AGENTS: HARNESSING RUNTIME TOOL ORCHESTRATION FOR TIME SERIES REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Time series reasoning, which involves understanding and reasoning over time series data to answer complicated questions, has been increasingly important across domains, such as healthcare, energy, and climate science. Modern time series reasoning models are mostly based on large language models and utilize statistical tools to extract temporal structure from numerical data. However, existing agents often couple tool use with model reasoning steps, where the model selects tools from the question and previous observations, limiting independent control over tool selection, execution, and output organization. Furthermore, accumulating tool outputs and execution histories can lead to long contexts that increase the overhead of subsequent reasoning. To address these problems, in this paper, we propose *AgenTS*, a runtime harness that controls tool selection, execution, and evidence construction as a separate stage from model reasoning. The core idea of AgenTS is to treat tool use as a runtime stage separate from model reasoning, enabling the harness to adapt analytical computation to each instance and explicitly manage the evidence for subsequent reasoning. Specifically, to make the analytical requirements explicit, we first propose a runtime state representation that couples the relevance and intensity of temporal properties from the input. Then, to govern tool execution at runtime, we propose a toolpath orchestration policy that effectively plans tool execution with an online learning memory to shift planning from repeated synthesis to procedural reuse. Moreover, to separate the analytical state from the execution trace, we propose a reasoning workspace reconstruction mechanism that curates tool outputs into structured evidence. Finally, we conduct experiments on two main benchmarks covering 14 datasets and 10 reasoning tasks, demonstrating that AgenTS achieves state-of-the-art performance, with an average improvement of more than 30% across tasks while using 33% fewer tools than strongest baseline.

1 INTRODUCTION

Time series reasoning, which aims to model numerical time series data to provide answers for complicated questions and tasks such as forecasting (Zhao et al., 2025), anomaly detection (Tao et al., 2026b), and causality discovery (Du et al., 2025), has become increasingly important across a range of application domains, including finance (Li et al., 2025), energy (Husom et al., 2025), and healthcare (Wang et al., 2025c). Currently, most time series reasoning methods are powered by large language models (LLMs), which demonstrate promising general reasoning capabilities in different modalities including time series data (Guan et al., 2025; Luo et al., 2025). Existing approaches for time series reasoning can be primarily categorized along three lines: modality adaptation that aligns numerical series with the token space of pretrained LLMs (Kong et al., 2025; Xie et al., 2025), post-training methods that enhance reasoning through reinforcement learning over multi-step processes (Guan et al., 2025), and agentic frameworks that augment LLMs with external computation tools for complex time series reasoning (Wu et al., 2026; Zhao et al., 2025).

While current agentic frameworks for time series reasoning use external statistical tools for numerical analysis (Liu et al., 2025; Wu et al., 2026), most of them couple tool use with model reasoning. Specifically, the model determines the next tool call from its current reasoning and the evidence collected, and then continues reasoning with the returned result (Yao et al., 2023; Liu et al., 2025). Analytical tools therefore unfold within the reasoning process, which may leave the tool configuration without an explicit control point and further lead to unstable tool use and repeated synthesis of

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.

analytical procedures (Ning et al., 2026). Moreover, given a time series instance, the required analysis depends on both the question and the input series. For example, a question may target a specific temporal property, such as periodicity, while the series may exhibit multiple coexisting patterns, such as trend, non-stationarity, or regime shifts. Thus, tool configuration should consider both the relevance and the intensity of temporal properties to enable governed tool execution at runtime.

To address the above problems, we propose *AgenTS*, a runtime harness for agentic orchestration in time series reasoning. The central idea of AgenTS is to formulate time series reasoning as a runtime harness problem, where the harness coordinates task analysis, tool execution, evidence construction, and model reasoning at inference time. In particular, we propose a runtime state construction mechanism for time series reasoning, implemented through the *Profiler*, which couples the relevance and intensity of temporal properties from the input as an explicit control point at runtime. To govern tool execution from this control point, we further introduce a time series toolpath orchestration policy through the *Orchestrator*, which combines a toolpath planning policy and accumulated experience through an online learning memory. To decouple the analytical evidence from the execution trace, we introduce a *Synthesizer* that executes the selected toolpaths and reconstructs a structured reasoning workspace from their outputs for reliable answer generation. In summary, our main contributions are:

- We formulate time series reasoning as a runtime harness problem that treats tool use as a stage separate from model reasoning. The harness governs tool execution from an explicit control point and provides the model with structured temporal evidence instead of raw execution traces.
- We propose *AgenTS*, a runtime harness for time series reasoning. The harness performs runtime state construction to couple the relevance and intensity of temporal properties through the *Profiler*, toolpath orchestration with online learning to plan tool execution from accumulated experience through the *Orchestrator*, and runtime execution with reasoning workspace reconstruction to separate the analytical state from the execution trace through the *Synthesizer*.
- We conduct extensive experiments on two main time series benchmarks (i.e., TSR-Suite (Guan et al., 2025) and MTBench (Chen et al., 2025)) covering 14 datasets and 10 reasoning tasks, which demonstrate AgenTS consistently achieves state-of-the-art performance. On TSR-Suite, AgenTS shows an average improvement of 33% compared to the strongest baseline, including nearly twofold improvements in decision-making tasks. On MTBench, AgenTS achieves performance improvements of up to 74% in financial data.

2 Related Work

Time Series Reasoning. Time series reasoning refers to understanding and reasoning over temporal data (Kauppinen & Hyvönen, 2007; Eichler & Didelez, 2012; Chow et al., 2024) to support tasks such as forecasting (Wang et al., 2025b), causality discovery (Du et al., 2025), decision making (Li et al., 2025), and scenario understanding (Kegel et al., 2017). Early studies primarily relied on statistical methods such as ARIMA, followed by deep neural forecasting models including Informer (Zhou et al., 2021), Autoformer (Wu et al., 2021), and PatchTST (Nie et al., 2023). More recent foundation models pretrained on large time series corpora extend this approach to cross-domain generalization (Ansari et al., 2024; Das et al., 2024; Woo et al., 2024). To further bridge temporal dynamics with semantic understanding, recent work adopts LLMs as the backbone for these tasks. Early approaches rely on textualization, serializing numerical sequences into text for zero-shot inference (Gruver et al., 2023; Xue & Salim, 2024). Subsequent methods move beyond textual representations by aligning series embeddings with the LLM token space to better preserve numerical information (Jin et al., 2024; Sun et al., 2024; Pan et al., 2024). More recently, these methods further enhance LLMs by incorporating dedicated time series encoders (Xie et al., 2025; Parker et al., 2025; Wang et al., 2025b) and strengthen reasoning through post-training, including reinforcement learning (Luo et al., 2025), step-level supervision (Zhou et al., 2026), and multi-stage training schemes (Guan et al., 2025).

Agentic AI for Time Series. Agentic AI tackles complex multi-step tasks by integrating planning, acting, self-reflection, and tool use (Xi et al., 2025; Wang et al., 2024b). Early frameworks like ReAct (Yao et al., 2023) couple reasoning with acting. Reflexion (Shinn et al., 2023) iteratively refines outputs via self-reflection. Tool-learning methods enable agents to invoke external functions through self-supervised tool induction (Schick et al., 2023), reinforcement-learning fine-tuning (Feng et al., 2025), and large-scale tool retrieval (Qin et al., 2024; Patil et al., 2024). These developments have been adapted to time series, motivating recent agentic frameworks for time series reasoning (Chang et al.,

2025). Within this direction, some approaches enhance reasoning through ReAct-style loops (Liu et al., 2025) or compositional query decomposition into operator-based workflows (Ye et al., 2026). Others focus on strengthening tool usage, learning tool-using policies from large-scale expert trajectories (Wu et al., 2026) or leveraging memory-augmented reinforcement learning for adaptive decision making (Tao et al., 2026a). Beyond these, specialized systems further tailor agents to applications such as anomaly detection (Tao et al., 2026b) and multi-agent time series analysis (Zhao et al., 2025).

3 AGENTS

AgenTS is a runtime harness that separates tool-based temporal abstraction from evidence-based reasoning (Figure 1). We first introduce its formulation and disassembly principle in Section 3.1, then describe runtime state construction in Section 3.2, toolpath orchestration with online learning in Section 3.3, and runtime execution with reasoning workspace reconstruction in Section 3.4.

Task Definition. This paper studies the task of *time series reasoning* (Kauppinen & Hyvönen, 2007; Eichler & Didelez, 2012; Chow et al., 2024), which requires answering questions over time series data. Formally, given a time series $\mathbf{x} \in \mathbb{R}^{T \times V}$ and a question $q \in \mathcal{Q}$, the goal is to produce an answer $a^* \in \mathcal{A}$, where $\mathcal{A}$ is a discrete set for reasoning tasks and $\mathcal{A} = \mathbb{R}^H$ for forecasting with horizon H. In general, this problem can be formulated as $a^* = f(\mathbf{x}, q)$, where f captures both extracting relevant information from the time series and reasoning over it to produce the answer. Performance is measured by a task-appropriate loss $\ell : \mathcal{A}^2 \to \mathbb{R}_{\geq 0}$. We approximate f with a language model $\mathcal{M}$ operating on $\mathrm{tok}(\mathbf{x})$, producing $\hat{a} = \mathcal{M}(\mathrm{tok}(\mathbf{x}), q)$.

3.1 FORMULATING TIME SERIES REASONING AS A RUNTIME HARNESS PROBLEM

Problem Formulation. Current tool-augmented agents select and invoke tools to solve tasks (Kong et al., 2025). In time series reasoning, selecting analytical tools requires considering both the temporal properties relevant to the question and whether they appear in the input series. These tools differ in their analytical capabilities and output structures, requiring coordination of tool selection, execution, and output organization (Cheng et al., 2026). These requirements motivate a runtime harness that connects task and data analysis with tool configuration and evidence preparation for reasoning.

We formulate the design of this runtime harness as an optimization problem for time series reasoning. Given a language model $\mathcal{M}$, a statistical tool library $\mathcal{T}$, and a procedural memory $\mathcal{W}_{\mathrm{mem}}$, the harness $\mathcal{H}$ governs the process from task analysis to tool execution and final answer generation:

$$\hat{a}_{\mathcal{H}} = \mathcal{H}\big[\mathcal{M}, \mathcal{T}, \mathcal{W}_{\mathrm{mem}}\big](\mathbf{x}, q), \tag{1}$$

where $\hat{a}_{\mathcal{H}}$ denotes the answer produced under $\mathcal{H}$. Let $C_{\mathcal{H}}(\mathbf{x}, q)$ denote the runtime cost incurred by the harness. We seek a harness that balances answer quality and runtime cost:

$$\mathcal{H}^* = \arg\min_{\mathcal{H}} \mathbb{E}_{(\mathbf{x},q,a^*)\sim\mathcal{D}} \left[\ell(\hat{a}_{\mathcal{H}}, a^*) + \eta\, C_{\mathcal{H}}(\mathbf{x}, q)\right], \tag{2}$$

where $\mathcal{D}$ is the task distribution and $\eta \geq 0$ controls the penalty on runtime cost.

Runtime Control through Disassembly. Time series reasoning requires extracting temporal structure and interpreting it to answer a question (Chang et al., 2025). Analytical tools support this extraction by computing temporal statistics directly from raw series. We therefore separate inference into tool-based abstraction and evidence-based model reasoning, connected through an explicit interface (Khattab et al., 2023). The harness controls abstraction by executing selected tool configurations and organizing their outputs into evidence:

$$f(\mathbf{x}, q) \approx \mathcal{M}\Big(q, \underbrace{\mathcal{F}_{\mathrm{obs}}\big(\mathcal{O}_{\mathcal{H}}\big)}_{\mathcal{V}\text{: temporal abstraction}}\Big). \tag{3}$$

where $\mathcal{O}_{\mathcal{H}}$ denotes the observations returned by executing the selected configurations $w_1^*, \ldots, w_K^*$ on $\mathbf{x}$. The observation interface organizes these observations into evidence $\mathcal{V}$. AgenTS constructs a runtime state from task and data characteristics, selects configurations using a criterion with a cost penalty to pursue the objective in Eq. 2, and aggregates outputs through a common evidence interface.

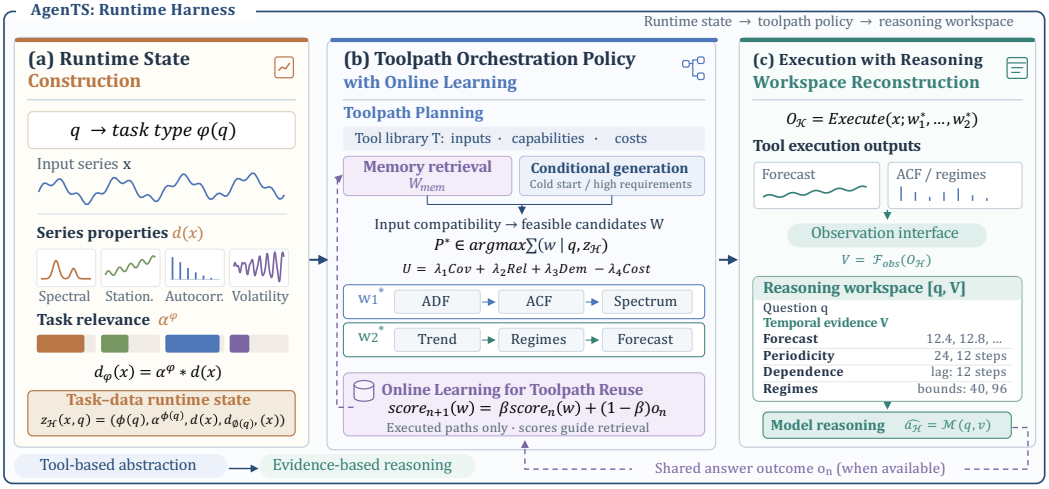


Figure 1: Overview of AgenTS. (a) The *Profiler* constructs a runtime state from the relevance of temporal properties to the question and their intensity in the input series. (b) The *Orchestrator* plans toolpaths with a cost-aware utility and reuses toolpaths stored in an online learning memory. (c) The *Synthesizer* executes the selected toolpaths and organizes their outputs into a reasoning workspace.

Tool Library. For the tool-based abstraction stage, the harness accesses a statistical tool library $\mathcal{T} = t_1, \ldots, t_{|\mathcal{T}|}$. Each tool computes temporal statistics from compatible raw series and returns an analytical output in its native format. Let $\mathcal{C}$ denote the set of capability tags. Each tool $t \in \mathcal{T}$ has input requirements, a capability set $\mathcal{C}t \subseteq \mathcal{C}$, and a computational cost $c_t \in \mathbb{R}+$. These specifications support compatibility filtering, capability matching, and cost evaluation during runtime orchestration. A *tool configuration* $w = (t_{i_1}, \ldots, t_{i_L})$, where $t_{i_j} \in \mathcal{T}$ and $L \leq L_{\max}$, specifies the selected tools and their execution order for an instance. The full tool registry is provided in Appendix D.

3.2 Time Series Agent Runtime State Construction

Following the disassembly above, controlling tool-based abstraction requires characterizing the analytical requirements of each instance. Existing time series agents interleave reasoning with tool execution, selecting tools based on the question and accumulated observations (Liu et al., 2025; Wu et al., 2026). However, tool orchestration remains coupled with model reasoning, complicating independent runtime control over the abstraction stage. Inspired by explicit state modeling in agent runtimes (Wang et al., 2025a), we construct a per-instance runtime state that represents these two factors for tool orchestration. Given $(\mathbf{x}, q)$, we propose the following runtime state representation:

$$\mathbf{z}_{\mathcal{H}}(\mathbf{x}, q) = \left(\phi(q), \boldsymbol{\alpha}^{\phi(q)}, \mathbf{d}(\mathbf{x}), d_{\phi(q)}(\mathbf{x})\right). \tag{4}$$

where $\phi(q)$ identifies the task type, $\boldsymbol{\alpha}^{\phi(q)}$ characterizes the relevance of different temporal properties to the task, and $\mathbf{d}(\mathbf{x})$ measures how strongly these properties are present in the input series. The scalar $d_{\phi(q)}(\mathbf{x})$ quantifies the degree to which the series exhibits properties relevant to the task.

Relevance and Intensity Estimation. To construct the runtime state in Eq. 4, the harness uses a profiling component, the *Profiler*, to characterize task relevance and input series properties. Given q, the Profiler identifies $\phi = \phi(q)$ and uses a relevance vector $\boldsymbol{\alpha}^{\phi} \in [0,1]^M$ shared across instances of that task type. It also probes $\mathbf{x}$ to obtain $\mathbf{d}(\mathbf{x}) \in [0,1]^M$. Each α_j^{ϕ} measures the task relevance of dimension j, while $d_j(\mathbf{x})$ measures its strength in the series. Their inner product, $d_{\phi}(\mathbf{x}) = \boldsymbol{\alpha}^{\phi} \cdot \mathbf{d}(\mathbf{x}) \in [0,1]$, provides an instance-level proxy for analytical requirements. This scalar determines the requirement bucket for configuration retrieval and whether additional candidates are generated, while the full vectors support dimension-specific capability matching. Full definitions are provided in Appendix C.1.

3.3 TIME SERIES TOOLPATH ORCHESTRATION POLICY WITH ONLINE LEARNING

A key challenge for the runtime harness in time series reasoning is selecting appropriate tools while maintaining efficiency. Existing work searches for execution structures using performance feedback (Zhang et al., 2025). However, the tool suitability and execution cost of these structures can vary across instances. To address this, our orchestration policy balances tool suitability and runtime cost through two mechanisms. The policy uses *toolpath planning* to select analytical tools efficiently based on the runtime state. Furthermore, repeatedly constructing similar toolpath across instances introduces planning overhead. Inspired by reusable agent memory (Gupta et al., 2025), the policy also uses *online learning* to store and reuse successful toolpaths, reducing repeated planning.

Toolpath Planning. Given the runtime state, the harness applies an orchestration policy to determine which analytical tools to execute and their order. We propose *toolpath planning*, in which the *Orchestrator* selects tool sequences by balancing analytical suitability and execution cost. Specifically, the Orchestrator uses $\mathbf{z}_{\mathcal{H}}(\mathbf{x}, q)$ to construct candidate sequences of analytical tools and filters them for input compatibility, yielding the feasible candidate set $\mathcal{W}$. It evaluates the feasible candidates using a utility that combines analytical suitability with estimated execution cost. We formulate planning as selecting $K \leq |\mathcal{W}|$ toolpaths that maximize utility:

$$\mathcal{P}^* \in \arg \max_{\substack{\mathcal{P} \subseteq \mathcal{W} \\ |\mathcal{P}|=K}} \sum_{w \in \mathcal{P}} U(w \mid q, \mathbf{z}_{\mathcal{H}}). \tag{5}$$

To evaluate each toolpath, we combine three proxies for suitability with an execution cost penalty:

$$\begin{aligned} U(w \mid q, \mathbf{z}_{\mathcal{H}}) = \lambda_1 \operatorname{Cov}(w, \phi) + \lambda_2 \operatorname{Rel}(w, q) \\ + \lambda_3 \operatorname{Dem}(w, \boldsymbol{\alpha}^{\phi}, \mathbf{d}(\mathbf{x})) - \lambda_4 \operatorname{Cost}(w). \end{aligned} \tag{6}$$

Here, Cov measures coverage of task-required statistical capabilities, while Rel measures the semantic similarity between tool descriptions and q in a sentence-embedding space. Dem measures capability coverage across temporal dimensions, weighted by $\alpha_j^{\phi} d_j(\mathbf{x})$ to emphasize properties relevant to the task and present in the series. Cost estimates tool execution cost. The Orchestrator solves Eq. 5 by ranking candidates under U and selecting the K highest-scoring paths. Their ordered tool sequences are passed to the *Synthesizer* for execution (§3.4). Full definitions are provided in Appendix C.2.

Online Learning for Toolpath Reuse. Inspired by online learning of reusable tool sequences (Gupta et al., 2025), we introduce a memory mechanism that refines toolpath quality estimates from execution outcomes to support subsequent planning. The mechanism maintains a dynamic memory (Sumers et al., 2024; Wang et al., 2024a), with buckets $\mathcal{W}_{\text{mem}}^{(\phi,b)}$ that store toolpaths and their quality estimates. Each bucket is indexed by task type ϕ and requirement level b, determined by $d_{\phi}(\mathbf{x})$.

For an instance $(\mathbf{x}_n, q_n)$, let $\mathcal{W}_{\text{mem},n}$ denote the available memory and $\mathbf{z}_n = \mathbf{z}_{\mathcal{H}}(\mathbf{x}_n, q_n)$ the constructed runtime state. We formulate planning and online memory adaptation as:

$$\mathcal{P}_n^* = \pi_{\mathcal{H}}\big(\mathbf{x}_n, q_n; \mathcal{W}_{\text{mem},n}\big), \qquad \mathcal{W}_{\text{mem},n+1} = \operatorname{Update}_{\text{mem}}\big(\mathcal{W}_{\text{mem},n}, \mathbf{z}_n, \mathcal{E}_n\big). \tag{7}$$

where $\pi_{\mathcal{H}}$ denotes the orchestration policy, $\mathcal{P}_n^*$ contains the selected toolpaths, and $\mathcal{E}_n = \{(w, o_n) : w \in \mathcal{P}_n^*\}$ pairs each selected toolpath with the shared instance-level outcome o_n obtained from the final answer. This feedback provides credit without isolating each toolpath's individual contribution.

At deployment, the memory is empty. For the first N_{warm} instances in each bucket, the Orchestrator draws candidates from LLM-generated toolpaths $\mathcal{W}_{\text{gen}}$ and builds the initial memory from their execution outcomes. Within $\operatorname{Update}_{\text{mem}}$, the *Synthesizer* refines the quality estimate of each executed toolpath using its outcome $o_n(w) \in [0, 1]$ through an exponential moving average:

$$\operatorname{score}_{n+1}(w) = \begin{cases} \beta \operatorname{score}_n(w) + (1 - \beta)\, o_n, & w \in \mathcal{P}_n^*, \\ \operatorname{score}_n(w), & \text{otherwise.} \end{cases} \tag{8}$$

where β controls the weight assigned to past outcomes. For subsequent instances, the scores determine retrieval priorities within the matching memory bucket. When $d_{\phi}(\mathbf{x}) > \theta$, the Orchestrator supplements retrieved toolpaths with generated candidates. The combined candidates undergo input compatibility filtering and are evaluated using the utility in Eq. 6. This process connects execution feedback to future toolpath selection, supporting reuse while reducing repeated generation. The EMA rate, outcome mapping, and memory policy are provided in Appendix C.3.

3.4 TIME SERIES RUNTIME EXECUTION WITH REASONING WORKSPACE RECONSTRUCTION

After toolpath selection, the runtime harness executes the toolpaths and prepares their outputs for model reasoning. Existing agent frameworks format tool outputs and pass them to the model for reasoning (Xiang et al., 2026). However, accumulating tool outputs can lead to long contexts that increase reasoning overhead. Motivated by workspace reconstruction (Chen et al., 2026), our design separates analytical evidence from the execution history. We therefore propose *reasoning workspace reconstruction*, where the *Synthesizer* executes toolpaths and structures outputs for model reasoning.

Reasoning Workspace Reconstruction. The Synthesizer first executes the selected toolpaths, obtaining $\mathcal{O}_{\mathcal{H}} = \mathrm{Execute}(\mathbf{x}; w_1^*, \ldots, w_K^*)$. For example, forecasts report values for future time steps, while autocorrelation peaks describe lagged dependence and regime boundaries locate transitions in the observed series. An observation interface $\mathcal{F}_{\mathrm{obs}}$ then applies tool-specific formatters to organize these outputs into structured evidence $\mathcal{V} = \mathcal{F}_{\mathrm{obs}}(\mathcal{O}_{\mathcal{H}})$. The formatters preserve the numerical results and their associated time or lag information within a common representation. The resulting workspace combines the question q with evidence $\mathcal{V}$ for final reasoning. We formalize our proposed workspace reconstruction through the following comparison with the full interaction history:

$$\underbrace{\big[q, (r_1, u_1, o_1), \ldots, (r_N, u_N, o_N)\big]}_{\text{Accumulated reasoning and execution history}} \quad \text{vs.} \quad \underbrace{\big[q, \mathcal{F}_{\mathrm{obs}}(\mathcal{O}_{\mathcal{H}})\big]}_{\text{AgenTS: structured temporal evidence}} . \tag{9}$$

Here, r_n, u_n, and o_n denote the reasoning step, tool call, and returned observation at step n. The accumulated history grows with the number of reasoning steps and mixes intermediate reasoning with tool outputs in their native formats, including redundant or failed calls, whereas the reconstructed workspace depends only on the selected toolpaths and follows a common representation. In constructing $\mathcal{V}$, tools shared across the K toolpaths are executed once, regime boundaries detected by different toolpaths are merged, and failed tools are recorded explicitly rather than dropped, so that the model knows which results are missing. The model $\mathcal{M}$ then reasons using q and $\mathcal{V}$, completing the separation of tool-based abstraction from model reasoning.

4 EXPERIMENTS

We conduct experiments on TSR-Suite (Guan et al., 2025) and MTBench (Chen et al., 2025), covering 14 datasets and 10 reasoning tasks, to evaluate AgenTS against tool-augmented baselines.

Benchmark. We evaluate on **TSR-Suite** (Guan et al., 2025), comprising 23,605 QA pairs across four tasks (scenario understanding, causality discovery, event-aware forecasting, and decision making) spanning 10 domains, each split into in-distribution (ID) and out-of-distribution (OOD) settings. The OOD split shifts the data source within each task, such as river discharge from different geographic regions for causality discovery and electricity load instead of taxi pickups for event-aware forecasting, and thus tests generalization beyond the training distribution. We further evaluate on **MTBench** (Chen et al., 2025), a multimodal benchmark of 22,000 paired time series and textual narratives in finance and weather. The finance domain pairs 20,000 articles from professional financial news websites with the corresponding stock price series, and the weather domain aligns hourly records from 50 U.S. weather stations with 2,000 severe weather event reports. Each domain includes three tasks: Short QA, Long QA, and Forecasting. The two QA tasks are news-driven multiple-choice questions whose correct and incorrect statements are grounded in the news. The short and long settings differ in the input window, which spans 7 and 30 days in finance and 7 and 14 days in weather.

Evaluation Metrics. We report Success Rate (SR), defined as the proportion of valid, extractable outputs based on predefined regular expressions (Guan et al., 2025). To isolate time series reasoning from instruction following, all metrics are computed on valid outputs only. For question answering tasks, we report Accuracy (ACC) with exact match, while for Event-aware Forecasting we report Mean Absolute Error (MAE). Higher ACC and lower MAE indicate better performance.

Baselines. We compare against two categories of methods based on how they model time series and reasoning. *Time Series Language Models* adapt LLMs for direct time series understanding. Time-MQA (Kong et al., 2025) enhances contextual representations for multi-task TSQA, while

Table 1: Overall results on TSR-Suite. All metrics are computed only on valid responses. "–" denotes SR below 10%, where results are omitted for lack of statistical significance. **Bold**: the best, underline: the 2nd best. Full results including proprietary and open-source baselines are in Table 5 (Appendix).

Model	Scenario Understanding				Causality Discovery				Event-aware Forecasting				Decision Making			
	ID		OOD		ID		OOD		ID		OOD		ID		OOD	
	ACC	SR%	ACC	SR%	ACC	SR%	ACC	SR%	MAE	SR%	MAE	SR%	ACC	SR%	ACC	SR%
Time Series Language Models Baseline																
Time-MQA	15.1	21.5	27.8	22.1	8.4	50.2	4.0	52.2	–	0.2	–	0.0	5.4	36.1	10.0	47.3
ChatTS	–	6.0	–	6.9	18.2	30.1	18.6	26.7	–	0.0	–	0.0	5.8	27.1	11.1	27.1
Agentic Time Series Language Model																
Time-MQA	32.2	29.5	25.1	32.6	30.1	44.3	31.2	37.2	–	1.4	–	0.4	12.0	13.3	11.6	15.8
+ ReAct	13.9	75.5	15.1	73.0	20.7	60.5	10.5	62.0	26.55	34.0	155.5	20.5	11.5	73.9	11.5	65.0
+ TS-Reasoner	19.5	100.0	16.6	99.5	39.5	100.0	39.0	100.0	–	4.5	–	3.0	24.3	98.4	29.1	99.5
+ TS-Agent	13.2	60.5	19.8	65.5	15.0	56.5	11.0	59.0	28.74	22.0	173.7	20.0	6.5	49.5	4.8	52.0
+ AgenTS	**34.3**	92.5	**33.9**	91.5	**43.9**	92.0	**40.5**	93.5	**14.60**	100.0	**48.2**	100.0	**58.6**	90.9	**68.4**	88.0
Time-R1	30.9	94.0	34.0	92.5	30.2	53.8	31.4	48.9	17.61	38.7	–	6.3	27.8	95.7	32.2	93.1
+ ReAct	40.4	89.0	37.1	87.5	3.2	92.5	8.1	86.0	24.15	21.5	267.4	2.0	17.4	61.2	21.7	57.5
+ TS-Reasoner	33.5	100.0	37.0	100.0	33.0	100.0	32.0	100.0	17.88	31.5	198.9	3.5	66.5	100.0	78.5	100.0
+ TS-Agent	44.1	72.5	40.9	68.5	26.1	82.5	26.4	87.0	24.53	15.5	–	–	15.3	90.4	16.8	92.5
+ AgenTS	**48.2**	95.5	**44.7**	99.0	**40.1**	83.0	**39.1**	92.5	**14.60**	100.0	**47.1**	100.0	**68.1**	93.3	**86.3**	96.5
Agentic Time Series Reasoning Model																
TimeOmni-1	85.4	97.5	87.7	98.3	69.3	99.8	64.0	99.8	15.0	93.8	145.5	82.3	47.9	100.0	58.9	100.0
+ ReAct	86.9	99.0	84.8	99.0	71.7	99.0	62.7	96.5	15.12	25.5	–	–	28.2	100.0	38.6	98.5
+ TS-Reasoner	75.0	100.0	73.0	100.0	71.5	100.0	63.0	100.0	18.14	75.5	141.2	21.0	48.4	100.0	39.0	100.0
+ TS-Agent	66.7	67.5	80.0	70.0	49.7	82.5	61.8	86.5	15.3	42.0	–	–	15.9	73.4	30.3	94.0
+ AgenTS	**89.5**	100.0	**89.0**	100.0	**77.0**	99.8	**68.0**	100.0	**14.60**	100.0	**45.4**	100.0	**51.1**	100.0	**69.0**	100.0

ChatTS (Xie et al., 2025) aligns time series with LLMs via large-scale synthetic data. *Time Series Reasoning Models* explicitly optimize reasoning over temporal data. Time-R1 (Luo et al., 2025) uses reinforcement learning with multi-objective rewards for forecasting, and TimeOmni-1 (Guan et al., 2025) adopts a multi-stage training pipeline for complex reasoning across tasks. We also include proprietary and open-source LLMs; full results are in Appendix A.

Additionally, we include three agent frameworks to further enrich our evaluation:

- **ReAct** (Yao et al., 2023): interleaves reasoning and tool use within a thought–action–observation loop. At each step, the model selects the next tool from the question and the observations collected so far, and each tool output is appended to its context.
- **TS-Reasoner** (Ye et al., 2026): integrates domain-specific time series tools with iterative error feedback. For each task type, it executes a fixed sequence of operators in a predefined order and retries the corresponding tool when a failure occurs.
- **TS-Agent** (Liu et al., 2025): incorporates a self-critique mechanism to refine multi-step reasoning over time series. After each tool call, the model checks whether the evidence collected so far is sufficient, and continues calling tools until it is ready to produce the final answer.

4.1 Main Results

Table 1 summarizes the results on TSR-Suite. Our method consistently outperforms both no-tool baselines and existing tool-augmented agents across all tasks. Decision making shows the largest gains, with up to a fivefold improvement on the weakest backbone, consistent with the advantage of tool selection over uniform tool use, while causality also improves substantially. The gains hold across backbones of different strength: weaker backbones benefit the most, and the strongest backbone, TimeOmni-1, still improves on every task. In contrast, existing agents do not always help, as ReAct and TS-Agent fall below the baseline on several tasks, indicating that tool use without control can hurt reasoning. For forecasting, many baselines fail to produce valid outputs, whereas AgenTS maintains a 100% success rate across all backbones with the best performance in each model group.

As shown in Table 2, AgenTS outperforms existing baselines across both Finance and Weather domains and on most task types. For QA tasks, AgenTS achieves the highest accuracy in nearly all settings, with particularly notable gains on Long QA and Short QA, indicating stronger capability in reasoning over both short- and long-horizon contexts. The only exception is Weather Short QA with TimeOmni-1, where ReAct achieves higher accuracy. For forecasting, AgenTS yields the lowest

Table 2: Overall results on MTBench. Models evaluated on Finance and Weather domains, each with Long QA, Short QA, and Forecasting. ACC for QA tasks, MAE for Forecasting; SR is the proportion of valid outputs. "–" denotes SR below 10%. **Bold**: best within each model group.

Model	Finance						Weather					
	Long QA		Short QA		Forecasting		Long QA		Short QA		Forecasting	
	ACC	SR%	ACC	SR%	MAE↓	SR%	ACC	SR%	ACC	SR%	MAE↓	SR%
Agentic Time Series Language Models												
Time-MQA	29.0	62.0	31.1	61.0	3.85	80.5	43.0	55.0	24.0	54.0	18.81	92.0
+ ReAct	58.6	70.0	14.7	75.0	5.00	95.0	45.0	70.0	39.0	70.0	18.00	95.0
+ TS-Reasoner	42.0	100.0	34.0	60.0	4.60	97.0	42.0	50.0	32.8	48.0	17.81	100.0
+ TS-Agent	53.8	87.0	19.7	66.0	5.00	93.0	37.0	65.0	40.0	65.0	18.05	93.0
+ AgenTS	**62.5**	77.0	**42.9**	76.0	**3.84**	100.0	**47.0**	85.0	**47.3**	84.0	**16.67**	95.5
Time-R1	75.0	20.0	74.4	60.0	14.61	62.5	45.0	10.0	44.5	78.0	17.69	78.0
+ ReAct	65.5	87.0	68.3	82.0	4.62	70.5	50.0	90.0	49.0	90.0	18.85	80.0
+ TS-Reasoner	74.0	100.0	78.0	100.0	–	–	55.0	100.0	53.2	100.0	17.38	92.0
+ TS-Agent	65.3	98.0	59.1	93.0	4.00	85.0	33.0	85.0	30.0	85.0	18.58	87.0
+ AgenTS	**86.7**	90.0	**85.0**	86.0	**3.84**	100.0	**63.0**	95.0	**62.5**	86.58	**16.00**	84.5
Agentic Time Series Reasoning Models												
TimeOmni-1	69.4	98.0	68.0	100.0	7.13	72.0	43.0	100.0	42.0	100.0	13.67	38.0
+ ReAct	61.8	89.0	57.8	90.0	4.41	83.5	30.0	86.0	78.0	86.0	13.77	88.5
+ TS-Reasoner	45.0	100.0	58.0	100.0	5.13	–	25.0	76.0	31.8	100.0	14.61	96.0
+ TS-Agent	33.8	65.0	45.5	66.0	6.30	71.5	37.0	85.0	23.2	86.2	14.08	92.0
+ AgenTS	**74.0**	99.0	**81.0**	98.2	**3.84**	100.0	**63.4**	88.0	**56.7**	94.5	**10.90**	96.0

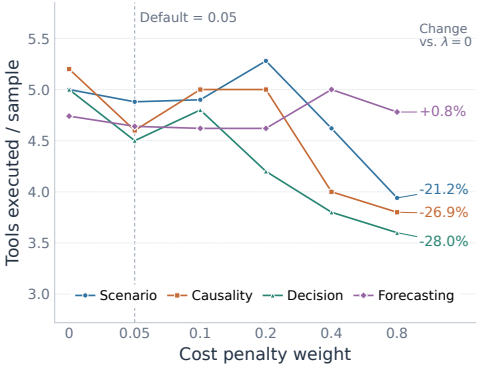


Figure 2: Effect of the cost penalty weight λ_4 on tool usage across four tasks.

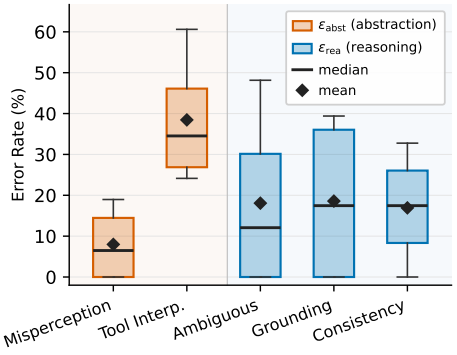


Figure 3: Distribution of AgenTS error types across four time series tasks.

MAE across most configurations, reflecting more precise numerical prediction, and maintains a 100% success rate on Finance forecasting for all backbones. In contrast, existing agents often fall below their tool-free backbones; for example, all three agent baselines reduce the Finance Long QA accuracy of TimeOmni-1, whereas AgenTS improves it.

4.2 Further Analysis

Sensitivity to the Cost Penalty. Figure 2 varies the cost penalty weight λ_4 in Eq. 6 and reports the number of tools executed per sample. For small penalties, tool usage remains relatively stable across tasks. As λ_4 increases further, the *Orchestrator* favors shorter toolpaths, reducing tool usage by up to 28% on scenario understanding, causality discovery, and decision making. In contrast, tool usage on forecasting remains nearly unchanged, since its toolpaths are dominated by forecasting tools that remain necessary under larger penalties. The default setting $\lambda_4 = 0.05$ keeps tool usage close to the non-penalized setting, while larger values provide direct control over tool usage at runtime.

Error Analysis. To better understand the sources of failure, we manually analyze 400 AgenTS errors using the TimeOmni-1 in TSR-Suite, and categorize them into five groups spanning abstraction and reasoning, including misperception, tool interpretation, ambiguous evidence, grounding, and

Table 3: Ablation study on TimeOmni-1. Each row removes one component while keeping the others unchanged. All metrics are computed on valid outputs. "w/o" denotes removing the corresponding component.

Variant	Scenario ACC↑	Causality ACC↑	Forecasting MAE↓	Decision ACC↑
(a) AgenTS (Full)	**89.5**	**77.0**	**14.6**	**51.1**
(b) w/o Property Intensity	85.0	72.5	15.1	48.3
(c) w/o Toolpath Planning	82.3	65.8	15.6	46.8
(d) w/o Online Learning	82.9	73.3	16.1	44.2
(e) w/o Workspace Reconstruction	81.1	69.1	17.3	42.6
(f) w/o Runtime Execution	84.5	73.0	15.3	45.2

Table 4: Efficiency-accuracy tradeoff on TSR-Suite. Avg ACC averaged over three MCQ tasks; Tool Calls denotes the per-sample averages. **Tool SR**: success rate of tool invocations. **Bold**: best, underline: second-best.

Metric	ReAct	TS-Reasoner	TS-Agent	Ours	Δ *vs. ReAct*
Tool Calls↓	3.17	2.51	6.81	**1.69**	−47%
Time (s)↓	80.8	**16.2**	225.5	57.6	−29%
Tool SR (%)↑	99.3	**100.0**	73.71	**100.0**	+0.7%
Avg ACC (%)↑	62.3	65.0	44.1	**72.5**	+16.4%
Eff.↑	19.7	25.9	6.47	**42.9**	+118%

consistency. We further group these categories by their source: an abstraction error ($\varepsilon_{\mathrm{abst}}$) occurs when the temporal structure of the series is not correctly obtained, either by misreading the series or by misinterpreting tool outputs, while a reasoning error ($\varepsilon_{\mathrm{rea}}$) occurs when the model draws a wrong conclusion from correctly obtained evidence. Full details are provided in Appendix A.5. Figure 3 shows that AgenTS substantially reduces direct misperception of numerical features, with a mean rate of around 8%, indicating that tool-grounded abstraction mitigates the core abstraction bottleneck. In these cases, the model correctly invokes the tools but misinterprets their textual outputs, suggesting that the representational gap shifts from raw time series abstraction to evidence interpretation.

Ablation Study. Table 3 reports an ablation study of each harness component on TimeOmni-1 in the in-distribution setting. Removing any component degrades performance on all four tasks, confirming that each component is necessary (full details in Appendix A.3). Toolpath planning (c) is the most critical for causality discovery, with the largest drop of -11.2. Property intensity (b) highlights the need for instance-level adaptation, as replacing it with a uniform vector leads to consistent degradation across tasks. Online learning (d) mainly benefits decision making among the four tasks. Removing workspace reconstruction (e) yields the largest degradation on the other three tasks, and this variant even underperforms the variant without any tool execution (f), indicating that raw tool outputs without structured organization can hinder reasoning more than providing no tool evidence at all. Overall, the three stages of AgenTS show complementary effects across tasks.

Efficiency Analysis. Table 4 compares the efficiency of AgenTS with tool-augmented baselines on TSR-Suite. The reported time is measured end to end for each sample, covering all stages from runtime state construction to final answer generation. AgenTS invokes the fewest tools per sample while achieving the highest average accuracy, which yields the best accuracy per tool call. Compared with ReAct, it reduces tool calls by 47% without any tool failure. Agents that decide tool calls at every reasoning step incur substantial overhead, as TS-Agent requires the most tool calls and the longest inference time. TS-Reasoner runs fastest by executing a fixed tool sequence, but this lowers its accuracy. AgenTS takes longer than TS-Reasoner but remains faster than ReAct and TS-Agent, since it plans toolpaths once for each instance before execution. The advantage of AgenTS comes from its runtime state and online learning. Tools are invoked according to the relevance of temporal properties to the question and their intensity in the input series. In addition, toolpaths are planned from accumulated experience instead of being generated from scratch for each instance.

5 Conclusion

In this study, we introduce AgenTS, a runtime harness for time series reasoning that treats tool use as a runtime stage separate from model reasoning. AgenTS constructs a runtime state that couples the relevance and intensity of temporal properties, orchestrates toolpaths with accumulated experience through online learning, and executes the selected toolpaths with reasoning workspace reconstruction to separate the analytical state from the execution trace. Across TSR-Suite and MTBench, AgenTS consistently achieves state-of-the-art performance while invoking fewer tools, with an average gain of more than 30% on TSR-Suite and up to 74% in the finance domain of MTBench. We hope this work can motivate further study of runtime harness design in time series reasoning. Our future work may extend this harness to broader agentic and visual reasoning tasks, as well as more dynamic environments and improved memory mechanisms.

AI USE STATEMENT

(This section is **required** and does not count toward the page limit.)

In this work, we used generative AI tools to polish the writing, including checking grammar, clarity, and phrasing of the sentences. We have not used AI tools for developing research ideas, designing methods, conducting experiments, or analyzing results. We have reviewed and edited all AI-assisted text to ensure its accuracy and consistency with our findings. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Türkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda-Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the language of time series. *Trans. Mach. Learn. Res.*, 2024, 2024.

Ching Chang, Yidan Shi, Defu Cao, Wei Yang, Jeehyun Hwang, Haixin Wang, Jiacheng Pang, Wei Wang, Yan Liu, Wen-Chih Peng, et al. A survey of reasoning and agentic systems in time series with large language models. *arXiv preprint arXiv:2509.11575*, 2025.

Guoxin Chen, Zile Qiao, Xuanzhong Chen, Donglei Yu, Haotian Xu, Xin Zhao, Ruihua Song, Wenbiao Yin, Huifeng Yin, Liwen Zhang, et al. Iterresearch: Rethinking long-horizon agents with interaction scaling. In *International Conference on Learning Representations*, volume 2026, pp. 30353–30383, 2026.

Jialin Chen, Aosong Feng, Ziyu Zhao, Juan Garza, Gaukhar Nurbek, Cheng Qin, Ali Maatouk, Leandros Tassiulas, Yifeng Gao, and Rex Ying. Mtbench: A multimodal time series benchmark for temporal reasoning and question answering. *CoRR*, abs/2503.16858, 2025. doi: 10.48550/ARXIV.2503.16858. URL https://doi.org/10.48550/arXiv.2503.16858.

Mingyue Cheng, Jiahao Wang, Daoyu Wang, Xiaoyu Tao, Qi Liu, and Enhong Chen. Can slow-thinking llms reason over time? empirical studies in time series forecasting. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining, WSDM 2026, Boise, ID, USA, February 22-26, 2026*, pp. 99–110. ACM, 2026.

Winnie Chow, Lauren Gardiner, Haraldur Tómas Hallgrímsson, Maxwell A. Xu, and Shirley You Ren. Towards time series reasoning with llms. *CoRR*, abs/2409.11376, 2024. doi: 10.48550/ARXIV.2409.11376. URL https://doi.org/10.48550/arXiv.2409.11376.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 10148–10167. PMLR / OpenReview.net, 2024.

Huaming Du, Yujia Zheng, Baoyu Jing, Yu Zhao, Gang Kou, Guisong Liu, Tao Gu, Weimin Li, and Carl Yang. Causal discovery through synergizing large language model and data-driven reasoning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 543–554, 2025.

Michael Eichler and Vanessa Didelez. Causal reasoning in graphical time series models. *arXiv preprint arXiv:1206.5246*, 2012.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *CoRR*, abs/2504.11536, 2025. doi: 10.48550/ARXIV.2504.11536. URL https://doi.org/10.48550/arXiv.2504.11536.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. Large language models are zero-shot time series forecasters. 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.

Tong Guan, Zijie Meng, Dianqi Li, Shiyu Wang, Chao-Han Huck Yang, Qingsong Wen, Zuozhu Liu, Sabato Marco Siniscalchi, Ming Jin, and Shirui Pan. Timeomni-1: Incentivizing complex reasoning with time series in large language models. *arXiv preprint arXiv:2509.24803*, 2025.

Advait Gupta, Rishie Raj, Dang Nguyen, and Tianyi Zhou. Fastaˆ∗: Fast-slow toolpath agent with subroutine mining for efficient multi-turn image editing. *arXiv preprint arXiv:2506.20911*, 2025.

Erik Johannes Husom, Arda Goknil, Merve Astekin, Lwin Khin Shar, Andre KãJPYsen, Sagar Sen, Benedikt Andreas Mithassel, and Ahmet Soylu. Sustainable LLM inference for edge AI: evaluating quantized llms for energy efficiency, output accuracy, and inference latency. *ACM Trans. Internet Things*, 6(4):1–35, 2025.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-llm: Time series forecasting by reprogramming large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Tomi Kauppinen and Eero Hyvönen. Modeling and reasoning about changes in ontology time series. In *Ontologies: A handbook of principles, concepts and applications in information systems*, pp. 319–338. Springer, 2007.

Lars Kegel, Martin Hahmann, and Wolfgang Lehner. Generating what-if scenarios for time series data. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, pp. 1–12, 2017.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

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 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. 29736–29753. Association for Computational Linguistics, 2025.

Haohang Li, Yupeng Cao, Yangyang Yu, Shashidhar Reddy Javaji, Zhiyang Deng, Yueru He, Yuechen Jiang, Zining Zhu, Kp Subbalakshmi, Jimin Huang, et al. Investorbench: A benchmark for financial decision-making tasks with llm-based agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2509–2525, 2025.

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. *CoRR*, abs/2510.07432, 2025. doi: 10.48550/ARXIV.2510.07432. URL https://doi.org/10.48550/arXiv.2510.07432.

Carl Henning Lubba, Sarab S. Sethi, Philip Knaute, Simon R. Schultz, Ben D. Fulcher, and Nick S. Jones. catch22: Canonical time-series characteristics - selected through highly comparative time-series analysis. *Data Min. Knowl. Discov.*, 33(6):1821–1852, 2019.

Yucong Luo, Yitong Zhou, Mingyue Cheng, Jiahao Wang, Daoyu Wang, Tingyue Pan, and Jintao Zhang. Time series forecasting as reasoning: A slow-thinking approach with reinforced llms. *CoRR*, abs/2506.10630, 2025. doi: 10.48550/ARXIV.2506.10630. URL https://doi.org/10.48550/arXiv.2506.10630.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Xuying Ning, Katherine Tieu, Dongqi Fu, Tianxin Wei, Zihao Li, Yuanchen Bei, Jiaru Zou, Mengting Ai, Zhining Liu, Ting-Wei Li, et al. Code as agent harness. *arXiv preprint arXiv:2605.18747*, 2026.

Zijie Pan, Yushan Jiang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. S2IP-LLM: semantic space informed prompt learning with LLM for time series forecasting. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 39135–39153. PMLR / OpenReview.net, 2024.

Felix Parker, Nimeesha Chan, Chi Zhang, and Kimia Ghobadi. Augmenting llms for general time series understanding and prediction. *CoRR*, abs/2510.01111, 2025. doi: 10.48550/ARXIV.2510.01111. URL https://doi.org/10.48550/arXiv.2510.01111.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

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.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. 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.

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.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents. *Trans. Mach. Learn. Res.*, 2024, 2024.

Chenxi Sun, Hongyan Li, Yaliang Li, and Shenda Hong. TEST: text prototype aligned embedding to activate llm's ability for time series. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Xiaoyu Tao, Mingyue Cheng, Chuang Jiang, Tian Gao, Huanjian Zhang, and Yaguo Liu. Cast-r1: Learning tool-augmented sequential decision policies for time series forecasting. *CoRR*, abs/2602.13802, 2026a. doi: 10.48550/ARXIV.2602.13802. URL https://doi.org/10.48550/arXiv.2602.13802.

Xiaoyu Tao, Yuchong Wu, Mingyue Cheng, Ze Guo, and Tian Gao. Anomamind: Agentic time series anomaly detection with tool-augmented reasoning. *CoRR*, abs/2602.13807, 2026b. doi: 10.48550/ARXIV.2602.13807. URL https://doi.org/10.48550/arXiv.2602.13807.

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. *Trans. Mach. Learn. Res.*, 2024, 2024a.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers Comput. Sci.*, 18(6):186345, 2024b.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. In *International Conference on Learning Representations*, volume 2025, pp. 65882–65919, 2025a.

Yilin Wang, Peixuan Lei, Jie Song, Yuzhe Hao, Tao Chen, Yuxuan Zhang, Lei Jia, Yuanxiang Li, and Zhongyu Wei. Itformer: Bridging time series and natural language for multi-modal QA with large-scale multitask dataset. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025b.

Ziyu Wang, Hao Li, Di Huang, Hye-Sung Kim, Chae-Won Shin, and Amir M Rahmani. Healthq: Unveiling questioning capabilities of llm chains in healthcare conversations. *Smart Health*, 36: 100570, 2025c.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 53140–53164. PMLR / OpenReview.net, 2024.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 22419–22430, 2021.

Xingjian Wu, Junkai Lu, Zhe Li, Xiangfei Qiu, Jilin Hu, Chenjuan Guo, Christian S. Jensen, and Bin Yang. Timeart: Towards agentic time series reasoning via tool-augmentation. *CoRR*, abs/2601.13653, 2026. doi: 10.48550/ARXIV.2601.13653. URL https://doi.org/10.48550/arXiv.2601.13653.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025.

Keyi Xiang, Zeyu Feng, Zhuoyi Lin, Yueming Lyu, Boyuan Shi, Yew-Soon Ong, Ivor Tsang, and Haiyan Yin. Flowsearcher: Synthesizing memory-guided agentic workflows for web information seeking. In *International Conference on Learning Representations*, volume 2026, pp. 149189–149217, 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. *Proc. VLDB Endow.*, 18(8):2385–2398, 2025.

Hao Xue and Flora D. Salim. Promptcast: A new prompt-based learning paradigm for time series forecasting. *IEEE Trans. Knowl. Data Eng.*, 36(11):6851–6864, 2024.

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.

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. *Trans. Mach. Learn. Res.*, 2026, 2026.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. In *International Conference on Learning Representations*, volume 2025, pp. 34040–34077, 2025.

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. *CoRR*, abs/2510.01538, 2025. doi: 10.48550/ARXIV.2510.01538. URL https://doi.org/10.48550/arXiv.2510.01538.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 11106–11115. AAAI Press, 2021.

Jiahui Zhou, Dan Li, Boxin Li, Xiao Zhang, Erli Meng, Lin Li, Zhuomin Chen, Jian Lou, and See-Kiong Ng. Time series reasoning via process-verifiable thinking data synthesis and scheduling for tailored llm reasoning. *arXiv preprint arXiv:2602.07830*, 2026.

A Extended Main Result

A.1 Full Baseline Comparison

Table 5 extends Table 1 by incorporating proprietary models, open-source baselines, and time series–specialized language models. Overall, AgenTS achieves the best performance on 7 out of 8 in-distribution (ID) and out-of-distribution (OOD) metrics across backbones, and AgenTS+TimeOmni-1 shows particularly large gains over GPT-4.1 in causality discovery (77.0 vs. 28.7 ID) and decision making (51.1 vs. 25.5 ID). The improvements are also consistent across smaller backbones. For instance, Time-MQA (Llama3-8B) shows nearly a fivefold increase in decision making performance and a 46% improvement in causality discovery when augmented with AgenTS, indicating strong compatibility between the orchestration framework and lightweight models. In forecasting, AgenTS also achieves the lowest mean absolute error (MAE) among open-source agent-based methods (14.60 ID and 45.4 OOD), demonstrating that tool-grounded numerical abstraction transfers effectively to quantitative prediction tasks. Importantly, OOD performance closely follows ID trends, suggesting that runtime state construction and toolpath orchestration generalize beyond in-distribution datasets.

A.2 Efficiency Performance

Table 6 reports the efficiency–accuracy trade-off on TSR-Suite (TimeOmni-1, OOD). AgenTS requires 2.25 tool calls per sample, the lowest among all agent-based baselines, while achieving the highest average accuracy of 75.3% across causality, scenario, and decision-making tasks. Compared with the closest agentic baseline, ReAct, AgenTS reduces tool usage by 39%, speeds up inference by 41%, and improves accuracy by 21.5 percentage points, resulting in nearly 2$\times$ higher efficiency (33.5 vs. 16.8). Although TS-Reasoner is the fastest method at 19 seconds per sample, it sacrifices accuracy (58.3%) due to its fixed pipeline design. Overall, these results demonstrate that cost-aware toolpath planning in AgenTS achieves a better balance between accuracy and computational cost than both stepwise and fixed tool-use approaches.

A.3 Full Ablation Study

Each row disables exactly one pipeline component while holding the rest at their default settings:

- **(b) w/o Property Intensity.** Replaces the property intensity vector $\mathbf{d}$ estimated by the *Profiler* with a uniform vector (d_j=0.5 for every j), so toolpath scoring no longer reflects the instance-level intensity of temporal properties.
- **(c) w/o Toolpath Planning.** Bypasses the cost-aware utility (Eq. 6) and selects the top-K toolpaths uniformly at random from the *admissible* pool (after the same input-compatibility filter used by the full harness).
- **(d) w/o Online Learning.** Disables the online learning memory entirely (no read, no write), so $\mathcal{W}_{\text{mem}}^{(\phi,b)}$ is unavailable to the candidate set and the score update (Eq. 8) is never invoked.
- **(e) w/o Runtime Execution.** No toolpath is executed; the model receives an empty-evidence placeholder (no tool outputs) and answers from the question alone.

All other hyperparameters (sample count N, top-K, temperature schedule) match the main run. Full results are reported in Table 7.

A.4 Online Learning Improves with Accumulated Experience

Figure 4 shows rolling performance over the first 200 tasks on TimeOmni-1. Initially, both variants perform similarly while toolpaths are accumulated. After this warm-up phase, AgenTS with online learning outperforms the variant without it across all tasks, with gains most pronounced in decision making and forecasting, followed by steady improvements in causality discovery and smaller gains in scenario understanding. This pattern shows that the value of online learning emerges as instances accumulate, when successful toolpaths in each requirement bucket become reliable enough to replace newly generated ones, which also reduces repeated toolpath generation.

Table 5: Comprehensive comparison on four tasks (full version of Table 1 with proprietary, open-source, and additional time-series language-model baselines). ACC is reported for Tasks 1, 2, 4 and MAE for Task 3, together with SR under ID and OOD testbeds. All metrics are computed only on valid responses. "–" denotes SR below 10%, where results are omitted for lack of statistical significance. **Bold**: the best, underline: the 2nd best. Baseline numbers for Time-MQA (Llama3-8B), Time-R1 and TimeOmni-1 are reported as the "Baseline" rows of their respective tool-augmentation.

Model	Base LLM	Scenario Understanding				Causality Discovery				Event-aware Forecasting				Decision Making			
		ID		OOD		ID		OOD		ID		OOD		ID		OOD	
		ACC	SR%	ACC	SR%	ACC	SR%	ACC	SR%	MAE	SR%	MAE	SR%	ACC	SR%	ACC	SR%
Proprietary Models																	
GPT-4.1	2025-04-14	85.5	100.0	87.8	100.0	28.7	99.9	35.9	100.0	**13.79**	97.4	170.7	76.1	25.5	100.0	27.8	100.0
GPT-4.1	Nano	66.2	97.5	62.6	98.7	29.8	98.6	28.0	98.4	18.98	92.8	170.7	76.1	28.9	99.5	34.1	97.8
Open-Source Models																	
Llama-3.1	70B-Instruct	56.4	100.0	59.6	100.0	23.4	100.0	28.9	99.9	24.67	92.8	238.9	97.0	20.3	96.8	17.7	97.4
Mistral-Small-3.1	24B-Ins	64.8	100.0	69.2	100.0	24.6	100.0	25.8	100.0	17.28	72.0	120.9	43.4	44.7	100.0	42.9	100.0
Llama-3.1	8B-Instruct	36.6	46.5	32.1	46.8	–	3.7	–	1.9	27.68	52.9	186.8	29.8	7.4	28.7	16.2	42.9
Mistral	7B-v0.3	40.5	92.2	34.7	87.6	29.0	86.0	26.9	82.6	–	5.3	–	0.0	24.3	94.2	16.7	96.7
Qwen2.5	Instruct-7B	48.5	100.0	42.8	100.0	21.6	99.8	26.3	100.0	23.28	53.1	146.1	55.5	25.5	100.0	24.9	100.0
Time Series Language Models																	
Time-MQA	Mistral-7B-v0.3	15.1	21.5	27.8	22.1	8.4	50.2	4.0	52.2	–	0.2	–	0.0	5.4	36.1	10.0	47.3
Time-MQA	Qwen2.5-7B	25.0	14.0	37.5	22.7	29.5	33.0	30.5	32.0	19.76	12.2	–	6.5	23.8	58.0	26.4	44.3
ChatTS	–	–	6.0	–	6.9	18.2	30.1	18.6	26.7	–	0.0	–	0.0	5.8	27.1	11.1	27.1
Time-MQA (Llama3-8B) + Agentic Methods																	
Baseline		32.2	29.5	25.1	32.6	30.1	44.3	31.2	37.2	–	1.4	–	0.4	12.0	13.3	11.6	15.8
+ ReAct		13.9	75.5	15.1	73.0	20.7	60.5	10.5	62.0	26.55	34.0	155.5	20.5	11.5	73.9	11.5	65.0
+ TS-Reasoner		19.5	100.0	16.6	99.5	39.5	100.0	39.0	100.0	–	4.5	–	3.0	24.3	98.4	29.1	99.5
+ TS-Agent		13.2	60.5	19.8	65.5	15.0	56.5	11.0	59.0	28.74	22.0	173.7	20.0	6.5	49.5	4.8	52.0
+ Ours		34.3	92.5	33.9	91.5	43.9	92.0	40.5	93.5	14.60	100.0	**48.2**	100.0	58.6	90.9	68.4	88.0
Time-R1 (Qwen2.5-7B) + Agentic Methods																	
Baseline		30.9	94.0	34.0	92.5	30.2	53.8	31.4	48.9	17.61	38.7	–	6.3	27.8	95.7	32.2	93.1
+ ReAct		40.4	89.0	37.1	87.5	3.2	92.5	8.1	86.0	24.15	21.5	267.4	2.0	17.4	61.2	21.7	57.5
+ TS-Reasoner		33.5	100.0	37.0	100.0	33.0	100.0	32.0	100.0	17.88	31.5	198.9	3.5	66.5	100.0	78.5	100.0
+ TS-Agent		44.1	72.5	40.9	68.5	26.1	82.5	26.4	87.0	24.53	15.5	–	–	15.3	90.4	16.8	92.5
+ Ours		48.2	95.5	44.7	99.0	40.1	83.0	39.1	92.5	14.60	100.0	**47.1**	100.0	**68.1**	93.3	**86.3**	96.5
TimeOmni-1 (Qwen2.5-7B) + Agentic Methods																	
Baseline		85.4	97.5	87.7	98.3	69.3	99.8	64.0	99.8	15.0	93.8	145.5	82.3	47.9	100.0	58.9	100.0
+ ReAct		86.9	99.0	84.8	99.0	71.7	99.0	62.7	96.5	15.12	25.5	–	–	28.2	100.0	38.6	98.5
+ TS-Reasoner		75.0	100.0	73.0	100.0	71.5	100.0	63.0	100.0	18.14	75.5	141.2	21.0	48.4	100.0	39.0	100.0
+ TS-Agent		66.7	67.5	80.0	70.0	49.7	82.5	61.8	86.5	15.3	42.0	–	–	15.9	73.4	30.3	94.0
+ Ours		**89.5**	100.0	**89.0**	100.0	**77.0**	99.8	**68.0**	100.0	14.60	100.0	**45.4**	100.0	51.1	100.0	69.0	100.0

Table 6: Efficiency-accuracy tradeoff on TSR-Suite (TimeOmni-1, **OOD**). Avg ACC averaged over three MCQ tasks (causality, scenario, decision); Tool Calls and Time are per-sample averages. **Tool SR**: success rate of tool invocations. **Eff.** = Avg ACC / Tool Calls (accuracy per tool invocation). **Bold**: best, underline: second-best. "–" denotes methods whose tool invocation collapsed on OOD (< 0.01 tools/sample), making per-tool ratios undefined.

Method	Tool Calls↓	Time (s)↓	Tool SR (%)↑	Avg ACC (%)↑	Eff.↑
ReAct	3.7	89.5	–	62.0	16.8
TS-Reasoner	2.50	**19.0**	**100.0**	58.3	23.3
TS-Agent	4.2	89.2	–	57.4	13.7
Ours	**2.25**	53.2	**100.0**	**75.3**	**33.5**
Δ *Ours vs. ReAct*	-39%	-41%	$+100$pp	$+21.5\%$	$+99\%$

A.5 Error Taxonomy

To analyze the failure cases of AgenTS, we classify each incorrect prediction into one of five categories, which we group into abstraction errors and reasoning errors: *abstraction errors* arise when the model cannot recover the underlying numerical structure, while *reasoning errors* arise when the structure is correct but the model misuses it. Per-task counts are shown in Figure 5.

Abstraction errors:

- **Misabstraction.** The model misreads numerical values from the tokenised series itself, e.g. wrong magnitude, missed peaks, or hallucinated trends. The pipeline never invokes a tool that would have surfaced the correct value.

Table 7: Component ablation of AgenTS on TimeOmni-1 (ID).

Variant	Scenario ACC↑	Causality ACC↑	Forecasting MAE↓	Decision ACC↑
AgenTS (Full)	**89.5**	**77.0**	**14.6**	**51.1**
Runtime state construction				
w/o Task Type	79.4	68.1	16.8	37.6
w/o Property Relevance	83.3	73.7	15.4	48.6
w/o Property Intensity	85.0	72.5	15.1	48.3
Toolpath orchestration with online learning				
w/o Cov	81.5	72.0	16.6	40.0
w/o Sim	80.7	70.0	17.4	41.0
w/o Dem	82.9	73.0	17.4	39.0
w/o Cost	80.5	70.0	18.0	40.0
w/o Toolpath Planning	82.3	65.8	15.6	46.8
w/o Online Learning	82.9	73.3	16.1	44.2
Runtime execution with reasoning workspace reconstruction				
w/o Workspace Reconstruction	81.1	69.1	17.3	42.6
w/o Runtime Execution	84.5	73.0	15.3	45.2

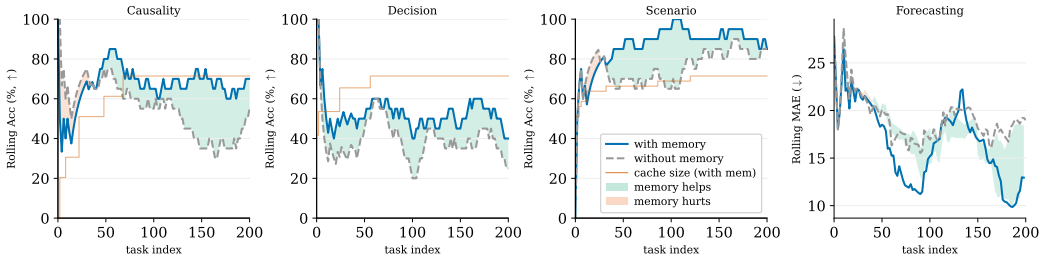


Figure 4: **Online learning improves with accumulated experience.** Rolling accuracy (or MAE) over the first 200 tasks on TimeOmni-1, comparing AgenTS with online learning (**blue**) and the variant without online learning (**dashed gray**). **Green** regions indicate where online learning improves performance, and the **orange** line shows the number of stored toolpaths as experience accumulates.

- **Tool Interpretation.** A tool is invoked and returns the correct statistic, but the model misinterprets its output — e.g. reading $p > 0.05$ as evidence of causality, or treating a high cross-correlation as a directional causal signal.

Reasoning errors:

- **Ambiguous evidence.** The tool returns multiple plausible signals (e.g. both Granger directions significant), and the model picks the wrong one despite valid inputs.
- **Grounding failure.** The tool ranks the correct option highest, but the model ignores the ranking and selects another option. This is a faithfulness failure of evidence-grounded generation.
- **Consistency failure.** The N voting drafts disagree, and majority vote selects the wrong answer — intrinsic reasoning variance that voting cannot eliminate.

B Case Study.

To demonstrate the impact of tool-based abstraction, Figure 6 shows a causality discovery instance. Without tools, the model misperceives the lag between the two series and infers a spurious causal direction. With AgenTS, the lagged-correlation analysis reveals a three-step misalignment between the peaks, so the significant Granger test is treated as weak evidence, and the model correctly concludes that the series are not causally related. This example shows that toolpaths selected from the runtime state handle the abstraction bottleneck more reliably than fixed tool configurations.

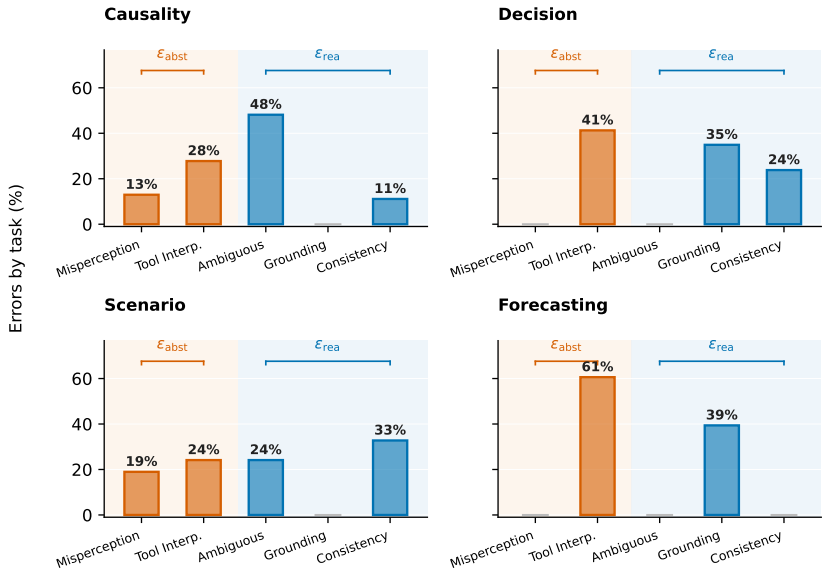


Figure 5: Per-task error breakdown on TimeOmni-1 in-distribution (ID). Each panel shows the share of errors falling into each category for one task. Categories are grouped into abstraction and reasoning errors as defined above.

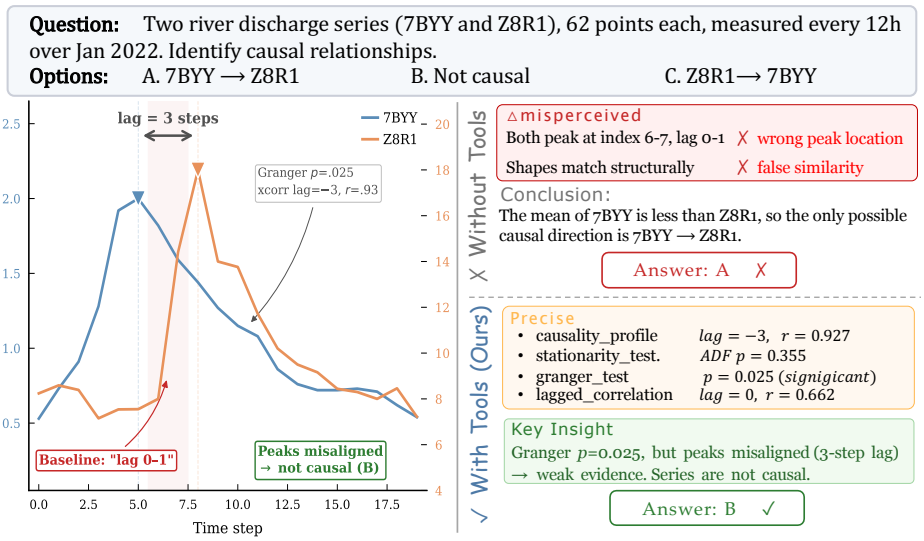


Figure 6: **Causality discovery example.** *Left:* Two river discharge series over time. *Right top:* without tools, misperceived lag leads to incorrect causal judgment. *Right bottom:* With our harness, exact tool outputs enable correct reasoning.

C AgenTS Framework

C.1 Runtime State Construction Details

The *Profiler* instantiates the property intensity vector $\mathbf{d} \in [0,1]^M$ (§3.2) with M=4 signals, each measuring the intensity of a distinct temporal property, drawn from the catch22 canonical feature set (Lubba et al., 2019) and classical descriptive statistics.

Spectral entropy (d_{spec}). Normalised Shannon entropy of the power spectral density:

$$d_{\text{spec}} = \frac{H\big(\text{PSD}(\mathbf{x})\big)}{\log K_f}, \tag{10}$$

where K_f is the number of frequency bins. High values indicate a spread spectrum (complex frequency structure requiring regime or anomaly tools); low values indicate a dominant frequency (simpler periodicity).

Stationarity (d_{struct}). The p-value of the Augmented Dickey–Fuller (ADF) test:

$$d_{\text{struct}} = p_{\text{ADF}}(\mathbf{x}). \tag{11}$$

High p-values indicate the unit-root null cannot be rejected (non-stationary series requiring differencing or causality tools); low values indicate stationarity.

Predictability (d_{acf}). One minus the maximum absolute autocorrelation over lags 1 to L:

$$d_{\text{acf}} = 1 - \max_{1\le \ell \le L} |\rho_\ell(\mathbf{x})|, \qquad L = \min\big(20,\ \lfloor T/3 \rfloor\big). \tag{12}$$

High values indicate weak autocorrelation (hard-to-predict series benefiting from decomposition or trend tools); low values indicate strong temporal structure.

Volatility (d_{vol}). Coefficient of variation, clipped to $[0, 1]$:

$$d_{\text{vol}} = \min\Big(1,\ \frac{\text{std}(\mathbf{x})}{|\text{mean}(\mathbf{x})| + \epsilon}\Big), \qquad \epsilon = 10^{-6}. \tag{13}$$

High values indicate large fluctuations relative to the level (risk-aware decision tasks or anomaly-prone series); low values indicate a stable signal whose temporal pattern dominates over its amplitude.

Task-conditioned relevance. Rather than fixing $\boldsymbol{\alpha}^\phi$ by hand, the *Profiler* generates it automatically at deployment via a single zero-shot LLM call per task type, using the same backbone $\mathcal{M}$ as the rest of the harness. The generated relevance vector is cached and reused for all subsequent instances of the same ϕ, so the total cost is $|\Phi|$ LLM calls over the entire deployment. Figure 7 shows the prompt template. The *Profiler* extracts the four values and renormalizes them so that $\sum_j \alpha_j^\phi = 1$ exactly, which handles rounding in the decimal output of the LLM. Sampling is deterministic (temperature 0), so re-running the same prompt against the same backbone produces the same $\boldsymbol{\alpha}^\phi$.

Task-type inference. The Profiler infers the task type ϕ from the question q via a zero-shot classification call to the same backbone $\mathcal{M}$ used in the rest of the pipeline. To force genuine discrimination, the prompt presents ten candidate task types (the four supported by AgenTS interspersed with six common time series question types as distractors): The agent extracts the first matching task name from the LLM's response via case-insensitive substring match. Of the ten candidates, four are supported by AgenTS (`causality_discovery`, `decision_making`, `scenario_understanding`, `event_aware_forecasting`); the remaining six act as distractors. If the inferred type falls outside the supported four (or no match is found), the agent defaults to `scenario_understanding`, the highest-recall fallback observed during calibration. The call adds a single short LLM round-trip per instance ($\sim$30 output tokens) and contributes negligibly to total pipeline latency.

Requirement bucketing and hard-instance gating. The requirement level d_ϕ lies in $[0, 1]$ by construction, as $\sum_j \alpha_j^\phi = 1$ and each $d_j \in [0, 1]$. The *Orchestrator* keys its online learning memory $\mathcal{W}_{\text{mem}}^{(\phi,b)}$ by the bucket:

$$b = \begin{cases} \text{low} & d_\phi < 0.4, \\ \text{mid} & 0.4 \le d_\phi < 0.7, \\ \text{high} & d_\phi \ge 0.7, \end{cases} \tag{14}$$

with each bucket retaining at most five toolpaths ranked by EMA-smoothed quality score.

The hard-instance threshold that gates LLM-based toolpath generation $\mathcal{W}_{\text{gen}}$ is a single global scalar $\theta = 0.6$ on the same $[0, 1]$ scale as d_ϕ: the *Orchestrator* generates new toolpaths for instances with $d_\phi > \theta$, and otherwise serves the request from memory.

```
You are calibrating a time-series reasoning system. Given a task type, assign four
real-valued weights in [0, 1] that sum to 1, indicating the relative importance of
four complexity axes for solving questions of this type:

  • spectral_entropy: frequency-domain richness; high if periodic, oscillatory, or
    multi-band structure matters.
  • stationarity: presence of trends, level shifts, or non-stationary drift; high if
    regime changes or trend dynamics matter.
  • autocorrelation: strength of temporal dependency and lag structure; high if
    lead-lag relationships or serial correlation matter.
  • volatility: amplitude variability relative to the signal level; high if risk,
    fluctuation magnitude, or anomaly susceptibility matters.

Task type: {phi}

Output exactly four decimals on a single line, comma-separated, summing to 1.0, in
the order: spectral_entropy, stationarity, autocorrelation, volatility.
```

Figure 7: Zero-shot prompt template used to generate the task-conditioned relevance vector $\boldsymbol{\alpha}^{\phi}$. Issued once per task type at deployment; the output is parsed into four values and renormalized to sum to one. The complexity axes in the prompt correspond to the temporal properties in §3.2.

```
You are classifying a time-series question into exactly one task type from the list
below. Respond with the task name only, no explanation.

Task types:
  • anomaly_detection: asks to identify anomalous points or outliers in the series.
  • decision_making: asks for the best action, optimal strategy, or correct decision
    given the series.
  • trend_analysis: asks to describe the overall direction or growth/decline of the
    series.
  • causality_discovery: asks which series causes which, the direction of a causal
    relationship, or whether two series are causally related.
  • classification: asks to assign the series to a discrete category from a
    predefined label set.
  • scenario_understanding: asks which scenario, situation, or condition the series
    corresponds to.
  • changepoint_detection: asks to find time points where the underlying
    distribution or regime shifts.
  • summarization: asks to summarize the series in natural language without making
    predictions.
  • event_aware_forecasting: asks to predict future values of a series, often
    conditioned on an event.
  • imputation: asks to fill in missing values within the observed series.

Question: {q}
Answer:
```

Figure 8: Zero-shot prompt template used to infer the task type ϕ from the natural-language question q. The four task types supported by AgenTS (`causality_discovery`, `decision_making`, `scenario_understanding`, `event_aware_forecasting`) are interspersed with six common time-series question types as distractors.

Configuration counts and lists. The *Profiler* uses $M = 4$ temporal properties (spectral entropy, stationarity, predictability, and volatility), $|\Phi| = 4$ task types (causality discovery, scenario understanding, decision making, and event-aware forecasting), and a tool library annotated with capability tags, a subset of which is mapped to the four properties through the anchor sets $\mathcal{B}_j$. The full library is listed in Table 8. Per-instance regeneration of $\boldsymbol{\alpha}^{\phi}$, which conditions the prompt on the specific

question q rather than only the task type ϕ, is supported by the harness but not used in the experiments of Section 4, since it offered no measurable accuracy gain in pilot runs while adding one LLM call.

C.2 Toolpath Orchestration Details

Utility terms. The task-required set $\mathcal{R}_\phi$ is induced from the relevance $\boldsymbol{\alpha}^\phi$:

$$\mathcal{R}_\phi = \bigcup_{j\,:\,\alpha_j^\phi > \eta} \mathcal{B}_j, \tag{15}$$

where $\mathcal{B}_j \subseteq \mathcal{C}$ is a fixed set of capability tags tied to temporal property j.

If no α_j^ϕ exceeds η, we set $\mathcal{R}_\phi = \mathcal{B}_{j^*}$ with $j^* = \arg\max_j \alpha_j^\phi$. Coverage measures the fraction of task-required capabilities provided by a toolpath w:

$$\mathrm{Cov}(w, \phi) = \frac{|\mathcal{C}(w) \cap \mathcal{R}_\phi|}{|\mathcal{R}_\phi|}, \qquad \mathcal{C}(w) = \bigcup_{t \in w} \mathcal{C}_t. \tag{16}$$

Relevance is computed with a frozen Sentence-BERT encoder $\mathbf{e}(\cdot)$ (`all-MiniLM-L6-v2`), with cosine values clipped to $[0, 1]$:

$$\mathrm{Rel}(w, q) = \frac{1}{|w|} \sum_{t \in w} \max\big(0,\, \cos\big(\mathbf{e}(q),\, \mathbf{e}(\mathrm{desc}(t))\big)\big). \tag{17}$$

Demand match is a weighted average of per-property capability coverage with weights $\alpha_j^\phi d_j$:

$$\mathrm{Dem}(w, \boldsymbol{\alpha}^\phi, \mathbf{d}) = \frac{\sum_j \alpha_j^\phi\, d_j\, m_j(w)}{\sum_j \alpha_j^\phi\, d_j}, \qquad m_j(w) = \frac{|\mathcal{C}(w) \cap \mathcal{B}_j|}{|\mathcal{B}_j|}, \tag{18}$$

with $\mathrm{Dem} = 0$ when $\sum_j \alpha_j^\phi d_j = 0$. Cost is the sum of per-tool costs, $\mathrm{Cost}(w) = \sum_{t \in w} c_t$.

Anchor sets and coefficients. The anchor sets $\mathcal{B}_j$ are fixed at library construction and shared across all task types:

Property j	Anchor capabilities $\mathcal{B}_j$
`spectral_entropy`	{`seasonality`, `autocorrelation`, `lag_structure`}
`stationarity`	{`stationarity`, `regime`, `changepoint`, `trend`}
`autocorrelation`	{`lag`, `correlation`, `causality`, `direction`}
`volatility`	{`anomaly`, `threshold`}

We set $\eta = 0.3$, so $\mathcal{R}_\phi$ usually includes one or two anchor sets per task. The utility coefficients are fixed across all tasks and datasets, with $\lambda_1 + \lambda_2 + \lambda_3 = 1$ and a separate cost penalty λ_4:

	λ_1 (Cov)	λ_2 (Rel)	λ_3 (Dem)	λ_4 (Cost)
Value	0.43	0.14	0.43	0.05

Input compatibility filtering. Before ranking, multivariate-only tools (`granger_test`, `lagged_correlation`, `causality_*`) are removed for univariate inputs, forecasting tools are removed for non-forecasting tasks, and duplicate tools within a toolpath are removed. If no candidate remains, the *Orchestrator* falls back to the single-tool toolpath `trend_decompose`.

Online learning memory. The memory is keyed by (ϕ, b), where $b \in \{\mathrm{low}, \mathrm{mid}, \mathrm{high}\}$ is the requirement bucket from Appendix C.1. Each bucket keeps at most $K_{\mathrm{mem}} = 5$ toolpaths ranked by quality score, and the lowest-scoring toolpath is removed when the bucket is full. New toolpaths start with score 0.5, and scores are updated by Eq. 8 with $\beta = 0.8$, using the outcome $o \in [0, 1]$. For the first $N_{\mathrm{warm}} = 5$ instances of each bucket, the *Orchestrator* generates new toolpaths for every instance; afterwards, generation is used only when $d_\phi > \theta$.

Top-K selection. The *Orchestrator* forwards the $K = 2$ toolpaths with the highest utility to the *Synthesizer*; pilot runs showed diminishing returns beyond $K = 2$. When two toolpaths differ in utility by less than 10^{-3}, the one with lower cost is selected.

Toolpath generation prompt. For toolpath generation, the *Orchestrator* issues the prompt in Figure 9 to $\mathcal{M}$. The output is split on commas, each token is matched to a registered tool name by case-insensitive substring match, and the toolpath is truncated to $L_{\max} = 4$ tools.

```
You are composing a tool workflow for a time-series reasoning system. Given a task
type, an instance summary, and an inventory of available tools, return an ordered
sequence of tool names whose outputs will jointly cover the task's perception
requirements.

Task type: {phi}
Instance summary: {textual summary of x produced by the Profiling Agent: dominant
    frequency, stationarity verdict, top-3 autocorrelation lags}
Available tools: {enumeration of T from the tool library, with name and capability
    tags}

Output a comma-separated ordered list of at most L_max = 4 tool names from the
inventory above, with no explanation. The first tool listed will be executed first.
```

Figure 9: Zero-shot prompt template used by the *Orchestrator* to generate new toolpaths.

Per-tool cost profiling. The cost c_t is measured once at library construction by running every tool on a profiling set $\mathcal{D}_{\text{prof}}$. For each run, we record the wall time $r_t(\mathbf{x})$ and the number of tokens $n_t(\mathbf{x})$ in the serialized tool output, and combine them in token-equivalent units:

$$c_t(\mathbf{x}) = n_t(\mathbf{x}) + \kappa \cdot r_t(\mathbf{x}), \qquad c_t = \frac{1}{|\mathcal{D}_{\text{prof}}|} \sum_{\mathbf{x} \in \mathcal{D}_{\text{prof}}} c_t(\mathbf{x}), \tag{19}$$

where $\kappa = 10$ tokens per second.

C.3 Runtime Execution Details

Evidence construction. The observation interface $\mathcal{F}_{\text{obs}}$ builds the structured evidence $\mathcal{V} = (\mathcal{S}_{\text{text}}, \mathcal{G}_{\text{reg}})$ in three steps. First, the output of each tool is written in the template `[tool_name]: {key: value, ...}`, and these entries are concatenated in toolpath order to form $\mathcal{S}_{\text{text}}$; forecasts use the same template with the key `forecast`. Second, regime boundaries reported by different toolpaths are clustered with a tolerance of ± 2 time steps, and the center of each cluster is kept in $\mathcal{G}_{\text{reg}}$. Third, a header line `confidence: <High|Med|Low>` is added to $\mathcal{S}_{\text{text}}$ based on the number of failed tools: `High` for no failure, `Med` for one or two failures, and `Low` otherwise.

Tool execution. Tools within a toolpath run sequentially, and tools shared across the K selected toolpaths are executed once. Each tool call is limited to 30 seconds. If a tool fails (e.g., timeout, NaN output, or dimension mismatch), it is recorded in $\mathcal{S}_{\text{text}}$ as `[tool_name]: {status: failed, reason: <one-line>}`, and tools that depend on its output are skipped; the remaining tools are still executed.

Answer prompt and extraction. The model receives the prompt in Figure 10, which contains the question, the evidence $\mathcal{V}$, and a task-specific output format. Multiple-choice tasks require `<answer>X</answer>` with $X \in \{A, B, C, D\}$, and event-aware forecasting requires `<answer>[v1, v2, ...]</answer>` with the length of the forecast horizon. Answers are extracted with a regular expression; a missing or malformed answer is scored as wrong for multiple-choice tasks and assigned the MAE of the no-tool baseline for forecasting.

Outcome for memory update. The outcome $o \in [0, 1]$ in Eq. 8 is $o = \mathbb{1}[\hat{a} = a^*]$ for multiple-choice tasks and $o = \exp(-\text{MAE}/\text{MAE}_{\text{ref}})$ for event-aware forecasting, where MAE_{ref} is the MAE of the no-tool baseline on held-out samples of the same task. This puts both outcome types on the

```
You are answering a time-series question grounded in the structured evidence below.
Use only the evidence and the question; do not invent values not present in the
evidence. Produce the final answer in the requested format only, with no surrounding
explanation.

Question: {q}

Evidence:
confidence: <High|Med|Low>
[tool_1]: {key: value, ...}
[tool_2]: {key: value, ...}
...
regimes: [t_1, t_2, ...]

Format:
 • Multiple-choice tasks (causality_discovery, decision_making,
   scenario_understanding): output <answer>X</answer> where X in {A, B, C, D}.
 • Event-aware forecasting: output <answer>[v_1, v_2, ..., v_H]</answer> with
   exactly H numeric values matching the gold horizon.

When confidence is Low, treat the evidence as advisory rather than authoritative and
fall back on the question text.

Answer:
```

Figure 10: Prompt template used by the *Synthesizer* for answer generation from the structured evidence $\mathcal{V}$. The output format depends on the task type, so the same prompt covers all four tasks.

same scale. After execution, the *Synthesizer* applies Eq. 8 to each of the K executed toolpaths in the corresponding (ϕ, b) bucket.

D TOOL DETAILS

Table 8: Tool library reference. All 21 atomic tools grouped by category. **Description** summarizes each function's behavior. **Capabilities** are the tags exposed to the orchestrator's utility function (Eq. 6). $\bar{r}_t$ (ms) and $\bar{n}_t$ (tokens) are the mean wall-clock runtime and serialized-output token count over 10 calls on a bivariate synthetic series of length 96 with the TimeOmni-1 tokenizer. c_t is the empirical token-equivalent cost $c_t = \min\big((\bar{n}_t + \kappa \bar{r}_t)/Z,\ 1\big)$ with $\kappa = 10$ tokens/s (Eq. 19) and $Z = 896$ (the second-largest profiled c_t).

Tool	Description	Capabilities	$\bar{r}_t$ (ms)	$\bar{n}_t$ (tok)	c_t
Statistical					
`trend_decompose`	Savitzky–Golay trend extraction; returns slope and residual std.	`trend, seasonality`	0.41	1918	1.0000
`stationarity_test`	Augmented Dickey–Fuller test for unit root.	`stationarity`	3.05	55	0.0614
`acf_pacf`	ACF/PACF up to n_{lags}; returns significant lag indices.	`autocorrelation, lag_structure`	1.13	896	1.0000
`anomaly_zscore`	Z-score anomaly detection; returns indices and summary stats.	`anomaly`	0.03	39	0.0435
`threshold_analysis`	Threshold-crossing analysis at a given percentile.	`decision, threshold`	0.08	66	0.0737
`option_score`	Composite option scorer for decision tasks.	`decision, option`	0.99	47	0.0525
`scenario_simulate`	Multiplicative perturbation counterfactual.	`counterfactual, scenario`	0.01	80	0.0893
Structural					
`changepoint_detect`	Sliding-window mean-difference changepoint detector.	`regime, changepoint`	0.66	37	0.0413
`salient_window`	Top-k highest-variance windows of the series.	`evidence, attribution`	0.23	116	0.1295
Relational					
`lagged_correlation`	Pearson correlation at each lag; returns best lag and coefficient.	`lag, correlation`	7.00	602	0.6719
`granger_test`	Bidirectional Granger causality test.	`causality, direction`	5.54	41	0.0458
Forecasting					
`naive_forecast`	Linear extrapolation from the last 10 observations.	`forecast`	0.03	108	0.1205
`arima_forecast`	ARIMA(1,1,1) forecast with CIs; falls back to naive on failure.	`forecast, trend`	18.44	108	0.1207
`forecast_profile`	Forecast reasoning profile: momentum, periodicity, extrapolation.	`forecast, trend, seasonality`	0.47	123	0.1373
Domain — Causality					
`causality_binary_gate`	Domain-rule binary gate over candidate causal directions.	`causality, direction`	11.70	103	0.1151
`causality_profile`	Bidirectional Granger + cross-correlation + lag profile.	`causality, correlation, lag, direction`	12.34	280	0.3126
`causality_option_analysis`	Per-option causality scoring against the question's MCQ choices.	`causality, direction, option, correlation`	12.14	11	0.0124
Domain — Decision					
`decision_profile`	Peak/off-peak load, volatility, and trend summary.	`decision, threshold`	0.23	88	0.0982
`decision_option_analysis`	Per-option pricing-rule compliance scoring.	`decision, option, pricing`	0.24	88	0.0982
Domain — Scenario					
`scenario_profile`	Length, magnitude, periodicity, and pattern descriptors.	`scenario, evidence`	0.30	155	0.1730
`scenario_option_analysis`	Per-option compatibility analysis with explicit length matching.	`scenario, option, evidence`	0.52	155	0.1730