TS-AGENTBENCH: DISENTANGLING FEATURE NECESSITY FROM ROBUSTNESS IN AGENTIC TIME-SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent benchmarks typically summarize performance with a single leaderboard score, which is useful for ranking but insufficient for assessing reliability. We introduce TS-AgentBench, a benchmark suite combining 20 real-world Kaggle competitions with a structural causal synthetic environment. Both support standard leaderboard evaluation, and their rankings strongly agree (Spearman $\rho = 0.786$), showing that the synthetic cohort provides a scalable, leakage-immune complement to real-world tasks.

Beyond ranking, TS-AgentBench introduces a three-condition diagnostic protocol – clean input, corrupted feature, and complete feature absence – to evaluate whether agents rely on useful information or are affected by irrelevant signals. We find that agents are more often harmed than helped by additional covariates, revealing reliability issues hidden by aggregate scores. These effects are difficult to detect within individual tasks because repeated runs can produce substantially different solution strategies, reflecting instability in agent decision-making rather than benchmark noise.

TS-AgentBench therefore provides both a realistic leaderboard and a diagnostic framework for understanding agent reliability beyond performance alone.

1 INTRODUCTION

Large language model (LLM) agents for time-series forecasting differ from traditional statistical and deep learning models (Liu et al., 2024; Nie et al., 2023; Oreshkin et al., 2020). They do not only output predictions: they plan, write code, run tools, and decide how to process raw tabular data (Cheng et al., 2026), often through scaffolds such as AIDE (Schmidt et al., 2024). As for other agentic systems (Liu et al., 2023; Mialon et al., 2023), these agents are usually evaluated with a single leaderboard score on held-out tasks. This is the protocol of MLE-bench (Chan et al., 2024), with 75 tasks, and TimeSeriesGym (Cai et al., 2025), with 34 tasks, and we use it for both cohorts of our benchmark (Section 3). A leaderboard score is cheap to compute and comparable across systems. It does not show which data an agent chose to use, or why it succeeded or failed.

Removing a column from a real dataset and re-running the agent gives only limited information. The informativeness of the removed column is unknown, and a real dataset does not allow us to corrupt one covariate (for example with lookahead leakage or drift) while keeping everything else fixed. A synthetic structural causal model (SCM) (Pearl, 2009; Peters et al., 2017) addresses both problems. The causal role of each covariate is known, a corruption can be applied to one feature only, generated tasks cannot appear in pretraining data, and task size and difficulty can be scaled as needed.

Synthetic control does not solve a second problem. Robustness evaluations usually corrupt a covariate at increasing intensity γ and report the degradation curve. An agent that ignores the covariate and an agent that is robust to its corruption both produce a flat curve, so the curve alone cannot separate the two.

We introduce **TS-AgentBench**, which pairs 20 real-world competition tasks with a synthetic causal environment (Figure 1). In addition to the **clean** and **corrupted** conditions, the synthetic environment provides an **absent** condition in which the covariate is removed. A hierarchical bootstrap test

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

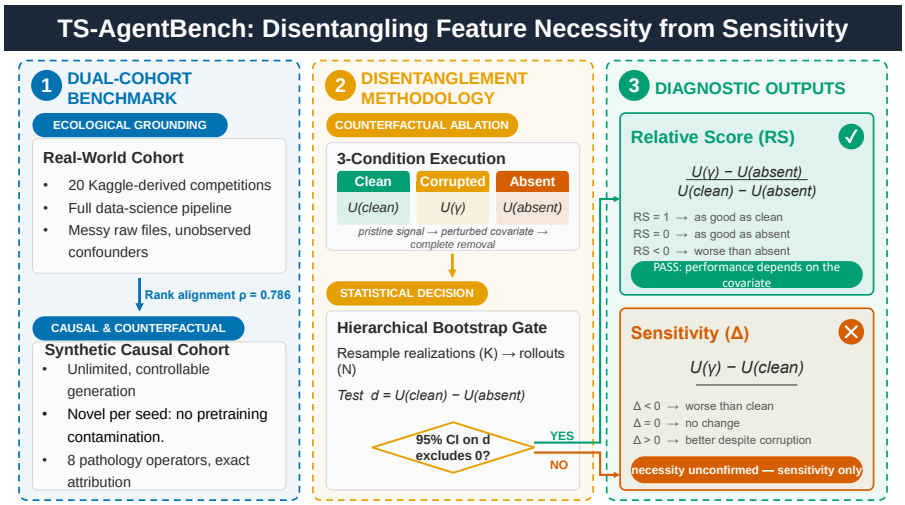


Figure 1: **TS-AgentBench overview.** Real-world and synthetic cohorts feed a three-condition protocol (clean, corrupted, absent). A hierarchical bootstrap test on d decides whether the Relative Score (gate passed) or the sensitivity curve (gate not passed) is reported.

Table 1: Comparison of TS-AgentBench with the closest benchmarks and with LOCO.

Approach	Real tasks	Synthetic causal	Agentic eval.	Absent cond.	Necessity vs. robustness
MLE-bench (Chan et al., 2024)	✓	–	✓	–	–
TimeSeriesGym (Cai et al., 2025)	✓	–	✓	–	–
CausalProfiler (Panayiotou et al., 2026)	–	✓	–	†	–
LOCO / permutation (Lei et al., 2018; Breiman, 2001)	✓	–	–	✓	–
TS-AgentBench (ours)	✓	✓	✓	✓	✓

† supports counterfactual ablation for statistical estimators, not for agents.

on the difference between clean and absent performance, which we call the necessity gate, separates whether performance depends on the covariate from how the agent reacts to its corruption. We apply this test to three operators (drift, missingness, lookahead-leakage) with K=15 seeds each, and run a coarser version on five real datasets. On the seven agent–model configurations evaluated on both cohorts, the real and synthetic rankings agree (Spearman $\rho = 0.786$, Kendall $\tau = 0.524$). We use the synthetic cohort as a diagnostic complement to the real tasks, not as a replacement.

Relation to prior work. Existing agent benchmarks (Chan et al., 2024; Cai et al., 2025) use fixed historical data and cannot run counterfactual ablations. Synthetic causal generators (Panayiotou et al., 2026) support counterfactual queries but evaluate statistical estimators, not code-generating agents. LOCO (Lei et al., 2018; Breiman, 2001) has no corrupted condition. LLM-as-a-Judge and Agent-as-a-Judge frameworks (Zheng et al., 2023; Zhuge et al., 2024) score individual traces and do not test whether an effect exceeds an agent's own run-to-run variation. Table 1 summarizes these differences, and Appendix A gives details.

Contributions. (i) A benchmark with two cohorts, 20 real-world tasks and a synthetic causal simulator, each with its own leaderboard. On seven agent–model configurations, the two rankings agree ($\rho = 0.786$, $\tau = 0.524$). (ii) A clean/corrupted/absent protocol with a hierarchical bootstrap test on the necessity effect, applied to three operators with K=15 seeds each. (iii) Evidence that agents perform worse when causally relevant covariates that end at the forecast origin are present than when they are removed: all six drift and missingness cells have a negative effect (one-sided sign test $p = 0.016$), while a pure-noise column available at forecast time has no consistent effect. (iv) A variance decomposition and a catastrophic-failure rate showing that repeated attempts at the same task vary almost as much as attempts at different tasks, and more than a deterministic baseline varies across tasks. (v) A coarse test of the necessity effect on five real datasets. Section 2 describes the

Table 2: Files in a synthetic task. "Forecast window": whether values are given for the 300 test hours.

File	Forecast window	Causal role for y
`train.csv`	–	history of y
`test.csv`	timestamps only	submission template
`observed_features.csv`	no	N sensors (parents of y via W; copy of y)
`exogenous_multivariate_macro.csv`	yes	none (OU index, 24-h signal)
`entity_metadata.csv`	static	σ_e, initial level; 2 unused columns

benchmark, Section 3 reports and compares the two leaderboards, Section 4 presents the necessity test and the instability analysis, and Section 5 reports the real-data test.

2 BENCHMARK CONSTRUCTION

Real-world cohort ($\mathcal{D}_{\text{eco}}$). Following MLE-bench and TimeSeriesGym (Chan et al., 2024; Cai et al., 2025), TSAGENTBENCH-REAL contains 20 competition-style forecasting tasks from 7 domains (including retail, energy, finance, web, and mobility), with granularities from 1 minute to 1 week. Unlike pre-processed benchmarks (Godahewa et al., 2021; Aksu et al., 2024), each task provides raw, multi-table relational files. Agents must parse irregular timestamps, join tables (for example sales logs to weather through key tables), and handle missing values. All tasks run in an isolated sandbox with a chronological 70/30 split applied to all tables, and auxiliary tables are truncated at the split date to prevent leakage. Appendix B (Table 5) gives schemas, metrics, and sandbox details.

Synthetic cohort ($\mathcal{D}_{\text{ctrl}}$). TSAGENTBENCH-SYNTH generates Kaggle-style tasks from a structural causal model (SCM). Each task is a panel of $E{=}4$ entities observed hourly for $T{=}4000$ steps. For each seed, we sample a causal graph over $N{=}4$ state variables (edges $A_{ij} \sim \text{Bernoulli}(0.35)$, signed weights $|W_{ij}| \sim \mathcal{U}(0.1, 0.6)$, rescaled so that the spectral radius of W is at most 0.9) and simulate each entity as

$$x_t = \max\Big(0.7\, x_{t-1} + 0.2\, W x_{t-1} + 0.1\,(B + S\, s_t)\, m_{r_t} + \epsilon_t,\ 10^{-3}\Big), \qquad \epsilon_t \sim \mathcal{N}(0, \sigma_e^2 I), \quad (1)$$

where B and S set the level and seasonal amplitude of the profile (energy grid, battery storage, or warehouse inventory), s_t contains daily and weekly cycles, σ_e is specific to each entity, and $m_{r_t} \in \mathbb{R}^N$ scales the input according to a latent Markov regime r_t (normal, surge, delay, failure). Since m_{r_t} scales only the input, the eigenvalues of $0.7I + 0.2W$ have modulus at most 0.88, and the recursion is stable for every seed. The target is the last state variable, $y_t = x_{t,N}$. Appendix C gives the full specification.

Task files. Each simulation is exported as a competition folder with five CSV files, joined on `entity_id` and `timestamp` (Table 2). The agent must forecast the last 300 hours of every entity, i.e., a horizon of 1 to 300 steps, and is scored by MASE on hidden values. The files differ in two ways that matter for forecasting: whether they cover the forecast window, and whether they cause the target. The sensor file contains the causal parents of y but ends at the forecast origin. The macro file covers the forecast window but has no causal link to y. A good pipeline must therefore decide which files to join and which to discard, as in real competitions. Because the generator fixes the role of every file, we know what information each file carries.

Pathology operators and absent condition. The compiler then applies one of eight pathology operators at intensity $\gamma \in [0, 1]$ (Table 7). Four change covariate or target values (lookahead leakage, trend extrapolation, drift, missingness), and four corrupt schema or timing (datetime format, resampling, asynchronous time, key noise). Each operator changes one file and is a no-op at $\gamma{=}0$. Six of the eight pathologies are detected in the real cohort (Table 8). For the three operators that act on separable covariates, we also build an **absent** condition that removes the corrupted columns: the sensor columns for drift and missingness, and the bait columns added by the lookahead operator. This condition cannot be built on fixed real data, and it is the basis of the necessity test in Section 4.

Table 3: **Matched evaluation results.** Mean within-dataset ranks (95% bootstrap CIs; lower is better). Wins count datasets beating the naive seasonal baseline (real/synthetic; $N{=}16/10$).

	Mean Rank [95% CI]		Wins
Agent + Model	**Real ($\mathcal{D}_{\text{eco}}$)**	**Synthetic ($\mathcal{D}_{\text{ctrl}}$)**	**(R / S)**
Claude Code + MiniMax M2.7	**2.38** [2.06, 3.50]	3.20 [2.10, 4.30]	14 / 6
Claude Code + DeepSeek V4 Flash	3.06 [2.06, 4.13]	**1.10** [1.00, 1.40]	13 / 9
Smolagents + GLM-5	3.88 [3.13, 5.06]	4.80 [3.80, 6.00]	15 / 2
Smolagents + Qwen3.7-Max	4.25 [3.13, 4.94]	4.30 [3.50, 5.30]	13 / 5
Smolagents + GLM-5.2	4.56 [3.56, 5.25]	3.80 [3.00, 4.70]	13 / 5
Smolagents + GLM-4.7	4.88 [3.81, 5.63]	5.70 [5.00, 6.40]	11 / 3
Smolagents + Qwen3.6-Plus	5.00 [3.94, 5.63]	5.10 [4.10, 5.70]	12 / 4

Seed depth. For the three operators with an absent condition, we generate $K{=}15$ seeds with the warehouse profile. For some (operator, model) cells, a few seeds have no valid clean or absent run because of execution failures. In these cases, the figure captions and Table 10 give the number of seeds used.

3 Real and Synthetic Leaderboards

3.1 Leaderboard Results

We evaluate ten agent–model configurations. Seven of them have a valid run on the same 16 real and 10 synthetic tasks, and we call this set the matched panel (Table 3 and Figure 2, left). Tasks where at least one configuration failed are excluded from the matched panel but still count in each configuration's own leaderboard. We report mean within-dataset ranks and the number of datasets on which each configuration beats the naive seasonal baseline.

On real data, the two Claude Code configurations have the best mean ranks (MiniMax M2.7: 2.38; DeepSeek V4 Flash: 3.06). Smolagents + GLM-5 beats the baseline most often (15/16) but has a worse mean rank (3.88). All seven configurations beat the baseline on at least 11 of the 16 real datasets. On synthetic tasks the number of wins ranges from 2 to 9 out of 10, so the synthetic cohort separates the configurations more clearly.

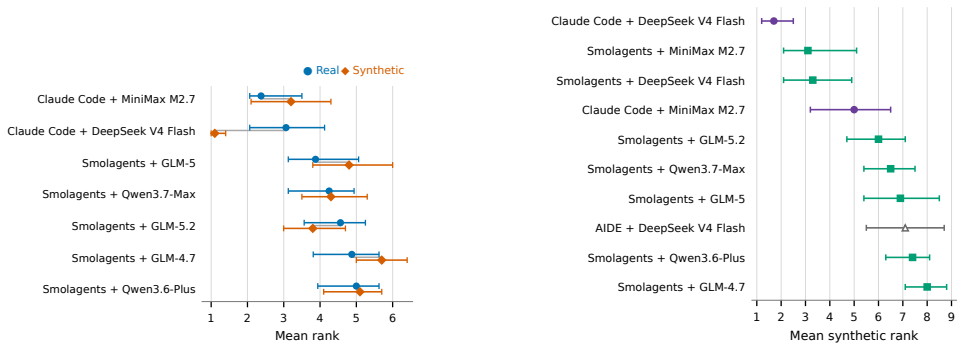
On the full synthetic leaderboard (Figure 2, right), ranks are computed among all ten configurations and therefore differ from Table 3. Claude Code + DeepSeek V4 Flash ranks first (1.70, 9/10 wins), followed by Smolagents + MiniMax M2.7 (3.10, 7/10) and Smolagents + DeepSeek V4 Flash (3.30, 8/10). The better scaffold depends on the model. With DeepSeek V4 Flash, Claude Code beats Smolagents on 7 of 10 tasks. With MiniMax M2.7, Smolagents beats Claude Code on 7 of 10 tasks. We therefore compare full agent–model pairs rather than scaffolds or models alone.

3.2 Agreement Between Cohorts

On the matched panel, the rankings agree across cohorts (Spearman $\rho = 0.786$, Kendall $\tau = 0.524$). Removing one configuration at a time gives $\rho \in [0.714, 0.829]$ and $\tau \in [0.467, 0.600]$. The same two configurations rank first and second in both cohorts, in reversed order: Claude Code + MiniMax M2.7 (1st real, 2nd synthetic) and Claude Code + DeepSeek V4 Flash (2nd real, 1st synthetic). With $n = 7$, a single swapped pair changes ρ noticeably, so we treat this agreement as preliminary evidence that the synthetic cohort preserves relative performance, not as proof of ecological validity.

3.3 Statistical Realism of the Synthetic Cohort

Rank agreement does not imply that the synthetic data looks like real data. We compare the two cohorts on distributional, temporal, and spectral statistics (Figure 3; Appendix D). The largest difference is in the tails. Real targets are heavy-tailed (median kurtosis 31.7, IQR $[1.7, 128.9]$), while synthetic targets are close to Gaussian (median 2.8, IQR $[1.1, 4.4]$). Synthetic series are also smoother: their autocorrelation decays more slowly (median of the mean absolute autocorrelation over lags



(a) Matched Real vs. Synthetic Ranks

(b) Full Synthetic Leaderboard

Figure 2: **Agent–model leaderboards.** (**a**) Matched real ($\mathcal{D}_{\text{eco}}$) and synthetic ($\mathcal{D}_{\text{ctrl}}$) rankings; lines connect identical configurations. (**b**) Synthetic leaderboard for all 10 configurations on 10 tasks; marker shapes denote scaffolds. Values are mean within-dataset ranks with 95% bootstrap CIs (lower is better).

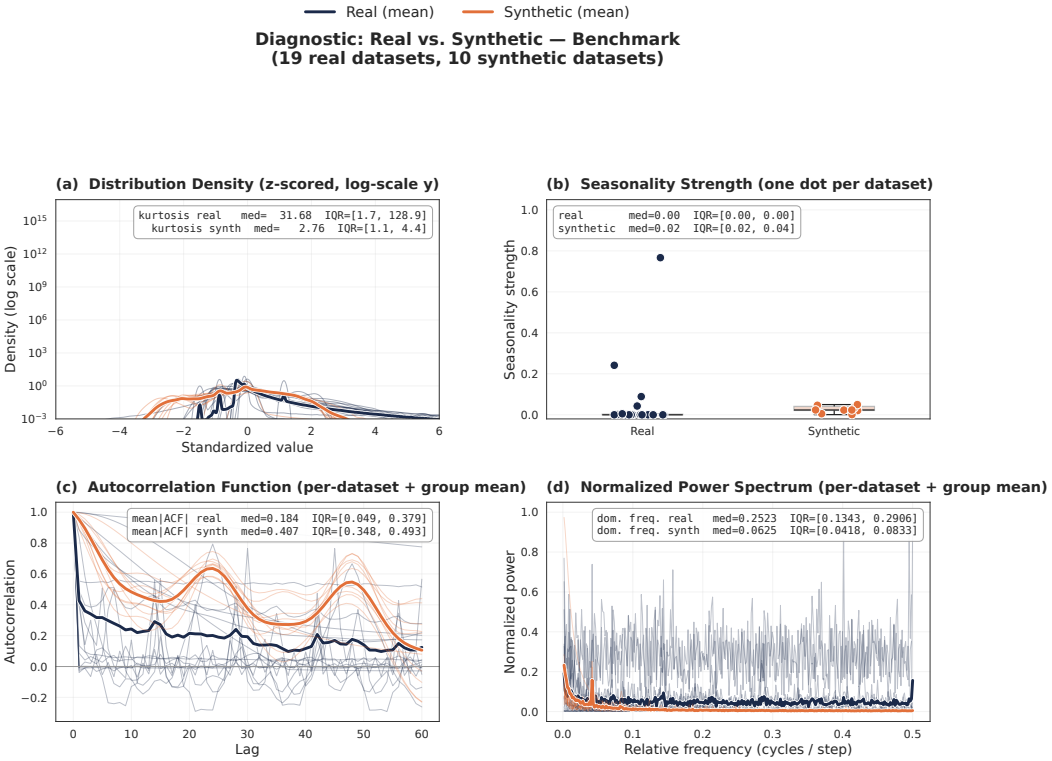


Figure 3: Statistical comparison of $\mathcal{D}_{\text{eco}}$ (n=19) and $\mathcal{D}_{\text{ctrl}}$ (n=10). Thin lines: individual datasets; bold lines: group means. (a) Z-scored target distributions (log density). (b) Seasonal strength at a common period, which does not match the natural cycle of most real datasets (Appendix D). (c) Autocorrelation decay. (d) Normalized power spectrum.

1–60: 0.41 vs. 0.18), and their spectral power is concentrated at low frequencies. Volatility clustering (ARCH statistic) is similar in the two cohorts (Table 9). The synthetic cohort is therefore a simplified environment: it is useful for controlled diagnosis, not as a copy of real data.

4 NECESSITY AND AGENT INSTABILITY

The leaderboards show how well each configuration performs, but not which information it depends on. This section defines the necessity test, applies it to one scaffold (Smolagents) with three models on the synthetic cohort, and analyzes why individual results are hard to resolve.

4.1 THE NECESSITY GATE

A robustness curve alone cannot separate an agent that ignores a covariate from an agent that is robust to its corruption. We therefore use the absent condition of Section 2. We measure utility $U = -\mathrm{MASE}$ (higher is better) in the clean, corrupted, and absent conditions, and define the necessity effect as $d = U(\text{clean}) - U(\text{absent})$. For each (operator, model) cell, we compute a 95% confidence interval on d with a hierarchical bootstrap that resamples seeds and then rollouts within each seed (Appendix G). A cell passes the necessity gate if this interval excludes zero.

What d measures. A nonzero d shows that the availability of the covariate changes the agent's performance. It does not show that the covariate is used in the final model: removing a column can also change the schema, the context length, or the features the agent tries. We inspect code-level use for three runs only (Appendix F). In this paper, "necessity" refers to this availability effect.

Sensitivity curve and Relative Score. We also report how performance changes with corruption intensity γ (Appendix E). The sensitivity curve $\Delta(\gamma) = U(\gamma) - U(0)$ measures the change relative to the clean condition and is always defined. The Relative Score $\mathrm{RS}(\gamma) = \big[U(\gamma) - U(\text{absent})\big]/d$ rescales this change by the necessity gap, so $\mathrm{RS}(0) = 1$ and $\mathrm{RS}(\gamma) = 0$ when corrupted performance equals absent performance. Because d is the denominator, RS is only stable when d is clearly different from zero, so we report it only for cells that pass the gate. No cell passes the gate in our experiments, so all curves in Appendix E show $\Delta(\gamma)$.

4.2 DIRECTIONAL EFFECT ACROSS CELLS

Figure 4 shows d for the nine (operator, model) cells. None of the nine intervals excludes zero; Section 4.3 explains why. We analyze drift and missingness separately from lookahead-leakage, because their covariates have different roles (Table 2).

In drift and missingness, the covariates are the sensor columns. They contain the causal parents of the target and a copy of the target, but they end at the forecast origin. Over a 300-step horizon they add little to the target's own history, because their influence decays by a factor of at most 0.88 per step. A pipeline that trains on their contemporaneous values has no inputs at forecast time (Appendix F, seed 0). All six cells have a negative point estimate: the agent performs worse when the sensors are present than when they are removed. A one-sided sign test against no directional bias gives $p = 0.016$. The six cells share three models and two operators, so they are not fully independent, and we read this test as a summary of a consistent pattern.

In lookahead-leakage, the bait columns cover the forecast window, and at $\gamma{=}0$ they are pure noise (Table 7). In the clean condition, d therefore measures the effect of a column that is available at forecast time but has no information. One of the three cells is negative and two are positive, and all three intervals include zero. Pooling all nine cells gives seven negative estimates (one-sided sign test $p = 0.090$). The consistent negative direction thus comes from covariates that are causally relevant but unavailable at forecast time, not from an uninformative column that is available.

4.3 AGENT-LEVEL INSTABILITY

To understand why individual cells do not resolve, we decompose the variance of performance. For every seed with several rollouts, we split the total variance into a within-seed part (differences between repeated attempts at the same task) and a between-seed part (differences between tasks), and compute the one-way intraclass correlation (ICC). If an agent solves a fixed task consistently, even imperfectly, the ICC is high. Table 4 shows ICC values between 0.000 and 0.332 in all nine cells, below the 0.50 threshold commonly used for poor reliability (Koo & Li, 2016). Repeated attempts at the same task differ almost as much as attempts at different tasks.

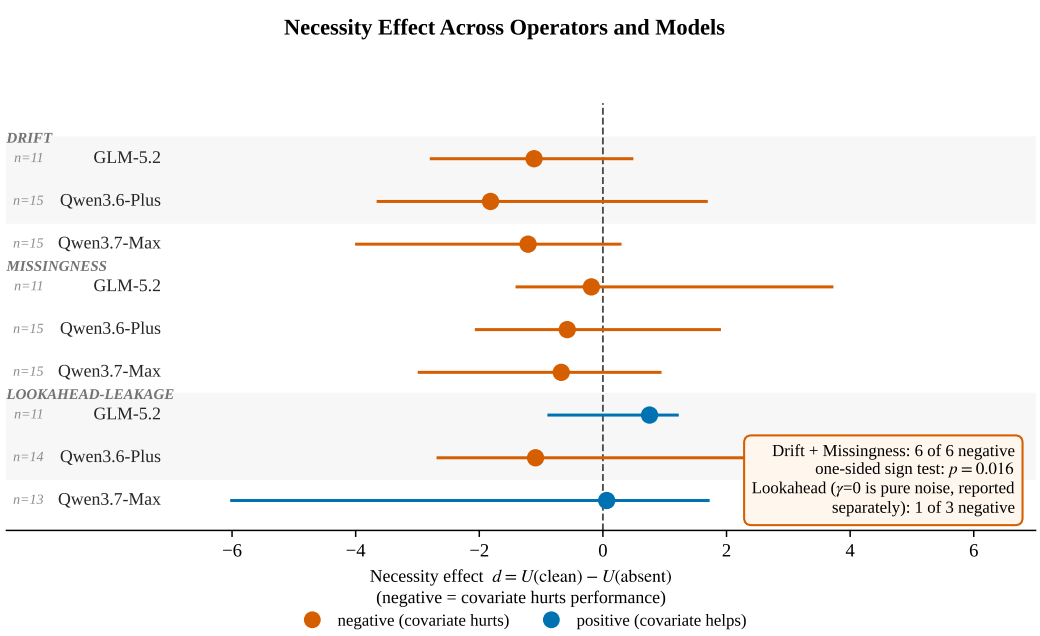


Figure 4: **Necessity effect for the nine (operator, model) cells.** Point estimates and 95% hierarchical bootstrap CIs on $d = U(\text{clean}) - U(\text{absent})$. Negative values mean the covariate hurts performance. Intervals can be asymmetric because point estimates are medians. All six drift and missingness cells are negative (one-sided sign test $p = 0.016$); lookahead-leakage is analyzed separately (Section 4.2).

Table 4: Intraclass correlation (share of performance variance due to differences between seeds rather than within seeds), clean condition. Low values mean that repeated attempts at the same task give different results.

	Drift	Missingness	Lookahead-Leakage
GLM-5.2	0.332	0.000	0.097
Qwen3.6-Plus	0.235	0.204	0.000
Qwen3.7-Max	0.000	0.000	0.299

Figure 5 (left) shows an example: rollouts of Qwen3.6-Plus on five lookahead-leakage seeds. For some seeds the rollouts are close together; for others they spread over almost the full range observed across the five seeds. Prior work on self-consistency shows that independently sampled reasoning paths for the same question often disagree (Wang et al., 2023). Here the variation concerns the whole pipeline the agent builds (features, preprocessing, and model), and it changes the final forecast error.

4.4 Deterministic Control

This variation could come from noise in the data generator. To check this, we use the deterministic baselines (Theta, SES, historical mean) that run on the same synthetic tasks. Across five seeds, their median cross-seed standard deviation is 0.80 utility units. In comparison, Qwen3.6-Plus has a pooled within-seed standard deviation of 2.49 utility units on the eleven lookahead-leakage seeds with several rollouts, after removing catastrophic rollouts (Section 4.5). In this cell, the agent varies about three times more across repeated attempts at one task than the baselines vary across different tasks. The two quantities are not measured on the same axis, since a deterministic method has no rerun variance. The comparison shows that task-to-task variation in the generator is too small to explain the agent's rerun variation.

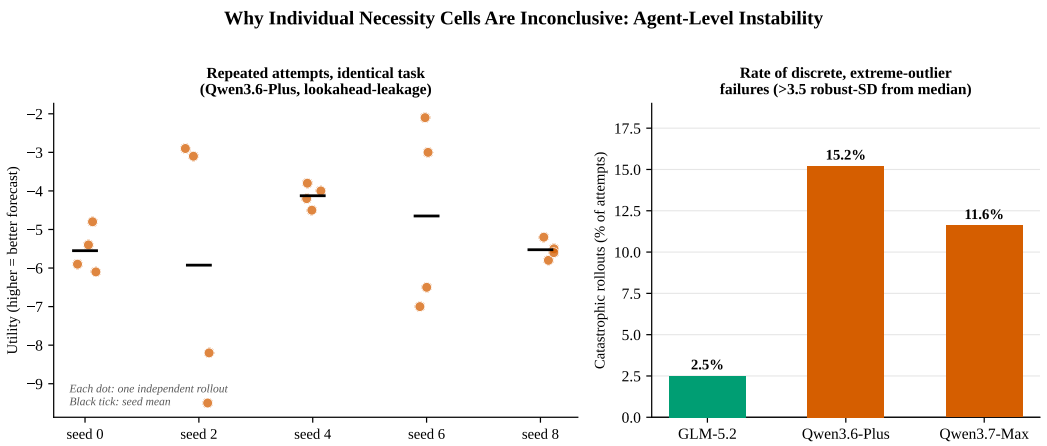


Figure 5: **Left:** rollouts of Qwen3.6-Plus on five lookahead-leakage seeds (one column per seed; each dot is one attempt at the same task). The spread within a seed is often close to the spread across seeds. **Right:** share of catastrophic rollouts per model (more than 3.5 robust SD from the cell median).

4.5 Catastrophic Rollouts

We flag a rollout as catastrophic if it lies more than 3.5 robust standard deviations from the median of its cell (Appendix G). The threshold was fixed before the analysis and is the same for all cells. Most rollouts lie close to the median, and a few are far below or above it. Figure 5 (right) gives the rate per model: 2.5% for GLM-5.2, 15.2% for Qwen3.6-Plus, and 11.6% for Qwen3.7-Max. The two Qwen models also have the widest intervals in Figure 4. Because the synthetic series are smooth and light-tailed (Section 3.3), these outliers do not come from extreme values in the input data.

We tested whether removing catastrophic rollouts makes individual cells significant. It does not: after trimming, none of the nine intervals excludes zero. Trimming moves some point estimates toward zero but changes no sign, so all six drift and missingness cells remain negative. Catastrophic rollouts therefore add to the size of the effect in Section 4.2 but do not determine its direction.

4.6 Implications for Evaluating Agentic Forecasters

Table 4 matters for anyone who wants to test whether an agentic pipeline still needs a data source. Two runs of the same test on the same task can produce different pipelines and different answers. In all nine cells, the variance between repeated attempts is larger than the variance between tasks. A single necessity test is therefore not a reliable measurement for an agent–task pair.

This has two consequences for evaluation. First, robustness claims based on one or a few seeds are underpowered and should be read with care. Second, the catastrophic-rollout rate needs only repeated rollouts and a fixed threshold, with no absent condition, so it can be added to existing agent benchmarks at low cost.

5 Real-World Necessity Replication

The results in Section 4 come from a simulator. We now test whether a similar effect appears on real data. Real datasets have no known causal structure, so we cannot remove a single covariate with a known role, and choosing one “dispensable” column would be arbitrary. We therefore test a coarser question: does any auxiliary data beyond the target history help the agent?

Design. We use five real tasks: IPBA-17, Red Hat Business, Recruit Restaurant, Rossmann Store Sales, and Store Sales (Favorita). For each, we build an absent variant that keeps only `train.csv`, `test.csv`, and the task infrastructure, and removes all auxiliary files (for example weather tables, entity metadata, and promotional calendars). We use the same hierarchical bootstrap as in Section 4,

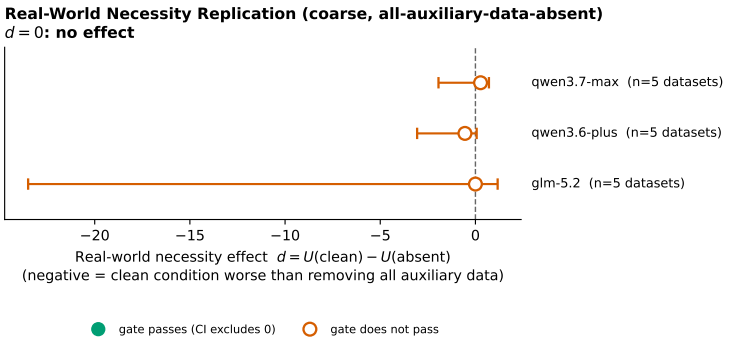


Figure 6: **Real-world necessity test, five datasets pooled per model.** Point estimates and 95% hierarchical bootstrap CIs on $d = U(\text{clean}) - U(\text{absent})$, in per-dataset standard-deviation units. Negative values mean that auxiliary data hurt performance, as in Figure 4.

with datasets as the outer level and repeated runs as the inner level. All tasks are scored with MAE (Appendix B) and converted to utility by negation. Because MAE depends on the scale of the target, we normalize each dataset's utilities by the robust center and scale of its pooled clean and absent runs before pooling. The effect d is therefore in per-dataset standard-deviation units.

Scope. This test has no corrupted condition. A nonzero d would show that some auxiliary data mattered, but not which covariate. It is a coarser question than the single-covariate test of Section 4.

Results. None of the three pooled effects excludes zero (Figure 6). Qwen3.6-Plus has a negative effect ($d = -0.55$, 95% CI $[-3.06, 0.06]$), in the same direction as the synthetic drift and missingness cells. Qwen3.7-Max has a small positive effect ($d = 0.27$, 95% CI $[-1.94, 0.71]$). GLM-5.2 has $d = 0.00$ with a very wide interval ($[-23.50, 1.17]$) because it has fewer absent-condition runs than the other two models. With five datasets, this test has much less power than the synthetic protocol. The result for Qwen3.6-Plus is consistent with the synthetic finding, but the test does not confirm it.

6 Conclusion

TS-AgentBench pairs 20 real forecasting competitions with a synthetic causal simulator. On seven configurations evaluated on both cohorts, the real and synthetic rankings agree ($\rho = 0.786$, $\tau = 0.524$). The clean/corrupted/absent protocol measures whether an agent's performance depends on a covariate, separately from its sensitivity to corruption. With three models and three operators, we find three results. First, in all six drift and missingness cells, agents perform worse when causally relevant covariates that end at the forecast origin are present than when they are removed (sign test $p = 0.016$), although no single cell is significant; a pure-noise column available at forecast time has no consistent effect. Second, repeated attempts at the same task vary almost as much as attempts at different tasks (ICC ≤ 0.33), more than a deterministic baseline varies across tasks, and 2.5–15.2% of rollouts fail catastrophically. Third, on five real datasets, one of three models shows the same negative direction, and no effect excludes zero. Because of this instability, a single necessity test on an agent–task pair is not reliable, and necessity estimates for agents need many seeds and repeated rollouts. These results give concrete targets for time-series agents, which today are mostly general-purpose ML engineering agents. A good agent should check which covariates exist at forecast time, build the same pipeline when rerun on the same task, and avoid rare collapses. TS-AgentBench measures each of these properties directly, and new tasks, operators, and seeds can be generated at low cost. We release the generator and the evaluation code so that new agents designed for time series can be tested and compared on the same protocol.

AI USE STATEMENT

In this work, we used generative AI tools for: providing feedback on and helping design research methodology (including the deterministic baseline control in Section 4.4, the outlier-trimming procedure in Section 4.5, and the cross-dataset normalization procedure in Section 5); implementing analysis methods (the statistical and figure-generation code released in Appendix G); cleaning and reformatting experimental data (ledger deduplication and coverage auditing, Appendix G); and interpreting results throughout the paper. We have not used generative AI tools to generate any of the synthetic benchmark data itself, which is produced entirely by TSAGENTBENCH-SYNTH's own structural causal simulator (Appendix C), or for translation. Formulating mathematical claims, providing critical ingredients for proofs, and assisting in the writing of proofs are not applicable to this work, which contains no formal theorems or proofs.

Additionally, we used generative AI tools for: creating and modifying scientific figures; suggesting experimental parameters (e.g., the outlier-trimming threshold and bootstrap resample count, both fixed before the corresponding analyses were run); writing and editing code; drafting portions of this paper's text; searching for, identifying, and summarizing related literature; brainstorming; editing for readability; formatting references; and suggesting paper structure and phrasing, including this paper's title. We have reviewed all AI-assisted work. All statistical code was tested against constructed cases with known ground truth before being applied to real experimental data (e.g., outlier detection and cross-dataset normalization procedures were each verified against synthetic test cases designed to reproduce the specific failure mode they address, prior to use); all reported figures were numerically cross-checked against the underlying data tables; and all citations to prior work were independently verified against their original sources rather than taken from model-recalled bibliographic detail. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses only publicly released Kaggle competition datasets (Appendix B) and does not involve new collection of human-subjects data. We are not aware of dual-use, safety, or fairness concerns specific to this benchmark: its outputs are forecasting error metrics and statistics about agent reliability, not personal or identifying information. The IPBA-17 dataset is redistributed via custom preparation scripts from a community-hosted competition (Table 5, footnote c); all other datasets are used as distributed by their Kaggle competitions, under their original licenses. We have no conflicts of interest or sponsorship to disclose.

REPRODUCIBILITY STATEMENT

Appendix G describes the full statistical procedure, including all fixed parameters (bootstrap resample count, outlier threshold, significance level), and lists the released code: benchmark generator, agent adapters, evaluation pipeline, and all analysis scripts. Appendix B (Table 5) lists the real datasets, including the five used in Section 5. Appendix C specifies the structural causal model, the pathology operators, and their validation against real data (Table 8).

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. *arXiv preprint arXiv:2410.10393*, 2024.

Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

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.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Madry. MLE-

bench: Evaluating machine learning agents on machine learning engineering. *arXiv preprint arXiv:2410.07095*, 2024.

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

Mingyue Cheng, Xiaoyu Tao, Qi Liu, Ze Guo, and Enhong Chen. Position: Beyond model-centric prediction – agentic time series forecasting. *arXiv preprint arXiv:2602.01776*, 2026.

Sarkar Snigdha Sarathi Das, Palash Goyal, Mihir Parmar, Nanyun Peng, Vishy Tirumalashetty, Chun-Liang Li, Rui Zhang, Jinsung Yoon, and Tomas Pfister. Nexus: An agentic framework for time series forecasting. *arXiv preprint arXiv:2605.14389*, 2026.

Kun Feng, Ziwei Shan, Yuchen Fang, Yiyang Tan, Sihan Lu, Shuqi Gu, Xingyu Lu, Lintao Ma, and Kan Ren. Kairosagent: Agentic time series forecasting with fused semantic reasoning, 2026. URL `https://arxiv.org/abs/2605.30002`.

Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *Neural Information Processing Systems (NeurIPS) Track on Datasets and Benchmarks*, 2021.

Terry K Koo and Mae Y Li. A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of Chiropractic Medicine*, 15(2):155–163, 2016. doi: 10.1016/j.jcm.2016.02.012.

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113 (523):1094–1111, 2018.

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. *arXiv preprint arXiv:2308.03688*, 2023.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2024.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pp. 4765–4774, 2017.

Aadyaa Maddi, Prakhar Naval, Deepti Mande, Shane Duan, Muckai Girish, and Vyas Sekar. Generating expressive and customizable evals for timeseries data analysis agents with AgentFuel. *arXiv preprint arXiv:2603.12483*, 2026.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. *arXiv preprint arXiv:2311.12983*, 2023.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2020.

Panayiotis Panayiotou, Audrey Poinsot, Alessandro Leite, Nicolas Chesneau, Marc Schoenauer, and Özgür Şimşek. CausalProfiler: Generating synthetic benchmarks for rigorous and transparent evaluation of causal machine learning. *arXiv preprint arXiv:2511.22842*, 2026.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition, 2009.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, 2017.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1135–1144, 2016.

Dominik Schmidt, Yuxiang Wu, and Zhengyao Jiang. AIDE: Human-level performance in data science competitions. `https://www.weco.ai/blog/technical-report`, 2024.

Patrick E. Shrout and Joseph L. Fleiss. Intraclass correlations: uses in assessing rater reliability. *Psychological Bulletin*, 86(2):420–428, 1979.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

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.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

Mingchen Zhuge, Changsheng Zhao, Dylan R. Ashley, Wenyi Wang, Dmitrii Khizbullin, Yunyang Xiong, Zechun Liu, Ernie Chang, Raghuraman Krishnamoorthi, Yuandong Tian, Yangyang Shi, Vikas Chandra, and Jürgen Schmidhuber. Agent-as-a-judge: Evaluate agents with agents. *arXiv preprint arXiv:2410.10934*, 2024.

A Extended Related Work

Real-world agentic ML and forecasting benchmarks. MLE-bench (Chan et al., 2024) contains 75 Kaggle machine-learning engineering competitions with human baselines from the Kaggle leaderboards; its best configuration paired the AIDE scaffold (Schmidt et al., 2024) with a frontier reasoning model. TimeSeriesGym (Cai et al., 2025) focuses on 34 time-series challenges from more than 15 domains, combining Kaggle-style and hand-designed tasks and grading code and model artifacts in addition to CSV submissions. Both use fixed historical data and therefore cannot run counterfactual ablations, which is what our synthetic cohort adds.

Agentic forecasting. Cheng et al. (2026) argue that forecasting should be treated as an agentic process built on perception, planning, action, reflection, and memory. Our necessity test measures one part of the perception component: whether the context an agent receives improves its forecast or adds noise that the agent fails to filter out.

Synthetic and causal benchmarks. Panayiotou et al. (2026) argue that synthetic generation is the only way to obtain ground-truth counterfactuals in causal machine learning, because counterfactual outcomes are never observed in real data. We apply the same idea to agent evaluation: the absent condition is a counterfactual that we can build because our tasks come from a known SCM.

Leave-one-covariate-out (LOCO). The closest existing technique to our absent condition is LOCO variable importance (Lei et al., 2018), related to permutation importance (Breiman, 2001) and feature attribution (Lundberg & Lee, 2017; Ribeiro et al., 2016). These methods compare model performance with and without a covariate. LOCO ranks features for a single fixed model, has no corrupted condition, and does not evaluate a system that decides how to use the data. Our necessity

test adds a corrupted condition and applies the comparison to agents. The coarse real-data test in Section 5 is close to a LOCO applied to all auxiliary files at once.

Agentic time-series systems. Several recent works build agentic time-series systems. TemporalBench (Weng et al., 2026) proposes a four-tier taxonomy that separates numerical forecasting from contextual and event-conditioned reasoning. TimeART (Wu et al., 2026) combines tool-augmented analysis with LLM reasoning for time-series question answering. Nexus (Das et al., 2026) splits forecasting into macro- and micro-level agentic stages with explicit context integration. KairosAgent (Feng et al., 2026) combines multimodal semantic reasoning with process-level reinforcement learning. AgentFuel (Maddi et al., 2026) generates domain-specific, stateful evaluations for conversational time-series data-analysis agents. None of these works tests whether an agent's performance depends on the auxiliary information it receives. Our necessity test and tools (Appendix G) can be applied to such systems. The survey of Chang et al. (2025) organizes time-series reasoning by reasoning topology (direct, chain, branch); our framework is orthogonal to this taxonomy, since it studies which information an agent needs rather than how it reasons.

Summary. To our knowledge, no existing benchmark combines real-world tasks, synthetic tasks that support counterfactual ablation, a statistical test that separates non-engagement from robustness, and a check of the result on real data (Table 1).

B Details on TSAgentBench-Real

Datasets. TSAgentBench-Real ($\mathcal{D}_{\text{eco}}$) contains 20 real-world competition datasets. Table 5 lists them.

Table 5: TSAgentBench-Real: the 20 real-world competition tasks, grouped by domain. All tasks are supervised prediction problems unless noted. Datasets marked † are used in the real-world necessity test (Section 5).

Dataset / Kaggle Source	Core Domain	Granularity
Favorita Grocery[b]	Retail	Daily
Store Sales (Favorita)[b,†]	Retail	Daily
M5 Forecasting	Retail	Daily
Walmart Stormy	Retail	Daily
Rossmann Store Sales[†]	Retail	Daily
Follow the Trend	Retail	Weekly
IPBA-17[c,†]	Retail	Daily
Playground S3E19	Retail	Daily
Playground S5E1	Retail	Daily
Smart Homes Temp	Energy	15-Minute
GEFCom 2012 Wind	Energy	Hourly
Energy Prosumers	Energy	Hourly
G-Research Crypto	Finance	1-Minute
Optiver Volatility	Finance	10-Minute
Web Traffic	Web Analytics	Daily
Recruit Restaurant[†]	Hospitality	Daily
Bike Sharing Demand	Urban Mobility	Hourly
Covid-19 Week 1	Health	Daily
MLB Engagement	Sports / Media	Daily
Red Hat Business[a,†]	CRM / Event	Event-based

[a] Binary classification, the only non-regression task; scored by MAE between predicted and true labels (see Metric paragraph).
[b] Same source (Favorita) at item and family level; treated as non-independent in statistical tests.
[c] Community-hosted competition, redistributed via custom preparation scripts.
[†] Used in the real-world necessity test (Section 5).

Metric. All 20 tasks are scored with mean absolute error (MAE) instead of each competition's original metric, so that scoring, aggregation, and the normalization in Section 5 are the same for all tasks. For Red Hat Business, the only classification task, this is the MAE between predicted and true labels, not a standard classification metric such as AUC-ROC.

Execution harness. For each task, the agent receives (1) a problem description (`README.md`) with domain context, metric, target columns, and the list of files; (2) raw relational data files (`.csv` or `.parquet`) in a local `/data` directory; and (3) a grading script (`evaluate.py`) that scores a submission against hidden test labels.

Temporal truncation. For tasks with auxiliary tables, we truncate every table at the training split date T_{split} to prevent leakage. Covariates that are legitimately known in advance (such as holiday flags) are placed in a separate `test_covariates.csv` file. This file is kept in the absent condition of Section 5, since it is part of the task definition rather than auxiliary data.

Sandboxing. Agents run in an isolated Docker container with `pandas`, `scikit-learn`, `lightgbm`, `xgboost`, `gluonts`, and `pytorch` installed. Internet access is restricted so that agents cannot retrieve Kaggle solutions or test labels. Each rollout has a 2-hour time limit.

C Details on TSAgentBench-Synth

Scope. The generator produces controlled stress tests, not a realistic model of a physical system. Section 3.3 shows that the generated series are smoother and lighter-tailed than real data.

Profiles. The three profiles share the equations below and differ in their constants and names (Table 6). All necessity experiments use the warehouse profile.

Table 6: Profile constants. The target is the last variable.

Profile	B	S	Variables ($x_1, \ldots, x_4$); periods $\mathcal{T}$ (h)
Energy grid	100	25	thermal, load, impedance, **frequency**; $\{24, 168\}$
Battery storage	80	15	throughput, resistance, temperature, **state of charge**; $\{12, 24\}$
Warehouse	1200	300	demand, lead time, backlog, **stock level**; $\{24, 168\}$

Causal graph. For each seed, we draw $A_{ij} \sim \text{Bernoulli}(0.35)$ for all i, j (self-loops allowed), $u_{ij} \sim \mathcal{U}(0.1, 0.6)$, and a random sign s_{ij}, and set $W_{ij} = A_{ij} u_{ij} s_{ij}$. $W_{ij} \neq 0$ means that $x_{t-1,j}$ affects $x_{t,i}$. If the spectral radius $\rho(W)$ exceeds 0.9, we rescale W by $0.85/\rho(W)$. All entities of a task share the same graph; different seeds give different graphs.

Dynamics and stability. Each entity follows Eq. (1). The seasonal term is $s_t = \sum_{\tau \in \mathcal{T}} \big[\sin(2\pi t/\tau) + 0.5\cos(2\pi t/2\tau)\big]$. The regime multiplier m_{r_t} equals 1 in the normal regime, multiplies x_1 by 1.3 in the surge regime and x_3 by 1.5 in the delay regime, and multiplies all variables by 0.6 in the failure regime. It scales only the input term, so the recursion matrix is always $0.7I + 0.2W$, whose eigenvalues $0.7 + 0.2\lambda(W)$ have modulus at most 0.88. The lower bound 10^{-3} keeps states positive and does not affect this contraction. As an additional check, a trajectory with $|x_{t,i}| > 10^5$ is discarded and simulated again.

Entities. Each of the E=4 entities has four parameters, all exported in `entity_metadata.csv`: an activity scale $\alpha_e \sim \mathcal{U}(0.8, 1.3)$, which sets the initial state $x_0 = B\alpha_e$; a noise scale $\sigma_e \sim \mathcal{U}(0.5, 2.5)$; and a capacity $\sim \mathcal{U}(0.7, 1.3)$ and a degradation rate $\sim \mathcal{U}(10^{-4}, 4 \cdot 10^{-4})$, which do not enter the dynamics. The last two are plausible but uninformative columns.

Regimes. At each step, the regime changes with probability 0.006 (mean duration about 170 hours). The new regime is drawn with probabilities proportional to $(0.5, 0.3, 0.2, p_f)$ for (normal, surge, delay, failure), with $p_f = 0.04$ initially. If the share of failure steps of an entity falls outside

Table 7: TSAGENTBENCH-SYNTH pathology operators. Operators marked ‡ support an absent condition.

Operator	File	Mechanism (intensity γ)	Capability
Lookahead leakage‡	macro	One bait column per entity: $(1-\gamma)\nu_t + \gamma\tilde{y}_{t+1}$ in the training window ($\tilde{y}$: standardized target) and $(1-\gamma)\nu_t + \gamma\eta_t$ in the test window; $\nu, \eta \sim \mathcal{N}(0,1)$.	Leakage detection
Trend extrapolation	target	Linear ramp with slope $\pm 0.15\gamma$ per step (random sign), over the whole series including the test window.	Detrending, extrapolation
Covariate drift‡	sensors	Independent cumulative Gaussian walk per entity, added to every sensor column; step mean 0.04γ, step s.d. 0.15γ.	Robustness to drift
Block missingness‡	sensors	Non-overlapping 24-step `NaN` blocks per column and entity, covering a share 0.4γ of rows.	Imputation
Datetime parsing	macro	A share γ of timestamps rewritten in up to five formats; more formats as γ grows.	Date parsing
Exogenous resampling	macro	Keep every $\mathrm{round}(1+9\gamma)$-th row.	Multi-rate alignment
Asynchronous time	macro	Timestamps shifted by $\mathrm{round}(12\gamma)$ hours.	Temporal alignment
Categorical noise	metadata	Character typos in the profile names of up to $\mathrm{round}(3\gamma)$ entities.	Robust joins

$[1\%, 20\%]$, the entity is simulated again, with p_f increased by 0.06 (too few failures) or decreased by 0.03 (too many), for at most 10 attempts. This check runs before any agent sees the task and does not depend on agent results.

Macro covariates. The macro table has one row per hour for the full timeline, including the forecast window. It contains a discretized Ornstein–Uhlenbeck index $z_t = z_{t-1} + 0.05(100 - z_{t-1}) + 0.45\,\eta_t$, $\eta_t \sim \mathcal{N}(0,1)$, and an ambient signal $15 + 4\sin(2\pi t/24)$. Neither enters Eq. (1); the ambient signal only repeats the daily cycle, which the agent can also recover from the timestamp.

Observed sensors. `observed_features.csv` contains one column per state variable (including the target) and a second copy of the target. It is exported only for the training window.

Task and split. The last 300 hours of each entity form the test window, and the preceding 3700 hours the training window. The agent receives the five CSV files of Table 2, a short description of the files, and the join keys (`entity_id`, `timestamp`). The hidden folder contains the ground truth and a specification file with the causal graph and the operator settings; the agent cannot access it. The simulator also produces regime and anomaly labels for other task types; this paper uses only the forecasting target.

Metric. Submissions are scored by MASE: the MAE on the test window divided by the in-sample MAE of the one-step naive forecast, computed per entity on the training window. Utility is $U = -\mathrm{MASE}$. Missing predictions are set to zero before scoring.

Pathology operators. Operators are applied in a fixed order (lookahead leakage, asynchronous time, resampling, categorical noise, drift, missingness, trend, datetime parsing), so that timestamp corruption comes last. Every operator is a no-op at γ=0, except that the lookahead operator still adds its bait columns, which are then pure noise. Table 7 gives each mechanism. In the necessity experiments, one operator varies at a time and all others are set to zero, with $\gamma \in \{0, 0.1, \ldots, 0.5\}$ for drift, missing ratios $\{0, 0.05, 0.10, 0.25, 0.40\}$ for missingness, and $\gamma \in \{0, 0.2, 0.5, 0.8, 1\}$ for lookahead leakage.

Challenge compiler. The compiler (1) simulates the SCM, (2) applies the operators in the order above, (3) splits each table at the forecast origin, and (4) writes the agent folder (data files and description) and the hidden folder (ground truth, specification, and scoring script).

Validation against real data. We test whether each pathology occurs in the real cohort, using hypothesis tests combined with Fisher's method ($\alpha = 0.01$). Table 8 gives the results.

Table 8: Validation of the synthetic operators against the real-world cohort $\mathcal{D}_{\text{eco}}$.

Operator	**Real-World Empirical Test**	**Validation Status**
Lookahead-Leakage	Feature/target correlation percentile	Confirmed ($p < 0.01$)
Trend-Extrapolation	Mann-Kendall test + extreme exceedance	Confirmed ($p < 0.01$)
Covariate-Drift	Per-entity AR(1) mean-reversion test	Confirmed ($p < 0.01$)
Block-Missingness	Per-entity Wald-Wolfowitz runs test	Confirmed ($p < 0.01$)
DateTime-Parsing	Schema parsing convention audit	Confirmed (4/11 audited sets)
Exogenous-Resampling	Modal timestamp-delta ratio	Insufficient real instances
Asynchronous-Time	Cross-correlation phase-lag analysis	Confirmed ($p < 0.01$)
Categorical-Noise	Key exact- vs. fuzzy-match rates	Insufficient real instances

D Realism Diagnostics

Statistics. For each dataset, we compute the variance, skewness, kurtosis, and differential entropy of the target; the Augmented Dickey–Fuller and KPSS stationarity tests; trend and seasonal strength from a classical moving-average decomposition (`statsmodels.tsa.seasonal_decompose`); the mean absolute autocorrelation over a fixed lag window; an ARCH statistic (autocorrelation of squared, demeaned values), which measures volatility clustering; and the dominant FFT frequency. The comparison uses 19 of the 20 real datasets and all 10 synthetic datasets. Per-dataset values are released with the code.

Aggregation. The main text reports medians and interquartile ranges, because a few datasets with extreme values dominate the mean. Table 9 reports means; for example, the mean kurtosis of the real cohort (4417) is driven by a small number of datasets.

Common analysis window. All series are analyzed with the same period and lag window, measured in time steps. Real granularities range from 1 minute to 1 week, so statistics tied to a period or a lag (seasonal strength, autocorrelation at a given lag, dominant frequency) do not correspond to the same physical time across datasets. Seasonal strength is close to zero for most real datasets because the common period does not match their natural cycle. For several real datasets, the dominant frequency corresponds to a period of 4 to 5 steps, which reflects short-lag noise rather than seasonality. We therefore do not use these statistics as evidence of a difference in periodicity. Distributional statistics such as kurtosis do not depend on the sampling period.

Figure panels. In Figure 3a, targets are z-scored so that datasets with different scales can be compared, and the density axis is logarithmic. In Figure 3d, each spectrum is normalized to its maximum and interpolated onto a common relative-frequency grid.

Table 9: Realism checklist: means across datasets in each cohort.

Statistic	Real	Synthetic	Comparison
Volatility clustering (ARCH)	0.428	0.410	Similar
Tail heaviness (kurtosis)	4417	53	Real much heavier
Seasonal / trend strength	0.006 / 0.48	0.032 / 0.71	Synthetic more regular

E Per-Model Gamma Curves

The figures below show the gamma curves for the three lookahead-leakage cells. No cell passes the necessity gate, so each figure shows $\Delta(\gamma)$ rather than $\mathrm{RS}(\gamma)$.

E.1 Competence vs. Robustness Across Operators

Figure 10 places each of the nine cells by clean utility and by sensitivity at high γ.

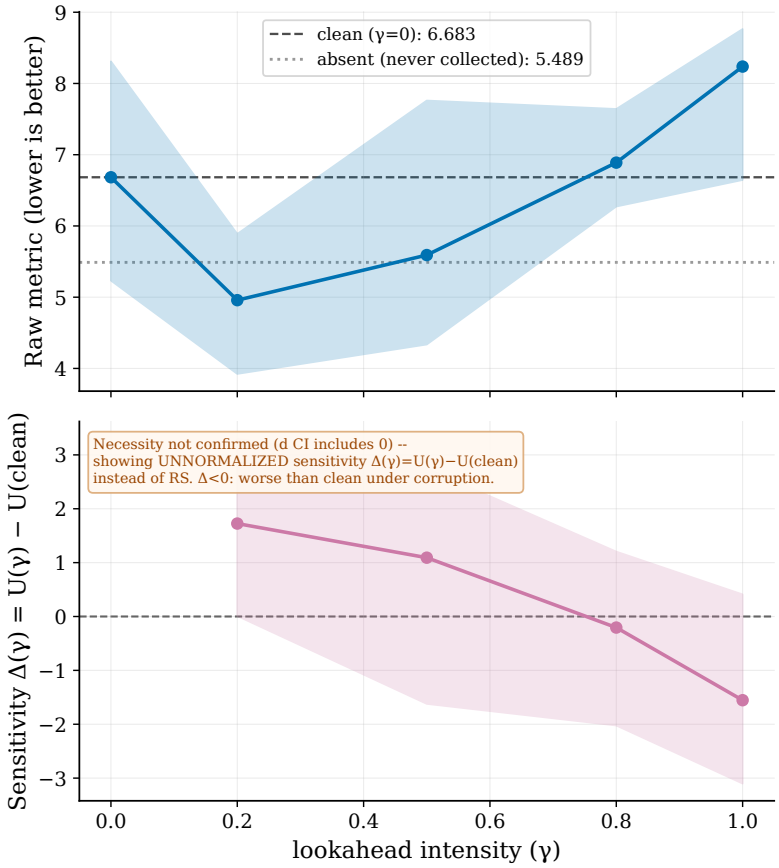


Figure 7: Qwen3.6-Plus, lookahead-leakage (n=14 seeds). The necessity gate does not pass (CI on d includes zero). Bottom: sensitivity curve $\Delta(\gamma)$.

Table 10: Point IDs and seed counts for Figure 10.

ID	Model	Operator	Seeds
1	GLM-5.2	Drift	11
2	Qwen3.6-Plus	Drift	15
3	Qwen3.7-Max	Drift	15
4	GLM-5.2	Lookahead-Leakage	11
5	Qwen3.6-Plus	Lookahead-Leakage	14
6	Qwen3.7-Max	Lookahead-Leakage	13
7	GLM-5.2	Missingness	10
8	Qwen3.6-Plus	Missingness	15
9	Qwen3.7-Max	Missingness	15

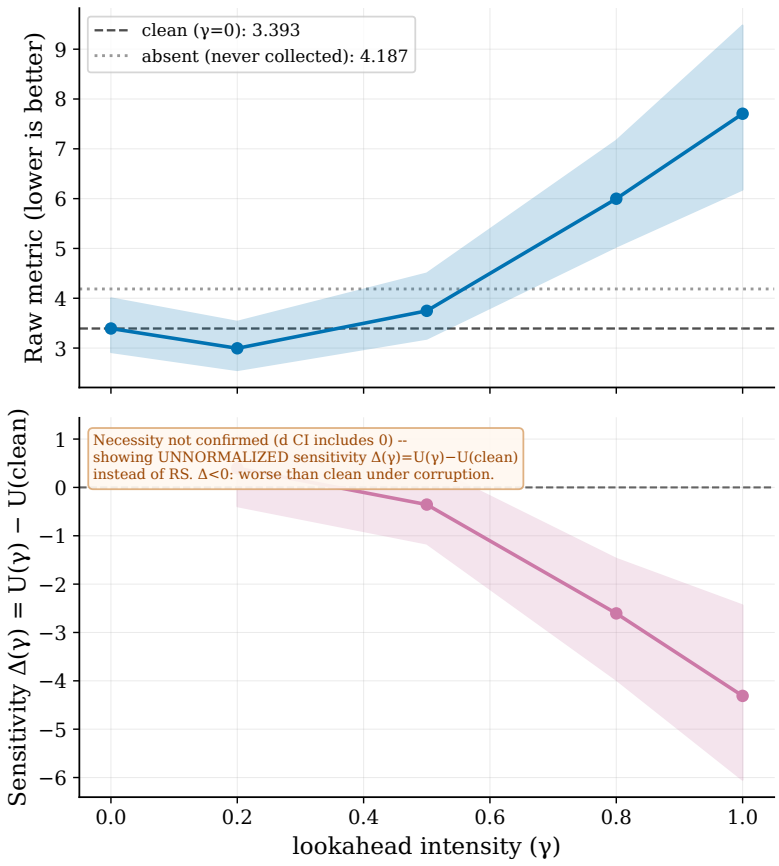


Figure 8: GLM-5.2, lookahead-leakage (n=11 seeds). The necessity gate does not pass (CI on d includes zero). Bottom: sensitivity curve $\Delta(\gamma)$.

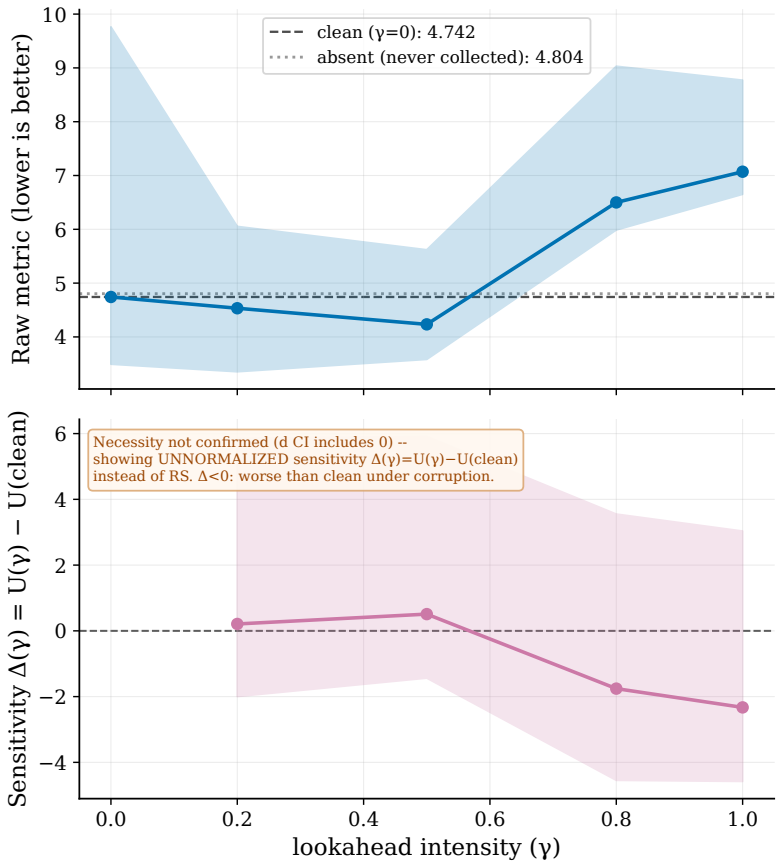


Figure 9: Qwen3.7-Max, lookahead-leakage (n=13 seeds). The necessity gate does not pass (CI on d includes zero). Bottom: sensitivity curve $\Delta(\gamma)$.

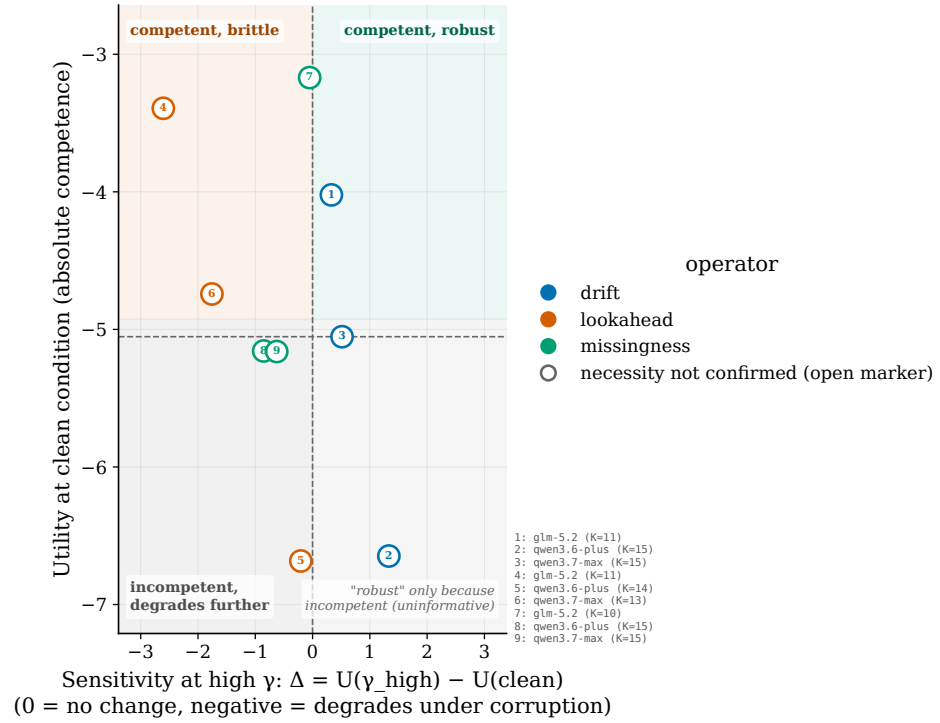


Figure 10: Competence (clean utility) vs. robustness (sensitivity at high γ) for the nine necessity-eligible cells, Smolagents only. Each point uses only seeds with clean, gamma-sweep, and absent runs all available, so seed counts (10–15, Table 10) can be lower than in Section 4.2. Filled markers: gate passed; open markers: gate not passed.

F CODE-LEVEL ENGAGEMENT: A CASE STUDY

A nonzero d shows that the availability of a covariate affects performance, but not that the agent uses the covariate in its model. Removing a column can also change the schema, the context length, or the features the agent tries. We therefore inspected the code produced by Qwen3.6-Plus in the lookahead-leakage cell, which has the largest negative point estimate of the three lookahead-leakage cells (its CI includes zero). For three clean-condition seeds, we checked whether the bait covariates (`macro_leakage_bait_*`) reach the fitted model.

Seed 0: used as a feature. A first feature set failed because sensor-derived variables were missing at test time (`NaN`). The agent then revised its pipeline and kept all four bait variables in the final feature set:

```
feature_cols_test = ['global_economic_index',
'ambient_environmental_noise',
                     'macro_leakage_bait_WH_000',
'macro_leakage_bait_WH_001',
                     'macro_leakage_bait_WH_002',
'macro_leakage_bait_WH_003', ...]
X_train = train_merged[feature_cols_test]
model.fit(X_train, y_train)
```

This model produces the submitted predictions; no later step changes the feature set.

Seed 1: available but unused. The bait variables are present in the merged data, pass the agent's column checks, and have no missing values. The agent nevertheless defines a manual feature set without them:

```
feature_cols = ['hour', 'dayofweek', 'ambient_environmental_noise',
'capacity',
                'base_activity_scale', 'sensor_noise_scale',
'structural_degradation',
                'entity_encoded']
X_train = train_full[feature_cols]
```

No bait variable appears in the final feature matrix or in the fitted model.

Seed 2: used in training and prediction. The bait variables are in the initial feature set, and the code retrieves them at each step of the sequential prediction:

```
feature_cols = [..., 'macro_leakage_bait_WH_000',
'macro_leakage_bait_WH_001',
                    'macro_leakage_bait_WH_002',
'macro_leakage_bait_WH_003', ...]
model.fit(X_train_filled, y_train)
...
lags[f'macro_leakage_bait_{entity}'] =
exo_row[f'macro_leakage_bait_{entity}'].values[0]
X_pred = pd.DataFrame([lags])[available_features]
pred = model.predict(X_pred)[0]
```

Summary. In two of the three runs, the bait variables are used by the final model. In the third, they are available but not used. The same agent, on the same cell, makes different choices about the same covariate.

Feature importance. In the clean condition (γ=0), the bait variables are pure noise. In seed 0, the fitted model gives them near-zero `feature_importances_`, and most of the importance ($\sim$0.98) goes to an autoregressive lag feature. The agent therefore keeps columns with no information in its model in two of the three runs, even though the model itself learns to give them little weight.

Scope. This is a case study of three runs for one model and one operator, not a population estimate. It shows at the code level the run-to-run variation measured in Section 4.3. An automated version would label each pipeline's use of a covariate (used as a feature, inspected only, never referenced) and be checked against hand labels. It needs only the stored code traces, and we leave it for future work.

G REPRODUCIBILITY

Code and data. We release the benchmark generator, agent adapters, evaluation pipeline (`tsbench` CLI), and all analysis scripts. These include `relative_robustness.py` (necessity gate, RS, and sensitivity), `build_absent_folders.py` (synthetic absent condition), `build_real_absent.py` (real-world absent condition, Section 5), and `hierarchical_bootstrap.py`/`compute_rs.py`.

Bootstrap. All confidence intervals come from a hierarchical bootstrap with $n_{\text{boot}} = 1000$ resamples. We first resample realizations with replacement, then resample the rollouts of each selected realization with replacement. In the synthetic experiments, realizations are seeds (K=15) and each seed has one to five rollouts, most often three to five. All seeds enter the necessity gate. The ICC (Section 4.3) needs at least two rollouts per seed, so it uses fewer seeds; for example, 11 of the 15 lookahead-leakage seeds of Qwen3.6-Plus. In the real-world test, realizations are the five datasets, with 3 to 21 runs per dataset depending on dataset and model. All intervals are two-sided 95% percentile intervals.

Sign convention. All metrics are converted to utilities (higher is better) before aggregation. With this convention, $d > 0$ means that the covariate helped.

Point estimates. All point estimates are medians over pooled rollouts, because a single failed rollout can shift a mean by orders of magnitude.

Necessity gate. A cell passes the gate if the 95% CI on d excludes zero. Cells where d is undefined are counted as not passing.

Intraclass correlation. Table 4 reports the one-way random-effects ICC (ICC(1) in the notation of Shrout & Fleiss (1979)), because each seed has its own independent rollouts rather than a shared set of raters. Negative estimates are clipped to zero, so a value of 0.000 can be an estimate of zero or a clipped negative value.

Robust standard deviation. The catastrophic-rollout threshold (Section 4.5) uses $\widehat{\sigma} = 1.4826 \times \mathrm{MAD}$, computed per (operator, model, condition) cell over all seeds and rollouts. The same rule removes catastrophic rollouts before computing the ICCs in Table 4, because the sums of squares in the ICC are sensitive to single extreme values.

H LIMITATIONS AND FUTURE WORK

Limitations. The drift and missingness sign test (six of six cells negative, $p = 0.016$) is our strongest directional result, but the six cells share three models and two operators and are not fully independent. No individual cell passes the necessity gate. The sensor covariates in these cells end at the forecast origin, so the result concerns covariates that are causally relevant but unavailable at forecast time; covariates that are both relevant and available were not tested. Lookahead-leakage is analyzed separately because, in the clean condition, its bait covariate is pure noise. The real-data test (Section 5) uses five datasets and has low power. All necessity experiments use one scaffold (Smolagents), three models, and the warehouse profile; the two Claude Code configurations that lead the real leaderboard are not included. The synthetic series are lighter-tailed and smoother than real data (Section 3.3), so effects that depend on heavy tails or abrupt shocks may be underrepresented in the synthetic cohort.

Operator scope. The necessity test applies to operators that corrupt a separable exogenous covariate (drift, missingness, lookahead-leakage). Trend extrapolation and datetime parsing change the target series itself, so no absent condition is defined for them. Exogenous resampling, asynchronous time, and categorical noise change schema and timing metadata rather than covariate values. These five operators are used in the leaderboard evaluation (Section 3) but not in the necessity test.

Future work. First, the necessity and instability analyses should be repeated with other scaffolds, since Section 3.1 shows that performance depends on the scaffold–model pair. Second, more operator–model cells would increase the power of the directional test; in particular, adding covariates that are available over the forecast window would test whether the negative effect depends on availability. Third, the case study in Appendix F can be turned into an automated engagement classifier validated on hand labels. Fourth, more real datasets would increase the power of the test in Section 5. Fifth, extending the framework to trend extrapolation and datetime parsing requires a different counterfactual, since no covariate can be removed for these operators.