Stonk Bench: Unified Benchmark for Synthetic Data Generation for Financial Time Series

Uyen Lam Ho*
Eddison Pham*
uyenlam.ho@mail.utoronto.ca
eddison.pham@mail.utoronto.ca
University of Toronto
Toronto, Ontario, Canda

Figure 1: Toronto Stock Exchange, Toronto, 1981

Abstract

Synthetic Data Generation (SDG) has become increasingly important in financial applications, particularly for generating Financial Time Series (FTS) data. However, evaluation of synthetic FTS remains inconsistent, with ad hoc methods limiting fair and reliable comparisons. To address this, we introduce StonkBench, a unified benchmark for synthetic data generation in financial time series (SDGFTS) that integrates four evaluation taxonomies: Fidelity, Diversity, Efficiency, and Utility, each comprising metrics specifically designed for forecasting tasks. Utility evaluations leverage portfolio assessments generated by a (deep) hedger. StonkBench builds upon Stenger's taxonomies and TSGBench's SDG framework, combining an enhanced standardization pipeline tailored for FTS, a comprehensive set of evaluation metrics, and an open-source framework that unifies evaluation procedures, enabling consistent, reproducible comparisons across datasets and models. Additionally, our benchmark evaluates statistical properties observed in financial time series, i.e., stylized facts, including volatility clustering and fat tails.

*Both authors contributed equally to this research.

Conference'17, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Keywords

Synthetic Data Generation, Financial Time Series, Benchmarking

ACM Reference Format:
Uyen Lam Ho and Eddison Pham. 2025. Stonk Bench: Unified Benchmark for Synthetic Data Generation for Financial Time Series. In . ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Synthetic Data Generation (SDG) has emerged as a crucial tool in financial technology, particularly for Financial Time Series (FTS) data [17]. The ability to generate high-quality synthetic financial data addresses several critical challenges in the field, like replicating and simulating fat-tail behaviors observed in financial returns time series and developing back-testing procedures for prediction and (deep) hedging algorithms [20].

Despite the growing importance of SDGFTS (Synthetic Data Generation for Financial Time Series), there is a notable lack of standardization in evaluating the quality and effectiveness of these generative models [20]. Current evaluation methods vary widely across studies, making it difficult to compare different approaches objectively and determine their relative strengths and weaknesses.

1.1 Limitations in the SDGFTS Literature

Among other things, we observed that there is currently no universal, generally accepted approach to evaluating synthetic time series [20]; this issue extends beyond FTS to generative frameworks like GANs as a whole [22]. Many evaluation measures are insufficiently defined and lack public implementations, making reuse

and reproduction troublesome and error-prone [20]. This presents a challenge unique to the generation task compared to areas like time series forecasting or classification [20]. Hence, future research would immensely benefit from a widely accepted, reasonably sized set of qualified measures for the central evaluation criteria. Here we outline observations on the current limitations in SDGFTS research and evaluation practices:

- **[L1] Inconsistent Evaluation Metrics:** Different studies employ varying metrics, hindering direct comparisons between models. For instance, some focus on statistical similarity, while others emphasize utility in downstream tasks [20].
- **[L2] Lack of Standardized Datasets and Preprocessing:** The absence of common benchmark datasets leads to evaluations on disparate data, making it challenging to assess model performance uniformly [20] in terms of both asset classes, data granularity, and data length. Furthermore, preprocessing steps (e.g., normalization, windowing) vary widely, affecting results and complicating comparisons.
- **[L3] Narrow Scope of Evaluation:** Many evaluations concentrate on a narrow set of criteria, such as marginal distributions, neglecting other important aspects like temporal dependencies and generation diversity [20].
- **[L4] Reproducibility Issues:** The lack of open-source implementations and detailed evaluation protocols impedes reproducibility and validation of results across different studies [17, 20]. It is important to note that most model papers did not provide code for their evaluations.

1.2 Our Contributions

To put it in simple terms, we are adopting the time series evaluation taxonomy outlined from Stenger et al. [20], develop a comprehensive and unified SDG benchmark expanded from the works of Ang et al. [4] in specifics to FTS by incorporating a utility evaluation framework inspired by Boursin et al. [6] through portfolio evaluations generated by a (deep) hedger. Our key contributions include:

- **[C1] Consolidated Evaluation Framework:** In effort to address **[L1]**, we systematically review and consolidate evaluation methods from leading papers (models and surveys) in SDG and financial time series [4, 6, 20], creating a comprehensive assessment toolkit based on established practices.
- **[C2] Unified Statistical and Utility Measures:** For the first time, we integrate both statistical fidelity metrics and practical utility measures in a single SDGFTS benchmark, providing a more complete evaluation of synthetic data quality.
- **[C3] Outline Benchmark Framework:** We deliver an open-source benchmark framework to share with the research community, aiming to establish cohesion and standard for future SDGFTS research in evaluation and comparison.

1.3 Research Questions

Our benchmark aims to address the following research questions:

- **[Q1] How do different sequence lengths during training affect STGFTS performance?** We investigate the impact of varying training sequence lengths on the performance of synthetic financial time series generation as an extension of existing work like Allen et al. [3]. The selected training sequence lengths are described in §4.1.4.
- **[Q2] Is there a trade-off between certain metrics or categories of metrics in SDGFTS?** In an effort to incorporate an ensemble of evaluation metrics, we are interested in understanding potential trade-offs between these metrics and how they impact the overall performance of SDG [4]. Particularly, we are interested in how a SDGFTS model's ability to balance fidelity and diversity as highlighted in recent studies [16]. Another obvious trade-off to investigate is between models' generation efficiency and the quality of generated data.
- **[Q3] Whether new deep learning models actually outperform classical parametric models in SDGFTS tasks?** Most existing literature focuses on deep learning-based methods tend to dominate the field. We aim to provide a comprehensive comparison between deep learning-based approaches and traditional methods, focusing on their respective strengths and weaknesses in generating high-quality synthetic financial time series data.

2 Preliminaries

2.1 Problem Definition

Let us formally define the Synthetic Data Generation problem for Financial Time Series. Given a financial time series $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_R)^T$ with R individual series represented as an L-dimensional vector $\mathbf{x}_i = (x_{i1}, ..., x_{iL})$, where x_{ij} denotes the j-th point of time series $\mathbf{x}_i$. Let $p(\mathbf{x}_1, ..., \mathbf{x}_R)$ denote the real distribution of the given time series $\mathbf{X}$. The objective of SDG for FTS is to generate a synthetic time series $\hat{\mathbf{X}} = (\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)^T$ such that its distribution $p(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)$ approximates $p(\mathbf{x}_1, ..., \mathbf{x}_R)$, while preserving the key properties exhibited in the real financial time series, such as autocorrelation structure, volatility clustering, and distributional moments (mean, variance, skewness, kurtosis).

2.2 Scope of Project

2.2.1 Scope of Methods. Our benchmark encompasses a diverse range of SDGFTS methods, from traditional parametric approaches to modern deep learning architectures. We selected models based on three main criteria: (1) proven industry adoption, (2) research community acceptance, and (3) recent methodological innovations.

Our evaluation includes classical parametric models, which rely on predefined statistical distributions and assumptions about the underlying data generation process. These models have been extensively used in financial institutions for their interpretability and theoretical foundations.

We also evaluate modern deep learning approaches, particularly deep learning-based models that learn the data distribution directly from observations without assuming a specific form. These include generative adversarial networks, and variational autoencoders, representing the cutting edge in synthetic data generation.

2.2.2 Scope of Datasets. We evaluate models on high-granularity financial time series indices prices from HistData.com [1]. Currently, we focus on univariate time series data generation. The S&P 500 (SPXUSD) index is chosen, using the minute-level closing price

from 2021 through 2023. Future work may extend to multivariate series involving multiple assets or additional features.

2.2.3 Scope of Evaluation Taxonomical Criteria. Our evaluation framework follows the taxonomy structure proposed by Stenger et al. [20], incorporating various evaluation measures from different papers in the field.

We systematically integrate evaluation metrics from multiple sources while maintaining this structured taxonomical approach, ensuring comprehensive coverage of all critical aspects of SDGFTS evaluation and setting a standard for future research in time series SDG as a whole.

3 Overview of Synthetic FTS Methods

We categorize synthetic financial time series generation methods into two main families: classical stochastic (parametric) models, which rely on explicit assumptions about the data-generating process, and modern deep learning (deep learning) models, which learn complex dependencies directly from data without predefined functional forms. The following subsections provide a detailed overview of the methods included in our benchmark.

3.1 Classical Stochastic (Parametric) Methods

Parametric models assume a specific functional form for the data generating process, characterised by a finite set of parameters. These models are interpretable and analytically tractable, allowing for closed-form solutions in many cases. However, they may struggle to capture complex stylised facts or high-dimensional dependencies beyond their structural assumptions. Below we outline several widely-used parametric models in FTS.

Table 1: Common notation and variables.

Symbol	Description
S_t	Asset price
$X_t = \ln S_t$	Log-price
$r_t = \ln(S_t/S_{t-1})$	Log-return
μ	Drift
σ	Volatility
W_t	Standard Brownian motion
Δt	Discrete time step

[A1] Geometric Brownian Motion (GBM). The price process follows a continuous-time stochastic process with constant drift and volatility. Model-specific parameters are μ and σ.

$$dS_t = \mu S_t\, dt + \sigma S_t\, dW_t. \tag{1}$$

[A2] Ornstein–Uhlenbeck (OU). A mean-reverting Gaussian process for log-prices or spreads. Model-specific parameters are $\theta > 0$ (reversion rate), μ (long-term mean), and σ (volatility).

$$dX_t = \theta(\mu - X_t)\, dt + \sigma\, dW_t. \tag{2}$$

[A3] Merton Jump Diffusion (MJD). Extends the GBM equation (1) by incorporating normally-distributed jumps. Model-specific parameters are λ (jump intensity), μ_j and σ_j (mean and standard deviation of jump sizes).

$$\frac{dS_t}{S_t} = \mu\, dt + \sigma\, dW_t + J\, dN_t, \tag{3}$$

where $J \sim \mathcal{N}(\mu_j, \sigma_j^2)$, $\quad N_t \sim \text{Poisson}(\lambda t)$.

[A4] Double-Exponential Jump Diffusion (DEJD). Another jump-diffusion model with asymmetric double-exponential jump sizes modifying equation (1). Model-specific parameters are p (probability of upward jump) and η_1, η_2 (exponential parameters for upward/downward jumps).

$$\frac{dS_t}{S_t} = \mu\, dt + \sigma\, dW_t + Y\, dN_t, \tag{4}$$

$$Y = \begin{cases} \text{Exp}(\eta_1), & \text{with probability } p, \\ -\text{Exp}(\eta_2), & \text{with probability } 1-p. \end{cases}$$

[A5] GARCH(1,1). A discrete-time conditional heteroskedastic model for returns. Model-specific parameters are $\omega > 0$, $\alpha \geq 0$, $\beta \geq 0$ (GARCH coefficients) and $\mathcal{D}$ (innovation distribution, usually standard Normal or Student-t). Captures volatility clustering via time-varying conditional variance.

$$r_t = \sigma_t z_t, \quad z_t \sim \mathcal{D}, \tag{5}$$
$$\sigma_t^2 = \omega + \alpha r_{t-1}^2 + \beta \sigma_{t-1}^2.$$

3.2 Deep Learning Methods

Deep learning models, in this context, refer to generative (often implicit) models that do not assume a fixed parametric form for the underlying data-generating process. Instead, they learn latent structures directly from data through flexible architectures such as neural networks, enabling the modeling of highly nonlinear, high-dimensional, and multi-modal dependencies (e.g., multivariate financial time series with interactions across assets, time horizons, and features). While these models trade off interpretability and analytical tractability, they offer substantially greater expressive power and flexibility. This expressiveness, however, comes at the cost of increased data requirements, more intensive hyperparameter tuning, and higher computational demands.

We categorize these approaches into two major families of deep generative models, considered in this paper: Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs).

3.2.1 Variational Autoencoders (VAEs). VAEs [12, 19] are latent-variable generative models that posit a probabilistic encoder $q_\phi(z \mid x)$ and decoder $p_\theta(x \mid z)$. One maximizes the evidence lower bound (ELBO):

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q_\phi(z|x)}\big[\log p_\theta(x \mid z)\big] - \text{KL}\big(q_\phi(z \mid x) \,\|\, p(z)\big).$$

Often one uses a standard Gaussian prior $p(z) = \mathcal{N}(0, I)$. The encoder–decoder structure allows sampling by first drawing $z \sim p(z)$, then generating $x \sim p_\theta(x \mid z)$.

In financial time-series generation, specialized versions such as the Time-Causal VAE (TC-VAE) have been proposed to enforce causality in the encoding/decoding of temporal data. For instance, Acciaio et al. (2024) propose TC-VAE, which ensures that generated

paths respect temporal causality and show that the model reproduces stylized facts (e.g., heavy tails, volatility clustering) on real markets [2]. Because VAEs provide a principled latent representation, they are useful in finance to model underlying latent drivers of asset dynamics and to sample coherent trajectories.

[A6] TimeVAE. A time-series specific VAE variant that combines causal temporal encoders (e.g., causal convolutions or RNNs) with an autoregressive decoder to preserve path coherence. Training typically augments the ELBO with prediction or stepwise reconstruction losses to improve short- and long-range dependencies; TimeVAE is used in our benchmark as a dedicated VAE baseline for temporal data [8].

3.2.2 Generative Adversarial Networks (GANs). GANs models approach generation as a two-player game between a generator G and a discriminator D [11]. The discriminator is trained to maximize the probability of correctly classifying real data and generated data, with loss

$$L_D = -\mathbb{E}_{x\sim p_{\text{data}}}\left[\log D(x)\right] - \mathbb{E}_{z\sim p(z)}\left[\log(1 - D(G(z)))\right].$$

The generator aims to fool the discriminator and is trained to minimize

$$L_G = -\mathbb{E}_{z\sim p(z)}[\log D(G(z))].$$

Many GAN variants, such as Wasserstein GAN or Least Squares GAN, modify these loss functions to promote more stable training and address issues like mode collapse [11] .

In financial data synthesis, GANs have been widely used to generate realistic synthetic time series. For example, GAN-based financial data generation models (often using WGAN or improvements) show that one can improve the authenticity and predictive ability of generated financial statements or time series [18]. Another example is VRNNGAN, which uses a recurrent VAE as the generator and a recurrent discriminator to capture temporal dependencies in synthetic sequence generation [13]. GANs are relevant in finance because they can capture complex joint distributions and dependencies in multivariate time-series without requiring explicit likelihood models.

[A7] QuantGAN. A GAN variant tailored for financial applications that incorporates quantile-aware objectives or conditional quantile matching and temporal architectures (e.g., temporal convnets or transformers). QuantGANs aim to better capture tail behaviour and risk-sensitive statistics (e.g., VaR/CVaR) important for downstream financial tasks [23].

3.3 Miscellaneous

This section consists of models that does not fall in the classification of §3.1 or § 3.2.

[A8] Block Bootstrap (Resampling Method). A resampling technique that preserves short-range dependence by sampling contiguous blocks of returns. Although not parametric, it is included here for completeness and baseline comparison.

4 Stonk Bench Architecture

4.1 Datasets and Preprocessing

The preprocessing procedure builds upon the TSGBench pipeline, which standardizes segmentation, normalization, and train/test splitting of the time series [4]. Several modifications were introduced to better accommodate financial time series and stochastic models.

4.1.1 Data type. Index price data at minute frequency at closing price are used. Let $X \in \mathbb{R}^{L\times N}$ denote a time series with $N = 1$ channel(s) and length L.

4.1.2 Transformations. Raw prices are converted to log-returns per channel via

$$r_t = \log S_t - \log S_{t-1} = \log(S_t/S_{t-1}), \quad r_t \in \mathbb{R}^N,$$

yielding a transformed series of length $L - 1$. Log returns are preferred in finance due to their variance-stabilizing properties, preserving heavy tails and volatility clustering, and removal of the first order integration (random walk) commonly exhibited in price series [21].

4.1.3 Sequence lengths specification. To address research question **[Q1]**, we experiment with varying training sequence lengths L. For deep learning models, overlapping sliding windows $\mathcal{W} \in \mathbb{R}^{R\times L\times N}$ with stride 1 are extracted. Specifically, we consider the following lengths L: 60, 120, 240, 300 minutes, and the maximum significant partial autocorrelation lag.

Unlike TSGBench, which employs an autocorrelation-based hard-coded window of 125 (often yielding a size of 1 for financial returns due to fast autocorrelation decay [7, 21]), the window length L is alternatively determined via the partial autocorrelation function (PACF). The PACF aims to remove indirect correlations from in-between lags due to auto-regressive structures by finding the best linear-predictor of the intermediate lagged values [9]. Particularly for a lag h, the PACF ϕ_{hh} is defined between X_t and X_{t+h} as

$$\phi_{hh} = \operatorname{corr}\big(X_t - \hat{X}_t, X_{t+h} - \hat{X}_{t+h}\big),$$

where $\hat{X}_t, \hat{X}_{t+h}$ is the best linear predictor of X_t, X_{t+h},respectively, given the intermediate lags $X_{t+1}, \ldots, X_{t+h-1}$. Specifically, PACF is computed up to $\lfloor 10 \log_{10} n \rfloor$ lags (this value is determined from TSGBench [4]), and the lag corresponding to the maximum significant PACF peak across channels is selected, resulting in $L = 52$. This approach captures relevant temporal dependencies within the data.

4.1.4 Model-specific preprocessing. The preprocessing follows the general TSGBench framework, with the following adaptations:

- Log returns are used directly instead of TSGBench's min-max normalization for the reasons outlined in §4.1.2.
- For parametric models, the transformed series is divided into contiguous training, validation, and test segments with ratios $(1 - \alpha - \beta) : \alpha : \beta$, where $\alpha = 0.1$ and $\beta = 0.1$. This ensures that parametric models fit parameters directly to contiguous historical data.
- The dataset is split into train, validation, and test sets while preserving temporal order to prevent future information from leaking into the past. Only the training set is shuffled during model training to improve convergence. In contrast, TSG-Bench shuffles time series samples before splitting, which introduces data leakage. Our approach ensures strictly forward-looking evaluation for realistic forecasting.
  proper evaluation, consistent with TSGBench.

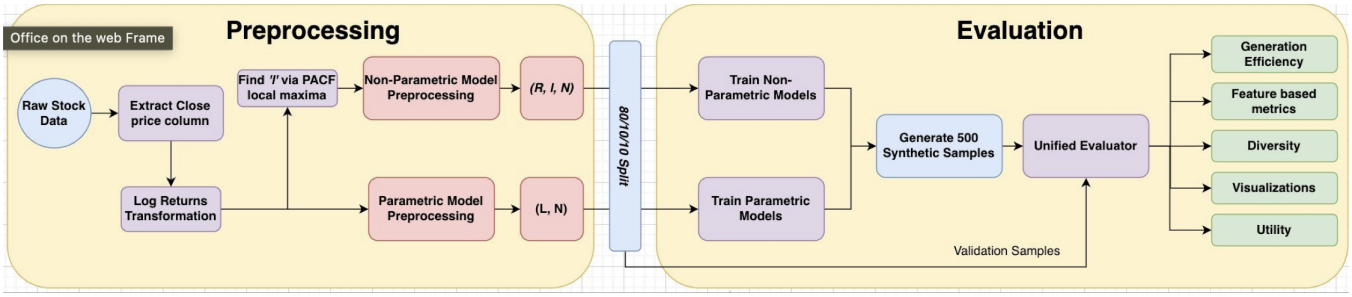

Figure 2: Stonk Bench system architecture overview.

4.1.5 Data loaders. For deep learning models, mini-batches are generated from the windowed data $\mathcal{W}$ using independent fixed seeds for training, validation, and test sets, enabling reproducible epoch-wise evaluation. Training batches are shuffled, while validation and test sets maintain sequential ordering. To ensure fair comparison, parametric models generate sequences of identical length L and matching sample count to the windowed data used by deep learning models.

4.2 Statistical Evaluation Measures

All metrics below are applied **per channel**, i.e., each univariate time series (OHLC column) is evaluated independently. For multivariate data, results are reported as channel-wise statistics (e.g., mean or distribution across channels).

4.2.1 Fidelity Measures. These metrics quantify how well synthetic data captures key marginal and second-order properties of real data. We employ the following feature-based metrics:

[M1] Marginal Distribution Difference (MDD): To measure the distance between empirical marginal distributions of real versus synthetic data, we use the 1-Wasserstein distance (a.k.a. Earth Mover's distance). This was chosen over the popular Kullback-Leibler divergence or the derivatives Jensen-Shannon divergence due to its better numerical stability and ability tocapture distributional differences even when supports do not overlap [5].

$$\mathrm{MDD} = W_1\big(\widehat{F}_{\mathrm{real}}, \widehat{F}_{\mathrm{synth}}\big),$$
$$W_1(F,G) = \int_0^1 \left|F^{-1}(u) - G^{-1}(u)\right| du,$$

where $\widehat{F}_{\mathrm{real}}$ and $\widehat{F}_{\mathrm{synth}}$ are empirical CDFs.

The following four metrics capture differences in the first four statistical (central) moments between real and synthetic data.

[M2] Mean Difference (MD):

$$\mathrm{MD} = \left|\mu_{\mathrm{real}} - \mu_{\mathrm{synth}}\right|.$$

[M3] Standard Deviation Difference (SDD):

$$\mathrm{SDD} = \left|\sigma_{\mathrm{real}} - \sigma_{\mathrm{synth}}\right|.$$

[M4] Skewness Difference (SD):

$$\mathrm{SD} = \left|\gamma(r_{\mathrm{real}}) - \gamma(r_{\mathrm{synth}})\right|,$$
$$\gamma(r) = \frac{\mathbb{E}[(r-\mu)^3]}{(\sigma)^3}.$$

[M5] Kurtosis Difference (KD):

$$\mathrm{KD} = \left|\kappa^{\mathrm{ex}}(r_{\mathrm{real}}) - \kappa^{\mathrm{ex}}(r_{\mathrm{synth}})\right|,$$
$$\kappa^{\mathrm{ex}}(r) = \frac{\mathbb{E}[(r-\mu)^4]}{(\sigma)^4} - 3.$$

4.2.2 Visualization Methods. Visual tools complement numeric metrics for face-validity and communication [4].

[M6] t-SNE: 2D embeddings of window-level features for real versus synthetic overlap and cluster structure.

[M7] Distribution: Histograms of returns to visualize empirical distribution.

4.2.3 Diversity Measures. We are also interested if the SDGFTS can generate data not observed but follow the same distribution as the real data. Therefore, we include the following intra-class distance metrics. The inclusion of these metrics are chose to ensure that mode collapse is prevented.

[M8] Intra-Class Euclidean Distance (ICD-ED):

$$\text{ICD-ED} = \frac{2}{R(R-1)} \sum_{1 \le a < b \le R} \|\mathbf{r}_a - \mathbf{r}_b\|_2,$$

where $\mathbf{r}_a \in \mathbb{R}^L$ is the a-th sample.

[M9] Intra-Class Dynamic Time Warping (ICD-DTW):

$$\text{ICD-DTW} = \frac{2}{R(R-1)} \sum_{1 \le a < b \le R} d_{\mathrm{DTW}}(\mathbf{r}_a, \mathbf{r}_b).$$

4.2.4 Efficiency Measures. Generation efficiency is crucial for practical applications, especially in high-frequency trading or real-time risk management, where rapid data synthesis is required. Thus, we include the following efficiency metric [20].

[M10] Generation Time: Wall-clock time to generate S samples:

$$T_{\mathrm{gen}} = t_1 - t_0 \quad \text{(seconds for 1,000 samples)}.$$

4.2.5 Stylized Facts Measures. Canonical stylized facts:

[M11] Autocorrelation Difference (ACD): Absolute difference in lag-1 autocorrelation:

$$\rho(1) = \frac{\sum_{t=2}^{L}(r_t - \bar{r})(r_{t-1} - \bar{r})}{\sum_{t=1}^{L}(r_t - \bar{r})^2},$$
$$\text{ACD} = \left|\rho_{\text{real}}(1) - \rho_{\text{synth}}(1)\right|.$$

[M12] Volatility Clustering (VC): Lag-1 autocorrelation of squared or absolute returns measures volatility clustering, a stylized fact where large price movements tend to cluster in time [7]:

$$\rho_{r^2}(1) = \text{Corr}\big((r_t)^2, (r_{t-1})^2\big),$$
$$\rho_{|r|}(1) = \text{Corr}\big(|r_t|, |r_{t-1}|\big).$$

[M13] Long Memory / Slow Decay (LMSD): To quantify long-range dependence in volatility, we follow Cont (2001) [7] and study the autocorrelation of absolute (or squared) returns:

$$C_\alpha(\tau) \;=\; \text{corr}\big(|r_{t+\tau}|^\alpha,\, |r_t|^\alpha\big), \qquad \alpha \in \{1, 2\}.$$

Empirically, this correlation decays slowly according to a power law,

$$C_\alpha(\tau) \;\sim\; \frac{A}{\tau^\beta}, \qquad \beta \in [0.2,\, 0.4],$$

indicating persistent volatility fluctuations (long memory). For evaluation, we estimate the decay exponent by a log–log linear fit over lags $\tau \in \{1, \ldots, \tau_{\max}\}$ where $C_\alpha(\tau) > 0$:

$$\widehat{\beta} \;=\; -\,\text{slope}\Big(\log C_\alpha(\tau) \text{ vs. } \log \tau\Big).$$

Our metric is the absolute difference between decay exponents estimated on real and synthetic data:

$$\text{LMSD} \;=\; \left|\widehat{\beta}_{\text{real}} - \widehat{\beta}_{\text{synth}}\right|.$$

Unless otherwise stated, we set $\alpha = 1$ (absolute returns) and report $\widehat{\beta}$ together with the goodness-of-fit of the linear regression.

4.3 Utility Evaluation Measures: (Deep) hedging

Utility measures are critical for evaluating synthetic data beyond statistical metrics, as they assess the practical value of generated data in real-world financial applications [6]. Our benchmark incorporates (deep) hedging as a utility measure for several key reasons:

- **[U1] Real-world Application Testing:** (Deep) hedging provides a concrete way to evaluate how synthetic data performs in actual financial tasks, particularly in derivatives pricing and risk management.
- **[U2] Industry-relevant Metrics:** By comparing hedging strategies trained on synthetic versus real data, we can assess the practical utility of synthetic data through metrics that matter to financial practitioners, such as replication errors and hedging performance.
- **[U3] Model Robustness Validation:** (Deep) hedging helps verify if synthetic data maintains the complex relationships and market dynamics necessary for developing reliable trading strategies.

4.3.1 (Deep) hedger Problem Defintion. We consider a continuous-time financial market defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, over a finite time horizon $0 < T < \infty$, equipped with a filtration $\mathcal{F} = (\mathcal{F}_t)_{0 \le t \le T}$ that represents the evolution of information through time. The market consists of asset S_t. For simplicity, we assume a zero interest rate environment. The asset S_t follow a stochastic differential equation. We focus on a single underlying asset S_t, and we are concerned with hedging a European call option with strike price K and maturity T so the payoff is $g(S_T) = \max(S_T - K, 0)$.

We study the hedging problem associated with a contingent claim delivering a payoff $g(S_T)$ at maturity T, where S_T represents the terminal value of the underlying asset vector. The hedging strategy is implemented at a discrete set of times $0 = t_0 < t_1 < \cdots < t_{N-1} < t_N = T$. A *self-financing portfolio* is described by a d-dimensional $\mathcal{F}_t$-adapted process Δ_t, and its terminal wealth, denoted $X_T^{\Delta,p}$, is given by

$$X_T^{\Delta,p} = p + \sum_{i=1}^{d}\sum_{j=0}^{N-1} \Delta_{t_j}^i \left(F_{t_{j+1}}^i - F_{t_j}^i\right), \tag{6}$$

where $p \in \mathbb{R}$ represents the initial premium.

The objective is to determine the optimal premium and trading strategy $(p^{\text{opt}}, \Delta^{\text{opt}})$ that minimize the expected squared hedging error:

$$(p^{\text{opt}}, \Delta^{\text{opt}}) = \underset{p,\Delta}{\text{Argmin}}\, \mathbb{E}\left[\left(X_T^\Delta - g(S_T)\right)^2\right]. \tag{7}$$

To address this optimization problem, we employ the global approach proposed by Fécamp *et al.* (2020), chosen for its computational efficiency and scalability. In this framework, the control policy Δ is approximated by a feed-forward neural network—referred to as a *deep hedger*—which jointly learns the optimal trading strategy and corresponding premium through end-to-end training [10].

Table 2: Notations for (deep) hedger architecture.

Symbol	Description
D	Time-series dataset
T	Task
S	Score
A	(deep) hedger algorithm
$\mathcal{A}_n$	Set of n (deep) hedger algorithms
M	(deep) hedger model
$\mathcal{M}_{TS}$	(SDGFTS) model

4.3.2 (Deep) hedger Architecture. Our (deep) hedger architecture centers around the Train-Synthetic-Test-Real (TSTR) evaluation framework [14], adapted for financial time series generation. We begin with our real FTS dataset D—partitioned to a training, validation, and test set respectively as $D_r = \{D_r^{train}, D_r^{validate}, D_r^{test}\}$—along with a specific hedging task T and a performance score S to evaluate hedging effectiveness, to which we will go indepth in the next subsubsection. We consider a (deep) hedger algorithm A (e.g. a 5-layer perceptron) that takes as input the dataset D_r and task T,

In the context of SDGFTS evaluation, we consider a SDGFTS model $\mathcal{M}_{TS}$ trained on D_r^{train} and validated with $D_r^{validate}$ to generate synthetic FTS data D_g.

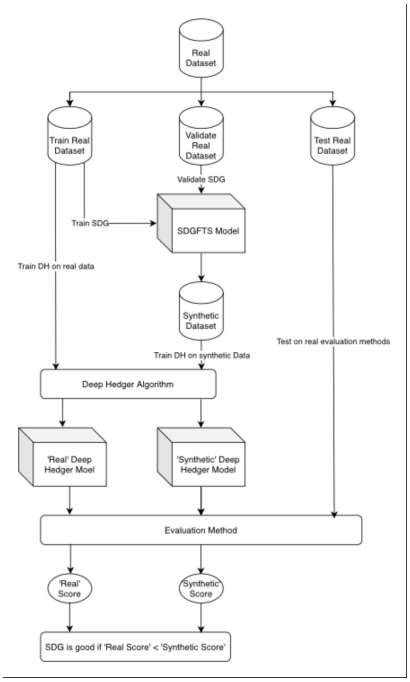

Figure 3: Architecture overview of the (deep) hedger evaluation framework, illustrating the flow from data preparation through model training to performance evaluation.

Now we can train a (deep) hedger algorithm A on the datasets D_r^{train} and D_g^{train} respectively, resulting in two (deep) hedger models M_r and M_g.

Finally, we evaluate both models on the real test set D_r^{test} to obtain the corresponding performance scores s_r and s_g.

The goal of the (deep) hedger portfolio evaluation is to evaluate the effectiveness of the model $\mathcal{M}_{TS}$ with downstream tasks (hedging) by comparing the scores s_r and s_g. The model $\mathcal{M}_{TS}$ is considered effective if the the results yielded similar scores, i.e. $s_r \approx s_g$ —preferably having higher performance with results of $s_g > s_r$ —indicating that the synthetic data generated by $\mathcal{M}_{TS}$ is useful for training a (deep) hedger.

4.3.3 Augmented Testing. To better evaluate the performance of (deep) hedgers trained on synthetic data, we train our (deep) hedger A on a new dataset D_{aug} that combines both real training data D_r^{train} and synthetic data D_g. In other words, we train model M_g on D_{aug} where $D_{aug} := D_r^{train} \cup D_g$. Then we proceed to evaluate scores between models M_g and M_r on the real test set, where M_r is still trained on only real data D_r^{train}. This extension, inspired by Stenger et al. [20], allows us to assess whether synthetic data provides additional value when combined with real training data.

4.3.4 Algorithm Comparison. Another extension is to compare multiple (deep) hedger algorithms to determine the degree to which each algorithm performs equally on generated data relative to other algorithms, compared to their performance on real data.

Given a set of n (deep) hedger algorithms $\mathcal{A}_n = \{A_1, \ldots, A_n\}$, we train each algorithm $A_i \in \mathcal{A}_n$ on both real training data D_r^{train} and synthetic data D_g^{train} respectively, resulting in two sets of (deep) hedger models $\{M_r^1, M_r^2, \ldots, M_r^n\}$ and $\{M_g^1, M_g^2, \ldots, M_g^n\}$ and two ordered sets of performance scores $s_r := \{s_r^1, s_r^2, \ldots, s_r^n\}$ and $s_g := \{s_g^1, s_g^2, \ldots, s_g^n\}$ respectively.

We compare the relative performance of each algorithm on real data versus synthetic data by computing the Spearman's rank correlation coefficient r_S [15] between the two score sets s_r and s_g:

$$r_S = 1 - \frac{6 \sum_{i=1}^{n} (\text{rank}(s_r^i) - \text{rank}(s_g^i))^2}{n(n^2 - 1)}, \tag{8}$$

where $\text{rank}(s_r^i)$ and $\text{rank}(s_g^i)$ denote the ranks of scores s_r^i and s_g^i within their respective sets.

A high positive correlation (i.e., r_S close to 1) indicates that the synthetic data generated by $\mathcal{M}_{TS}$ preserves the relative performance of different (deep) hedger algorithms, suggesting that the synthetic data is suitable for algorithm evaluation and comparison.

The (deep) hedger algorithms $\mathcal{A}_n$ considered in this benchmark can be found in appendix A.2.

4.3.5 (Deep) hedger Portfolio Evaluation. We evaluate the suite of hedger algorithms trained on synthetic and real data using the portfolio replication error or replication loss metric [6].

5 Results and Analysis

We present a concise, comprehensive evaluation of the synthetic financial time series produced by each model. The analysis covers marginal and second-order statistics, temporal dependencies, diversity measures, stylized-facts verification, and generation efficiency. Results are reported on the minute-level closing price and summarized in the tables and figures that follow: fidelity metrics, diversity metrics, stylized-fact comparisons, and generation-time measurements. Best-performing entries are highlighted where appropriate and comparative discussion appears in the subsequent analysis subsections.

In §5.1 and §5.2, results displayed and analysis are presented based on SDGFTS traingin with sequenc length determined by the maximum significant PACF lag (§4.1.4).

5.1 Statistical Evaluation Results

The following 2 sections are basesd on the results summarized in figure 4.

5.1.1 Stylized Facts and Fidelity. The block bootstrap **[A8]** consistently achieves the strongest performance across both distributional metrics and stylized facts. This outcome is expected: by resampling contiguous blocks from the training data, the block bootstrap directly inherits empirical marginal distributions **[M1]** and short- to medium-range dependence structures. As a result, it naturally preserves key stylized facts (e.g., heavy tails **[M5]**, volatility clustering **[M12]**) as long as they are present within the chosen block length.

Among parametric models, GARCH **[A5]** performs second best overall. This is consistent with its widespread use in financial time series modeling: GARCH is explicitly designed to capture volatility clustering and volatility persistence. While standard GARCH models do not exhibit true long-memory behavior (in the strict statistical sense), they nonetheless approximate slowly decaying volatility autocorrelations well enough to score favorably on stylized-fact

Figure 4: Normalized ranking heatmap across all evaluation taxonomies. For diversity metrics, a score of (1) indicates highest metric value (most diverse), while for other metrics, a score of (1) indicates lowest metric value (best performance). Raw minimum and maximum values are provided for each metric on top of each respective column. (Specification: sequence length based on maximum significant PACF lag)

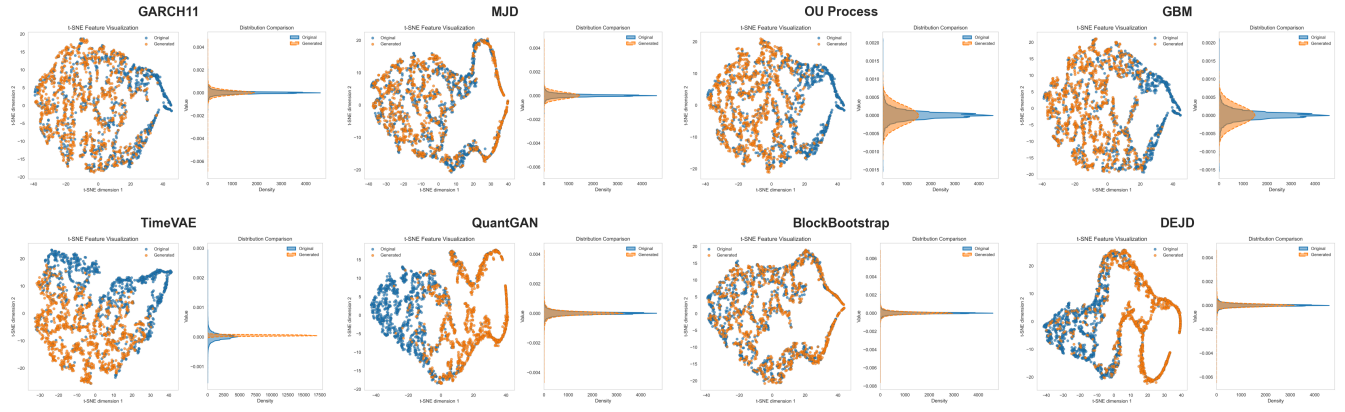

Figure 5: t-SNE visualization and distribution plots comparing real and synthetic financial time series data across models. (Specification: sequence length based on maximum significant PACF lag)

metrics relative to simpler parametric models such as OU process and GBM.

The DEJD **[A4]** model underperforms on kurtosis- based metrics **[M5]** despite its ability to generate fat tails. Although the jump component introduces excess kurtosis relative to pure diffusion processes, the model is not inherently calibrated to reproduce empirical tail thickness precisely. Its weaker performance likely reflects limitations in parameter estimation and mismatch between assumed jump dynamics and the empirical return distribution.

Volatility models such as GBM **[A1]** and the OU process **[A2]** perform poorly on stylized facts (§ 4.2.5). This result is unsurprising: both models assume conditionally Gaussian increments with constant (GBM) or mean-reverting (OU) volatility, and therefore cannot reproduce volatility clustering or persistent heteroskedasticity observed in real financial time series.

A notable result is that QuantGAN performs significantly worse on distributional metrics while capturing stylized facts relatively well. This suggests that the adversarial objective is effective at matching temporal dependence and higher-order dynamics, but less effective at aligning marginal distributions without explicit distributional constraints. TimeVAE performs poorly overall, which is consistent with posterior collapse when trained on log returns—limiting its ability to generate meaningful variability and capture higher-order financial structure.

5.1.2 Efficiency. Efficiency results largely follow model complexity. GBM, as the simplest model both conceptually and computationally, is the fastest to train and sample from. In contrast, QuantGAN is the slowest by design, reflecting the cost of adversarial training and high-capacity neural architectures.

5.1.3 Visualization. Figure 5 presents t-SNE embeddings and return distribution plots (see §4.2.2) for real versus synthetic data across models. The block bootstrap **[A8]**shows near-perfect overlap in t-SNE space, reflecting its direct resampling from real data. QuantGAN **[A7]** and DEJD **[A4]** also shows better distribution alignment, but does not full capture the clusters in the t-SNE plots.

While other parametric models do not capture these clusters fully, they generally corroborate the quantitative findings: GARCH **[A5]** shows better distributional alignment than GBM **[A1]** and OU process **[A2]**, while TimeVAE **[A6]** exhibits the worst overlap and distributional mismatch, as it appears to suffer from posterior collapse.

5.1.4 Diveristy metrics. Diversity alone is ambiguous: high variability does not guarantee realism and may violate structural dependencies. We therefore evaluate diversity jointly with fidelity to ensure variability remains plausible. We revisit this in research question **[Q2]** and in §5.4.

5.2 Utility Evaluation Results

As see in Figure 6, for the augmented testing 4.3.3, we can make the observation that for the hedging models, the neural network models generally perform better than the parametric models in estimating the deltas. This is expected as the neural network models are able to capture the higher-order dynamics of the index series, which the parametric models are not able to capture [10].

In the augmented-testing setting, with algorithms compared across datasets (see §4.3.4), we observe that the rank-normalized values of the heatmaps obtained using QuantGAN **[A7]** and BlockBootstrap **[A8]** remain highly consistent with those derived from real-only data. In particular, rank-normalized hedging performance under QuantGAN augmentation closely matches the baseline ordering, while BlockBootstrap augmentation yields nearly identical results. This behavior is expected for BlockBootstrap, which preserves empirical dependence structures by construction, and suggests that QuantGAN-generated samples do not distort the hedger-relevant dynamics when mixed with real data. Thus, both augmentation strategies maintain the relative sensitivities and preferences of hedger models observed under real-data training, indicating that these synthetic generators can be used for augmentation without materially altering hedging-based performance assessments.

For Spearman correlation (see Equation 8), recommended by Stenger et al., it's important to note that this metric considers only the ranking of model performances, completely ignoring the actual delta values between them [20]. This can be problematic: for example, TimeVAE **[A6]** —despite suffering from mode collapse and generating less realistic outputs —may still achieve a high Spearman correlation if it preserves the order of model performances, even though the deltas may be substantially off or meaningless. Conversely, QuantGAN, which better captures the structure of financial data, might produce more realistic values but achieve a worse Spearman score if the ranking changes slightly. Thus, relying solely on Spearman correlation can be misleading, as it overlooks the magnitude and quality of differences between models and may obscure meaningful distinctions in model performance.

5.3 [Q1] Metric Response on Sequence Length Training

We investigate how training sequence length affects evaluation metrics as posed in research question **[Q1]**. As previous sections describe how sequence length is determined based on model-specific PACF analysis (§4.1.4) how they affect the Stonk Bench architecture.

Recall that the sequence lengths considered are 60, 120, 180, 240, and 300 minutes respectively.

We analyze each metric as the training sequence length varies from 60 to 300 minutes.

5.3.1 Fidelity. Rank-normalized MDD values are essentially invariant to sequence length, which is expected for a marginal distribution metric unless models generate time-varying distributions (not observed here). Skewness differences show minimal variation for GARCH(1,1), GBM, OU, and MJD, suggesting limited sensitivity to sequence length. Modest rank improvements for BlockBootstrap and DEJD likely reflect better capture of asymmetric tail events. Kurtosis differences for GARCH, MJD, OU, and GBM change marginally with sequence length, consistent with limited extreme-tail generation; BlockBootstrap improves with longer sequences by resampling rare extremes. See the fidelity ablation heatmap in Figure 8.

5.3.2 Diversity. Diversity rankings remain stable across sequence lengths, consistent with largely unchanged marginal distributions. However, the minimum and maximum diversity values increase with sequence length, as longer windows permit greater variability across time steps. See the diversity ablation heatmap in Figure 9.

5.3.3 Efficiency. Maximum generation time increases monotonically with sequence length, while model rankings remain unchanged, reflecting computational complexity rather than algorithmic reordering. See the efficiency ablation heatmap in Figure 11.

5.3.4 Stylized facts. The maximum values for lag-1 autocorrelation of absolute returns and volatility clustering increase with sequence length, whereas long memory measures tend to decrease. Rankings for QuantGAN, BlockBootstrap, and DEJD are largely preserved across lengths. See the stylized-facts ablation heatmap in Figure 10.

5.3.5 Spearman correlation. Spearman's rank correlation summarizes ordering only and can be misleading when magnitude differences matter; reliance on it alone is therefore discouraged. See the utility correlation comparison in Figure 12.

5.4 [Q2] Metric Tradeoffs and Interactions

We dedicate this subsection to answer the research question **[Q2]** regarding trade-offs and interactions between different evaluation metrics.

5.4.1 Diversity versus Fidelity. As introduced in §5.1.4, diversity metrics must be interpreted in conjunction with fidelity and stylized-fact metrics. How should we interpret these relationships? Recall that "difference" here means the discrepancy between real and synthetic data—the lower the value, the better the model is matching the real data.

The MDD **[M1]** and SDD **[M3]** both show a notably strong positive correlation with ICD Euclidean diversity **[M8]**. For example, MDD has a Pearson correlation coefficient of $r = 0.68$ ($p = 0.09$),

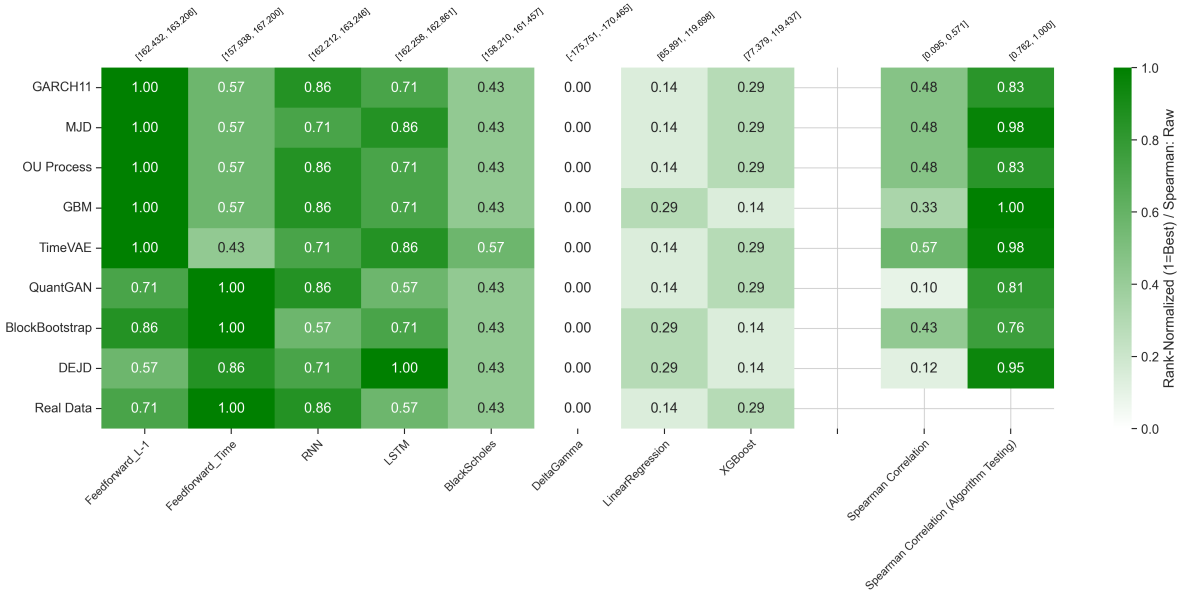

Figure 6: Normalized ranking heatmap for mean replication values of each hedger trained on SDGFTS/real data. Raw minimum and maximum values are provided for each metric column. (Specification: Augmented datasets and sequence length based on maximum significant PACF lag).

and STD difference has an even higher $r = 0.97$ (p = 1.90e-4) (see Figure 13). This indicates that as the marginal or standard deviation difference between real and synthetic data increases (i.e., as fidelity to the real data worsens), the diversity, as measured by ICD Euclidean, also tends to increase.

The findings indicate that increased diversity, as measured by ICD Euclidean, often correlates with reduced fidelity in basic distributional statistics. This suggests that models generating more diverse samples may deviate significantly from the real data distribution, highlighting the necessity of balancing diversity with fidelity. While some diversity is beneficial for generative models, excessive diversity can lead to less realistic outputs. Therefore, it is crucial to evaluate both fidelity and diversity metrics together when assessing generative time series models.

5.4.2 Diversity versus Stylized Facts. Figure 7 presents scatter plots illustrating the relationship between ICD Euclidean diversity **[M8]** and key stylized-fact metrics: ACD **[M11]**, volatility clustering **[M12]**, and LMSD **[M13]**.

Volatility clustering shows a weak positive correlation with diversity ($p \approx 0.1$). This suggests that more diverse models tend to preserve stronger volatility clustering, indicating that allowing a broader range of sample paths may help retain persistent volatility dynamics, though the evidence is marginal.

Long memory in volatility is also positively and mildly correlated with diversity. This implies that models producing more varied samples are more likely to exhibit persistent, slowly decaying volatility correlations, consistent with richer temporal dynamics emerging from increased generative flexibility.

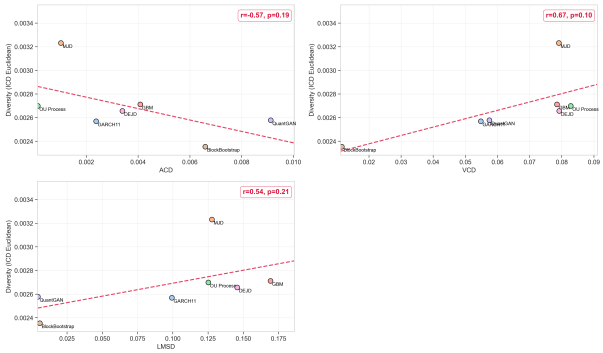

Figure 7: Scatter plots showing the relationship between ICD Euclidean diversity and stylized facts metrics (ACD, volatility clustering, and LMSD). Each point represents a model, with trend lines indicating correlation direction and strength.

5.4.3 Efficiency versus other metrics. Efficiency, measured by generation time, shows no significant correlation with fidelity, diversity, or stylized-fact metrics. This indicates that faster models do not necessarily compromise on quality or realism, nor do slower models guarantee better performance. Thus, efficiency appears to be largely independent of other evaluation dimensions in this benchmark.

6 Conclusion and Future Work

In our research, we recognize the existing limitations in the evaluation of synthetic time series, particularly the absence of a universally accepted framework. To address this, we adopt the evaluation

taxonomy proposed by Stenger et al. and expand upon it to create a comprehensive benchmark specifically tailored for Synthetic Data Generation for Financial Time Series (SDGFTS). Our contributions include the development of a consolidated evaluation framework that systematically reviews and integrates various assessment methods from leading studies in the field. This approach not only enhances the rigor of evaluations but also facilitates the comparison of different SDGFTS models, ultimately providing a more standardized and reliable means of assessing synthetic data quality.

We believe that Stonk Bench will serve as a valuable resource for researchers and practitioners in the field of synthetic financial time series generation. We hope that our benchmark will facilitate more rigorous and standardized evaluations of SDGFTS models, ultimately advancing the state of the art in this important area of research. For the broader field of synthetic data generation, our work highlights the importance of a comprehensive and unifying evaluation frameworks that consider both statistical properties and practical utility that is proposed by Stenger *et al.*.

6.1 Summary of Findings

We presented Stonk-Bench, a comprehensive benchmark for evaluating Synthetic Data Generation for Financial Time Series (SDGFTS). This benchmark integrates statistical fidelity, diversity, stylized facts verification, and utility evaluation through (deep) hedging. We evaluated a suite of 5 traditional, 3 deep learning-based, and 1 other SDGFTS models on the S&P 500 minute-level dataset.

6.2 Stonk Bench Shortcomings

While Stonk Bench represents a significant step forward in the evaluation of SDGFTS models, we acknowledge several limitations in our current work:

- **[S1]** **Incompatibility with other models:** Stonk Bench has so far been evaluated mainly on classical stochastic models (e.g., GBM, OU, GARCH) and a subset of deep-learning approaches (GANs, VAEs). Other model families — such as reinforcement-learning-based generators, normalizing flows, autoregressive networks, and hybrid methods — have not yet been benchmarked.
- **[S2]** **No hyperparameter tuning:** For consistency and comparability across models, StonkBench evaluations were performed using default model configurations, without performing hyperparameter optimization. This ensures a fair baseline but may not reflect the absolute best performance achievable by each model.
- **[S3]** **Metric and procedure generalizability:** As a consequence, some metrics or evaluation procedures may not directly translate to these untested families, which limits the framework's generalizability across all synthetic financial time-series generation methods.
- **[S4]** **Data Leakage from Public Datasets:** Current models can potentially overfit to publicly available datasets —the same datasets that Stonk Bench is using —leading to inflated performance metrics that do not generalize well to unseen data. One possible solution is to either curate proprietary datasets or implement a grace period so the model can be tested on new data that was not publicly available during its development.
- **[S5]** **Computational Resource Requirements:** The comprehensive nature of Stonk Bench —particularly accommodating any submitted (deep) hedging model and the deep hedging utility evaluation —demands significant computational resources. This may limit accessibility for researchers with constrained resources.

6.3 Future Research Directions

6.3.1 Remaining progress. In regard to the next portion of our project, we aim to expand Stonk Bench in several key areas.

- **[F1]** **Multivariate time series predictions.** Extend Stonk Bench to evaluate models on multivariate time series data, enabling a more comprehensive assessment of their performance in realistic scenarios. We wish test SDGFTS models on datasets with multiple correlated assets and datasets with Bid-Ask spreads and trading volume to evaluate their ability to capture inter-asset dependencies and market microstructure effects [21].
- **[F2]** **Data granularity evaluation.** We wish to test SDGFTS models across multiple data granularities (minute-level, daily, and weekly) to evaluate model performance on different temporal scales and their ability to capture both short-term and long-term dependencies. The most relevant granularities that are commonly used in financial analysis and trading strategies are 5-minute, 30-minute, and daily data [7, 22].

6.3.2 Beyond the project. We envision Stonk Bench evolving into a community-driven benchmark for SDGFTS evaluation. As we acknowledge the shortcomings outlined in §6.2, we propose the following future directions:

- **[F3]** **Continuous model inclusion.** Maintain Stonk Bench as a living benchmark by incorporating newly developed SDGFTS techniques (normalizing flows, autoregressive generators, advanced diffusion models, reinforcement-learning-based generators, and hybrids) [17].
- **[F4]** **Rigorous metric selection and ablations.** As mentioned in Stenger et al., we wish to rigorously conduct ablation studies to identify the most informative and robust metrics for different model families and stylized facts; produce guidance for a compact, interpretable metric set [20].
- **[F5]** **Categories of datasets** We wish to format Stonk Bench to cover a variaty datasets categorized by asset class (equities, forex, commodities, cryptocurrencies), granularity (minute, five-minute, daily), and datatypes (OHCL, BAV).
- **[F6]** **Public benchmark platform.** Launch a website with documentation, code, datasets, and a leaderboard for submitted SDGFTS models; provide CI-enabled evaluation pipelines for reproducible comparisons.
- **[F7]** **Community engagement.** Foster collaboration via contribution guidelines, open issues, and reproducible example workflows so researchers can extend and validate the benchmark.

Acknowledgments

To professor Irene Huang, University of Toronto, for her supervision and guidance throughout the project.

References

[1] 2019. *HistData.com Free Forex Historical Data.* Retrieved Dec 1, 2025 from https://www.histdata.com

[2] Beatrice Acciaio, Stephan Eckstein, and Songyan Hou. 2024. Time-Causal VAE: Robust Financial Time Series Generator. arXiv:2411.02947 [cs.LG] https://arxiv.org/abs/2411.02947

[3] David E. Allen, Leonard Mushunje, and Shelton Peiris. 2025. GANs and synthetic financial data: calculating VaR*. *Applied Economics* 57, 37 (2025), 5680–5695. arXiv:https://doi.org/10.1080/00036846.2024.2365456 doi:10.1080/00036846.2024.2365456

[4] Yihao Ang, Qiang Huang, Yifan Bao, Anthony K. H. Tung, and Zhiyong Huang. 2023. TSGBench: Time Series Generation Benchmark. *Proc. VLDB Endow.* 17, 3 (2023), 305–318.

[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. *Proceedings of the 34th International Conference on Machine Learning* 70 (2017), 214–223.

[6] Nicolas Boursin, Carl Remlinger, Joseph Mikael, and Carol Anne Hargreaves. 2022. Deep Generators on Commodity Markets; application to Deep Hedging. arXiv:2205.13942 [q-fin.RM] https://arxiv.org/abs/2205.13942

[7] R. Cont. 2001. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance* 1, 2 (2001), 223–236. arXiv:https://doi.org/10.1080/713665670 doi:10.1080/713665670

[8] Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. 2021. TimeVAE: A Variational Auto-Encoder for Multivariate Time Series Generation. arXiv:2111.08095 [cs.LG] https://arxiv.org/abs/2111.08095

[9] J. Durbin. 1960. The Fitting of Time-Series Models. *Revue de l'Institut International de Statistique / Review of the International Statistical Institute* 28, 3 (1960), 233–244. http://www.jstor.org/stable/1401322

[10] Simon Fecamp, Joseph Mikael, and Xavier Warin. 2021. Deep learning for discrete-time hedging in incomplete markets. *The Journal of Computational Finance* (01 2021). doi:10.21314/JCF.2021.006

[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Networks. arXiv:1406.2661 [stat.ML] https://arxiv.org/abs/1406.2661

[12] Diederik P Kingma and Max Welling. 2022. Auto-Encoding Variational Bayes. arXiv:1312.6114 [stat.ML] https://arxiv.org/abs/1312.6114

[13] J. Lee et al. 2022. VRNNGAN: A Recurrent VAE-GAN Framework for Synthetic Time-Series Generation. In *Conference.*

[14] Mark Leznik, Patrick Michalsky, Peter Willis, Benjamin Schanzel, Per-Olov Östberg, and Jörg Domaschka. 2021. Multivariate Time Series Synthesis Using Generative Adversarial Networks. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering* (Virtual Event, France) *(ICPE '21).* Association for Computing Machinery, New York, NY, USA, 43–50. doi:10.1145/3427921.3450257

[15] Zinan Lin, Alankar Jain, Chen Wang, Giulia Fanti, and Vyas Sekar. 2021. Using GANs for Sharing Networked Time Series Data: Challenges, Initial Promise, and Open Questions. arXiv:1909.13403 [cs.LG] doi:10.1145/3419394.3423643

[16] Chung I Lu and Julian Sester. 2025. Generative modelling of financial time series with structured noise and MMD-based signature learning. arXiv:2407.19848 [q-fin.MF] https://arxiv.org/abs/2407.19848

[17] Vamsi K. Potluru, Daniel Borrajo, Andrea Coletta, Niccolo Dalmasso, Yousef El-Laham, Elizabeth Fons, Mohsen Ghassemi, Sriram Gopalakrishnan, Vikesh Gosai, Eleonora Kreacic, Ganapathy Mani, Saheed Obitayo, Deepak Paramanand, Natraj Raman, Mikhail Solonin, Srijan Sood, Svitlana Vyetrenko, Haibei Zhu, Manuela Veloso, and Tucker Balch. 2024. Synthetic Data Applications in Finance. arXiv:2401.00081 [cs.LG] https://arxiv.org/abs/2401.00081

[18] Feng Qi. 2025. GAN-Based Financial Data Generation and Prediction: Improving The Authenticity and Prediction Ability of Financial Statements. *Informatica* (2025).

[19] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32* (Beijing, China) *(ICML'14).* JMLR.org, II–1278–II–1286.

[20] Michael Stenger, Robert Leppich, Ian Foster, Samuel Kounev, and André Bauer. 2024. Evaluation is key: a survey on evaluation measures for synthetic time series. *J Big Data* 11, 66 (May 2024). doi:10.1186/s40537-024-00924-7

[21] Tomonori Takahashi and Takayuki Mizuno. 2024. Generation of synthetic financial time series by diffusion models. arXiv:2410.18897 [q-fin.CP] https://arxiv.org/abs/2410.18897

[22] Zhengwei Wang, Qi She, and Tomas E. Ward. 2020. Generative Adversarial Networks in Computer Vision: A Survey and Taxonomy. arXiv:1906.01529 [cs.LG] https://arxiv.org/abs/1906.01529

[23] Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. 2020. Quant GANs: Deep Generation of Financial Time Series. *Quantitative Finance* 20, 9 (April 2020), 1419–1440. doi:10.1080/14697688.2020.1730426

A Utility Evaluation Logistics

Detailed procedures for (deep) hedging evaluation, including dataset splits, model architectures, training hyperparameters, and evaluation metrics.

A.1 Hedging Tasks Specifications

We consider the hedging of a European call option on a single underlying asset S_t with strike price K and maturity T. The option payoff at maturity is given by $g(S_T) = \max(S_T - K, 0)$.

For each sequence length tested in §4.1.3, we have set the corresponding option maturity T to match the length of the generated sequences. For example, if the sequence length is 60 minutes, then the option maturity T is set to 60 minutes.

A.2 Algorithm Comparison Models

In §4.3.4, we described the procedure for comparing multiple (deep) hedger algorithms to evaluate the relative performance preservation on synthetic data versus real data.

We have considered the following deep hedger algorithms:

[H1] Feedforward Layers Neural Network (FLNN). A standard multi-layer perceptron with fully connected layers, ReLU activations, and dropout for regularization.

[H2] Feedforward Time Neural Network (FTNN). A feedforward neural network designed to process time series data by accepting lagged observations as input features. This architecture uses fully connected layers with ReLU activations and dropout for regularization, enabling it to learn non-linear relationships between past and future values without explicit recurrence.

[H3] Long Short-Term Memory (LSTM). A recurrent neural network architecture designed to capture temporal dependencies in sequential data, particularly effective for time series tasks.

[H4] Recurrent Neural Network (RNN). An attention-based architecture that models long-range dependencies in sequences without relying on recurrence, utilizing self-attention mechanisms to enhance performance on sequential data.

We also considered the following (non-deep) hedging models:

[H5] Black-Scholes Delta Hedging (BSD). A classical hedging strategy based on the Black-Scholes option pricing model, which computes the delta of the option to determine the optimal hedge ratio.

[H6] Binomial Delta-Gamma Hedging (BDG). A discrete-time hedging strategy that uses a binomial tree model to compute both delta and gamma of the option, allowing for more accurate hedging in the presence of non-linear price movements.

[H7] Linear Regression Hedging (LR). A hedging strategy that uses linear regression to estimate the hedge ratios based on historical data.

[H8] XGBoost Hedging (XGB). A machine learning-based hedging strategy that employs gradient boosting decision trees to learn optimal hedge ratios from historical data.

Figure 8: Ablation study: fidelity per-metric rank-normalized heatmap across sequence lengths.

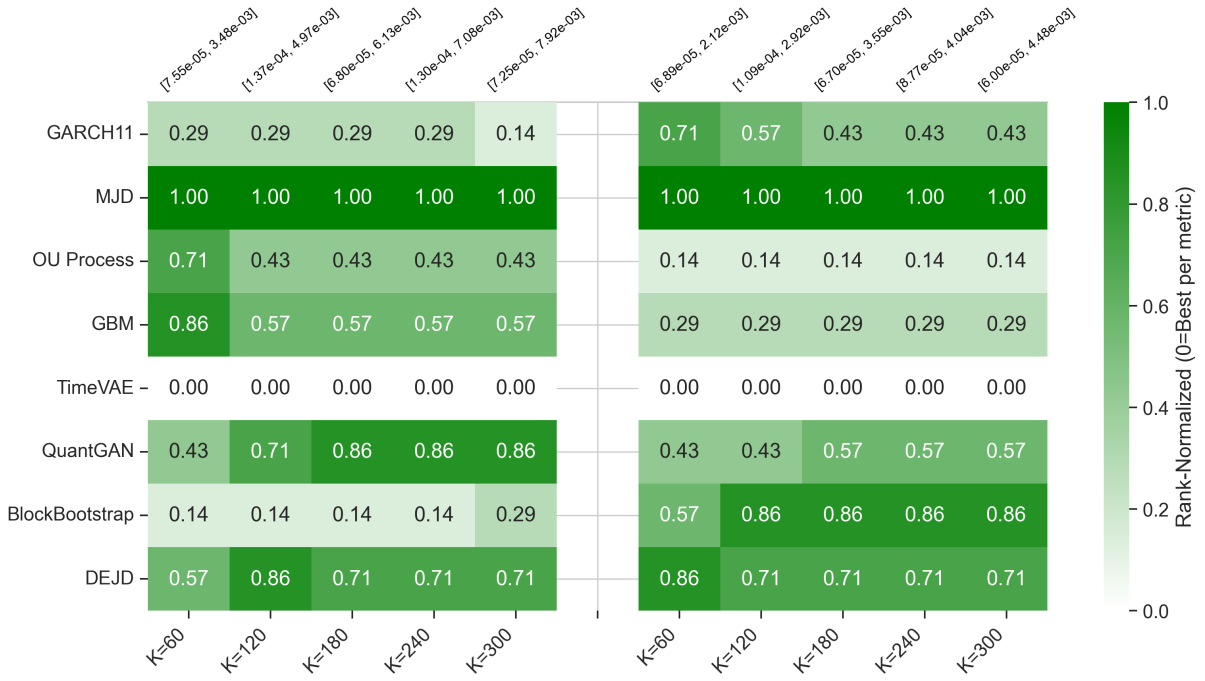

Figure 9: Ablation study: diversity per-metric rank-normalized heatmap across sequence lengths.

B Supplementary t-SNE Visualizations

C Augmented Testing Analysis

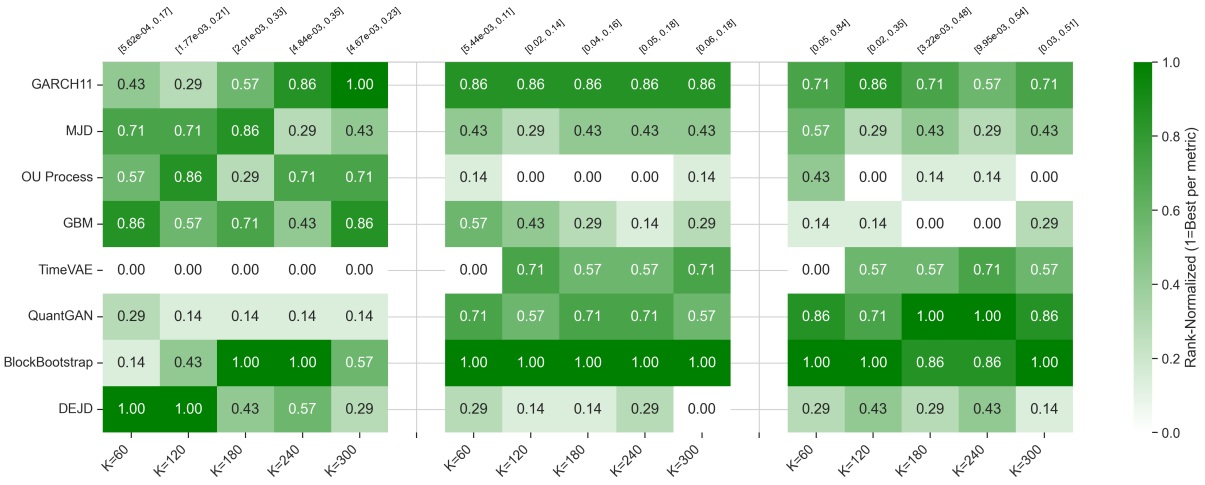

Figure 10: Ablation study: stylized-facts per-metric rank-normalized heatmap across sequence lengths.

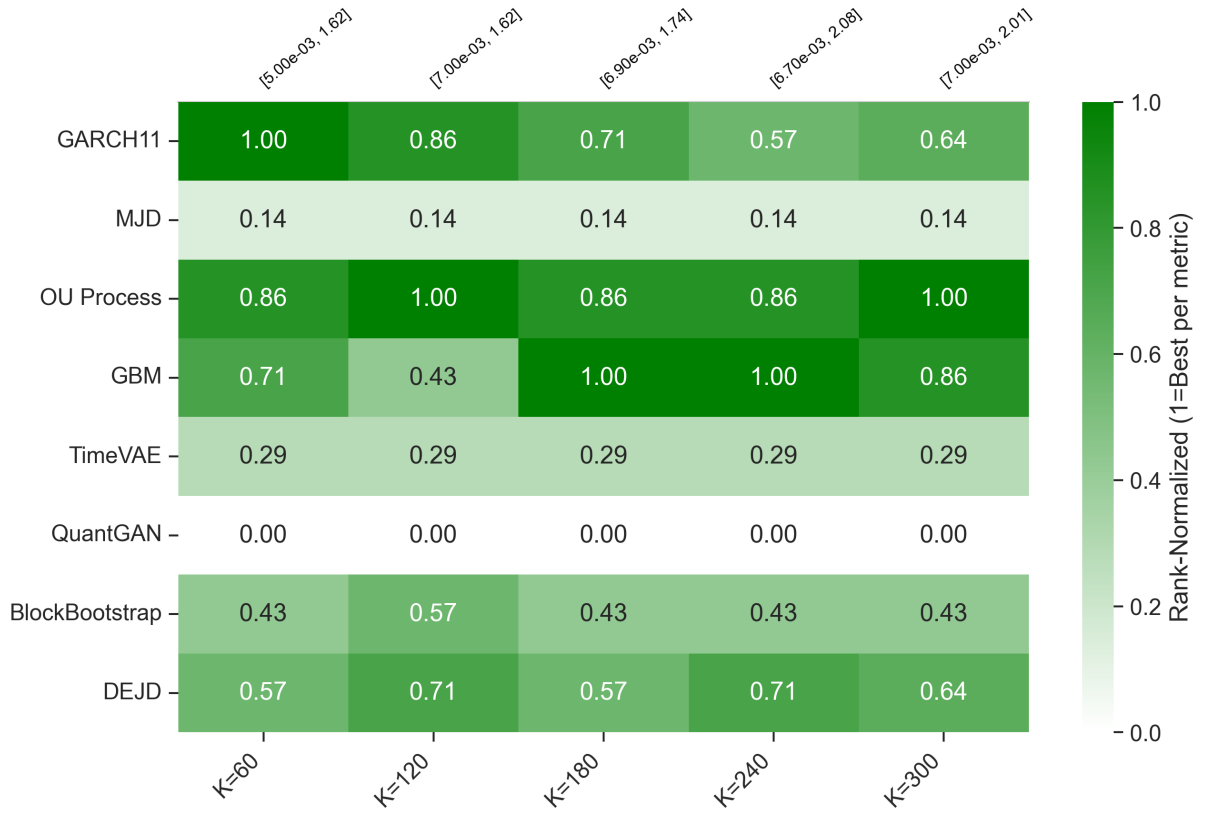

Figure 11: Ablation study: efficiency per-metric rank-normalized heatmap across sequence lengths.

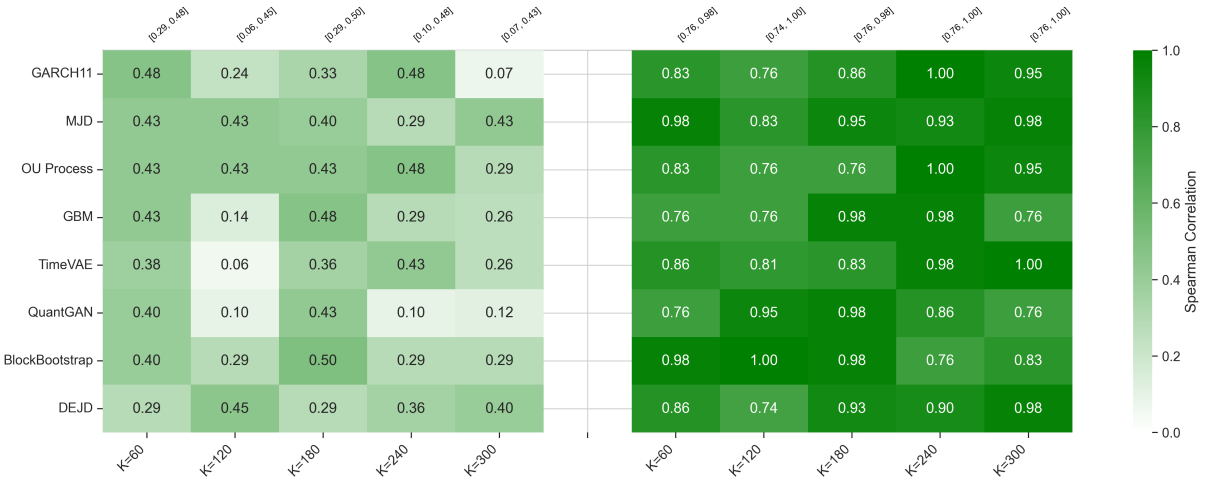

Figure 12: Ablation study: utility (Spearman) per-sequence-length comparison across hedger algorithms between real and sythetic data.

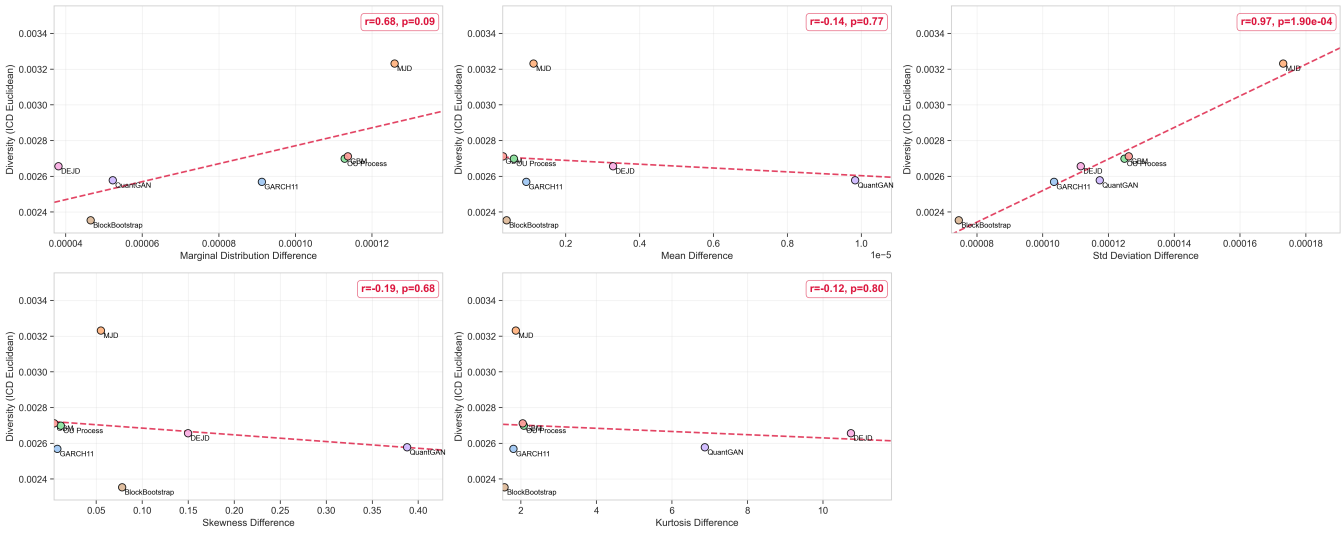

Figure 13: Scatter plots and linear regression analysis showing the relationship between fidelity metrics (MDD, MD, SDD, SD, KD, by row) and intra-class Euclidean distance (ICD-ED) across all SDGFTS models. Each subplot displays the fitted regression line with corresponding R^2 value and p-value, indicating the strength and statistical significance of the linear relationship between fidelity and diversity.

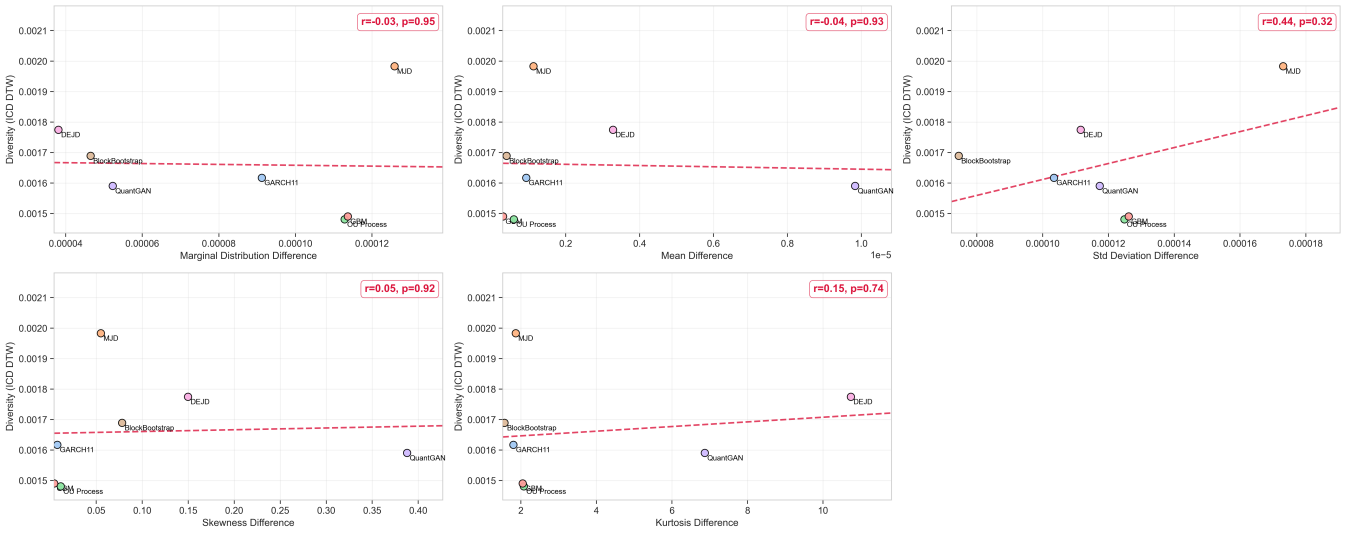

Figure 14: Scatter plots and linear regression analysis showing the relationship between fidelity metrics (MDD, MD, SDD, SD, KD, by row) and intra-class DTW distance (ICD-DTW) across all SDGFTS models. Each subplot displays the fitted regression line with corresponding R^2 value and p-value, indicating the strength and statistical significance of the linear relationship between fidelity and diversity measured by DTW.

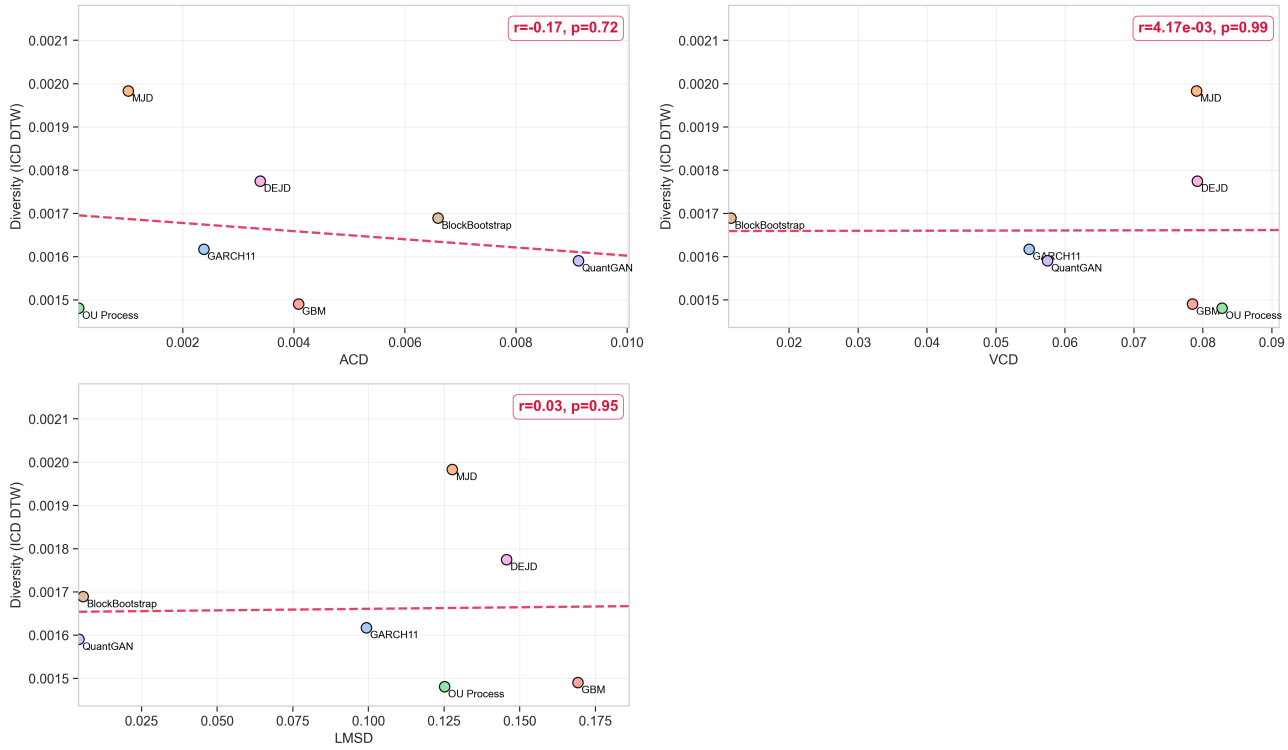

Figure 15: Scatter plots showing the relationship between stylized facts metrics (ACD, volatility clustering, and LMSD) and intra-class DTW distance (ICD-DTW) across all SDGFTS models. Each point represents a model, with trend lines indicating correlation direction and strength.

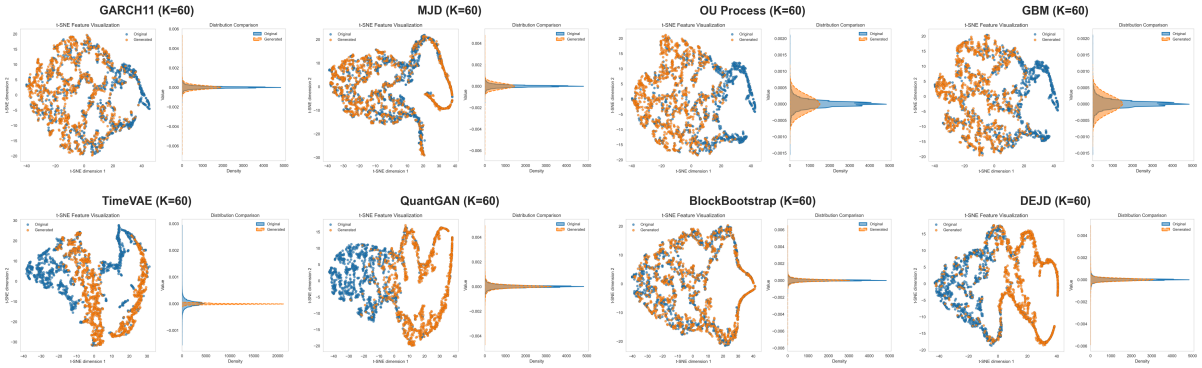

Figure 16: Supplementary t-SNE visualization for sequence length 60 minutes.

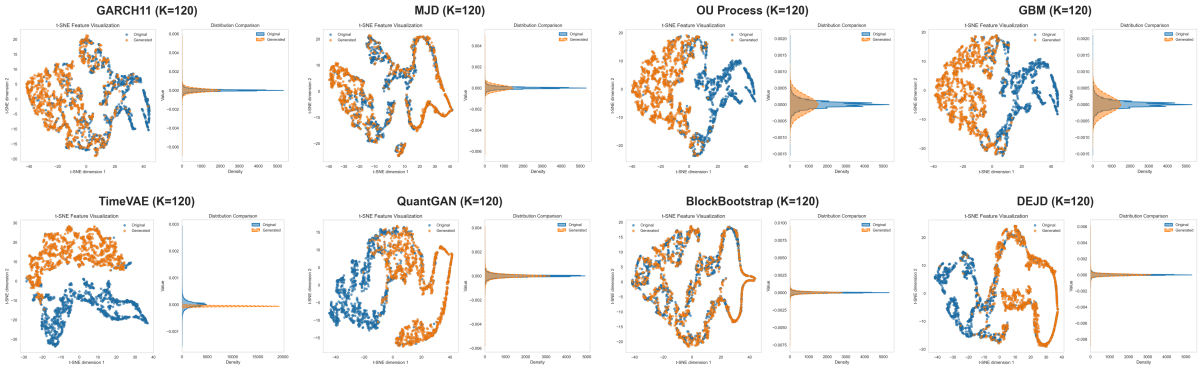

Figure 17: Supplementary t-SNE visualization for sequence length 120 minutes.

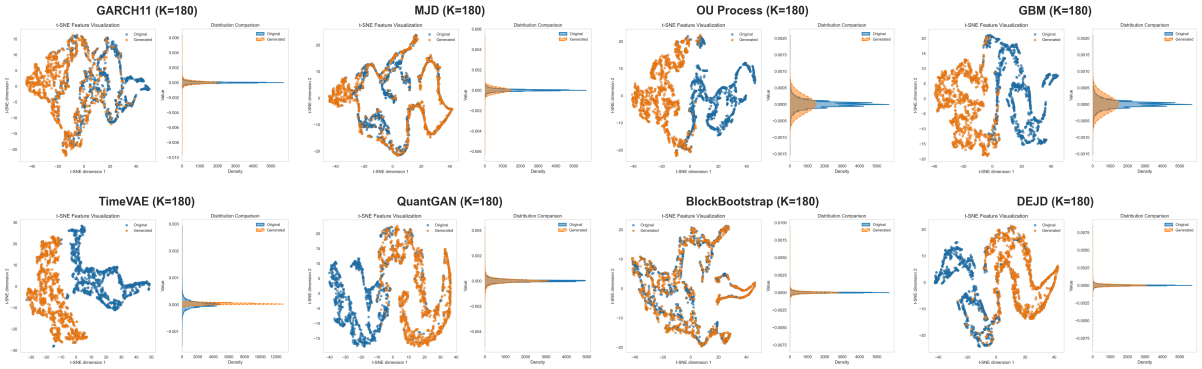

Figure 18: Supplementary t-SNE visualization for sequence length 180 minutes.

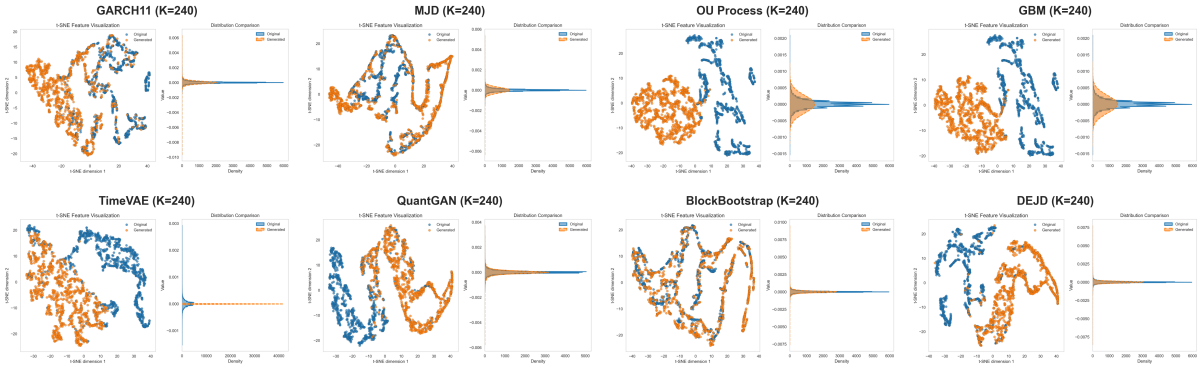

Figure 19: Supplementary t-SNE visualization for sequence length 240 minutes.

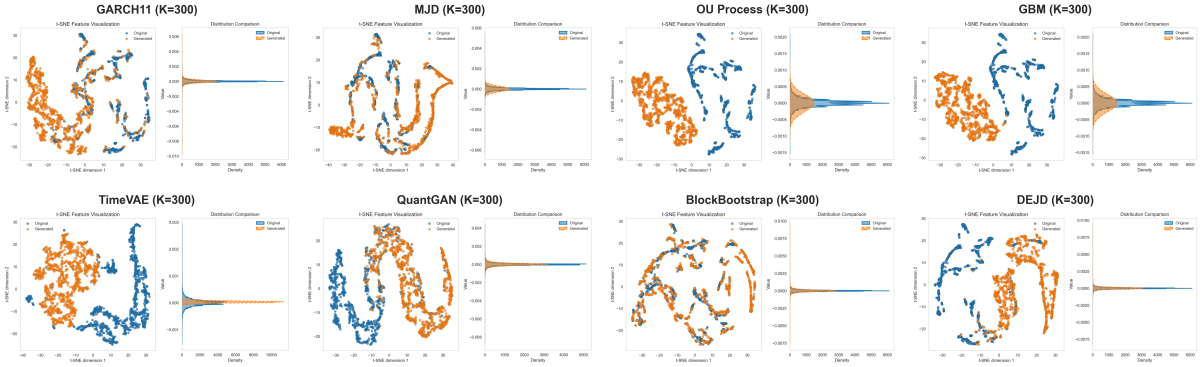

Figure 20: Supplementary t-SNE visualization for sequence length 300 minutes.

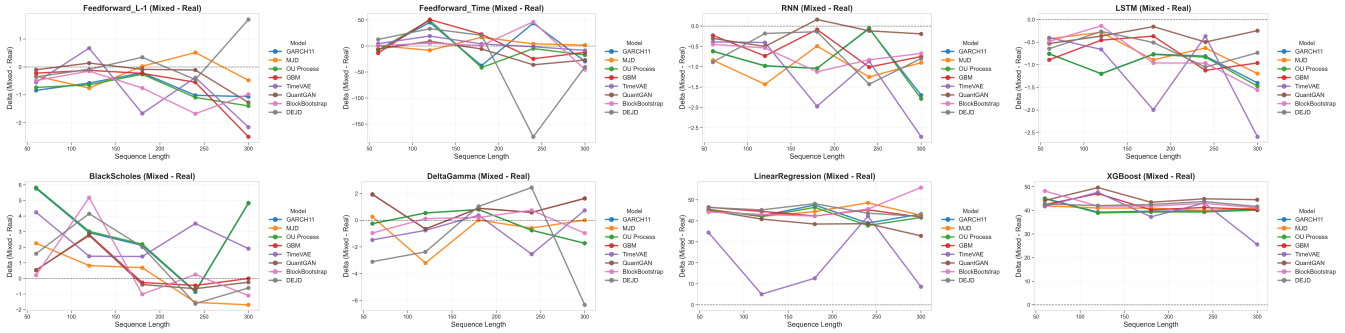

Figure 21: Augmented testing performance comparison showing delta metrics across different hedger algorithms and SDGFTS models. Each subplot compares performance when training on real data only versus augmented data (real + synthetic).