TIMEEVO: FAILURE-DRIVEN SELF-EVOLUTION OF A TIME SERIES AGENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series agents answer analytical questions by calling external tools, and which tools they carry is decided by people before the agent runs. However, we identify two failures in this setup. **Human–Agent Tool Misalignment**: a library of 21 expert-curated tools helps on some tasks and hurts on others, dropping anomaly accuracy under every backbone we test. **Silent Harm**: one round of generic self-revision changes 147 answers and breaks 56 of them, while the final score moves by less than a point. Both follow from the same gap: whether a tool helps is decided question by question at runtime, while tools are supplied in advance and judged by a single average. To address this, we propose TimeEvo, which clusters an agent's diagnosed failures into capability gaps, plans a measurement for each, synthesizes evidence-only tools that fill them, and admits the candidate library only through a paired admission gate. Experiments on ten time series QA tasks and three backbones show that TimeEvo, starting from an empty library, improves accuracy on every task and every backbone, and that a library grown on a cheap model still gains when it is installed into stronger ones. Code is available at `https://anonymous.4open.science/r/TimeEvo-652E`.

1 INTRODUCTION

Time series analysis (Hu et al.; Beqari et al., 2024) is a fundamental problem with broad impact in real-world applications such as finance (Yang et al., 2025b; Cheng et al., 2025), traffic (Hu et al., 2026), and healthcare (Yang et al., 2025a), and spans a wide range of tasks, including forecasting (Yang et al., 2026a; Wang et al., 2025), imputation (Yang et al., 2026b), and time series understanding (Xie et al., 2024). To address this diverse task landscape, model-based approaches follow two broad designs. Dedicated models (Liu et al., 2024; Zeng et al., 2023) provide efficient and competitive performance on well-defined objectives, whereas multimodal language models (Jin et al., 2024; Xie et al., 2024) treat time series as an additional modality for natural-language interaction and open-ended questions (Kong et al., 2025). Despite their different interfaces, both acquire analytical capabilities through task-oriented training and encode them in model parameters. Therefore, supporting a new task or capability typically requires corresponding data construction, training, or tuning (Dong et al., 2026; Kong et al., 2026).

Large language model (LLM) agents (Yao et al., 2022; Zhu et al., 2026) offer an alternative: rather than internalizing every time series operation in model parameters (Dong et al., 2025), they interpret user requests, plan analyses, and invoke external analytical tools (Wu et al., 2026). Across tasks ranging from forecasting (Weng et al., 2026) to open-ended time series understanding (Kong et al., 2025; Yu et al., 2026), they synthesize tool outputs and intermediate evidence into coherent natural-language responses (Ye et al., 2024; Zhao et al., 2025). This design implements many domain-specific analytical operations as reusable external modules that can be inspected, recombined, and modified without retraining the underlying model (Wu et al., 2026). Building on this modular design, recent agent frameworks further refine how agents use these external modules by distilling execution experience into reusable guidance (Liu et al., 2026; Ye et al., 2024).

However, task-specific development persists: human effort does not disappear but shifts from model training (Fan et al., 2026a;b; Fang et al., 2026) to configuring prompts, workflows, and tools for tasks anticipated before deployment (Hao et al., 2026). It remains unclear whether supplying an agent with more tools makes it better at the tasks those tools were written for, and whether the

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.

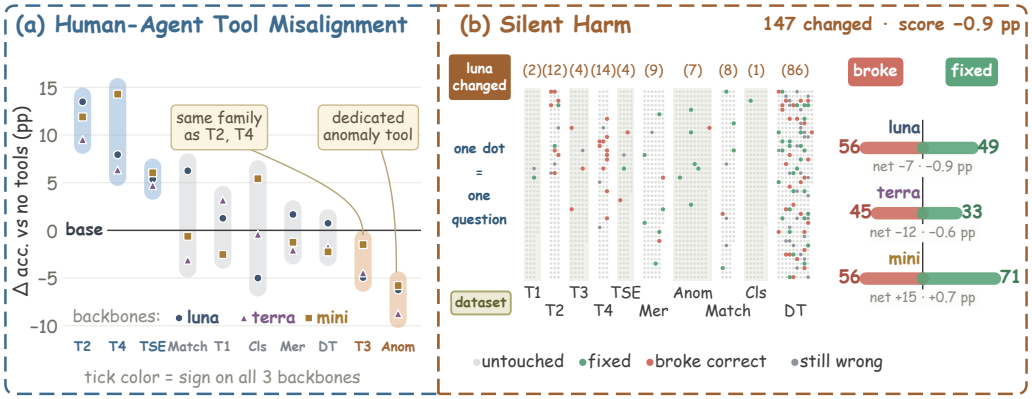


Figure 1: **Two failure phenomena in TSAgent.** We compare the same frozen agent with and without 21 expert-curated tools **(a)**, and before and after one generic self-refinement pass **(b)**, pairing every question across ten QA tasks and three backbone LLMs. Improved and degraded tasks are shaded **blue** and **orange** in (a), and fixed and broken answers are shown in **green** and **red** in (b).

agent's own revisions can be trusted to improve it (Miao et al., 2025). To investigate, we analyze the errors and repair attempts of a frozen-LLM time series agent (TSAgent) across time series QA tasks as shown in Fig. 1. Our analysis reveals two key phenomena:

❶ **Human–Agent Tool Misalignment**: ***what humans prefer to supply is not what the agent benefits from.*** Equipping the same frozen agent with a library of 21 expert-curated analysis tools of TimeART (Wu et al., 2026) makes it better on some tasks and worse on others (Fig. 1a). The largest gains and the largest losses both recur in the same direction under all three backbone LLMs. The losses are not a coverage problem: the library includes a dedicated anomaly-detection tool, yet the anomaly task drops by 5.8 to 8.8 points under every backbone. T3 shares the same task family with the improving T2 and T4 (Weng et al., 2026), yet it is degraded under all three backbones. The misalignment is structural: the supply is fixed per task before deployment, but whether a tool helps is decided question by question at runtime.

❷ **Silent Harm**: ***self-revision breaks answers that the final score never shows.*** A single pass of the Self-Refine (Madaan et al., 2023) changes 147 answers across the ten tasks: 49 are fixed, 56 previously correct answers break, and the remaining 42 swap one wrong answer for another. Yet the final score moves by less than one point (Fig. 1b). The pattern is not specific to one model: on two further backbones the pass touches 97 and 163 answers, breaks 45 and 56 of them, and again never moves the final score by a full point. On T4 alone, the pass on luna changes 14 answers, fixes 0, and breaks 11. The silence is arithmetic: inside a single average, the harm silently erases the fixes, and the final score never moves.

Together, these findings show that the tools agents are given do not match what they need at runtime, while unexamined self-revision silently erases much of what it fixes. This leads to a central question: ***Can a time series agent supply its own missing tools from diagnosed failures while measurably protecting what it already answers correctly from silent harm?***

To answer this question, we propose **TimeEvo**, a self-evolving agent for time series analysis. The key idea is to let the agent discover the tools it actually needs from the failures it makes, and to trust no self-made update until it proves that it helps more than it harms. Specifically, TimeEvo clusters the agent's errors into failure buckets, plans a measurement contract for each, and synthesizes evidence-only tools to fulfill them. The whole candidate library must then pass two levels of validation, and a failing library is pruned and retested once. Ultimately, the agent evolves its own task-specific tool library, in which every tool has earned its place without harming silently.

Our main contributions are summarized as follows:

- We identify two phenomena in time series agents, **Human–Agent Tool Misalignment** and **Silent Harm**: what humans prefer to supply is not what the agent benefits from, and self-revision silently erases much of what it fixes.
- We propose **TimeEvo**, a self-evolving agent that discovers the tools it needs from the failures it makes, and admits every self-made update only after two levels of validation.
- Extensive experiments on ten time series QA benchmarks show that TimeEvo improves its base agent on every task, with pooled gains from $+1.5$ to $+14.7$ points. Neither expert-curated tools nor generic self-refinement reproduces this.

2 Related Work

2.1 Time Series Analysis

Time series analysis has long been dominated by specialized models built for individual tasks such as forecasting (Yang et al., 2026a), classification (Dempster et al., 2020), and anomaly detection (Liu & Paparrizos, 2024). Pretrained foundation models (e.g., Chronos (Ansari et al., 2025), TimesFM (Das et al., 2023), and Moirai (Liu et al., 2025a)) extend this line by transferring forecasting capability across domains. Furthermore, the emergence of LLMs has enabled more flexible and powerful approaches to time series analysis. Multimodal language models such as ChatTS (Xie et al., 2024) and TimeLLM (Jin et al., 2024) take time series as an additional input modality and answer open-ended questions end to end. Yet facing a new task, these models still depend on collecting task-specific data and training or tuning the model. In contrast, agent-based systems (e.g., TS-Reasoner (Ye et al., 2024), TS-Agent (Liu et al., 2025b), TimeCopilot (Garza & Rosillo, 2025)) offer a training-free alternative that moves analytical capabilities outside the model: the agent plans over external tools and synthesizes their outputs into natural-language answers. Training-free, however, does not mean effort-free: human effort merely shifts from model training to designing prompts, workflows, and tools before deployment. To reduce this reliance on human design, we propose TimeEvo, which treats diagnosed failures as the supervision signal for evolving the capability library itself.

2.2 Self-Evolving Agents

Self-evolution has been extensively explored for general LLM agents. One line, exemplified by Reflexion (Shinn et al., 2023) and EvolveR (Wu et al., 2025), distills experience from past trajectories into reusable lessons or principles. Another line maintains growing skill libraries such as Voyager (Wang et al., 2023), and recent work like SkillOpt (Yang et al., 2026c) and CoEvoSkills (Zhang et al., 2026) further tests learned skills before deployment with held-out scores or per-task checks. Within time series, TimeClaw distills usage experience over a fixed tool library (Liu et al., 2026). However, these frameworks admit an update through the final score or a per-task check at best, and some skip validation entirely. The admission never hinges on how many previously correct answers the update breaks. Inside a single final score, such breaks cancel against the fixes, so an update can cause real harm while the score barely moves, exactly the Silent Harm identified in our analysis. This motivates TimeEvo, which evolves the tool library from diagnosed failures and admits every update only after two levels of validation that weigh what it fixes against what it breaks.

3 Method

3.1 Problem Definition

Each time series question is a triple $x = (q, \mathbf{S}, \mathcal{O})$: a natural-language question q that may carry event context, an observed input $\mathbf{S}$ collecting one or two primary series together with any named covariate or candidate series the question refers to, and a finite option set $\mathcal{O}$. An agent is a frozen LLM equipped with a library $\mathcal{L}$ of executable analysis tools, and it answers by combining tool evidence with its own reasoning to select one option from $\mathcal{O}$. We write $\mathcal{A}_{\mathcal{L}}$ for the agent carrying $\mathcal{L}$, and the base agent starts from an empty library $\mathcal{L}_0$ (a pre-installed forecasting root is kept only as an ablation). The gold answer is determined by the observed input alone, so whether the agent answers a question correctly can be checked automatically. A final score alone cannot judge a library update

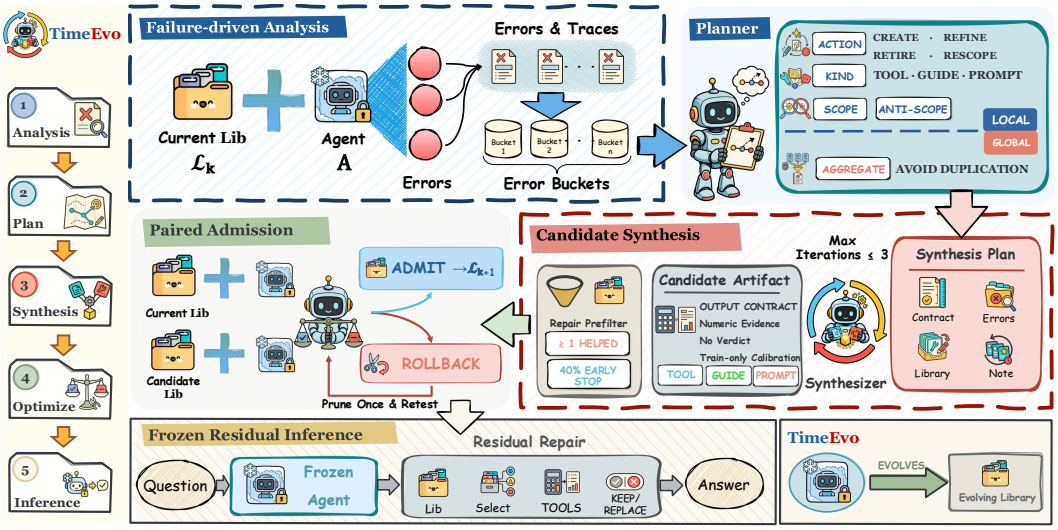


Figure 2: **Overview of TimeEvo.** Each round turns the agent's own failures into new tools: errors are clustered into failure buckets, each bucket is planned into a measurement contract, and contracts are synthesized into evidence-only candidate tools. The candidate library is admitted only after two levels of validation, a per-tool prefilter and a whole-library paired gate. A failing library is pruned and retested once. Admitted tools join the agent's library, while rejected rounds leave the agent untouched. At inference, the frozen agent answers first, and admitted tools review only in-scope questions, keeping or replacing the answer.

$\mathcal{L} \to \mathcal{L}'$, which is the Silent Harm of Fig. 1b. We therefore track what an update fixes and what it breaks on a question set D:

$$\begin{aligned} \text{helped}(D) &= \{x \in D : v(x) = 0,\ v'(x) = 1\}, \\ \text{harmed}(D) &= \{x \in D : v(x) = 1,\ v'(x) = 0\}, \end{aligned} \tag{1}$$

where v and v' denote correctness under $\mathcal{L}$ and $\mathcal{L}'$ on the same questions. Given train, validation, and test splits $D_{\text{train}}, D_{\text{val}}, D_{\text{test}}$, the goal is to evolve $\mathcal{L}$ from the agent's failures on D_{train} so that helped outnumbers harmed on unseen questions, while D_{test} is never used for any decision.

3.2 Overview and Frozen Residual Inference

TimeEvo runs in rounds, and each round turns the agent's current failures into new tools (Fig. 2). A round has four stages: diagnose the failures on the training split into measurement contracts, synthesize evidence-only tools for the contracts, screen each tool with a cheap prefilter, and submit the surviving library as a whole to a paired admission gate. After r accepted rounds the deployed library is a chain $\mathcal{L}_r = (\mathcal{L}_0, \mathcal{T}_1, \ldots, \mathcal{T}_r)$, where $\mathcal{T}_k$ is the tool set admitted in round k, and a question is threaded through the stages:

$$\mathcal{A}_{\mathcal{L}_k}(x) = \begin{cases} \rho_k\big(x,\ E_k(x),\ \mathcal{A}_{\mathcal{L}_{k-1}}(x)\big) & \text{if } x \in \text{scope}(\mathcal{T}_k), \\ \mathcal{A}_{\mathcal{L}_{k-1}}(x) & \text{otherwise}, \end{cases} \tag{2}$$

where $\text{scope}(\mathcal{T}_k)$ matches the stage's declared scope fields (task types, evidence types, single- or dual-series input, and an anti-scope), $E_k(x)$ is the numeric evidence the stage's tools compute on x, and ρ_k is the same agent reviewing the incoming answer with that evidence. At answer time, as the *Frozen Residual Inference* strip of Fig. 2 shows, the agent consults a structured scope catalog and inspects at most three shortlisted tools per question. The review is conservative by construction: the incoming answer is the default, and a replacement is honored only when a stage tool actually ran on the question. If a tool errors, is never invoked, or the review does not explicitly decide to replace, the question falls back to the incoming answer. Out-of-scope questions are therefore never degraded. In-scope harm remains possible, and the admission gate measures it alongside repair and screens candidate libraries for positive net improvement on validation data.

3.3 Failure-Driven Analysis and Planning

As shown in the *Failure-driven Analysis* panel of Fig. 2, round r first replays the current library on the training split, collects the questions it answers wrong, and partitions them into failure buckets:

$$\mathcal{F}_r = \{x \in D_{\text{train}} : v_{r-1}(x) = 0\}, \qquad \mathcal{F}_r = \mathcal{B}_1 \cup \cdots \cup \mathcal{B}_J, \tag{3}$$

where v_{r-1} denotes correctness under the current library $\mathcal{L}_{r-1}$ and the buckets are disjoint. The agent builds this partition in two steps: it first defines a small set of failure categories from a sample of the errors, then assigns every error to exactly one category. Errors on which a tool fired and errors that no tool reached are clustered separately, because the former indicate a defective or misused tool while the latter indicate a coverage gap. The definition step never sees task metadata, so buckets group errors by the missing capability rather than by question type. Oversized buckets are split once, and buckets with fewer than two errors are set aside.

Each bucket $\mathcal{B}_j$ is then diagnosed by a local planner (the *Planner* panel in Fig. 2), which must return a *measurement contract* c_j: the diagnosed root cause, the required measurement, the inputs it needs, a success criterion, and a scope with an anti-scope of applicability. The contract also declares one library action, typed by the diagnosed failure mode: **CREATE** a new tool when evidence is missing, **REFINE** an existing tool with usage guidance when its evidence is ignored or misread, **RESCOPE** a tool that fires on the wrong questions, and **RETIRE** a tool that should leave the library. Because buckets are planned in parallel, a global planner coordinates the proposals afterwards: duplicated measurements are merged and overlapping scopes are narrowed, while a merge of incompatible contracts is rejected mechanically and falls back to the independent per-bucket plans. Planning is thus committed to measurements rather than answers: what a bucket receives is a contract for the evidence its failures lack, not a rule for how to answer them.

3.4 Candidate Synthesis and Prefilter

As shown in the *Candidate Synthesis* panel of Fig. 2, each coordinated contract c_j receives up to three sequential synthesis attempts:

$$t_j^{(i)} = \text{Synth}\big(c_j,\ f_j^{(i-1)}\big), \qquad t_j^{(i)} : (\mathbf{S}, q) \mapsto \mathbf{e} \in \mathbb{R}^{d_j}, \qquad i \le 3, \tag{4}$$

where $f_j^{(i-1)}$ is the structured feedback of the previous failed attempt ($f_j^{(0)} = \emptyset$), and the delivered tool t_j is the first passing attempt, one deterministic Python function. A tool is evidence-only: it never outputs a verdict, an option string, or a dataset-specific constant. It may read a covariate only when the question mentions it, and when its input is unavailable it abstains rather than fabricates a value. A sandbox that permits only a fixed allowlist of operations enforces determinism and isolation, and a candidate that fails any of these checks is discarded. A tool is not limited to plain arithmetic on the series: it may call the raw Chronos-2 model (Ansari et al., 2025) as a forecasting primitive, and when its contract requests a decision boundary it may carry a *train-only calibration*, a shallow decision tree fitted on training rows alone and compiled into its source:

$$g_j = \text{Fit}\big(\{(\, t_j(x),\ y(x)\,) : x \in D_{\text{train}}\}\big), \qquad t_j \leftarrow t_j \oplus g_j \ \text{ iff } \ \text{BalAcc}(g_j) \ge 0.55, \tag{5}$$

where $y(x)$ is the gold answer, $\oplus$ denotes compiling the fitted tree into the tool's source code, and BalAcc is balanced accuracy on an internal train-side split with at least 16 samples per class. Thresholds are therefore measured on training rows rather than invented by the agent.

The first level of validation is a per-tool prefilter, run on at most 32 held-out errors $\mathcal{H}_j \subset \mathcal{B}_j$ of the tool's own bucket. A candidate survives only if it delivers at least one *attributable* repair: a question fixed with the tool actually attached, actually executed, and visible in the execution record. It must also break few previously correct answers and rarely fail to execute. Two consecutive attempts without an attributable repair send the bucket back for re-diagnosis, which may change the artifact kind, for example from a new tool to usage guidance. The prefilter only discards clearly useless candidates: every statistical decision is reserved for the admission gate, the only decision whose mistakes would reach deployment.

3.5 Paired Admission

The second level of validation (the *Paired Admission* panel in Fig. 2) decides admission for the round's surviving tools $\mathcal{T}_r$ as a whole rather than tool by tool, because tools that look harmless in

isolation can interfere once deployed together. The current library and the candidate library answer the full validation split in one paired run, and only questions that both sides complete are compared. Applying Eq. (1) to D_{val} gives $h = |\text{helped}(D_{\text{val}})|$ and $m = |\text{harmed}(D_{\text{val}})|$. Over the n paired questions this yields the gate statistic:

$$\delta = \frac{h-m}{n}, \qquad \delta_{\text{lb}} = \delta - Z \cdot \frac{\sqrt{\max(h+m,\,1)}}{n}, \qquad Z = 1.96, \tag{6}$$

and the library passes strictly when $h > 0$ and $\delta_{\text{lb}} > 0$. A clean 3-versus-0 round on a small split still fails, because its lower bound stays negative. A single-round run has no later rounds in which to accumulate evidence, so a supplementary rule admits a library that satisfies:

$$\delta > 0, \qquad h - m \;\geq\; \max\big(3,\; \lceil 0.02\,n \rceil\big), \qquad \frac{m}{n} \;\leq\; 0.10, \tag{7}$$

that is, a positive paired delta, a net repair of at least three questions or two percent of the split, and an absolute harm rate within ten percent.

When the gate rejects the library, TimeEvo attempts one repair before giving up. The validation records show which tool fixed or broke which question, so every update u receives its own counts h_u and m_u, and an update is removed only when it is clearly harmful:

$$\mathcal{P} = \big\{\, u \in \mathcal{T}_r \;:\; m_u > 0 \;\wedge\; \big(h_u = 0 \;\vee\; m_u - h_u \geq 3\big) \,\big\}. \tag{8}$$

A tool is left alone whenever the evidence against it is thin: one that changed no answers may simply have met no matching questions, and one whose counts are small and mixed cannot be told apart from noise. The pruned library is retested in exactly one more full paired run and commits only if it strictly improves the lower bound. There is no second prune, because repeated retesting would slowly fit the library to the validation split. Whether a library passes at once or only after the prune, it then reconciles its scope fields against how its tools behaved, and because that edit changes the library, it passes the same paired gate one final time before joining the chain. A rejected round changes nothing.

4 Experiments

4.1 Experimental Settings

Tasks. The formal suite covers ten time series QA tasks from six public sources: Temporal-Bench T1–T4 (Weng et al., 2026), TimeSeriesExam (Cai et al., 2024), Merrill (Merrill et al., 2024), TimeMQA Anomaly and Classification (Kong et al., 2025), MMTS Match (Yin et al., 2026), and TSAQA Data-Transformation (Jing et al., 2026). Split details are in Appendix A.

Protocol. Every run uses the same frozen protocol: empty root library, one evolution round, train-only calibration, and a report-only test split. We evaluate on three backbones, GPT-5.6-luna, GPT-5.6-terra, and GPT-5.4-mini, with all four LM roles switched together. Every number we report is a within-run paired delta (Eq. (1)).

Baselines. We compare against five non-evolving baselines: a fixed Chronos-2 forecasting tool (Ansari et al., 2025), the 21 expert-curated tools of TimeART (Wu et al., 2026), Self-Refine (Madaan et al., 2023), five-sample self-consistency voting, and few-shot in-context learning with four training examples. Each tests one alternative explanation of the gains: that a strong pre-installed tool, a hand-engineered library, one more round of reflection, the same budget spent on sampling, or direct contact with the training data would suffice. All are measured with the same paired protocol on the same splits.

4.2 Overall Comparison

We compare TimeEvo against the five baselines on all ten tasks, under each of the three backbones. Results are in Table 1, and Fig. 3a takes a closer look at TimeMQA Anomaly, where the human library fails. Based on these results, we summarize our observations (**Obs.**) as follows:

Obs. ❶: TimeEvo is the only method that improves every task on every backbone. All thirty of its task results are positive. Its mean gain is the largest in every block: $+8.79$ on luna, $+8.73$

Table 1: Main comparison on the ten tasks across three backbones. Each cell shows test accuracy with the within-run paired delta (pp) in parentheses, where **bold** marks the best method per column within a block and underline the second best. The last block installs the library that GPT-5.6-luna evolved into three stronger models, and its shaded rows carry that library and are not ranked.

Method	T1	T2	T3	T4	TSExam	Match	Merrill	Anomaly	Cls	TSAQA-DT	Mean
GPT-5.6-luna											
Base (no tool)	41.5	33.9	31.9	36.5	72.2	78.4	85.6	60.2	49.0	71.0	56.0
Chronos-2 fixed	43.4 (+1.90)	**49.0** (+15.08)	34.8 (+2.97)	**54.8** (+18.25)	76.9 (+4.70)	77.8 (−0.62)	86.4 (+0.83)	57.2 (−3.00)	56.9 (+7.92)	69.3 (−1.75)	60.7 (+4.63)
TimeART 21 tools	42.8 (+1.27)	47.4 (+13.49)	26.9 (−4.95)	44.4 (+7.94)	77.6 (+5.37)	84.7 (+6.25)	**87.2** (+1.67)	53.9 (−6.25)	44.0 (−5.00)	71.8 (+0.75)	58.1 (+2.05)
Self-Refine	42.2 (+0.63)	30.0 (−3.97)	30.4 (−1.49)	27.8 (−8.73)	71.6 (−0.67)	80.3 (+1.88)	**87.2** (+1.67)	61.4 (+1.25)	49.4 (+0.42)	70.8 (−0.25)	55.1 (−0.93)
vote@5	39.0 (−2.53)	34.7 (+0.79)	32.9 (+0.99)	40.5 (+3.97)	72.9 (+0.67)	80.3 (+1.88)	86.8 (+1.25)	59.7 (−0.50)	49.4 (+0.42)	72.0 (+1.00)	56.8 (+0.79)
few-shot ICL k=4	**48.5** (+6.96)	33.9 (0.00)	**42.8** (+10.89)	38.1 (+1.59)	76.9 (+4.70)	81.6 (+3.12)	86.4 (+0.83)	56.7 (−3.50)	39.8 (−9.17)	73.0 (+2.00)	57.8 (+1.74)
TimeEvo (ours)	44.7 (+3.16)	45.0 (+11.11)	37.2 (+5.28)	47.1 (+10.58)	**77.8** (+5.59)	**90.9** (+12.50)	87.1 (+1.53)	**74.9** (+14.69)	**61.1** (+12.08)	**82.4** (+11.42)	**64.8** (+8.79)
GPT-5.6-terra											
Base (no tool)	37.9	34.0	32.7	37.2	73.6	78.3	87.4	59.8	54.0	71.5	56.6
Chronos-2 fixed	37.9 (0.00)	**50.7** (+16.67)	32.2 (−0.50)	**48.3** (+11.11)	76.3 (+2.68)	80.2 (+1.88)	88.2 (+0.83)	61.8 (+2.00)	52.8 (−1.27)	73.8 (+2.25)	60.2 (+3.56)
TimeART 21 tools	41.0 (+3.16)	43.5 (+9.52)	28.2 (−4.46)	43.5 (+6.35)	**78.3** (+4.70)	75.2 (−3.12)	85.3 (−2.08)	51.0 (−8.75)	53.6 (−0.42)	69.8 (−1.75)	56.9 (+0.31)
Self-Refine	38.5 (+0.63)	31.6 (−2.38)	32.7 (0.00)	33.2 (−3.97)	74.3 (+0.67)	79.0 (+0.62)	87.8 (+0.42)	59.5 (−0.25)	54.5 (+0.42)	69.5 (−2.00)	56.1 (−0.58)
vote@5	36.0 (−1.90)	32.4 (−1.59)	32.7 (0.00)	40.3 (+3.17)	68.9 (−4.70)	79.6 (+1.25)	86.9 (−0.42)	64.8 (+5.00)	49.9 (−4.17)	72.8 (+1.25)	56.4 (−0.21)
few-shot ICL k=4	**46.1** (+8.23)	33.2 (−0.79)	29.7 (−2.97)	38.8 (+1.59)	77.0 (+3.36)	77.7 (−0.62)	89.4 (+2.08)	64.3 (+4.50)	50.7 (−3.33)	67.0 (−4.50)	57.4 (+0.76)
TimeEvo (ours)	41.7 (+3.80)	44.3 (+10.32)	**36.1** (+3.47)	45.1 (+7.94)	77.0 (+3.36)	**90.8** (+12.50)	**90.3** (+2.92)	**72.3** (+12.50)	**54.8** (+0.83)	**86.0** (+14.50)	**63.9** (+7.21)
GPT-5.4-mini											
Base (no tool)	40.9	34.3	32.5	37.6	70.8	72.1	86.1	61.0	43.1	67.3	54.6
Chronos-2 fixed	45.4 (+4.43)	**47.8** (+13.49)	28.5 (−3.96)	**59.8** (+22.22)	70.2 (−0.67)	69.0 (−3.12)	84.0 (−2.08)	56.2 (−4.75)	45.2 (+2.08)	66.8 (−0.50)	57.3 (+2.71)
TimeART 21 tools	38.4 (−2.53)	46.2 (+11.90)	31.0 (−1.49)	51.9 (+14.29)	**76.9** (+6.04)	71.5 (−0.62)	84.9 (−1.25)	55.2 (−5.79)	48.5 (+5.42)	65.1 (−2.25)	56.9 (+2.37)
Self-Refine	41.6 (+0.63)	32.7 (−1.59)	32.0 (−0.50)	37.6 (0.00)	66.8 (−4.03)	80.2 (+8.12)	86.5 (+0.42)	60.2 (−0.75)	46.5 (+3.38)	68.3 (+1.00)	55.2 (+0.67)
vote@5	41.6 (+0.63)	34.3 (0.00)	34.5 (+1.98)	39.2 (+1.59)	69.5 (−1.34)	75.8 (+3.75)	88.6 (+2.50)	63.0 (+2.00)	47.7 (+4.58)	68.1 (+0.75)	56.2 (+1.64)
few-shot ICL k=4	**49.2** (+8.23)	29.5 (−4.76)	30.5 (−1.98)	28.8 (−8.73)	72.9 (+2.01)	70.8 (−1.25)	**89.0** (+2.92)	63.0 (+2.00)	32.7 (−10.42)	67.8 (+0.50)	53.4 (−1.15)
TimeEvo (ours)	46.6 (+5.70)	45.4 (+11.11)	**37.0** (+4.46)	39.2 (+1.59)	72.9 (+2.01)	**89.0** (+16.88)	**89.0** (+2.92)	**77.2** (+16.25)	**53.5** (+10.42)	**83.3** (+16.00)	**63.3** (+8.73)
Strong Models + the Library Evolved by GPT-5.6-luna											
Claude-opus-5 (bare)	36.1	38.9	30.2	41.3	74.5	72.5	91.7	60.2	51.2	59.8	55.6
+ evolved library	40.5 (+4.43)	40.5 (+1.59)	34.2 (+3.96)	46.8 (+5.56)	81.2 (+6.71)	96.2 (+23.75)	94.2 (+2.50)	86.5 (+26.25)	65.0 (+13.75)	76.2 (+16.50)	66.1 (+10.50)
Claude-sonnet-5 (bare)	38.0	38.1	30.2	42.9	71.1	76.9	91.7	59.0	50.8	60.0	55.9
+ evolved library	41.8 (+3.80)	43.6 (+5.56)	33.2 (+2.97)	44.4 (+1.59)	83.2 (+12.08)	96.2 (+19.38)	94.2 (+2.50)	85.5 (+26.50)	66.2 (+15.42)	76.8 (+16.75)	66.5 (+10.65)
GPT-5.6-sol (bare)	38.0	30.9	30.7	35.7	75.8	80.6	90.0	59.5	57.9	71.8	57.1
+ evolved library	43.7 (+5.70)	39.7 (+8.73)	36.1 (+5.45)	43.6 (+7.94)	81.9 (+6.04)	95.0 (+14.37)	92.9 (+2.92)	81.2 (+21.75)	55.4 (−2.50)	84.0 (+12.25)	65.4 (+8.27)

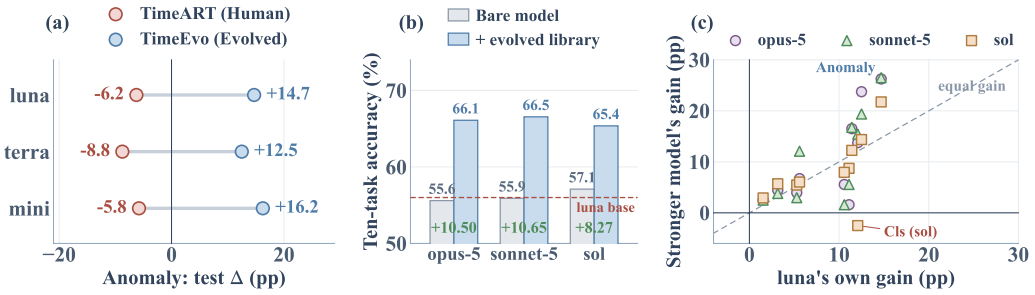


Figure 3: Human tools, and the transfer of an evolved library. **(a)**: Test delta on TimeMQA Anomaly for the 21 expert-curated tools of TimeART and for the evolved library. **(b)**: Ten-task mean accuracy of three stronger models, carrying the library GPT-5.6-luna evolved. **(c)**: Per-task gain of those models against the gain luna itself obtained with the same library. Diagonal marks equal gain.

on mini, and +7.21 on terra, the strongest backbone of the three. Each backbone evolves its own library from its own failures, so the gains come from the loop rather than from one lucky set of tools. These results answer the first half of the central question of Section 1: an agent can supply its own missing tools from its own diagnosed failures.

Obs. ❷: Human-supplied tools and generic self-improvement pay for their peaks with losses elsewhere. The fixed Chronos-2 tool is strongly positive only on the two forecasting tiers it was built for (T2/T4, up to +22.22) and flat or negative elsewhere. As shown in Fig. 3a, the TimeART library includes a dedicated anomaly tool yet loses $-6.25/-8.75/-5.79$ on Anomaly under every backbone, and one Self-Refine pass breaks nearly as many answers as it fixes (Fig. 1b). Voting

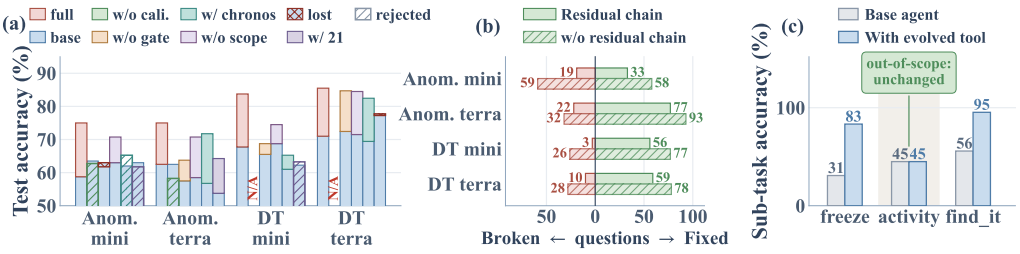


Figure 4: Ablations on TimeMQA Anomaly and TSAQA-DT under two backbones. **(a)**: Test accuracy with one component removed, each bar rising from that run's own no-tool base (blue). A red cross marks accuracy the library lost, and a hatched bar marks a gate-rejected round. **(b)**: Answers fixed and broken by the update, under the protected review and under unconditional overwrite of the same library. **(c)**: Sub-task accuracy with and without the evolved tool, on two in-scope sub-tasks and one out-of-scope (shaded). Full numbers in Appendix D.

with five samples spends five times the budget for ten-task means of $+0.79/-0.21/+1.64$, and few-shot ICL, which reads the same training split our method learns from, is strongly negative on Classification under all three backbones. None of these alternatives is free of human design effort.

4.3 Transfer Analysis

We test whether a library evolved on one model still helps a different one. For each task we install the library that GPT-5.6-luna evolved into Claude-opus-5, Claude-sonnet-5, and GPT-5.6-sol, and no test data is used to pick it. From the last block of Table 1 and from Fig. 3b-c, we observe:

Obs. ❸: A stronger model is not a substitute for an evolved library. As shown in Fig. 3b, without tools the three expensive models score no better than the cheapest one, with everything between 54.6 and 57.1. Installing luna's library lifts each of them on the ten-task mean by $+10.50$, $+10.65$, and $+8.27$, and only one of the thirty task results is negative. As a result, we can conclude that what they were missing was not a better model but the right tools, and that what the library carries is task knowledge rather than model-specific habits.

Obs. ❹: The library gains more on models stronger than the one that grew it. As shown in Fig. 3c, 21 of the 30 task results sit above the diagonal, so the same library usually gains more on the stronger model than it did on luna. On Anomaly the two Claude models gain $+26.25$ and $+26.50$, against $+14.69$ on luna itself. The tools compute the same evidence in every case, and what differs is how deeply a model can exploit it. Therefore, growing the tools need not happen on an expensive model at all.

4.4 Ablation and General Analysis

We first remove one component at a time, then open the deployed libraries and check how the gate behaves. We ablate on two datasets, Anomaly and TSAQA-DT, each under mini and terra, and the four cells were chosen before the runs. Against the full method we compare six changes, named as in Fig. 4. (**1**) **w/o cali.** drops the train-only decision trees, which only the Anomaly libraries carry. (**2**) **w/o gate** force-accepts the candidate library. (**3**) **w/o scope** removes the structured scopes. (**4**) **w/ chronos** and (**5**) **w/ 21 tools** replace the empty root by a pre-installed forecasting tool and by the 21 expert-curated tools of TimeART. (**6**) **w/o residual chain** overwrites the base answers unconditionally instead of reviewing them. From Figs. 4 and 5, we observe:

Obs. ❺: Every component earns its place, and none can be removed for free. The full method reaches the highest test accuracy in all four cells. Calibration and the gate cost the most: without calibration both Anomaly runs deploy nothing, and without the gate the mini run deploys a library that ends 1.25 below its own base, where the full method ends 16.25 above. Removing structured scopes costs 1.00 to 9.25 points. In Fig. 4b, overwriting fixes 16 to 25 more answers per cell than reviewing does, and breaks far more: 59 versus 19, 32 versus 22, 26 versus 3, and 28 versus 10. The chain adds no repairs itself, but keeps a good library from breaking answers that were already right.

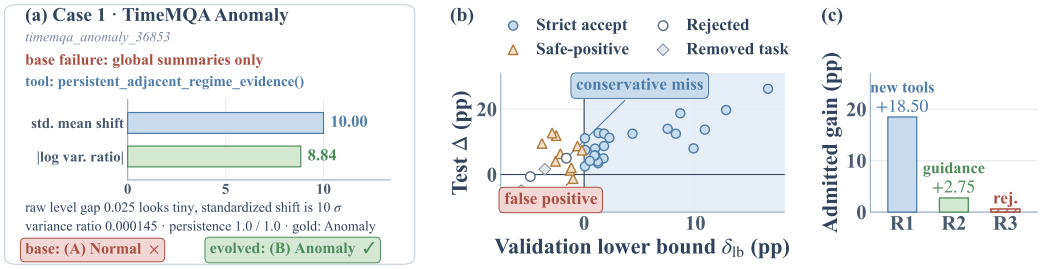


Figure 5: What the agent builds, and how the gate behaves. **(a)**: One question from the report-only test trace: the tool the agent built, the numbers it returned, and the answer before and after. **(b)**: Gate calibration across all formal runs, validation lower bound against deployed test delta, with the report-only diagnostic for rejected rounds. **(c)**: Admitted gain per round on a three-round run (terra, Anomaly). Two further cases in Appendix B.

Obs. ❻: A pre-installed root is not a shortcut. As shown in Fig. 4a, both alternative roots end below the empty root in all four cells: the Chronos wrapper by 3.04 to 18.50 points and the 21 expert-curated tools of TimeART by 8.25 to 21.50. The human library is the weaker start of the two, and evolution from it is rejected by the gate on both mini tasks, and on TSAQA-DT under terra it ends 0.50 below its own base. Tools written before deployment do not become the right tools when an agent evolves on top of them, which is the **Human–Agent Tool Misalignment** of Phenomenon ❶ appearing inside our own method.

Obs. ❼: The admitted tools are targeted and used exactly where they claim. As illustrated in Fig. 4c, on TimeMQA Classification the deployed tool computes exactly the band powers that the question's criteria name, and the out-of-scope activity sub-task stays at exactly 45.0. On TSAQA-DT, the libraries evolved under all three backbones independently rediscover the same Fourier and wavelet family, so the measurement contract, not a lucky sample, determines what gets built. 25 of the 29 deployed libraries are pure tools, and the other four admitted a textual patch, because the gate arbitrates by measured repair and harm rather than by artifact type.

Obs. ❽: The repairs are single missing measurements, not better reasoning. As shown in Fig. 5a, the base agent fails this question without computing anything, reading a global summary of a series whose overall spread looks ordinary. The evolved tool supplies exactly one measurement and the answer flips. The raw level gap between the two adjacent windows is 0.025, but standardized it is $10\,\sigma$, and the variance ratio of 0.000145 says the second window is almost flat. The two cases in Appendix B fail the same way: the base agent is not bad at arithmetic but simply never does it.

Obs. ❾: The gate errs visibly, in both directions, and the loop stops itself. As shown in Fig. 5b, every strict-region acceptance ($\delta_{lb} > 0$) lands test-positive. The borderline band contains one false positive (a T1 library at -1.27, the only accepted library that lands test-negative), and the rejected region contains one conservative false negative (a Merrill library whose report-only diagnostic scored $+5.00$). A rejected round rolls back cleanly and counts as zero. As illustrated in Fig. 5c, on a three-round run, round 1 admits new tools (validation gain $+18.50$), round 2 admits only usage guidance ($+2.75$), and round 3 is rejected outright. Every tool in the final library carries a round-1 identifier, so the tools stabilize in one round and later rounds only teach how to use them.

5 Conclusion

This paper studies two failures of tool-based time series agents: the tools humans supply are not the ones the agent needs at runtime, and an agent's own revisions break answers that the final score hides. To address them, we introduce TimeEvo, which turns diagnosed failures into evidence-only tools and admits a candidate library only when a paired comparison shows it fixes more than it breaks, rolling the round back otherwise. Experiments on ten time series QA tasks and three backbones show that TimeEvo, starting from an empty library, improves accuracy on every task and every backbone, and that a library grown on a cheap model still gains when it is installed into stronger ones. An agent's own failures are therefore a better specification for tools than human anticipation.

AI USE STATEMENT

In this work, we used generative AI tools to polish the writing, to draft some of the plotting scripts, and to help debug and correct parts of the code. We did not use generative AI tools for research ideation, experimental design, or the literature survey. All AI-assisted text was read and verified by the authors, all AI-assisted code was reviewed and tested by the authors before any reported run, and every number that appears in a figure or table was checked against the experiment logs. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Appendix C records the complete frozen protocol, every configuration knob, and the step-by-step round procedure. Appendix A gives the source and the frozen split sizes of each task, and Appendix D lists the full numbers behind every ablation and the per-run evolution funnel. The plotting scripts read their values directly from the result tables, and the code is available at `https://anonymous.4open.science/r/TimeEvo-652E`.

REFERENCES

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, et al. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025.

Jorind Beqari, Joseph Powell, Jacob Hurd, Alexandra L Potter, Meghan McCarthy, Deepti Srinivasan, Danny Wang, James Cranor, Lizi Zhang, Kyle Webster, et al. A pilot study using machine learning algorithms and wearable technology for the early detection of postoperative complications after cardiothoracic surgery. *Annals of surgery*, 281(3):514, 2024.

Yifu Cai, Arjun Choudhry, Mononito Goswami, and Artur Dubrawski. Timeseriesexam: A time series understanding exam. *arXiv preprint arXiv:2410.14752*, 2024.

Ziyang Cheng, Jie Yang, Yixin Song, Dawei Cheng, Guang Yang, and Bo Wang. Neighbor-enhanced graph pre-training and prompt learning framework for fraud detection. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 5617–5625, 2025.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023.

Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels: A. dempster et al. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

Haonan Dong, Wenhao Zhu, Guojie Song, and Liang Wang. Aurora: Breaking low-rank bottleneck of lora with nonlinear mapping. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 36929–36961. Curran Associates, Inc., 2025. doi: 10.52202/085713-1241. URL `https://proceedings.neurips.cc/paper_files/paper/2025/file/34aae5b7991a4fb4fe3081ca4f94549c-Paper-Conference.pdf`.

Haonan Dong, Qiguan Feng, Kehan Jiang, Haoran Ye, Xin Zhang, and Guojie Song. Agent-valuebench: A comprehensive benchmark for evaluating agent values, 2026. URL `https://arxiv.org/abs/2605.10365`.

Shicheng Fan, Nour Elhendawy, Jianle Sun, Ke Fang, Kun Zhang, Yihang Wang, and Lu Cheng. Mosaic: Module discovery via sparse additive identifiable causal learning for scientific time series. *arXiv preprint arXiv:2605.05524*, 2026a.

Shicheng Fan, Kun Zhang, and Lu Cheng. Trace: Trajectory recovery for continuous mechanism evolution in causal representation learning. *arXiv preprint arXiv:2601.21135*, 2026b.

Zhengyu Fang, Danny Wang, Alexandra L Potter, Lizi Zhang, Yuhang Zhang, Bryan Rettner, Alexander Zhu, Meghan McCarthy, Quiana Guo, Deepti Srinivasan, et al. A novel wearables-based sleep quality index to quantify postoperative sleep quality. *The Journal of Thoracic and Cardiovascular Surgery*, 2026.

Azul Garza and Renée Rosillo. Timecopilot. *arXiv preprint arXiv:2509.00616*, 2025.

Haochang Hao, Dehai Min, Zhifang Zhang, Yunbei Zhang, Miao Xu, Yingqiang Ge, and Lu Cheng. Poise: Position-aware undetectable skill injection on llm agents. *arXiv preprint arXiv:2606.07943*, 2026.

Yifan Hu, Jie Yang, Xilin Dai, Wanxu Cai, Kuiye Ding, Yuante Li, Qinghua Liu, Enze Ma, Zhiyuan Qu, Yixin Wang, et al. The landscape of agentic time series systems: Architectures, reliability, and frontiers.

Yifan Hu, Jie Yang, Tian Zhou, Peiyuan Liu, Yujin Tang, Rong Jin, and Liang Sun. Bridging past and future: Distribution-aware alignment for time series forecasting. In *International Conference on Learning Representations*, volume 2026, pp. 126148–126172, 2026.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. In *International conference on learning representations*, volume 2024, pp. 23857–23880, 2024.

Baoyu Jing, Sanhorn Chen, Lecheng Zheng, Boyu Liu, Zihao Li, Jiaru Zou, Tianxin Wei, Zhining Liu, Zhichen Zeng, Ruizhong Qiu, et al. Tsaqa: Time series analysis question and answering benchmark. In *Proceedings of the Fifth Workshop on Generation, Evaluation and Metrics (GEM)*, pp. 944–979, 2026.

Yaxuan Kong, Yiyuan Yang, Yoontae Hwang, Wenjie Du, Stefan Zohren, Zhangyang Wang, Ming Jin, and Qingsong Wen. Time-mqa: Time series multi-task question answering with context enhancement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29736–29753, 2025.

Yaxuan Kong, Qingren Yao, Yuqi Nie, Yichen Li, Yilei Shao, Stefan Zohren, Anna Vettoruzzo, Joaquin Vanschoren, Ming Jin, and Qingsong Wen. Timesage-mt: A multi-turn benchmark for evaluating agentic time series reasoning. *arXiv preprint arXiv:2606.01498*, 2026.

Chenghao Liu, Taha Aksu, Juncheng Liu, Xu Liu, Hanshu Yan, Quang Pham, Silvio Savarese, Doyen Sahoo, Caiming Xiong, and Junnan Li. Moirai 2.0: When less is more for time series forecasting. *arXiv preprint arXiv:2511.11698*, 2025a.

Hangchen Liu, Dongyuan Li, Renhe Jiang, Jiewen Deng, Weiwei Ye, and Yoshihide Sekimoto. Timeclaw: A time-series ai agent with exploratory execution learning. *arXiv preprint arXiv:2605.10038*, 2026.

Penghang Liu, Elizabeth Fons, Svitlana Vyetrenko, Daniel Borrajo, Vamsi K Potluru, and Manuela Veloso. Ts-agent: A time series reasoning agent with iterative statistical insight gathering. In *First Workshop on Foundations of Reasoning in Language Models*, 2025b.

Qinghua Liu and John Paparrizos. The elephant in the room: Towards a reliable time-series anomaly detection benchmark. *Advances in Neural Information Processing Systems*, 37:108231–108261, 2024.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *International conference on learning representations*, volume 2024, pp. 11116–11140, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Mike A Merrill, Mingtian Tan, Vinayak Gupta, Thomas Hartvigsen, and Tim Althoff. Language models still struggle to zero-shot reason about time series. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 3512–3533, 2024.

Chunyu Miao, Henry Peng Zou, Yangning Li, Yankai Chen, Yibo Wang, Fangxin Wang, Yifan Li, Wooseong Yang, Bowei He, Xinni Zhang, et al. Recode-h: A benchmark for research code development with interactive human feedback. *arXiv preprint arXiv:2510.06186*, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Hao Wang, Lichen Pan, Yuan Shen, Zhichao Chen, Degui Yang, Yifei Yang, Sen Zhang, Xinggao Liu, Haoxuan Li, and Dacheng Tao. Fredf: Learning to forecast in the frequency domain. In *International Conference on Learning Representations*, volume 2025, pp. 6893–6922, 2025.

Muyan Weng, Defu Cao, Wei Yang, Yashaswi Sharma, and Yan Liu. Temporalbench: A benchmark for evaluating llm-based agents on contextual and event-informed time series tasks. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 9997–10008, 2026.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, et al. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025.

Xingjian Wu, Junkai Lu, Zhengyu Li, Xiangfei Qiu, Jilin Hu, Chenjuan Guo, Christian S Jensen, and Bin Yang. Timeart: Towards agentic time series reasoning via tool-augmentation. *arXiv preprint arXiv:2601.13653*, 2026.

Zhe Xie, Zeyan Li, Xiao He, Longlong Xu, Xidao Wen, Tieying Zhang, Jianjun Chen, Rui Shi, and Dan Pei. Chatts: Aligning time series with llms via synthetic data for enhanced understanding and reasoning. *arXiv preprint arXiv:2412.03104*, 2024.

Jie Yang, Yifan Hu, Kexin Zhang, Luyang Niu, Philip S Yu, and Kaize Ding. Revisiting multivariate time series forecasting with missing values. *arXiv preprint arXiv:2509.23494*, 2025a.

Jie Yang, Rui Zhang, Ziyang Cheng, Dawei Cheng, Guang Yang, and Bo Wang. Grad: Guided relation diffusion generation for graph augmentation in graph fraud detection. In *Proceedings of the ACM on Web Conference 2025*, pp. 5308–5319, 2025b.

Jie Yang, Yifan Hu, Yuante Li, Kexin Zhang, Kaize Ding, and Philip S Yu. From observations to states: Latent time series forecasting. *arXiv preprint arXiv:2602.00297*, 2026a.

Jie Yang, Kexin Zhang, Guibin Zhang, Philip S Yu, and Kaize Ding. Glocal information bottleneck for time series imputation. *Advances in Neural Information Processing Systems*, 38:104452–104484, 2026b.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, et al. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026c.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Wen Ye, Wei Yang, Defu Cao, Yizhou Zhang, Lumingyuan Tang, Jie Cai, and Yan Liu. Ts-reasoner: Domain-oriented time series inference agents for reasoning and automated analysis. *arXiv preprint arXiv:2410.04047*, 2024.

Yao Yin, Zhenyu Xiao, Musheng Li, Yiwen Liu, Sutong Nan, Yiting He, Ruiqi Wang, Zhenwei Zhang, Qingmin Liao, and Yuantao Gu. Mmts-bench: A comprehensive benchmark for time series understanding and reasoning. *arXiv preprint arXiv:2602.08588*, 2026.

Fangxu Yu, Tao Feng, Dehai Min, Lu Cheng, Ge Liu, and Tianyi Zhou. Tsrouter: Dynamic modality-model selection for time series reasoning. *arXiv preprint arXiv:2607.08940*, 2026.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, et al. Coevoskills: Self-evolving agent skills via co-evolutionary verification. *arXiv preprint arXiv:2604.01687*, 2026.

Haokun Zhao, Xiang Zhang, Jiaqi Wei, Yiwei Xu, Yuting He, Siqi Sun, and Chenyu You. Timeseriesscientist: A general-purpose ai agent for time series analysis. *arXiv preprint arXiv:2510.01538*, 2025.

Sibo Zhu, Shicheng Fan, Xinyue Wang, Wenyi Wu, Kun Zhou, and Biwei Huang. Rsiagent: Autonomous exploration for recursive self-improvement in new environments, 2026. URL https://arxiv.org/abs/2609.15364.

A Dataset Details

Table 2: The ten-task suite: data sources and frozen train/validation/test sizes. Merrill is downsampled from its official 5779/1000/1036 split, and every other task uses its full frozen pool. Splits are grouped by source series, so questions from the same series never cross splits.

Task	Source	Train	Val	Test
T1	TemporalBench (Weng et al., 2026)	400	156	158
T2	TemporalBench (Weng et al., 2026)	324	123	126
T3	TemporalBench (Weng et al., 2026)	517	196	202
T4	TemporalBench (Weng et al., 2026)	324	123	126
TSExam	TimeSeriesExam (Cai et al., 2024)	448	149	149
Merrill	Time-Series Reasoning (Merrill et al., 2024)	240	120	240
Anomaly	Time-MQA (Kong et al., 2025)	400	400	400
Classification	Time-MQA (Kong et al., 2025)	240	120	240
Match	MMTS-Bench (Yin et al., 2026)	160	80	160
TSAQA-DT	TSAQA (Jing et al., 2026)	400	400	400

The ten tasks come from six public sources, and every source is adapted to one row contract: a question, a closed option set whose gold answer appears verbatim among the options, one or two primary series, and optional named covariate or candidate series. This appendix describes each source and the split design, and Table 2 lists the frozen sizes.

TemporalBench (T1–T4) (Weng et al., 2026). The four tiers share one source pool and are split by source sample, so all questions expanded from the same sample stay in the same split. T1 asks property questions about the observed history (trend, volatility, seasonality, anomaly), with label spaces read from the source prompts rather than invented. T2 and T4 are forecasting-style multiple-choice questions, distinguished only by whether event context is present, and T3 packs several sub-questions per sample with clinical covariates such as temperature and respiratory rate. All series are univariate with a median length of 300 points, and only the observed history is ever exposed, never another tier's future targets.

TimeSeriesExam (Cai et al., 2024). An exam of time series knowledge and the only task with two-series questions (99 single and 50 dual in the test split). Its series are the longest in the suite (median 1024 points). The split groups all questions that share the exact same input series and stratifies by

question type, difficulty, option count, and even the position of the correct option, using no model outputs.

Merrill (Merrill et al., 2024). Scenario matching: every question shows one series and four candidate natural-language scenario descriptions, under one shared question text. We keep the official train/validation/test files and downsample them to 240/120/240 for cost. Series lengths span 12 to 1460 points, the widest range in the suite.

Time-MQA (Anomaly and Classification) (Kong et al., 2025). The source embeds each series inside the question text. We extract the longest numeric list into a machine-readable series so tools can compute on it, and turn the closed-set free-text answer into explicit options. From 37,000 source rows, one representative row is kept per unique series, and one group whose duplicate series carried contradictory labels is excluded entirely. Anomaly is a two-way normal-versus-anomaly task with the shortest series in the suite (8 to 64 points, median 16). Classification mixes a six-way activity-recognition half with a two-way freeze-of-gait half, over accelerometer snippets of 9 to 30 points.

MMTS Match (Yin et al., 2026). Each question shows a target series and four candidate series and asks which candidate is most similar. The four candidates are carried as named covariate series, and the task is kept as a four-way choice rather than rewritten into easier pairwise comparisons. The 400 source rows contain only 112 unique target series, so whole target groups are assigned to one split.

TSAQA (Data-Transformation) (Jing et al., 2026). Closed-set questions about transformed series (multiple-choice and true-or-false halves), with options lifted from the source text and one series per question. Row identifiers encode the official split, so our frozen manifest never crosses the official train/validation/test boundaries.

Leakage control and heterogeneity. Across the suite, splits are grouped at the series level (all questions sharing a series move together), duplicate series are collapsed to one representative, contradictory groups are dropped, official split boundaries are preserved, and future windows are never exposed to the agent or its tools. Series lengths span two orders of magnitude (8 to 1460 points) and option counts range from two to six, so absolute accuracies are not comparable across tasks. All comparisons in this paper are therefore within-task paired deltas.

B Additional Case Studies

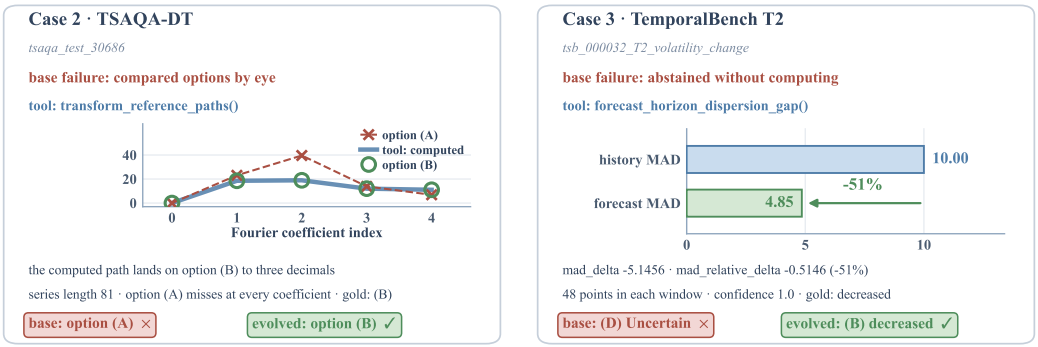


Figure 6: Two further repairs. **Left**: the agent picked the Fourier spectrum that looked right, and the synthesized tool computed the real one, which matches the correct option to three decimals. **Right**: the agent abstained because it never compared the two windows, and the tool measured the dispersion drop directly.

Both cases below are single questions from the report-only test traces of accepted rounds, quoted with the tool's own returned values, and both are drawn in Fig. 6. For each case we give the question, the answer before and after the round, the failure the planner diagnosed, the measurement contract it wrote, and the numbers the synthesized tool returned on that question.

Comparing options by eye. On TSAQA-DT (`tsaqa_test_30686`, terra), the question asks which of the listed option vectors is the Fourier transform of the given series. The base agent

answers (A), whose leading coefficients $[0, 23.09, 39.60, 13.86, 6.90]$ are the largest and therefore look the most like a spectrum, and the gold answer is (B), $[0, 18.55, 18.96, 12.00, 11.07]$. The planner diagnoses the bucket as a candidate-comparison error, namely that the solver never produces aligned quantitative comparisons between the requested transform and the candidates, and writes a contract for candidate-wise normalized residuals after computing the requested representation and aligning each candidate by valid length and coefficient index. The synthesized tool `transform_reference_paths` returns a series length of 81 and a magnitude path of $[0.0005, 18.5500, 18.9578, 12.0046, 11.0670, 7.5275, \ldots]$, which matches option (B) to three decimals at every index, and the evolved agent answers (B).

Abstaining without computing. On TemporalBench T2 (`tsb_000032_T2_volatility_change`, mini), the question gives a history window and asks whether volatility over the forecast horizon rises, falls, or cannot be determined. The base agent answers (D) Uncertain, which is the fallback option rather than a wrong measurement, and the gold answer is (B) decreased. The planner diagnoses that the agent never computed the cross-window spread comparison the question needs, and writes a contract for a deterministic comparison of forecast-horizon dispersion against historical dispersion. The synthesized tool `forecast_horizon_dispersion_gap` returns a history MAD of 10.0000 against a forecast MAD of 4.8544, a delta of -5.1456, a relative delta of -0.5146, and 48 points in each window, and the evolved agent answers (B).

Neither repair required better reasoning about time series. In both cases the agent had already read the question correctly and simply had no number to read off, and one deterministic measurement was enough to settle the answer.

C Frozen Protocol and Round Procedure

This appendix records the complete frozen configuration and the step-by-step round procedure for reproducibility.

C.1 Round Procedure

0. **Root library.** The root is configurable and immutable. `empty` installs no tool and is the frozen protocol used for every reported result, while `chronos` installs a Chronos-2 wrapper and is retained only as an ablation.
1. **Base pass.** The current chain answers the full training split under a semantic-fingerprint cache. This yields the base accuracy of the round and the error set that everything downstream is built from.
2. **Failure clustering.** Errors are partitioned by whether a tool fired on them, and each partition is clustered independently by a triage LM in two stages. A clustering failure invalidates the round as measurement-invalid, which is a rollback rather than a quality verdict.
   - **Partition.** The `tool_fired` side collects questions where a tool ran and the answer was still wrong, which points at a defective or misused tool, and the `no_tool` side collects questions no tool reached, which points at a coverage gap. An empty root puts every error on the second side.
   - **Define.** The stage samples at most 150 errors and returns at most eight categories, each carrying a short label and a criteria string, with at most two retries.
   - **Assign.** Every error is sent in batches of 32 under a strict JSON contract that must cover each id in the batch, with at most two retries per batch and at most three passes over what remains. Small batches are deliberate, since one long response that fails validation would otherwise void a large batch.
   - **Resplit and drop.** A bucket larger than 64 errors is split once and not recursively. Errors that stay unassigned, and errors in buckets with fewer than two members, are dropped.
3. **Two-level planning.** A local planner diagnoses each bucket independently, and a global planner then coordinates the proposals before anything is synthesized.
   - **Local planner.** It sees the bucket's error dossier, at most 12 support examples, and at most 48 background examples, and returns one proposal card.

- **Descriptive fields.** The action (**CREATE**, **REFINE**, **RESCOPE**, or **RETIRE**), the artifact kind, the target tool, the failure type, the root cause, and the focus. These carry context only.
- **Contract fields.** Required measurement, required signature, success criterion, scope, anti-scope, matcher, and an optional calibration request. These are checked repeatedly downstream.
- **Global planner.** It merges proposals that ask for the same measurement and narrows scopes that overlap. Merging incompatible contracts is rejected mechanically rather than discouraged in the prompt, and after two failed attempts the round falls back to the validated local plans as singleton groups and continues.

4. **Synthesis.** Each work group gets at most K_{update}=3 sequential attempts.
   - **What the synthesizer sees.** The current library, rendered separately, together with a digest carrying the bucket dossier and its support examples, the planner's diagnosis, the contract, a compact timeline of previous failed attempts, and the numerically strongest previous attempt kept apart so that a weak later retry cannot displace it.
   - **Hard constraints.** The prompt fixes a single-function output with no `while` loops and a zero-argument signature.
   - **Checks before execution.** A single-function check, a smoke call built from the signature, and a context-selection check against the contract.
   - **Calibration.** When a contract asks for a decision boundary, the tool may carry a shallow decision tree of kind `binary_decision_tree_v1`, fitted on training rows alone with validation and test never read and answers stored only as hashes. The tree needs at least 16 samples per class and an internal 80/20 selection split grouped by input fingerprint, and it is accepted only at an internal balanced accuracy of at least $\beta = 0.55$. An accepted tree is AST-compiled into the tool source, and a calibrated tool may not call the forecasting primitive.
   - **Retry policy.** A contract-violating artifact receives one targeted repair, an undeliverable input signature is adapted between retries, and two consecutive zero-repair outcomes trigger a replan that may change the artifact kind, for example from a new tool to usage guidance.
5. **Candidate prefilter.** Each candidate is screened cheaply before any candidate reaches the round-level gate.
   - **Sampling.** The first 32 held-out errors of the candidate's own bucket, taken in a deterministic order that interleaves the members of a coordinated work group so that a bounded screen does not sample only the largest source bucket.
   - **Attribution.** A newly created tool is first checked for signature deliverability on those questions, while lifecycle updates are attributed through action-specific traces instead, because narrowing or rescoping can help precisely by preventing delivery.
   - **Criterion.** At least one attributable repair, meaning a question fixed with the tool bound, executed, and visible in the trace, within coarse caps on harm and execution errors.
   - **Stopping.** A repair rate of at least 0.40 stops the search early, and otherwise the best positive candidate after three attempts still reaches the gate.
6. **Whole-library gate.** The current library and the candidate library each answer the full validation split, and only questions both arms completed are compared, so helped and harmed are counted on identical questions.
   - **Statistic.** With n paired questions the gate computes $\delta = (h - m)/n$ and $\delta_{\text{lb}} = \delta - 1.96\sqrt{\max(h + m, 1)}/n$.
   - **Strict path.** A library passes when $h > 0$ and $\delta_{\text{lb}} > 0$.
   - **Single-round path.** A single-round run has no later round in which to accumulate evidence, so a supplementary rule admits a library when $\delta > 0$, the net repair is at least $\max(3, \lceil 0.02n \rceil)$, and the absolute harm rate is at most 0.10.
   - **Invalidity.** An execution error rate above 2% renders the measurement invalid, which is separated from a quality failure.
7. **One-shot attribution pruning.** On a failed gate with clearly harmful updates, all of them and their dependents are deleted in one transaction, the pruned library is re-verified in

exactly one more full paired run, and it is deployed only on strict improvement together with a passing gate, else the round rolls back. There is no second prune.

8. **Chain write and report-only test.** An accepted library first reconciles its declared scope fields against how its tools actually behaved, and because that edit changes the library it passes the same paired gate one final time. The round then appends exactly one repair stage to the chain. The test split is answered once per round for reporting only, and on a rollback the best failed candidate may receive one diagnostic test run that participates in no selection.

C.2 Frozen Configuration

Table 3 lists every knob that was fixed before the first run and never touched again, so that a reader can reproduce a round without reading the code. Three groups appear in the table. The first fixes the models and the shape of a round: all four agent roles run on the same backbone, synthesis runs at a higher reasoning effort because it writes code, the root library is empty, and a run is a single round with at most three synthesis attempts per work group. The second fixes the two levels of validation: the prefilter samples 32 held-out errors and demands one attributable repair, while the gate compares the whole library on the full validation split at $Z = 1.96$ and admits either through the strict lower bound or through the single-round rule with a net floor of three and a harm ceiling of 0.10. The third fixes how the agent answers: it consults a structured scope catalog before inspecting at most three tools, tools return numbers rather than verdicts, and no hand-written prompt is added for any task. Two entries deserve emphasis, because they are what makes a rejected round harmless: `AUTO_PRUNE` is off so that only one bounded prune can run, and `TEST_REJECTED_CANDIDATE` is on so that a rolled-back library still receives one diagnostic test run that participates in no selection.

Table 3: The frozen protocol. Every reported run uses these settings without exception.

Knob	Value	Meaning
Models and evolution		
`TASK_LM` / `TRIAGE_LM` / `PLANNER_LM`	backbone under evaluation, effort low	all agent roles
`SYNTH_LM`	same backbone, effort medium	tool synthesis
`ROOT_LIBRARY`	`empty`	evolution starts from no tools
`RESIDUAL_REPAIR`	`1`	residual chain, mandatory
`TOOL_CALIBRATION`	`1`	train-only calibration on
`ROUNDS` / K / `K_KIND` / `K_UPDATE`	`1 / 3 / 1 / 3`	single round, three retries
`SUPPORT_MAX` / `BACKGROUND_N`	`12 / 48`	planner-visible samples
`MAX_FAILURES` / `MAX_BUCKETS`	`all / all`	no truncation
`ROUND_PLAN_COORDINATION`	`1`	global coordinator on
Prefilter and gate		
`PREFILTER_N` / `MIN_HELPED`	`32 / 1`	prefilter sample and floor
`CANDIDATE_EARLY_STOP_REPAIR`	`0.40`	early-stop threshold
`GATE_MODE` / `LLM_GATE`	`library / 0`	whole-library gate, no LLM judge
`LIBGATE_N` / `LIBGATE_Z`	`all` / `1.96`	full split, lower-bound Z
`ONE_ROUND_SAFE_POSITIVE`	`1`	safe-positive rule active
`SAFE_MIN_NET` / `SAFE_MAX_HARM_RATE`	`3 / 0.10`	net floor, harm ceiling
`AUTO_PRUNE` / `ONE_SHOT_ATTRIBUTION_PRUNE`	`0 / 1`	one bounded prune only
`RESCOPE_AFTER_ROUND`	`1`	scope recalibration before chain write
`TEST_REJECTED_CANDIDATE`	`1`	diagnostic test on rollback
Routing and answering		
`ROUTING` / `ENCODING` / `CONTEXT`	`inline / shape / preview`	scope catalog first, ≤ 3 inspects
`TOOL_VERDICT` / `STRONG_PROMPT` / `TS_PROMPT`	`0 / 0 / 0`	no verdicts, no hand-written prompts
`BASE_CACHE` / `EVAL_CACHE`	`1 / 0`	base answers cached once per run

D Full Ablation Results

D.1 Component Ablations

Table 4 is the numeric table behind Fig. 4a, with one row per configuration on each of the four cells. The two *Val* columns are what the gate actually saw before it made its decision: the paired delta on the validation split, its lower bound, and the helped and harmed counts behind them. Reading them next to *Final* shows how well the gate's own evidence predicted the deployed outcome, which

Table 4: Component ablations on TimeMQA Anomaly (main) and TSAQA-DT (reproduction), each under two backbones. Val columns show what the gate saw (paired delta, lower bound, helped/harmed). Base and Final are that run's own test accuracy before and after the round, with the gain the round itself produced in parentheses, and the last column gives the gap to the full method. A rejected library is not deployed, so its final accuracy equals its base and the diagnostic column reports what the rejected library would have scored. Calibration is marked N/A on TSAQA-DT, whose libraries carry no calibration trees.

Dataset / backbone	Configuration	Val δ / δ_{1b} (h/m)	Rejected diagnostic	Base	Final (evo gain)	Gap to full	Tools
Anomaly / mini	full method	$+12.25$ / $+7.63$ (69/20)	—	58.75	**75.00** ($+16.25$)	—	7
	w/o calibration	-0.50 / -1.70 (2/4)	-0.75	63.50	63.50 ($+0.00$)	-16.25	0 (rejected)
	w/o gate	-2.00 / -5.67 (24/32)	—	63.00	61.75 (-1.25)	-17.50	4
	w/o scope	$+5.25$ / $+1.06$ (47/26)	—	63.00	70.75 ($+7.75$)	-8.50	5
	w/ chronos root	-0.25 / -2.15 (7/8)	$+3.25$	62.00	62.00 ($+0.00$)	-16.25	1
	w/ 21 tools root	$+0.50$ / -2.00 (14/12)	-1.25	63.00	63.00 ($+0.00$)	-16.25	21
Anomaly / terra	full method	$+13.50$ / $+9.06$ (68/14)	—	62.50	**75.00** ($+12.50$)	—	8
	w/o calibration	-4.50 / -9.50 (43/61)	-4.25	62.50	62.50 ($+0.00$)	-12.50	0 (rejected)
	w/o gate	$+3.50$ / -0.83 (46/32)	—	57.50	63.75 ($+6.25$)	-6.25	8
	w/o scope	$+10.50$ / $+5.96$ (64/22)	—	58.50	70.75 ($+12.25$)	-0.25	8
	w/ chronos root	$+16.25$ / $+10.90$ (92/27)	—	56.75	71.75 ($+15.00$)	$+2.50$	9
	w/ 21 tools root	$+10.50$ / $+6.28$ (58/16)	—	53.75	64.25 ($+10.50$)	-2.00	26
TSAQA-DT / mini	full method	$+19.75$ / $+15.39$ (79/0)	—	67.75	**83.75** ($+16.00$)	—	4
	w/o calibration	N/A	N/A	N/A	N/A	N/A	N/A
	w/o gate	$+0.50$ / -1.33 (8/6)	—	65.50	68.75 ($+3.25$)	-12.75	4
	w/o scope	$+6.25$ / $+3.52$ (28/3)	—	68.75	74.50 ($+5.75$)	-10.25	6
	w/ chronos root	$+3.00$ / $+1.17$ (13/1)	—	61.00	65.25 ($+4.25$)	-11.75	2
	w/ 21 tools root	$+0.50$ / -1.05 (6/4)	$+1.00$	62.25	62.25 ($+0.00$)	-16.00	21
TSAQA-DT / terra	full method	$+15.00$ / $+10.67$ (69/9)	—	71.00	**85.50** ($+14.50$)	—	4
	w/o calibration	N/A	N/A	N/A	N/A	N/A	N/A
	w/o gate	$+15.50$ / $+11.64$ (62/0)	—	72.43	84.71 ($+12.28$)	-2.22	4
	w/o scope	$+14.50$ / $+10.64$ (60/2)	—	71.50	84.50 ($+13.00$)	-1.50	2
	w/ chronos root	$+11.50$ / $+7.23$ (61/15)	—	69.42	82.46 ($+13.04$)	-1.46	3
	w/ 21 tools root	$+3.25$ / $+0.52$ (22/9)	—	77.75	77.25 (-0.50)	-15.00	25

Table 5: The residual chain against unconditional overwrite. Both arms deploy the same library on the same questions, and the only difference is whether the library reviews the base answer or replaces it outright. Fixed counts previously wrong answers the update repaired, and harmed counts previously correct answers it broke.

	Fixed		**Harmed**	
Dataset / backbone	Chain	Overwrite	Chain	Overwrite
Anomaly / mini	33	58	19	59
Anomaly / terra	77	93	22	32
TSAQA-DT / mini	56	77	3	26
TSAQA-DT / terra	59	78	10	28

is the calibration plotted in Fig. 5b. The *Rejected diagnostic* column is filled only for rounds the gate rejected. Such a round deploys nothing, so its final accuracy equals its base, and the diagnostic reports what the rejected library would have scored had it been deployed, measured once and used in no decision. *Final* carries the gain the round itself produced, *Gap to full* compares it against the full method on the same cell, and *Tools* gives the size of the deployed library, with *rejected* marking a round that shipped none.

D.2 Residual Chain against Overwrite

The residual chain is not a removed component but a second way of applying the same library, so Table 5 reports it separately. Both arms deploy the same tools on the same questions, and the only difference is whether an admitted tool reviews the base answer or replaces it outright. The two arms are what Fig. 4b draws.

D.3 Evolution Funnel across Tasks and Backbones

Table 6 summarizes attrition across the one-round process, and Table 7 opens the full trace of one complete GPT-5.6-luna run per task, one row per task. Every field is read from that run's sole round, with no cross-run imputation and no missing fields.

Across the 30 runs, 4,898 of 4,910 errors were assigned to a bucket: only 12 remained unassigned and none were discarded for an undersized bucket. Coordination reduced 197 local proposals to 194 contracts, while nine of 203 failure buckets, spread across seven runs, produced no candidate. The prefilter retained 227 of 483 synthesis attempts (47.0%), after which 113 artifacts were deployed: 108 tools, five prompt patches, and no usage-only artifacts. The recorded final outcomes were 22 passes, seven passes after one-shot pruning (17 artifacts removed in total), and one rollback. Thus most cost occurs at executable synthesis, the prefilter, and the library gate rather than during clustering or coordination.

Funnel width reflects error diversity rather than the eventual payoff. For example, the T3 run has 367 errors, 11 buckets, 31 synthesis attempts, and six deployed artifacts, yet a test gain of $+7.43$ pp, while the MMTS Match run has only 28 errors, three buckets, six attempts, and two deployed artifacts, yet gains $+18.75$ pp. More failures therefore create more search work, but not necessarily more useful headroom.

Table 6: Aggregate one-round evolution funnel over the 30 task–backbone runs. Totals count events across runs, and medians and ranges are computed per run. Yield compares each row with the stage above it.

Stage	**Total**	**Median [min–max]**	**Yield**
Errors on the training split	4,910	156 [26–369]	—
Errors assigned to a bucket	4,898	155.5 [24–369]	99.76%
Buckets	203	6 [2–12]	—
Buckets that produced a candidate	194	6 [2–12]	95.57%
Local proposals	197	6 [2–12]	—
Contracts after coordination	194	6 [2–12]	98.48%
Synthesis attempts	483	16 [4–32]	—
Candidates past the prefilter	227	7 [1–17]	47.00%
Deployed artifacts	113	3.5 [0–8]	49.78%

Table 7: One-round evolution funnel of one complete GPT-5.6-luna run per task, recorded end to end for process analysis. *Errors* is the base pass's error count on the training split and *Tagged* the errors assigned to a bucket. *Local* and *Contracts* are proposals before and after coordination, *Synth.* counts every synthesis attempt, *Prefilter* the attempts that passed it, and *Deploy T/P* the deployed tools and prompt patches. *Final gate* records a direct pass, a rollback, or the number of artifacts removed by the one-shot prune before a passing re-verification. Each test delta is that run's own paired delta rather than the number reported in Table 1.

Task	Errors	Tagged	Buckets	Local	Contracts	Synth.	Prefilter	Deploy T/P	Final gate	Δ_{test} (pp)
T1	246	246	8	8	8	20	11	4/0	prune 3 → pass	+9.49
T2	218	218	8	8	8	22	8	4/0	pass	+12.70
T3	367	367	11	11	11	31	12	6/0	pass	+7.43
T4	198	198	5	5	4	11	4	2/0	pass	+12.70
TSExam	119	119	8	8	8	22	13	6/0	pass	+8.72
Merrill	27	25	6	4	4	10	7	2/1	pass	+2.50
Anomaly	164	164	8	8	8	12	11	8/0	pass	+19.75
Classification	158	158	6	6	6	16	1	1/0	pass	+26.25
Match	28	28	3	3	3	6	4	2/0	prune 1 → pass	+18.75
TSAQA-DT	121	121	7	7	6	14	4	3/0	pass	+13.75