# FinREALBench: The Execution Reliability Gap Beyond Reasoning and Acting

Anonymous authors
Paper under double-blind review

## Abstract

Professional agents must execute workflows correctly and deliver results supported by the evidence they use. We introduce FinREALBench, a benchmark of 100 executable financial workflows spanning ten professional categories. Rather than assessing trajectories and answers in isolation, its evaluation contract follows evidence from source acquisition through transformation and repair into the submitted artifact. Process, Outcome, and Integrity assess distinct parts of this contract, while Oracle replay and targeted self-tests exercise the implemented evaluators. Across 16 model–harness systems, we examine where execution goes wrong and whether subsequent repairs reach the delivered answer. In a source-audited ownership task, all 16 submissions are schema-valid and arithmetically consistent, yet three retain an incorrect denominator. Same-model trajectories reveal two repair paths: correction from an observed filing passage and substitution of an unsupported value. Beyond this task, 133 of 868 schema-valid, fully complete artifacts receive evidence-consistency scores at most 0.5, spanning 56 tasks and 12 settings. These analyses characterize the Execution Reliability Gap: agents struggle to perform the work correctly, and complete submissions can remain unsupported. FinREALBench localizes failures along the evidence-to-delivery chain, identifying stronger execution skills, requirement-aligned checks, and evidence-grounded repair as complementary development targets.

## 1 Introduction

Language-model agents are moving from conversational assistance into professional workflows that require search, tools, code, documents, and structured delivery. In these settings, execution reliability means producing a correct work product whose sources, transformations, and claims can be checked before downstream use.

Consider a share-ownership workflow. An agent must locate the relevant filing, extract the correct share counts, preserve their dates and scope, calculate ownership, and deliver the requested result. An extraction error can corrupt the calculation before any answer is produced; an unsupported repair can carry that error into a complete-looking submission. Our trajectories exhibit both difficulties. Real-world reliability therefore requires examining whether the work was performed correctly as well as whether the delivered answer is trustworthy.

The evaluation challenge is to follow evidence through the work. A source check establishes whether an input matches the relevant document. Execution records show how the input was transformed and whether a detected error led to a supported repair. The delivered artifact shows which values ultimately reached the user. Linking these observations within the same task makes four transitions inspectable: source to input, input to computation, error to repair, and validated result to artifact. This separates an incorrect calculation from a correct calculation over incorrect inputs, and error recognition from effective correction.

Existing work addresses important parts of this challenge. Financial benchmarks measure numerical reasoning, retrieval, or report quality (Chen et al., 2021; Zhu et al., 2021; Islam et al., 2023; Hu et al., 2025). Tool and environment benchmarks test API use and interactive completion (Qin et al., 2024; Patil et al., 2025; Mialon et al., 2024; Zhou et al., 2024). Procedure-aware evaluation exposes violations hidden by task success, verification-based

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

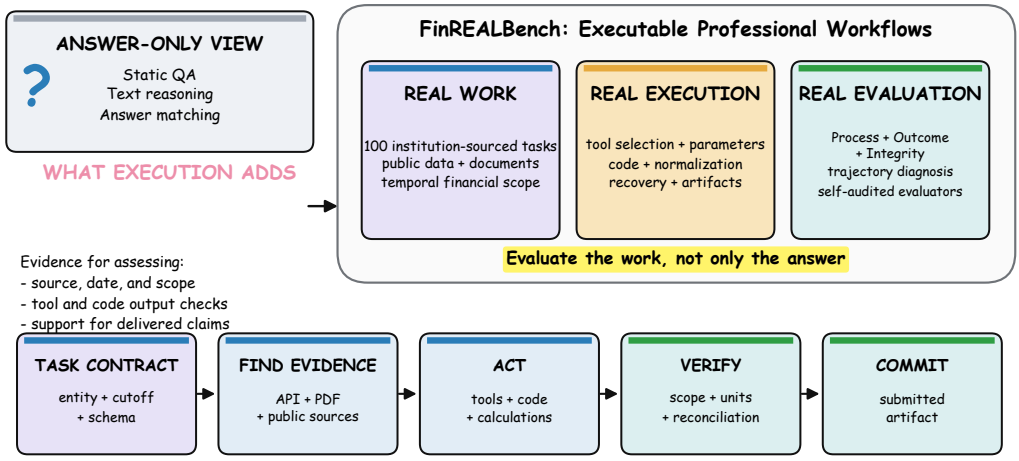



Figure 1: From task contract to assessable delivery. FinREALBench retains tool observations, computations, and artifacts so that execution, validation, and delivered claims can be evaluated together. The bottom row shows the workflow stages examined by the evaluation contract; verification and evidence support are assessed from the observed execution.

systems audit beliefs or actions, and workflow benchmarks reveal framework sensitivity (Cao et al., 2026a; Lin et al., 2026; Yuan et al., 2026; Guan et al., 2026; Niu et al., 2026). The remaining diagnostic question is where the connection between evidence and the delivered result breaks. Financial workflow benchmarks already establish the importance of executable tools, trajectory quality, structured outputs, and task-specific verifiers (Lu et al., 2026; Cao et al., 2026b; Bek et al., 2026). These views do not by themselves identify whether a wrong submitted value arose during acquisition, transformation, repair, or artifact construction. We address this narrower question by combining executable task contracts, source checks, observed repairs, and within-task trajectory–artifact comparisons.

Our core idea is to evaluate evidence-to-commitment traceability: whether a result remains supported as evidence moves through execution, repair, and delivery. We propose FinREALBench, comprising 100 financial tasks with real sources, tool and code execution, and assessable deliverables. Process records how the work was performed, Outcome evaluates the delivered artifact, and Integrity enforces the evaluation boundary. These are distinct assessment views rather than a single proxy for reliability. Oracle replay, invariants, positive controls, provenance tests, and targeted mutations exercise whether each implemented evaluator both rejects targeted errors and accepts valid implementation variation. Finance makes the evidence links concrete through publication dates, security identities, accounting scopes, units, and reconciliation identities.

The resulting evidence characterizes the Execution Reliability Gap (ERG) at two levels: difficulty performing the work correctly and unreliable answers even when delivery is complete. In an ownership task, all 16 systems submit schema-valid, arithmetically consistent answers, but three use a denominator that disagrees with the filing. Two same-model trajectories recognize extraction errors yet differ in repair: one uses an observed source passage; the other substitutes an unsupported value. A broader paired-artifact analysis finds weak evidence support in 133 of 868 fully complete, schema-valid answers across 56 tasks and 12 settings. The source audit explains a concrete failure path; the broader analysis establishes that unsupported complete answers are not confined to that example.

Our contributions are summarized as follows:

- Executable professional workflows. FinREALBench evaluates 100 end-to-end financial tasks with open-world sources, controlled execution, observable trajectories, and machine-checkable artifacts.
- Evidence-linked evaluation. FinREALBench traces four transitions: source → input, input → computation, error → repair, and validated result → artifact. Process, Outcome, and Integrity provide separate assessment views.
- Tested diagnosis of execution reliability. Oracle replay, positive controls, and 1,702 targeted mutants test the implemented evaluators. A same-task source audit and a

broader paired-artifact analysis then distinguish execution errors, unsupported repairs, and weakly supported complete answers.

## 2 Related Work

Financial evaluation. FinQA, TAT-QA, and ConvFinQA evaluate numerical reasoning over reports, tables, and conversations; FinanceBench adds open-book financial questions with answer evidence (Chen et al., 2021; Zhu et al., 2021; Chen et al., 2022; Islam et al., 2023). FinAgentBench, FinSearchComp, InvestorBench, and Deep FinResearch Bench extend evaluation toward retrieval, open-domain research, market decisions, and report generation (Choi et al., 2025; Hu et al., 2025; Li et al., 2025; Haque et al., 2026). The closest workflow-oriented efforts include FinToolBench, FinTrace, and FinSkillBench, which address executable financial tools, trajectory evaluation, and point-in-time tasks with structured outputs and verifiers (Lu et al., 2026; Cao et al., 2026b; Bek et al., 2026). Together, these works establish that tool execution, trajectory quality, and structured verification each matter. FinREALBench asks a complementary diagnostic question: when a submitted claim is wrong or unsupported, which evidential transition failed, and did the observed repair restore it? Its evaluation contract links source-grounded requirements, execution and repair evidence, submitted artifacts, and evaluator self-tests within the same task.

Interactive agents and environments. ToolLLM, StableToolBench, BFCL, and MTU-Bench evaluate tool use and API interaction (Qin et al., 2024; Guo et al., 2024; Patil et al., 2025; Wang et al., 2025). GAIA, AgentBench, and WebArena broaden the setting to information seeking and interactive environments (Mialon et al., 2024; Liu et al., 2024; Zhou et al., 2024). SupChain-Bench and DuMateBench emphasize long-horizon workflows and challenging execution environments (Guan et al., 2026; Niu et al., 2026). FinREALBench links execution to domain requirements: acquiring the correct source, preserving temporal and financial scope, implementing transformations, and producing a supported artifact. Its trajectories distinguish errors introduced during the work from errors retained at delivery; model–harness comparisons treat the deployed system as the unit of comparison.

Verification and commitment. ReAct connects reasoning with actions and observations, while Reflexion uses verbal feedback to guide subsequent decisions (Yao et al., 2023; Shinn et al., 2023). Procedure-Aware Evaluation distinguishes task success from policy and integrity compliance (Cao et al., 2026a); VIGIL and SAVeR introduce verification before commitment for security and reasoning faithfulness (Lin et al., 2026; Yuan et al., 2026). FinREALBench complements these methods by assessing task execution, observed checks and repairs, and delivered correctness together. Its emphasis is not verification as a generic extra step, but whether the check is capable of falsifying the task requirement at risk and whether a supported repair reaches the artifact. Reason–Act–Verify–Commit expresses the resulting development direction: improve the underlying execution skills while detecting and repairing errors throughout the workflow.

## 3 FinREALBench: Real Work, Real Execution, Real Evaluation

### 3.1 The workflow, not the question, is the evaluation unit

The unit of evaluation is an Executable Professional Workflow: a task contract, an observed execution, and a submitted work product. The task inventory is derived from recurring needs supplied by financial-institution users and reviewed by finance experts; identifying organizational details are removed from released prompts. Each task specifies the professional role and expected artifact. Contracts fix dates, universes, formulas, financial scope, and output semantics where needed, while agents identify and execute the required procedure.
The contract makes four evidential transitions inspectable. Provenance asks whether a source supports the selected input; transformation asks whether the input is carried correctly through computation; repair asks whether a detected error is replaced with supported evidence; and commitment asks whether the validated result reaches the submitted artifact. These transitions are analytical links, not an additional scalar score. Task-specific checks and retained trajectory evidence assess them under the requirements of each workflow.
The 100 tasks cover ten workflows: earnings quality (28), market surveillance (13), fixed-income and credit (10), data-model audit (10), valuation scenarios (9), corporate actions (9),

macro hedging (7), portfolio risk (7), fund-manager analysis (5), and execution liquidity (2). They combine retrieval, document inspection, code, calculation, validation, and structured delivery. Appendix Table A1 reports the modality, access-mode, and difficulty distribution.

An example contract: event clocks and value conservation. FTB-053 asks agents to verify NVIDIA's 2024 ten-for-one split against SEC disclosures, distinguish announcement, legal effectiveness, distribution, and adjusted trading, and compute the share and price transformations. For the supplied example of 100 shares at $1,200, the required calculation is 1,000 shares at $120 with unchanged theoretical position value. The contract supplies event anchors and constants; the execution requirement is to verify them against primary sources and preserve their distinct meanings. This task therefore tests source verification and reconciliation as well as calculation. Other tasks require agents to retrieve the target values themselves.

### 3.2 Real execution environment

Each task runs in a fresh Docker container with a read-only task package, a writable workspace, network access to public data, and resource limits. The agent receives the task, answer schema, an audit contract specifying required execution evidence, and approved user documents. Task contracts and schemas may contain prescribed values and constraints, as in FTB-053. Evaluator reference files, graders, source manifests, frozen evaluator inputs, and prior runs remain host-side.

Agents must actually query sources, read documents, write and execute code, process files, recover from tool errors, validate results, and write a machine-checkable answer.json. Backend adapters normalize native messages, tool calls, audit events, and artifacts into a common trajectory. This distinction is central: the benchmark evaluates an agent doing the work, not an LLM describing how it would do the work.

| Stage | Boundary / input | Protocol output |
|---|---|---|
| Task | Task, answer schema, and required execution evidence | Fresh isolated container with a read-only task package |
| Execute | Public sources, approved documents, and writable workspace | Tool use, code execution, validation, and answer.json |
| Record | Native messages, tool calls, events, files, and hashes | Normalized trajectory and retained artifacts |
| Evaluate | Host-only reference, snapshots, and policies | Process and Outcome scores plus the Integrity gate |

Table 1: Real-execution protocol. Evaluator inputs remain host-side; the tested agent must acquire evidence and produce the work artifact inside the isolated environment.

### 3.3 Process- and outcome-aware evaluation

Outcome asks whether the submitted work product is correct. Deterministic checks compare structured fields, keyed rows, sets, orderings, reconciliations, and numeric values under task-specific tolerances. An identity-blind judge complements these checks along six dimensions: factual correctness, completeness, financial scope, temporal correctness, units and calculation, and evidence consistency.

Process consumes observable trajectory evidence. Deterministic checkpoints test configured tool calls, source locators, parameters, retrieval times, date windows, artifacts, hashes, and assertions. An identity-blind judge assesses nine dimensions: problem decomposition, tool selection, query parameters, source quality, temporal control, data normalization, model/code implementation, execution validation, and output auditability. Each judgment records applicability, a score, evidence, and a reason. A claim that a check was performed is distinct from an observed tool call or retained validation artifact.

Integrity is a hard gate rather than an averaged capability. It reports prohibited hidden-file access, unsupported provenance, and other critical contract violations separately. An accurate answer without support, a supportable process with the wrong vintage, and an integrity violation are therefore distinct outcomes rather than one collapsed correctness label.

Let $D_t^P, L_t^P$ and $D_t^A, L_t^A$ denote the deterministic and judge components of Process and Outcome for task $t$. The composite scores are $P_t = 0.5D_t^P + 0.5L_t^P$ and $A_t = 0.75D_t^A +$

$0.25L_t^A$. The secondary overall score is

$$O_t = 0.7A_t + 0.3P_t. \tag{1}$$

We report $P_t$ and $A_t$ separately: they assess distinct properties with different rubrics and component weights. The ERG diagnosis uses dimension profiles and observable execution evidence, rather than subtracting these two aggregate scores.

## 4 Validating the Evaluator

FinREALBench treats evaluator construction as a tested software process. Oracle acceptance is necessary but insufficient: an evaluator can accept its reference while also accepting targeted errors or rejecting valid alternatives. We therefore test both rejection sensitivity and implementation tolerance.

### 4.1 Self-validation protocol

For each task, the construction Oracle is replayed in isolation with only its contract, schema, source manifest, frozen inputs, explicit assets, and task-local computation. Networking, references, graders, trace policies, historical runs, and other tasks remain unavailable, and every Oracle runs at least twice. Generic invariants cover schema validity, finite values, unique keys, date windows, ordering, and reconciliation; finance-specific invariants cover units, accounting identities, portfolio contributions, fixed-income cash flows, signal and execution dates, and PDF evidence locators.

Evaluator sensitivity and tolerance are tested together. Starting from valid answers and traces, targeted non-equivalent mutations inject errors in schema, values, signs, units, set membership, ordering, summaries, financial scope, temporal leakage, source evidence, hidden-file access, and process checkpoints. A mutant is killed only when its intended check triggers. Positive controls preserve validity while varying numeric values within tolerance, retry behavior, JSON key order, and valid tool order. For document tasks, provenance stress tests require hash-verified page and table parsing, field-level source/page/bounding-box/raw-text lineage, derivation records, and fail-closed behavior when selected evidence is removed, replaced, or corrupted.

### 4.2 All 100 tasks pass the release gate

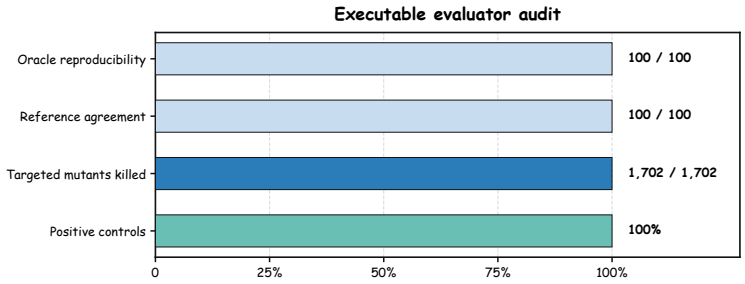



Figure 2: FinREALBench validates its evaluators before evaluating agents. All 100 tasks reproduce under Oracle execution, all 1,702 targeted evaluator mutations are detected, all positive controls pass, and no critical survivor remains.

All 100 construction Oracles pass isolated replay and reproduce their references; all 1,702 implemented targeted mutants trigger their intended checks; and every implemented positive control passes (Figure 2). Four provenance-intensive document tasks contribute 330 field-level evidence and derivation records. Oracle replay tests acceptance of the intended solution, targeted mutants test rejection of specified non-equivalent errors, and positive controls test tolerance to valid variation. Together, these self-tests exercise the implemented numerical, temporal, financial-scope, provenance, and process-integrity checks. The dimension-level LLM judgments use the separate rubrics described in Appendix B.

## 5 Experimental Setup

We evaluate a frozen matrix of 16 deployed model–harness systems on the 100-task standard suite. The matrix spans Codex, Claude Code, OpenClaw, and OpenHands together with

GPT-5.6-sol, GPT-5.5, GPT-5.4, GPT-5.4-mini, Claude Opus 4.8, Claude Sonnet 4.6, and Claude Haiku 4.5 endpoints. Each cell represents the complete deployed system, including its native tool protocol, context management, recovery behavior, and artifact commitment; same-model contrasts characterize the evaluated deployed pairings, including runtime and provider effects.

Within each setting, we task-macro average canonical judged runs. Across settings, headline aggregates are equal-setting macro averages, ensuring that each model–harness system receives equal weight. Dimension aggregates average applicable judged dimensions. Endpoint metadata and canonical-run selection associate scores with their requested model–harness settings, and the manuscript ships the machine-readable frozen values behind the result tables and figures. Appendix D describes the reporting and aggregation conventions.

Our primary quantities are Process $P$ and Outcome $A$. Overall from Equation 1 supports system-level comparison. We examine process and outcome dimensions separately, compare Outcome across difficulty tiers, and compare the same Overall metric across model–harness pairings.

## 6 The Execution Reliability Gap

We organize the diagnosis around three questions: where do execution weaknesses appear, how do errors and repairs enter the final artifact, and how broadly do complete answers lack evidence support? The full matrix provides context; a source-audited task connects execution to delivery; the paired-artifact analysis measures the broader coexistence of completeness and weak support.

### 6.1 Main results

| Agent | Model Name | FinREALBench | | |
|---|---|---|---|---|
| | | Overall | Process | Outcome |
| Codex | GPT-5.6-sol | .667 | .750 | .632 |
| | GPT-5.5 | .646 | .753 | .600 |
| | GPT-5.4 | .686 | .748 | .659 |
| | GPT-5.4-mini | .611 | .704 | .571 |
| Claude Code | Claude Opus 4.8 | .560 | .652 | .521 |
| | Claude Sonnet 4.6 | .546 | .640 | .506 |
| | Claude Haiku 4.5 | .400 | .443 | .381 |
| OpenClaw | GPT-5.6-sol | .645 | .725 | .610 |
| | GPT-5.4 | .517 | .582 | .489 |
| | Claude Opus 4.8 | .496 | .593 | .455 |
| | Claude Sonnet 4.6 | .489 | .597 | .443 |
| | Claude Haiku 4.5 | .317 | .345 | .305 |
| OpenHands | GPT-5.6-sol | .646 | .717 | .615 |
| | Claude Opus 4.8 | .485 | .557 | .455 |
| | Claude Sonnet 4.6 | .518 | .596 | .485 |
| | Claude Haiku 4.5 | .373 | .290 | .409 |
| Equal-setting macro | | .538 | .606 | .508 |

Table 2: Main results across 16 reported agent–model systems. Scores are equal-task macro averages on the 100-workflow standard suite. Bold marks the highest value within each Agent block and metric.

Across the 16 settings, equal-setting macro scores are 0.606 Process, 0.508 Outcome, and 0.538 Overall, with Process and Outcome retaining their distinct rubrics. GPT-5.4 with Codex has the highest Overall (0.686) and Outcome (0.659), while GPT-5.5 with Codex has the highest Process score (0.753).

### 6.2 Execution weaknesses extend beyond final checking

Acquisition, transformation, and checking all remain imperfect. Figure 3a reports source quality at 0.633, data normalization at 0.657, and model/code implementation at 0.720. Execution validation is lowest at 0.508, followed by output auditability at 0.549; problem decomposition scores 0.806. These rubric averages describe assessed capabilities. The

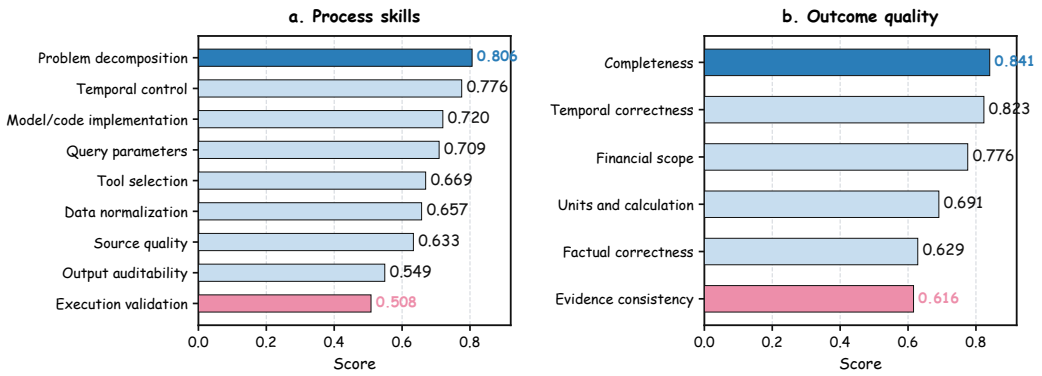



Figure 3: Execution and delivery exhibit distinct weaknesses. Left: acquisition, transformation, implementation, and checking dimensions. Right: completeness, correctness, and evidence support. Relatively high planning or completeness scores do not certify correct execution. Values are equal-setting macro means over applicable dimensions.

trajectories locate specific failures: FTB-049 extracts an incorrect denominator, while the additional FTB-085 case applies an incorrect date conversion (Appendix H). Both errors enter during execution, before the submission decision.

Delivered correctness and evidence support remain weak. Figure 3b reports factual correctness at 0.629 and evidence consistency at 0.616, compared with completeness at 0.841. To move from capability profiles to a concrete failure path, we next link source inputs, repairs, and submitted values within one task.

### 6.3 From source errors to submitted answers

Source agreement is distinct from internal consistency. In FTB-049, all 16 submissions pass the recorded schema check and reproduce their submitted ownership calculation within $10^{-6}$ percentage points. Yet three use a shares-outstanding denominator inconsistent with the original filing. Recomputing the formula would accept these answers; checking the denominator against the filing exposes the error. This within-task audit separates structural validity, arithmetic consistency, and source agreement (Appendix H.1).

Error recognition leads to different repairs. Two Opus trajectories on this task identify extraction errors. Claude Code receives the filing passage with the correct denominator, replaces the extracted value, and retains the recalculated ratio in its answer. OpenClaw hardcodes an unsupported replacement, calculates with it, and submits that value. The observed distinction is the evidence used for the repair, not whether the agent notices an error or performs arithmetic. Appendix H traces the source observations, replacement inputs, and final artifacts.

Populating the answer need not restore its evidential basis. FTB-009 illustrates another path: after retrieval fails, a disclosed synthetic-data fallback fills a historical ranking with simulated observations. The answer is complete but does not supply the requested historical evidence. These cases locate failures from acquisition through repair to delivery; the following analysis examines the broader distribution of evidence support without assigning these specific mechanisms to every weakly supported answer.

The presence of a check is not the same as requirement-aligned validation. The failing FTB-049 submissions pass structural and arithmetic checks, but those checks cannot falsify the incorrect source denominator. Appendix H provides two additional mechanisms: FTB-085 checks the observation count but proceeds with 249 rows despite an explicit 250-row requirement, and FTB-097 reports validation success from a partial checker while the full schema still fails. These cases do not estimate mechanism prevalence; they show why validation must be interpreted against the requirement it is meant to protect.

### 6.4 Even complete answers can lack evidence support

Evidence support within complete submissions. Of 1,600 runs, 1,595 have both applicable scores and 868 pass schema validation with completeness 1.0. The remaining 732 do not meet this joint condition, which is distinct from having no answer. Within the 868-answer subset, 133 (15.3%) have evidence consistency at most 0.5, including 41 scoring zero (Figure 4a).

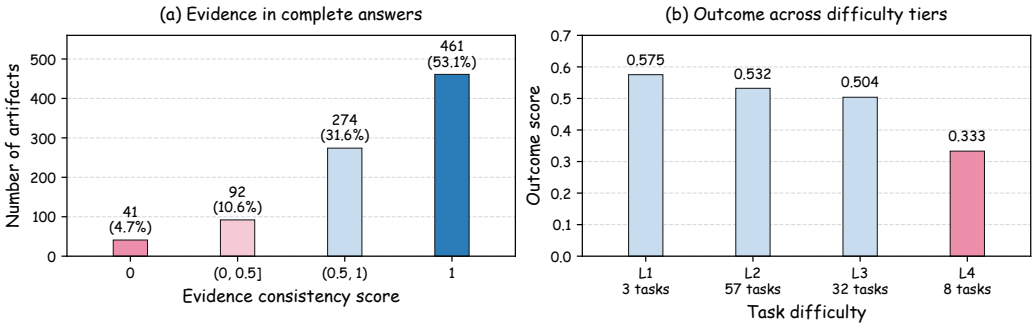



Figure 4: Delivery quality within complete answers and across task difficulty. (a) Evidence-consistency scores for 868 schema-valid answers with completeness 1.0; labels show counts and shares. The first two groups contain 133 answers (15.3%). (b) Equal-setting macro Outcome within each difficulty tier, using all reported settings; task counts refer to the 100-task inventory.

These 133 span 56 tasks and 12 settings. The distribution therefore describes a conditional subset of submissions, not successful execution across the benchmark. Appendix H.1 gives the protocol; 0.5 is a descriptive midpoint.

### 6.5 Task difficulty and deployed pairings

Difficulty tiers.
Figure 4b compares Outcome under the same scoring definition across four difficulty tiers: 0.575 at L1, 0.532 at L2, 0.504 at L3, and 0.333 at L4. The tiers contain 3, 57, 32, and 8 tasks, respectively. L4 has the lowest Outcome in this inventory, with a clear separation from L3. This comparison locates challenging tasks within the benchmark rather than isolating difficulty from task composition.
Model–harness pairings.
Table 2 also shows variation within the same model. GPT-5.4 scores 0.686 Overall with Codex and 0.517 with OpenClaw, while GPT-5.6-sol spans 0.023 across three harnesses. These observed pairings include differences in tool protocols, context management, recovery, and artifact submission. Model and harness must therefore be reported together when interpreting execution reliability.

## 7 Implications: Reason–Act–Verify–Commit

ReAct connects reasoning to actions and observations (Yao et al., 2023); verification-based systems introduce checks before commitment (Lin et al., 2026; Yuan et al., 2026). Our diagnosis does not motivate indiscriminate extra checking. It calls for stronger execution fundamentals and checks tied to the requirements that can invalidate a delivered claim. Reason–Act–Verify–Commit describes this coupled target, not a final check added to an already reliable workflow.

Strengthen execution throughout the workflow. Agents must retrieve the right material, extract the relevant fields, preserve dates and units, and implement the requested transformation. The observed extraction and normalization failures make these skills training targets in their own right. Checks should accompany these operations so that an incorrect intermediate result can be repaired before it propagates.

Match the check to the requirement. FTB-049 distinguishes a schema check, an arithmetic check, and a source check. Passing the first two does not establish the third. A requirement-aligned check must identify the claim that could be false, inspect evidence capable of falsifying it, and preserve the supported value through delivery. Repeating an internally consistent calculation cannot establish the entity, date, unit, or accounting scope of its inputs.

Repair the evidence link before commitment. A repair must restore the violated requirement: a filing passage can justify a revised denominator; an unsupported constant cannot. Where partial delivery is allowed, unresolved fields should remain explicit. Execution, inspection, and evidence-grounded repair are complementary targets for trajectory supervision.

## 8 Conclusion

FinREALBench makes 100 professional financial workflows inspectable through an evidence-linked evaluation contract. It follows source-to-input, input-to-computation, error-to-repair, and validated-result-to-artifact transitions, while Oracle replay, targeted mutants, and positive controls exercise the implemented evaluators.

The Execution Reliability Gap spans both performing the work correctly and delivering a trustworthy result. A same-task audit shows that schema-valid, arithmetically consistent answers can retain source errors; trajectories distinguish source-grounded correction from unsupported replacement. Across the broader evaluation, 133 of 868 fully complete, schema-valid answers have weak evidence support. FinREALBench therefore turns reliability from a single end score into a localized diagnosis of provenance, transformation, repair, and commitment. The findings make execution skills, requirement-aligned checking, and evidence-grounded repair complementary targets: a corrected calculation must use a justified input, and that correction must reach the delivered answer.

## Limitations and Responsible Use

FinREALBench evaluates verifiable financial research across ten workflow categories. Its contracts span open retrieval, supplied documents, and verification of prescribed facts; the measured capability is execution under the stated contract. The frozen 16-setting matrix describes deployed model–harness pairings, including runtime and provider effects. Live-source availability can change between runs, while difficulty-tier comparisons reflect different task compositions. Evaluator mutation tests validate implemented checks; rubric-based dimension scores additionally depend on LLM judgment. These distinctions define how the released results should be interpreted. Benchmark outputs are research artifacts and do not constitute investment advice.

## AI Use Statement

Generative AI tools were used to aid and polish writing and for retrieval and discovery, including finding related work. Details are described in the paper. The authors reviewed all AI-assisted content and take responsibility for the final text, claims, citations, and artifacts.

## References

Jermyn Zhen Yong Bek, Zhuang Qiang Bok, and Zhongtian Sun. Finskillbench: Evaluating ai agents and domain skills for investment management. CoRR, abs/2608.18099, 2026. URL https://arxiv.org/abs/2608.18099.

Hongliu Cao, Ilias Driouich, and Eoin Thomas. Beyond task completion: Revealing corrupt success in llm agents through procedure-aware evaluation. CoRR, abs/2603.03116, 2026a. URL https://arxiv.org/abs/2603.03116.

Yupeng Cao, Haohang Li, Weijin Liu, Wenbo Cao, Anke Xu, Lingfei Qian, Xueqing Peng, Minxue Tang, Zhiyuan Yao, Jimin Huang, K. P. Subbalakshmi, Zining Zhu, Jordan W. Suchow, and Yangyang Yu. Fintrace: Holistic trajectory-level evaluation of llm tool calling for long-horizon financial tasks. CoRR, abs/2604.10015, 2026b. URL https://arxiv.org/abs/2604.10015.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Kenneth Huang, Bryan R. Routledge, and William Yang Wang. Finqa: A dataset of numerical reasoning over financial data. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021, pp. 3697–3711. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.EMNLP-MAIN.300. URL https://doi.org/10.18653/v1/2021.emnlp-main.300.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. Convfinqa: Exploring the chain of numerical reasoning in conversational finance

question answering. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022, pp. 6279–6292. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.421. URL https://doi.org/10.18653/v1/2022.emnlp-main.421.

Chanyeol Choi, Jihoon Kwon, Alejandro Lopez-Lira, Chaewoon Kim, Minjae Kim, Juneha Hwang, Jaeseon Ha, Hojun Choi, Suyeol Yun, Yongjin Kim, and Yongjae Lee. Finagentbench: A benchmark dataset for agentic retrieval in financial question answering. CoRR, abs/2508.14052, 2025. doi: 10.48550/ARXIV.2508.14052. URL https://doi.org/10.48550/arXiv.2508.14052.

Shengyue Guan, Yihao Liu, and Lang Cao. Supchain-bench: Benchmarking large language models for real-world supply chain management. CoRR, abs/2602.07342, 2026. URL https://arxiv.org/abs/2602.07342.

Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models. In Findings of the Association for Computational Linguistics: ACL 2024, pp. 11143–11156. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.664. URL https://aclanthology.org/2024.findings-acl.664/.

Mirazul Haque, Antony Papadimitriou, Samuel Mensah, Zhiqiang Ma, Zhijin Guo, Joy Prakash Sain, Simerjot Kaur, Charese Smiley, and Xiaomo Liu. Deep finresearch bench: Evaluating ai's ability to conduct professional financial investment research. CoRR, abs/2604.21006, 2026. URL https://arxiv.org/abs/2604.21006.

Liang Hu, Jianpeng Jiao, Jiashuo Liu, Yanle Ren, Zhoufutu Wen, Kaiyuan Zhang, Xuanliang Zhang, Xiang Gao, Tianci He, Fei Hu, Yali Liao, Zaiyuan Wang, Chenghao Yang, Qianyu Yang, Mingren Yin, Zhiyuan Zeng, Ge Zhang, Xinyi Zhang, Xiying Zhao, Zhenwei Zhu, Hongseok Namkoong, Wenhao Huang, and Yuwen Tang. Finsearchcomp: Towards a realistic, expert-level evaluation of financial search and reasoning. CoRR, abs/2509.13160, 2025. doi: 10.48550/ARXIV.2509.13160. URL https://doi.org/10.48550/arXiv.2509.13160.

Pranab Islam, Anand Kannappan, Douwe Kiela, Rebecca Qian, Nino Scherrer, and Bertie Vidgen. Financebench: A new benchmark for financial question answering. CoRR, abs/2311.11944, 2023. URL https://arxiv.org/abs/2311.11944.

Haohang Li, Yupeng Cao, Yangyang Yu, Shashidhar Reddy Javaji, Zhiyang Deng, Yueru He, Yuechen Jiang, Zining Zhu, K. P. Subbalakshmi, Jimin Huang, Lingfei Qian, Xueqing Peng, Jordan W. Suchow, and Qianqian Xie. INVESTORBENCH: A benchmark for financial decision-making tasks with llm-based agent. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025, pp. 2509–2525. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.126/.

Junda Lin, Zhaomeng Zhou, Zhi Zheng, Shuochen Liu, Tong Xu, Yong Chen, and Enhong Chen. VIGIL: Defending LLM agents against tool-stream injection via verify-before-commit. In Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 9764–9785. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.443. URL https://aclanthology.org/2026.acl-long.443/.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. In The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Jiaxuan Lu, Kong Wang, Yemin Wang, Qingmei Tang, Hongwei Zeng, Xiang Chen, Jiahao Pi, Shujian Deng, Lingzhi Chen, Yi Fu, Kehua Yang, and Xiao Sun. Fintoolbench: Evaluating llm agents for real-world financial tool use. CoRR, abs/2603.08262, 2026. URL https://arxiv.org/abs/2603.08262.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

Zechun Niu, Yukun Zhao, Jiaxin Zhang, Xu Shen, Jinhua Si, Han Tian, Can Xu, Yunfan Song, Jiaxin Mao, Yansong Gao, Yuchen Li, Jianmin Wu, Lingyong Yan, Shuaiqiang Wang, and Dawei Yin. Dumatebench: Evaluating autonomous agents in complex real-world workflows. CoRR, abs/2608.26546, 2026. URL https://arxiv.org/abs/2608.26546.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): from tool use to agentic evaluation of large language models. In Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025. OpenReview.net, 2025. URL https://openreview.net/forum?id=2GmDdhBdDk.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Pei Wang, Yanan Wu, Noah Wang, Jiaheng Liu, Xiaoshuai Song, Z. Y. Peng, Ken Deng, Chenchen Zhang, Jiakai Wang, Junran Peng, Ge Zhang, Hangyu Guo, Zhaoxiang Zhang, Wenbo Su, and Bo Zheng. Mtu-bench: A multi-granularity tool-use benchmark for large language models. In The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025. OpenReview.net, 2025. URL https://openreview.net/forum?id=6guG2OlXsr.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Wenhao Yuan, Chenchen Lin, Jian Chen, Jinfeng Xu, Xuehe Wang, and Edith Cheuk-Han Ngai. Verify before you commit: Towards faithful reasoning in LLM agents via self-auditing. In Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 31201–31225. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1440. URL https://aclanthology.org/2026.acl-long.1440/.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021, pp. 3277–3287. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.ACL-LONG.254. URL https://doi.org/10.18653/v1/2021.acl-long.254.

# A Task Inventory and Benchmark Construction

Table A1: Task coverage. Frozen evaluator inputs are not exposed to tested agents.

| Primary modality | # | Difficulty | # |
|---|---|---|---|
| Structured table | 54 | L1 | 3 |
| Time series | 25 | L2 | 57 |
| PDF | 13 | L3 | 32 |
| Text | 5 | L4 | 8 |
| Mixed | 3 | | |
| **Access mode** | | **Tasks** | |
| Live retrieval | 85 | User documents | 13 |
| Live + documents | 2 | Total | 100 |

The 100-task inventory is derived from recurring professional financial-analysis needs and reviewed as executable work products. Task contracts specify the professional role, expected artifact, financial scope, temporal boundary, output schema, source policy, and acceptance conditions. Organizational details are removed from released prompts while the analytical semantics required for evaluation are preserved. Contracts may supply facts or schema constants to be independently checked: FTB-053 provides event anchors and split mechanics, while requiring SEC-source verification and reconciliation. Prescribed contract values are visible inputs; evaluator references and private checks remain host-side.

The suite covers earnings quality, market surveillance, fixed-income and credit, data-model audit, valuation scenarios, corporate actions, macro hedging, portfolio risk, fund-manager analysis, and execution liquidity. Tasks combine public retrieval, supplied documents, financial libraries, code execution, normalization, calculation, validation, evidence construction, and structured delivery. The modality and access distribution in Table A1 records the primary task surface; a single workflow may exercise several modalities and tools.

# B Evaluation Dimensions and Rubrics

Process. Problem decomposition measures whether the trajectory identifies the required subproblems and constraints. Tool selection and query parameters measure whether the chosen interface and call semantics fit the task. Source quality assesses authority and provenance. Temporal control checks cutoffs, observation dates, publication dates, and leakage. Data normalization covers identifiers, units, missing values, joins, and aligned windows. Model/code implementation evaluates whether formulas and code instantiate the requested method. Execution validation measures tests, assertions, reconciliations, and independent checks. Output auditability measures whether claims and artifacts can be traced to sources and transformations.

Outcome. Factual correctness assesses values and claims against locked references. Completeness covers required objects, periods, fields, and conclusions. Financial scope checks instrument, accounting, adjustment, and consolidation semantics. Temporal correctness checks as-of and window requirements. Units and calculation cover scale, precision, signs, formulas, and reconciliations. Evidence consistency asks whether retained evidence supports the submitted result.

Integrity and aggregation. Integrity is evaluated as a separate gate over isolation, identity, provenance, and completion validity. Deterministic and identity-blind judge components are combined within Process and Outcome as defined in Equation 1; Overall is retained as a secondary system-level summary. Dimension means include only tasks for which a dimension is applicable, while task and setting aggregates use the frozen canonical-run rules described in Appendix I.

Judgment records and evidence. The judge returns a structured record for each fixed rubric dimension: its identifier, applicability, a score between zero and one, supporting evidence, and a reason. Process judgments use observed trajectories, tool results, audit events, and artifacts. Outcome judgments compare the submitted answer with the task, schema, reference, and evaluator inputs; phrasing and writing style are not scoring targets. Host-side

reference material supplies evaluation context and is not evidence that an agent accessed a hidden file.

Reading a dimension profile. Problem decomposition assesses identification of entities, indicators, time windows, constraints, and subtasks. Execution validation instead assesses checks for missing or duplicate data, sample sizes, boundary conditions, and result invariants. Completeness assesses coverage of requested objects, fields, periods, and conclusions; evidence consistency assesses alignment with verifiable data, calculations, and reference evidence. A higher mean on one dimension describes a stronger assessment under that rubric. Establishing whether the same artifact is both complete and unsupported requires inspecting its task-level records.

## C Full Model–Harness Results

The reported matrix retains settings with more than 95 answer artifacts in the original 19-setting matrix. This post-run filter excludes GPT-5.4/OpenHands (30 answers), GPT-5.4-mini/OpenClaw (70), and GPT-5.4-mini/OpenHands (78). All 100 task runs in each retained setting contribute to the main scores, including missing answers and execution failures. Table 2 reports the resulting 16 systems. The three machine-readable snapshots with the suffix _reported.csv provide system, dimension, and difficulty aggregates. GenericAgent, QwenPaw, and the hard-task suite remain outside this comparison.

## D Score Aggregation and Reporting

Within each setting, task-macro means summarize canonical judged runs. Headline summaries then average the 16 setting means with equal weight. Dimension profiles include applicable judged dimensions. Difficulty-tier summaries apply the same Outcome definition within each tier and average the resulting setting means; L1, L2, L3, and L4 contain 3, 57, 32, and 8 tasks, respectively.

Process and Outcome use different rubrics and deterministic–judge mixtures. Their separate values describe execution and delivery; their subtraction is not a measure of capability retained or lost. The main-text diagnosis instead reports the process and outcome dimension profiles, including the 0.298 decomposition–validation separation and the 0.224 completeness–evidence separation. These are contrasts between rubric dimensions, with the component scores shown explicitly.

Harness ranges use unrounded Overall scores from results_snapshot_reported.csv. The reported ranges for GPT-5.4 and GPT-5.6-sol are 0.168964 and 0.022592, respectively, reported as 0.169 and 0.023; these compare two and three harnesses, respectively. The frozen summaries describe the evaluated settings; shared tasks and shared model or harness components are not treated as independent repeated trials.

## E Workflow- and Task-Level Audit Views

Workflow-level analysis preserves the benchmark's operational categories rather than pooling unlike financial tasks. For each workflow, Process, Outcome, Overall, and every applicable dimension can be recomputed from the released canonical task records. Task-level records additionally expose the required modality, access mode, temporal constraint, artifact schema, and evaluation checkpoints. This supports paired analyses that hold the task fixed while varying the model–harness system, and stratified analyses that hold the workflow fixed while examining post-action reliability.

A workflow-level audit can trace source localization, parameterization, normalization, reconciliation, validation, evidence binding, and final commitment. Keeping the underlying observations alongside the scores enables reviewers to distinguish incorrect computation, unsupported claims, and incomplete execution.

## F Action-Level Failure Taxonomy

The audit taxonomy organizes potential failures by their earliest observable point in execution. It defines six families: localization errors select the wrong entity, document, field, or endpoint; parameterization errors use the wrong date, window, adjustment, frequency, or financial scope; normalization errors mishandle units, identifiers, missing values, joins, or

time alignment; implementation errors instantiate an incorrect formula, transformation, or code path; verification errors omit assertions, reconciliations, independent checks, or contradiction handling; and commitment errors submit unsupported claims, incomplete evidence, or an invalid artifact despite unresolved checks.

This decomposition distinguishes infrastructure recovery from semantic repair. Transient API and transport failures may justify bounded retry; a successful call with the wrong vintage, unit, entity, or scope requires inspection and repair rather than repetition. This taxonomy provides a vocabulary for inspecting concrete trajectories; the aggregate dimension profiles summarize rubric assessments rather than counts of root causes.

# G Evaluator Self-Validation

Construction Oracles run in isolation at least twice with network disabled and frozen-input hashes checked before and after execution. Mutation families cover schema structure, values, signs, units, set membership, ordering, summaries, financial scope, temporal leakage, source evidence, hidden-file access, and missing process checkpoints. A mutation is counted as killed only when the intended check or integrity rule triggers. Positive controls include permitted numeric perturbations, retries, alternative valid tool orders, JSON key reordering, and path-agnostic evidence. Together, these tests cover the benchmark's specified schema, numerical, temporal, financial-scope, provenance, and process-integrity failure families.

For four provenance-intensive PDF tasks, source fields record source ID, page, bounding box, raw text, unit, and transformation. Derived fields record inputs, formula, raw result, and reported value. Removing a selected source, replacing a page, or corrupting a table anchor fails closed. The resulting audit contains 161 source-field and 169 derivation records, complementing the release-gate summary in Figure 2.

# H Trajectory Case Studies

We first quantify artifact-level evidence support and independently recheck a fixed task's arithmetic and source agreement. We then trace requirements, tool observations, repair decisions, and submitted artifacts in selected runs. The first two cases come from the 16-setting main matrix; two additional same-task contrasts use GenericAgent. Case selection is exploratory, based on observable failures and contrasting repair behavior. The observations come from executed code, tool output, and retained artifacts, rather than judge summaries alone. These cases establish concrete execution sequences, not their prevalence or the causal benefit of adding a verifier.

## H.1 Quantitative artifact checks

We pair completeness and evidence-consistency judgments within each submitted artifact, rather than comparing only their marginal means. Of the 1,600 runs in the reported matrix, 1,595 have both applicable scores and 1,553 of these pass the host schema check. Among them, 868 receive completeness exactly 1.0. Table A2 shows the evidence-score distribution for this fixed subset. In particular, 133/868 (15.3%) have evidence consistency at most 0.5, including 41 with zero evidence consistency. The midpoint 0.5 is a descriptive threshold, not an evaluator pass/fail rule. These are artifact counts based on existing rubric assessments, not independently adjudicated factual-error rates or equal-setting macro averages.

The 133 low-evidence records cover 56 tasks and 12 of the 16 settings; four settings have no such record under this definition. By answer judge, the corresponding counts are 113/629 for GLM-5.2 and 20/239 for Claude Opus 4.8 among schema-valid, fully complete answers. These descriptive strata preserve evaluator provenance; judge assignments are not randomized and their rates are not a judge comparison.

Independent arithmetic versus source checks. For FTB-049, we recompute ownership as submitted beneficially owned shares divided by submitted shares outstanding, times 100. All 16 artifacts agree with that calculation within $10^{-6}$ percentage points and pass schema validation. Only 13 use the filing's denominator of 800,641,166. The remaining three use 796,456,803, 798,000,000, and 797,000,000. Thus, correct arithmetic on the submitted inputs would not expose these input errors; a source-grounded denominator check would. The task is selected for the repair analysis below, so 3/16 is a within-task count, not a benchmark-wide error estimate.

| Condition or check | Count | Share |
|---|---|---|
| A. Schema-valid answers with completeness 1.0 ($n = 868$) | | |
| Evidence consistency $= 1$ | 461 | 53.1% |
| Evidence consistency $\in (0.5, 1)$ | 274 | 31.6% |
| Evidence consistency $\in (0, 0.5]$ | 92 | 10.6% |
| Evidence consistency $= 0$ | 41 | 4.7% |
| B. FTB-049 submitted artifacts across all settings ($n = 16$) | | |
| Pass host schema validation | 16 | 100.0% |
| Ownership arithmetic agrees with submitted inputs | 16 | 100.0% |
| Shares-outstanding denominator agrees with filing | 13 | 81.3% |

Table A2: Coverage and internal consistency do not certify source agreement. Panel A pairs two rubric scores on the same artifacts. Panel B independently rechecks submitted values; it does not count how often workers executed validation.

The scripts mine_delivery_evidence.py and audit_ownership_identity.py in the paper's analysis directory retain run-level records, denominators, evaluator identities, and source hashes. The latter verifies identical task and schema hashes across all 16 FTB-049 runs. No new model calls or judge calls are used.

Trajectory coding boundary. For the case studies below, an executed check requires a tool call and its returned observation, not merely an intention or a success message. A repair requires an observable change to the affected input or artifact; a recheck requires a subsequent executed check. Check-tool failures, detected content errors, and incomplete check coverage are recorded separately. The case traces support these distinctions, but we do not infer a benchmark-wide repair rate from keyword matches or unobserved steps.

### H.2 Recognizing an error is not an evidence-grounded repair

FTB-049 requires extracting a shareholding numerator and a dated shares-outstanding denominator from the specified original filings, then recomputing ownership. We compare Claude Opus 4.8 under Claude Code and OpenClaw. Task and schema hashes match; both runs use GLM-5.2 for their Process and Outcome judges. Both final answers pass schema validation and receive completeness 1.0.

| Stage | Opus / Claude Code | Opus / OpenClaw |
|---|---|---|
| Initial extraction | Tool reports 512,460 shares outstanding. | Tool reports 2,000,000 shares outstanding. |
| Error recognition | Compares extraction with a source passage stating 800,641,166. | Explicitly identifies incorrect shares and other extraction errors. |
| Repair choice | Replaces the denominator using the observed filing text. | After further extraction difficulty, hardcodes 796,456,803 as a claimed known fact. |
| Recalculation | Computes 9.178511% with the corrected denominator. | Computes 9.226732% with the unsupported denominator. |
| Submission | Corrected denominator retained; schema valid. | Unsupported denominator retained; schema valid; no missing data reported. |

Table A3: The repair decision separates two same-model trajectories. Both runs recognize extraction errors and subsequently perform arithmetic. The difference is whether the replacement input is grounded in the retrieved source.

Successful correction. In the Claude Code run, tool message 27 reports 512,460 shares. Tool message 38 returns the filing passage specifying 800,641,166 shares as of February 10, 2022. Assistant message 39 explicitly recognizes the discrepancy, and message 42 initiates correction. Tool message 46 recomputes $73{,}486{,}938/800{,}641{,}166 \times 100 = 9.178511\%$, which is retained in the final artifact.

Unsupported replacement. In the OpenClaw run, assistant message 46 recognizes several extraction errors. After unsuccessful attempts to isolate the relevant HTML passage, message 56 switches to hardcoding a claimed public-record value, 796,456,803. Tool message 59 uses that denominator in the final calculation. The retained Schedule 13G and 10-K files both contain 800,641,166: the authoritative source was successfully downloaded, but

the replacement value was not grounded in it. Availability in the downloaded file does not establish that the correct passage was visible in the worker's tool context.
What the contrast establishes. The failing run does not lack error awareness, and its arithmetic is consistent with its chosen denominator. It substitutes an unsupported denominator, computes the ownership ratio, and retains both in the submitted artifact. The successful run instead uses observed source evidence to correct that input before recalculation. This is an extraction-and-repair failure, not evidence that all upstream facts were correct and only final serialization failed. Outcome scores are 1.000 and 0.173, respectively, but their magnitude also reflects grouped recursive checks: an incorrect denominator fails the filing-facts and calculation blocks. The behavioral sequence, rather than the large score gap, is the mechanism evidence.

### H.3 A disclosed fallback can still violate the task

FTB-009 asks for historical 2024 ETF returns and maximum-drawdown rankings. In the Claude Sonnet 4.6 / OpenHands run, API retrieval fails. Tool message 19 then shows executed code constructing synthetic price paths: it seeds a random generator with the ETF code and samples daily returns. Tool message 23 and the submitted artifact contain four populated ETF rows and a ranking calculated from those simulated paths.
The answer explicitly labels the fallback as synthetic and documents the connection failure; it does not conceal the substitution. Nevertheless, simulated paths do not satisfy the request for historical observations. The artifact passes the host schema check and receives completeness 1.0, while evidence consistency is 0.0 and Outcome is 0.109. The visible failure is not the network outage itself, but the subsequent decision to replace unavailable observations with synthetic inputs and deliver the requested numerical ranking. Recording a limitation does not make the affected numerical claims evidence-grounded.

Evidence locations. Run IDs are 95898fec61ab (FTB-049, Claude Code), 003358bec57c (FTB-049, OpenClaw), and 2e33aaed1fba (FTB-009, OpenHands), with the directory prefix run_. Message numbers in these two cases are zero-based indices in the normalized trajectory/messages.json list, not worker turn counts. For FTB-049/OpenClaw, the retained source files are schedule_13g_musk_twitter.htm and twitter_10k_2021.htm in the workspace. Final artifacts and host checks are retained alongside each trajectory.

Additional GenericAgent contrasts. The following two tasks supplement, but do not enter, the main matrix. Within each task, the compared runs have identical task-text and answer-schema hashes. Their existing Outcome scores use GLM-5.2 for the answer judge.

| Task | GA model | Outcome | Observed artifact |
|---|---|---|---|
| FTB-085 | Claude Opus 4.8 | 1.000 | 250 common dates; schema valid |
| | Claude Haiku 4.5 | 0.042 | 249 common dates; schema invalid |
| FTB-097 | Claude Opus 4.8 | 0.192 | Core ratios agree; schema invalid |
| | Claude Haiku 4.5 | 1.000 | Core ratios agree; schema valid |

Table A4: Same-task execution and delivery contrasts. Outcome is the existing composite answer score, not a schema pass rate. In FTB-097, the three annual values of both coverage ratios agree between the two submitted artifacts despite their different schema status.

### H.4 Counts and joins do not verify date semantics

Contract. FTB-085 constructs a 2024 gasoline crack-price proxy from EIA WTI and gasoline prices. The agent must join on common quotation dates, select the last common date in each month, and compute gasoline price per gallon times 42 minus WTI price per barrel.
Execution. Haiku's turn 10 converts Excel serial dates with datetime(1900, 1, 1) + timedelta(days=serial - 1). For the dates in this task, that expression shifts the date forward by one day. Both series receive the same erroneous conversion, so their inner join does not expose the error. The next turn accepts 249 common observations and proceeds to monthly aggregation. The final artifact reports June 29, a Saturday, as June's last common quotation date; the Opus artifact reports June 28. January's proxy also differs: 17.670 versus 16.960 dollars per barrel.
Verification boundary. Haiku checks record counts and the presence of all 12 months, but proceeds with 249 common observations despite the schema's explicit requirement of 250.

Its checks neither enforce this count constraint nor establish that parsed dates preserve the source dates. Opus parses the EIA HTML date labels, filters to 2024, retains 250 common observations, and runs schema validation before submission. Haiku's answer also violates the output schema, so the Outcome difference is not attributable solely to date conversion. The trace shows both a semantic transformation error and an observed count mismatch reaching the delivered artifact.

### H.5 A validation message is not a complete validation

Contract and computation. FTB-097 computes Home Depot's fixed-charge and lease-adjusted coverage for fiscal years 2022–2024. Opus retrieves the filing and computes fixed-charge coverage of 7.886364, 6.212777, and 5.222735; its lease-adjusted values also agree with Haiku's submitted values.

Verification and submission. In turn 16, Opus loads the answer and schema, checks top-level required fields and selected constants, and prints annual values. It then unconditionally prints Answer structure validated. This custom routine does not validate nested required fields or disallowed properties. The delivered artifact uses operating_income instead of the required operating_income_musd, among other missing and mismatched fields. The host evaluator reports an invalid schema. In contrast, Haiku's turn 13 calls jsonschema.validate(instance=answer, schema=schema); the tool reports success and the host schema check agrees.

Interpretation. The contrast separates successful core calculation from contract-compliant delivery. Opus does attempt verification, but its check covers less than the contract it claims to validate. Haiku's higher score in this case is therefore not evidence of uniformly stronger financial reasoning: the visible distinction is a concrete difference in validation coverage and artifact construction.

Trace locators. The retained run IDs are 9ec438c01181 (FTB-085, Opus), 32a9277c18fb (FTB-085, Haiku), cf28327a1a6b (FTB-097, Opus), and 5b7ab07e196d (FTB-097, Haiku), each prefixed by run_ in the release. Turn numbers refer to trajectory/message.json; submitted artifacts and host checks are retained in workspace/answer.json and eval/answer_result.json.

## I Reproducibility Details

The release records the frozen model roster, model endpoint identity, agent and harness version, canonical-run selection, evaluator provenance, result snapshots, task packages, and container configuration needed to reproduce the reported matrix. Each run retains its requested and observed model identity, exit status, normalized trajectory, artifacts, answer, Process result, Outcome result, Integrity record, and evaluator model metadata. Selection counts each model–harness–task unit once, using its latest-started retained run. Missing answers and execution failures within retained settings remain in task-macro scores. The analysis script and hashed run manifest in analysis/reported_subset.json document the reported subset; the original 19-setting snapshot is preserved for audit.

Agents execute in fresh Docker containers with read-only task packages, writable workspaces, public-network access under the task policy, and bounded resources. Host-only references, frozen evaluator inputs, graders, and prior runs remain outside the tested environment.