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, ie, stylized facts, including volatility clustering, fat tails, absence of autocorrelations in returns, and leverage effects. By formalizing SDGFTS evaluation, StonkBench establishes a gold standard that facilitates the development of more effective generation methods, supports fair benchmarking across studies, and advances both current and future financial forecasting and analysis.

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, 15 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 [7]. The ability to generate high-quality synthetic financial data addresses several critical challenges in the field, replicating and simulating fat-tail behaviors observed in financial returns time series and develop back-testing procedures for prediction and deep hedging algorithms. [12].

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 [10]. Current evaluation methods vary widely across studies, making it difficult to compare different approaches objectively and determine their relative strengths and weaknesses.

Our research addresses this gap by introducing a comprehensive benchmark framework that encompasses:

*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

- Statistical fidelity measures for comparing synthetic and real FTS
- Utility metrics that assess the practical value of synthetic data in downstream tasks
- Performance evaluation across multiple SDGFTS models and datasets that is applicable to many real-world financial applications.

By analyzing the results from our MLflow experiments and applying our unified benchmark, we aim to provide clear guidelines for evaluating SDGFTS models and establish a standard for future research in this domain.

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[10]; this issue extends beyond FTS to generative frameworks like GANs as a whole [13]. Many evaluation measures are insufficiently defined and lack public implementations, making reuse and reproduction troublesome and error-prone [10]. This presents a challenge unique to the generation task compared to areas like time series forecasting or classification [10]. Hence, future research would immensely benefit from a widely accepted, reasonably sized set of qualified measures for the central evaluation criteria. Here are our 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 solely on statistical similarity, while others emphasize utility in downstream tasks [1].

[L2] **Lack of Standardized Datasets:** The absence of common benchmark datasets leads to evaluations on disparate data, making it challenging to assess model performance uniformly [10] in terms of both asset classes, data granularity, and data length.

[L3] **Limited Scope of Evaluation:** Many evaluations concentrate on a narrow set of criteria, such as marginal distributions, neglecting other important aspects like temporal dependencies and multivariate relationships [2].

[L4] **Reproducibility Issues:** The lack of open-source implementations and detailed evaluation protocols impedes reproducibility and validation of results across different studies [10]. 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. [10], develop a comprehensive and unified SDG benchmark expanded from the works of Ang et al. [1] in specifics to FTS by incorporating a utility evaluation framework inspired by Boursin et al. [2] through portfolio evaluations generated by a deep hedger. Our key contributions include:

[C1] **Consolidated Evaluation Framework:** We systematically review and consolidate evaluation methods from leading papers (models and surveys) in SDG and financial time series [1, 2, 10], 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] **Open Benchmark Platform:** We deliver an open-source benchmark framework for the research community, aiming to establish a gold standard for SDGFTS model evaluation and comparison.

2 Preliminaries

2.1 Problem Definition

Let us formally define the Synthetic Data Generation problem for Financial Time Series (FTS). Consider a multivariate financial time series $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)^T \in \mathbb{R}^{T \times N}$ of length T with N features (e.g., OHLC prices). We extract overlapping subseries of length L using a sliding window with stride S, denoted as $\mathbf{x}_i = (\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,L}) \in \mathbb{R}^{L \times N}$ for $i = 1, \ldots, R = \lfloor \frac{T-L}{S} \rfloor + 1$, where each $\mathbf{x}_{i,j}$ is the N-dimensional feature vector at time step j within the i-th subseries.

Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_R\}$ denote the set of R subseries extracted from the original time series, with underlying distribution $p_{\mathcal{X}}$. The objective of synthetic data generation for financial time series is to generate a synthetic set of subseries $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_K\}$ of arbitrary size K, such that the distribution $p_{\hat{\mathbf{X}}}$ closely approximates $p_{\mathcal{X}}$, while preserving the key temporal and statistical characteristics of the original series, including autocorrelation structure, volatility clustering, and distributional moments (mean, variance, skewness, and 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 non-parametric approaches, particularly deep learning-based models that learn the data distribution directly from observations without assuming a specific form. These include generative adversarial networks, variational autoencoders, and diffusion models, representing the cutting edge in synthetic data generation.

2.2.2 Scope of Datasets. We evaluate models on diverse financial datasets covering multiple asset classes and temporal granularities, ranging from high-frequency (1-minute, 5-minute) to low-frequency (daily) resolutions. This range enables the assessment of generative performance across varying market dynamics and forecasting horizons.

- **Stock market indices:** S&P 500, NASDAQ, and TSX.

- **Individual equities:** Price series from major exchanges (NYSE, NASDAQ, TSX).
- **Foreign exchange (Forex):** Major trading pairs (e.g., EUR/USD, USD/JPY).

2.2.3 Scope of Evaluation Taxonomical Criteria. Our evaluation framework adopts the taxonomy proposed by Stenger et al. as an organizing structure, while drawing specific evaluation metrics primarily from TSGBench, which provides an existing and well-established benchmark for general synthetic time series benchmarking. Although Stenger et al. introduce an extensive set of potential measures, our selection focuses on metrics that are both widely used in recent synthetic time series benchmarks and empirically meaningful for financial applications. Our reformulation leverages Stenger's taxonomy to provide a coherent organizational hierarchy, particularly Fidelity, Diversity, Efficiency, and Utility, while ensuring that each chosen metric remains interpretable, computationally feasible, and directly comparable to prior benchmarks. By grounding our framework in the pragmatic evaluation design of TSGBench and aligning it with Stenger's structured taxonomy, we ensure both methodological rigor and cross-study consistency in SDGFTS assessment.

3 Overview of Synthetic FTS Methods

We categorize synthetic financial time series generation methods into two main families: classical parametric (stochastic) models, which rely on explicit assumptions about the data-generating process, and modern non-parametric (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.

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

The GBM parameters are estimated from historical log returns: the drift μ is set to the sample mean and the volatility σ is set to the sample standard deviation. This corresponds to maximum likelihood estimation under the assumption that log returns are normally distributed.

[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).

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

The OU process parameters are estimated from historical log-prices by fitting a discrete-time AR(1) model $X_{t+1} = \phi X_t + c + \varepsilon_t$ and mapping the coefficients to continuous-time parameters. The mean-reversion rate θ, long-term mean μ, and volatility σ are derived from ϕ, c, and the residual standard deviation, corresponding to statistical estimation from data.

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

$$\frac{dS_t}{S_{t^-}} = \mu\, dt + \sigma\, dW_t + (e^Z - 1)\, dN_t, \tag{3}$$
$$Z \sim \mathcal{N}(\mu_j, \sigma_j^2), \quad N_t \sim \text{Poisson}(\lambda t).$$

Jumps in the log-returns r_t are identified as returns exceeding a threshold of k standard deviations from the mean, and the jump intensity λ is estimated as

$$\lambda = \frac{\text{number of jumps}}{T}.$$

The jump size mean and variance are computed from the detected jumps:

$$\mu_J = \frac{1}{n_{\text{jumps}}} \sum_{\text{jumps}} r_t, \tag{4}$$

$$\sigma_J^2 = \frac{1}{n_{\text{jumps}} - 1} \sum_{\text{jumps}} (r_t - \mu_J)^2. \tag{5}$$

The diffusion volatility σ is obtained by removing the jump contribution from the total variance per unit time:

$$\sigma = \sqrt{\frac{\text{Var}(r_t)}{\delta t} - \lambda(\mu_J^2 + \sigma_J^2)},$$

and the drift μ is corrected for the expected jump contribution:

$$\mu = \frac{\mathbb{E}[r_t]}{\delta t} - \lambda \mu_J.$$

[A4] Double-Exponential Jump Diffusion (DEJD). A jump-diffusion model with asymmetric double-exponential jump sizes. Model-specific parameters are p (probability of upward

jump) and η_1, η_2 (exponential parameters for upward/downward jumps):

$$\frac{dS_t}{S_{t^-}} = \mu\, dt + \sigma\, dW_t + (e^Y - 1)\, dN_t, \tag{6}$$

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

Jumps in the log-returns r_t are identified using a robust threshold based on the median absolute deviation, and the jump intensity λ is estimated as

$$\lambda = \frac{\text{number of jumps}}{T},$$

corresponding to the compound Poisson process N_t.
The jump size distribution Y is separated into positive and negative jumps to estimate the double-exponential parameters:

$$p = \frac{\text{number of positive jumps}}{\text{total jumps}}, \tag{7}$$

$$\eta_1 = \frac{1}{\text{mean of positive jumps}}, \tag{8}$$

$$\eta_2 = \frac{1}{-\text{mean of negative jumps}}. \tag{9}$$

Finally, the drift μ and diffusion volatility σ of the continuous part are corrected for the expected jump contribution:

$$\mu = \frac{\mathbb{E}[r_t]}{\delta t} - \lambda\, \mathbb{E}[Y], \tag{10}$$

$$\sigma = \sqrt{\frac{\text{Var}(r_t)}{\delta t} - \lambda\, \mathbb{E}[Y^2]}, \tag{11}$$

with

$$\mathbb{E}[Y] = \frac{p}{\eta_1} - \frac{1-p}{\eta_2}, \tag{12}$$

$$\mathbb{E}[Y^2] = 2\left(\frac{p}{\eta_1^2} + \frac{1-p}{\eta_2^2}\right). \tag{13}$$

[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). The model captures volatility clustering via time-varying conditional variance:

$$r_t = \sigma_t z_t, \quad z_t \sim \mathcal{D}, \tag{14}$$
$$\sigma_t^2 = \omega + \alpha r_{t-1}^2 + \beta \sigma_{t-1}^2.$$

The parameters (ω, α, β) are estimated via maximum likelihood from the observed log-return series.

3.2 Deep Learning (Non-parametric) Methods

Non-parametric 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 three major families of deep generative models: Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Diffusion Models.

3.2.1 Diffusion Models. Diffusion models (or score-based generative models) have emerged as powerful methods for modeling complex data distributions via a forward (noise-adding) process and a learned reverse (denoising) process [3?]. In the forward direction, one gradually corrupts a data sample x_0 into noise via a Markov chain:

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}), \tag{15}$$
$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t I\right).$$

The reverse (generative) model is parameterized as

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\big(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\big),$$

where μ_θ and Σ_θ are often expressed via a neural network that estimates the score $\nabla_\theta \log p_\theta(x_t)$. A common training objective is the simplified denoising score-matching loss:

$$\mathbb{E}_{t, x_0, \epsilon}\left[\left\|\epsilon - \epsilon_\theta(x_t, t)\right\|^2\right],$$

where $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon$.

In finance/time-series, diffusion models have recently been adapted to generate synthetic asset paths by converting multivariate time series into suitable representations (e.g. wavelet-transformed images) and then sampling via reverse diffusion [12]. For instance, Takahashi and Mizuno shows that such an approach can reproduce key statistical properties of financial data (e.g. volatility clustering, tail behavior) [11]. Another relevant work is "A Financial Time Series Denoiser Based on Diffusion Model", which shows how diffusion can help improve downstream predictability and reduce noise in financial signals [14].

3.2.2 Variational Autoencoders (VAEs). Variational Autoencoders [9?] 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 [**?**]. 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.

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

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

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.

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 [8]. Another example is VRNNGAN, which uses a recurrent VAE as the generator and a recurrent discriminator to capture temporal dependencies in synthetic sequence generation [4]. GANs are relevant in finance because they can capture complex joint distributions and dependencies in multivariate time-series without requiring explicit likelihood models.

3.3 Miscellaneous

This section consists of models that are classified as neither parametric nor non-parametric.

[A1] Block Bootstrap (non-parametric). 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 multivariate time series. Several modifications were introduced to better accommodate financial time series and stochastic models.

4.1.1 Data type. Stock price data (e.g., AAPL) at daily frequency with OHLC columns (Open, High, Low, Close) are used. Let $X \in \mathbb{R}^{L\times N}$ denote a multivariate time series with $N = 4$ channels and length L.

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

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

yielding a transformed series of length $L - 1$.

4.1.3 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, preserving the time serie's distribution along with financial properties such as heavy tails and volatility clustering.
- 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.
- For non-parametric models, overlapping sliding windows $\mathcal{W} \in \mathbb{R}^{R\times L\times N}$ with stride 1 are extracted. Unlike TSGBench, which employs an autocorrelation-based hard-coded window of 125 (often yielding a size of 1 for stock data due to fast autocorrelation decay), the window length L is determined via the partial autocorrelation function (PACF). Specifically, PACF is computed for each channel up to $\lfloor 10 \log_{10} n \rfloor$ lags, and the lag corresponding to the maximum significant PACF peak across channels is selected, resulting in $L = 13$. This approach captures relevant temporal dependencies within the 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.

4.1.4 Data loaders. For non-parametric 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 non-parametric 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 Feature-based Distance Measures. These metrics quantify how well synthetic data captures key marginal and second-order properties of real data.

[M1] Marginal Distribution Difference (MDD): Distance between real and synthetic marginals (e.g., 1-Wasserstein distance) per channel:

$$\text{MDD}_j = W_1(\widehat{F}^j_{real}, \widehat{F}^j_{synth}),$$

$$W_1(F, G) = \int_0^1 \left|F^{-1}(u) - G^{-1}(u)\right| du,$$

where $\widehat{F}^j_{real}$ and $\widehat{F}^j_{synth}$ are empirical CDFs of channel j.

[M2] Mean Difference (MD): Absolute difference of per-channel means:

$$\mathrm{MD}_j = \left|\mu^j_{real} - \mu^j_{synth}\right|.$$

[M3] Standard Deviation Difference (SDD): Absolute difference of per-channel standard deviations:

$$\mathrm{SDD}_j = \left|\sigma^j_{real} - \sigma^j_{synth}\right|.$$

[M4] Kurtosis Difference (KD): Absolute difference of per-channel excess kurtosis:

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

[M5] AutoCorrelation Difference (ACD): Absolute difference in lag-1 autocorrelation per channel:

$$\rho^j(1) = \frac{\sum_{t=2}^{L}(x^j_t - \bar{x}^j)(x^j_{t-1} - \bar{x}^j)}{\sum_{t=1}^{L}(x^j_t - \bar{x}^j)^2},$$
$$\mathrm{ACD}_j = \left|\rho^j_{real}(1) - \rho^j_{synth}(1)\right|.$$

4.2.2 Visualization Methods. Visual tools complement numeric metrics for face-validity and communication [1].

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

[M7] Distribution: Channel-wise histograms of returns to visualize empirical distribution.

4.2.3 Diversity Metrics. Measures that prevent mode collapse, computed per channel. Pairwise distances only consider the same channel across samples.

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

$$\text{ICD-ED}^j = \frac{2}{R(R-1)} \sum_{1 \le a < b \le R} \left\| \mathbf{x}^j_a - \mathbf{x}^j_b \right\|_2,$$

where $\mathbf{x}^j_a \in \mathbb{R}^L$ is the j-th channel of the a-th sample.

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

$$\text{ICD-DTW}^j = \frac{2}{R(R-1)} \sum_{1 \le a < b \le R} d_{\mathrm{DTW}}\left(\mathbf{x}^j_a, \mathbf{x}^j_b\right).$$

4.2.4 Efficiency Assessment.

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

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

4.2.5 Stylized Facts Verification. Canonical stylized facts, measured per channel:

[M11] Heavy Tails: Per-channel excess kurtosis:

$$\kappa^{\mathrm{ex}}(X^j) = \frac{\mathbb{E}[(X^j - \mathbb{E}[X^j])^4]}{(\mathrm{Var}[X^j])^2} - 3.$$

[M12] Return Autocorrelation: Lag-1 autocorrelation per channel, should be close to zero for liquid assets.

[M13] Volatility Clustering: Lag-1 autocorrelation of squared or absolute returns per channel:

$$\rho^j_{x^2}(1) = \mathrm{Corr}\big((x^j_t)^2, (x^j_{t-1})^2\big),$$
$$\rho^j_{|x|}(1) = \mathrm{Corr}\big(|x^j_t|, |x^j_{t-1}|\big).$$

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 [2]. 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 $d+1$ tradable assets $S = (S^0, \ldots, S^d)$, where S^j_t denotes the price of asset j at time t. For simplicity, we assume a zero interest rate environment.

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 $X^{\Delta,p}_T$, is given by

$$X^{\Delta,p}_T = p + \sum_{i=1}^{d} \sum_{j=0}^{N-1} \Delta^i_{t_j} \left(F^i_{t_{j+1}} - F^i_{t_j} \right), \tag{16}$$

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

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

$$(p^{\mathrm{opt}}, \Delta^{\mathrm{opt}}) = \underset{p,\Delta}{\mathrm{Argmin}}\, \mathbb{E}\left[\left(X^{\Delta}_T - g(S_T) \right)^2 \right]. \tag{17}$$

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.

4.3.2 Deep Hedger Architecture. Our deep hedger architecture centeres around the Train-Synthetic-Test-Real (TSTR) evaluation framework [5], adapted for financial time series generation. We begin with our real FTS dataset D—partitioned to a training, validation, and test set respectively as $D_r = \{D^{train}_r, D^{validate}_r, D^{test}_r\}$—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

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

subsubsection. We consider a deep hedger algorithm A (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.

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 (deep 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 deep hedger models [10].

4.3.3 Deep Hedger Extentions. We proposed two extensions to the standard deep hedger architecture to better suit our benchmark need that drew inspiration from recent literature [10].

[E1] Augmented Testing [10] 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 traid model M_g on D_{aug} where $D_{aug} := D_r^{train} \cup D_g$. Then we proceed to evaluate scores between modelsM_g and M_r on the real test set, where M_r i s still trained on only real data D_r^{train}.

[E2] Algorithm Comparison [6] Another extension is to compare multiple deep hedger algorithms indicating to what degree each algorithm performs equally on the generated data relative to the 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 $A_i \in mathcalA_n$ 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 setes 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.

Finally we compare the relative performance of each algorithm on real data versus synthetic data by computing the Spearman's rank correlation coefficient r_S [6] between the two score sets s_r and s_g:

$$r_S = 1 - \frac{6 \sum_{i=1}^{n} (\text{rank}(s_r^i) - \text{rank}(s_g^i))^2}{n(n^2 - 1)}, \tag{18}$$

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.

We interpret a high positive correlation (i.e., r_S close to 1) as an indication that the synthetic data generated by $mathcalM_{TS}$ preserves the relative performance of different deep hedger algorithms.

5 Results and Analysis

In this section, we present a comprehensive evaluation of the synthetic financial time series generated by all models under consideration. The evaluation focuses on multiple statistical, temporal, and diversity metrics to assess how well each model reproduces the characteristics of real financial data. Detailed tables of evaluation results can be found in Appendix A, while visualizations including T-SNE embeddings and distribution comparisons are presented in Appendix B. A comparative view of evaluation metrics across models is provided in Appendix C.

5.1 Statistical Evaluation Results

Analysis of the evaluation metrics reveals several noteworthy findings. First, the GBM, OU, and GARCH(1,1) model all fail to fully capture key stylized facts of financial time series, particularly extreme events and fat-tailed behavior. This limitation is evident in the kurtosis metrics, where these models substantially underestimate the frequency and magnitude of rare, high-impact returns (Table **??**). These shortcomings reflect the inherent assumptions of each model: GBM assumes log-normal returns with constant volatility, OU is mean-reverting with Gaussian noise, and GARCH(1,1), despite modeling conditional heteroskedasticity, does not sufficiently reproduce the tails of the empirical return distribution. Collectively, these models exhibit large deviations in kurtosis and other distributional metrics, highlighting their inability to generate the extreme events observed in real financial time series.

Among all models, the Block Bootstrap approach consistently performs well across nearly all statistical metrics. This strong performance is expected, as Block Bootstrap samples directly from historical log returns, inherently preserving the empirical distribution, including autocorrelation structures and heavy tails. Kurtosis and skewness deviations are minimal, and temporal similarity metrics remain high (Appendix A), confirming its ability to retain higher-order moments and realistic sequence dynamics.

Deep learning-based models generally underperform relative to parametric approaches. In particular, TimeVAE and Takahashi Diffusion generate distributional plots that fail to reproduce key features of the real data (Appendix B). This underperformance is likely attributable to insufficient data expressivity and limited training: the batch size was set to 32, the number of training epochs to 20, and the window length selected via the Partial Correlation Function (PCF) was 13. As a result, only 620/9035 time series samples were effectively used to train each model. Under these conditions, the models were unable to capture complex temporal dependencies and higher-order moments, leading to underfitting, as evidenced

by deviations in both statistical and temporal metrics. Future work will address these limitations by increasing batch size and training epochs, as well as experimenting with higher-granularity index data (e.g., minute-level stock prices), consistent with the data granularity used to successfully train the Takahashi DDPM.

Merton Jump Diffusion (MJD) and, especially, Double Exponential Jump Diffusion (DEJD) display extremely high kurtosis, consistent with their intended design. DEJD was explicitly developed to model extreme market movements and heavy tails for robust backtesting, producing rare but impactful price jumps. In T-SNE visualizations (Appendix B), DEJD-generated data forms distinct clusters separated from the main cluster, highlighting the presence of extreme outliers. MJD exhibits moderate kurtosis increases relative to GBM, reflecting its capacity to generate jumps.

Overall, these results illustrate the trade-offs between model simplicity, parametric assumptions, and the ability to replicate complex statistical properties of financial time series. Parametric jump models effectively capture extreme events, Block Bootstrap preserves empirical distribution fidelity, and deep learning models require larger and more expressive datasets to reach their theoretical potential. The comparative metrics in Appendix C further highlight the relative strengths and weaknesses of each approach in both quantitative and visual terms.

5.2 Utility Evaluation Results

Results of the deep hedging evaluation (replication error, P&L distribution, risk-adjusted metrics) can be summarized in tables analogous to the above once computed.

5.3 Comprehensive Model Comparison

Provide radar plots or scorecards that combine normalized metrics (fidelity, diversity, stylized facts, efficiency) into composite ranks per task.

5.4 Ranking Analysis

Discuss trade-offs: fidelity vs. diversity, training time vs. quality, and sensitivity to window length and sample count.

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.

6.1 Summary of Findings

We presented Stonk-Bench, a comprehensive benchmark for evaluating Synthetic Data Generation for Financial Time Series (SDGFTS) models. Our 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

6.2 Implications for SDGFTS Research

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.* .

6.3 Stonk-Bench Limitation

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

- **Incompatibility with other models:** StonkBench has so far been primarily evaluated on classical stochastic models (e.g., GBM, OU, GARCH) and select deep learning models—Diffusion models, GANs, and VAEs. Other model families (e.g., reinforcement learning-based generators, normalizing flows, autoregressive networks, hybrid approaches) have not been benchmarked, so StonkBench's applicability to them is not yet established. As a result, some metrics or procedures may not directly translate, limiting the framework's generalizability to all synthetic financial time series generation methods.
- **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.
- **Limited input data:** The benchmark strictly uses raw historical stock price data (Open, High, Low, Close) as input. External information, such as causal graphs, news sentiment, macroeconomic indicators, or alternative data sources, is not incorporated, even though such information can significantly influence the accuracy of future price predictions.
- **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.
- **Computational Resource Requirements:** The comprehensive nature of Stonk-Bench —particularly accomodating any submited deep hedging model and the deep hedging

utility evaluation —demands significant computational resources. This may limit accessibility for researchers with constrained resources.

6.4 Future Research Directions

6.4.1 Progress for the remaining duration of the project. In regard to the next portion of our project, we aim to expand Stonk-Bench in several key areas:

- **Complete Utility Evaluations:** As of now, our utility evaluation through deep hedging is still in progress. We plan to finalize this component and integrate the results into the overall benchmark.
- **Matching Distribution Metrics:** We intend to incorporate additional distribution matching metrics, such as the the Komogorov-Smirnov (KS) statistic to further enhance the fidelity evaluation.
- **Expansive Datasets:** We plan to test Stonk-Bench on a wider variety of financial time series datasets, including indices and intraday (1-minute) data, to assess the generalizability of our benchmark and its performance in downstreamq tasks evaluations.

6.4.2 Beyond this project. In regards the larger scope of our Stonk-Bennch project, we envision several avenues for future research and project development:

- **Incorporation of Advanced SDGFTS Models:** As new models are developed, we plan to continuously update Stonk-Bench to include these advancements, ensuring that our benchmark remains relevant and comprehensive.
- **Develop rigorous evaluation metric selection:** We aim to refine and expand the selection of evaluation metrics used in Stonk-Bench, ensuring they capture the multifaceted nature of synthetic financial time series data. A proposed direction is to perform ablation studies to identify the most informative metrics for different types of SDGFTS models with its known stylized facts and characteristics [10].
- **Host a benchmark website:** To facilitate wider adoption and accessibility, we plan to develop a dedicated website for Stonk-Bench. This platform will provide comprehensive documentation, code repositories, and resources for researchers and practitioners interested in utilizing the benchmark. We envision this website to also feature a ranking leaderboard where researchers can submit their SDGFTS models and compare their performance against existing models evaluated using Stonk-Bench.
- **Community Engagement and Collaboration:** We hope to foster a collaborative environment where researchers can contribute to the development and refinement of Stonk-Bench, promoting shared progress in the field of synthetic data generation.

Acknowledgments

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

References

[1] 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.

[2] 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

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 574, 12 pages.

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

[5] 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

[6] 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

[7] 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

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

[9] 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.

[10] 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

[11] 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

[12] Yuki Tanaka, Ryuji Hashimoto, Takehiro Takayanagi, Zhe Piao, Yuri Murayama, and Kiyoshi Izumi. 2025. CoFinDiff: Controllable Financial Diffusion Model for Time Series Generation. arXiv:2503.04164 [q-fin.CP] https://arxiv.org/abs/2503.04164

[13] 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

[14] Zhuohan Wang and Carmine Ventre. 2024. A Financial Time Series Denoiser Based on Diffusion Models. In *Proceedings of the 5th ACM International Conference on AI in Finance* (Brooklyn, NY, USA) *(ICAIF '24)*. Association for Computing Machinery, New York, NY, USA, 72–80. doi:10.1145/3677052.3698649

A Evaluation Tables

Table 3: Fidelity metrics by model and channel (lower ↓ is better for differences; best value in each column bolded). Channels O, H, L, C correspond to dataset channels (Open, High, Low, Close).

	MDD ↓				MD ↓				SDD ↓			
Model	O	H	L	C	O	H	L	C	O	H	L	C
GBM	4.35	4.48	4.48	3.41	0.0010	0.0004	0.0005	0.0014	0.0115	0.0108	0.0119	0.0110
OU_Process	4.31	4.48	4.43	3.37	0.00095	**0.00027**	0.00046	0.0014	0.0115	0.0107	0.0119	0.0110
MJD	3.74	3.89	3.86	2.87	0.0014	0.00041	0.00042	0.00045	0.0104	0.0115	0.0133	0.0107
GARCH11	**2.60**	**2.47**	2.88	**1.92**	**0.000004**	0.000079	0.000096	0.000080	**0.00041**	**0.00045**	0.0020	**0.0012**
DEJD	2.79	**2.38**	**2.55**	2.17	0.0015	0.00094	0.0018	0.0011	0.0138	0.0116	0.0160	0.0141
BlockBootstrap	2.82	2.75	2.78	2.20	0.0014	0.00049	0.00033	0.00073	0.0102	0.0102	0.0108	0.0118
TimeGAN	5.32	5.55	6.00	4.42	0.0026	0.0022	0.0034	0.0018	0.0081	0.0084	0.0094	0.0077
QuantGAN	3.97	4.04	4.67	2.83	0.0056	0.00034	0.0079	0.0032	0.0085	0.0084	0.0118	0.0068
TimeVAE	9.98	9.08	9.32	7.25	0.0018	0.0017	0.0011	0.0017	0.0188	0.0157	0.0170	0.0194
Takahashi	5.81	6.33	6.26	5.10	0.145	0.0170	0.0222	0.104	1.23	1.22	1.28	1.26

Table 4: Efficiency: time (in seconds) taken by each model to generate 500 sample time series. Lower ↓ is better.

Model	Time to generate 500 samples (s) ↓
GBM	0.0024
OU_Process	**0.0017**
MJD	0.0023
GARCH11	0.0029
DEJD	0.0078
BlockBootstrap	0.0175
TimeGAN	0.0145
QuantGAN	0.4991
TimeVAE	0.0082
Takahashi	102.51

Table 5: Temporal and diversity metrics by model and channel (lower ↓ is better for differences; higher ↑ is better for diversity distances). Best value in each column bolded.

	ICD-ED ↑				ICD-DTW ↑			
Model	O	H	L	C	O	H	L	C
GBM	0.150	0.133	0.145	0.150	0.100	0.088	0.097	0.099
OU_Process	0.150	0.133	0.145	0.149	0.100	0.088	0.096	0.100
MJD	0.140	0.130	0.143	0.142	0.096	0.092	0.101	0.098
GARCH11	0.090	0.077	0.095	0.088	0.061	0.051	0.064	0.060
DEJD	0.135	0.118	0.132	0.143	0.102	0.091	0.104	0.108
BlockBootstrap	0.138	0.125	0.132	0.146	0.100	0.091	0.097	0.107
TimeGAN	0.133	0.121	0.132	0.131	0.081	0.073	0.080	0.079
QuantGAN	0.137	0.122	0.146	0.130	0.092	0.081	0.097	0.086
TimeVAE	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002
Takahashi	**6.091**	**6.076**	**6.366**	**6.253**	**4.204**	**4.245**	**4.358**	**4.364**

B Visualizations

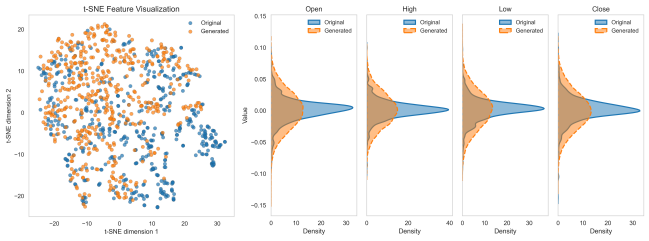


Figure 2: Geometric Brownian Motion (GBM)

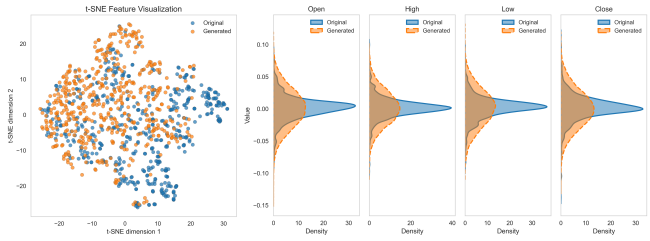


Figure 3: Ornstein-Uhlenbeck (OU) Process

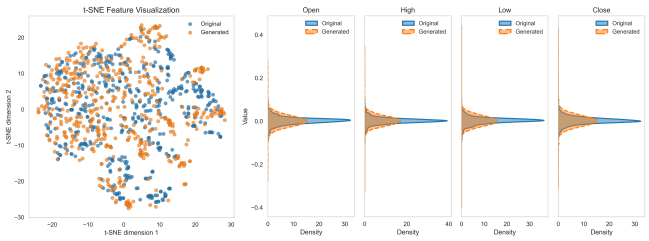


Figure 4: Merton Jump Diffusion (MJD)

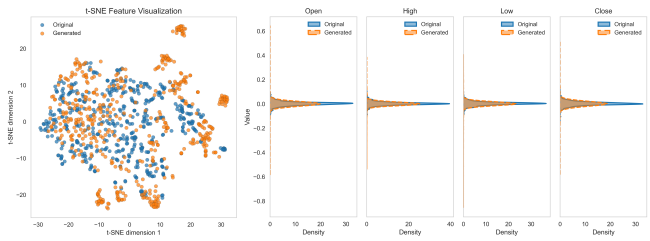


Figure 5: Double-Exponential Jump Diffusion (DEJD)

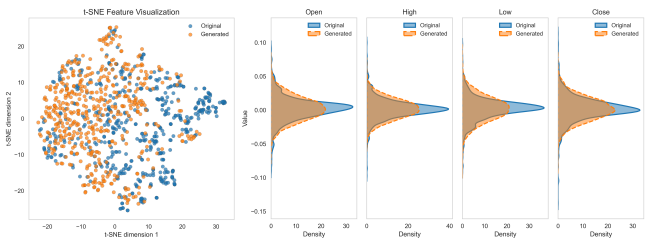


Figure 6: GARCH(1,1)

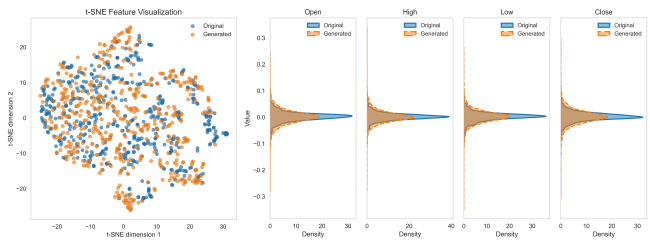


Figure 7: Block Bootstrap

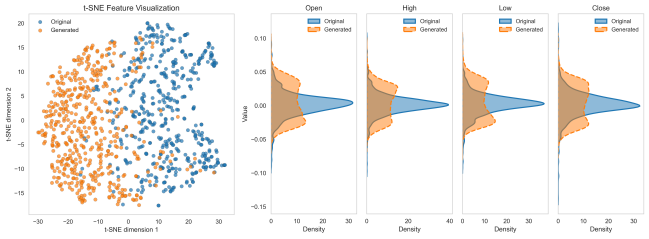


Figure 8: TimeGAN

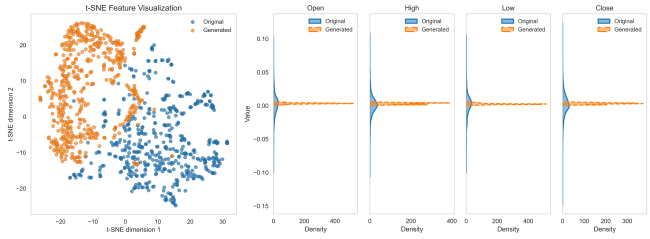


Figure 9: TimeVAE

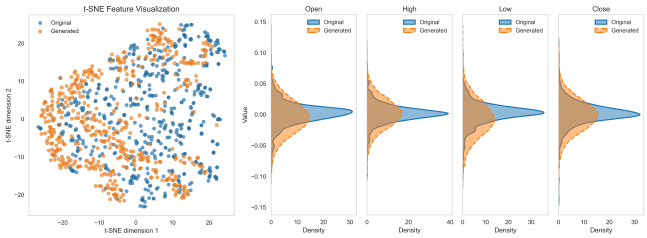


Figure 10: QuantGAN

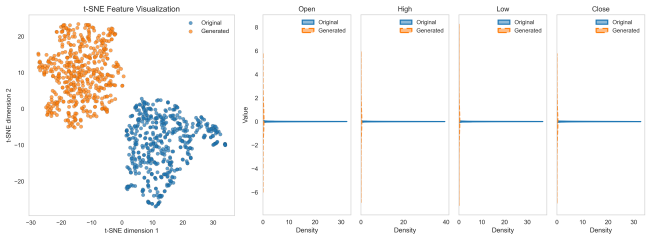


Figure 11: Takahashi DDPM

C Evaluation Metrics Comparisons

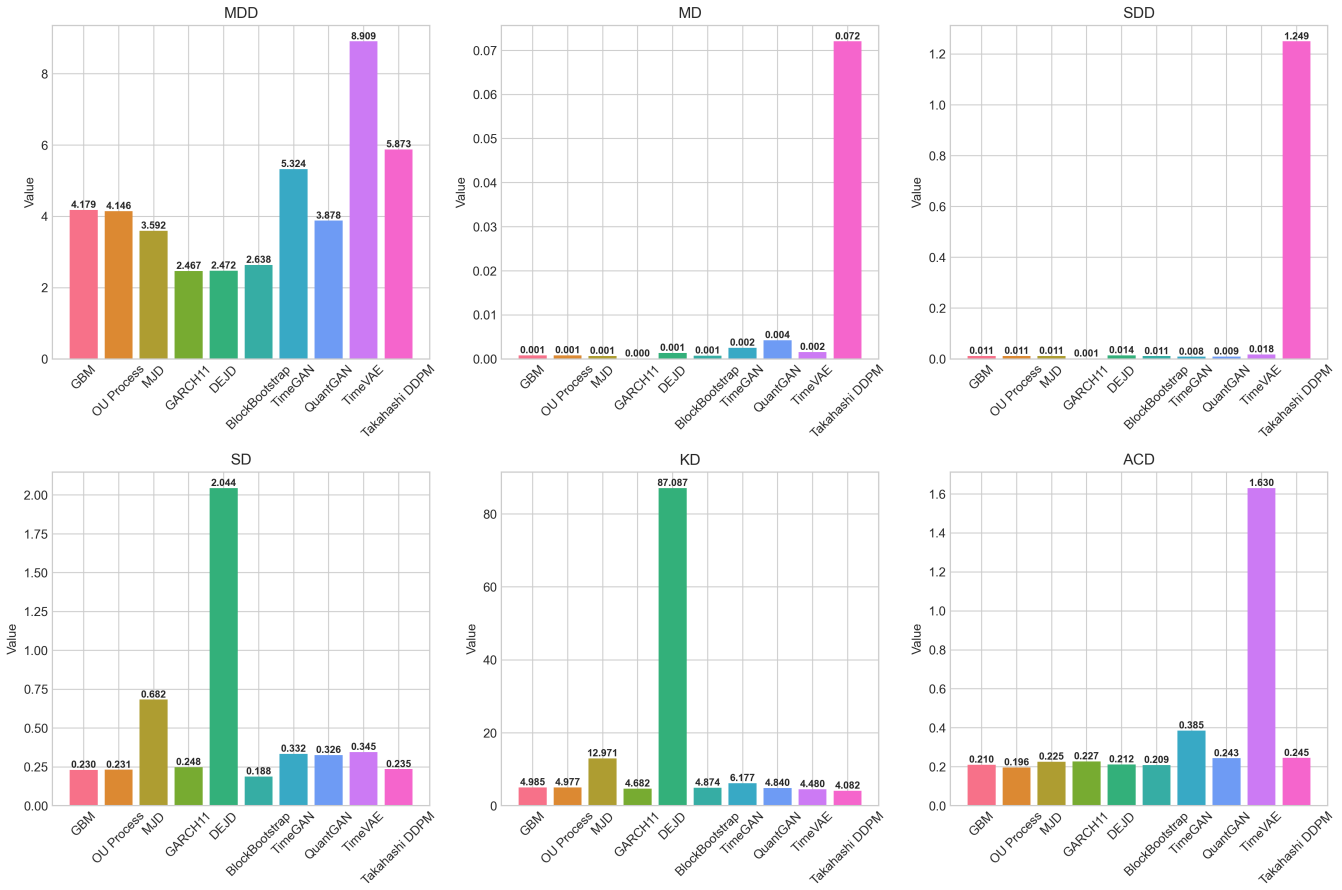
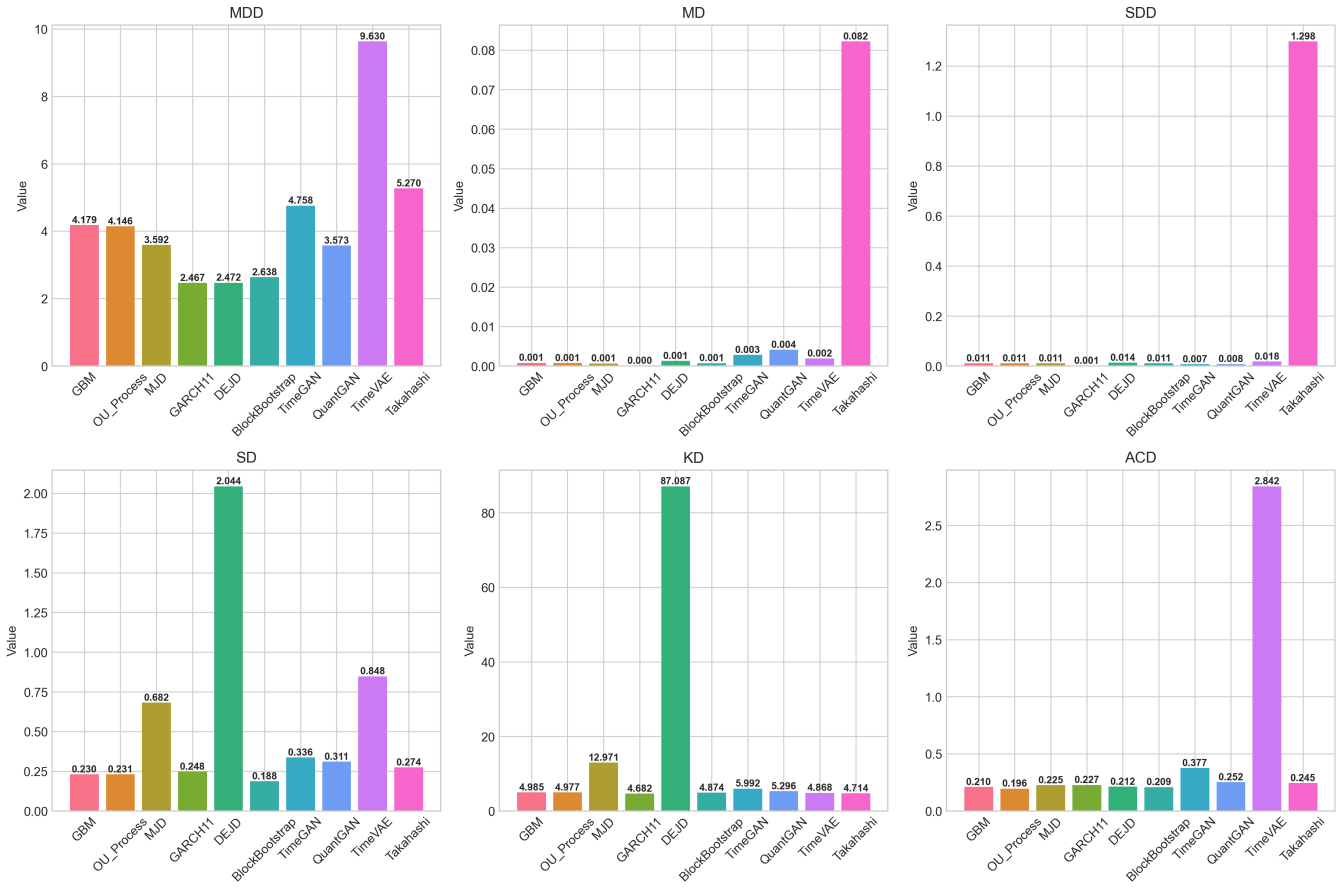


Figure 12: Distribution Metrics Comparison



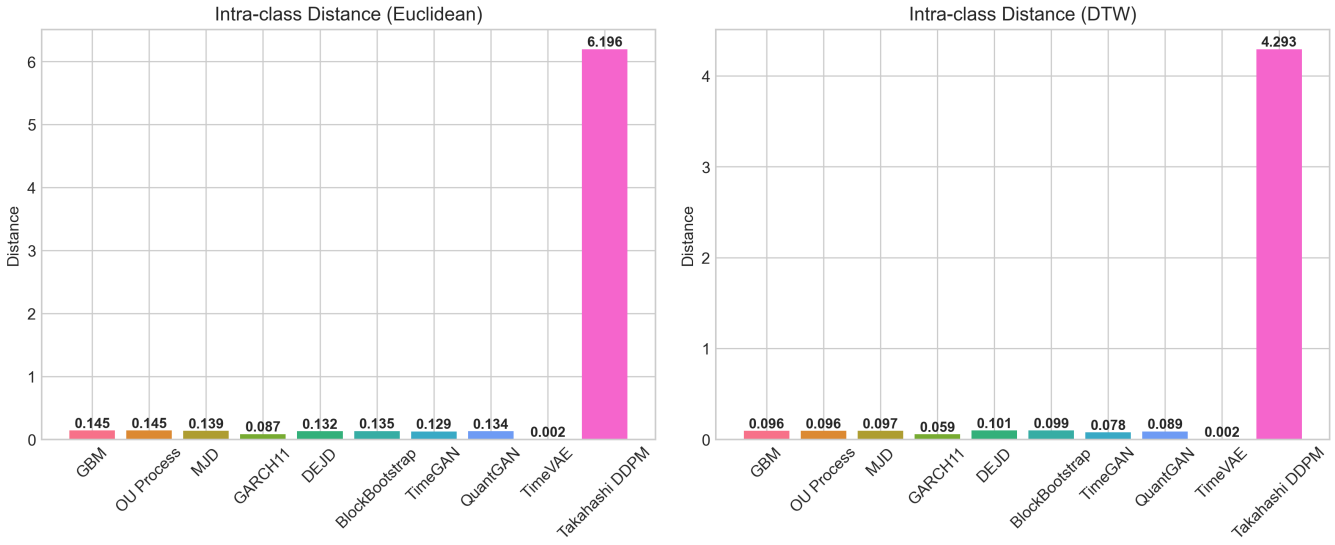


Figure 14: Similarity Metrics Comparison

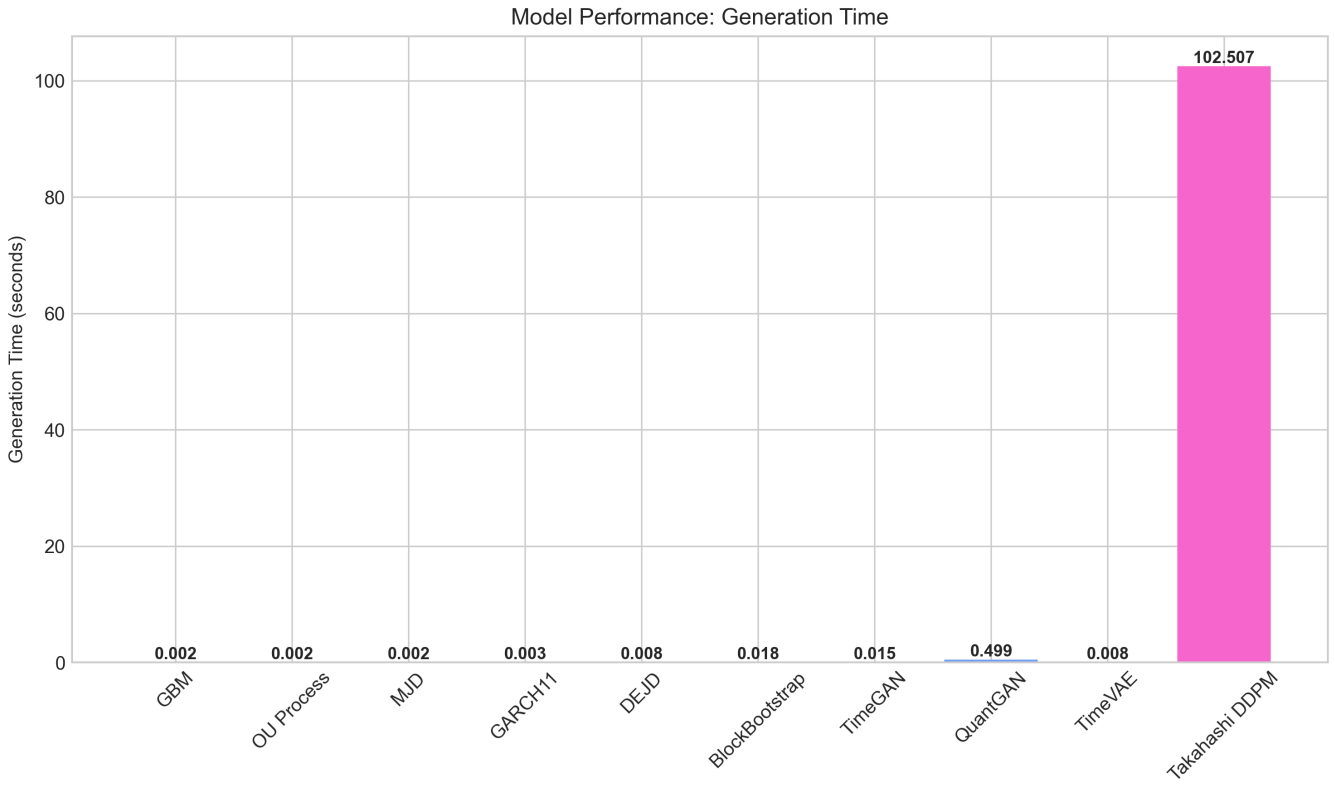


Figure 15: Generation Time Performance

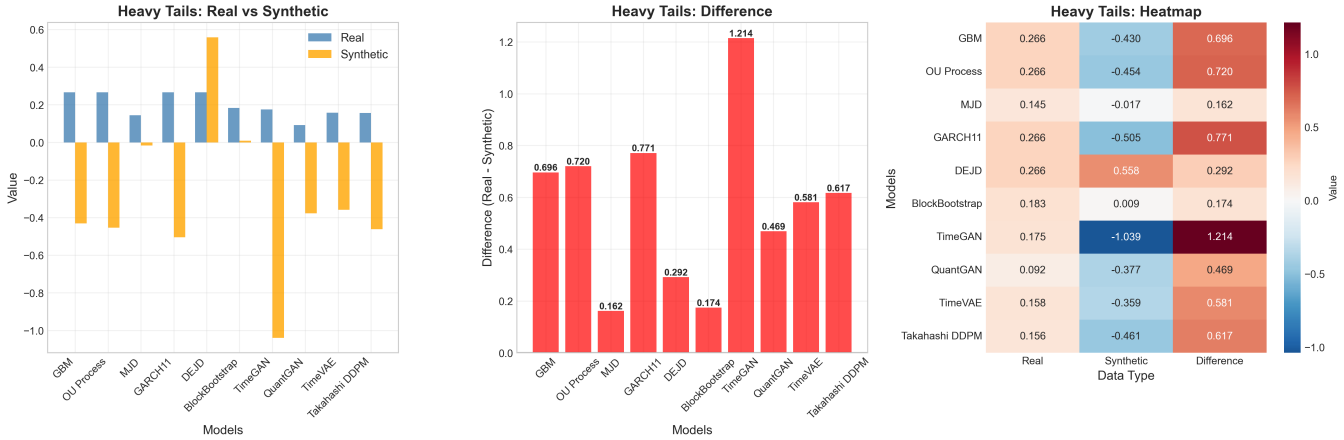


Figure 16: Heavy Tails Stylized Fact

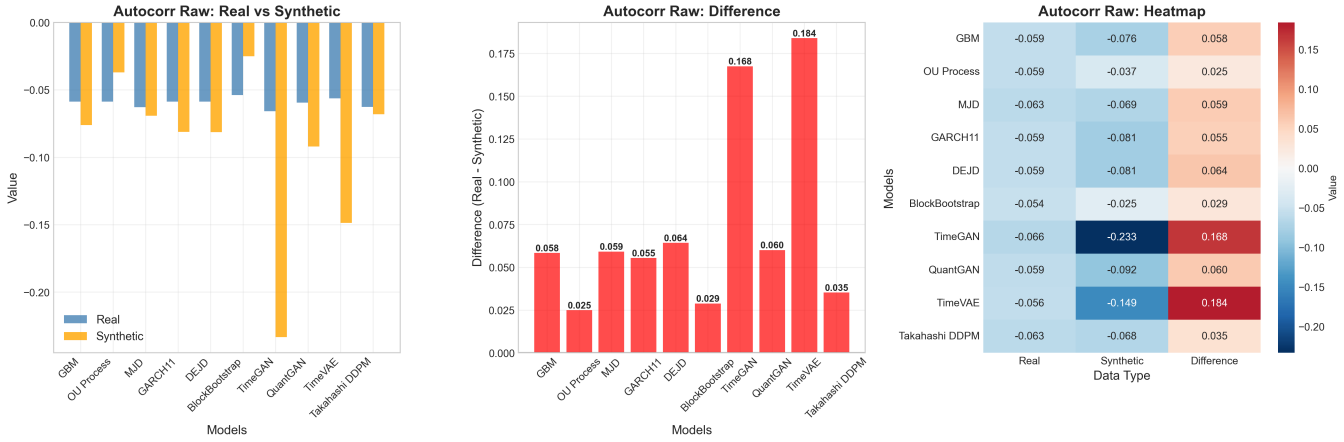


Figure 17: Return Autocorrelation Stylized Fact

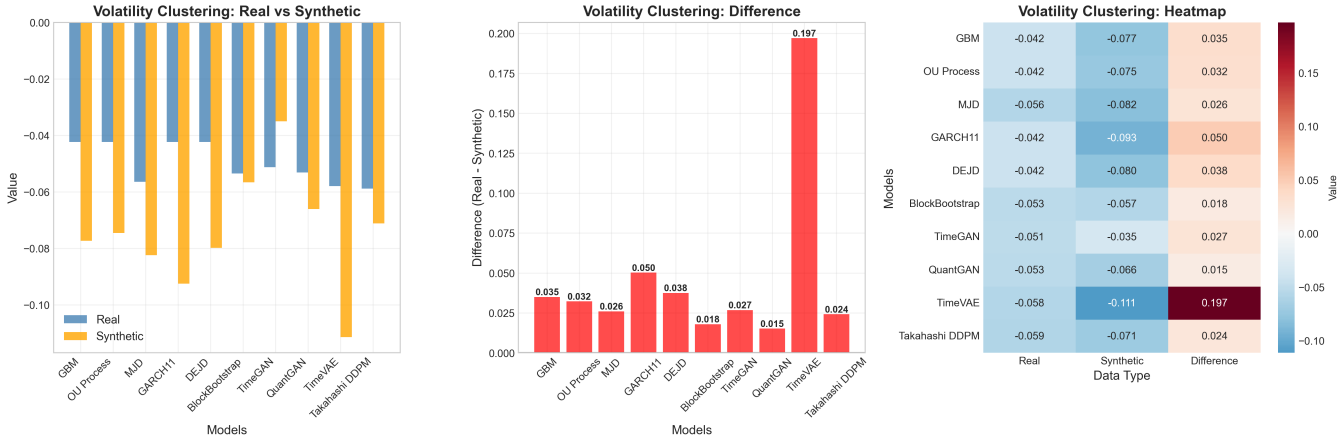


Figure 18: Volatility Clustering Stylized Fact