Institut Polytechnique de Paris

Monte Carlo Methods: from MCMC to Data-based Generative Model

---

Methodology of data generation

Students:
Nasr El Hamzaoui
Martin Boutier

Code is available at https://github.com/nabz78350/MCMC-project

December 21, 2023

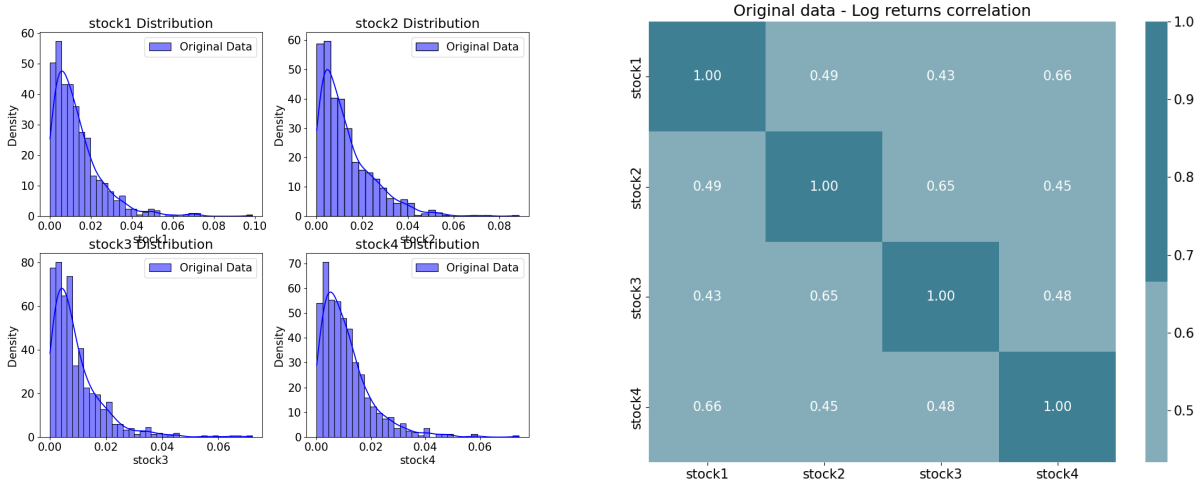
Contents

1 Training Dataset Description

The aim of this project is to simulate negative returns of 4 assets based on a training corpus characterized by the following properties:

	stock1	stock2	stock3	stock4
count	746.000000	746.000000	746.000000	746.000000
mean	0.013144	0.012822	0.009366	0.010788
std	0.011914	0.011712	0.009283	0.009338
min	0.000012	0.000057	0.000014	0.000067
25%	0.004761	0.003878	0.003202	0.004226
50%	0.010030	0.009423	0.006641	0.008508
75%	0.017771	0.017810	0.012354	0.014221
max	0.098709	0.088502	0.072016	0.074291

Table 1: Distribution of the training set



(a) Distribution of the training set

(b) Correlation matrix of the training set

Figure 1: Training Dataset Properties

The returns of the assets were computed on the same day (each row of the dataset represents the four returns for a certain day). So the correlation between the 4 assets must be taken into account in order to have plausible synthetic data.

2 Analytical Solution Using Cholesky Decomposition

The analytical solution to generating synthetic data involves the use of **Cholesky decomposition**, a crucial method for simulating data that maintains the statistical properties of the original dataset. This approach is particularly effective in the context of financial data modeling, where the correlation between different assets is significant.

2.1 Correlation Matrix and Cholesky Decomposition

The first step in this process involves computing the correlation matrix of the original financial data. This matrix, denoted as $\mathbf{R}$, encapsulates the pairwise correlations between the different financial assets in our dataset.

Given the correlation matrix, we apply the Cholesky decomposition. This decomposition transforms $\mathbf{R}$ into a lower triangular matrix $\mathbf{L}$ such that:

$$\mathbf{R} = \mathbf{L}\mathbf{L}^T \tag{1}$$

2.2 Generating Synthetic Data

To generate synthetic data, we start with a sample of independent and identically distributed (i.i.d.) random variables from a normal distribution. This sample is then transformed using the Cholesky matrix $\mathbf{L}$ to introduce the desired correlations. The synthetic data $\mathbf{S}$ is given by:

$$\mathbf{S} = \mathbf{Z}\mathbf{L}^T \tag{2}$$

where $\mathbf{Z}$ represents the matrix of i.i.d. normal random variables.

2.3 Filtering and Adjusting the Synthetic Data

Post-generation, the synthetic data undergoes a filtering process to exclude extreme values, ensuring that the synthetic dataset reflects realistic financial scenarios. Additionally, the synthetic data is adjusted to more closely match the original data's correlation structure, enhancing the authenticity of the simulated dataset.

2.4 Evaluation of the Approach

Finally, the effectiveness of this method is evaluated by comparing the statistical properties, particularly the correlation structure, of the synthetic data against the original dataset. This comparison involves visual and quantitative assessments to ensure the synthetic data accurately reflects the characteristics of the original financial data.

The objective of this experiment is to generate synthetic data that closely resembles the statistical properties of a given financial dataset, specifically in terms of its correlation structure and distribution. The approach involves two key statistical tests: the **Anderson-Darling** distance and the **Kendall Tau** distance.

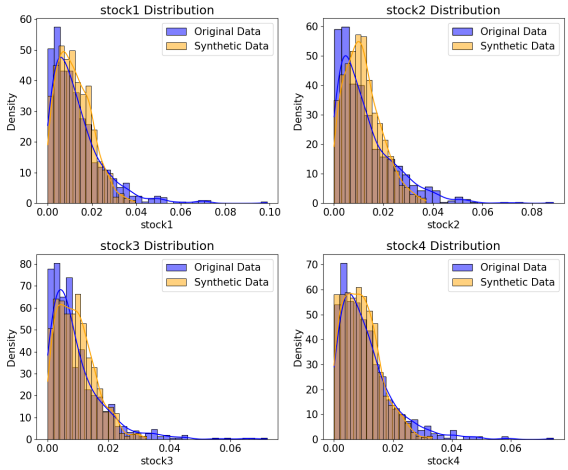
The procedure consists on the following steps, iterated over 200 simulations:

- Randomly sample a subset of the original dataset, of size 410. This sampling introduces variability in the covariance structure, representing different market conditions.
- Compute the covariance matrix, $\mathbf{C}$, for each sampled subset.
- Apply Cholesky decomposition on $\mathbf{C}$ to generate synthetic data. The Cholesky decomposition ensures that the synthetic data maintains the correlation structure of the sampled subset.
- Evaluate the similarity between the synthetic data and the sampled subset using two metrics:

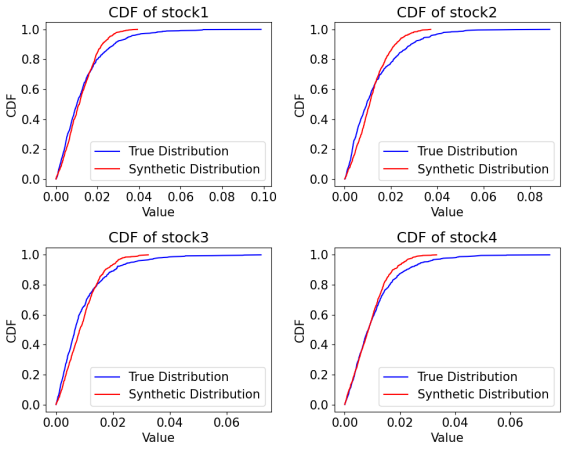
– **The Anderson-Darling Test**, which assesses how well the synthetic data matches the distribution of the original data.

– **The Kendall Tau Distance**, which measures the similarity in rank correlation between the two datasets.

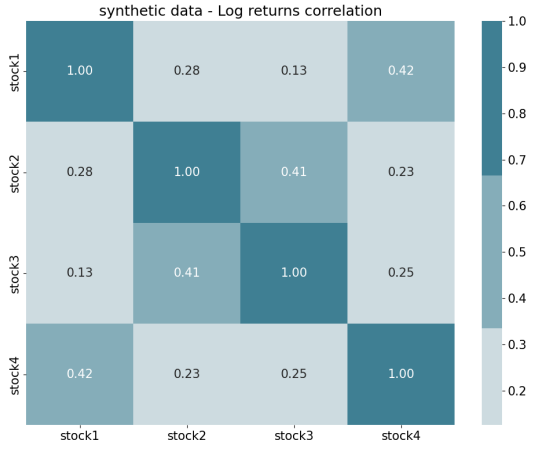
This methodology provides a robust framework for evaluating the quality of synthetic financial datasets, ensuring that they closely mirror the statistical properties of real market data.



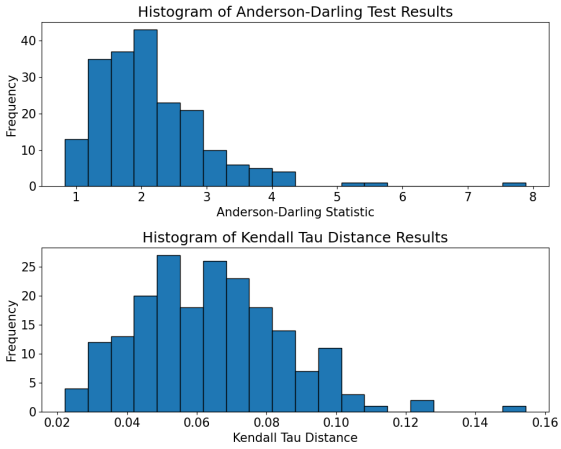
Distribution of synthetic data and original data



Cumulative distribution of synthetic data and original data



Correlation matrix of synthetic data



Histogram of Anderson Darling and Kendall distances over the iterations

Figure 3: Cholesky synthetic data properties

	AndersonDarling	KendallTau
count	200	200
mean	1.521147	0.063962
std	0.615859	0.018951
min	0.434653	0.016224
25%	1.099616	0.049637
50%	1.374299	0.064378
75%	1.784606	0.076926
max	4.919981	0.119883

Table 2: 200 test iterations of the Cholesky method over 410 samples

3 Gaussian Mixture Model (GMM) Approach

3.1 Overview of GMM

The Gaussian Mixture Model (GMM) is a probabilistic model that assumes all the data points are generated from a mixture of several Gaussian distributions with unknown parameters. It's a soft clustering algorithm that assigns a probability to each point for belonging to each of the Gaussian distributions. The Gaussian Mixture Model (GMM) with 20 components is formulated as follows:

$$p(\mathbf{x}) = \sum_{i=1}^{20} \pi_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \tag{3}$$

where $\mathbf{x}$ is the data vector, π_i are the mixture weights satisfying $\sum_{i=1}^{20} \pi_i = 1$, $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ represents the Gaussian components with means $\boldsymbol{\mu}_i$ and covariance matrices $\boldsymbol{\Sigma}_i$.

Each component contributes to the overall model by capturing a subset of the data's structure. The model's parameters, including the means, covariances, and mixture weights, are typically estimated using the **Expectation-Maximization (EM)** algorithm.

3.2 Assumptions

When applying GMM to financial data, the key assumption is that the data can be modeled using Gaussian distributions. This approach is particularly useful due to its simplicity and the natural fit for many types of financial data.

3.3 Covariance Structure

The covariance type 'full' in the GMM indicates that each component has its own general covariance matrix. This choice allows for the capturing of a more comprehensive relationship between variables in the data, which is crucial for financial datasets where asset relationships can be complex.

3.4 Expectation-Maximization (EM) Algorithm

GMMs utilize the Expectation-Maximization (EM) algorithm for parameter estimation. The EM algorithm iteratively improves the parameter estimates, with two main steps:

1. **Expectation (E) step:** Calculate the expected value of the log likelihood, given the current estimate of the parameters.

2. **Maximization (M) step:** Maximize the expected log likelihood found in the E step.

3.5 Recursive Training Algorithm

The implemented algorithm involves the following steps:

1. Split the dataset into training and testing subsets.

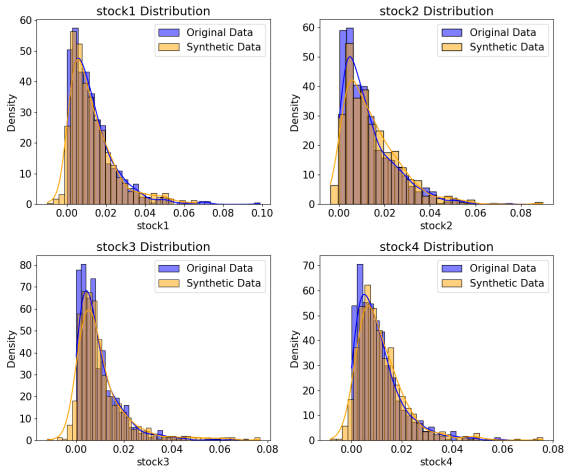
2. Fit the GMM on the training data, potentially initializing with the best weights from previous iterations.

3. Generate synthetic data using the fitted model.

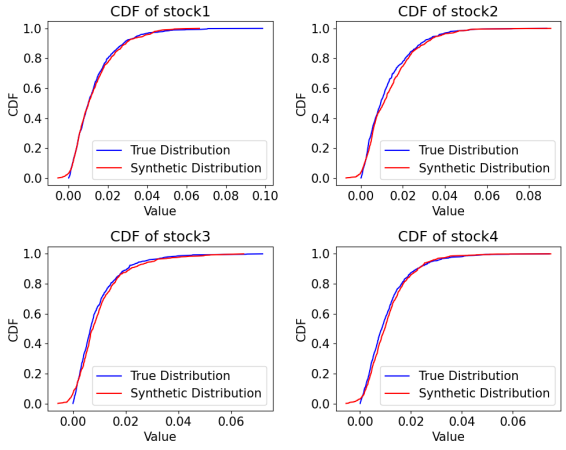
4. Calculate the Anderson-Darling distance and Kendall Tau distance between the test data and the synthetic data.

5. Update the best model parameters if the current model outperforms the previous best in terms of the Anderson-Darling distance.

This recursive training process aims to minimize the Anderson-Darling distance, continuously improving the GMM's fit to the data.



GMM - distribution of synthetic data and original data



GMM - cumulative distribution of synthetic data and original data

Figure 4: Gaussian Mixture Model

We repeated the procedure, performing 200 iterations in which we sampled noise for 410 data points. For each iteration, we used the GAN to generate synthetic data from this noise, followed by calculating the Anderson-Darling distance and the Kendall Tau distance.

	AndersonDarling	KendallTau
count	200	200
mean	1.219633	0.065480
std	0.736206	0.020026
min	0.170186	0.017467
25%	0.651728	0.052897
50%	1.069109	0.063767
75%	1.633829	0.079114
max	4.519083	0.122324

Table 3: 200 test iterations of the GMM method over 410 samples

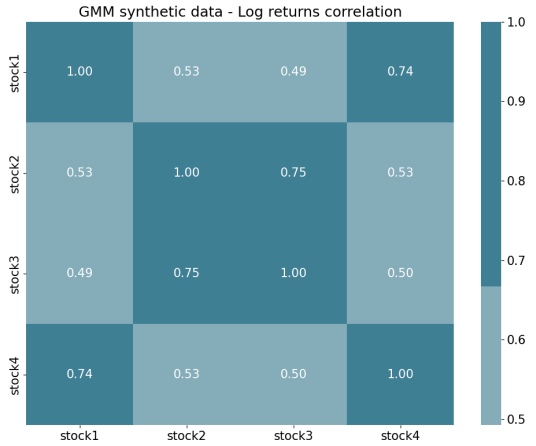


Figure 5: GMM - Correlation matrix of synthetic data

	stock1	stock2	stock3	stock4
count	746.000000	746.000000	746.000000	746.000000
mean	0.013164	0.013078	0.009731	0.011259
std	0.011667	0.012428	0.010075	0.009649
min	-0.004159	-0.004370	-0.006576	-0.007873
25%	0.005026	0.004366	0.003548	0.004960
50%	0.010366	0.008894	0.007186	0.009234
75%	0.017774	0.019914	0.012764	0.015271
max	0.068125	0.089732	0.067345	0.074802

Table 4: Distribution statistics for the GMM synthetic data with 746 samples

We note that certain predictions yield negative values, an outcome that seems inescapable under the stringent assumption that the true distribution follows a Gaussian model.

4 Generative Adversarial Network (GAN)

4.1 Introduction

Generative Adversarial Networks (GANs) represent a class of artificial intelligence algorithms used in unsupervised machine learning, implemented by a system of two neural networks contesting with each other in a zero-sum game framework. This technique can generate data with

the same statistics as the training set. For this project, GANs were chosen due to their effectiveness in generating complex, high-dimensional data distributions, making them suitable for tasks such as synthetic data generation in financial markets.

4.2 GAN Architecture

Our GAN architecture comprises two main components: the Generator and the Discriminator, both constructed as neural networks with specific configurations.

4.2.1 Generator

The generator network aims to generate data that is indistinguishable from real data. It takes a latent space vector as input and outputs data in the desired format. The architecture of the Generator is defined as follows:

- Input: Latent space vector of dimension latent_dim $\times$ 3.
- Network Structure: A series of fully connected layers, each followed by Batch Normalization, with the number of neurons and activation functions specified per layer.
- Output: Data in the target shape, passed through an output activation function.

4.2.2 Discriminator

The discriminator network is a binary classifier whose task is the distinction between real and fake (generated) data. Its architecture is as follows:

- Input: Data in the target shape.
- Network Structure: Similar to the generator, comprising fully connected layers with Batch Normalization.
- Output: A single neuron with a binary classification activation function (e.g., sigmoid) indicating real or fake data.

4.3 Noise Generation and Moment Matching

In the context of our GAN framework, the noise vectors $\mathbf{x}$ are generated from a multivariate Gaussian distribution, crucial for the diversity and richness of the generated synthetic data. Specifically, we utilize a multivariate Gaussian distribution with a mean vector $\boldsymbol{\mu}$ of zeros and a *Toeplitz* covariance matrix $\boldsymbol{\Sigma}$, which is a function of the latent dimension size, determined to be 100 as a result of extensive grid search optimization.

4.3.1 Covariance Structure

The Toeplitz structure of the covariance matrix ensures consistent correlation properties across the noise vector, with the correlation coefficient ρ between elements i and j of the noise vector given by:

$$\text{corr}(x_i, x_j) = \rho^{|i-j|}, \quad \text{where } \rho = 0.75. \tag{4}$$

This correlation structure introduces a controlled decay of correlation as the distance between the elements increases, emulating the temporal decay in autocorrelation often observed in financial time series.

4.3.2 Inclusion of Higher Moments

To better capture the statistical properties of financial market returns, we augment the noise vector with its square and cube, corresponding to the second and third moments—variance and skewness—respectively:

$$\mathbf{x}_{\text{augmented}} = [\mathbf{x}, \mathbf{x}^2, \mathbf{x}^3]. \tag{5}$$

This augmentation ensures that our GAN accounts for both the magnitude of fluctuations (variance) and the asymmetry of the return distribution (skewness), which are essential characteristics of asset returns.

4.3.3 Noise Generation Algorithm

The algorithm to generate the noise vectors is as follows:

```
def generate_noise(n_samples):
    # Toeplitz covariance matrix with decay factor rho
    covariance_matrix = 0.75 ** np.abs(np.subtract.outer(
        np.arange(opt.latent_dim), np.arange(opt.latent_dim)))
    # Sample from the multivariate normal distribution
    noise = np.random.multivariate_normal(
        mean=np.zeros(opt.latent_dim), cov=covariance_matrix, size=n_samples)
    # Augment noise with squared and cubed terms
    squared_noise = noise**2
    cube_noise = noise**3
    noise_augmented = np.concatenate([noise, squared_noise, cube_noise], axis=1)
    return noise_augmented
```

4.4 Scaling of Training Data

A significant enhancement in our GAN's performance was observed by scaling the training data. This process involved multiplying all values in the training set by a factor of 100, effectively transforming the data to operate within the unit interval. Such normalization is a common practice in machine learning to stabilize and speed up the training of neural networks.

Impact on GAN Predictions: The scaling of input data necessitates a corresponding adjustment in the GAN's output to ensure the synthetic data generated is comparable to the original scale of the financial market data. To address this, the outputs of the GAN are divided by 100. This rescaling ensures that the synthetic data generated by the GAN aligns accurately with the real-world data scale.

Pre-Processed Synthetic Data: In line with this adjustment, the file 'submissions/synthetic_data.csv' contains the final synthetic data, which has already been rescaled back to the original data scale. This pre-processing step simplifies the integration of GAN-generated data with existing financial data analysis pipelines, ensuring seamless usability and compatibility.

4.5 Training the GAN

Training involves alternating between updating the discriminator to better differentiate real data from fake, and updating the generator to produce increasingly realistic data. The training process is summarized as follows:

1. Generate a batch of fake data using the generator.
2. Sample a batch of real data.
3. Train the discriminator on both this real and fake data.
4. Generate a new batch of fake data and use it to train the generator, with the goal of fooling the discriminator.

4.6 Model Evaluation

Each GAN configuration is evaluated based on its ability to replicate the statistical properties of real financial market data. This is quantified using the Anderson-Darling distance and the Kendall Tau distance, calculated between the synthetic and real data distributions. These metrics provide a measure of similarity, with lower values indicating better model performance.

4.7 Optimization and Results

A grid search is conducted over various hyperparameters, including the number of layers, neuron count, and activation functions for both the generator and discriminator, as well as the dimensionality of the latent space. The optimal configuration is determined based on the lowest average Anderson-Darling and Kendall Tau distances across multiple evaluations.

4.7.1 Best Performing Configuration

The best-performing GAN configuration in our experiments is characterized by the following architecture:

Generator:

- Number of Layers: 1
- Neurons per Layer: [16]
- Hidden Activation: ['relu']
- Output Activation: 'softplus'

Discriminator:

- Number of Layers: 1
- Neurons per Layer: [64]
- Hidden Activation: ['leaky_relu']
- Output Activation: 'sigmoid'

General Configuration:

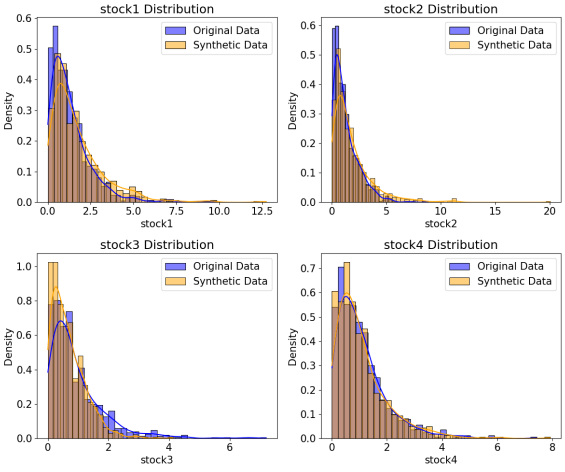
- Latent Dimension: 100

This model configuration achieved an Anderson-Darling distance of 0.375017 and a Kendall Tau distance of 0.080277, indicating its high effectiveness in generating synthetic data that closely matches the statistical properties of the real financial market data. Note that if the latent dimension is 100, the total noise is of dimension 300 after we added square and cube noise.

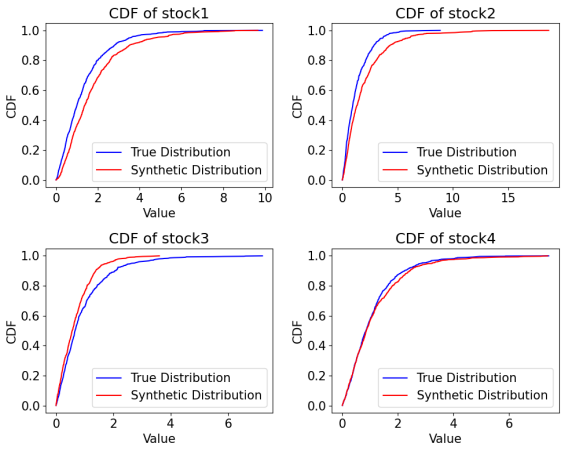
We repeated the process of 200 test iterations, each time generating samples of size 410, and computing distance metrics. Here are the results we obtained with our best GAN model

	AndersonDarling	KendallTau
count	200	200
mean	0.428039	0.088514
std	0.298841	0.022480
min	0.066944	0.039216
25%	0.240877	0.072494
50%	0.339624	0.087447
75%	0.526102	0.102222
max	2.309868	0.162861

Table 5: 200 test iterations of the GAN model over 410 samples

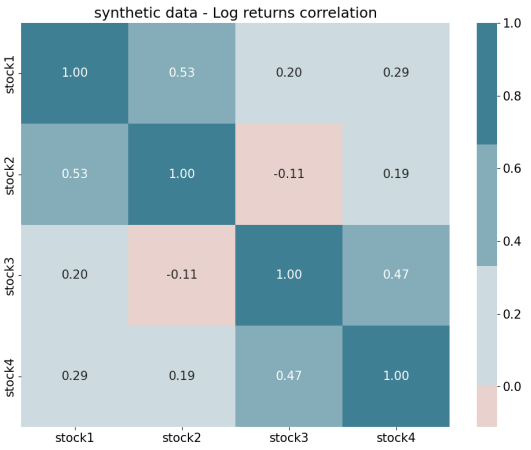


GAN - distribution of synthetic data and original data

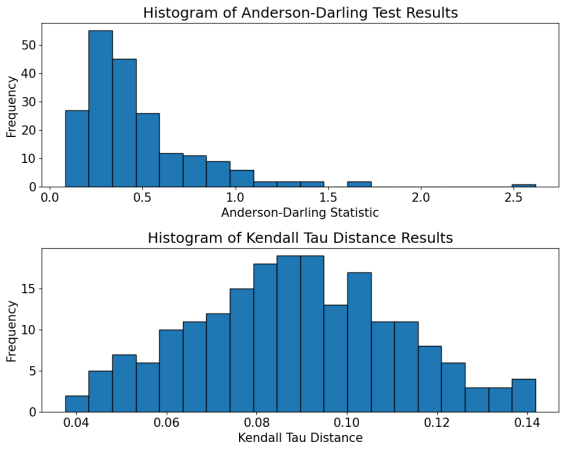


GAN - cumulative distribution of synthetic data and original data

Figure 6: GAN



Correlation matrix of synthetic data



Histogram of Anderson Darling and Kendall distances over the iterations

Figure 7: GAN synthetic data properties

5 Conclusion

We have explored three distinct methods: the Cholesky-based simple generation, a Gaussian Mixture Model (GMM), and a Generative Adversarial Network (GAN).
The Cholesky approach exhibits robust performance, effectively replicating the correlation patterns found in actual data. However, its heavy dependence on a dataset of merely 746 samples poses a critical drawback. This overreliance potentially leads to excessively supervised outcomes, which might not be representative or scalable to broader contexts.
The Gaussian Mixture Model (GMM) also presents encouraging outcomes. Its primary issue, however, lies in its propensity to produce negative values, which may not align with certain types of real-world data. Additionally, the GMM is predicated on a substantial assumption: the data is a composite of Gaussian random variables. This assumption might hold some validity for intraday returns due to their relatively lower tail risk. But for daily financial returns, characterized by significant fat tails, this model becomes less convincing. A more realistic approach might be a leptokurtic distribution centered at zero, yet even this assumption is considerably stringent.
In contrast, the Generative Adversarial Network (GAN) was selected as the model for our project submission. Unlike the Cholesky method, the GAN does not heavily rely on the training data's specific characteristics and avoids the stringent assumptions inherent in the GMM. Importantly, when equipped with appropriate activation functions, the GAN can generate exclusively positive values, aligning more closely with certain types of real data. Enhancements in the GAN's performance were achieved by integrating Batch Normalization layers and amplifying the input by a factor of 100. A comprehensive, parallelized grid search across over 500 different architectural configurations revealed a consistent trend: simpler models generally delivered superior results. This finding aligns logically with the limited scope of our training dataset, which includes just 746 data points.