UGAPSYN: UTILITY-GAP SYNTHESIS FOR ADAPTING FINANCIAL TIME-SERIES FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Synthetic price paths can supplement the limited market history used to adapt time-series foundation models to financial forecasting, and generators are usually judged by how realistic these paths are. Under a fixed adaptation budget, however, each synthetic window replaces a real one, so even a realistic path can hurt the adapted model if the displaced real window was more useful. To capture this trade-off, we define the utility gap as the reduction in forecasting risk relative to the same adaptation without replacement. Because this gap depends on the model being adapted and on the real windows being replaced, it must be measured by adapting the same model with and without the replacement. Utility-Gap Synthesis (UGapSyn) makes this comparison by continuing adaptation from a shared state with synthetic or real windows at the replaced positions, and selects the configuration of a controllable GARCH-jump generator by the gap measured on validation data. During adaptation, it adjusts the amount and kind of synthetic data only when repeated measurements on historical data agree. With three foundation models on two US equity panels and one Chinese panel, UGapSyn achieves lower mean test MSE than real-data adaptation in all nine settings and than all five synthesis and condensation baselines on both US panels for every model. On Chronos-Bolt-Base, it lowers MSE relative to real-data adaptation by 2.4% on US_71 and 2.7% on US_14L.

1 INTRODUCTION

Time-series foundation models learn temporal patterns from large datasets and transfer them to new forecasting tasks (Ansari et al., 2024; Das et al., 2023). In finance, changing market conditions can reduce the relevance of patterns learned from earlier data (Zhu et al., 2025; Rahimikia et al., 2025), so pretrained models are often adapted to a target market through fine-tuning, continual pretraining, adapters or refinement modules (Fu et al., 2025; Chitsaz & Haratizadeh, 2025; Bolton et al., 2026). These methods change how the model is updated, but the adaptation examples still come from a finite market history. Recent observations cover only a limited range of price dynamics, while extending the history can introduce regimes that are less aligned with the target period (Taga et al., 2026). Financial generators offer a way to expose the model to additional realizations of plausible price dynamics (Huang et al., 2024; Zhang et al., 2026).

We study how to decide which synthetic data are worth using when adapting a pretrained financial forecaster. Under a fixed adaptation budget, each generated window replaces a real training window, so a statistically plausible path can still reduce forecasting performance if the displaced real example would have been more useful. We make this trade-off explicit through *replacement utility*, defined as the reduction in forecasting risk relative to the same adaptation on the displaced real windows (Figure 1).

Existing financial generators are commonly evaluated by statistical fidelity and downstream forecasting (Huang et al., 2024). Fidelity measures how well generated data reproduce observed patterns, but it does not determine whether replacing real windows with them improves a particular adapted forecaster. In our experiments, fidelity scores that track utility on the US panels show no consistent relation to it on CSI300 (Section 4.3). Model-dependent methods use learning performance to value or select data, and recent work also adapts financial synthesis to market drift (Ghorbani & Zou, 2019; Fan et al., 2026; Taga et al., 2026; Xia et al., 2026). For fixed-budget adaptation, generated

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

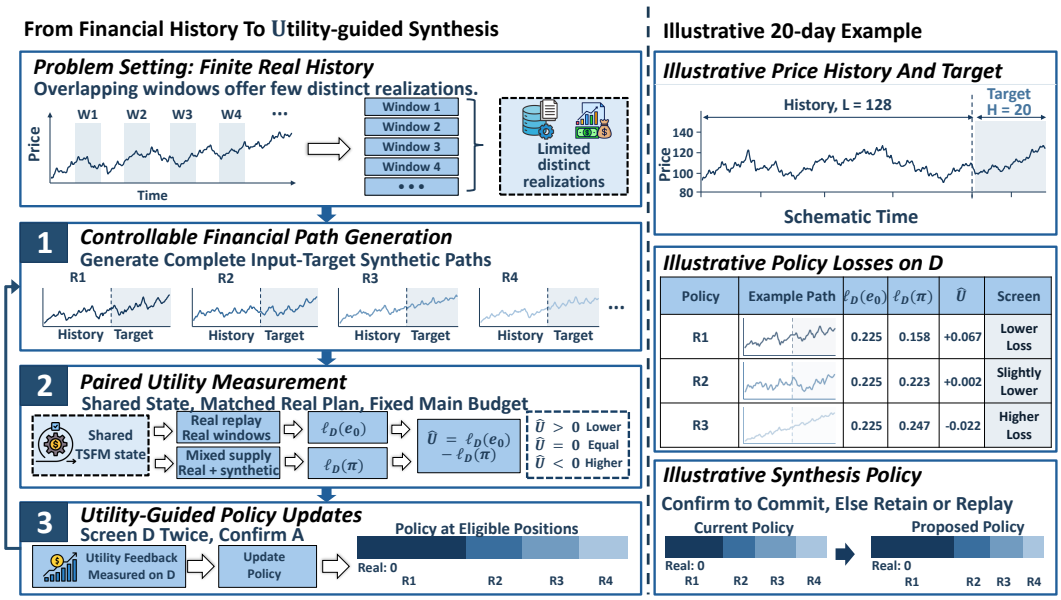

Figure 1: Synthetic windows replace real ones under a fixed budget. UGapSyn measures their contribution through paired continuations from the same adaptation state and selects the synthetic-data configuration.

data must be evaluated against the real data they displace under matched training conditions, before the market period in which the choice is evaluated.

1.1 Challenges

Turning replacement utility into reliable synthesis decisions raises two difficulties.

(1) Synthetic-data utility depends on the adaptation state and the real data it replaces. Real and synthetic windows jointly update the same model, so the effect of a candidate depends on the starting model and the real-data alternative. Comparing runs under different training conditions mixes this effect with unrelated training variation, which can obscure small differences (Bouthillier et al., 2021). Reliable feedback requires paired continuations that isolate the replacement effect.

(2) Historical utility may not transfer to later periods. Even a paired comparison measures utility on historical observations, and market changes can weaken its relation to the benefit in a later period (Yao et al., 2022; Zhu et al., 2025). Repeating the comparison reveals measurement variability but does not establish temporal transfer. Historical feedback should therefore support policy changes only when the evidence is consistent, and its ability to predict later gains should be evaluated separately.

1.2 Contributions

This paper makes three contributions.

(1) Utility-gap synthesis. We introduce UGapSyn, short for Utility-Gap Synthesis (Figure 2). It generates synthetic windows from a controllable GARCH-jump process and selects the configuration with the largest utility gap on validation data, the reduction in forecasting risk compared with adaptation that keeps the displaced real windows. During adaptation, it revises the region weights and replacement level only when repeated screening and a later confirmation agree, so a single noisy comparison cannot change the validated configuration (Section 3).

(2) Replacement utility and paired measurement. We formulate fixed-budget synthetic adaptation as a replacement problem and formalize the utility gap as replacement utility. Paired continuations from the same adaptation state estimate this utility and differ only in the synthesis policy, separating its effect from unrelated training variation (Sections 2 and 3.2).

(3) Evidence across markets, backbones and periods. Across three US and Chinese equity datasets and three pretrained backbones, UGapSyn achieves lower mean test MSE than real-data adaptation in all nine settings and the lowest mean among seven adapted methods on both US datasets with every backbone. On Chronos-Bolt-Base, its MSE is 2.4% lower than that of real-data adaptation on US_71 and 2.7% lower on US_14L, with ten-seed paired intervals excluding zero, and its mean error stays lower in a later period. Most of the gain comes from the validation-selected synthetic data. Because historical utility transfers weakly, runtime revision adds little in the main settings and reduces mean error in two of three harmful initial configurations (Section 4).

Related work. Appendix A positions UGapSyn against financial foundation-model adaptation, synthetic data for forecasting and model-dependent data selection.

2 Problem Definition

Given a real training set $\mathcal{R}$, a pretrained forecaster f_{θ_0}, historical feedback data $\mathcal{H}$, and a fixed adaptation budget E, the goal is to choose a synthetic-data configuration and replacement level that minimize forecasting risk after adaptation. All decisions use only information available before the evaluation period.

For asset a with observed price $p_{a,t} > 0$, a forecasting example is

$$x_{a,t} = p_{a,t-L+1:t}, \qquad y_{a,t} = p_{a,t+1:t+H}, \tag{1}$$

where L is the input length and H is the forecast horizon. Both parts of every example in $\mathcal{R}$ lie within the training period. The feedback data $\mathcal{H}$ are observed after this period and before evaluation, and are used to compare candidate synthesis decisions. Let $\mathcal{L}(f) = \mathbb{E}_{(x,y)\sim P_{\text{future}}}[\ell(f(x), y)]$ denote forecasting risk under the future evaluation distribution, where ℓ is a fixed prediction loss.

Definition 2.1 (Synthesis policy)**.** Let G_ϕ generate a price path of length $L+H$, split into a training pair $(\tilde{x}, \tilde{y})$, with admissible parameters $\phi \in \Phi$ determined from training data and process constraints. Let $T_{\text{real}} = (z_1, \ldots, z_N)$ be the ordered real-data sequence used under budget E, and let $\mathcal{I} \subseteq \{1, \ldots, N\}$ contain the positions eligible for replacement. A policy π specifies a replacement decision $c_i \in \{0, 1\}$ and, when $c_i = 1$, a distribution for sampling $\phi_i \in \Phi$. It induces $T_\pi = (z_1^\pi, \ldots, z_N^\pi)$ with

$$z_i^\pi = \begin{cases} z_i, & c_i = 0, \\ (\tilde{x}_i, \tilde{y}_i), & c_i = 1. \end{cases} \tag{2}$$

Each replacement pair is generated by G_{ϕ_i}, and $c_i = 0$ for $i \notin \mathcal{I}$. The realized replacement proportion is $N^{-1}\sum_{i=1}^N c_i$. The policy may use historical feedback and the current adaptation state, but cannot access evaluation data. UGapSyn considers a structured subclass parameterized by the generation configuration and replacement level.

Definition 2.2 (Replacement utility)**.** Let $\mathcal{A}(\theta_0, T, E)$ denote the model obtained by adapting from θ_0 on sequence T under budget E. The replacement utility, or utility gap, of policy π is

$$U(T_\pi; T_{\text{real}}, \theta_0, E) = \mathcal{L}(\mathcal{A}(\theta_0, T_{\text{real}}, E)) - \mathcal{L}(\mathcal{A}(\theta_0, T_\pi, E)) \,. \tag{3}$$

The two adaptations start from the same model state and use the same update budget and training schedule, differing only at the designated replacement positions. Positive utility means that the generated examples reduce future forecasting risk relative to the real examples they replace.

Definition 2.3 (Utility-guided synthesis)**.** Let $\Pi_{\mathcal{H}}$ be the class of admissible synthesis policies that use only training data, historical feedback $\mathcal{H}$, and adaptation states. The population objective is

$$\begin{aligned} \max_{\pi\in\Pi_{\mathcal{H}}} \quad & \mathbb{E}_\xi[U(T_\pi; T_{\text{real}}, \theta_0, E)] \\ \text{s.t.} \quad & c_i = 0 \quad \text{for } i \notin \mathcal{I}, \\ & \phi_i \in \Phi \quad \text{whenever } c_i = 1, \\ & |T_\pi| = |T_{\text{real}}| = N. \end{aligned} \tag{4}$$

The expectation is over generation and adaptation randomness ξ, conditional on the observed training and feedback data. With the real-data reference fixed, maximizing expected replacement utility is equivalent to minimizing expected forecasting risk after adaptation. Because the future evaluation distribution is unavailable when the policy is selected, UGapSyn estimates candidate utility on $\mathcal{H}$ and evaluates its temporal transfer separately. The main budget E is fixed across policies, while computation used for policy evaluation is reported separately.

3 THE UGAPSYN METHOD

UGapSyn evaluates synthetic data by their effect on adaptation. Figure 2 organizes the method into three stages. A controllable financial process defines the candidate synthetic data, paired continuations estimate their replacement utility, and utility feedback selects and revises the synthesis policy. Validation uses this utility to fix the configuration before adaptation. During adaptation, historical screening and confirmation act as a conservative safeguard and revise the policy only when the evidence consistently supports a change (Algorithm 1).

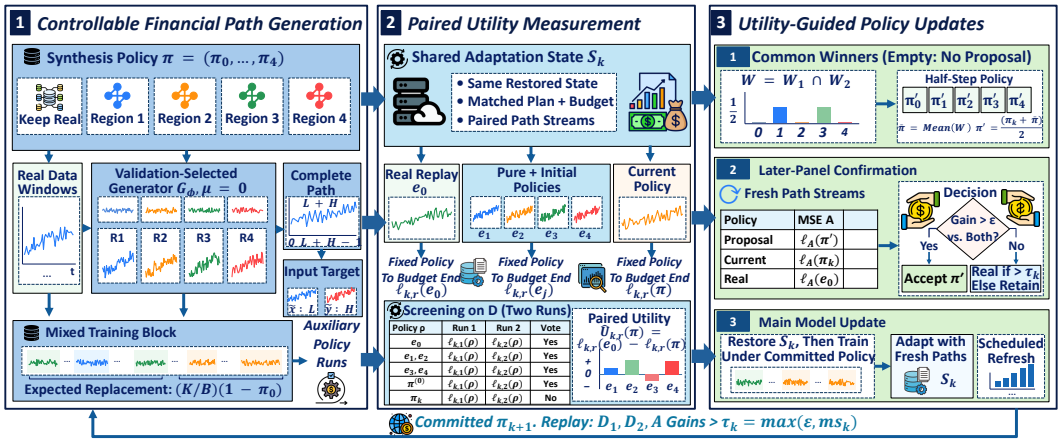

Figure 2: The UGapSyn framework: controllable generation, paired measurement of replacement utility from a common state, and utility-guided selection and conservative revision of the synthesis policy.

The feedback data $\mathcal{H}$ of Section 2 consist of validation data D_{val} and two calibration panels D and A that lie between the training and validation periods. This is an offline adaptation setting: all feedback labels are available before main adaptation begins. D_{val} selects the configuration before adaptation. During adaptation, D provides screening feedback and the later panel A provides confirmation.

3.1 CONTROLLABLE FINANCIAL PATH GENERATION

The generator defines a controllable synthetic candidate space. Stage 1 of Figure 2 turns the policy in Section 2 into synthetic training examples. We use GARCH-type conditional variance and a jump component to represent changing volatility and abrupt price movements (Bollerslev, 1986; Merton, 1976):

$$r_t = \mu + \sqrt{h_t} z_t + J_t, \qquad h_{t+1} = \omega + \alpha r_t^2 + \beta h_t, \tag{5}$$

where z_t is a standardized Student-t innovation and J_t is a Gaussian jump activated by a Bernoulli draw. The parameters control drift, variance dynamics, tail weight and jumps. Each path starts from an anchor, the first price of a uniformly drawn training window, and is then split into an input and target. Each backbone normalizes a context by its own location and scale and applies the same transform to the target, so the anchor sets the price level without changing the normalized training pair. A generated path carries no asset identity and takes the position of the real window it replaces.

Training-window return statistics determine proposal ranges for drift, variance scale, tail weight and persistence. We partition variance scale and persistence at their midpoints, producing four coarse regions $\Phi_1, \ldots, \Phi_4$ while keeping the candidate space small. Generated paths must be finite, positive and distinct from real windows and other synthetic paths in the same block. Rejected paths are resampled within the same region (Appendix E).

For each block of B training positions, a fixed real-data plan marks K positions as eligible for replacement. At an eligible position, the action probabilities $\pi = (\pi_0, \ldots, \pi_4)$ sum to one. Action zero retains the real window, while action $j > 0$ generates a path from region Φ_j using proposal distribution q_j. The expected replacement proportion is $\frac{K}{B}(1 - \pi_0)$, and conditional on generation,

$$q_\pi = \frac{\sum_{j=1}^{4} \pi_j q_j}{1 - \pi_0}, \qquad \pi_0 < 1. \tag{6}$$

Algorithm 1 Utility-guided synthesis and adaptation

Require: θ_0, real windows $\mathcal{R}$, validation data D_{val}, calibration panels D, A, budget E, refresh schedule

$(G, m, C_{\mathrm{aux}}) \leftarrow \mathrm{ValidationSelect}(\theta_0, \mathcal{R}, D_{\mathrm{val}})$ ▷ Appendix B.3
$\mathsf{S} \leftarrow \mathrm{RealWarmup}(\theta_0, \mathcal{R}), \quad \pi \leftarrow \pi^{(0)} = (0, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$
$\mathcal{V} \leftarrow \{e_0, \ldots, e_4, \pi^{(0)}\}, \quad \eta \leftarrow \frac{1}{2}, \quad \mathcal{J} \leftarrow [\,]$
for each scheduled refresh k **do**
  $\mathsf{S}_k \leftarrow \mathsf{S}, \quad \pi_k \leftarrow \pi, \quad C_k \leftarrow \mathcal{V} \cup \{\pi_k\}$
  for $r \in \{1, 2\}$ **do**
    $(\ell_{k,r}, c) \leftarrow \mathrm{Probe}(\mathsf{S}_k, C_k, E_k, D, r), \quad C_{\mathrm{aux}} \leftarrow C_{\mathrm{aux}} + c$
    $b_r \leftarrow \min_{\rho \in \mathcal{V}} \ell_{k,r}(\rho), \quad W_r \leftarrow \{\rho \in \mathcal{V} : \ell_{k,r}(\rho) \le b_r + \varepsilon(b_r)\}$
  end for
  $W \leftarrow W_1 \cap W_2, \quad \pi' \leftarrow \pi_k$
  if $W \neq \varnothing$ **then**
    $\bar{\pi} \leftarrow |W|^{-1} \sum_{\rho \in W} \rho, \quad \pi' \leftarrow (1 - \eta)\pi_k + \eta\bar{\pi}$
  end if
  $(a_k, c) \leftarrow \mathrm{Probe}(\mathsf{S}_k, \{\pi', \pi_k, e_0\}, E_k, A, 3), \quad C_{\mathrm{aux}} \leftarrow C_{\mathrm{aux}} + c$
  $\widehat{U}_{k,r} \leftarrow$ Eq. 7, $\quad s_k \leftarrow$ Eq. 8, $\quad \tau_k \leftarrow$ Eq. 10
  $g_k \leftarrow \bigwedge_{r=1}^{2} [\ell_{k,r}(\pi_k) - \ell_{k,r}(e_0) > \varepsilon(\ell_{k,r}(e_0))]$
  if $\pi' \notin \{\pi_k, e_0\}$ and $a_k(\pi') + \varepsilon(a_k(\pi')) < \min\{a_k(\pi_k), a_k(e_0)\}$ **then**
    $\pi \leftarrow \pi'$ ▷ accept the proposal
  else if g_k and $a_k(\pi_k) - a_k(e_0) > \tau_k$ **then**
    $\pi \leftarrow e_0$ ▷ return to real data
  else
    $\pi \leftarrow \pi_k$ ▷ retain the current policy
  end if
  $\mathcal{J} \leftarrow \mathcal{J} \,\|\, (\pi_k, \pi, \widehat{U}_{k,1}, \widehat{U}_{k,2}, a_k, s_k, \tau_k)$
  $\mathsf{S} \leftarrow \mathrm{Advance}(\mathsf{S}_k, \pi, G, \Delta_k)$ ▷ main adaptation only
end for
return adapted forecaster f_{S}, decision log $\mathcal{J}$, main budget E, auxiliary cost C_{aux}

Thus, π jointly controls the synthetic-path distribution and the amount of real data replaced.

3.2 Paired Measurement of Replacement Utility

Paired continuations isolate the effect of the synthesis policy. Stage 2 of Figure 2 measures this effect on the adapted forecaster. At refresh k, the state S_k contains the model parameters, optimizer state, buffers, module modes and random states needed to reproduce the continuation. Each candidate is held fixed while adaptation continues from S_k through the remaining budget E_k. Candidates share the same real-data plan, replacement positions and batch order. Within a repetition, each eligible position has separate random streams for the action, parameter proposal, anchor and innovations. Candidate policies share the corresponding primitive draws, while a rejected path affects only its own position (Appendix D).

Let $\ell_{k,r}(\rho)$ be the screening mean squared error on panel D after continuing under policy ρ in repetition r. With e_0 denoting the all-real policy, the measured replacement utility is

$$\widehat{U}_{k,r}(\rho) = \ell_{k,r}(e_0) - \ell_{k,r}(\rho). \tag{7}$$

This is the historical proxy for the future utility in Eq. 3. Each refresh evaluates every distinct voting policy and the current policy in two matched repetitions, after which all exploratory states are discarded.

What a continuation score estimates. A continuation score is the historical loss obtained by committing to ρ from the current state for the remaining schedule. It is a local decision score for the current refresh because the controller may revise the policy later. With two refreshes, this distinction affects only the first decision.

The repetitions also provide a scale for measurement disagreement. Let $d_k(\rho) = |\ell_{k,1}(\rho) - \ell_{k,2}(\rho)|$. We define

$$s_k = \mathrm{median}\big(d_k(e_1), \ldots, d_k(e_4), d_k(\pi^{(0)}), d_k(\pi_k)\big). \tag{8}$$

Because real replay contains no generated paths, $\ell_{k,1}(e_0) = \ell_{k,2}(e_0)$, so $d_k(\rho)$ also measures disagreement between repeated paired-utility estimates. The quantity s_k only scales the retreat margin below and is not a calibrated statistical test.

3.3 Utility-Guided Policy Updates

Validation fixes the generation configuration, and runtime feedback revises only the region weights and replacement level. Before main adaptation, paired validation comparisons select the synthetic-data configuration and retreat rule (Appendix B.3). Stage 3 of Figure 2 then uses the calibration panels to decide whether the current policy π should be revised.

Screening and proposal. Let e_j denote the pure policy for action j, and initialize $\pi^{(0)} = (0, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$. The voting set is $\mathcal{V} = \{e_0, \ldots, e_4, \pi^{(0)}\}$. In each repetition r, W_r contains the voting policies with minimum screening loss up to numerical tolerance. A proposal is formed only from policies supported by both repetitions. For $W = W_1 \cap W_2 \neq \varnothing$,

$$\bar{\pi} = \frac{1}{|W|} \sum_{\rho \in W} \rho, \qquad \pi' = (1 - \eta)\pi_k + \eta\bar{\pi}, \tag{9}$$

where $\eta = \frac{1}{2}$. If W is empty, no proposal is formed. Each accepted proposal therefore moves at most half of the policy mass. A larger action set was evaluated during validation but not selected (Appendix B.3).

Confirmation and commitment. Before changing the main trajectory, the proposal is compared with the current policy and e_0 on panel A. All candidates continue from the same state using a fresh matched stream. Let $a_k(\rho)$ denote the confirmation loss and $\varepsilon(x) = 10^{-12}\max(1, |x|)$ the numerical tolerance. A proposal distinct from both references is accepted only if its confirmation loss is lower than both by more than numerical tolerance.

If no proposal is accepted, the method can return to real data only when real replay improves on the current policy in both screening repetitions and its confirmation gain exceeds

$$\tau_k = \max\big(\varepsilon(a_k(e_0)), m s_k\big), \tag{10}$$

where m is selected on validation data. Otherwise the current policy is retained.

Conservative retreat. A proposal changes at most half of the policy mass, while a retreat moves all of it to real replay, so m governs only this stronger intervention. Setting $m = 0$ applies the proposal standard to retreat, $m = 1$ requires a gain beyond the repetition disagreement, and $m = \infty$ disables retreat. Validation selects $m = \infty$ for the main configuration. Retreat-enabled variants are evaluated in the decision audit and on weaker initial configurations (Sections 4.4 and 4.5).

After each decision, UGapSyn restores S_k and resumes the main trajectory under the committed policy with fresh paths, counting candidate continuations only in the auxiliary cost C_{aux}. In Algorithm 1, $\mathrm{Probe}(\mathsf{S}, C, E', P, r)$ evaluates candidate policies from a common state over budget E' and restores that state, and $\mathrm{Advance}$ performs the next Δ_k main updates (Appendix D).

4 Experiments

We compare forecasting performance under a fixed main adaptation budget, separate the contributions of the selected synthetic-data configuration and runtime revision, and test whether historical feedback reflects later utility.

4.1 Experimental Setup

Data and baselines. We evaluate on three financial panels from two equity markets. US_71 and US_14L contain daily closing prices for 71 and 14 disjoint stocks from the supervised FinCast release (Zhu et al., 2025). CSI300 provides a point-in-time panel from the Chinese market. The context length is 128 days and the forecast horizon is 20 days. Generator parameters and asset scales are estimated from training data. Chronological splits appear in Appendix C.

Table 1: Main comparison. Mean test MSE. Chronos-Bolt-Base uses ten seeds for every adapted method, the runs behind Table 2. The other two backbones use three seeds. Zero-shot is deterministic. Bold marks the lowest mean and underlining the next distinct mean.

	Bolt-Base			Bolt-Small			TimesFM-2.5		
Method	US_71	US_14L	CSI300	US_71	US_14L	CSI300	US_71	US_14L	CSI300
Zero-shot	0.1940	1.1089	2.0022	0.1915	1.0690	2.0480	0.1973	1.1648	2.3463
Real adaptation	0.1877	1.1229	1.9575	<u>0.1828</u>	1.0684	1.9950	0.1954	1.1455	<u>2.0596</u>
FTS-Diffusion	0.1923	<u>1.0970</u>	1.9994	0.1909	1.0683	2.0320	0.1939	1.0924	2.1181
SDForger	0.2192	1.1152	2.0081	0.2203	1.0697	2.0523	0.2006	<u>1.0911</u>	2.0866
L2D-Diff	0.1976	1.1193	2.0879	0.1906	1.0709	2.4175	0.2009	1.0938	2.1553
Diffusion-TS	<u>0.1865</u>	1.1041	1.9609	0.1830	<u>1.0557</u>	1.9902	<u>0.1935</u>	1.0961	2.1070
MTT+CondTSF	0.1916	1.1195	**1.9365**	0.1877	1.0896	**1.9689**	0.2000	1.1246	2.1193
UGapSyn	**0.1832**	**1.0930**	<u>1.9536</u>	**0.1784**	**1.0462**	<u>1.9771</u>	**0.1860**	**1.0599**	**2.0466**

We use Chronos-Bolt-Base, Chronos-Bolt-Small (Ansari et al., 2024; Simeone, 2026) and TimesFM-2.5-200M (Das et al., 2023) as pretrained backbones. We compare UGapSyn with real-data adaptation and five synthesis or condensation methods, FTS-Diffusion, SDForger, L2D-Diff, Diffusion-TS and MTT+CondTSF (Huang et al., 2024; Rousseau et al., 2026; Shen et al., 2026; Yuan & Qiao, 2024; Cazenavette et al., 2022; Ding et al., 2024). Zero-shot prediction provides a reference without adaptation.

Matched adaptation and evaluation. For each backbone, adapted methods share pretrained weights, the ordered real-window plan and 52 main updates, and each 128-position block has the same 64 positions eligible for replacement. External methods fill all 64 positions from their pools, a 50% replacement level equal to that of the initial UGapSyn policy. These pools are fixed across seeds, while the parametric generator draws fresh paths. MSE averages squared errors of the direct model forecasts after scaling by each asset's training-price standard deviation, using common evaluation masks. Configurations are chosen on validation under recorded rules and frozen before the independent test and forward evaluations (Appendix B.3). For Chronos-Bolt-Base, the main table reports ten-seed means, and significance statements use seed-paired bootstrap intervals over the same seeds (Appendix B.1). The other two backbones use three seeds per panel.

4.2 Forecasting Performance

UGapSyn improves on real-data adaptation across all three backbones. UGapSyn has lower mean test MSE than real-data adaptation in all nine settings of Table 1 and the lowest mean among the adapted methods on both US panels with every backbone. The largest reduction occurs on US_14L with TimesFM-2.5, where MSE falls from 1.1455 to 1.0599. On CSI300, MTT+CondTSF gives the lowest mean with both Bolt backbones and UGapSyn is second, ahead of real-data adaptation and the other generators. With TimesFM-2.5, UGapSyn has the lowest mean on all three panels, while every external method is worse than real-data adaptation on CSI300.

The evaluation supports the US gains over real-data adaptation. Over ten seeds with Chronos-Bolt-Base, UGapSyn lowers MSE relative to real-data adaptation by 0.0045 [0.0025, 0.0066] on US_71 and 0.0300 [0.0229, 0.0372] on US_14L, with 95% seed-paired bootstrap intervals (Table 2). It is below every external method on US_71. On US_14L it is below all except FTS-Diffusion, for which the paired interval includes zero. On CSI300, UGapSyn is below FTS-Diffusion and SDForger by 0.0458 and 0.0545, while its 0.0039 gain over real-data adaptation is not significant and MTT+CondTSF is lower by 0.0171. Seed standard deviations of UGapSyn are 0.0026, 0.0173 and 0.0178.

On the independent later US period, UGapSyn has lower mean error than real-data adaptation and all five external methods on both panels (Figure 3(f)). Table 3(b) reports the seed standard deviations. The fixed zero-drift configuration matches UGapSyn on US_71 and the fixed sampled-drift configuration is lower on US_14L, so the mean advantage in the later period holds without runtime revision.

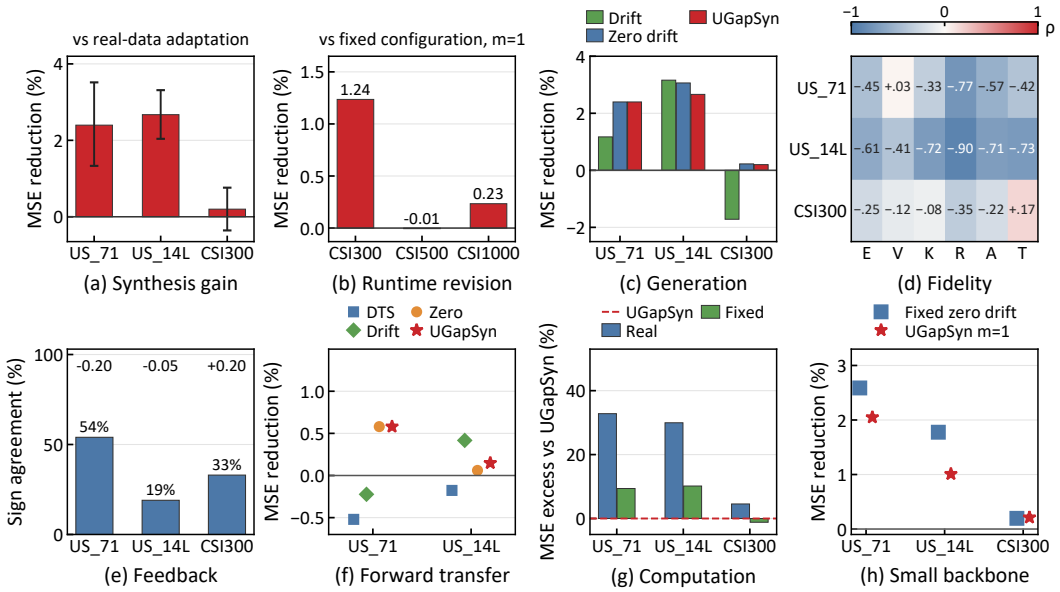

Figure 3: Synthesis and adaptation controls. (a,c) Ten-seed MSE reductions against real-data adaptation on Chronos-Bolt-Base (Tables 1 and 3). Error bars in (a) are the 95% seed-paired bootstrap intervals of Table 2, relative to the real-data mean. (b) Runtime-revision gains over fixed configurations for $m = 1$, using 10, 5 and 10 seeds. (d) Rank correlations between fidelity scores and utility across the eleven synthetic sources of each panel. E, V, K, R, A and T denote energy distance, volatility, kurtosis, return ACF, volatility ACF and tail scores. (e) Sign agreement between historical and test utility, with rank correlations above the bars. (f) Ten-seed forward gains. DTS denotes Diffusion-TS. (g) Excess MSE against UGapSyn at similar total computation. (h) Ten-seed gains for the $m = 1$ control on Chronos-Bolt-Small. The other panels omit uncertainty intervals. Appendices B.1, B.2, B.4 and B.6 give the full comparisons.

Allocation of computation. The gains over real-data adaptation do not require UGapSyn's 1,039 auxiliary updates, which serve only runtime revision, because the fixed zero-drift configuration reaches equal or lower mean MSE with the 52 main updates alone (Section 4.3). Giving real-data and fixed-configuration training a similar total of 1,092 main updates makes both worse on the US panels, while longer fixed-configuration training is better on CSI300 (Figure 3(g), Appendix B.4).

4.3 Generation Configuration and Runtime Updates

The validation-selected configuration accounts for the main gain. The primary control keeps the selected zero-drift configuration and fixes its initial policy, removing runtime revision and nothing else. Over ten paired seeds, UGapSyn minus this control is 0.0000 on US_71, 0.0045 [0.0000, 0.0098] on US_14L and 0.0005 [−0.0008, 0.0024] on CSI300 (Table 2). Runtime revision therefore adds no clear gain in these settings, with the fixed configuration slightly ahead on US_14L, and the improvement over real-data adaptation is primarily explained by the configuration selected on validation. Relative to UGapSyn, sampled drift raises MSE by 0.0375 on CSI300 and lowers it by 0.0055 on US_14L (Table 3(a)).

Other runtime rules lose part of the gain on the US panels. Setting $m = 0$ and binary voting, which restricts the vote to real replay and the initial mixture, both raise US_14L MSE from 1.0930 to 1.1030. Expanded actions, which add the external generator pools as pure policies, raise the mean on every panel (Table 3(a)), and on the ten-seed Small control the $m = 1$ rule improves on the fixed configuration only on CSI300 (Figure 3(h)).

Fidelity and utility. Smaller fidelity distances generally accompany larger replacement utility on the US panels, but correlations on CSI300 range from -0.35 to $+0.17$ (Figure 3(d), Appendix B.5). Even on US_71, SDForger falls within the energy-distance range of the parametric sources but has the lowest utility. In this diagnostic, no fidelity metric provides a consistent utility ranking across all three panels, supporting direct evaluation through the adapted forecaster.

4.4 Reliability of Historical Feedback

Repeatability and temporal transfer impose different requirements. The coupling experiment varies which training conditions candidate continuations share (Table 7). Repetition pairs select the same winner 36% of the time with paired continuations, compared with 14% with independent continuations. The same training conditions with private generation streams give 44% agreement. Sharing the real-data plan, eligible positions and batch order therefore improves repeatability in this diagnostic, while matched generation randomness adds no further gain.

Agreement across repetitions does not ensure agreement across periods. After freezing the method, we audit completed candidate branches on test data without changing any policy, threshold or reported trajectory. Every audited candidate improves on real replay in the test period, while the screening panel judges 46%, 81% and 67% of them worse (Table 9). Historical and test utilities thus agree in sign only for the remaining 54%, 19% and 33%, with rank correlations of -0.20, -0.05 and 0.20 (Figure 3(e)). Utility measured on this panel, which directly follows the training period, systematically understates the later value of synthetic data.

Across 26 audited refreshes with retreat enabled, mean test regret is 0.0005 for retaining the current policy, 0.0003 for accepting a proposal and 0.0335 for returning to real data. The last mean covers only two retreats and shows the possible cost of an unnecessary retreat, which agrees with validation disabling retreat in the main configuration.

4.5 Runtime Control under Harmful Initial Configurations

Runtime revision recovers part of the loss from a harmful initial configuration. Figure 3(b) evaluates the $m = 1$ rule when the initial configuration is weaker than in the main ablation. CSI500 and CSI1000 were registered before construction and did not inform any configuration choice, and every setting uses ten seeds except CSI500 with five. On CSI300, sampled drift raises fixed-configuration MSE to 1.9911, above real-data adaptation at 1.9575, and runtime revision reduces it to 1.9665, recovering 0.0246 of the 0.0336 loss. On CSI1000, the rule lowers mean MSE from 1.3671 to 1.3639, slightly below real-data adaptation at 1.3646. On CSI500, the measured advantage of real replay remains below the confirmation threshold, and the retained configuration gives 1.8006 against 1.7802 for real-data adaptation. The margin can therefore block changes driven by small historical differences, but may also delay removal of an unhelpful configuration (Appendix B.4).

5 Discussion

The evidence supports utility-guided selection more strongly than repeated runtime revision. Most of the benefit comes from the configuration selected by paired validation, and its mean advantage persists in a later US period. Runtime revision adds no clear gain in the main settings, consistent with the weak temporal transfer of historical utility, and is better viewed as a conservative safeguard when the initial configuration is poor. Replacement utility remains task-dependent because it depends on the model state, adaptation budget, and real-data alternative. The current generator is univariate, matching the per-series interface of the evaluated backbones.

6 Conclusion

The value of synthetic data depends on their effect on the adapted forecaster. UGapSyn measures this effect against the displaced real data through paired continuations from the same adaptation state. Utility-guided validation selects a synthetic configuration that achieves lower mean test MSE than real-data adaptation in all nine primary dataset-backbone settings. On Chronos-Bolt-Base, ten-seed paired intervals exclude zero on both US panels, and the mean advantage remains in an independent later US evaluation. Because historical utility transfers only weakly across periods, utility guidance works best when it selects synthetic data before adaptation, with runtime revision as a conservative safeguard for unfavorable initial configurations.

Ethics statement

The experiments use publicly available daily prices of listed equities and collect no personal or demographic data. US_71 and US_14L come from the FinCast release (Zhu et al., 2025), the Chinese panels come from a public point-in-time release, and the later US period extends the same panels with public daily closes where available (Appendix C). Generated paths come from a parametric process fitted to training returns and carry no asset identity. Forecasters adapted in this way could inform trading decisions. Gains measured on historical panels need not carry over to later market conditions (Section 4.4), so such forecasters require separate validation before any use in investment decisions.

Reproducibility statement

Appendix C gives the chronological split manifest, the adaptation configuration and the baseline implementations. Appendix D gives the paired-continuation procedure, the random-stream coupling and the auxiliary-update accounting, and Appendix E gives the generator construction. Appendix B.3 records the registered selection rules, the validation ratios behind every configuration choice and the freeze before the test and forward evaluations, and Appendix B.4 records that the CSI500 and CSI1000 panels were registered before they were built. Appendix B.6 describes the audit replay behind Section 4.4. Each table or its accompanying text states the number of seeds, and stochastic methods within a panel share seeds. Code, configurations and the frozen synthetic pools will be released with the camera-ready version.

AI use statement

Generative AI tools assisted with writing, literature search and reference checking. Chronos-Bolt (Ansari et al., 2024; Simeone, 2026) and TimesFM-2.5 (Das et al., 2023) are the pretrained forecasters adapted in the experiments. The authors verified all AI-assisted content and take full responsibility for it.

References

Ahmed Alaa, Boris Van Breugel, Evgeny S Saveliev, and Mihaela Van Der Schaar. How faithful is your synthetic data? sample-level metrics for evaluating and auditing generative models. In *International conference on machine learning*, pp. 290–306. PMLR, 2022.

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327, 1986.

Anthony Bolton, Wuyang Zhou, Zehua Chen, Giorgos Iacovides, and Danilo Mandic. Refinebridge: Generative bridge models improve financial forecasting by foundation models. In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1796–1800. IEEE, 2026.

Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Nazanin Mohammadi Sepahvand, Edward Raff, Kanika Madan, Vikram Voleti, et al. Accounting for variance in machine learning benchmarks. *Proceedings of machine learning and systems*, 3:747–769, 2021.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10708–10717. IEEE, 2022.

Liyuan Chen, Shuoling Liu, Jiangpeng Yan, Xiaoyu Wang, Henglin Liu, Chuang Li, Kecheng Jiao, Jixuan Ying, Yang Veronica Liu, Qiang Yang, et al. Advancing financial engineering with foundation models: progress, applications, and challenges. *Engineering*, 2025.

Fatemeh Chitsaz and Saman Haratizadeh. Dual adaptation of time-series foundation models for financial forecasting. In *1st ICML Workshop on Foundation Models for Structured Data*, 2025.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023.

Jianrong Ding, Zhanyu Liu, Guanjie Zheng, Haiming Jin, and Linghe Kong. Condtsf: One-line plugin of dataset condensation for time series forecasting. *Advances in Neural Information Processing Systems*, 37:128227–128259, 2024.

Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha V Naidu, and Colin White. Forecastpfn: Synthetically-trained zero-shot forecasting. *Advances in Neural Information Processing Systems*, 36:2403–2426, 2023.

Zijun Dou, Zhenhe Yao, Zhe Xie, Xidao Wen, Tong Xiao, and Dan Pei. Autoda-timeseries: Automated data augmentation for time series. In *The Fourteenth International Conference on Learning Representations*, 2026.

Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

Zhiting Fan, Ruizhe Chen, Tianxiang Hu, Ru Peng, Zenan Huang, Haokai Xu, Yixin Chen, Jian Wu, Junbo Zhao, and Zuozhu Liu. Optimsyn: Influence-guided rubrics optimization for synthetic data generation. *arXiv preprint arXiv:2604.00536*, 2026.

Ziqing Fan, Siyuan Du, Shengchao Hu, Pingjie Wang, Li Shen, Ya Zhang, Dacheng Tao, and Yanfeng Wang. Combatting dimensional collapse in llm pre-training data via submodular file selection. In *International Conference on Learning Representations*, volume 2025, pp. 6655–6679, 2025.

Xinghong Fu, Masanori Hirano, and Kentaro Imajo. Financial fine-tuning a large time series model. In *2025 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CiFer)*, pp. 1–9. IEEE, 2025.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In *International conference on machine learning*, pp. 2242–2251. PMLR, 2019.

Hongbin Huang, Minghua Chen, and Xiao Qiao. Generative learning for financial time series with irregular and scale-invariant patterns. In *International Conference on Learning Representations*, volume 2024, pp. 56374–56394, 2024.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Understanding predictions with data and data with predictions. In *International Conference on Machine Learning*, pp. 9525–9587. PMLR, 2022.

Yiding Jiang, Allan Zhou, Zhili Feng, Sadhika Malladi, and Zico Kolter. Adaptive data optimization: Dynamic sample selection with scaling laws. In *International Conference on Learning Representations*, volume 2025, pp. 58014–58037, 2025.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. Pmlr, 2017.

Yong Liu, Guo Qin, Zhiyuan Shi, Zhi Chen, Caiyin Yang, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Sundial: A family of highly capable time series foundation models. *arXiv preprint arXiv:2502.00816*, 2025.

Robert C Merton. Option pricing when underlying stock returns are discontinuous. *Journal of financial economics*, 3(1-2):125–144, 1976.

Eghbal Rahimikia, Hao Ni, and Weiguan Wang. Re (visiting) time series foundation models in finance. *arXiv preprint arXiv:2511.18578*, 2025.

Cécile Rousseau, Tobia Boschi, Giandomenico Cornacchia, Dhaval Salwala, Alessandra Pascale, and Juan Moreno. Forging time series with language: A large language model approach to synthetic data generation. *Advances in neural information processing systems*, 38:78402–78432, 2026.

Lifeng Shen, Kai Syun Hou, Weiyu Chen, and James Kwok. Latent-to-data cascaded diffusion models for unconditional time series generation. In *The Fourteenth International Conference on Learning Representations*, 2026.

Luigi Simeone. Time series foundation models for energy load forecasting on consumer hardware: A multi-dimensional zero-shot benchmark. *arXiv preprint arXiv:2602.10848*, 2026.

Ege Onur Taga, Halil Alperen Gozeten, Kutay Tire, Rahul Dalvi, Reinhard Heckel, and Samet Oymak. Filter, augment, forecast: Online data selection for robust time series forecasting. In *The 29th International Conference on Artificial Intelligence and Statistics*, 2026.

Haoru Tan, Sitong Wu, Yanfeng Chen, Jun Xia, Ruobing Xie, Bin Xia, Samm Sun, and Xiaojuan Qi. Fast data mixture optimization via gradient descent. In *International Conference on Learning Representations*, volume 2026, pp. 702–715, 2026.

Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. Quant gans: deep generation of financial time series. *Quantitative Finance*, 20(9):1419–1440, 2020.

Haochong Xia, Yao Long Teng, Regan Tan, Molei Qin, Xinrun Wang, and Bo An. History is not enough: An adaptive dataflow system for financial time-series synthesis. Technical report, 2026.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 36:69798–69818, 2023.

Huaxiu Yao, Caroline Choi, Bochuan Cao, Yoonho Lee, Pang Wei W Koh, and Chelsea Finn. Wild-time: A benchmark of in-the-wild distribution shift over time. *Advances in Neural Information Processing Systems*, 35:10309–10324, 2022.

Jiexia Ye, Yongzi Yu, Weiqi Zhang, Le Wang, Jia Li, and Fugee Tsung. Empowering time series analysis with foundation models: A comprehensive survey. *Information Fusion*, pp. 104601, 2026.

Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32, 2019.

Zichun Yu, Spandan Das, and Chenyan Xiong. Mates: Model-aware data selection for efficient pretraining with data influence models. *Advances in Neural Information Processing Systems*, 37: 108735–108759, 2024.

Xinyu Yuan and Yan Qiao. Diffusion-ts: Interpretable diffusion for general time series generation. In *International Conference on Learning Representations*, volume 2024, pp. 41582–41610, 2024.

Yingxiao Zhang, Jiaxin Duan, Junfu Zhang, and Ke Feng. Tf-codit: Conditional time series synthesis with diffusion transformers for treasury futures. *arXiv preprint arXiv:2601.11880*, 2026.

Zhuohang Zhu, Haodong Chen, Qiang Qu, and Vera Chung. Fincast: A foundation model for financial time-series forecasting. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 4539–4549, 2025.

A Related Work

This appendix positions UGapSyn within three lines of research, **(1) financial foundation-model adaptation**, **(2) synthetic data for forecasting** and **(3) model-dependent data selection**. The comparison concerns what each approach changes or measures and how that measurement informs the use of generated training data.

Financial foundation-model adaptation. Surveys of time-series and financial foundation models identify limited and non-stationary financial data as a central obstacle to transfer (Ye et al., 2026; Chen et al., 2025). Large-scale evaluations find that off-the-shelf models transfer poorly to financial returns and that domain-specific training helps (Rahimikia et al., 2025; Zhu et al., 2025). Adaptation methods change the model through continual pretraining on financial prices (Fu et al., 2025), adapters that separate shared and asset-specific patterns (Chitsaz & Haratizadeh, 2025), or a generative module that refines the forecasts of a frozen model (Bolton et al., 2026). UGapSyn keeps the adaptation recipe and its budget fixed and changes the data it consumes, so these methods are complementary to it. A matched comparison would require their recipes to share the backbone, real-data plan and main budget used here.

Synthetic data for forecasting. Pretraining-scale forecasters already mix generated or transformed series with real corpora or train on synthetic series alone (Ansari et al., 2024; Das et al., 2023; Liu et al., 2025; Dooley et al., 2023). Task-specific generators further target financial patterns, conditional market scenarios, temporal distributions or compressed training sets (Wiese et al., 2020; Yoon et al., 2019; Huang et al., 2024; Zhang et al., 2026; Yuan & Qiao, 2024; Shen et al., 2026; Rousseau et al., 2026; Ding et al., 2024). Their evaluation commonly relies on fidelity, diversity or downstream statistics (Alaa et al., 2022). Those criteria answer how closely a generator reproduces data, and train-on-synthetic, test-on-real evaluation (Esteban et al., 2017) measures downstream use without comparing the generated data with the real data they would displace. Section 4.3 compares these criteria with replacement utility under a matched adaptation budget. Adaptive financial synthesis adjusts augmentation and curation operations to market drift through bi-level optimization of a downstream objective (Xia et al., 2026). UGapSyn instead compares each synthetic-data configuration with the real windows it displaces under a matched budget, using paired continuations from one training state. The parametric process used here, with conditional heteroskedasticity and jumps, serves as a synthetic-data source whose type and amount can be varied on purpose (Bollerslev, 1986; Merton, 1976). This controllability makes the process suitable for the matched utility comparisons studied here.

Model-dependent data selection. A second line values training data through the model. Retraining-based values, influence functions and datamodels relate training data to predictions (Ghorbani & Zou, 2019; Koh & Liang, 2017; Ilyas et al., 2022), and learned proxies and decorrelated selection reduce the cost (Yu et al., 2024; Fan et al., 2025). Feedback-driven methods select or synthesize data from model signals (Taga et al., 2026; Dou et al., 2026; Fan et al., 2026), and mixture methods take feedback from proxy models, learning curves or validation objectives (Xie et al., 2023; Jiang et al., 2025; Tan et al., 2026). These methods relate data choices to model performance. The object here is a synthetic window and the comparison is against the specific real window it displaces, which makes the alternative explicit and the two branches paired by construction. The measurement is taken during adaptation on held-out data. Section 4.4 examines its relationship with later utility.

B Additional Experimental Evidence

This appendix expands the evidence for synthesis and its use during adaptation through seven analyses. **(1)** Paired differences across ten seeds quantify the significance of the main comparisons. **(2)** Component controls and a later evaluation period separate the validation-selected configuration from runtime revision and test temporal transfer. **(3)** Validation comparisons document configuration selection. **(4)** Additional adaptation settings and an equal-computation comparison test the runtime rule and alternative training allocations. **(5)** Generation scores test fidelity as a proxy for utility. **(6)** Decision audits examine the feedback used by the controller. **(7)** Repetition controls test a stricter screening rule. Each analysis retains its stated seed set and evaluation conditions.

B.1 Paired Differences Across Seeds

Table 2 gives paired differences for the ten Chronos-Bolt-Base seeds behind Table 1. Each pair uses the same adaptation seed, accounting for variation shared by the two methods. Intervals use 4,000 bootstrap resamples of the ten pairs with random seed 20260923. Negative differences favor UGap-

Syn. The intervals describe seed variation conditional on the frozen generator pools and exclude variation from generator fitting or configuration selection.

Table 2: Ten-seed paired comparison on the test period. Entries give UGapSyn minus each competitor, with a 95% interval from 4,000 bootstrap resamples of the ten paired seeds. Negative favors UGapSyn. An asterisk marks an interval that excludes zero. Every comparator has a complete ten-seed evaluation on all three panels. The Chronos-Bolt-Base columns of Table 1 report the means of the same runs.

Competitor	US_71	US_14L	CSI300
Real adaptation	-0.0045^{*} $[-0.0066, -0.0025]$	-0.0300^{*} $[-0.0372, -0.0229]$	-0.0039 $[-0.0149, +0.0070]$
FTS-Diffusion	-0.0091^{*} $[-0.0105, -0.0074]$	-0.0040 $[-0.0119, +0.0024]$	-0.0458^{*} $[-0.0553, -0.0368]$
SDForger	-0.0360^{*} $[-0.0386, -0.0330]$	-0.0223^{*} $[-0.0282, -0.0170]$	-0.0545^{*} $[-0.0704, -0.0393]$
L2D-Diff	-0.0144^{*} $[-0.0161, -0.0122]$	-0.0263^{*} $[-0.0327, -0.0203]$	-0.1343^{*} $[-0.2366, -0.0583]$
Diffusion-TS	-0.0033^{*} $[-0.0046, -0.0018]$	-0.0112^{*} $[-0.0174, -0.0055]$	-0.0073 $[-0.0175, +0.0017]$
MTT+CondTSF	-0.0084^{*} $[-0.0098, -0.0069]$	-0.0265^{*} $[-0.0324, -0.0220]$	$+0.0171^{*}$ $[+0.0071, +0.0268]$
Fixed, sampled drift	-0.0023^{*} $[-0.0031, -0.0015]$	$+0.0055^{*}$ $[+0.0004, +0.0119]$	-0.0375^{*} $[-0.0644, -0.0158]$
Fixed, zero drift	$+0.0000$ $[-0.0000, +0.0000]$	$+0.0045^{*}$ $[+0.0000, +0.0098]$	$+0.0005$ $[-0.0008, +0.0024]$
UGapSyn, $m = 0$	-0.0003 $[-0.0008, +0.0000]$	-0.0100^{*} $[-0.0201, -0.0000]$	$+0.0001$ $[-0.0030, +0.0039]$

The intervals describe variation across the ten seeds, and every significance statement in the main text uses them.

B.2 Component Controls and Later-Period Evaluation

This subsection reports the complete component controls behind Section 4.3 and the evaluation of every method on a later period (Table 3).

Component controls. Table 3(a) reports ten-seed mean results for the component controls discussed in Section 4.3. Every configuration uses the same seeds within a panel and the same main adaptation budget. The fixed zero-drift configuration removes runtime decisions. Sampling the drift changes the generated data while keeping the policy fixed. The other controls change the retreat threshold, voting alternatives or available actions.

Keeping the zero-drift configuration fixed matches UGapSyn on US_71 and gives lower means on US_14L and CSI300, 1.0885 and 1.9531 against 1.0930 and 1.9536. Sampling the drift raises the mean on US_71 and CSI300 and gives the lowest mean on US_14L. Every configuration except sampled drift on CSI300 improves on real-only training, and runtime decisions add no gain over the fixed zero-drift configuration (Table 2).

Setting $m = 1$ raises the mean on both US panels and matches UGapSyn on CSI300. Setting $m = 0$ and binary voting raise US_14L MSE from 1.0930 to 1.1030 and stay within 0.0001 of UGapSyn on CSI300. Expanding the action set raises all three means. The comparison shows that additional choices do not consistently improve prediction. Appendix B.3 records the selection rules, and Appendix B.7 evaluates additional screening repetitions.

Later-period evaluation. The archived US panels end on 2023-12-29. The extension adds daily closes from 2024-01-02 to 2026-09-18. Each asset is rescaled by the median ratio of archived to retrieved prices over the final twenty overlapping trading days. Assets are excluded when that ratio varies by more than one part in a thousand or their calendar is incomplete. US_71 loses three assets,

one unavailable from the service and two affected by corporate actions in the join window. The resulting panels have 68 and 14 assets, 681 added days, and 534 horizon-20 forecast origins each.

Table 3: Component controls and later-period evaluation. (a) Ten-seed mean test MSE under the same main adaptation budget, with the seeds of Table 2 shared by every configuration. Bold marks the lowest mean and underlining the next distinct mean, including displayed ties. (b) The forward period begins after the last window used by any method for fitting or selection. MSE is reported with the seed standard deviation. The reported configuration was frozen before this evaluation, and no configuration was changed or selected using forward outcomes. Ten seeds are used for every stochastic method. The deterministic zero-shot reference has no seed standard deviation.

(a) Component controls

	Mean test MSE		
Configuration	US_71	US_14L	CSI300
Real only	0.1877	1.1229	1.9575
Fixed, zero drift	**0.1832**	<u>1.0885</u>	**1.9531**
Fixed, sampled drift	0.1855	**1.0874**	1.9911
UGapSyn, $m = 1$	<u>0.1835</u>	1.1006	1.9536
UGapSyn, $m = 0$	<u>0.1835</u>	1.1030	<u>1.9535</u>
Binary voting	<u>0.1835</u>	1.1030	1.9536
Expanded actions	0.1837	1.0959	1.9539
UGapSyn	**0.1832**	1.0930	1.9536

(b) Later-period evaluation

	US_71		US_14L	
Method	MSE	SD	MSE	SD
Zero-shot	0.7153	n/a	4.9804	n/a
Real adaptation	<u>0.6723</u>	0.0097	4.8002	0.0794
FTS-Diffusion	0.7068	0.0082	4.8122	0.0312
SDForger	0.7999	0.0219	4.8739	0.0455
L2D-Diff	0.7270	0.0093	4.9101	0.0300
Diffusion-TS	0.6758	0.0074	4.8087	0.0632
MTT+CondTSF	0.6912	0.0093	4.8179	0.0421
Fixed, sampled drift	0.6738	0.0108	**4.7802**	0.0353
Fixed, zero drift	**0.6684**	0.0067	4.7973	0.0302
UGapSyn	**0.6684**	0.0067	<u>4.7931</u>	0.0287

Figure 3(f) and Table 3(b) compare the methods on the later period. UGapSyn has lower mean MSE than real-data adaptation and all five external synthesis or condensation methods on both panels. The fixed zero-drift configuration matches UGapSyn on US_71, while fixed sampled drift is better on US_14L. The forward results therefore support the value of the synthetic-data configuration beyond the original test period, while its comparison with the fixed configurations leaves limited evidence for an additional gain from runtime revision.

Archived observations remain unchanged, and rebuilding the original test panels reproduces window hashes and asset scales. The extension is assembled under a recorded rule, and the reported configuration is frozen before forward evaluation. Forward outcomes do not enter configuration selection or any subsequent method revision.

The fixed-configuration controls distinguish persistence of the generation benefit from any additional benefit of runtime revision.

B.3 Validation-Based Configuration Selection

The registered rule selects one configuration across all three datasets from three matched pilot seeds. It averages the ratio of each candidate's mean validation MSE to that of the full-step control. Ties within 10^{-6} prefer full step, half step, later panel and zero drift in that order. Half step both retains the initial mixture and halves proposal steps. Later panel adds confirmation on the later calibration panel. Zero drift additionally suppresses simulated drift. The half-step contrast therefore combines two controller changes. Figure 4 reports the relative validation MSE of each candidate on the three panels.

The frozen configuration is the zero-drift candidate, which lowers validation MSE relative to the full-step control on each panel. Its global mean validation ratio is 0.99314, compared with 1.00000 for full step, 1.00108 for half step and 0.99932 for later panel. The choice uses all three pilot seeds on every panel and is applied across datasets. The selected configuration is frozen before the independent test and forward evaluations, and neither evaluation is used for subsequent configuration selection.

Subsequent configuration comparisons. The retreat comparison evaluates $m = 0$, $m = 1$ and a disabled fallback under a registered validation rule. Their mean validation MSE ratios are 1.00000, 0.99960 and 0.99644, with 7, 3 and 0 retreats across sixty refreshes. The rule selects the disabled

fallback, represented by $m = \infty$ in Eq. 10. This is the configuration reported as UGapSyn in the main table and the component controls. Proposal formation and confirmation remain active.

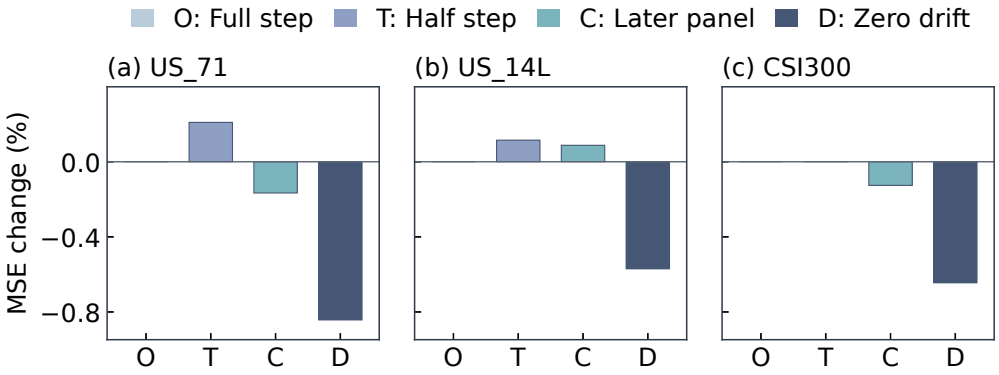

Figure 4: Four registered candidates on validation. Each relative MSE compares the candidate and full-step control on common seeds 0–2. Selection averages the three panel ratios and chooses one configuration for all three panels. Half step also retains the initial mixture among the voting policies.

The $m = 1$ configuration is retained as a separate control for conditions in which the initial configuration can be harmful. It has a higher validation ratio than the disabled configuration and does not replace the main method row. The audit with retreat enabled and the additional adaptation settings in Appendix B.4 examine its decisions and forecasting effects. Keeping these configurations separate avoids attributing the gains of one rule to another.

The action-space comparison adds each available external generator pool as a pure policy with zero initial weight. Its validation MSE ratio is 1.00612, compared with 1.00000 for the unchanged action space, so it is not selected. The resulting test performance is reported as the expanded-actions ablation. Accepted external pools differ across panels, with Diffusion-TS used on US_71 and L2D-Diff on US_14L. Adding useful candidates is insufficient when the feedback does not reliably identify when to use them.

The repetition comparison increases screening repetitions from two to four under an unchanged decision rule. Its validation ratio is 1.000128 against 1.000000, and the two-repetition configuration is retained. Appendix B.7 reports its effect on agreement and computation. CSI500 and CSI1000 have separate recorded membership, split and evaluation rules. Their additional $m = 1$ runs are controls for runtime revision and do not determine the main configuration.

B.4 Additional Adaptation Settings and Equal Computation

Additional adaptation settings. Table 4 examines a second pretrained backbone, a harmful initial configuration and two additional panels. The runtime configuration uses $m = 1$ in Eq. 10. It is distinct from the validation-selected configuration in Table 1. The first three rows use the second backbone. The remaining rows use the main backbone. CSI500 has five seeds, reduced from ten before any of its models was trained, and the other settings have ten.

On the second backbone, fixed generation lowers MSE relative to real adaptation by 0.0048, 0.0190 and 0.0039 on US_71, US_14L and CSI300, using the displayed means. Runtime control raises the first two means by 0.0010 and 0.0082 and lowers the CSI300 mean by 0.0003. Thus, generation gains extend to this backbone, but a further benefit from runtime updating remains limited.

Starting CSI300 with sampled drift gives a fixed-configuration MSE of 1.9911, compared with 1.9575 for real adaptation. The $m = 1$ rule reduces it to 1.9665, recovering part of the loss without reaching the real-data baseline. On CSI1000, the same runtime variant reaches 1.3639, below the fixed configuration at 1.3671 and real adaptation at 1.3646.

CSI500 shows a limitation of the threshold. The fixed configuration reaches 1.8005 against 1.7802 for real adaptation, and the $m = 1$ variant remains at 1.8006. Its measured real-data advantage does not exceed the threshold, so the harmful configuration is retained. Appendix B.3 reports the

Table 4: Test MSE in additional adaptation settings. The first block evaluates Chronos-Bolt-Small on the three development panels. The second starts CSI300 from the sampled-drift configuration. The third evaluates two panels whose rules were registered and hashed before construction. The runtime control uses $m = 1$, distinct from the main configuration.

Setting	Real adaptation	Fixed configuration	UGapSyn, $m = 1$	Seeds
Small, US_71	0.1855	0.1807	0.1817	10
Small, US_14L	1.0700	1.0510	1.0592	10
Small, CSI300	1.9876	1.9837	1.9834	10
Base, CSI300 from sampled drift	1.9575	1.9911	1.9665	10
Base, CSI500	1.7802	1.8005	1.8006	5
Base, CSI1000	1.3646	1.3671	1.3639	10

unthresholded alternative. The additional panels were registered before their construction and were not used to select the reported configuration.

Table 5: Equal total computation, test MSE. Real adaptation and the matched fixed configuration are rerun with 272 main blocks, giving 1,092 main updates against the 52 main and about 1,039 auxiliary updates UGapSyn spends. The last column lists the numbers of seeds for the extended-training controls and the UGapSyn reference, respectively.

	Real adaptation		Fixed configuration		UGapSyn	
Panel	52	1092	52	1092	52	Seeds
US_71	0.1877	0.2433	0.1832	0.2004	0.1832	2, 10
US_14L	1.1229	1.4204	1.0885	1.2041	1.0930	2, 10
CSI300	1.9575	2.0425	1.9531	1.9306	1.9536	2, 10

Equal total computation. Matching the main adaptation budget leaves a separate question about the cost of candidate evaluation. Figure 3(g) compares UGapSyn with real-data and fixed-configuration adaptation given 1,092 main updates. UGapSyn uses 52 main updates and approximately 1,039 auxiliary updates. On both US panels, UGapSyn has lower mean MSE than either extended-training alternative. On CSI300, longer fixed-configuration training gives a lower mean, while longer real-data training does not. Allocating computation to candidate evaluation can therefore be useful, but the benefit depends on the task and the alternative use of that computation. The extended controls repeat the main block schedule with the same learning rate and optimizer, and a schedule tuned for longer training could perform differently.

B.5 Generation Scores and Replacement Utility

This comparison asks whether similarity to real training windows identifies synthetic sources that improve the forecaster. Each source is evaluated as a fixed policy under the same adaptation budget. This analysis is diagnostic only and does not enter configuration selection. Table 8 summarizes the correlations, and Table 6 gives the underlying scores.

Table 6: Per-source generation scores and replacement utility, the entries behind Table 8. Utility is the test MSE of real adaptation minus that of the source under the same main budget. Parametric sources use three seeds. External generators use their recorded evaluations, three seeds on the US panels and ten on CSI300.

Panel	Source	Energy dist.	Kurtosis gap	Vol. ACF gap	Utility
US_71	Region 1	0.0041	1.815	0.0223	+0.0031
	Region 2	0.0024	1.537	0.0168	+0.0032
	Region 3	0.0052	1.820	0.0214	+0.0029
	Region 4	0.0031	1.582	0.0158	+0.0031
	Uniform, zero drift	0.0028	1.659	0.0070	+0.0015
	Uniform, sampled drift	0.0033	1.686	0.0076	−0.0017
	FTS-Diffusion	0.1214	9.006	0.0936	−0.0036
	L2D-Diff	0.0111	0.208	0.2280	−0.0076
	MTT+CondTSF	0.0203	5.692	0.1525	−0.0031
	SDForger	0.0035	2.337	0.0567	−0.0284
	Diffusion-TS	0.0473	3.507	0.0237	+0.0024
US_14L	Region 1	0.0015	0.938	0.0212	+0.0343
	Region 2	0.0013	0.642	0.0256	+0.0341
	Region 3	0.0022	0.934	0.0207	+0.0329
	Region 4	0.0008	0.700	0.0250	+0.0325
	Uniform, zero drift	0.0008	0.783	0.0155	+0.0288
	Uniform, sampled drift	0.0010	0.791	0.0166	+0.0279
	FTS-Diffusion	0.1175	19.017	0.1119	+0.0253
	L2D-Diff	0.0153	1.559	0.2646	−0.0010
	MTT+CondTSF	0.0307	6.885	0.1492	−0.0010
	SDForger	0.0043	1.365	0.0696	+0.0068
	Diffusion-TS	0.0015	1.324	0.0400	+0.0109
CSI300	Region 1	0.0032	1.340	0.0305	+0.0163
	Region 2	0.0030	1.055	0.0222	+0.0180
	Region 3	0.0047	1.334	0.0296	+0.0165
	Region 4	0.0032	1.096	0.0224	+0.0184
	Uniform, zero drift	0.0036	1.188	0.0190	+0.0154
	Uniform, sampled drift	0.0046	1.181	0.0196	−0.0438
	Diffusion-TS	0.0341	2.806	0.0279	−0.0034
	L2D-Diff	0.0038	2.077	0.2275	−0.1304
	MTT+CondTSF	0.0226	6.672	0.1425	+0.0210
	FTS-Diffusion	0.0068	0.807	0.0366	−0.0419
	SDForger	0.0104	2.029	0.0548	−0.0506

These measurements describe different properties of a synthetic source. Their association with utility must be checked for each target forecasting task.

B.6 Decision Audits and Comparison Conditions

The audit addresses two questions, whether candidate comparisons repeat under controlled conditions and whether their measured utility transfers to test data. Table 7 separates comparison variability, historical-to-test agreement and the consequences of each decision.

Table 8: Generation quality against replacement utility. Each entry is the Spearman correlation, across the synthetic sources of a panel, between a generation score and the replacement utility that source obtained as a fixed policy under the same main budget. The gap scores are distances to the real training windows, so a negative correlation means closer reproduction goes with more utility. Table 6 gives the corresponding per-source scores.

Panel	Energy dist.	Volatility	Kurtosis	Return ACF	Vol. ACF	Tail	Sources
US_71	−0.45	+0.03	−0.33	−0.77	−0.57	−0.42	11
US_14L	−0.61	−0.41	−0.72	−0.90	−0.71	−0.73	11
CSI300	−0.25	−0.12	−0.08	−0.35	−0.22	+0.17	11
Pooled	−0.57	−0.03	−0.44	−0.54	−0.35	−0.08	33

Table 7: Measurement repeatability and temporal validity. (a) One screen is repeated four times on each of three panels, with policy updates disabled. Spread is the standard deviation of candidate utility relative to the paired design. Same winner is the fraction of repetition pairs selecting the same candidate. Separation divides the spread of candidate means by repetition spread. (b) The 26 audited refreshes with retreat enabled, grouped by committed decision. Regret is the committed policy's test loss minus the lowest candidate test loss. (c) The same refreshes by panel. Rank correlation and sign agreement compare historical with test utility. Confirmed proposals count accepted proposals whose later outcome agrees with the acceptance decision. Test scores are used only for this retrospective audit.

(a) Repeatability across three panels

Design	Spread	Same winner	Separation
Private generation streams	0.77	0.44	2.62
Independent continuations	1.08	0.14	1.42
Paired, UGapSyn	1.00	0.36	1.03

(b) Regret by committed decision

Committed	Refreshes	Regret
Incumbent	20	+0.0005
Proposal	4	+0.0003
Real replay	2	+0.0335

(c) Transfer of measured utility to the test period

Panel	Refreshes	Rank corr.	Sign agreement	Confirmed	Regret
US_71	10	−0.20	0.54	0 of 0	+0.0003
US_14L	10	−0.05	0.19	2 of 4	+0.0045
CSI300	6	+0.20	0.33	1 of 1	+0.0028

Evaluating policy decisions. For an evaluation panel P, let $\mathcal{L}_P(\rho)$ be the loss after continuing policy ρ from S_k under the remaining budget. The regret of the committed policy is

$$\mathrm{Reg}_k^P = \mathcal{L}_P(\pi_{k+1}) - \min_{\rho \in \mathcal{V} \cup \{\pi_k\}} \mathcal{L}_P(\rho). \tag{11}$$

Using $P = A$ evaluates the decision on confirmation data. Using a later panel evaluates whether the decision transfers beyond the feedback period.

The utilities in Section 4.4 come from a replay with one switch added. After a screening or confirmation branch has trained to the end of the schedule and has been scored on the panel the controller reads, the same state is scored again on validation and on test. Only then is the main trajectory restored. Quantile prediction draws no random numbers, so the branch that follows is the one the frozen runner produced. The replay writes the same refresh records as the frozen run. After the added scores and wall-clock fields are dropped, a field-by-field comparison confirms that every committed decision, proposal, winner set and acceptance reason is identical in all thirteen replayed runs.

The replayed configuration keeps the retreat enabled, because the regret of a retreat can only be measured where retreats occur, and the reported configuration disables them. Every other rule is the reported one. Table 7(b) reports decision regret for this diagnostic configuration. Its enabled fallback must be distinguished from the disabled fallback selected for the main comparison.

Effect of matched comparison conditions. The measured utility is a difference of two losses, so its variance is

$$\mathrm{Var}\,\widehat{U}_{k,r}(\pi) = \mathrm{Var}\,\ell_{k,r}(e_0) + \mathrm{Var}\,\ell_{k,r}(\pi) - 2\,\mathrm{Cov}\big(\ell_{k,r}(e_0), \ell_{k,r}(\pi)\big). \tag{12}$$

Matched conditions can lower the variance of the difference when they induce positive covariance between branch losses. This benefit is not guaranteed for every source of randomness. The comparison in Table 7 therefore separates matched training conditions from identical generation streams. It reports both repetition variability and agreement on the best candidate.

The coupling levels. Table 7(a) compares three designs on one screening round, with the controller switched off so no policy moves and only the measurement is exercised. *Paired* is the method of Section 3.2. All candidates continue from one snapshot, under one real-data plan and one set of eligible positions, and two candidates that place a synthetic window at the same position draw the same numbers for it. *Private generation streams* keeps the same snapshot, plan and positions but gives each candidate its own generation randomness. *Independent continuations* shares only the snapshot, so the plan, the positions and the streams all differ. Each level is replayed four times on

each of the three panels. The columns report the standard deviation of a candidate utility across those repetitions relative to the paired design on the same panel, the fraction of repetition pairs naming the same best candidate, and the spread of the candidate means divided by the repetition spread.

Table 9 compares the judgment of each replayed candidate on the screening panel with the utility its continuation realized on the test period.

Table 9: The screening panel against the later period, from the audit replay. A candidate counts as judged worse than real replay when its utility measured on the screening panel is negative, and as better than real replay when the utility the same continuation realized on the test period is positive. The last two columns give the median rank correlation from the measured utility to the utility realized on the validation and test periods. Every candidate of every replayed refresh is included.

Panel	Candidates	Judged worse	Realized better	ρ to validation	ρ to test
US_71	50	23 (46%)	50 (100%)	−0.85	−0.10
US_14L	52	42 (81%)	52 (100%)	+0.70	−0.17
CSI300	30	20 (67%)	30 (100%)	+0.85	+0.20
All panels	132	85 (64%)	132 (100%)	+0.05	−0.07

These retrospective scores explain recorded decisions without changing the policies that produced the reported forecasts.

B.7 Number of Screening Repetitions

This analysis compares two and four screening repetitions with the decision rule held fixed. More repetitions can improve the description of variability, but requiring an intersection of all winner sets also makes agreement more restrictive. Proposition 1 in Appendix E.3 describes that effect.

For $K = 6$, matching the observed two-repetition agreement rate of 10 out of 18 gives $p = 0.736$. Equation 22 then predicts agreement of approximately 0.29 with four repetitions, or 5.3 of 18 refreshes. The observed count is 5. This calculation uses a simplified independent-winner model and illustrates the effect of the intersection rule rather than establishing independence in the actual measurements.

The registered comparison changes only the number of screening repetitions. Across three panels and three common seeds, agreement falls from 10 of 18 refreshes to 5 of 18, and policy changes fall from 4 to 2. On CSI300, agreement falls from 4 of 6 to 0 of 6. Auxiliary computation increases from approximately 1,015 to 1,872 updates per run, while the validation MSE ratio rises from 1.000000 to 1.000128. The comparison retains two repetitions.

Additional repetitions can describe measurement variation more fully, but the present rule requires an intersection of all winner sets. A disagreement in any added repetition can remove a previously supported proposal. On US_71, three of the four repetitions name the same winner at both refreshes, yet the remaining repetition prevents agreement. The experiment therefore evaluates more repetitions together with a stricter acceptance condition, rather than isolating a gain in measurement precision.

Real replay remains deterministic across repetitions, while synthetic continuations vary with generation. Increasing repetitions does not remove this asymmetry. A rule using pooled losses or majority agreement would make a different use of the repeated measurements and would require a separate validation comparison.

C Evaluation Setup and Reproducibility

This appendix specifies three parts of the evaluation, **(1) chronological splits and feedback observations**, **(2) adaptation settings and computation accounting** and **(3) baseline implementations**. Together they define which observations and resources are available to each comparison.

Order of the evaluation and size of the feedback panel. The evaluation compares synthesis with matched real adaptation, controls the composition of training data, and tests feedback against a fixed

policy. Both paired branches share real selection, model origin, and main budget. Panel D contains 16 evenly spaced admissible calibration origins. It supplies 1,136 labeled windows on US_71 and 224 on US_14L. Every auxiliary branch holds its candidate policy through the remaining schedule. This fixed-policy continuation measures a decision proxy for the next policy update. The main trajectory may change its policy at a later refresh.

C.1 SPLIT MANIFEST

Table 10 gives the period boundaries. Validation and test require complete windows inside each period, giving the period length minus 147 possible horizon-20 origins on a complete calendar. The per-asset error scale is the standard deviation of its training prices. US prices are the release's close prices, with no additional corporate-action adjustment. CSI300 admits an asset only when it belongs to the index at the forecast origin and all 148 window prices are finite and positive. Assets without a positive training scale are excluded. All adapted models share the same pretrained checkpoint. Possible pretraining overlap remains distinct from the chronological separation used during adaptation. Financial calendar differences are preserved, and scaling, interpolation and filtering do not use future target information.

Table 10: Chronological split manifest. Validation and test windows lie within their periods. US calibration contexts may extend into training. The admissible target-index construction above defines D and A. Forward evaluation uses the archived later-period extension.

Panel	Period	First day	Last day	Days	Origins
US_71, 71 assets	Training	2016-01-04	2020-06-23	1,126	n/a
	Calibration	2020-06-24	2021-08-05	282	16 in D
	Validation	2021-08-06	2022-05-24	202	55
	Test	2022-05-25	2023-12-29	402	255
	Forward	2024-01-02	2026-09-18	681	534
US_14L, 14 assets	Training	2005-01-04	2015-08-20	2,676	n/a
	Calibration	2015-08-21	2018-04-19	670	16 in D
	Validation	2018-04-20	2020-03-13	478	331
	Test	2020-03-16	2023-12-29	956	809
	Forward	2024-01-02	2026-09-18	681	534
CSI300, point-in-time	Training	2005-04-08	2021-10-28	4,025	n/a
	Calibration	2021-10-29	2023-06-20	400	16 in D
	Validation	2023-06-21	2024-09-11	300	153
	Test	2024-09-12	2026-09-15	486	339

CSI300 uses the fixed 2026-09-15 release of `investment_data`.[1] The reader consumes Qlib-format adjusted close prices and dated membership intervals. Membership follows the forecast origin, with no back-filling from the final constituent list or survivor filter. Training data use a stride of twenty origins and contains 27,359 complete windows. Evaluation uses stride one. Validation contains 44,426 windows over 153 origins, with 283 to 297 valid assets per origin. Test contains 93,840 windows over 339 origins, with 273 to 280 valid assets per origin. The source audit matches all 950 consumed files to the archived release. FTS-Diffusion and SDForger were outside the initial CSI300 evaluation. The extension adds both, each fitted once on training data under the recorded generator budget and shared by the three backbones and by all ten Bolt-Base seeds. After the FTS-Diffusion pool was written, its manifest failed to serialize a non-finite training loss. The pool file was kept unchanged, and the manifest was rebuilt from a replay that reproduced the stored cluster centroids and labels without refitting.

Let $[c_{\mathrm{lo}}, c_{\mathrm{hi}})$ denote the calibration interval and o_a an asset's first available calendar index. The US admissible first-target indices are

$$\mathcal{F}_{\mathrm{US}} = \{\max(c_{\mathrm{lo}}, \max_a o_a + L), \ldots, c_{\mathrm{hi}} - H\}. \tag{13}$$

Context and target indices are $[f - L, f)$ and $[f, f + H)$, respectively. CSI300 starts its candidate indices at $c_{\mathrm{lo}} + L$. Write the ordered admissible indices as $(f_0, \ldots, f_{n-1})$ and set $m = \lfloor \frac{n-H}{2} \rfloor$.

[1] https://github.com/chenditc/investment_data/releases/tag/2026-09-15

Panel D samples positions $0, \ldots, m$, while panel A samples positions $m + H, \ldots, n - 1$. Each uses up to 16 evenly spaced positions and requires at least eight. The final target date in D must precede the first target date in A. Their historical contexts may overlap, so target separation does not imply statistical independence.

C.2 Main configuration

Table 11 lists the settings that define the adaptation applied to each data source. The seed counts distinguish the main comparison from the separate diagnostic controls.

Table 11: Adaptation configuration.

Component	Setting
Backbones	Chronos-Bolt-Base, Chronos-Bolt-Small and TimesFM-2.5-200M, with fixed checkpoint identities. TimesFM uses its official causal patch normalization and median point forecast
Context and horizon	$L = 128$, $H = 20$
Warmup and mixed blocks	128 real exposures, followed by 12×128 exposures
Optimizer	AdamW, learning rate 10^{-5}, weight decay 0.01
Batch and precision	Batch size 32, float32 main adaptation
Real candidate pool	Up to 512 training windows per stage
Synthetic replacement	Five-action policy over 64 replaceable positions
Seeds	Three matched pilot seeds for configuration selection. Ten seeds for Chronos-Bolt-Base and three per panel for the other backbones
Metric	Horizon-20 MSE
Hardware	One 32 GB RTX 5090. Capacity-admitted training jobs

These settings fix the main-model budget while allowing the synthesis policy to change. The real coverage representation uses eight normalized log-price shape features with scales estimated from the permitted real data. Complete labeled windows from D supply the coverage reference. Feature definitions, reference indices, and weights form part of the data-construction configuration.

Timing of feedback. All calibration targets are observed before adaptation begins. The calibration period precedes each evaluation period, so the controller uses only past labeled data. Refresh times are fixed in advance, and each committed policy remains active until the next refresh. This is an offline adaptation setting. It does not evaluate rolling deployment with newly arriving labels.

Computation accounting. Every adapted method receives 1,664 exposures and 52 main updates. Auxiliary updates total $\sum_k E_k(2|S_k|+|C_k|)$ over distinct policies, without changing the main model state. Main-budget matching therefore does not imply equal total computation. Capacity admission permits concurrent training subject to memory reservations and a concurrency cap.

C.3 Baseline implementations

The five comparison methods retain their generation or condensation mechanisms, and each produces a frozen window pool. In Table 1, windows from each pool fill all 64 eligible positions of every 128-position main block, so each external method replaces 50% of the windows in every main block on every panel and backbone, the same level as the initial UGapSyn policy. This setting differs from the original papers' evaluations, which include fidelity assessment, synthetic-only training, and mixtures with real data. Reduced schedules are labeled as adaptations. Generator pools may be reused across adaptation seeds, while model-dependent experts must match the backbone. The configuration log records implementation changes, trials, stopping rules, and auxiliary costs. The reported L2D-Diff row is a paper-derived adaptation, and its code identity is recorded separately.

Frozen pools retain their original training and model-selection procedures. Current matching applies to the real-data plan, eligible positions, pretrained predictor, main adaptation budget, and evaluation panels. It does not equalize generator fitting, calibration feedback, or configuration-search costs. Auxiliary models may use the full permitted training set where the original algorithm requires it. The current candidate is selected on validation and frozen before the independent test and forward evaluations.

D Paired Continuation and Implementation Details

This appendix makes two implementation choices explicit, **(1) how each candidate continues from the same saved training state** and **(2) how the evaluation schedule determines auxiliary cost**. Algorithm 2 specifies the continuation used by the main algorithm, followed by the batch construction and update counts.

Algorithm 1 gives the policy decisions. Algorithm 2 specifies the continuation used for each comparison. Screening uses two generation streams on D, while confirmation uses a fresh stream on A. Returning to real data requires a gain beyond numerical tolerance in each screening repetition and a confirmation gain exceeding τ_k. No average across repetitions substitutes for either screening condition.

Algorithm 2 Paired continuation $\mathrm{Probe}(\mathsf{S}, C, E', P, r)$

Require: saved state S, policies C, remaining budget E', scoring panel P, stream r

$C \leftarrow \mathrm{Distinct}(C), \quad c \leftarrow 0, \quad \ell \leftarrow \{\}$
for $\rho \in C$ **do**
  $\mathsf{S}' \leftarrow \mathrm{Copy}(\mathsf{S})$
  for each remaining training block b **do**
    $T_b \leftarrow \mathrm{GenerateBlock}(G, \rho, T_{\mathrm{real},b}, \mathcal{I}_b, r)$
    $\mathsf{S}' \leftarrow \mathrm{TrainBlock}(\mathsf{S}', T_b)$
    $c \leftarrow c + \mathrm{Cost}(T_b)$
  end for
  $\ell(\rho) \leftarrow \mathrm{MSE}(f_{\mathsf{S}'}, P)$
  $\mathrm{Discard}(\mathsf{S}'), \quad \mathrm{Restore}(\mathsf{S})$
end for
return loss map ℓ, auxiliary cost c

Here G is the selected generator, $T_{\mathrm{real},b}$ is the fixed real block and $\mathcal{I}_b$ contains its replacement-eligible positions. Every policy uses the same inputs and batch order. The stream index r identifies common primitive generation draws at corresponding positions.

Coupling map. Each random stream is seeded by the adaptation seed, a tag for main, screening or confirmation use, the stage, the repetition, the block position, the attempt and a role. The four roles are action, parameters, anchor and innovations. The action stream uses attempt zero and one uniform draw u, and the action is the smallest j with $u < \sum_{i \le j} \pi_i$. Every policy therefore draws the same u at a position, and a pure policy still consumes it. The parameter stream draws its proposals as uniforms and maps the variance level and persistence affinely into the selected region, so two policies that choose different regions receive the same primitive draws at different locations. The anchor stream selects the training window whose first price anchors the path, and the innovation stream supplies the Student-t and jump draws. A rejected path moves only its own position to the next attempt. Other positions and the action at that position are unchanged. Real-window rows, eligible positions and batch order come from the stored plan and are identical across policies. Screening and confirmation use different tags, so confirmation paths are fresh. Table 7(a) separates the contribution of shared training conditions from that of shared generation streams.

Each continuation uses the complete remaining schedule with its policy held fixed, and all of its updates are charged to c. Numerical ties use tolerance $10^{-12}\max(1, |\ell|)$. Blocks contain $B = 128$ real windows selected by greedy coverage, with $K = 64$ replacement-eligible positions fixed across paired branches. State fingerprints check restoration. Refreshes leave 48 and 24 updates, while the main trajectory has 52 updates. Identical policies reuse one continuation. The real-replay fallback is applied without proposal shrinkage. The screening set contains the six voting policies and the incumbent after deduplication, giving at most six distinct policies at the opening refresh and seven at the second. With at most three confirmation policies per refresh, the auxiliary-update upper bound is $48 \times (2 \times 6 + 3) + 24 \times (2 \times 7 + 3) = 1{,}128$. Actual costs can be lower after deduplication.

E Generator Construction and Mathematical Analysis

This appendix provides three details supporting the method, **(1) parameter proposals and path validity**, **(2) second moments of the untruncated process** and **(3) the effect of repeated winner**

intersection. These calculations characterize the generator and comparison rules without assuming that either guarantees a forecasting improvement.

E.1 Parameter proposals and path construction

The formulas below use the effective simulated drift μ. The selected configuration fixes $\mu = 0$. It still draws the original drift proposal, denoted μ_{raw}, and records both values to preserve random-stream alignment. The general moment identities below retain μ explicitly and specialize to this configuration by setting it to zero.

For $\phi = (\mu, \omega, \alpha, \beta, \nu, \lambda, \sigma_J)$ the path follows

$$z_t \sim t_\nu \sqrt{\frac{\nu-2}{\nu}}, \qquad J_t = b_t \eta_t, \quad b_t \sim \text{Bernoulli}(\lambda), \quad \eta_t \sim \mathcal{N}(0, \sigma_J^2), \tag{14}$$

$$r_t = \mu + \sqrt{h_t} z_t + J_t, \qquad h_{t+1} = \omega + \alpha r_t^2 + \beta h_t, \tag{15}$$

and with a positive anchor a the prices are

$$\tilde{p}_t = a \exp\left(\sum_{j=0}^{t} \text{clip}(r_j, -0.5, 0.5) \right), \quad 0 \le t < L + H. \tag{16}$$

The anchor a precedes the generated path. It is the first price of a training window drawn uniformly from the real bank by the anchor stream of the position. Every generated price is proportional to a. Chronos-Bolt subtracts the context mean, divides by the context standard deviation and applies the same transform to the target, so the normalized training pair does not depend on a beyond floating-point effects. The sequence contains $L + H$ prices, with the first L forming the input and the next H the target. Parameters and variance dynamics remain unchanged across that boundary. The variance recursion uses the full return, including drift and jumps. This differs from a residual-based GARCH recursion, with second-moment consequences given in Section E.2. Independent action draws at eligible positions give

$$\mathbb{E}[m \mid \pi] = K(1 - \pi_0), \qquad \Pr(j \mid \text{synthetic}) = \frac{\pi_j}{1 - \pi_0}, \quad j = 1, \dots, 4, \tag{17}$$

when $\pi_0 < 1$.

For each training window, let μ_i, v_i, κ_i denote log-return mean, variance, and excess kurtosis. The tail proposal is

$$\nu_i = \text{clip}\left(4 + \frac{6}{\max(\kappa_i, 0.05)}, 3, 30 \right). \tag{18}$$

The fifth and ninety-fifth percentiles of μ_i, v_i, and ν_i define uniform proposal intervals for μ_{raw}, v, and ν. After the parameter draw, the selected configuration sets the simulated drift to $\mu = 0$. Squared standardized returns supply a lag-one correlation proxy s_i, clipped to $[0, 0.97]$. The persistence range is $[Q_{0.05}(\max(s_i, 0.02)), 0.97]$. Within the selected region, draw v and p uniformly and a response share $s \sim \mathcal{U}[0.05, 0.45]$. Set

$$\begin{aligned} &\alpha = ps, \qquad \beta = p(1-s), \\ &\omega = \max\{v \max(1-p, 10^{-3}), 10^{-10}\}, \qquad h_0 = \frac{\omega}{\max(1-\alpha-\beta, 10^{-6})}. \end{aligned} \tag{19}$$

Regions 1 and 2 use the lower half of the v range, and regions 3 and 4 use the upper half. Within each pair, the first region uses lower p and the second uses higher p. Jump frequency is uniform on $[0, 0.05]$, and $\sigma_J = \sqrt{v}\, z$ with $z \sim \mathcal{U}[0, 3]$. The primitive draws are independent conditional on the region. The recursion coefficients are linked by the relations above. The persistence cap, response-share range, and jump ranges are fixed priors. The initialization of h_0 is a numerical convention and does not identify v with the stationary return variance. Clipping applies only when converting returns to prices and does not enter the variance recursion.

For each synthetic position, rejection sampling stays within its selected region. Invalid lengths, nonfinite or nonpositive prices, and forbidden float32 path hashes are rejected. Each position allows at most 100 attempts, after which the run raises an error. Successfully constructed main blocks

exclude the real bank and previously committed synthetic paths. Auxiliary blocks exclude the real bank and duplicates within the same block. The laws q_j describe parameter proposals. Validity and duplicate checks also affect the distribution of accepted paths. Equation 17 applies to successfully constructed blocks.

E.2 Second moments of the untruncated process

Assume fixed parameters, independent zero-mean innovations, $\nu > 2$, $p = \alpha + \beta < 1$, and assume that a stationary solution with finite second moments exists. The following identities are conditional on this assumption. We do not use them as an existence proof. Put $j = \lambda\sigma_J^2$. Taking expectations in Eq. 15 gives

$$\mathbb{E}[h] = \omega + \alpha\{\mu^2 + \mathbb{E}[h] + j\} + \beta\mathbb{E}[h] = \frac{\omega + \alpha(\mu^2 + j)}{1 - p}, \tag{20}$$

$$\mathrm{Var}(r) = \mathbb{E}[h] + j = \frac{\omega + \alpha\mu^2 + (1 - \beta)j}{1 - p}. \tag{21}$$

Consequently, setting $\omega = v(1 - p)$ does not make v the unconditional variance when drift or jumps contribute to the recursion. This calculation describes the untruncated process only. It neither predicts a forecasting improvement from variance calibration nor characterizes the clipped output exactly.

E.3 Effect of repeated winner intersection

The following result isolates the acceptance effect of requiring the same winner across repetitions. The empirical comparison in Appendix B.7 also includes the computation required by those repetitions.

Proposition 1. *Let each of r screening repetitions name the best of K candidates with probability p and otherwise name one of the remaining $K - 1$ uniformly, independently across repetitions. Then $W = \bigcap_r W_r$ is nonempty with probability*

$$P_\cap(r) = p^r + \frac{(1 - p)^r}{(K - 1)^{r-1}}, \tag{22}$$

which is nonincreasing in r and strictly decreasing when no candidate has selection probability one.

Proof. Each repetition names exactly one candidate, so the intersection is nonempty precisely when all r name the same one. All naming the best gives p^r. For a fixed candidate other than the best, each repetition names it with probability $\frac{1-p}{K-1}$, so all r do with probability $(\frac{1-p}{K-1})^r$, and summing over the $K - 1$ such candidates gives the second term. Each term has the form q^r for a candidate selection probability $q \in [0, 1]$. Their sum is nonincreasing in r. It decreases strictly when every positive candidate probability is below one. □