Time series forecasting via Network Science

Filipe Godinho Justiça
Master's degree in Data Science
Computer Science Department
2022

Supervisor
Pedro Manuel Pinto Ribeiro, Assistant Professor, Faculty of Science, University of Porto

Co-supervisor
Maria Eduarda da Rocha Pinto Augusto da Silva, Associate Professor, Faculty of Economics, University of Porto

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, ______/______/_________

Sworn Statement

Filipe Godinho Justiça

30/9/2022

Abstract

In the past decades, the field of time series forecasting has been growing at an unprecedented pace. Such an event is due to the growing interest in this area, but, more importantly, to the fact that with the evolution of the modern world, time series data is widely available in all domains.

One of the fundamental steps of any forecasting method is mining important information on the time series being predicted. These features are used to estimate which time points should be used to reduce the error of the prediction task. However, most methods fall into a common issue, good data is bounded to observations recent in time; e.g. when someone is predicting the weather for tomorrow, this person will, most likely, think about the recent days and how the meteorology was in the last year or few years, though, the weather of the past few days might be more similar to what happened in decades or even centuries ago.

A new area that maps time series into complex networks and finds new characteristics of the data by calculating topological measures of the graphs has been receiving increasing interest from the research community. In such a way that in recent years forecasting algorithms based on these complex network mappings were developed.

The core of the present work is based on the use of network science to solve the limitations of most forecasting methods described above. To do so, firstly there is a description of the basic concepts used in this thesis. Then, a taxonomy of the field of "Forecasting via network science" is provided. Through it, it is possible to conclude that the methods developed until now make predictions by either using linear regressions between the last observation and its most similar neighbours or by computing embeddings of the graph to enrich the information given to a forecasting method already available.

Since none of these strategies solves the issue of temporal distance a novel implementation of a window encoding strategy is proposed. With this algorithm, it is possible to calculate topological features based on subgraphs of the series which can be used to make predictions. Also, the code that was developed to achieve this end is available in the *networktsf* package, as well as all other methods developed in the present thesis.

Having defined a way to characterize time series through the analysis of complex networks, the next step taken was to analyse which size of windows should be used, and which subgraphs and topological features capture the time series patterns the best. In order to make this study, the correlation of these topological features computed for all observations in multiple time series with well-known time series patterns was computed. In the end, it was shown that these measures are worth exploring for windows of size ranging from half the period to two times its value.

A new similarity algorithm based on the most relevant topological features found in the study described above was implemented. This algorithm allows calculating the coincidence similarity between any observation and all time points before it. To do so, the method uses the vector encoding strategy proposed in this work to compare the recent patterns of each observation.

The culmination of this thesis is the introduction of the *Forecasting based on Networks Similarities*, FbNS, prediction algorithm. This method finally solves the issue of not using good but old data, because it has no time restrictions. In fact, the used time points can be very far in the past if they are similar to the last observation before the prediction. This similarity is calculated with the similarity algorithm developed, which already uses the other implementations of this work and the knowledge gathered from the provided analysis. In the end, it is shown that the FbNS model can be used to generate good forecasts and that it actually uses old data to enhance its performance.

Keywords: Time Series, Time Series Forecasting, Similarity, Complex Networks, Characterization, Topological Features, Encoding, Embedding, Pattern Recognition

Resumo

Nas últimas décadas, a área da previsão de séries temporais tem vindo a crescer a um ritmo sem precedentes. Tal desenvolvimento deve-se ao crescente interesse neste ramo, mas, acima de tudo, ao facto de que, com a evolução do mundo moderno, as séries temporais estão amplamente disponíveis em todos os domínios.

Uma das etapas fundamentais de qualquer método de previsão é a recolha de informações importantes sobre a série temporal que está a ser prevista. Por sua vez, essas variáveis são utilizadas para estimar quais as observações que devem ser usadas para reduzir o erro na tarefa de previsão. Apesar disso, a maioria dos métodos têm um problema comum: bons dados estão vinculados a observações recentes no tempo; e.g., quando alguém está a prever a meteorologia para o dia seguinte, essa previsão será provavelmente baseada nos últimos dias e, quanto muito, na meteorologia do ano anterior ou dos últimos anos; no entanto, o clima dos últimos dias poderá ser mais semelhante ao de algumas décadas, ou mesmo séculos, atrás.

Uma nova área que mapeia séries temporais em redes complexas e encontra novas características dos dados através do cálculo de métricas topológicas dos grafos, tem vindo a crescer na comunidade científica. De tal forma que, nos últimos anos, foram desenvolvidos algoritmos de previsão baseados nestes mapeamentos em redes complexas.

O foco do presente trabalho é o uso da ciência de redes para resolver as limitações da maioria dos métodos de previsão descritos acima. Para tal, primeiramente, é feita uma descrição dos conceitos básicos utilizados nesta tese. De seguida, é fornecida uma taxonomia do campo de "Previsão via ciência de rede". Deste modo, é possível concluir que os métodos desenvolvidos até agora fazem previsões através de regressões lineares entre a última observação e os seus vizinhos mais semelhantes ou ao encontrar *embeddings* do grafo para enriquecer a informação dada a um método de previsão já disponível.

Uma vez que nenhuma dessas estratégias resolve a questão da distância temporal, é proposta a implementação de uma nova estratégia *window encoding*. Com este algoritmo, é possível calcular características topológicas, tendo por base subgrafos da série, que podem ser usados para fazer previsões. O código que foi desenvolvido para atingir este fim está disponível no pacote *networktsf*, assim como todos os restantes métodos desenvolvidos na presente tese.

Após a definição de uma forma de caracterizar séries temporais, através da análise de redes complexas, o passo seguinte foi analisar qual o tamanho de janela que deveria ser utilizado, e quais os subgrafos e as características topológicas que capturam melhor os padrões das séries temporais. Para elaborar tal estudo, foi calculada a correlação dessas variáveis topológicas de todas as observações em múltiplas séries temporais, com padrões bem conhecidos das mesmas. Em consequência, mostrou-se que vale a pena explorar essas medidas para janelas cujo tamanho varia entre metade do período e duas vezes o seu valor.

Posteriormente, foi implementado um novo algoritmo de similaridade, baseado nas características topológicas mais relevantes encontradas no estudo descrito acima. Este algoritmo permite calcular a semelhança entre qualquer observação e todos as anteriores a esta. Para isso, o método utiliza *vector encoding*, proposto neste trabalho para comparar os padrões recentes de cada observação.

O culminar deste estudo é a introdução do algoritmo de previsão *Forecasting based on Networks Similarities*, FbNS. Este método resolve o problema de não usar apenas dados relevantes, mas também antigos, dado que não tem restrições de tempo. Efetivamente, as observações utilizadas podem estar muito distantes no passado se forem semelhantes à última observação antes da previsão. Esta similaridade é calculada recorrendo ao *similarity finder*, que já utiliza as demais implementações deste trabalho e o conhecimento obtido a partir das análises fornecidas. Por fim, é demonstrado que o modelo FbNS pode ser usado para gerar boas previsões e que, de facto, utiliza dados antigos para melhorar o seu desempenho.

Palavras-chave: Séries Temporais, Previsão de Séries Temporais, Similaridade, Redes Complexas, Caracterização, Variáveis Topológicas, *Encoding*, *Embedding*, Reconhecimento de Padrões

Acknowledgements

The present thesis represents the closure of my master's degree journey. Hence, I dedicate this section to thanking some of the people who contributed to this project somehow.

Firstly, I would like to share my deepest gratitude to my supervisors, Professor Pedro Ribeiro and Professor Maria Eduarda da Silva, for all the knowledge shared and guidance provided, always supporting me and leading me towards the right path.

I would also like to express my sincere appreciation to Vanessa Silva for sharing her expertise in the field of this thesis from the very beginning of the present study.

Moreover, I want to truly thank my parents for all the support and for giving me the fundamental education to achieve this goal.

The support DareData and my friends have given me was much appreciated. DareData provided me with the time needed to complete the present work and my friends encouraged me and kept me optimistic.

Last but not least, I would like to thank my girlfriend. Throughout this journey, she was very patient and helpful with me, something that she did without flinching, while giving me the right mindset to complete this thesis.

Contents

List of Tables

List of Figures

Acronyms

ACF	Autocorrelation Function
ARIMA	Autoregressive Integrated Moving Average
CCF	Cross-Correlation Function
CNN	Convolutional Neural Network
DNVG	Directed Natural Visibility Graph
DWNVG	Directed Weighted Natural Visibility Graph
FbNS	Forecasting based on Networks' Similarities
MA	Moving Average
MAPE	Mean Absolute Percentage Error
MASE	Mean Absolute Scale Error
NVG	Natural Visibility Graph
VG	Visibility Graph
WNVG	Weighted Natural Visibility Graph

Chapter 1

Introduction

The time series forecasting field has been gathering interest in all domains, in fact, this area can be considered to be almost as old as humanity, since even the *Homo erectus* needed to predict where it was best to hunt. In recent years the need to make better forecasts increased at an unparalleled speed, this is due to the evolution of biomedical devices which have very precise measurements; the growing interest in data science by companies; the internet of things; and many other areas. With this in mind, the scientific community developed many forecasting algorithms, ranging from autoregressive models, like the ARIMA, to modern methodologies like deep learning. All these methods share a common step while being fitted: mining features from the time series.

Network Science is a recent area that is being well recognized for its capability of describing a wide range of systems in nature and society. The impact of this field has been so enormous, that in the past decade a new domain connecting network science to time series was introduced. With the evolution of this conciliation, some authors have shown that besides analysing characteristics of the series through the topological measures gathered from complex networks mapped from the time series, network science can also be used to develop forecasting models.

1.1 Goal and Contributions

Even though many of the forecasting models available are able to generate great predictions with minimal errors, most of them are limited by recent data. It is true, that when the models are being fitted the entire dataset is used to estimate the parameters of the algorithm, yet, usually, when making predictions, only recent observations are used to forecast.

The main goal of this thesis is to present a solution to the problem previously described by using the new domain connecting time series forecasting to network science. With the purpose of achieving this goal, this work is divided into three main components: the characterization of observations by computing topological features and finding which ones make a better representation of the data; using these topological measures to find similar observations; the combination of the findings of the previous chapters to create a forecasting algorithm not bounded by time.

The main contributions of the work described in this thesis are the following:

1. A taxonomy of the field of time series forecasting via network science is purposed
2. An implementation of a window encoding strategy is presented. This algorithm allows computing topological features of the subgraphs of size m ending at the desired observation
3. The degree, average shortest path length, harmonic centrality and betweenness topological features computed with the purposed encoding strategy is then analysed for different windows sizes, different mappings and compared between each other.
4. Both the window encoding and the study of the topological features are then used to build a similarity algorithm that computes the coincidence similarity between all observations and their past.
5. The combination of all algorithms developed and the studies made with them lead to the creation of a powerful new forecasting method, the *Forecasting based on Networks Simmilarities* (FbNS), which is not limited by the common issue of many forecasting models.
6. Network time series forecasting, **networktsf**, is a python library that has three main modules:

   Mappings : Visibility mappings explained in section 2.3

   Network Features :
   - Window Encoding (section 4.1)
   - Pipeline that collects topological features from a selected mapping method.
   - Similarity Finder (section 5.1)

   Forecasting Models :
   - Enriched ARIMA model (section 6.3);
   - Forecasting based on Networks Similarities (section 6.1).

1.2 Organization

This thesis is structured into seven major chapters. A brief description of each one of them is now provided.

Chapter 1 - Introduction. Provides an introduction to the research area, the goals and contributions, as well as the organization of the thesis.

Chapter 2 - Basic Concepts. Introduces time series and complex networks terminology. Furthermore, the main methods of time series to complex networks mapping that will be used in the present work are briefly described.

Chapter 3 - Complex Networks and Time Series Forecasting. Presents a summary of the prediction algorithms developed in this novel field. A taxonomy of the area is also proposed.

Chapter 4 - Topological Characterization of Observations. A window encoding strategy to compute topological features based on their past is proposed. Then a study of the usefulness of this implementation is made with an extensive analysis of the window size, mapping methods and features that represent multiple time series in the best way possible.

Chapter 5 - Topological Similarity between Observations. Presents a novel similarity detection algorithm that makes use of the findings of the previous chapters. The method allows finding correlated observations based on the patterns of their past.

Chapter 6 - Forecasting based on Networks Similarities. The culmination of the developed work. The FbNS method is proposed to solve the issue of recent time dependency. Also, an analysis of the performance of the model in multiple time series is provided.

Chapter 7 - Conclusions. Discusses the research done and summarizes the contributions done.

Chapter 2

Basic Concepts

The necessary notation for the remaining thesis about time series, complex networks, forecasting models, and existing methods to map time series into complex networks is explained in this chapter.

2.1 Time Series

A time series $Y = (y_1, \dots, y_T)$ is an ordered sequence of random variables $\{Y_t\}_t$, that are collected from measurements observed at uniformly spaced instants [8], t. The time series' principal characteristic is the dependence among its observations, which, although extremely interesting, limits the applicability of conventional statistical methods that assume independent and identically distributed (i.i.d) observations. To solve this problem, time series analysis provides methods that describe the characteristics of the data with the objective of forecasting and simulation [3], always taking into consideration the time correlations.

The study of univariate time series is immensely relevant; however, it is often necessary to include several other variables measured over time, $Y_{i,t}, i = 1, 2, \dots m$, that may be correlated. For instance, the blood glucose levels and the amount of insulin of a patient need to be analysed jointly over time. $Y_t = [Y_{1,t}, Y_{2,t}, \dots, Y_{m,t}]'$ is the usual representation of multivariate time series, where $'$ is the transpose operator and $Y_{i,t}$ is the i-th component time series (a random variable for each i and t).

Similar to univariate time series, the principal characteristic of the multivariate case is that its observations are correlated; yet, in the latter, there is not only serial dependence within each component series $Y_{i,t}$, but also interdependence between the different components $Y_{i,t}$ and $Y_{j,s}$, when $i \neq j$, regardless of whether the time s and t are the same or not. Although the theory of univariate time series extends naturally to the multivariate case, new issues need to be considered and even fundamental concepts must be re-defined [32].

2.1.1 Stochastic Processes and Measures of Dependence

A stochastic process is a sequence of random variables indexed by time t, and, since time series are sequences of N successive observations, they are regarded as finite realizations of such processes. From this definition, arises that a time series can be described by the joint distribution function $F(c_1, \ldots, c_n) = P(y_{t_1} \leq c_1, \ldots, y_{t_n} \leq c_n)$, which can only be easily obtained in the particular case of jointly Gaussian random variables.

Stochastic processes' characteristics allow describing the serial correlation of time series data. Since each time step is composed of a random variable, it is possible to define a **mean function** that corresponds to the mean of the random variable of each instant

$$\mu_t = E(y_t) \tag{2.1}$$

in which E denotes the expected value operator.

Although very useful, the mean function is not able to represent the dependence between the random variables. In this sense, other measures that lead to a better understanding of the correlation between observations have to be defined. One of which is the second-moment product, **autocovariance function**, and is defined as

$$\gamma_y(s,t) = cov(y_s, y_t) = E[(y_s - \mu_s)(y_t - \mu_t)] \tag{2.2}$$

for all s and t.

The autocovariance measures the linear dependence between two points on the same series observed at different times [28], leading to the conclusion that, for $s = t$, the autocovariance reduces to the variance of the random variable y_t, since

$$\gamma_y(t,t) = E[(y_t - \mu_t)^2] = var(y_t). \tag{2.3}$$

In order to study the main characteristic of time series, the serial correlation between observations, the **autocorrelation function (ACF)** was defined, and it is written as follows

$$\rho_y(s,t) = \frac{\gamma_y(s,t)}{\sqrt{\gamma_y(s,s)\gamma_y(t,t)}}. \tag{2.4}$$

It is worth noticing that $\rho_y(s,t) \in [-1,1]$, thus, when $|\rho_y(s,t)| = 1$, y_s and y_t are perfectly correlated, and, if $\rho_y(s,t) = 0$, the pair is perfectly uncorrelated. Hence, as correlation measures a linear relationship between two variables, autocorrelation measures the linear relationship between lagged values of a time series [16].

In the case of multivariate time series, the concepts of autocorrelation and autocovariance still apply for representing dependence within each component series $Y_{i,t}$. Nevertheless, the interdependence between the different components $Y_{i,t}$ and $Y_{j,s}$ is not captured. The **cross-covariance** and **cross-correlation functions** serve exactly for that purpose.

The cross-covariance between two series $Y_{i,t}$ and $Y_{j,s}$ is

$$\gamma_{Y_i,Y_j}(s,t) = E[(Y_{i,s} - \mu_{Y_{i,s}})(Y_{j,t} - \mu_{Y_{j,t}})]. \tag{2.5}$$

And the cross-correlation (**CCF**)

$$\rho_{Y_i,Y_j}(s,t) = \frac{\gamma_{Y_i,Y_j}(s,t)}{\sqrt{\gamma_{Y_i}(s,s)\gamma_{Y_j}(t,t)}}. \tag{2.6}$$

Both functions can easily be re-defined for the case of the m series. In this situation, a **covariance matrix** is defined as

$$\Gamma_Y(s,t) = \begin{bmatrix} \gamma_{Y_1}(s,t) & \gamma_{Y_1,Y_2}(s,t) & \dots & \gamma_{Y_1,Y_m}(s,t) \\ \gamma_{Y_2,Y_1}(s,t) & \gamma_{Y_2}(s,t) & \dots & \gamma_{Y_2,Y_m}(s,t) \\ \vdots & \vdots & \vdots & \vdots \\ \gamma_{Y_m,Y_1}(s,t) & \gamma_{Y_m,Y_2}(s,t) & \dots & \gamma_{Y_m}(s,t) \end{bmatrix} \tag{2.7}$$

in which, when $i = j$, $\gamma_{Y_i}(s,t)$ is the autocovariance function for the i-th component process; and when $i \neq j$, $\gamma_{Y_i,Y_j}(s,t)$ is the cross-covariance function between component series Y_i and Y_j [32].

The following matrix is especially helpful, as it captures the dependence within and between components. Also, it is clear that the same idea can be applied to create a **correlation matrix**

$$P_Y(s,t) = \begin{bmatrix} \rho_{Y_1}(s,t) & \rho_{Y_1,Y_2}(s,t) & \dots & \rho_{Y_1,Y_m}(s,t) \\ \rho_{Y_2,Y_1}(s,t) & \rho_{Y_2}(s,t) & \dots & \rho_{Y_2,Y_m}(s,t) \\ \vdots & \vdots & \vdots & \vdots \\ \rho_{Y_m,Y_1}(s,t) & \rho_{Y_m,Y_2}(s,t) & \dots & \rho_{Y_m}(s,t) \end{bmatrix}. \tag{2.8}$$

A particular case of stochastic processes is obtained when the process is in a state of *statistical equilibrium* [28]. In the case of time series, this type of behaviour is represented by regularity over time, which leads to the notion of **stationarity**.

Time series can either be **strictly or weakly stationary**. The first defines a series for which the probabilist behaviour is not affected by shifting all the times by any integer amount h,

$$P\{y_{t_1} \leq c_1, \dots, y_{t_k} \leq c_k\} = P\{y_{t_1+h} \leq c_1, \dots, y_{t_k+h} \leq c_k\} \tag{2.9}$$

for all $k = 1, 2, \dots$, all time points $t_1, t_2, \dots, t_k$, all numbers $c_1, c_2, \dots, c_k$, and all time shifts $h = 0, \pm 1, \pm 2, \dots$.

As the name suggests, this definition is too strict, given that the whole probabilistic structure of the process must only depend on time differences. A weakly stationary time series is derived from the same concept, but is less restrictive since it only imposes conditions on the first two moments of the series, such that:

(i) the mean value function, μ_t, is constant and does not depend on time t;

(ii) the autocovariance function, $\gamma_y(s,t)$, depends on s and t only through their difference $h = |s-t|$.

Herewith, the mean of each random variable of a weakly stationary series writes as follows

$$\mu_t = \mu. \tag{2.10}$$

Also, the corollary of the second statement is that the autocovariance function of such series does not depend on the time argument t, considering that

$$\begin{aligned} \gamma_y(t+h,t) &= E[(y_{t+h}-\mu)(y_t-\mu)] \\ &= E[(y_h-\mu)(y_0-\mu)] \\ &= \gamma_y(h,0). \end{aligned} \tag{2.11}$$

Consequently, the autocorrelation function of a stationary time series will be written as

$$\rho_y(h) = \frac{\gamma_y(t+h,t)}{\sqrt{\gamma_y(t+h,t+h)\gamma_y(t,t)}} = \frac{\gamma_y(h)}{\gamma_y(0)}. \tag{2.12}$$

Henceforth, a time series is stationary when its first two momentums (mean, variance) are constant over time, implying that correlation only depends on the time lag between observations.

When considering multivariate analysis, two time series are considered to be **jointly stationary** if:

(i) they are each stationary;

(ii) the cross-covariance is a function only of lag h [28].

The same concept applies to m series, and the correlation matrix would be written as

$$P_Y(h) = \begin{bmatrix} \rho_{Y_1}(h) & \rho_{Y_1,Y_2}(h) & \dots & \rho_{Y_1,Y_m}(h) \\ \rho_{Y_2,Y_1}(h) & \rho_{Y_2}(h) & \dots & \rho_{Y_2,Y_m}(h) \\ \vdots & \vdots & \vdots & \vdots \\ \rho_{Y_m,Y_1}(h) & \rho_{Y_m,Y_2}(h) & \dots & \rho_{Y_m}(h) \end{bmatrix} \tag{2.13}$$

where $\rho_{Y_i}(h)$ is the autocorrelation function of the i-th component stationary time series; and the cross-correlation [28]

$$\rho_{Y_i,Y_j}(h) = \frac{\gamma_{Y_i,Y_j}(s,t)}{\sqrt{\gamma_{Y_i}(0)\gamma_{Y_j}(0)}}. \tag{2.14}$$

In conclusion, m series are said to be jointly stationary when all of them are stationary and the cross-covariances are constant over time, which entails that the autocorrelations and cross-correlations only depend on the time lag h.

2.1.2 Time Series Patterns

Stationary is of utmost importance for many forecasting models. However, most real-life processes' first two momentums (mean, variance) are not constant over time. Usually, time series are characterized by three fundamental patterns [16]:

Seasonality This pattern occurs when a time series presents similar characteristics between observations separated by a fixed amount of lags. Plus, univariate sequences may have more than one seasonal pattern.

Trend When the oscillation between consequent observations corresponds to an increase or decrease for a long time, the time series may present a trend, which does not need to be linear, and sometimes it may "change direction" (when it goes from an increasing trend to a decreasing trend or vice-versa).

Cycles Fluctuations that occur at a none fixed frequency are interpreted as cycles that usually happen due to external conditions.

Besides these characteristics, a series may also present changes in the variance, which makes those series very difficult to predict. Solving this problem passes by normalizing the series values by performing a Box-Cox transformation [2].

Performing a decomposition of the time series in the three main patterns described above results in distinct processes, one representing the trend, another one the seasonality and the third is the random component of the series. The classical additive approach of this process follows the steps below:

1. The trend-cycle, $\hat{T}_t$, patterns are computed by using a Moving Average [15].
2. Then, the trend-cycle is removed from the series $y_t - \hat{T}_t$.
3. Finally, computing the mean value for each season in the detrended series, while replicating each value for all seasons across the entire series, results in the seasonal component $\hat{S}_t$.
4. In the end, the residuals are calculated by subtracting the estimated seasonal and trend-cycle components: $\hat{R}_t = y_t - \hat{T}_t - \hat{S}_t$.

The classical multiplicative decomposition is similar, except that the subtractions are replaced by divisions. The results of the application of this method are seen in figure 2.1, and the series used is the Monthly Milk Production by cow [5], which presents clear seasonal characteristics, as well as trend-cycle ones.

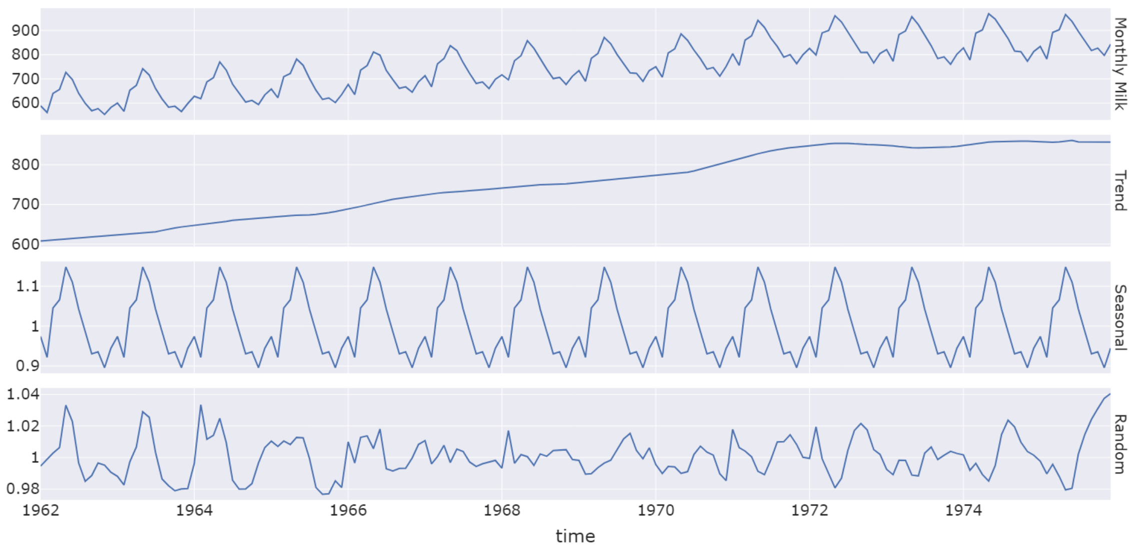


Figure 2.1: A classical additive decomposition of the Monthly Milk production by cow series.

2.1.3 Time Series Forecasting

The measures presented in subsection 2.1.1 and the patterns presented in the previous subsection help express the current and past values of a time series, so that future values can be predicted. Forecasting is one of the most important topics of time series analysis, especially because it has countless applications across several fields [3], by being a great aid to effective and efficient planning [16].

From the statistical perspective, the aim of forecasting is to predict Y_{T+h}, while knowing $Y_T = (y_1, \ldots, y_T)$, even if Y represents a multivariate time series. The forecast horizon, $T + h$, highly depends on the field of study, and influences the predictability of the event. For instance, if one is trying to forecast the weather for tomorrow, there is a high chance of making a good prediction. On the other hand, when trying to forecast it for the following year, it is very likely to make a bad prediction. This concept of predictability is of great importance in the context of time series forecasting and depends on several factors, including the characteristics of the process and the goal of the prediction.

In order to reach the necessities of the modern world in terms of time series forecasting, several algorithms were introduced in the past decades. These methods are divided into two categories: parametric and nonparametric.

In a classic model (parametric) the goal of the algorithm is to estimate the parameters of the probability distribution assumed for the time series data [10]. Furthermore, this set of methods is either linear or nonlinear. Essentially, the first ones are models for which the conditional mean is a linear function of past time series values. Whilst, nonlinear algorithms focus on using nonlinear parametric forms to provide better forecasts in series that present changes in their characteristics across time, like the presence of different regimes (where the mean, variance and autocorrelation depend on the regime) [9]. The following list presents a brief description of some popular forecasting methods:

AutoRegressive Model (AR) A time series can be described by this type of model when it satisfies the following equation

$$\begin{aligned} y_t &= \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t \\ (1 - \sum_{i=1}^{p} \phi_i B^i) y_t &= \epsilon_t \\ \Phi(B) y_t &= \epsilon_t \end{aligned} \tag{2.15}$$

where B represents the backshift operator, $By_t = y_{t-1}$, p is the number of autoregressive terms, and ϵ_t is a white noise process.

AutoRegressive Moving Average (ARMA) This method is described by a combination of AR and Moving Average (MA) processes. As such, a stationary time series, y_t, is an ARMA process of order (p, q) if it satisfies the equation:

$$\begin{aligned} y_t &= \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t \\ \left(1 - \sum_{i=1}^{p} \phi_i B^i\right) y_t &= (1 + \sum_{i=1}^{q} \theta_i B^i) \epsilon_t \\ \Phi(B) y_t &= \Theta(B) \epsilon_t \end{aligned} \tag{2.16}$$

where the white noise ϵ_t is usually a Gaussian process, ϕ_i, $i = 1, \dots, p$ are constants such that $\Phi(z) = 1 - \sum_{i=1}^{p} \phi_i z^i \neq 0$ for $|z| \leq 1$, and θ_i, $i = 1, \dots, q$ are constants such that $\Theta(z) = 1 + \sum_{i=1}^{q} \theta_i z^i \neq 0$ for $|z| \leq 1$.

AutoRegressive Integrated Moving Average (ARIMA) A nonstationary time series, x_t, but whose dth-difference $y_t = \nabla^d x_t$ is stationary, may be described by a ARMA(p, q) process:

$$\begin{aligned} \Phi(B) y_t &= \Theta(B) \epsilon_t \\ \Phi(B) y_t (1 - B)^d x_t &= \Theta(B) \epsilon_t \end{aligned} \tag{2.17}$$

Thus, the ARIMA(p, d, q) algorithm generalizes ARMA(p, q) models, making it very useful to predict series that present trend-cycle patterns, without seasonality.

Seasonal AutoRegressive Integrated Moving Average (SARIMA) In the context of time series that present both trend and seasonal patterns, ARIMA processes are insufficient to describe the sequences. Adding seasonal terms to the ARIMA models is written as follows:

$$ARIMA(p, d, q)(P, D, Q)_m \tag{2.18}$$

where m is the seasonal period, and the upper case notation corresponds to seasonal parts of the model, which consist of terms that are similar to the non-seasonal components of the model but involve backshifts of the seasonal period.

The scientific community developed other linear models, or even non linear. Besides that, non-parametric methods like regression models, deep learning, and machine learning models are becoming a standard in this field as well. In the present work, the SARIMA model is used to produce forecasts and compare them with the developed methods, since it is one of the most used prediction algorithms and usually establishes a good baseline in terms of the results obtained. Moreover, this type of model can be used with exogenous regressors by forcing the model to be a linear regression:

$$\begin{aligned} y_t &= \beta_0 + \sum_{i=1}^{n} \beta_i x_{i,t} + \eta_t \\ \eta_t &= ARIMA(p,d,q)(P,D,Q)_m \end{aligned} \tag{2.19}$$

where x_i is the i^{th} exogenous variable.

Every model described above has multiple parameters that are estimated through ***Maximum Likelihood Estimation***, which is similar to the least squares estimates that would be obtained by minimizing:

$$\sum_{t=1}^{N} \epsilon_t^2 \tag{2.20}$$

$$\epsilon_t = y_j - \hat{y}_t \tag{2.21}$$

where $\hat{y}$ is the prediction of the model, and ϵ_t the residuals.

Another important step to find the best possible model is to select the best hyperparameters of the regressor (e.g in the context of the SARIMA: (p,d,q), (P,D,Q) are the hyperparameters). This task is not that simple, as the number of possible models is endless. As such, several approaches have been developed to find the best model automatically. The ***Auto-Arima*** method is the same as an ARIMA or SARIMA algorithm, though, the hyperparameters are selected by computing an information criterion of each model obtained by a specific order, and choosing the model that minimizes its value [18].

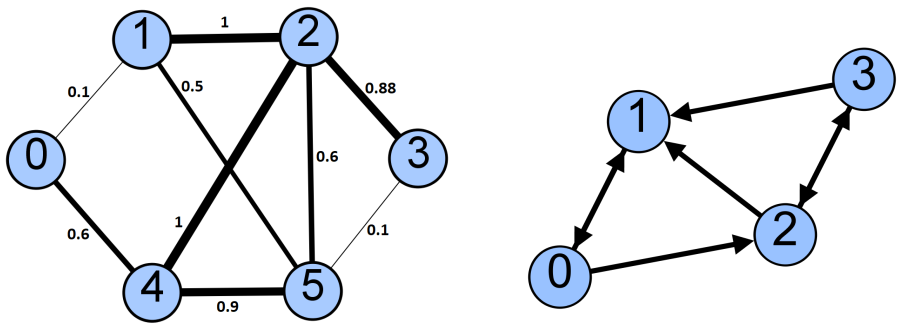
Analysing the performance of a model is a necessary step in order to understand if it is able to describe the series the model was fitted on. To do so, several metrics have been introduced, namely, the *Mean Absolute Percentage Error*, *Mean Squared Error*, or the *Mean Absolute Scaled Error*. In the present thesis, the models will be compared by computing the ***Root Mean Squared Error*** (RMSE) and the ***Mean Absolute Error*** (MAE):

$$RMSE = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(y_j - \hat{y}_t)^2} \tag{2.22}$$

$$MAE = \frac{1}{N}\sum_{j=1}^{N}|y_j - \hat{y}_t| \tag{2.23}$$

2.2 Complex Networks

The branch of mathematics dedicated to studying the properties of the interactions between elements of a complex system is called *graph theory*. In this area, networks (or graphs), G, are used to make a graphical representation of the problem, where the elements are represented by nodes (or vertices), $V(G)$, and their interactions by links (or edges), E(G). Therefore, graphs are ordered pairs of nodes and edges, $G = (V(G), E(G))$. Also, the number of nodes (or size of the graph) is represented by $|V(G)|$, and the number of links by $|E(G)|$.



(a) Undirected weighted graph (b) Directed graph

Figure 2.2: A representation of (a) a simple directed graph and (b) a simple undirected weighted graph.

The networks shown in Figure 2.2 are defined as *simple* since they do not contain multiple links (two or more links connecting the same pair of nodes) or any self-loops (a link connecting a node to itself).

In Figure 2.2a, the node v_0 is a *neighbour* of the node v_1 given that they are connected by a link. This means that two nodes v_i and v_j are neighbours if $(v_i, v_j) \in E(G)$. Moreover, the graph also is a *weighted* graph, since each link (v_i, v_j) has an associated weight $w_{i,j}$.

One of the fundamental differences between Figures 2.2a and 2.2b is the fact that graph 2.2a is an undirected graph while 2.2b is a *digraph*. This means that in second graph each link has a source and target node, so the edges are *directed*.

Figure 2.2 is a graphical representation of the networks. Even so, another possible way of presenting graphs is through their *adjacency matrix*, G_{Adj}, where:

$$G_{Adj}[i][j] = \begin{cases} 1 & \text{if } (v_i, v_j) \in E(G) \wedge G \text{ is unweighted} \\ w_{i,j} & \text{if } (v_i, v_j) \in E(G) \wedge G \text{ is weighted} \\ 0 & \text{otherwise} \end{cases} \tag{2.24}$$

Accordingly, the adjacency matrix is a mathematical way of representing the network while preserving its properties. Although these matrices are hugely useful to perform calculations, they are not as helpful for quickly visualizing fundamental features of the graph, like its *paths*, *connectivity* or the properties of subgraphs of the original network.

A **path** is a sequence of nodes in which each consecutive pair of nodes in the sequence is connected by a link.

The **connectivity** studies the paths between nodes in the entire network. Two nodes are said to be connected if a path exists between them, while the graph is defined as connected if there is a path between every pair of nodes. This property is extremely practical to partition the nodes in non-overlapping subsets of connected nodes known as connected components [4], since there are no paths between distinct components.

A **subgraph** SG_k of size k of a graph G is a graph with k nodes in which $V(SG_k) \subseteq V(G)$ and $E(SG_k) \subseteq E(G)$.

2.2.1 Topological Features

Network science helps capture the patterns of interactions between the parts of a system, for instance, made through visualization for graphs up to a few hundreds or thousands of nodes, and for networks that are relatively sparse, meaning that the number of edges is quite small [25]. The capture of these patterns can also occur by using topological measures of the networks. The majority of these features can be labeled as local or global measures, i.e., those related to individual nodes or links, or those related to the graph as a whole. In this thesis, we are focused on local metrics, and a global measure based on the average number of links between any two nodes in a graph.

The **Average Shortest Path Length** is denoted by $\bar{d}$, the arithmetic mean of the shortest paths (d) among all pairs of nodes:

$$\bar{d} = \frac{1}{n(n-1)} \sum_{i}^{n} \sum_{j \neq i}^{n} d(i,j) \tag{2.25}$$

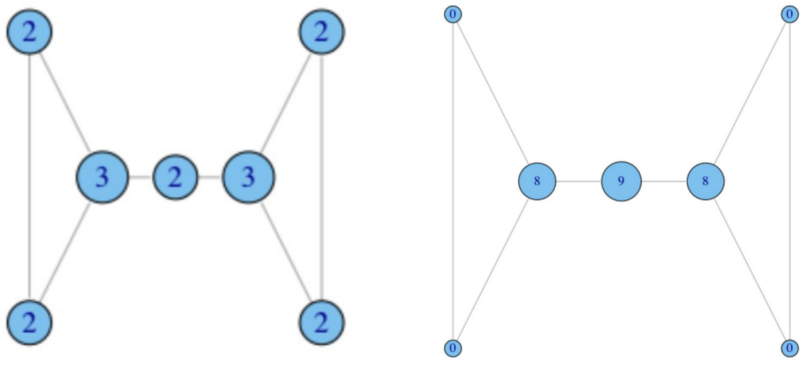
Where the path length is the number of links, or the sum of the links weights if the graph is weighted, in a path between any two nodes in the same component [1].

The concept of centrality addresses the question, "What are the most important or central nodes in a network?" [25]. Answering this question leads to the development of several centrality measures that define importance differently, meaning that the combination of multiple centrality measures can help describe the network patterns in a richer and more useful way.

The *simplest* centrality measure, the **degree**, $k(i)$, of a node v_i, is equal to its number of neighbours. In the context of digraphs, this metric is extended by calculating both the number of edges that originate in the given node, **in-degree**, $k^{in}(i)$, and the number of links ending in that node, **out-degree**, $k^{out}(i)$. Besides, if the graph is weighted, the weight of each connection that the node belongs to is taken into account. Mathematically, the **weighted degree**, $k^{w}(i)$, of a vertex is calculated by summing the weights of its edges:

$$k^{w}(i) = \sum_{j=1}^{|V(G)|} w_{i,j}. \tag{2.26}$$

The degree is a very practical measure to capture important nodes in the network, though, vertices may be influential without having many neighbors. One example of such nodes is the ones that work as a bridge between two connected components. This situation is exemplified in the graph from Figure 2.3, since the node in the middle of the graph is very important to the global features of the network, but its degree is smaller than the degree of its neighbors.



(a) Node degrees (b) Node betweenness centrality's

Figure 2.3: A representation of (a) a simple graph with the degree of each node and (b) the same network, but with the non-normalized betweenness centrality of each node.

One way to capture the importance of such nodes is by calculating how many shortest paths between two other nodes in the network pass through the given node. This concept led to the definition of a centrality measure called **betweenness centrality**, calculated as follows:

$$bc(i) = \sum_{j<k} \frac{g_{j,k}(i)}{g_{j,k}} \tag{2.27}$$

where $g_{j,k}$ is the number of shortest paths connecting the node j to k, and $g_{j,k}(i)$ the number where i is on.

The $bc(i)$ is usually normalized by the number of pairs of vertices, excluding the vertex itself:

$$bc^{'}(i) = \frac{bc(i)}{(n-1)(n-2)/2} \tag{2.28}$$

Another noteworthy centrality measure is the **harmonic centrality**, which gives more importance to nodes close to vertices with high degree, independently of the number of neighbours or how between "others" they are. This metric is based on the length of the average shortest path between a node and all other nodes in the network:

$$hc(i) = \sum_{d(i,j)<\infty, j\neq i} \frac{1}{d(i,j)} \tag{2.29}$$

$$hc^{'}(i) = \frac{hc(i)}{(n-1)} \tag{2.30}$$

All of this measures, including the Average Shortest Path Length, help getting a fuller picture of the graph or node they are computed on. Other metrics could be used to achieve the same purpose, or even to give another perspective, however in this thesis the topological measures studied are the ones described above.

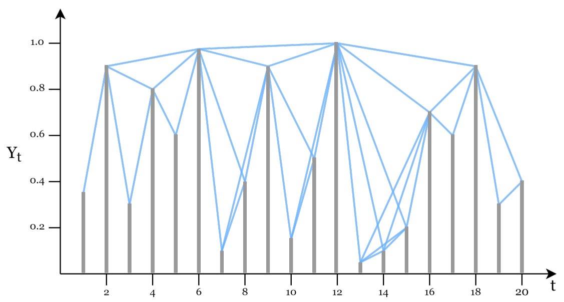
2.3 Time Series Mappings

In the last decade several network-based time series analysis approaches have been proposed. These approaches are based on the mapping of univariate and multivariate time series to the network domain, namely in single layer or multiple layer networks [30]. The mappings proposed in the literature are essentially based on concepts of visibility, transition probability, proximity, time series models and statistics [30]. So far, the biggest focus has been on approaches to mapping univariate time series into single layer structure.

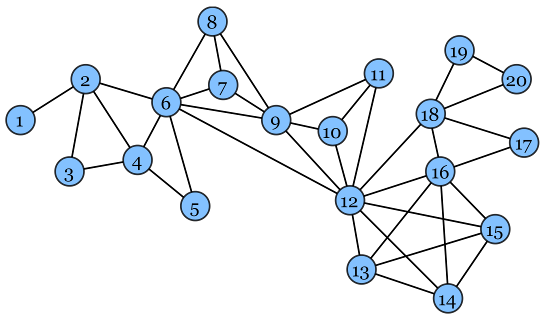
Visibility mappings establish connections between observations (nodes in the graph) of the time series using visibility lines (with or without restrictions). The transition mappings are based on the transition probabilities between states/partitions defined by dividing the time space in a set of temporal states (or dividing the series support/observations into partitions) that will be the vertices of the network. Proximity mappings establish connections using measures of distance or similarity between observations (or states), which become network vertices.

2.3.1 Natural Visibility Graphs

Lacasa and co-authors [20] proposed the first method based upon the concept of visibility: the natural visibility graph (NVG), or simply visibility graph (VG). This method stands on the idea that each observation of the time series is seen as a vertical bar with a height equal to the numerical value of the observation, and that these vertical bars are laid in a landscape, where the top of a bar is visible from the tops of other bars. Each node in the graph corresponds to a time stamp t of the time series, so the nodes are serially ordered. Two nodes are connected if there is a line of visibility between the corresponding data bars that are not intercepted. This idea is illustrated in Figure 2.4a.



(a) Toy time series



(b) NVG

Figure 2.4: In (a), a plot of a toy time series is presented, and, in (b), the respective graph, generated by the natural visibility graph algorithm. The color lines in the time series plot represent the visibility lines (and hence the edges of the graph) between the observations. *Adapted from:* [30]

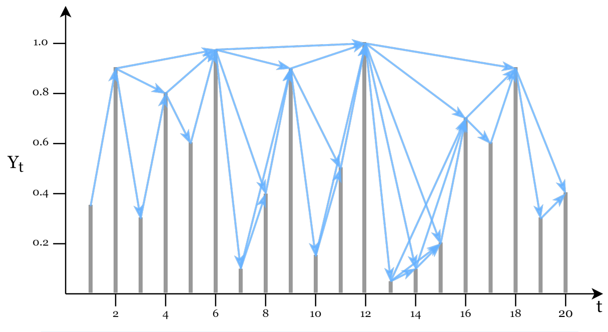
Formally, the set of nodes $\{v_i\}$ of an NVG is numbered sequentially in time and two nodes, v_i and v_j, are connected (have visibility) if any other observation, (t_k, Y_k) with $t_i < t_k < t_j$, satisfies:

$$Y_k < Y_j + (Y_i - Y_j)\frac{(t_j - t_k)}{(t_j - t_i)}. \tag{2.31}$$

Visibility graphs are connected at all times, since each node v_i has visibility for at least its neighbors v_{i-1} and v_{i+1}. Besides, they are always undirected. Nonetheless, a direction can be defined considering the direction of the time axis or the series values. The network is also invariant under affine transformations of the data [20], because the visibility criterion is invariant under rescheduling of both the horizontal and vertical axis, as well as in vector translations, that is, each transformation $Y' = aY + b$, for $a \in \mathbb{R}$ and $b \in \mathbb{R}$, leads to the same NVG.

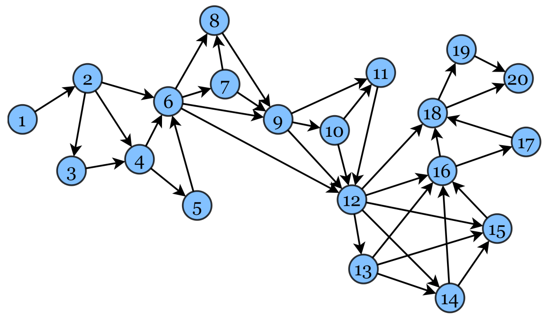
2.3.2 Directed Visibility Graphs

Given that time has a natural direction, the directed natural visibility graphs, DNVG, can be derived by defining an NVG with edges that follow the **time direction** (v_i, v_j), $i < j$, or the node with a larger value (v_i, v_j), $y_i < y_j$. (**the series direction**). Note that the adjacency matrix is not a symmetric matrix. An example of the representation of this algorithm is illustrated in Figure 2.5.



k_i^{out}:	1	3	1	2	1	4	2	1	3	2	1	5	3	2	1	2	1	2	1	0
k_i^{in} :	0	1	1	2	1	3	1	2	3	1	2	4	1	2	3	4	1	3	1	2
k_i :	1	4	2	4	1	7	3	3	6	3	3	9	4	4	4	6	2	5	2	2

(a) Toy time series



(b) DNVG

Figure 2.5: Illustrative example of directed natural visibility algorithm and corresponding *out*-degree ($k(i)^{out}$), *in*-degree ($k(i)^{in}$) and degree ($k(i)$). In (a), a plot of a toy time series is presented, and, in (b), the network that is generated by the directed natural visibility algorithm. The color-directed lines represent the directed visibility lines between the observations.
Modified from: [30]

This version of VGs was used to propose a set of rigorous statistical tests for time series irreversibility [1] [7].

2.3.3 Weighted Visibility Graphs

The method proposed by Supriya and co-authors [31] is a fairly simple modification to the traditional NVG algorithm, and it considers the NVG edges as directed and weighted. For a given time series, the corresponding weighted visibility graph (WVG) (or weighted directed visibility graph, WDVG) is built as follows: a directed NVG is constructed as described above, and a weight, $w_{i,j}$, equal to the view angle between the observations (t_i, Y_i) and (t_j, Y_j) in time series, is assigned to the edge that connects the corresponding nodes. The angle is given by:

$$\alpha_{i,j} = \tan^{-1}\left(\frac{Y_j - Y_i}{t_j - t_i}\right), \quad i < j. \tag{2.32}$$

It should be kept in mind that Equation (2.32) allows the attribution of not only positive weights but also negative weights to the edges of the WVG. However, the analysis of networks with negative weights is more complex, as standard methods and techniques do not apply straightforwardly [19], and, therefore, it is less common.

The problem of using negative weights is too big to ignore. As such other authors proposed other metrics, like the absolute value of the angle or distances between the nodes in the time series space. The inverse of this distances is also used in some studies with the goal of giving

[1] A stationary time series is reversible if $\{Y_1, \ldots, Y_T\}$ and $\{Y_T, \ldots, Y_1\}$ have the same joint probability distributions.

more value to the edges between close observations. As such, besides the view angle, other measures can be considered for the weight of the edges of an NVG, namely:

- Vertical distance: $e_{i,j} = |Y_j - Y_i|$
- Inverse vertical distance $e_{i,j} = 1/|Y_j - Y_i|$
- Euclidean distance: $w_{i,j} = \sqrt{(t_j - t_i)^2 + (Y_j - Y_i)^2}$
- Inverse of Euclidean distance: $w_{i,j} = 1/\sqrt{(t_j - t_i)^2 + (Y_j - Y_i)^2}$

Chapter 3

Complex Networks and Time Series Forecasting

The field of forecasting dates goes back to thousands of years when ancient civilizations observed natural patterns to anticipate the future; e.g. storms at the sea were predicted by observing an abundance of seagulls taking refuge inland, and nomads would travel from one place to another accordingly to the seasons because they foresaw that certain locations would have more resources at a given time. More recently, forecasting is required in all kinds of areas, and with the technological advances of the last decades, as well as the adoption of the Internet of Things, time series data is widely available in all domains.

Due to the heterogeneity of available data and the growing interest in predicting future events, new problems arise. Forecasting accurately depends not only on how well we understand the factors that contribute to the event being forecasted or the amount of data available [16] but also on the methods used. Since traditional prediction algorithms have some limitations, new approaches that combine multiple areas are being developed.

In recent years, the field that connects network science to time series forecasting is starting to captivate the scientific community. This field, although recently founded, seems to be promising, as several approaches for time series analysis based on network science methodologies lead to great results. For instance, Lacasa and co-authors ([20]) showed that NVGs inherit several properties of the time series. Visibility graphs were used to study energy dissipation rates in three-dimensional turbulence [22], financial time series [34], heart rate variability [26], and sleep stages [37].

Despite very few studies having been done in this field, it is already possible to establish connections between past works. In this chapter, we provide a conceptual division of the forecasting methods that use network science, which encompasses all available algorithms in the literature until now. The proposed taxonomy divides the models accordingly to the space, network, or time series, in which the predictions are made. In figure 3.1, the diagram that represents the conceptual division is depicted, and, even though existing forecasting methods via

Network science only use visibility graphs mappings, the proposed high-level taxonomy does not specify that, since other mappings could be applied.

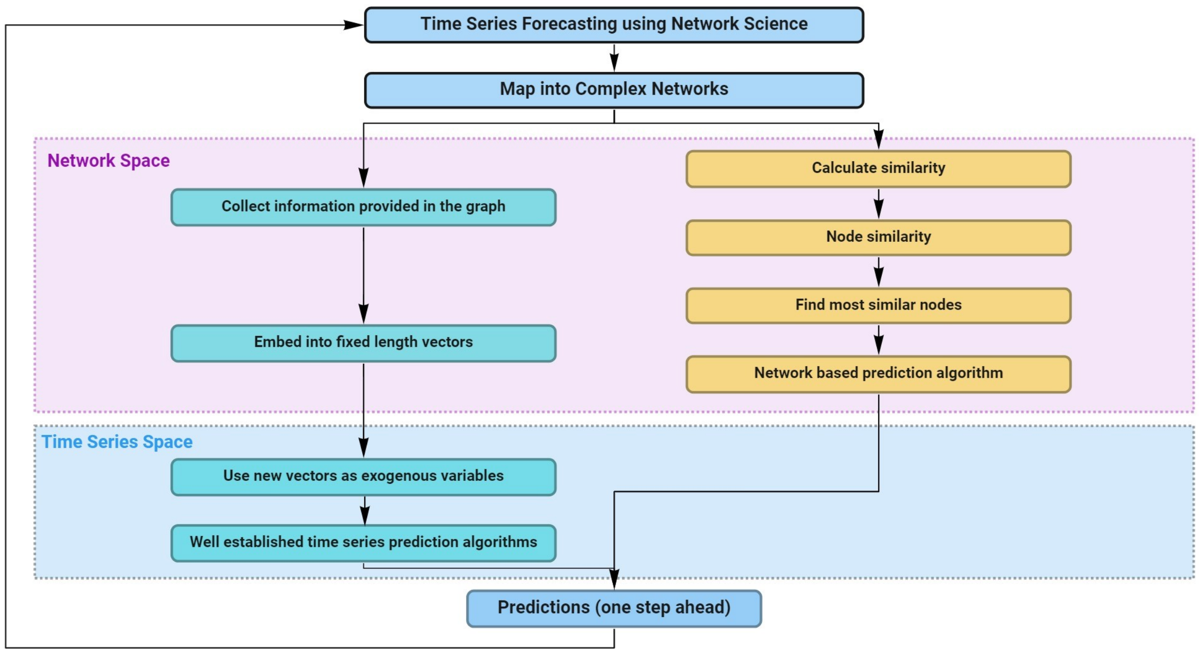

Time Series Forecasting using Network Science
Map into Complex Networks
Network Space
Collect information provided in the graph
Embed into fixed length vectors
Calculate similarity
Node similarity
Find most similar nodes
Network based prediction algorithm
Time Series Space
Use new vectors as exogenous variables
Well established time series prediction algorithms
Predictions (one step ahead)

Figure 3.1: Taxonomy of algorithms for time series forecasting that use complex networks.

The most noticeable affinity between the prediction algorithms developed in this area is the fact that the first step of all of them is to map the time series into a complex network. Then, most methods make predictions in the network space by making use of similarity between nodes and the geometrical properties of the graph; while in a recent study [13], the predictions are made by combining information obtained through the graph with modern time series forecasting approaches. The authors of the algorithms in question only implemented them with univariate time series and one-step ahead forecasts. However, once a prediction is made, all methods in the literature map the new forecast into the network space and repeat the process if the goal is to predict more than one-step ahead.

A more comprehensive description of the groups presented in the proposed taxonomy is given in the following sections.

3.1 Forecasting in the Network Space

For the first time in the literature, in 2017, Zhang and co-authors [35] used complex networks for time series forecasting, which did not only provide reasonable forecasts but also incited other authors to develop similar methods that follow the same principles.

Zhang and co-authors´ algorithm consists of two phases:

1. Firstly the time series is mapped into a complex network using a VG; then, the similarity between the last known node, N, and all previous $(N-1)$ nodes, is calculated based on a local random walk; finally, the node M with maximum similarity to node N is selected to make an initial one-step ahead prediction defined as
$$\hat{y}_{N+1} = \frac{y_N - y_M}{N - M} + y_N \tag{3.1}$$
where y_M is the value of the observation of the node M and t_M is the time of that observation.

2. The second phase generates the final prediction by making a weighted sum of the initial forecast and the value of the last known node, in which the weights of each value are calculated based on fuzzy rules.

The first stage of Zhang's method was the stepping stone for all algorithms encompassed in this branch of the taxonomy (figure 3.1). Most of them follow the same sequence of steps and take advantage of the geometric properties of the VG by calculating the slope between nodes to generate predictions.

Mao and co-authors [24] implemented the same steps in their algorithm, but replaced the similarity measure with a novel one that led to a slightly improved performance. Besides that, those authors also made a weighted sum between the result of the slope equation (3.1) and the value of the last known node, though, in this case, the weights were calculated based on the distance between the last known node and the most similar node used to calculate the slope.

Liu and co-authors [23] developed an algorithm that instead of selecting the most similar node, the prediction is made by doing a weighted sum of all slopes (3.1) between the K most similar node and the last known node, where the weights are the relative similarities of that set
$$y_{t+1} = \sum_{r=1}^{k} \frac{V_r}{\sum_{j=1}^{k} V_j} \left(\frac{y_t - y_r}{t - t_r} N y_t \right) \tag{3.2}$$
and where $V_x = V_{Nx}^t$ represents the stationary distribution of the probability transfer matrix, a measure of similarity obtained by customization of the Markov chain.

In [36], Zhao and co-authors propose a slightly different approach. In this method, the final forecast is given by making a weighted sum of the slopes between the last known node and all nodes connected/visible to it,
$$y_{t+1} = \sum_{i=t_f}^{t_l} \frac{w_i}{2} \left(\frac{y_t - y_r}{t - t_r} + 2y_t \right) \tag{3.3}$$
where w_i is a measure of similarity that takes into consideration the degree of each node and the temporal distance between nodes, defined as
$$w_i = \frac{\frac{d(t_i^*)}{t - t_i}}{\sum_{j=t_f}^{t_l} \frac{d(t_j^*)}{t - t_j}} \tag{3.4}$$

where $d(t_i^*)$ is the degree of node t_i^*, t_f is the time of the first node visible to the node of time t, and t_l is the time of the last node having visibility relation with that node.

3.2 Forecasting with Graph Embeddings

Up until the method developed by Huang *et al.* [13] was released, all prediction algorithms that use complex networks made predictions in the network space and followed similar steps to the ones explained in the previous section. Huang's algorithm takes advantage of the properties of the VG distinctly from the methods that belong to the other branch of the taxonomy. In this case, the focus is not to use the complex network to make link prediction. Instead, the goal of mapping the time series with a VG is to collect information from the data, which, otherwise, would be hugely hard to collect. Then, this information can be used by time series forecasting methods to generate forecasts that have a better description of the data.

Huang's method introduces the concept of **NV-encoding**, which extracts the local motif information of the whole series by using a moving-window strategy. This information is determined by the degree sequence of the VG (the degree of the corresponding node of the time point in the complex network) with some extra conditions. The natural visibility encoding of the observation y_i of the time series is written as

$$nv_{y_i} = \sum_{\substack{j=t_f \\ j\neq i}}^{j=t_l} \delta_{i,j}, \;\; \delta_{i,j} = \begin{cases} 1 & \text{if } y_i \geq y_j \wedge \;\; A_{i,j} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

As stated by Huang et al., "The global motif information was already embedded in the time series itself. Thus, conducting NV-encoding on the whole time series would lack necessity.". As such, a moving-window strategy with windows of arbitrary size, $size_w$, was developed. The NV-enconding is now calculated iteratively, from $i = 0$ to $i = N$, where each network originated from the NVG of the time series is formed by the observations $y_i, \ldots, y_{i+size_w}$.

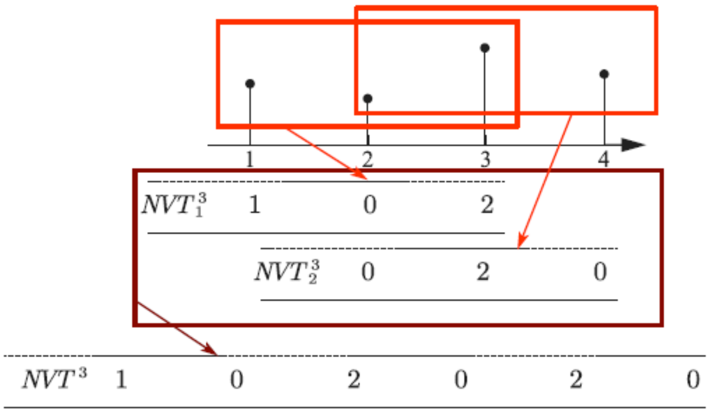


Figure 3.2: An example of the moving-window encoding strategy for the NV-encoding. *Extracted from* [13].

Figure 3.2 is an example of the application of the moving-window encoding with a time series of size 4 and windows of size 3. The final result is a vector composed of the encodings of each window, which leads to a vector of size $(N - size_w + 1) \times size_w$. Since $N = 4$ and $size_w = 3$, the length of the NV-encoded series is 6.

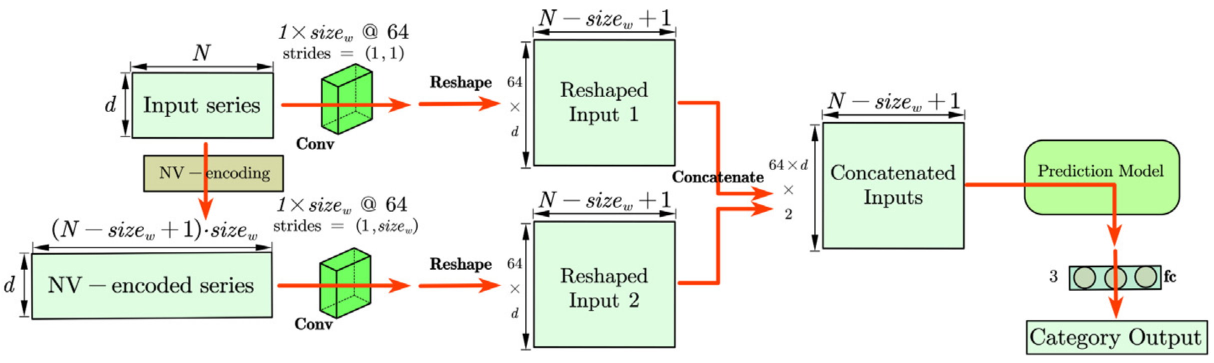


Figure 3.3: The architecture of Huang et al. framework. *Extracted from* [13]

This difference between the sizes of the input time series and the outcome of the NV-encoding leads to the necessity of a pre-processing step so that both series can be used with the same size. Huang et al. opted to use a CNN to reshape both series and combine them to generate predictions, something that can be done with any prediction model, as observed in Figure 3.3.

The NV-encoding can be seen as a graph embedding, in which the entire graph is encoded into a single vector, the NV-encoded series. This concept can be extended to other properties collected from the network, graph embedding methods, or even by generating a vector for each node or edge instead of one vector for the entire graph.

Chapter 4

Topological Characterization of Observations

One of the basic steps in a forecasting task is doing an exploratory analysis of the data [16]. Usually, this is done by finding consistent patterns like trends, seasonality, outliers, and changes in the variance of the time series. Tools like lag and seasonal plots, decomposition methods, and hypothesis tests are widely used to infer these characteristics.

As stated in Chapter 3, topological features of VGs can be used to characterize the time series while capturing several properties of the data, something that is explored by the algorithms described in Sections 3.1 and 3.2. However, these methods are only focused on the topological features used in the algorithms (e.g.: degree sequence by Huang's method [13]). This Chapter describes our implementation of a window strategy (Figure 3.2), which allows collecting both local and global topological features of the subgraphs formed in each window. Furthermore, an analysis of centrality measures collected with the window encoding for different types of VGs is presented.

4.1 Window Encoding

The characterization of time series through global topological features in its networks has already been proven to be a great tool for several analyses, like clustering [29]. Though, using the topology of the graphs for forecasting requires collecting such measures for each observation while considering only its past. This condition already implies that each metric is going to be calculated in a subgraph mapped from time points before the one being studied:

$$SG(v_i) = G[v_0, \dots, v_i]. \tag{4.1}$$

Moreover, these variables are only valuable if they are comparable between them. In this sense, the size of the subgraphs in which the features are computed should be the same. For instance, the maximum possible degree of the node v_i in a subgraph that only contains nodes that precede it, is smaller than the maximum degree of a node V_j in the same condition, where $j > i$. This is because the subgraph to which v_j belongs to has more $j - i$ nodes than the network ending in v_i.

The conditions described above lead us to define our own window encoding, in which each observation, i, of the time series is described by a topological feature of the subgraph of size m ending at i:

$$SG_m(v_i) = G[v_{i-(m+1)}, \ldots, v_i]. \tag{4.2}$$

In order to have a complete and flexible way of characterizing observations, we defined three encoding strategies:

Global Encoding : Features are computed for the subgraph as a whole

$$TF_m(v_i) = TF(SG_m(v_i)) \tag{4.3}$$

where TF is any global topological feature.

Local Encoding : Local measure, tf, for a single vertex in the subgraph (4.2) it

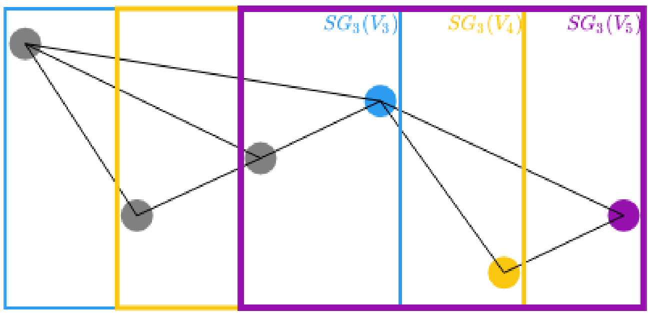
$$tf_m(v_i, j) = tf(SG_m(v_i))_j \tag{4.4}$$

where $j \in [i - (m+1), i]$.

Vector Encoding : Local topological feature for all nodes in the network

$$\overrightarrow{tf}_m(v_i) = [tf(v_i, i-(m+1)), \ldots, tf(v_i, i)]. \tag{4.5}$$

This way, it is possible to capture the global information of a subgraph of size m ending at each observation, i; the local measures of an interesting node in the network; or even the topological pattern which precedes i in m time points. An example of the application of the window encoding in a NVG with 5 nodes, where the features collected are the average degree of each subgraph, the degree sequence and the degree of the last node for each observation, is presented in Table 4.1.



Node (i)	$\bar{k}_3(v_i)$	$\overrightarrow{k}_3(v_i)$	$k_3(v_i, i)$
0	-	-	-
1	-	-	-
2	-	-	-
3	2.5	$[3, 2, 3, 2]$	2
4	1.5	$[1, 2, 2, 1]$	1
5	2	$[1, 3, 2, 2]$	1

Table 4.1: Window degree encoding of size 3 on a NVG with 6 nodes

Since no implementation is found online for this sort of the explained procedure, as mentioned in chapter 1 two modules of the *networktsf* library were developed to solve this issue:

Mappings : Visibility mappings implemented with the divide & conquer algorithm proposed in Lan et al. [21]. The module allows computing 15 different types of VGs:

- NVG;
- DNVG, where the edges in the graph follow the natural **direction** of **time**, or of the **series values**, the node with the highest value between the two connected nodes;
- WNVG, where the **weight** of the edges can be the **angle** between the two observations, the **euclidean** or **vertical** distance between them (or the inverse);
- DWNVG, any combination of the DNVG with the WNVG.

Network Features : Encodings of observations through the topological features of the network generated by a mapping method in the module *Mappings*:

- Window Encoding 4.1 which accepts any *networkx* [11] algorithm to compute global or local measures;
- Pipeline that maps a time series into a complex network with one of the methods in the package, and then computes as many window encodings as wished for all observations.
- ...

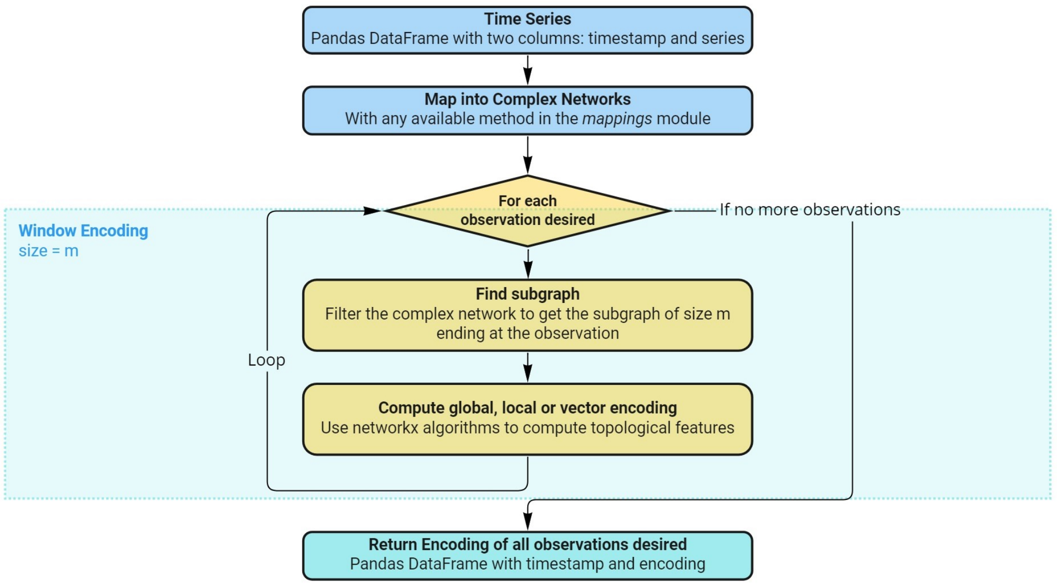

Time Series
Pandas DataFrame with two columns: timestamp and series
Map into Complex Networks
With any available method in the *mappings* module
Window Encoding
size = m
For each observation desired
If no more observations
Find subgraph
Filter the complex network to get the subgraph of size m ending at the observation
Loop
Compute global, local or vector encoding
Use networkx algorithms to compute topological features
Return Encoding of all observations desired
Pandas DataFrame with timestamp and encoding

Figure 4.1: Schematic diagram for the window encoding

.

Figure 4.1 is a representation of our window encoding strategy implementation. This strategy allows computing any topological feature available in *networkx* for each observation, while considering only the subgraph that contains the past m time points before each one of the observation. The process is dependent on other classes in the library, like the *mappings*, as well as three widely known packages, *numpy* [12], *pandas* [33] and *networkx*. Besides that, the mapping methods have a plotting function that produces a visualization of the graph by using *plotly* [17] to reproduce interactive plots.

4.2 Empirical Study

The conditions applied to the window encoding allow the calculation of topological features that can be used for forecasting. However, using this strategy implies choosing the size of the window, something that needs to be done wisely, as it should be based on the context of the series, the measures being collected, and the mapping method selected. On top of that, topological features may represent different patterns, depending on the mapping and the series under analysis. To address this question, we studied the correlation between the encodings and four time series patterns: seasonality, $\hat{S}_t$; and trend $\hat{T}_t$; moving average of order 3, $3-MA$; first-order difference, ∇^1 (subsection 2.1.2).

This study is divided into three different stages: selecting the size, m, of the window encoding; choosing the best VG mapping for each topological feature; and comparing the features between themselves. In order to do such an analysis, we computed the average correlation between the topological features and the patterns of 35 time series:

1. **FJ Glucose Values** Self-collected 15-minutely glucose values from a device used by people with type-1 diabetes.
2. **Monthly Milk production by cow** [5]
3. **Monthly Air Passengers in the USA** [3]
4. **Quarterly beer production in Australia** (Australian Bureau of Statistics. Cat. 8301.0.55.001.)
5. **Total monthly wine sales in Australia** [14]
6. **Quarterly production of woollen yarn in Australia** [14]
7. **29 monthly or quarterly** series from the **M2 Dataset**

4.2.1 Window Size Analysis

Selecting the size of the window to perform the encoding is of uttermost importance, as topological features vary with the length of the subgraph they are computed on, and the time series patterns may be impossible to identify in smaller samples of the data. With this in mind, the degree and the harmonic centrality of the last node in each subgraph, $k_m(i)$, the betweenness centrality of the observation in the middle of the window, $bc_m(i - \frac{m}{2})$, as well as the average shortest path length of each subgraph, $D(i)$, are computed for the NVGs mapped from each series. Then we calculated the average correlation between these measures and the time series patterns for window sizes ranging from a quarter to two times the period of the series, $m \in [P/4, 2P]$. The results of the study, for each topological feature computed, are presented in the tables below.

m/P	$\overline{\rho}(k_m(i), \text{3-MA})$	$\overline{\rho}(k_m(i), \nabla^1)$	$\overline{\rho}(k_m(i), \hat{S})$	$\overline{\rho}(k_m(i), \hat{T})$
0.25	-0.048	0.671	0.198	0.017
0.33	-0.037	0.690	0.206	0.006
0.42	-0.034	0.702	0.230	-0.009
0.50	-0.012	0.645	0.230	-0.009
0.58	0.012	0.721	0.276	-0.009
0.67	0.033	0.722	0.293	-0.007
0.75	0.044	0.674	0.327	-0.004
0.83	0.070	0.718	0.345	-0.007
0.92	0.097	0.716	0.367	0.002
1.00	0.082	0.682	0.412	-0.009
1.08	0.100	0.715	0.412	-0.019
1.17	0.096	0.715	0.407	-0.027
1.25	0.086	0.685	0.406	-0.012
1.33	0.106	0.713	0.394	-0.021
1.42	0.113	0.717	0.397	-0.014
1.50	0.093	0.690	0.406	-0.007
1.58	0.120	0.714	0.395	-0.009
1.67	0.117	0.712	0.397	-0.019
1.75	0.097	0.690	0.405	-0.012
1.83	0.131	0.710	0.390	0.002
1.92	0.131	0.707	0.391	-0.006
2.00	0.114	0.687	0.416	-0.002

Table 4.2: Mean correlation between the degree, $k_m(i)$, and time series patterns

Based on Table 4.2 it is possible to realize that $k_m(i)$ is highly correlated with the first order difference of the series, given that the degree of a node in a NVG is bigger for local maximums and smaller for minimums, which is translated into a variable susceptible to local variations. Moreover, this topological feature can also capture seasonality, something that can be explained by the fact that the maximum values of each period of the series are mapped into the nodes with more visibility to other nodes before them.

It is worth noticing that the first order difference is always captured, regardless of the window. The correlation with seasonality is better detected for $m > \frac{P}{2}$, though, the maximum value is obtained for windows of the same size as the period, stabilizing for bigger lengths. Considering this, it seems that selecting $m = P$ is a safe choice to represent time series patterns such as the ones seasonality and local variations.

m/P	$\overline{\rho}(D_m(i), \text{3-MA})$	$\overline{\rho}(D_m(i), \nabla^1)$	$\overline{\rho}(D_m(i), \hat{S})$	$\overline{\rho}(D_m(i), \hat{T})$
0.25	0.037	-0.527	-0.167	0.001
0.33	0.007	-0.356	-0.059	-0.000
0.42	-0.022	-0.389	-0.140	-0.001
0.50	-0.054	-0.374	-0.193	-0.004
0.58	-0.125	-0.293	-0.248	0.003
0.67	-0.182	-0.285	-0.272	-0.004
0.75	-0.181	-0.316	-0.368	-0.017
0.83	-0.251	-0.193	-0.354	-0.019
0.92	-0.297	-0.188	-0.379	-0.028
1.00	-0.249	-0.266	-0.522	-0.032
1.08	-0.283	-0.039	-0.380	-0.028
1.17	-0.215	-0.038	-0.289	-0.019
1.25	-0.137	-0.107	-0.233	-0.026
1.33	-0.095	-0.008	-0.095	-0.012
1.42	-0.048	-0.048	-0.045	-0.014
1.50	-0.038	-0.128	-0.013	-0.032
1.58	-0.031	-0.086	-0.006	-0.028
1.67	-0.060	-0.104	-0.025	-0.034
1.75	-0.106	-0.191	-0.123	-0.052
1.83	-0.148	-0.116	-0.149	-0.044
1.92	-0.204	-0.118	-0.215	-0.042
2.00	-0.206	-0.194	-0.384	-0.051

Table 4.3: Mean correlation between the avg. shortest path length, $D_m(i)$, and time series patterns

The average shortest path length of a graph is always going to be smaller for networks with nodes that are linked to many other nodes, and bigger when the number of edges is smaller. In the context of the global encoding applied to VGs, the subgraphs that end in a maximum of the series have a smaller value of D since the visibility of that node to the past m observations is big, and, consequently, most shorter paths pass trough that vertex to reach others in fewer steps. Therefore, we expected this measure to be a good representation of the inverse of the seasonality, as it is corroborated by the correlation values in table 4.3. Besides the seasonality, the feature is also negatively correlated to the first-order difference and the moving average of order 3.

In this case, small changes in the window size lead to many different patterns being captured. For instance, if the window size is equal to the period, the feature is highly correlated to the seasonality, first-order difference and $3 - MA$; yet, if the length of the subgraph is half the period, the correlation with any of the patterns being studied, expcept for ∇^1, is much smaller. In fact, the window lengths that are worth exploring with this measure range in the interval $m \in [P/2, P]$, as well as $2P$, and, once again, selecting a size equal to the period is a safe option to capture multiple patterns with this global topological feature.

m/P	$\overline{\rho}(hc_m(i), \text{3-MA})$	$\overline{\rho}(hc_m(i), \nabla^1)$	$\overline{\rho}(hc_m(i), \hat{S})$	$\overline{\rho}(hc_m(i), \hat{T})$
0.25	-0.052	0.672	0.197	0.012
0.33	-0.041	0.687	0.199	0.003
0.42	-0.033	0.702	0.227	-0.007
0.50	-0.007	0.645	0.237	-0.005
0.58	0.020	0.717	0.289	-0.009
0.67	0.048	0.716	0.312	-0.009
0.75	0.063	0.669	0.353	-0.003
0.83	0.098	0.706	0.380	-0.010
0.92	0.134	0.702	0.408	-0.001
1.00	0.116	0.673	0.464	-0.008
1.08	0.148	0.686	0.462	-0.021
1.17	0.141	0.685	0.452	-0.032
1.25	0.119	0.661	0.447	-0.015
1.33	0.139	0.680	0.426	-0.029
1.42	0.143	0.683	0.419	-0.020
1.50	0.119	0.662	0.421	-0.006
1.58	0.147	0.677	0.409	-0.014
1.67	0.146	0.674	0.411	-0.023
1.75	0.126	0.660	0.424	-0.011
1.83	0.165	0.667	0.415	-0.006
1.92	0.174	0.665	0.426	-0.015
2.00	0.151	0.657	0.464	-0.006

Table 4.4: Mean correlation between the harmonic centrality, $hc_m(i)$, and time series patterns

Another way of analyzing the influence of a node in the network is by inferring how close it is to the center of the graph. Translating this into VGs results in attributing more relevance to observations near the maximums of the series. This way, harmonic centrality is a measure closely related to the degree but shifted one lag into the future. Actually, as presented in table 4.4, the values of correlations with the time series patterns are very similar to the ones obtained for the degree. Nonetheless, this feature can capture the seasonality and the local variations of the series, and since it is another way of studying the centrality of nodes, it is a feature worth taking into consideration, as well as the degree.

The first-order difference is highly correlated with the feature for both small and big windows; yet, the seasonality is only captured for window sizes above half the period. As mentioned, the correlations are very close to the ones obtained in the degree, the maximum correlation with $\hat{S}_t$ is obtained for $m = P$, but, in this case, the values obtained for the seasonality cease to increase for lengths bigger than the period until $m = 2P$. Selecting a window with the same length as the period of the time series is still the best choice to take the best out of this feature.

m/P	$\overline{\rho}(bc_m(i-\frac{m}{2}), \text{3-MA})$	$\overline{\rho}(bc_m(i-\frac{m}{2}), \nabla^1)$	$\overline{\rho}(bc_m(i-\frac{m}{2}), \hat{S})$	$\overline{\rho}(bc_m(i-\frac{m}{2}), \hat{T})$
0.25	-0.028	-0.019	-0.036	0.004
0.33	-0.048	-0.008	-0.033	-0.013
0.42	-0.024	0.098	0.100	-0.010
0.50	-0.015	-0.023	0.072	0.007
0.58	-0.036	-0.133	-0.180	0.012
0.67	-0.055	-0.140	-0.190	-0.001
0.75	-0.087	-0.059	-0.153	-0.003
0.83	-0.098	-0.016	-0.123	0.008
0.92	-0.093	0.002	-0.097	0.010
1.00	-0.085	-0.051	-0.155	0.011
1.08	-0.084	0.015	-0.088	-0.001
1.17	-0.075	0.014	-0.083	0.002
1.25	-0.034	0.006	-0.117	-0.006
1.33	-0.033	0.032	-0.100	-0.004
1.42	0.014	0.054	0.023	0.008
1.50	0.005	0.027	0.008	-0.003
1.58	0.057	0.033	0.103	-0.010
1.67	0.058	0.035	0.101	-0.007
1.75	0.132	0.053	0.120	-0.008
1.83	0.178	0.056	0.073	0.001
1.92	0.134	0.205	0.481	0.000
2.00	0.097	0.162	0.487	-0.010

Table 4.5: Mean correlation between the betweenness centrality, $bc_m(i-\frac{m}{2})$, and time series patterns

Measuring how many shortest paths pass through a node is extremely useful to understand how important a node is in terms of graph connectivity. The betweenness of a node is precisely that, but, when considering the conditions of the window encoding, the first and last observations of each subgraph are not between anything. Table 4.5 shows the correlation between the betweenness centrality of the node in the middle of each window and the patterns of the series. Despite having selected this node as an example, all vertexes except the first and the last of the window could be selected.

Such a choice makes this feature the one that varies the most with the size of the window. Thus, for a window where $m \approx P$, that ends at the maximum value of each period, it is probable that the observation in the middle of it is near the minimum of the period, which means that fewer shortest paths pass through that node. Consequently, the value of betweenness centrality is smaller, and, therefore, negatively correlated to the seasonality. On the other hand, for a window where $m \approx 2P$, ending at the maximum value of each period results in a high correlation with the seasonality.

Selecting a window size near two times the period, leads to a small correlation with all patterns except the trend, while subgraphs of length equal to the period result in negative correlations. These patterns are way better captured in both of these cases than with other sizes. Hence, for $bc_m(i - \frac{m}{2})$, the lengths that seem to result best are $m = P \vee m = 2P$.

The study made for each topological feature allows an understanding of which window sizes are appropriate to capture interesting patterns in the series. Window sizes that are smaller than half the period lead to low correlations with the patterns. Based on the conclusions written for each feature, we will use $m = P$ in the remainder of this study. Yet, the size of the window should be studied for each context, at least in the range:

$$m \in [P/2, 2P]. \tag{4.6}$$

4.2.2 Mapping Methods Analysis

Since all topological features computed in subsection 4.2.1 were obtained for NVGs, in this part of the investigation we study the influence of attributing weight and/or a direction to the edges in the network by inspecting how it changes the correlations between the measures and the time series patterns. Thus, in this subsection, we present the correlation between the same encodings and the patterns studied for all mapping methods possible to compute with the *networktsf* library.

The mappings names presented in the tables are shortened in order to fit the page, as such, the weights are represented by: *a*, the angle between observations; *ed* and *ie* the euclidean distance and the inverse; *d* and *id* the vertical distance and the inverse. The direction can follow the natural sequence of time, *t*, or the highest value between the two connected nodes, *s*.

Mapping	$\overline{\rho}(k_P(i), \text{3-MA})$	$\overline{\rho}(k_P(i), \nabla^1)$	$\overline{\rho}(k_P(i), \hat{S})$	$\overline{\rho}(k_P(i), \hat{T})$
NVG	0.082	0.682	0.412	-0.009
WNVG (W: a)	0.215	0.644	0.543	0.047
WNVG (W: ed)	0.226	0.553	0.321	0.162
WNVG (W: ie)	-0.118	0.114	0.066	-0.136
WNVG (W: id)	-0.108	0.135	0.094	-0.133
WNVG (W: d)	0.239	0.562	0.320	0.175

Table 4.6: Mean correlation between the degree, $k_P(i)$, and time series patterns for different mappings

From table 4.6, it is possible to conclude that calculating $k_P(i)$ in distinct VGs leads to highly different results in terms of the patterns captured by the feature. The local variations of the time series are better captured in the NVG, which means the degree of the nodes in this mapping is more susceptible to the noise of the data. All weighted networks, except the ones with weights that represent the inverse of the distance, result in high correlations between $k_P(i)$ and the first order difference, as well as with the seasonality, and a small correlation with the moving average of order 3. Besides that, the trend is somewhat captured when the weights correspond to distances and the rest of the patterns are also represented by this feature. This is particularly interesting since the measure can be used to analyze different nuances of the time series simultaneously.

Mapping	$\overline{\rho}(k_P^{in}(i), \text{3-MA})$	$\overline{\rho}(k_P^{in}(i), \nabla^1)$	$\overline{\rho}(k_P^{in}(i), \hat{S})$	$\overline{\rho}(k_P^{in}(i), \hat{T})$
DNVG (D: s)	0.151	0.678	0.511	0.009
DNVG (D: t)	0.082	0.682	0.412	-0.009
DWNVG (W: a; D: s)	0.189	0.715	0.511	0.047
DWNVG (W: a; D: t)	0.215	0.644	0.543	0.047
DWNVG (W: ed; D: s)	0.262	0.634	0.476	0.123
DWNVG (W: ed; D: t)	0.226	0.553	0.321	0.162
DWNVG (W: ie; D: s)	-0.074	0.195	0.128	-0.112
DWNVG (W: ie; D: t)	-0.118	0.114	0.066	-0.136
DWNVG (W: id; D: s)	-0.078	0.146	0.123	-0.117
DWNVG (W: id; D: t)	-0.108	0.135	0.094	-0.133
DWNVG (W: d; D: s)	0.274	0.642	0.476	0.135
DWNVG (W: d; D: t)	0.239	0.562	0.320	0.175

(a) Mean correlation between $k_P^{in}(i)$ and time series patterns for different mappings

Mapping	$\overline{\rho}(k_P^{out}(i), \text{3-MA})$	$\overline{\rho}(k_P^{out}(i), \nabla^1)$	$\overline{\rho}(k_P^{out}(i), \hat{S})$	$\overline{\rho}(k_P^{out}(i), \hat{T})$
DNVG (D: s)	-0.185	-0.162	-0.404	-0.020
DWNVG (W: a; D: s)	0.170	0.183	0.418	-0.003
DWNVG (W: ed; D: s)	-0.096	-0.312	-0.490	0.122
DWNVG (W: ie; D: s)	-0.078	0.012	-0.030	-0.065
DWNVG (W: id; D: s)	-0.068	0.062	-0.011	-0.062
DWNVG (W: d; D: s)	-0.085	-0.318	-0.490	0.140

(b) Mean correlation between $k_P^{out}(i)$ and time series patterns for different mappings

Table 4.7: In and out degree correlation with time series patterns for directed VGs

Calculating the degree for directed networks results in collecting two new features, in-degree, and out-degree, therefore the tables in 4.7 show the correlation between these metrics and the patterns of the series. It must be pointed out that $k_P^{out}(i)$ is zero for VGs with edges directed by time, which is obvious since the last node of each subgraph corresponds to the most recent one for each window. Also, computing $k_P^{in}(i)$ in networks with links following the direction of time is the same as doing it with an NVG because i is only linked to past observations in the windows being studied.

Combining the in and out degree seems to lead to a better description of the series when compared to only using the degree, something that can be concluded by the results in the three tables above. Mappings with links that follow the time direction are not worth exploring in the context of this feature because of the reasons mentioned in the previous paragraph, however, the correlation of the degrees and the patterns of the series for networks with edges that point to the highest value between the two connected nodes are very promising. For these mappings, $k_P^{in}(i)$ captures the seasonality and local variations, as well as the $3 - MA$ if the weight of the edges corresponds to the vertical distance of the series; while $k_P^{out}(i)$ is negatively correlated to the seasonality and the first order difference, and positively correlated to the trend.

In conclusion, calculating both of these features for a *DWNVG(W: y distance; D: series)* allows distinguishing between minimums and maximums of the series more easily, as well as capturing other relevant patterns of the data. On the other hand, using the angle between observations as the weight of the links makes this encoding not useful to represent the trend, and the combination of both features is redundant since the in-degree gives a better picture of the patterns that are also captured by the out-degree.

Even though *WNVGs (W: angle)* can be used to compute the degree, most topological features algorithms can not deal with negative weights. With this mind tables 4.8 to 4.10 only present correlations for NVGs and WNVGs, where the weight represents distances.

Mapping	$\overline{\rho}(D_P(i), 3\text{-MA})$	$\overline{\rho}(D_P(i), \nabla^1)$	$\overline{\rho}(D_P(i), \hat{S})$	$\overline{\rho}(D_P(i), \hat{T})$
DNVG (D: s)	0.117	0.132	0.441	-0.073
DNVG (D: t)	-0.234	-0.249	-0.499	-0.031
DWNVG (W: ed; D: s)	0.507	0.178	0.337	0.410
DWNVG (W: ed; D: t)	0.217	-0.026	-0.376	0.452
DWNVG (W: ie; D: s)	-0.148	-0.023	0.131	-0.211
DWNVG (W: ie; D: t)	-0.203	-0.115	-0.033	-0.224
DWNVG (W: id; D: s)	-0.149	-0.024	0.145	-0.219
DWNVG (W: id; D: t)	-0.225	-0.097	-0.047	-0.245
DWNVG (W: d; D: s)	0.512	0.178	0.337	0.415
DWNVG (W: d; D: t)	0.221	-0.026	-0.376	0.456
NVG	-0.249	-0.266	-0.522	-0.032
WNVG (W: ed)	0.217	-0.026	-0.376	0.452
WNVG (W: ie)	-0.345	-0.164	-0.183	-0.310
WNVG (W: id)	-0.337	-0.141	-0.178	-0.302
WNVG (W: d)	0.221	-0.026	-0.376	0.456

Table 4.8: Mean correlation between the avg. shortest path length, $D_P(i)$, and time series patterns for different mappings

Changing the direction and weight of the network results in very different average shortest path lengths. The fact that this metric is negatively correlated to the seasonality when computed for all undirected or time-directed VGs, and positively correlated to all other mappings is a good case in point. The influence of using the distance as the weight is very clear since the correlation with $3 - MA$ is significantly higher in such cases. Also, the trend is far better expressed in this situation.

With this in mind, calculating $D_P(i)$ for a *WNVG (W: y distance)* and combining it with the in-degree and out-degree of a *DWNVG(W: y distance; D: series)* gives a richer description of the series.

Mapping	$\overline{\rho}(hc_P(i), \text{3-MA})$	$\overline{\rho}(hc_P(i), \nabla^1)$	$\overline{\rho}(hc_P(i), \hat{S})$	$\overline{\rho}(hc_P(i), \hat{T})$
DNVG (D: s)	0.162	0.651	0.549	-0.008
DNVG (D: t)	0.113	0.672	0.453	-0.006
DWNVG (W: ed; D: s)	-0.078	0.191	0.160	-0.128
DWNVG (W: ed; D: t)	-0.165	0.049	0.057	-0.192
DWNVG (W: ie; D: s)	0.285	0.644	0.506	0.129
DWNVG (W: ie; D: t)	0.275	0.388	0.185	0.264
DWNVG (W: id; D: s)	0.297	0.651	0.506	0.141
DWNVG (W: id; D: t)	0.289	0.397	0.185	0.278
DWNVG (W: d; D: s)	-0.082	0.155	0.154	-0.132
DWNVG (W: d; D: t)	-0.141	0.110	0.085	-0.175
NVG	0.116	0.673	0.464	-0.008
WNVG (W: ed)	-0.165	0.049	0.057	-0.192
WNVG (W: ie)	0.254	0.393	0.164	0.255
WNVG (W: id)	0.268	0.402	0.163	0.270
WNVG (W: d)	-0.141	0.110	0.085	-0.175

Table 4.9: Mean correlation between the harmonic centrality, $hc_P(i)$, and time series patterns for different mappings

Table 4.9 shows the same conclusion as the previous section on the comparison between the degree and the harmonic centrality. Nonetheless, computing the $hc_P(i)$ in WVGs where the weight is the distance, results in low correlations with all patterns. Then, this feature can be useful to capture other phenomenons of the series that are not represented by the other encodings above.

Seasonality is always negatively represented by the betweenness centrality, except for networks with edges that follow the positive direction of the values. This can be explained by the fact that, for these types of VGs, none of the nodes work as a connector between components, i.e., local maximums have many in-links but few out-links, and the inverse happens for minimums, which means that when accounting for this type of direction no node is between a lot of other vertexes. Table 4.10 shows that the patterns captured by $bc_P(i - \frac{m}{2})$ in a *WNVG (W: y distance)* lead to an encoding that is not able to capture any pattern, thus, with the same purpose as the harmonic centrality, this feature can be used to represent other non-classical characteristics of the series.

Mapping	$\overline{\rho}(bc_P, \text{3-MA})$	$\overline{\rho}(bc_P, \nabla^1)$	$\overline{\rho}(bc_P, \hat{S})$	$\overline{\rho}(bc_P, \hat{T})$
DNVG (D: s)	0.001	0.013	0.059	-0.023
DNVG (D: t)	-0.079	-0.026	-0.135	0.003
DWNVG (W: ed; D: s)	-0.002	0.027	0.065	-0.029
DWNVG (W: ed; D: t)	-0.080	-0.027	-0.135	0.002
DWNVG (W: ie; D: s)	-0.001	0.010	0.049	-0.023
DWNVG (W: ie; D: t)	-0.074	-0.029	-0.124	0.002
DWNVG (W: id; D: s)	-0.002	0.011	0.050	-0.025
DWNVG (W: id; D: t)	-0.077	-0.029	-0.123	-0.001
DWNVG (W: d; D: s)	0.048	0.070	0.152	-0.029
DWNVG (W: d; D: t)	-0.100	-0.032	-0.184	0.003
NVG	-0.085	-0.051	-0.155	0.011
WNVG (W: ed)	-0.080	-0.027	-0.135	0.002
WNVG (W: ie)	-0.080	-0.046	-0.154	0.023
WNVG (W: id)	-0.088	-0.047	-0.146	0.013
WNVG (W: d)	-0.084	-0.026	-0.150	-0.002

Table 4.10: Mean correlation between the betweenness centrality, $bc_P(i - \frac{m}{2})$, and time series patterns for different mappings

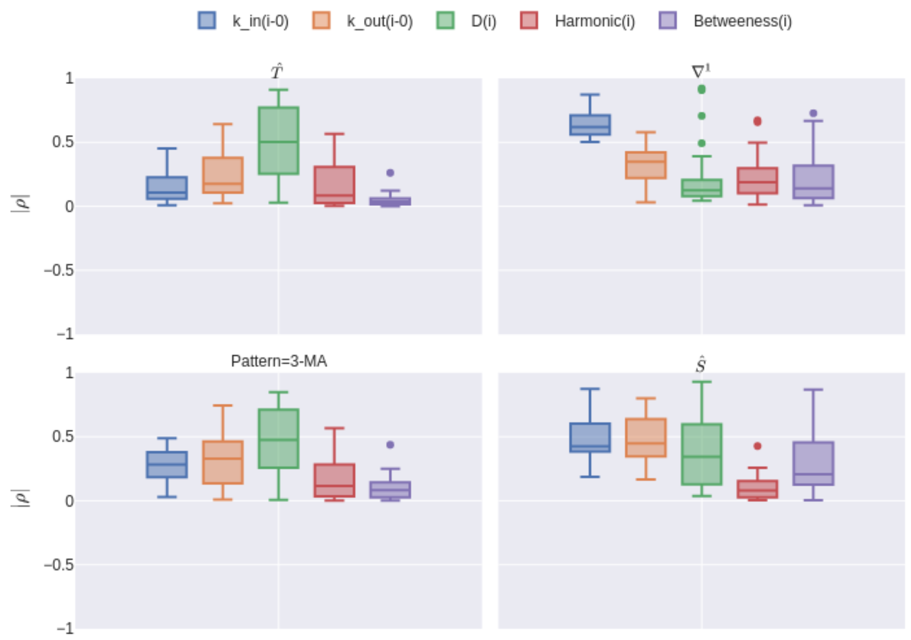
Not surprisingly, computing the encodings for different mappings led to distinct descriptions of the series' patterns. The features collected from WVGs where the weights correspond to the inverse of a distance or the angle, do not seem to be very relevant in the context of the time series components analyzed. Additionally, the conditions of the window encoding imply that using a DNVG where the edges follow the direction of time is redundant, since the way the features are calculated already takes time into consideration. All other methods to convert time series to complex networks produce encodings worth exploring, and the combination of the different features collected for similar graphs can originate a fuller representation of the time series.

4.2.3 Topological Features Comparison

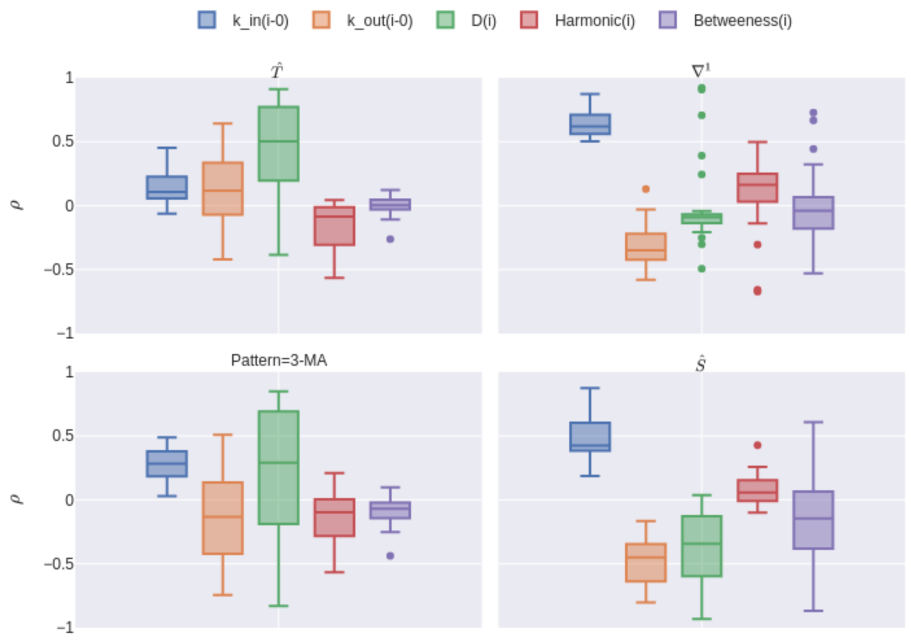
As mentioned in 4.2.2, the topological features studied complement each other regarding the patterns they capture. Even so, different measures can represent the same depending on the type of VG each one is computed on, making the combination of these encodings superfluous. One way to avoid this is to use the same weight in the WVGs each feature is calculated for.

This subsection compares the in and out-degree of a *DWNVG (D: series; W: y distance)*, with the other features obtained from a similar but undirected graph. The decision of selecting these mappings was based on the previous analysis, which demonstrates that computing the encodings for these networks originates differences between the patterns each feature expresses. In addition, in this section, it is shown an example of the measures computed for a time series.

Although the correlations with the time series characteristics were already studied in this chapter, the way it was done was solely based on its average for each metric. A more detailed view of the distribution of the correlations and their absolute values is displayed in the box plots of figure 4.2.



(a) Box-plot of absolute correlations between topological features and patterns



(b) Box-plot of correlations between topological features and patterns

Figure 4.2: Comparison between topological features

The distributions of the absolute correlations allow an understanding of which patterns are better represented by each one of the topological measures studied. With the help of figure 4.2a, it is possible to conclude that the trend is not expressed by the majority of the features. In fact, this is a consequence of setting the window encoding size to be equal to the period because the trend is not as notable between observations of the same subgraph. The exception is when the variance of the series is not constant; if this is the case, the average shortest path length can represent this trait of the data, something that can be seen by the absolute correlations with T and $3 - MA$. Figure 4.2b upholds this statement, while also adding more information, for instance, the fact that these topological features are usually positively correlated with the trend, but negatively with the third-order moving average.

Clearly, the seasonality is the better represented pattern by four out of five encodings. Such a statement is not surprising, as the observations separated by one period will likely originate nodes with similar topological properties. Therefore, seasonality is captured by the repetition of the same values for each feature. It is important to note that this trait of the series is either positively or negatively correlated with each measure, independently of the series.

One pattern that is mostly portrayed by the in-degree is the local variations of the series. As stated, being too correlated to the first-order difference may produce an encoding that resembles the noise of the time series. Nonetheless, having features positively correlated and others negatively correlated is helpful to distinguish between local minimums or maximums and further values. Thereby, analyzing the series with the in and out degree, the average shortest path length, and the betweenness centrality allow inferring which node corresponds to each situation explained above.

The harmonic centrality is the only encoding that is not very much correlated to any component of the series investigated. However, in the previous subsection, it was already mentioned that this is not necessarily damaging, considering the encoding may capture other characteristics of the data.

4.3 Example of Application

Figure 4.3 is an example of a visual representation of the complex network obtained by mapping the average monthly production of milk per cow series with an NVG. A window encoding with size $m = 12$ was used to calculate the in-degree of the last observation (size of nodes) and the average shortest path length of the subgraph (color of nodes) for each window. Plots for the other time series studied can be found in the appendix A (figures A.2 to A.5).

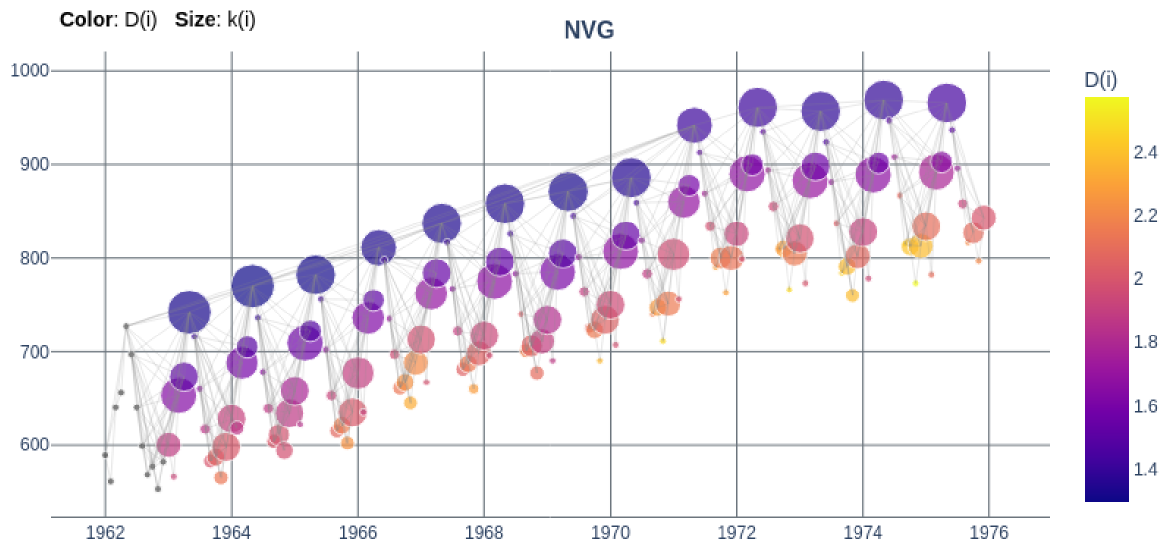
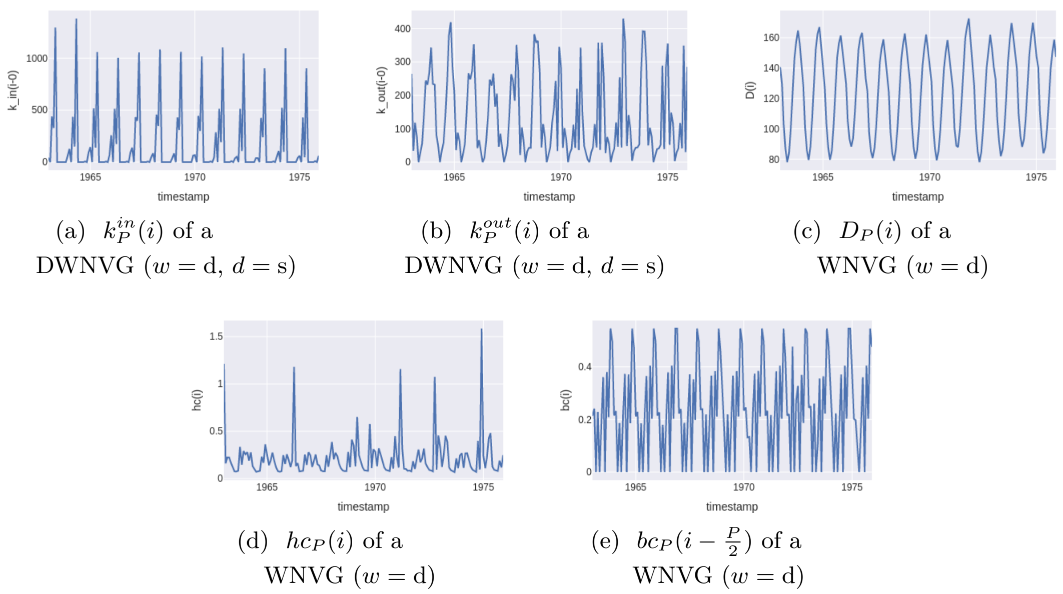


Figure 4.3: NVG of the monthly production of milk per cow

Just by doing a quick analysis of the image, it is clear that the topological features capture some important patterns: the average shortest path, $D_{12}(i)$, resembles the inverse of the seasonality; while the in-degree, $k_{12}^{in}(i)$, is bigger for time points that represent local maximums of the series, and smaller for observations that present a negative variation with the value of the previous time point.

A more detailed evaluation of how each encoding considered in this subsection varies for each observation is available in the plots below for the same time series as above. Images A.7 to A.10 show the same charts but for each one of the other series.



(a) $k_P^{in}(i)$ of a DWNVG (w = d, d = s)
(b) $k_P^{out}(i)$ of a DWNVG (w = d, d = s)
(c) $D_P(i)$ of a WNVG (w = d)
(d) $hc_P(i)$ of a WNVG (w = d)
(e) $bc_P(i - \frac{P}{2})$ of a WNVG (w = d)

Figure 4.4: Topological features encoded from monthly production of milk per cow series

In this example, it is quite evident that all five topological features are influenced by the seasonality of the series they were calculated for. Curiously, none of the encodings echos the trend. Yet, the maximums of the harmonic centrality (that can almost be considered outliers) match the changes in this pattern. Moreover, the lag of size P between the maximums of the in and out-degree represents the local variations of the series. Lastly, since there is no change in the variance of the series, the $3 - MA$ is only correlated with the trend, and, as such, the topological measures can not correlate to a none-existing component of the series.

4.4 Summary

In summary, when using WNVGs where the weight is the vertical distance between the values of the series, some conclusions were drawn. Firstly, the trend and the $3 - MA$ are expressed by the out-degree and the average shortest path length. Also, it was verified that all patterns except $hc_P(i)$ are able to describe the seasonality. Besides, a positive variation among consecutive observations originates a bigger in-degree of the later node, while a negative one does the same for the out-degree. Finally, harmonic centrality can be seen as another way of describing the centrality of each node. Hence, combining these topological measures computed with the window encoding can provide a good characterization of the observations.

Each one of the topological features explored can represent slightly different characteristics of the data, something that can be induced by the plots in the appendix A and the figures in 4.2b. It is tremendously useful to know this, for the reason that it shows the adaptability of encodings to the most relevant traits of the series. For instance, the air passenger series A.2 has increasing variance, something that is represented by the average shortest path length computed with the window encoding A.7c.

In the end, this investigation reveals that both local and global topologies of complex networks mapped from time series can provide a great tool to characterize each observation in the sequence. The description provided can be used to find similar data points and/or outliers, explore hidden nuances, or enhance the knowledge that forecasting models can pick up from the series.

With all of this in mind, one can wonder if these features can somehow be used to characterize entire past events in a way that can be compared to other time periods. One way to achieve this is by using a vector encoding 4.5 instead of a global or local one, because, this way, all nodes in each subgraph will be characterized similarly to the last one. A thorough examination of this solution is brought off in chapter 5.

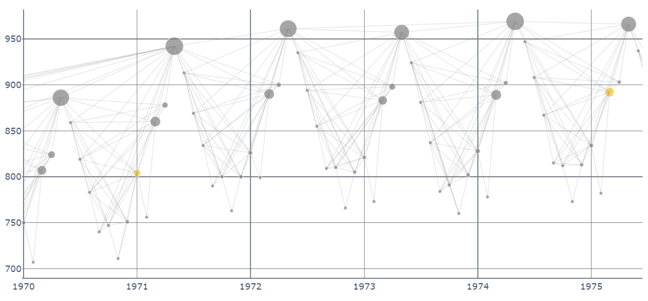
Chapter 5

Topological Similarity between Observations

Time Series forecasting is strongly supported by the analysis of the data, specially by studying the relationship between observations (subsection 2.1.3). The classical approach to do this is by computing the ACF, which presents the correlation between lagged values. One problem behind this methodology is that two similar observations but very distinct from the others in the series would not have much influence on the values of the autocorrelation function.

The empirical study in chapter 4 demonstrates another way of computing the correlation of time pairs by comparing the topological features calculated with the window encoding strategy for different types of VGs. If these measures are treated as a node embedding, meaning that each time point corresponds to a vector of local and global encodings, a pair of observations can be compared by finding the similarity between the vectors.

Even though this strategy seems to be promising, it may be hard to distinguish between local maximums and observation values bigger than their preceding. An example of this situation is displayed below. The plot shows a sample of the NVG obtained for the monthly production of milk per cow series. The two yellow nodes (01/03/1975 and 01/01/1971) are topologically similar, and the table presents the embedding for these two vertexes (each feature is normalized between zero and one).



	01/01/1971	01/03/1975
$k_{12}^{in}(i)$	0.21	0.31
$k_{12}^{out}(i)$	0.35	0.34
$D_{12}(i)$	0.60	0.34
$hc_{12}(i)$	0.11	0.10
$bc_{12}(i-6)$	0.39	0.36

Table 5.1: Two topological similar observations of the monthly production of milk per cow series

Following the methodology proposed above, the next step is computing the similarity between the nodes. Since this corresponds to comparing two vectors of single time points, one good metric to evaluate how alike they are is the cosine similarity [6]. Therefore, this function can be used to find the most similar previous node of any vertex:

$$S(\overrightarrow{y}(t), \overrightarrow{y}(t-n)) = cos(\theta) = \frac{\sum_i y_t(i) y_{t-n}(i)}{|\overrightarrow{y}(t)||\overrightarrow{y}(t-n)|}, \text{ where } n \in]0, t]. \tag{5.1}$$

In fact, this procedure discovers that the most similar time point to 01/03/1975 is 01/01/1971, with a similarity value of 95%. The problem with this result is that the first observation is a local maximum, while the latter one is not. While it is true that the two vectors are significantly alike, given that the seasonality of this series is so well-defined, the distance between all node embeddings is always small.

The present chapter, presents a solution to this predicament, which focuses on the description of the subgraph of each window to characterize each observation. Furthermore, an analysis of this approach through the application in real time series is made available.

5.1 Visibility Networks Subgraphs Similarity

Chapter 4 study was entirely based on four local encodings computed for one observation in each window and a global measure to describe each subgraph as a whole. Undoubtedly, all nodes of each network carry some information from the graph, specially the last one. Still, the topologies of a specific node do not allow characterizing a graph completely and the fact that, for seasonal series, windows are almost identical between themselves: their topological structures expressed by global measures are quite similar. Besides, both local and global encodings do not take advantage of the time sequence inherent to VGs.

In the summary of the previous chapter, it is possible to find a suggestion of a way to characterize past events by computing the vector encoding of each local feature, instead of doing it for just one node in each window. The advantages of describing observations by portraying the subgraph of size m ending in each of them are:

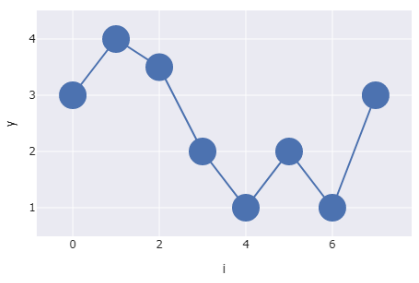
Topological patterns : Rather than attributing the entire importance of each feature to a single time point, topological measures of all nodes in the window are in their vector;

Resistant to extreme values : Values on extremes of the range of each feature have a big influence on the average of the local encodings of those observations, while the vector encoding corresponds only to one entry;

Time direction : Vector order is governed by the natural order of time, allowing to compare the vectors entry by entry, while considering the sequence of the time series;

Time distance : The combination of the three perks already described enables the consideration of the time distance between extreme values, and repetition of the same measure.

Figure 5.1 is an example of the vector encoding for each local topological measure analysed in subsection 4.2.3, computed for the time series on the left.



i	$\overrightarrow{k}^{in}_3(i)$	$\overrightarrow{k}^{out}_3(i)$	$D_m(i)$	$\overrightarrow{hc}_3(i)$	$\overrightarrow{bc}_3(i)$
3	[0, 2, 1, 0]	[1, 0, 1, 1]	1.7	[1.8, 2.5, 2.5, 1.8]	[0, 0.7, 0.7, 0]
4	[1, 1, 1, 0]	[0, 1, 1, 1]	1.7	[1.8, 2.5, 2.5, 1.8]	[0, 0.7, 0.7, 0]
5	[2, 1, 0, 1]	[0, 1, 2, 1]	1.0	[2.5, 2.0, 2.5, 2.0]	[0, 1.0, 0, 0.2]
6	[1, 0, 2, 0]	[0, 2, 0, 1]	1.0	[2.0, 2.5, 2.0, 2.5]	[0.6, 0, 0.9, 0]
7	[0, 2, 0, 3]	[1, 1, 3, 0]	1.5	[2.0, 3.0, 2.0, 2.0]	[0, 0.8, 0, 0]

Figure 5.1: Portrait of the subgraphs of the windows of size 3 obtained from the time series on the left

The benefits of vector encoding are represented in the table above. For instance, if all features were computed for the first node in each window, the 3^{rd} and the 6^{th} observation would be considered almost identical, even though the second one corresponds to a minimum and the first to a value preceding another minimum.

$$\begin{aligned}\overrightarrow{e}_m(i) &= [k^{in}_m(i), k^{out}_m(i), D_m(i), hc_m(i), bc_m(i)]\\ S(\overrightarrow{e}_3(6), \overrightarrow{e}_3(3)) &= 95\%\\ S(\overrightarrow{e}_3(6), \overrightarrow{e}_3(4)) &= 95\%\end{aligned}$$

On the contrary, comparing the **topological patterns** entry by entry between these time points, which implies that the comparison considers the **time direction** and **distance**, shows that they are quite different from each other, and that actually the most similar observation to the 6^{th} is the 4^{th}, both corresponding to minimums of the series. Also, the betweenness centrality is always zero for the first and last observation in each window, so, considering the two elements of the vector is irrelevant. Removing them results in the following similarity scores:

$$\begin{aligned}\overrightarrow{e}_m(i) &= c(\overrightarrow{k}^{in}_m(i), D_m(i), \overrightarrow{hc}_m(i), \overrightarrow{bc}_m(i))\\ S(\overrightarrow{e}_3(6), \overrightarrow{e}_3(3)) &= 78\%\\ S(\overrightarrow{e}_3(6), \overrightarrow{e}_3(4)) &= 90\%\end{aligned}$$

where, for simplicity, $\overrightarrow{bc}_m(i) = [bc_m(i,1), \ldots, bc_m(i, m-1)]$ and $c(\overrightarrow{x}, \overrightarrow{y})$ is the concatenations of the two vectors.

Still, the similarity between the 3^{rd} and 6^{th} node is high due to the limits of the cosine similarity, which does not consider the magnitude of the compared vectors. Another possible similarity measure is the **Coincidence Similarity**:

"The coincidence similarity, consisting of a combination of the Jaccard and interiority indices (...) has been found to allow particularly complete and strict quantification of similarity, presenting enhanced performance in important tasks such as pattern recognition (...)"

Luciano da F.Costa [6]

which is calculated by the following equations:

$$s_x = \text{sign}(x) = \frac{x}{|x|} \tag{5.2}$$

$$s_{xy} = \text{sign}(xy) = \text{sign}(x)\,\text{sign}(y) = s_x s_y \tag{5.3}$$

$$J(\vec{x}, \vec{y}) = \frac{\sum_i s_{x_i y_i} \min\{s_{x_i} x_i, s_{y_i} y_i\}}{\sum_i \max\{s_{x_i} x_i, s_{y_i} y_i\}} \tag{5.4}$$

$$CS(\vec{x}, \vec{y}) = \frac{\sum_i \min\{s_{x_i} x_i, s_{y_i} y_i\}}{\min\{\sum_i s_{x_i} x_i, \sum_{i \in S} s_{y_i} y_i\}} s_1(\vec{x}, \vec{y}) \tag{5.5}$$

Using the coincidence index instead of the cosine leads to very distinct results:

$$CS(\overrightarrow{e}_3(6), \overrightarrow{e}_3(3)) = 36\%$$
$$CS(\overrightarrow{e}_3(6), \overrightarrow{e}_3(4)) = 54\%$$

Instantly, it is possible to realize that the two values are considerably smaller than the ones obtained before. This is absolutely reasonable since the 4 observations in the window ending at the 6^{th} time point are composed of two local minimums in the 4^{th} and 2^{nd} positions, a local maximum in the 3^{rd} position, and one value in a downwards trend in the first. In contrast, the subgraph ending at the 3^{rd} node has two vertexes in a declining tendency, the maximum of the series and one value preceding a maximum. As such, it is not plausible to attribute a similarity score so high as the cosine when the two patterns are so different. Even the similarity between the 6^{th} and the 4^{th} observations should not be as high as 90%, because the subgraphs are still distinct from one another in a way that should be captured by the similarity score. Furthermore, the gap between the two similarity scores is greater than the one obtained before, which shows that the coincidence index can enhance the spread of the distribution of the indexes.

Although this improvement is notable, considering both the in and out degree vectors is redundant. The two measures are almost symmetrical to each other and the sum of their elements is the same. Consequently, the differences between the two subgraphs will be doubly penalized by the two identical features representing the degree. One option to deal with this problem would be to calculate the degree in an undirected graph, but, considering the conclusions of chapter 4, both the in-degree and the out-degree are more helpful to distinguish between local minimums and maximums. Figure 4.2a demonstrates that the in-degree is more correlated with patterns that are not well represented by other features than the out-degree, leading us to select this measure rather than using both metrics or the degree.

With these changes the similarity between the observations being studied is now:

$$CS(\overrightarrow{e}_3(6), \overrightarrow{e}_3(3)) = 47\%$$
$$CS(\overrightarrow{e}_3(6), \overrightarrow{e}_3(4)) = 60\%$$

Thus, the penalization of the differences in the degree component is smaller, resulting in higher similarities, but still in a reasonable range.

Another possible refinement of this strategy is to *standardize* the elements of the vectors based on the same components in other vectors. As an outcome of this pre-processing method, the same entries across all vectors have now null mean and unit standard deviation. Though, applying this step to a network as small as the one in figure 5.1 leads to poor results given that the distribution is not based on enough values. In conclusion, standardizing the components of the vectors only makes sense for time series with enough data, particularly because it is helpful to not give more weight to one topological feature than others.

5.2 Similarity Finder

Finding the similarities scores between all observations and all previous time points in a time series based on their topological historical patterns has many benefits for any type of analysis:

No temporal restrictions : The most similar observations may be from before the last seasonal period or even very far in time;

Independent of seasonality : It is very much possible that one observation, i, is more similar to time points not belonging to $\{i - nP\}$, where $n \in \mathbb{Z}_+ \land n \leq \frac{i}{p}$ and P is the period;

Pattern Recognition : Each time point is not only described by its position in the time series but by itself and its m past observations;

Outside-the-box characterization : Each pattern is characterized by topological features computed from VGs that represent classical time series characteristics and other hidden traits of the data.

Based on the previous section's strategy to compare observations based on their subgraphs of size m, it is now feasible to implement the similarity finder algorithm represented above, which can be mathematically described by the following equations for small time series:

$$\overrightarrow{S}_m^{tf}(i) = [CS(tf_m(i), tf_m(0)), \ldots, CS(tf_m(i), tf_m(t-1))] \tag{5.6}$$

$$\overrightarrow{S}_m^{G}(i) = \frac{\overrightarrow{S}_m^{\overrightarrow{k}^{in}}(i) + \overrightarrow{S}_m^{D}(i) + \overrightarrow{S}_m^{\overrightarrow{hc}}(i) + \overrightarrow{S}_m^{\overrightarrow{bc}}(i)}{4} \tag{5.7}$$

This methodology compares the different observations by calculating a global similarity score based on the concatenation of the different vector encodings and the average shortest path length. However, as demonstrated in the comparison between the features on subsection 4.2.3, some topological features may be more important to describe specific time series than others. In light of this conclusion, the similarity finder algorithm implemented allows computing a weighted average of the similarities between each topological feature:

$$W\vec{S}^{G}_{m}(i,a,b,c,d) = \frac{a\vec{S}^{\vec{k}^{in}}_{m}(i) + b\vec{S}^{D}_{m}(i) + c\vec{S}^{\vec{hc}}_{m}(i) + d\vec{S}^{\vec{bc}}_{m}(i)}{a+b+c+d} \tag{5.8}$$

and (a, b, c, d) are parameters of the algorithm which represent the weights of the similarities between each topological measure.

The schematic implementation of the similarity finder class available in the *networktsf* package is presented in the figure below. This class has two main methods, *fit_transform(series)*, where all possible pairs of similarity indexes are calculated, and the *transform(series2)*, where only the pairs of the new observations are computed (*series2* is a continuation of *series*).

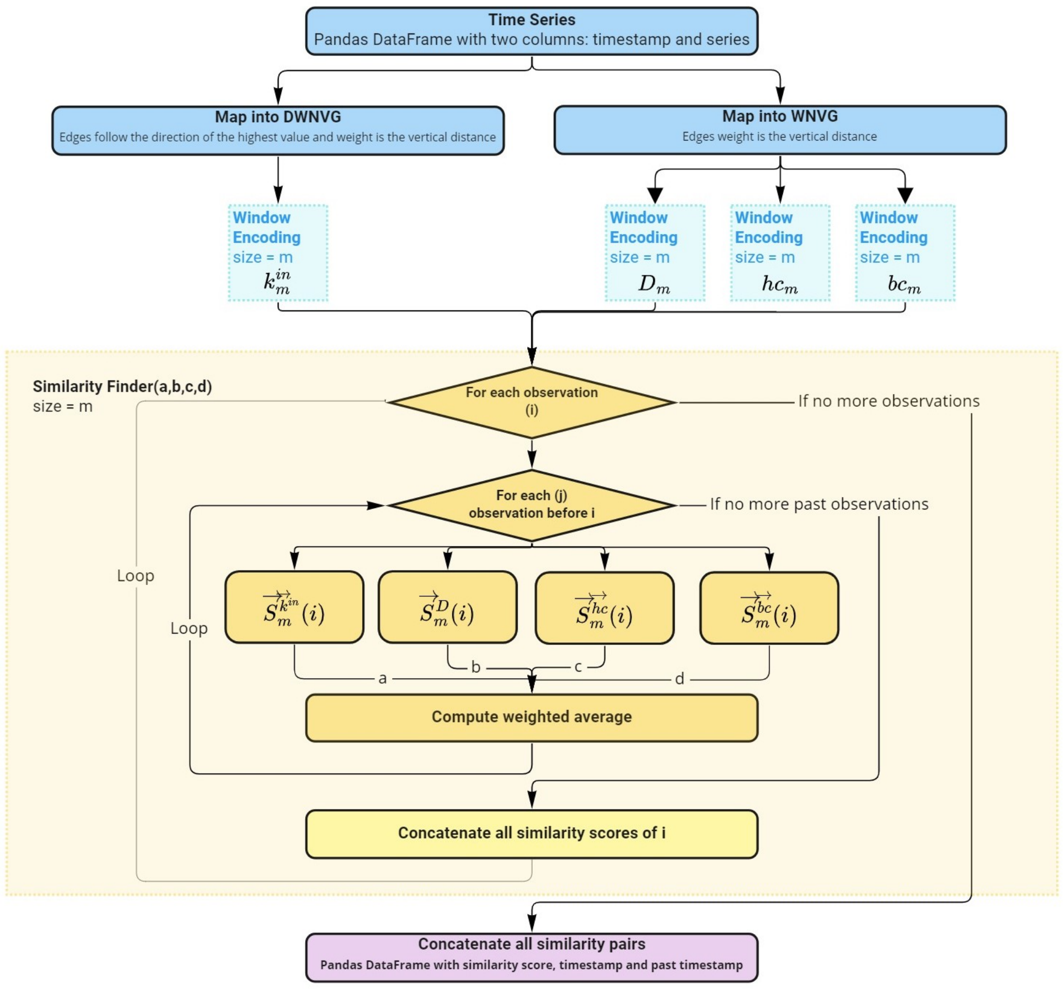

Time Series
Pandas DataFrame with two columns: timestamp and series
Map into DWNVG
Edges follow the direction of the highest value and weight is the vertical distance
Map into WNVG
Edges weight is the vertical distance
Window Encoding size = m k^{in}_m
Window Encoding size = m D_m
Window Encoding size = m hc_m
Window Encoding size = m bc_m
Similarity Finder(a,b,c,d)
size = m
For each observation (i)
If no more observations
For each (j) observation before i
If no more past observations
Loop
Loop
$\vec{S}^{\vec{k}^{in}}_{m}(i)$
$\vec{S}^{D}_{m}(i)$
$\vec{S}^{\vec{hc}}_{m}(i)$
$\vec{S}^{\vec{bc}}_{m}(i)$
a
b
c
d
Compute weighted average
Concatenate all similarity scores of i
Concatenate all similarity pairs
Pandas DataFrame with similarity score, timestamp and past timestamp

Figure 5.2: Schematic diagram for the Similarity Finder

Chapter 6

Forecasting via Complex Networks

"Occasionally, old data will be less useful due to structural changes in the system being forecast; then we may choose to use only the most recent data. However, remember that good statistical models will handle evolutionary changes in the system; don't throw away good data unnecessarily."

Hyndman, R.J., & Athanasopoulos, G. (2018) [16]

In the last decades, a plenitude of forecasting models was developed, and most provide great forecasts when the time series being forecasted respects the assumptions of the methods. Many of these algorithms use the autocovariance and autocorrelation functions for the estimation of parameters, which carry the historical information of the time series. Anyhow, forecasting models usually neglect relations between observations distant in time, in particular when such events are very distinct from the rest of the series.

Based on the quote from Hyndman R.J *et all* [16], it is crucial for prediction models to not discard good data. The problem with this statement regards the definition of *"good data"* since some doubts may arise from it. Is it important to take into account data that is distant in time? Are outliers worth considering? If the system is very different from a specific period, should the model gather information from it to predict this distinct and more recent stage of the series? The present study interprets *"good data"* as all observations in a time series, which is evidently a bold assumption that can be justified by answering the questions above.

Data from a long time ago : Models should consider old observations because their future may be similar to what is being predicted;

Outliers : Considering outliers is trickier. If they exist due to an error in the measurements, then they should not be considered. However, if that is not the case, patterns before them may help predict new outliers, and what happens afterward also helps to forecast what will occur after a new one;

Distinct Systems : Even if a new system is completely different from the past, the patterns in the past may repeat once again. For instance, if there are cycles in the series, the series may change its system from cycle to cycle.

In this regard, the next question would be "how?" (e.g. "How can a model consider all observations to predict the future?"). In fact, most forecasting methods only consider the entire time series for parameter estimation. One good example is any auto-regressive model in which the parameters define which lags are going to be used to predict all observations, independently of changes in the patterns (e.g. an $ARIMA(1,0,0)(1,0,0)_{12}$ only considers observations y_{t-1}, y_{t-12} and y_{t-13} to estimate y_t). In conclusion, the main goal of the present work is to answer the "how?" question by finding an algorithm flexible enough to use any observation to predict ahead of the most recent one in which the model was fitted.

A solution to the aforementioned question is provided in this chapter through the proposal of a new forecasting method called *Forecasting based on Networks' Similarities* (**FbNS**). Moreover, an evaluation of this model's performance is presented, while comparing it to baseline models.

6.1 Forecasting based on Networks Similarities

The two previous chapters already provide a hint to achieving this study's primary objective. By using the similarity finder algorithm (Chapter 5), it is possible to get similar time points to the most recent observation based on the past topological patterns. In theory, such strategy answers the "how?" question because, as mentioned in section 5.1, this methodology has no **temporal restrictions** and is **independent of seasonality**, which means it is capable of finding valuable observations to predict y_t very far in time and outside $\{t-nP\}$, $n \in \mathbb{Z}_+ \wedge n \leq \frac{t}{P}$ (where P is the frequency). Additionally, the algorithm works by **recognizing patterns** with an **out-of-the-box characterization** of each measurement in the time series, something that can be extremely valuable given that, most likely, similar events have similar futures.

As such, this section introduces a novel time series prediction algorithm built on the principles proposed in the present work. Its name, *Forecasting based on Networks' Similarities* (FbNS), arises from the core step of the algorithm, which is finding similar observations based on the subgraphs of size $m+1$ ending in them. The implementation of the FbNS was developed to answer all the questions made in this chapter, leading to a flexible and easy class to use, available in the *forecasting* module in the *networktsf* library. As per usual, this class only encompasses two main methods: *fit(time series)* and *predict(N)*, in which N is the number of steps ahead desired to predict. Also, the practicality of the model can be justified by the fact that it has only one hyperparameter: the size of the window, m, in which the topological features of each observation are computed to calculate the similarity between themselves.

FbNS first fitting step is to **map the time series** into two VGs, a WNVG and a DWNVG. Both mappings edges weight is based on the vertical distance, and the directed graph edges point to the node with higher value. Then, the window encoding strategy is used to compute the in-degree vector in the directed network, while the average shortest path length, the harmonic centrality, and the betweeness centrality vectors are calculated in the undirected graph. Subsequently, these topological measures are used to calculate the similarity between all available time points for each feature.

At this stage, the weighted average of the similarity scores must be calculated to generate predictions. Nonetheless, observations with low similarity may not be helpful to produce a good forecast. Given this, a limit to which time points are considered must be delineated based on the each time point similarity distribution. FbNS selects the relevant observations, $RO(t-1)$, to generate the forecast of y_t by finding the time points with similarity scores to y_{t-1} above the pth-quantile of the $W\overrightarrow{S}_m^G(t-1,a,b,c,d)$ (5.8) distribution:

$$WS_m^G(t-1,i,a,b,c,d) = W\overrightarrow{S}_m^G(t-1,a,b,c,d)(i) \tag{6.1}$$

$$WS_m^G(t-1,i) = WS_m^G(t-1,i,a,b,c,d) \tag{6.2}$$

$$W\overrightarrow{S}_m^G(t-1) = W\overrightarrow{S}_m^G(t-1,a,b,c,d) \tag{6.3}$$

$$RO_{t-1} = \{i\} \in WS_m^G(t-1,i) > \mathrm{Q(p)}\, \overrightarrow{WS}_m^G(t-1) \tag{6.4}$$

and, in the end, the forecast is simply a random walk with corrections:

$$y_t(m,a,b,c,d,p) = y_{t-1} + \sum_{i \in RO_{t-1}} \frac{WS_m^G(t-1,i)}{\sum_{j \in RO_{t-1}} WS_m^G(t-1,i)} (y_{i+1} - y_i) \tag{6.5}$$

With these equations, it is possible to make predictions based on the subgraphs similarities. However, this model would not be easy to use with so many hyperparameters. To solve this issue, the implementation of the model estimates the weights of the similarities (a,b,c,d) and p through a **least squares** estimation (subsection 2.1.3). This step is of utmost importance because, as described in the previous chapter, some features may be more pertinent to describe similar events than others in specific time series, and the definition of the p-quantile allows finding all relevant observations that minimize the forecasting error.

At long last, $FbNS$ can make predictions, yet, only one-step-ahead which is not useful for most cases. Thus, the prediction method must be capable of predicting n steps ahead. To that end, for each forecast, FbNS computes the similarity scores to all past observations (and forecasts if $n > 1$). Then, it uses the estimated parameters to produce each forecast iteratively:

$$y_{t+(n-1)} = y_{t+(n-2)} + \sum_{i \in RO_{t+(n-2)}} \frac{WS_m^G(t+(n-2),i)}{\sum_{j \in RO_{t+(n-2)}} WS_m^G(t+(n-2),i)} (y_{i+1} - y_i) \tag{6.6}$$

where $n \in \mathbb{Z}_+$

One problem that may be hard for the model to solve is when the mean difference between consecutive observations is changing. Series that present such characteristic have a non-normal distributions of their values. To solve this, FbNS performs a Shapiro-Wilk test [27] to verify if the series follows a normal distribution, then, if it does not, a box-cox transformation [2] is performed to normalize it.

All of these steps combined represent the fitting method of the FbNS, from which the parameters necessary to make a one-step-ahead prediction are computed and saved in the model class.

The schematic representation of the two methods is presented on the following page.

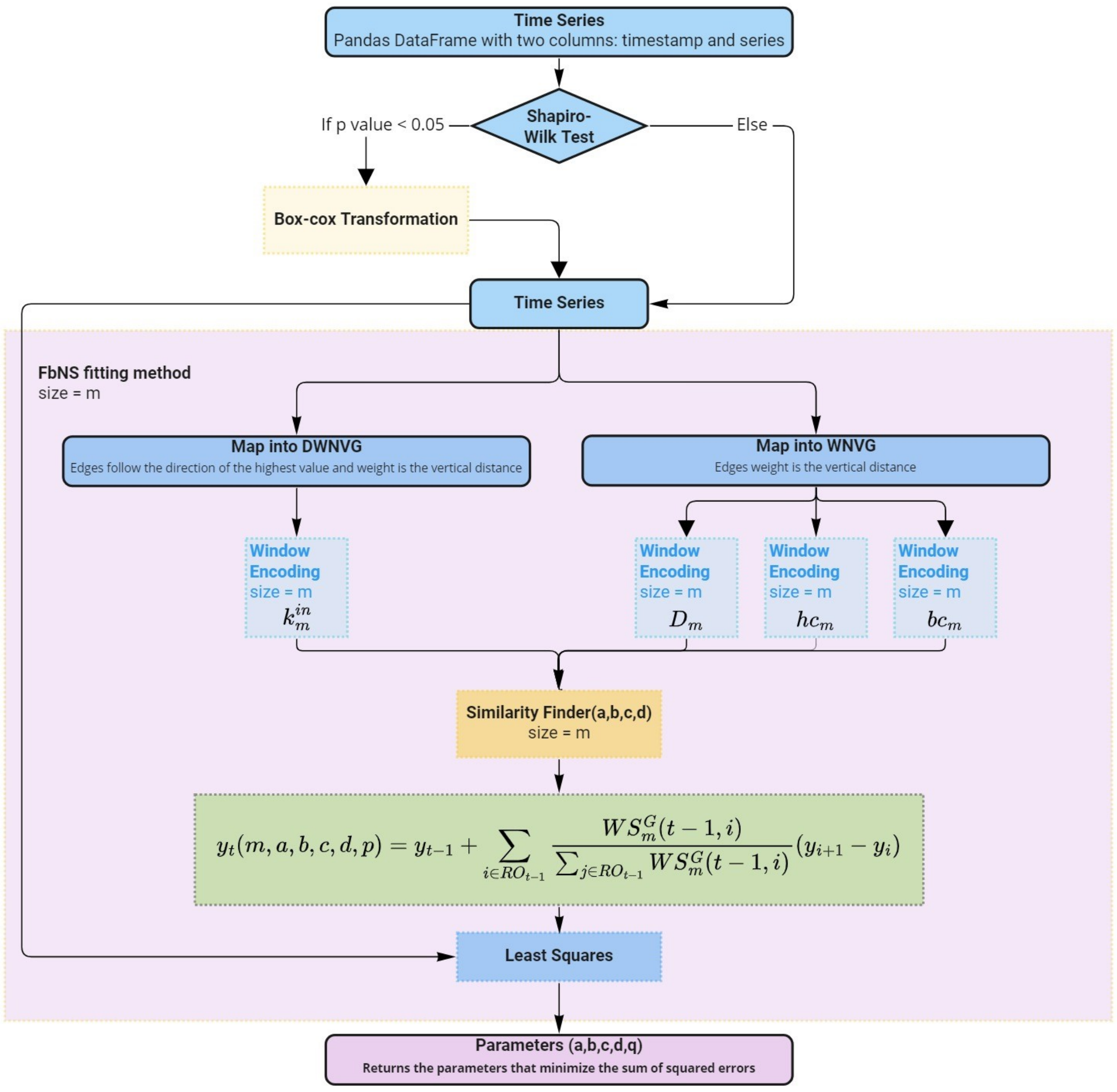

Time Series
Pandas DataFrame with two columns: timestamp and series
Shapiro-Wilk Test
If p value < 0.05
Else
Box-cox Transformation
Time Series
FbNS fitting method
size = m
Map into DWNVG
Edges follow the direction of the highest value and weight is the vertical distance
Map into WNVG
Edges weight is the vertical distance
Window Encoding size = m k_m^{in}
Window Encoding size = m D_m
Window Encoding size = m hc_m
Window Encoding size = m bc_m
Similarity Finder(a,b,c,d) size = m
$y_t(m,a,b,c,d,p) = y_{t-1} + \sum_{i \in RO_{t-1}} \frac{WS_m^G(t-1,i)}{\sum_{j \in RO_{t-1}} WS_m^G(t-1,i)} (y_{i+1} - y_i)$
Least Squares
Parameters (a,b,c,d,q)
Returns the parameters that minimize the sum of squared errors

Figure 6.1: Schematic diagram for the fitting method of the FbNS

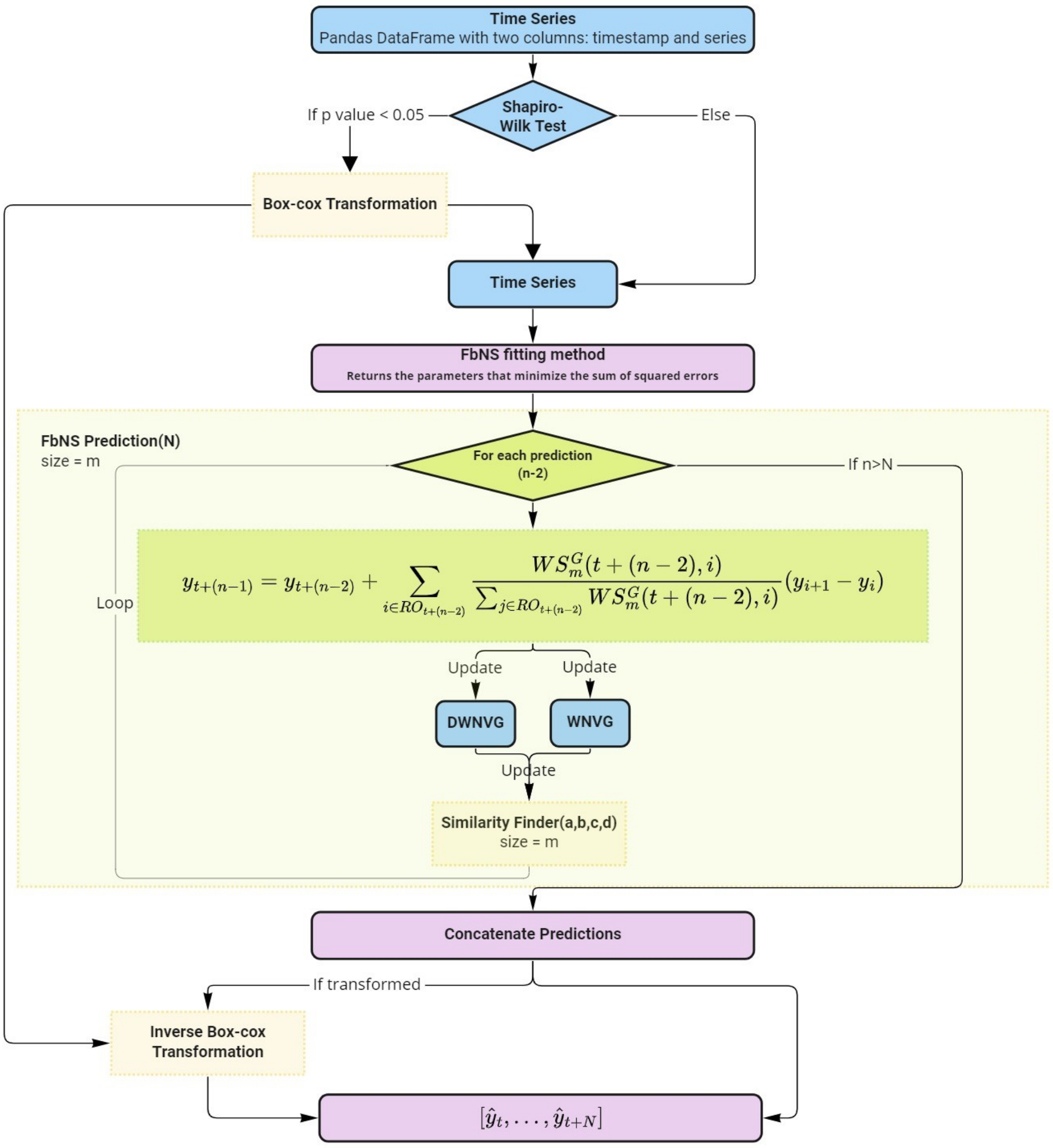
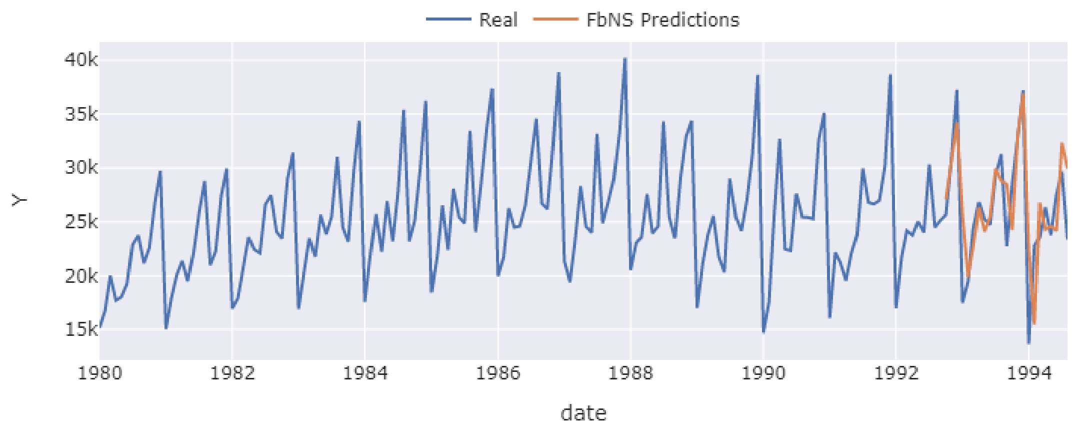

Time Series
Pandas DataFrame with two columns: timestamp and series
Shapiro-Wilk Test
If p value < 0.05
Else
Box-cox Transformation
Time Series
FbNS fitting method
Returns the parameters that minimize the sum of squared errors
FbNS Prediction(N)
size = m
For each prediction (n-2)
If n>N
Loop
$y_{t+(n-1)} = y_{t+(n-2)} + \sum_{i \in RO_{t+(n-2)}} \frac{WS_m^G(t+(n-2),i)}{\sum_{j \in RO_{t+(n-2)}} WS_m^G(t+(n-2),i)} (y_{i+1} - y_i)$
Update
Update
DWNVG
WNVG
Update
Similarity Finder(a,b,c,d)
size = m
Concatenate Predictions
If transformed
Inverse Box-cox Transformation
$[\hat{y}_t, \ldots, \hat{y}_{t+N}]$

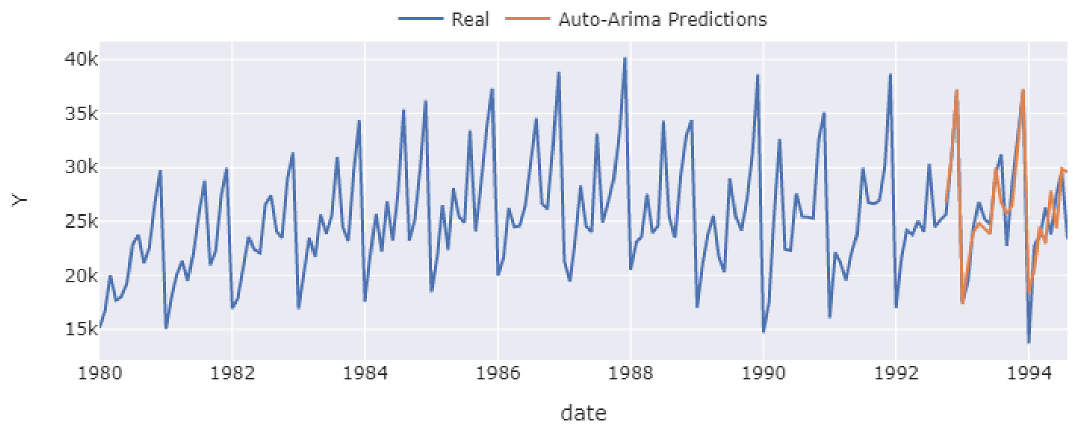
Figure 6.2: Schematic diagram for the prediction method of the FbNS

6.2 Example of Application

Given that the proposed model is now defined, the next step is to check if it works. An example of the applicability of the FbNS algorithm is presented in this section. To do so, the value of the total monthly wine sales in the Australia series was divided into a train set between January 1980 and September 1992, and a test set with months ranging from October 1992 and August 1994. Cross-validation with one-step-ahead forecasts is displayed to analyse the model without the accumulated error of previous predictions that occurs in multi-step forecasts. Afterwards, the model was fitted in the train set and the predictions were generated for the following 24 months after the end of the training in order to compare with the test set.



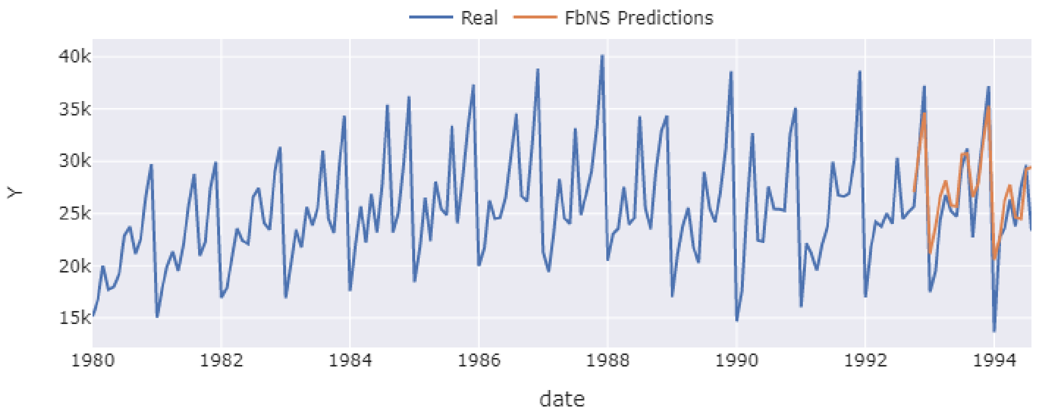
(a) FbNS One-step-ahead forecast



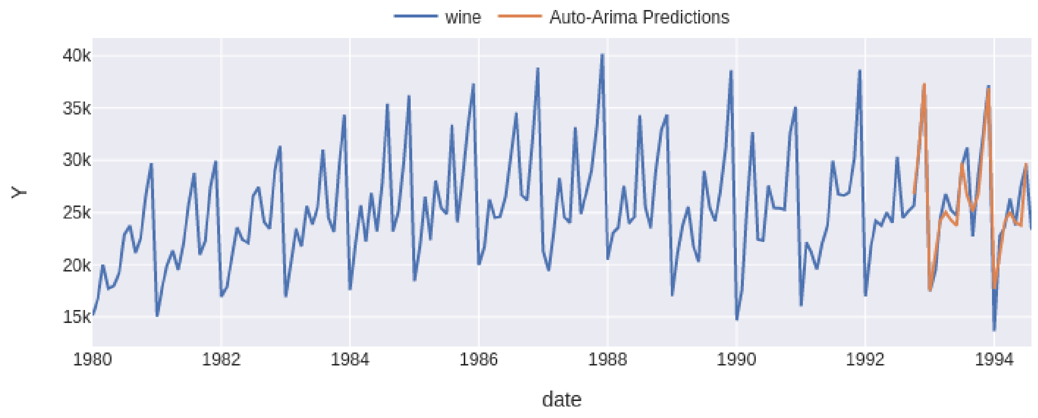
(b) Auto-Arima One-step-ahead forecast

Figure 6.3: One-step-ahead forecasts of the total monthly wine sales in Australia series

From the analysis of the plots in figure 6.3 it stands out that both models are a really good fit for the series. Not only that, the one-step-ahead predictions made with the FbNS generate better forecasts in the local minimums and maximums of each year, meaning that it is more capable of capturing local variations; while the Auto-Arima model is much better in predicting the minimums of each season. The mean absolute percentage error of the FbNS is 13%, yet the Arima does a better job with an 8% *MAPE*.



(a) FbNS 24-steps-ahead forecast



(b) Auto-Arima 24-steps-ahead forecast

Figure 6.4: 24-steps-ahead forecasts of the total monthly wine sales in Australia series

When making a 24-steps-ahead forecast the scenery is different. In fact, the performance of the FbNS improves (its MAPE is now 9.6%), which can be verified by an even more stable representation of the local variations, far better than the one obtained by the Auto-Arima. The second method still has better performance ($MAPE = 5.7\%$), something that arises from the same problem as in the one-step-ahead forecast, the FbNS model is not able to capture the minimums of each year as well as the Arima.

One point worth exploring is the local maximum that happens in August 1993, because since 1987 this month actually corresponds to a local minimum, but before that date, it used to be a maximum. Obviously, the Auto-Arima model is not able to predict this variation similar to the ones that happened at least 7 years before. In contrast, since the FbNS method has no temporal restrictions and the topological patterns in 1993 are similar to the ones observed in 1986, the model actually predicts that this month is a local maximum, which contradicts the tendency from more recent years. The main goal of building this model was to capture situations such as this one. Hence, this example shows that it is possible to do so.

Besides the *fit* and *predict* methods, the FbNS also has a *plot_diagnostic()* function. This diagnostic contains the similarity distributions for each feature and the estimated weight of each of them to compute the *global similarity distribution*. Moreover, the estimated *pth*-quantile of the weighted sum of the similarities distribution is presented in the histogram of this variable.

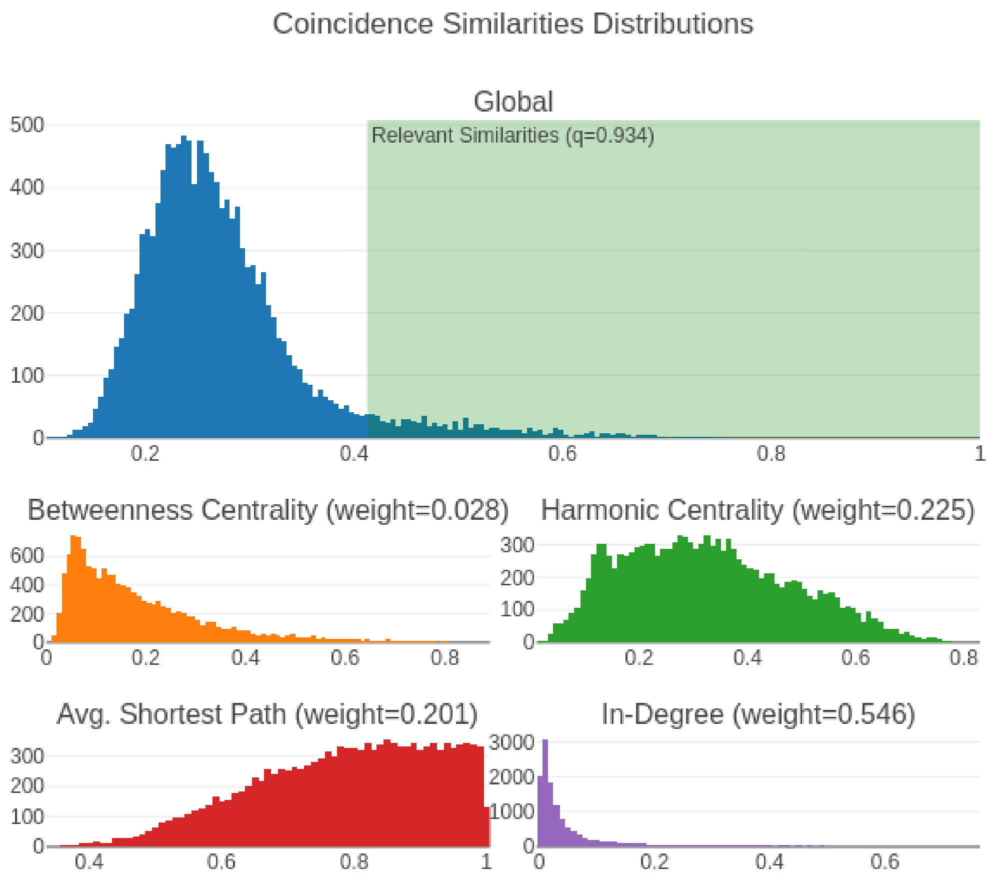


Figure 6.5: Similarity distributions of the FbNS model in the total monthly wine sales in Australia series

The diagnostic produced for wine series (Figure 6.5) demonstrates that most pairs of observations are dissimilar, resulting in only considering the observations with similarity above Q(93.4%) $\overrightarrow{WS}^{Z}_{m}(i)$ to predict $i+1$. Another important analysis to make with these plots is the study of the global similarity distribution. Ideally, this distribution would be bimodal, with the first maximum near 0 and another one near 1. This is not the case for the wine series, since the distribution is actually right-skewed, which is not so bad because, most likely, the valuable time points are in the right tail. Probably, if the distribution was normal, the model would not be able to make predictions. In this sense, paying attention to this diagnosis is fundamental to understand if the model can be used to predict the series it was fitted in.

6.3 Experimental Studies

A further evaluation of the model performance is displayed in this chapter and compared with the Auto-ARIMA results in multiple time series. This comparison is done by calculating the mean absolute and root mean squared errors (MAE and RMSE) 2.1.3 for each model.

In addition, a regression with ARIMA errors is also included in the study. In this model ($k_{in}(f)k_{out}(f)$ AUTO-ARIMA), the regressors are the in-degree and out-degree computed for the last node in each subgraph of size equal to the frequency of each series mapped with a DWNVG (weight is the vertical distance between nodes and the edges point to the node with higher value).

Each model was fitted in almost $4k$ time series, with different prediction horizons. These sequences are divided into different datasets with categories that describe the field the observations were collected from:

1. **FJ Glucose Values**:
   - **Category** - Medicine
   - **Horizon** - 24 steps
2. **Other time series in section 6.3** (except M2 Dataset) (5 series)
   - **Category** - Macro Economics
   - **Horizon** - Twice the frequency
3. **M1 Dataset** (1001 series)
   - **Category** - categories of M1-competition
   - **Horizon** - horizon from M1-competition
4. **M2 Dataset** (29 series)
   - **Category** - categories of M2-competition
   - **Horizon** - horizon from M2-competition
5. **M3 Dataset** (2829 series)
   - **Category** - categories of M3-competition
   - **Horizon** - horizon from M3-competition

In order to perform this experiment, the developed code uses parallel computing so that each model fits and makes predictions for 16 series at the same time (the machine used has 16 processors). Then, the performance is compared in the following tables considering the dataset, frequency and category.

Model	Auto-Arima	FbNS	Auto-Arima ($k_{in}(f)k_{out}(f)$)
Dataset			
FJ Glucose 15 minutely (1 series)	48.2	58.6	49.9
M1 monthly (617 series)	1971.1	3340.2	2246.7
M1 quarterly (203 series)	1983.1	4639.5	2957.8
M1 yearly (177 series)	119067.2	136094.9	198662.9
M2 monthly (23 series)	89590.1	165924.2	104560.4
M2 quarterly (6 series)	33.2	31.5	36.6
M3 monthly (1428 series)	731.7	1173.9	728.7
M3 quarterly (756 series)	631.8	731.0	693.2
M3 yearly (645 series)	1502.8	3912.0	1694.8
air passengers monthly (1 series)	68.6	37.6	71.1
ausbeer quarterly (1 series)	5.8	17.7	6.3
monthly milk monthly (1 series)	7.8	14.4	8.4
wine monthly (1 series)	1411.9	1906.3	1604.7
wooly quarterly (1 series)	760.5	726.4	813.2

Table 6.1: Comparison between the average **MAE** for each dataset and frequency

Model	Auto-Arima	FbNS	Auto-Arima ($k_{in}(f)k_{out}(f)$)
Dataset			
FJ Glucose 15 minutely (1 series)	54.2	65.9	56.4
M1 monthly (617 series)	2336.5	4039.2	2671.2
M1 quarterly (203 series)	2265.5	5386.9	3427.6
M1 yearly (177 series)	136695.6	151608.6	219643.9
M2 monthly (23 series)	111494.4	195456.0	129129.7
M2 quarterly (6 series)	37.0	34.8	40.3
M3 monthly (1428 series)	870.9	1419.7	878.0
M3 quarterly (756 series)	735.7	871.9	808.5
M3 yearly (645 series)	1706.2	5859.0	1939.7
air passengers monthly (1 series)	74.3	41.0	76.9
ausbeer quarterly (1 series)	7.1	20.6	8.4
monthly milk monthly (1 series)	9.6	16.8	11.0
wine monthly (1 series)	1968.9	2461.4	2101.8
wooly quarterly (1 series)	1028.5	948.5	1060.1

Table 6.2: Comparison between the average **RMSE** for each dataset and frequency

Tables 6.1 and 6.2 demonstrate that the regression with ARIMA errors is very similar to the Auto-Arima, and, in all datasets its RMSE is slightly bigger. With this analysis it is possible to conclude that using the in-degree and out-degree in a regression model may add noise to the method, therefore, its performance is worse than a simpler version.

The average MAE and RMSE for each dataset and frequency shows that the proposed model is competitive enough to get better results than an Auto-Arima in some cases. In fact, FbNS performs better in 1409 **time series**. However, the classical method is still a better forecasting method for most time series studied. This observation is due to the assumptions of each model: FbNS gives more importance to patterns similar to the ones observed in the past than recent changes of the series. If the pattern is new but can be induced by the recent behaviour of the sequence, FbNS won't be able to predict it. On the other hand, an ARIMA model always considers the most recent observations as the most important ones, providing stability to the model.

Only computing the average of the errors makes this analysis susceptible to outliers. Figure 6.6 allows observing the distributions of the mean absolute error, for each model, dataset and frequency.

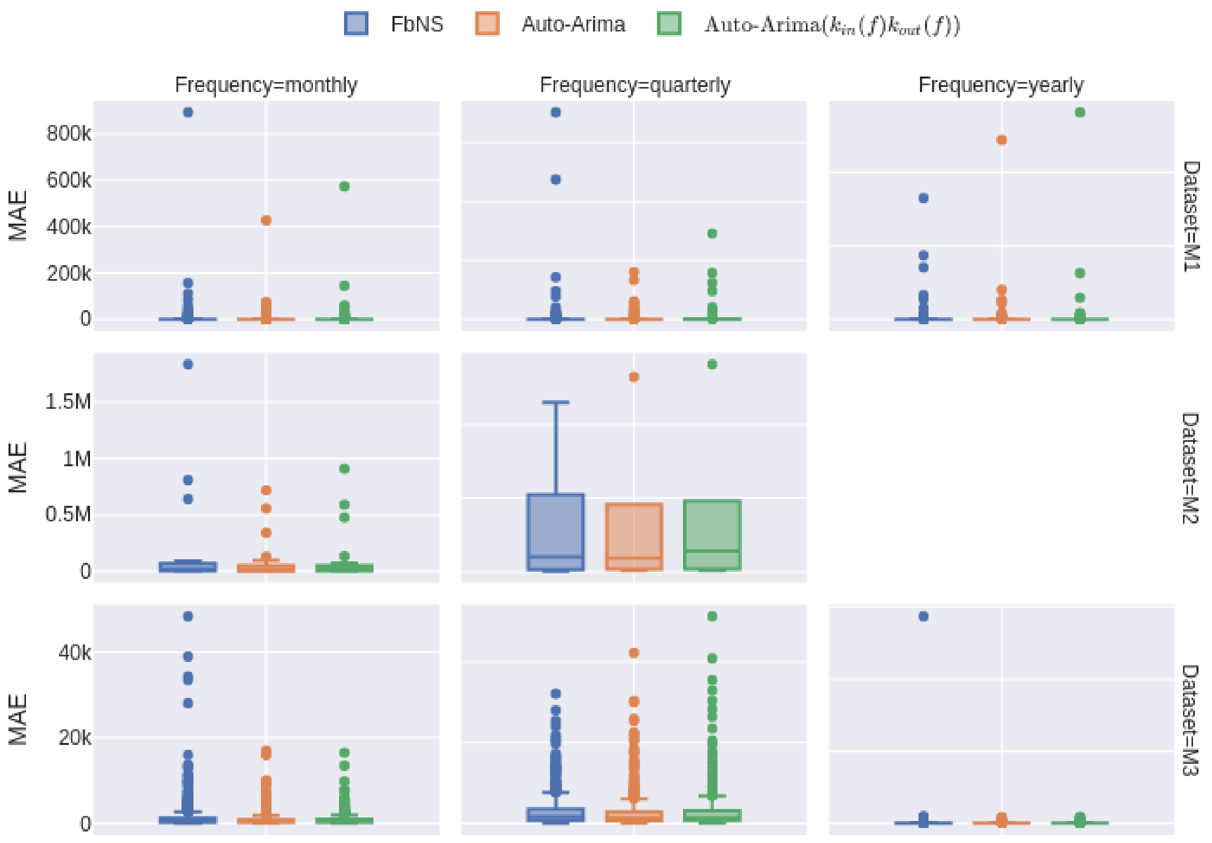


Figure 6.6: Distribution of the MAE for each model, dataset and frequency

The distributions above help comprehending the big differences in the average MAE shown in table 6.1, which are due to outliers, especially in the results from the FbNS model. Actually, when removing the outliers, the distributions are very similar. Therefore, the proposed algorithm is clearly worth exploring, in particular, when the series might have repeating patterns from the past.

Another interesting point of view is to evaluate the models for each category in the datasets. Tables 6.3 and 6.4 present the average value of the errors of each model in each category.

Model	Auto-Arima	FbNS	NV Auto Arima
Category			
BIOLOGY (1 series)	7.8	14.4	8.4
DEMOGRAPHIC (554 series)	1152.2	1529.8	1273.3
FINANCE (279 series)	1337.0	1957.9	1377.6
INDUSTRY (755 series)	2623.9	6807.5	2901.6
MACRO (1054 series)	1600.6	8807.1	3590.2
MEDICINE (1 series)	48.2	58.6	49.9
MICRO (1154 series)	19593.5	16478.9	18631.6
OTHER (63 series)	751.2	1167.0	767.1

Table 6.3: Comparison between **MAE**'s for each category

Model	Auto-Arima	FbNS	NV Auto Arima
Category			
BIOLOGY (1 series)	9.6	16.8	11.0
DEMOGRAPHIC (554 series)	1332.8	1883.1	1485.6
FINANCE (279 series)	1511.9	2304.7	1573.3
INDUSTRY (755 series)	2946.6	7377.0	3260.9
MACRO (1054 series)	1937.5	9632.2	4382.4
MEDICINE (1 series)	54.2	65.9	56.4
MICRO (1154 series)	22690.0	19975.4	20707.2
OTHER (63 series)	903.5	1419.2	921.3

Table 6.4: Comparison between **RMSE**'s for each category

Analysing the results in both tables leads to similar conclusions from the ones made previously (i.e. the scale of the errors is the same for all categories, which means that the FbNS is somewhat similar in performance to the Auto-Arima). There is no clear difference between the two models, though the FbNS ig globally better in micro economics series, while the Auto-Arima is superior in all other categories.

It should be noted that the window size of the FbNS model was not adjusted to any series (the frequency of each series was always selected as the window size), because the objective of this study was to show that the FbNS model was competitive enough, even when not searching for its only hyperparameter. In contrast, the Auto-Arima model already does this for each series it is fitted in.

With all this in mind, the study clearly shows that the proposed model is worth exploring. Finding the optimal window size may also be an important task to get a better fit. Though, if the series is not very much correlated to any event in the past, then a Auto-Arima is probably the best choice out of these two; contrarily, when historical information is very important the FbNS is a good candidate to generate good predictions. Furthermore, the results also support the use of the similarity finder algorithm proposed in Chapter 5, since it finds similar observations based on their past, which are used by the FbNS to make predictions.

Chapter 7

Conclusions

Most forecasting models ignore data far in time when making predictions, which is a limitation because a pattern that may be happening or will happen in the future might be similar to one distant in time. In the past decade, a new area that maps time series into complex networks to mine their characteristics started gaining attention. Moreover, forecasting algorithms based on these complex network mappings were developed in recent years and proved that this new research topic can provide good tools to build useful forecasting methods. The work developed in this thesis was focused on analysing the advantages of characterizing observations with topological features collected from complex graphs, while trying to reduce the limitation of most forecasting models described above.

This Chapter summarizes our main contributions and presents suggestions for future research based on the work developed.

7.1 Main Contributions

Chapter 3 displays a taxonomy of the forecasting algorithms based on network science. This contribution creates a global picture of the different algorithms previous authors developed. Making this overview was an important stepping stone to materialize the present study.

The second contribution is a new implementation of a window encoding (section 4.1) strategy that allows computing topological features in subgraphs of the series that can be used to forecast. Then, this strategy was used to compute features for each observation in several time series, so that they could be compared to the classical patterns of every sequence. In the end, the study conducted to find the optimal window size and mapping method for each topological feature, as well as the advantages of using these metrics to describe time series lead to the following conclusions:

- The study made in subsection 4.2.1 suggests that, for periodic series, the best window size m is probably in the interval $m \in [P/2, 2P]$, where P is the period of the series.
- Most mapping methods result in interesting variables that represent several classic characteristics of the time series they are computed for. When the features are computed in graphs where the edges correspond to distances between observations their correlation with the patterns in subsection 2.1.2 is higher.
- Lastly, the chapter closes by suggesting the use of the combination of in and out degree computed for a directed weighted VG, in which the direction is the series and the weight is vertical distance, and the average shortest path length, betweenness centrality and harmonic centrality for the same, but undirected network.

One of the core implementations of this thesis is the similarity finder algorithm described in Chapter 5. This methodology, supported by the vector encoding introduced in the window encoding section (4.1), allows finding similar observations in a time series. This algorithm uses the conclusions of Chapter 4, by combining the topological features suggested in order to describe each observation. Then, the similarity between the vectors obtained is calculated with the coincidence similarity metric [6].

Finally, the main goal of the present thesis was achieved by the combination of all contributions made in this work. A novel forecasting model, FbNS, was introduced in Chapter 6. The window encoding strategy was used to compute the vectors of the topological features suggested in chapter 4; then with these features, the similarity finder algorithm calculates the coincidence index to compare patterns in the series; after all, the predictions are made based on the most similar observations to the one being forecasted. The study of the performance of the FbNS shows that the model is not limited by time distance, since in some cases the most similar patterns are from distinct periods of the series. Also, the model performance proved that it can provide better results than the Auto-Arima, even without finding the optimal window size.

In summary, this work addressed the common issue of neglecting old data in many forecasting algorithms. To do so, complex network mappings were used to build a novel forecasting method, FbNS, based on the similarity between the subgraphs topological patterns. Using the proposed window encoding strategy, and features that proved capable of capturing important traits of the series, was essential to make the present thesis possible.

7.2 Future Work

All objectives of this work were achieved, making it a good foundation for further developments. While developing the algorithms proposed and testing them in distinct contexts, other studies and improvements seemed to be appropriate to tackle some limitations of the methods and to strengthen their analysis.

Studying other topological features and mappings might give other perspectives that enrich the characterization of the observations in a time series. Even though, section 6.3 is focused in four topological features, other measures were tested before deciding the ones which would be presented in this work. Nonetheless, the study made for other graph topologies was not as deep. As such, understating the benefits of using other measures or other mappings would increase the knowledge in this field.

Since the main goal of this thesis was to address some limitations of prediction algorithms, all developments made were focused on time series forecasting. It is true that this subject is particularly interesting nowadays, but other fields could benefit from the window encoding strategy (4) and the similarity finder algorithm (5). Applying these methodologies for time series analysis, clustering, outlier detection or even just to find if something very similar already happened in the past, may help other fields grow (e.g. if data from a heartbeat sensor in a patient shows an odd observation recently, and the doctor wants to know if it was the first time, the similarity finder can help selecting time periods that may show the same behaviour).

In terms of further developments, the proposed forecasting method could benefit from an evolution in the way it is predicting. One of the biggest limitations of the algorithm is that, most of the times, it does not make use of recent changes in the flow of the series to generate better forecasts. Exploring the combination of the FbNS with an autoregressive model (AR(p)) could solve this.

Furthermore, FbNS does not consider the possible error of its predictions, which, when the error is high for a certain prediction step, leads to a situation where all further forecasts carry that error and their values are very far from the real ones. Adding an MA process to the prediction algorithm, to model the errors, could make the method more stable.

Including exogenous variables as regressors of a method can be important depending on the cross-correlation between the feature and the endogenous series. This method could be easily applied to the developed model. Though, including these features in the similarity finder algorithm would allow finding close correlated time points both in terms of the characteristics of the time series itself, and the variables which may have influenced its behaviour.

The promising results obtained show that time series forecasting via complex networks has the potential to really boost this field of study. Besides, all findings in this thesis may enrich other fields, and help solving difficult problems across several subjects.

Appendix A

Characterization of Observations using Topological Features

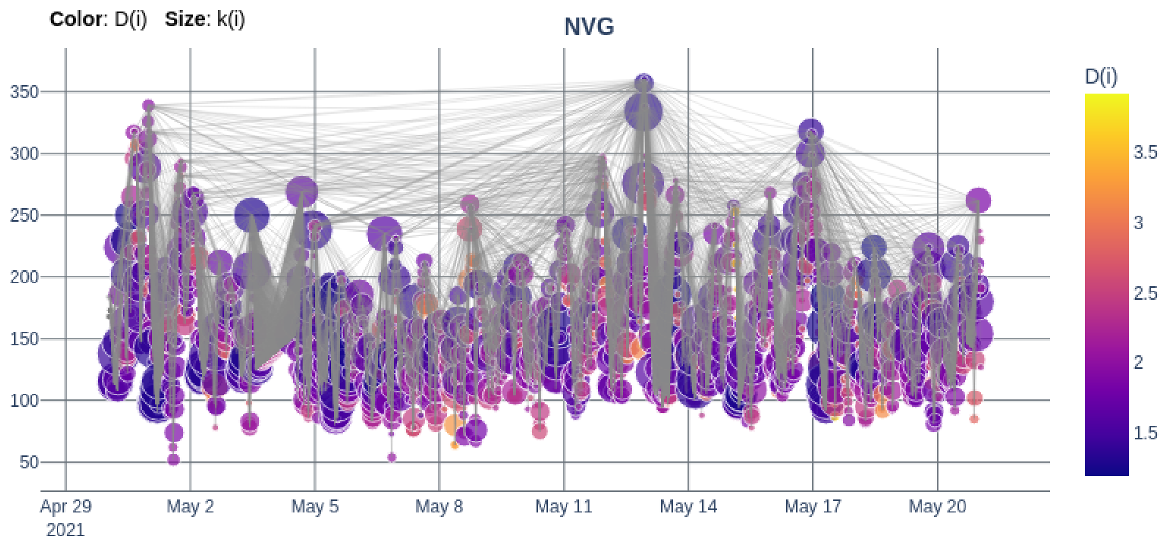


Figure A.1: NVG of the FJ Glucose Values

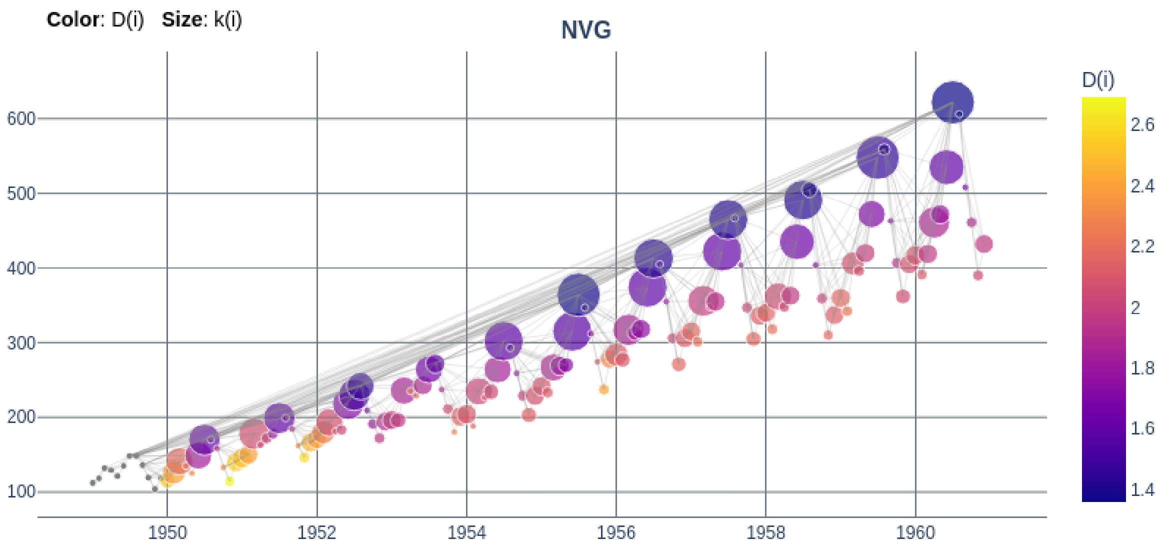


Figure A.2: NVG of the monthly air passengers

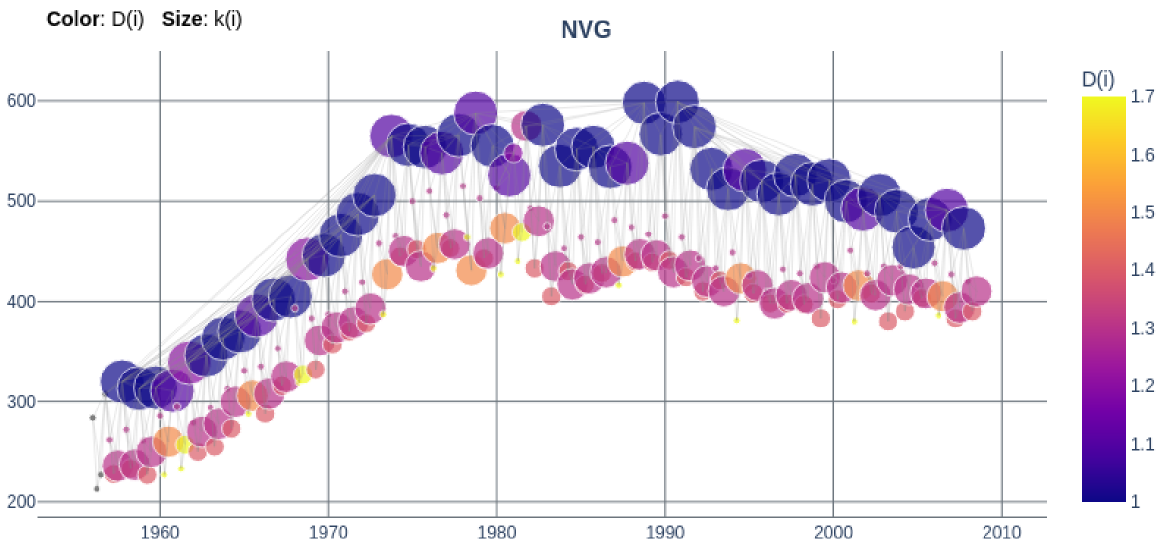


Figure A.3: NVG of the quarterly beer production in Australia

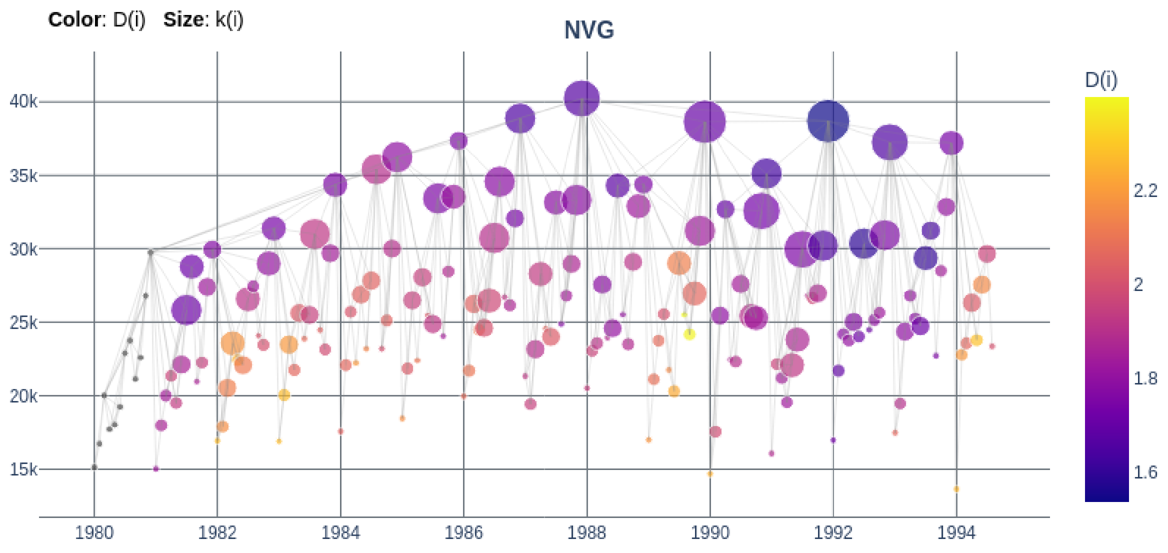


Figure A.4: NVG of the total monthly wine sales in Australia

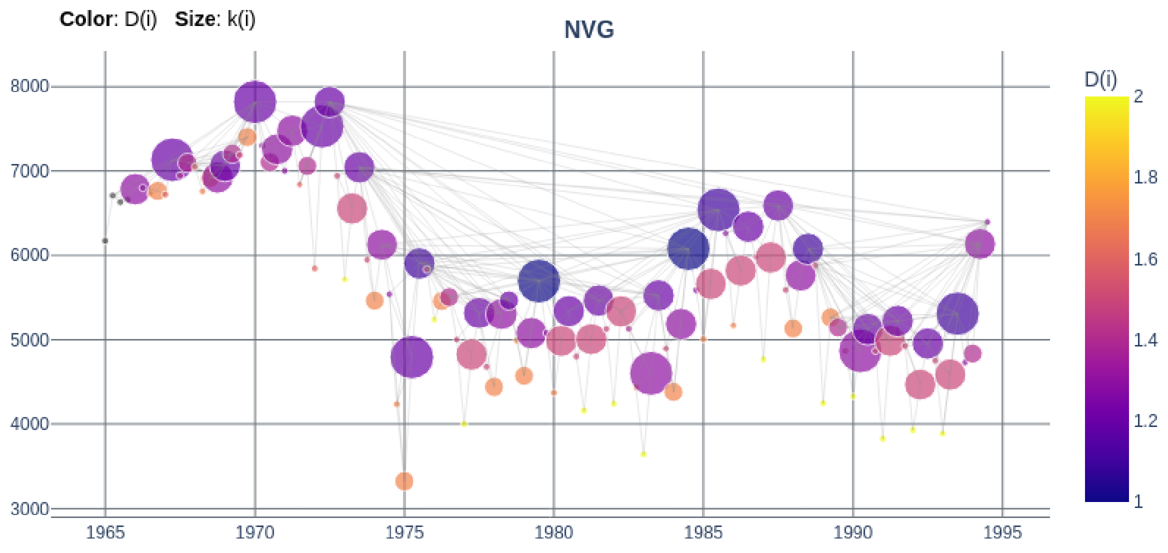
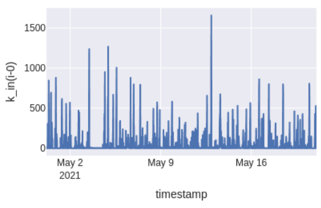
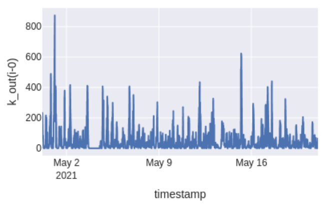


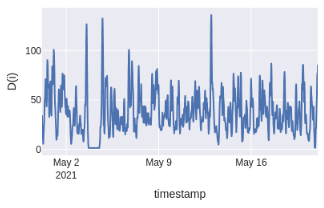
Figure A.5: NVG of the quarterly production of woollen yarn in Australia



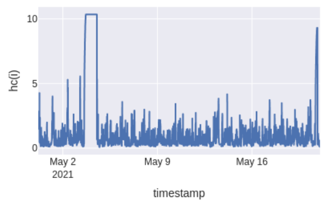
(a) $k_P^{in}(i)$ from a DWNVG ($w =$ d, $d =$ s)



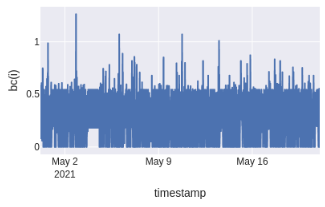
(b) $k_P^{out}(i)$ from a DWNVG ($w =$ d, $d =$ s)



(c) $D^{(}_{P}i)$ from a WNVG ($w =$ d)

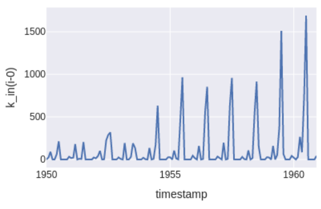


(d) $hc^{(}_{P}i)$ from a WNVG ($w =$ d)

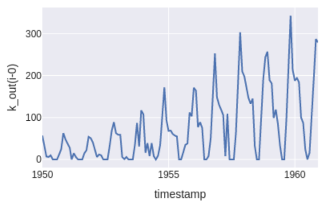


(e) $bc^{(}_{P}i - \frac{P}{2}$ from a WNVG ($w =$ d)

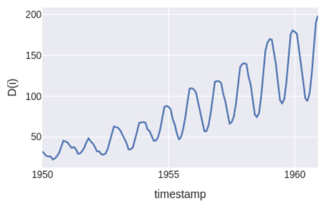
Figure A.6: Topological features encoded from FJ Glucose Values series



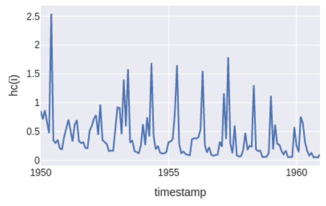
(a) $k_P^{in}(i)$ from a DWNVG ($w =$ d, $d =$ s)



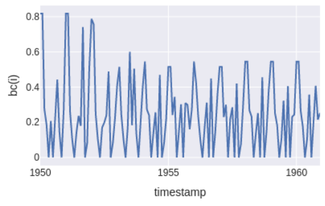
(b) $k_P^{out}(i)$ from a DWNVG ($w =$ d, $d =$ s)



(c) $D^{(}_{P}i)$ from a WNVG ($w =$ d)

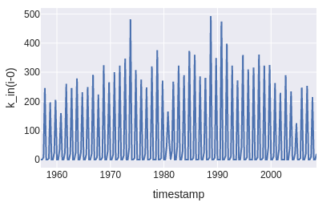


(d) $hc^{(}_{P}i)$ from a WNVG ($w =$ d)

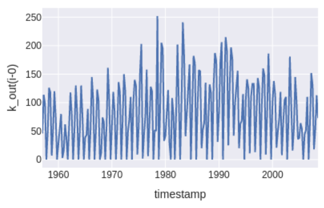


(e) $bc^{(}_{P}i - \frac{P}{2}$ from a WNVG ($w =$ d)

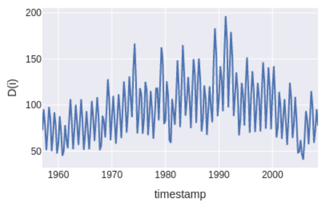
Figure A.7: Topological features encoded from monthly air passengers series



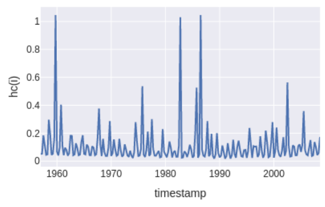
(a) $k_P^{in}(i)$ from a DWNVG ($w = \mathrm{d}$, $d = \mathrm{s}$)



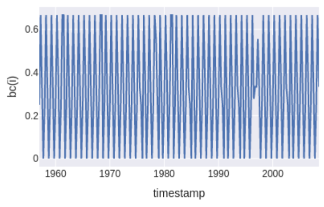
(b) $k_P^{out}(i)$ from a DWNVG ($w = \mathrm{d}$, $d = \mathrm{s}$)



(c) $D^{(}_{P}i)$ from a WNVG ($w = \mathrm{d}$)

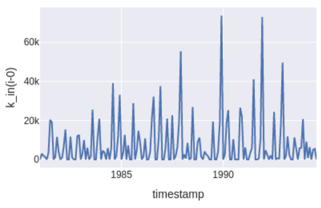


(d) $hc^{(}_{P}i)$ from a WNVG ($w = \mathrm{d}$)

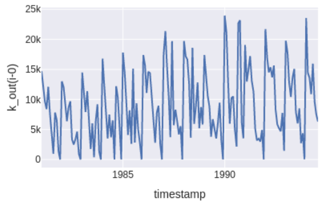


(e) $bc^{(}_{P}i - \frac{P}{2}$ from a WNVG ($w = \mathrm{d}$)

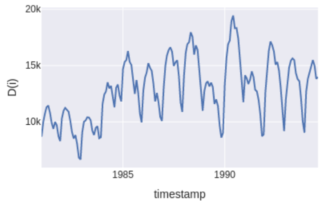
Figure A.8: Topological features encoded from quarterly beer production in Australia series



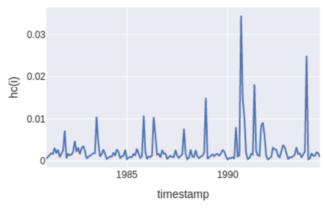
(a) $k_P^{in}(i)$ from a DWNVG ($w = \mathrm{d}$, $d = \mathrm{s}$)



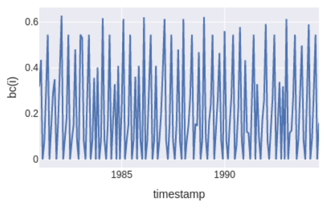
(b) $k_P^{out}(i)$ from a DWNVG ($w = \mathrm{d}$, $d = \mathrm{s}$)



(c) $D^{(}_{P}i)$ from a WNVG ($w = \mathrm{d}$)

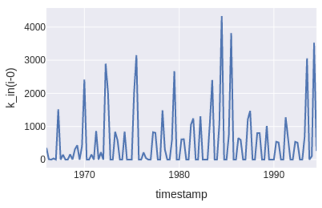


(d) $hc^{(}_{P}i)$ from a WNVG ($w = \mathrm{d}$)

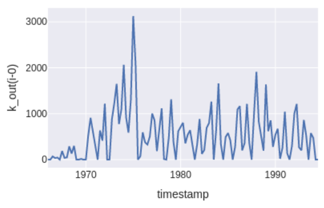


(e) $bc^{(}_{P}i - \frac{P}{2}$ from a WNVG ($w = \mathrm{d}$)

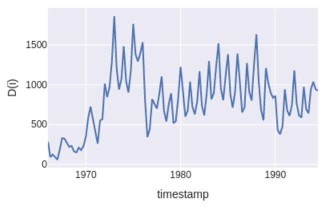
Figure A.9: Topological features encoded from total monthly wine sales in Australia series



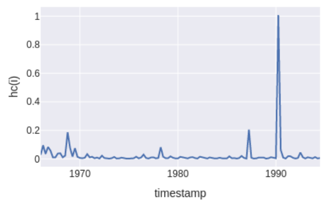
(a) $k_P^{in}(i)$ from a DWNVG ($w =$ d, $d =$ s)



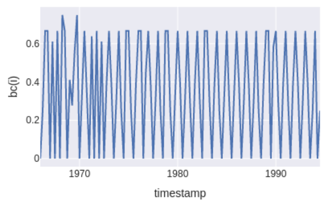
(b) $k_P^{out}(i)$ from a DWNVG ($w =$ d, $d =$ s)



(c) $D^{(}_{P}i)$ from a WNVG ($w =$ d)



(d) $hc^{(}_{P}i)$ from a WNVG ($w =$ d)

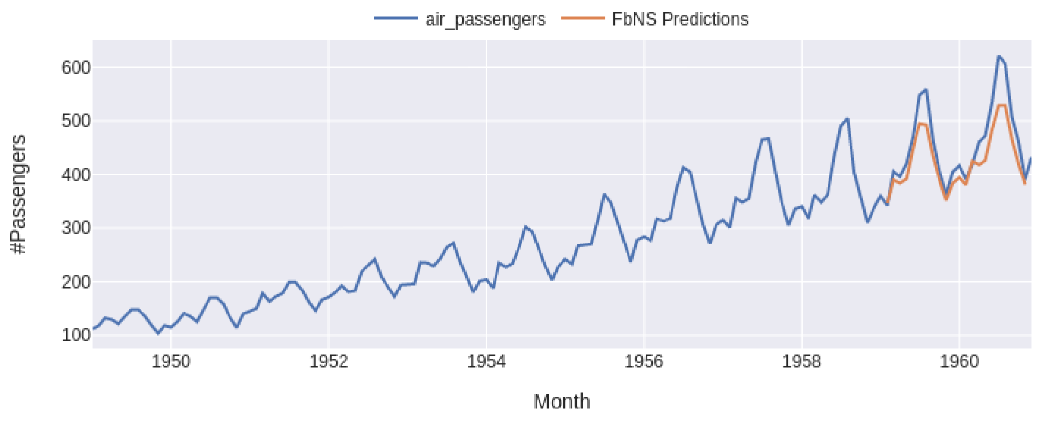


(e) $bc^{(}_{P}i - \frac{P}{2}$ from a WNVG ($w =$ d)

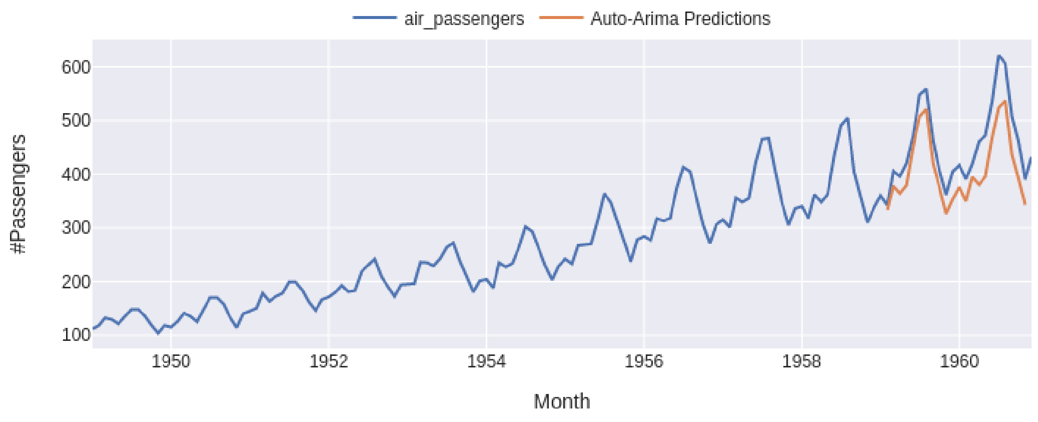
Figure A.10: Topological features encoded from quarterly production of woollen yarn in Australia series

Appendix B

Forecasting based on Networks Similarities



(a) FbNS 24-steps-ahead forecast



(b) Auto-Arima 24-steps-ahead forecast

Figure B.1: 24-steps-ahead forecasts of the total monthly air passengers series

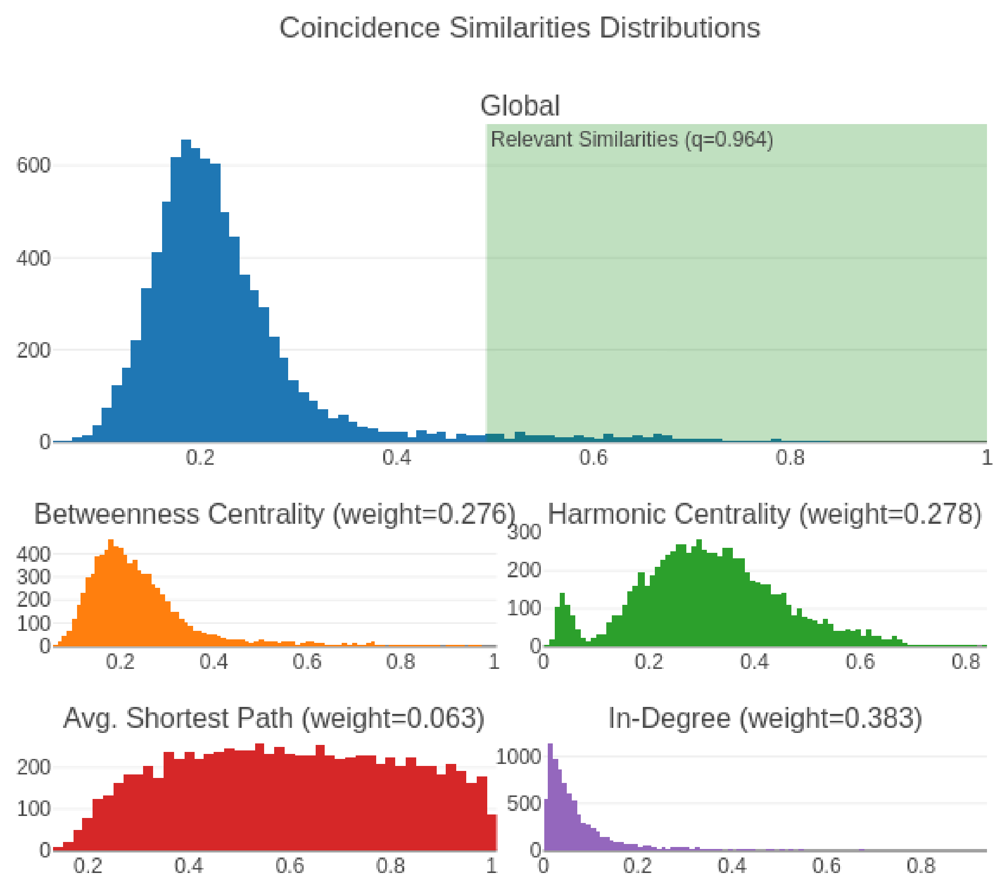
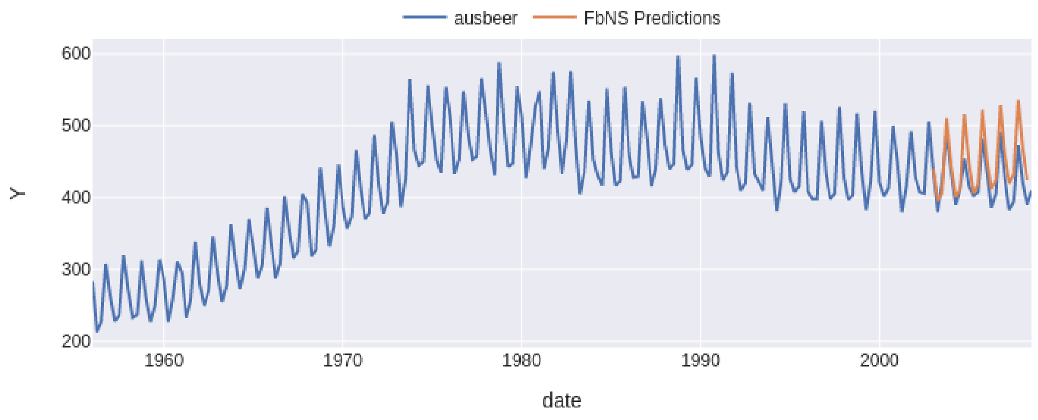
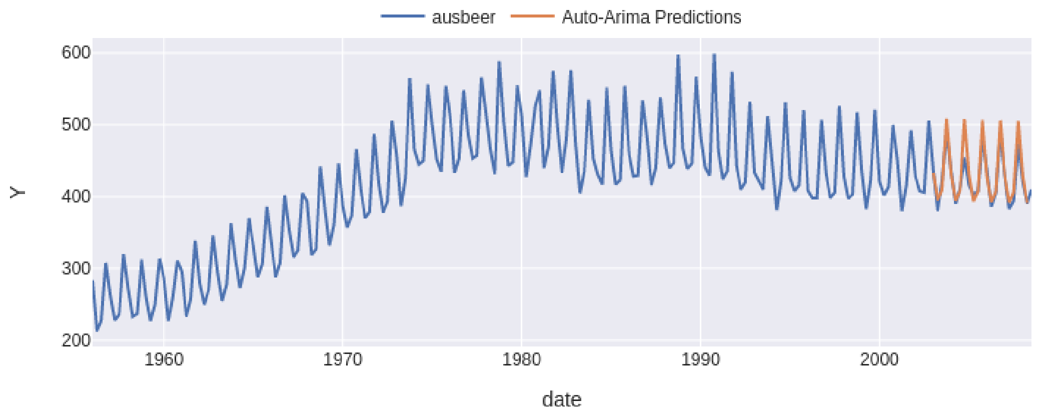


Figure B.2: Similarity distributions of the FbNS model in the monthly air passengers



(a) FbNS 24-steps-ahead forecast



(b) Auto-Arima 24-steps-ahead forecast

Figure B.3: 24-steps-ahead forecasts of the total quarterly beer production in Australia series

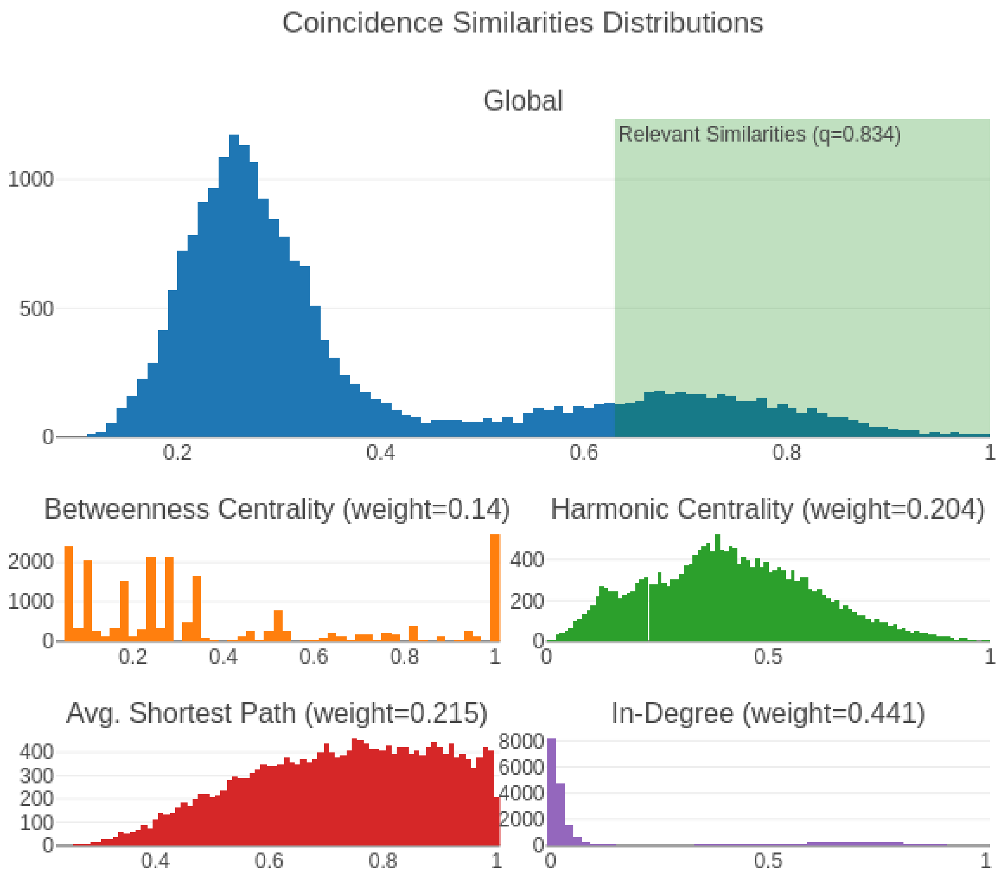
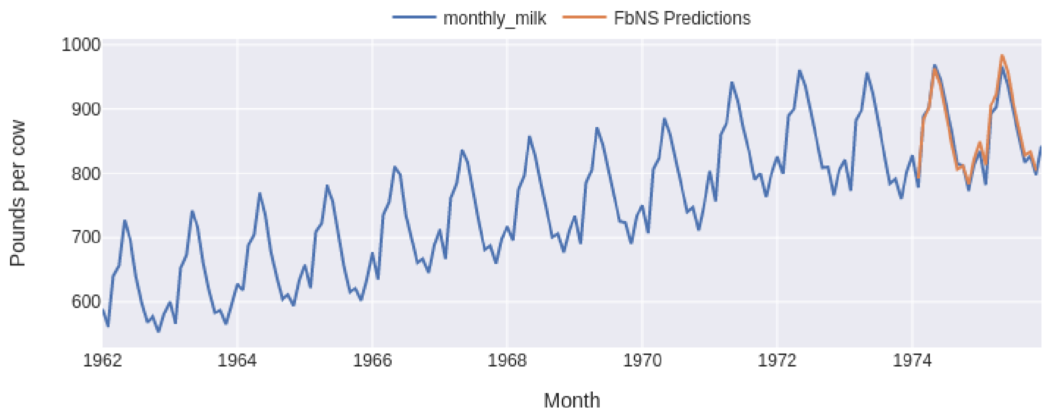
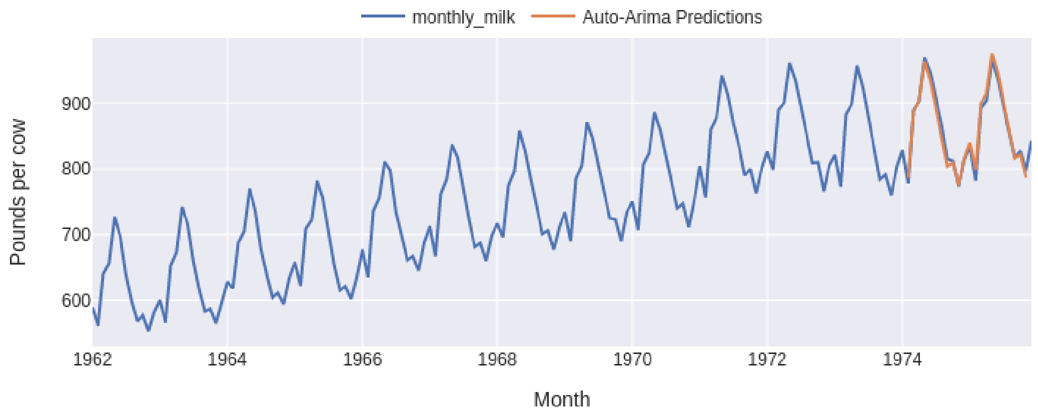


Figure B.4: Similarity distributions of the FbNS model in the quarterly beer production in Australia



(a) FbNS 24-steps-ahead forecast



(b) Auto-Arima 24-steps-ahead forecast

Figure B.5: 24-steps-ahead forecasts of the total monthly production of milk per cow series

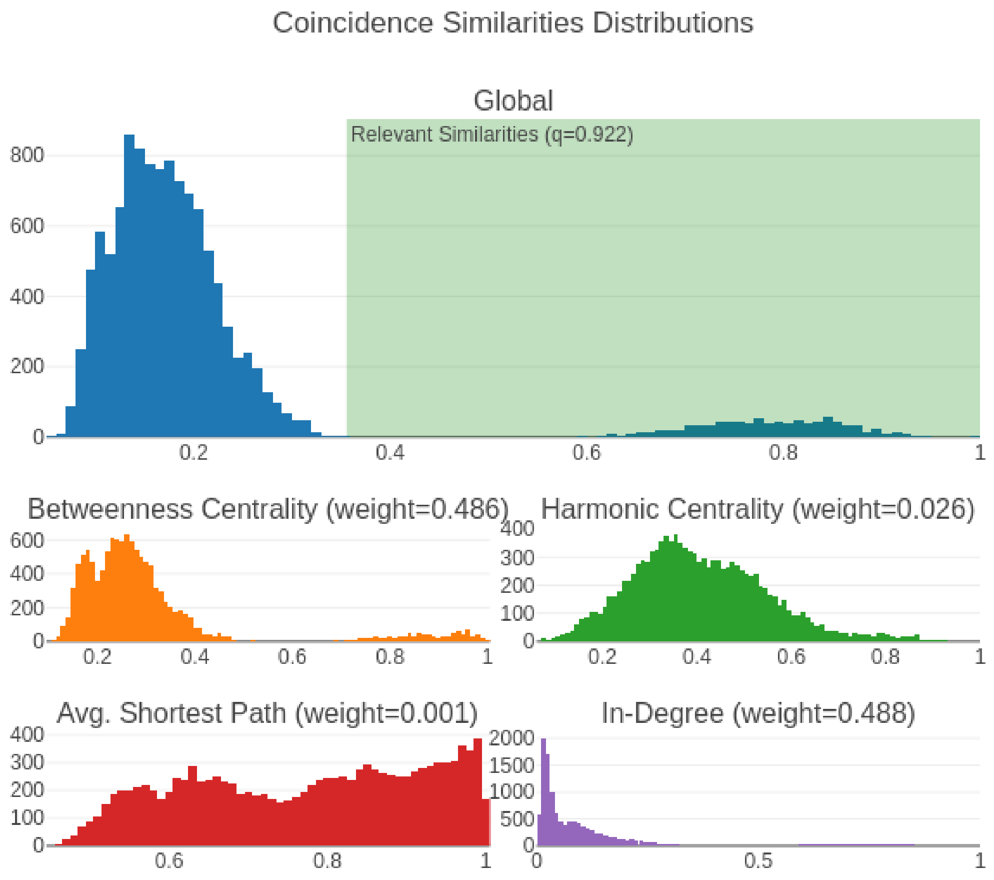
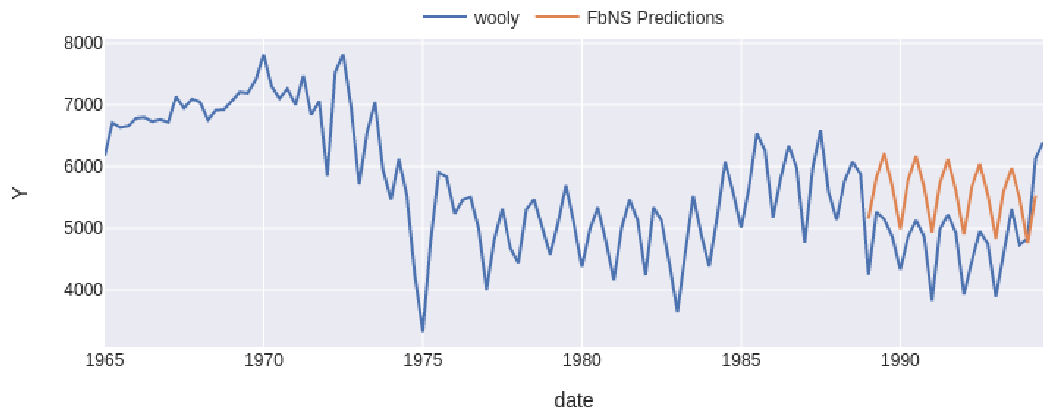
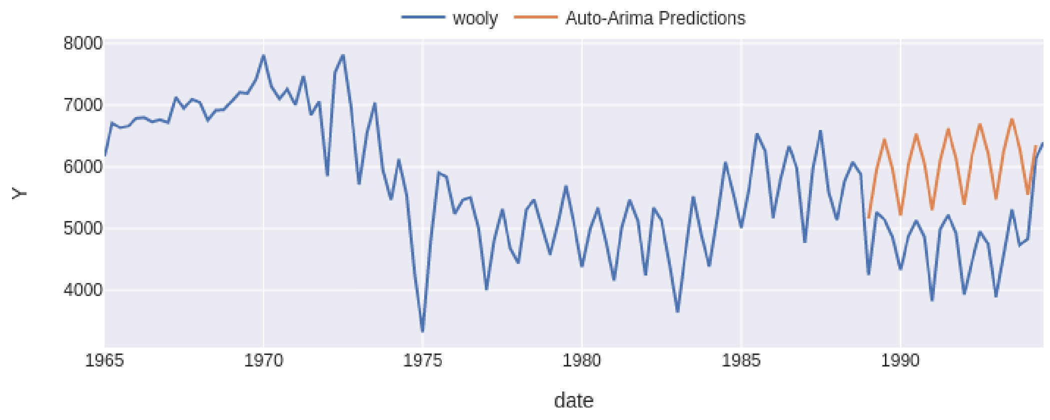


Figure B.6: Similarity distributions of the FbNS model in the monthly production of milk per cow



(a) FbNS 24-steps-ahead forecast



(b) Auto-Arima 24-steps-ahead forecast

Figure B.7: 24-steps-ahead forecasts of the total quarterly production of woollen yarn in Australia series

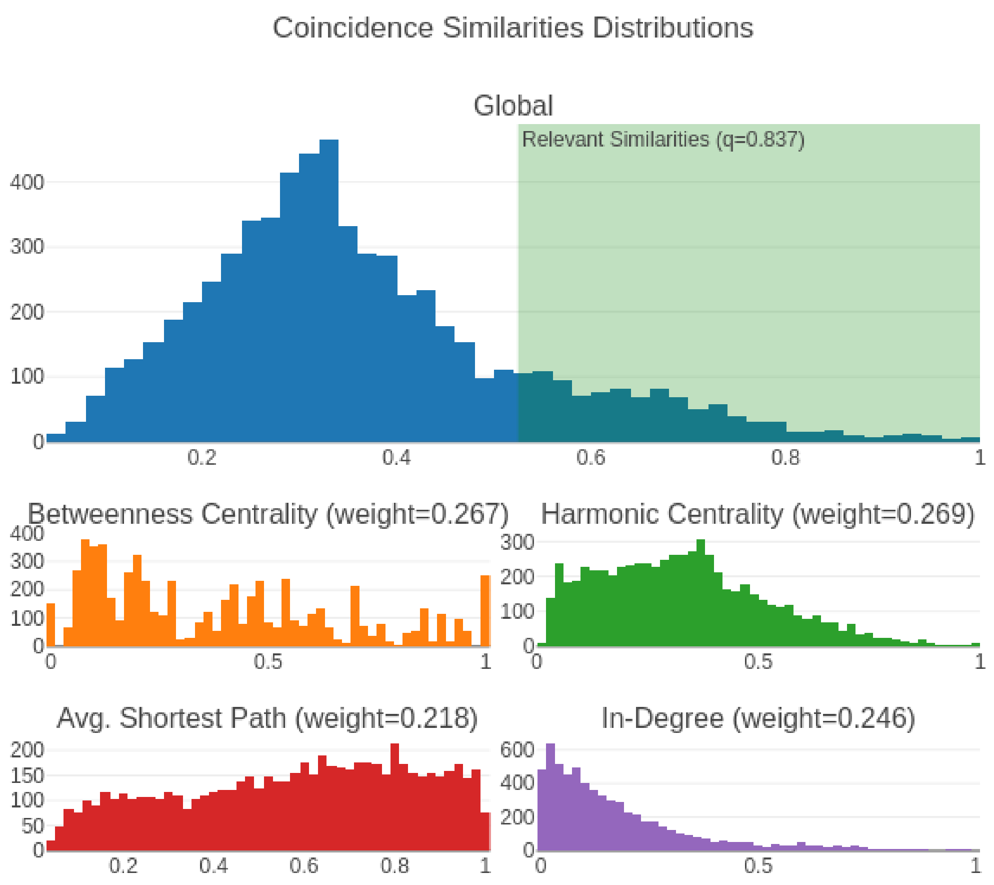


Figure B.8: Similarity distributions of the FbNS model in the quarterly production of woollen yarn in Australia

Bibliography

[1] Albert-László Barabási. *Network Science.* Cambridge University Press, 2016.

[2] G. E. P. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–243, 1964. doi:https://doi.org/10.1111/j.2517-6161.1964.tb00553.x.

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

[4] L da F Costa, Francisco A Rodrigues, Gonzalo Travieso, and Paulino Ribeiro Villas Boas. Characterization of complex networks: A survey of measurements. *Advances in physics*, 56 (1):167–242, 2007.

[5] Jonathan D Cryer and Kung-Sik Chan. *Time Series Analysis With Applications in R.* New York: Springer, 2008.

[6] Luciano da F. Costa. On similarity. *Physica A: Statistical Mechanics and its Applications*, 599:127456, 2022. ISSN: 0378-4371. doi:https://doi.org/10.1016/j.physa.2022.127456.

[7] Jonathan F Donges, Reik V Donner, and Jürgen Kurths. Testing time series irreversibility using complex network methods. *EPL (Europhysics Letters)*, 102(1):10004, 2013.

[8] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Computing Surveys (CSUR)*, 45(1):12, 2012.

[9] Philip Hans Franses and Dick Van Dijk. *Non-linear time series models in empirical finance.* Cambridge University Press, 2000.

[10] Anjali Gautam and Vrijendra Singh. Parametric versus non-parametric time series forecasting methods: A review. *Journal of Engineering Science and Technology Review*, 13:165–171, 06 2020. doi:10.25103/jestr.133.18.

[11] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[12] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi:10.1038/s41586-020-2649-2.

[13] Yusheng Huang, Xiaoyan Mao, and Yong Deng. Natural visibility encoding for time series and its application in stock trend prediction. *Knowledge-Based Systems*, 232:107478, 2021. ISSN: 0950-7051. doi:https://doi.org/10.1016/j.knosys.2021.107478.

[14] Rob Hyndman and Yangzhuoran Yang. tsdl: Time series data library. v0.1.0. 2018.

[15] Rob J Hyndman. Moving averages., 2011.

[16] Athanasopoulos G Hyndman R.J. *Forecasting: principles and practice, 2nd edition.* OTexts, 2018.

[17] Plotly Technologies Inc. Collaborative data science. Montreal, QC, 2015. Plotly Technologies Inc.

[18] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R.* 08 2013. ISBN: 9781461471370.

[19] Mankirat Kaur and Sarbjeet Singh. Analyzing negative ties in social networks: A survey. *Egyptian Informatics Journal*, 17(1):21–43, 2016.

[20] Lucas Lacasa, Bartolo Luque, Fernando Ballesteros, Jordi Luque, and Juan Carlos Nuno. From time series to complex networks: The visibility graph. *Proceedings of the National Academy of Sciences*, 105(13):4972–4975, 2008.

[21] Xin Lan, Hongming Mo, Shiyu Chen, Qi Liu, and Yong Deng. Fast transformation from time series to visibility graphs. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 25(8):083105, 2015.

[22] Chuang Liu, Wei-Xing Zhou, and Wei-Kang Yuan. Statistical properties of visibility graph of energy dissipation rates in three-dimensional fully developed turbulence. *Physica A: Statistical Mechanics and its Applications*, 389(13):2675–2681, 2010.

[23] Fan Liu and Yong Deng. A fast algorithm for network forecasting time series. *IEEE Access*, 7:102554–102560, 2019. doi:10.1109/ACCESS.2019.2926986.

[24] Shengzhong Mao and Fuyuan Xiao. Time series forecasting based on complex network analysis. *IEEE Access*, 7:40220–40229, 2019. doi:10.1109/ACCESS.2019.2906268.

[25] Mark Newman. *Networks: An Introduction.* Oxford University Press, 2010.

[26] Zhi-Gang Shao. Network analysis of human heartbeat dynamics. *Applied Physics Letters*, 96(7):073703, 2010.

[27] Samuel S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52:591–611, 1965.

[28] Robert H Shumway and David S Stoffer. *Time series analysis and its applications.* Springer, 2017.

[29] Vanessa Silva, Maria Silva, Pedro Ribeiro, and Fernando Silva. Novel features for time series analysis: a complex networks approach. *Data Mining and Knowledge Discovery*, 36, 05 2022. doi:10.1007/s10618-022-00826-3.

[30] 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(3):e1404, 2021. doi:https://doi.org/10.1002/widm.1404.

[31] Supriya Supriya, Siuly Siuly, Hua Wang, Jinli Cao, and Yanchun Zhang. Weighted visibility graph with complex network features in the detection of epilepsy. *IEEE Access*, 4:6554–6566, 2016.

[32] William W. S. Wei. *Multivariate Time Series Analysis and Applications.* John Wiley & Sons, 2019.

[33] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. doi:10.25080/Majora-92bf1922-00a.

[34] Yue Yang, Jianbo Wang, Huijie Yang, and Jingshi Mang. Visibility graph approach to exchange rate series. *Physica A: Statistical Mechanics and its Applications*, 388(20):4431–4437, 2009.

[35] Rong Zhang, Baabak Ashuri, and Yong Deng. A novel method for forecasting time series based on fuzzy logic and visibility graph. *Advances in Data Analysis and Classification*, 11 (4):759–783, 2017. ISSN: 1862-5355. doi:10.1007/s11634-017-0300-3.

[36] Junyin Zhao, Hongming Mo, and Yong Deng. An efficient network method for time series forecasting based on the dc algorithm and visibility relation. *IEEE Access*, 8:7598–7608, 2020. doi:10.1109/ACCESS.2020.2964067.

[37] Guohun Zhu, Yan Li, and Peng Paul Wen. Analysis and classification of sleep stages based on difference visibility graphs from a single-channel eeg signal. *IEEE journal of biomedical and health informatics*, 18(6):1813–1821, 2014.