TS-Agent: A Time Series Reasoning Agent with Iterative Statistical Insight Gathering

Penghang Liu
JPMorgan AI Research
penghang.liu@jpmchase.com

Elizabeth Fons
JPMorgan AI Research
elizabeth.fons@jpmchase.com

Svitlana Vyetrenko
JPMorgan AI Research
svitlana.s.vyetrenko@jpmchase.com

Daniel Borrajo
JPMorgan AI Research
daniel.borrajo@jpmchase.com

Vamsi K. Potluru
JPMorgan AI Research
vamsi.k.potluru@jpmchase.com

Manuela Veloso
JPMorgan AI Research
manuela.veloso@jpmchase.com

Abstract

Large language models (LLMs) have shown strong abilities in reasoning and problem solving, but recent studies reveal that they still struggle with time series reasoning tasks, where outputs are often affected by hallucination or knowledge leakage. In this work we propose TS-Agent, a time series reasoning agent that leverages LLMs strictly for what they excel at, i.e., gathering evidence and synthesizing it into conclusions through step-by-step reasoning, while delegating the extraction of statistical and structural information to time series analytical tools. Instead of mapping time series into text tokens, images, or embeddings, our agent interacts with raw numeric sequences through atomic operators, records outputs in an explicit evidence log, and iteratively refines its reasoning under the guidance of a self-critic and a final quality gate. This design avoids multi-modal alignment training, preserves the native form of time series, ensures interpretability and verifiability, and mitigates knowledge leakage or hallucination. Empirically, we evaluate the agent on established benchmarks. Our experiments show that TS-Agent achieves performance comparable to state-of-the-art LLMs on *understanding* benchmarks, and delivers significant improvements on *reasoning* tasks, where existing models often rely on memorization and fail in zero-shot settings.

1 Introduction

Time series data play a central role in many high-stakes domains such as finance, healthcare, climate science, and energy systems. Understanding and reasoning about these data is often crucial for decision-making. For example, identifying whether a sudden change in a patient's heart rate signals a genuine medical risk, or determining whether a sharp movement in stock prices is driven by structural market factors or short-term noise. Such scenarios require more than simple forecasting, which demands factual understanding, mathematical reasoning, and the ability to uncover causal or correlational relationships in sequential data [12, 3, 5, 16].

Large language models (LLMs) have recently demonstrated remarkable capabilities in reasoning across a wide range of textual and mathematical tasks. Techniques such as chain-of-thought prompting and self-consistency have shown that LLMs can decompose complex questions into step-by-step reasoning trajectories and arrive at more reliable answers [15, 14, 6, 23]. Recent works also explore the use of LLMs for time series forecasting, understanding, and question answering, often through multimodal approaches that represent time series as text, images, or embeddings [8, 22, 4, 21, 7, 18, 2, 20]. These developments suggest that LLMs hold potential as general-purpose reasoning engines for time series data.

However, recent evidence shows that reasoning over time series remains a significant challenge for LLMs. Merrill et al. [10] systematically evaluate language models on time series reasoning tasks and find that, despite their strengths in textual reasoning, models consistently underperform in zero-shot reasoning tasks such as etiological reasoning, context-aided forecasting, and question answering. They attribute these limitations to the mismatch between continuous quantitative inputs and token-based representations, the difficulty of precise numerical manipulation, and the specialized forms of inductive and causal reasoning required in time series domains. Moreover, LLMs can sometimes answer correctly without consulting the data due to knowledge leakage [10] or fabricate unsupported details as chains grow longer [18].

Motivated by this gap, we propose an AI agent for time series reasoning that uses the LLM strictly where it is strongest, i.e., *reasoning* (integrating properties to infer higher-level conclusions), and delegates *understanding* (extracting properties) of raw time series to domain-appropriate analytical functions (difference between reasoning and understanding are shown in fig. 1). This yields four concrete advantages aligned with our design motivations: (i) it removes the need for multi-modal alignment training to make LLMs "perceive" time series, since the LLM never has to parse numeric sequences as text or images; (ii) it preserves the native, quantitative form of time series (raw numeric sequences) so no information is lost compared to multi-modal approaches, and precise statistics and characteristics are obtained by exact computation; (iii) it guides problem solving in a human-like, auditable manner via an interleaved think-act-observe loop (in the spirit of [19]), where the LLM decomposes the task and iteratively calls atomic tools to gather evidence; and (iv) it enables self-refinement via a step-wise critic and a final quality gate that enforce evidence grounding, thereby mitigating knowledge leakage and hallucination by rejecting answers not supported by the computed facts [10, 18].

Taken together, this *reason-with-tools* paradigm treats time series as they are (numeric sequences) and reserves natural-language modeling for coordinating analysis and forming conclusions, rather than for perceiving the data itself. In the rest of the paper we formalize the distinction between time-series *understanding* and *reasoning* , present our agent framework, and evaluate it on established benchmarks.

2 Related Works

Time Series Reasoning Benchmarks. With the rapid advancement of LLMs, several works have proposed benchmarks and multimodal methods for handling time-series reasoning problems [7, 22, 4, 21]. Among these, TimeSeriesExam [1] provides a large-scale multiple-choice exam across five categories (pattern recognition, noise understanding, anomaly detection, similarity, and causality), and Fons et al. [2] introduce a taxonomy and benchmark for time-series feature understanding, retrieval, and arithmetic reasoning. Xie et al. [18] propose ChatTS, a multimodal LLM aligned with synthetic time series–text pairs, showing improvements on both forecasting and question answering. Despite these promising advances, Merrill et al. [10] demonstrate that current LLMs still struggle on genuine reasoning tasks, often producing nearly random answers or achieving high accuracy even when the time-series input is removed, revealing that their performance frequently reflects memorized knowledge rather than actual data analysis.

AI Agents for Reasoning. General-purpose reasoning agents such as ReAct [19], Reflexion [11], and Self-Refine [9] combine natural-language reasoning with tool use and self-correction. Recent surveys further highlight LLM-based agents as a promising paradigm for complex reasoning tasks across domains [13, 17]. In the time series domain, Ye et al. [20] propose TS-Reasoner, a domain-specific agent for electricity load forecasting. Their approach uses the LLM purely as a task decomposer that produces a complete plan executed by external programs. Our work differs in two key aspects: (i) we address general time-series reasoning across diverse domains and evaluate on non–domain-specific benchmarks; and (ii) our TS-Agent performs iterative ReAct-style reasoning, where each

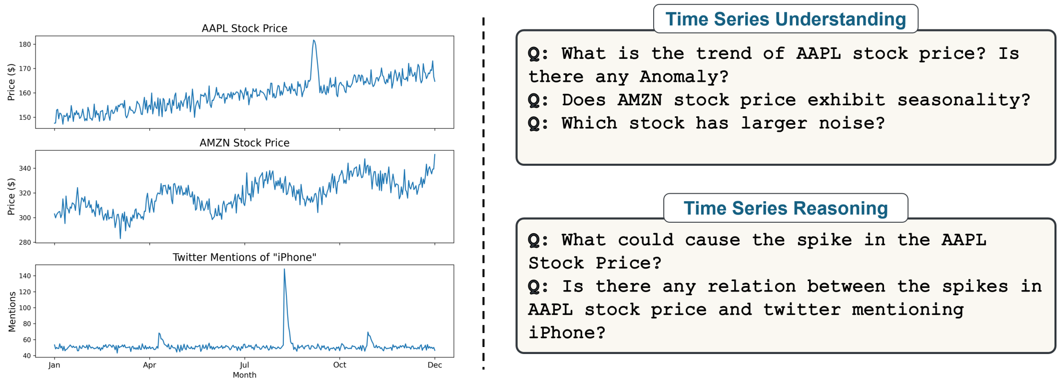


Figure 1: The two types of time series questions.

step is logged in an evidence log, reviewed by a critic, and verified by a final quality gate, ensuring transparency and verifiability rather than relying on a one-shot plan.

3 The Time-Series Reasoning Problem

Language models have been used to address time series questions from a wide spectrum, but the nature of these questions spans two fundamentally different categories. We distinguish between *time-series understanding*, where the answer is a direct property or characteristic of the series, and *time-series reasoning*, where the answer requires integrating multiple aspects of the series dynamics to infer higher-level conclusions. This distinction is critical: while many prior works evaluate LLMs on understanding tasks [1, 2], recent studies show that reasoning remains a major challenge [10]. In this paper we evaluate our agent on both categories, but our framework is designed primarily to tackle the reasoning cases. Examples of the time series understanding and reasoning questions are given in fig. 1

3.1 Time-Series Understanding

Understanding tasks ask for explicit concepts or features of a time series. Formally, Given a multivariate time series $X \in \mathbb{R}^{d \times T}$ and a natural language question q, the target output y is a *characteristic* $c \in \mathcal{C}$ or *statistic* $r \in \mathbb{R}$ that can be directly extracted from the data:

$$y = f(X; \theta)$$

using some task-specific function or model f with parameter θ. Typical examples include pattern recognition (trend, seasonality, anomaly, shifts), computation (variance, auto-correlation, forecasting), and comparison (similarity, correlation, causal relation). Such problems can often be solved by a single analytic operator or by a specialized predictive/detection model, which correspond to the problems in TimeSeriesExam [1] and the time series feature understanding benchmark [2]. While sometimes described as "reasoning," these tasks in fact primarily require the ability to extract and identify well-defined properties from data.

3.2 Time-Series Reasoning

Reasoning tasks go beyond extracting properties to require logical inference about the *dynamics and mechanics* of the series. The answer itself is not a time-series concept but a conclusion derived from combining several such concepts. Formally, given (X, q), the solution requires a composition

$$y = \Phi\big(f_1(X), f_2(X), \ldots, f_n(X)\big)$$

where Φ integrates multiple intermediate properties into a higher-level inference. Some typical examples are: (1) Given temperature and electricity consumption series, predict the effect of a spike in temperature. Solving requires understanding causal relations (temperature $\rightarrow$ demand) and daily seasonality of consumption. (2) Given solar panel output over one month, identify a cloudy period.

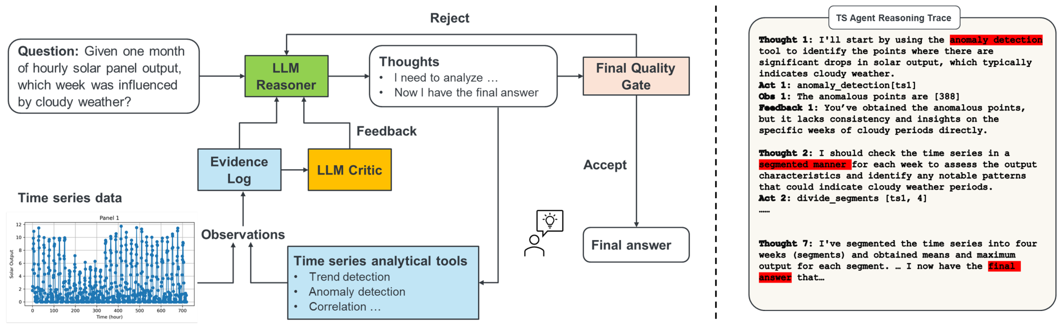


Figure 2: The TS Agent framework (left) and example of reasoning trace (right).

This demands integrating knowledge of diurnal cycles (peaks at noon, zero at night) with deviations in daily maxima. (3) Choosing the best investment among several stocks, which requires analyzing trends, volatilities, correlations, and forecasts jointly.

Merrill et al. [10] define several time series reasoning tasks and find that current LLMs perform near random or substantially below human baselines, underscoring that true reasoning about time series remains unsolved.

3.3 Our Focus

We adopt this two-way distinction to structure our evaluation. We benchmark our agent on both *understanding* and *reasoning* tasks, using existing benchmark datasets [1, 2, 10]. However, our work is specifically motivated by the defects of AI model in addressing the reasoning tasks. Therefore we propose a reasoning agent to decomposing complex queries into a iterative reasoning trace that conduct atomic analyses and integrate evidence into a concrete answer.

4 Time Series Agent Framework

We design a time series reasoning agent that address complex reasoning problems through an iterative process of reasoning, evidence gathering, and self-refinement. Figure 2 presents the pipeline of TS Agent. Given a multivariate time series $X \in \mathbb{R}^{d\times T}$ and a natural language question q, the agent decomposes the problem into reasoning steps, calls analytical tools iteratively to extract evidence from X, and integrates observations into its chain of thought until reaching a final answer. In addition, we employ a critic to review each step for self-refinement, and a quality gate for final answer verification. This design ensures accurate quantitative analysis via tools, interpretability via step-by-step reasoning trace, and robustness through iterative self-refinement and final quality control, contrasting with non-agentic LLM approaches that directly map time series input into final answer without structured reasoning [1, 22].

4.1 Iterative LLM Reasoner

At the core of our framework is an *iterative LLM reasoner* that drives the reasoning process through cycles of thought, action, and observation. Rather than producing a fixed plan upfront, the agent adaptively generates the next reasoning step only after incorporating the structured outcome of the previous step. This makes the agent responsive to evidence as it emerges, and robust against early mistakes.

Formally, let $\mathcal{T} = \{t_1, \ldots, t_M\}$ be the set of callable tools, and $\mathcal{L}_k = \{e_1, \ldots, e_M\}$ be *evidence log*, i.e., the memory module that records of all accumulated observations produced during reasoning. The state of the agent at step k consists of the query, a reasoning trace $\tau_{1:k-1}$ of past thoughts and actions, and the *evidence log* $\mathcal{L}_{k-1}$ containing accumulated observations. The LLM generates the next *thought*:

$$\theta_k = f_{\mathrm{LLM}}(q, \tau_{1:k-1}, \mathcal{L}_{k-1}),$$

where θ_k is a natural-language hypothesis or sub-goal.

Conditioned on this thought, the agent selects a tool $t_k \in \mathcal{T}$ and structured arguments α_k, forming an *action* $a_k = (t_k, \alpha_k)$. Execution of this action produces a structured *observation*: $o_k = t_k(X; \alpha_k)$, where o_k may be numeric, categorical, or relational depending on the tool type. The observation is saved in the *evidence log*, then the cycle proceeds with the updated context $(\tau_{1:k}, \mathcal{L}_k)$, ensuring that subsequent reasoning steps can build on all accumulated evidence. The trajectory continues until the LLM concludes that sufficient evidence has been gathered to answer the query or declares the problem undecidable.

This iterative mechanism distinguishes our design from static planners: at each step, reasoning is grounded not only in the original question q but also in the dynamically growing buffer of evidence $\mathcal{L}_k$. The explicit recording of intermediate observations improves interpretability and provides the foundation for later modules such as the critic and quality gate.

4.2 Time Series Analytical Tools

To support fine-grained reasoning, we equip the agent with a library of callable functions (APIs) that return structured information from the time series data. Inspired by prior works on agents [19] and time series LLM integration [1], we develop the following toolkit that decomposes complex reasoning into atomic analytical steps while anchoring the reasoning process in precise numerical computations.

Data Processing ($f_{\text{proc}} : \mathcal{X} \times \Theta \to \mathcal{X}$). These operators enable the agent to select a part of the original time series for later analysis. Examples include segmentation, slicing, and temporal resampling.

Detection & Classification ($f_{\text{det}} : \mathcal{X} \times \Theta \to \mathcal{C}$). This include operators that map a series to categorical descriptors of its dynamics, including trend direction, anomaly flags, regime shifts, or seasonality type. Such tools allow the agent to bridge between raw data and interpretable symbolic evidence.

Numerical Operations ($f_{\text{num}} : \mathcal{X} \times \Theta \to \mathbb{R}$). The numerical operators extract real-valued statistics including mean, variance, volatility, quantiles, autocorrelation, and returns. These operators provide precise numerical evidence to support quantitative reasoning.

Relations & Comparison ($f_{\text{rel}} : \mathcal{X}^k \times \Theta \to \mathbb{R}$ or $\mathcal{C}$). These functions evaluate similarity or dependence between series based on correlations, distances, shape similarities, or causal effects across multiple time series. Such relations enable reasoning about interactions rather than single-series properties.

Custom Operator Synthesis ($f_{\text{custom}} : q \mapsto f \in \mathcal{F}$). This is an extension mechanism where the agent can construct new operators from natural-language specifications. For instance, synthesizing a "volatility-adjusted moving average" operator if not already present. This mechanism allows the agent to generalize beyond the fixed library when solving novel tasks.

This comprehensive tool set ensures that every reasoning step is grounded in verifiable computations directly tied to the numerical structure of the time series. The outputs are always typed (time series, real values, or categories), which makes them composable within the iterative reasoning loop.

4.3 Self-Refinement and Final Quality Gate

Beyond the iterative reasoner, our framework introduces two complementary mechanisms that improve reliability and auditability: a step-wise *critic* for *self-refinement* inspired by [11, 9] and a *quality gate* for final answer verification. In particular, we extract *question intents*, which specifies the requriements in the question and what must be conducted on the data to answer the question. Formally,

$$\phi_{\text{intent}} : \; q \;\mapsto\; \big(\underbrace{\texttt{task}}_{\text{type}},\; \underbrace{\texttt{schema}}_{\text{output format}},\; \underbrace{\mathcal{R}}_{\text{required predicates}}\big),$$

where $\mathcal{R} = \{r_1, \ldots, r_m\}$ is a set of predicates that must be supported by evidence, inferred from the keywords in the question. After k steps, the evidence log $\mathcal{L}_k$ induces the covered set

$$\mathcal{C}_k \;=\; \{\, r \in \mathcal{R} \;:\; \exists\, e \in \mathcal{L}_k \text{ that verifies } r \,\}, \qquad \mathcal{G}_k \;=\; \mathcal{R} \setminus \mathcal{C}_k$$

as the current *gap set*. Our critic and quality gate operate by comparing $\mathcal{G}_k$ against $\mathcal{R}$. In practice (`detect_question_intents()`), ϕ_{intent} is realized via transparent rule bundles: keyword/pattern matches map questions into a small set of intent classes (e.g., `MCQ_anomaly_location`, `trend_direction`, `seasonality_type`, `relation_lagged`), each declaring its `schema` and $\mathcal{R}$. For example, a question of "In which part of the time series does the anomaly occur?" triggers `MCQ_anomaly_location` with

$$\mathcal{R} = \{\texttt{has\_anomaly},\ \texttt{anomaly\_segment} \in \{\texttt{beginning}, \texttt{middle}, \texttt{end}\}\}.$$

As tools run, the log populates $\mathcal{C}_k$ (e.g., anomaly detected in $[t_1, t_2]$); any unmapped predicate remains in $\mathcal{G}_k$ and drives subsequent steps.

Step-wise Critic. After the execution of each action, we invoke the same LLM in a critic role to assess the new evidence in the context of the accumulated reasoning trace and the evidence log. Specifically, the critic reviews $(\theta_k, \mathcal{L}_k,\ \phi_{\text{intent}}(q))$ and checks: (i) *tool suitability*, whether the chosen operator aligns with the current sub-goal and question intent; (ii) *output plausibility*, whether the observed values are consistent with prior evidence; and (iii) *evidence sufficiency*, whether the log contains all predicates required to satisfy the question intent. Based on the review, the critic may raise suggestion for correction of a wrong usage, using a different tool, or gather missing evidence. This dynamic loop allows the agent to backtrack from weak evidence, avoid compounding errors, and iteratively converge toward stronger conclusions.

Final Quality Gate. When the agent proposes an answer $\hat{y}$, the quality gate enforces data-grounding before output. We formalize the gate as

$$\Gamma(q, \mathcal{L}_k, \phi_{\text{intent}}(q), \hat{y}) \;\in\; \{\texttt{accept}\} \,\cup\, \{\texttt{reject}\} \times \Delta_k,$$

where Δ_k is a set of concrete rejection reasons. The gate checks: (i) *schema compliance*—$\hat{y}$ matches `schema` (e.g., belongs to the MCQ option set); and (ii) *evidence sufficiency*—all required predicates are verified ($\mathcal{G}_k = \varnothing$) and no contradictions remain. If any check fails, the gate returns `reject` along with Δ_k (e.g., "missing `trend_direction`", "contradictory `seasonality` labels"), which is fed back to the reasoner to continue gathering evidence. The process repeats until acceptance or the step budget is exhausted; only then do we emit `AGENT_FAILURE` with the unresolved Δ_k. This is crucial because LLMs may sometimes answer correctly without consulting data due to knowledge leakage [10] or fabricate unsupported details in longer chains [18]. By requiring acceptance from Γ, our framework ensures final outputs are justified solely by verifiable predicates extracted from the input time series and recorded in the evidence log.

5 Experiments and Results

We evaluate our agent on two aspects: *time series understanding*, where the goal is to extract and identify fundamental concepts or characteristics, and *time series reasoning*, which requires integrating such characteristics to answer higher-level inference questions. Across all experiments we use `gpt-4o-mini` as the LLM reasoner. Although this is not the strongest available model, our results demonstrate that even with a relatively lightweight backbone the proposed framework achieves competitive performance on understanding benchmarks and substantial gains on reasoning tasks.

5.1 Time Series Understanding

We first evaluate our agent on *time series understanding* tasks using the TimeSeriesExam benchmark [1]. This dataset is designed as a comprehensive exam for evaluating whether models can recognize fundamental time-series concepts. It contains 763 multiple-choice questions generated from over one hundred templates and refined through iterative item response theory (IRT). The questions span five categories: **(i)** pattern recognition, including trend, stationarity, and regime switching; **(ii)** noise understanding, such as distinguishing between white and red noise or estimating signal-to-noise ratios; **(iii)** anomaly detection, including both local and global anomalies; **(iv)** similarity analysis,

Model	Pattern Rec.	Noise Und.	Anomaly Det.	Similarity	Causality
Phi-3.5	0.44	0.24	0.25	0.41	0.25
DeepSeek	0.63	0.58	0.43	**0.66**	0.28
GPT-4o	0.46	**0.65**	0.42	0.65	0.29
GPT-o1	0.65	0.62	**0.58**	0.62	0.37
TS-Agent	**0.71**	0.61	0.57	0.57	**0.55**

Table 1: Comparison of model accuracy on five time series understanding tasks [1]: pattern recognition, noise understanding, anomaly detection, similarity analysis, and causality analysis. Baseline performance are obtained from [1, 20].

Metric	GPT-4	GPT-3.5	Llama2	Vicuna	Phi-3	TS-Agent
Univariate time series characteristics						
Feature detection						
Trend	0.79	0.45	0.51	0.58	0.72	**0.82**
Seasonality	**0.94**	0.43	0.64	0.49	0.82	0.86
Anomalies	0.84	0.57	0.47	0.49	0.43	**0.85**
Volatility	0.68	0.43	0.42	0.45	0.73	**0.77**
Struct. break	**0.59**	0.57	0.39	0.48	0.44	0.53
Feature classification						
Trend	**0.98**	0.78	0.43	0.53	0.48	**0.98**
Seasonality	0.17	0.17	0.31	0.23	0.48	**0.58**
Anomalies	**0.87**	0.20	0.30	0.37	0.53	**0.87**
Volatility	0.18	0.07	0.12	0.15	0.08	**0.25**
Struct. break	0.42	**0.56**	0.30	0.41	0.51	0.55
Multivariate time series characteristics						
Fixed corr.	0.48	0.39	0.38	0.40	0.43	**0.53**
Lagged corr.	0.54	0.52	0.45	0.42	0.41	**0.55**
Changing corr.	0.48	0.43	0.52	0.50	0.48	**0.53**

Table 2: Performance (F1 scores) on time series feature understanding benchmark [2]).

testing whether two series share shapes or distributions; and **(v)** causality analysis, focusing primarily on Granger-type causal relations. Each question is grounded in either synthetic or counterfactual data generated with explicit control over the target characteristics.

Table 1 compares our TS-Agent with baseline LLMs reported in prior work as well as newly obtained results. While GPT-o1 and DeepSeek achieve strong performance on several categories, our agent consistently matches or outperforms them, attaining the highest scores on pattern recognition (0.71) and causality analysis (0.55). Notably, causality questions have been reported as among the hardest categories in TimeSeriesExam, where open-source and proprietary LLMs typically struggle. To further assess understanding capabilities, we also evaluate on the time series feature understanding benchmark [2], which measures feature detection and classification on univariate and multivariate series. Table 2 shows that TS-Agent achieves the best performance on most tasks, including trend detection (0.82), anomaly detection (0.85), volatility estimation (0.77), and nearly all classification tasks. While GPT-4 remains slightly stronger on seasonality detection and structural break classification, our agent matches or exceeds it in nearly all other categories, especially for multivariate time series characteristics. These results show that by grounding answers in explicit analytic computations, our framework can achieve performance comparable to or stronger than state-of-the-art LLMs, despite using a relatively lightweight backbone (`gpt-4o-mini`) as the reasoning engine.

5.2 Time Series Reasoning

We next evaluate our agent on time-series reasoning tasks as defined by Merrill et al. [10]. These tasks move beyond simple extraction to require inference about dynamics and causality, often integrating multiple streams or temporal patterns.

	One TS	Two TS**
LLAMA-7B	78.8%*	25.2%*
LLAMA-13B	82.5%*	25.8%*
GPT-3.5	88.2%*	27.4%*
GPT-4	**92.3**%*	52.7%*
GPT-4-Vision	91.8%*	53.6%*
TS-Agent	80.5%	**57.3%**

Table 3: Comparison of model accuracy on time series reasoning tasks [10]

* Merrill et al. [10] showed that pretrained LLMs achieved 78–92% accuracy even without time series data, indicating that the answers stem from knowledge leakage rather than time series reasoning.
** Human annotators achieved 67% accuracy on the two time series tasks.

Table 3 compares our agent (TS-Agent) to various LLMs on the "one time series" and "two time series" reasoning benchmarks. While GPT-4 achieves 92.3% accuracy for one series and 52.7% for two series, it is showed that most of this performance reflects knowledge leakage rather than true reasoning, as pretrained LLMs without access to data already achieve similar scores. With black-box LLMs it is therefore difficult to determine whether an answer was derived from the input series or simply recalled from training. In contrast, our agent framework provides a transparent reasoning trace and a verifiable evidence log. The final quality gate ensures that only answers grounded in explicitly gathered evidence are accepted, preventing the LLM reasoner from producing unsupported outputs through hallucination or knowledge leakage.

On the other hand, our agent obtains 80.5 % accuracy on the single-series task (competitive with larger models) and surpasses all baselines on the two-series reasoning task, achieving 57.3 % accuracy. The latter result is especially significant, as two-series reasoning requires integrating evidence across multiple signals where large LLMs often fail without genuine computation. By decomposing the question into analytical tool calls, recording results in an explicit evidence log, and iteratively refining through a critic and quality gate, TS-Agent demonstrates authentic reasoning over time series.

6 Conclusion

In this paper, we presented a time series reasoning agent that leverages LLMs for step-by-step iteratively reasoning while delegating quantitative analysis to explicit time-series tools. By separating reasoning from understanding, the agent avoids the limitations of black-box LLM approaches, maintains verifiable evidence through an explicit log, and enforces correctness via a critic and quality gate. Experiments show that our agent achieves performance comparable to strong baselines on time-series understanding benchmarks and delivers significant improvements on reasoning tasks, where existing models often rely on knowledge leakage rather than genuine analysis. This framework highlights a path toward interpretable, auditable, and effective LLM-based systems for reasoning over time series data.

Disclaimer

References

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

[2] Elizabeth Fons, Rachneet Kaur, Soham Palande, Zhen Zeng, Tucker Balch, Manuela Veloso, and Svitlana Vyetrenko. Evaluating large language models on time series feature understanding: A comprehensive taxonomy and benchmark. *arXiv preprint arXiv:2404.16563*, 2024.

[3] Ben D Fulcher and Nick S Jones. Hctsa: A computational framework for automated time-series phenotyping using massive feature extraction. *Cell Systems*, 5(5):527–531, 2017.

[4] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36:19622–19635, 2023.

[5] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time-series classification: A review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

[6] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[7] 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. *arXiv preprint arXiv:2503.01875*, 2025.

[8] Yaxuan Kong, Yiyuan Yang, Shiyu Wang, Chenghao Liu, Yuxuan Liang, Ming Jin, Stefan Zohren, Dan Pei, Yan Liu, and Qingsong Wen. Position: Empowering time series reasoning with multimodal llms. *arXiv preprint arXiv:2502.01477*, 2025.

[9] 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.

[10] Mike A Merrill, Mingtian Tan, Vinayak Gupta, Tom Hartvigsen, and Tim Althoff. Language models still struggle to zero-shot reason about time series. *arXiv preprint arXiv:2404.11757*, 2024.

[11] 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.

[12] Robert H Shumway and David S Stoffer. *Time Series Analysis and Its Applications*. Springer, 2017.

[13] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

[14] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, et al. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[15] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, et al. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[16] Qingsong Wen, Liang Sun, Fan Yang, Xue Wang Song, Jianmin Gao, Xian Wang, and Huan Xu. A survey on time series forecasting. *arXiv preprint arXiv:2004.13408*, 2020.

[17] 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.

[18] 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.

[19] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[20] Wen Ye, Yizhou Zhang, Wei Yang, Lumingyuan Tang, Defu Cao, Jie Cai, and Yan Liu. Beyond forecasting: Compositional time series reasoning for end-to-end task execution. *arXiv preprint arXiv:2410.04047*, 2024.

[21] Xiyuan Zhang, Ranak Roy Chowdhury, Rajesh K Gupta, and Jingbo Shang. Large language models for time series: A survey. *arXiv preprint arXiv:2402.01801*, 2024.

[22] Siru Zhong, Weilin Ruan, Ming Jin, Huan Li, Qingsong Wen, and Yuxuan Liang. Time-vlm: Exploring multimodal vision-language models for augmented time series forecasting. *arXiv preprint arXiv:2502.04395*, 2025.

[23] Denny Zhou, Quoc V Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations (ICLR)*, 2023.

A Benchmark Datasets

Here we provide detailed information on the benchmark datasets used in our evaluation. We cover three recent benchmarks that target complementary aspects of time series question answering: *TimeSeriesExam* [1], the *time series feature understanding benchmark* [2], and the *time series reasoning benchmark* [10]. Together they span tasks from low-level feature detection to high-level reasoning about dynamics and causality.

TimeSeriesExam. TimeSeriesExam is designed as a comprehensive multiple-choice exam for testing time series understanding. It consists of 763 questions generated from 104 templates and refined with item response theory (IRT) to balance difficulty. The benchmark is divided into five major categories: pattern recognition, noise understanding, anomaly detection, similarity analysis, and causality analysis. Each category is further divided into subcategories such as trend recognition, stationarity detection, or Granger causality. All questions are grounded in synthetic or counterfactual series with known ground truth.

Time Series Feature Understanding Benchmark. The feature understanding benchmark focuses on more fine-grained feature identification and classification tasks. Each task type includes 200 multiple-choice questions constructed from synthetic time series with controlled attributes. The benchmark is divided into *univariate* and *multivariate* settings. Univariate tasks include detection and classification of trend, seasonality, anomalies, volatility, and structural breaks. Multivariate tasks include identifying fixed, lagged, or changing correlation structures.

Time Series Reasoning Benchmark. The reasoning benchmark proposed by Merrill et al. [10] evaluates whether LLMs can reason about time series in a zero-shot manner. It covers three families of tasks: *etiological reasoning* (choosing the most plausible generative scenario or description for a given series), *time-series question answering* (answering factual questions based on one or two input series), and *context-aided forecasting* (using auxiliary textual context in addition to the series to forecast future behavior). The dataset is large-scale, consisting of thousands of question–series pairs, and has been used to demonstrate that LLMs often succeed without even accessing the series data, raising concerns about knowledge leakage and memorization.

B Detailed Description of Time Series Tools

Below we provide concise, implementation-oriented notes on the analytical tools used by TS-Agent. Tools are grouped by category as in the main text. Each tool takes typed arguments, operates on raw numeric series, and returns structured outputs (e.g., numeric value(s), categorical label(s), index ranges), which are written to the evidence log with provenance.

Data Processing.

– `slice_series`. (`name`, `start`, `end`}). Extract a subsequence by timestamps and returns sliced series.

– `segment_series` (`name`, `k` *or* `lengths`). Partition a series into k equal segments or a user-specified list of segment lengths; returns list of segments with boundaries.

– `resample_series` (`name`, `interval` / `rate`, `method=mean`/ `sum`/ `last`). Temporal down/up-sampling with aggregation.

– `select, channel` (`names` *or* `indices`). Extract one or more channels from a multivariate series

– `normalize_series` (`name`, `method=z-score`/ `minmax`, `ref_window=None`). Normalize values globally or over a reference window.

Detection & Classification.

– `trend_classifier` (`name`, `window=None`). Classify global/segment trend as {`up`, `down`, `flat`}.

– `anomaly_classifier` (`name`, `threshold`, `window=None`). Flag anomalies; returns indices/spans and brief type label (`spike`, `dip`, `level shift`).

– `seasonality_detector` (`name`, `max_period`). Detect periodicity; returns period estimate and a strength label.

Task	Category	Example Question
TimeSeriesExam [1]		
Pattern recognition	Trend recognition	Does the series show an upward trend?
	Stationarity detection	Is the series stationary over time?
	Regime switching	Does the series switch between regimes?
	Cycle recognition	Does the series exhibit repeating cycles?
	AR/MA recognition	Which ARMA model fits the series?
Noise understanding	White noise recognition	Is the series indistinguishable from white noise?
	Red noise recognition	Does the series resemble red noise?
	Signal-to-noise ratio understanding	Is the signal-to-noise ratio high or low?
Anomaly detection	General anomaly detection	Where does an anomaly occur in the series?
Similarity analysis	Shape similarity	Which series is most similar in shape?
	Distributional similarity	Which series has a similar distribution?
Causality analysis	Granger causality	Does series A Granger-cause series B?
Feature Understanding Benchmark [2]		
Univariate detection	Trend	Does the series increase or decrease?
	Seasonality	Does the series show seasonality?
	Anomalies	Does the series contain anomalies?
	Volatility	Is the volatility high or low?
	Structural break	Is there a structural break?
Univariate classification	Trend	Which type of trend is present?
	Seasonality	Which seasonal pattern is present?
	Anomalies	Which anomaly type is present?
	Volatility	What is the volatility level?
	Structural break	What type of break occurred?
Multivariate	Fixed correlation	Are the two series correlated?
	Lagged correlation	Does one series lag the other?
	Changing correlation	Does correlation change over time?
Reasoning Benchmark [10]		
Etiological reasoning	Generative scenario	Which description best explains the series?
TS QA (one series)	Question answering	At what time does the series peak?
TS QA (two series)	Comparative reasoning	Which series reacts first to a shock?
Context-aided forecasting	Forecasting	Given context, what is the next value?

Table 4: Overview of tasks in the bechmark dataset[1, 2, 10].

– `change_point_detector` (`name`, `penalty/n_cp`). Identify structural breaks in mean/variance; returns change-point indices.

– `noise_profile` (`name`, `window=None`). Qualitative noise label (e.g., `white/red`) based on simple tests.

– `stationarity_test` (`name`, `test=adf|kpss`). Return {`stationary`, `nonstationary`} plus test statistic.

– `spike_detector` (`name`, `threshold`, `min_sep`). Classify and locate isolated spikes/dips.

Numerical Operations.

– `series_info` (`name`). Return basic metadata: length T, dimension d, channel names/indices, sampling interval (if available), missingness stats.

– `datapoint_value` (`name`, `index` *or* `timestamp`). Return the value at a specific time.

– `datarange_value` (`name`, `start`, `end`, `stat=mean|sum|max|min`). Compute a statistic over a specified window.

– `summary_stats` (`name`, `range=None`). Mean, std, min, max (optionally over a window/segment).

– `return_calc` (`name`, `t1`, `t2`, `kind=pct|diff`). Simple or percentage return between two times.

– `autocorr` (`name`, `lag`). Autocorrelation at a specified lag.

– `rolling_stat` (`name`, `stat`, `window`, `step=1`). Rolling mean/std/quantile with window size and step.

– `quantile_value` (`name`, `q`). Empirical quantile at level $q \in (0, 1)$.

– `volatility` (`name`, `window`). Windowed volatility (std of differences or returns).

Relations & Comparison.

– `corr_relation` (`name1`, `name2`, `lag=0`, `method=pearson|spearman`).Correlation (optionally lagged).

– `cross_correlation` (`name1`, `name2`, `max_lag`). Cross-correlation function and best lag.

– `dtw_distance` (`name1`, `name2`). Dynamic time warping distance; lower is more similar.

– `shape_similarity` (`name1`, `name2`, `norm=zscore`). Scale-invariant shape comparison score.

– `granger_causality` (`name1`, `name2`, `maxlag`). Test whether X_1 Granger-causes X_2; returns p-value and decision.

Custom Operator Synthesis.

– `custom_operator` (`prompt`). Generate an ad-hoc operator from a natural-language description (e.g., "volatility-adjusted moving average"); returns a callable signature and a short schema.

Note that, (1) All tools accept `name/names` to identify the target series/channel(s). (2) Many tools support an optional `window/range` to localize computation. (3) Outputs are typed and include minimal diagnostics (e.g., indices, confidence/test statistics) for the critic and quality gate.

C Prompts for LLM Modules

In this section we provide the prompts used for the two language model modules in our framework. The LLM Reasoner prompt guides the model to follow a structured reason-act-observe procedure, invoking time series tools when necessary and producing a final answer in the required format. The LLM Critic prompt specifies how the critic model evaluates each reasoning step by checking tool appropriateness, the plausibility of outputs, and the sufficiency of accumulated evidence. Both prompts are shown below.

Reasoner Prompt

You are tasked with answering the following question. You have access to a set of tools:
{tool_descs}
These tools allow you to access and analyze the time series. When producing the Final Answer, you must follow the required format and respond in plain English text (no Markdown). The final output must be written exactly as: "Final Answer: the final answer to the original input question".
Follow this reasoning format:
Question: the input question to solve
Thought: you should always think about what to do
Action: the tool to use, chosen from [{tool_names}]
Action Input: the arguments to the tool
Observation: the tool's returned result
Feedback: the critic's feedback
......(The Thought/Action/Action Input/Observation/Feedback block may be repeated until you get final answer.)
Thought: I now know the final answer
Final Answer: the final answer to the original input question
Begin!
Question: {query}

Critic Prompt

You are the critic reviewing the reasoning process. You are given the current observation and the accumulated evidence log. Your task is to assess:
(i) Tool suitability: does the chosen operator match the sub-goal and the question intent?
(ii) Output plausibility: are the observed values consistent with prior evidence and expectations?
(iii) Evidence sufficiency: does the log contain enough information to fully satisfy the question intent?
Provide concise feedback indicating whether the reasoning is appropriate, highlight potential issues, and suggest corrections if necessary.

D Case Study: Reasoning Trace on Cloudy Period Detection

To illustrate how the agent performs iterative reasoning with self-refinement, we present a representative example of its reasoning trace on a time series analysis task. The question asks the agent to identify the weeks during which solar panel output is reduced due to cloudy weather. The input consists of two month-long hourly solar output series with daily seasonality (peaks at noon, zero output at night), along with periods of cloudy weather.

The agent's initial intuition is to treat cloudy conditions as anomalies, since cloudy weather reduces output abruptly. Therefore, the first attempt is to call the `anomaly_classifier` tool. However, as shown in the first reasoning trace box, the agent misuses the anomaly detection tool by providing two time series simultaneously, which causes an error. The critic provides feedback about the expected input format, leading the agent to correct its tool usage.

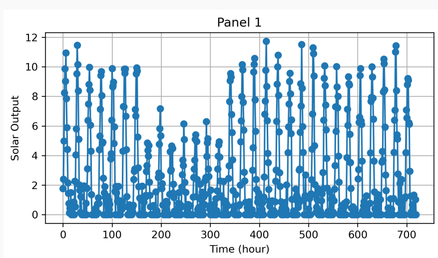
After correcting the usage, the agent applies anomaly detection separately to each series. As illustrated in the second reasoning trace box, this produces very limited and misleading results: the anomaly detector does not capture cloudy periods well because cloudy spans are continuous intervals rather than isolated anomalous points. The critic suggests that segmentation would be a more appropriate approach.

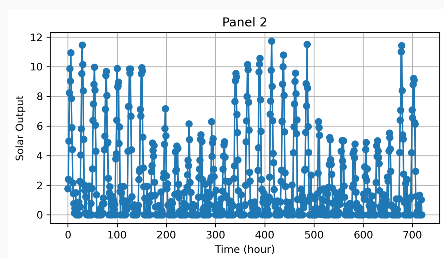
In the third reasoning trace box, the agent revises its plan: it segments each monthly time series into four weeks and compares the average outputs across weeks. This approach directly aligns with the question's intent, as cloudy weeks can be identified by sustained reductions in mean or maximum solar output. By following this refined reasoning path, the agent arrives at the correct final answer.

This case study highlights how the agent uses step-by-step reasoning and critic feedback to refine its approach: starting with an intuitive but unsuitable choice of anomaly detection, correcting tool misuse, and finally adopting segmentation as the appropriate method. Such iterative refinement enables the agent to better align its tool selection with the characteristics of the data and the requirements of the question.

Example Question

Question. You are a time series analysis expert. The time series represents hourly solar output from a panel over a month, influenced by a period of cloudy weather. Please analyze the time series features and answer the following questions:
How does the duration of cloudy periods within the month compare between the first and second time series?
Please strictly follow the output format as:
`|the first time series has cloudy periods for the (select all that apply from first, second, third, fourth) week, whereas the second time series are cloudy for the (select all that apply from first, second, third, fourth) week.|`
Then briefly explain your answer.





Self-refinement: Error Correction

```
Thought: To analyze the duration of cloudy periods in the two time
series, I first need to determine the characteristics of the time
series, particularly focusing on anomalies related to cloudy weather.
I'll start by using the anomaly detection tool to identify the
points where there are significant drops in solar output, which
typically indicates cloudy weather.

Action: anomaly_classifier, Action Input: {Solar panel 1,
Solar panel 2}

Observation: Error when calling anomaly_classifier: x must be 1-D

Feedback: Error calling anomaly_classifier, Input: {Solar panel
1, Solar panel 2}, It seems there was an error when trying to
perform anomaly detection across multiple time series. The correct
usage i {'parameters':[ {'name': 'name', 'required': True},
{'name':'threshold', 'required': False}].
```

Reasoning Trace: Corrected Tool Usage

```
Thought: I'll try performing the anomaly detection on the single
time series "Solar panel 1" to analyze the cloudy periods more
effectively.
Action: anomaly_classifier, Action Input: {Solar panel 1}
...
Observation: For Solar panel 1, no anomaly is detected. For Solar
panel 2, the anomalous points are [388].
Feedback: The anomaly detection indicated that there are no cloudy
periods detected in Solar panel 1, while Solar panel 2 has an
anomalous point. To gain a clearer understanding of how cloudy
periods compare between the two time series, it would be prudent
to segment the time series into weeks and analyze the output to
determine the presence of reduced output due to cloudy weather in
each week.
Thought: I need to divide the time series for "Solar panel 1" into
four segments to analyze the solar output during each week of the
month. This will allow me to compare the presence of cloudy periods
across the weeks.
Action: segment_series, Action Input: {Solar panel 1, 4}
...
Thought: Now that I have segmented both time series into four weeks
and have their mean solar outputs, I can compare them to assess the
presence of cloudy periods. A lower mean output during a week would
indicate cloudy weather conditions.
...
Thought: I now know the final answer.
Final Answer: the first time series has cloudy periods for the
second week, whereas the second time series are cloudy for the
second and fourth week.
```