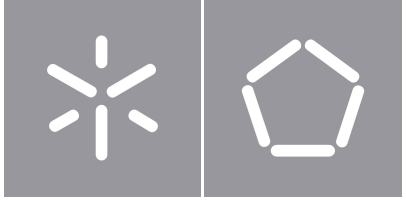
University of Minho
School of Engineering

Luís Henrique Ferreira Amorim

Grasynda: Graph-based Synthetic Time Series Data Augmentation

july 2025



University of Minho
School of Engineering

Luís Henrique Ferreira Amorim

Grasynda: Graph-based Synthetic Time Series Data Augmentation

Master's Dissertation in Physics Engineering

Dissertation supervised by
Professor Paulo Azevedo (1)
Dr. Vítor Cerqueira (2)

(1) Departamento de Informática da Universidade do Minho, Braga, Portugal
(2) Faculdade de Engenharia da Universidade do Porto, Porto, Portugal

july 2025

Copyright and Terms of Use for Third Party Work

License granted to users of this work:

Acknowledgements

First and foremost, I would like to express my deepest gratitude to Professor Carlos Soares, Professor Paulo Azevedo, and Dr. Vítor Cerqueira for their invaluable supervision throughout this project. I am especially thankful to them for the opportunity to develop this work, as well as for their continuous support and guidance. Their insights and mentorship were crucial to the success of this thesis.

I would like to thank my family for their unwavering support and essential encouragement. To my close friends, I also want to say I'm truly grateful, their presence has made these years not only unforgettable but also the best of my life. Finally, I want to give a special thanks to Diogo, Carolina and Eduardo for the countless laughs and fun we shared during our master's degree. This journey would have been much more difficult without you.

This work is a result of Agenda "Center for Responsible AI", nr. C645008882-00000055, investment project nr. 62, financed by the Recovery and Resilience Plan (PRR) and by European Union - NextGeneration EU. Funded by the European Union – NextGenerationEU. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them. This work is also funded by the European Union under the Horizon Europe Framework Programme Grant Agreement N°: 101095387. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the European Health and Digital Executive Agency can be held responsible for them, and by: UID/00027 of the LIACC - Artificial Intelligence and Computer Science Laboratory - funded by Fundação para a Ciência e a Tecnologia, I.P./ MCTES through the national funds.

Statement of Integrity

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, Braga, july 2025

Assinado por: Luís Henrique Ferreira Amorim
Identificação: BI15393321
Data: 2025-07-11 às 12:23:16

Luís Henrique Ferreira Amorim

Abstract

Data augmentation is a crucial tool in time series forecasting, especially for deep learning architectures that require large training samples to generalize effectively. However, extensive datasets are not always available in real-world scenarios. While many state-of-the-art data augmentation methods exist, they face crucial limitations including high computational costs, limited control over variability, or poor generalization across tasks. This paper introduces `Grasynda`, a novel graph-based approach to synthetic time series generation designed to address these challenges. `Grasynda` represents univariate time series as quantile graphs and encodes their temporal dynamics using a transition probability matrix. This representation enables the generation of realistic synthetic sequences that maintain the statistical properties and temporal patterns of the original data, while introducing controlled variability. We performed an extensive evaluation using three neural network variations and six benchmark datasets. The results indicate that `Grasynda` consistently outperforms other time series data augmentation methods. The method and all experiments are publicly available.

Keywords Time series, Synthetic data, Forecasting, Data Augmentation.

Resumo

A capacidade de prever o comportamento de séries temporais é fundamental em diversos domínios como indústria, finanças e saúde. Com o avanço das técnicas de deep learning, têm-se alcançado progressos significativos nesta área. No entanto, estas abordagens dependem de grande volume e variedade de dados para treinar modelos eficazes e generalizáveis. Apesar desta dependência, os dados temporais são frequentemente limitados ou incompletos, em contextos reais. Neste contexto, o aumento de dados (data augmentation) surge como uma solução promissora para colmatar esta limitação, ao permitir gerar novas amostras a partir de dados existentes. Ainda assim, os métodos atuais de aumento de dados enfrentam vários desafios, nomeadamente elevados custos computacionais, dificuldade em controlar a variabilidade das amostras geradas e fraca capacidade de generalização entre diferentes tarefas e conjuntos de dados.

Esta tese apresenta `Grasynda`, um novo método de geração sintética de séries temporais baseado em grafos. A abordagem tem por base a representação de séries temporais univariadas em grafos de quantis. Os grafos capturam os padrões temporais através de uma matriz de probabilidades de transição. Esta representação permite gerar sequências sintéticas que mantêm as propriedades essenciais dos dados originais, enquanto introduz variação de forma controlada. O modelo `Grasynda` tem como objetivo melhorar em relação a algumas das limitações dos métodos existentes.

A eficácia do método foi validada através de uma avaliação experimental com três redes neuronais de previsão e seis conjuntos de dados de referência amplamente utilizados na literatura. Os resultados obtidos sugerem que `Grasynda` obtém um desempenho superior a outras abordagens na melhoria do erro dos modelos de previsão. Todas as experiências realizadas, bem como o código desenvolvido, estão disponíveis publicamente.

Palavras-chave Séries temporais, Previsão, Dados Sintéticos

Contents

I Introductory material 1

1 Introduction 2

1.1 Time Series Forecasting . . . 2
1.2 Data Augmentation . . . 3
1.3 Contribution . . . 3

2 State of the Art 6

2.1 Time series . . . 6
2.1.1 Time Series Forecasting . . . 6
2.1.2 Time series Forecasting models . . . 7
2.2 Data Augmentation . . . 10
2.2.1 Data Augmentation Techniques . . . 11
2.3 Time Series Analysis using Network Science . . . 13
2.4 Motivation . . . 14

II Core of the Dissertation 15

3 Methodology 16

3.1 Time Series Quantile Graph . . . 16
3.1.1 Quantile Discretization . . . 16
3.1.2 Graph Construction . . . 17
3.2 Time series generation . . . 17
3.2.1 Quantile Series Generation . . . 17
3.2.2 Time Series Generation . . . 18

3.3 `Grasynda(E)`: transition matrix ensembles . . . 18
3.4 Handling non-stationarity . . . 19

4 Experiments **21**
4.1 Experimental Design . . . 21
4.1.1 Datasets . . . 21
4.1.2 Parameter setup . . . 22
4.1.3 Performance estimation . . . 22
4.2 Methods . . . 23
4.2.1 Neural Networks and Baseline . . . 23
4.2.2 Synthetic time series generators . . . 23

5 Results **24**
5.1 Main results . . . 24
5.2 Sensitivity Analysis . . . 26
5.3 Discussion . . . 27

6 Conclusion **29**

III Appendices **35**

A Details of results **36**

List of Figures

1 Example plot of a time series in the time domain (a); the corresponding network representation using a quantile graph (b); and the corresponding graph transition matrix used for data generation in `Grasynda` (c). 4

2 Illustration of the time-series to graph conversion scheme from (7), depicting the transformation of temporal data into a graph probability matrix representation. 14

3 Workflow behind `Grasynda` for building a quantile graph from a time series and using it to create synthetic time series. 16

4 Example plot of a time series in the time domain (a); the corresponding network representation using a quantile graph (b); and a synthetic time series generated from the graph using `Grasynda` (c). 20

5 Median MASE values for different quantile parameters across different forecasting models. 26

6 Median MASE values for different ensemble parameters across forecasting models. . . . 27

7 forecasting architetures mean MASE by dataset and data augmentation model 36

8 Average MASE across forecasting models, for different data augmentation methods. . . 36

9 Average rank across forecasting models, for different data augmentation methods. . . . 36

10 Average rank for different number of quantiles. 37

11 Example plot of generated vs original time series for two unique ID's. 37

List of Tables

1 Summary of the datasets: average value, number of time series, number of observations, seasonal period, and forecasting horizon (h). 21

2 MASE of each approach across different neural networks and datasets, and corresponding model average and average rank. Best and second-best values are **bolded** and underlined in the respective dataset and neural network architecture. The '*' symbol denotes a significantly better performance of the respective method relative to `Original`. 25

Part I

Introductory material

Chapter 1

Introduction

This dissertation is organized into six chapters with the following organization. Chapter 1 outlines the importance of series forecasting in data-limited scenarios the motivation for developing a novel approach to synthetic data generation. A summary of the main contributions of this work is also provided. Chapter 2 reviews the relevant literature, covering traditional and deep learning forecasting models, existing data augmentation techniques, and state of the art developments in graph-based time series analysis. Chapter 3 presents the proposed methodology for `Grasynda`, detailing the process of transforming time series into graph representations and generating synthetic sequences through the probability matrix. Chapter 4 describes the experimental setup: datasets, forecasting models, parameter settings and evaluation metrics employed. In Chapter 5 `Grasynda`'s performance is compared to that of other augmentation methods and its effectiveness across different scenarios is analyzed, then the results obtained are presented and discussed. Finally, Chapter 6 concludes the dissertation by summarizing the key findings and suggesting directions for future research.

1.1 Time Series Forecasting

Time series forecasting is a critical task across a wide range of domains, including finance, industry, and retail. Accurate forecasts support decision-making in inventory management, energy consumption planning, and predictive maintenance, among many other applications. Traditionally, statistical models such as ARIMA and exponential smoothing (20) have been widely used. However, in recent work deep neural networks have emerged as state the art solutions for forecasting tasks, capable of modeling complex nonlinear dependencies and handling high-dimensional, multivariate inputs. Architectures like NHITS (10) and N-BEATS (33) have achieved state-of-the-art performance in forecasting competitions and benchmark datasets.

These deep learning methods typically require a sufficiently large and diverse training sample to per-

form effectively. In many real-world scenarios, time series datasets are limited in size, either having a small number of time series or containing series with few observations (3). Moreover, time series often exhibit non-stationarity, noise, and structural variability, which makes learning reliable patterns even more challenging in low-data regimes. These limitations pose a barrier to the application of modern forecasting architectures in practical settings, and highlight the need for effective strategies to improve generalization when data is scarce or fragmented.

1.2 Data Augmentation

Synthetic time series data augmentation methods tackle the issue of limited data availability by increasing the training set with artificial but realistic time series. Various data generation methods have been developed for time series, ranging from simple approaches such as jittering to more sophisticated techniques involving pattern mixing (averaging multiple time series) (17) or generative models (46).

Despite the various approaches in the literature, existing time series generation methods face important limitations. Simple techniques often fail to preserve temporal dependencies or introduce meaningful yet realistic diversity. As a result, these methods may generate data that does not contribute to improved forecasting performance. On the other hand, sophisticated generative models, such as generative adversarial networks (46), typically require substantial computational resources and are prone to issues such as mode collapse or overfitting, especially when applied to small training sets. Moreover, many of these methods operate as black boxes and offer little control or interpretability over the generation process.

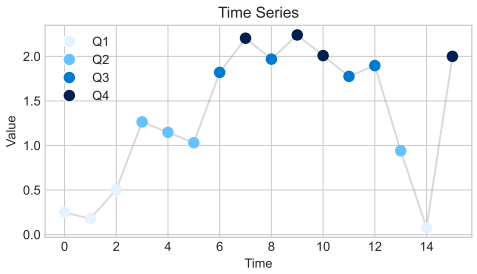
These challenges create a gap between the need for robust data augmentation and the current capabilities of available methods. In practice, there remains a need for synthetic data generation approaches that are consistent, controllable, and accurately replicate the characteristics of time series data. In particular, to improve downstream forecasting tasks.

1.3 Contribution

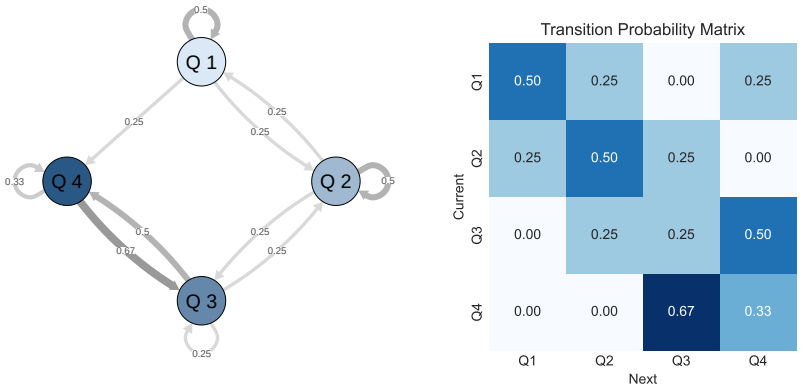
We address these limitations with `Grasynda` (GRAph-based SYNthetic time series DAta generation), a novel approach that creates synthetic data by transforming time series into a network representation using quantile graphs (7). Our approach builds upon the framework of time series graph duality (7), which establishes a bijective mapping between time series and their graph representations. Our working hypothesis is that this transformation encodes the temporal dynamics into a transition probability matrix that effectively captures the underlying data generation process. The graph representation inherently captures both local

patterns (immediate state transitions) and global structures (overall network connectivity patterns).

Figure 1 shows an example time series in time domain (a), the corresponding network representation using a quantile graph (b), and the graph quantile transition matrix (c).



(a) Example time series in the time domain



(b) Time series quantile graph representation (c) Quantile transition matrix

Figure 1: Example plot of a time series in the time domain (a); the corresponding network representation using a quantile graph (b); and the corresponding graph transition matrix used for data generation in `Grasynda` (c).

The transition matrix provides a compact and interpretable structure that summarizes the dynamics of time series. By sampling from the transition matrix according to the encoded probabilities, `Grasynda` generates new synthetic time series that maintain statistical fidelity to the original data while introducing controlled variability.

This thesis addresses the following key questions:

- **RQ1**: How can we transform time series into a graph-based representation that effectively captures their temporal dynamics while also handling non-stationarity variations?
- **RQ2**: How effective is this graph-based representation for generating synthetic time series and augmenting training datasets to improve forecasting performance?

- **RQ3**: How does the proposed method compare with existing synthetic data generation approaches in terms of downstream forecasting performance across different neural network architectures?

We address these questions with extensive experiments using six benchmark datasets and three neural network architectures. The results suggest that augmenting time series datasets using `Grasynda` improves the forecasting accuracy of different neural networks. The proposed method also shows competitive forecast accuracy when compared with other data augmentation methods. The experiments and method are publicly available online.[1]

In summary, the contributions of this work are:

- `Grasynda`, a novel graph-based method for synthetic time series generation that transforms temporal dynamics into network structures via quantile graphs;
- Extensive empirical evaluation showing that `Grasynda`-generated synthetic data improves forecasting accuracy across six benchmark datasets and three neural network architectures.

[1] https://github.com/GrasyndaDataAugmentation/Grasynda.git

Chapter 2

State of the Art

This chapter provides theoretical and methodological foundations for understanding the contributions of the thesis. We begin by reviewing the fundamental concepts of time series data, forecasting problem formulation, and state of the art forecasting architectures. Emphasis is placed on the limitations and necessities of each method when trying to compute meaningful predictions.

Sequentially, we present an overview of data augmentation methods for time series, highlighting the main advantages and limitations of different approaches, particularly in the context of data-limited forecasting tasks. Given that `Grasynda` foundation is a graph-based representation of time series, we review the main work on time series analysis using network science.

The chapter ends with a discussion on how current methods are evaluated and we identify the key research gaps that motivate the development of `Grasynda`.

2.1 Time series

2.1.1 Time Series Forecasting

A univariate time series is a time-ordered set of observations $Y = \{y_1, y_2, \ldots, y_T\}$, where each value $y_t \in \mathbb{R}$ corresponds to the observed state at a specific time t, and T denotes the total length of the series.

Time series are characterized by temporal dependencies: **trend**, **seasonality**, and **remainder** (or residuals) (45). The *trend* reflects the long-term evolution of the series, capturing systematic increases or decreases over time. The *seasonal* component accounts for cyclical patterns that repeat at regular intervals (e.g., daily, weekly, yearly). Finally, the *remainder* encompasses irregular fluctuations and noise that cannot be described by the trend or seasonality. The models understanding of these fundamental properties is essential to compute meaningful forecasts.

Forecasting refers to predicting future values of a time series, $y_{T+1}, \ldots, y_{T+h}$, where h is the fore-

casting horizon. Many forecasting problems often extend beyond a single univariate series, and involve collections of time series $\mathcal{Y} = \{Y^{(1)}, Y^{(2)}, \ldots, Y^{(N)}\}$ (20), where N is the number of time series in the collection. In this project we only work with univariate time series data.

Most traditional forecasting approaches train separate forecasting models for each series in the collection. This makes the capture of shared patterns and dependencies that may exist across the time series impossible. Global forecasting models (3) overcome this limitation by using the data from all available time series to build a forecasting model. Machine learning approaches to time series forecasting typically adopt an auto-regressive modeling framework (3). This approach models future values as a function of the l preceding observations (lags).

2.1.2 Time series Forecasting models

Statistical Models

Statistical models have long been foundational tools in time series forecasting. These classical methods are designed to capture linear relationships, short-term dependencies, and periodic behavior in stationary or moderately evolving time series. In this section, we review the most widely used statistical forecasting models and discuss their strengths and inherent limitations.

The Autoregressive (AR) model (13) is a well-established technique in time series forecasting. It operates on the principle that the value of a time series at a particular time, t, can be predicted using a weighted combination of its p previous values, where p represents the model's order. Mathematically, it is defined:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \epsilon_t,$$

where ϕ_i are the coefficients, and ϵ_t represents random noise. The model (13) works by estimating these coefficients to minimize the difference between observed and predicted values. When working with this method, it is assumed that the time series is stationary; otherwise, transformations like differencing may be applied. By capturing temporal dependencies, the model provides a simple and effective approach for short-term forecasting.

The Moving Average (MA) model (18) is another fundamental approach in time series analysis that models the dependency of a time series on past error terms. Unlike AR, which uses previous observations, MA assumes that the current value of the time series is a linear combination of past forecast errors. An MA(q) model of order q is mathematically expressed as:

$$X_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q},$$

where μ is the mean of the series, ϵ_t represents white noise, and θ_i are the model coefficients. This approach captures short-term dependencies and helps smooth out fluctuations in the data. It is particularly effective for identifying patterns in the residual noise of a time series. By focusing on the impact of past errors, MA is useful for identifying and correcting short-term irregularities in time series data, making it a complementary method to AR models.

Meanwhile, the Autoregressive Moving Average (ARMA) model (11) combines the strengths of the Autoregressive (AR) (13) and Moving Average (MA) (18) models to provide a versatile approach for modeling stationary time series data. ARMA(p, q) incorporates both the dependency on past observations (AR) and the impact of past error terms (MA). Mathematically, it is expressed as:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + \cdots + \phi_p X_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \cdots + \theta_q \epsilon_{t-q},$$

where ϕ_i are the autoregressive coefficients, θ_i are the moving average coefficients, and ϵ_t is the white noise. The AR component captures the linear relationships between the series' current and past values, while the MA component accounts for the influence of past forecast errors. Together, these components allow ARMA (11) to capture both short-term dependencies and residual noise, making it a powerful tool for time series forecasting and analysis.

ARIMA (6) extends the ARMA (11) model to handle non-stationary time series, making it one of the most widely used tools for time series forecasting. While the ARMA model is suited for stationary data, ARIMA introduces an additional component called integration. The model is characterized by three parameters: p, the order of the autoregressive part; d, the degree of differencing required to achieve stationarity; and q, the order of the moving average part. Mathematically, it is represented as:

$$\Delta^d X_t = \phi_1 \Delta^d X_{t-1} + \cdots + \phi_p \Delta^d X_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \cdots + \theta_q \epsilon_{t-q},$$

where $\Delta^d X_t$ denotes the d-th differenced series. Differencing is applied to remove trends or other non-stationary elements from the data, ensuring that the underlying series satisfies stationarity requirements. The model combines the strengths of autoregression, moving averages, and differencing to handle a wide variety of time series data, from purely stationary to those with complex trends and seasonality. Its versatility makes ARIMA (6) a reliable approach for time series forecasting.

Classic models are well suited for applications in inventory management (8), demand forecasting (43), finance (40), and short-term predictions (32). However, despite their historical importance and effectiveness in many structured domains, they present several significant limitations (38; 21).

Statistical models rely on strong statistical assumptions, such as linearity and stationarity, which rarely hold in real-world time series, especially those involving complex systems. These methods capacity to

capture long-term dependencies, abrupt structural changes, and non-linear patterns is limited. As a result, this leads to poor generalization and ineffective model training. These limitations motivate more flexible and data-driven forecasting models that can learn complex patterns from the data. This leads to the study of deep learning approaches, which are discussed in the following section.(34) (21)

Deep learning models

While statistical methods are the core of classic forecasting and remain foundational tools, new techniques (4), emerged to address some of their limitations. With the increased production of real-world time series data and advances in neural architectures, deep learning solutions (4) have become increasingly used for time series forecasting.

Several types of deep learning architectures have been developed, ranging from feedforward neural networks and recurrent-based architectures (e.g., LSTMs, GRUs) designed to capture temporal dependencies, to more recent Transformer-based models (26) which leverage attention mechanisms. Despite the development of these diverse architectures, models based on multi-layer perceptrons (MLPs) have consistently shown competitive forecasting performance (10). MLPs are neural network architectures composed of stacks of fully connected layers.

One such state-of-the-art architecture based on MLPs is NHITS (10). NHITS is built upon stacked blocks of MLP layers connected with residual links. Each block learns hierarchical temporal representations by processing inputs sampled at multiple rates. Formally, the hierarchical layer of NHITS can be expressed as

$$\mathbf{H}_t = f_{\text{hier}}(\mathbf{Y}_t, \mathbf{Y}_{t-1}, \ldots, \mathbf{Y}_{t-n}),$$

where f_{hier} represents the neural network function capturing hierarchical relationships across different time scales, and n is the number of previous time steps considered. To combine information across different granularities, NHITS employs a hierarchical interpolation layer defined as

$$\mathbf{I}_t = f_{\text{interp}}(\mathbf{H}_t, \mathbf{Y}_{t-n}, \mathbf{Y}_{t-n+1}, \ldots),$$

which refines the forecast by integrating these learned hierarchical patterns. Through the combination of multi-rate input sampling and hierarchical interpolation, NHITS achieves state-of-the-art forecasting performance while maintaining computational efficiency.

Time series forecasting using Kolmogorov-Arnold Networks (KANs) (27) recently emerged as a competitive alternative to MLPs (12). KANs are inspired by the Kolmogorov-Arnold representation theorem, which states that any multivariate continuous function can be represented as a superposition of continuous

univariate functions. These networks learn interpretable transformations by applying learnable activation functions directly on the connections between neurons, rather than only at the nodes as in traditional MLPs. This design allows KANs to capture complex non-linear relationships in the data more efficiently. The architecture has shown promising forecasting performance (27).

In conclusion, recent deep learning models offer greater flexibility in modeling complex, non-linear, and non-stationary time series, where traditional statistical models often struggle (38). Architectures such as MLP-based models (e.g., NHITS) and KANs effectively learn from multi-scale patterns and achieve state-of-the-art forecasting accuracy across diverse domains. (4) As the field continues to evolve, further research in model efficiency, data efficiency, and generalization will be critical to unlocking the full potential of deep learning for time series forecasting (25).

However, these models have critical limitations (31). A key challenge is their high data dependency, deep learning networks require very large quantities of data for robust model training and diverse datasets to avoid overfitting. In many practical forecasting settings, time series data is limited, noisy, or imbalanced, making it difficult for deep learning models to generalize well (3).

2.2 Data Augmentation

Machine learning algorithms, particularly neural networks which are known to excel in data-rich environments, generally achieve better performance with larger training sets (9). The need for an adequate sample size is especially important when using deep learning models. These models typically contain a large number of parameters and rely on learning feature representations directly from raw data. To capture meaningful patterns and avoid overfitting, deep learning models must be exposed to a wide variety of examples. Without sufficient data, they tend to memorize the training set rather than generalize to new inputs.

Data augmentation addresses these challenges by generating realistic variations of the original data, increasing diversity, and enabling models to generalize better to new, unseen conditions (3; 24). The augmentation process works by generating artificial but realistic time series that mimic the characteristics of the original data. Then, the synthetic data is added to the original dataset to increase the size of the training set. Data augmentation has become an increasingly important tool in addressing the limitations of forecasting models in time series forecasting.

2.2.1 Data Augmentation Techniques

Despite significant advances, data generation techniques for time series forecasting still face critical limitations (44). A broad range of approaches have been developed, each with varying degrees of complexity, effectiveness, and domain specificity. These can be broadly categorized into transformation-based techniques, pattern mixing methods, and more recent generative approaches.

Transformation-based methods

Transformation-based methods simple and not expensive computationally (41) (46). They apply predefined alterations to existing time series data to simulate variability while preserving the overall structure.

- `Jittering` involves adding small random noise to each data point. This technique improves robustness by exposing the model to slight perturbations in the signal, helping it generalize better in noisy environments (42). However, because the added noise is often unstructured, this method may not introduce meaningful diversity and can risk degrading the original temporal patterns if overused.

- `Scaling` modifies the amplitude of the time series by multiplying its values by a random factor, typically drawn from a standard normal distribution. This method simulates changes in intensity, making it useful in domains where signal strength varies, such as wearable sensor data. The downside is that excessive scaling can distort the semantic meaning of the time series and compromise interpretability.

- `Time Warping (T-Warp)` introduces smooth distortions along the time axis, typically using interpolation through cubic splines. The warped timeline is then used to resample the original series. This allows the model to learn invariance to time shifts and varying speeds of patterns. However, care must be taken to avoid unrealistic temporal deformations that misalign important features or events in the sequence (36).

- `Magnitude Warping (M-Warp)` applies smooth, spline-based distortions directly to the values of the series rather than its timestamps. This results in more gradual, shape-preserving modifications than `Scaling`.(36)

These transformation techniques are domain-agnostic and have been adapted for use in fields such as healthcare and activity recognition (19). Nonetheless, their major limitation is that they do not model the

underlying data generation process. As a result, they may fail to introduce sufficient variability or preserve contextual dependencies critical for accurate forecasting (44).

Pattern mixing methods

Pattern mixing methods aim to generate synthetic time series by combining segments or features from different original series, offering greater diversity in the augmented data.

- `Moving Block Bootstrap (MBB)` constructs synthetic series by sampling blocks of consecutive observations from the original data, preserving local temporal structure. When combined with seasonal decomposition, it allows for more control over trend and seasonality components. However, MBB relies on the assumption of local stationarity and may replicate redundant segments, limiting the diversity of synthetic samples. (5)

- `DTW Barycentric Averaging (DBA)` computes the average of multiple series using Dynamic Time Warping alignment. This technique produces smooth prototypes that preserve key dynamic behaviors across input sequences . While DBA generates more coherent synthetic sequences than simple averaging, it can be sensitive to outliers and is computationally intensive for large datasets. (16)

- `TSMixup` creates new time series by mixing randomly sampled segments from different sequences using weighted averages. This method introduces greater diversity and can be adapted to supervised learning settings by interpolating both input and target sequences. However, the synthetic sequences produced may lack interpretability, and mixing unrelated patterns can yield unrealistic examples that negatively impact learning. (1)

Generative models

Generative models such as GANs and VAEs offer a more principled approach to data augmentation by learning to approximate the data distribution directly. These methods can, in theory, produce highly diverse and realistic synthetic sequences. However, their practical application remains challenging. They require significant computational resources, are sensitive to hyperparameter tuning, and often struggle to generate temporally consistent or semantically meaningful data in forecasting contexts (44). Moreover, mode collapse and training instability may remain persistent issues.

In summary, while existing augmentation strategies offer important tools for improving model generalization, each comes with trade-offs. Their effectiveness often depends heavily on the specific character-

istics of the dataset and the forecasting task at hand. The limitations of these techniques highlight the need for data augmentation strategies that effectively increase diversity and quantity of training data, while maintaining the temporal dependencies critical for model training.

2.3 Time Series Analysis using Network Science

The application of network science to time series analysis has proven to be an effective approach that bridges two distinct domains: temporal data analysis and graph theory (7). By transforming temporal data into network representations, this approach allows the use of graph theory to reveal complex patterns, nonlinear relationships, or multi-scale dynamics that conventional time and frequency-domain analyses may fail to detect. Furthermore, it offers more intuitive visualizations of dynamic behavior, which can reveal hidden structures in temporal data and improve interpretability (39).

Using network science for time series analysis involves transforming the temporal data into a graph representation. Several approaches have been proposed for this conversion. Quantile graphs (7) discretize the time series into bins or quantiles and represent transitions between these states as edges in a graph, capturing the dynamics of state changes. Visibility graphs (39) connect time series observations that can *see* each other based on a geometric visibility criterion, converting the temporal sequence into a complex network where structural properties correspond to time series characteristics.

Other common graph-based transformations include recurrence plots, which map the recurrence of states in the time series to a graph structure, enabling the study of periodicity and regime shifts. Horizontal visibility graphs, are also being used due to their ability to preserve temporal ordering while filtering out small-scale fluctuations (28). Transition graphs, which model the probabilities or frequencies of transitions between discrete states of the time series, are also widely used for capturing dynamic behavior (39). Silva et al. (39) overview several other methods for time series analysis using network science.

The concept of duality between time series and graphs, as introduced by (7), formalizes the relationship between temporal data and their corresponding network representations. This framework offers an alternative perspective for analyzing complex temporal characteristics inherent in time series data.

Figure 2, adapted from (7), illustrates the conversion process from a time series to its graph-based probability matrix representation. This transformation highlights the method proposed by (7) to map temporal sequences into a graph structure, enabling novel analytical approaches.

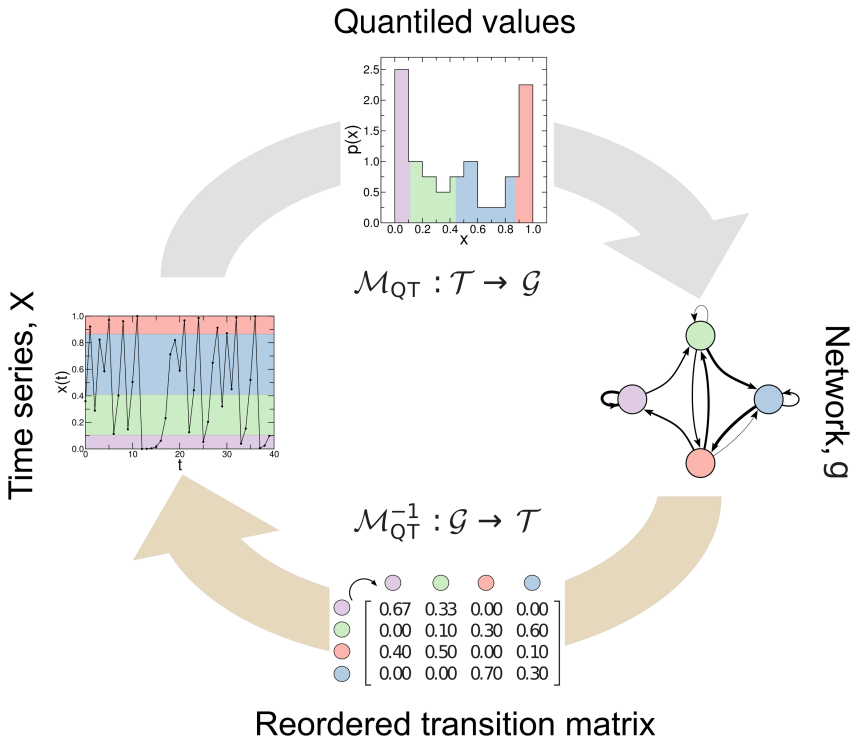


Figure 2: Illustration of the time-series to graph conversion scheme from (7), depicting the transformation of temporal data into a graph probability matrix representation.

2.4 Motivation

In this chapter, we addressed state-of-the-art deep learning architectures for time series forecasting, which have demonstrated strong performance in capturing complex temporal dynamics. However, these models are highly dependent on large and diverse training datasets. In real-world scenarios, such data is often scarce or unavailable, making robust model training a significant challenge.

To mitigate this issue, data augmentation techniques generate synthetic time series to enhance the training dataset. Despite recent advances, current data augmentation methods still face important limitations. These include limited control over variability, the introduction of unwanted noise, poor preservation of essential temporal dependencies, and high computational costs.

Therefore, developing augmentation methods capable of producing realistic and diverse time series while preserving temporal coherence remains an essential task.

To address these limitations, we introduce `Grasynda`, a graph-based approach designed for time series data augmentation. In the next chapter, we detail the methodology and formalization of Grasynda.

Part II

Core of the Dissertation

Chapter 3

Methodology

This section describes the methodology behind `Grasynda`. We start by detailing the graph construction process based on quantile discretization (Section 3.1). Section 3.2 shows how to leverage the transition matrix to generate new time series. Then, we present a variant of `Grasynda` that combines several transition matrices for more robust time series generation (Section 3.3). Finally, we show how to use `Grasynda` to handle non-stationary time series (Section 3.4).

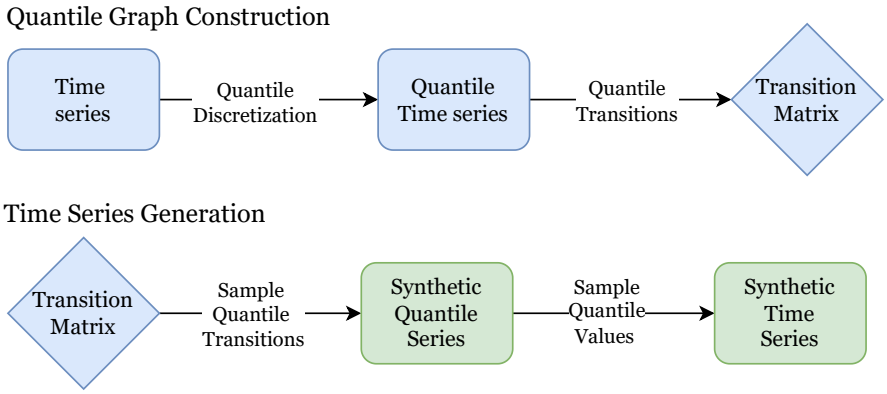


Figure 3: Workflow behind `Grasynda` for building a quantile graph from a time series and using it to create synthetic time series.

The workflow behind `Grasynda`, illustrated in Figure 3, is based on two main steps: transforming time series into quantile graphs; and generating time series using this representation.

3.1 Time Series Quantile Graph

3.1.1 Quantile Discretization

The process begins by performing quantile discretization on the original time series $Y = \{y_1, y_2, \ldots, y_T\}$ (15). The series is divided into k quantiles based on equal-frequency binning. So, each quantile bin is

defined to contain an approximately equal number of observations (T/k).

Each value $y_t \in Y$ is mapped to a discrete label $q_t \in \{1, 2, \ldots, k\}$ corresponding to the index of the quantile bin it belongs to. This results in a new discretized time series Q:

$$Q = \{q_1, q_2, \ldots, q_T\}, \quad q_t \in \{1, 2, \ldots, k\} \tag{3.1}$$

where q_t represents the quantile bin associated with the value y_t. Let S_j denote the set of original values assigned to quantile bin q_j:

$$S_j = \{y_t \in Y \mid q_t = j\}, \quad j \in \{1, \ldots, k\} \tag{3.2}$$

3.1.2 Graph Construction

After discretization, we create the graph structure based on the resulting quantiles. This structure results in a directed weighted graph $G = (V, E, P)$, where:

- $V = \{1, 2, \ldots, k\}$ is the set of nodes, each representing a quantile bin q_i,
- $E \subseteq V \times V$ is the set of directed edges, where an edge $(q_i, q_j) \in E$ exists if there is at least one transition from q_i to q_j in the discretized time series Q,
- $P = [p_{ij}]$ is the normalized edge weight matrix (transition matrix), where each weight p_{ij} corresponds to the empirical conditional probability of transitioning from quantile q_i to quantile q_j:

The elements of P satisfy:

$$p_{ij} = P(q_{t+1} = q_j \mid q_t = q_i) = \frac{\text{count of transitions from } q_i \text{ to } q_j}{\text{total transitions from } q_i} \tag{3.3}$$

Since these are conditional probabilities, each row of P sums to 1: $\sum_{j=1}^{k} p_{ij} = 1$ for all $i \in \{1, \ldots, k\}$. Generally $p_{ij} \neq p_{ji}$, indicating the directed nature of the graph and the non-symmetric property of the transition matrix.

3.2 Time series generation

3.2.1 Quantile Series Generation

By sampling from the transition matrix P, we can generate synthetic quantile sequences $\hat{Q} = \{\hat{q}_1, \hat{q}_2, \ldots, \hat{q}_T\}$. We start by selecting an initial quantile $\hat{q}_1$, matching the first quantile from the original time series. The

subsequent quantiles are sampled iteratively based on the transition probabilities:

$$\hat{q}_{t+1} \sim P(\hat{q}_t) \tag{3.4}$$

This means that $\hat{q}_{t+1}$ is chosen according to the probability distribution given by the row of P corresponding to $\hat{q}_t$. In effect, the synthetic quantile sequence follows the same probabilistic transitions as the original quantile series. We can sample any number of quantiles to generate synthetic sequences of arbitrary length. Notwithstanding, in the experiments presented in the next section, we sample a number of quantiles equal to the size of the original time series.

3.2.2 Time Series Generation

After obtaining a synthetic quantile time series $\hat{Q}$, we convert it back to the original time series space. This is achieved by sampling from the original time series. Specifically, for each quantile $\hat{q}_i$ in $\hat{Q}$, a corresponding value $\hat{y}_i$ is randomly sampled from the original values assigned to that quantile bin:

$$\hat{y}_t \sim \text{Uniform}(S_{\hat{q}_i}) \tag{3.5}$$

This results in a synthetic time series $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T\}$. This sampling approach preserves the original temporal dynamics of the time series while introducing controlled variability by leveraging the probabilistic nature of the quantile state transitions.

3.3 `Grasynda(E)`: transition matrix ensembles

Some time series are composed of a small number of observations that may be insufficient for computing a transition matrix that effectively captures its underlying dynamics. In this context, we introduce an ensemble-based variant of the proposed method, `Grasynda(E)`.:

For a given target time series, this variant operates by averaging the N nearest transition matrices from the dataset. Let $\{P^{(1)}, P^{(2)}, \ldots, P^{(M)}\}$ denote the set of transition matrices derived from all M time series in the dataset, where each $P^{(i)} \in \mathbb{R}^{k \times k}$ represents the quantile graph transition matrix of the i-th time series. We identify the most similar matrices using the Euclidean norm:

$$d_{ab} = \|P^{(a)} - P^{(b)}\| = \sqrt{\sum_{m=1}^{k}\sum_{n=1}^{k}\left(p_{mn}^{(a)} - p_{mn}^{(b)}\right)^2}. \tag{3.6}$$

From the dataset at hand, the N matrices in $\{P^{(1)}, P^{(2)}, \ldots, P^{(M)}\}$ with the smallest distances d_{ab} are selected and averaged element-wise to produce the final aggregated transition matrix:

$$\bar{P}^{(a)} = \frac{1}{N} \sum_{b \in \mathcal{N}_a} P^{(b)}, \tag{3.7}$$

where $\mathcal{N}_a$ represents the set of indices corresponding to the N most similar matrices. After averaging the selected transition matrices, the generation process is the same as described in Section 3.2. Besides mitigating small sample size issues, blending information from multiple transition matrices promotes greater diversity in the generated series while, in principle, maintaining realistic temporal dynamics.

3.4 Handling non-stationarity

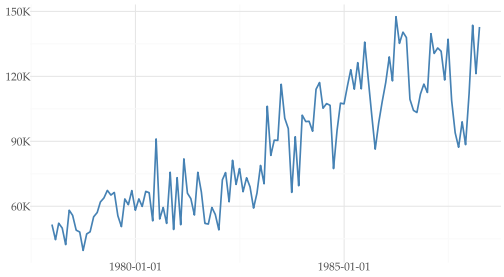
The methodology described above effectively models temporal dynamics through quantile state transitions between consecutive observations. While the network structure captures local patterns through edge connections, it has limitations in preserving long-term temporal dependencies such as trend and seasonality that characterize non-stationary time series.

To address these limitations, we adopt a decomposition-based approach following Bergmeir et al. (5). Our solution leverages Seasonal-Trend decomposition using LOESS (STL) (14) to decompose each time series into three distinct components:

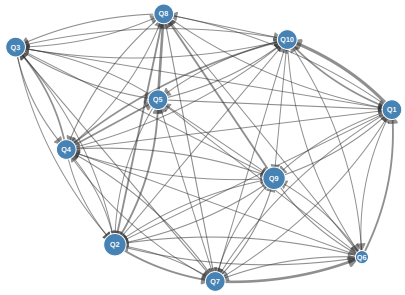
$$y_t = \text{Trend}_t + \text{Seasonal}_t + \text{Remainder}_t \tag{3.8}$$

We then apply `Grasynda` to the remainder component, which typically exhibits stationarity. After obtaining a synthetic remainder component $Remainder_{\texttt{Grasynda}}$ is generated, we recombine it with the original trend and seasonal components to produce the final synthetic time series:

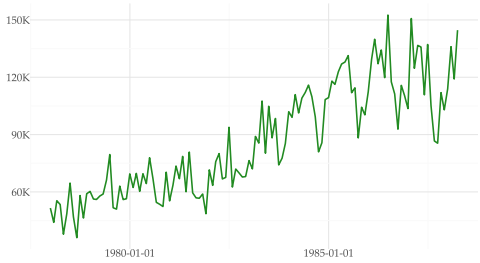
$$\hat{y}_t = \text{Trend}_t + \text{Seasonal}_t + \text{Remainder}_{\texttt{Grasynda}} \tag{3.9}$$



(a) Original time series in the time domain



(b) Time series quantile graph representation



(c) Synthetic time series generated with `Grasynda`

Figure 4: Example plot of a time series in the time domain (a); the corresponding network representation using a quantile graph (b); and a synthetic time series generated from the graph using `Grasynda` (c).

Figure 4 shows an example application of `Grasynda` using the time series with identifier *ID90* from the M3 Monthly dataset (c.f. Section 4.1.1). The figure shows the time series in the standard time domain (a), the corresponding network representation using a quantile graph (b), and a synthetic time series generated from the graph using `Grasynda` (c).

Chapter 4

Experiments

We conduct extensive experiments to evaluate the effectiveness of `Grasynda` as a data augmentation method for time series forecasting, comparing its performance against other state-of-the-art approaches for synthetic time series generation.

4.1 Experimental Design

4.1.1 Datasets

We use six public benchmark datasets used in previous forecasting competitions: M1 (29), M3 (30), and Tourism (2). These cover different application domains, including industry, demography, and economics. We present a summary of the datasets in Table 1. Overall, these comprise a total of 3797 time series, with 409602 observations.

Table 1: Summary of the datasets: average value, number of time series, number of observations, seasonal period, and forecasting horizon (h).

Dataset	Average value	# time series	# observations	period	h
M1 Monthly	72.7	617	44892	12	12
M1 Quarterly	40.9	203	8320	4	8
M3 Monthly	117.3	1428	167562	12	12
M3 Quarterly	48.9	756	37004	4	8
Tourism Monthly	298.5	366	109280	12	12
Tourism Quarterly	99.6	427	42544	4	8
Total	-	3797	409602	-	-

Each dataset is used in its original form without additional pre processing, aside from standard train-test splitting, to maintain consistency with prior work.

4.1.2 Parameter setup

We set the forecasting horizon (h) to 12 and 8, for monthly and quarterly time series, respectively. The input size (number of lags) is set to 24 for monthly time series and 8 for quarterly ones. These values correspond to two seasonal periods.`Grasynda` and `Grasynda(E)` are applied with 25 quantiles and an ensemble size of 50. A sensitivity analysis is shown in Figure 5.

4.1.3 Performance estimation

To evaluate performance, the final h observations of each time series are held out for testing. The model is trained using the preceding observations.

The evaluation metric used is the **Mean Absolute Scaled Error (MASE)** (22), a scale-independent measure of forecasting accuracy. MASE is calculated by dividing the Mean Absolute Error (MAE) of the forecast by the MAE of a seasonal naive baseline computed on the training data.

Let the original training dataset be

$$\mathcal{D} = \left\{ Y^{(1)}, Y^{(2)}, \ldots, Y^{(M)} \right\}, \tag{4.1}$$

where each time series

$$Y^{(i)} = \left\{ y_1^{(i)}, y_2^{(i)}, \ldots, y_T^{(i)} \right\} \tag{4.2}$$

has length T.

For each $Y^{(i)}$, a synthetic series

$$\hat{Y}^{(i)} = \left\{ \hat{y}_1^{(i)}, \hat{y}_2^{(i)}, \ldots, \hat{y}_T^{(i)} \right\} \tag{4.3}$$

is generated.

The augmented training dataset is then

$$\mathcal{D}_{\text{aug}} = \left\{ Y^{(1)}, \ldots, Y^{(M)}, \hat{Y}^{(1)}, \ldots, \hat{Y}^{(M)} \right\}. \tag{4.4}$$

4.2 Methods

4.2.1 Neural Networks and Baseline

To evaluate the effectiveness of `Grasynda`, we use three state-of-the-art neural network forecasting models: NHITS (10), KAN (27), and MLP (12). These models were selected for their established competitive forecasting accuracy (c.f. Section 2) and computational efficiency when compared to other deep learning architectures (47). All neural network models are used with a fixed configuration across all datasets. Specifically, we use the default hyperparameters and settings from the neuralforecast Python library. This ensures a consistent and fair comparison across models and datasets.

The seasonal naive method is included as a forecasting baseline to provide a benchmark for the experimental results.

4.2.2 Synthetic time series generators

The proposed method is compared against several established data augmentation techniques: `Scaling` (46), `TSMixup` (1), `DBA` (17), jittering (46) (`Jitter`), magnitude warping (23) (`M-Warp`), time warping (35) (`T-Warp`), and seasonal MBB (37) (`MBB`). The data augmenation methods are used with the default configuration from metaforecast Python library. This methods are reviewed in detail in Section 2. We also include the original training data with no augmentation (`Original`).

Chapter 5

Results

This chapter presents the experimental results of the proposed `Grasynda` and `Grasynda(E)` data augmentation methods across a diverse set of time series forecasting tasks. We begin by analyzing the forecasting performance of each augmentation method across multiple datasets and neural forecasting architectures, with emphasis on MASE (22) and statistical significance. An effectiveness analysis is employed to quantify how frequently each method outperforms the non-augmented baseline. Subsequently, we examine the sensitivity of `Grasynda(E)` to the number of quantiles and the ensemble parameter. Collectively, these results are used to assess the performance of the proposed approaches.

5.1 Main results

The results of the experiments are presented in Table 2. These indicate that both `Grasynda` and `Grasynda(E)` consistently improve forecasting accuracy relative to `Original`. Specifically, across the 18 evaluated experiments (6 datasets times 3 forecasting models), `Grasynda` and `Grasynda(E)` shows an improvement in 13 cases (72%). This level of effectiveness is greater than the one obtained by the other augmentation methods. The effectiveness score denotes the ratio of times the respective method outperforms the Original reference approach.

Table 2: MASE of each approach across different neural networks and datasets, and corresponding model average and average rank. Best and second-best values are **bolded** and underlined in the respective dataset and neural network architecture. The '*' symbol denotes a significantly better performance of the respective method relative to `Original`.

		Original	Grasynda	Grasynda(E)	DBA	Jitter	M-Warp	MBB	Scaling	T-Warp	TSMixup	SNaive
NHITS	M1-M	0.977	0.9449	**0.9415***	0.9476*	0.9697	0.9807	0.952	0.9477*	0.9772	0.9779	1.2207
	M1-Q	1.0371	1.0341	1.0208	1.0935	1.0359	**0.9738***	1.0289	1.0389	1.0655	1.0487	1.647
	M3-M	0.8014	0.759*	0.7641*	0.7787	0.7855	0.7733	0.7642*	0.7702	**0.7569***	0.7703	1.091
	M3-Q	1.1994	1.2196	1.2045	1.1363*	1.1851	1.1806	1.1651*	**1.1069***	1.1305*	1.1933	1.4168
	T-M	1.208	1.1904*	1.1881*	1.1933	1.2004*	1.2083	**1.184***	1.2056	1.2298	1.1904*	1.3448
	T-Q	1.6348	1.5549*	**1.528***	1.5647*	1.608*	1.6282	1.6634	1.6034*	1.6161	1.6215	1.7016
	Avg.	1.143	1.1172	**1.1078**	1.119	1.1307	1.1242	1.1263	1.1121	1.1293	1.1337	1.4036
	Avg. Rank	8.0	3.83	**3.0**	5.33	6.17	6.67	4.5	4.67	6.0	6.83	11.0
MLP	M1-M	0.9298	0.9482	0.9518	0.9847	0.9395	0.9685	0.9364	**0.9264**	0.9774	0.9535	1.2207
	M1-Q	1.0704	1.0745	1.0812	1.0641	1.0336*	**0.9957***	1.0361*	1.0403	1.0763	1.062	1.647
	M3-M	0.7735	0.7681	0.765	0.7774	0.7597*	0.7796	0.7654*	0.7667	**0.7573**	0.7612	1.091
	M3-Q	1.1428	1.1809	1.1989	1.1518	1.1481	1.1592	1.1543	**1.1115**	1.1324	1.2094	1.4168
	T-M	1.1981	1.1947	1.1884	2.5449	1.2126	1.2164	1.1963*	1.204	1.2574	**1.183***	1.3448
	T-Q	1.5676	1.5166*	**1.4976***	1.6052	1.6539	1.6154	1.6369	1.6453	1.7196	1.5976	1.7016
	Avg.	**1.1137**	1.1138	1.1138	1.3547	1.1246	1.1224	1.1209	1.1157	1.1534	1.1278	1.4036
	Avg. Rank	4.67	5.5	5.33	7.67	4.67	6.67	4.67	**4.33**	6.83	5.0	10.67
KAN	M1-M	0.9614	0.9411	0.9392	0.9573	0.9618	0.9788	0.9393	**0.9364**	1.0076	0.9552	1.2207
	M1-Q	1.0128	1.0216	1.0244	1.0431	1.013	**0.9661***	1.0158	1.0297	1.0708	1.0438	1.647
	M3-M	0.7844	0.7791	0.775	0.7766	0.7828	0.7955	**0.7725***	0.7971	0.7984	0.7754*	1.091
	M3-Q	1.2225	1.2069*	1.2124	**1.1463***	1.2292	1.213	1.2212	1.1651*	1.2063	1.2711	1.4168
	T-M	1.2273	1.1904*	**1.1827***	1.2306	1.2276	1.2294	1.2171	1.2216	1.2725	1.2133*	1.3448
	T-Q	1.5709	1.5476	**1.545***	1.5488	1.6312	1.5908	1.6002	1.6234	1.6701	1.6419	1.7016
	Avg.	1.1299	1.1145	**1.1131**	1.1171	1.1409	1.1289	1.1277	1.1289	1.1709	1.1501	1.4036
	Avg. Rank	5.67	3.67	**2.83**	5.17	6.83	6.17	4.17	5.17	8.83	6.5	11.0
All	Effectiveness	–	**0.72**	**0.72**	0.56	0.5	0.39	0.67	0.67	0.33	0.56	0.0

`Grasynda(E)` shows the best average MASE in two of the three tested forecasting models, and the second best in the other. The average rank score (grouped by forecasting model) also shows that `Grasynda(E)` outperforms the other tested data augmentation methods.

`Grasynda(E)` generally shows improved or comparable performance to `Grasynda`, particularly in average MASE and rank across models, suggesting the ensembling step is often beneficial, though not always superior in every specific instance.

In most scenarios, the neural networks outperform the seasonal naive baseline. This outcome validates the forecasting performance of the trained models. We also assess the significance of the results using the Wilcoxon signed-rank test. In Table 2, the '*' symbol denotes a significant better performance of the respective method relative to `Original`. In 6 (7) out of the 18 tests, `Grasynda` (`Grasynda(E)`) shows a significant better performance than Original. Except for `MBB` (also 7 times), other approaches, in most cases, do not show significant better performance than `Original`.

5.2 Sensitivity Analysis

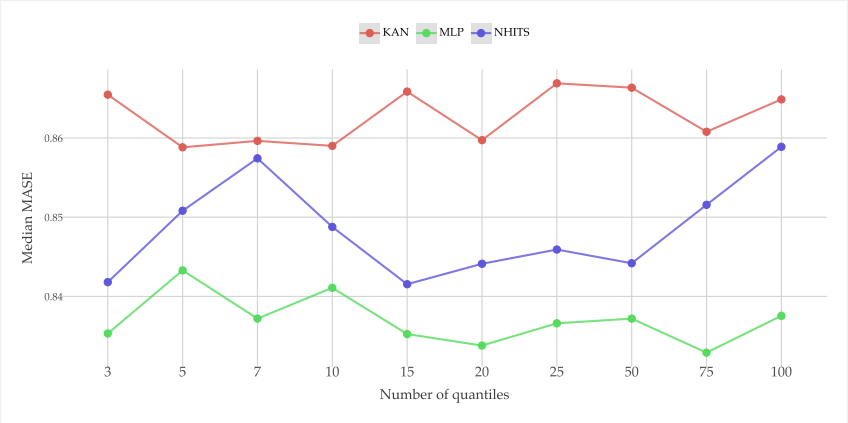


Figure 5: Median MASE values for different quantile parameters across different forecasting models.

We conducted an analysis to determine how the choice of quantile parameter influences `Grasynda(E)`'s performance. The results are shown in Figure 5, illustrating the median[1] MASE values for different quantile parameters across different forecasting models. The results vary by forecasting model, but in general, the performance is poorer and more inconsistent when the number of quantiles is either too low (<10) or too high (>50). Apart from that, the performance is somewhat consistent over the tested range of quantiles.

We also conducted a sensitivity analysis on the ensemble size parameter, with the results show in Figure 6. The conclusions are similar to the quantile parameter analysis. In particular, both very low and very high values for the ensemble size should be avoided, as they tend to result in poorer performance.

[1] The median is used as a more robust measure of central tendency compared to the mean.

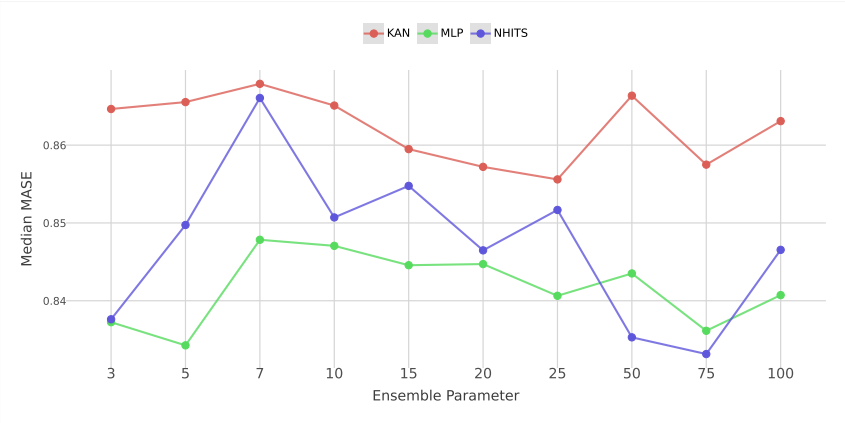


Figure 6: Median MASE values for different ensemble parameters across forecasting models.

5.3 Discussion

This section discusses the main findings of our research in relation to the questions posed in the introduction, highlighting the contributions and implications of `Grasynda`.

The first research question (**RQ1**) addresses the development of a novel data augmentation method based on a graph-to-time-series conversion. The primary contribution of this work is the formalization of such a novel method, `Grasynda`. By converting time series into graph representations and using transition probability matrices, we have introduced an approach to generating synthetic time series.

Regarding **RQ2**, which assesses the effectiveness of `Grasynda` and `Grasynda(E)`, and **RQ3**, which compares them to existing data augmentation methods, our extensive experiments show their ability to improve forecasting accuracy. The results (Table 2) show that `Grasynda` and `Grasynda(E)` improves forecasting accuracy in 72% of scenarios compared to not using data augmentation (`Original`). Furthermore, in comparison to state-of-the-art data augmentation techniques and a seasonal naive baseline, both `Grasynda` and `Grasynda(E)` show competitive forecasting accuracy, and outperform these benchmarks. Notably, `Grasynda(E)` consistently ranked among the top two performing methods across all three forecasting architectures and achieved the best average rank in two of these models.

Regarding the sensitivity of `Grasynda(E)` to its parameters, such as the number of quantiles and ensemble size (an implicit fourth research question), the analysis presented in Section 5.2 (Figure 5) indicates that performance is generally robust within a certain range. However, extreme values (too low or too high) for these parameters can lead to poorer outcomes. This highlights the need for careful parameter selection or tuning, a common aspect in machine learning models.

In future work, the objective will be methodological improvements to `Grasynda`, with a focus on

optimizing parameter sensitivity—particularly regarding the number of quantiles and the ensemble size—to enable more refined model tuning. Additionally, testing alternative discretization approaches, such as visibility graphs (39), may lead to improved performance. Another objective is to leverage `Grasynda`'s probability matrix representation for scenario simulation and stress-testing of forecasting architectures. This could enable the evaluation of model robustness under varied conditions, including the presence of outliers and other real-world scenarios.

Chapter 6

Conclusion

This work introduced `Grasynda`, a novel synthetic time series generation method. By converting time series into graph representations and using transition probability matrices, `Grasynda` generates synthetic data.

Using extensive experiments, we show that `Grasynda`, and its ensemble-based variant `Grasynda(E)`, improve forecasting accuracy and outperform state-of-the-art time series data augmentation approaches. Both `Grasynda` and `Grasynda(E)` consistently outperformed established state-of-the-art data augmentation methods and baseline models, with `Grasynda(E)` frequently emerging among the top performing methods. Overall, we believe `Grasynda` offers a significant contribution to the time series literature by providing a novel and effective graph-based synthetic data generation strategy.

Future work will pursue several promising directions. These include methodological improvements to `Grasynda`, such as exploring alternative graph construction or discretization techniques (e.g., visibility graphs) (39). Furthermore, the probabilistic transition matrices generated by `Grasynda` open up opportunities for advanced applications like scenario simulation and stress-testing forecasting models.

Bibliography

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

[2] George Athanasopoulos, Rob J Hyndman, Haiyan Song, and Doris C Wu. The tourism forecasting competition. *International Journal of Forecasting*, 27(3):822–844, 2011.

[3] Kasun Bandara, Hansika Hewamalage, Yuan-Hao Liu, Yanfei Kang, and Christoph Bergmeir. Improving the accuracy of global forecasting models using time series data augmentation, 2020.

[4] Konstantinos Benidis, Syama Sundar Rangapuram, Valentin Flunkert, Yuyang Wang, Danielle Maddix, Caner Turkmen, Jan Gasthaus, Michael Bohlke-Schneider, David Salinas, Lorenzo Stella, François-Xavier Aubet, Laurent Callot, and Tim Januschowski. Deep learning for time series forecasting: Tutorial and literature survey. 2022.

[5] Christoph Bergmeir, Rob J Hyndman, and José M Benítez. Bagging exponential smoothing methods using stl decomposition and box–cox transformation. *International journal of forecasting*, 32(2):303–312, 2016.

[6] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[7] Andriana S. L. O. Campanharo, M. Irmak Sirer, R. Dean Malmgren, Fernando M. Ramos, and Luís A. Nunes. Amaral. Duality between time series and networks. *PLOS ONE*, 2011.

[8] Real Carbonneau, Kevin Laframboise, and Rustam Vahidov. Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, pages 1140–1154, 2008.

[9] Vitor Cerqueira, Luis Torgo, and Carlos Soares. A case study comparing machine learning with statistical methods for time series forecasting: size matters. *Journal of Intelligent Information Systems*, 59(2):415–433, 2022.

[10] Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. Nhits: Neural hierarchical interpolation for time series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[11] Jiann-Fuh Chen, Wei-Ming Wang, and Chao-Ming Huang. Analysis of an adaptive time-series autoregressive moving-average (arma) model for short-term load forecasting. *Electric Power Systems Research*, 1995.

[12] Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O. Arik, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecasting, 2023.

[13] Yongping Chen, Min Gan, Shunqi Pan, Haidong Pan, Xian Zhu, and Zhengjin Tao. Application of auto-regressive (ar) analysis to improve short-term prediction of water levels in the yangtze estuary. *Journal of Hydrology*, 2020.

[14] Robert B Cleveland, William S Cleveland, Jean E McRae, Irma Terpenning, et al. Stl: A seasonal-trend decomposition. *J. off. Stat*, 6(1):3–73, 1990.

[15] Elena S Dimitrova, M Paola Vera Licona, John McGee, and Reinhard Laubenbacher. Discretization of time series data. *Journal of Computational Biology*, 17(6):853–868, 2010.

[16] Germain Forestier, François Petitjean, Hoang Anh Dau, Geoffrey I Webb, and Eamonn Keogh. Generating synthetic time series to augment sparse datasets. In *2017 IEEE international conference on data mining (ICDM)*, pages 865–870. IEEE, 2017.

[17] Germain Forestier, François Petitjean, Hoang Anh Dau, Geoffrey I. Webb, and Eamonn Keogh. Generating synthetic time series to augment sparse datasets. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 865–870, 2017.

[18] Seng Hansun. A new approach of moving average method in time series analysis. In *2013 Conference on New Media Studies (CoNMedia)*, 2013.

[19] Zeshan Hussain, Francisco Gimenez, Darvin Yi, and Daniel Rubin. Differential data augmentation techniques for medical imaging classification tasks. In *AMIA annual symposium proceedings*, volume 2017, page 979. American Medical Informatics Association, 2017.

[20] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2014.

[21] Rob Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Australia, 3rd edition, 2021.

[22] Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 2006.

[23] Brian Kenji Iwana and Seiichi Uchida. An empirical survey of data augmentation for time series classification with neural networks. *PLOS ONE*, 16(7):1–32, 07 2021.

[24] Cherry Khosla and Baljit Singh Saini. Enhancing performance of deep learning models with different data augmentation techniques: A survey. In *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, 2020.

[25] Milind Kolambe and Sandhya Arora. Forecasting the future: A comprehensive review of time series prediction techniques. *Journal of Electrical Systems*, 20(2s):575–586, 2024.

[26] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

[27] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks, 2024.

[28] B. Luque, L. Lacasa, F. Ballesteros, and J. Luque. Horizontal visibility graphs: Exact results for random time series. *Phys. Rev. E*, 2009.

[29] Spyros Makridakis, Allan Andersen, Robert Carbone, Robert Fildes, Michele Hibon, Rudolf Lewandowski, Joseph Newton, Emanuel Parzen, and Robert Winkler. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of forecasting*, 1(2):111–153, 1982.

[30] Spyros Makridakis and Michele Hibon. The m3-competition: results, conclusions and implications. *International journal of forecasting*, 16(4):451–476, 2000.

[31] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 2018.

[32] Maurice Omane-Adjepong, Francis Oduro, and Samuel Oduro. Determining the better approach for short-term forecasting of ghana's inflation: Seasonal arima vs holt-winters. *International Journal of Business, Humanities and Technology*, 3:69–79, 01 2013.

[33] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. 2019.

[34] Raydonal Ospina, João A. M. Gondim, Víctor Leiva, and Cecilia Castro. An overview of forecast analysis with arima models during the covid-19 pandemic: Methodology and case study in brazil. *Mathematics*, 11, 2023.

[35] Khandakar M. Rashid and Joseph Louis. Time-warping: A time series data augmentation of imu data for construction equipment activity identification. In *Proceedings of the 36th International Symposium on Automation and Robotics in Construction (ISARC)*, pages 651–657, May 2019.

[36] Luis Roque, Carlos Soares, and Luís Torgo. Rhiots: A framework for evaluating hierarchical time series forecasting algorithms. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2491–2499, 2024.

[37] Xiaofeng Shao. The Dependent Wild Bootstrap. *Journal of the American Statistical Association*, pages 218–235, 2010.

[38] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401, 2018.

[39] Vanessa Freitas Silva, Maria Eduarda Silva, Pedro Ribeiro, and Fernando Silva. Time series analysis via network science: Concepts and algorithms. *WIREs Data Mining and Knowledge Discovery*, 11, March 2021.

[40] Granville Tunnicliffe Wilson. Time series analysis: Forecasting and control,5th edition, by george e. p. box, gwilym m. jenkins, gregory c. reinsel and greta m. ljung, 2015. published by john wiley and sons inc., hoboken, new jersey, pp. 712. isbn: 978-1-118-67502-1. *Journal of Time Series Analysis*, 37:n/a–n/a, 03 2016.

[41] Terry T. Um, Franz M. J. Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić. Data augmentation of wearable sensor data for parkinson's disease moni-

toring using convolutional neural networks. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, ICMI '17. ACM, 2017.

[42] Terry T Um, Franz MJ Pfister, Daniel Pichler, Satoshi Endo, Muriel Lang, Sandra Hirche, Urban Fietzek, and Dana Kulić. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM international conference on multimodal interaction*, pages 216–220, 2017.

[43] Claudimar Veiga, Cassia Rita Veiga, Anderson Catapan, Ubiratã Tortato, and Wesley Silva. Demand forecasting in food retail: A comparison between the holt-winters and arima models. *WSEAS Transactions on Business and Economics*, 2014.

[44] Alexander Okhuese Victor and Muhammad Intizar Ali. Enhancing time series data predictions: A survey of augmentation techniques and model performances. In *Proceedings of the 2024 Australasian Computer Science Week*, 2024.

[45] Qingsong Wen, Jingkun Gao, Xiaomin Song, Liang Sun, Huan Xu, and Shenghuo Zhu. Robuststl: A robust seasonal-trend decomposition algorithm for long time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5409–5416, 2019.

[46] Qingsong Wen, Liang Sun, Fan Yang, Xiaomin Song, Jingkun Gao, Xue Wang, and Huan Xu. Time series data augmentation for deep learning: A survey. 2020.

[47] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128, Jun. 2023.

Part III
Appendices

Appendix A
Details of results

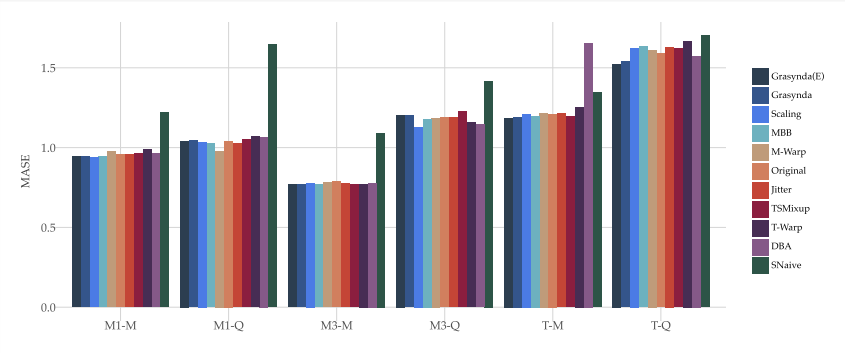


Figure 7: forecasting architetures mean MASE by dataset and data augmentation model

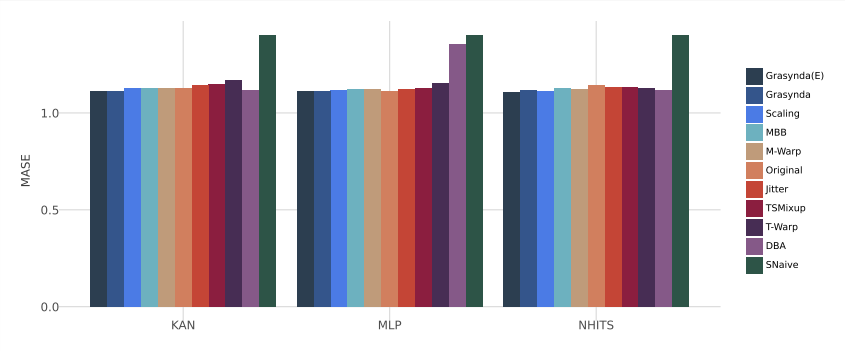


Figure 8: Average MASE across forecasting models, for different data augmentation methods.

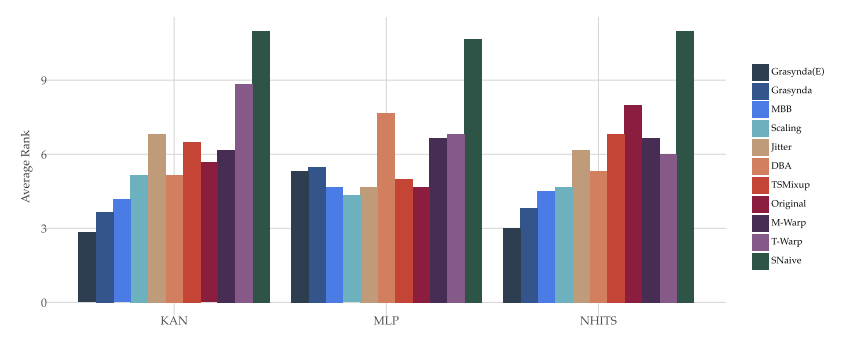


Figure 9: Average rank across forecasting models, for different data augmentation methods.

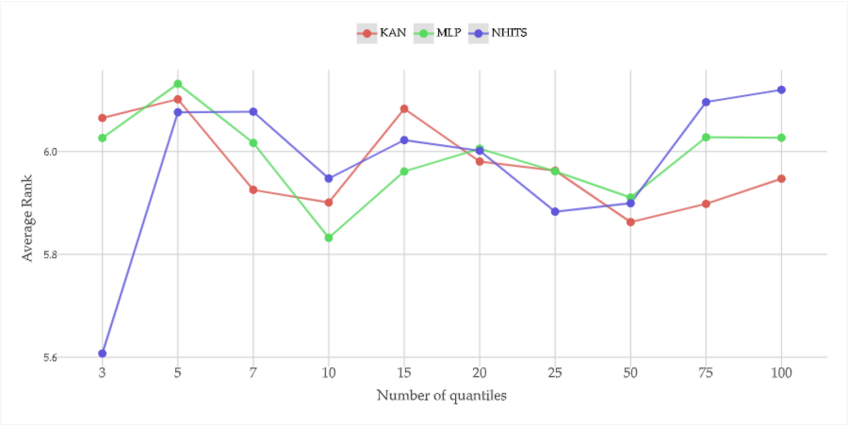


Figure 10: Average rank for different number of quantiles.

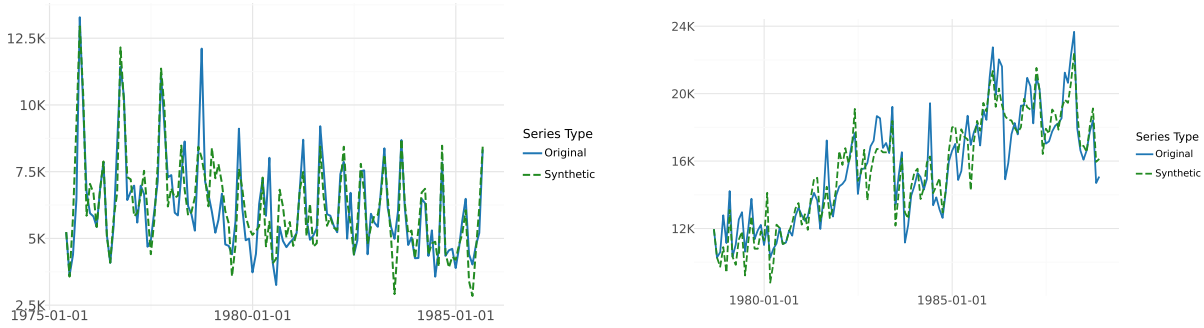


Figure 11: Example plot of generated vs original time series for two unique ID's.

This work is a result of Agenda “Center for Responsible AI”, nr. C645008882-00000055, investment project nr. 62, financed by the Recovery and Resilience Plan (PRR) and by European Union - NextGeneration EU. Funded by the European Union – NextGenerationEU. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them. This work is also funded by the European Union under the Horizon Europe Framework Programme Grant Agreement N°: 101095387. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the European Health and Digital Executive Agency can be held responsible for them, and by: UID/00027 of the LIACC - Artificial Intelligence and Computer Science Laboratory - funded by Fundação para a Ciência e a Tecnologia, I.P./ MCTES through the national funds.