indicator logic but suffers from formatting, indentation, interface, or framework-call issues that can prevent reliable execution.