

WHATIF-BENCH: Benchmarking Intervention-Aware Time Series Forecasting

Arjun Ashok^{1, 2, 3}, Mihir Parmar¹, Palash Goyal¹, Bhavana Dalvi Mishra¹, Chun-Liang Li¹, Irina Rish^{2, 3}, Alexandre Drouin^{2, 4}, Jinsung Yoon¹ and Tomas Pfister¹

¹Google, ²Mila - Quebec AI Institute, ³Université de Montréal, ⁴Université Laval

Recent time series foundation models (TSFMs) can consume covariates in-context, extending zero-shot forecasting to problems driven by external factors. This enables scenario analysis: forecasting how a target responds to *events* that set covariates to chosen future values. Whether they can do this reliably is an open question. We introduce WHATIF-BENCH, a benchmark measuring how well forecasters use covariates to forecast a target’s response to future events. WHATIF-BENCH is *semi-synthetic*: time series come from mechanistic simulators of data centers, power grids, road traffic, and water networks, with controlled covariate-target relationships grounded in each system. Each task presents a history of events intervening on covariates, then asks to forecast a future event that either replays seen values (*Replay*) or sets them to unseen ones (*Novel*). Varying the number of covariates, historical events, and event length yields 37,665 tasks. Evaluating TSFMs, frontier LLMs, and simple regression baselines, we find the LLMs reach the highest skill (improvement over a naive baseline), ahead of the best TSFMs. Every model forecasts well on Replay futures (48 to 85% skill) but drops to -7 to 50% on Novel futures, with the regression baselines falling the most. Skill on Novel futures also falls as events set more covariates jointly. More historical events raise skill on Novel futures across the board, but large differences remain. WHATIF-BENCH reveals generalization to Novel futures as the core challenge, and offers a testbed toward arbitrary scenario analysis.

🏆 [Leaderboard](#)

🔗 [Code](#)

1. Introduction

Forecasting guides decisions across science, engineering, and commerce (Hyndman and Athanasopoulos, 2018), and models have advanced from classical methods and deep networks (Benidis et al., 2022; Box et al., 2015; Gardner Jr., 1985; Nie et al., 2023; Salinas et al., 2020) to time series foundation models (TSFMs) that forecast unseen series zero-shot (Ansari et al., 2024; Das et al., 2024; Liang et al., 2024; Woo et al., 2024). Most TSFMs are univariate, yet many real targets are shaped by external drivers such as weather, prices, and policies; recent TSFMs can instead consume such covariates *in-context* and factor them into the forecast without task-specific training (Ansari et al., 2025; Google Research, 2026; Le Naour et al., 2026; Meyer et al., 2026; Podest et al., 2026).

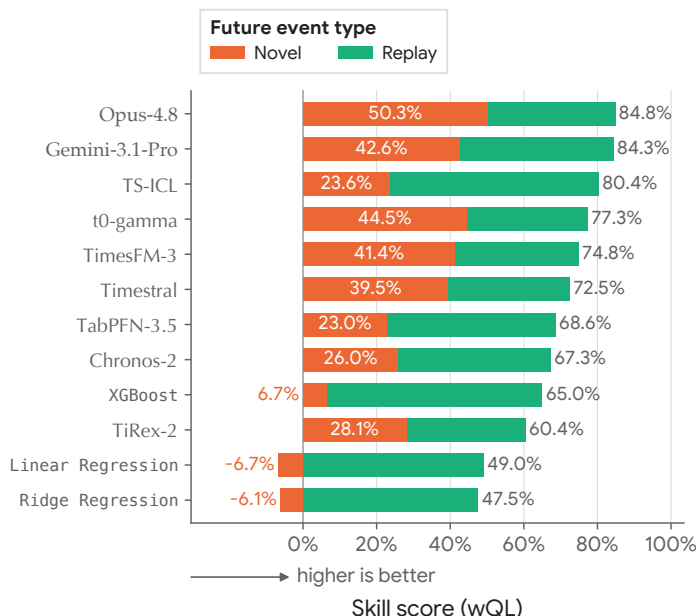


Figure 1 | Performance of LLMs, numerical foundation models, and regression models on WHATIF-BENCH tasks. Every model’s skill score drops sharply from Replay to Novel futures.

*Corresponding Authors: Arjun Ashok <arjun.ashok.psg@gmail.com>, Mihir Parmar <mihirparmar@google.com> and Palash Goyal <palashgoyal@google.com>

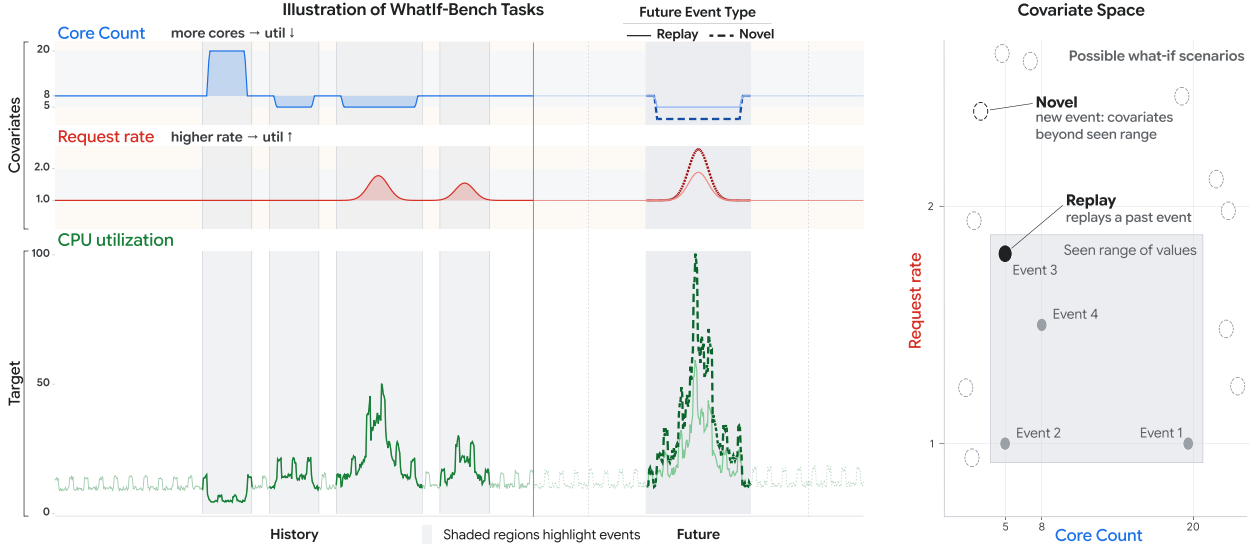


Figure 2 | **Illustration of WHATIF-BENCH tasks.** Left: a WHATIF-BENCH task contains a history of events, in which covariates are set and the target responds, and asks a model to forecast the target under one further event. Right: the same events in covariate space. A Replay future repeats a past event (here, the third), so it lies inside the range of values the history has shown; a Novel future sets the covariates beyond that range, and is one of many possible what-if scenarios. Novel futures thus test whether a model extends what it learned to unseen values. Figure 1 shows that current models do this poorly: their skill on Novel futures is far below their skill on Replay futures.

When the future covariates are known, this enables *scenario analysis*: predicting how the target would evolve under different futures, which is central to planning (Bańbura et al., 2015; Doan et al., 1984; Waggoner and Zha, 1999). Existing benchmarks, however, evaluate only on the covariate values their datasets happen to contain (Lago et al., 2021; Shchur et al., 2025), so they cannot tell whether a model forecasts accurately when covariates take values beyond that range. They also measure performance only in aggregate over the conditions each dataset provides, so they cannot reveal where a model systematically fails at using covariates.

We present WHATIF-BENCH, a *semi-synthetic* benchmark that measures how well a forecaster uses in-context covariates to forecast a target’s response to future events, including scenarios that set covariates to values never seen in the history. Its target series come from mechanistic simulators of four domains - OpenDC (data centers) (Iosup et al., 2017), GridLAB-D (power grids) (Chassin et al., 2008), SUMO (road traffic) (Lopez et al., 2018), and EPANET (water networks) (Rossman, 2000), on which we impose controlled covariate-target relationships grounded in each system. Because these relationships are known exactly, we can isolate the effect of any single covariate, so a model’s error reflects how well it handles the covariate rather than irreducible noise. Each task presents a history of *events*, each setting covariates to chosen values and letting the target respond, and asks the model to forecast the target under one further event: a *Replay* future that repeats a past event, or a *Novel* future that sets covariates to unseen values, as illustrated in Figure 2. WHATIF-BENCH is a controllable framework: we vary one axis at a time – the future type, the number of covariates set jointly, the number and length of historical events, and the context and prediction length – yielding 37,665 tasks over 100 target series, so each axis can be studied on its own.

We evaluate covariate-aware foundation models and frontier LLMs. The LLMs consume the series and its covariates as text, and have proven competitive at forecasting (Gruver et al., 2023; Requeima et al., 2024). We also evaluate the simplest versions of regression baselines (XGBoost, Ridge Regression, and Linear Regression), which map the covariates at each step to the target. We use the skill score as our

Benchmark	Data source	Includes covariates	Covariate effect known exactly	Evaluates arbitrary what-if scenarios	Systematic difficulty (# covariates, historical evidence)	Tasks with covariates
Monash (Godaheva et al., 2021)	Real-world	✗	✗	✗	✗	0
GIFT-Eval (Aksu et al., 2024)	Real-world	✗	✗	✗	✗	0
TSFM-Bench (Li et al., 2025)	Real-world	✗	✗	✗	✗	0
TIME (Qiao et al., 2026)	Real-world	✗	✗	✗	✗	0
fev-bench (Shchur et al., 2025)	Real-world	✓	✗	✗	✗	46
Bertheliet et al. (2026)	Semi-synthetic (toy relations)	✓	✓	✗	✗	— [†]
WHATIF-BENCH (Ours)	Semi-synthetic (simulator-grounded)	✓	✓	✓	✓	37,665

[†]Number of tasks not reported.

Table 1 | **Comparison of WHATIF-BENCH with existing forecasting benchmarks.** WHATIF-BENCH supports evaluation on arbitrary what-if scenarios and lets difficulty be varied systematically, from few to many covariates and from few to many past observations of their effects, exposing differences between models that existing benchmarks cannot show.

main metric: a model’s relative improvement over a naive baseline that repeats the last value. First, on the full benchmark, the frontier LLMs lead: Claude-Opus-4.8 and Gemini-3.1-Pro reach the top skill scores, 72.5% and 70%, with the best covariate-aware TSFMs, t0-gamma, TimesFM-3 and TS-ICL, close behind at 61 to 65%, and a simple, untuned XGBoost baseline (43%) below every foundation model. Next, every model shares a failure mode: all forecast well on *Replay* futures (48–85% skill) but drop to between −7% and 50% on *Novel* ones (Figure 1). The regression baselines collapse the most: Linear Regression has a skill of 49% on *Replay* futures but −7% on *Novel* futures. Therefore, despite their own drop, the LLMs and foundation models keep a large advantage over the regression baselines on *Novel* futures. Further, the number of covariates has no clear effect on *Replay* futures, but skill on *Novel* futures also falls as events set more covariates jointly. Finally, a longer history of events helps: models improve their skill, especially on *Novel* futures, though large differences between models remain. Together, these results show that generalizing to *Novel* futures is the core challenge, and the further axes of WHATIF-BENCH show how it depends on the number of covariates and on the history available. WHATIF-BENCH provides a testbed toward models that support arbitrary scenario analysis.

2. Related Work

Below, we position WHATIF-BENCH with respect to existing forecasting benchmarks and related work on causal inference. Background on the models we evaluate is given in Appendix A.

2.1. Time Series Forecasting Benchmarks

Early benchmarks such as the M-competitions and the Monash archive standardized evaluation but largely measure univariate accuracy (Alexandrov et al., 2020; Godahewa et al., 2021; Makridakis and Hibon, 2000; Makridakis et al., 2018), while more recent ones target TSFMs with large, leakage-free test sets, such as GIFT-Eval (Aksu et al., 2024), TSFM-Bench (Li et al., 2025), and TIME (Qiao et al., 2026). All of them, however, evaluate univariate forecasting and do not contain covariates. fev-bench (Shchur et al., 2025) includes tasks with real-world covariates drawn from datasets such as M5 (Makridakis et al., 2022), the EPF toolbox (Lago et al., 2021), Rohlik Sales (Kaggle, 2025), and Walmart (Kaggle, 2014). It evaluates only on covariate values that appear in the data, so it cannot tell whether a model forecasts accurately at unseen values. It also measures performance only under the conditions each dataset happens to provide, so it cannot reveal where a model systematically fails. It evaluates only on the covariate values its datasets happen to contain, so it cannot tell whether a model forecasts accurately when covariates take values beyond that range. It also measures performance only in aggregate over the conditions each dataset provides, so it cannot reveal where a model systematically fails at using covariates. WHATIF-BENCH instead controls these conditions directly:

it sets covariates to unseen values to test extrapolation, and varies one axis at a time to pinpoint where a model fails. Xu et al. (2024) also benchmark forecasting under interventions; they study forecasting under textual events, rather than as numeric covariates as we do; this places it in the different setting of multimodal forecasting, as in Williams et al. (2025), where covariate values and their effects are not tested. Closest to our work, Bertheliet et al. (2026) probe whether two foundation models capture very simple covariate-target relationships - identity, sum, aggregate, and quadratic, and find that the two behave differently even on such simple relationships. WHATIF-BENCH, by contrast, is large-scale and diverse (Table 1): it imposes many covariate-target relationships grounded in mechanistic simulators, and varies axes such as the future type (*Replay* or *Novel*), the number of covariates, and the amount of in-context evidence. Across these conditions, we evaluate a broad range of numerical foundation models and LLMs.

2.2. Causality and Interventional Estimation

Measuring whether a forecast responds correctly to a covariate is an interventional question: it requires the target under new, chosen covariate settings, which real data seldom provides (Holland, 1986; Imbens and Rubin, 2015). The causal-inference community has long worked around this by evaluating on simulators and semi-synthetic data whose data-generating process is known, so the effect of any intervention can be computed exactly (Dorie et al., 2019; Hill, 2011); WHATIF-BENCH borrows this methodology, but applies it to forecasting rather than treatment-effect estimation. A related thread predicts outcomes under time-varying treatments (Robins et al., 2000), such as Counterfactual Recurrent Networks (Bica et al., 2020), which forecast counterfactual trajectories and score them by forecast error on simulated environments. These methods are trained on each system and rely on identification assumptions. Similarly, DoTime (Thumm et al., 2026) generates fully synthetic causal models with interventions to benchmark treatment-effect estimation. We instead evaluate zero-shot forecasters, which must learn the covariate effects in-context from the history alone, with no training on the system.

3. WHATIF-BENCH

3.1. Problem setting

Let $\mathbf{X}_H = [X_1, \dots, X_t]$ be the observed history of a scalar target, each $X_\tau \in \mathbb{R}$. Assume the target is driven by K exogenous, numerical covariates, observed over the same period as $\mathbf{Z}_H = [\mathbf{z}_1, \dots, \mathbf{z}_t]$, each $\mathbf{z}_\tau \in \mathbb{R}^K$. Canonically, forecasting concerns predicting the target’s evolution $\mathbf{X}_F = [X_{t+1}, \dots, X_T]$ from this history alone, by estimating $P(\mathbf{X}_F \mid \mathbf{X}_H, \mathbf{Z}_H)$. When the covariates’ future values $\mathbf{Z}_F = [\mathbf{z}_{t+1}, \dots, \mathbf{z}_T]$ are themselves known, a forecaster can condition on them as well, to instead estimate $P(\mathbf{X}_F \mid \mathbf{X}_H, \mathbf{Z}_H, \mathbf{Z}_F)$. The future covariates $\mathbf{Z}_F$ allow one to express various future *what-if scenarios*, and produce a conditional forecast of the target under each scenario. This can reveal which drivers move the target and how strongly, and supports planning against many possible futures rather than the single one implied by holding covariates at their most recent values (Brodersen et al., 2015).

The core challenge is to learn the covariate-target relationships the history reveals, separating the target’s own variation from covariate-driven changes, and then apply them at the arbitrary values $\mathbf{Z}_F$ the scenario specifies. Since a scenario may set $\mathbf{Z}_F$ to values the history has shown, or to values beyond anything it contains, the relationships must be learned well enough to extrapolate: to apply at covariate values never seen, and predict the target at levels it never reached. We call this setting *intervention-aware forecasting*, as it requires models to learn from past “interventions” in the system - from the past changes in the covariates that drive the target, and to respond to the future interventions

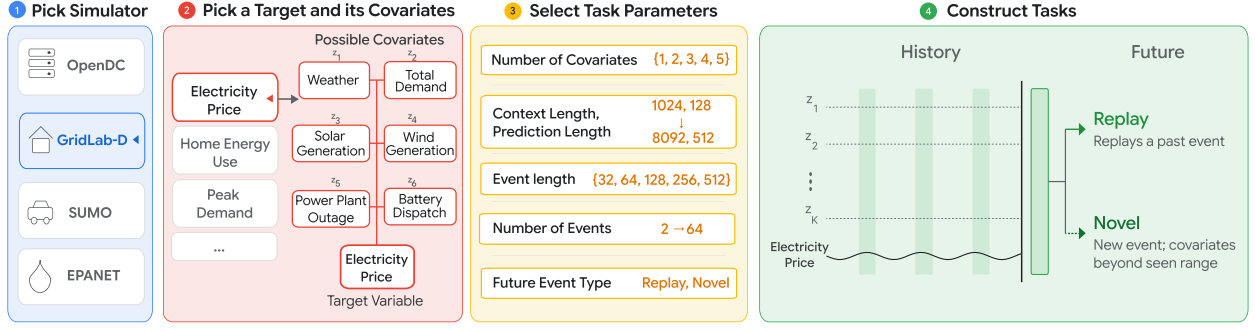


Figure 3 | **The WHATIF-BENCH construction recipe.** We (1) pick a simulator and a base setting; (2) choose a target and the covariates that drive it; (3) select the parameters of the task; finally (4) construct the events in the history, to produce two futures over the same history. This yields 2 tasks. We generate the benchmark by repeating this process for every target across all simulators, and over all axes: number of covariates, number of events, event length, and context length.

a scenario specifies ¹. We operate in the *in-context learning* setting, where these relationships must be learned *in-context* from the dynamics seen in the history, across the segments in which the covariates take on different values (Shchur et al., 2025).

3.2. Semi-synthetic design

Motivation. Evaluating intervention-aware forecasting requires assessing whether a model forecasts the target accurately under different settings of the future covariates. Real-world data, however, gives only one realization of the covariates and target, and A/B tests that change covariates to observe their effects are usually infeasible (Holland, 1986; Imbens and Rubin, 2015). We instead need an environment where we control the covariates, record their effects on the target, and test how sensitive a model is to changing them in the future. Research in causality has long addressed this with *semi-synthetic* data, where a known, generated process supplies exact counterfactuals (Dorie et al., 2019; Hill, 2011). Building on this principle, we introduce WHATIF-BENCH, a benchmark for intervention-aware forecasting that combines realistic time series from mechanistic simulators of real-world systems with controlled covariate-target relationships whose effects are known exactly.

The simulators. We use four established, open-source simulators from distinct domains, each developed by domain experts and widely used in research and practice to study and plan real-world systems: **OpenDC** (Andreadis et al., 2022; Iosup et al., 2017; Mastenbroek et al., 2021) models a data center’s resource usage for capacity planning; **GridLAB-D** (Chassin et al., 2008; Schneider et al., 2010) models a power-distribution grid and the buildings on it; **SUMO** (Lopez et al., 2018; Lu et al., 2023) models road traffic vehicle by vehicle; and **EPANET** (Kępa and Deska, 2025; Rossman, 2000) models pressurized water-distribution networks.

Targets, covariates, and their relationships. We run each simulator on a fixed *base setting*: a real configuration of the system together with the recorded input that drives it – for example, a data center’s setup and the demand placed on it. This produces time series of the system’s variables, such as the CPU utilization of each host, as they evolve under the simulator’s dynamics. From each run, a system variable serves as the target and covariates drawn from the simulator’s adjustable

¹We use “intervention” to mean setting the values of the covariates, which are among the parents of the target, i.e. $\text{do}(\mathbf{Z} = \mathbf{z})$ (Pearl, 2009). The target is never intervened on.

properties drive it. On top of these initial targets and their associated covariates, we impose controlled relationships that govern how changes in the covariates affect the target, grounded in each system, so their effect is known exactly at any value they take. This gives us a semi-synthetic environment of realistic time series targets and covariates with known, exactly reproducible effects. The rationale behind this design, the target variables we extract from each simulator, and the relationships we impose for how changes in the covariates affect the target are given in Appendix B.

3.3. Events

The building block. An *event* is an intervention held over a bounded window of time, acting on one or more covariates at once. Within the window the covariates are moved away from their base setting, and the target responds for as long as the covariates are held there. Such a setup reflects how change happens in real systems, where discrete interventions move the drivers over a bounded window and the target responds, a structure long studied as interrupted time series and event studies (Box and Tiao, 1975; Brodersen et al., 2015).

Event design. An event specifies two properties for each covariate it acts on: its *magnitude* - how far to move it, and its *shape* - how it moves across the window. For the magnitude, we apply a *multiplicative factor* to the base value, which scales the covariate and moves the target by the amount the imposed relationship prescribes. This relationship is fixed for each covariate-target pair and defined at every factor value, but need not be proportional: it may be inverse, sublinear, or piecewise, so a factor of two on the covariate need not mean a factor of two on the target. Within the factors we use, each relationship moves the target in one direction and never reverses, so it can be learned from the history. For the shape, we use two: a step, which holds the new value constant throughout, and a gaussian bell, which rises to it and decays back. Each covariate in our benchmark is assigned one. Additional details are given in Appendix B.6.

3.4. Task structure and axes

From events to tasks. A WHATIF-BENCH task simply places a sequence of events in the history, each leaving its effect on the target, and asks the model to forecast the target’s response to one further event in the future. We build the tasks of WHATIF-BENCH by varying five axes of this structure, described next (illustrated in Figure 3).

Replay and Novel Futures. The first axis is the future event: it is designed to evaluate whether a model can apply the relationships learned from history to *arbitrary* future covariate values. We generate tasks with two futures over the same history, differing only in the future covariate values. We call the choice between them the *future event type*:

- **Replay.** The future event takes on the covariate values of a past event, so the covariates take values they have already taken in the history.
- **Novel.** The future event is unseen: the covariates are driven to values beyond the range seen in the history.

In both cases, the model must forecast the target’s intrinsic dynamics (its trend, seasonality, and level) and, on top of that, the covariate effect at the specified values. The two futures give complementary evaluation sets: Replay measures a model’s basic covariate-conditioned forecasting, while a model that has learned the relationships *robustly* should do equally well on Novel. This is closely tied to the principle of invariance in causality (Peters et al., 2016; Rojas-Carulla et al., 2018): a model that has

captured the true mechanism linking the covariates to the target should transfer to new covariate values.² The degradation from Replay to Novel indicates whether a model is reliable for arbitrary scenario analysis. Figure 2 illustrates the structure of WHATIF-BENCH tasks.

Covariate settings. The second axis is the number of covariates in a task $K \in \mathcal{K} = \{1, \dots, 5\}$, which sets how many covariates every event acts on at once. Across the events in the history, these covariates take a range of values, from which a model can learn how each acts on the target and how they act together. Larger K means more relationships to learn, and therefore harder tasks.

Context and prediction length. The third axis is the context and prediction length: events sit within a history of fixed length, the *context length* – and the model forecasts a fixed horizon beyond it, the *prediction length*. We use four pairs, $c \in C = \{C_1, \dots, C_4\}$, from the smallest C_1 (context 1024, prediction 128) to the largest C_4 (context about 8,000, prediction 512 to 864, depending on the simulator; details in App. Table 4). The context sets the budget the events must fit within; for the longest events, which would not fit at the smallest contexts, we extend the context slightly (Appendix B.3).

Event length. The fourth axis is the event length $\ell \in \mathcal{L} = \{32, 64, 128, 256, 512\}$, which sets how long each event holds its covariates in place, spanning brief interventions to sustained ones. A short event gives only a brief glimpse of a relationship, whereas a long one lets the model see it settle and hold, but takes up more of the history and so leaves room for fewer events.

Number of events. The fifth and final axis is the number of events E , which controls how often the covariates are set across the history: more events expose their relationships at more values, which can let a model fit them more robustly. How many fit is bounded by the context and event lengths – at a context of 1024, only a few long events fit, but far more short ones do. For each event length and context/prediction pair, we build tasks at two settings: the fewest events (one or two) and the most the context allows, up to 64. We design WHATIF-BENCH around real-world regimes rather than guaranteed identifiability: since limited-evidence tasks (e.g., many covariates, few events) do occur in practice, we include them and expect a reliable forecaster to respond with calibrated uncertainty rather than overconfidence; we discuss this design choice further in Appendix B.5.

Instantiating WHATIF-BENCH. To recap the recipe (Figure 3): we (1) pick a simulator and a base setting; (2) choose a target variable and the covariates that drive it; (3) select the task’s parameters; and (4) construct the events in the history, producing two futures over the same history. We repeat this recipe over all targets and every combination of the axes above, giving 40,000 tasks in all (Table 2). Out of these, we keep only tasks whose covariates have a non-negligible effect in every event, leaving

Benchmark composition		
Simulator	Domain	# Targets
OpenDC (Iosup et al., 2017)	Data center	22
GridLAB-D (Chassin et al., 2008)	Power grid	11
SUMO (Lopez et al., 2018)	Road traffic	30
EPANET (Rossman, 2000)	Water network	37
Total		100
Design axes		
Axis	Values	Count
Targets	across 4 domains	100
Covariates	1, 2, 3, 4, 5	5
Context / prediction length	1k/128, 2k/256, 4k/384, ~8k/512–864	4
Event length	32, 64, 128, 256, 512	5
# events	1–64 (fewest, most)	2
Futures	Replay, Novel	2
Tasks Generated		37,665

Table 2 | **The WHATIF-BENCH instantiation.** The number of series contributed by the four simulators, and the axes we vary over them to build the tasks. The exact context and prediction lengths per simulator are given in Table 4.

²Novel thus tests this consequence of learning the mechanism. Doing well on Novel does not in turn guarantee that the mechanism has been learned, since Novel covers a finite set of covariate shifts; we use it to measure how well this transfer holds in practice.

37,665 of the 40,000 tasks. Additional details on the benchmark and the various design choices are in Appendix B.6. Example tasks are provided in Appendix L.

3.5. Metrics and aggregation

Metrics We score each task with standard forecasting metrics. Our primary metric is the weighted quantile loss (wQL), which evaluates probabilistic forecasts across a set of quantile levels; for point-forecast accuracy we also report the weighted absolute percentage error (WAPE) on the median forecast (definitions in Appendix C).

Aggregating performance across tasks Following fev-bench (Shchur et al., 2025), we summarize the per-task errors (wQL or WAPE) along two dimensions. The primary one is the *skill score*, a model’s relative reduction in error against a fixed naive baseline that repeats the last observed value. As a secondary measure, we use the *average win rate*: the fraction of tasks on which a model has a lower error than another model, averaged over all other models in the set (higher is better for both). We use it to compare models directly against one another. We report win rates in Appendix K and give formal definitions of both in Appendix C.

3.6. Models and Evaluation Protocol

Numerical forecasters. Our main candidates for evaluation are foundation models that can forecast time series while consuming past and future covariates in-context. Six such models exist, and we evaluate all of them: Chronos-2 (Ansari et al., 2025), t0-gamma (Meyer et al., 2026), Timestral (Mistral AI, 2026), TiRex-2 (Podest et al., 2026), TS-ICL (Le Naour et al., 2026), and TimesFM-3.0 (Google Research, 2026). We also evaluate TabPFN-3.5 (Jäger et al., 2026), a tabular foundation model, which we run for time series forecasting through the TabPFN-TS harness (Hoo et al., 2025). These models output nine quantile levels from 0.1 to 0.9 directly.

Large language models. We also evaluate frontier LLMs such as Gemini-3.1-Pro (Google DeepMind, 2026) and Claude-Opus-4.8 (Anthropic, 2026), prompted directly with the target and covariates as text. We run each at two reasoning efforts, Medium and High, to measure the effect of reasoning. Unlike the foundation models, the LLMs are prompted for the median rather than for quantiles: we find that prompting them for many quantiles exceeds the output length and causes formatting failures, so we score their wQL on the median alone, at the 0.5 level, which disadvantages them under probabilistic metrics. Implementation details on all models are given in Appendix D.

Baselines. We include two kinds of baselines. The naive baseline copies the last observed value and does not use covariates. The regression baselines - Linear Regression, Ridge Regression, and XGBoost - use both past and future covariates, in their simplest form: we fit each one on a task’s history with one training row per time step, predicting the target from the covariates at that step only, and then predict each future step separately from its known covariates. Our problem setting does not assume a separate training or validation set for each task, so these baselines cannot be tuned on held-out data. We use the simplest versions to understand how far the covariates alone take a forecaster; variants with minor tuning on the test set are explored in Appendix J. Like the LLMs, they only provide point forecasts. We use the naive baseline as the reference for the skill score, since it gives the same forecast on Replay and Novel futures over a shared history, which makes it a consistent reference to measure skill against. We also explore using the covariate-aware Linear Regression as the reference in Appendix I.

Model	wQL Skill	WAPE Skill	Inf. time / series (s)
Frontier LLMs			
Claude-Opus-4.8 (High)	72.5% 	72.5% 	270.13
Claude-Opus-4.8 (Medium)	<u>70.3%</u> 	<u>70.3%</u> 	206.29
Gemini-3.1-Pro (High)	<u>70.0%</u> 	<u>70.0%</u> 	355.28
Gemini-3.1-Pro (Medium)	58.2% 	58.2% 	97.35
Numerical Foundation Models			
t0-gamma	64.5% 	58.4% 	0.043
TimesFM-3	61.6% 	54.5% 	0.419
TS-ICL	61.3% 	55.3% 	0.083
Timestral	59.2% 	52.7% 	0.048
TabPFN-3.5	50.8% 	42.3% 	21.26
Chronos-2	50.8% 	42.0% 	0.067
TiRex-2	46.7% 	36.5% 	0.079
Regression Models[†]			
XGBoost	42.9% 	42.9% 	0.068
Linear Regression	26.2% 	26.2% 	0.002
Ridge Regression	25.4% 	25.4% 	0.002
Baseline			
Naive*	0.0%	0.0%	0.072

Table 3 | Performance of models on WHATIF-BENCH, measured as skill score under both wQL (probabilistic) and WAPE (point) metrics, alongside inference latency. The best score per column is **bolded** and the second-best is underlined. Naive* represents a naive baseline which does not consume covariates. [†]The simplest, untuned versions of the regression models; tuned variants are in Appendix J. Win rates are provided in Appendix K.

4. Results

4.1. Performance on WHATIF-BENCH

We present results on WHATIF-BENCH in Table 3. The regression baselines, in their simplest form, already obtain substantial skill over the naive baseline: XGBoost reaches 42.9%, and even Ridge and Linear Regression reach 25.4% and 26.2%. Among the foundation models, t0-gamma (64.5%), TimesFM-3 (61.6%) and TS-ICL (61.3%) obtain the best skill scores, about 19 to 22 points above XGBoost. Timestral (59.2%) follows, then TabPFN-3.5 and Chronos-2 (50.8% each), about 8 points above XGBoost, followed by TiRex-2 (46.7%), about 4 points above it. Notably, the LLMs obtain the highest skill scores on the benchmark, with Claude-Opus-4.8 (High) at 72.5% followed by Gemini-3.1-Pro (High) at 70%. Reasoning helps here: moving from Medium to High effort raises skill from 70.3% to 72.5% for Claude and from 58.2% to 70% for Gemini. On inference time, the foundation models are the fastest: most run in under half a second per series, and TabPFN-3.5 is the slowest at 21s. The LLMs are far slower, at around 300 seconds per series, over an order of magnitude beyond TabPFN-3.5, and almost 4 orders of magnitude beyond the rest.

Takeaway. Frontier LLMs score highest. Among the foundation models, t0-gamma leads, followed by TimesFM-3 and TS-ICL, and all of them are ahead of a simple, untuned XGBoost baseline that uses the covariates. The LLMs are, however, about 4 orders of magnitude slower at inference.

4.2. Performance on Replay vs Novel tasks

Figure 1, in the introduction, compares model performance on the Replay and Novel tasks in WHATIF-BENCH. On Replay tasks, every model achieves a high skill score, indicating that the models forecast

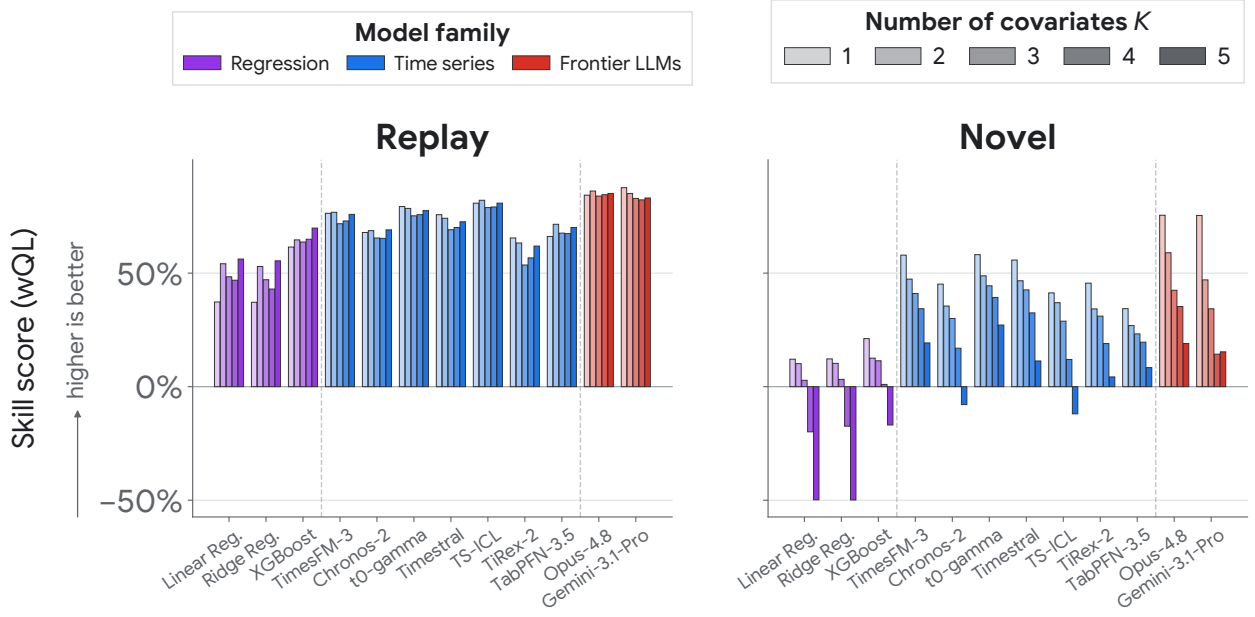


Figure 4 | Skill scores on tasks with different number of covariates on WHATIF-BENCH on Replay (left) and Novel (right) future types respectively.

well when the future event repeats covariate patterns already seen in the history. The LLMs are strongest here, with Claude Opus 4.8 (High) at 84.8% and Gemini 3.1 Pro (High) at 84.3%, followed by TS-ICL (80.4%) and t0-gamma (77.3%) among the foundation models. The regression baselines are competitive on Replay: XGBoost reaches 65.0%. On Novel futures, however, the skill of every model drops sharply, and the LLMs fall just as much: Claude Opus 4.8 drops from 84.8% to 50.3% and Gemini 3.1 Pro from 84.3% to 42.6%, while TS-ICL falls furthest among the foundation models, from 80.4% to 23.6%. This degradation appears for every model, indicating the difficulty of extrapolating to unseen values, and points to a lack of robustness. The regression baselines fall the most: XGBoost drops from 65.0% to 6.7% and Linear Regression from 49.0% to -6.7%, worse than the naive baseline. Novel futures are thus where the LLMs and foundation models have their largest advantage over the regression baselines: TimesFM-3 is only 10 points above XGBoost on Replay but 35 points above it on Novel.

Takeaway. Every model forecasts well on Replay, but skill drops sharply on Novel. The regression baselines fall the most, so Novel is where the LLMs and foundation models have their largest advantage over simple models. Yet even the LLMs and foundation models drop substantially, so strong Replay performance does not carry over to Novel futures. Closing this gap is the core challenge WHATIF-BENCH exposes.

4.3. Effect of the number of covariates

Figure 4 shows how skill varies with the number of covariates K , for Replay and Novel futures. On Replay futures, skill varies marginally as the number of covariates grows, for both foundation models and LLMs, which indicates that adding covariates does not make the Replay task harder. However, on Novel futures, skill decreases consistently as the number of covariates increases. Claude Opus 4.8 (High) and Gemini 3.1 Pro (High) both reach 75% at one covariate, far ahead of the other models. But their performance falls steeply, to 19% and 15% respectively at five covariates. By five covariates the best foundation models catch up with them: t0-gamma is strongest at 27%, ahead of both LLMs, and TimesFM-3 (19%) is level with Claude Opus 4.8 (High). This drop also reshuffles the ranking: TS-ICL, for instance, is among the best models on Replay at every covariate count (around 80%), yet

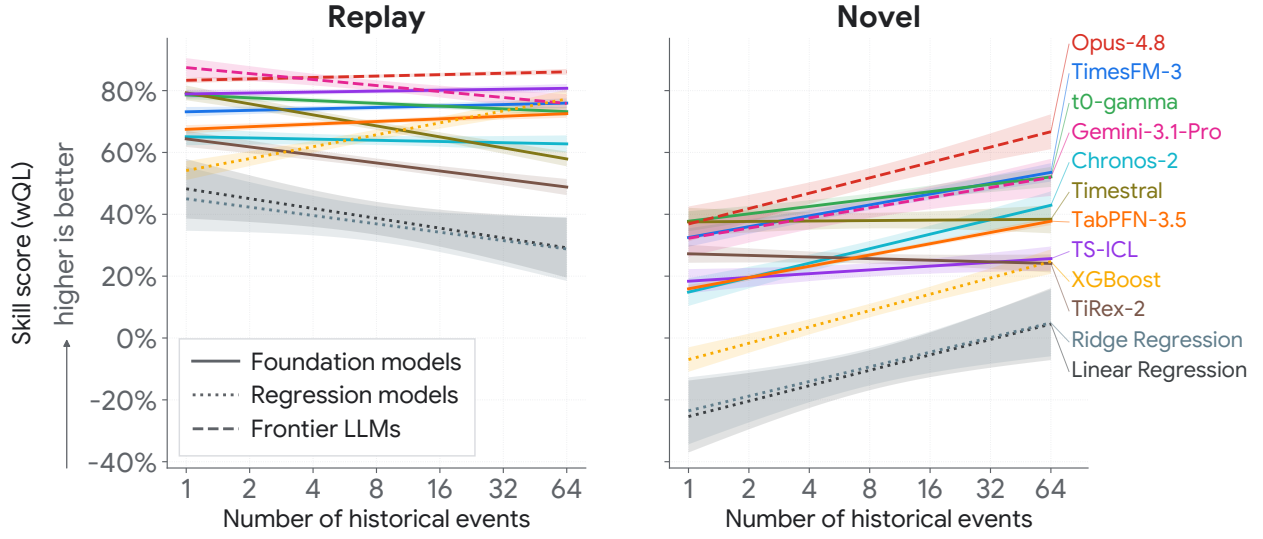


Figure 5 | Skill scores with different number of historical events on WHATIF-BENCH, on tasks with Replay (left) and Novel (right) futures respectively. Lines are least-squares trends of skill on $\log_2$ of the number of events; they summarise the direction of change.

drops to -12% at five covariates on Novel. The regression baselines end lowest at five covariates: Linear Regression drops from 12.1% to -49.8% and Ridge Regression from 12.2% to -49.9% , so even a linear model loses 62 points on Novel. Since the same models handle five covariates as well as one on Replay, the difficulty lies not in the number of covariates but in extrapolating their joint effect to unseen values, so strong Replay performance does not carry over to the Novel futures that scenario analysis requires.

Takeaway. The number of covariates has no clear effect on Replay, but on Novel skill degrades progressively for every model as more are set together, including the regression baselines. The LLMs lead at one covariate, but by five covariates, t0-gamma is ahead of the best LLM and TimesFM-3 is on par with it. This consistent drop points again to Novel futures as the core challenge, across all covariate settings.

4.4. Effect of the number of historical events

Figure 5 shows skill as a function of the number of historical events in the context, separately for Replay and Novel futures. On Replay futures, more history brings only marginal gains: skill stays roughly flat for most models. XGBoost is the one model that clearly gains, by about 25 points. This shows that Replay futures can be predicted with very little history, as little as a single event. However, on Novel futures the trend is far stronger: skill rises sharply as the number of historical events grows. Both the LLMs and the foundation models improve substantially: Claude Opus 4.8 (High) more than doubles from 24% to 58% and Gemini 3.1 Pro (High) from 19% to 42% , while Chronos-2 rises the most among the foundation models, from 6% to 35% . The regression baselines gain as well, Linear Regression from -27% at one event to 9% at 64, but they remain well below the best models. This indicates most models can use a longer history to forecast Novel futures more accurately. We find the same pattern with event length and context length (Appendix F): on Novel futures, skill improves in both cases whenever the context can hold more past events - i.e., both when events are shorter or the context is longer. Even with the most evidence, however, large differences between models remain: more history helps nearly every model, but models differ widely in how effectively they use it. We analyze this further in Appendix F.

Takeaway. Replay needs very little history, while on Novel most models improve as they see more past events, including the regression baselines. Even with the most events, however, Novel futures stay well below Replay, and models differ widely in how well they use the history.

5. Discussion and Future Work

We propose WHATIF-BENCH, a semi-synthetic benchmark for intervention-aware forecasting: its target series come from mechanistic simulators of real systems, on which we impose controlled effects that determine how the target responds to changes in the covariates. Each task presents a history of events and asks a model to forecast the target’s response to a further event, either a Replay event that repeats values seen in the history or a Novel event that sets them beyond that range; WHATIF-BENCH varies the number of covariates, the number of events, the event length, and the context length. Across these conditions, the frontier LLMs reach the highest skill scores overall, with the best foundation models close behind. Yet every model forecasts well on Replay futures but sees its skill drop sharply on Novel futures, and the regression baselines collapse the most. Despite their own drop, the LLMs and foundation models thus keep their clearest advantage over simple models on Novel futures. On Novel futures, skill also decreases as more covariates are set jointly, while more past events help. These results point to generalizing to Novel futures as the central open problem, and open new directions for building models that forecast reliably under novel scenarios.

We discuss the limitations of WHATIF-BENCH and future work in Appendix E. Addressing the limitations of WHATIF-BENCH opens natural extensions: richer system dynamics, and larger covariate spaces where many drivers are irrelevant, confounded, or misleading. Beyond these, the covariate-aware forecasting ability of frontier LLMs, despite their far slower inference, opens up more flexible what-if analysis, letting users explore queries that are not directly expressible through covariates, such as those posed in natural language, which could bring scenario analysis to many more users. Finally, beyond evaluating models, training them in such simulation environments could yield forecasters that learn a system in-context and reason over its possible futures - a step toward *world models* for planning and decision-making (Ha and Schmidhuber, 2018; Hafner et al., 2023; LeCun, 2022).

References

- A. Abadie, A. Diamond, and J. Hainmueller. Synthetic control methods for comparative case studies: Estimating the effect of california’s tobacco control program. *Journal of the American Statistical Association*, 105(490):493–505, 2010. DOI: 10.1198/jasa.2009.ap08746.
- T. Aksu, G. Woo, J. Liu, X. Liu, C. Liu, S. Savarese, C. Xiong, and D. Sahoo. GIFT-Eval: A benchmark for general time series forecasting model evaluation. *arXiv preprint arXiv:2410.10393*, 2024.
- A. Alexandrov, K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. Rangapuram, D. Salinas, J. Schulz, L. Stella, A. C. Türkmen, and Y. Wang. GluonTS: Probabilistic and neural time series modeling in Python. *Journal of Machine Learning Research*, 21(116):1–6, 2020. arXiv:1906.05264.
- G. Andreadis, F. Mastenbroek, V. van Beek, and A. Iosup. Capelin: Data-driven compute capacity procurement for cloud datacenters using portfolios of scenarios. *IEEE Transactions on Parallel and Distributed Systems*, 33(1):26–39, 2022.
- A. F. Ansari, L. Stella, C. Turkmen, X. Zhang, P. Mercado, H. Shen, O. Shchur, S. S. Rangapuram, S. P. Arango, S. Kapoor, et al. Chronos: Learning the language of time series. *Transactions on Machine Learning Research (TMLR)*, 2024. arXiv:2403.07815.

- A. F. Ansari, O. Shchur, J. Küken, A. Auer, B. Han, P. Mercado, S. S. Rangapuram, H. Shen, L. Stella, X. Zhang, M. Goswami, S. Kapoor, D. C. Maddix, P. Guerron, T. Hu, J. Yin, N. Erickson, P. M. Desai, H. Wang, H. Rangwala, G. Karypis, Y. Wang, and M. Bohlke-Schneider. Chronos-2: From univariate to universal forecasting, 2025. URL <https://arxiv.org/abs/2510.15821>.
- Anthropic. Claude opus 4.8 release. Announced via Anthropic News, May 2026. URL <https://www.anthropic.com/news/claude-opus-4-8>.
- A. Ashok, É. Marcotte, V. Zantedeschi, N. Chapados, and A. Drouin. Tactis-2: Better, faster, simpler attentional copulas for multivariate time series. In *International conference on learning representations*, volume 2024, pages 22897–22924, 2024.
- A. Ashok, A. R. Williams, V. Z. Zheng, I. Rish, N. Chapados, É. Marcotte, V. Zantedeschi, and A. Drouin. Beyond naïve prompting: Strategies for improved context-aided forecasting with LLMs. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL <https://openreview.net/forum?id=dkjHHFJkVI>.
- C. K. Assaad, E. Devijver, and E. Gaussier. Survey and evaluation of causal discovery methods for time series. *Journal of Artificial Intelligence Research*, 73, 2022. DOI: 10.1613/jair.1.13428.
- M. Bańbura, D. Giannone, and M. Lenza. Conditional forecasts and scenario analysis with vector autoregressions for large cross-sections. *International Journal of Forecasting*, 31(3):739–756, 2015. doi: 10.1016/j.ijforecast.2014.08.013.
- K. Benidis, S. S. Rangapuram, V. Flunkert, Y. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella, F.-X. Aubet, L. Callot, and T. Januschowski. Deep learning for time series forecasting: Tutorial and literature survey. *ACM Computing Surveys*, 55(6):1–36, 2022. doi: 10.1145/3533382.
- G. Berthelier, M. Baranova, A.-T. Pantea, E. Le Naour, A. Petralia, T. Nabil, and T. Palpanas. Investigating simple target-covariate relationships for Chronos-2 and TabPFN-TS. *arXiv preprint*, 2026. arXiv:2605.12200.
- I. Bica, A. M. Alaa, J. Jordon, and M. van der Schaar. Estimating counterfactual treatment outcomes over time through adversarially balanced representations. In *International Conference on Learning Representations (ICLR)*, 2020. arXiv:2002.04083; CRN.
- G. E. P. Box and G. C. Tiao. Intervention analysis with applications to economic and environmental problems. *Journal of the American Statistical Association*, 70(349):70–79, 1975. ISSN 01621459, 1537274X. URL <http://www.jstor.org/stable/2285379>.
- G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, fifth edition, 2015.
- K. H. Brodersen, F. Gallusser, J. Koehler, N. Remy, and S. L. Scott. Inferring causal impact using bayesian structural time-series models. *The Annals of Applied Statistics*, 9(1):247–274, 2015. DOI: 10.1214/14-AOAS788.
- D. P. Chassin, K. P. Schneider, and C. Gerkenmeyer. GridLAB-D: An open-source power systems modeling and simulation environment. In *2008 IEEE/PES Transmission and Distribution Conference and Exposition*, pages 1–5. IEEE, 2008. doi: 10.1109/TDC.2008.4517260.
- A. Das, W. Kong, R. Sen, and Y. Zhou. A decoder-only foundation model for time-series forecasting. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2310.10688.

- T. Doan, R. Litterman, and C. Sims. Forecasting and conditional projection using realistic prior distributions. *Econometric Reviews*, 3(1):1–100, 1984. doi: 10.1080/07474938408800053.
- V. Dorie, J. Hill, U. Shalit, M. Scott, and D. Cervone. Automated versus do-it-yourself methods for causal inference: Lessons learned from a data analysis competition. *Statistical Science*, 34(1):43–68, 2019.
- A. Drouin, É. Marcotte, and N. Chapados. Tactis: Transformer-attentional copulas for time series. In *International Conference on Machine Learning*, pages 5447–5493. PMLR, 2022.
- V. Ekambaram, A. Jati, P. Dayama, S. Mukherjee, N. H. Nguyen, W. M. Gifford, C. Reddy, and J. Kalagnanam. Tiny time mixers (TTMs): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. *Advances in Neural Information Processing Systems*, 37: 74147–74181, 2024.
- E. S. Gardner Jr. Exponential smoothing: The state of the art. *Journal of Forecasting*, 4(1):1–28, 1985.
- R. Godahewa, C. Bergmeir, G. I. Webb, R. J. Hyndman, and P. Montero-Manso. Monash time series forecasting archive. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track (NeurIPS)*, 2021. arXiv:2105.06643.
- Google DeepMind. Gemini 3.1 Pro: Model Card, 2026. URL <https://deepmind.google/models/model-cards/gemini-3-1-pro/>. Accessed: 2026-09-20.
- Google Research. TimesFM 3.0: A zero-shot foundation model for multivariate forecasting. <https://research.google/blog/timesfm-3-a-zero-shot-foundation-model-for-multivariate-forecasting/>, 2026. Model release; successor to TimesFM 2.5 with multivariate/covariate support.
- M. Goswami, K. Szafer, A. Choudhry, Y. Cai, S. Li, and A. Dubrawski. MOMENT: A family of open time-series foundation models. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.03885.
- N. Gruver, M. Finzi, S. Qiu, and A. G. Wilson. Large language models are zero-shot time series forecasters. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2310.07820.
- D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.
- D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint*, 2023. arXiv:2301.04104.
- J. L. Hill. Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240, 2011.
- P. W. Holland. Statistics and causal inference. *Journal of the American Statistical Association*, 81(396): 945–960, 1986.
- S. B. Hoo, S. Müller, D. Salinas, and F. Hutter. From tables to time: Extending TabPFN-v2 to time series forecasting, 2025. URL <https://arxiv.org/abs/2501.02945>.
- R. J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.
- G. W. Imbens and D. B. Rubin. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press, 2015.

- A. Iosup, G. Andreadis, V. van Beek, M. Bijman, E. Van Eyk, M. Neacsu, L. Overweel, S. Talluri, L. Versluis, and M. Visser. The OpenDC vision: Towards collaborative datacenter simulation and exploration for everybody. In *2017 16th International Symposium on Parallel and Distributed Computing (ISPDC)*, pages 85–94. IEEE, 2017.
- B. Jäger, N. Erickson, L. Grinsztajn, F. Birkel, K. Flöge, O. Key, K. Kaya, J. Kübler, A. Frankel, T. Schröder, A. Garg, J. H. Metzen, D. Salinas, S. Bing, K. Collins, T. Çelik, V. Balazadeh, L. Sidhoum, T. Pereda, B. Roof, A. Tschalzev, S. Guo, P. Singer, L. Purucker, J. Robertson, M. Salmon, P. Jund, J. Chen, D. Kriuchkova, A. Cahu, E. Kalfon, A. Hayler, G. Grab, V. Monteiro, L. Wehrhahn, D. Safaric, C. Cornu, A. Arazi, R. Grace, S. Alessi, M. Manium, B. Schölkopf, Y. LeCun, M. Hulsebos, S. Gambhir, N. Hollmann, and F. Hutter. TabPFN-3.5: Technical report, 2026. URL <https://arxiv.org/abs/2609.17895>.
- M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan, and Q. Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2310.01728.
- Kaggle. Walmart Recruiting – Store Sales Forecasting. Kaggle competition, 2014. URL <https://www.kaggle.com/competitions/walmart-recruiting-store-sales-forecasting>.
- Kaggle. Rohlik Sales Forecasting Challenge V2. Kaggle competition, 2025. URL <https://www.kaggle.com/competitions/rohlik-sales-forecasting-challenge-v2/data>.
- U. Kępa and I. Deska. Using a hydraulic model for conceptual planning of rural water supply network reconstruction—case study. *Water*, 17(20):2961, 2025. doi: 10.3390/w17202961.
- J. Lago, G. Marcjasz, B. De Schutter, and R. Weron. Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark. *Applied Energy*, 293: 116983, 2021. arXiv:2008.08004.
- E. Le Naour, T. Nabil, and A. Petralia. TS-ICL: A flexible time-indexed foundation model for time series via in-context learning, 2026. URL <https://arxiv.org/abs/2606.05878>.
- Y. LeCun. A path towards autonomous machine intelligence. *OpenReview preprint*, 2022. OpenReview:BZ5a1r-kVsf.
- G. Lee, W. Yu, K. Shin, W. Cheng, and H. Chen. TimeCAP: Learning to contextualize, augment, and predict time series events with large language model agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025. arXiv:2502.11418.
- Z. Li, X. Qiu, P. Chen, Y. Wang, H. Cheng, Y. Shu, J. Hu, C. Guo, A. Zhou, C. S. Jensen, and B. Yang. TSFM-Bench: A comprehensive and unified benchmark of foundation models for time series forecasting. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2025. arXiv:2410.11802.
- Y. Liang, H. Wen, Y. Nie, Y. Jiang, M. Jin, D. Song, S. Pan, and Q. Wen. Foundation models for time series analysis: A tutorial and survey. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2024. doi: 10.1145/3637528.3671451.
- B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.
- H. Liu, S. Xu, Z. Zhao, L. Kong, H. Kamarthi, A. B. Sasanur, M. Sharma, J. Cui, Q. Wen, C. Zhang, and B. A. Prakash. Time-MMD: Multi-domain multimodal dataset for time series analysis. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2024a. arXiv:2406.08627.

- X. Liu, J. Hu, Y. Li, S. Diao, Y. Liang, B. Hooi, and R. Zimmermann. Unitime: A language-empowered unified model for cross-domain time series forecasting. In *Proceedings of the ACM Web Conference 2024*, pages 4095–4106, 2024b.
- P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using SUMO. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582. IEEE, 2018. doi: 10.1109/ITSC.2018.8569938.
- Q.-L. Lu, M. Qurashi, and C. Antoniou. Simulation-based policy analysis: The case of urban speed limits. *Transportation Research Part A: Policy and Practice*, 175:103754, 2023.
- S. Makridakis and M. Hibon. The M3-Competition: Results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476, 2000.
- S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The M4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018.
- S. Makridakis, E. Spiliotis, and V. Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364, 2022.
- F. Mastenbroek, G. Andreadis, S. Jounaid, W. Lai, J. Burley, J. Bosch, E. Van Eyk, L. Versluis, V. van Beek, and A. Iosup. OpenDC 2.0: Convenient modeling and simulation of emerging technologies in cloud datacenters. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 455–464. IEEE, 2021.
- L. Meyer, C. Sole, H. Xiang, N. Li, L. Franceschino, A. Quera-Bofarull, M. P. Scholl, J. Fainberg, and G. Négier. t_0 : A time-series foundation model for forecasting with context, 2026. URL <https://arxiv.org/abs/2609.24559>.
- Mistral AI. Timestral. <https://mistral.ai/news/timestral/>, 2026. Model release; checkpoint mistralai/Timestral-209M.
- Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2211.14730.
- B. N. Oreshkin, D. Carpo, N. Chapados, and Y. Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020.
- J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, NY, USA, 2nd edition, 2009. ISBN 978-0521895606.
- J. Peters, P. Bühlmann, and N. Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 78(5):947–1012, 2016.
- P. Podest, M. Pichler, E. Bürger, L. Zólyomi, B. Voggenberger, W. Berghammer, D. Klotz, S. Böck, G. Klambauer, and S. Hochreiter. TiRex-2: Generalizing TiRex to multivariate data and streaming. *arXiv preprint*, 2026. arXiv:2607.01204.
- Z. Qiao, S. Pan, A. Wang, V. Zhukova, Y. Liu, X. Jiang, Q. Wen, M. Long, M. Jin, and C. Liu. It’s TIME: Towards the next generation of time series forecasting benchmarks. *arXiv preprint arXiv:2602.12147*, 2026.

- K. Rasul, A. Ashok, A. R. Williams, A. Khorasani, G. Adamopoulos, R. Bhagwatkar, M. Biloš, H. Ghonia, N. V. Hassen, A. Schneider, et al. Lag-Llama: Towards foundation models for probabilistic time series forecasting. *arXiv preprint arXiv:2310.08278*, 2023.
- J. Requeima, J. Bronskill, D. Choi, R. E. Turner, and D. Duvenaud. LLM processes: Numerical predictive distributions conditioned on natural language. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2405.12856.
- J. M. Robins, M. Á. Hernán, and B. Brumback. Marginal structural models and causal inference in epidemiology. *Epidemiology*, 11(5):550–560, 2000. DOI: 10.1097/00001648-200009000-00011.
- M. Rojas-Carulla, B. Schölkopf, R. Turner, and J. Peters. Invariant models for causal transfer learning. *Journal of Machine Learning Research*, 19(36):1–34, 2018.
- L. A. Rossman. EPANET 2 users manual. Technical Report EPA/600/R-00/057, U.S. Environmental Protection Agency, Cincinnati, OH, 2000.
- J. Runge, P. Nowack, M. Kretschmer, S. Flaxman, and D. Sejdinovic. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science Advances*, 5(11), 2019. PCMCi; DOI: 10.1126/sciadv.aau4996.
- D. Salinas, V. Flunkert, and J. Gasthaus. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020. arXiv:1704.04110.
- K. P. Schneider, J. C. Fuller, F. K. Tuffner, and R. Singh. Evaluation of conservation voltage reduction (CVR) on a national level. Technical Report PNNL-19596, Pacific Northwest National Laboratory, 2010.
- O. Shchur, A. F. Ansari, C. Turkmen, L. Stella, N. Erickson, P. Guerron, M. Bohlke-Schneider, and Y. Wang. fev-bench: A realistic benchmark for time series forecasting. *arXiv preprint arXiv:2509.26468*, 2025.
- D. Thumm, B. T. Anthony, and Y. Chen. DoTime: A synthetic benchmark generator for interventional and counterfactual time series. *arXiv preprint arXiv:2607.27263*, 2026.
- D. F. Waggoner and T. Zha. Conditional forecasts in dynamic multivariate models. *The Review of Economics and Statistics*, 81(4):639–651, 1999. doi: 10.1162/003465399558508.
- X. Wang, M. Feng, J. Qiu, J. Gu, and J. Zhao. From news to forecast: Integrating event analysis in LLM-based time series forecasting with reflection. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2409.17515.
- A. R. Williams, A. Ashok, É. Marcotte, V. Zantedeschi, J. Subramanian, R. Riachi, J. Requeima, A. Lacoste, I. Rish, N. Chapados, and A. Drouin. Context is key: A benchmark for forecasting with essential textual information. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. arXiv:2410.18959.
- G. Woo, C. Liu, A. Kumar, C. Xiong, S. Savarese, and D. Sahoo. Unified training of universal time series forecasting transformers. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.02592.
- H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 22419–22430, 2021. arXiv:2106.13008.

- Z. Xu, H. Wang, and Q. Xu. Intervention-aware forecasting: Breaking historical limits from a system perspective. *arXiv preprint arXiv:2405.13522*, 2024.
- H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11106–11115, 2021. arXiv:2012.07436.

Table of Contents for Appendix

A	Extended Related Work	21
A.1	Time Series Foundation Models	21
A.2	Large Language Models for Forecasting	21
B	Benchmark details.	21
B.1	Rationale behind semi-synthetic setup	21
B.2	Verification	22
B.3	Task layout	23
B.4	Task generation procedure	23
B.5	Design choices: identifiability and evaluation	24
B.6	Simulator Details	25
C	Aggregation and Metrics.	36
C.1	Per-task scoring	36
C.2	Aggregates	36
C.3	Metrics	37
D	Experimental Setup.	37
D.1	Foundation models	37
D.2	Large language models.	38
D.3	Regression baselines	39
E	Limitations and Future Work	39
F	Skill across the number of events, event length and context length	40
G	Skill scores under WAPE.	43
H	Results by Simulator	47
I	Skill Against a Covariate-Aware Baseline	50
J	Tuned Regression Baselines	56
K	Win Rates.	61
L	Example tasks	65
L.1	Replay and Novel futures	65
L.2	Event length	69

L.3	Number of events	71
L.4	Context and prediction length	73
L.5	Number of covariates	74

A. Extended Related Work

A.1. Time Series Foundation Models

Time series forecasting has moved from classical methods (Box et al., 2015; Gardner Jr., 1985; Hyndman and Athanasopoulos, 2018) and deep networks (Ashok et al., 2024; Benidis et al., 2022; Drouin et al., 2022; Lim et al., 2021; Nie et al., 2023; Salinas et al., 2020; Wu et al., 2021; Zhou et al., 2021) to time series foundation models (TSFMs), pretrained once on large corpora to forecast unseen series zero-shot, often outperforming dataset-specific models (Ansari et al., 2024; Goswami et al., 2024; Rasul et al., 2023; Woo et al., 2024). These models were long univariate; using the external covariates many tasks depend on required task-specific fine-tuning (Shchur et al., 2025). Some foundation models can use covariates only after such adaptation, such as Moirai (Woo et al., 2024) and TTM (Ekambaram et al., 2024). The most recent TSFMs instead consume covariates in-context, inferring how they affect the target from the examples alone and factoring this into the forecast without any training (Ansari et al., 2025; Google Research, 2026; Le Naour et al., 2026; Meyer et al., 2026; Podest et al., 2026). We focus on this zero-shot setting, as in fev-bench (Shchur et al., 2025). WHATIF-BENCH evaluates these covariate-aware TSFMs for arbitrary scenario analysis: whether they can forecast the target accurately when covariates are set to unseen future values.

A.2. Large Language Models for Forecasting

Large language models (LLMs), though not built for forecasting, have also been repurposed for it: prompted with a numerical time series as text, a pretrained LLM can continue it zero-shot (Gruver et al., 2023; Requeima et al., 2024), and later work uses LLMs as strong forecasting baselines (Ashok et al., 2026; Williams et al., 2025). Their performance improves with fine-tuning or adaptation for time series, as in Time-LLM (Jin et al., 2024) and UniTime (Liu et al., 2024b), and they have been explored for multimodal forecasting, using textual context such as news (Lee et al., 2025; Liu et al., 2024a; Wang et al., 2024; Williams et al., 2025). Our focus is on zero-shot forecasting, so we prompt LLMs directly rather than adapting them, as in a few recent forecasting studies (Ashok et al., 2026; Williams et al., 2025). We evaluate frontier LLMs in WHATIF-BENCH, and find they are in fact the strongest models we test, outperforming covariate-aware TSFMs both on tasks with repeated future covariate values and on tasks requiring extrapolation to unseen values.

B. Benchmark details

B.1. Rationale behind semi-synthetic setup

Intervention-aware forecasting needs an environment where the target’s response to a covariate is known exactly at any value the covariate takes, including values never observed in the history. Real data does not meet this need: it gives one realization of the covariates and targets, and running controlled A/B tests to see how the target moves is usually infeasible (Holland, 1986; Imbens and Rubin, 2015). This is what motivates a semi-synthetic setup, where a known, generated process supplies exact counterfactuals (Dorie et al., 2019; Hill, 2011).

The simulators alone do not meet this need either. They are built mainly to generate realistic dynamics for a domain, and they do this well: running one on a base setting gives a target series that is realistic and forecastable on its own. Interventions can be performed on them, but not the kind we need. To construct our tasks we would have to re-run the simulator for every event and every covariate value, including the novel values that fall outside the range a simulator is calibrated for, and we found the simulators were not robust to this. Interventions in OpenDC often shift the system into a new

regime, producing a change that is not forecastable from the history; in GridLAB-D most interventions required several attempts to run and some failed outright; in SUMO the covariates we wanted moved the target reliably only at large factors, too coarse to span the range that novel extrapolation needs; and across simulators, setting several covariates at once rarely produced a clear, separable effect on the target.

A further problem is noise. The simulators add stochastic variation to the target, so the ground-truth series carries noise unrelated to the covariates. This makes evaluation unreliable: even when a model uses the covariates correctly, the noise in the ground truth inflates the error and drives metrics down, so scores reflect the noise rather than the model’s grasp of the covariate–target relationships. Imposing the relationships ourselves gives a clean ground truth, on which such scores are meaningful.

We therefore keep the simulators for what they are good at, the realistic baseline series, and impose the covariate–target relationships ourselves. Each relationship is derived from the simulator’s documented model and reduced to a smooth, fixed form that holds at every factor value: faithful to the system under simplifying assumptions, but deliberately simple so it stays reliable and exactly reproducible, not an attempt to reproduce the simulator’s exact dynamics. Under other operating regimes some effects could take a different sign or shape (for instance, a network quantity’s response to added demand depends on where the bottleneck is), but we do not consider this a limitation for benchmarking: WHATIF-BENCH does not test physics, only whether a model learns a covariate–target relationship from its context. Every foundation model sees only numbers, and the LLMs see dummy variable names (Appendix D), so no model can tell one form of a relationship from another except through the data, and results do not depend on whether a relationship matches a practitioner’s expectation. We further document every relationship used in the benchmark (Appendix B.6).

This setup has limitations. The relationships are hand-engineered, and building them for each simulator took manual effort. We see domain experts building bespoke, faithful benchmarks for each use case as the ideal, and leave this to future work. We view WHATIF-BENCH as complementary to benchmarks such as fev-bench (Shchur et al., 2025), and recommend using it alongside them: our axes are designed to cover several regimes of real-world data as proxies, and to test regimes that are only reachable with semi-synthetic data, such as novel interventions and events with several covariates.

B.2. Verification

For a task to be useful, each event in it must leave an effect on the target that is large enough to measure and clearly attributable to the covariates. We keep a covariate–target pair only if the covariate moves the target by a measurable amount and in a consistent direction, so the change can be attributed to the covariate rather than the target’s own variation. Some relationships are not monotone over all factors (for example, SUMO flow is piecewise), but the factors we use reach only one branch, so each target moves the same way in every event. Concretely, we apply a 10% criterion: every event must move the target at least 10% away from its intervention-free baseline over the event window, measured on average across the window rather than at each timestep. This threshold keeps the effect above the level at which the metrics could measure it reliably. We check every event in a task, and drop the whole task if any single event fails the criterion. One alternative would be to retune the equations or factors to force the effect through, but we instead drop the failing tasks, since only a small fraction fall short: this leaves 37,665 of the 40,000 generated tasks.

B.3. Task layout

Within a task’s history, the events do not fill the whole context. We lay out each history as an observational prefix, followed by the events, each separated from the next by a gap, and a small buffer at the end. The layout is:

- **Observational prefix.** The first 40% of the context holds no events: the covariates sit at their base setting and the target evolves on its own. This lets a model see the target’s intrinsic dynamics, apart from any intervention, so it can separate them from the covariate-driven changes later.
- **Events.** Each event is exactly the event length ℓ long, and the events are placed one after another through the rest of the history.
- **Gaps.** Every event is followed by a gap of at least 32 timesteps, so that one event never runs into the next and the target can settle back toward its baseline between events.
- **End buffer.** The last 2% of the context is left clear, keeping the final event away from the forecast boundary.

The observational prefix, events, and gaps together fill the context, and the split depends on how many events a task has. At the most events a context and event length allow, the layout reduces to the 40% prefix above; with fewer events, the extra room is added to the prefix, so the events stay anchored toward the recent end of the history. The exact prefix length is therefore set per task, from its context length, event length, and number of events. If the events and gaps do not fit in the nominal context, we extend the context instead of dropping events (Table 4). This layout is a design choice specific to WHATIF-BENCH, made mainly for clear benchmarking and evaluation; we keep these parameters fixed and instead vary the main axes described in Section 3.3.

Table 4 gives the nominal context and prediction lengths of the four pairs C_1 – C_4 for each simulator. The three smallest pairs are shared; only the largest, C_4 , varies slightly across simulators. For the longest events (length 512), we extend the context so that the events fit (to 1,876 at C_1 , and to 7,504 at C_4 on OpenDC).

Pair	Context / prediction length		
	EPANET, GridLAB-D	OpenDC	SUMO
C_1	1024 / 128	1024 / 128	1024 / 128
C_2	2048 / 256	2048 / 256	2048 / 256
C_3	4096 / 384	4096 / 384	4096 / 384
C_4	8192 / 512	7328 / 864	7776 / 864

Table 4 | Nominal context and prediction lengths (in timesteps) of the four pairs C_1 – C_4 , per simulator.

B.4. Task generation procedure

We generate the benchmark by repeating one procedure over every point of the axes grid. We iterate over each simulator, each of its targets, and each setting of the axes: the number of covariates K , the context/prediction pair, the event length, and the number of events (Table 2). For a single such setting, we build a task as follows.

1. **Take the baseline series.** Use the intervention-free target series, produced by running the simulator once on its base setting, and lay out the history with the observational prefix, events, and gaps described above.

2. **Pick the covariates.** For a task with K covariates, select K of the target’s covariates, including its primary one, to act together in each event.
3. **Pick the factors.** For each event in the history, draw a multiplicative factor for each active covariate from a fixed pool of factors (given per simulator in Appendix B.6). Across the events, the factors take a spread of values, so the covariate–target relationship is seen at several points.
4. **Apply the events.** At each event, scale the active covariates by their factors over the event window, following the assigned shape, and move the target by the amount the imposed covariate–target relationship prescribes. When several covariates are active, combine their effects using the rules in Appendix B.6.
5. **Build the two futures.** Over the same history, form two future events. The Replay future reuses the covariate values of a past event; the Novel future pushes them beyond the range seen in the history. The Novel factor is set relative to the most extreme factor in the history: the largest one plus an offset drawn uniformly from a fixed range (or, for a covariate with an inverse relationship, the smallest one scaled by a factor drawn from a fixed range), so the extrapolation distance beyond the seen range does not depend on the number of events. This yields two tasks that share a history and differ only in the future.
6. **Verify and keep.** Check the 10% criterion on every event, and keep the two tasks only if all events pass; otherwise drop them.

Repeating this over all targets and axis settings produces the full benchmark of 37,665 verified tasks.

Generation parameters. For completeness, we give the values fixed in the generation code.

- **Covariate selection.** Each simulator fixes an ordered list of covariates. A task with K covariates uses the target’s primary covariate followed by the first $K - 1$ other covariates in that order, so the covariate set is deterministic given the target and K .
- **Bell shape.** Over an event window of length ℓ with timesteps $t = 0, \dots, \ell - 1$, a bell-shaped covariate with peak factor f takes the factor $1 + (f - 1) \exp(-\frac{1}{2}((t - t_m)/\sigma)^2)$, with $t_m = (\ell - 1)/2$ and $\sigma = \ell/6$. It rises from the base value to f at the window midpoint and decays back.
- **History factors.** Each event draws a multiplicative factor per active covariate from a fixed pool: $\{0.1, 0.25, 0.5, 2, 4, 6, 8\}$ for OpenDC, GridLAB-D, and SUMO, and $\{2, 4, 6, 8\}$ for EPANET, whose covariates are only ever raised. Within an event, all active covariates push the target in one direction, chosen at random per event: a covariate with a direct relationship draws from the pool values above 1 and one with an inverse relationship from the values below 1, or the reverse for the opposite direction. This keeps the covariates’ effects from cancelling each other within an event.
- **Novel factors.** For a covariate with a direct relationship, the Novel factor is the largest factor it took in the history plus an offset drawn uniformly from $[5, 12]$. For an inverse relationship, it is the smallest history factor times a value drawn uniformly from $[0.1, 0.5]$.

B.5. Design choices: identifiability and evaluation

Identifiability. We do not constrain the number or placement of events to guarantee that every covariate–target relationship is identifiable from a task’s history. The evidence a task offers may be insufficient to learn them – most clearly when many covariates are set but few events occur, but in other cases too, since it also depends on where each event acts on the target (for instance, an event applied at an already-high level can confound its effect). We treat the available evidence as a controlled axis: the number of historical events sets how much a model has to learn from, ranging from evidence-poor tasks (few events, many covariates) to evidence-rich ones. We report these regimes separately, and practitioners can filter to the high-event regime to focus on near-identifiable tasks.

Including tasks with limited historical evidence is deliberate and, we argue, principled, for three reasons. First, they are the norm in real scenario analysis, where a useful forecast is still expected under limited evidence. Second, our metric rewards the correct response to this: wQL is a proper scoring rule (Appendix C), so a forecaster that widens its predictive distribution under weak evidence is scored well, while an overconfident one is penalized – especially on Novel futures.³ Third, the skill score measures improvement over a fixed baseline on the same task, so tasks that no forecaster can resolve do not distort the aggregate. The task also remains well-posed throughout: the simulator defines one true target trajectory, so limited identifiability reflects limited evidence, not an undefined target – as in real forecasting. With enough events, every relationship can in principle be learned: it is exact, and it does not change direction over the factors we use. We leave designs that target identifiability directly – for example, restricting to identifiable tasks to test identification of every relationship, or scoring calibrated abstention on tasks that are unidentifiable – to future work.

B.6. Simulator Details

For each simulator we give the base setting we run it on, the variables we take as targets and the properties we use as covariates, the shape assigned to each covariate, and the covariate–target relationships we impose. Throughout, y denotes the intervention-free baseline value of a target at a timestep, produced by running the simulator, and f denotes the multiplicative factor an event applies to a covariate (so $f = 1$ leaves the covariate at its base setting). A relationship maps (y, f) to the target value under the event.

The four simulators span distinct domains. **OpenDC** models a data center’s resource usage for capacity planning – for example, how CPU utilization and power draw respond as servers are added to a cluster. **GridLAB-D** models a power-distribution grid and the buildings on it, used to study how electricity demand responds to control – for example, how a building’s power use and indoor temperature respond when its thermostat set-point is raised. **SUMO** models road traffic vehicle by vehicle to evaluate traffic policies – for example, how traffic flow responds when a road’s speed limit is lowered. **EPANET** models pressurized water-distribution networks to study pump operation and demand growth – for example, how network pressure responds to a surge in demand. All covariates are numerical, since the simulator properties they come from take numerical values.

How the relationships are designed. Every covariate–target relationship below follows the same rules:

- **Grounded in the simulator.** Each relationship is derived from the simulator’s documented model of the system, and restated in closed form. These are the power and performance model of OpenDC, the residential load and power-flow model of GridLAB-D, the hydraulic model of EPANET, and the car-following, emission, and detector models of SUMO.
- **Direction from the system.** A covariate moves the target in the direction the real system would. For example, more cores lower the per-core utilization, and a higher clock speed raises the power draw.
- **Monotonic within the factors we use.** Each relationship is monotonic over the factors an event can draw, so the target always moves in one direction as the factor grows. Relationships that are piecewise in general, such as SUMO flow, stay on one branch.
- **Varied in form.** The forms include proportional, inverse, sublinear, faster-than-linear, exponential, and asymmetric relationships. A factor of two on a covariate need not mean a factor of two on the target.
- **Combined in a fixed way.** When an event sets several covariates at once, the secondary covariates act on the target either multiplicatively or additively, as given for each target.
- **Large enough to measure.** Every event moves the target by at least 10% from its baseline, averaged over the event window. We drop any task in which an event does not.

³This applies to the foundation models, which emit quantiles; the LLMs return only a median point forecast and so cannot express this uncertainty through their intervals, as noted in Appendix D.

The rest of this section gives, for each simulator, the exact forms and constants, for reproducibility.

B.6.1. OpenDC

Base setting. We run OpenDC on a single rack of four identical hosts (host-0–host-3), each with 16 cores at a core speed of 3000 and 64 GB of memory, driven by the Bitbrains workload trace of four virtual machines.⁴ Running it exports the resource usage of the hosts and virtual machines at a fixed interval, which forms the baseline series.

Targets and covariates. We take 22 target series and use six covariates, named as they appear in the released benchmark:

- **Host targets (10).** `cpu_demand_host-0`, `cpu_usage_host-0`, `cpu_utilization_host-0`, `cpu_time_active_host-0`, `power_draw_host-0`, and the same five for host-1.
- **Virtual machine targets (12).** For each of the VMs 1019, 1023 (running on host-0) and 1026 (running on host-1): `cpu_demand_<vm>`, `cpu_usage_<vm>`, `cpu_time_active_<vm>`, and `cpu_time_idle_<vm>`, e.g. `cpu_demand_1019`.
- **Covariates (6).** `coreCount_host-0`, `coreSpeed_host-0`, `memorySize_host-0`, and the same three for host-1.

Each target has one *primary* covariate that drives it most directly; the pairing is given in Table 5. The other covariates can also act on a target as *secondary* drivers when an event sets several covariates at once (described below).

Covariate shapes. The `coreSpeed` covariates (`coreSpeed_host-0`, `coreSpeed_host-1`) are assigned the bell shape (clock speed could be ramped up for a period and then brought back down). The `coreCount` and `memorySize` covariates are assigned the step shape (a hardware allocation is set to a level and held there for the duration of the event). Each event draws its multiplicative factor for a covariate from the pool {0.1, 0.25, 0.5, 2, 4, 6, 8}.

Primary relationships. Each primary relationship maps the baseline value y and the primary covariate’s factor f to the target under the event. The relationships below are derived from OpenDC’s documented power and performance model (Mastenbroek et al., 2021);⁵ we restate them in closed form and give a short description of the behavior each one captures. We give the forms for host-0 below; host-1 uses the same forms with slightly different constants, and the virtual machine targets use the same forms with per-machine constants.

- **`cpu_demand_host-0`, `cpu_usage_host-0` (from `coreSpeed_host-0`).** Throughput scales close to linearly with clock frequency, with a small fixed overhead:

$$\begin{aligned}\text{cpu_demand_host-0} : y' &= f y + 0.05 (f - 1), \\ \text{cpu_usage_host-0} : y' &= (0.98f + 0.02) y + 0.15 (f - 1).\end{aligned}$$

- **`power_draw_host-0` (from `coreSpeed_host-0`).** A host draws a fixed idle power (about 105.5 W) plus a dynamic part that scales with frequency, with a small switching term:

$$y' = 105.5 + f (y - 105.5) + 2.2 |f - 1|.$$

⁴<https://github.com/atlarge-research/quantifying-api-design/tree/master/traces/bitbrains-small>

⁵<https://opendc.org>

Target	Primary covariate	Combination
cpu_demand_host-0	coreSpeed_host-0	multiplicative
cpu_usage_host-0	coreSpeed_host-0	multiplicative
power_draw_host-0	coreSpeed_host-0	additive
cpu_utilization_host-0	coreCount_host-0	additive
cpu_time_active_host-0	coreCount_host-0	multiplicative
cpu_demand_1019	coreSpeed_host-0	multiplicative
cpu_usage_1019	coreSpeed_host-0	multiplicative
cpu_time_active_1019	coreCount_host-0	multiplicative
cpu_time_idle_1019	memorySize_host-0	multiplicative
cpu_demand_1023	coreSpeed_host-0	multiplicative
cpu_usage_1023	coreSpeed_host-0	multiplicative
cpu_time_active_1023	coreCount_host-0	multiplicative
cpu_time_idle_1023	memorySize_host-0	multiplicative

Table 5 | **OpenDC target-covariate pairing (host-0 side).** Each target is driven by one primary covariate; whether secondary covariates combine multiplicatively or additively is set by the target’s class. The host-1 targets and VM 1026 pair with the host-1 covariates in the same way.

- **cpu_utilization_host-0 (from coreCount_host-0).** The same work spread over more cores lowers per-core utilization inversely:

$$y' = \frac{1}{f} y.$$

- **cpu_time_active_host-0 (from coreCount_host-0).** Adding cores raises the total active core-time sublinearly, reflecting imperfect parallel speedup:

$$y' = (1 + 0.4(f - 1)) y.$$

- **cpu_time_idle_1019 (from memorySize_host-0).** Adding memory slightly reduces idle waiting; shrinking memory raises it sharply as the machine begins to swap pages. The relationship is therefore asymmetric in f :

$$y' = \begin{cases} (0.28 + \frac{0.72}{f}) y, & f \geq 1, \\ (1 + 3.2(1 - f)) y + 12(1 - f), & f < 1. \end{cases}$$

Combining multiple covariates. When an event sets more than one covariate, the target starts from its primary relationship and each additional active covariate applies a further, dampened effect. A target’s class fixes whether these effects combine multiplicatively (cpu_demand, cpu_usage, cpu_time_active, cpu_time_idle) or additively (power_draw, cpu_utilization). Let n be the number of active secondary covariates in the event and set the joint dampening

$$s = \begin{cases} 1, & n \leq 1, \\ \frac{1}{\sqrt{n}}, & n > 1, \end{cases}$$

so that the more covariates move together, the less each one contributes. Each secondary covariate c also carries a fixed weight w_c that orders it by importance. For a multiplicative target with a secondary covariate whose own effect is r , and an additive target with a secondary offset o , the running target value is updated as

$$\text{multiplicative: } y \leftarrow y [1 + (s w_c) (r - 1)], \quad \text{additive: } y \leftarrow y + (s w_c) o.$$

Target class	Secondary covariate	Weight w_c	Effect (r or o)
CPU demand / usage (mult.)	core count, same host	0.60	$r = f^2$
	memory size, same host ($f < 1$)	0.40	$r = 10/f$
	core speed, other host ($f < 1$)	0.35	$r = 5(2 - f)$
	core count, other host	0.30	$r = (2 - f)^3$
	memory size, other host	0.20	$r = 1 + 3/f$
Active CPU time (mult.)	core speed, same host	0.80	$r = 1/f^3$
	memory size, same host ($f < 1$)	0.50	$r = 500/f$
	core speed, other host ($f < 1$)	0.35	$r = 400(2 - f)$
	core count, other host	0.30	$r = 300(2 - f)$
	memory size, other host	0.20	$r = 200(2 - f)$
Idle CPU time (mult.)	core speed, same host	0.60	$r = f^2$ if $f_m \geq 1$, else $1/f^2$
	core count, same host	0.50	$r = 1/f^3$
	core speed, other host ($f < 1$)	0.30	$r = f$
	core count, other host ($f < 1$)	0.20	$r = f$
	memory size, other host ($f < 1$)	0.20	$r = f$
Power draw (add.)	core count, same host	0.70	$o = 50(f - 1)$ if $f \geq 1$, else $-50(1 - f)$
	memory size, same host	0.50	$o = 150(f - 1)$ if $f \geq 1$, else $300(1 - f)$
	core speed, other host ($f < 1$)	0.35	$o = 250(1 - f)$
	core count, other host	0.30	$o = 200(1 - f)$
	memory size, other host	0.20	$o = 100(1 - f)$
CPU utilization (add.)	core speed, same host	0.70	$o = -0.05(f - 1)$ if $f \geq 1$, else $0.15(1 - f)$
	memory size, same host ($f < 1$)	0.55	$o = 2(1 - f)$
	core speed, other host ($f < 1$)	0.35	$o = 3(1 - f)$
	core count, other host	0.30	$o = 2(1 - f)$
	memory size, other host	0.20	$o = 1(1 - f)$

Table 6 | **OpenDC secondary covariate effects (host-0 targets).** For a target, each secondary covariate contributes an effect scaled by its weight w_c and the joint dampening s . Multiplicative classes use the raw effect r ; additive classes use the offset o . Here f is the secondary covariate’s own factor, and f_m the memory-size factor. Effects marked ($f < 1$) apply only when that covariate is reduced. host-1 targets use the same template with mirrored hosts.

The weights w_c and the effects r and o are listed in Table 6. Two patterns are worth noting. First, a covariate on the *same* host as the target acts in the opposite direction to the *other* host’s covariate: reducing a remote host’s resources pushes load onto the local host and raises its demand, active time, and power, reflecting consolidation and contention across the shared rack. Second, some effects only apply in one direction. For example, memory acts only when it shrinks (when $f < 1$, as swapping begins), and a remote host’s speed acts only when it is reduced. The final target value is clipped at zero.

The direction and qualitative form of every effect follow the mechanisms above. The magnitudes (the weights w_c and the constants in r and o) are chosen so that each covariate leaves a clear, separable imprint on the target, rather than being derived from the simulator.

B.6.2. GridLAB-D

Base setting. We run GridLAB-D 5.3.0 with its `climate`, `powerflow`, `residential`, and `tape` modules, driven by Seattle TMY3 weather data. The topology is a single-house feeder: a SWING source node at 2400 V feeds a 75 kVA center-tapped transformer, then a 120 V triplex node, then a single house (1550 ft², heating set-point 60 °F, cooling set-point 77 °F, thermostat deadband 1.6 °F) with a constant 10 kW ZIP load (impedance/current/power fractions 0.3/0.5/0.2). Outputs are recorded every 60 seconds, which forms the baseline series.

Target	Primary covariate	Combination
actual_power	base_power	multiplicative
total_load	nominal_power	multiplicative
fan_power	power_pf	multiplicative
hvac_load	voltage_factor	multiplicative
hvac_power	duty_cycle	multiplicative
heating_demand	rate_of_heating	multiplicative
air_temperature	heat_fraction	additive
mass_temperature	temperature	additive
is_COOL_on	thermostatic_control_range	additive
is_HEAT_on	recovery_duty_cycle	additive
is_AUX_on	period	additive

Table 7 | **GridLAB-D target-covariate pairing.** Each target is driven by one primary covariate; whether secondary covariates combine multiplicatively or additively is set by the target’s class.

Targets and covariates. We generate tasks for the 11 target series below and use 21 covariates, named as they appear in the released benchmark. One target, `is_COOL_on`, fails the 10% verification criterion in every task and is dropped entirely, leaving 10 GridLAB-D targets in the released benchmark:

- **Power targets (5).** `actual_power`, `total_load`, `hvac_load`, `hvac_power`, `fan_power`.
- **Temperature targets (2).** `air_temperature`, `mass_temperature`.
- **Demand target (1).** `heating_demand`.
- **State targets (3).** `is_AUX_on`, `is_HEAT_on`, `is_COOL_on`. These are on/off indicators; each is stored offset by 1 and kept in the window $[1, 2]$.
- **Covariates (21).** `base_power`, `current_pf`, `power_pf`, `impedance_pf`, `duty_cycle`, `period`, `phase`, `heat_fraction`, `multiplier`, `voltage_factor`, `breaker_amps`, `power_factor`, `number_of_devices`, `thermostatic_control_range`, `rate_of_cooling`, `rate_of_heating`, `temperature`, `phi`, `demand_rate`, `nominal_power`, `recovery_duty_cycle`.

Each target has one *primary* covariate that drives it most directly; the pairing is given in Table 7. Eleven of the covariates act as a primary driver; the remaining ten only act as *secondary* drivers when an event sets several covariates at once (described below).

Covariate shapes. The covariates `temperature`, `base_power`, `multiplier`, `voltage_factor`, `duty_cycle`, and `recovery_duty_cycle` are assigned the bell shape (a setting is ramped up for a period and then brought back down). The remaining covariates are assigned the step shape (a setting is moved to a level and held there for the duration of the event). Each event draws its multiplicative factor for a covariate from the pool $\{0.1, 0.25, 0.5, 2, 4, 6, 8\}$.

Primary relationships. Each primary relationship maps the baseline value y and the primary covariate’s factor f to the target under the event. The relationships below are derived from GridLAB-D’s documented residential and power-flow model (Chassin et al., 2008);⁶ we restate them in closed form and give a short description of the behavior each one captures. For brevity we write $g(f) = \text{sign}(f - 1) |f - 1|^{0.4}$ for the damped shift used by several relationships.

- **actual_power, total_load, fan_power (from base_power, nominal_power, power_pf).** Each power draw scales close to linearly with its driving quantity, with a small fixed overhead.

⁶<https://www.gridlabd.org/>

All three take the same form, e.g.:

$$\text{actual_power} : y' = f y + 0.05 (f - 1).$$

- **hvac_load, hvac_power (from voltage_factor, duty_cycle).** Raising the driver lowers the HVAC draw by a damped offset:

$$\begin{aligned} \text{hvac_load} : y' &= y - 0.4 g(f) - 0.02 (f - 1), \\ \text{hvac_power} : y' &= y - 0.35 g(f) - 0.02 (f - 1). \end{aligned}$$

- **heating_demand (from rate_of_heating).** Heating demand rises super-linearly when the heating rate goes up and falls sub-linearly when it goes down, so the relationship is asymmetric in f :

$$y' = \begin{cases} f^{1.5} y, & f \geq 1, \\ f^{0.7} y, & f < 1. \end{cases}$$

- **air_temperature, mass_temperature (from heat_fraction, temperature).** Each temperature shifts by a damped amount plus a small linear term:

$$\begin{aligned} \text{air_temperature} : y' &= y + 2.0 g(f) + 0.05 (f - 1), \\ \text{mass_temperature} : y' &= y + 1.5 g(f) + 0.04 (f - 1). \end{aligned}$$

- **is_COOL_on, is_HEAT_on, is_AUX_on (from thermostatic_control_range, recovery_duty_cycle, period).** Each on/off indicator shifts within its clipped window $[1, 2]$:

$$\begin{aligned} \text{is_COOL_on} : y' &= \text{clip}(y + 0.25 g(f) + 0.01 (f - 1), 1, 2), \\ \text{is_HEAT_on} : y' &= \text{clip}(y - 0.25 g(f) - 0.01 (f - 1), 1, 2), \\ \text{is_AUX_on} : y' &= \text{clip}(y - 0.20 g(f) - 0.01 (f - 1), 1, 2). \end{aligned}$$

Combining multiple covariates. When an event sets more than one covariate, the target starts from its primary relationship and each additional active covariate applies a further, dampened effect. A target's class fixes whether these effects combine multiplicatively (power, demand) or additively (temperature, state). The covariates also split into a *thermal* group, which acts on the building's thermal state, and a *non-thermal* group, which acts on the electrical load; the two groups get different weights and effect shapes, listed in Table 8. Let n be the number of active secondary covariates in the event and set the joint dampening

$$s = \begin{cases} 1, & n \leq 1, \\ \frac{1}{\sqrt{n}}, & n > 1, \end{cases}$$

so that the more covariates move together, the less each one contributes. Each secondary covariate c also carries a fixed weight w_c that orders it by importance. Writing r for a multiplicative effect and δ for an additive one, the running target value is updated as

$$\begin{aligned} \text{power, demand} : y &\leftarrow y [1 + (s w_c) (r - 1)], \\ \text{temperature} : y &\leftarrow y + (s \cdot 0.5) \delta, \\ \text{state} : y &\leftarrow y + (s w_c) \delta. \end{aligned}$$

The weights w_c and the effects r and δ are listed in Table 8. After all effects are applied, the state targets are clipped back to their $[1, 2]$ window.

The direction and qualitative form of every effect follow the mechanisms above. The magnitudes, i.e. the weights w_c and the constants in r and δ , are chosen so that each covariate leaves a clear and separable imprint on the target, rather than being derived from the simulator.

Target class	Secondary covariate group	Weight w_c	Effect (r or δ)
Power (mult.)	voltage_factor	0.50	$r = f^2$
	breaker_amps	0.40	$r = 1 + 0.1(f - 1)$
	phi	0.30	$r = \cos^2(\text{clip}(1.57(f - 1), \pm 1.57))$
	other non-thermal	0.20	$r = f + 0.05(f - 1)$
	thermal	0.30	$r = 1 + 0.15g(f)$
Demand (mult.)	thermal	0.40	$r = 1 + 0.15g(f)$
	non-thermal	0.20	$r = 1 + 0.05g(f)$
Temperature (add.)	heat_fraction	1.50	$\delta = w_c g(f) + 0.02(f - 1)$
	temperature	2.50	$\delta = w_c g(f) + 0.02(f - 1)$
	thermostatic_control_range	0.50	$\delta = w_c g(f) + 0.02(f - 1)$
	other thermal	0.40	$\delta = w_c g(f) + 0.02(f - 1)$
	non-thermal	0.05	$\delta = w_c g(f) + 0.005(f - 1)$
State (add.)	thermal	0.40	$\delta = -0.1g(f)$
	non-thermal	0.10	$\delta = 0.02g(f)$

Table 8 | **GridLAB-D secondary covariate effects.** For a target, each secondary covariate contributes an effect scaled by its weight w_c and the joint dampening s . Multiplicative classes use the raw effect r ; additive classes use the offset δ . Here f is the secondary covariate’s own factor and $g(f) = \text{sign}(f - 1)|f - 1|^{0.4}$. Covariates split into a *thermal* group (heat_fraction, temperature, thermostatic_control_range, rate_of_heating, rate_of_cooling, duty_cycle, recovery_duty_cycle, period), which act on the building’s thermal state, and a *non-thermal* group of all remaining covariates, which act on the electrical load.

B.6.3. EPANET

Base setting. We run EPANET 2.2 through WNTR on the stock Net1 network (EPANET Example 1). The network has 11 nodes (9 junctions, one reservoir with id 9, and one tank with id 2) and 13 links (12 pipes and one pump with id 9), solved with the Hazen–Williams head-loss model in GPM units. A control opens the pump when the tank level drops below a lower bound and closes it above an upper bound, so the tank cycles over the run. Outputs are recorded at a fixed interval, which forms the baseline series.

Targets and covariates. We take 37 target series and use the covariates below, named as they appear in the released benchmark:

- **Pressure targets** (10). pressure_{10, 11, 12, 13, 21, 22, 23, 31, 32} at the nine junctions, and pressure_2 at the tank.
- **Demand targets** (2). demand_9 (reservoir outflow) and demand_2 (tank flow).
- **Flowrate targets** (13). flowrate_p for the twelve pipes $p \in \{10, 11, 12, 21, 22, 31, 110, 111, 112, 113, 121, 122\}$ and flowrate_9 at the pump.
- **Velocity targets** (12). velocity_p for the same twelve pipes.
- **Covariates.** junction_j_base_demand for the nine junctions, reservoir_9_base_head, tank_2_init_level, pump_9_setting, and pipe_p_diameter and pipe_p_roughness for the twelve pipes.

Each target has one *primary* covariate that drives it most directly; the pairing is given in Table 9. Any covariate that an event also sets acts as a *secondary* driver (described below). Every event only *raises* its covariates (the factor $f > 1$), so targets only increase under an event.

Target	Primary covariate	a	c	Combination
pressure_j (junctions)	junction_j_base_demand	0.09	$1 + 0.05 (j \bmod 10)$	multiplicative
pressure_2 (tank)	tank_2_init_level	0.09	1	multiplicative
demand_9 (reservoir)	reservoir_9_base_head	0.09	1	multiplicative
demand_2 (tank)	tank_2_init_level	0.09	1	multiplicative
flowrate_p (pipes)	pipe_p_diameter	0.09	$1 + 0.1 (p \bmod 10)$	multiplicative
flowrate_9 (pump)	pump_9_setting	0.09	1	multiplicative
velocity_p (pipes)	pipe_p_roughness	0.10	$1 + 0.02 (p \bmod 10)$	multiplicative

Table 9 | **EPANET target-covariate pairing.** Each target is driven by one primary covariate through the shared form in the text, with per-element constants a and c . All targets combine multiplicatively. Here j ranges over the nine junction ids and p over the twelve pipe ids.

Target class	Weight w_c	Ratio r
Pressure	$0.10 + 0.05 (h \bmod 5)$	$r = 1 + (s w_c) (f^{0.4} - 1) + 0.01 (s w_c) (f - 1)$
Demand	$0.10 + 0.02 (h \bmod 5)$	
Flowrate	$0.10 + 0.04 (h \bmod 6)$	
Velocity	$0.10 + 0.03 (h \bmod 6)$	

Table 10 | **EPANET secondary covariate effects.** Each secondary covariate contributes a multiplicative ratio r scaled by its weight w_c and the joint dampening s . All classes share the same ratio shape and differ only in the weight. Here f is the secondary covariate’s own factor, and h is a fixed per-covariate constant (the sum of the character codes of its name) that spreads the weights across covariates.

Covariate shapes. The covariates pump_9_setting, reservoir_9_base_head, and tank_2_init_level are assigned the bell shape (a setting is ramped up for a period and then brought back down). The remaining covariates are assigned the step shape (a setting is moved to a level and held there for the duration of the event). Each event draws its multiplicative factor for a covariate from the pool $\{2, 4, 6, 8\}$; all factors exceed 1, so the targets only increase under an event.

Primary relationships. Each primary relationship maps the baseline value y and the primary covariate’s factor f to the target under the event. The relationships are derived from EPANET’s documented hydraulic model (Rossman, 2000);⁷ we restate them in closed form. Every target follows the same multiplicative form, with a small per-element sensitivity:

$$y' = y \left[1 + a c g(f) + 0.01 c (f - 1) \right], \quad g(f) = \text{sign}(f - 1) |f - 1|^{0.4},$$

where the constants a and c for each target are listed in Table 9. The target rises with its driver by a damped amount; the constant c scales how sensitive each junction or pipe is, and because events only raise covariates ($f > 1$) the target always increases.

Combining multiple covariates. When an event sets more than one covariate, the target starts from its primary relationship and each additional active covariate applies a further, dampened effect. For EPANET every class combines multiplicatively. Let n be the number of active secondary covariates in the event and set the joint dampening

$$s = \begin{cases} 1, & n \leq 1, \\ \frac{1}{\sqrt{n}}, & n > 1, \end{cases}$$

⁷<https://www.epa.gov/water-research/epanet>

so that the more covariates move together, the less each one contributes. Each secondary covariate c also carries a fixed weight w_c that orders it by importance. For a secondary covariate whose ratio is r , the running target value is updated as

$$y \leftarrow y \cdot r, \quad r = 1 + (s w_c) (f^{0.4} - 1) + 0.01 (s w_c) (f - 1).$$

The weights w_c are listed in Table 10.

The direction and qualitative form of every effect follow the mechanisms above. The magnitudes, i.e. the weights w_c and the constants in the primary and secondary relationships, are chosen so that each covariate leaves a clear and separable imprint on the target, rather than being derived from the simulator.

B.6.4. SUMO

Base setting. We run SUMO on a single signalized four-arm intersection: a center node with four approach nodes at ± 500 m, every edge two lanes at 13.89 m/s (50 km/h) in both directions, and a fixed traffic-light plan on a 90 s cycle (42 s green, 3 s yellow, twice). Traffic follows a rush-hour demand of four through-movements, with five periods per day (from 120 vehicles/hour at night up to 2000 vehicles/hour at peak) repeated over 30 days. Induction-loop and lane-area detectors, plus an edge-level emissions record, are written every 300 s, which forms the baseline series.

Targets and covariates. We generate tasks for 30 target series and use 16 covariates. One target fails the 10% verification criterion in every task and is dropped entirely, leaving the 29 targets below, named as they appear in the released benchmark. Detectors on the two measured edges carry the suffixes `_e_0` and `_e_1`; we write $\{0, 1\}$ for the pair.

- **Emissions targets (4).** `CO2_abs_edge_e2c`, `NOx_abs_edge_e2c`, `PMx_abs_edge_e2c`, `fuel_abs_edge_e2c`.
- **Flow targets (5).** `flow_loop_e_{0,1}_near`, `flow_loop_e_0_far`, `flow_loop_exit_e_{0,1}`.
- **Occupancy targets (6).** `occupancy_loop_e_{0,1}_near`, `occupancy_loop_exit_e_{0,1}`, `meanOccupancy_e2_e_{0,1}`.
- **Speed targets (6).** `speed_loop_e_{0,1}_near`, `speed_loop_exit_e_{0,1}`, `meanSpeed_e2_e_{0,1}`.
- **Delay targets (6).** `meanHaltingDuration_e2_e_{0,1}`, `meanTimeLoss_e2_e_{0,1}`, `meanVehicleNumber_e2_e_{0,1}`.
- **Jam-length targets (2).** `meanMaxJamLengthInMeters_e2_e_{0,1}`.
- **Covariates (16).** `minGap`, `accel`, `decel`, `maxSpeed`, `sigma`, `tau`, `length`, `speedDev`, `offset`, `width`, `height`, `lcStrategic`, `lcCooperative`, `lcSpeedGain`, `impatience`, `actionStepLength`.

Each target has one *primary* covariate; the pairing is given in Table 11. Fifteen covariates act as a primary driver; `speedDev` has no target in the released set and only acts as a *secondary* driver (described below), as do the other covariates when an event sets several at once. Before the relationships are applied, every target series is shifted up by a constant so all values stay positive.

Covariate shapes. The covariates `sigma`, `impatience`, and `lcSpeedGain` are assigned the bell shape (a setting is ramped up for a period and then brought back down). The remaining covariates are assigned the step shape (a setting is moved to a level and held there for the duration of the event). Each event draws its multiplicative factor for a covariate from the pool $\{0.1, 0.25, 0.5, 2, 4, 6, 8\}$.

Target	Primary covariate	Combination
meanMaxJamLengthInMeters_e2_e_{0,1}	minGap	additive
meanTimeLoss_e2_e_{0,1}	accel	additive
meanHaltingDuration_e2_e_{0,1}	offset	additive
meanVehicleNumber_e2_e_{0,1}	impatience	additive
speed_loop_e_{0,1}_near	decel	multiplicative
meanSpeed_e2_e_{0,1}	maxSpeed	multiplicative
speed_loop_exit_e_{0,1}	lcSpeedGain	multiplicative
CO2/NOx/PMx/fuel_abs_edge_e2c	sigma	multiplicative
flow_loop_e_{0,1}_near	tau	multiplicative
flow_loop_e_0_far	lcStrategic	multiplicative
flow_loop_exit_e_{0,1}	lcCooperative	multiplicative
meanOccupancy_e2_e_{0,1}	length	multiplicative
occupancy_loop_e_0_near	width	multiplicative
occupancy_loop_e_1_near	height	multiplicative
occupancy_loop_exit_e_{0,1}	actionStepLength	multiplicative

Table 11 | **SUMO target-covariate pairing.** Each target is driven by one primary covariate. The Combination column is the rule by which *secondary* covariates combine, set by the target’s class (additive for delay and jam length, multiplicative for emissions, flow, occupancy, and speed).

Primary relationships. Each primary relationship maps the baseline value y and the primary covariate’s factor f to the target under the event. The relationships below are derived from SUMO’s documented car-following, emission, and detector models (Lopez et al., 2018);⁸ we restate them in closed form. A covariate can be raised or lowered, but each metric here responds mainly to added pressure on the network, so we first map the factor through an *effective factor*

$$f_{\text{eff}} = \begin{cases} f, & f \geq 1, \\ 1 + 1.5 (1/f - 1), & f < 1, \end{cases}$$

which turns a reduction into an equivalent increase. Every relationship below uses f_{eff} .

- **Delay (from accel, offset, impatience).** Delays grow exponentially with the driver:

$$\begin{aligned} \text{meanTimeLoss} : y' &= y + 6.0 (e^{1.1 (f_{\text{eff}} - 1)} - 1), \\ \text{meanHaltingDuration} : y' &= y + 8.0 (e^{1.2 (f_{\text{eff}} - 1)} - 1), \\ \text{meanVehicleNumber} : y' &= y + 4.0 (e^{1.15 (f_{\text{eff}} - 1)} - 1). \end{aligned}$$

- **Jam length (from minGap).** Queue spillback grows faster than linearly:

$$\text{meanMaxJamLengthInMeters} : y' = y + 4.5 (f_{\text{eff}} - 1)^{2.5}.$$

- **Speed (from decel, maxSpeed, lcSpeedGain).** More pressure lowers speed through a decay:

$$\begin{aligned} \text{speed_loop_near} : y' &= y / f_{\text{eff}}^{0.4}, \\ \text{meanSpeed} : y' &= y / f_{\text{eff}}^{0.35}, \\ \text{speed_loop_exit} : y' &= y / f_{\text{eff}}^{0.3}. \end{aligned}$$

- **Emissions (from sigma).** All four pollutants scale exponentially:

$$\text{CO2/NOx/PMx/fuel} : y' = y e^{0.12 (f_{\text{eff}} - 1)}.$$

⁸<https://sumo.dlr.de/docs/>

	minGap	accel	decel	maxSpeed	sigma	tau	length	speedDev	offset	width	height	lcStrategic	lcCooperative	lcSpeedGain	impatience	actionStepLength
w_c^{add}	.60	.20	.20	.20	.20	.20	.50	.20	.70	.20	.20	.20	.20	.20	.45	.20
w_c^{mul}	.45	.15	.30	.15	.35	.50	.35	.25	.30	.15	.15	.20	.20	.20	.15	.15

Table 12 | **SUMO secondary covariate weights.** w_c^{add} is used for the additive classes (delay, jam length) and w_c^{mul} for the multiplicative classes (emissions, flow, occupancy, speed). The effect shapes e and ρ are given in the text.

- **Flow (from tau, lcStrategic, lcCooperative).** Flow follows a piecewise curve, with a rising branch below $f_{\text{eff}} = 2$ and a falling branch above it as stop-and-go sets in. With our factor pool and the effective-factor map, every event has $f_{\text{eff}} \geq 2$ (the smallest is $f = 2$ itself, and reductions map to 2.5 and above), so only the falling branch is reached and flow always decreases under an event. The curve is discontinuous at $f_{\text{eff}} = 2$, which affects no generated task since no event lands on the rising side:

$$y' = \begin{cases} y (1 + 0.1 (f_{\text{eff}} - 1)), & f_{\text{eff}} < 2, \\ y \cdot 2.2 / f_{\text{eff}}^{1.5}, & f_{\text{eff}} \geq 2. \end{cases}$$

- **Occupancy (from length, width, height, actionStepLength).** Occupancy rises faster than linearly:

$$\begin{aligned} \text{meanOccupancy} : y' &= y + 0.15 (f_{\text{eff}} - 1)^{2.2}, \\ \text{occupancy_loop_e_0_near} : y' &= y (1 + 0.05 (f_{\text{eff}} - 1)^{1.5}), \\ \text{occupancy_loop_e_1_near} : y' &= y (1 + 0.04 (f_{\text{eff}} - 1)^{1.5}), \\ \text{occupancy_loop_exit} : y' &= y (1 + 0.02 (f_{\text{eff}} - 1)^{1.9}). \end{aligned}$$

Combining multiple covariates. When an event sets more than one covariate, the target starts from its primary relationship and each additional active covariate applies a further, dampened effect. A target’s class fixes whether these effects combine additively (delay, jam length) or multiplicatively (emissions, flow, occupancy, speed). Let n be the number of active secondary covariates in the event and set the joint dampening

$$s = \begin{cases} 1, & n \leq 1, \\ \frac{1}{\sqrt{n}}, & n > 1. \end{cases}$$

Each secondary covariate c carries a weight w_c (Table 12) and the running target value is updated as

$$\text{additive: } y \leftarrow y + (s w_c^{\text{add}}) e(f_{\text{eff}}), \quad \text{multiplicative: } y \leftarrow y [1 + (s w_c^{\text{mul}}) (\rho(f_{\text{eff}}) - 1)].$$

The additive shift $e(\cdot)$ and multiplicative ratio $\rho(\cdot)$ reuse the custom curves above for a few sensitive covariates, and take a generic form otherwise:

- $e(f_{\text{eff}}) = 8.0 (e^{1.2 (f_{\text{eff}} - 1)} - 1)$ for **offset**, $4.5 (f_{\text{eff}} - 1)^{2.5}$ for **minGap**, $4.0 (e^{1.15 (f_{\text{eff}} - 1)} - 1)$ for **impatience**, $10.0 (f_{\text{eff}} - 1)^2$ for **length**, and $8.0 (f_{\text{eff}} - 1)^2$ otherwise.

- $\rho(f_{\text{eff}}) = f_{\text{eff}}^{-0.4}$ for `decel`, $f_{\text{eff}}^{-0.35}$ for `maxSpeed`, $e^{0.12(f_{\text{eff}}-1)}$ for `sigma`, the flow piecewise form above for `tau`, and $1 + \frac{m}{2}(f_{\text{eff}} - 1)^{1.5}$ otherwise, with $m = 0.28$ for `actionStepLength` and $m = 0.25$ for the rest.

The direction and qualitative form of every effect follow the mechanisms above. The magnitudes, i.e. the weights w_c and the constants in e and ρ , are chosen so that each covariate leaves a clear and separable imprint on the target, rather than being derived from the simulator.

C. Aggregation and Metrics

We provide here the details of how each task is scored and how the per-task scores are aggregated in the benchmark.

C.1. Per-task scoring

Each metric reduces a task’s forecast to a single number. A point-forecast metric takes the median (point) forecast $\widehat{\mathbf{X}}_F = [\widehat{X}_{t+1}, \dots, \widehat{X}_T]$ and the ground-truth future $\mathbf{X}_F = [X_{t+1}, \dots, X_T]$; a probabilistic metric takes the predicted quantiles $\widehat{\mathbf{X}}_F^{(q)}$ over a set of levels Q . The forecast horizon runs from $t + 1$ to T , so it has $T - t$ steps.

C.2. Aggregates

Following [Shchur et al. \(2025\)](#), we report two aggregates over a task set $\mathcal{S}$, defined in Section 3.5 and restated here. Let $\mathcal{M}$ be the set of evaluated models, $M = |\mathcal{M}|$, and $m_j(\mathcal{T})$ the error (wQL or WAPE) of model j on task $\mathcal{T}$.

Skill score. We use the skill score ([Hyndman and Athanasopoulos, 2018](#)), a model’s relative reduction in error against a fixed reference model β , the naive baseline that repeats the last observed value. For model j over $\mathcal{S}$,

$$\text{Skill}_j(\mathcal{S}) = 1 - \exp\left(\frac{1}{|\mathcal{S}|} \sum_{\mathcal{T} \in \mathcal{S}} \log \text{clip}\left(\frac{m_j(\mathcal{T})}{m_\beta(\mathcal{T})}\right)\right), \quad \text{clip}(x) = \max(10^{-2}, \min(x, 10^2)).$$

The clip constrains each ratio to $[10^{-2}, 10^2]$, limiting the influence of a few extreme tasks. The score is a geometric mean of the per-task error ratios: it equals 0 when a model matches the baseline, is positive when it improves on the baseline, and approaches 1 (or 100%) as forecasts approach perfect accuracy.

Average win rate. The average win rate is the fraction of tasks on which a model attains a lower error than each competing model, with ties counted as half a win. For model j over $\mathcal{S}$,

$$\text{Win}_j(\mathcal{S}) = \frac{1}{|\mathcal{S}|(M-1)} \sum_{\mathcal{T} \in \mathcal{S}} \sum_{\substack{k \in \mathcal{M} \\ k \neq j}} \left(\mathbb{1}[m_j(\mathcal{T}) < m_k(\mathcal{T})] + \frac{1}{2} \mathbb{1}[m_j(\mathcal{T}) = m_k(\mathcal{T})] \right).$$

The win rate ranges from 0 to 1 (or 100%). It is magnitude-invariant, reflecting only the ordering of models on each task and not the size of their differences, and its value depends on the composition of the model set $\mathcal{M}$, changing as models are added or removed.

C.3. Metrics

The main paper reports the weighted quantile loss (wQL) and the weighted absolute percentage error (WAPE), defined in Section 3.5. We restate them here. For a quantile level q , let $\Lambda_q(X, \hat{X}) = q(X - \hat{X})_+ + (1 - q)(\hat{X} - X)_+$ be the quantile (pinball) loss, with $(\cdot)_+ = \max(\cdot, 0)$. wQL averages this over the levels Q and normalizes by the target’s aggregate magnitude; WAPE is the point-forecast analogue on the median forecast:

$$\text{wQL}(\mathcal{T}) = \frac{1}{|Q|} \sum_{q \in Q} \frac{\sum_{\tau=t+1}^T \Lambda_q(X_\tau, \hat{X}_\tau^{(q)})}{\sum_{\tau=t+1}^T |X_\tau|}, \quad \text{WAPE}(\mathcal{T}) = \frac{\sum_{\tau=t+1}^T |X_\tau - \hat{X}_\tau|}{\sum_{\tau=t+1}^T |X_\tau|}.$$

We adopt these metrics for their scale-invariance: our targets span four simulators with disparate scales, and normalizing by each target’s aggregate magnitude keeps errors comparable across targets and well suited to the relative aggregation above.

D. Experimental Setup

Every model is given the target’s history together with the covariates: the past covariates over the history and the future covariates over the forecast horizon. The foundation models produce nine quantile levels, 0.1, 0.2, $\dots$, 0.9, which we score with the metrics of Appendix C. The LLMs output a point forecast, which we score as the median: their wQL is the quantile loss at the 0.5 level only (Section 3.6). We prompt the LLMs for the median rather than many quantiles because asking for many quantiles lengthens the output and causes formatting errors that persist across retries. Scoring the median alone inherently disadvantages the LLMs under probabilistic metrics, since the foundation models’ wQL averages over nine levels, 0.1 to 0.9, most of them away from the median, where a calibrated forecast incurs a lower loss; we leave more efficient ways to elicit quantiles from LLMs to future work.

We run the LLMs through Google Cloud Vertex Model Garden and each foundation model on a single A100 GPU per task. The inference times in Table 3 thus compare on-device GPU times with API round-trip times. These differ, but they are the conditions under which each family is used in practice, and represent the best each can achieve.

D.1. Foundation models

We use the following public checkpoints.

- Chronos-2: [amazon/chronos-2](https://huggingface.co/amazon/chronos-2)⁹
- TimesFM-3.0: [google/timesfm-3.0-pytorch](https://huggingface.co/google/timesfm-3.0-pytorch)¹⁰
- t0-gamma: [theforecastingcompany/t0-gamma](https://huggingface.co/theforecastingcompany/t0-gamma)¹¹
- Timestral: [mistralai/Timestral-209M](https://huggingface.co/mistralai/Timestral-209M)¹²
- TiRex-2: [NX-AI/TiRex-2](https://huggingface.co/NX-AI/TiRex-2)¹³
- TS-ICL: [taharnbl/TS-ICL](https://huggingface.co/taharnbl/TS-ICL)¹⁴
- TabPFN-3.5: [Prior-Labs/tabpfm_3_5](https://huggingface.co/Prior-Labs/tabpfm_3_5)¹⁵, which we run for time series through the TabPFN-TS harness (Hoo et al., 2025).

⁹<https://huggingface.co/amazon/chronos-2>

¹⁰<https://huggingface.co/google/timesfm-3.0-pytorch>

¹¹<https://huggingface.co/theforecastingcompany/t0-gamma>

¹²<https://huggingface.co/mistralai/Timestral-209M>

¹³<https://huggingface.co/NX-AI/TiRex-2>

¹⁴<https://huggingface.co/taharnbl/TS-ICL>

¹⁵https://huggingface.co/Prior-Labs/tabpfm_3_5

D.2. Large language models

We evaluate two frontier LLMs: Gemini-3.1-Pro (gemini-3.1-pro-preview) and Claude-Opus-4.8 (claude-opus-4-8). We run both at medium and high reasoning effort.

We pass the target history and the covariates to the model as text, as four CSV files, and ask it to return the forecast in a fixed tagged format. The variable names in these files are replaced by dummy identifiers such as <target_1> and <covariate_1>. The exact prompt is:

```
# Task Definition

## File Descriptions
- covariates.csv: Time-series data of input variables (features) that
  might influence the system or be affected by events.
- variates.csv: Time-series data of output variables (targets) that
  you need to forecast.
- future_variates_blanked.csv: Time-series data of output variables
  (targets) that you need to forecast.
- future_covariates.csv: Time-series data of input variables (features)
  for the future timestamps - the same timestamps you need to forecast
  the output variables for.

## Goal
You must provide the forecast values in a strict structured format. Do
not include any conversational text, analysis, or explanations in your
final response. Only output the forecast data within the specified tags.
For each variable listed in future_variates_blanked.csv, output the
forecast values enclosed in tags named after the variable. The content
inside the tags must be a list of (timestamp, value) pairs separated by
commas. You must output the variables in the exact same order as the
columns appear in the header of future_variates_blanked.csv (ignoring
the timestamp column).

Example Format:
<variable_name>
(timestamp, value), (timestamp, value), ...
</variable_name>

Replace variable_name with the actual name of the variable you are
forecasting. Ensure you provide values for all expected timestamps
found in future_variates_blanked.csv.
```

If a call fails or the output cannot be parsed into a forecast for every target and timestamp, we retry the same call, up to 10 attempts per task, with a short wait that grows with each attempt. A task with no valid forecast after all attempts is imputed with the naive baseline forecast. We find that this happens on at most 16 of 37,665 tasks for any LLM, under 0.05% in every case, so it does not affect the reported aggregates. We report the number of failures in Table 13.

Model	Imputed windows	% of windows
Claude-Opus-4.8 (High)	16	0.04%
Claude-Opus-4.8 (Medium)	2	0.01%
Gemini-3.1-Pro (High)	10	0.03%
Gemini-3.1-Pro (Medium)	0	0.00%

Table 13 | Number of task windows on which each LLM returned no valid forecast after all retries, out of 37,665 windows. These windows are imputed with the naive baseline forecast, so they receive zero skill. All other models return a forecast on every window and have no imputed windows.

D.3. Regression baselines

We fit each regression baseline separately on the history of every task. Each training row is one time step of the history: the input is the covariates at that step, $\mathbf{z}_\tau$, and the label is the target X_τ . At test time, the model predicts each forecast step X_{t+h} separately, given its known covariates $\mathbf{z}_{t+h}$. We use Linear Regression and Ridge Regression from scikit-learn, with the covariates standardized on the history and a regularization strength of 10 for Ridge Regression, and XGBoost with 150 trees of depth 6, a learning rate of 0.05, and row and column subsampling of 0.9. These settings are fixed across all tasks and are not tuned. Like the LLMs, the regression baselines output a point forecast, which we score as the median. We run them on CPU. Tuned variants of these models are explored in Appendix J.

E. Limitations and Future Work

We discuss the limitations of WHATIF-BENCH and several areas for future work.

Harder system dynamics. Our tasks keep the system in a single regime: each event sets the covariates, the target responds, and the system returns to its usual range. Within this range, each relationship is exact and does not reverse. Many real-world systems may not always be so well-behaved. An intervention can push a system into a new state, such as a power grid moving from normal operation into an overload. Modelling such regime changes is important, and common in domains such as finance. An intervention can also drive a system into a hard limit, where the target saturates and stops responding. Capturing these limits is important, so a model learns when a covariate stops affecting the target. Further, interventions can also stack on one another, so the system never settles back before the next one begins (Bica et al., 2020; Robins et al., 2000). Building benchmarks that cover these cases would test whether a model can reason about the behavior of the system as a whole.

Larger covariate spaces. Our tasks use a refined set of covariates, where every one of them affects the target, however, real systems can expose a vast number of possible drivers, of which only a few matter, and sometimes none do. Further, many drivers could be unrelated to the target, while some could be correlational. A model must then find the drivers that truly matter, and ignore the rest (Assaad et al., 2022; Runge et al., 2019). Metadata and other side information could help a model identify the true drivers. Building benchmarks with such irrelevant and misleading covariates is an important future direction.

Grounding in real-world data. WHATIF-BENCH is designed to discover failure modes in a controlled semi-synthetic environment, where the covariate–target relationships are known exactly and the ground truth carries no noise unrelated to the covariates, so a model’s error reflects how it handles

the covariates. We do not test how these findings hold up under the irreducible noise and unobserved drivers of real systems. Validating our proposed axes on real-world data with covariates would be a useful next step and may reveal further findings. Building benchmarks for what-if forecasting from purely real-world data is a grand challenge. Real data records only one course of events, where it is impossible to simulate counterfactuals (Holland, 1986; Imbens and Rubin, 2015). Synthetic control offers a partial answer: it estimates the missing counterfactual for a unit from a weighted combination of comparable units whose outcomes were recorded (Abadie et al., 2010). Bringing these ideas to forecasting would move benchmarks closer to real applications, while keeping the ground truth needed to measure a model’s response.

Natural-language what-if queries. Our tasks still require engineering the right covariates: we assume the right drivers have been chosen already, and are set by hand to simulate scenarios. A more general system would instead take a what-if query as posed, in natural language, and construct the covariates itself. Benchmarks that evaluate this end to end, from a free-form query to a forecast are promising directions for future work (Lee et al., 2025; Liu et al., 2024a; Williams et al., 2025). Such queries carry their own nuances: they may be precise, broad, or ambiguous, and may name the drivers only indirectly. LLMs and agents are natural candidates to interpret them, turning a query into concrete covariates and opening scenario analysis to many more users, while keeping those covariates explicit for inspection. A capable system should also know when a query cannot be answered from the available history, and say so, instead of returning a confident but unsupported forecast. Benchmarks that capture these needs are important for bringing such systems into real-world use and deployment.

From evaluation to training. Most broadly, the simulators behind WHATIF-BENCH need not serve only for evaluation; they can also be used for training. A model that learns a system in-context and forecasts its response to unseen interventions would do more than predict: it could plan, weigh possible actions, and help make decisions about the system it has learned, akin to *world models* (Ha and Schmidhuber, 2018; Hafner et al., 2023; LeCun, 2022). We see WHATIF-BENCH as a first step toward forecasters that can answer such questions.

F. Skill across the number of events, event length and context length

The main text reports how skill varies with the number of historical events. Here we show the analogous trends for event length and context length, again split by Replay and Novel futures. Both echo the number-of-events result: on Novel futures, skill improves as the context holds more past events: that is, when events are shorter (Figure 6) or the context is longer (Figure 7). We also report how the regression baselines and TiRex-2 respond to the number of events (Tables 14 and 15).

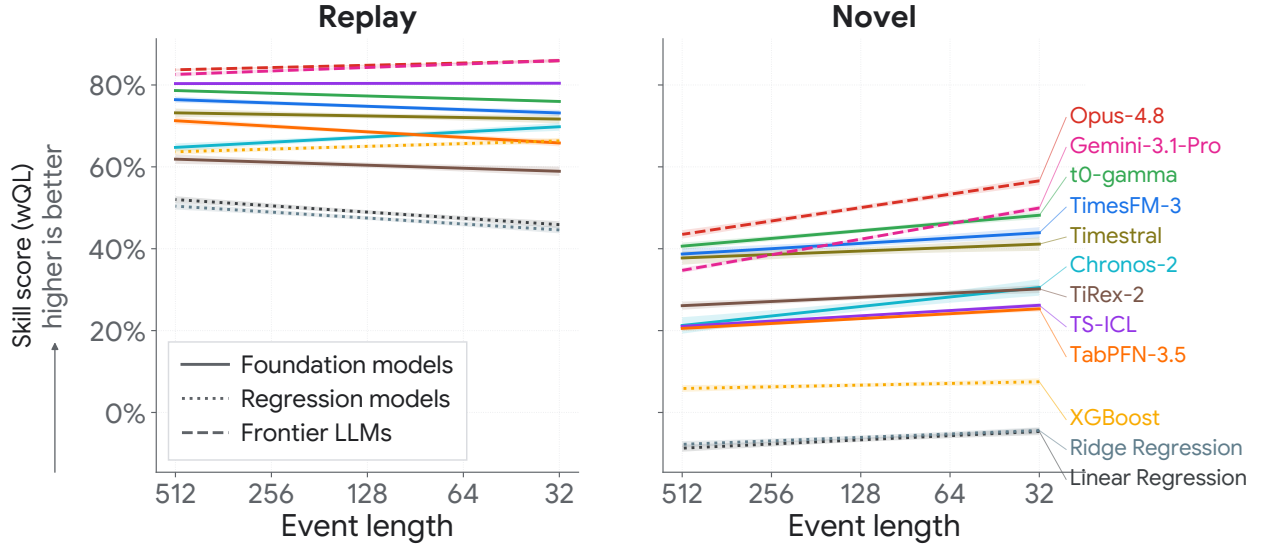


Figure 6 | Skill scores with different event lengths on WHATIF-BENCH, on tasks with Replay (left) and Novel (right) futures respectively. Lines are least-squares trends of skill on $\log_2$ of the event length; they summarise the direction of change.

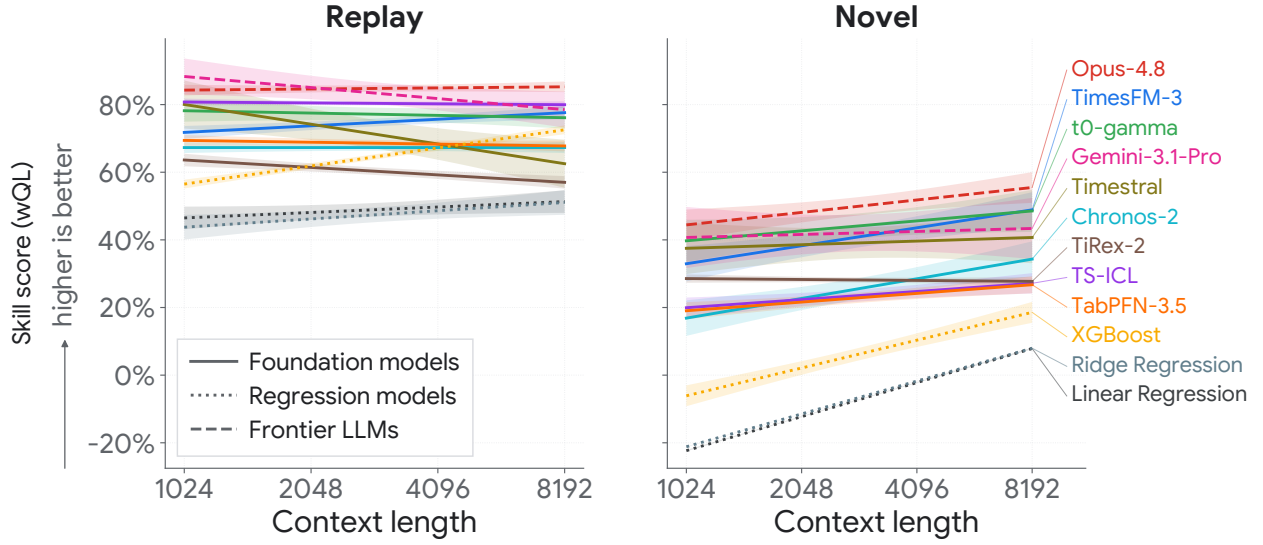


Figure 7 | Skill scores with different context lengths on WHATIF-BENCH, on tasks with Replay (left) and Novel (right) futures respectively. All models receive the same input and produce the same output: each consumes the full history at every context length, without truncation, and forecasts the same horizon. Tasks are grouped by their nominal context length; for the longest events, the actual context is longer (Appendix B.3). Lines are least-squares trends of skill on $\log_2$ of the context length; they summarise the direction of change.

Linear and Ridge Regression and the number of events. Most models change little on Replay futures as the number of events grows, but Linear and Ridge Regression get worse on two simulators. Table 14 compares tasks that differ only in the number of events, sharing the target, event length, context length and number of covariates. On SUMO, their Replay skill falls from about 80% to 5% and their Novel skill from -4% to -30% ; on OpenDC, Replay skill falls from 80% to 54% and Novel

skill from 29% to −35%. On EPANET and GridLAB-D, both improve with more events. We believe this is because, with only a few events, a single linear fit can pass close to the few covariate values it has seen, while with many events it has to average over many points and loses the fit, even at values it has already seen; this would explain why skill drops on both Replay and Novel futures.

Simulator	Replay				Novel			
	Fewest	Most	Change	Worse	Fewest	Most	Change	Worse
OpenDC	80.2%	54.4%	−25.8	22/22	28.7%	−35.2%	−63.9	11/22
GridLAB-D	34.8%	39.8%	+5.0	4/10	6.2%	18.2%	+12.0	4/10
EPANET	24.4%	39.4%	+15.1	6/33 [†]	10.9%	32.6%	+21.7	10/33 [†]
SUMO	79.8%	5.5%	−74.3	25/29	−4.2%	−29.7%	−25.5	22/29

Table 14 | Skill score (wQL) of Linear Regression with the fewest and the most historical events, on Replay and Novel futures. Each comparison uses tasks that share the target, event length, context length and number of covariates, where the most events are at least 8× the fewest. Change is in percentage points. Worse is the number of targets whose skill is lower with the most events, out of the benchmark targets (Table 2) that have at least one such comparison. [†]Out of the 37 EPANET targets: the other 4 (demand_9, flowrate_10, flowrate_9, velocity_10) have at most 2 events in every task, so no such comparison exists for them. Ridge Regression is within 1 point of Linear Regression in every cell.

TiRex-2 and the number of events. TiRex-2 is the only foundation model whose skill falls with more events on both Replay and Novel futures (Figure 5). Comparing tasks that differ only in the number of events (Table 15), its skill falls from 61.4% to 52.2% on Replay futures and from 33.8% to 26.5% on Novel futures, while TimesFM-3 gains on both (+2.0 and +4.6 points). The drop comes from two simulators: on SUMO, Replay skill falls from 78.5% to 55.3%, on all 29 targets, and on OpenDC, Novel skill falls from 23.1% to −24.9%, on 14 of 22 targets. On EPANET and GridLAB-D, it improves slightly. Other foundation models show weaker versions of the same pattern, for example Chronos-2 on SUMO Replay futures (−21.4 points), so these settings become harder with many events for several models, and TiRex-2 is the most affected.

Simulator	Replay				Novel			
	Fewest	Most	Change	Worse	Fewest	Most	Change	Worse
OpenDC	63.3%	58.8%	−4.4	17/22	23.1%	−24.9%	−48.1	14/22
GridLAB-D	53.8%	55.7%	+1.9	4/10	35.1%	37.3%	+2.2	3/10
EPANET	40.9%	43.6%	+2.7	15/33 [†]	43.6%	47.7%	+4.0	13/33 [†]
SUMO	78.5%	55.3%	−23.2	29/29	26.4%	18.5%	−7.9	27/29

Table 15 | Skill score (wQL) of TiRex-2 with the fewest and the most historical events, on Replay and Novel futures. Each comparison uses tasks that share the target, event length, context length and number of covariates, where the most events are at least 8× the fewest. Change is in percentage points. Worse is the number of targets whose skill is lower with the most events, out of the benchmark targets (Table 2) that have at least one such comparison. [†]Out of the 37 EPANET targets: the other 4 (demand_9, flowrate_10, flowrate_9, velocity_10) have at most 2 events in every task, so no such comparison exists for them.

Gemini-3.1-Pro and the context length. Gemini-3.1-Pro’s drop at 64 events in Figure 5 comes from the longest context, where all such tasks lie: its skill falls sharply at C_4 on both Replay and

Novel futures, while more events at a fixed context change it little (Table 16). On Replay futures, Claude-Opus-4.8 and TimesFM-3 change much less with context. We believe Gemini-3.1-Pro handles the longest inputs less well.

Model	Replay		Novel	
	Context $C_3 \rightarrow C_4$	Events fewest $\rightarrow$ most	Context $C_3 \rightarrow C_4$	Events fewest $\rightarrow$ most
Gemini-3.1-Pro (High)	87.2 $\rightarrow$ 73.7 (−13.5)	82.9 $\rightarrow$ 80.2 (−2.7)	52.3 $\rightarrow$ 35.1 (−17.2)	42.8 $\rightarrow$ 49.6 (+6.9)
Gemini-3.1-Pro (Medium)	75.9 $\rightarrow$ 63.1 (−12.9)	73.0 $\rightarrow$ 64.6 (−8.4)	35.9 $\rightarrow$ 24.2 (−11.6)	31.6 $\rightarrow$ 31.1 (−0.5)
Claude-Opus-4.8 (High)	86.6 $\rightarrow$ 83.8 (−2.8)	83.2 $\rightarrow$ 85.5 (+2.3)	57.8 $\rightarrow$ 51.3 (−6.5)	48.8 $\rightarrow$ 60.7 (+11.9)
Claude-Opus-4.8 (Medium)	85.5 $\rightarrow$ 82.2 (−3.3)	81.5 $\rightarrow$ 84.0 (+2.5)	53.7 $\rightarrow$ 44.3 (−9.4)	42.6 $\rightarrow$ 55.1 (+12.5)
TimesFM-3	78.2 $\rightarrow$ 76.4 (−1.8)	73.5 $\rightarrow$ 75.5 (+2.0)	50.9 $\rightarrow$ 44.6 (−6.3)	45.2 $\rightarrow$ 49.7 (+4.6)

Table 16 | Skill score (wQL, %) of the LLMs and TimesFM-3 when moving from the second-longest to the longest context ($C_3 \rightarrow C_4$), and from the fewest to the most historical events at a fixed target, event length, context length and number of covariates (most events at least $8\times$ the fewest). Change in percentage points in brackets. Gemini-3.1-Pro drops at the longest context, but changes little with the number of events.

G. Skill scores under WAPE

Figure 8 gives the aggregate skill leaderboard under WAPE, and fig. 9 splits it into Replay vs Novel skill.

Further, the main text shows how wQL skill falls as the number of covariates K grows (fig. 4). The same trend holds under WAPE: every model loses skill on Novel futures as K grows. We provide here the aggregate skill leaderboard partitioned by the number of covariates, under WAPE, in fig. 10. Finally, figs. 11 to 13 give the WAPE counterparts of the number-of-events, event-length, and context-length figures (figs. 5 to 7); the trends match those under wQL.

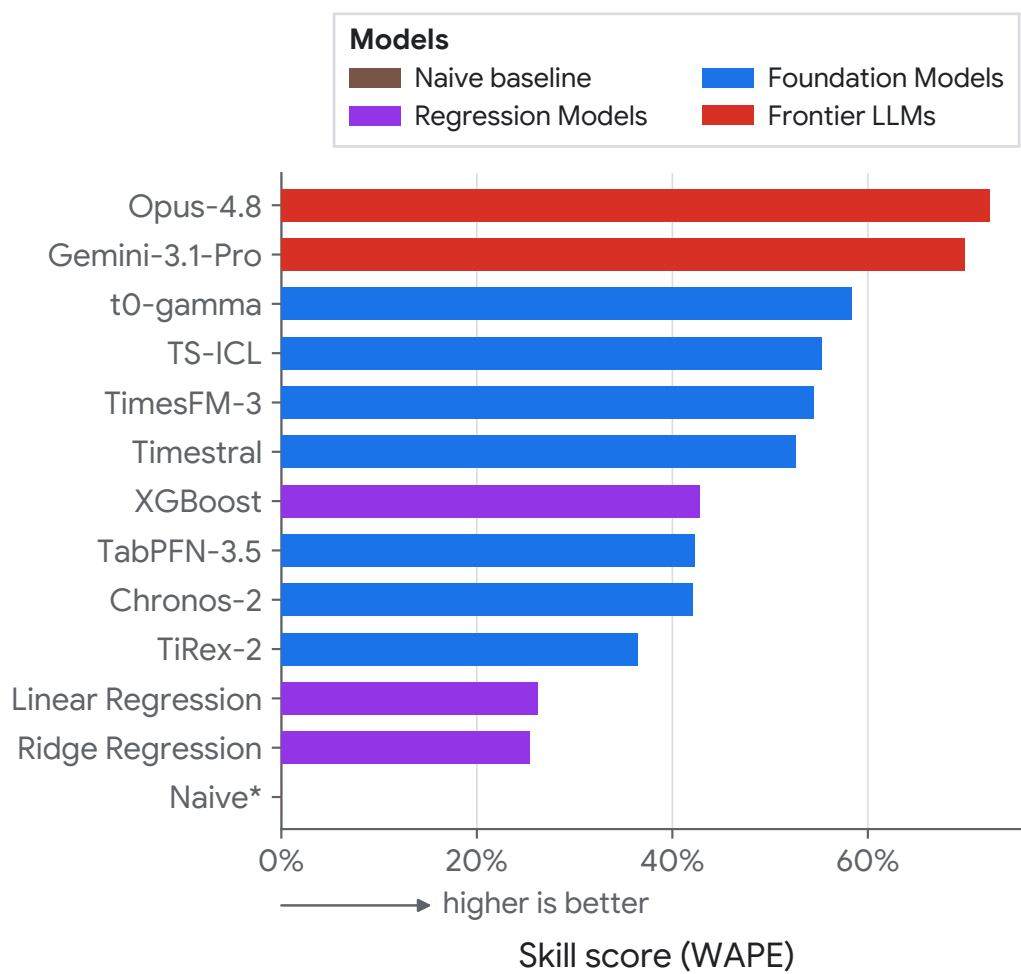


Figure 8 | Aggregate skill leaderboard on WHATIF-BENCH as per the WAPE metric.

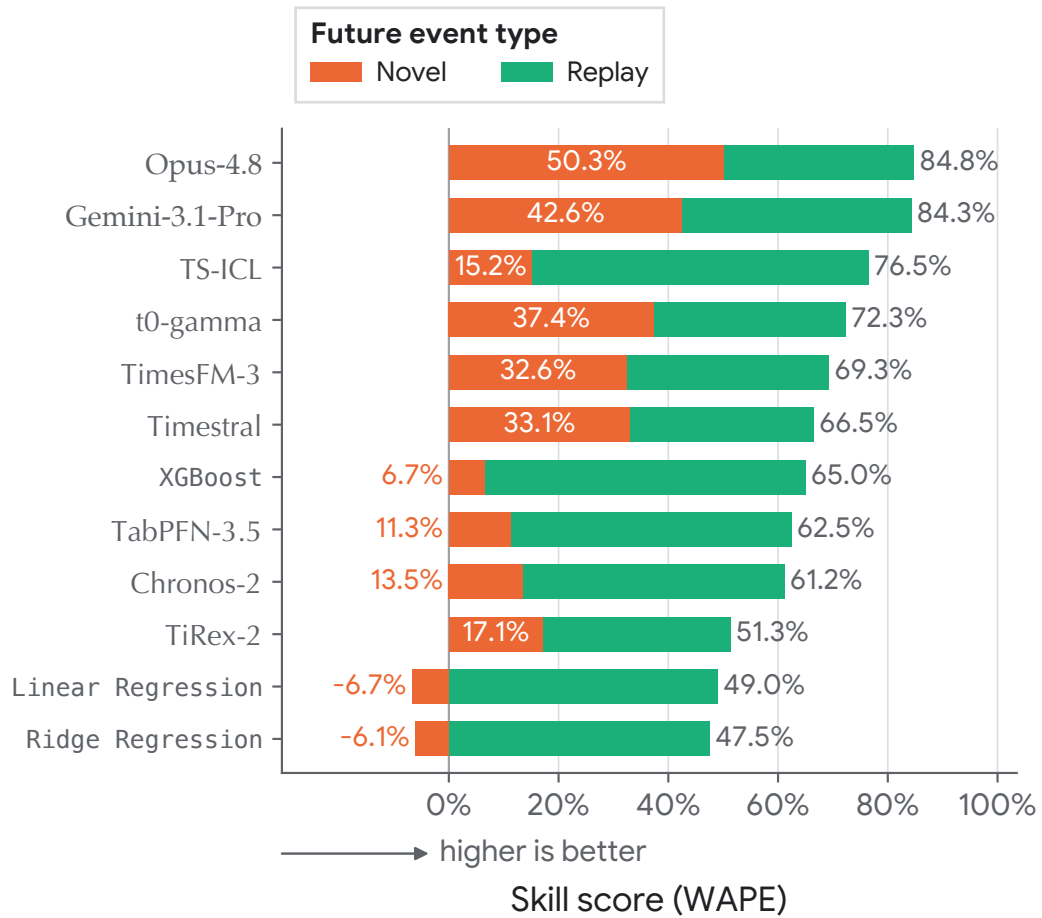


Figure 9 | Replay vs Novel skill of LLMs, numerical foundation models, and regression models on WHATIF-BENCH as per the WAPE metric.

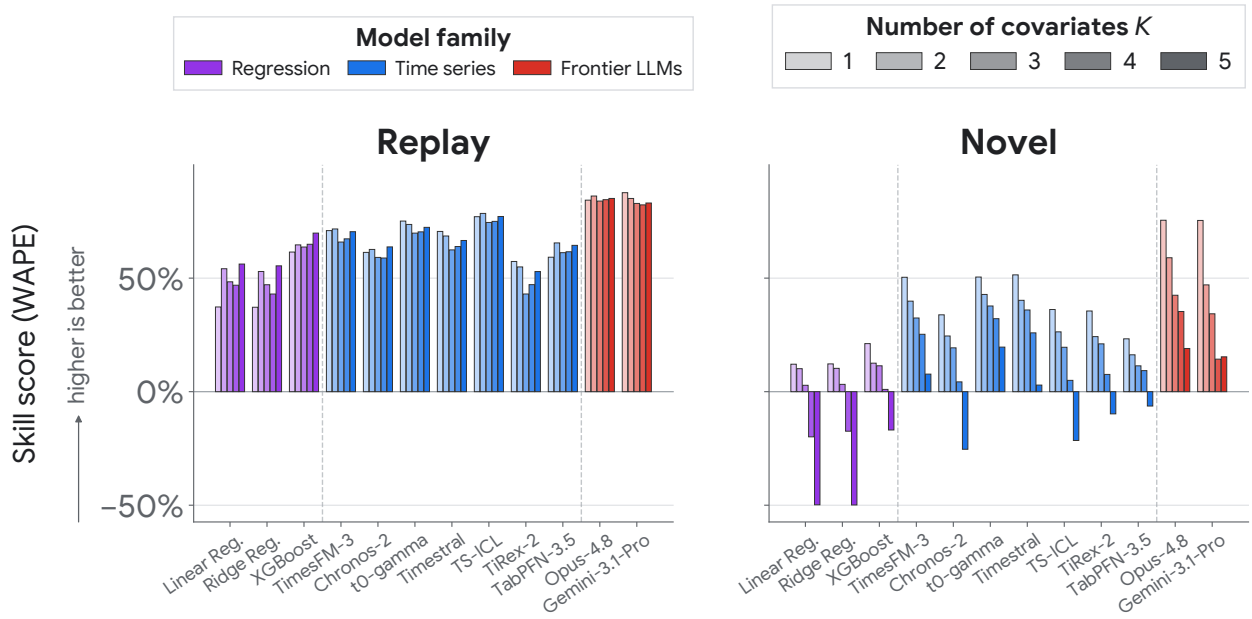


Figure 10 | Skill scores as per the WAPE metric on tasks with different number of covariates on WHATIF-BENCH, on Replay (left) and Novel (right) future types respectively.

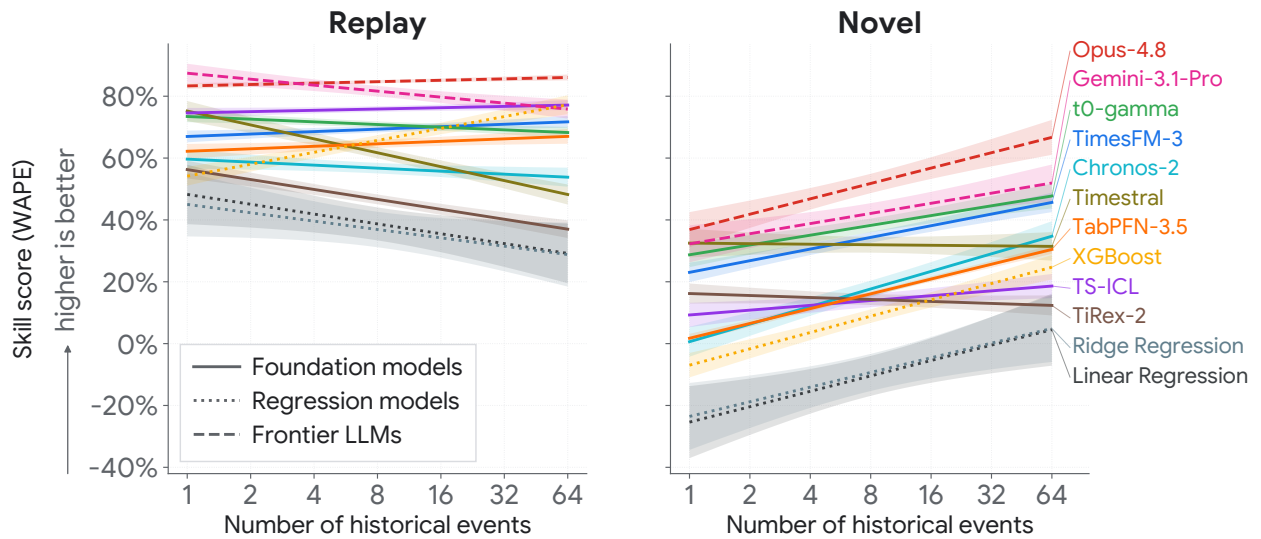


Figure 11 | Skill scores as per the WAPE metric with different number of historical events on WHATIF-BENCH, on tasks with Replay (left) and Novel (right) futures respectively. Lines as in Figure 5.

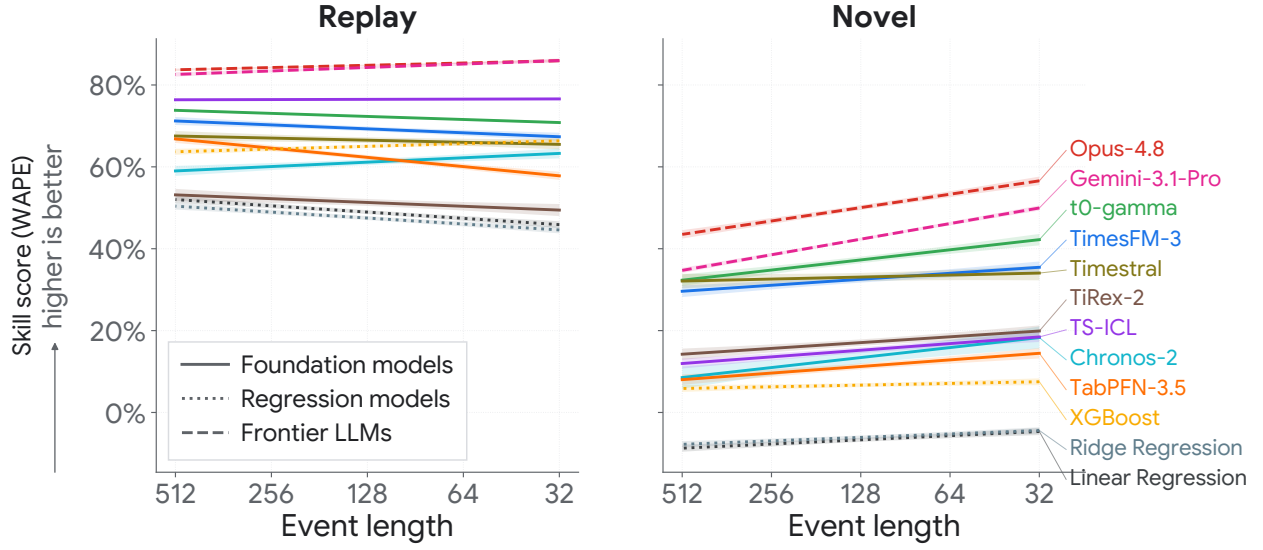


Figure 12 | Skill scores as per the WAPE metric with different event lengths on WHATIF-BENCH, on tasks with Replay (left) and Novel (right) futures respectively. Lines as in Figure 5.

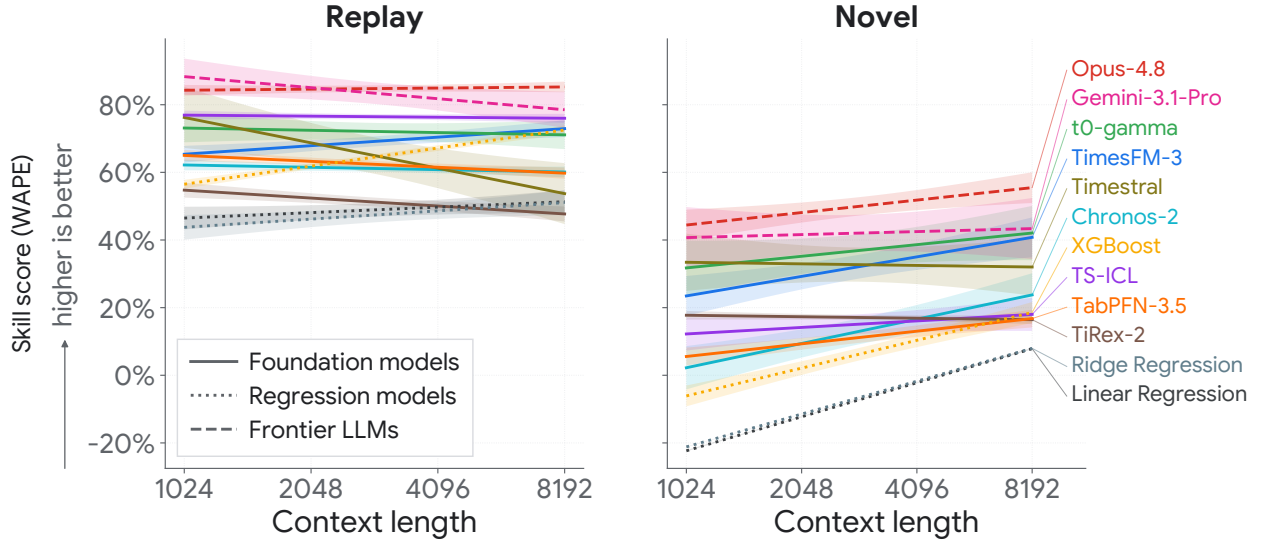


Figure 13 | Skill scores as per the WAPE metric with different context lengths on WHATIF-BENCH, on tasks with Replay (left) and Novel (right) futures respectively. All models receive the same input and produce the same output: each consumes the full history at every context length, without truncation, and forecasts the same horizon. Tasks are grouped by their nominal context length; for the longest events, the actual context is longer (Appendix B.3). Lines as in Figure 5.

H. Results by Simulator

The main text pools the four simulators. Here we separate the results by simulator: OpenDC (data centers), GridLAB-D (power grids), EPANET (water networks), and SUMO (road traffic). The simulators differ in size: EPANET contributes 13,265 tasks, SUMO 11,600, OpenDC 8,800, and GridLAB-D 4,000, so the pooled score is weighted toward the larger ones. All scores are wQL skill scores, measured against the naive baseline on the windows of the same simulator.

Overall score by simulator. Table 17 gives the overall skill score of each model on each simulator, next to the pooled score over all four.

Model (# Tasks)	OpenDC (8,800)	GridLAB-D (4,000)	EPANET (13,265)	SUMO (11,600)	All (37,665)
<i>Frontier LLMs</i>					
Claude-Opus-4.8 (High)	75.1% 	58.7% 	45.9% 	88.1% 	72.5% 
Claude-Opus-4.8 (Medium)	<u>73.4%</u> 	56.2% 	41.4% 	<u>87.0%</u> 	<u>70.3%</u> 
Gemini-3.1-Pro (High)	<u>73.0%</u> 	52.4% 	52.0% 	<u>83.7%</u> 	<u>70.0%</u> 
Gemini-3.1-Pro (Medium)	62.9% 	35.0% 	25.7% 	79.6% 	58.2% 
<i>Numerical Foundation Models</i>					
t0-gamma	58.6% 	58.4% 	59.1% 	74.6% 	64.5% 
TimesFM-3	56.0% 	54.8% 	57.9% 	70.4% 	61.6% 
TS-ICL	61.1% 	54.2% 	42.1% 	77.0% 	61.3% 
Timestral	47.2% 	55.1% 	56.4% 	69.8% 	59.2% 
TabPFN-3.5	59.7% 	48.3% 	37.6% 	57.1% 	50.8% 
Chronos-2	44.4% 	50.1% 	43.7% 	61.8% 	50.8% 
TiRex-2	40.9% 	45.6% 	41.4% 	56.0% 	46.7% 
<i>Regression Models[†]</i>					
XGBoost	58.2% 	21.4% 	4.6% 	63.9% 	42.9% 
Linear Regression	48.8% 	22.4% 	-1.7% 	33.7% 	26.2% 
Ridge Regression	48.3% 	22.8% 	-1.7% 	31.7% 	25.4% 
<i>Baseline</i>					
Naive*	0.0%	0.0%	0.0%	0.0%	0.0%

Table 17 | Skill score (wQL) of each model on WHATIF-BENCH, separately for each of the four simulators, and pooled over all of them (All). The number under each simulator name is its number of tasks (# Tasks). The best score per column is **bolded and highlighted** and the second-best is underlined. Note that Naive* represents a naive baseline which does not consume covariates. All other models are evaluated with covariates. [†]The simplest, untuned versions of the regression models; tuned variants are in Appendix J.

Takeaway. No model is best everywhere. Claude-Opus-4.8 (High) leads on OpenDC, GridLAB-D, and SUMO, but t0-gamma leads on EPANET. Scores are highest on SUMO for both LLM families and for six of the seven foundation models.

Replay and Novel futures by simulator. Figure 14 splits each score into Replay futures, which repeat covariate values already seen in the history, and Novel futures, which set the covariates beyond that range.

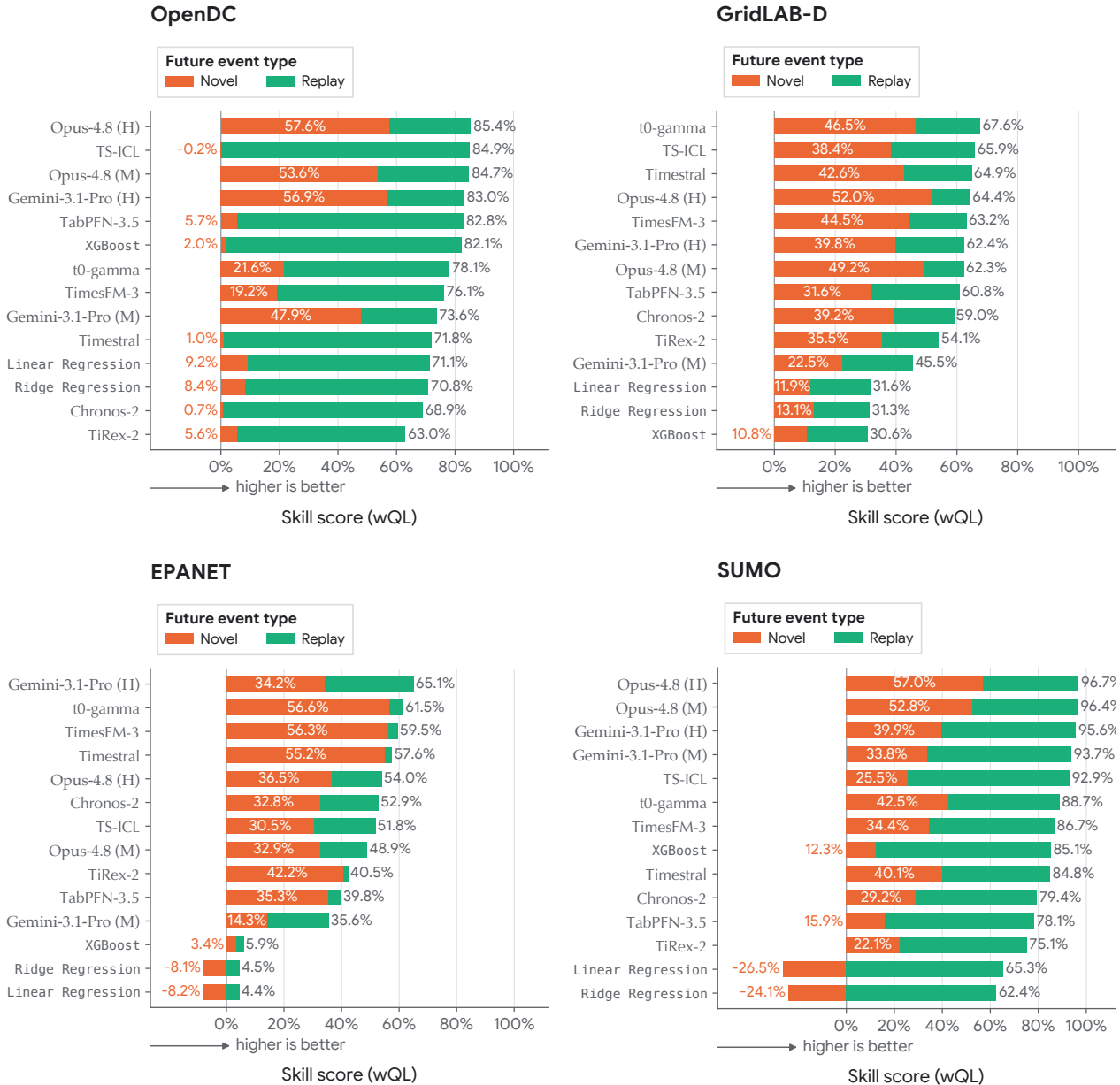


Figure 14 | Skill score (wQL) of LLMs, numerical foundation models, and regression models on Replay and Novel futures, separately for each simulator. In each panel, the orange segment is a model’s skill on Novel futures and the full bar is its skill on Replay futures; models are sorted by Replay skill within each panel, and all panels share the same horizontal axis. (H) and (M) mark High and Medium reasoning effort.

Takeaway. Novel futures are much harder than Replay futures on OpenDC and SUMO. On these two simulators, several models fall to about zero or below on Novel futures, for example TS-ICL on OpenDC and the linear and ridge regressions on SUMO. On EPANET and GridLAB-D, the drop from Replay to Novel is smaller.

The Replay–Novel gap. To compare the drop across models and simulators in one view, we define the *Replay–Novel gap* of a model on a simulator as its Replay skill minus its Novel skill, in percentage points. Both skills are measured against the naive baseline. The gap is the number of skill points a model loses when the future sets the covariates beyond the range seen in the history. A gap of

zero means the model handles Novel futures as well as Replay futures. A large gap means the model forecasts well when covariate values repeat, but does not carry what it has learned to values it has not seen. Figure 15 shows the gap for every model and simulator, with the mean over simulators in the last column and the mean over models in the last row.

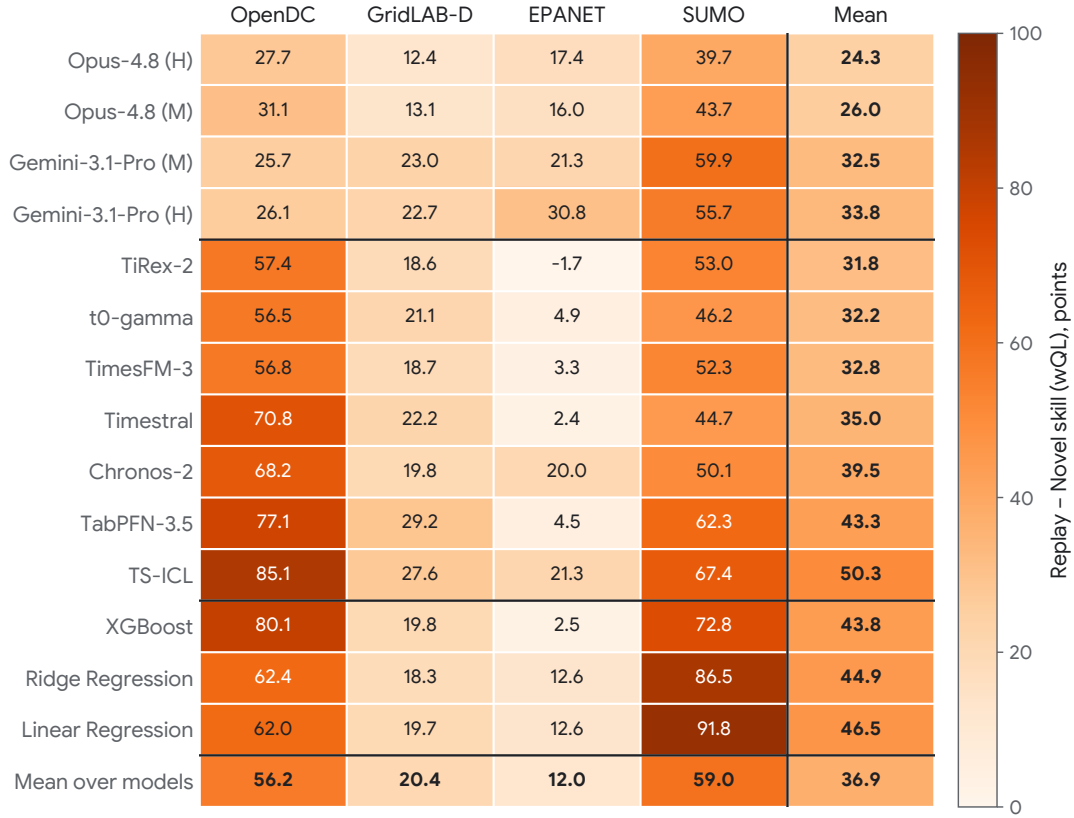


Figure 15 | Replay–Novel gap (Replay skill minus Novel skill, wQL, in percentage points) for each model and simulator. Larger values mean a larger drop from Replay to Novel futures. The last column is the mean over simulators and the last row is the mean over models. Rows are sorted by the mean gap within each group of models.

Takeaway. The gap between Replay and Novel skill is much larger on OpenDC and SUMO than on EPANET and GridLAB-D. Novel futures are therefore the harder case on OpenDC and SUMO, while on EPANET and GridLAB-D models lose relatively little. On OpenDC the four LLMs have the smallest gaps, and on SUMO the two Claude models do.

I. Skill Against a Covariate-Aware Baseline

The skill scores in the main text are measured against a naive baseline that copies the last value. We use it because it does not use covariates, so it is the same kind of forecast on every task and in every slice of the benchmark. Here we measure skill against a covariate-aware baseline, Linear Regression, to see how much models gain over a simple model that also uses the covariates. It sets a higher bar: it has a lower wQL than the naive baseline on 69.1% of the Replay tasks and on 55.7% of the Novel tasks.

We do not use it as the main reference because its strength depends on the slice of the benchmark. Its skill against the naive baseline is 49.0% on Replay futures and –6.7% on Novel futures; on SUMO, it

is 65.3% and -26.5%. A skill score against it therefore mixes how well a model forecasts with how well linear regression forecasts on the same slice. This makes comparisons across Replay and Novel futures, across simulators, and across the number of covariates harder to interpret.

Even so, linear regression is a valid reference for WHATIF-BENCH: it uses the covariates, as a practitioner’s first model would, and it sets a higher bar than copying the last value. We therefore offer it alongside the naive baseline, as an option in our benchmark leaderboard. Other models, such as XGBoost, can also be used as reference baselines for the skill score.

Takeaway. Measured against linear regression instead of the naive baseline, skill scores differ in three ways. First, every model scores much lower against linear regression than against the naive baseline, but the order of the models does not change. Second, compared with the skill against the naive baseline, skill against linear regression is much lower on Replay futures, where linear regression is strong, but not on Novel futures, where it is close to the naive baseline: for every model, skill on Novel futures is slightly higher against linear regression. As a result, the gap between Replay and Novel skill mostly disappears for the foundation models, while the LLMs, TS-ICL, and XGBoost keep it. Third, the trends with the number of covariates, the number of events, and the event length differ from those against the naive baseline, and some are reversed, because linear regression is affected by these factors too. The LLMs have the highest skill on Replay futures, and Claude-Opus-4.8 (High) has the highest on Novel futures, against either reference.

Leaderboard. Table 18 gives the skill score of each model against both references, under wQL and WAPE.

Model	wQL		WAPE	
	vs. Naive	vs. Linear	vs. Naive	vs. Linear
<i>Frontier LLMs</i>				
Claude-Opus-4.8 (High)	72.5%	63.1%	72.5%	63.1%
Claude-Opus-4.8 (Medium)	<u>70.3%</u>	<u>60.1%</u>	<u>70.3%</u>	<u>60.1%</u>
Gemini-3.1-Pro (High)	<u>70.0%</u>	<u>59.6%</u>	<u>70.0%</u>	<u>59.6%</u>
Gemini-3.1-Pro (Medium)	58.2%	43.4%	58.2%	43.4%
<i>Numerical Foundation Models</i>				
t0-gamma	64.5%	51.1%	58.4%	42.5%
TimesFM-3	61.6%	46.9%	54.5%	37.4%
TS-ICL	61.3%	45.3%	55.3%	37.8%
Timestral	59.2%	43.5%	52.7%	34.8%
TabPFN-3.5	50.8%	32.8%	42.3%	21.7%
Chronos-2	50.8%	32.2%	42.0%	20.2%
TiRex-2	46.7%	26.7%	36.5%	13.0%
<i>Regression Models</i> [†]				
XGBoost	42.9%	21.9%	42.9%	21.9%
Linear Regression	26.2%	0.0%	26.2%	0.0%
Ridge Regression	25.4%	-5.6%	25.4%	-5.6%
<i>Baseline</i>				
Naive*	0.0%	-35.5%	0.0%	-35.5%

Table 18 | Skill score of each model on WHATIF-BENCH under wQL and WAPE, measured against two reference models: the naive baseline that copies the last value (as in the main text) and a covariate-aware linear regression (Linear Regression). Each reference has a skill of 0 by construction. The best score per column is **bolded and highlighted** and the second-best is underlined. Note that Naive* represents a naive baseline which does not consume covariates. All other models are evaluated with covariates. [†]The simplest, untuned versions of the regression models; tuned variants are in Appendix J.

Takeaway. Every model has a much lower skill score against linear regression than against the naive baseline, but the order of the models does not change. For example, under wQL, Claude-Opus-4.8 (High) goes from 72.5% to 63.1% and TimesFM-3 from 61.6% to 46.9%. The same holds under WAPE.

Replay and Novel futures. Figure 16 shows the skill of each model on Replay and Novel futures against linear regression.

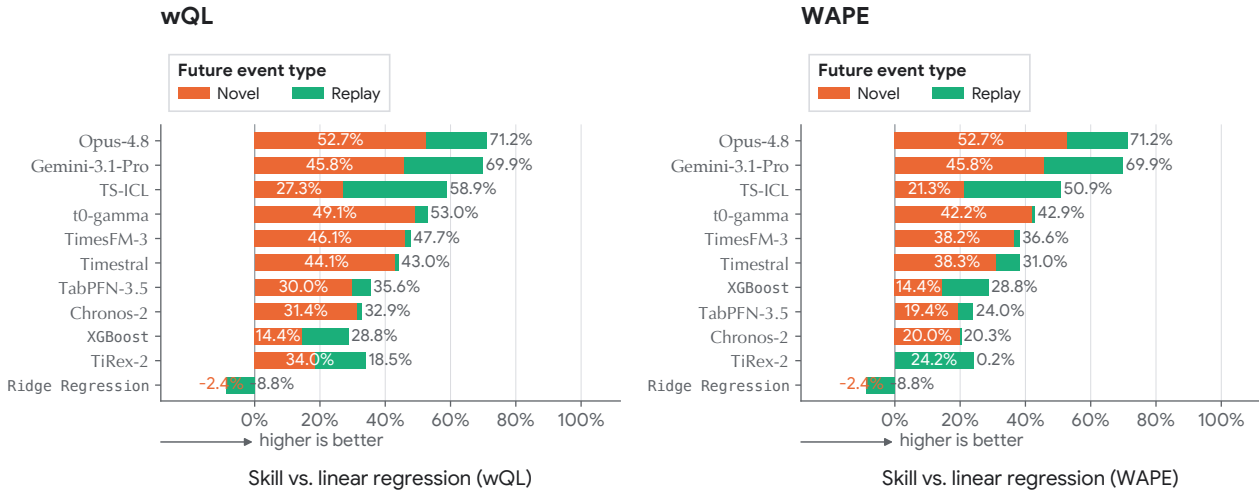


Figure 16 | Skill score against linear regression of LLMs, numerical foundation models, and regression models on Replay and Novel futures, under wQL (left) and WAPE (right). The orange segment is a model’s skill on Novel futures, and the full bar is its skill on Replay futures. Models are sorted by Replay skill.

Takeaway. Linear regression itself drops sharply from Replay to Novel futures (from 49.0% to −6.7% against the naive baseline). This is why the gap between Replay and Novel skill is much smaller against linear regression than against the naive baseline: the reference is much weaker on Novel futures than on Replay futures. For TimesFM-3, Timestrat, Chronos-2, and TiRex-2, skill on Novel futures is similar to or higher than skill on Replay futures, and t0-gamma and TabPFN-3.5 lose only about 4 to 6 points, which shows that these models are more reliable than linear regression in novel scenarios. Against linear regression, the LLMs, TS-ICL, and XGBoost still have a lower skill on Novel futures than on Replay futures. Note that skill scores against linear regression are overall much smaller than against the naive baseline. The same holds under WAPE.

Number of covariates. Figures 17 and 18 show how skill varies with the number of covariates K , under wQL and WAPE.

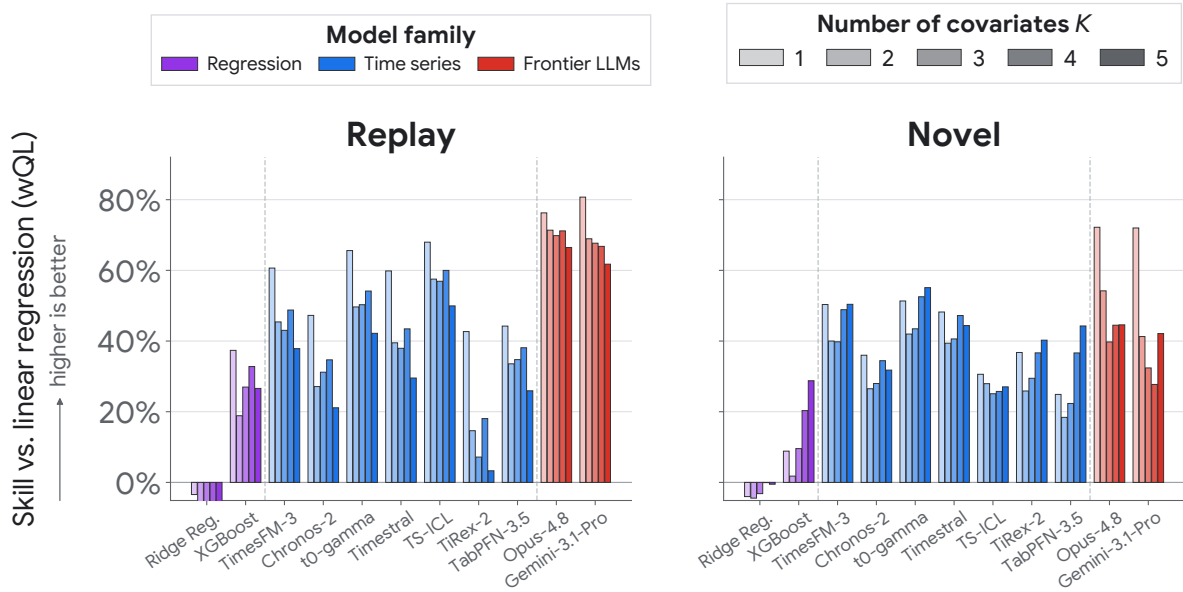
wQL

Figure 17 | Skill score (wQL) against linear regression for different numbers of covariates K , on Replay (left) and Novel (right) futures.

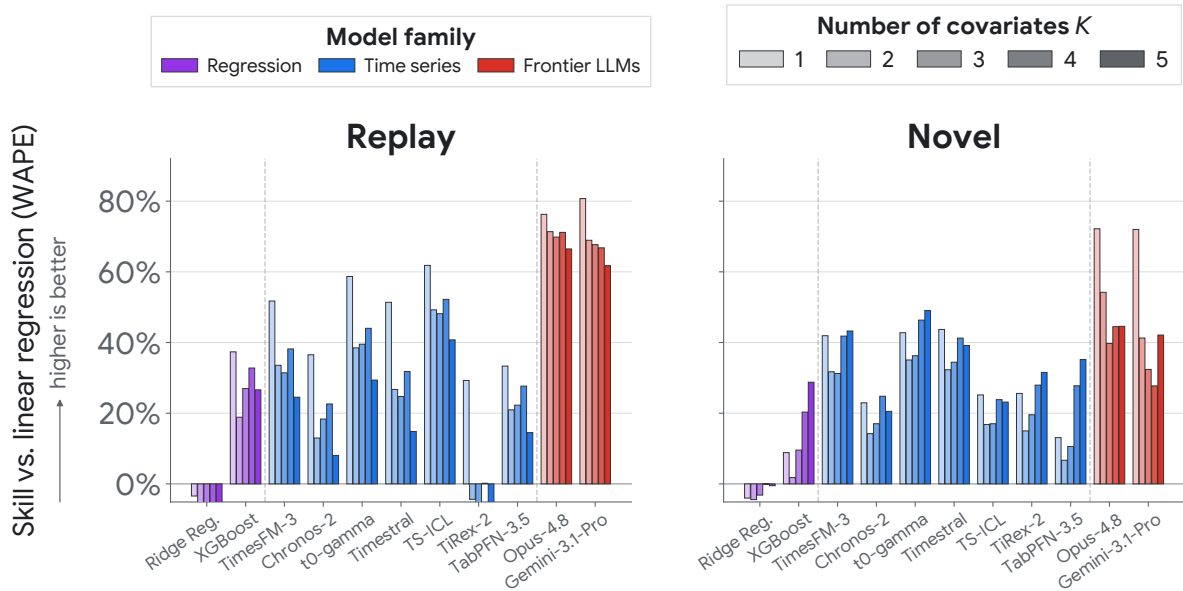
WAPE

Figure 18 | Skill score (WAPE) against linear regression for different numbers of covariates K , on Replay (left) and Novel (right) futures.

Takeaway. On Replay futures, skill against linear regression falls as covariates are added, whereas against the naive baseline it is nearly flat. This is because linear regression stays strong on Replay futures as covariates are added (its skill against the naive baseline is 37.3% at $K = 1$ and 56.2% at $K = 5$), while the models do not improve. On Novel futures, skill against the naive baseline falls sharply with more covariates. Against linear regression, it falls little or not at all for the foundation models, because linear regression itself falls as fast (from 12.1% at $K = 1$ to -49.8% at $K = 5$ against the naive baseline). Against linear regression, the skill of the LLMs on Novel futures still falls as covariates are added. Skill scores against linear regression are overall much smaller than against the naive baseline. The same holds under WAPE.

Number of historical events. Figures 19 and 20 show how skill varies with the number of historical events in the context, under wQL and WAPE.

wQL

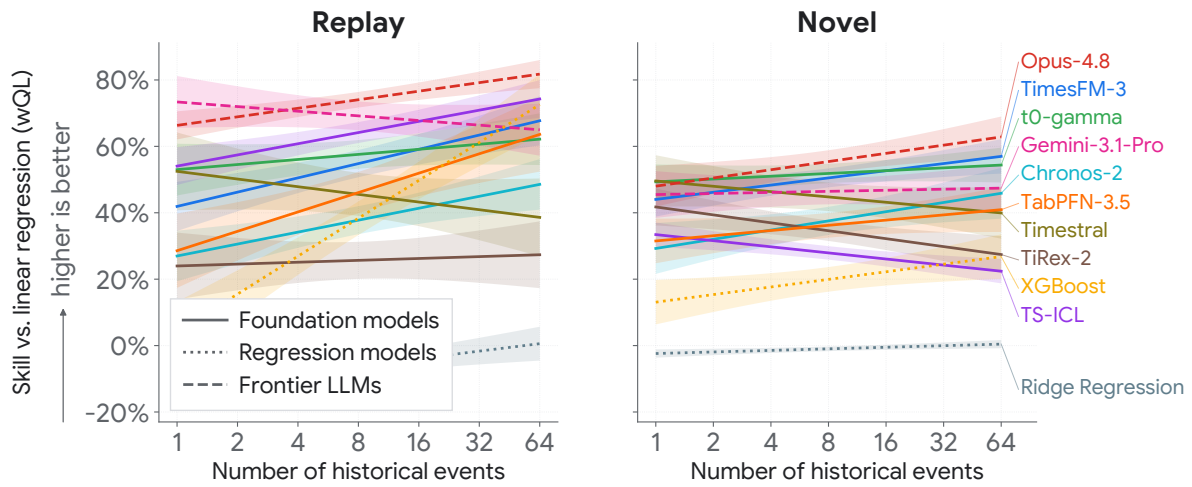


Figure 19 | Skill score (wQL) against linear regression for different numbers of historical events, on Replay (left) and Novel (right) futures. Lines as in Figure 5.

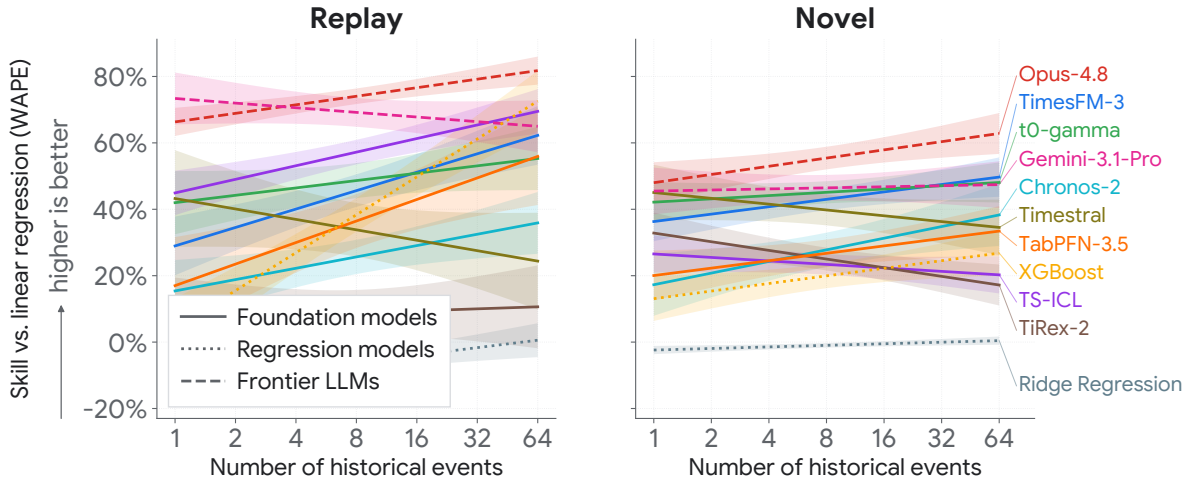
WAPE

Figure 20 | Skill score (WAPE) against linear regression for different numbers of historical events, on Replay (left) and Novel (right) futures. Lines as in Figure 5.

Takeaway. Against the naive baseline, skill on Replay futures is flat as the number of historical events grows, and skill on Novel futures rises. Against linear regression, more events raise skill on Replay futures for most models, and raise it less on Novel futures. Linear regression itself improves on Novel futures with more events (from -27.3% at one event to 9.3% at 64, against the naive baseline), so most models gain only somewhat more from history than it does there. On Replay futures, the models gain more from history than linear regression does. The same holds under WAPE.

Event length. Figures 21 and 22 show how skill varies with the length of the events, under wQL and WAPE.

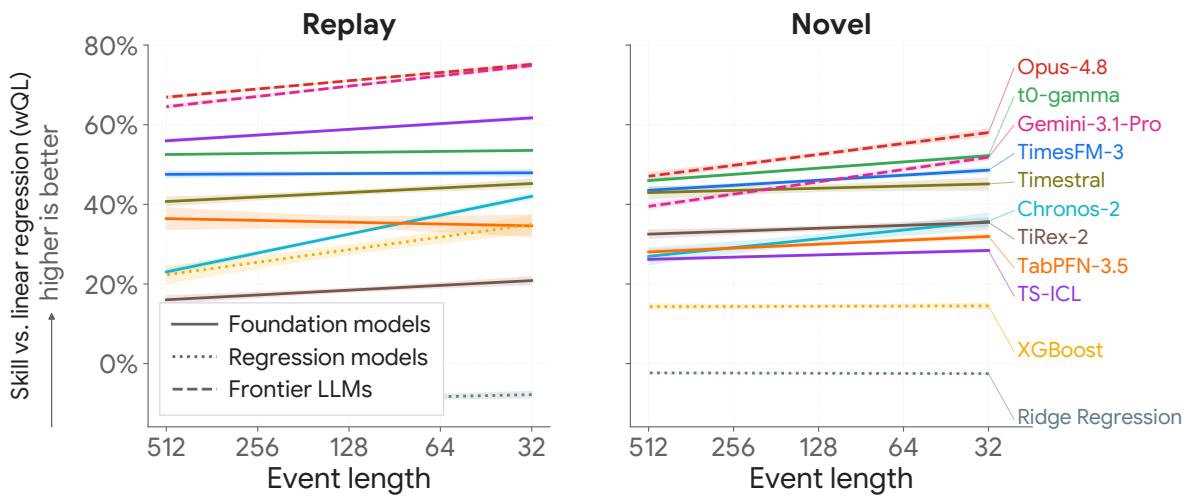
wQL

Figure 21 | Skill score (wQL) against linear regression for different event lengths, on Replay (left) and Novel (right) futures. Lines as in Figure 5.

WAPE

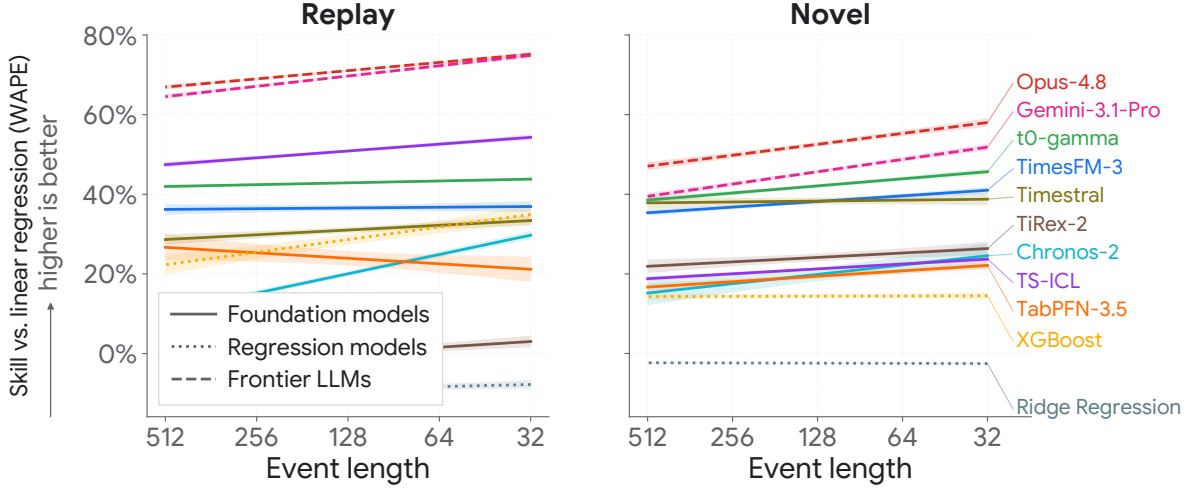


Figure 22 | Skill score (WAPE) against linear regression for different event lengths, on Replay (left) and Novel (right) futures. Lines as in Figure 5.

Takeaway. Against the naive baseline, skill on Novel futures falls as events get longer. Linear regression itself falls only slightly on Novel futures with longer events (from -4.1% at length 32 to -7.5% at length 512, against the naive baseline). Against it, the skill of the models therefore still falls as events get longer, by about as much as against the naive baseline. Skill on Replay futures changes little with event length against the naive baseline, but falls slightly against linear regression, which itself improves on Replay futures with longer events. The same holds under WAPE.

Other views. Our code supports further views that we do not show here, such as results for each simulator, and other aggregations and metrics, such as win rates. It can also compute every table and figure in this appendix against a different reference model.

J. Tuned Regression Baselines

The regression baselines in the main text are kept as simple as possible: each one is fit on the history of a task with one training row per time step, predicting the target from the covariates at that step, with default settings. Here we explore whether a minimal amount of tuning improves them, to check whether our conclusions depend on how weak these baselines are. **Tuning improves their scores, but does not change any of our conclusions.** We tune three things: how the training rows are built, which features the models use, and how many past values of the target they see (and, for Ridge Regression, its regularization strength). We select the variants on the test set, so we do not consider them valid baselines; this is also why we use the plain models for the main benchmark. The tuning is also not exhaustive: we do not tune other model hyperparameters, such as the learning rate or the tree depth of XGBoost. Even so, it gives a sense of how much these simple baselines can be improved.

Paired training. With one training row per time step, the regression models cannot directly use past values of the target: a feature such as the target one step earlier is unknown at test time for every forecast step after the first, unless forecasts are fed back as inputs. We instead take every feature relative to a *forecast origin*: the last time step at which the target is known when the forecast is made, which is t at test time. We design the training rows as pairs of an origin τ and a horizon h , where

each row predicts the target $X_{\tau+h}$. We sample the origins from the history, as N-BEATS does to create its training examples (Oreshkin et al., 2020). For each task, we draw a horizon h uniformly from 1 to the prediction length $T - t$, and then an origin τ uniformly from the history, such that $\tau + h \leq t$ and enough history precedes τ to compute the features. We draw up to 8,000 such pairs per task. At test time, the origin is $\tau = t$, and the model predicts each step X_{t+h} , $h = 1, \dots, T - t$, separately, using the known future covariates.

Variants. We vary the features and the number of target lags. For predicting X_{t+h} from the origin t , the features are:

- $\mathbf{z}_{t+h}$: the covariates at the step being predicted (always used, and the only feature of the plain models);
- $\mathbf{z}_{t+h-1}$: the covariates one step earlier;
- $\mathbf{z}_t$: the covariates at the origin;
- $\Delta\mathbf{z} = \mathbf{z}_{t+h} - \mathbf{z}_t$: the change in the covariates since the origin;
- s_t : the mean, standard deviation, and slope of the last 32 target values up to the origin, $X_{t-31}, \dots, X_t$;
- h : the horizon.

In training, the sampled origin τ takes the place of t . Lags $\leq \ell$ are the values of the target at lags 1, 2, 3, 6, 12, 24, 48, 96, 128, 256 up to ℓ , where lag L is X_{t-L+1} . We explored 34 variants across the three models. Table 19 shows a selection that adds features and lags step by step, as a guide for tuning these models, together with the best variant of each model.

Training	Features	Lags	Skill	Replay	Novel	Gap
Linear Regression						
Plain (main text)	$\mathbf{z}_{t+h}$	—	26.2%	49.0%	-6.7%	55.7
Paired	$\mathbf{z}_{t+h}$	—	25.8%	48.7%	-7.1%	55.8
Paired	$\mathbf{z}_{t+h}, s_t$	—	30.7%	52.3%	-0.6%	52.9
Paired	$\mathbf{z}_{t+h}, \mathbf{z}_t, s_t, h$	—	32.9%	54.8%	0.2%	54.6
Paired	all	—	28.9%	52.5%	-6.5%	59.0
Paired	all	≤ 3	29.5%	53.1%	-5.7%	58.8
Paired	all	≤ 24	29.4%	53.0%	-5.8%	58.8
Ridge Regression						
Plain (main text)	$\mathbf{z}_{t+h}$	—	25.4%	47.5%	-6.1%	53.6
Paired	$\mathbf{z}_{t+h}$	—	25.4%	47.9%	-6.8%	54.7
Paired	$\mathbf{z}_{t+h}$	1	30.5%	51.5%	0.5%	51.0
Paired	$\mathbf{z}_{t+h}$	≤ 24	30.6%	51.6%	0.6%	50.9
Paired	$\mathbf{z}_{t+h}$	≤ 48	30.8%	51.7%	0.9%	50.7
Paired	all	≤ 48	32.7%	54.5%	0.7%	53.8
XGBoost						
Plain (main text)	$\mathbf{z}_{t+h}$	—	42.9%	65.0%	6.7%	58.3
Paired	$\mathbf{z}_{t+h}$	—	42.5%	64.7%	6.4%	58.3
Paired	$\mathbf{z}_{t+h}$	1	45.8%	67.0%	11.1%	56.0
Paired	$\mathbf{z}_{t+h}$	≤ 24	46.7%	68.4%	10.1%	58.3
Paired	$\mathbf{z}_{t+h}$	≤ 48	47.2%	68.7%	11.2%	57.5
Paired	all	≤ 24	50.4%	71.4%	13.9%	57.5
Paired	all	≤ 48	50.8%	71.8%	14.2%	57.6
Paired	all	≤ 96	51.6%	72.2%	15.8%	56.3
Paired	all	≤ 128	51.4%	72.0%	15.7%	56.3
Paired	all	≤ 256	50.7%	71.4%	15.0%	56.4

Table 19 | Skill score (wQL) of the tuned regression variants, measured against the naive baseline. Rows add features and target lags step by step; the best variant of each model is in bold. *all* is $\mathbf{z}_{t+h}, \mathbf{z}_{t+h-1}, \mathbf{z}_t, \Delta \mathbf{z}, s_t, h$. Ridge Regression uses a regularization strength of 10 in every row shown. Gap is Replay skill minus Novel skill, in percentage points. The variants are selected on the test set and are not valid baselines.

Takeaway. Minimal tuning raises the skill of the regression baselines by 7 to 9 points. Paired training alone does not help; the gains come from the features it allows, mainly the recent values of the target, as lags or as a summary. Even tuned, XGBoost only reaches TabPFN-3.5 and Chronos-2, and stays well below t0-gamma, TimesFM-3, TS-ICL, Timestral, and the LLMs. The drop from Replay to Novel futures stays as large as for the plain models.

Replay and Novel futures. Figure 23 shows the best variant of each model on Replay and Novel futures, next to the other models in the paper.

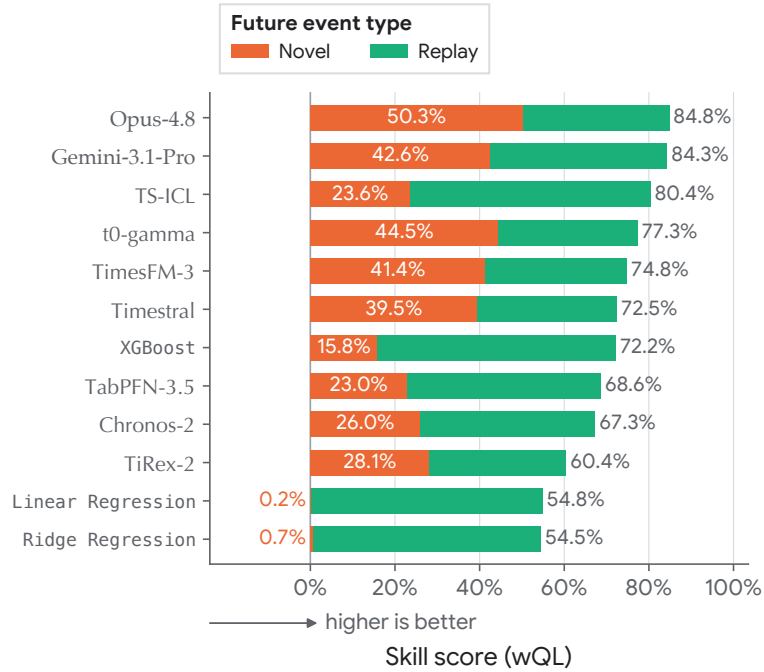


Figure 23 | Skill score (wQL) on Replay and Novel futures, with the tuned variants of Linear Regression, Ridge Regression, and XGBoost (best on the test set) in place of the plain ones. The orange segment is a model’s skill on Novel futures, and the full bar is its skill on Replay futures. Models are sorted by Replay skill.

Takeaway. Tuning raises the regression baselines on both Replay and Novel futures, but they still drop sharply between the two: tuned XGBoost goes from 72.2% to 15.8%, and tuned Linear Regression from 54.8% to 0.2%. Tuned XGBoost is ahead of several foundation models on Replay futures, but below every foundation model on Novel futures.

Other axes. Figures 24 to 27 show the tuned variants along the number of covariates, the number of historical events, the event length, and the context length.

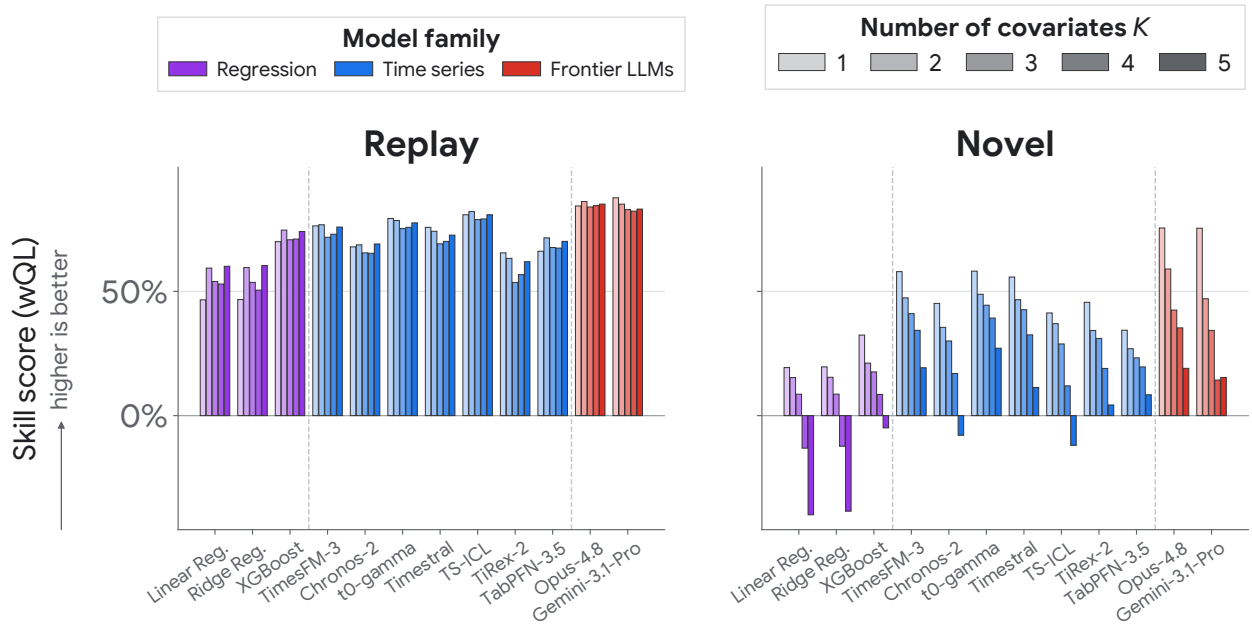


Figure 24 | Skill score (wQL) for different numbers of covariates, on Replay (left) and Novel (right) futures, with the tuned regression variants.

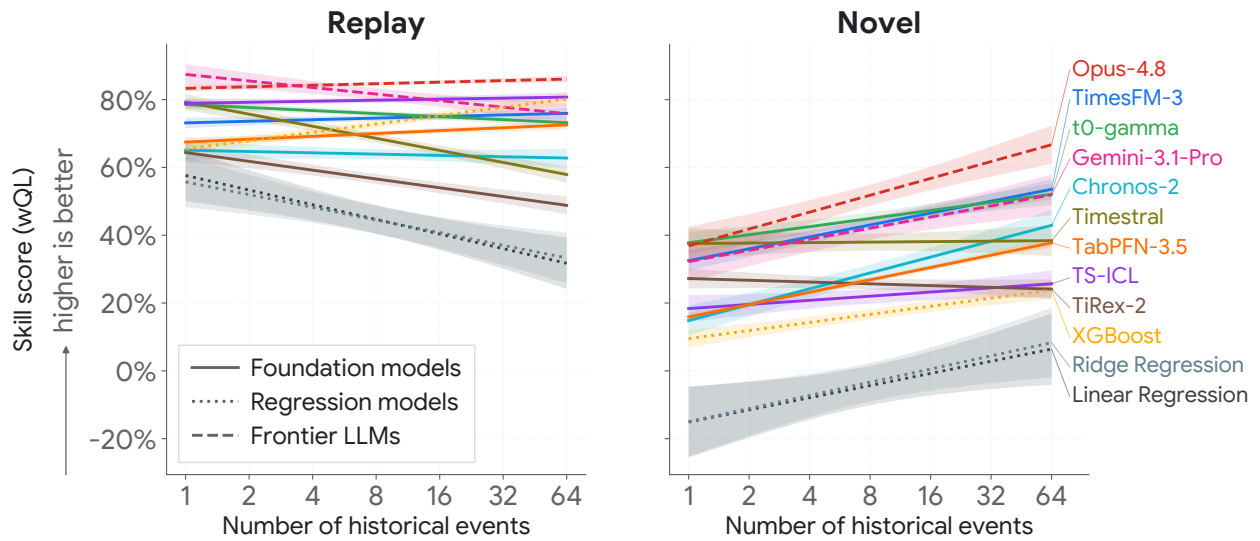


Figure 25 | Skill score (wQL) for different numbers of historical events, on Replay (left) and Novel (right) futures, with the tuned regression variants. Lines as in Figure 5.

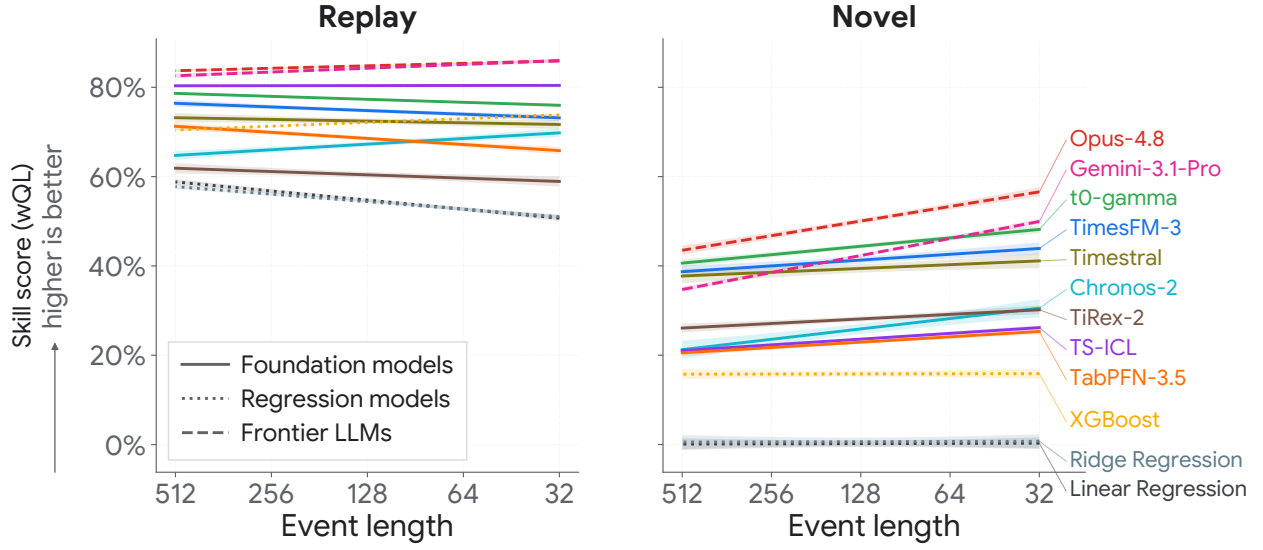


Figure 26 | Skill score (wQL) for different event lengths, on Replay (left) and Novel (right) futures, with the tuned regression variants. Lines as in Figure 5.

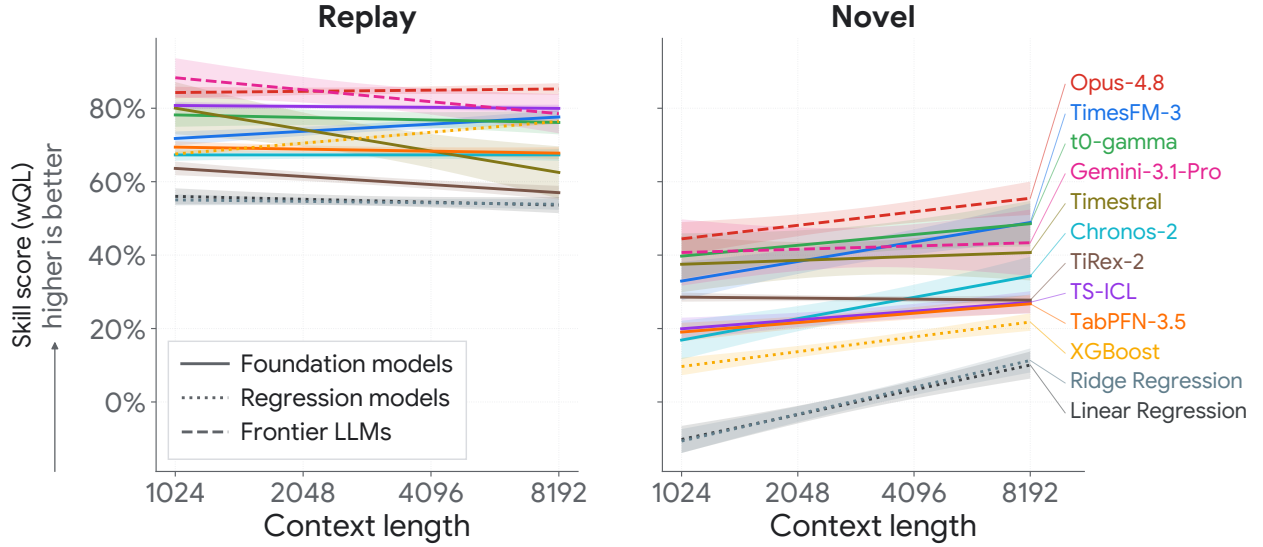


Figure 27 | Skill score (wQL) for different context lengths, on Replay (left) and Novel (right) futures, with the tuned regression variants. Lines as in Figure 5.

Takeaway. The tuned variants follow the same trends as the plain ones. On Novel futures, their skill falls as more covariates are set together, rises with more historical events and a longer context, and changes little with the event length. Along every axis, they stay well below the best models on Novel futures.

K. Win Rates

The main text reports skill scores. Here we report *win rates*: the fraction of task windows on which one model has a lower error than another, with ties counted as half a win. Table 20 gives each model's average win rate over all other models. Figures 28 and 29 show every pairwise comparison.

Model	wQL Win Rate	WAPE Win Rate	Inf. time / series (s)
Frontier LLMs			
Claude-Opus-4.8 (High)	63.2% 	68.6% 	270.13
Claude-Opus-4.8 (Medium)	60.1% 	<u>65.5%</u> 	206.29
Gemini-3.1-Pro (High)	58.4% 	63.4% 	355.28
Gemini-3.1-Pro (Medium)	46.0% 	50.9% 	97.35
Numerical Foundation Models			
t0-gamma	66.2% 	60.6% 	0.043
Timestral	<u>63.5%</u> 	58.3% 	0.048
TimesFM-3	<u>62.7%</u> 	57.1% 	0.419
TS-ICL	61.6% 	54.8% 	0.083
Chronos-2	53.8% 	48.8% 	0.067
TabPFN-3.5	50.3% 	44.2% 	21.26
TiRex-2	46.7% 	41.2% 	0.079
Regression Models[†]			
XGBoost	35.4% 	41.1% 	0.068
Ridge Regression	31.5% 	36.7% 	0.002
Linear Regression	31.1% 	36.2% 	0.002
Baseline			
Naive*	19.3% 	22.5% 	0.072

Table 20 | Average win rate of each model on WHATIF-BENCH under wQL (probabilistic) and WAPE (point) metrics, alongside inference latency. A model’s win rate is the fraction of task windows on which it has a lower error than another model, averaged over all other models in the table (ties count half). The best score per column is **bolded** and the second-best is underlined. Note that Naive* represents a naive baseline which does not consume covariates. All other models are evaluated with covariates. [†]The simplest, untuned versions of the regression models; tuned variants are in Appendix J.

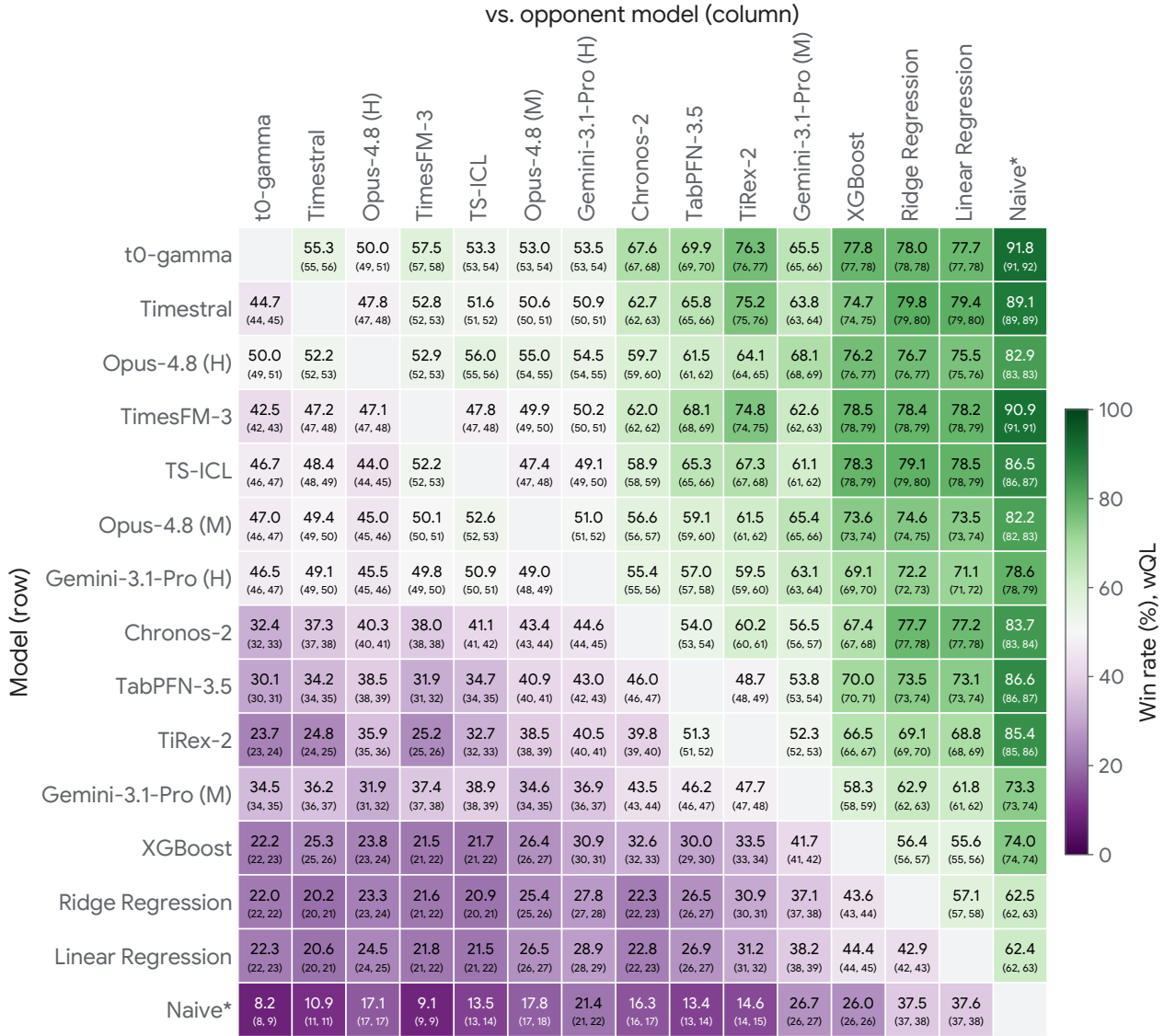


Figure 28 | Pairwise win rates on WHATIF-BENCH under the wQL metric. Each cell is the win rate (%) of the row model against the column model, with a bootstrap 95% confidence interval below it; the two cells across the diagonal add up to 100%. Models are ordered by their average win rate. Naive* does not consume covariates.

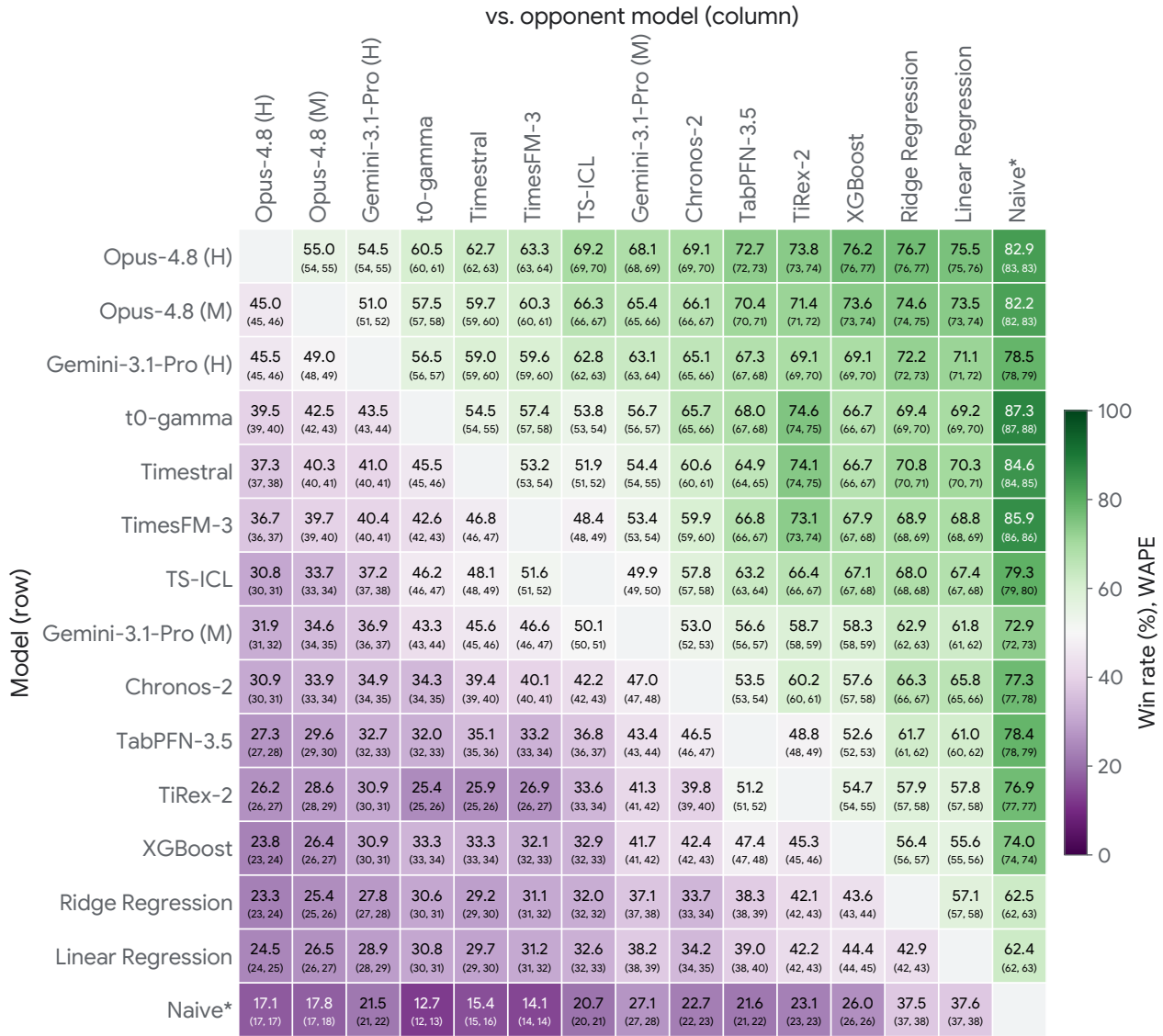


Figure 29 | Pairwise win rates on WHATIF-BENCH under the WAPE metric. Each cell is the win rate (%) of the row model against the column model, with a bootstrap 95% confidence interval below it; the two cells across the diagonal add up to 100%. Models are ordered by their average win rate. Naive* does not consume covariates.

L. Example tasks

We show example tasks here to make the benchmark construction concrete. As described in the main text (Section 3.3), the axes along which we build WHATIF-BENCH are:

- **Future type:** Replay or Novel.
- **Number of covariates** $K \in \{1, 2, 3, 4, 5\}$.
- **Context / prediction length** c : from C_1 (context 1024, prediction 128) to C_4 (context about 8,000, prediction 512 to 864, depending on the simulator), with longer contexts for the longest events (Appendix B.3).
- **Event length** $\ell \in \{32, 64, 128, 256, 512\}$.
- **Number of events** E : from the fewest (one or two) to the most the context permits, up to 64.

As set out in the construction recipe (Section 3.3), this is repeated for each target in each simulator.

Crucially, variations along one axis are built with every other axis held fixed. For example, the Replay and Novel tasks come from the same history, and only the future differs. This lets each axis be read on its own.

Each example below shows the history and the future of the target, together with its covariates. Except for the covariate axis, every example uses a single covariate, to keep the figures easy to read. History and future are separated by a dashed forecast-start line; in-context events are shaded grey and the future event red. Only the Replay future is drawn for every axis except the Replay-versus-Novel one. Some panels also show the pre-intervention target, the dotted line marking what the target would be with no event; it is shown purely for illustration and is not part of any task, and models never receive it.

L.1. Replay and Novel futures

Each figure shows a Replay task and a Novel task over the same history. The history and its covariate are identical across the two; only the future event changes. Replay reuses covariate values already seen in the history, while Novel pushes them beyond that range.

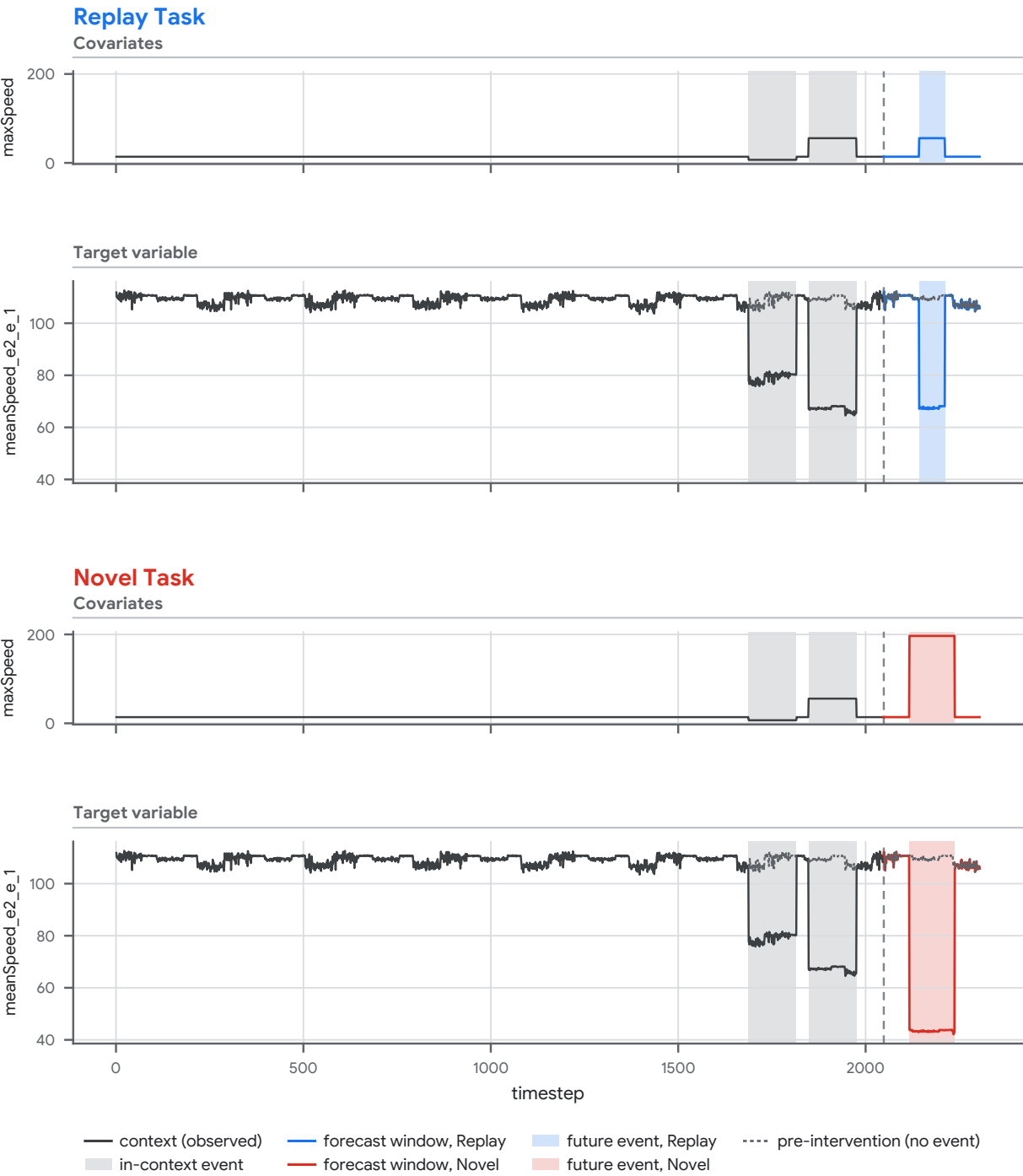


Figure 30 | A Replay task and a Novel task over a shared history. The dotted pre-intervention target is shown for illustration only and is not part of the task.

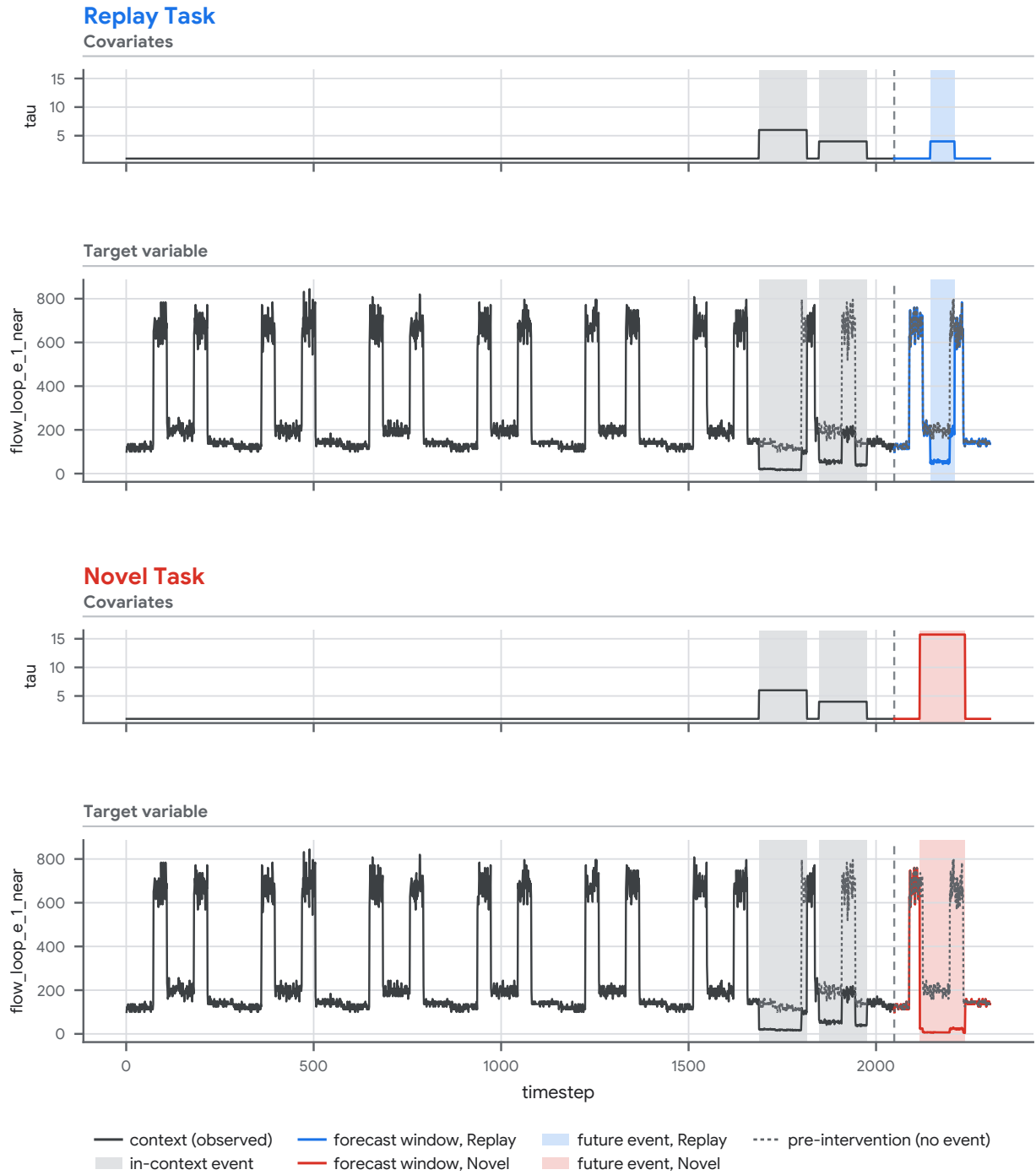


Figure 31 | A Replay task and a Novel task over a shared history. The dotted pre-intervention target is shown for illustration only and is not part of the task.

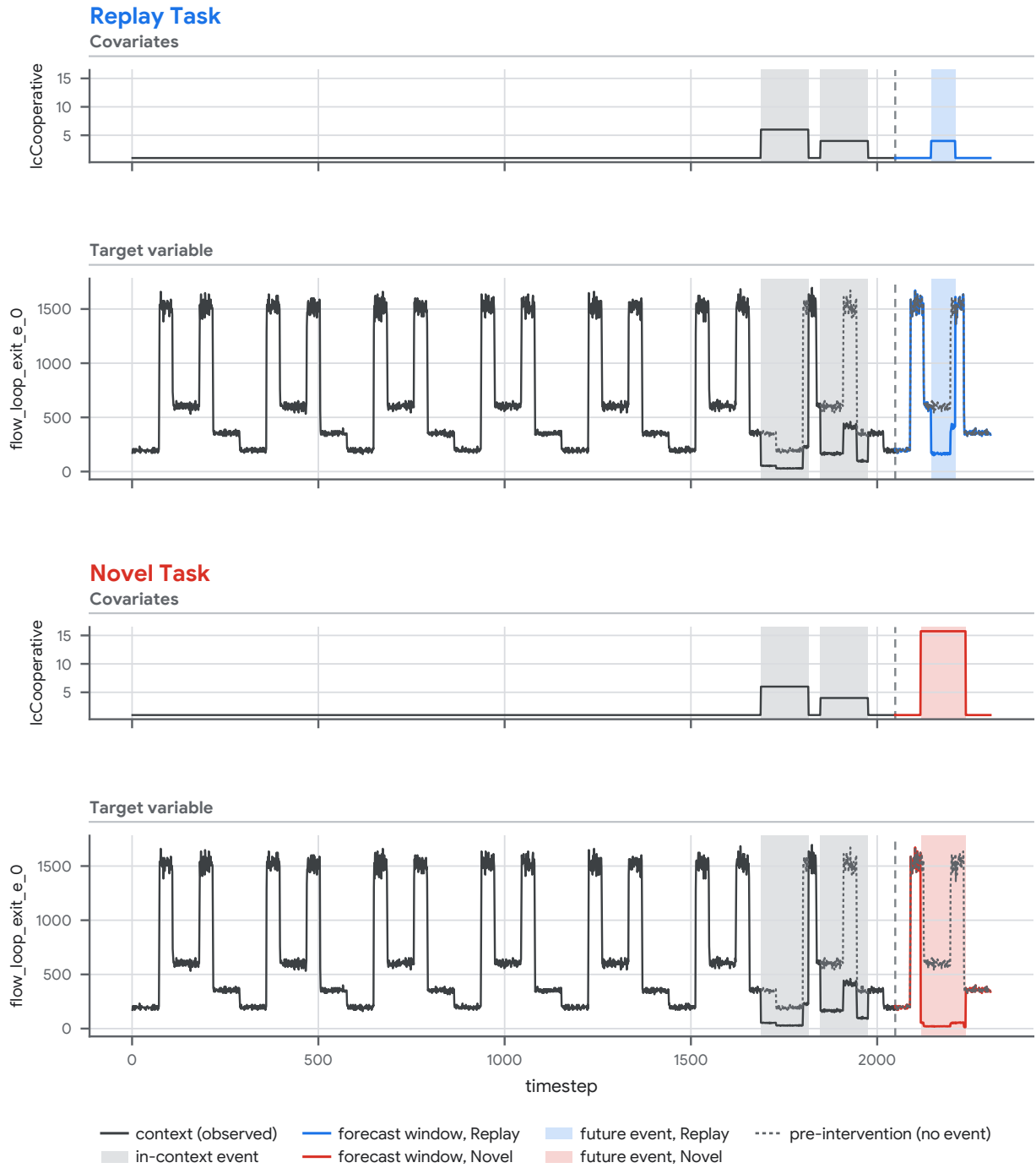


Figure 32 | A Replay task and a Novel task over a shared history. The dotted pre-intervention target is shown for illustration only and is not part of the task.

L.2. Event length

Each figure varies the event length ℓ across columns, holding the target, covariate, context, and number of events fixed. Longer events hold the covariate in place for longer, so the target's response is seen over a longer window.

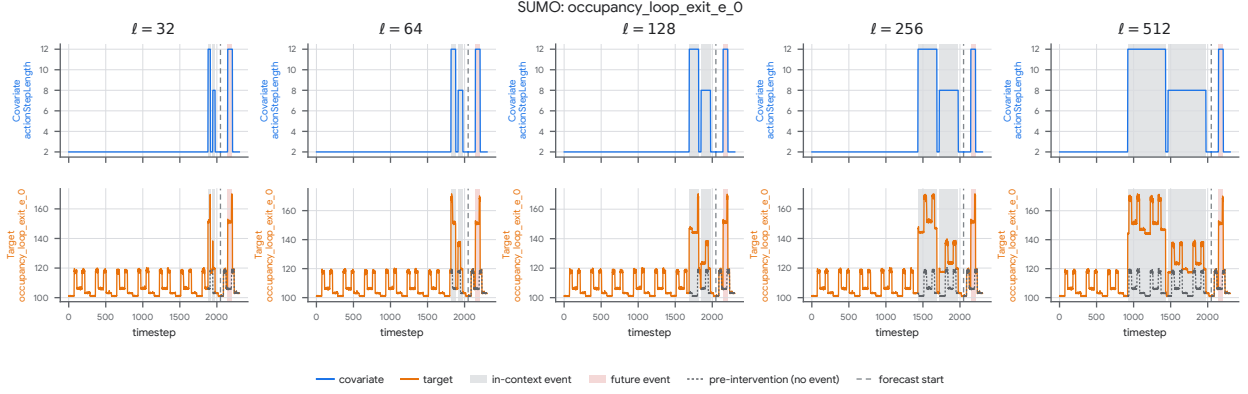


Figure 33 | Tasks that differ only in event length $\ell = 32, 64, 128, 256, 512$. The dotted pre-intervention target is shown for illustration only and is not part of the task.

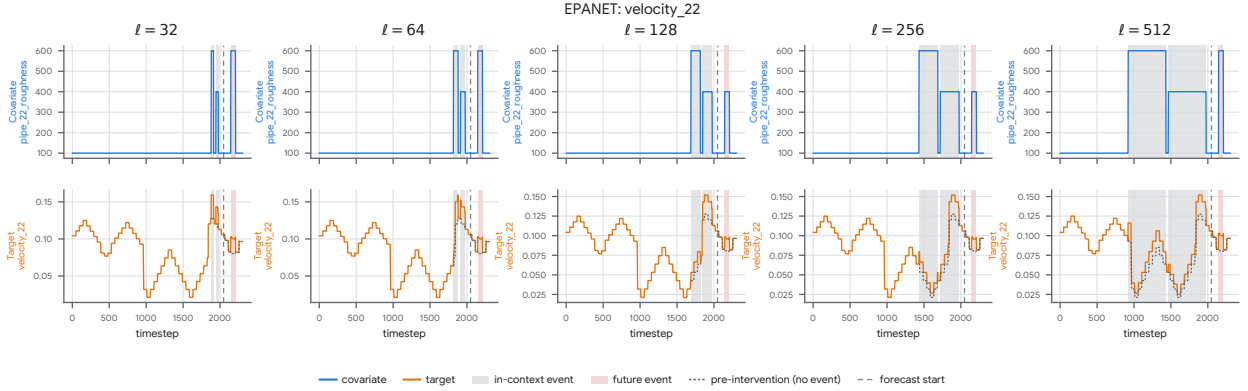


Figure 34 | Tasks that differ only in event length $\ell = 32, 64, 128, 256, 512$. The dotted pre-intervention target is shown for illustration only and is not part of the task.

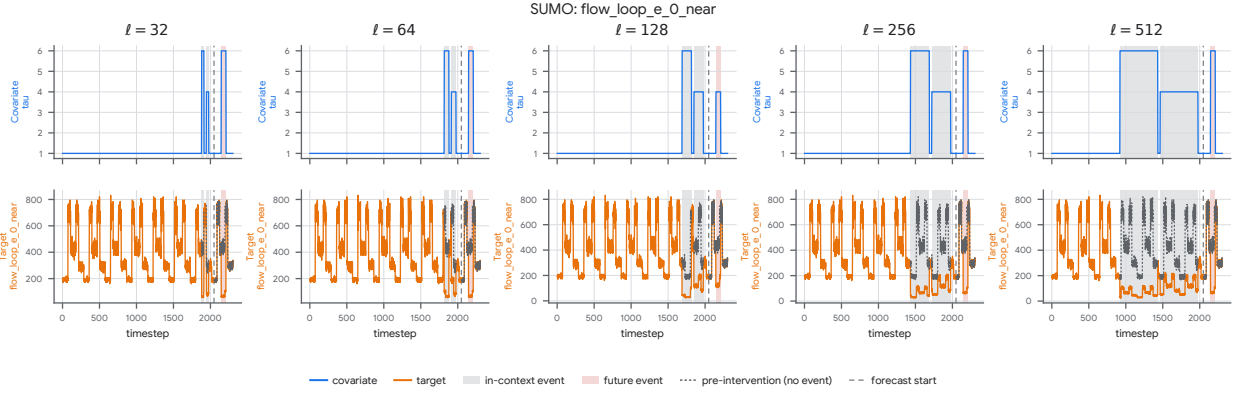


Figure 35 | Tasks that differ only in event length $\ell = 32, 64, 128, 256, 512$. The dotted pre-intervention target is shown for illustration only and is not part of the task.

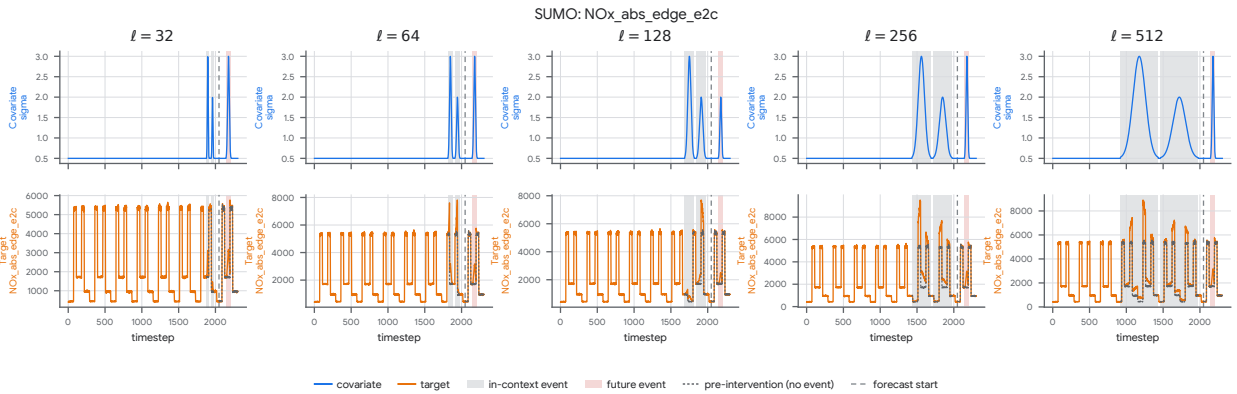


Figure 36 | Tasks that differ only in event length $\ell = 32, 64, 128, 256, 512$. The dotted pre-intervention target is shown for illustration only and is not part of the task.

L.3. Number of events

Each figure varies the number of events E across columns, holding every other axis fixed. More events set the covariate at more points across the history, giving more views of the covariate–target relationship.

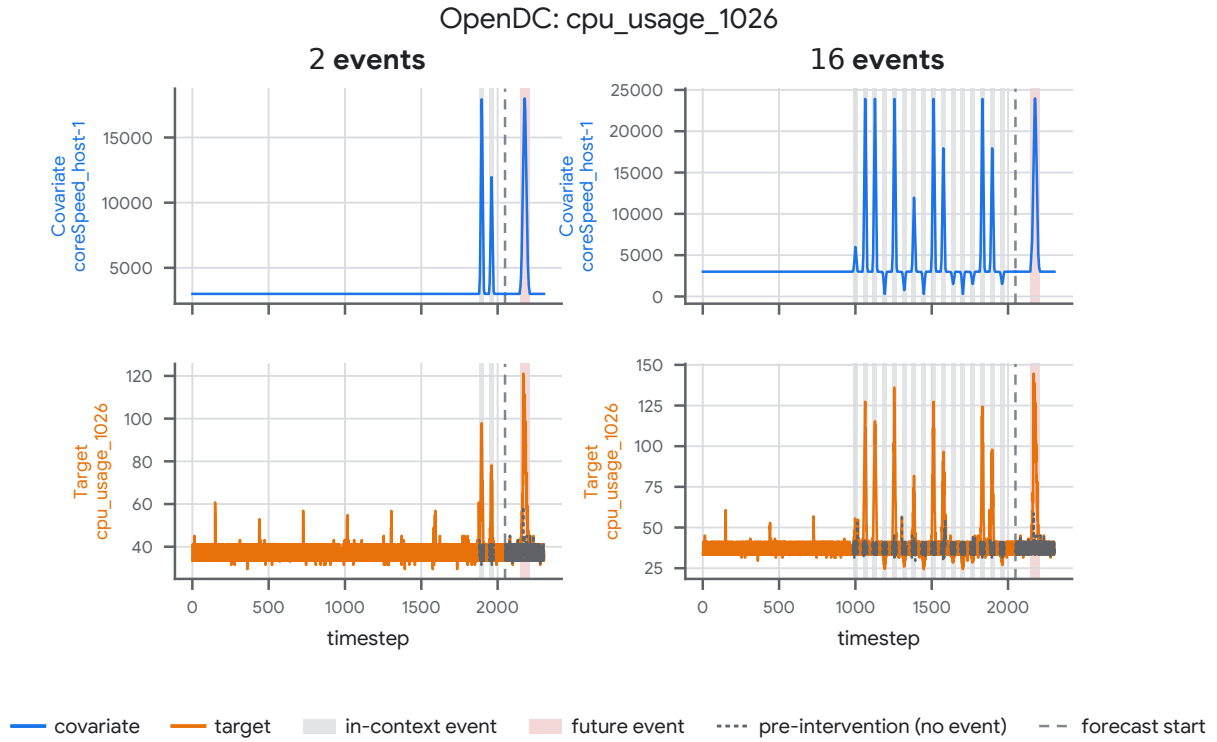


Figure 37 | Tasks that differ only in the number of events (few versus many). The dotted pre-intervention target is shown for illustration only and is not part of the task.

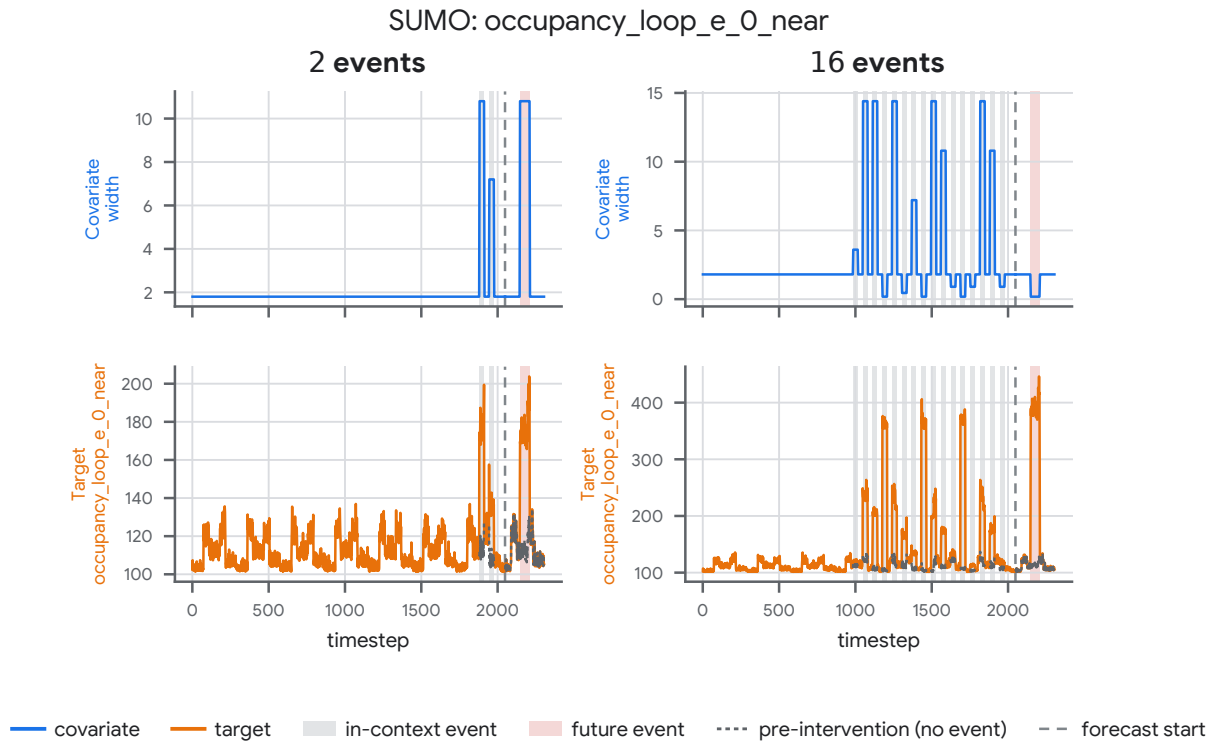


Figure 38 | Tasks that differ only in the number of events (few versus many). The dotted pre-intervention target is shown for illustration only and is not part of the task.

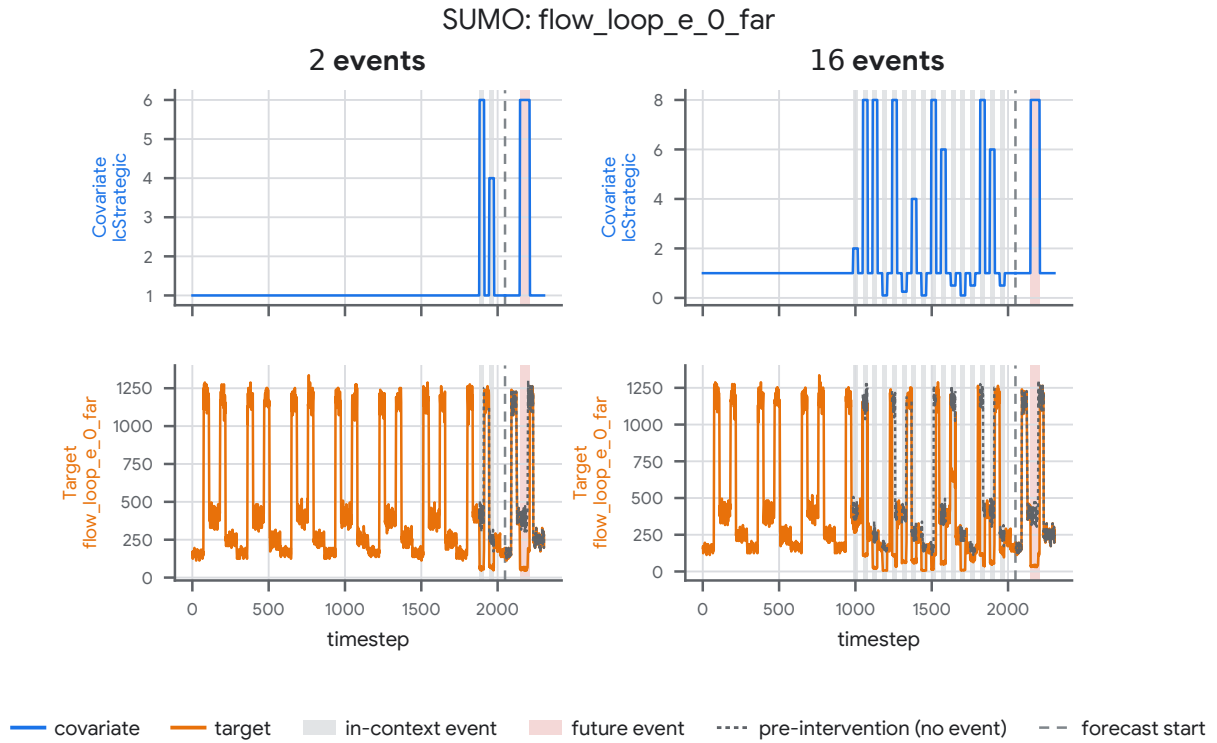


Figure 39 | Tasks that differ only in the number of events (few versus many). The dotted pre-intervention target is shown for illustration only and is not part of the task.

L.4. Context and prediction length

Each figure varies the context and prediction length across columns, holding every other axis fixed. Each column is labelled with its context length (CTX) and prediction length (PRED).

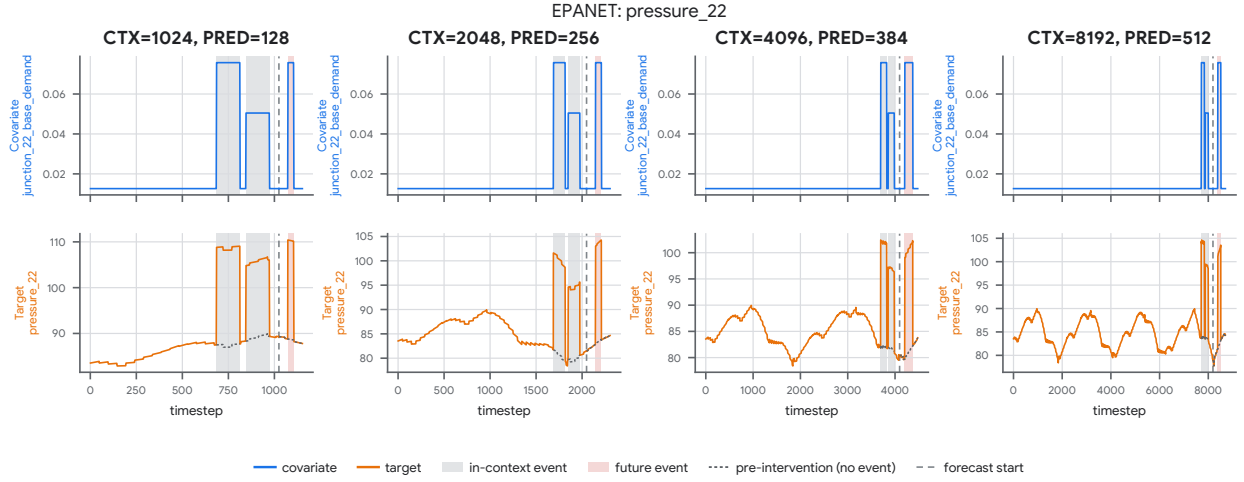


Figure 40 | Tasks that differ only in context/prediction pair, C_1 through C_4 . The dotted pre-intervention target is shown for illustration only and is not part of the task.

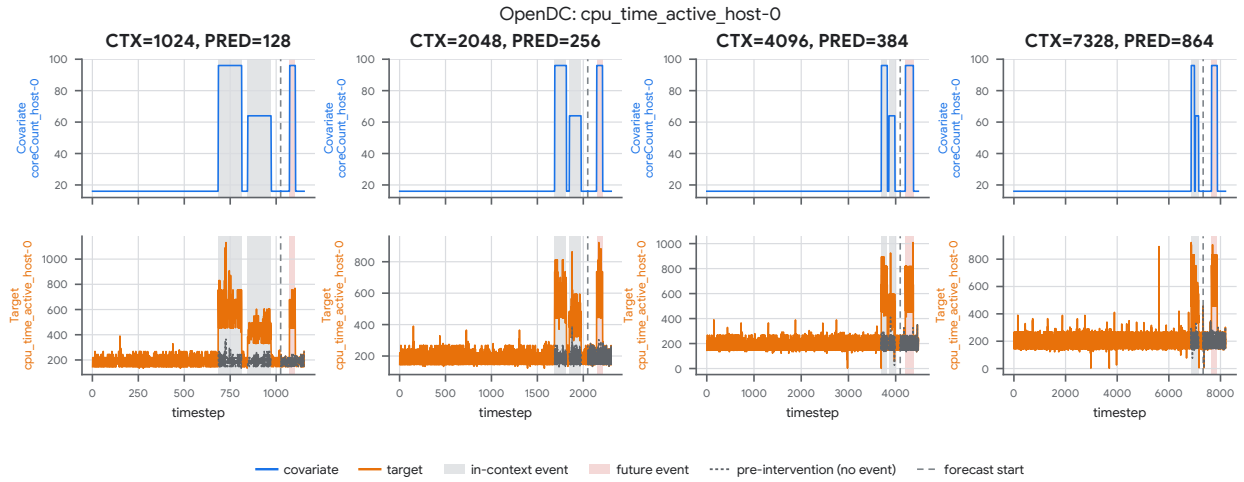


Figure 41 | Tasks that differ only in context/prediction pair, C_1 through C_4 . The dotted pre-intervention target is shown for illustration only and is not part of the task.

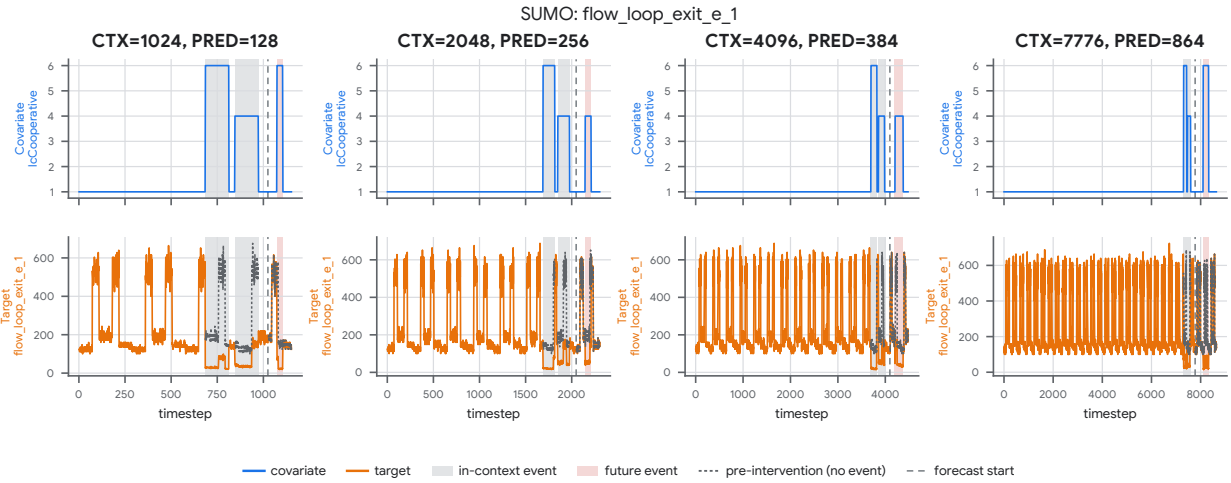


Figure 42 | Tasks that differ only in context/prediction pair, C_1 through C_4 . The dotted pre-intervention target is shown for illustration only and is not part of the task.

L.5. Number of covariates

Each figure varies the number of covariates K across columns, holding every other axis fixed. As K grows, each event acts on more covariates at once, and the model must learn more relationships from the history. Here every covariate is plotted above the shared target.

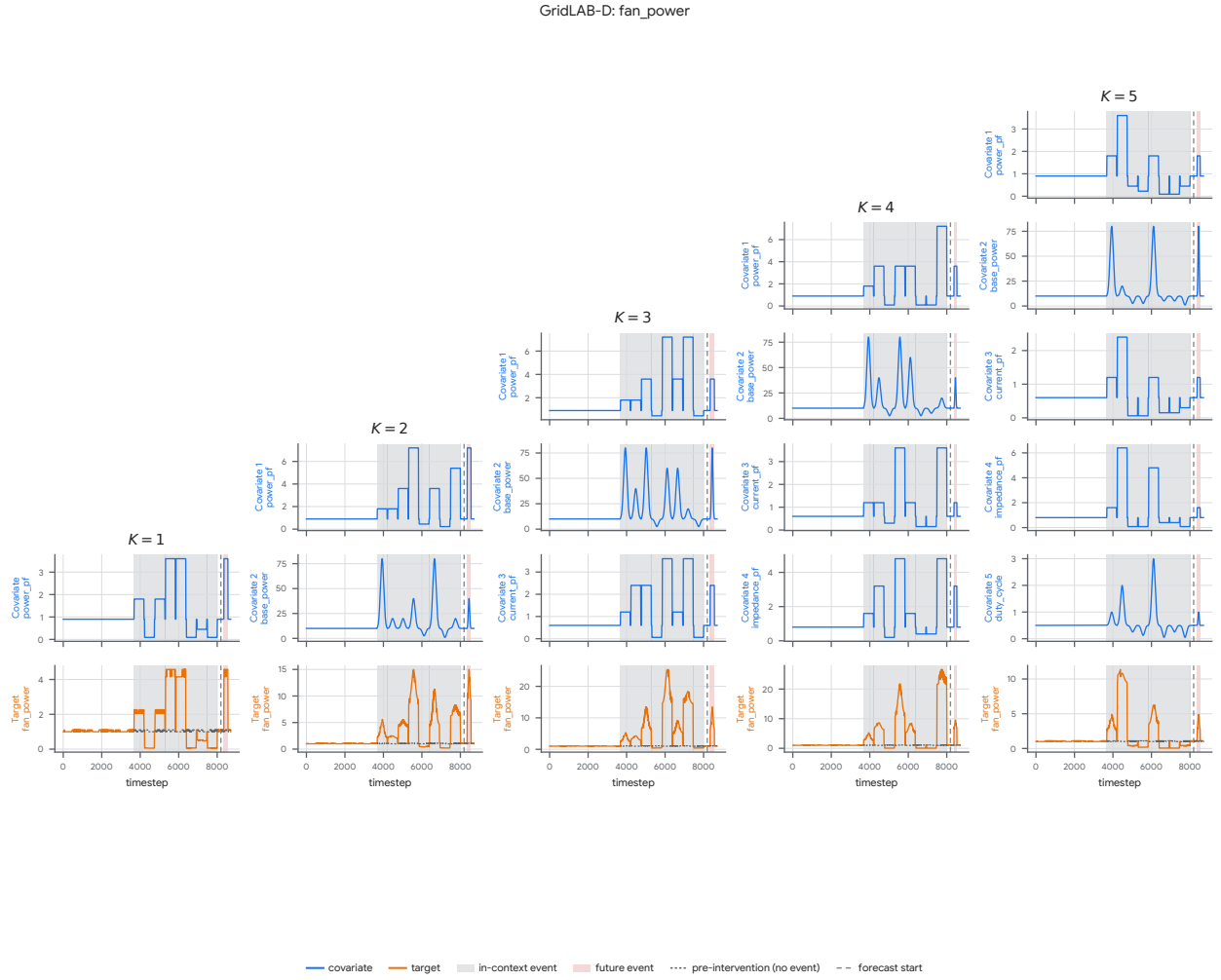


Figure 43 | Tasks that differ only in the number of covariates, $K = 1$ through 5. The dotted pre-intervention target is shown for illustration only and is not part of the task.

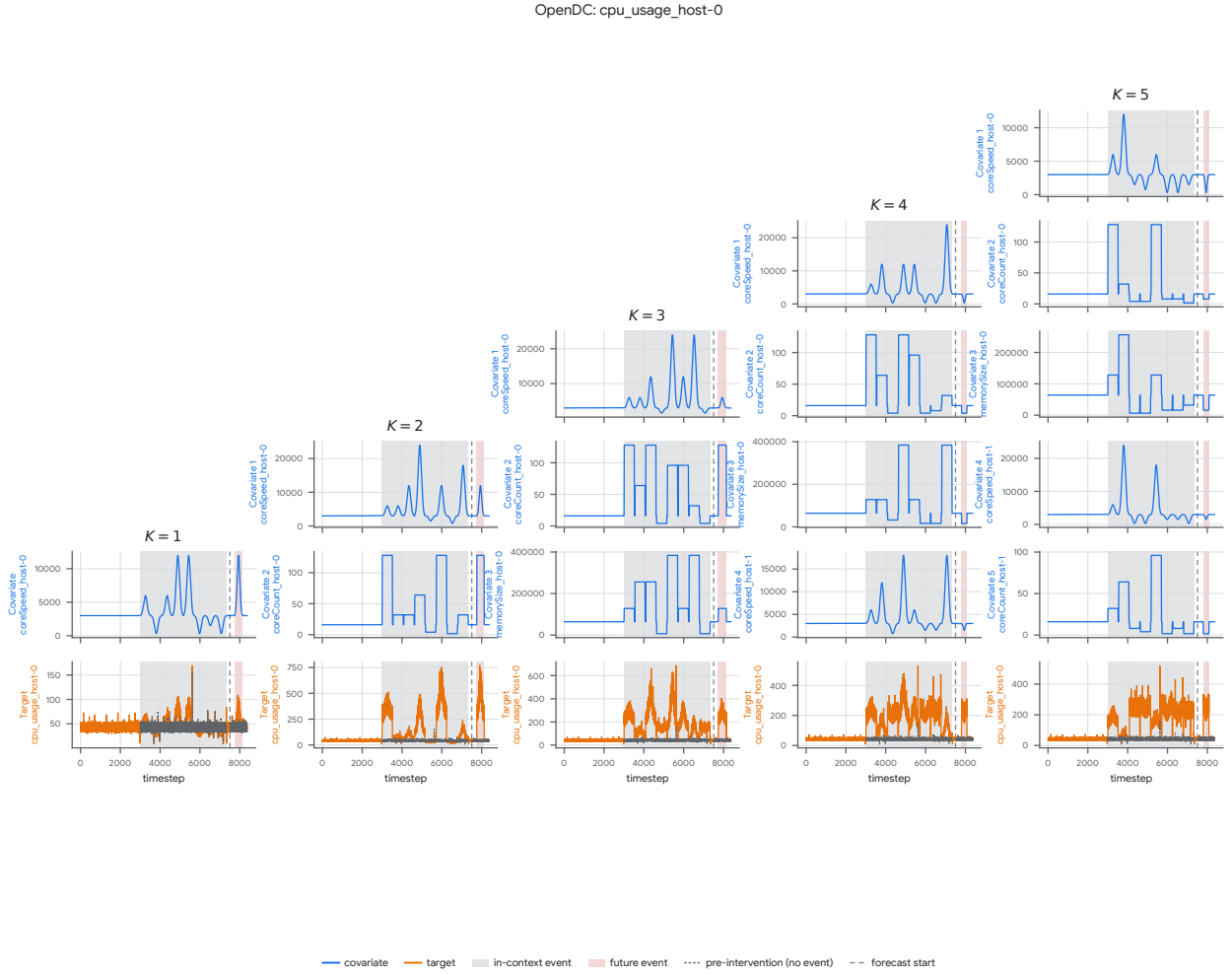


Figure 44 | Tasks that differ only in the number of covariates, $K = 1$ through 5. The dotted pre-intervention target is shown for illustration only and is not part of the task.