Giving agents the tools to reason about time series

An engineer asks an agent: “Will we have enough capacity next week?”

The agent can find a traffic dashboard, read a launch plan and retrieve the infrastructure configuration. But answering the question requires it to connect those sources. It needs to understand how demand varies through the week, whether recent growth is unusual, what the launch might change and how much uncertainty the available history leaves.

It also needs to distinguish several different claims. Traffic increased last week. Traffic is likely to increase next week. The launch will cause an increase. Additional capacity is worth its cost. Each requires different evidence.

This is the kind of work we mean by time-series reasoning: using observations over time, relevant context and explicit analytical steps to reach a supported conclusion about a changing system.

A time series is a sequence of measurements, such as hourly requests, daily sales or readings from a sensor. Their order and timing carry information. A value of 100 can mean something very different after weeks at 20 than after weeks at 200. A spike can be routine every Monday morning and unusual on Sunday night. Missing observations can hide a change or create the appearance of one.

Reasoning about these data means working out which patterns matter for the question, choosing appropriate analyses and connecting their results to a conclusion. Forecasting is one task within that process. Others include investigating changes, identifying unusual behaviour, comparing explanations and deciding whether the evidence supports action.

Chang and colleagues’ survey of reasoning and agentic systems in time series gives this emerging field a useful structure. It distinguishes direct answers, sequential reasoning and branching processes that explore or revise alternatives. It also separates the objective of an analysis from the way it is carried out. Forecasting, explanation, decision support and generation can require different workflows. The survey emphasises that a more elaborate reasoning process still needs its correctness tested. Survey, sections 2 and 7

For an agent builder, the practical question is how to make those steps dependable enough to use.

Everyday operating decisions make this important. A retailer needs stock before demand arrives. A platform team needs capacity before traffic overwhelms it. A maintenance team needs to recognise a developing problem while there is still time to respond. These tasks combine measurements, lead times, uncertainty and the consequences of being wrong.

When an agent participates in that work, the numerical answer becomes an input to another process. A forecast for seven observations can be mistaken for seven days. An average can be replaced by the most recent value. A historical test can accidentally use a correction that was published after the date being simulated. The resulting answer may look plausible while addressing the wrong question.

The opportunity is to make careful temporal analysis accessible through the agent people already use. Doing that well requires reliable interfaces between the question, the data, the models and the evidence returned to the user.

Language models and forecasting models have complementary roles here. A language model can interpret a request, locate relevant documents, propose analyses and explain results. A specialist forecaster can estimate future values from numerical history and supported additional inputs. Neither role, on its own, establishes that the whole workflow was appropriate.

Context matters too. An announced closure or planned launch may change the outlook without appearing in the historical measurements. The *Context is Key* benchmark studies precisely this problem by pairing numerical histories with textual information needed to solve its forecasting tasks. It demonstrates why evaluating context use deserves attention alongside numerical forecasting. Context is Key

Turning a document into a defensible numerical adjustment remains a separate challenge. Knowing that a launch is planned does not establish how much traffic it will generate. An agent should expose that assumption and seek supporting evidence.

Gnomon provides an execution layer for the numerical and evidential parts of this workflow. Its current interface lets people and agents inspect time-series data, calculate observed statistics, run their chosen forecasting models, compare historical performance and retrieve the resulting evidence. The same core is accessible through Python, the command line and the Model Context Protocol, which lets an agent call these operations as tools. Gnomon overview

Consider the capacity question again. The following is an illustrative workflow, rather than a claim of a completed customer deployment.

The agent first establishes what is being measured: requests per hour, the relevant service, the timestamp convention and the planning horizon. Gnomon can inspect the supplied data and freeze a reusable snapshot. Subsequent calls can work from that same inspected version, even if the original file changes. Basic statistics can be calculated over explicit time windows, with the observation count and declared unit retained.

The agent can then discover the configured models and their supported inputs. Gnomon accepts user-provided forecasting software through a common request and result interface. That can be a local model, a statistical method or an optional remote connector such as Ephemeris. It checks supported capabilities and the alignment of returned values, timestamps and uncertainty outputs. Unsupported inputs produce an error instead of being silently discarded. Execution and provider integration

If the task calls for a model comparison, the agent can explicitly request a backtest with a baseline and a limit on evaluation work. Gnomon evaluates providers at matched historical forecast origins. Each receives the corresponding history, and the report preserves failed and unfinished evaluations alongside the scores. This gives the agent evidence for discussing model performance on the task at hand. A successful inference call alone would not supply that evidence.

Historical availability is especially important. Suppose Monday's traffic measurement was corrected on Thursday. A simulation of Tuesday's forecast should use the version available on Tuesday. Gnomon's temporal data machinery distinguishes when a measurement applies, when its source made it available and, where recorded history exists, when the local system received it. Its evaluator narrows the supplied observation history at each forecast origin. Ordinary files without availability information carry an explicit assumption; they cannot reconstruct publication delays or missing revision history. These controls also do not establish what a pretrained model saw during training. Backtesting implementation

The agent now has a firmer basis for its answer: a defined dataset, a specified horizon, actual calculations and a comparison against a reference method. It can explain which parts of the capacity outlook follow from the measurements and which depend on assumptions about the launch. The host application still needs to supply the decision policy, any additional analysis and the authority to change infrastructure.

The workflow can continue when next week arrives. With Gnomon's optional ledger, forecasts can be retained and scored against actual observations. Later corrections append new records while preserving the original predictions and earlier scores. Historical model comparisons can use

matched recorded forecasts, and the system can identify records awaiting outcomes or rescoring. This creates a basis for checking whether past recommendations rested on useful forecasts. It does not automatically retrain models or change a deployment. Ledger operations

This is where a harness can help an agent reason: it gives the agent explicit operations and inspectable results to work with. A hypothesis can lead to a calculation. A model choice can lead to a comparison. A later observation can lead to an updated assessment. The agent remains responsible for connecting that evidence to the user's question.

Gnomon's current contribution is bounded. It does not cover the whole field of time-series reasoning, establish causal explanations or prove that a provider's uncertainty is calibrated. Its repository also states that a matched evaluation of real-agent performance remains pending. The implementation supports a concrete claim about validated execution and retained evidence; whether that produces better agent decisions must be measured. Validation and limitations

We are building Gnomon around the expectation that agents will increasingly work with systems whose state keeps changing. Those agents need ways to inspect the past, test predictions and revisit their conclusions as new evidence arrives. Making those operations accessible and their results inspectable is a practical step towards agents that can make better use of time-series data.