BEYOND ISOLATED TASKS: BENCHMARKING AND AUTOMATING FULL-CYCLE TIME SERIES ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world time series analysis is a complex, multi-stage process that typically requires human experts to explore heterogeneous records, model temporal patterns, and support downstream decisions. Yet, existing time series models and benchmarks largely focus on carefully curated, isolated tasks, leaving dependencies between stages largely unexplored. We argue that reliable downstream performance requires both accurate upstream results and the ability to effectively leverage them. To address the scarcity of data for evaluating the cross-stage dependencies in full-cycle time series analysis (FC-TSA), we collect real hydrologic records across 15 gauged catchments and introduce the first benchmark for evaluating **F**ull-Cycle **A**nalysis **C**apabilities for **T**ime **S**eries: **FACTS-BENCH**. It follows the complete real-world hydrologic analysis process, containing over 1.5K scored instances across four core capabilities: (1) **Environment Exploration** prepares analysis-ready series from unfamiliar workspaces for subsequent stages; (2) **Situation Perception** identifies events, assesses anomalies, and establishes current conditions; (3) **Forecast Orchestration** builds forecasts to inform; (4) **Warning Decisions** produces evidence-backed warning and dispatch plans. Crucially, FACTS-BENCH traces how intermediate results affect downstream outputs, capturing cross-stage dependencies overlooked by existing task-specific benchmarks. To further understand cross-stage dependencies, we present **AUTOTSER**, an autonomous analyst for FC-TSA. It coordinates a Planner, Worker, and Auditor through two mechanisms: **(1) Evidence-Grounded Collaboration** checks intermediate results against source data, while **(2) Boundary-Constrained Progression** checks whether those results meet the requirements of the next analytical stage. AUTOTSER further learns from alternative operations and their downstream effects, using offline reinforcement learning to train a compact controller. This enables **(3) Lightweight Self-Evolution** without fine-tuning the underlying language models. Full-cycle evaluations demonstrate improved downstream decision quality through more reliable use of intermediate results. Further analysis shows that agent self-evolution helps retain beneficial operations and avoid harmful ones. Code: repository.

1 INTRODUCTION

Time series capture the evolving states of complex real-world systems (Ansari et al., 2024), and analyzing these states supports downstream decision-making across domains such as energy, transportation, and healthcare (Vazquez-Canteli et al., 2020; Zou et al., 2026; Choudhary et al., 2024). In practice, making decisions often requires several connected stages of analysis (Guan et al., 2026a; Nearing et al., 2024). For example, to prepare a flood warning, analysts first align rainfall and river measurements, then assess whether an observed rise indicates a developing flood or an observation error. Next, they use rainfall forecasts to predict how the event may evolve. Finally, they interpret the predicted conditions against operating rules to recommend a response (Yang et al., 2026). In this multi-stage process, an incorrect interpretation of the observations can affect both the forecast and the resulting decision. We refer to the complete analytical process needed to address an overall time series problem as full-cycle time series analysis (**FC-TSA**). A key capability in this process is handling **cross-stage dependencies**, where later stages rely on intermediate results produced by earlier ones. This requires both accurate intermediate results and valid use of those results in subsequent stages.

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.

Recent LLM agents have demonstrated the ability to plan and execute complex, multi-stage tasks (Lu et al., 2026; Dong et al., 2026). Corresponding benchmarks evaluate these capabilities through end-to-end scientific research and workflow completion (Wang et al., 2026; Su et al., 2026). In contrast, time series evaluation remains largely centered on isolated tasks, such as forecasting and anomaly detection, or single-turn question answering (Feng et al., 2025; Aksu et al., 2024; Guan et al., 2026b). These settings provide limited support for studying how intermediate analytical results affect downstream decisions. Recent benchmarks extend evaluation to multi-turn reasoning and evolving observations (Kong et al., 2026; Yao et al., 2026), but do not explicitly evaluate dependencies across analytical stages leading to those decisions. Meanwhile, time series agents primarily use planning, tools, and iterative refinement to improve individual tasks, such as producing a forecast, rather than to connect successive analytical tasks toward an overall objective (Tao et al., 2026a; Pan et al., 2026; Tao et al., 2026c). Together, these gaps leave FC-TSA and its cross-stage dependencies underexplored in both evaluation and model design.

The lack of evaluation that explicitly evaluates cross-stage dependencies has driven us to introduce the first systematic benchmark for evaluating **F**ull-Cycle **A**nalysis **C**apabilities for **T**ime **S**eries: **FACTS-BENCH**. We collect real-world hydrologic records across 15 gauged catchments and construct over 1.5K scored instances through source-specific data processing, event and anomaly annotation, reconstruction of operational decisions, and expert review. FACTS-BENCH covers four core capabilities within a full-cycle hydrologic analysis process. Agents begin with **(C1) Environment Exploration** to construct analysis-ready series from unfamiliar workspaces, then use these records in **(C2) Situation Perception** to identify events, assess anomalies, and establish current conditions. This assessment informs **(C3) Forecast Orchestration**, where agents select, combine, or develop models to predict how conditions may evolve. The resulting forecasts feed into **(C4) Warning Decisions**, where they are interpreted alongside operating conditions and rules to produce evidence-backed warning and dispatch plans. Throughout this workflow, intermediate results serve as both evaluation targets and inputs to subsequent stages. By evaluating these results against verified references and comparing downstream outcomes under reference and agent-produced inputs, FACTS-BENCH traces how intermediate analyses affect subsequent forecasts and final decisions.

Building on this data foundation, we investigate two questions. *(1) Can explicitly verifying intermediate results and their conditions of use improve downstream decisions?* To study this question, we present **AUTOTSER**, an autonomous analyst for FC-TSA that coordinates a Planner, Worker, and Auditor through two mechanisms. **Evidence-Grounded Collaboration** checks intermediate conclusions against source records and computed evidence, enabling agents to revise unsupported judgments. **Boundary-Constrained Progression** checks whether those results meet the requirements of subsequent stages, including temporal alignment, model applicability, and operating constraints. Together, these mechanisms make the correctness of intermediate results and the validity of their downstream use explicit parts of the analytical process. *(2) Can agents learn from past executions to better handle cross-stage dependencies?* We investigate this question through **Lightweight Self-Evolution**, which learns from the downstream effects of analytical operations. Starting from the same analytical state, AUTOTSER executes alternative operations and compares their downstream workflow outcomes. These comparisons provide an explicit learning signal based on whether an operation improves downstream results, accounting for constraint satisfaction and execution cost. We use these execution records to train a lightweight controller through offline reinforcement learning while keeping the underlying language models fixed. The controller learns when additional retrieval, verification, or model development is likely to benefit subsequent analysis and final decisions.

Our contributions lie in three aspects:

1. **New Setting and Testbed.** We reflect on existing time series evaluation and find that FC-TSA and its cross-stage dependencies remain underexplored. To address this gap, we introduce **FACTS-BENCH**. Built from real-world hydrologic records across 15 gauged catchments, it connects Environment Exploration (C1), Situation Perception (C2), Forecast Orchestration (C3), and Warning Decisions (C4) into complete analytical workflows. The benchmark systematically evaluates intermediate results and traces how their subsequent use affects downstream decisions.
2. **New Models.** We present **AUTOTSER**, an autonomous analyst that explicitly verifies intermediate results through (1) Evidence-Grounded Collaboration and checks their conditions of use across stages through (2) Boundary-Constrained Progression. We further observe that providing more execution memories does not consistently improve agent performance. We introduce

Lightweight Self-Evolution to train a compact controller using feedback on the downstream effects of alternative operations, while keeping the underlying language models fixed.

3. **Comprehensive Evaluation of Cross-Stage Dependencies.** Our experiments examine how intermediate results affect subsequent analysis and decisions, showing that AUTOTSER improves decision quality through more reliable use of these results. Further analyses reveal when verification and model development are beneficial or detrimental, and show that learning from their downstream effects helps retain beneficial operations and avoid harmful ones.

2 RELATED WORK

Agentic Time Series Benchmarks. Conventional benchmarks for time series analysis evaluate individual tasks, including forecasting (Godahewa et al., 2021; Aksu et al., 2024) and anomaly detection (Liu & Paparrizos, 2024). Question-answering benchmarks extend evaluation to temporal patterns and relationships (Cai et al., 2024; Kong et al., 2025; Jing et al., 2026), while contextual benchmarks assess how external information informs predictions and decisions (Guan et al., 2026a; Weng et al., 2026). Recent benchmarks further examine agents that conduct analyses through code, tools, and interaction, with two complementary directions. **(1) Data analysis and modeling.** These evaluations assess data processing and model development (Cai et al., 2025), tool-based reasoning over irregular observations (Chen et al., 2026), and root-cause analysis (Bendinelli et al., 2026). **(2) Interactive analysis.** Other evaluations examine how agents build on earlier analyses across user interactions and revise conclusions as new observations and reports become available (Kong et al., 2026; Yao et al., 2026). FACTS-BENCH extends this evaluation to complete tasks connecting unfamiliar data workspaces, situation analysis, model construction and validation, and operational decision support. By scoring these capabilities both separately and in connected executions, it distinguishes errors within an individual task from errors inherited from earlier results. Paired forecasting tasks further measure whether developing new models improves on using the available candidates.

Large Time Series Models and Agents. Deep learning methods model temporal dependencies for forecasting and other time series tasks (Zhou et al., 2021; Wu et al., 2023). Large-scale pretraining supports transfer across datasets (Ansari et al., 2024; Das et al., 2024; Goswami et al., 2024), while language-based approaches connect numerical sequences with contextual information and reasoning (Jin et al., 2024; Guan et al., 2026a). Building on these capabilities, time series agents use language models to plan analyses, execute tools, and revise results. Existing designs emphasize three approaches: **(1) Tool orchestration**, coordinating diagnostics, forecasting, validation, and domain procedures (Zhao et al., 2025; Zhan et al., 2026; Yan et al., 2026; Yang et al., 2026); **(2) Experience reuse**, retrieving analytical experience or distilling earlier executions to guide later tasks (Tao et al., 2026a; Liu et al., 2026); and **(3) Policy learning**, learning tool use for forecasting, anomaly detection, or simulator-based calibration (Tao et al., 2026b; Pan et al., 2026; Tao et al., 2026c; Li et al., 2026a). Our focus is on verifying and improving the intermediate conclusions that connect analysis to decisions. AUTOTSER coordinates planning, execution, and independent premise verification, preserving temporal cutoffs and domain constraints when conclusions are reused. It also compares alternative operations from the same state and uses their downstream outcomes to train a compact risk-aware controller. This enables adaptation of the agent's execution strategy without updating its underlying language models. Appendix A discusses general agent frameworks and optimization methods.

3 TASK FORMULATION

FACTS-BENCH formulates full-cycle time series analysis as an end-to-end task of chained stages, from data preparation through model development to a final decision. At each stage, an agent must follow the request, select accessible records and resources, and build on earlier results under the given constraints. Formally, a task is a tuple $\mathcal{T} = (r, \mathcal{W}, \mathcal{K}, \mathcal{B})$, where r is the taskrequest, $\mathcal{W}$ contains data records, $\mathcal{K}$ provides metadata, tools and domain knowledge, and $\mathcal{B}$ specifies background information, model, domain and authority constraints. An agent π solves the task in N steps, at each step k taking an action that follows the request under the constraints, given the history of earlier results, to yield the next result:

$$a_k \sim \pi\big(\cdot \mid h_k;\, r, \mathcal{W}, \mathcal{K}, \mathcal{B}\big), \qquad (s_{k+1},\, o_k) = \mathcal{E}(s_k, a_k). \tag{1}$$

At each step k, the agent's policy π selects an action a_k conditioned on the interaction history $h_k = (a_1, o_1, \ldots, a_{k-1}, o_{k-1})$. An action is an operation such as writing or running code, or invoking a tool. The action is executed in a Docker-based environment $\mathcal{E}$, which transitions from state s_k to s_{k+1} and returns an observation o_k. The environment state s_k includes all intermediate artifacts generated, such as processed data, trained models, and evaluation reports. The episode terminates after N steps, and the delivered outcome is extracted from the final state s_{N+1}.

4 FACTS-BENCH: A BENCHMARK FOR FULL-CYCLE TIME SERIES ANALYSIS

We introduce FACTS-BENCH, an executable benchmark for full-cycle time series analysis, built from real-world hydrologic observations, operational documents, and expert-reviewed decision-support records. This section presents its design principles, construction procedure, and evaluation protocol.

As shown in Figure 1, FACTS-BENCH evaluates agents on full-cycle time series analysis through eight subtasks, which we group into four capabilities: **(C1) Environment Exploration** asks agents to identify useful data and reconstruct station records via ① *Workspace Grounding*. **(C2) Situation Perception** covers ② *Flood Event Segmentation*, ③ *Observation Anomaly Adjudication*, and ④ *Preforecast Data and State Analysis*. Agents are tasked with locating event boundaries and peaks, determine event types, classify anomalies and select treatments, and summarize data availability and current conditions. **(C3) Forecast Orchestration** tests how agents conduct forecasting. ⑤ *Candidate Forecast Orchestration* requires agents to select and combine supplied predictions using model descriptions and validation records. ⑥ *Forecast Model Development* tasks agents with developing a forecasting model with training and validation data. Both task output require a forecast with an uncertainty interval and event probability. **(C4) Warning Decisions** asks agents to recommend warning levels and engineering operations in ⑦ *Warning and Dispatch Planning*, and assess the support for decision statements in ⑧ *Decision Dependency Verification*. Plans specify the affected asset, operation, operating state, and approval requirement. Each subtask can be evaluated independently or as part of a larger analysis task in which intermediate results are passed forward.

4.1 DESIGN PRINCIPLES

FACTS-BENCH follows three design principles. Tasks are heterogeneous, retaining differences in source formats, temporal resolutions, variable roles, and completeness, and combining observations with model resources and operating information. Tasks are also sequential, as source selection determines the records analyzed, situation analysis guides subsequent modeling, and forecasts inform decisions; intermediate outputs preserve these dependencies. Finally, tasks are verifiable, since source-derived series, computed facts, observed outcomes, and expert annotations provide references for scoring and for identifying ground-truth errors.

4.2 BENCHMARK CONSTRUCTION

Stage 1: Data Collection. We first gather hydrological records that cover diverse flood conditions and the corresponding decisions. Our primary source comprises seven Chinese catchments that differ in rainfall regime, terrain, and degree of regulation. Each provides hydrometeorological observations together with flood-response records, including engineering operations and operating rules. And we complement them with eight CAMELS-LUX catchments (Nijzink et al., 2025), which add multi-year observations, meteorological forcing, and catchment attributes. Then, data names are replaced with consistent codes. This stage yields an anonymized raw data pool spanning 15 catchments.

Stage 2: Processing and Annotation. We then convert the heterogeneous files into consistently labeled records. To unify the data schema, we reconcile variable names, units, timestamps, and missing-value conventions across sources. We then annotate flood boundaries and types, separate recording errors from genuine extremes, and link decisions to observations by visual inspection of the plots. This stage yields harmonized series with event, anomaly, and decision annotations.

Stage 3: Task Construction. We convert the annotated records into task instances for the eight subtasks. Grounding instances are station-level requests with paired raw-file and profile settings.

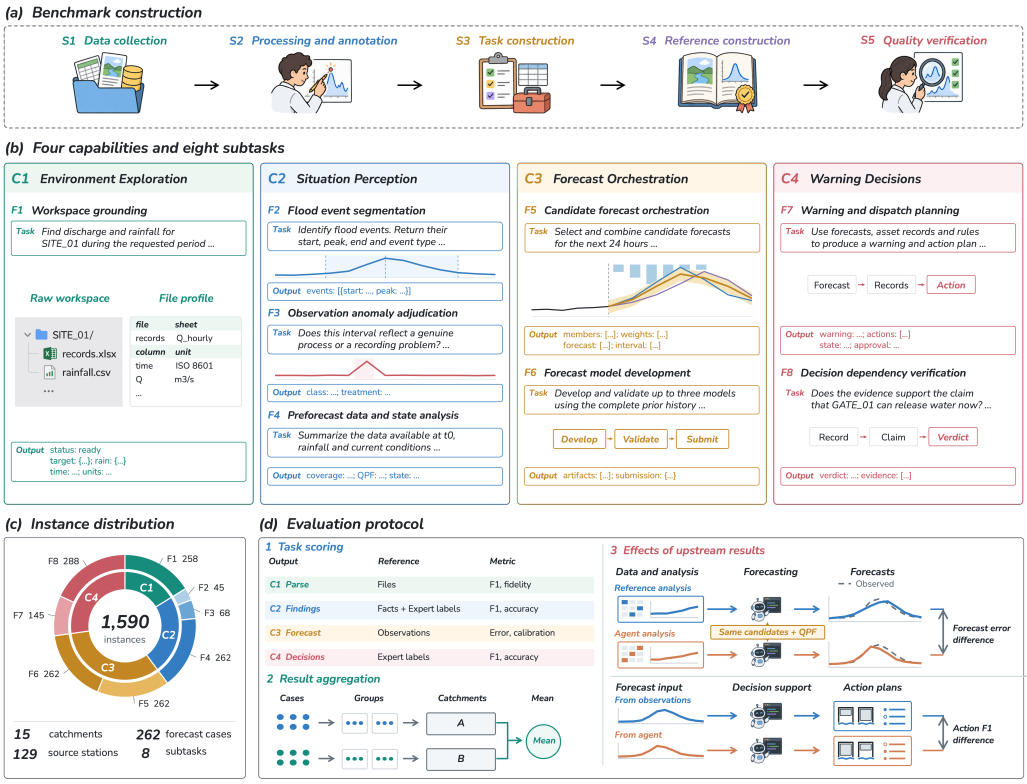


Figure 1: Overview of FACTS-BENCH. Construction combines operational records, human annotation, and reference verification. Short cases illustrate the inputs and outputs of eight subtasks under four capabilities. Evaluation scores individual results and their use in complete analysis tasks.

Event and anomaly instances provide surrounding observations and covariates. Forecasting instances comprise 262 cases spanning ordinary, rainfall-onset, and developing-flood conditions, shared by preforecast analysis and both forecasting subtasks. Decision instances pair reconstructed operating situations with statements about their supporting observations and rules.

Stage 4: Reference Construction. We then build a verified reference for every task instance. We reconstruct reference series, compute data and state facts for preforecast analysis, and extract observed forecast targets. Experts verify event and anomaly annotations, classify proposed actions as necessary, acceptable, or unsupported, and assign verdicts to decision statements, while warning levels follow the applicable thresholds. We also confirm that reference forecasting pipelines execute in the task environment. This stage yields a reference answer paired with each instance.

Stage 5: Quality Control and Verification. Finally, we check that instances, references, and scoring are correct. Reviewers inspect observations, source records, proposed answers, and operating clauses in a common interface, and we incorporate their corrections and signed judgments. We further check numerical consistency and program execution, and verify output scoring on representative instances from all eight subtasks. Appendix B provides detailed specifications.

4.3 EVALUATION PROTOCOL

Here we introduce how agents perform the benchmark tasks, how to score their outputs, and how to aggregate scores across subtasks. **Task execution:** We evaluate both LLMs and agent frameworks. They receive the same task instructions, inputs, and output requirements: as text for LLMs, and through their native interfaces for agent frameworks. To limit repetitive execution without a valid submission, agent runs are capped at 40 model calls, 200 tool calls, and 30 minutes per instance. Task-specific budgets for agent configurations are provided in Appendix D.1. **Task evaluation:** Our evaluation covers eight subtasks, in each of which the outputs (e.g., ...) are compared against the

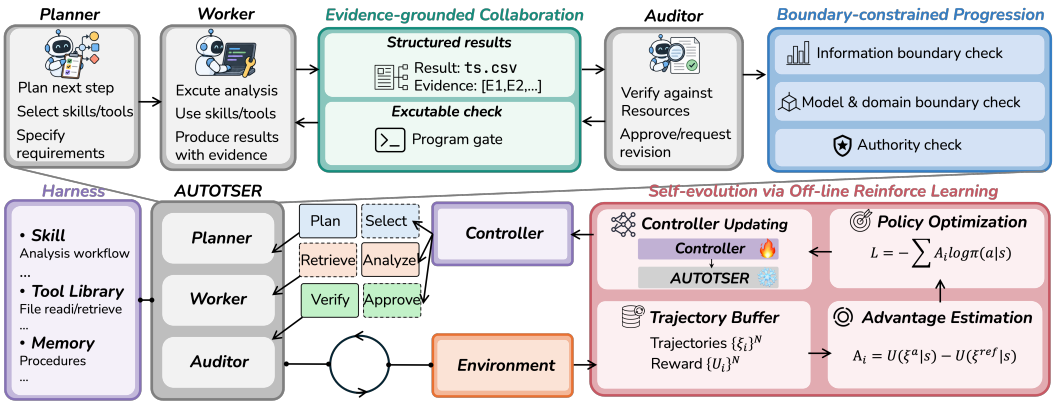


Figure 2: Overview of AUTOTSER. Planner, Worker, and Auditor coordinate analysis through shared skills, tools, and memory. Evidence-grounded collaboration and boundary-constrained progression connect intermediate results. Matched action-effect traces support offline adaptation of a lightweight controller while the language models remain frozen.

preserved ground truth (Table 5). Forecasts are scored against subsequent target observations. For paired orchestration and development cases, development gain measures the reduction in normalized forecast error. Final forecast quality, model validity, and resource use are reported separately. For multi-step analyses, the same task metrics are applied to both intermediate and final outputs. **Result aggregation:** We summarize cases within workspace, event, or decision groups, average group scores within each catchment, and then average across catchments with equal weights. Forecast errors are normalized using predefined reference scales; physical-unit errors are reported separately. Confidence intervals are estimated by resampling complete groups. Appendices C and D.1 provide metric definitions and detailed execution settings.

5 AUTOTSER: A SELF-EVOLVING MULTI-AGENT FRAMEWORK FOR FULL-CYCLE TIME SERIES ANALYSIS

We introduce AUTOTSER, a multi-agent framework for full-cycle time series analysis. AUTOTSER implements three core mechanisms: ① **evidence-grounded collaboration** to tie judgments to verifiable information, ② **boundary-constrained progression** to control the results passed to later steps, and ③ **lightweight self-evolution** to learn when additional operations improve task outcomes.

Multi-agent Framework. We organize collaboration through three agent types. The *Planner* identifies the next analytical task and selects relevant skills and tools. The *Worker* executes the analysis and proposes a judgment or plan with concise evidence references. The *Auditor* independently checks its critical premises against source records and tool results. We also introduce reusable analytical procedures and domain knowledge as task-specific skills. Agents load the selected skills to guide analysis and tool use. Tool specifications define the required inputs and supported operations, while task constraints specify data availability, temporal coverage, and operational limits. These components support three mechanisms for verifying intermediate results, preserving their conditions of use, and learning from execution experience.

Evidence-grounded collaboration. Decision-making on time series is sensitive to the choice of variables, observation conditions, and time intervals. In our multi-agent framework, the *Worker* submits a result, the *Auditor* checks it against the underlying data, and the *Worker* revises it accordingly. To ensure that evidence is conveyed correctly through this loop, each submission must follow a fixed output format and cite its sources, which we enforce by specifying the format in the Worker's prompt and programmatically validating the returned JSON via tools provided to *Auditor*. When the Auditor disputes part of a result, it runs the code-based tool that retrieves the cited records or recomputes the relevant quantities, so the Worker revises only the contradicted part based on verifiable facts. Because the loop is task-agnostic, it transfers directly to new datasets and extends to new task types by adapting only the output format and checking rules.

Boundary-constrained progression. An intermediate result is useful only when the conditions under which it was produced still hold. AUTOTSER carries these conditions forward by attaching three types of boundaries to each intermediate result. Information boundaries specify variable roles, temporal coverage, and which future covariates are known in advance; model and domain boundaries specify tool requirements, validation conditions, and operational constraints; and authority boundaries specify the information and approvals required before an operation can proceed. At state s, these boundaries define the applicable operations $\mathcal{L}(s; \mathcal{B})$. Boundaries that can be decided by comparing recorded fields, such as timestamps, action names, and variable dimensions, are checked programmatically before an operation runs. Boundaries that depend on what a rule or measurement means, such as whether a water-level reading can establish a condition on flow, are passed to the *Auditor* as factual questions about the relevant variable, time interval, and rule clause. Since boundaries travel with each result, the *Planner* can request missing facts or revise unsupported operations.

Lightweight self-evolution. Memory enables AUTOTSER to reuse summaries of previous executions. On decision dependency verification tasks processed in the same order, access to these summaries improves some batches but degrades others (Figure 3). We therefore complement memory with a lightweight controller that learns from paired executions when optional retrieval, review, or model development is worth its cost in the current state. We build on Yi & Song (2026), who separate agent execution from learning (Luo et al., 2025): a lightweight controller selects the next operation for a frozen language model and is trained offline to imitate logged actions, weighted by how well their trajectories scored. An action can thus earn credit merely by co-occurring with high scores, whatever its own effect. AUTOTSER learns a similar controller but measures this effect directly. Starting from the same state s, it completes one branch with operation a and another with the reference action, namely the default choice, and then compares the resulting traces ξ^a and ξ^{ref}:

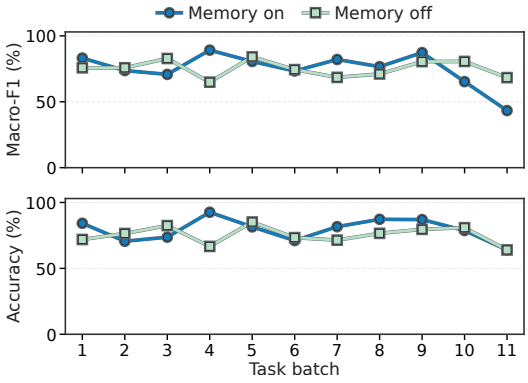


Figure 3: Memory effects on decision verification.

$$\Delta(s, a) = U(\xi^a \mid s) - U(\xi^{\text{ref}} \mid s), \tag{2}$$

where the utility U combines task quality, constraint satisfaction, and execution cost. Using these evaluated branches, we fit tree-based action-value estimators $\hat{Q}(s, a)$ by fitted Q-iteration (Ernst et al., 2005) in reverse decision order to capture delayed payoffs. For example, a developed model affects the forecast only after it is adopted and combined into an ensemble, so the target for adopting it is the fitted value of the subsequent ensemble construction, not its standalone error, crediting models that are weaker alone but improve the combined forecast. At run time, the controller takes operation a only if its estimated gain $\hat{Q}(s, a) - \hat{Q}(s, \text{ref})$ is large, consistent across the trees of $\hat{Q}$, and supported by similar training states. New executions add evaluated branches on which the controller is refit, closing the self-evolution loop without fine-tuning any language model.

6 EXPERIMENTS

Baselines. We compare Direct Prompt, Claude Code, OpenCode, TimeSeriesScientist, AION, AQUAH, and HydroAgent (Zhao et al., 2025; Zhan et al., 2026; Yan et al., 2026; Li et al., 2026a), retaining native workflows. Step-3.7-Flash is the shared backbone unless varied explicitly.

6.1 MAIN RESULTS

Setup. Tables 1–2 evaluate eight subtasks across 15 catchments with 24-hour forecasts (Section 4.3). Isolated tasks share reference inputs; orchestration and development use matched cases. Each instance runs once with reasoning enabled (Appendix D.1). **Results. Strong perception and decisions.** Self-evolved AUTOTSER leads grounding (0.949), event F1 (0.851), state accuracy (0.891), and decision subtasks (0.297/0.765); AION leads anomaly F1 (0.694). **Selective development helps.** Only AUTOTSER attains positive development gain (+0.0167). Its orchestration nMAE (2.612)

Table 1: Framework comparison. Gain: development nMAE reduction; PTE: hours; tokens: 262 forecast requests. −: unsupported. **Red**: best; blue: second best.

Method	C1: Environment Exploration	C2: Situation Perception			C3: Forecast Orchestration			C4: Warning Decisions		Cost
	Grounding Fidelity↑	Event F1↑	Anomaly F1↑	State Acc.↑	Orchestration nMAE↓	PTE↓	Development Gain↑	Plan F1↑	Verify F1↑	Tokens (M)↓
General-purpose LLM Agents										
Direct Prompt	–	0.507	0.434	0.307	2.629	6.978	–	0.241	0.696	24.267
Claude Code	0.513	0.224	0.392	0.279	2.771	6.622	-1.709	0.282	0.639	125.998
OpenCode	0.548	0.632	0.348	0.135	2.696	7.234	-1.279	0.272	0.582	150.084
Time Series Agents										
AION	0.373	0.500	**0.694**	0.555	2.641	9.400	-1.382	0.128	0.450	310.606
TimeSeriesScientist	–	–	–	–	**2.610**	5.871	-1.586	–	–	**1.256**
AQUAH	–	–	–	–	2.769	7.023	-1.601	–	–	76.282
HydroAgent (calibration)	–	–	–	–	3.267	6.498	-1.630	–	–	71.563
Ours										
AUTOTSER (self-evolved)	**0.949**	**0.851**	0.592	**0.891**	2.612	**5.833**	**0.0167**	**0.297**	**0.765**	7.242
AUTOTSER w/o Self-Evolution	**0.949**	**0.851**	0.544	0.887	2.634	5.864	-1.568	**0.297**	0.756	7.242
AUTOTSER w/o Evidence	0.687	0.561	0.578	0.890	2.634	5.864	-1.375	0.294	0.741	7.139

Table 2: Backbone comparison within AUTOTSER. Bold: best; underline: second best.

Backbone	C1: Environment Exploration	C2: Situation Perception			C3: Forecast Orchestration		C4: Warning Decisions		Cost
	Grounding Fidelity↑	Event F1↑	Anomaly F1↑	State Acc.↑	Orchestration nMAE↓	Development Gain↑	Plan F1↑	Verify F1↑	Tokens (M)↓
Step-3.7-Flash	0.949	**0.851**	**0.544**	0.887	2.634	-1.568	0.297	0.756	7.242
DeepSeek V4.1 Flash	**1.000**	0.833	0.501	0.999	2.634	**-1.396**	0.284	0.909	**6.206**
GLM-5.3	0.939	0.760	0.524	0.991	2.634	-1.532	0.295	0.845	6.239
GPT-5.6-Sol	0.806	0.706	0.476	**1.000**	2.634	-1.610	0.263	**0.980**	6.517
Opus-5	0.874	0.842	0.496	**1.000**	2.634	-1.582	0.277	0.928	7.133
Qwen3.8-27B	0.954	0.462	0.514	0.951	2.634	-1.542	0.326	0.452	7.856
Qwen3.5-122B-A3B	0.862	0.578	0.463	0.998	**2.633**	-1.547	**0.327**	0.715	9.134

approaches TimeSeriesScientist (2.610), with lower peak-time error (5.833 vs. 5.871 h). **Lower cost.** It uses 7.242M tokens versus 125.998–310.606M for other full-coverage agents. **Backbones mainly affect reasoning.** Across seven backbones, nMAE stays at 2.633–2.634, but verification F1 ranges from 0.452 to 0.980.

6.2 FULL-CYCLE ANALYSIS AND ERROR PROPAGATION

Setup. Three frameworks start from identical raw workspaces in four flood scenarios and pass their own results through C1–C4. Missing C1/C2/C4 outputs score zero; nMAE averages valid forecasts, and tokens include cached inputs. **Results. Reliable handoffs improve completion.** AUTOTSER completes all four chains, versus two/three for Claude Code/OpenCode (Table 3). It leads C2 accuracy (0.932) and action F1 (0.844), using 0.269M tokens per case. Claude Code reaches 0.959 extraction fidelity in Cases 2–3 but produces no forecast or final plan. Appendix G.1 reports each case.

Table 3: Full-cycle means (four cases). CC/OC: Claude Code/OpenCode. Bold: best; underline: second best.

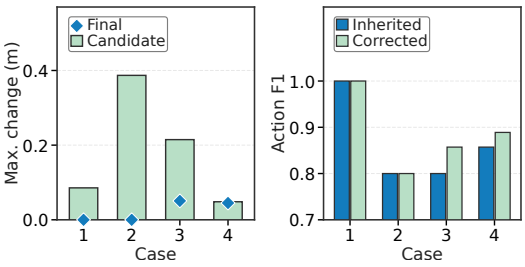
Metric	CC	OC	AUTOTSER	without Boundary
Complete↑	50%	75%	**100%**	75%
C1 Fidelity↑	**0.849**	0.609	0.797	0.797
C2 Acc.↑	0.773	0.636	**0.932**	0.705
C3 nMAE↓	44.292	9.628	**8.750**	10.724
C4 F1↑	0.415	0.604	**0.844**	**0.844**
Tokens (M)↓	3.706	4.358	**0.269**	0.272

Upstream correction. Setup. At the C1–C2 checkpoint, we replace incomplete rainfall values and source bindings with a reference parse, keeping other inputs fixed. We rerun C2–C4 and recompute

rainfall-dependent candidates under both conditions. **Results. Propagation depends on the selected models.** C1 fidelity becomes 1.0 in all cases. In Cases 1–2, selected models do not use rainfall, so downstream metrics stay unchanged. In Cases 3–4, forecasts move by 0.051/0.045 m and action F1 improves by 0.057/0.032 (Figure 4), although nMAE worsens by 0.035/0.065. Better upstream inputs can alter decisions without reducing forecast error. Details appear in Appendix G.3.

6.3 More Analysis

Ablation Studies. Setup. We remove evidence checks and self-evolution in isolated subtasks (Table 1), and boundary checks at all three full-cycle handoffs (Table 3, Appendix G.2 details the matched comparisons.). Other components and budgets stay fixed. The controller uses station-held-out evaluation on 261 valid development pairs. **Results. Evidence supports source-verifiable outputs.** Removing source checks lowers grounding from 0.949 to 0.687 and event F1 from 0.851 to 0.561. **Self-evolution improves development selection.** Disabling the controller changes gain from $+0.0167$ to -1.568. **Boundaries prevent invalid handoffs.** Without them, Case 2 omits `target_role` and loses its forecast; completion falls to 75% and C2 accuracy to 0.705. Forecasts on the three common valid cases are identical, so the mean nMAE difference reflects coverage, not forecast improvement.



(a) Forecast changes (b) Action F1

Figure 4: Rainfall correction: forecast changes and action F1 (axis starts at 0.7).

Failure Analysis. Setup. We sample 12 scored erroneous runs per framework with identical task-type composition, excluding external faults. Docent categorizes errors using answers, scores, and traces; we check labels against logs. **Results. Temporal and observational reasoning remain bottlenecks.** Data binding contributes 8.3% of AutoTSer errors, the lowest sampled share, while observation judgment and temporal reasoning jointly contribute 50% (Figure 5). One run treats a future operation as already executed. These conditional distributions describe error types, not overall failure rates.

	Data binding	Temporal reasoning	Observ. judgment	Model choice	Decision reasoning
Claude Code	25%	8%	17%	25%	25%
OpenCode	17%	0%	25%	25%	33%
AION	17%	0%	33%	25%	25%
AutoTSer	8%	17%	33%	25%	17%

Figure 5: Error types among 12 erroneous runs per framework (row percentages).

Generalization. Setup. We compare the three frameworks on 20 TimeSage-MT conversations (Kong et al., 2026), five per level, with Step-3.7-Flash and a common DeepSeek Flash analytical-quality evaluator. **Results. Gains extend to multi-turn synthesis.** AutoTSer scores 70.2 overall, versus 67.8/67.5 for Claude Code/OpenCode (Table 4). Its largest advantage is L3 grounded synthesis (66.3 vs. 58.0/57.4); OpenCode leads L4. The sampled comparison supports transfer beyond the hydrologic tasks.

Table 4: TimeSage-MT analytical quality (0–100). Five conversations per level.

Method	L1	L2	L3	L4	Mean
Claude Code	77.8	**77.7**	58.0	57.6	67.8
OpenCode	74.7	76.3	57.4	**61.6**	67.5
AutoTSer	**79.8**	76.3	**66.3**	58.5	**70.2**

7 Conclusion

We introduced FacTS-Bench and AutoTSer for full-cycle time series analysis. The benchmark connects eight subtasks through verifiable intermediate results; the framework combines evidence checks, boundary constraints, and a learned controller. Experiments show stronger perception and decision support, more complete analytical chains, and selective gains from model development. Controlled reruns reveal how upstream changes propagate through selected forecasts into decisions. Temporal interpretation and action selection remain challenges for future work.

AI USE STATEMENT

In this work, we used generative AI tools only for language polishing, proofreading, error checking, and reviewing logical consistency and clarity. We did not use them for research design, data collection, analysis, interpretation, or scientific content generation; other required disclosure items are not applicable. We also used AI tools to improve readability. All AI-assisted work was reviewed by the authors. We take responsibility for the final content of this work.

ETHICS STATEMENT

Our work focuses solely on scientific challenges and does not involve human subjects, animals, or environmentally sensitive materials. We foresee no ethical risks or conflicts of interest. We are committed to upholding the highest standards of scientific integrity and ethical conduct to ensure the validity and reliability of our findings.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we provide implementation and evaluation details (Appendix B, C and D) and a real case study for our FC-TSA (Appendix F). Our code are publicly available.

REFERENCES

Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. GIFT-Eval: A Benchmark For General Time Series Forecasting Model Evaluation, 2024. URL https://arxiv.org/abs/2410.10393.

Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, 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. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=gerNCVqqtR. Expert Certification.

Tommaso Bendinelli, Artur Dox, and Christian Holz. TraceBench: Controlled evaluation of LLM agents for time-series root-cause attribution. *arXiv preprint arXiv:2608.27182*, 2026. URL https://arxiv.org/abs/2608.27182.

Yifu Cai, Arjun Choudhry, Mononito Goswami, and Artur Dubrawski. TimeSeriesExam: A time series understanding exam. *arXiv preprint arXiv:2410.14752*, 2024. URL https://arxiv.org/abs/2410.14752.

Yifu Cai, Xinyu Li, Mononito Goswami, Michał Wiliński, Gus Welter, and Artur Dubrawski. TimeSeriesGym: A scalable benchmark for (time series) machine learning engineering agents. *arXiv preprint arXiv:2505.13291*, 2025. URL https://arxiv.org/abs/2505.13291.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/7e3767db483c942b883eb4f8cfb74e31-Abstract-Conference.html.

Sanhorn Chen, Xiaoyang Chen, Boyu Liu, and Roy Zhao. Towards verifiable agentic data science: Solving irregular TSQA via tool-grounded reasoning. *arXiv preprint arXiv:2606.15107*, 2026. URL https://arxiv.org/abs/2606.15107.

Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, Vishal Dey, Mingyi Xue, Frazier N. Baker, Benjamin Burns, Daniel Adu-Ampratwum, Xuhui Huang, Xia Ning, Song Gao, Yu Su, and

Huan Sun. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f12b4df26344f3be803c06b555252efe-Abstract-Conference.html.

Kartik Choudhary, Dhawal Gupta, and Philip S. Thomas. ICU-Sepsis: A benchmark MDP built from real medical data. In *Reinforcement Learning Conference*, 2024. URL https://arxiv.org/abs/2406.05646.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10148–10167, 2024. URL https://proceedings.mlr.press/v235/das24c.html.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, Jiajie Jin, Yutao Zhu, Hanbin Wang, Fangyu Lei, Qinyu Luo, Mingyang Chen, Zehui Chen, Jiazhan Feng, Ji-Rong Wen, and Zhicheng Dou. Agent-World: Scaling Real-World Environment Synthesis for Evolving General Agent Intelligence, 2026. URL https://arxiv.org/abs/2604.18292.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6, 2005.

Shengda Fan, Xin Cong, Yuepeng Fu, Zhong Zhang, Shuyan Zhang, Yuanwei Liu, Yesai Wu, Yankai Lin, Zhiyuan Liu, and Maosong Sun. WorkflowLLM: Enhancing workflow orchestration capability of large language models. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/3d7259031023c5aa463187c4a31c95c8-Abstract-Conference.html.

Austin Feng, Andreas Varvarigos, Ioannis Panitsas, Daniela Fernandez, Jinbiao Wei, Yuwei Guo, Jialin Chen, Ali Maatouk, Leandros Tassiulas, and Rex Ying. TelecomTS: A Multi-Modal Observability Dataset for Time Series and Language Analysis, 2025. URL https://arxiv.org/abs/2510.06063.

Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/eddea82ad2755b24c4e168c5fc2ebd40-Abstract-round2.html.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16115–16152, 2024. URL https://proceedings.mlr.press/v235/goswami24a.html.

Tong Guan, Zijie Meng, Dianqi Li, Shiyu Wang, Chao-Han Huck Yang, Qingsong Wen, Zuozhu Liu, Sabato Siniscalchi, Ming Jin, and Shirui Pan. TimeOmni-1: Incentivizing complex reasoning with time series in large language models. In *International Conference on Learning Representations*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/f7227d037ee7202106f511f78e9ecb80-Abstract-Conference.html.

Tong Guan, Sheng Pan, Johan Barthelemy, Zhao Li, Yujun Cai, Cesare Alippi, Ming Jin, and Shirui Pan. Timeomni-VL: Unified models for time series understanding and generation. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=uUnIAtZUfX.

Hoshin V. Gupta, Harald Kling, Koray K. Yilmaz, and Guillermo F. Martinez. Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling. *Journal of Hydrology*, 377(1–2):80–91, 2009. doi: 10.1016/j.jhydrol.2009.08.003.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/6507b115562bb0a305f1958ccc87355a-Abstract-Conference.html.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/36b7acf6f6010652b3f2a433774a66fe-Abstract-Conference.html.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James 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 *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/680b2a8135b9c71278a09cafb605869e-Abstract-Conference.html.

Baoyu Jing, Sanhorn Chen, Lecheng Zheng, Boyu Liu, Zihao Li, Jiaru Zou, Tianxin Wei, Zhining Liu, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Yuchen Yan, Dongqi Fu, Jingchao Ni, Jingrui He, and Hanghang Tong. TSAQA: Time series analysis question and answering benchmark. *arXiv preprint arXiv:2601.23204*, 2026. URL https://arxiv.org/abs/2601.23204.

Liqiang Jing, Zhehui Huang, Xiaoyang Wang, Wenlin Yao, Wenhao Yu, Kaixin Ma, Hongming Zhang, Xinya Du, and Dong Yu. DSBench: How far are data science agents from becoming data science experts? In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/50e9ad960ae78b741a6b4fea533f2eaf-Abstract-Conference.html.

Yaxuan Kong, Yiyuan Yang, Yoontae Hwang, Wenjie Du, Stefan Zohren, Zhangyang Wang, Ming Jin, and Qingsong Wen. Time-MQA: Time series multi-task question answering with context enhancement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29736–29753. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1437. URL https://aclanthology.org/2025.acl-long.1437/.

Yaxuan Kong, Qingren Yao, Yuqi Nie, Yichen Li, Yilei Shao, Stefan Zohren, Anna Vettoruzzo, Joaquin Vanschoren, Ming Jin, and Qingsong Wen. TimeSage-MT: A Multi-Turn Benchmark for Evaluating Agentic Time Series Reasoning, 2026. URL https://arxiv.org/abs/2606.01498.

Zhi Li, Songkun Yan, Jie Cao, Mofan Zhang, Anjiang Wei, Jinwoong Yoo, and Yang Hong. HydroAgent: Closing the gap between frontier LLMs and human experts in hydrologic model calibration via simulator-grounded RL. *arXiv preprint arXiv:2605.17792*, 2026a. URL https://arxiv.org/abs/2605.17792.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Y. Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. In *International Conference on Learning Representations*, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/52c4a736cd63b3c7bf52011a1f66f3f6-Abstract-Conference.html.

Hangchen Liu, Dongyuan Li, Renhe Jiang, Jiewen Deng, Weiwei Ye, and Yoshihide Sekimoto. TimeClaw: A time-series AI agent with exploratory execution learning. *arXiv preprint arXiv:2605.10038*, 2026. URL https://arxiv.org/abs/2605.10038.

Jie Liu, Wenxuan Wang, Zizhan Ma, Guolin Huang, Yihang Su, Kao-Jung Chang, Haoliang Li, Linlin Shen, Michael R. Lyu, and Wenting Chen. MedChain: Bridging the gap between LLM agents and clinical practice with interactive sequence. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-2498. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/6bc02900c74234aa5a0c1a7148ed933f-Abstract-Datasets_and_Benchmarks_Track.html.

Qinghua Liu and John Paparrizos. The elephant in the room: Towards a reliable time-series anomaly detection benchmark. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/c3f3c690b7a99fba16d0efd35cb83b2c-Abstract-Datasets_and_Benchmarks_Track.html.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of AI research. *Nature*, 651(8107):914–919, 2026. ISSN 1476-4687. doi: 10.1038/s41586-026-10265-5. URL https://www.nature.com/articles/s41586-026-10265-5.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent Lightning: Train ANY AI agents with reinforcement learning. *arXiv preprint arXiv:2508.03680*, 2025. URL https://arxiv.org/abs/2508.03680.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Jenia Jitsev, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-Bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/444a3737adaee10d86ad2ef5f74468e6-Abstract-Conference.html.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general ai assistants. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/25ae35b5b1738d80f1f03a8713e405ec-Paper-Conference.pdf.

J. E. Nash and J. V. Sutcliffe. River flow forecasting through conceptual models part i: A discussion of principles. *Journal of Hydrology*, 10(3):282–290, 1970. doi: 10.1016/0022-1694(70)90255-6.

Grey Nearing, Deborah Cohen, Vusumuzi Dube, Martin Gauch, Oren Gilon, Shaun Harrigan, Avinatan Hassidim, Daniel Klotz, Frederik Kratzert, Asher Metzger, Sella Nevo, Florian Pappenberger, Christel Prudhomme, Guy Shalev, Shlomo Shenzis, Tadele Yednkachw Tekalign, Dana Weitzner, and Yossi Matias. Global prediction of extreme floods in ungauged watersheds. *Nature*, 627(8004):559–563, 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-07145-1. URL https://www.nature.com/articles/s41586-024-07145-1.

Judith Nijzink, Davide Zoccatelli, Laurent Gourdol, Jean François Iffly, Ralf Loritz, and Laurent Pfister. CAMELS-LUX: Highly resolved hydro-meteorological and atmospheric data for physiographically characterized catchments around luxembourg, 2025. URL https://doi.org/10.5281/zenodo.17621594. Version 2.

Bokai Pan, Mingyue Cheng, Zhiding Liu, Shuo Yu, Xiaoyu Tao, Yuchong Wu, Qi Liu, Defu Lian, and Enhong Chen. CastFlow: Learning Role-Specialized Agentic Workflows for Time Series Forecasting, 2026. URL https://arxiv.org/abs/2604.27840.

Liangcai Su, Zhaopeng Feng, Zhuo Chen, Zhen Zhang, Xiang Lin, Ruilin Li, Handuo Zhang, Ning Wang, Kailong Wen, Yueqi Guo, Feng Xing, Yiling Guo, Chenxiong Qian, Simon Shaolei Du, Lidong Bing, and Xinyu Wang. FrontierChallenge: Evaluating scientific workflow completion. *arXiv preprint arXiv:2608.24979*, 2026. URL https://arxiv.org/abs/2608.24979.

Xiaoyu Tao, Mingyue Cheng, Ze Guo, Shuo Yu, Yaguo Liu, Qi Liu, and Shijin Wang. MemCast: Memory-Driven Time Series Forecasting with Experience-Conditioned Reasoning, 2026a. URL https://arxiv.org/abs/2602.03164.

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. *arXiv preprint arXiv:2602.13802*, 2026b. URL https://arxiv.org/abs/2602.13802.

Xiaoyu Tao, Yuchong Wu, Mingyue Cheng, Ze Guo, and Tian Gao. AnomaMind: Agentic Time Series Anomaly Detection with Tool-Augmented Reasoning, 2026c. URL https://arxiv.org/abs/2602.13807.

Jose R Vazquez-Canteli, Sourav Dey, Gregor Henze, and Zoltan Nagy. Citylearn: Standardizing research in multi-agent reinforcement learning for demand response and urban energy management, 2020. URL https://arxiv.org/abs/2012.10504.

Zhen Wang, Fan Bai, Zhongyan Luo, Jinyan Su, Kaiser Sun, Xinle Yu, Jieyuan Liu, Kun Zhou, Claire Cardie, Mark Dredze, Zhiting Hu, and Eric P. Xing. Fire-bench: Evaluating ai agents on the rediscovery of scientific insights, 2026. URL https://arxiv.org/abs/2602.02905.

Muyan Weng, Defu Cao, Wei Yang, Yashaswi Sharma, and Yan Liu. TemporalBench: A benchmark for evaluating LLM-based agents on contextual and event-informed time series tasks. *arXiv preprint arXiv:2602.13272*, 2026. URL https://arxiv.org/abs/2602.13272.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. TimesNet: Temporal 2D-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ju_Uqw384Oq.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=BAakY1hNKS.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

Songkun Yan, Mengye Chen, Zhi Li, Yixin Wen, Siyu Zhu, Mofan Zhang, Di Liu, Jie Cao, Xiaodong Chen, Chengbin Deng, Tiantian Yang, and Yang Hong. AI agent for hydrologic modeling: Definition, development, and application. *Geophysical Research Letters*, 53(13):e2025GL119814, 2026. doi: 10.1029/2025GL119814. URL https://agupubs.onlinelibrary.wiley.com/doi/full/10.1029/2025GL119814.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5a7c947568c1b1328ccc5230172e1e7c-Abstract-Conference.html.

Qingyi Yang, Siqian Qiu, Bing Li, Xu Shan, Jia Feng, Shunan Zhou, Xudong Zhou, Tiantian Xing, Jiale Guo, Xiaoyi Dong, Gaoyu Liu, Xiaohuan Liu, Haiqing Pu, Qingwen Deng, Xun Zhang,

Zhongrun Xiang, Haiyang Qian, Ying Yan, Yongkang Xu, Nuo Lei, Tianlong Jia, Baoying Shan, and Carlo De Michele. Hydroagent: Formalizing forecaster expertise into skill-orchestrated flood forecasting workflows, 2026. URL https://arxiv.org/abs/2607.23983.

Qingren Yao, Yaxuan Kong, Yuqi Nie, Yichen Li, Stefan Zohren, Anna Vettoruzzo, Qingsong Wen, Ming Jin, and Joaquin Vanschoren. TimeSage-EV: A Live Benchmark for Agentic Time Series Analysis in Evolving Environments, 2026. URL https://arxiv.org/abs/2608.14270.

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*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b126cc38b8638e07bef37e7b2bb72bf-Abstract-Conference.html.

Haiwen Yi and Xinyuan Song. Learning to control LLM agent harnesses with offline reinforcement learning. *arXiv preprint arXiv:2607.05458*, 2026. URL https://arxiv.org/abs/2607.05458.

Tianxiang Zhan, Xiaobao Song, Tong Guan, Shirui Pan, and Ming Jin. AION: Next-generation tasks and practical harness for time series. *arXiv preprint arXiv:2605.25045*, 2026. URL https://arxiv.org/abs/2605.25045.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5492ecbce4439401798dcd2c90be94cd-Abstract-Conference.html.

Haokun Zhao, Xiang Zhang, Jiaqi Wei, Yiwei Xu, Yuting He, Siqi Sun, and Chenyu You. TimeSeries-Scientist: A general-purpose AI agent for time series analysis. *arXiv preprint arXiv:2510.01538*, 2025. URL https://arxiv.org/abs/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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11106–11115, 2021. doi: 10.1609/aaai.v35i12.17325. URL https://ojs.aaai.org/index.php/AAAI/article/view/17325.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4410c0711e9154a7a2d26f9b3816d1ef-Abstract-Conference.html.

Xingchen Zou, Yuhao Yang, Zheng Chen, Xixuan Hao, Yiqi Chen, Chao Huang, and Yuxuan Liang. Traffic-R1: Reinforced LLMs bring human-like reasoning to traffic signal control systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21823–21838. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.995. URL https://aclanthology.org/2026.acl-long.995/.

A Further Related Work

A.1 General Agent Frameworks and Evaluation Benchmarks

General LLM agents combine reasoning with environment interaction to solve tasks that require several operations. Tool-using agents alternate reasoning and action, while coding agents expose files, commands, and execution results through dedicated interfaces (Yao et al., 2023; Yang et al., 2024). Multi-agent frameworks distribute work across roles and coordinate their outputs through conversation or structured procedures (Wu et al., 2024; Hong et al., 2024). These approaches provide the planning, execution, and collaboration mechanisms used in analytical agents.

Evaluation benchmarks examine different aspects of this interaction. **(1) General task completion.** Diverse environments and assistant tasks test reasoning, tool use, and information gathering (Liu et al., 2024; Mialon et al., 2024). **(2) Environment interaction.** Web, desktop, and terminal tasks evaluate whether agents can execute operations and reach verifiable outcomes (Zhou et al., 2024; Xie et al., 2024; Merrill et al., 2026). **(3) Domain and analytical tasks.** Other benchmarks assess tool use under business policies, interactive clinical decisions, machine learning engineering, and scientific data analysis (Yao et al., 2025; Liu et al., 2025; Chan et al., 2025; Jing et al., 2025; Chen et al., 2025). Full-cycle time series analysis brings these concerns to numerical records whose availability, sampling, and variable roles change across analytical steps. FacTS-Bench evaluates the resulting intermediate outputs and their contribution to downstream decisions.

A.2 Agent Optimization

Agent optimization targets different parts of an agentic system: its language model, the organization of its components, or the operations selected during execution. These targets distinguish three lines of work. **(1) Model parameter optimization.** Supervised fine-tuning and reinforcement learning improve planning and tool use by updating language model parameters (Fan et al., 2025; Li et al., 2026b). Training interfaces can also separate agent execution from reinforcement learning and assign credit to individual transitions (Luo et al., 2025). **(2) Agent structure optimization.** Search over agent programs improves prompts, component connections, and execution procedures (Zhang et al., 2025; Hu et al., 2025). **(3) Runtime control.** Offline harness learning trains an external policy to select execution actions while keeping the underlying language model fixed (Yi & Song, 2026).

AutoTSer follows the third approach and learns from matched action-effect records. Alternative operations begin from the same recorded state, and their outcomes are compared after the resulting analyses have been used downstream. These comparisons connect optional work to changes in task quality, risk, and cost. New executions can supply additional training records, allowing the framework to adapt its control policy without updating its language model executors.

B Benchmark Construction and Task Specifications

B.1 Data Sources and Coverage

The benchmark covers 15 catchments and 129 stations across multiple flood seasons. Seven Chinese catchments provide hourly rainfall, water-level, and discharge observations alongside operational records. These records include gate and pump states, engineering operations, river-network connections, operating rules, and approval requirements. Eight CAMELS-LUX catchments (Nijzink et al., 2025) provide multi-year observations, meteorological forcing, soil moisture, and catchment attributes. We anonymize catchment and station names to reduce direct answer lookup. These records yield 1,590 scored instances, with counts for each capability listed in Table 5.

B.2 Data Processing, Annotation, and Review

Data preparation combines manual inspection with source-specific processing scripts. We reconcile variable names, units, timestamps, and missing-value conventions under a common schema. Annotators inspect observation plots, covariates, and operating records to identify flood events, distinguish recording errors from genuine extremes, and link decisions to supporting evidence.

Scripts reconstruct reference series, compute data and state facts, and extract observed forecast targets. Domain experts verify event and anomaly annotations, classify candidate actions as necessary, acceptable, or unsupported, and assess decision statements against observations and operating rules. Warning levels follow the thresholds specified by the applicable rules.

Reviewers record corrections and final judgments through a common interface showing observations, source records, proposed answers, and operating clauses. Final checks cover numerical consistency, program execution, and scoring on representative instances from all eight subtasks.

B.3 Instance Construction

We use scripts to assemble processed records and reviewed annotations into task instances for the eight subtasks. Each grounding request is evaluated under two input settings: the original workspace or its directory and table descriptions. Event windows vary in the amount of context before and after annotated events. Reviewed windows without flood events provide negative examples for event segmentation. Each forecast case specifies a station, a target variable, and a forecast start time. The 262 cases cover ordinary, rainfall-onset, and developing-flood conditions with varying historical coverage. Preforecast analysis and both forecasting subtasks share these 262 forecast cases.

Decision situations combine flood histories with current observations, forecasts, engineering states, network connections, and applicable rules. Related instances share workspace, event, or decision contexts and are grouped for evaluation as described in Appendix C.

B.4 Subtask Specifications

The following specifications describe task inputs and expected responses; Appendix C defines the scoring rules.

C1 Workspace Grounding. The request specifies a station, a target variable, and an analysis period. The agent identifies source records and returns source mappings and an executable parsing plan. The executor reconstructs the requested series and compares it with the reference at matching timestamps.

C2 Flood Event Segmentation. The input contains an observation window and the associated covariates. The agent reports event presence and the start, peak, end, and type of each detected event. Predicted and reference intervals are matched one to one for event detection, timing, and type evaluation.

C2 Observation Anomaly Adjudication. The input contains a suspect interval with surrounding observations and covariates. The agent classifies the interval as a recording error or a genuine extreme and selects a treatment. Evaluation scores the classification, the treatment, and the retention of genuine extremes.

C2 Preforecast Data and State Analysis. The input contains the observation window, the variable inventory, and the quantitative precipitation forecast available at the forecast origin. The agent summarizes variable roles, sampling cadence, history length and coverage, precipitation forecast availability and totals, missingness, and the current trend and conditions. These facts are compared with computed or annotated references, with task-specific tolerances applied to numerical values.

C3 Candidate Forecast Orchestration. The input contains candidate trajectories, model descriptions, and validation records from before the forecast origin. The agent selects members and weights and submits a 24-hour forecast with an uncertainty interval and an event probability. Forecasts are evaluated against subsequent observations using trajectory, peak, interval, and probability metrics.

C3 Forecast Model Development. The input contains the full pre-origin history and a training interface that accepts up to three development plans. Agents can train models, combine executed artifacts, or decline model development. The final single-model or ensemble forecast follows the same output requirements as candidate orchestration. The paired orchestration output and score are excluded from development inputs. Artifact validity is checked by execution, and development gain compares paired forecasts on the same case.

Table 5: Capabilities, subtasks, and evaluation in FACTS-BENCH.

Capability	Instances	Subtask	Required output	Evaluation
C1 Environment Exploration	258	Workspace Grounding	Source mappings, parsing plan, and reconstructed records	Source F1, schema score, numerical fidelity
C2 Situation Perception	375	Flood Event Segmentation	Event presence, start, peak, end, and type	Event F1, temporal IoU, boundary error, type macro-F1
		Observation Anomaly Adjudication	Observation class and treatment	Class and treatment macro-F1, genuine-extreme retention
		Preforecast Data and State Analysis	Data availability, numerical summaries, and current state	Fact accuracy
C3 Forecast Orchestration	524	Candidate Forecast Orchestration	Forecast members, weights, trajectory, interval, and event probability	Trajectory and peak errors, timing error, interval score, Brier score
		Forecast Model Development	Executable models and final single-model or ensemble forecast	Forecast quality, paired development gain, artifact validity
C4 Warning Decisions	433	Warning and Dispatch Planning	Warning, asset actions, operating states, and approval requirements	Warning accuracy, action precision, recall and F1, unsupported-action rate, state accuracy, authority compliance
		Decision Dependency Verification	Statement verdicts, supporting records, and explanations	Verdict accuracy and macro-F1
Total	**1,590**	**Eight subtasks**		

C4 Warning and Dispatch Planning. The input contains the operating situation, a forecast, engineering states, the asset registry, and applicable rules. The agent returns a warning level and proposed actions, specifying the asset, operation, execution state, and approval requirement for each action. Actions are compared with necessary and acceptable reference actions, while warning levels, operating states, and authority compliance are scored separately.

C4 Decision Dependency Verification. Given an operating situation and a decision statement, the agent checks whether the supplied evidence supports the statement. It returns a supported, invalid, or unresolved verdict with the relevant source records. Reference verdicts are established through source review and numerical verification.

C Evaluation Metrics and Protocol

This appendix defines the task metrics and aggregation procedure used in Section 4.3.

C.1 Capability-Level Metrics

For subtask f, the submitted result $\hat{o}$ is evaluated against reference g through the score vector $\mathbf{q}_f = \mathrm{Eval}_f(\hat{o}, g; \Gamma_f)$, where Γ_f specifies the numerical tolerances and matching rules of the subtask. References are computed facts, observed outcomes, or expert annotations. Submission validity, output coverage, and missing responses are recorded separately from the quality of returned answers, and every metric retains its own denominator, including the number of forecasts or judgments actually compared.

Environment exploration. Source F1 compares the selected and reference sources. The schema score averages table F1, column F1, and the correctness of units, timezone, and target-variable binding. Numerical fidelity is the fraction of non-missing reference values matched within tolerance on the reference time grid, pooling target and rainfall values and counting missing or non-finite predictions as incorrect. Raw-workspace and profile-based inputs are evaluated separately on matched requests.

Situation perception. For a predicted interval $\hat{E}$ and reference interval E, temporal intersection over union is $|E \cap \hat{E}|/|E \cup \hat{E}|$. The event scorer matches intervals one to one, event F1 uses matched events as true positives and unmatched predictions and references as false positives and false negatives, and start, peak, and end timing errors are reported on matched events. Anomaly classification and treatment are scored with separate macro-F1 values. Preforecast analysis checks the requested data and state facts, and fact accuracy is the fraction agreeing with the reference under the stated tolerance for numerical facts.

Warning and decision support. Let P_i be the submitted asset-action pairs, N_i the necessary pairs, and A_i the additional acceptable pairs. Action precision, recall, and F1 use

$$\mathrm{Precision}_i = \frac{|P_i \cap (N_i \cup A_i)|}{|P_i|}, \qquad \mathrm{Recall}_i = \frac{|P_i \cap N_i|}{|N_i|}, \\ \mathrm{F1}_i = \frac{2\,\mathrm{Precision}_i\,\mathrm{Recall}_i}{\mathrm{Precision}_i + \mathrm{Recall}_i}. \tag{3}$$

Empty necessary sets have recall one, and an empty prediction has precision one only when no supported action exists. F1 is zero when both precision and recall are zero. The unsupported-action rate is $|P_i \setminus (N_i \cup A_i)|/|P_i|$ with zero for an empty prediction. Warning level, operating state, and approval requirements are scored separately. Dependency verification uses macro-F1 over the supported, invalid, and unresolved classes and reports class-specific recall.

C.2 Forecast Quality and Development Gain

Let y_t and $\hat{y}_t$ be the observation and forecast over H steps. We report

$$\begin{aligned} \mathrm{MAE} &= \frac{1}{H}\sum_{t=1}^{H}|\hat{y}_t - y_t|, \quad \mathrm{nMAE} = \frac{\mathrm{MAE}}{s_b}, \\ \mathrm{MSE} &= \frac{1}{H}\sum_{t=1}^{H}(\hat{y}_t - y_t)^2, \quad \mathrm{nMSE} = \frac{\mathrm{MSE}}{s_b^2}, \end{aligned} \tag{4}$$

where $s_b > 0$ is the predefined reference scale of catchment b computed from its development observations. NSE and KGE follow Nash & Sutcliffe (1970); Gupta et al. (2009). Normalized peak error and peak-time error are

$$\mathrm{NPE} = \frac{|\max_t \hat{y}_t - \max_t y_t|}{s_b}, \qquad \mathrm{PTE} = \big|\,\mathrm{time}(\arg\max_t \hat{y}_t) - \mathrm{time}(\arg\max_t y_t)\big|, \tag{5}$$

with PTE in hours. For a 90% interval $[l_t, u_t]$, empirical coverage is the fraction of observations inside the interval and the normalized interval score is

$$\mathrm{IS}_{90} = \frac{1}{H s_b}\sum_{t=1}^{H}\Big[(u_t - l_t) + 20(l_t - y_t)\mathbf{1}\{y_t < l_t\} + 20(y_t - u_t)\mathbf{1}\{y_t > u_t\}\Big]. \tag{6}$$

For the submitted event probability p_i, the Brier score is $(p_i - z_i)^2$ with $z_i = \mathbf{1}\{\max_t y_{it} \geq \theta_i\}$ and the threshold θ_i supplied with the task, and event detection reports probability of detection, false-alarm ratio, and critical success index.

For paired case i, development gain is $\Delta_i^{\mathrm{dev}} = \mathrm{nMAE}_i^{\mathrm{orchestration}} - \mathrm{nMAE}_i^{\mathrm{development}}$ on the same station, prediction time, horizon, and normalization scale. Positive values indicate improvement. We compute gains for cases with both forecasts and aggregate them using the group and catchment weights defined below. Paired coverage is reported separately. Artifact validity measures whether the submitted artifacts execute and satisfy the development requirements and is reported separately from forecast accuracy. The main tables report this paired reduction in forecast error. Controller training uses the utility U defined in Section 5.

C.3 Full-Cycle Evaluation and Aggregation

Isolated decision tasks supply all methods with the same reference hydrograph constructed from observations. Full-cycle evaluations pass agent-produced outputs through C1–C4 and score both intermediate results and final decisions. A complete chain produces accepted outputs at all four stages, including the final decision; conditional advisories count as completion. Missing C1, C2, or C4 outputs score zero. Forecast nMAE is averaged over valid forecasts, and chain completion is reported separately.

For a per-instance metric q_i, related cases are averaged within each workspace, event, or decision group, group scores are averaged within each catchment, and catchments are averaged with equal weights. Classification metrics pool their confusion counts within the comparison group. Confidence intervals resample complete groups, following Section 4.3.

D Implementation and Experimental Settings

D.1 Language Models and Execution Settings

The framework comparison and all additional experiments use Step-3.7-Flash. The backbone comparison within AutoTSer compares this model with DeepSeek V4.1 Flash, GLM-5.3, GPT-5.6-Sol, Opus-5, Qwen3.8-27B, and Qwen3.5-122B-A3B. The selected backbone is used for every role in AutoTSer. Self-evolution is disabled for all backbones in Table 2.

Direct Prompt uses temperature zero when this setting is supported by the model. Reasoning is enabled for all backbones, with high effort selected where configurable. Hydrologic values are

serialized to three decimal places, and scores and weights to four. Agent runs are capped at 40 model calls, 200 tool calls, and 1,800 seconds per instance. Model development accepts up to three plans through the supplied training interface. Transport errors permit retries of the identical request; format errors permit one identical resubmission.

For the task-level comparisons in Tables 1 and 2, each instance runs once in a fresh runtime without memory of preceding test instances. The read-only memory remains available unless it is ablated in the corresponding experiment. Each run records the model identifier, reasoning setting, prompt, framework and controller versions, task list, and scorer version.

D.2 Agent Execution and Boundary Checks

The Planner selects the next task and its associated skills and tools. Hydrologic skills cover tabular repair, event and anomaly perception, model development, and dispatch planning. The transfer experiment uses a general time-series skill in place of these hydrologic skills. Local tools execute parsing plans, run the forecast interface, retrieve cited records, and recompute quantities.

Each Worker request contains the task, inputs, output schema, applicable boundary, and selected skill. The Worker returns one JSON object, which is validated programmatically through the tools available to the Auditor. An Auditor request specifies the result, factual question, evidence identifiers, and tool needed for verification. Time-dependent questions also specify the interval over which the evidence is checked. The executor resolves these identifiers to source records or recomputes the requested quantities. The Worker uses the returned evidence to revise the contradicted part of its submission. Public claims and proposed operations are shared across roles, while private reasoning is not. Full prompts for each role and subtask are included in the released repository.

Each intermediate result records its time scope, variable roles, source references, and conditions for use in subsequent stages. Table 6 summarizes the checks applied before the next stage consumes that output. Recorded fields support programmatic checks; conditions requiring interpretation are passed to the Auditor for verification, as described in Section 5.

Table 6: Boundary checks at the three handoffs of the full-cycle chain.

Handoff	Checks
C1 $\rightarrow$ C2	Variable roles and units; agreement of stations and time grids.
C2 $\rightarrow$ C3	Forecast origin; coverage of historical observations and available precipitation forecasts; compatibility with model input requirements.
C3 $\rightarrow$ C4	Forecast origin and horizon; uncertainty intervals and event probabilities; operating conditions and authority for each proposed action.

D.3 Model Development and Forecast Combination

The model bank contains six candidates: persistence, trend, arima, ets, rain_ridge, and gbm. Candidate parameters are fitted on the visible pre-origin history, and backtests use three windows before the origin. Candidate orchestration reuses these artifacts without training new models for the task. Model development can train on the supplied history and combine executed model artifacts into an ensemble.

Four deterministic procedures provide reference combinations for each evaluated forecasting case. Best backtest selects the candidate with the lowest mean pre-origin error. Equal weighting averages all candidates, while inverse-error weighting assigns weights inversely proportional to their pre-origin errors. The fourth procedure uses SLSQP to fit non-negative weights that sum to one. It uses strictly pre-origin out-of-fold forecasts and a regularization term that draws weights towards equal weighting. This fit accounts for correlated model errors and allows candidates to receive zero weight. The native selection procedure conditions model membership and weights on the declared state. Reference combinations produce point forecasts; interval and probability metrics are reported for methods that submit the corresponding outputs.

D.4 Trace Collection and Offline Controller Training

The lightweight controller selects operations across all eight subtasks using the current analytical state and estimated action values. Training uses evaluated execution traces collected by the framework.

Matched branches replay the same verified execution prefix before the candidate and reference operations. The shared prefix uses identical cached messages, role routes, and generation settings. A mismatch stops collection of that pair before it enters the training dataset. Each record stores the state, action, evaluated outcomes of both branches, next state, and termination status. Branch outcomes supply the utility difference defined in Eq. 2, and records enter training after outcome evaluation.

Tree-based action-value estimators are fitted from evaluated branches in reverse decision order. For model development, the value of subsequent ensemble construction supplies the training target for model adoption. This links model adoption to the resulting ensemble forecast, as described in Section 5.

At run time, each tree estimates an operation's gain relative to the reference action. Selection uses the mean gain, a lower quantile, the fraction of negative gains, and support from similar training states. An operation must have a positive mean gain and pass the lower-quantile, negative-gain, and state-support checks. New evaluated branch records are used to refit the controller while the language models remain fixed.

Model-development gains are evaluated on 261 valid pairs using station-held-out splits.

E Additional Forecasting Results

Table 7 reports point forecasting, peak, event, and interval metrics for the framework and backbone comparisons, alongside development gain. Metric definitions follow Appendix C.

F Case Study

Task and information boundaries. Figure 6 shows a full-cycle analysis of a historical flood event in Basin QH. The user requests a 24-hour water-level forecast and dispatch advice from 08:00 on 4 July 2016. AutoTSer maps this basin-level request to control station `QH-ST-DSH`, where the warning thresholds and operating rules are defined. The task record fixes the target variable, forecast origin, and horizon for subsequent stages. This historical replay supplies observed rainfall over the forecast horizon as a known input.

Evidence-grounded forecasting. The observation and perception cards expose data coverage and review status alongside the analytical results. The available history contains 4,254 observations, with a current water level of 11.13 m after a sustained rise. Model checks require a validated conversion for candidates that predict discharge instead of the requested water level. These checks preserve the target definition as model outputs enter forecast construction. The resulting forecast reaches a peak of 11.354 m, with a reported MAE of 0.076 m.

Boundary-constrained decisions. The current water level exceeds the 8.80 m warning threshold, and the supplied 24-hour rainfall totals 77.7 mm. Using these conditions and the operating rules, AutoTSer recommends increasing discharge at `QH-GATE-WDM` and `QH-GATE-QHXH`. It also identifies `QH-LOCK-QHXH` as an additional outlet when both main gates are at full capacity. Each recommendation links an asset and an action to its supporting rules and operating conditions, including backflow checks at the two gates. These conditions remain visible at the human-confirmation step, where the operator can review and authorize the proposed actions.

Execution traces for self-evolution. The interface presents the execution trace alongside analytical states, supporting evidence, and boundary status. The development, adoption, and forecast-construction panels expose decision points and their validation context. These records preserve the context for evaluating alternative operations from the same state. Under the training protocol in Appendix D.4, matched candidate and reference branches supply supervision through their evaluated outcomes. The controller uses expected gain, downside risk, and support from similar training states for risk-aware operation selection. Further evaluated branches update this policy while the language models remain fixed.

Table 7: Additional forecasting metrics. Cov._{90} is the empirical coverage of 90% prediction intervals; gain is the development nMAE reduction. N/A: model development is unsupported. Blank cells indicate unavailable detailed results.

(a) Point forecasting (F5)

Method / Backbone	MAE↓	MSE↓ (10^5)	nMAE↓	nMSE↓	NSE↑	KGE↑
Framework comparison (Table 1)						
Direct Prompt	135.400	2.421	2.629	330.224	-11.654	-0.408
Claude Code	124.974	2.007	2.771	372.455	-9.582	-0.355
OpenCode	124.978	2.047	2.696	341.268	-11.035	-0.423
AION	124.558	2.105	2.641	375.560	-8.783	-0.255
TimeSeriesScientist	124.049	2.047	2.610	366.850	-4.436	-0.151
AQUAH	117.691	1.678	2.769	370.586	-6.374	-0.280
HydroAgent (calibration)	111.140	1.528	3.267	435.767	-7.160	-0.335
AUTOTSER w/o Self-Evolution	139.333	2.495	2.634	361.305	-4.114	-0.186
Backbone comparison (Table 2)						
Step-3.7-Flash	139.333	2.495	2.634	361.305	-4.114	-0.186
DeepSeek V4.1 Flash	139.333	2.495	2.634	361.305	-4.114	-0.186
GLM-5.3	139.333	2.495	2.634	361.305	-4.114	-0.186
GPT-5.6-Sol	139.333	2.495	2.634	361.305	-4.114	-0.186
Qwen3.8-27B	139.333	2.495	2.634	361.305	-4.114	-0.186

(b) Peaks, events, and intervals (F5) and development gain (F6)

Method / Backbone	NPE↓	PTE↓ (h)	Event MAE↓	Brier↓	Cov._{90}	IS_{90} ↓	Gain↑
Framework comparison (Table 1)							
Direct Prompt	7.290	6.978	228.827	0.204	0.402	41.596	*N/A*
Claude Code	7.591	6.622	206.329	0.161	0.530	60.389	-1.709
OpenCode	7.732	7.234	213.195	0.151	0.616	47.979	-1.279
AION	7.865	9.400	238.762	0.203	0.624	58.199	-1.382
TimeSeriesScientist	7.564	5.871	214.126	0.173	0.611	36.174	-1.586
AQUAH	7.782	7.023	177.653	0.163	0.477	42.782	-1.601
HydroAgent (calibration)	9.038	6.498	164.129	0.167	0.396	52.474	-1.630
AUTOTSER w/o Self-Evolution	7.489	5.864	233.087	0.207	0.685	58.822	-1.568
Backbone comparison (Table 2)							
Step-3.7-Flash	7.489	5.864	233.087	0.207	0.685	58.822	-1.568
DeepSeek V4.1 Flash	7.489	5.864	233.087	0.207	0.685	58.822	-1.396
GLM-5.3	7.489	5.864	233.087	0.207	0.685	58.822	-1.532
GPT-5.6-Sol	7.489	5.864	233.087	0.207	0.685	58.822	-1.610
Qwen3.8-27B	7.489	5.864	233.087	0.207	0.685	58.822	-1.542

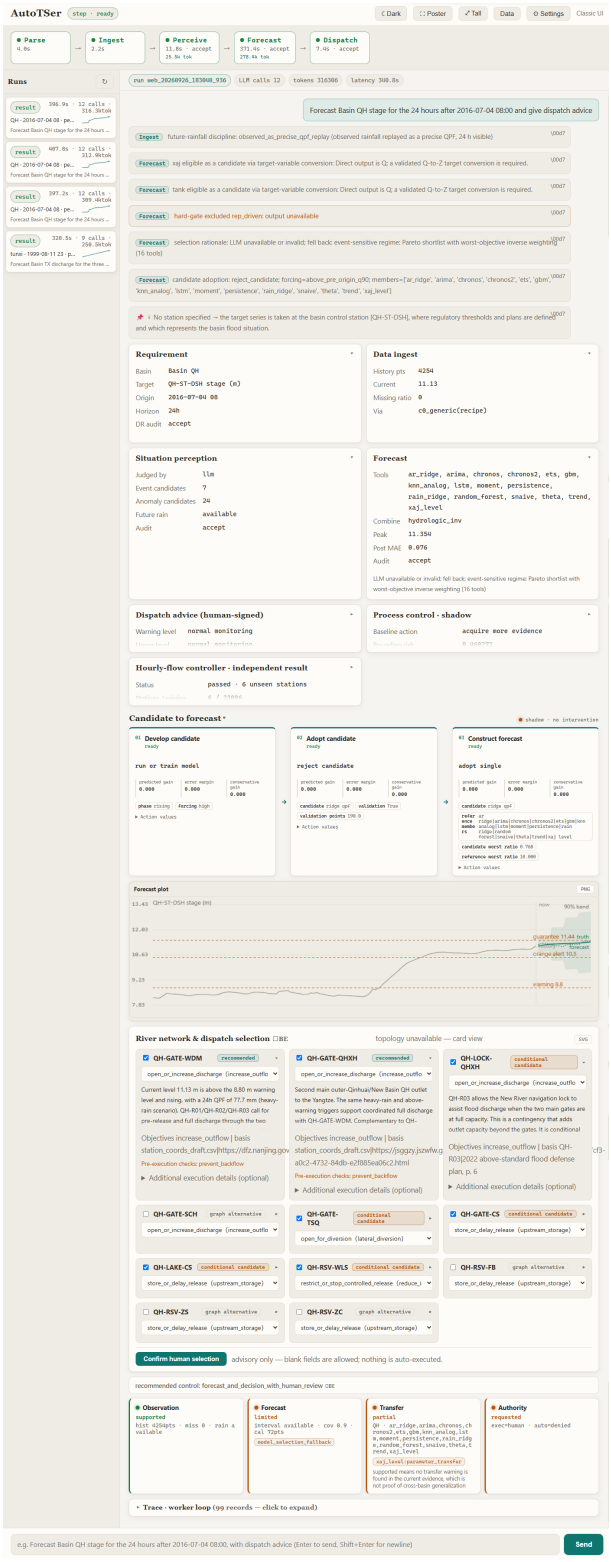


Figure 6: A full-cycle analysis with AUTOTSER, from a basin-level request to water-level forecasting and dispatch advice. The interface shows intermediate results and the rules and checks supporting each recommendation.

G Full-Cycle Evaluation and Ablation Details

G.1 Full-Cycle Evaluation

Full-cycle evaluation tests whether a framework can complete a connected analysis using its own intermediate results. Each stage passes its outputs to the next, linking data reconstruction and interpretation to forecasting and decisions. For each of four flood scenarios, Claude Code, OpenCode, and AutoTSer start from the same raw workspace and execute C1–C4. Table 8 reports stage quality, chain completion, and token usage for every case underlying the means in Table 3.

Table 8: Per-case full-cycle results underlying Table 3. Missing C1/C2/C4 outputs score zero; nMAE averages valid forecasts. Tokens include cached inputs.

Method	Case	Complete	C1 Fidelity↑	C2 Acc.↑	C3 nMAE↓	C4 F1↑	Tokens (M)↓
Claude Code	1	Yes	0.479	0.545	88.088	0.750	3.142
	2	No	0.959	0.818	–	0.000	1.878
	3	No	0.959	0.818	–	0.000	3.090
	4	Yes	1.000	0.909	0.496	0.909	6.713
	Mean	50%	**0.849**	0.773	44.292	0.415	3.706
OpenCode	1	Yes	0.479	0.727	27.881	0.800	2.553
	2	No	0.000	0.000	–	0.000	1.325
	3	Yes	1.000	1.000	0.281	0.727	3.096
	4	Yes	0.956	0.818	0.722	0.889	10.457
	Mean	75%	0.609	0.636	9.628	0.604	4.358
AutoTSer	1	Yes	0.891	0.909	31.466	1.000	0.291
	2	Yes	0.769	0.909	2.827	0.500	0.313
	3	Yes	0.725	1.000	0.211	0.875	0.212
	4	Yes	0.805	0.909	0.496	1.000	0.261
	Mean	**100%**	0.797	**0.932**	**8.750**	**0.844**	**0.269**
AutoTSer w/o Boundary	1	Yes	0.891	0.909	31.466	1.000	0.267
	2	No	0.769	0.000	–	0.500	0.376
	3	Yes	0.725	0.909	0.211	0.941	0.198
	4	Yes	0.805	1.000	0.496	1.000	0.249
	Mean	75%	0.797	0.705	10.724	**0.844**	0.272

G.2 Ablation Design

Evidence checks and self-evolution are evaluated on isolated subtasks with shared reference inputs. This setting isolates their effects on task performance from errors introduced by earlier stages. Boundary constraints are evaluated in full-cycle chains, where later stages consume agent-produced outputs. This setting tests whether intermediate results remain usable across the three handoffs.

Evidence checks. We disable source-based verification by the Auditor and keep self-evolution disabled in both variants. The comparison uses the *w/o Self-Evolution* and *w/o Evidence* rows of Table 1.

Self-evolution. We compare the self-evolved and *w/o Self-Evolution* variants across all eight subtasks in Table 1. Evidence checks remain enabled in both variants, isolating the contribution of learned operation selection.

Boundary constraints. In the comparison reported in Table 3, we disable applicability checks at the C1–C2, C2–C3, and C3–C4 handoffs. The Worker, Auditor, and source-based verification remain active throughout the chain.

The backbone, tools, skills, and execution budgets remain fixed within each ablation comparison.

G.3 Controlled Upstream Correction

The correction experiment traces how a correction to the C1 output affects forecasts and decisions through the selected models. At the C1–C2 checkpoint, we create two conditions using the original rainfall reconstruction and a reference parse. The correction updates rainfall values and their associated source bindings. Water-level history, forecast origin, precipitation forecasts, engineering observations, rules, skills, and the model bank are held fixed. We rerun C2–C4 under both conditions and recompute rainfall-dependent candidates and their validation records. Figure 4 reports these paired reruns; Table 8 reports the original full-cycle executions.

We trace the resulting changes through candidate forecasts, the selected ensemble, and the final action set. Ensemble membership remains unchanged between the two conditions in all four cases. Forecast change in Figure 4a is the maximum absolute difference over the 24 forecast steps.