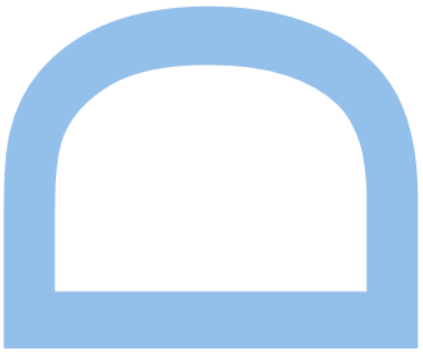
Multidimensional Time Series Analysis: A Complex Networks Approach

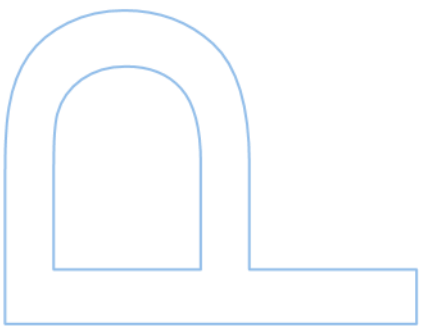
Vanessa Alexandra Freitas da Silva
Doctoral Program in Computer Science of the Universities of Minho, Aveiro and Porto (MAPi)
Computer Science Department
2023

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

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

Universidade do Minho

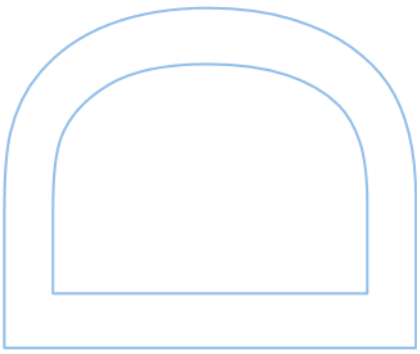


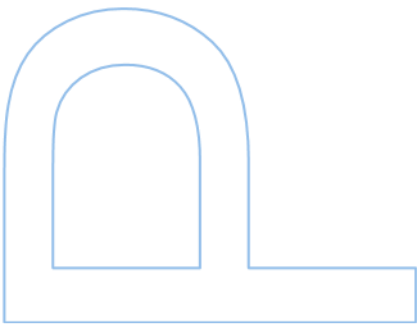


All the corrections determined by the jury, and only those, were made.

The Supervisor,

Porto, _____/_____/_____





Dedicated to Ricardo and my parents

Sworn Statement

I, Vanessa Alexandra Freitas da Silva, enrolled in the Doctor's Degree in Computer Science at the Faculty of Sciences of the University of Porto hereby declare, in accordance with the provisions of paragraph a) of Article 14 of the Code of Ethical Conduct of the University of Porto, that the content of this thesis reflects perspectives, research work and my own interpretations at the time of its submission.

By submitting this thesis, I also declare that it contains the results of my own research work and contributions that have not been previously submitted to this or any other institution.

I further declare that all references to other authors fully comply with the rules of attribution and are referenced in the text by citation and identified in the bibliographic references section. This thesis does not include any content whose reproduction is protected by copyright laws.

I am aware that the practice of plagiarism and self-plagiarism constitute a form of academic offense.

Vanessa Alexandra Freitas da Silva

16/01/2023

Acknowledgements

First and foremost, I would like to thank my advisors, who guided me through this wonderful journey. Over the years, they have always been there when I needed them, providing guidance, support, and a lot of patience. They proposed and discussed new ideas and encouraged me to pursue new challenges. That is why I am immensely grateful for all the enriching meetings and the intellectual freedom given to me, which made me become more autonomous and grow as a person. I want to thank Professor Pedro Ribeiro and Professor Maria Eduarda Silva for the guidance and interest they have shown in continuing to work with me. I am grateful for the enthusiasm and encouragement of Professor Pedro Ribeiro and the solicitude and dedication of Professor Maria Eduarda Silva. I want also to thank Professor Fernando Silva for his support and for encouragement to move forward and find new challenges. When I started my research journey, I had no expectations of doing a PhD, but through your advice, I followed this great adventure and for that I am grateful. These Professors are among the most passionate, dedicated, and motivational Professors that I ever had and I thank them for their help throughout this thesis and in my life. I truly feel indebted for all their help and trust.

I want to thank MAP-i's organization for providing me with a good research environment. This allowed me to get to know other Universities and meet other PhD students and experienced researchers, with whom I shared ideas and pains. I also want to thank the financial support of FCT that permitted me to be fully dedicated to pursuing this research [PhD grant SFRH/BD/139630/2018].

Finally, I would like to thank all my friends and family, who have always supported me. In particular, my parents, my brothers, my parents-in-law, and the love of my life, Ricardo. You kept me strong and sane, and your company is invaluable. I want to thank my parents for everything they do for me. Thank you for all the security, comfort, and love you give me. I cannot fail to thank my brother João Paulo. He was the primary motivator for me to follow the path of computer science, and with that being able to reach this point. That was the initial timestamp and I could not fail to mention it. Thank you also for all the tenderness and for being a constant inspiration. Thank you all for everything, without you it would not have been possible to complete this step of my life. I also want to thank Ricardo for being my strength, for listening to all my cries and despair, in moments of greatest insecurity, and for encouraging me and getting up again. He believed in my capabilities even before I did. Thank you for all your love, comfort, security, and stability. During this PhD we can give the first steps to complete our

dreams, starting a new step in our life, after of beautiful 10 years, and it was the best decision I ever made. I will always love you.

Abstract

Nowadays, we live surrounded by sensing devices, which collect and store data over time, measuring different and dependent variables. In univariate settings, time series analysis is already a well-established and mature field. However, in multivariate contexts, it still presents many limitations, due to their serial and cross dependencies and high-dimensionality. Recently, network science has been emerging as a promising approach to tackle these issues. These methods involve transforming an initial time series data set into one or more complex networks, which can then be analyzed in depth to provide insight into the original time series, but the majority of the existing work focuses on the univariate setting.

The overall goal of this thesis is precisely to advance time series analysis via network science, and in particular to find new and useful methods to study and understand complex temporal data using multidimensional network structures and their topological characteristics. Our first major contribution is comprehensive overview of the existing methodologies both on the univariate and multivariate setting, identifying their main characteristics and their differences and giving insight into their advantages and limitations in a unified notation and language. Our second major contribution is a proposal of two novel mapping methods that convert multivariate time series data into multilayer networks: the *multilayer horizontal visibility graph*, based on a new concept of cross visibility, and the *multilayer quantile graph*, based on the concept of transitions.

Being able to capture the properties of a time series with a feature vector is a crucial task. Here we propose to use features based on topological measures of the mapped networks. We first introduce the *NetF* for the univariate case, which we then extend to the multivariate case with *MNetF*. Both of these feature sets incorporate representative metrics from different mappings, and we provide an extensive exploratory study, giving insight and interpretability into what properties are being captured from the underlying time series. Furthermore, we apply the proposed features for clustering both synthetic and real data with very promising results, showcasing its general applicability. Our final contribution is *tsmnet*, a framework with parallel computation capabilities written in C++ that makes available all of the developed methods to any practitioner.

Keywords: Time Series, Complex Networks, Nonparametric Methods, Time Series Features

Resumo

Nos dias de hoje, vivemos rodeados de dispositivos móveis e sensores, que recolhem e armazenam dados ao longo do tempo, medindo variáveis diferentes e dependentes. Em contextos univariados, a análise de séries temporais é uma área bem estabelecida e sólida. No entanto, em contextos multivariados, ainda apresenta muitas limitações, devido às dependências seriais e cruzadas e à alta dimensionalidade dos dados. Recentemente, a ciência de redes tem emergido como uma abordagem promissora para lidar com essas questões. Os métodos envolvem a transformação de um conjunto de dados inicial de séries temporais em uma ou mais redes complexas, que podem ser analisadas a fundo para fornecer informações sobre as séries temporais originais, mas a maioria dos trabalhos existentes foca-se nos contextos univariados.

O objetivo geral desta tese é precisamente avançar na análise de séries temporais via ciência de redes e, em particular, encontrar métodos novos e úteis para estudar e compreender dados temporais complexos usando estruturas de redes multidimensionais e suas características topológicas. A nossa primeira grande contribuição é uma visão abrangente das metodologias existentes tanto no contexto univariado quanto no multivariado, identificando as suas principais características e diferenças e dando uma visão das suas vantagens e limitações em uma notação e linguagem unificadas. A nossa segunda maior contribuição é a proposta de dois novos métodos de mapeamento que transformam dados de séries temporais multivariadas em redes de múltiplas camadas: o *multilayer horizontal visibility graph*, que tem por base um novo conceito de visibilidade cruzada, e o *multilayer quantile graph*, que tem por base o conceito de transição.

Ser capaz de capturar as propriedades de uma série temporal com um vetor de características é uma tarefa crucial. Aqui propomos usar características baseadas em medidas topológicas das redes mapeadas. Primeiro introduzimos o *NetF* para o caso univariado, que então estendemos para o caso multivariado com o *MNetF*. Ambos os conjuntos de características incorporam métricas representativas de diferentes mapeamentos, e fornecemos um extenso estudo exploratório, fornecendo informações e interpretabilidade sobre quais propriedades são capturadas das séries temporais subjacentes. Além disso, aplicamos as características propostas para agrupar dados sintéticos e dados reais com resultados muito promissores, mostrando a sua aplicabilidade geral. A nossa contribuição final é o *tsmnet*, uma framework com capacidades de computação paralela escrito em C++ que disponibiliza todos os métodos desenvolvidos para qualquer profissional.

Palavras-chave: Séries temporais, Redes complexas, Métodos não paramétricos, Características de séries temporais

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

> Everything is theoretically impossible until it is done.
>
> *Robert A. Heinlein*

Multidimensional data has become, in recent years, predominant in various fields of research. Technological developments have led to the generation and collection of huge volumes of data associated with various real world systems, e.g., through the most diverse and advanced sensing technologies. These datasets typically exhibit an inherent temporal dimension that is crucial to their understanding and cannot be overlooked. The analysis of a single collection of observations indexed in time in a univariate setting is a mature and solid field, usually referred to as time series analysis (Shumway and Stoffer, 2017). There are several approaches to analyse time series, such as those stemming from statistics or from dynamic systems (Bradley and Kantz, 2015; Douc et al., 2014; Shumway and Stoffer, 2017), often tailored to data from specific areas and under several assumptions. Therefore, the analysis of univariate time series in many contexts (e.g. high frequency signals, non-stationarity) present, to this day and age, limitations. The data that are currently gathered, e.g., from large number of sensors which measure different variables and are positioned at different locations, are time indexed and high dimensional and leads to multivariate, spatio-temporal and panels of time series. The methods available to analyse these data sets are limited and have often been developed for specific areas and under several assumptions that hinder their wide application (Wei, 2019).

Given the multidisciplinary of multidimensional time series data and the growing interest from other research communities in analyzing these data, alternative approaches have emerged. This thesis focuses on a network-based approach for analyzing time series. The idea is to map the time series into a complex network, leveraging the large body of research in network analysis and providing new insights and novel angles on which to understand the structure of time series (Silva et al., 2021; Zou et al., 2019). Univariate time series are mapped into single-layer

networks based on concepts of visibility, transition, and proximity. Multivariate time series may be mapped into a single-layer network or a network of multiple layers. So far, the literature has been focused on mappings that result in single-layer network structures which imply significant losses of information from the original time series data. This thesis aims at bridging this gap in the literature. Three main challenges in analyzing time series via complex networks are covered in this thesis.

- Represent multidimensional time indexed data as a complex network capturing both serial and cross dimensional dependencies.
- Develop new tools for extracting features from univariate and multivariate time series.
- Develop a computational framework that implements time series analysis methods via complex networks.

1.1 Thesis Motivation

We can think of time series data as collections of observations indexed by time, which are ubiquitous in all domains from climate studies or health monitoring to financial data analysis. The main characteristic of a time series is the serial dependence between the observations, however, this restricts the applicability of many conventional statistical models and methods developed under the assumption of independent and identically distributed observations. Time series analysis seeks to develop a set of procedures and mathematical models, capable of systematically solving the statistical problems posed by the serial correlations and providing plausible descriptions of data characteristics with a view to forecasting and simulation (Box et al., 2015).

Multivariate time series present not only serial dependence within each variable but also interdependence between the different variables. Although the theory of univariate time series extends in a natural way to the multivariate case, new concepts inevitably arise (Wei, 2019). The analysis of multivariate time series requires tools, methods, and models for processing information contained in multiple variables that have both temporal and cross-sectional dependence which must be necessarily different from standard statistical theory and methods based on a random sample. Mathematical models for multivariate time series have often been developed tailored to model data from specific areas and under several assumptions. In recent years, procedures to analyze high dimensional time series have been developed but many issues remain open (see Wei, 2019 for more details). For example, the high dimension leads to limitations in computational and memory capacities that make the application of many statistics models difficult and impractical.

1.1. THESIS MOTIVATION

Recently, approaches based on statistics of time series data have gained attention from researchers in several areas. Such approaches have proved to be an important preliminary task in many applications of time series analysis (Fulcher and Jones, 2014; Hyndman et al., 2015; Kang et al., 2020; Montero-Manso et al., 2020) and an advantage for some of the challenges mentioned above, in particular for dimensionality reduction. Finding a set of features that summarizes the main characteristics of such data is therefore a crucial task, which usually involves conventional statistical and nonlinear measures of time series analysis (Montero-Manso et al., 2020). These methods usually involve parametric assumptions, parameter estimation, non-trivial calculations, and approximations, as well as preprocessing tasks such as finding time series components, differencing, and whitening. Computation problems related to the nature of the data, such as the length of the time series and dimension size, often appear and make such methods impractical.

Over the last decades, several techniques for extracting univariate time series features have been developed (see Barandas et al. 2020; Christ et al. 2018; Fulcher and Jones 2017; Fulcher et al. 2013; Hyndman et al. 2020; Lubba et al. 2019; O'Hara-Wild et al. 2021 for more details). Most of these techniques have in common the definition of a finite set of statistical features, such as autocorrelation, the existence of unit roots, periodicity, nonlinearity, and volatility among others, to capture the global nature of the time series. Although extending these techniques to describe and represent a multivariate time series dataset by means of global features is a promising plan, it is more complex and less trivial. The discriminatory features cannot just be about an individual time series, related to the serial dependence of the data, but can be about the interactions between the different components, related to the cross dependence of the data. The existing literature focuses mainly on correlation and causality measures between the individual time series components and more recent approaches focus on extracting features individually for each time series component, combining them into a single feature vector (Baldán et al., 2021). This can be done either by concatenating the resulting feature vectors, or calculating the correlation between the features in order to reduce the redundancy and the number of features. Although these approaches allow adapting existing features of packages for univariate time series to multivariate settings, the cross dependencies are not taken into account.

Many efforts in developing innovative methods to respond to the requirements mentioned earlier are being made in the data mining, machine learning, and network science fields. In fact, network science has available a vast set of topological features to characterize several properties of the structure of networks that can be from more trivial features to non-trivial features (Barabási, 2016; Costa et al., 2007). Furthermore, recent developments have led to the generation of more complex and high-level graph structures, which allow the modeling of multidimensional data without losing important properties, such as the different connections that can be intra and inter-dimension. These more complex structures are called multilayer networks and are complex structures capable of establishing internal connections (within the same layer) and also external connections (between different layers). Despite being a recent branch of network science, well-established methods and methodologies in the area can be easily extended and adapted to the new concept (Kivelä et al., 2014).

Univariate time series networks achieved several results, including theoretical results (Luque et al., 2009) and topological feature sets useful to time series problems (Silva et al., 2021; Zou et al., 2019). Different mappings methods capture different characteristics, which depend on the mapping construction. This can be shown in Figure 1.1 which illustrates different network structures obtained from a toy univariate time series using three different types of mappings proposed in the literature. Despite the range of existing methods and methodologies for the study of a univariate time series through networks, understanding the properties captured by the different mapping methods is not trivial. Furthermore, most works focus on the analysis of only one type of mapping method, limiting the information extracted from resulting networks to the chosen method. Research on multivariate time series mappings is still in its infancy, especially those that result in multilayer networks. Most focus on reducing multivariate time series into a single-layer network losing a set of data information that is pertinent to the analysis. Recent works use multiplex networks to analyze multivariate time series. Multiplex networks are a special type of multilayer network, which only have external connections between the same node present in different adjacent layers, ignoring possible direct external connections between different nodes.

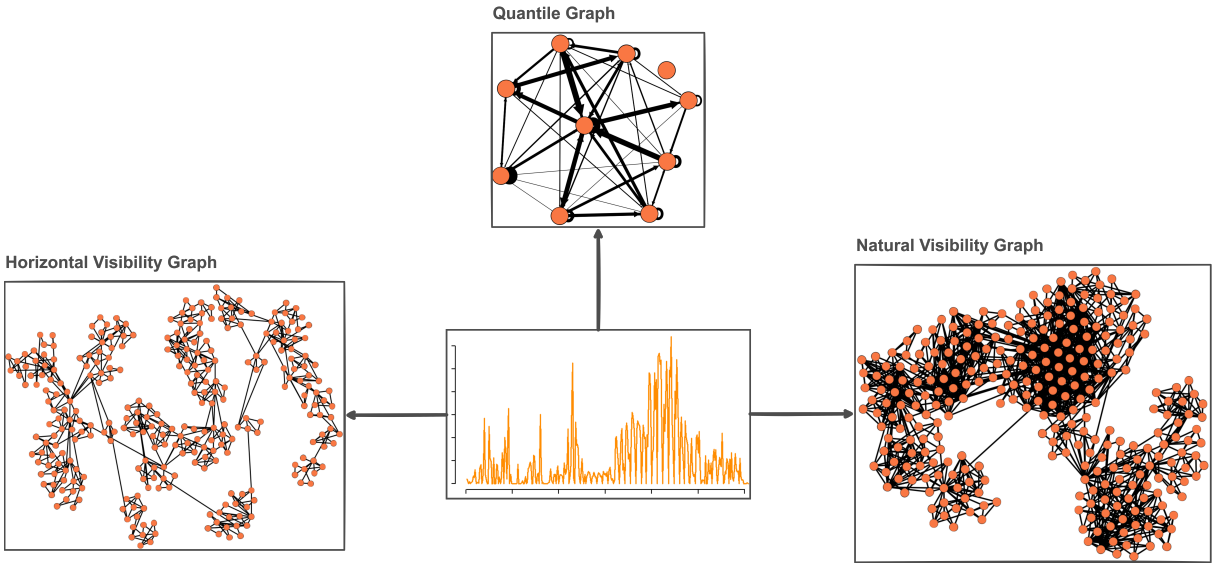


Figure 1.1: Schematic diagrams of different networks results from different mappings of univariate time series.

Although the literature presents a range of methods and methodologies for the study of time series through networks the focus has been mainly on univariate time series data. Therefore, studying multidimensional time series through these network methods and interpreting the characteristics captured by them still presents many gaps and challenges. Some of these challenges are as follows.

- **Univariate and multivariate time series features.** Feature-based approaches involve a set of procedures to describe the original time series by a set of features (or statistics, as it is better known in statistical literature) that describe different behaviors and characteristics of the data. These approaches have proven to be a powerful approach to help respond to the most diverse problems addressed in time series analysis, becoming more and more critical. There is a vast set of descriptive features to univariate time series data, including linear statistics and features based on nonlinear dynamics, symbolic representation, and other innovative features such as topological features. However, there is a sparse list of descriptive features in multivariate time series settings. The existing ones in the literature are adapted statistical measures from the univariate settings, or computed on each time series variable independently. In terms of global features (which describe the multivariate data through measures that describe the data as a whole), these kinds of features are very difficult (or even impossible) to find. Therefore, new features for multivariate time series need to be created.

- **Topological properties of time series networks.** In the literature, there is a large body of research using topological features to analyze time series data. However, few research works study and analyze the meaning of the resulting transformation, that is, the interpretability of features at the network and time series level. In fact, different topological features resulting from different mapping methods describe different (topological and dynamic) properties of the time series. Empirical studies using those mapping methods are still required to establish which mappings are most appropriate for a particular dataset or purpose.

- **High-dimensionality.** A very worrying and relevant issue in time series data is dimensionality. Time series data can be high-dimensional in both temporal dimension and variable dimension. With the advancement of technology, we can increasingly find this type of data, and statistical methodologies and methods, which normally involve parameter estimation, are not completely designed for high-dimensional variables. In fact, the number of parameters can exceed the length of the data. Common issues related to this topic are related to computation (time and memory), extraction of appropriate information, and visualization. More efficient versions of mathematical models of time series are required, which maintain equal (or better) results and reduce the number of parameters, or innovative methods that find new relationships and knowledge of multivariate datasets.

In this work, we propose two new mapping methods aimed at analyzing multivariate time series data via multilayer networks. We study high-level topological properties through features based on resulting multilayer graphs and we study these properties to respond to the above challenges.

1.2 Main Contributions

This thesis aims to use graph-based methods to develop new tools for time series analysis, in particular, new tools for time series feature extraction via mapping the time series (univariate and multivariate) into complex networks. Thus, this thesis involves two distinct but related main research fields: *time series analysis* and *network science*. Time series analysis consists on the study of time indexed data via statistical and mathematical methodologies. Network science is the study of complex systems via graph theory and topological features. Recent advances in the literature have provided several methods that unite these two areas around a common goal: to analyze complex data that present time-dependent properties. The overall purpose of this thesis is to find new useful methods to study and understand such data, with a particular interest in multidimensional time series data through multidimensional network structures and their topological features.

For univariate time series we propose a new set of features based on topological measures of networks obtained from different mappings. For multivariate time series, we extend the concept of visibility to pairs of time series and extend two univariate mappings to the multivariate context leading to two new multilayer networks. One, *Multilayer Horizontal Visibility Graph* (MHVG), involves a new concept of visibility, cross-horizontal visibility, between pairs of time series and the other, *Multilayer Quantile Graph* (MQG), extends the transition probability between patterns to the transition between patterns of different time series components. Figure 1.2 shows three different high-level networks obtained from mapping a toy multivariate time series according to the mapping methods proposed in this work that include cross dimensional connections between lagged data (timestamps or patterns). Additionally, we introduce a new set of network-based features for univariate time series, *NetF*, and propose a set of features for multivariate time series, *MNetF*. Both sets of features may be applied in time series mining tasks. To support and consolidate the proposed methodologies, we develop a flexible framework to implement the new mapping methods and the underlying mapping methods from the literature and to implement the various network topological features.

A detailed description of our contributions follows.

Survey The first contribution of this work is a thorough survey of the state of the art of time series analysis via network science. It provides a high-level conceptual division (at the level of the dimensionality and concomitant choice of the resulting network structure) of the existing mapping methods in the literature, as well as a comprehensive overview of these methods and their algorithms. It gives a quick overview table that highlights the differences between the strategies of the different methods and introduces a set of figures illustrating the main mapping methods, using the same dataset, with the purpose of comparing the different strategies. For each method, the survey describes the characteristics of the data it captures and its main references and results.

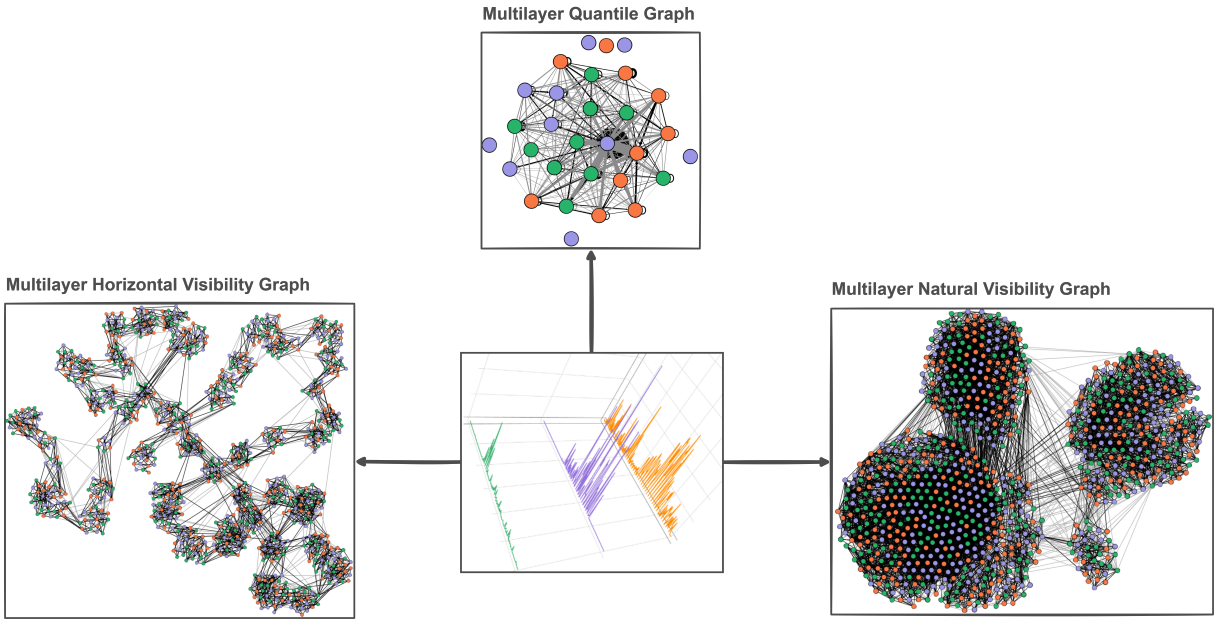


Figure 1.2: Schematic diagrams of different networks results from different mappings of multivariate time series. The nodes of different colors correspond to the different time series components data, the black lines (back edges) represent connections within dimensions and the gray lines (gray edges) represent connections between dimensions.

Multilayer time series networks The second contribution is the conception of the two new methods of mapping multivariate time series into a complete multilayer network structure described below. Conventional approaches mainly involve reducing the temporal dimensions in sets of nodes in a single-layer network, losing relevant information. And works that use multidimensional mappings, limit the mapping to only external connections between the same nodes of adjacent dimensions. To the best of our knowledge, multilayer time series networks that incorporate inter-layer edges between different entities (i.e., unique nodes) of the network have not yet been used in the literature.

MHVG: We propose a new mapping method called *multilayer horizontal visibility graph*. It is based on an extension of the horizontal visibility concept (Luque et al., 2009), which involves a new definition of visibility, the *cross-horizontal visibility*, and based on multiplex visibility graphs (Lacasa et al., 2015) to better utilize the structural capacity of multilayer networks,

MQG: The second mapping here proposed is based on a different mapping concept. It is called *multilayer quantile graph* and aims to extend the concept of quantile graphs (Campanharo et al., 2011) (for the univariate case), introducing edges between layers that capture the contemporaneous dynamic transitions between the different time series dimensions. MQG was designed with the main purpose of reducing the dimensionality of time series data with high dimensionality. The resulting multilayer networks have a smaller dimension, in comparison with the MHVG networks, and with the multivariate time series itself.

Topological features for time series networks. The third contribution is the development of methodological tools for the introduction of novel sets of descriptive features for both univariate and multivariate time series data. For univariate case, we propose to combine feature vectors resulting from different concepts of time series networks. Most of the previous works aimed at studying univariate time series from a single mapping method that captures particular characteristics underlying the concept of the chosen method, limiting data analysis to these characteristics. Our method allows the combination of different characteristics, which can enrich the data analysis. For multivariate settings, we propose a set of topological features for multilayer networks (described below) related to the substructures that we can extract from the network. We also combine different features originating from different multivariate mappings. As far as we know, no other work presents different topological features of different time series networks and of multilayer networks that take advantage of both intra- and inter-layer edges to (univariate and multivariate) time series analysis. In particular, we propose the following feature vectors.

***NetF*:** *NetF* is new set of network-based univariate time series features targeted at univariate time series mining tasks. The proposed procedure includes mapping the time series into (natural and horizontal) visibility graphs and quantile graphs, from which 5 global topological features are extracted (average weighted degree, average path length, number of communities, clustering coefficient, and modularity) for each type of graph, thus forming a set of 15 features, the *NetF*. *NetF* is always computable, which is not always possible using classic time series features.

***MNetF*:** *MNetF* is an extention of *NetF* for multivariate time series. We propose a new topological feature *ratio degree* that relates intra-layer and inter-layer connections of the nodes in the network, and we use extended versions of the common topological features (degree, path length, number of communities, and modularity) to the substructures of the multilayer networks, that is, topological features computed on both intra-layer and inter-layer edges, which we named *intra-layer features*, *inter-layer features*, *all-layer features*, and *relational features*. These topological features are computed from different multivariate time series mapping methods (the MHVG and MQG proposed in this work), forming the *MNetF*.

- We carry out a detailed exploratory and empirical analysis of the proposed mapping methods and topological feature sets on particular sets of univariate and multivariate time series models. The results show that different topological features capture different properties of the data complementing each other. In the multivariate case, we verified that the inter-layer edges between different nodes of different dimensions better capture the different properties of the series (intra-layer and inter-layer edges), when compared to the features on the individual layers (intra-layer edges). We observed that despite the dimensionality reduction, MQG mapping preserves the dynamic characteristics of the time series and between series during the mapping process. We compare the proposed methodologies with other benchmark feature sets on real (univariate and multivariate) time series clustering problems. Overall, the results

(for both *NetF* and *MNetF*, respectively) were balanced when compared to other features sets of benchmark, and when we combine *NetF* (or *MNetF*) with the respective benchmark features the results improve in some of the datasets.

tsmnet The fourth contribution of this thesis is the development of a computational framework that implements the mappings methods and the topological features proposed in this work and supports all the results obtained. The *tsmnet* is a framework in C++ designed and thought for the analysis of multidimensional time series via multilayer networks. It is capable of extracting (multilayer) time series networks resulting from univariate and multivariate mappings and allows extracting topological feature sets directed to perform classification and clustering tasks, for example. We make *tsmnet* available to practitioners.

1.3 Thesis Organization

This thesis is structured into eight major chapters described next.

Chapter 1 The present chapter presents the main context of this thesis, introducing the time series data problems, networks, and the mapping methods of time series analysis through networks. This chapter also describes the main motivations of our research, presenting the main challenges and our main contributions, which will be detailed in the following chapters. Finally, it presents an overview of the thesis contents and a bibliographic note.

Chapter 2 This chapter details the background knowledge necessary for this thesis. Introduces the common terminology in the time series analysis and network science literature that is used throughout the thesis, focusing on four relevant concepts for this thesis: univariate time series, multivariate time series, graphs, and multilayer networks. It also provides a brief description of time series statistical methods and time series data mining problems. A brief description of the topological features that serve as the basis for this work is also provided.

Chapter 3 This chapter corresponds to the published content of a paper in the form of a survey of the state of the art in which this thesis is inserted, plus the incorporation of more recent published works (Silva et al., 2021). It gives a detailed and structured depiction of the state-of-the-art methods for mapping time series in networks, complete with overview tables and figures illustrating the algorithms of the main mapping strategies. It also reviews the existing mapping approaches to convert (univariate or multivariate) time series data in (single-layer or multiple layers) network structure, describes their similarities and differences, the data characteristics captured, and the main results.

Chapter 4 Introduces the novel mapping methods for mapping multivariate time series data in a multilayer network, giving the motivation behind it. Explains and gives pseudo-code for the associated methods, namely the transformation of multivariate data into multilayer structures and their inter-layer edges. It is divided into two parts, each describing the two novel methods proposed in this work. Parts of this chapter correspond to part of the work published in Silva et al. (2023).

Chapter 5 Introduces *NetF*, a set of global topological features, as a novel set of features for univariate time series analysis, which is based on several concepts of univariate mappings. An empirical analysis of time series properties that are captured by each feature of *NetF* is carried out, by analyzing the feature space. Additionally, performs cluster analysis of univariate time series, highlighting the advantage of combining different mapping methods. This chapter corresponds to part of the work in Silva et al. (2022).

Chapter 6 Introduces the set of global topological features of multilayer networks proposed in this work, namely: intra-layer topological features, inter-layer topological features, all-layer topological features, and relational topological features. Presents a new topological feature aimed at representing intra and inter-layer relationships. The topological features of empirical analysis of the set of topological features on the new mapping method based on the concept of cross-visibility. And finally, similarly to *NetF*, the multivariate version is introduced, that is, *MNetF*, as a new set of features for multivariate time series analysis. An empirical analysis of this feature set is also performed.

Chapter 7 Performs an exhaustive empirical analysis of proposed *NetF* and of *MNetF* feature vectors. Respectively, we perform a clustering analysis of a large set of synthetic multivariate and univariate time series models, as well as a clustering analysis of a large set of real-world and benchmark univariate and multivariate time series datasets, respectively. A comparison of the proposed approaches with two standard approaches to time series analysis using a feature vector of conventional statistics is presented. Parts of this chapter correspond to the experimental part of the works published in Silva et al. (2022) and in Silva et al. (2023).

Chapter 8 This chapter introduces the *tsmnet*, a framework in C++ developed in the course of this work. Presents the main data structures implemented and used for the development of this work, also presents the module corresponding to the implemented mapping algorithms (the new mappings proposed in this work and the existing base mappings in the literature), the module corresponding to the computation of topological features of multilayer networks, and the modules of utility functions and data reading and data writing of time series and networks.

Chapter 9 Summarizes the work and the contributions, discusses obtained results and points to future research directions.

1.4 Bibliographic Note

In this section, we summarize the publications associated with the content of this thesis.

Journal Papers

- Vanessa Freitas Silva, Maria Eduarda Silva, Pedro Ribeiro, and Fernando Silva. *MHVG2MTS: Multilayer Horizontal Visibility Graphs for Multivariate Time Series Analysis*. Submitted to Transactions on Knowledge Discovery from Data, December 2022. A first version is available at arXiv https://arxiv.org/abs/2301.02333.
- Vanessa Freitas Silva, Maria Eduarda Silva, Pedro Ribeiro, and Fernando Silva. *Novel features for time series analysis: a complex networks approach*. Data Mining and Knowledge Discovery, March 2022.
- Vanessa Freitas Silva, Maria Eduarda Silva, Pedro Ribeiro, and Fernando Silva. *Time series analysis via network science: Concepts and algorithms*. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, e1404, Wiley, March 2021.

Talks

- Vanessa Freitas Silva, Maria Eduarda Silva, Pedro Ribeiro, and Fernando Silva. *Are multilayer networks useful for mining multivariate time series?*. In JOCLAD2023 - Book of Abstracts - Viana do Castelo, 20-22 April, April 2023.
- Vanessa Freitas Silva, Maria Eduarda Silva, Pedro Ribeiro, and Fernando Silva. *Multivariate Time Series Feature Extraction via Multilayer Networks*. In IFCS 2022 - Book of Abstracts - Porto, 19-23 July, July 2022.
- Vanessa Freitas Silva, Maria Eduarda Silva, and Pedro Ribeiro. *Time series analysis via complex networks: a first approach*. In JOCLAD2019 - Book of Abstracts - Viseu, 11-13 April, April 2019.

Chapter 2

Background

The purpose of this chapter is to introduce the reader to the background concepts and terminology necessary for understanding the remaining content of this thesis. This chapter consists of three parts. The first two present the reader with basic (univariate and multivariate) time series concepts and time series mining tasks. The third part introduces the reader to basic network science concepts emphasizing multilayer network structures and topological features.

2.1 Time Series Analysis

Time series analysis is the research area that studies observations indexed in time. In this section, we will present the basic concepts of univariate and multivariate time series analysis that are the basis for understanding this work.

2.1.1 Univariate Time Series

Formally, a *Univariate Time Series* (UTS) is a finite realization of a *stochastic process* which is a sequence of random variables indexed by time t, usually denoted by $\{Y_t\}_{t=1}^{T}$. A time series is discrete if $t \in \mathbb{Z}^+$ and is continuous if the time observations are taken continuously through time, that is, $t \in \mathbb{R}^+$. In this work, we consider that the observations are taken at discrete and equidistant time instants, that is, $t = 0, \pm 1, \pm 2, \ldots$. The main characteristic of a time series is the serial dependence between the observations that restricts the applicability of many conventional statistical methods developed under the assumption of independent and identically distributed (i.i.d.) observations. Time series analysis refers to the collection of procedures developed to systematically solve the statistical problems posed by the serial correlation. Thus, its main purpose is to develop mathematical models that provide plausible descriptions of data characteristics with a view to forecasting and simulation (Box et al., 2015).

The main statistical tools in time series analysis are the following measures of dependence. The *mean function* is defined as $\mu_t = \mathrm{E}(Y_t)$. The *autocovariance function* is defined as the covariance between the variables at times t and s:

$$\gamma(s,t) = \mathrm{cov}(Y_s, Y_t) = \mathrm{E}[(Y_s - \mu_s)(Y_t - \mu_t)], \tag{2.1}$$

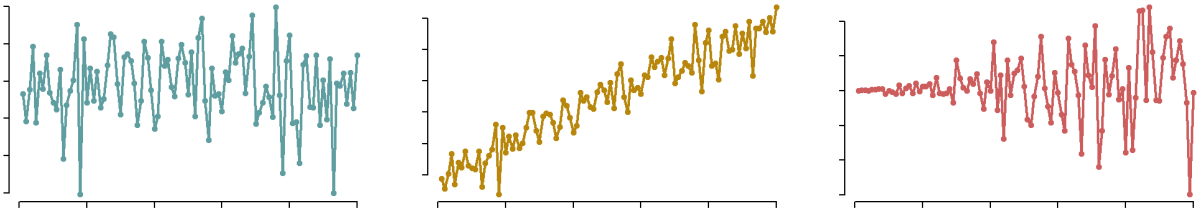
for all t and s. We should note that $\gamma(s,t) = \gamma(t,s)$ for all-time points s and t and $\gamma(t,t) = \mathrm{E}[(Y_t - \mu_t)^2] = \mathrm{var}(Y_t)$. The autocovariance measures the linear dependence between variables at different times. The *autocorrelation function* (ACF), which measures the linear predictability of the series at time t using only the variable at time s, Y_s, is defined as:

$$\rho(s,t) = \mathrm{corr}(Y_t, Y_s) = \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\gamma(t,t)}}, \tag{2.2}$$

$-1 \leq \rho(s,t) \leq 1$ with $\rho(s,t) = 1$ indicating a perfect positive correlation and $\rho(s,t) = -1$ a perfect negative correlation. Hence, the ACF provides a rough measure of the ability to forecast the series at time t from the value at time s (Shumway and Stoffer, 2017).

The statistical analysis of a time series relies on the concept of *stationarity*. Technically, we may distinguish strict (strong) stationarity from weak (wide or second-order) stationarity. A time series is said to be strictly stationary if the probabilistic behavior of every collection of variables $\{Y_1, Y_2, \ldots, Y_T\}$ is identical to that of the time-shifted set $\{Y_{1+h}, Y_{2+h}, \ldots, Y_{T+h}\}$. However, this condition is too strong for most applications and difficult to assess for a single dataset. Instead of imposing conditions on all possible distributions of a time series, weak stationarity only conditions the first two moments of the series. Thus, a time series is said to be weakly stationary if it is a finite variance process with a constant mean function, that does not depend on time, $\mathrm{E}(Y_t) = \mu$, and its autocovariance function depends only on the lag, $h = s - t, \gamma(s,t) = \gamma(s-t,0) = \gamma(h)$. In short, a time series is stationary when its statistical characteristics (mean and variance) are constant over time. That is, data fluctuate around a constant mean, with the variance of the fluctuations remaining essentially the same over time. An illustrative example of these characteristics for toy random time series is presented in Figure 2.1. Moreover, stationarity implies that the correlation between observations depends only on the time lag between them. Stationarity is essential in time series analysis since it enables the meaningful computation of statistics such as the mean, variance, and ACF from the data.

A simple time series process is the *white noise* (or purely random) process, henceforth denoted by ϵ_t. It is a sequence of i.i.d. random variables with mean 0 and constant variance σ_ϵ^2. A particular case is the *Gaussian white noise*, where ϵ_t are independent normal random variables. White noise is essential for time series analysis procedures.



(a) Stationary mean and variance (b) Non-stationary mean (c) Non-stationary variance

Figure 2.1: A illustration of a stationary toy time series in (a), and two non-stationary toy time series with non-constant mean and constant variance in (b) and constant mean and non-constant variance in (c).

Time series are seldom stationary presenting characteristics such as trend, seasonality and other cycles. In recent years several authors have proposed a feature based approach for time series analysis which has the advantage of reducing the dimensionality. Examples of statistical features include trend, seasonality, periodicity, autocorrelation, skewness, kurtosis, and heteroscedasticity (Wang et al., 2006). Furthermore, concepts such as self-similarity, nonlinearity structure, and chaos, derived from nonlinear science, are also used to characterize time series (Bradley and Kantz, 2015).

There is a plethora of (linear and nonlinear) statistical models in the literature adequate to describe the behavior of UTS (Shumway and Stoffer, 2017). *Linear time series models* are models for which the conditional mean is a linear function of past values of the time series. The most popular class is the AutoRegressive Moving Average (ARMA) model. Particular cases of ARMA models are white noise (WN), which is a sequence of i.i.d. observations, the AutoRegression (AR) models, which specify a linear regression between the current and past values, and the Moving Average (MA) models, which define a linear regression between the current value of the series and the past stochastic terms. ARMA models have been extended to incorporate non-stationarity (unit root) ARIMA models and long memory characteristics, ARFIMA models. However, many real-world time series data exhibit characteristics that cannot be described by linear models, examples include volatility, asymmetry, different regimes, and clustering effects. To model these effects, nonlinear specifications for the conditional mean and the conditional variance lead to classes of *nonlinear time series models*. The most common examples are Generalized AutoRegressive Conditional Heteroskedastic (GARCH) models, which are characterized by conditional variance and developed mainly for financial time series, the threshold and hidden Markov-based models, which allow modeling data from different regimes, and INAR models, which describe integer-valued data. Appendix B.1 presents additional details about these linear and nonlinear models, describing the models used in this work.

2.1.2 Multivariate Time Series

Often at each time t we can obtain a vector of m observations, $\boldsymbol{Y}_t = [Y_{1,t}, Y_{2,t}, \ldots, Y_{m,t}]'$, where $'$ represents the transpose of the vector. Then the dataset $\boldsymbol{Y} = \{\boldsymbol{Y}_t\}_{t=1}^{T}$ is called a *Multivariate Time Series* (MTS). Henceforward, the UTS components of the MTS $\boldsymbol{Y}$ are denoted by $\boldsymbol{Y}^{\alpha} = [Y_{\alpha,1}, Y_{\alpha,2}, \ldots, Y_{\alpha,T}]$, $\alpha = 1, \ldots, m$ and thus we can denote the MTS by its components, $\boldsymbol{Y} = \{\boldsymbol{Y}^{\alpha}\}_{\alpha=1}^{m}$. MTS data present not only serial correlation within each component, $\boldsymbol{Y}^{\alpha}$, but also correlation between the different UTS, $\boldsymbol{Y}^{\alpha}$ and $\boldsymbol{Y}^{\beta}$ with $\alpha \neq \beta$, both *contemporaneous* and *lagged* correlation. Thus, analyzing MTS depends on key dependence measures such as the ACF, which measures the linear predictability of a UTS, and the *cross-correlation function* (CCF), which measures the correlation between any two components of the MTS, α and β, say, at times s and t,

$$\rho_{\alpha,\beta}(s,t) = \text{corr}(Y_{\alpha,s}, Y_{\beta,t}). \tag{2.3}$$

Figure 2.2 shows, as an example, a bivariate autoregressive process and the respective plots of the ACF and CCF values for up to a lag of 15. We can see that the MTS is strongly correlated both serially and crosswise.

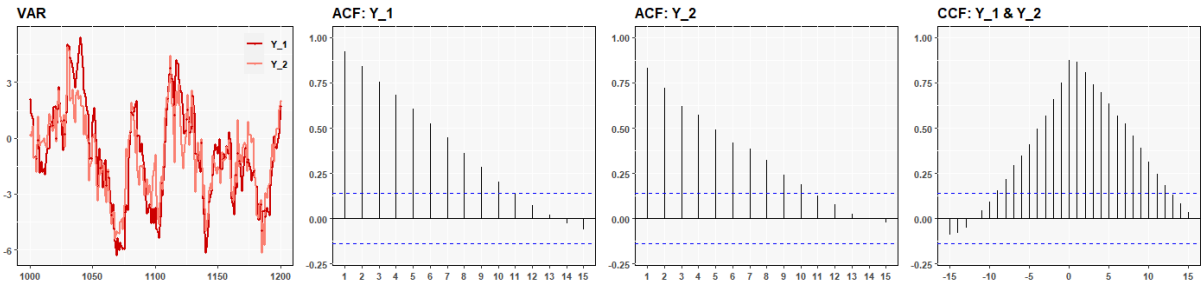


Figure 2.2: A bivariate autoregressive process and the corresponding ACF and CCF plots.

A m-dimensional time series process is stationary if each of its component time series is a univariate stationary process and the covariance matrix function depends only on the lag. Thus, $\text{E}(\boldsymbol{Y}_t) = \boldsymbol{\mu}$ is a vector of size $m \times 1$ of constants $\{\mu_\alpha\}_{\alpha=1}^{m}$, and the autocovariance function is a matrix function defined for lag h as the matrix of size $m \times m$:

$$\boldsymbol{\Gamma}(h) = \text{cov}(\boldsymbol{Y}_t, \boldsymbol{Y}_{t+h}) = \text{E}\left[(\boldsymbol{Y}_t - \boldsymbol{\mu})(\boldsymbol{Y}_{t+h} - \boldsymbol{\mu})'\right], \tag{2.4}$$

with matrix elements $\gamma_{\alpha,\beta}(h) = \text{cov}(Y_{\alpha,t}, Y_{\beta,t+h})$. Note that $\boldsymbol{\Gamma}(h) = \boldsymbol{\Gamma}'(-h)$ and the element (α,β) of matrix $\boldsymbol{\Gamma}(0)$ is the contemporaneous correlation between $\boldsymbol{Y}^{\alpha}$ and $\boldsymbol{Y}^{\beta}$ (see Wei, 2019 for more details). Similarly, we can define the correlation matrix at lag h with elements $\rho_{\alpha,\beta}(h) = \text{corr}(Y_{\alpha,t}, Y_{\beta,t+h})$.

The simplest example of a m-dimensional vector process is the *vector white noise process*, $\{\boldsymbol{\epsilon}_t\}$, with mean vector $\mathbf{0}$ and covariance matrix function $\boldsymbol{\Sigma}$, where $\boldsymbol{\Sigma}$ is a symmetric positive definite matrix of size $m \times m$. The white noise components are serially uncorrelated, $\text{corr}(\epsilon_{\alpha,t}, \epsilon_{\alpha,s}) = 0$, for $t \neq s$ but can be contemporaneously correlated, $\text{corr}(\epsilon_{\alpha,t}, \epsilon_{\beta,t}) \neq 0$. The most common vector white noise process is the Gaussian white noise process, for which $\boldsymbol{\epsilon}_t$ follows a multivariate normal distribution (Wei, 2019).

Several measures and statistical methods from UTS analysis can naturally extend to MTS analysis. Linear and nonlinear models can be extended to MTS in vector form. We can highlight the Vector AutoRegressive Moving Average (VARMA) model, which is a combined formulation of Vector AutoRegressive (VAR) and Vector Moving Average (VMA) models. The VAR model represents each variable as a linear regression of the past values of itself and the past values of all the other variables. The VMA model defines a linear regression of the past stochastic terms of the variable and the stochastic terms of all other variables. Like the UTS models, the VARMA model can also be extended to incorporate non-stationarity models (ARIMA model) and long memory models (VARFIMA model). To study the volatility problem in nonlinear multivariate settings, the GARCH processes characterized by their conditional heteroscedasticity are extended to the vector GARCH (VGARCH) models following the principles similar to VARMA models. We present in Appendix C.1 additional details about the linear and nonlinear models used in this work.

Although the theory of UTS extends naturally to the multivariate case, such as the mean, covariance, correlation functions, and linear and nonlinear models, new concepts arise. MTS analysis requires tools, methods, and models for mining information from multiple dependent variables - i.e., variables with both temporal and cross-sectional correlations. Such a property implies methods necessarily different from the standard statistical theory and methodologies based on random samples that assume independence (Wei, 2019). As mentioned above, numerous studies have already resulted in measures and methods that try to solve these questions. Examples include introducing variance-covariance and correlation matrix functions, measuring causality between UTS components, and extending linear and nonlinear models to vector time series models that incorporate matrix dependency functions (Wei, 2019). However, such models are often designed tailor to model data from specific areas and under several assumptions. Vector time series models require high computational effort due to the number of parameters that must be estimated, the most suitable model for each dataset also needs to be identified, and the need for data preprocessing to resolve non-stationarity. All these problems increase with the increasing high dimension of multivariate time series.

2.2 Time Series Data Mining

Time series data is ubiquitous in all domains from climate studies or health monitoring to financial data analysis. Although technological advances have facilitated the availability of time-indexed data, they have also led to the dimensionality problems we mentioned above that are often designated as the curse of dimensionality. All these factors have led to a growing interest in this type of data from fields other than time series analysis. The data mining field is one of them and consists of a set of procedures for automatically extracting knowledge from large amounts of datasets using techniques such as data warehousing, visualization, machine learning, pattern recognition, and statistics. In recent years, several methodologies have been developed especially for mining time-indexed datasets (Esling and Agon, 2012), where adding a time

dimension to the usual databases introduces new challenges. Several of these methodologies are for answering the fundamental tasks of time series analysis such as classification (Fulcher and Jones, 2014), clustering (Wang et al., 2006), forecasting (Montero-Manso et al., 2020; Talagala et al., 2018), pattern detection (Geurts, 2001), motif discovery (Chiu et al., 2003), outlier or anomalies detection (Hyndman et al., 2015), visualization (Kang et al., 2017), and generation of new data (Kang et al., 2020), among others.

2.2.1 Time Series Feature Extraction

Broadly speaking, we can divide time series data mining approaches into three main categories. *Shape-based* approaches, which consist of methods that explore the raw data of the time series, more precisely, the shape of the trajectory of the data. The *model-based* approaches, consist of selecting a statistical time series model and estimating a set of parameters for this model. And the *feature-based* approaches, which consist of extracting a set of statistical characteristics that describe the time series. Recently, feature-based time series characterization has become a popular approach among time series data researchers and has proven to be helpful for a wide range of the temporal data mining tasks (Fulcher, 2018), as well as a promising approach to battle problems related to the high dimensionality of data (Wang et al., 2006). More specifically, in feature-based approaches, a time series dataset is formed by a collection of time series *samples* and each time series sample is represented by a vector of statistical characteristics. These characteristics vary from sample to sample and define the dataset. Usually, these characteristics are called *features* in the data mining literature and *statistics* in the statistics literature.

In the context of UTS, there are several software packages available for feature extraction (Henderson and Fulcher, 2021), the most popular examples are the `tsfeatures` (Hyndman et al., 2020) and `feasts` (O'Hara-Wild et al., 2021), two R software packages with 63 and 42 features, respectively, the `tsfresh` (Christ et al., 2018), `Kats` (Science), and `TSFEL` (Barandas et al., 2020), Python software packages with 1558, 40, and 390 features available, respectively, the `hctsa` (Fulcher and Jones, 2017; Fulcher et al., 2013), a Matlab software package of 7730 features, and the `catch22`, a package of 22 features (a subset of `hctsa` package) available in Matlab, R, Python, and Julia software's (Lubba et al., 2019). Combined, these packages provide a rich and very large set of features available for characterizing UTS, they include from basic statistical measures to linear and nonlinear autocorrelation measures, spectral quantities and wavelet decompositions, to more specific methodologies, such as measures adapted to seasonal, autoregressive and heteroskedastic processes, among other. All these packages try to take advantage of the various methodologies that have been developed for time series analysis and try to express them in the form of global features that describe the time series dataset. However, many of these measures are highly redundant within the same package and have overlapping measures between packages (Henderson and Fulcher, 2021).

Recent approaches aim to reuse the existing univariate time series feature extracting methodologies for multivariate time series feature extracting. Those consist of extracting features individually for each UTS component and combining the resulting feature vectors into a single feature vector (Baldán et al., 2021). This approach adapts the existing literature features for UTS mentioned above to the multivariate problems.

2.2.2 Clustering and Classification

Time series classification problems are one of the main topics currently addressed in temporal data analysis work, given their wide applicability. Real-world examples might include classifying healthy and unhealthy patients, predicting behaviors, categorization of photos, fake detection, and sentiment analysis, to name a few. A classification problem aims to build a predictive model that receives as input a set of data samples and a set of descriptive attributes for each of the data samples and returns a class label for each of the input samples. The predictive model is learned through a set of data samples and respective example attributes where the labels are known, this process is called *supervised learning*. In the context of UTS classification, predictive models receive a set of UTS and a set of attributes from each UTS and return the labels assigned to each UTS. While in the context of MTS classification, the input is a set of MTS and a set of attributes for each MTS and the return is the learned classes for each MTS (Ruiz et al., 2021). There is a wide range of methods available for UTS classification (see Bagnall et al. (2017) for a review), whereas for MTS data current approaches are essentially combinations of univariate models over each individual component of the MTS (see Ruiz et al. (2021) for a review of more recent works).

Another important time series mining task is the clustering problem, which, in a general sense, has a similar purpose to the classification problem, that is, it aims to group/classify a set of data samples. However, clustering methods do not receive a training subset to learn the model and are therefore known as *unsupervised learning*. Clustering methods aim to group similar data samples into the same group, called a cluster, and different data samples into different clusters, classifying all samples in an input dataset. The definition of similarity between samples is dictated by the used clustering methods that can be based on the distance between data samples and respective clusters, on the density in space of the data samples, and on intervals or statistical distributions of the data. We can say that the time series clustering problem is less explored in literature when compared to the classification problem. However, it is equally important in terms of application and does not require a training dataset, so recently research interest in this problem has increased (Maharaj et al., 2019). Several research fields can use clustering methods to respond to real-world problems, some examples include pattern recognition, information retrieval, bioinformatics, data analysis and understanding, and more recently data privacy.

Generally, we can have three categories of clustering time series (Aghabozorgi et al., 2015). *Whole time series clustering*, where the set of input data samples is the set of UTS and clustering is performed by calculating the similarity between the UTS, the *sub-sequence time series clustering*, where the set of samples are sub-sequences of contiguous observations from a UTS of way to group similar patterns, and the *time point clustering*, where the samples are the observations of a UTS joining observations similar in value and temporal distance.

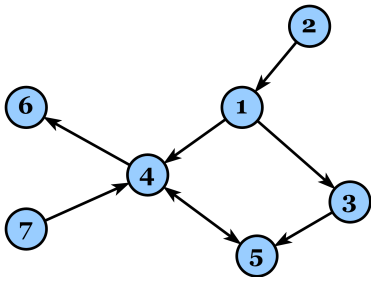
2.3 Network Science

Many real-world systems can be seen as complex systems with a set of elements that interact with each other and which exhibit emergent collective properties (Costa et al., 2011). *Graphs* provide a powerful and general abstraction for this kind of scenario, in which the elements are represented by *nodes* and the interactions by *edges*. Such graphs are often described as *complex networks*, since they typically exhibit non-trivial topological properties, due to the characteristics of the underlying complex system, which are neither random nor purely regular (Albert and Barabási, 2002).

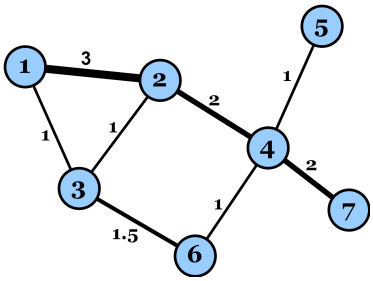
Network science is the research area that studies complex networks and has origins in discrete mathematics and graph theory. In the last decades, graph theory has become one of the most important tools in the analysis of networks. This approach allows representing complex systems in a unified mathematical form, making it possible to find common characteristics and properties. We use graph and network terms interchangeably in this document. In this section, we initially present the basic concepts of graph theory used in this thesis and then the more general concept of complex networks, the *multilayer networks*, ending with the presentation of topological features of networks explored in the course of this work.

2.3.1 Graph Terminology and Concepts

A *graph* (or *network*), G, is an ordered pair (V, E), where V represents the set of *nodes* (or vertices) and E the set of *edges* (or *links*, , or *arcs*) between pairs of elements of V. The number of nodes, also known as the *size* of the graph, is written as $N = |V|$ and the number of edges simply as $|E|$. Two nodes v_i and v_j are called *neighbors* (or *adjacent*) if they are linked/connected by an edge $(v_i, v_j) \in E$. We can distinguish between *directed* edges, which connect a source node to a target node, and *undirected* edges, when there is no such concept of orientation. In the first case the graph is called *directed* or *digraph*. A graph is termed *weighted* if there is a weight (or cost), $w_{i,j}$, associated the edge (v_i, v_j). It is classified as *simple* if it does not contain multiple edges (two or more edges connecting the same pair of nodes) and it does not contain self-loops (an edge connecting a node to itself). In the context of this thesis, all nodes are labeled with consecutive integer numbers starting from 1 to N. Figure 2.3 illustrates the concepts here defined and depicts a simple directed graph and a simple undirected weighted graph, respectively.



(a) Directed graph



(b) Undirected weighted graph

	1	2	3	4	5	6	7
1	0	0	1	1	0	0	0
2	1	0	0	0	0	0	0
3	0	0	0	0	1	0	0
4	0	0	0	0	1	1	0
5	0	0	0	1	0	0	0
6	0	0	0	0	0	0	0
7	0	0	0	1	0	0	0

(c) Non-symmetric binary adjacency matrix

	1	2	3	4	5	6	7
1	0	3	1	0	0	0	0
2	3	0	1	2	0	0	0
3	1	1	0	0	0	1.5	0
4	0	2	0	0	1	1	2
5	0	0	0	1	0	0	0
6	0	0	1.5	1	0	0	0
7	0	0	0	2	0	0	0

(d) Symmetric non-binary adjacency matrix

Figure 2.3: A representation of (a) a simple directed graph and (b) a simple undirected weighted graph. And the corresponding adjacency matrices in (c) and (d).

A graph G is usually represented mathematically as an *adjacency matrix* denoted as $\boldsymbol{A}$, and $A_{i,j}$ is 1 (or $w_{i,j}$) when $(v_i, v_j) \in E$ and is 0 otherwise. The corresponding adjacency matrices of the example graphs displayed in Figure 2.3 are illustrated in Figure 2.3. Note that undirected graphs result in symmetric matrices, while digraphs result in asymmetric matrices.

Most topological features are related to the concepts of *paths* and graph *connectivity*. A *path* is a sequence of nodes in which each consecutive pair of nodes in the sequence is connected by an edge. It may also be useful to think of the path as the sequence of edges that connect those nodes. Two nodes are *connected* if there is a path between them and are *disconnected* if no such path exists. A set of nodes is called a connected component if all its node pairs are connected.

The analysis of complex networks has made enormous progress in the last decade, leading to the development of statistical properties of certain types of complex systems that are related and codified in their topology (Albert and Barabási, 2002; Newman, 2003). There exists, therefore, a vast set of topological features that can characterize a network, each reflecting some specific property of the system under analysis. A more detailed list of the most important topological features can be found in the literature (Albert and Barabási, 2002; Barabási, 2016; Costa et al., 2007).

2.3.2 Multilayer Networks

Most real-world systems are not isolated and interact with other systems. Furthermore, a single entity may belong to several subsystems, establishing both internal (in the same system) and external (with other systems) interactions. The latter type of connectivity can be interpreted as an interaction between levels (or layers), leading to the concept of *multilayer networks*. In recent years, depending on the type of connections established between systems, several definitions and nomenclatures of multilayer networks had been proposed, until a universal mathematical definition is proposed in Kivelä et al. (2014) (see Kivelä et al. (2014) also for a more detailed review).

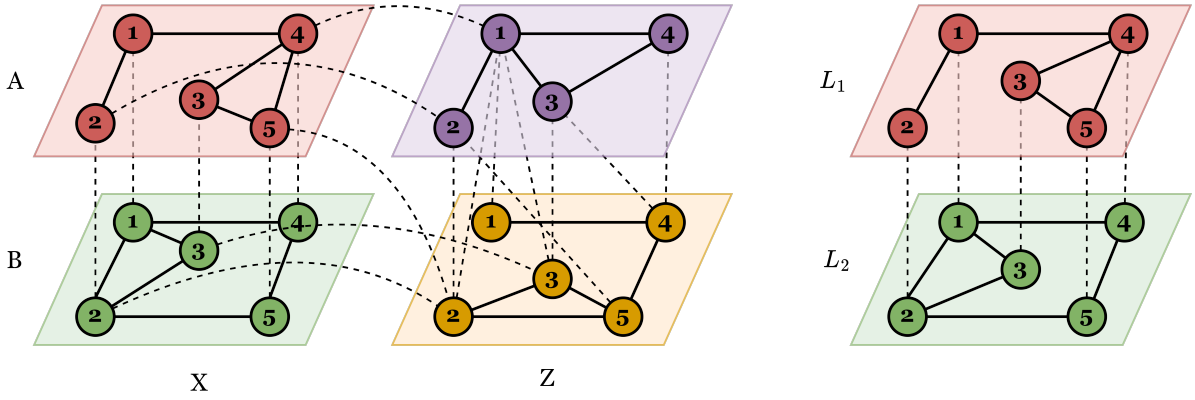
A *Multilayer Network* (MNet), also called multiple layer network, is defined as a quadruplet $M = (V_M, E_M, V, \mathcal{L})$ where $\mathcal{L} = \{\boldsymbol{L}_a\}_{a=1}^{d}$ are sets of elementary layers (there is one set of elementary layers $\boldsymbol{L}_a$ for each *aspect* a [1]), V is a set of entities of M, and V_M and E_M represent the global sets of nodes and edges, respectively. The $V_M \subseteq V \times \boldsymbol{L}_1 \times \ldots \times \boldsymbol{L}_d$ is a set of node-layer combinations in which a node is present in the corresponding layer. A node-layer specifies the role that an entity has on a specific layer, defining its occurrence on that layer. Therefore, we can have multiple nodes specifying the same entity in an MNet. The $E_M \subseteq V_M \times V_M$ is the set of edges that contains the pairs of possible combinations of nodes and elementary layers (Kivelä et al., 2014). A general simple representation of a multilayer network of two aspects can be seen in Figure 2.4a. In this work, we focus on single-aspect multilayer networks ($d = 1$) and, we denominate as *intra-layer edges*, the connections between nodes of the same layer L_α, (v_i^α, v_j^α), and *inter-layer edges* the connections between nodes of different layers L_α and L_β, (v_i^α, v_j^β) with $\alpha \neq \beta$.

Two particular cases of multilayer networks are the *monoplex network* when $d = 0$ and, consequently, $m = 1$ and the M reduces to a single-layer graph, G, and the *multiplex network*, when M is a sequence of m graphs, $\{G^\alpha\}_{\alpha=1}^{m} = \{(V^\alpha, E^\alpha)\}_{\alpha=1}^{m}$, usually with a node set common to all elementary layers, and inter-layer edges connecting only the counterpart nodes across the layers, that is connecting (v_i^α, v_i^β), $\alpha \neq \beta$ (Boccaletti et al., 2014). Figure 2.4b represents a simple multiplex network.

A node-aligned[2] MNet has an associated *adjacency tensor* of order $2(d+1)$, $\boldsymbol{\mathcal{A}}$, where the tensor element $\mathcal{A}_{i,j,\alpha,\beta}$ is 1 (or $w_{i,j}^{\alpha,\beta}$ for weighted MNet's) when $(v_i^\alpha, v_j^\beta) \in E_M$ and is 0 otherwise (Kivelä et al., 2014). If the MNet is not node-aligned, we can consider empty nodes to complete the tensor structure. Another representation is obtained by flattening $\boldsymbol{\mathcal{A}}$ into a *supra-adjacency matrix*, $\boldsymbol{A}$, where intra-layer edges are associated with diagonal element blocks and inter-layer edges with off-diagonal element blocks. Figure 2.5 represents this concept, displaying the supra-adjacency matrices of the networks illustrated in Figure 2.4. From these element blocks we can infer three types of subgraphs:

[1] We can generate a set of layers in a multilayer network by combining all elementary layers using the Cartesian product over the sequence of sets of elementary layers, $\boldsymbol{L}_1 \times \ldots \times \boldsymbol{L}_d$ (Kivelä et al., 2014).

[2] A multilayer network is node-aligned if all elementary layers contain all entities, that is, $V_M = V \times \boldsymbol{L}_1 \times \ldots \times \boldsymbol{L}_s$.



(a) Multilayer Network ($d = 2$)

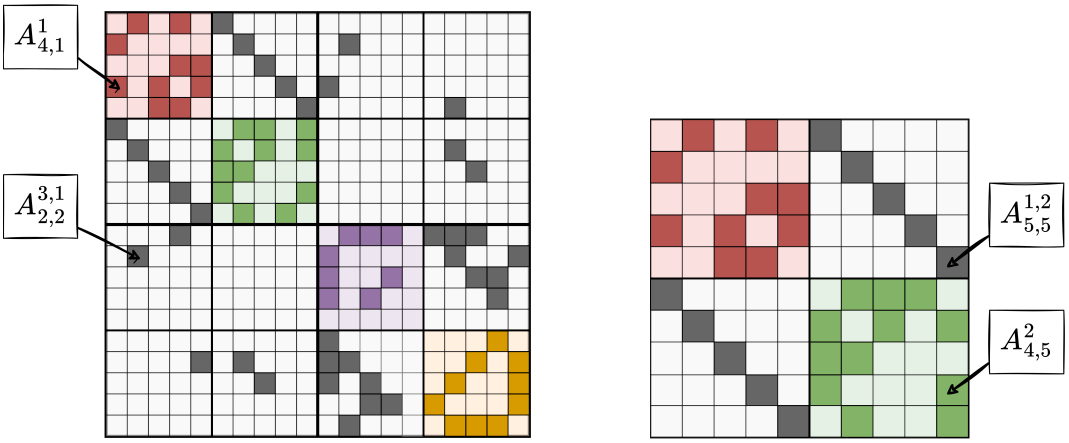
(b) Multiplex Network ($d = 1$)

Figure 2.4: (a) A toy example of the most general type of multilayer network. The network has five entities $V = \{1, 2, 3, 4, 5\}$ and two aspects, which have the corresponding elementary-layers sets $\boldsymbol{L}_1 = \{\mathrm{A}, \mathrm{B}\}$ and $\boldsymbol{L}_2 = \{\mathrm{X}, \mathrm{Z}\}$. There are thus a total of four different layers: $(\mathrm{A}, \mathrm{X}), (\mathrm{A}, \mathrm{Z}), (\mathrm{B}, \mathrm{X})$ and (B, Z). (b) An example of a toy multiplex network with five entities $V = \{1, 2, 3, 4, 5\}$ and one aspect. In both figures, solid lines represent intra-layer edges, and dashed lines represent inter-layer edges.

- *intra-layer graphs*, G^{α}, represented by the square matrices of order $|V^{\alpha}|$ formed by the diagonal element blocks (intra-layer edges, $A^{\alpha}_{i,j}$), ie., $\begin{bmatrix} A^{\alpha} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$,
- *inter-layer graphs*, $G^{\alpha,\beta}$, represented by the square matrices of order $|V^{\alpha}| + |V^{\beta}|$ constructed from off-diagonal element blocks (inter-layer edges, $A^{\alpha,\beta}_{i,j}$ and $A^{\beta,\alpha}_{j,i}$, and no intra-layer edges, $A^{\alpha}_{i,j} = 0$ and $A^{\beta}_{i,j} = 0$) [3], ie., $\begin{bmatrix} \mathbf{0} & \boldsymbol{A}^{\alpha,\beta} \\ \boldsymbol{A}^{\beta,\alpha} & \mathbf{0} \end{bmatrix}$, and
- *all-layer graphs*, $G^{\alpha,\beta}_{all}$, represented by the square matrices of size $|V^{\alpha}| + |V^{\beta}|$ constructed by both on and off-diagonal element blocks (intra-layer edges, $A^{\alpha}_{i,j}$ and $A^{\beta}_{i,j}$, and inter-layer edges, $A^{\alpha,\beta}_{i,j}$ and $A^{\beta,\alpha}_{j,i}$), ie., $\begin{bmatrix} \boldsymbol{A}^{\alpha} & \boldsymbol{A}^{\alpha,\beta} \\ \boldsymbol{A}^{\beta,\alpha} & \boldsymbol{A}^{\beta} \end{bmatrix}$.

The definition of MNet presented above is the most general definition, where the concept of *aspects* allows the modeling of real-world systems that can represent relations among various types. An illustrative example is the social networks, composed of Facebook, Instagram, and LinkedIn, where we can analyze different interactions between the networks, namely interactions between users through text messages, family interactions, and professional interactions between the users of the network. In this work, we will focus on single-aspect MNets.

[3] Note also that the inter-layer graphs have the characteristics of a bipartite graph. Where a bipartite graph is a graph $G^{\alpha,\beta}$ whose node set $V^{\alpha,\beta}$ can be divided into two disjoint and independent sets V^{α} and V^{β} ($V^{\alpha,\beta} = V^{\alpha} \cup V^{\beta}$ and $V^{\alpha} \cap V^{\beta} = \emptyset$) and every edge connects a node in V^{α} to a node in V^{β}.



(a) Multilayer Network (b) Multiplex Network

Figure 2.5: An illustrative example of two supra-adjacency matrices. (a) a supra-adjacency matrix of a toy multilayer network with two aspects (two layers in each) and (b) a supra-adjacency matrix of a toy multiplex network with a single aspect. Colored blocks represent the intra-layer edges and gray blocks represent the inter-layer edges.

Network science has been a very useful tool to answer the most diverse problems in several scientific fields (Vespignani, 2018), applying the vast arsenal of methodologies and topological features (Barabási, 2016; Peach et al., 2021) to data that are mapped directly or indirectly into networks. The topological features used in monoplex contexts can be extended to MNets. Most of the common features can be extended straightforwardly to *intra-layer features* by just computing them over the intra-layer edges. These features can also be extended to the whole MNet, computing them over both intra-layer and inter-layer edges, by transformation from flattening to supra-adjacency matrix (Huang et al., 2021a; Kivelä et al., 2014). Other approaches rely on measurements and properties in the tensor analysis literature (Kivelä et al., 2014). The analysis of multilayer networks is still a recent and not very mature topic, with most of the research being applied to networks with a single aspect (i.e., $d = 1$), and in the context of multiplex networks (Kivelä et al., 2014; Porter, 2018).

2.3.3 Network Topological Features

As mentioned above, a large number of topological, statistical, spectral, and combinatorial properties features that extract information from networks are available in the literature (Barabási, 2016; Peach et al., 2021; Silva et al., 2022). We can group these topological features into *global*, *local*, and "*intermediate*" features. The first group quantifies properties involving all network elements, the second properties over a given node or edge, and the last properties that involve subsets of the network, such as subgraphs. More common examples include features of node centrality (Oldham et al., 2019), graph distances (Li et al., 2021), clustering and community detection (Malliaros and Vazirgiannis, 2013), among others. Centrality features aim to quantify the importance of nodes and edges in the network depending on their connection topology.

Path-based features refer to sequences of edges that connect pairs of nodes, depend on the overall network structure, and are useful for measuring network efficiency and information propagation capability. Communities and node connectivity are also very relevant features of networks, which measure how and which groups of nodes are better connected, measuring the clustering and resilience of the network.

In this work, we are particularly interested in the following features frequently used in the network literature, which will serve as a base reference for the rest of the document thesis.

Degree The *degree* k_i of a node v_i is a quite important local feature of a graph, it represents the number of edges that the node v_i shares with other nodes v_j. In digraphs, we distinguish between *in-degree*, k_i^{in}, the number of edges that point to node v_i, and *out-degree*, k_i^{out}, the number of edges that point from node v_i to other nodes. And the total degree is given by the sum of the two, $k_i = k_i^{in} + k_i^{out}$. For weighted graphs, we can compute the *weighted degree* (or also called *strength*) by adding the edge weights instead of the number of edges, that is,

$$k_i = \sum_{j=1}^{N} A_{i,j}. \tag{2.5}$$

Degree shows the intensity of connectivity in the node neighbourhood. In many networks, there are some nodes that have a fairly high degree compared to others, these nodes are called *hubs*.

The *average degree*, $\bar{k}$, is a global feature related to node degree, and is obtained by calculating the arithmetic mean of the degrees of all nodes in the graph:

$$\bar{k} = \frac{1}{N} \sum_{i=1}^{N} k_i. \tag{2.6}$$

Path Length A path is a sequence of nodes in which each consecutive pair of nodes in the sequence is connected by an edge. In digraphs, the path follows the direction of from the source node to the target node. Paths can contain repeated nodes or not and paths without repeated nodes are called simple paths. The *path length* is defined by the number of edges in the path or the sum of the edge weights if the graph is weighted. *Average path length*, $\bar{d}$, is the arithmetic mean of the shortest paths, $d_{i,j}$, among all pairs of nodes in the graph:

$$\bar{d} = \frac{1}{N(N-1)} \sum_{\substack{i,j=1 \\ i \neq j}}^{|V|} d_{i,j}. \tag{2.7}$$

It is a good global feature of the efficiency of information flow on a network.

Clustering Coefficient The *global clustering coefficient*, C, also known as *transitivity*, is a feature that captures the degree to which the nodes of a graph tend to cluster, that is, the probability that two nodes connected to a given node are also connected. We can measure this property

through the total number of closed triangles in the graph. Then, C is given by the ratio of the number of closed triangles ($N_{\triangle}$) to the number of possible triangles (N_3):

$$C = \frac{3N_{\triangle}}{N_3}. \tag{2.8}$$

The constant factor 3 explains the fact that each triangle can be seen to consist of three different triangles, one with each of the nodes as the central node, guaranteeing $0 \leq C \leq 1$. In this work, we will call C simply as *clustering coefficient*.

Communities In this work, we refer to the set of communities in the network, $\mathcal{C}$, as the grouping of nodes (potentially overlapping) that are densely connected internally and that can also be considered as a group of nodes that share common or similar characteristics. The *number of communities* is the amount of these groups on the network, $S = |\mathcal{C}|$.

The *modularity*, Q, measures how good a specific division of the graph is into communities, that is, how different are the different nodes belonging to different communities. A high modularity value Q indicates a graph with a dense internal community structure, that is, with many edges between nodes within communities, and sparse connections between nodes of different communities. If a particular network is split into $\mathcal{C} = \{\mathcal{C}_i\}_{i=0}^{S}$ communities, Q can be calculated from as follows:

$$Q = \frac{1}{2|E|} \sum_{i,j} \left[A_{i,j} - \frac{k_i k_j}{2|E|} \right] \delta\left(\mathcal{C}_i, \mathcal{C}_j\right), \tag{2.9}$$

where $\delta\left(\mathcal{C}_i, \mathcal{C}_j\right) = 1$ if v_i and v_j belong to the same community (i.e., $\mathcal{C}_i = \mathcal{C}_j$) and $\delta\left(\mathcal{C}_i, \mathcal{C}_j\right) = 0$ otherwise.

These features can naturally be extended to a MNet structure and to its subgraphs mentioned in the above section. In the literature we can find the following approaches (Kivelä et al., 2014):

- **Flattening approaches**: as previously mentioned, it consists of translating the topological information of an MNet into a monoplex network defined by the supra-adjacency matrix. Traditional topological features can be calculated from this matrix. The main disadvantages pointed out are the potential loss of information related to the aspects, when it comes to the analysis of MNets with multiple aspects.
- **Aggregation approaches**: consists of creating a weighted monoplex network formed by the entity set of MNet. The inter-layer edges are ignored and the amount (or weights) of intra-layer edges between the same pair of entities in all layers is summed, forming the set of weighted edges of the aggregated network. From this structure, it is possible to calculate all common topological features. However, information encoded by inter-layer relationships is inevitably lost.
- **Direct approaches**: can be seen as reformulations and adaptations of topological features directed to the structure of the multilayer network, giving rise to new topological features of networks. In these approaches methodologies from the theory of tensor can be used.

Chapter 3

Time Series Analysis via Network Science: Concepts and Algorithms

> What is research but a blind date with knowledge?
>
> *Will Harvey*

The analysis of UTS is a mature and solid field (Shumway and Stoffer, 2017), which mostly involves statistical linear analysis. However, many observed datasets exhibit non-Gaussian and nonlinear characteristics and therefore several statistical non-Gaussian and nonlinear models have been proposed in the literature (Douc et al., 2014). As already mentioned in Section 2.1, these models have often been developed tailored to model data from specific areas and under several assumptions that hinder their wide application (Wei, 2019). In fact, in many settings both univariate (e.g. high-frequency signals, non-stationarity) and high dimensional (e.g. multivariate, spatio-temporal, panels of time series) time series analysis still presents many limitations. An alternative approach to time series data analysis has been developed under the framework of dynamical systems theory. This approach, denominated nonlinear time-series analysis (Bradley and Kantz, 2015), can be overly effective when the data model is based on deterministic dynamics in some state space.

In the last decade, several approaches for time series analysis based on network science methodologies were proposed, leveraging the large body of research in network analysis and providing new insight and novel angles on which to understand the structure of time series (Zou et al., 2019). This new framework for time series analysis inevitably involves mapping the time series into networks. In this chapter, we analyze the state-of-the-art of existing mapping methods for this purpose. Our main contribution is to produce a general and structured review of existing mapping algorithms, highlighting their similarities and differences, the data characteristics they capture, and the main references and results. We propose here a conceptual division of the

methods at the level of the time series dimensionality and concomitant choice of the resulting network structure. From our point of view, this high-level conceptual division encompasses all existing methods in the literature, for both univariate and multivariate frameworks.

The proposed high-level conceptual division is depicted in the diagram of Figure 3.1, and corresponds to the organization of this chapter. We first divide the mappings into two large groups, depending on the source dataset is a univariate or a multivariate time series. The first group results in single-layer networks, while multivariate series can be mapped both in single-layer and multiple layer networks (such as multiplex networks).

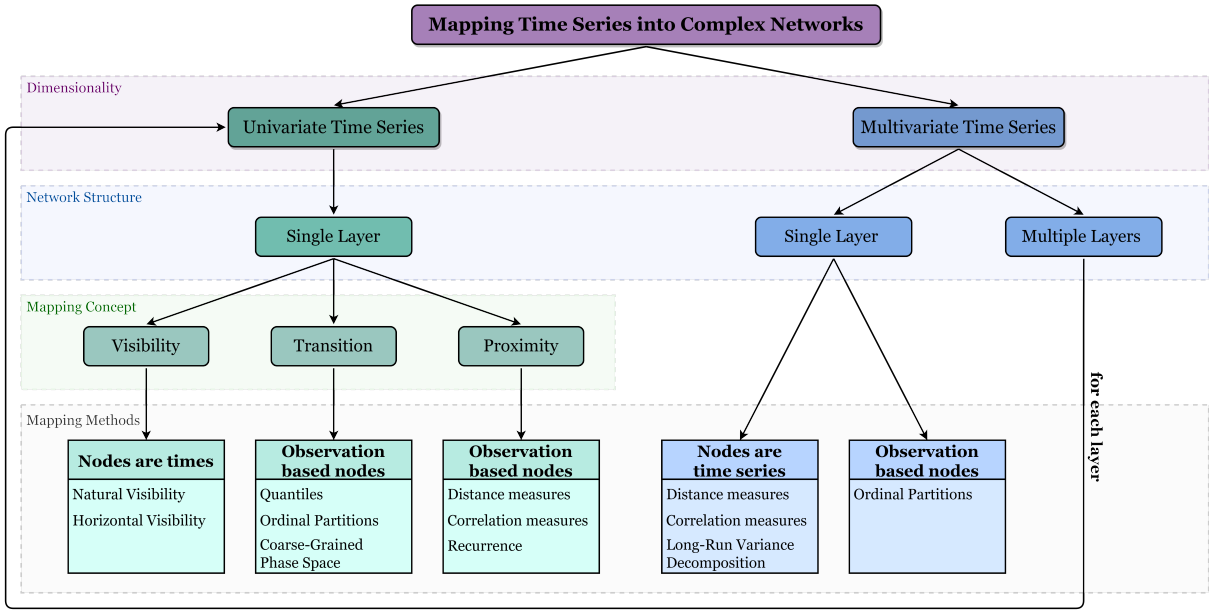


Figure 3.1: Overview of mapping methods. Taxonomy of algorithms for mapping time series into complex networks based on the dimensionality of time series, resulting network structure, mapping concept, and main mapping methods.

The set of univariate series mappings proposed in the literature may be divided into three main types depending on the underlying concept: visibility, transition, and proximity. Visibility mapping concepts establish connections (edges) between the timestamps of the series (as nodes) using visibility lines (with or without restrictions) between observations. For transition mappings, the methods are based on the transition probabilities between states (or 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 nodes of the network. Finally, proximity mappings establish connections using measures of distance or similarity between time points (or states) to become network nodes. Regarding multivariate time series, there are methods leading to a single-layer network, where each series (or patterns based on observations from multivariate series) is represented by a node in the network and the connections are established based on relationships between the series (or patterns). We can also find mappings leading to multiple layer network structures, where each layer represents a time series and, in fact, each layer can be mapped based on one of the univariate time series mapping methods, separately. Research on multivariate mappings is however still in its infancy.

3.1 Univariate Time Series Mappings

In this section, we review the state of the art of mapping univariate time series into the network domain. As previously represented in Figure 3.1, from a high-level perspective, and as in previous papers (Donner et al., 2011b; Zou et al., 2019), we divide the existing approaches into three different main classes, according to the underlying mapping concept. An overview of the main characteristics of the methods here described is given in Table 3.1. In the following sections, we will give an in-depth description of how these approaches emerged, how they work, what variants exist, and what are the core results.

3.1.1 Visibility Networks

Lacasa and co-authors (Lacasa et al., 2008) proposed visibility mappings from (univariate) time series to complex networks, based on traditional visibility algorithms from computational geometry (Ghosh, 2007). The visibility networks have a geometric criterion associated with the natural ordering of the time series. Given its ease of interpretation and implementation, this mapping quickly attracted the research community and several results as well as variants of these methods began to emerge. Visibility-based algorithms incorporate, as we shall see, global and local topological characteristics of the time series in the graph characteristics, are easy to implement, computationally fast, and, except for the limited and parametric versions, are parameter-free.

Throughout this section, we present the different types of visibility algorithms proposed in the literature, which are essentially based on the natural visibility (Lacasa et al., 2008) and horizontal visibility (Luque et al., 2009) algorithms.

3.1.1.1 Natural Visibility Graphs

Lacasa and co-authors (Lacasa et al., 2008) proposed the first method based on the concept of visibility, the natural visibility graph (NVG), or simply visibility graph (VG). This method is based 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, 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 3.2.

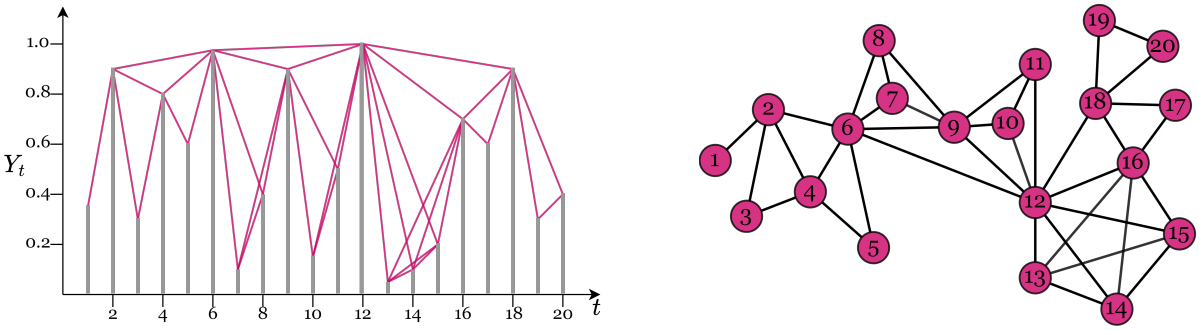
Formally, the set of nodes $\{v_i\}$ of a NVG are numbered sequentially in time and two nodes v_i and v_j are connected (have visibility) if any other observation (t_k, Y_k) with $t_i < t_k < t_j$ satisfies:

$$Y_k < Y_j + (Y_i - Y_j)\frac{(t_j - t_k)}{(t_j - t_i)}. \tag{3.1}$$

	Network Type (original reference)	Node	Edge	Directed	Weighted	Param-free	Pre proc
Visibility	Natural Visibility Network (Lacasa et al., 2008)	t	NV	✗	✗	✓	✗
	Horizontal Visibility Network (Luque et al., 2009)	t	HV	✗	✗	✓	✗
	Direct Horizontal Visibility Network (Lacasa et al., 2012)	t	HV	✓	✗	✓	✗
	Limited Penetrable Natural Visibility Network (Zhou et al., 2012)	t	NV	✗	✗	✗	✗
	Parametric Natural Visibility Network (Bezsudnov and Snarskii, 2014)	t	NV	✓	✓	✗	✗
	Weighted Visibility Network (Bezsudnov and Snarskii, 2014)	t	NV	✓	✓	✓	✗
	Difference Visibility Network (Zhu et al., 2014a)	t	V	✗	✗	✓	✗
	Absolute Visibility Graph (Yan and van Tuyll van Serooskerken, 2015)	t	NV	✗	✗	✓	✗
	Limited Penetrable Horizontal Visibility Network (Gao et al., 2016a)	t	HV	✗	✗	✗	✗
	Dual Perspective Visibility Graph (Zheng et al., 2021)	t	RNV	✗	✗	✓	NV
Transition	Coarse-Grained Phase Space Network (Gao and Jin, 2009)	$\vec{z}_i$	TP	✓	✓	✗	PS
	Quantile Network (Campanharo et al., 2011)	q_i	TP	✓	✓	✗	DS
	Ordinal Partitions Network (Small, 2013)	π_i	TP	✓	✓	✗	AR
	Visibility Graphlets Network (Mutua et al., 2015)	G_i	TP	✓	✓	✓	PS;VG
Proximity	Cycle Network (Zhang et al., 2006)	c_i	CM	✗	✗	✗	DC
	Correlation Network (Yang and Yang, 2008)	$\vec{z}_i$	CM	✗	✓	✗	PS
	κ-Nearest Neighbor Network (Small et al., 2009)	$\vec{z}_i$	DM	✓	✗	✗	PS
	Adaptive Nearest Neighbor Network (Donner et al., 2011b; Small et al., 2009; Xu et al., 2008)	$\vec{z}_i$	DM	✗	✗	✗	PS
	ϵ-Recurrence Network (Donner et al., 2010)	$\vec{z}_i$	DM	✗	✗	✗	PS

Table 3.1: Comparison of (univariate) time series mappings based on the properties of the corresponding algorithms and of the resulting networks. Notation: NV - natural visibility, HV - horizontal visibility, V - natural and horizontal visibility, TP - transition probability, CM - correlation measures, DM - distance measures, RNV - reflected natural visibility, PS - phase space reconstruction, DS - division of the support, AR - amplitude rank, VG - directed visibility graph, DC - division into cycles.

The NVG algorithm is easy to implement and, in terms of computational complexity, has quadratic complexity ($\mathcal{O}(T^2)$) and thus becomes very slow for very long time series. However, a more efficient algorithm based on the divide and conquer technique proposed in Lan et al. (2015) has a complexity of $\mathcal{O}(T \log T)$.



(a) Toy time series

(b) NVG

Figure 3.2: On the left side, we present the plot of a toy time series and, on the right side, the network generated by the natural visibility algorithm. The purple lines in the time series plot represent the lines of visibility (and hence the edges of the graph) between data points.

Visibility graphs are always connected because each node v_i sees at least its neighbors, v_{i-1} and v_{i+1}, and are always undirected. However, a direction can be defined considering the direction of the time axis. The network is also invariant under affine transformations of the data (Lacasa et al., 2008) 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 $X = aY + b$, for $a \in \mathbb{R}$ and $b \in \mathbb{R}$, leads to the same NVG.

A NVG has, typically, a distinct topology characterized by hubs representing maximum of the time series, since these have visibility contact with more nodes when compared to other points (Donner and Donges, 2012). However, this is not always necessarily true since a process with a concave behavior over a certain period of time, as for example the Conway series (Lacasa et al., 2008), can lead to highly connected nodes that do not coincide with local maximum (Donner and Donges, 2012). The presence of hubs that correspond to (local) maxima of time series typically gives rise to a topological structure represented by communities. These communities usually reflect the temporal order of the observations. The communities may also be formed by fragments of the time series, in which case do not reflect the temporal order of the observations (Yao and Lin, 2017).

Lacasa and co-authors (Lacasa et al., 2008) have shown that NVGs inherit several structural properties of the time series. In fact, the method maps periodic series [1] on regular graphs,

[1]Periodic time series are series where the process has a regular seasonal pattern. However, there are many processes where the data appears highly periodic but does not repeat itself exactly. These processes are called pseudo-periodic time series.

random series on random graphs [2] and fractal series on scale-free graphs [3]. In particular, for periodic series, the graphs have a regular structure where the degree distribution presents a number of peaks related to the period of the series. For white noise processes, the resulting graph is completely random, producing a degree distribution that is an exponential function. For fractal time series, the degree distribution is a power law related to the fractality of series.

In Lacasa et al. (2009), the authors used the NVGs to quantify long-range dependence and fractality in time series. These authors concluded that fractal processes result in scale-free visibility graphs with a degree distribution that is a power law $P(k) \sim k^{\gamma}$ with exponent γ which is a linear function of the Hurst coefficient. These results were applied to study the fractality of energy dissipation rates in turbulent flows (Liu et al., 2010), to study turbulent heated jets behaviors (Charakopoulos et al., 2014), and to diagnose Alzheimer's disease automatically based on the data electroencephalography (EEG) (Ahmadlou et al., 2010), among others.

Recently, a new definition of motif [4] in the context of NVG was presented in Iacovacci and Lacasa (2016b): *sequential NVG n-node motifs*. According to this definition, motifs are the set of NVG subgraphs formed by the sequence of nodes $\{g, g+1, \ldots, g+n-1\}$, where $n < T$ and $g \in [1, T-n+1]$). This definition differs from traditional network motifs in that they require nodes to be labeled according to the temporal order of the nodes induced by the construction of NVG. One advantage of this definition is that the computation of these motifs is extremely efficient, linear time $\mathcal{O}(T)$, using the theory developed in Iacovacci and Lacasa (2016a). Similarly to traditional motifs networks, we can compare the relative occurrence of each motif to distinguish different time series processes, see more details in Iacovacci and Lacasa (2016b).

Donner and Donges (Donner and Donges, 2012) address the problem of missing data in time series. They assessed the effect that missing data produces in the topological properties of NVGs associated with Gaussian white noise processes, using two strategies: missing data is simply ignored in the generation of the NVG; the NVG is fragmented into subgraphs corresponding to times before and after the missing observations. Donner and Donges (2012) concluded that ignoring the missing data had a considerable effect mainly on closeness centrality but not on

[2] Random networks are graphs obtained by a random process or by a probability distribution (Erdős and Rényi, 1960). There are many random graph models, but the *Erdős Rényi model* (Erdős and Rényi, 1960) is the most fundamental and widely studied of them (Newman, 2010). This model is mathematically referred by $G(n, p)$ where n is the number of nodes and p is the probability with that the edges between each distinct pair of nodes are connected. In this model, the probability that a node has k edges (degree distribution) follows a Poisson distribution $P(k) = \frac{e^{\lambda}\lambda^{k}}{k!}$ (Barabási and Albert, 1999).

[3] Scale-free networks have a degree distribution (the probability $P(k)$ that a given node has a degree, i.e., a number of edges that connects the node to the other nodes, equal to k) that follows the power law $P(k) \sim k^{\gamma}$. This means that most nodes have few edges in contrast to the existence of some nodes with a high degree (hubs) (Barabási and Albert, 1999).

[4] Network *motifs* (Milo et al., 2002) are small subgraphs (typically with 3 to 5 nodes) originally defined as patterns that occur more often than expected, that is, whose frequency is higher than in randomized networks; nowadays the term motif is also used to refer to all such small patterns, regardless of its overrepresentation, and their frequencies can serve as a rich network fingerprint (Milo et al., 2004).

degree distribution and local clustering coefficient, as well. When the missing data is taken into account and the NVG becomes a set of subgraphs, path-based metrics are the most impacted. As expected the impacts become stronger as the missing data increases. Interestingly, those authors also found out that the effects of node-level metrics are slightly reduced for missing data occurring in runs rather than randomly.

Given the potential demonstrated, the method was applied in the study of energy dissipation rates in three-dimensional turbulence (Liu et al., 2010), financial time series (Long, 2013; Qian et al., 2010; Yang et al., 2009; Zhuang et al., 2014), heart rate variability (Hou et al., 2016; Shao, 2010), classification of sleep stages (Zhu et al., 2014a), classification of time series (Li et al., 2018), hurricane occurrence in the United States (Elsner et al., 2009), to perform forecasting (Huang et al., 2021b; Zhang et al., 2017b, 2018b), and more recently to preserve data privacy (Xiu et al., 2022).

Note that NVGs are not able to distinguish time series with certain properties. As an example, consider a time series with a deterministic increasing trend. Its NVG and the NVG of its symmetric time series which presents a deterministic decreasing trend are similar graphs with the same properties. This disadvantage may be overcome by defining NVGs with directed edges.

3.1.1.2 Horizontal Visibility Graphs

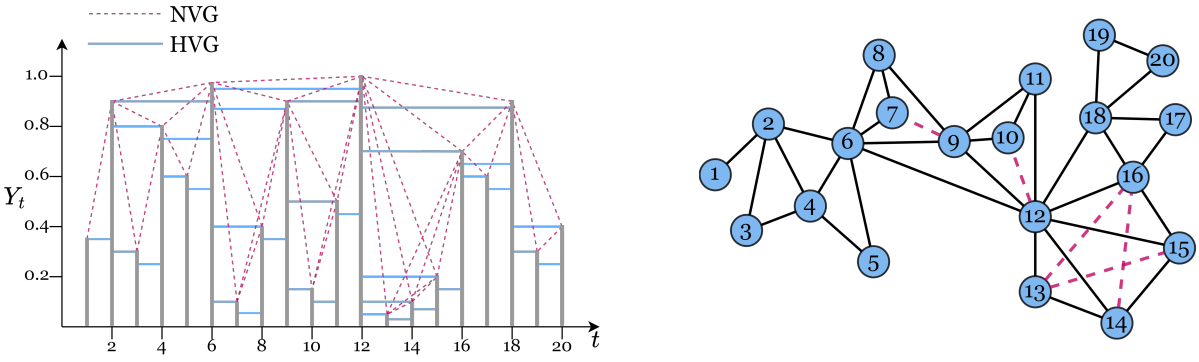
In order to reduce the computational complexity associated with NVGs, Luque et al. (2009) proposed a simplified NVG method called the horizontal visibility graph, HVG, which inherits all NVG graph characteristics mentioned above. The construction of HVGs differs from that of NVGs in that the visibility lines are only horizontal. Figure 3.3 gives a simple illustration of this algorithm, with a toy time series and the resulting network.

Formally, two nodes v_i and v_j are connected, if for all (t_k, Y_k) such that $t_i < t_k < t_j$ the following condition is met:

$$Y_i, Y_j > Y_k. \tag{3.2}$$

In terms of computational complexity, the generation of HVGs has a time complexity of the construction $\mathcal{O}(T \log T)$ (Zhu et al., 2014b) and, in the case of noisy (stochastic or chaotic) processes, the algorithm has an average-case time complexity $\mathcal{O}(T)$ (Luque et al., 2009).

The HVG is always a subgraph of the NVG for a particular time series. This is well illustrated in Figures 3.2 and 3.3 where we can easily verify that all the edges present in the HVG are present in NVG, but the converse is not true, e.g. edges $(6, 8)$ and $(12, 15)$. HVG nodes will always have a degree less than or equal to that of the corresponding NVG nodes. Therefore, there is some loss of quantitative information in HVGs in comparison with NVGs (Luque et al., 2009) which may be crucial in the analysis of certain time series. However, it has no impact on the qualitative characteristics of the graphs, since the graphs preserve some part of the data information, namely the local information (the closest timestamps). Another characteristic of



(a) Toy time series

(b) HVG

Figure 3.3: On the left side, we present the plot of a toy time series and, on the right side, the network generated by the horizontal visibility algorithm. The green lines represent the horizontal lines of visibility between the data points and the purple lines the natural visibility, for comparison.

HVGs is that it is always an outerplanar graph [5] and has a Hamilton path, that is, a path that passes through all the nodes of a graph only once. In fact, Gutin et al. (2011) have shown that this is a necessary and sufficient condition for a graph to be an HVG.

Luque and co-authors (Luque et al., 2009) have formally established several relationships between properties of HVGs and characteristics of the underlying time series.

The first such property states that uncorrelated random series is mapped into an HVG with an exponential degree distribution given by:

$$P(k) = \frac{1}{3}\left(\frac{2}{3}\right)^{k-2}, \quad \text{for } k \geq 2. \tag{3.3}$$

This result holds for all probability distributions. It follows that, for uncorrelated random series, the associated average degree is (Nuñez et al., 2012a):

$$\bar{k} = \sum kP(k) = \sum_{k=2}^{\infty} \frac{k}{3}\left(\frac{2}{3}\right)^{k-2} = 4. \tag{3.4}$$

Additionally, Lacasa and Toral (2010) suggested that the degree distribution follows the exponential law, $P(k) \sim \exp^{(-\lambda k)}$, where the value of λ depends on the type of process generating the time series: $\lambda < \ln \frac{3}{2}$ for chaotic processes, $\lambda > \ln \frac{3}{2}$ for stochastic processes and $\lambda = \ln \frac{3}{2}$ for uncorrelated processes. However, note that Ravetti et al. (2014) found that for the Rössler system [6] the rule is not valid, and Zhang et al. (2017c) found results in which negatively

[5] An outerplanar graph is a planar graph, (i.e., it can be drawn on the plane so that no edges cross each other), where all nodes are incident to the infinite face, that is, no node is totally surrounded by edges.

[6] The Rössler system is a complex system of three ordinary differential equations (Rössler, 1976), which define a continuous-time process that exhibits chaotic behavior, where the predictability of the behavior decreases exponentially with lead time.

correlated processes lead to lower λ values than the critical value. These results show that there are exceptions to the rule and that using λ to distinguish chaotic from stochastic processes requires further investigation.

A second property obtained by Luque et al. (2009) refers to the relationship between the hubs of the graph and the extreme values of the data. This equivalence was obtained numerically. Furthermore, based on geometric arguments, Luque et al. (2009) obtained the following relation between the clustering coefficient, a measure to capture the degree to which the nodes of a graph tend to cluster, and the degree in an HVG:

$$C_i = \frac{k_i - 1}{\binom{k_i}{2}} = \frac{2}{k_i}, \tag{3.5}$$

where $\binom{k_i}{2}$ denotes the number of possible triangles and $k_i - 1$ is the number of visible nodes that are also visible from v_i. This relation allows also to deduce the local clustering coefficient distribution (Nuñez et al., 2012b), substituting k_i by C_i in Equation (3.3).

The HVG associated with an infinite periodic series of period P is a representation of a concatenation of a motif. Nuñez and co-authors (Nuñez et al., 2012a) proved that the average degree of the resulting HVG is given by:

$$\bar{k} = 4\left(1 - \frac{1}{2P}\right). \tag{3.6}$$

This result implies that time series are mapped on HVGs with $2 \leq \bar{k} \leq 4$. The lower bound is reached for constant series (HVG is a chain graph and so each node is only connected to its two closest neighbors) and the upper bound for aperiodic series (random and chaotic process).

Nuñez and co-authors (Nuñez et al., 2012a) proposed a method for calculating the hidden periodicity in a periodic noisy signal, a very common problem in real world data analysis and difficult to solve with traditional periodicity detection algorithms, such as spectral analysis. The method involves the construction of a modified HVG, called the filtered HVG (f-HVG), where two nodes v_i and v_j are connected if the following condition is met:

$$Y_i, Y_j > Y_k + f, \tag{3.7}$$

for all $t_i < t_k < t_j$ and $f \in [\min(Y_t), \max(Y_t)]$ is a real-valued scalar that acts as a filter. To filter the noise of a time series, its f-HVG is generated for increasing values of f, and at each step, the average degree is calculated. The result is a decrease of the average degree, with initial value 4 (for $f = 0$), an asymptotic value of 2 (lower bound of $\bar{k}$) and the plateau obtained for the distribution allows to obtain the period through Equation (3.6). This method delivered very promising results including, specific cases where autocorrelation analysis yields misleading results.

The definition of sequential motif presented for NVGs in Iacovacci and Lacasa (2016b) also applies similarly to HVGs. The definition of *sequential HVG n-node motifs* is the same way as that for an NVG. The frequency of sequential HVG motifs is also useful for discriminating different dynamic processes.

HVGs were used to analyze seismic signals (Telesca and Lovallo, 2012), evaluate the complex dynamics of tourism systems (Baggio and Sainaghi, 2016), construct the so-called Feigenbaum graphs in order to study unimodal maps and their spectral properties (Flanagan et al., 2019), study the volatility behavior of returns for financial time series (Zhang et al., 2015), analyze heartbeat rates of healthy subjects, congestive heart failure subjects, and atrial fibrillation subjects (Xie et al., 2019) and predicting catastrophes (Zhang et al., 2018a).

In recent work, Li et al. (2018) show that the topological properties of the NVGs and HVGs are useful features for constructing accurate classification models using generic classifiers for the classification of real-time series, since combining both graphs allows capturing global (in the case of NVGs) and local (in the case of HVGs because they are more sensitive to local variations) features. This approach allows to increase the classification accuracy of a vast set of time series of different domains and it turns out to be better when compared to the traditional approaches, such as time series features obtained from common statistical measures.

Several variants of NVGs and HVGs have been proposed in the literature. The most relevant ones are presented in the following sections.

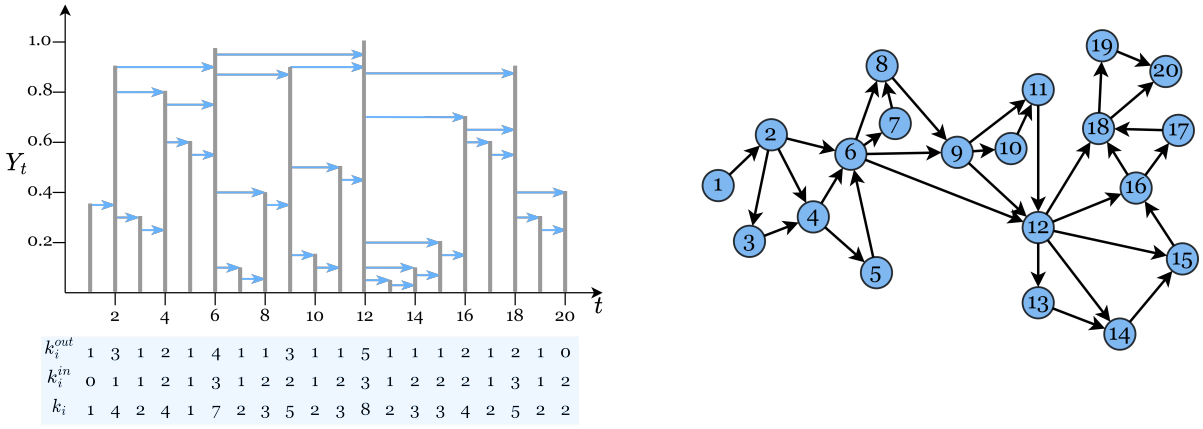
3.1.1.3 Directed Visibility Graphs

Given that time has a natural direction, Lacasa et al. (2012) introduced directed horizontal visibility graphs, DHVG, by defining an HVG with edges (v_i, v_j), $i < j$. The adjacency matrix is no longer a symmetric matrix. An example of the representation of this algorithm is illustrated in Figure 3.4. This variant is extended straightforwardly to the NVGs.

DHVG was proposed as a simple and well-defined tool for measuring time series irreversibility [7] notoriously difficult to assess with traditional algorithms, like algorithms that involve time series symbolization, which normally involve the choice of extra parameters and the results may depend on that choice. The degree of irreversibility of a time series is calculated as the Kullback-Leibler distance between the *in*- and *out*-degree distributions, $P(k^{in})$ and $P(k^{out})$ (Lacasa et al., 2012). These authors proved that for a bi-infinite sequence of i.i.d. random variables, both the *in* and *out*-degree distributions of the corresponding DHVGs are equal and given by:

$$P(k^{in}) = P(k^{out}) = \left(\frac{1}{2}\right)^k, \quad k = 1, 2, 3, \ldots. \tag{3.8}$$

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



(a) Toy time series

(b) DHVG

Figure 3.4: Illustrative example of directed horizontal visibility algorithm and corresponding *out*-degree (k_i^{out}), *in*-degree (k_i^{in}) and *total*-degree (k_i). On the left side, we present the plot of a toy time series and, on the right side, the network generated by the directed horizontal visibility algorithm. The green directed lines represent the directed horizontal lines of visibility between data points.

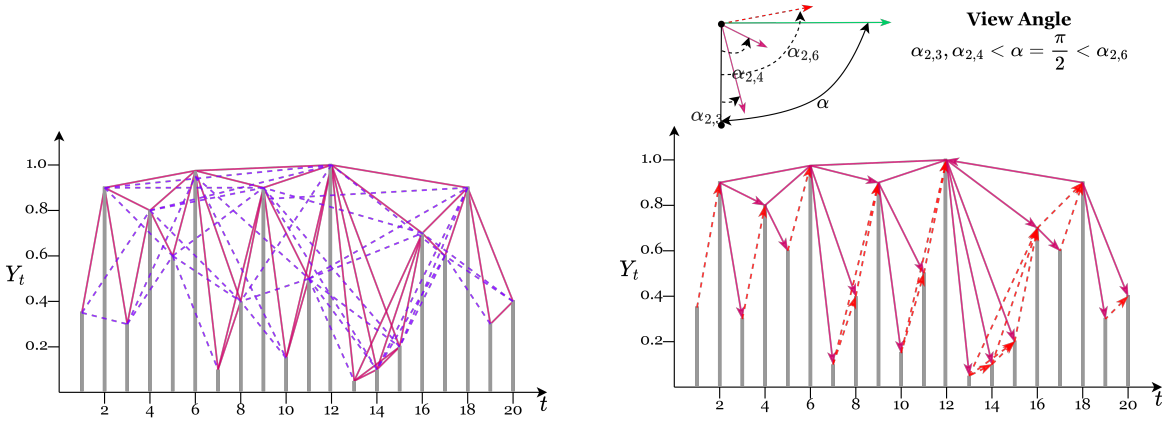
In the same line of research, Donges et al. (2013) followed a similar approach. Donges and co-authors proposed a set of rigorous statistical tests for time series irreversibility based on both visibility algorithms (NVG and HVG). The authors compare the degree and local clustering coefficient distributions taking into account the past ($P(k^r)$ and $P(C^r)$) and the future ($P(k^a)$ and $P(C^a)$) separately. It is conjectured that if a time series is reversible the distributions $P(k^r)$ and $P(k^a)$ (or $P(C^r)$ and $P(C^a)$) should be similar, while for irreversible time series we should find statistically significant deviations between the distributions.

3.1.1.4 Limited Penetrable Visibility Graphs

The limited penetrable visibility graph, LPVG, was first proposed by Zhou et al. (2012) to improve the NVG and HVG mappings. Similarly to the NVG method, the nodes are numbered sequentially in time but now have a limit l of visibility, and the nodes v_i and v_j are connected if:

$$Y_{i+l} < Y_j + (Y_i - Y_j)\frac{t_j - (t_i + t_l)}{(t_j - t_i)}, \quad l < j - i, \tag{3.9}$$

where l is a limited penetrable distance established to reduce the effect of noise intrinsic to data. So, the nodes v_i and v_j have mutual visibility through l intermediate bars (data) in the time series. The presence of noise in the data causes the connections on NVGs to break easily, while the nodes should, in principle, have more connections to other nodes, except for the maxima data (Pei et al., 2014). A small illustration of this method for $l = 1$ is represented in Figure 3.5a.



(a) LPVG algorithm

(b) PNVG vs. DNVG algorithm

Figure 3.5: (a) Illustrative example of limited penetrable visibility graph algorithm where the limit $l = 1$. The purple lines show the edges between points that have direct visibility (as in NVG, $l = 0$) and the blue dashed lines are the extra edges imposed by the LPVG algorithm, where two points can be seen with only one higher intermediate point. (b) Illustrative example of parametric natural visibility graph algorithm with a comparison with NVG algorithm.

A limited penetrable version for HVG was developed by Gao et al. (2016a) and called limited penetrable horizontal visibility graph (LPHVG). The graphs are constructed using the above algorithm, except that the visibility condition is horizontal visibility. Note that when $l = 0$, the LPVG (LPHVG) is reduced to NVG (HVG). Other versions of this method exist, namely, a directed version, the directed limited penetrable horizontal visibility graph (DLPHVG), and an image version, the image limited penetrable horizontal visibility graph (ILPHVG) (Wang et al., 2018b).

LPVGs are very promising: as the NVGs they inherit several properties of the time series and are able to detect differences between random and chaotic series, detect the location of inverse bifurcations in chaotic dynamical systems and, in addition, as an advantage over NVGs, they show good tolerance to noise interference (Wang et al., 2016a). However, it is necessary to pay attention to the choice of parameter l so that not too much information is incorporated into the graphs. The idea of limited natural visibility was applied to analyze abnormalities of EEG signals from Alzheimer's disease (Wang et al., 2016a), EEG signals under manual acupuncture (Pei et al., 2014), and signals from an electromechanical system in the process industry (Wang et al., 2016b).

Ren and Jin (Ren and Jin, 2020) applied the idea of sequential motifs (see Section 3.1.1.1) to LPVGs and introduced a measure of motif entropy to estimate the complexity of network structure, rather than looking at the motif occurrence frequency. They obtained better robustness and the ability to distinguish different processes when compared to visibility-graph motif entropy.

Wang et al. (2018a) presents some exact results (similar to those obtained for the HVGs presented in Section 3.1.1.2) on the topological properties of the LPHVG associated with bi-infinite time series of i.i.d. random variables with a probability density $f(x)$. The degree distribution of the associated LPHVG with the limited penetrable distance l is given by:

$$P(k) = \frac{1}{2l+3}\left(\frac{2l+2}{2l+3}\right)^{k-2(l+1)}, \tag{3.10}$$

where $l = 0, 1, 2, \ldots$, and k is the degree of a node. From Equation (3.10) we obtain the average degree $\bar{k}$:

$$\bar{k} = \sum kP(k) = 4(l+1). \tag{3.11}$$

And based on the Equation (3.10) we can deduce the minimum and maximum local clustering coefficient of the LPHVG associated with i.i.d. random series as (see Wang et al. 2018a for more details):

$$C_{min}(k) = \frac{2}{k} + \frac{2l(k-2)}{k(k-1)}, \quad l = 0, 1, 2, \ldots; \quad k \geq 2(l+1) \tag{3.12}$$

$$C_{max}(k) = \frac{2}{k} + \frac{4l(k-3)}{k(k-1)}, \quad l = 0, 1, 2, \ldots; \quad k \geq 2(2l+1). \tag{3.13}$$

For an infinite periodic series of period, P, the average degree depends on the period P:

$$\bar{k} = 4(l+1)\left(1 - \frac{2l+1}{2P}\right), \quad l \ll P. \tag{3.14}$$

LPHVG was employed in the analysis of EEG signals and biphasic-flow signals where they characterize the behaviors underlying the systems (Gao et al., 2016a), in the analysis of chaotic series and energy and oil price series (Wang et al., 2018a), and to distinguish between random, periodic and chaotic signals using motifs (Wang et al., 2018b).

3.1.1.5 Weighted Visibility Graphs

The method proposed by Supriya and co-authors (Supriya et al., 2016) is a fairly simple modification to the traditional NVG algorithm, 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 constructed as follows: a directed NVG is constructed as described in Section 3.1.1.3, and 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:

$$\alpha_{i,j} = \tan^{-1}\left(\frac{Y_j - Y_i}{t_j - t_i}\right), \quad i < j. \tag{3.15}$$

We should note that Equation (3.15) 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, standard methods and techniques do not apply straightforwardly (Kaur and Singh, 2016), and therefore is less common. For this reason, Supriya et al. (2016) consider the absolute value of the view angle, $|\alpha_{i,j}|$, between the observations of the time series. We can see that the sign of the angle between the observations is an indicator of an increase (positive angle) or decrease (negative angle) of the values of the underlying process, thus providing information about changes in trends throughout the series.

We can consider a parametric version of WVGs, called parametric natural visibility graph, PNVG or PNVG(α), proposed by Bezsudnov and Snarskii (Bezsudnov and Snarskii, 2014). This algorithm adds a restriction to the edge assignment in WVG determined by a threshold parameter α associated with the visibility angle. So, the PNVG(α) imposes that a weighted edge $(v_i, v_j, w_{i,j})$ is established from node v_i to node v_j only if the view angle between those nodes is less than threshold parameter, that is, $\{(v_i, v_j, w_{i,j}) \in E \mid \alpha_{i,j} < \alpha, w_{i,j} = |\alpha_{i,j}|\}$, as shown in the Figure 3.5b.

The PNVG(α) is a directed acyclic graph, it is always a subgraph of the underlying NVG (and WVG) and, consequently, invariant under affine transformations of the time series; and it can be a connected or disconnected graph.

PNVG and WVG were designed to try to overcome the loss of quantitative information of the NVG's binary matrix. Other than *arc tangent* weights can be associated with the edges of the graphs. The arc tangent is a direct measure of the concept of visibility and a direction for future work may be to analyze its properties. However, the characteristics of the resulting network will always be dependent on this weight.

Specific topological features were defined for this mapping, namely, *relative average degree*, *relative average length of edge* and *relative number of clusters*, which aim to compare the average degree, the average length of edges and number of clusters with the corresponding features in the underlying NVG (i.e., when $\alpha = \pi$). The authors in Bezsudnov and Snarskii (2014) showed that these features are useful for distinguishing, identifying, and describing various time series. They performed tests in different synthetic time series and in real heart rate data to distinguish different series associated with people with different health conditions, where they obtained good results. However, PNVG is not a parameter-free method and it always involves choosing the parameter α that conditions the final results.

In Supriya et al. (2016) the authors applied this approach to EEG reference data associated with epileptic activity and showed that the metrics such as modularity (that measures how good a division of the graph is in specific communities) and weighted average degree of WVG constructed for this data help to distinguish convulsion signals from normal signals, detecting the sudden fluctuations in signals. The accuracy of the results surpassed many other methods.

Note that other strategies for assigning weights to the edges of (natural or horizontal) visibility graphs, that are different from those of the visibility angles, can be considered. Examples include weights based on Euclidean distance (Bianchi et al., 2017) and weights based on the VG own degree distribution (Song and Xiao, 2022; Xu et al., 2018) leading to undirected and weighted horizontal and natural visibility graphs.

3.1.1.6 Difference Visibility Graphs

A variant of the NVG and HVG consists of the creation of difference visibility graphs, DVG (Zhu et al., 2014a). This algorithm subtracts the HVG edge set from the NVG edge set. Thus, the degree k_i of a node v_i of the DVG is: $k_i = k_{i,NVG} - k_{i,HVG}$, since the HVG is always a subgraph of the NVG. For the average degree we have $\bar{k} = \bar{k}_{NVG} - \bar{k}_{HVG}$. DVGs will very possibly be disconnected and may have isolated nodes (which have no connections). The DVGs were used together with HVGs in An et al. (2019); Zhu et al. (2014a) to extract graph metrics, such as average degree, degree distribution, and degree sequence, to classify automatically real EEG data in order to detect different sleep stages. DVGs obtained intermediate precision values when compared to other strategies (An et al., 2019).

3.1.1.7 Absolute Invisibility Graph

Based on the NVG concept, Yan and van Tuyll van Serooskerken (2015) extend the concept to a new algorithm that builds the so-called absolute invisibility graph (AIG), whose definition is the opposite of NVG, that is, two nodes v_i and v_j are connected if there is no natural line of visibility between corresponding data values (t_i, Y_i) and (t_j, Y_j):

$$Y_k > Y_j + (Y_i - Y_j)\frac{(t_j - t_k)}{(t_j - t_i)}, \quad \text{for all } t_k, \;\; t_i < t_k < t_j. \tag{3.16}$$

This algorithm was conceived with finance time series in mind, namely stock prices, in order to understand and measure the magnitude of the super-exponential growth of these data. More specifically, the authors take advantage of the topological features of NVG degree and AIG degree to predict the peaks (high values) and troughs (low values) in the financial market prices (Chen et al., 2019; Liu and Chen, 2020; Yan and van Tuyll van Serooskerken, 2015).

3.1.1.8 Dual Perspective Visibility Graph

The dual perspective visibility graph, DPVG (Zheng et al., 2021), is a proposed method that combines the concept of the natural view of NVGs with the concept of *reflected view*. It is projected to culminate the limitation of VGs to distinguish positive and negative intensity changes from time series data.

To build a DPVG from a time series $\{Y_t\}_{t=1}^{T}$ we need first map the time data into a NVG following the Equation 3.1, which is described by the adjacency matrix $\boldsymbol{A}$, and when we calculate the reflected perspective of the NVG. This is made by inverting the time series across the time axis, that is, for each Y_i we compute the inverted time series values: $X_i = -Y_i$. The second to last step is to map the inverted time series $\{X_t\}_{t=1}^{T}$ into corresponding NVGs, generating the reflected perspective NVG described by the adjacency matrix $\boldsymbol{B}$. Finally, we combine the NVG and the reflected perspective NVG by the following criteria,

$$A_{i,j} = \max\left(\mathrm{A_{i,j}}, \mathrm{B_{i,j}}\right), \tag{3.17}$$

where the adjacency matrix $\boldsymbol{A}$ now defines the adjacency matrix of the DPVG. This new mapping method can be extended to the horizontal visibility concept and to the directed and weighted versions.

Note that if the interest is not in data changes below a given baseline or if the time series does not present negative values, we do not need to generate the DPVG.

The authors in (Zheng et al., 2021) also propose a community detection method in VG, which is based on calculating the shortest path between nodes in the VG, preserving temporal information, and using it to characterize biological time series.

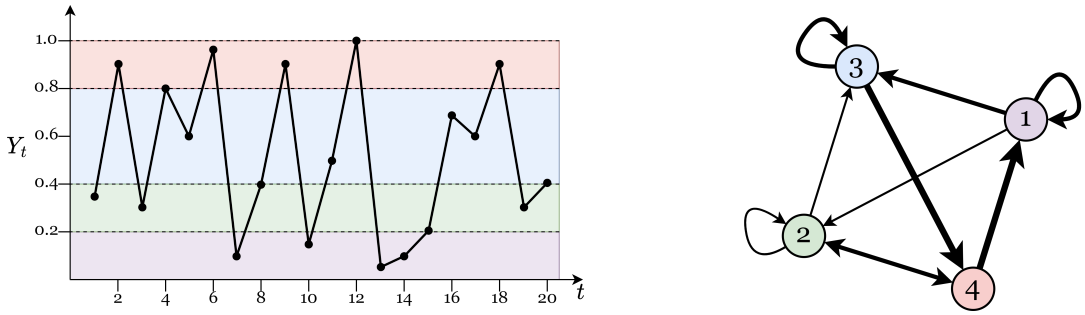
3.1.2 Transition Networks

Transition networks are a type of network that is constructed from time series based on the concept of transition between symbols that are assigned to represent the series. In general, the construction of transition networks involves two fundamental steps: assigning a symbol encoding to the time series data; mapping the symbols and the transition function between the symbols into the nodes and the edges, respectively. The first step transforms the time series into a set of symbols or states by partitioning either the time range, reconstructing the time series in a new phase space [8], or the support of the time series. The first partition strategy leads to coarse-graining phase space graphs, to ordinal partition transition networks, and to visibility graphlets networks, while the second leads to quantile graphs. In the second step of the transition network construction process, the edges are established by transition probabilities between the symbols obtained in the first step, computed as the relative frequency of symbol sequences s_i, s_j, leading to directed and weighted edges. The resulting transition network is a directed and weighted graph whose adjacency matrix is a Markov matrix. An unweighted representation can be obtained by omitting the information on the probability/quantity of transitions between different states, or a less dense graph can be obtained considering a limit for the probability of transitions between states in order to exclude rare transitions (for example, due to noise in deterministic dynamical systems) (Zou et al., 2019).

[8] The reconstruction of phase space (or state space) is the basis of the analysis of nonlinear processes, mostly adopted in areas such as physics. It allows reconstructing the complete dynamics of a system from a single time series (Bradley and Kantz, 2015). It consists of embedding $(Y_1, \ldots, Y_t)$ into a w-dimensional space with a time delay of embedding τ with states $\{\vec{z}_1, \vec{z}_2, \ldots, \vec{z}_{T-(w-1)\tau}\}$, where $\vec{z}_i = (Y_i, Y_{i+\tau}, \ldots, Y_{i+(w-1)\tau})$ (Takens, 1981).

3.1.2.1 Quantile Graphs

Quantile graphs (QG) were introduced by Campanharo and co-authors (Campanharo et al., 2011) based on the idea of assigning the time series observations to bins (Shirazi et al., 2009). In the construction of quantile graphs, the bins are defined by η quantiles, $q_1, q_2, ..., q_\eta$. Each quantile, q_i, is associated to a node v_i of the graph so the graph has as many nodes as the number of quantiles. Two nodes v_i and v_j are connected by a weighted directed edge $(v_i, v_j, w_{i,j})$, where the weight $w_{i,j}$ represents the transition probability between quantile ranges. The adjacency matrix becomes a Markov transition matrix, where $\sum_{j=1}^{\eta} w_{i,j} = 1$, for each $i = 1, \ldots, \eta$. This mapping is illustrated in Figure 3.6.



(a) Toy time series

(b) QG

Figure 3.6: Illustrative example of the quantile graph algorithm for $\eta = 4$. On the left panel, we present the plot of a toy time series, and on the right panel the network generated by the quantile graph algorithm. The different colors in the time series plot represent the regions corresponding to the different quantiles. In the network, edges with larger weights represented by thicker lines correspond to the repeated transitions between quantiles.

Later, Campanharo and Ramos (2016) extended the QG method to a more general version that not only allows representing the transition between quantiles corresponding to consecutive timestamps but also allows the construction of the QG that represents the transition between quantiles corresponding to lagged timestamps. That is, two nodes v_i and v_j are connected by a weighted directed edge $(v_i, v_j, w_{i,j,h})$, whenever two timestamp values Y_t and T_{t+h} belong respectively to quantiles q_i and q_j, with $t = 1, \ldots, T$ and the lagged $h = 1, \ldots, h_{max} < T$. The weight $w_{i,j,h}$ represents the probability of transiting from the quantile q_i at time t to the quantile q_j at time $t + h$. If $h = 1$, QG reduces to the original definition.

The number of quantiles is usually much less than the length of the time series ($\eta \ll T$) and the resulting networks are weighted, directed, and contain self-loops. If the number of quantiles is too large the resulting graph may not be connected. On the other hand, QGs present a significant loss of information, represented by large weights assigned to self-loops, when η is small. Shirazi et al. (2009) proposed a method based on the chi-squared statistic to obtain the optimal number of bins (quantiles). Some works define the number of quantiles based on the following relationship with the time series length: $\eta \approx 2T^{1/3}$ (Campanharo et al., 2018). The connectivity of QGs represents the causal relationships contained in the dynamics of the process.

Campanharo and co-authors (Campanharo et al., 2011) reconstructed a time series from a QG using only the information contained in the adjacency matrix and without prior knowledge about the original series. These authors have shown that the resulting series presents statistical properties namely, autocorrelation, power spectrum, and marginal distributions equivalent to those of the original series.

Recently, Campanharo and Ramos (2016) developed a variation of the QGs method in order to estimate the Hurst exponent of fractional noise. This variation involves the introduction of a parameter ϕ such that two nodes v_i and v_j are connected by an edge with weight $w_{i,j}^{\phi}$ whenever two observations Y_t and $Y_{t+\phi}$ belong to the quantiles q_i and q_j, respectively, where $\phi = 1, \ldots \phi_{max} < T$. If $\phi = 1$, the method is reduced to QGs. The Hurst exponent, H, can be estimated as a function of the mean jump length: $\Delta(\phi) \sim \phi^H$, where

$$\Delta(\phi) = \frac{1}{L} \sum_{l=1}^{L} \delta_{l,\phi}(i,j), \tag{3.18}$$

$\delta_{l,\phi}(i,j) = |i-j|$, with $i, j = 1, \ldots, \eta$, is the jumps of length in a random walk on the graph and L is the number of jumps.

Shirazi and co-authors (Shirazi et al., 2009) were able to quantify the effect of long-range correlations and of the marginal distributions, using distance and clustering measures. They tested the method on synthetic white noise data and on real data from turbulence and stock market index series. Liu and Wang (2018) also used this approach to identify behavior patterns of the processes underlying synthetic and real-world time series data from the visual analysis as well as the analysis of the communities of the resulting networks. Campanharo and co-authors (Campanharo et al., 2011) were able to show that time series with different properties are mapped into networks with different topological properties. In particular, as randomness increases in time series, the corresponding QG networks become increasingly random as well, becoming similar to small-world models [9]. Later, Campanharo (2016) showed that the QGs are capable of capturing and quantifying time series derived from long-range correlated processes and chaotic processes. They also verified that as the time series changes from periodic to chaotic, the corresponding networks, initially regular, become more and more random with larger values of the average path length [10] and clustering coefficient. The work developed by Campanharo et al. (2018) and Pineda et al. (2020) takes advantage of the quantile graphs to classify EEG data.

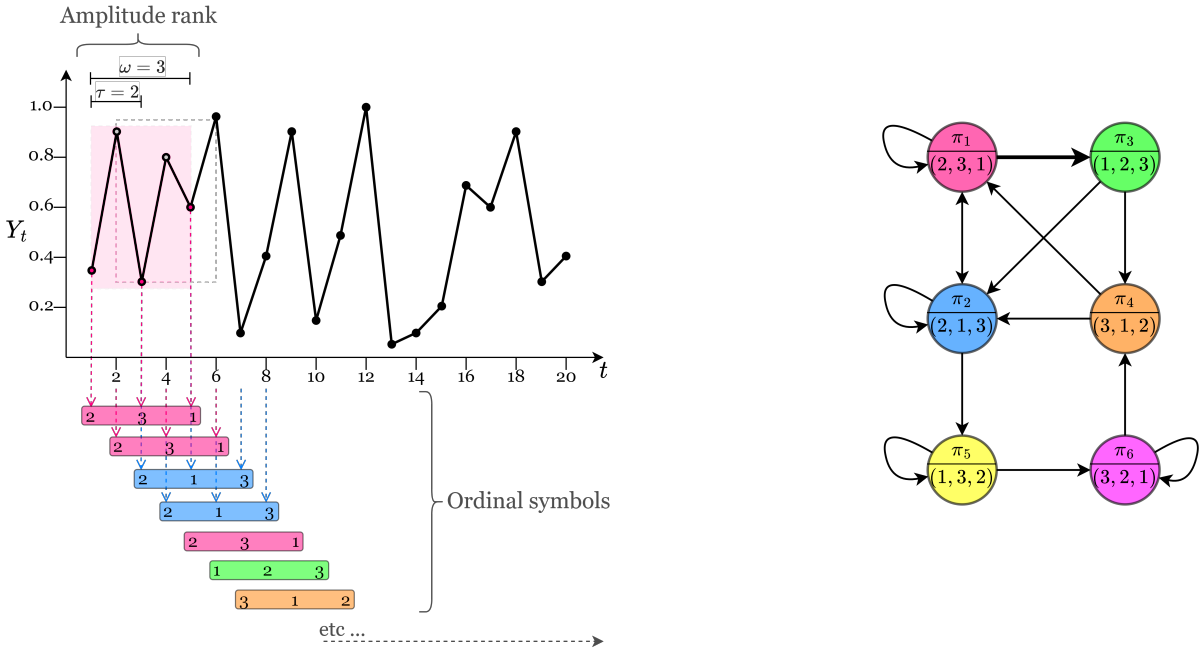
[9] Small-world networks are networks characterized by a high clustering coefficient, like a regular graph, and a low characteristic path length, like a random graph. This means that most nodes have few edges but can be reached from another node through a small number of edges (Watts and Strogatz, 1998).

[10] The arithmetic mean of the shortest path lengths among all pairs of nodes in the graph, where the path length is the number of edges, or the sum of the edge weights if the graph is weighted, in the path.

3.1.2.2 Ordinal Partition Transition Networks

Ordinal pattern methods are based on the idea of a set of sequential patterns defined for a sequence of consecutive observations, where each node of the network represents one of the defined patterns and the edges are weighted according to the transition frequency between two consecutive patterns.

Formally, to construct a network of ordinal patterns (or ordinal partition transition network, OPTN) for a univariate time series the time series is embedded in a w-dimensional space using a time delay τ. For each of these vectors $\vec{z}_i$, define the corresponding ordinal pattern by assigning ranks to the data Y_t in descending order, $\pi_i = (R_1, R_2, \ldots, R_w)$, where $R_j \in \{1, 2, \ldots, w\}$, $R_j \neq R_k$ if $j \neq k$. If the observations Y_j and Y_k for time j and k have equal amplitudes we consider the order of time, and we take the first to occur as smallest (McCullough et al., 2015; Small, 2013) (see Figure 3.7a). Finally, the set of patterns obtained is mapped to nodes in a network where the edges are allocated between nodes based on the transition sequence of the symbols π_i. The weight $(w_{i,j})$ associated with the edge (v_i, v_j) is the probability of transition between consecutive symbols.



(a) Toy time series

(b) OPTN

Figure 3.7: Illustrative example of ordinal partition transition network algorithm. On the left side, we illustrate the method of embedding with window size $w = 3$ and lag $\tau = 2$ and the method of finding its ordinal pattern, based on the amplitude rank of its elements. On the right side, we show the resulting networks.

McCullough and co-authors (McCullough et al., 2015), applied the ordinal partitions method to the Rössler system and to other chaotic series. They found that both the visual topological structure of the networks, time series with periodic behavior are mapped in networks with a ring structure and chaotic time series in networks with the band or tube-like structures, and the simple topological metrics as average *out*-degree and variance of *out*-degrees, convey the dynamics underlying the processes. In particular, the average *out*-degree and the variance of *out*-degrees allow detecting changes in the dynamic behavior in a similar way to the Lyapunov exponent [11], distinguishing dynamics of different states and determining points of change. These networks also proved to be useful for the analysis of real-world time series, such as: series from the externally driven diode resonator circuit (McCullough et al., 2015), and electrocardiograms (ECG) series for analysis of dynamical changes, such as human cardiac dynamics (Cao et al., 2004; McCullough et al., 2017).

A number of studies have begun to emerge around the so-called *forbidden patterns*, which are patterns that do not occur in a time series. These studies have shown that features around these patterns, namely the counting of forbidden patterns, are important to distinguish deterministic time series with very high levels of noise from random series (Amigó et al., 2007), to detect determinism in noisy data (McCullough et al., 2016; Sakellariou et al., 2016), and to distinguish between healthy patients and unhealthy patients with varying heart conditions based in ECG data (Kulp et al., 2016).

Recently, Pessa and Ribeiro (2019) showed that properties of ordinal networks (namely, the average weighted shortest path lengths) can be used for estimating the Hurst exponent of time series with high precision, outperforming the quantile graphs (Campanharo and Ramos, 2016) and the detrended fluctuations analysis (Shao et al., 2012).

3.1.2.3 Coarse-Grained Phase Space Graphs

Networks based on coarse-graining of phase space consist of the idea of reconstructing (creating sets of symbol groups) the phase space of a dynamic process, where each of these symbol groups is mapped into a node of the network and the edges are established between nodes representing successive partitions in time. A weight representing the amount or probability of transition between these partitions (nodes) is assigned to the edge.

The algorithm involves the following steps (Gao and Jin, 2009; Wang and Tian, 2016): start by embedding a time series in a w-dimensional space with a time delay τ, similarly to OPTN (Section 3.1.2.2); then the vectors (points in the phase space) $\vec{z}_t = (Y_t, Y_{t+\tau}, \ldots, Y_{t+(w-1)\tau})$ are classified into $D = w$ different symbol groups ($\Psi_j = \{\psi_i\}_{i=1}^{D}$), according to intervals predefined by a set of values that delimit the phase space $\{\alpha_1, \ldots, \alpha_{D-1}\}$ (Wang and Tian, 2016; Zou et al.,

[11] The Lyapunov exponent, λ, (Lyapunov, 1892) measures the speed with which trajectories in phase space approach (contraction) or move away from (expansion) each other (Kiel and Elliott, 1996). It can be interpreted as a qualitative measure of the sensitivity to the initial conditions of the chaotic systems and, consequently, instability of a process.

2019):

$$\begin{cases} \psi_1 & \text{if } Y_t < \alpha_1, \\ \psi_i & \text{if } \alpha_{i-1} \leq Y_t < \alpha_i,\ 1 < i < D-1 \\ \psi_D & \text{if } Y_t \geq \alpha_{D-1}. \end{cases} \tag{3.19}$$

Note that this transformation process is similar to the transformation underlying the QG algorithm (Section 3.1.2.1), but applied separately to each component of the embedding vector. While in QGs, quantiles delimit the time series support, here the set of values $\{\alpha_1, \ldots, \alpha_{D-1}\}$ delimit the phase space. Finally, each *unique* vector of symbols Ψ_j is mapped into a node of the network. Directed and weighted edges $(v_i, v_j, w_{i,j})$ are defined by the transition probabilities between symbol vectors Ψ_i and Ψ_j.

Wang and Tian (2016) have shown, based on the distribution of weighted degree of nodes, that increasing the dimension w of the phase space leads to enhancing the heterogeneity of the networks and that the method allows distinguishing and characterizing different series, such as white noise, chaotic Lorenz systems[12], and gasoline price series.

Weng and co-authors (Weng et al., 2017) proposed an extension of this algorithm with the purpose of characterizing the memory (short or long) of the process. Taking into account that the temporal information is lost after the coarse-graining transformation and that the transition probability matrix is an estimate of the process dynamics, Weng and co-authors proposed to incorporate the temporal information into an additional topological dimension, such as an edge attribute. In this network, a node v_i is connected to the node v_j with an attribute t, $(v_i, v_j; t)$, when a transition from Ψ_i to Ψ_j occurs at time stamp t. This makes it possible to construct a memory network whose memory effect appears to be able to accurately differentiate various types of time series, namely, white noise, 1/f noise, AR model, and periodic and chaotic time series.

3.1.2.4 Visibility Graphlets Networks

Given the ability to incorporate the topological characteristics (global and local) of the time series in the visibility graphs, visibility graphlets networks were proposed by Mutua et al. (2015) in order to monitor and, consequently, understand the evolutionary behaviors of a time series. The method consists of five fundamental steps (Mutua et al., 2015, 2016). First, a time series is incorporated into a w-dimensional space without time delay ($\tau = 1$), similarly to sliding a window of length w across the time series; each resulting vector $\vec{z}_k = (Y_k, Y_{k+1}, \ldots, Y_{k+w-1})$, with $k = 1, 2, \ldots, T - w + 1$, is mapped to a directed visibility graph (visibility graphlet) G_k, following the algorithms described in Sections 3.1.1.1 and 3.1.1.3, representing the k-th state of the time series; the successive visibility graphs are then connected by a directed edge resulting in the following state chain network: $G_1 \rightarrow G_2 \rightarrow \ldots \rightarrow G_{T-w+1}$. In order to obtain a network of distinguishable states, each graph G_k is, iteratively, compared to the others across the chain

[12] Lorenz's system, as Rössler, is a complex system of three ordinary differential equations (Lorenz, 1963). It defines a chaotic continuous-time process.

network and if any two graphs are identical, i.e., the adjacency matrices are the same, the latter graph is replaced by the former which is the reference state. For example, if $\boldsymbol{A}_1 = \boldsymbol{A}_5$, the graph (state) G_5 is replaced by G_1. Finally, the visibility graphlets network is defined so that the nodes represent the unique graphlets, and the directed and weighted edges are established between pairs of nodes with weight equal to the number of edges between the successive corresponding graphlets in the chain network.

A visibility graphlets network is a directed and weighted transition network where the time series states are represented by visibility graphlets and the edges reflect the relative frequency of states sequences, i.e., the transfer behaviors of the unique states. This network can also be seen as a temporal network (Holme and Saramäki, 2012) and a network of networks (Kivelä et al., 2014), where the corresponding network science methodologies can be used (Mutua et al., 2015). This mapping method is a "mixture" of two underlying concepts, visibility and transition probability, which can benefit from the characteristics mapped by both, namely, structural features that fully describe the states of the time series and the dynamic transfers of the underlying processes. However, like all methods that involve incorporation in the phase space, the selection of the parameter w is a problem. If w is too small, the number of different graphlets is very limited. On the other hand, the number of different graphlets increases geometrically with the increase of w implying that the time series is sufficiently large. See Mutua et al. (2015) for a more detailed discussion.

In Mutua et al. (2015), the authors studied some properties in a set of synthetic and real-world time series, namely, the degree of nodes and transmission probability related to edge weights for short-term prediction, and long-term persistence related to the occurring positions of some motifs on time series. Mutua et al. (2016) used the method in discrete and continuous chaotic time series and showed that the generated networks capture the dynamic properties of the systems, distinguishing chaotic zones that result in networks with a complex structure characterized by hubs and non-chaotic zones, such as periodic ones that result in regular networks.

3.1.3 Proximity Networks

Mappings based on the concept of proximity use measures of distance or similarity to calculate the distance between the points of the time series incorporated in the multidimensional phase space. These methods map states of the time series into nodes of the network and create edges between those nodes based on some measure of distance or similarity. The ability to connect different nodes based on proximity measures allows capturing significant information about the topology of the dynamical systems, enabling to identify, for example, different regimes or patterns along the series via the similarity of successive states (Donner et al., 2011b).

Proximity networks include *cycle networks*, *correlation networks*, and *recurrence networks*.

3.1.3.1 Cycle Networks

Cycle networks were proposed by Zhang and co-authors (Zhang et al., 2006; Zhang and Small, 2006) to represent pseudo-periodic time series.

Formally, to construct a cycle network from a pseudo-periodic time series $\{Y_t\}_{t=1}^{T}$, first segment the series into u consecutive cycles $\{c_1, c_2, \ldots, c_u\}$, not necessarily of the same length, according to the local minima (or maxima). For each pair of cycles $c_i = \{Y_i, \ldots, Y_{l_i}\}$ and $c_j = \{Y_j, \ldots, Y_{l_j}\}$ ($i, j = 1, 2, \ldots, u$ and $i \neq j$) compute the correlation index $r_{i,j}$ defined as the maximum of the cross-correlation between the two cycles (assuming that $l_i \leq l_j$ without loss of generality) (Zhang et al., 2006):

$$r_{i,j} = \max_{l=0,1,\ldots,l_j-l_i} \frac{\mathrm{cov}[(Y_1, \ldots, Y_{l_i}), (Y_{1+l}, \ldots, Y_{l_j+l})]}{\sqrt{\mathrm{var}(Y_1, \ldots, Y_{l_i})}\sqrt{\mathrm{var}(Y_{1+l}, \ldots, Y_{l_j+l})}}, \tag{3.20}$$

where cov is the covariance and var is the variance. This means that, if the cycles are not of the same length, the shortest, c_i is shifted relative to the longest, c_j by $l_j - l_i$ steps and the correlation coefficient between c_i and the corresponding part of c_j, in each step, is calculated. Finally, the highest value is chosen as the correlation coefficient between c_i and c_j. The graph G is constructed by assigning a node v_i to each cycle c_i and defining an undirected and unweighted edge between two nodes v_i and v_j if the correlation index is above a certain threshold $r_{i,j} > \alpha$.

The definition of the edges may be alternatively based on other distance measures, such as

$$d_{i,j} = \min_{l=0,1,\ldots,l_j-l_i} \frac{1}{l_i} \sum_{k=1}^{l_i} \|Z_k - X_{k+l}\|, \tag{3.21}$$

for which thresholds must also be set (Zhang and Small, 2006) and where Z_k and X_k are the k-th point of c_i and c_j in the constructed cycle, respectively.

Zhang and Small (Zhang and Small, 2006) applied cycle networks to the study of the Rössler system and of electrocardiogram (ECG) signals. The authors studied topological metrics of the resulting graphs, such as degree distribution, among others, and concluded that noisy periodic signals are mapped into random networks and chaotic time series into networks that exhibit small-world and scale-free features. In particular, they observed peaks in the degree distribution function of the graph that corresponds to the unstable periodic orbits of the system, the nodes corresponding to these orbits form communities in the network. The metrics studied also allowed to distinguish between the ECG of healthy volunteers, $P(k)$ varies smoothly, and those of unhealthy patients, $P(k)$ shows more prominent variations.

3.1.3.2 Correlation Networks

Correlation networks are essentially constructed based on a correlation measure, such as the correlation matrix of the time series Y_t proposed by Yang and Yang (2008) based on the ideas of functional networks (Eguiluz et al., 2005), correlation of stock markets (Bonanno et al., 2004) and the results obtained by cycle networks (Zhang and Small, 2006). The process consists of first considering individual state vectors $\vec{z}_i$ of time series which are extracted by embedding the series into a sufficiently large w-dimensional space. Each state vector is mapped into a node v_i of an undirected and weighted network where the set of edges $\{(v_i, v_j, w_{i,j})\}$ are established in terms of the Pearson correlation coefficient calculated between the corresponding state vectors $(\langle \vec{z}_i, \vec{z}_j \rangle)$. Note that the Pearson correlation takes values between -1 and 1, so a negative weight may be assigned to the edges. However, as already mentioned, network analysis with negative weights is unusual and so the weights of correlation networks are usually established as the absolute value of the correlations, $w_{i,j} = |\rho_{i,j}|$. The correlation matrix $[|\rho_{i,j}|]$ is the matrix of the correlation network, and note that other measures of correlation may be considered.

As with cycle networks, a threshold r_c also can be considered and a binary adjacency matrix can be produced. This threshold must be chosen properly because it determines the characteristics of the resulting network. If it is extremely small, pairs with weak correlations are also connected (noise). But if it is too large important information may be lost too (Yang and Yang, 2008).

Correlation networks were applied to the stock price series (return and amplitude series) resulting in a degree distribution function that follows a perfect Gaussian distribution. Additionally, the return series point to a random behavior, indicating that the global correlation characteristic can be modeled by an Erdős-Rényi network (Yang and Yang, 2008). The authors used the degree distribution to decide the best parameters for the algorithm. In contrast, Feng and He (Feng and He, 2017), used the cross-correlation as a measure of similarity between two points in phase space and used the clustering coefficient and efficiency to decide the best parameters for the model. They analyzed the Lorenz system, white Gaussian noise, and sea clutter time series. The results show that, for an unknown complex system, dynamic states, i.e., the behavior changes of the system, can be discovered by studying the community structures of complex networks. In particular, Feng and He identified changes in behavior from the contractive state to the open state in the sea clutter time series and the white Gaussian noise did not identify clear community structures, the behavior changes of white Gaussian noise are completely random and therefore the resulting networks do not present a topology in communities.

3.1.3.3 Recurrence Networks

Recurrence networks are quite popular in the research community. These networks are based on recurrence plots, a tool used in the study of dynamical (nonlinear or chaotic) systems (Marwan, 2008). For constructing a recurrence plot for a dynamical system from an observed time series first embed the time series in a w-dimensional space (phase space reconstruction) thus defining

a set of vectors $\vec{z}_i$. Then define a matrix $\boldsymbol{A}$, also called the recurrence matrix, where the elements (i, j) are defined based on the proximity relation in phase space of the vectors $\vec{z}_i$ and $\vec{z}_j$ (Eckmann et al., 1987). The proximity relation can be defined by different criteria, namely, fixed (probability) mass, where a fixed amount of nearest neighbor vectors is established, or fixed volume, where an ε-threshold of proximity is established. The matrix $\boldsymbol{A}$ is commonly defined with elements:

$$A_{i,j} = \begin{cases} 1 & \text{if } \|\vec{z}_i - \vec{z}_j\| \leq \varepsilon \\ 0 & \text{otherwise} \end{cases}, \tag{3.22}$$

where ε is a distance threshold and $\|\cdot\|$ can be any norm in phase space (for example, Euclidean, Manhattan, or maximum norm) (Marwan et al., 2007). The matrix $\boldsymbol{A}$ is a symmetric ($A_{i,j} = A_{j,i}$) matrix with constant values on the diagonal ($A_{i,i} = 1$), and it can be interpreted as the adjacency matrix of an undirected and unweighted network, the recurrence network, where the nodes represent $\{\vec{z}_i\}$ and the edges are defined by $A_{i,j}$, usually the self-loops are removed ($A_{i,i} = 0$). A direct generalization is considering the associated distance matrix between pairs of states, resulting in a weighted network. This type of network, both weighted and unweighted, are spatial networks, where the nodes (states) are located in an m-dimensional space equipped with a norm in the phase space and which is reflected in the topological characteristics/features of the networks (Donner et al., 2010). In the unweighted network's case, the probability of finding an edge between two states will decrease with the distance. For weighted networks, the direct use of the distance matrix is generally avoided because the more distant/dissimilar two states are, the stronger the connection between them. To avoid this other alternatives can be used, such as the inverse of the distance $w_{i,j} = \frac{1}{\|\vec{z}_i - \vec{z}_j\|}$ (Strozzi et al., 2011). Recently, a general alternative, that can be applied to any kind of network, was presented to recurrence networks where weights are given by: $w_{i,j} = \frac{\sqrt{k_i k_j}}{k_{max}}$, k_{max} is the maximum degree in the unweighted recurrence network (Jacob et al., 2019).

The construction of these networks implies that they do not incorporate any temporal information on the underlying process and therefore do not explicitly depend on the presence of equally spaced observations, which is a major issue in the analysis of time series observed in many phenomena (Donner et al., 2011b). Recurrence plots are also one of the few time series analysis techniques that work well with non-stationary time series data. However, in the case of more complex systems, the rich geometric structure of recurrence plots can become difficult to interpret, for example, for chaotic systems this happens due to their unstable periodic orbits (Bradley and Kantz, 2015). The characteristics and topological properties of recurrence networks depend essentially on their construction, namely on the following parameters: the dimension of states w, time delay τ and threshold ε, and the measure of proximity between states. It is important to note $w = 1$ meaning that the embedding in the phase space is not required in which case the distance is computed between the points of time series (Marwan et al., 2009; Zhao et al., 2020).

The literature essentially highlights three types of recurrence networks, the *κ-nearest neighbor networks*, the *adaptive nearest neighbor network*, and the *ε-recurrence networks*, which we describe in more detail below.

κ-Nearest Neighbor Networks, κ-NNN

The κ-nearest neighbor version imposes a constraint on the quantity κ of the nearest neighbor points (nodes v_j) of a given point in the phase space (node v_i). This implies a direction in the edge (v_i, v_j) since it is not necessarily true that, if the node v_j is one of the κ the closest neighbors of the node v_i then v_i is also one of the κ closest neighbors to v_j, and so the adjacency matrix becomes an asymmetric matrix. This method was proposed by Small and co-authors (Small et al., 2009), in order to overcome the issue of cycle network mappings that requires time series with an oscillatory nature.

κ-NNN imply that all nodes have the same amount κ of *out*-edges, $k_i^{out} = \kappa$, preserving a constant mass of the considered neighborhoods, i.e., the number of nodes is the same in all neighborhoods (Donner et al., 2011b). However, the distribution of *in*-edges k_i^{in} is allowed to vary but the average degree is $\bar{k}^{in} = \kappa$. In particular, if $k_i^{in} \ll \kappa$, the node v_i must be in a phase space region with decreased density compared to the remaining, and if $k_i^{in} \gg \kappa$, v_i must be located in a densely populated region (Donner et al., 2011b), thus giving information about the local geometry of the phase space.

Xiang et al. (2012) studied the distribution of network motifs in κ-NNN and were able to classify different dynamic behaviors (such as maps and flows) using the frequency of motifs. These authors have also shown that degree variations accompany the changes in the dynamics throughout the bifurcation process of the Rössler system.

Adaptive Nearest Neighbor Networks, ANNN

In order to overcome the imposition that all nodes have the same $k_i^{out} = \kappa$ in κ-nearest neighbor networks, a new version, called adaptive nearest neighbor networks, was proposed (Small et al., 2009; Xu et al., 2008). In ANNN if v_j is one of the κ closest neighbors of v_i, then v_i is also considered a neighbor of v_j, even if it is not one of the κ closest neighbors to v_j. This leads to a symmetric adjacency matrix and the node degree to be variable, unlike κ-NNN, since there may be more than κ neighbors to a given node. Xu et al. (2008) and Liu and Zhou (2010) have shown that the motif distributions allow characterizing different processes, namely periodic, chaotic, noise and fractional, creating super-families. Unique fingerprints have also been found for specific dynamical systems within a family. Additionally, Liu and Zhou (2010) analyzed three stock market indexes and concluded that the motif distributions are equivalent for the three series (since they have the same dynamic behavior characteristic of return stock series) and are also very similar to the distributions obtained for the fractional Brownian motion [13].

[13] Fractional Brownian motion is a continuous-time Gaussian process where the increments need not be independent. The Hurst exponent, H, describes the raggedness of the motion, $H \in [0, 1]$, the higher the value is smoother (correlated) motion (Zunino et al., 2007). It is an important measure that quantifies the correlation of a time series, and it is used as a measure of long-term memory, the persistence of the process (Zunino et al., 2007).

ε-Recurrence Networks

Equation (3.22) suggests that the "neighborhood" of a single state vector $\vec{z}_i$ can be defined by a fixed distance in the phase space ε, considering fixed volumes (communities) in phase space (Donner et al., 2011b). Thus, an undirected and unweighted network can be built.

This version implies choosing the distance threshold ε which allows controlling of the phase space resolution. If ε is too small, the volume of the neighborhood will be small and therefore there will be almost no recurrence points and the information incorporated in the network will be insufficient. On the other hand, if ε is too large we observe a general qualitative change in the network topology (Jacob et al., 2016), namely, each node will behave like a hub, leading to an excess of recurring points and misleading information.

Donner and co-authors (Donner et al., 2010, 2011b) studied properties of ε-recurrence networks at three different scales, namely local, intermediate and global on several paradigmatic systems: Hénon map, Bernoulli map, Lorenz system and Rössler system. The authors studied graph properties as a function of the distance threshold ε and proposed specific features like the local clustering coefficient to detect dynamically invariant objects, saddle points, or unstable periodic orbits. Moreover, those authors suggested varying the embedding dimension as a means to distinguish between chaotic and stochastic systems. Zou et al. (2012) have shown that networks created for one-dimensional maps with local power-law in the invariant density result in scale-free networks since the degree distribution is scale-free. Overall, the exponent of the resulting degree distribution does not need to coincide with the fractal dimension. Thus, Donner et al. (2011a) demonstrated that the local and global transitivity of the networks is closely related to a generalized notion of fractal dimensionality (local and global, respectively).

ε-recurrence networks were the most exploited and led to the well-established relationships between some topological metrics of networks and the properties and measurements of phase space. Donges and co-authors (Donges et al., 2012) propose an analytical framework for ε-recurrence network analysis describing graph-theoretical recurrence network quantifiers. This framework shows that several standard features of network analysis can represent discrete estimators of continuous measures of certain complex phase space properties. A theoretical relation is established between features of the recurrence networks and phase space properties. ε-recurrence networks were also widely used to determine changes in the dynamics of theoretical (Iwayama et al., 2013; Marwan et al., 2009) and real world (Donges et al., 2011b; Fukino et al., 2016) systems.

3.2 Multivariate Time Series Mappings

So far we have focused on approaches to mapping univariate time series into network structures. However, technological developments are producing a wealth of inter-connected multidimensional data, such as multivariate spatio-temporal data. Tools for the analysis of these high dimensional datasets are yet scarce, hinting at the possibility of using network science approaches. The literature on mapping multivariate time series to the network domain, summarized in Table 3.2, is not as developed as for the univariate case. Even so, we can distinguish two classes of methods, introduced in Figure 3.1: those that map the multivariate time series into a single-layer (or monoplex) network and those that map the multivariate time series into a multiplex[14] network. The mappings in the first class construct networks with nodes representing the (component) time series and edges representing the relationship between the node time series, computed as, e.g., causal relation. Mappings on the second class lead to multiplex networks where each layer is a network resulting from a univariate time series mapping. The layers usually have the same nodes and are connected via the edges which connect the same node across adjacent layers. Details on each approach and corresponding core results are given in the following sections.

3.2.1 Single Layer Networks

3.2.1.1 Correlation Networks

Formally, a correlation network (CN) is defined as an undirected and weighed graph $G = (V, E)$ where $V = \{Y_{i,t}\}_{i=1}^{m}$ and $E = \{(v_i, v_j, w_{i,j}) | (v_i, v_j) \in V \wedge w_{i,j} = \rho_{i,j}(0) \wedge i \neq j\}$ (see in Section 2.1.2). Thus, inter-dependencies between the m time series are represented by the (contemporaneous) correlation.

Other approaches to represent the inter-dependencies and establish the edges of the network rely on suitable correlation measures such as cross-correlations (Nakamura et al., 2016) or partial correlations (Epskamp and Fried, 2018), eventually subject to thresholding. The thresholding (edges are established only if the correlation exceeds a predefined value) helps to remove spurious edges. Alternatively, correlations can be replaced by any similarity measure, including distance measures. The resulting similarity/distance matrix is then used to construct a monoplex network (Mori et al., 2016).

Figure 3.8 presents an example of a correlation network representation of a toy multivariate time series. Nodes that represent similar time series are linked by thicker edges.

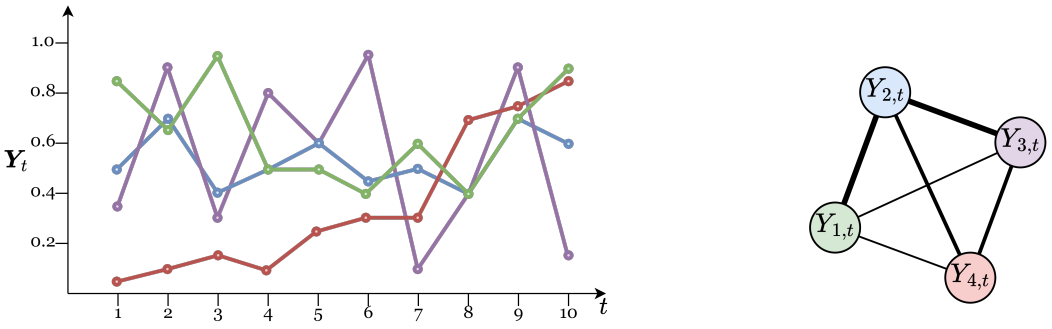
[14] Remember that a multiplex network is just a particular case of multilayer networks so the terms are not equivalent, see Section 2.3.2.

	Network Type (original reference)	Node	Edge	Directed	Weighted	Param-free	Pre pross
Single Layer	Correlation Network (Eguiluz et al., 2005)	$Y_{\alpha,t}$	CM	✗	✓	✓	✗
	Long-Run Variance Decomposition Network (Diebold and Yılmaz, 2014)	$Y_{\alpha,t}$	VD	✓	✓	✗	VAR
	Causal Effect Network (Runge et al., 2015)	t	LCR	✓	✓	✗	CDA
	Ordinal Partition Transition Network (Ruan et al., 2019; Zhang et al., 2017a)	π_i	TP	✓	✓	✗	PS
	Pattern Interdependent Network (Ren et al., 2020)	G_i	TP	✓	✓	✓	PS;VG
	Inter-system recurrence network (Feldhoff et al., 2012)	$\vec{z}_i^{[\alpha]}$	DM	✗	✗	✗	PS
	Joint Recurrence Network (Feldhoff et al., 2013)	$\vec{z}_i^{[\alpha]}$	DM	✗	✗	✗	PS
Multiple Layer	Multiplex Visibility Network (Lacasa et al., 2015)	t	V	✗	✗	✓	✗
	Multiplex Recurrence Network (Eroglu et al., 2018)	t	DM	✗	✗	✗	PS
	Multiplex Directed Visibility Network (Flori et al., 2021)	t	V	✓	✗	✓	✗

Table 3.2: Comparison of (multivariate) time series mappings based on the properties of the corresponding algorithms and of the resulting networks. Notation: CM - correlation measures, VD - variance decomposition, LCR - lagged causal regression, TP - transition probability, DM - distance measures, V - natural and/or horizontal visibility, VAR - vector autoregression model, CDA - causal discovery algorithm, PS - phase space, VG - directed visibility graph.

Correlation networks (also known as functional networks) have been widely used in neuroscience (Bullmore and Sporns, 2009; Eguiluz et al., 2005), financial (Cai et al., 2010; Gao et al., 2015; Tumminello et al., 2010) and climate science (Dijkstra et al., 2019; Tsonis and Swanson, 2012) areas. In particular, Eguiluz and co-authors (Eguiluz et al., 2005) applied this approach to functional fMRI data in order to connect brain zones based on their similarities and their relationships. They have shown that the degree distribution obeys a power law, indicating scale-free networks which imply that there are always a few zones of the brain (hubs) with relations to most other regions of the brain, that is, predominant zones.

Gao and co-authors (Gao et al., 2015) used correlation networks to analyze the interactions between companies of the different sectors using clustering metrics. The objective is also to study the influence that some companies and/or sectors have on others. The results show important information about fluctuations of the series and show that correlation networks can be very useful for financial risk analysis.



(a) Toy time series

(b) CN

Figure 3.8: Illustrative example of the correlation network algorithm. On the left side we present the plot of a toy multivariate time series and on the right side the network generated by the correlation algorithm (using contemporaneous cross-correlation). The different colors represent the time series. Higher correlation values result in edges in the network with larger weights represented by thicker lines.

3.2.1.2 Long-Run Variance Decomposition Networks

The mapping of multivariate time series into correlation networks is based on the contemporary dependence of time series, but in multivariate time series, cross-sectional dependencies may arise in different leads/lags (Lanne and Nyberg, 2016). We introduce now a class of networks that reflect, in the directed and weighted edges, the presence of contemporaneous as well as lagged partial correlations between time series $Y_{i,t}$ and $Y_{j,t}$. The edges of the network are established via VAR models and their analysis, namely Forecast Error Variance Decomposition (FEVD). The process $\boldsymbol{Y}_t$ follows a VAR(p) model, where p is the order of the VAR if it satisfies the following equation:

$$\boldsymbol{Y}_t = \boldsymbol{\varphi} + \sum_{i=1}^{p} \boldsymbol{\phi}_i \boldsymbol{Y}_{t-i} + \boldsymbol{\epsilon}_t, \tag{3.23}$$

where $\boldsymbol{\varphi}$ is an m-vector of constants, $\boldsymbol{\phi}_i$ $(i = 1, 2, \ldots, p)$ is an $m \times m$-matrix of coefficients to be estimated and $\boldsymbol{\epsilon}_t$ is an m-vector of uncorrelated errors with zero mean and covariance matrix $\boldsymbol{\Sigma}$, with elements $\sigma_{i,j}$. Under stationarity conditions $\boldsymbol{Y}_t$ may be represented as an infinite moving-average representation (MA model):

$$\boldsymbol{Y}_t = \sum_{j=0}^{\infty} \boldsymbol{\Theta}_j \boldsymbol{\epsilon}_{t-j}. \tag{3.24}$$

The FEVD, carried out typically on the infinite moving-average representation of the VAR process, calculates the proportion of the prediction error variance of the i-th variable which is assignable to its own (lagged) shocks and to the (lagged) shocks of the other variables. A problem with the FEVD is that it is dependent of the ordering of the variables. To overcome this issue, Diebold and Yılmaz (2014) and Lanne and Nyberg (2016) propose to use the generalized

forecast error variance decomposition (GFEVD)[15] which is independent of the ordering of the variables. Thus, a long-run variance decomposition network (LVDN) is a network where the nodes v_i and v_j represent the time series $Y_{i,t}$ and $Y_{j,t}$, respectively, and the edges $(v_i, v_j, w_{i,j})$ are given by $w_{i,j} = \frac{\theta^h_{i,j}}{\sum_{j=1}^{T} \theta^h_{i,j}}$, where $\theta^h_{i,j}$, denotes the (i,j)-th h-step ahead variance decomposition component, that is, the fraction of variable i h-step ahead forecast error variance due to shocks in variable j (Diebold and Yılmaz, 2014), calculated as:

$$\theta^h_{i,j} = \frac{\sigma^{-1}_{j,j} \sum_{k=0}^{h-1} (\boldsymbol{e}_i' \boldsymbol{\Theta}_k \boldsymbol{\Sigma} \boldsymbol{e}_j)^2}{\sum_{k=0}^{h-1} (\boldsymbol{e}_i' \boldsymbol{\Theta}_k \boldsymbol{\Sigma} \boldsymbol{\Theta}_k' \boldsymbol{e}_i)}, \tag{3.25}$$

where $\boldsymbol{e}_j$ is a vector of orthogonalized shocks with the j-th element unity and zeros elsewhere. Note that the adjacency matrix $D^h = [\theta^h_{i,j}]$ is then normalized.

Diebold and Yilmaz (Diebold and Yılmaz, 2014) applied the method to 13 years of major US financial institutions' stock return volatilities, and they found qualitative relations between the features of *in* and *out*-degree with traditional economic risk features. Given the interpretability and qualities of the results obtained, Barigozzi and Hallin (Barigozzi and Hallin, 2017) proposed an improvement of this approach to make it applicable in the analysis of large sets of time series. Note that the LVDN involves the estimation of VAR models, which becomes unbearable for large sets of time series. This approach involves the use of a generalized dynamic factor model (Barigozzi and Hallin, 2016).

In finance, these methods have been widely applied to study interconnections between financial institutions to identify possible contagion channels. The main disadvantage of these methods is that they are heavily parameterized, as they are based on time series models that involve the estimation of several parameters and restrictive assumptions, and this is one of the major problems of the analysis of multivariate series.

3.2.1.3 Causal Effect Networks

Causal effect networks (Runge et al., 2015), like LVDNs, allow to encode of the inter-relations between components of multivariate time series in different time lags and to distinguish the directionality of these relations. The corresponding graphs are directed and weighed, the nodes represent the individual observations $Y_{i,t}$ with $i = 1, \ldots, m$ at each time t, and the edges are established based on a causal discovery algorithm[16] (Runge et al., 2019a). More specifically, the causal discovery algorithm is only used as a variable selection for a subsequent lagged causal regression and the construction of the network consists of the following three main steps. First, we select causal parents, $\mathcal{P}(Y_{j,t})$, for each component, $Y_{j,t}$, using the causal discovery algorithm (Runge et al., 2014) that iteratively tests the conditional correlation between $Y_{j,t}$ and the remaining components at a range of time lags $0 < h \leq h_{max}$, h_{max} is a maximum time

[15] See Pesaran and Shin (1998) for more details.

[16] There are several methods of causal discovery based on different approaches, examples include approaches based on Granger causality, structural causal models, among others. For an overview of methodological frameworks and challenges of these approaches see Runge et al. (2019a).

lag. Then we estimate the lagged causal regression matrix $\boldsymbol{C}(h)$ of shape (m, m, h_{max}) using the selected parents:

$$C_{i\to j}(h) = b_{j,i\cdot\mathcal{P}(Y_{j,t})}(h) \quad \text{for } h = 1,\ldots,h_{max} \text{ and } i,j = 1,\ldots,m, \tag{3.26}$$

where b is the standardized regression coefficient of $Y_{i,t-h}$ in the multiple regression model of $Y_{j,t}$ on $\{Y_{i,t-h}, \mathcal{P}(Y_{j,t})\}$ using ordinary least squares regression (Runge et al., 2015). Finally, we construct the causal effect graph where the edges are established from the threshold of the causal regression matrix $|C_{i\to j}(h)| \geq \theta$ chosen to obtain a given link density. Self-loops are not counted and multiple edges are only counted once. Note that the advantage of causal effect networks over LVDNs is that the latter can exploit sophisticated versions of conditional mutual information and therefore do not need to resort to parametric time series models.

Causal effects networks also arise from the need to eliminate possible spurious edges added by pairwise association measures, such as cross-correlation. These edges result from transitivity effects (leading to indirect paths) or other processes making it difficult to analyze the causal interactions among multiple nodes (Runge et al., 2015). However, the construction of the causal effect networks implies several assumptions given the underlying statistical methods: causal sufficiency (common drivers of all variables are taken into account), causal Markov condition (all error terms of the nodes in the graph are independent), and stationarity. Note that different causal discovery methods can lead to different assumptions (Runge et al., 2019a).

This network mapping approach can be complemented by multiple information at the edges, such as weights indicating causal edge strengths and additional attributes with the associated time lags. This also allows an extension from a single-layer to a multilayer network, where each layer would correspond to a different time lag h.

Causal effect networks have been extensively explored for the analysis of climate data, especially in complex spatio-temporal systems (Kretschmer et al., 2016; Runge et al., 2015, 2019a,b). In particular, Runge and co-authors (Runge et al., 2015), introduce novel network features based on causal effect theory, which differ from the standard complex network tools by distinguishing direct from indirect paths, in order to identify components with high cumulative causal effect either as sources or as intermediate nodes on path of the causal network.

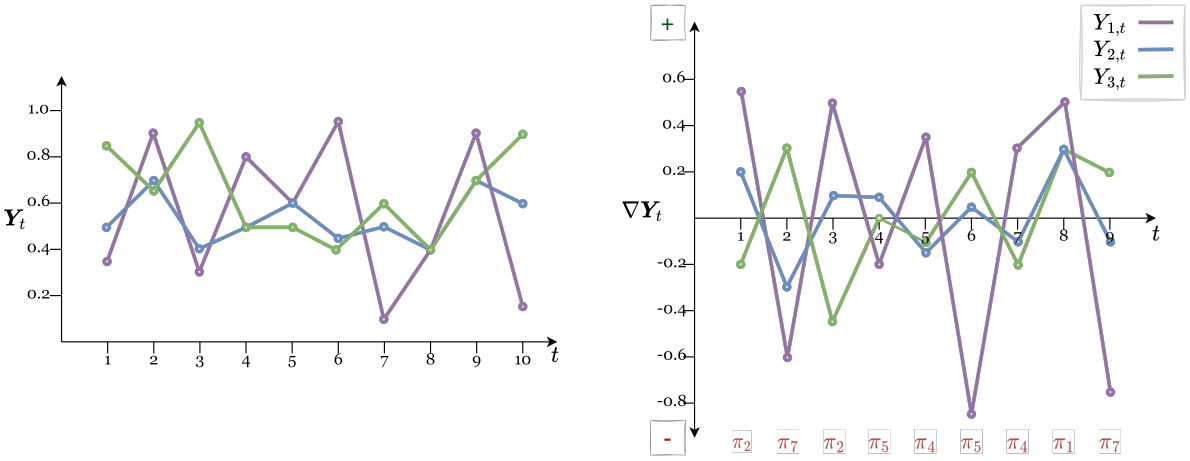
3.2.1.4 Ordinal Partition Transition Networks

The success of OPTN presented in Section 3.1.2.2 led to the extension of these types of networks to the multivariate context. Zhang and co-authors (Zhang et al., 2017a), built the first ordinal partition transition networks for multivariate time series $\{Y_{i,t}\}_{i=1}^{m}$, with an algorithm that differs from the univariate case. Start by constructing the set of order patterns $\Pi = \{\pi_j\}$ with $j = 1,\ldots,2^m$, where each π_j represents a pattern set $(\pi^{\gamma}_{Y_{1,t}},\ldots,\pi^{\gamma}_{Y_{m,t}})$, $\gamma \in [0,1]$, with $\pi^{1}_{Y_{i,t}}$ capturing the increasing trend and $\pi^{0}_{Y_{i,t}}$ the decreasing trend of the time series $Y_{i,t}$. In short, 2^m different combinations of order patterns are constructed depending on the signs of first-order differences of the m time series, as exemplified in Table 3.3 for $m = 3$.

Π	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8
∇Y_1	$\pi^1_{Y_1},+$	$\pi^1_{Y_1},+$	$\pi^1_{Y_1},+$	$\pi^1_{Y_1},+$	$\pi^0_{Y_1},-$	$\pi^0_{Y_1},-$	$\pi^0_{Y_1},-$	$\pi^0_{Y_1},-$
∇Y_2	$\pi^1_{Y_2},+$	$\pi^1_{Y_2},+$	$\pi^0_{Y_2},-$	$\pi^0_{Y_2},-$	$\pi^1_{Y_2},+$	$\pi^1_{Y_2},+$	$\pi^0_{Y_2},-$	$\pi^0_{Y_2},-$
∇Y_3	$\pi^1_{Y_3},+$	$\pi^0_{Y_3},-$	$\pi^1_{Y_3},+$	$\pi^0_{Y_3},-$	$\pi^1_{Y_3},+$	$\pi^0_{Y_3},-$	$\pi^1_{Y_3},+$	$\pi^0_{Y_3},-$

Table 3.3: Order patterns in multivariate time series with three variables $(Y_{1,t}, Y_{2,t}, Y_{3,t})$. (Adapted from Zhang et al., 2017a)

Then apply the first-order differences $\nabla Y_{i,t} = Y_{i,t} - Y_{i,t-1}$, to each of the m time series. Note that this corresponds to considering the time series of changes and that much non-stationary time series become stationary with just this procedure. At each time t, associate the order pattern of the increasing trend with the positive sign of the difference $\nabla Y_{i,t}$ or the order pattern of the decreasing trend with the negative sign of the difference $\nabla Y_{i,t}$ thus constructing the order pattern $\pi_{j_1}, \ldots, \pi_{j_{T-1}}, j_1, \ldots, j_{T-1} \in \{1, \ldots, 2^m\}$, see Figure 3.9 for an example.



(a) Toy multivariate time series

(b) Ordinal pattern association

Figure 3.9: (A) Illustrative example of a toy three-dimensional series $(Y_{1,t}, Y_{2,t}, Y_{3,t})$. (B) First-order differences and corresponding ordinal patterns.

Finally, the ordinal partition transition network is the network with 2^m nodes representing the patterns π_j and with directed weighted edges $(v_i, v_j, w_{i,j})$ established based on the frequency of the transitions from pattern π_i to pattern π_j. In Zhang et al. (2017a), the authors also propose a measure of entropy to characterize ordinal partition transition dynamics, useful for capturing local geometric changes of trajectories. Those authors have shown that for periodic multivariate time series, with different combinations of periods, the resulting network presents clear evidence of possible forbidden patterns and so the network has several disconnected nodes. This shows the determinism of the time series, while random uniform noise multivariate time series are mapped into a complete connected graph.

Guo and co-authors (Guo et al., 2018) propose the *cross ordinal pattern transition network* and the *joint ordinal pattern transition networks* in order to compare the relative rate of change between the two processes by the signs of $(\nabla Y_{i_1,t} - \nabla Y_{i_2,t})$ or $(\nabla Y_{i_1,t} \cdot \nabla Y_{i_2,t})$, respectively. Note that the last sign is related to the signs of changes in the component series while the former is related to the amplitude of the changes.

Recently, Ruan and co-authors (Ruan et al., 2019) proposed a method of ordinal partition transition networks to analyze bivariate time series, $\{Y_{i,t}\}_{i=1}^{2}$, based on the initial method proposed for the analysis of univariate series (see Section 3.1.2.2). The method consists in generating the ordinal partition transition network (following the algorithm presented in Section 3.1.2.2) for each time series $Y_{1,t}$ and $Y_{2,t}$ and extracting the successive sequence of ordinal patterns of each series, removing self-transitions (self-loops in the network). Having the two sequences of ordinal patterns $\pi_i^{Y_{1,t}}$ and $\pi_j^{Y_{2,t}}$, as shown in Figure 3.10a, the next step is to calculate at each time t the frequencies of co-occurrence of the ordinal patterns of $Y_{1,t}$ with the ordinal patterns of $Y_{2,t+h}$ i.e., $p(\pi_j^{Y_{2,t+h}} | \pi_i^{Y_{1,t}})$. In particular, when $h = 0$ we analyze the simultaneous co-occurrence of ordinal patterns, and when $h \neq 0$ we analyze the possible indications of causal relationships between the two systems. The resulting network has two different types of nodes corresponding to the ordinal patterns $\pi_i^{Y_{1,t}}$ of $Y_{1,t}$ and $\pi_j^{Y_{2,t}}$ of $Y_{2,t}$, respectively, and have directed and weighted edges $(\pi_i^{Y_{1,t}}, \pi_j^{Y_{2,t}}, w_{i,j})$ with $w_{i,j} = p(\pi_j^{Y_{2,t+h}} | \pi_i^{Y_{1,t}})$. This results in a bipartite OPTN (Figure 3.10b).

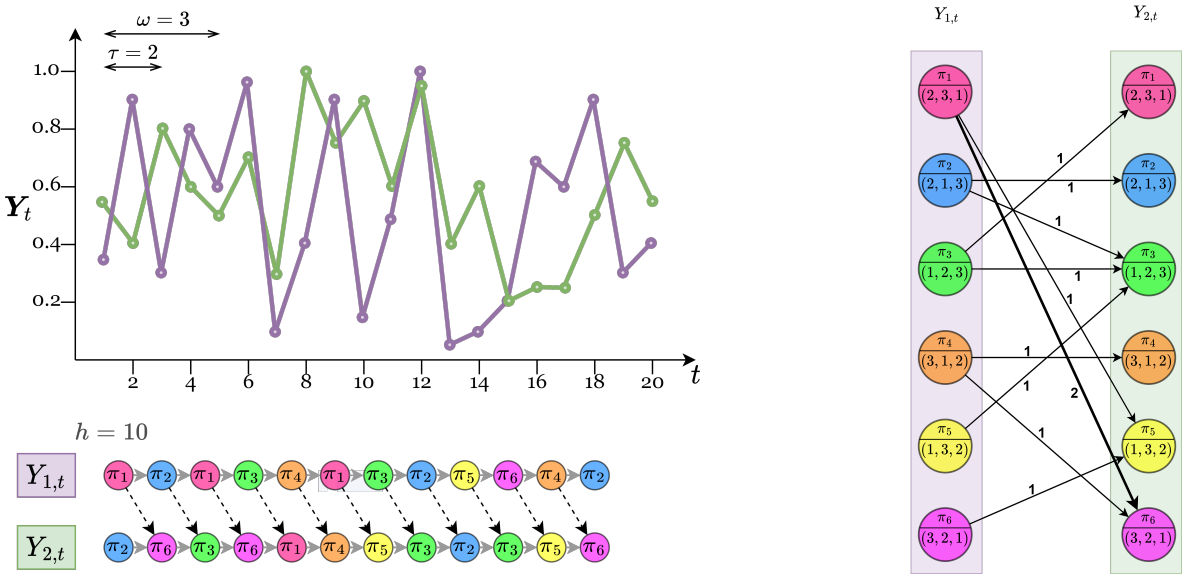
Compared to the univariate case, the bipartite OPTN measures co-occurrence probabilities between symbols in two series rather than succession probabilities of symbols in a single time series. Ruan and co-authors (Ruan et al., 2019) have also introduced a set of OPTN-based complexity features to infer the coupling direction and infer causality between two systems. Those authors used these features in coupled stochastic processes and in climate time series.

3.2.1.5 Pattern Interdependent Networks

Based on the visibility graphlets networks (Section 3.1.2.4) recently, Ren et al. (2020) proposed the pattern interdependent networks to represent the cross-correlation patterns in a stationary bivariate time series, $\{Y_{i,t}\}_{i=1}^{2}$. The method consists in generating the state chain network (following the algorithm presented in 3.1.2.4) for each time series component, resulting in a bi-graphlet series:

$$\begin{pmatrix} G_1^1 \rightarrow G_2^1 \rightarrow \ldots \rightarrow G_{T-w+1}^1 \\ G_1^2 \rightarrow G_2^2 \rightarrow \ldots \rightarrow G_{T-w+1}^2 \end{pmatrix}.$$

Similar to visibility graphlets networks, all unique graphlets G_i are identified. From all the co-occurrent pairs G_k^1 and G_k^2, with $k = 1, 2, \ldots, T - w + 1$, the graphlet co-occurrent frequency matrix, $\boldsymbol{A}$, is generated by calculating the co-occurrent frequencies of all the possible pairs of the unique graphlets, where the element $A_{i,j}$ represents the occurring frequency of $\begin{pmatrix} G_i \\ G_j \end{pmatrix}$ in $\begin{pmatrix} G_k^1 \\ G_k^2 \end{pmatrix}$. $\boldsymbol{A}$ represents a bipartite network where the rows correspond to the total of unique graphlets (nodes) across the time series $Y_{1,t}$ and the columns to the unique graphlets (nodes) across the $Y_{2,t}$,



(a) Toy bivariate time series and sequence of ordinal patterns

(b) Bipartite OPTN

Figure 3.10: (a) Illustrative example of a toy bivariate time series $(Y_{1,t}, Y_{2,t})$ and its ordinal pattern definitions and evolution. We show an embedding in a 3-dimensional space using a time delay $\tau = 2$ and the schematic illustration of the OPTN analysis of the two series of ordinal patterns with a unidirectional coupling $Y_{1,t} \rightarrow Y_{2,t}$ with a coupling delay of $h = 1$. The time-lagged conditional co-occurrences of the patterns $\pi_i^{Y_{1,t}}$ and $\pi_j^{Y_{2,t+h}}$ are indicated by dashed arrows. (b) The network generated by the ordinal partition transition algorithm proposed in Ruan et al. (2019).

and the edges are only established between different node types (similar to the bipartite OPTN illustrated in Figure 3.10b). Since the sets of nodes, unique graphlets G_i of each time series, are identical, they are merged and the bipartite network is converted into a directed and weighted network that represents the co-occurrent relationships between the unique visibility graphlets.

Ren and co-authors (Ren et al., 2020) applied this method to synthetic bivariate time series and showed that a set of unique graphlets and the topological structure of the resulting networks is determined and dependent on the cross-correlation and that the differences in features, such as Hurst exponent, of the time series components determine the symmetry of the edges of the network.

3.2.1.6 Inter-System Recurrence Networks

Two extensions of recurrence networks (Section 3.1.3.3) have been proposed in the literature, one directed to bivariate time series and the other to multivariate time series (Feldhoff et al., 2012, 2013). As in recurrence-based methods for univariate time series, we embed each time series $Y_{\alpha,t}$ with $\alpha = 1, \ldots, m$ in a w-dimensional, defining a set of vectors $\vec{z}_i^{[\alpha]}$ $(i = 1, \ldots, N_\alpha)$.

Cross-recurrence networks are based on cross-recurrence plots that aim to compare the dynamics of two time series $Y_{\alpha,t}$ and $Y_{\beta,t}$ in the same phase space (Marwan and Kurths, 2002; Zbilut et al., 1998). So, both time series are simultaneously embedded in the same phase space and the cross-recurrence plot is defined by the matrix $\boldsymbol{CR}^{[\alpha\beta]}$ with elements (Marwan and Kurths, 2002):

$$CR_{i,j}^{[\alpha\beta]} = \begin{cases} 1 & \text{if } \|\vec{z}_i^{[\alpha]} - \vec{z}_j^{[\beta]}\| \leq \varepsilon \\ 0 & \text{otherwise} \end{cases}, \tag{3.27}$$

where $i = 1, \ldots, N_\alpha$, $j = 1, \ldots, N_\beta$, and ε is a distance threshold in the joint phase space of both processes. $\boldsymbol{CR}^{[\alpha\beta]}$ is asymmetric, since $\|\vec{z}_i^{[\alpha]} - \vec{z}_j^{[\beta]}\| = \|\vec{z}_i^{[\beta]} - \vec{z}_j^{[\alpha]}\|$ does not hold for all i, j, α, β, and it can be non-square if we consider time series of different lengths, $N_\alpha \neq N_\beta$. This matrix can represent the adjacency matrix of a bipartite graph, corresponding to the cross-recurrence network, where the nodes belong to two distinct groups, namely, the state vectors $\{\vec{z}_i^{[\alpha]}\}$ and $\{\vec{z}_j^{[\beta]}\}$ and the edges are established only between nodes of different groups.

Inter-system recurrence networks (given by adjacency matrix $\boldsymbol{IR}$) arise from the combination of the recurrence matrices $\boldsymbol{R}^{[\alpha]}$ (for univariate time series $Y_{\alpha,t}$), with the cross-recurrence matrix $\boldsymbol{CR}^{[\alpha\beta]}$ (Feldhoff et al., 2012):

$$\boldsymbol{IR} = \begin{pmatrix} \boldsymbol{R}^{[1]} & \boldsymbol{CR}^{[12]} & \ldots & \boldsymbol{CR}^{[1m]} \\ \boldsymbol{CR}^{[21]} & \boldsymbol{R}^{[2]} & \ldots & \boldsymbol{CR}^{[2m]} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{CR}^{[m1]} & \boldsymbol{CR}^{[m2]} & \ldots & \boldsymbol{R}^{[m]} \end{pmatrix}. \tag{3.28}$$

The adjacency matrix of an inter-system recurrence network is defined as

$$\boldsymbol{A} = \boldsymbol{IR} - \boldsymbol{I}_N, \tag{3.29}$$

where $\boldsymbol{I}$ is an identity matrix of size $N = \sum_{\alpha=1}^{m} N_\alpha$. This results in an undirected and unweighted simple graph, where the nodes and edges obey a natural partition: subgraphs G^α correspond to $\boldsymbol{R}^{[\alpha]}$ (intra-system connectivity) and subgraphs $G^{\alpha\beta}$ to $\boldsymbol{CR}^{[\alpha\beta]}$ that includes only edges between nodes from different systems (inter-system connectivity). Note that the definition of closeness can vary between different pairs of systems (Zou et al., 2019) if we consider different thresholds $\varepsilon_{\alpha\beta}$ for all $\alpha, \beta = 1, \ldots, m$.

Inter-system recurrence networks can be analyzed as a network of networks where the nodes represent subgraphs G^α. As such and in line with previous work for the univariate case, Donges and co-authors (Donges et al., 2011a) used specific graph features as discrete approximations of more general geometric properties to study the interconnections between systems. These features proved to be useful to coupling detection from two paleoclimate records (Feldhoff et al., 2012), distinguish between the dynamics of focal and non-focal EEG signals (Subramaniyam and Hyttinen, 2015) and characterize different oil–water flow patterns from multi-channel measurements (Gao et al., 2013, 2016b).

3.2.1.7 Joint Recurrence Networks

A joint recurrence network is another extension of the recurrence networks. The difference for the networks introduced in the previous section is that joint recurrence networks study the recurrence of different time series in their individual phase spaces, contrary to the same phase space. The joint recurrence matrix $\boldsymbol{JR}$ is defined as follows (Romano et al., 2004):

$$JR_{i,j}^{[\alpha]} = \begin{cases} 1 & \text{if } \|\vec{z}_i^{[\alpha]} - \vec{z}_j^{[\alpha]}\| \leq \varepsilon_\alpha,\ \alpha = 1, \ldots, m \\ 0 & \text{otherwise} \end{cases}, \tag{3.30}$$

where ε_α is the selected distance threshold for the individual time series $Y_{\alpha,t}$. This matrix can be seen as the adjacency matrix $\boldsymbol{A}$ of a joint recurrence network:

$$\boldsymbol{A} = \boldsymbol{JR} - \boldsymbol{I}_N, \tag{3.31}$$

where $\boldsymbol{I}_N$ is an identity matrix of size $N = \sum_{\alpha=1}^{m} N_\alpha$. $\boldsymbol{A}$ represents the joint probability of m simultaneous recurrences ($\|\vec{z}_i^{[\alpha]} - \vec{z}_j^{[\alpha]}\|$, $\alpha = 1, \ldots, m$) in the phase spaces (Marwan et al., 2007). Joint recurrence networks are undirected and unweighted simple graphs. They can be constructed for time series with different phase spaces and require simultaneous observations, i.e., time series of the same length. So, unlike the recurrence networks and inter-system recurrence networks, the time information is taken into account. This type of network can be analyzed from the same point of view of the recurrence networks for univariate time series, however, need to be reinterpreted in terms of the underlying joint recurrence structure (Feldhoff et al., 2013). Note that joint recurrence networks are similar in construction to multiplex recurrence networks, as we will see in Section 3.2.2.3, except that the former does not establish inter-layer edges (Zou et al., 2019).

This method has been successfully used to detect generalized synchronization in time series (Feldhoff et al., 2013). It is expected that joint recurrence will be increasingly unlikely with an increase in the number m of processes. Therefore, Donner and co-authors (Donner et al., 2015) proposed a version called f-joint recurrence networks that reduces the requirement of occurrence of simultaneous recurrences in all subsystems.

3.2.2 Multiple Layer Networks

3.2.2.1 Multiplex Visibility Networks

Based on the visibility methods of Section 3.1.1 and the definition of multilayer networks, Lacasa and co-authors (Lacasa et al., 2015) proposed an extension of the visibility mapping method for multivariate series analysis. The networks resulting from this approach have been called *multiplex visibility graphs* (MVG). A multiplex visibility graph M of m layers is constructed so that each layer α corresponds to the NVG (Section 3.1.1) associated with the series $Y_{\alpha,t}$, as illustrated in Figure 3.11.

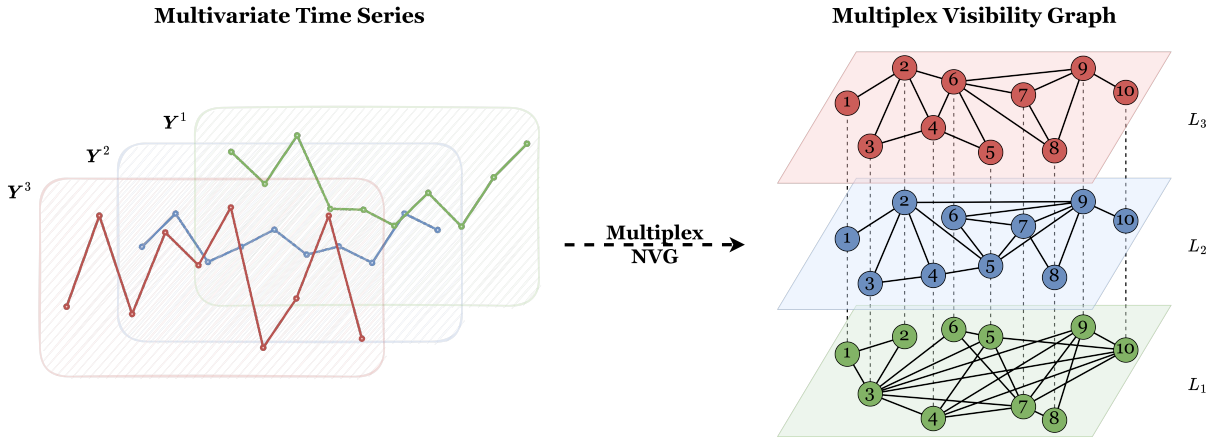


Figure 3.11: Illustration of multiplex natural visibility graph algorithm for a toy multivariate time series.

The resulting graph M is now represented by the adjacency matrix vector $[\boldsymbol{A}^1, \boldsymbol{A}^2, \ldots, \boldsymbol{A}^m]$, whose elements are the adjacency matrices of each layers and $A^{\alpha}_{i,j} = 1$ if and only if the nodes v^{α}_i and v^{α}_j are connected by an edge in layer L_{α}.

A *graph of layers* (or *network of networks*) can be built by projecting the original multiplex visibility network into a (single-layer) weighted graph of m nodes, where each node represents one layer. The edge weights denote a relation measure between layers, e.g., the magnitude of mutual information (Lacasa et al., 2015) calculated by *interlayer mutual information* [17]. We should note that the construction of multiplex visibility networks can be based on any of the visibility algorithms for univariate time series mentioned in Section 3.1.1.

The multiplex structure was used for the study of cellular networks of diffusively coupled maps lattices (Lacasa et al., 2015), to model spatio-temporal complex dynamics. They show that multiplex visibility graphs allow quantifying the amount of information flow between different series through the interlayer mutual information metric, and also to distinguish between different signal behaviors, between chaotic, periodic, and multiband patterns. The analysis of this metric allowed locating data points of change between different patterns of behavior of a system.

Bianchi and co-authors (Bianchi et al., 2017) used multiplex visibility networks method based on the weighted HVG algorithm (Section 3.1.1.5) where the edge weights are given by:

$$w_{i,j} = 1/\sqrt{(j-i)^2 + (Y_i - Y_j)^2},$$

incorporating temporal and amplitude information of the data. They studied the topological features of the networks to characterize neuron activation. For example, in certain types of neuronal activities, the local clustering coefficient of the corresponding nodes at the moment

[17] Interlayer mutual information, $I_{\alpha,\beta}$, quantifies the common information by every two different layers α and β based on the similarity of the degree distributions. Higher values of $I_{\alpha,\beta}$ indicate that the corresponding layers are associated/correlated and, consequently, the series they represent (Lacasa et al., 2015).

of the activity has high values which are replicated by the same nodes in the remaining layers. This type of metric allows identifying neuronal activities and the propagation of this activity by different neurons. Another set of real data studied using these networks was the fMRI series (Sannino et al., 2017) that are of extreme importance for the diagnosis of mental and neurological diseases. The results show differences in brain activities connected to psychiatric disorders.

In Gao et al. (2019a) the authors studied multiplex limited penetrable horizontal visibility graphs to try to explain the difference in the accuracy of the results obtained in the task of classifying time series from individuals in normal and fatigue states. They found that some network metrics decrease in states of fatigue and that, consequently, there is less efficiency in global information transfer. They concluded, therefore, that this transfer loss leads to a reduction in the accuracy of the classification between individuals in normal states and in fatigue.

More recent works begin to look at multiplex network structures as a rich and promising framework for analyzing multivariate time series. An example is the work carried out in De Giuli et al. (2022) which studies MTS of asset prices to study the reaction of capital markets to the Brexit announcements. The authors used the MVG algorithm to map the MTS and use tensor decomposition over the multiplex structure to obtain global centrality features through scores resulting from the tensor decomposition. These features allowed the authors to analyze and identify volatility behaviors of the data, in both intra and inter-dimension.

3.2.2.2 Multiplex Directed Visibility Network

Based on MVG, Flori et al. (2021) propose the multiplex directed visibility graph (MDVG) version. In this configuration, each time series component in a MTS set is mapped to a directed natural visibility graph-layer of a multiplex network, where intra-layer edges (v_i^α, v_j^α) between two nodes (v_i^α andv_j^α) are directed from the node corresponding to the timestamp that has the lowest value to the node corresponding to the timestamp that has the highest value in the series if there is a natural visibility line between these timestamps in the series. Formally, this directed natural visibility concept is described by:

$$Y_{\alpha,i} < Y_{\alpha,j} \wedge Y_{\alpha,k} < Y_{\alpha,j} + (Y_{\alpha,i} - Y_{\alpha,j})\frac{(t_j - t_k)}{(t_j - t_i)}. \tag{3.32}$$

This definition allows mapping local maximums and minimums of time series on nodes with high in-degree and out-degree values, respectively.

Flori et al. (2021), uses tensor decomposition, which takes into account the direction of the intra-layer edges, to produce centrality features. The authors were able to discriminate positive and negative trends market from the in- and out-degree features and correlate these properties with scores extracted from tensor decomposition.

3.2.2.3 Multiplex Recurrence Networks

Recently, Eroglu and co-authors (Eroglu et al., 2018), motivated by the idea that recurrences are a fingerprint of the characteristic properties of dynamic systems, the results from recurrence networks, the current and ever greater need for analysis and exploration of large amounts of data, and the previous method, multiplex visibility networks, proposed by Lacasa and co-authors (Lacasa et al., 2015), proposed recurrence networks for the domain of multivariate time series, the so-called multiplex recurrence networks. The algorithm follows exactly the same process as the multiplex visibility graphs presented in Section 3.2.2.1, but using recurrence networks presented in 3.1.3.3 instead of the visibility graphs. Remembering, the quantities of interlayer mutual information allow us to deduce a single-layer weighted network from the multiplex recurrence network.

Eroglu and co-authors (Eroglu et al., 2018), used metrics as the interlayer mutual information and the average edge overlap [18] in the multiplex network and also weighted network metrics, clustering coefficient, and average path length, in the single-layer weighted network, in order to detect the transitions between different dynamical regimes in coupled map lattices and real-world paleoclimate time series. They show that all the metrics capture similarities in the topological structures of the m recurrence networks. In particular, high values of the first three metrics have been obtained for periodic behaviors, while lower values have been obtained for chaotic behaviors. Additionally, average path length showed the opposite behavior.

In Gao et al. (2019b), the authors propose to combine the multiplex recurrence networks and deep learning techniques to detect driver fatigue in the EEG signals of subjects under alert and fatigue states. They construct a multiplex recurrence network from the EEG signal, used convolutional neural networks to extract and learn the features of the multiplex networks, and they perform a classification task. Those authors showed that this combination can achieve high accuracy and better results than traditional methods.

3.3 Software Frameworks

In recent years, some software packages have been made available to the research community that allows implementing and putting into practice some of the methodologies presented in this chapter. Below we name and briefly describe these software packages.

[18] The average edge overlap, ω, calculates the expected number of layers in which a given edge is present.

3.3.1 pyunicorn

The `pyunicorn` (Donges et al., 2015) is an available Python software package that integrates methods of network science and of nonlinear time series analysis. In addition, it implements methods for time series analysis via graph theory, which includes the following multivariate and univariate time series mapping methods. Recurrence networks methods for both UTS and MTS that were presented in Sections 3.1.3.3, 3.2.1.6, 3.2.1.7 and 3.2.2.1. The available methods focus mainly on the literature of recurrence quantification analysis. And methods of natural visibility graph presented in Section 3.1.1.1 and horizontal visibility graph presented in Section 3.1.1.2. Where the `pyunicorn` package provides network topological features related to centrality and clustering/community properties. The source code for this package is available in the following public repository: https://github.com/pik-copan/pyunicorn.

3.3.2 nets

An R software package `nets` (Network Estimation for Time Series) (Barigozzi and Brownlees, 2019) has been developed based on Diebold and Yılmaz (2014) and Barigozzi and Hallin (2017) and on the concept of long-run partial correlation to convert a multivariate time series into a single-layer weighted graph structure. The `nets` algorithm (presented in Section 3.2.1.2) allows estimating sparse long-run partial correlation networks from multivariate time series data, based on the estimated VAR parameters and the concentration matrix of the VAR residuals using LASSO (Meinshausen et al., 2006) regressions on the data. The package is available in the following public repository: https://github.com/ctbrownlees/R-Package-nets.

3.3.3 PyIOmica

A Python software package named `PyIOmica` (Domanskyi et al., 2020) is a package with a focus on biological datasets, characterizing and categorizing temporal trends. Adding to several useful bioinformatics tools, the `PyIOmica` incorporates a module with implementations of visibility graphs methods. These methods can be used to visualize time series as networks, characterizing time series in terms of autocorrelations, and categorizing temporal behavior, among others. In this package, we can also find the DPVG algorithm (Zheng et al., 2021) and the community detection method based on the shortest paths, presented in Section 3.1.1.8. The source code of `PyIOmica` can be found in the following repository: https://github.com/gmiaslab/pyiomica.

3.3.4 ts2net

The latest `ts2net` (Ferreira, 2022) software package is an R software package dedicated to transforming UTS or MTS datasets into a monoplex network structure. This framework allows us to represent complete time series, segments of times series, or timestamp values as node structures of networks, where connections between pairs of nodes, ie. edges, are defined by associations or similarity measures calculated between the corresponding time series granularity. The main mapping methods incorporated in this package are recurrence networks, visibility networks, and transition networks. To map MTS data we can use several distance functions to measure the similarity between pairs of nodes (the time series components) and establish the edges based on these measures, or user distance matrix based on recurrence networks approaches. To map UTS data into a monoplex network the methods are based on proximity networks, specifically, recurrence networks presented in Section 3.1.3.3, on transition networks, namely the quantile networks presented in Section 3.1.2.1, and on the visibility graphs presented in Sections 3.1.1.1 and 3.1.1.2. The package also takes advantage of other available packages to compute complex network methods and data mining tools to analyze time series data. The `ts2net` is available in: https://github.com/lnferreira/ts2net.

3.4 Final Remarks

In this chapter, we present several algorithms proposed in the literature to map univariate and multivariate time series into the complex network domain with the aim of producing new insights and overcoming open issues in time series analysis, such as high-dimensionality, finding periodicities, classifying different dynamic processes, among others. For univariate time series, there is a large body of literature, and the mappings may be classified according to three main underlying concepts: visibility, transition, and proximity.

Visibility concepts map times series into graphs with nodes representing timestamps and edges defined by a geometric relationship between data values. The graph reflects both local and global properties of the time series, especially via local maximum. Visibility-based mapping methods do not require preprocessing of the time series data. They are also completely parameter-free, except for the limited and parametric versions. These networks have become very popular in the literature due to the fact that the geometric criteria associated to visibility mappings are intuitive and easy to understand.

Intuitively, transition networks represent the transition pattern of a time series based on different types of symbolic encoding of the data. With the exception to time series with clear trending behavior, these networks are not explicit about the underlying time order since time series with sharp trend behavior are mapped into quantile graphs that are chain graphs allowing us to perceive the time order of the data. Construct these networks involves the definition of symbols representing quantiles, order patterns, or different dynamic states, requiring the choice of parameters that must balance the loss of information induced by the partitioning.

The proximity networks represent the similarity of sliding windows of data over time, thus, reflecting how local properties of the time series evolve over time. The construction of proximity networks involves the definition of states of the time series as vectors, such as cycles or vectors in phase space, requiring the selection of parameters. The resulting networks have nodes that represent these states of the time series and edges that are established using measures of similarity or distance between the states, with or without thresholding. The construction of transition and of proximity networks thus requires choosing several parameters that may influence the resulting graph and consequently the analysis of the time series. These networks are quite popular in the study of dynamic systems.

Several approaches to mapping multivariate time series into complex networks have been developed. Most focus on the idea of constructing a single-layer network from multivariate time series, capable of capturing dependencies between the component processes, both contemporary and lagged, as well as the serial dependencies. However, with technological advances and the growing need to analyze complex and high-dimensional data, new techniques have emerged, namely, methods that give rise to high-level structures: multiple layers networks. Although preliminary, this type of multivariate time series mapping has an important characteristic: all the methods developed for the univariate context can be extended to the context of multilayer networks, allowing the reuse of knowledge already acquired in the univariate case.

A network science approach to time series analysis has been used in different application domains and allowed us to characterize system dynamics, distinguish different dynamics, identify regime shifts and dynamical transitions, and test for reversibility and forecast. This survey hints at the conclusion that this approach provides complementary information to traditional time series analysis. This approach can be leveraged to address fundamental open issues, such as: time series mining problems and visualization and analysis of high-dimensional data. Mining large collections of time series data, such as univariate and multivariate time series clustering and classification problems, is a non-trivial and currently hot problem in the research areas involved. More and more eminent approaches are based on time series features, e.g. trend, length of time series, the strength of the trend, autocorrelations (Kang et al., 2017). Features extracted from the time series networks may be added to the classical set of time series features to enhance the data characterization. In particular, Silva (2018) has shown that (single-layer) network features can be used to accurately cluster univariate time series from different models, a relevant and important contribution to time series mining. In this direction, in the remaining of this work, we propose mappings from MTS multilayer networks and extend the MTS features via multilayer topological features.

Chapter 4

Multivariate Time Series Mappings

> Networks are present everywhere. All we need is an eye for them.
>
> *Albert-László Barabási*

Multivariate time series may be mapped into single or multiple layer networks. In the former, the nodes represent the component time series and the edges represent the relationships between the nodes (component time series) computed using statistical methods or models. These methods imply that all important information on the dynamics of each time series component, such as serial correlation, is inevitably lost in the mapping process. Mapping methods that represent multivariate time series as multiplex networks were proposed with the objective of suppressing this loss (see Section 3.2.2), preserving both the dynamical (over time) and the cross-sectional information contained in the multivariate data (Eroglu et al., 2018; Lacasa et al., 2015; Sannino et al., 2017; Silva et al., 2021). These multiplex networks map each component UTS into a layer (using a UTS mapping in which each time stamp, or a representation thereof, is represented by a node) and connect different layers via the common nodes (timestamps) through the underlying concept of mapping. Inevitably, lagged cross-correlations, which sometimes are the most important information, are not preserved during the mapping process.

To overcome this limitation, in the course of this work, we propose two new mapping methods to represent a multivariate time series as a multiple layers complex network. Remembering the definition in Section 2.3.2, MNets are complex structures capable of establishing internal connections (within the same layer) and external connections (between different layers). Therefore, an MNet allows us to create a very complete and flexible data structure (Kivelä et al., 2014). From a high-level view, multilayer networks present a compatible structure with the complex structure of multivariate time series. This allows us to keep more information from the time series data after the mappings, information that is inevitably lost using the conventional mapping methods referred to above.

This chapter is divided into two parts, where we present the two mapping methods of MTS proposed in this work. We describe the concepts and properties inherent to each of the mappings and present the implementation details and respective algorithms.

4.1 *MHVG*: a New Multilayer Visibility Graph

Visibility methods have shown to be very promising in capturing time series characteristics reflecting local and global properties of the data, and not requiring preprocessing of the data.

Recent literature (see Section 3.2.2) shows that multiplex versions of visibility graphs applied to MTS can achieve better accuracy and more promising results than conventional mappings into monoplex networks (Eroglu et al., 2018; Lacasa et al., 2015; Sannino et al., 2017; Silva et al., 2021). However, the method results in a multiplex network where the inter-layer edges connect nodes (timestamps) only between their counterparts, that is, the edges that connect two layers are only between nodes of subsequent layers that represent the same entity (timestamp). In the context of networks, this means that two nodes from different layers are reachable only using a path that contains one or more inter-layer edges necessarily between the same entity. This prevents direct relations between different entities on different layers. In the context of MTS and VGs, we can think of this property as the lack of a direct geometric relationship (visibility lines) between timestamps lagged of different variables. And thinking about the native concept of the visibility method (Section 3.1.1.1) this seems restrictive.

In this section, we present the new visibility algorithm to map an MTS into a *multilayer horizontal visibility graph*. This algorithm is based on a new visibility concept, *cross-horizontal visibility* which is an extension of the traditional horizontal visibility. Next, we begin by defining this concept.

4.1.1 Cross-Horizontal Visibility

Consider two time series $\boldsymbol{Z}^{\alpha} = (Z_{\alpha,1}, \ldots, Z_{\alpha,T})$ and $\boldsymbol{Z}^{\beta} = (Z_{\beta,1}, \ldots, Z_{\beta,T})$ on the same scale. Two arbitrary data values (t_i, Z_{α,t_i}) and (t_j, Z_{β,t_j}) are said to have cross-horizontal visibility, Cross-HV if

$$Z_{\alpha,t_i}, Z_{\beta,t_j} > \max\left(Z_{\alpha,t}, Z_{\beta,t}\right), \quad \text{for all } t, \;\; t_i < t < t_j, \;\; i, j = 1, \ldots, T, i \neq j. \tag{4.1}$$

This definition implies that all data values have Cross-HV to its neighbours and that the visibility is reciprocal, meaning that if $(t, Z_{\alpha,t})$ has Cross-HV to $(s, Z_{\beta,s})$, then $(s, Z_{\beta,s})$, has Cross-HV to $(t, Z_{\alpha,t})$. The concept of cross-horizontal visibility, Cross-HV, is illustrated in Figure 4.1 with two toy time series and for the first four data points, with the bi-coloured lines indicating (the reciprocal) visibility between the corresponding time series.

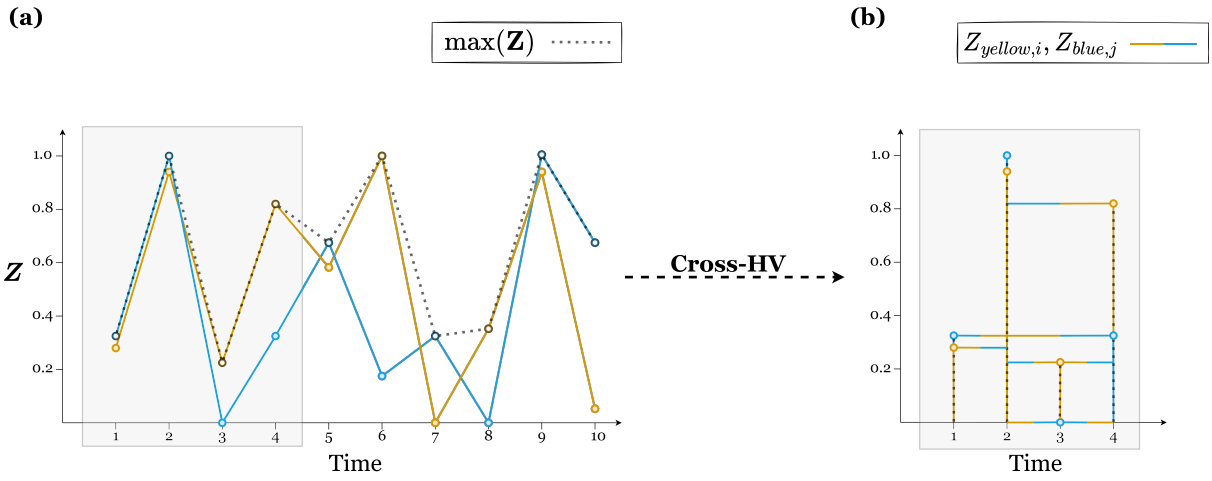


Figure 4.1: Schematic diagram of the cross-horizontal visibility concept. (a) Illustrates a toy bivariate time series $\boldsymbol{Z}^{yellow}, \boldsymbol{Z}^{blue}$, (same scale) and the corresponding *maximum* time series; (b) represents the cross-horizontal visibility, Cross-HVG, by solid bi-color lines (yellow and blue) connecting the data bars of the time series components $\boldsymbol{Z}^{yellow}$ and $\boldsymbol{Z}^{blue}$, for the first four timestamps.

4.1.2 Multilayer Horizontal Visibility Graph

A *Multilayer Horizontal Visibility Graph* (MHVG) is obtained by mapping a MTS, $\boldsymbol{Y} = \{Y^{\alpha}\}_{\alpha=1}^{m}$, into a MNet structure, $M = (V_M, E_M, V, \boldsymbol{L})$, using the concepts of HV and Cross-HV, as follows. Each unique time stamp, t, is mapped into an unique entity in V_M and each component time series, $\boldsymbol{Y}^{\alpha}$ is mapped into a layer, $L_{\alpha} \in \boldsymbol{L}$, $\alpha = 1, \ldots, m$, using the HVG method described in Section 3.1.1.2, thus establishing the intra-layer edges, $(v_i^{\alpha}, v_j^{\alpha}) \in E_M$, $i, j = 1, \ldots, T$, $i \neq j$. Then inter-layer edges $(v_i^{\alpha}, v_j^{\beta}) \in E_M$, between any two layers L_{α} and L_{β}, $\alpha, \beta = 1, \ldots, m$, $\alpha \neq \beta$ and $i, j = 1, \ldots, T$, $i \neq j$ are established using the Cross-HV described in the previous Section. Note that to establish Cross-HV all the time series $\boldsymbol{Y}^{\alpha}$, $\alpha = 1, \ldots, m$ must be in the same scale[1] which may require a preprocessing step of the dataset $\boldsymbol{Y}$, comprising the Min-Max scaling of each time series. The mapping is illustrated in Figure 4.2, with toy bivariate time series, for the sake of simplicity.

From the generated MHVG, we can identify the *intra-layer graphs*, $\{G^{\alpha}\}_{\alpha=1}^{m}$ and the *inter-layer graphs*, $G^{\alpha,\beta}$, for $\alpha, \beta = 1, \ldots, m$ and $\alpha \neq \beta$. $\{G^{\alpha}\}_{\alpha=1}^{m}$ correspond to the HVG of each individual time series component and it is represented by the adjacency matrix $\boldsymbol{A}^{\alpha}$ with $A_{i,j}^{\alpha} = 1$ if $(v_i^{\alpha}, v_j^{\alpha}) \in E_M$ and 0 otherwise. $G^{\alpha,\beta}$ correspond to the cross-horizontal visibility graph (Cross-HVG) of each pair of time series components and it is represented by the adjacency matrix $\boldsymbol{B}^{\alpha,\beta} = \begin{bmatrix} \mathbf{0} & \boldsymbol{A}^{\alpha,\beta} \\ \boldsymbol{A}^{\beta,\alpha} & \mathbf{0} \end{bmatrix}$ with $A_{i,j}^{\alpha,\beta} = 1$ and $A_{j,i}^{\beta,\alpha} = 1$ if $(v_i^{\alpha}, v_j^{\beta}) \in E_M$ and 0 otherwise.

[1]Normalization is necessary so that it is possible to make the condition of horizontal visibility (Eq. 4.1). Otherwise, time series with very different value ranges would lead to edges only for the immediate neighbors, for example.

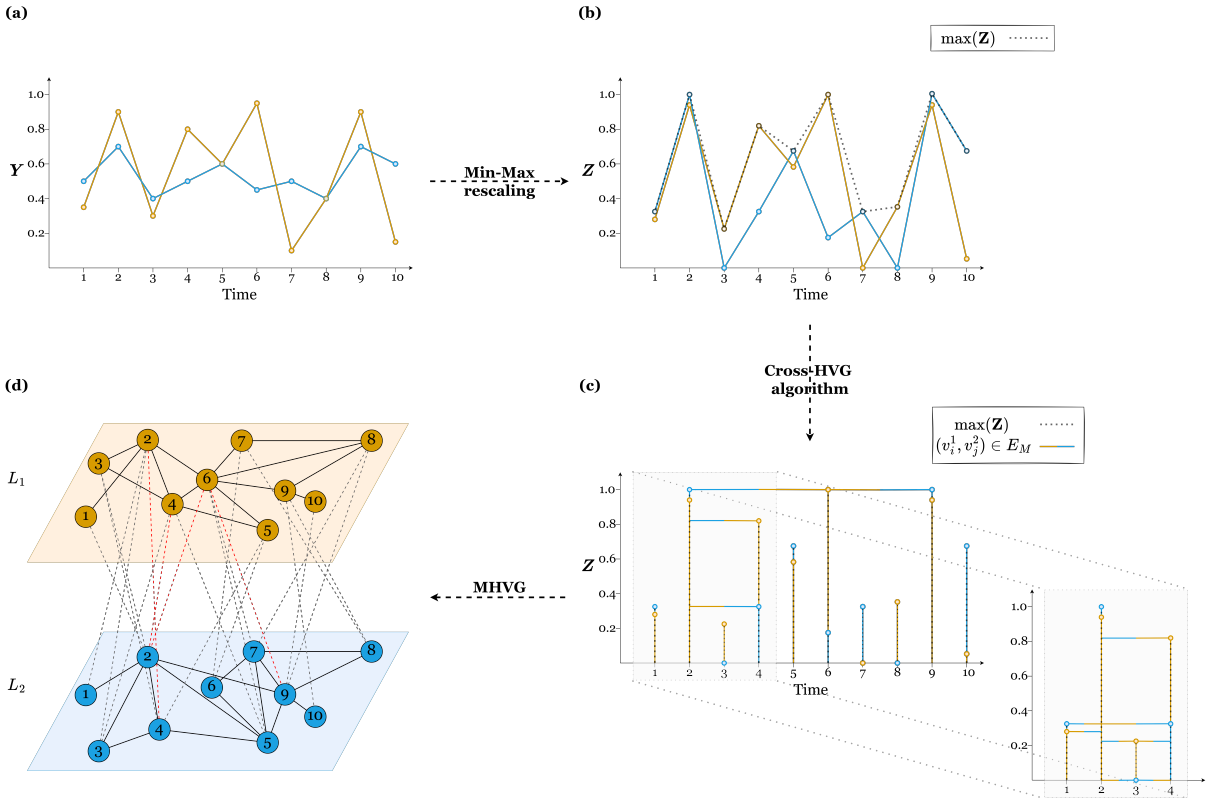


Figure 4.2: Schematic diagram of the multilayer horizontal visibility graph algorithm: (a) original time series, (b) Min-Max rescaled time series and *maximum* time series, (c) illustration of cross-HV with the edges between adjacent timestamps omitted for simplicity (detail for the first four timestamps), (d) cross-horizontal visibility graph: solid black lines represent the intra-layer edges (the HVGs), dashed lines the inter-layer edges (the Cross-HVGs) and the red lines highlight inter-layer edges between nodes corresponding to non-adjacent timestamps.

Algorithm 1 describes the concept of Cross-HV and Algorithm 2 describes mapping a multivariate time series into a Multilayer Horizontal Visibility Graph. In the Appendix A, we describe the auxiliary functions to support the implementation of the method, Algorithm 5 describes the function that creates an HVG and Algorithm 6 describes the function that creates the inter-layer edges.

The new method of multilayer visibility network is based on the HVG algorithm. However, the definitions presented above, the cross-HVG algorithm and MHVG algorithm, naturally extend to the NVG method and versions of both NVG and HVG. In the case of the NVG we can build the *Multilayer Natural Visibility Graph* (MNVG), built based on the Cross-Natural Visibility Graph (Cross-NVG) method, given by the following condition:

$$\max\left(Z_{\alpha,t}, Z_{\beta,t}\right) < Z_{\beta,t_j} + (Z_{\alpha,t_i} - Z_{\beta,t_j})\frac{(t_j - t)}{(t_j - t_i)}, \tag{4.2}$$

where $\boldsymbol{Z}^{\alpha} = (Z_{\alpha,1}, \ldots, Z_{\alpha,T})$ and $\boldsymbol{Z}^{\beta} = (Z_{\beta,1}, \ldots, Z_{\beta,T})$ are on the same scale, and two arbitrary data values (t_i, Z_{α,t_i}) and (t_j, Z_{β,t_j}) have cross-natural visibility, Cross-NV.

Algorithm 1: Cross-Horizontal Visibility Graph

Input: Two rescaled time series, $\boldsymbol{Z}^a$ and $\boldsymbol{Z}^b$, (***tsA***, ***tsB***), the corresponding layers, L_a and L_b, (***layerA***, ***layerB***), and the maximum time series, $\max\left(\boldsymbol{Z}^a, \boldsymbol{Z}^b\right)$, (***tsMax***)

Procedure `CHVG(`*tsA, tsB, tsMax, layerA, layerB*`)`

```
1   T ← tsMax.size()                                   ▷ The time series lengths
    for node i in layerA.get_Nodes() do
        for node j from i + 1 to T in layerB.get_Nodes() do
            if node i can 'see' j then
2               mnet.add_Edge(i, j, layerA, layerB)    ▷ Add inter-layer edge (v_i^a, v_j^b)
            end
        end
        for node j from i − 1 to 0 in layerB.get_Nodes() do
            if node i can 'see' j then
3               mnet.add_Edge(i, j, layerA, layerB)    ▷ Add inter-layer edge (v_i^a, v_j^b)
            end
        end
    end
4   return
```

Algorithm 2: Multilayer Horizontal Visibility Graph

Input: A set of time series components, $\{\boldsymbol{Y}^a\}_{a=1}^m$, (***mts***)

Output: A multilayer network, M, (***mnet***)

Procedure `MHVG(`*mts*`)`

```
1   m ← mts.size()                                     ▷ The number of time series components
2   mnet ← {}                                          ▷ The empty MNet M
3   n_mts ← {}                                         ▷ List to store the rescaled time
                                                         series components
    for a ← 1 to m do
4       mnet.layers[a] ← {}                            ▷ The empty layer L_a
5       HVG(mts[a], mnet.layers[a])                    ▷ Map the time series Y^a on the HVG
                                                         L_a (Eq. 3.2)
6       n_mts[a] ← MinMax(mts[a])                      ▷ Rescale time series Z^a
    end
    for a ← 1 to m − 1 do
        for b ← a + 1 to m do
7           tsMax ← MaxTS(n_mts[a], n_mts[b])          ▷ Get the maximum rescaled time series
8           CrossHVG(n_mts[a], n_mts[b], tsMax, mnet.layers[a], mnet.layers[b])  ▷ Map the pairwise
                                                         time series Z^a and Z^b
                                                         on Cross-HVG (Eq. 4.1)
        end
    end
9   return mnet
```

4.2 *MQG*: a New Multilayer Transition Graph

In the univariate context, time series transition properties can be captured by QG (presented in Section 3.1.2.1) with the additional advantage of allowing dimensionality reduction. As far as we are aware, there are no mappings of MTS to multiple layer networks that capture the data transition properties both serially and cross-dimension wise while reducing the dimensionality of the data. Given the advantages and thinking about the problem of the dimensionality curse that is more and more prominent, in this section, we propose a multivariate version of the QG method for multivariate settings of temporal data, the *Multilayer Quantile Graph*.

In a basic sense, a *Multilayer Quantile Graph* (MQG) extends the concept underlying QG presented in Section 3.1.2.1. Given a pair of time series, $\boldsymbol{Y}^{\alpha}$ and $\boldsymbol{Y}^{\beta}$, each one is replaced by the corresponding QG and the contemporaneous quantiles (q_i^{α} and q_j^{β} at time t) are associated by inter-layer edges that represent the cross-dimensions contemporary transitions. In more detail, the MQG involves the following steps (Figure 4.3):

Step 1: each time series component, $\boldsymbol{Y}^{\alpha}, \alpha = 1, \ldots, m$, is mapped into the corresponding QG L_{α} by applying the QG mapping described in Section 3.1.2.1. This step is illustrated at the top of Figure 4.3.

Step 2: for each pair of QGs, L_{α} and L_{β} with $\alpha, \beta = 1, \ldots, m$ and $\alpha \neq \beta$, the corresponding contemporary quantiles (at time $t = 1, \ldots, T$), q_i^{α} and q_j^{β} with $i, j = 1, \ldots, \eta$, are linked. The quantiles q_i^{α} and q_j^{β} belong, respectively, to quantile sequences $\boldsymbol{Q}_{\alpha}$ and $\boldsymbol{Q}_{\beta}$ over time $t = 1, \ldots, T$ (see panel **(b)** of Figure 4.3).

Step 3: the MTS, $\boldsymbol{Y}$, is then mapped into a MQG, M, as illustrated in panel **(c)** of Figure 4.3. The set of layers refers to each QG, i.e., $L_{\alpha} \in \boldsymbol{L}$, $\alpha = 1, \ldots, m$, and the set of node-layer refers to the sample quantiles, $V_M = \{q_i^{\alpha}\}_{i=1}^{\eta}$. From the individual QGs the directed weighted intra-layer edges are established in layer L_{α}, i.e., $(q_i^{\alpha}, q_j^{\alpha}, w_{i,j}^{\alpha}) \in E_M$, and from pair-wise QGs the bidirectional weighted inter-layer edges are established between dimensions $(q_i^{\alpha}, q_j^{\beta}, w_{i,j}^{\alpha,\beta}) \in E_M, \alpha \neq \beta$. The weight of inter-layer edge, $w_{i,j}^{\alpha,\beta}$, represents the probability that $Y_{\alpha,t}$ and $Y_{\beta,t}$ belong to the quantiles q_i^{α} and q_j^{β}, respectively, at the same time t.

The multilayer network corresponding to the MQG is a directed and weighted network. Note that the inter-layer edges are bi-directed, that is, whenever there is a transition from q_i^{α} to q_j^{β} there is also a transition from q_j^{β} to q_i^{α}. Therefore, inter-layer edges can also be represented as undirected edges. Similar to the MHVG, we can also identify from the MQG the intra-layer graphs, corresponding to the individual QGs, and the inter-layer graphs corresponding to the bipartite graphs that represent the contemporary transitions. We will refer to these bipartite graphs as *Contemporaneous Quantile Graph*.

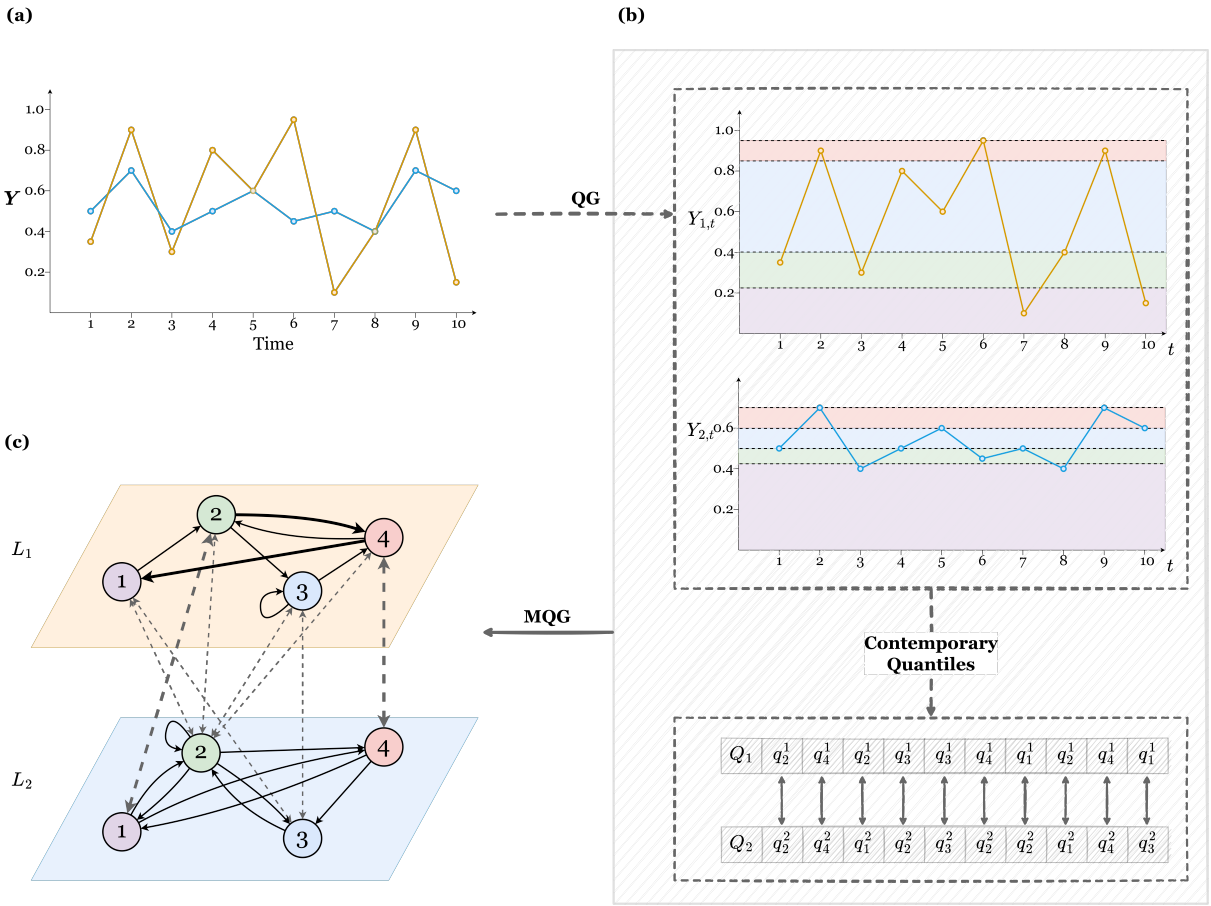


Figure 4.3: Schematic diagram of the multilayer quantile graph algorithm for $\eta = 4$: (a) original time series, (b) illustration of the intra-layer quantile graphs (coloured regions representing the different sample quantiles) and inter-layer contemporaneous edges mapping, (c) multilayer quantile graph: black lines represent the intra-layer edges (the QGs), dashed lines the inter-layer edges between nodes contemporaneous quantile nodes, and the thickness of the lines represent the weighted intensities of the edges.

The Algorithm 7 describes mapping a multivariate time series into a Multilayer Quantile Graph and Algorithm 4 describes the method that adds the set of inter-layer edges between contemporary nodes to the multilayer networks. In Appendix A, we present the Algorithm 7 that describes the QG method implementation.

In the same way that the univariate quantile graphs method can be extended to represent transitions between quantiles corresponding to lagged timestamps (rather than just between consecutive quantiles), the inter-layer edges of the contemporaneous quantile graphs can be extended to represent transitions between quantiles corresponding to lagged timestamps from two different layers (see Section 3.1.2.1 for more details).

Algorithm 3: Multilayer quantile graph

Input: A set of time series components, $\{Y_{a,t}\}$, (***mts***), and a value with the number of quantiles, (***numQ***)
Output: A multilayer network, M, (***mnet***)
Procedure `MQG(`*mts*, *numQ*`)`

```
1   m ← mts.size()                                  ▷ The number of time series components
2   mnet ← {}                                       ▷ The empty MNet M
    for a ← 1 to m do
3       mnet.layers[a] ← {}                         ▷ The empty layer L_a
4       set_direction(mnet.layers[a], true)         ▷ Set the directionality of the intra-
                                                      layer edges as directed
5       QG(mts[a], mnet.layers[a], numQ)            ▷ Map the time series Y_{a,t} on the QG L_a
                                                      (Section 3.1.2.1)
    end
    for a ← 1 to m − 1 do
        for b ← a + 1 to m do
6           set_direction(mnet.layers[a], mnet.layers[b], false)   ▷ Set the directionality of
                                                                     the inter-layer edges as
                                                                     undirected
7           Contemp_QG(mnet.layers[a], mnet.layers[b])             ▷ Map the pairwise time
                                                                     series Y_{a,t} and Y_{b,t} on
                                                                     Contemporaneous QG
                                                                     (Section 4.2)
        end
    end
8   return mnet
```

Algorithm 4: Contemporaneous quantile graph

Input: Two layers, L_a and L_b, (***layerA***, ***layerB***)
Procedure `Contemp_QG(`*layerA*, *layerB*`)`

```
1   T ← layerA.size()                               ▷ The time series length
2   qA ← layerA.q_seq                               ▷ The quantiles sequence of layer L_a
3   qB ← layerB.q_seq                               ▷ The quantiles sequence of layer L_b
    for i ← 1 to T do
4       edge ← mnet.get_Edge(qA[i], qB[i], layerA, layerB)   ▷ Get inter-layer edge between
                                                               contemporaneous time if
                                                               quantiles it already exists
        if !edge then
5           mnet.add_Edge(qA[i], qB[i], layerA, layerB, 1)   ▷ Add inter-layer edge with
                                                               weight 1
        end
        else
6           w ← mnet.get_Weight(edge)               ▷ Get the weight of the edge
7           mnet.set_Weight(edge, w+1)              ▷ Increase the weight of the edge
        end
    end
8   return
```

4.3 Final Remarks

In this chapter we introduce two new mapping methods to convert a MTS into a MNet.

The first mapping is based on a new horizontal visibility concept, the *cross-horizontal visibility*, developed to capture the cross dependencies between pairs of component time series. Thus, the multiplex visibility graphs of Lacasa et al. (2015) (presented in Section 3.2.2.1) are extended with the incorporation of *inter-layer edges* established according to the cross-horizontal visibility between different nodes (timestamps). These new edges/connections can capture dependencies between different timestamps of different variables. The resulting networks are denoted as *multilayer horizontal visibility graphs*.

The second mapping is based on the concept of transition and the definition of QGs presented in Section 3.1.2.1. The objective is to capture the transition probabilities of the dynamic (data variations) between the timestamps of different time series components. In this way, we are able to create a reduced structure to represent high-dimensions of multivariate data (since QGs allow reduce the dimensionality) that characterize the serial dynamic transitions, through the QG mapping over each time series component, and the dynamic transitions cross-dimension, through weighted inter-layer edges between contemporary quantiles. This new mapping method is designated by *multilayer quantile graphs*.

The taxonomy of algorithms for mapping time series into complex networks, proposed in Chapter 3 and represented in Figure 3.1 must now be updated with these two new mappings. Thus Figure 4.4 represents the updated taxonomy.

These two MNets are used in the remainder of this work to extract (univariate and multivariate) time series features via topological features from the whole MNet structure and its substructures.

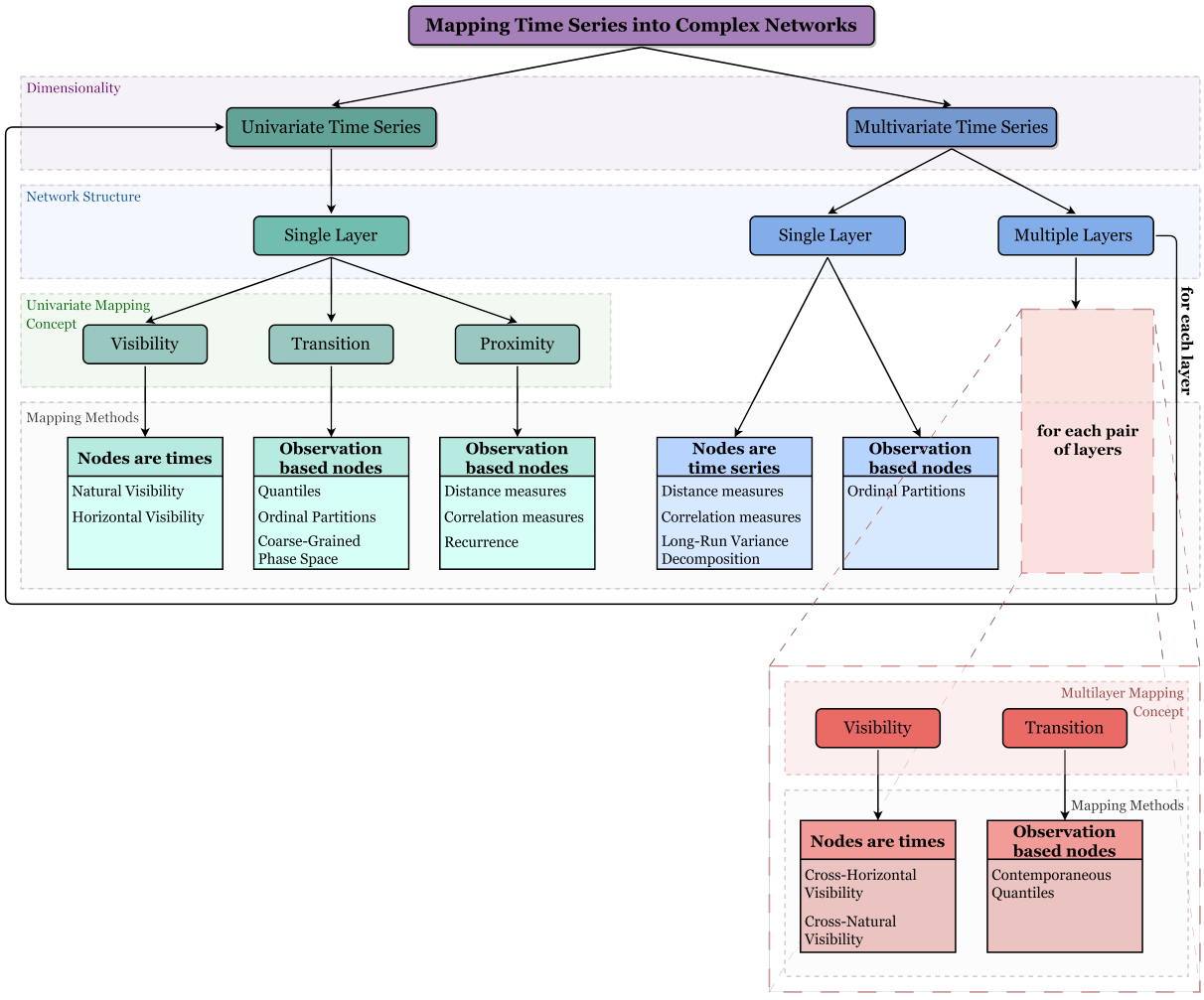


Figure 4.4: Overview of mapping methods. Taxonomy of algorithms for mapping time series into complex networks based on the dimensionality of time series, resulting network structure, mapping concept, and main mapping methods.

Chapter 5

Features for Univariate Time Series

The main idea behind time series feature-based approaches is to construct vectors of features that aim to represent specific properties of the time series data by characterizing the underlying dynamic processes (Fulcher, 2018; Fulcher and Jones, 2017). The usual features for UTS (see Sections 2.1 and 2.2) include concepts and methods from the linear time series analysis literature (Shumway and Stoffer, 2017), such as autocorrelation, stationarity, seasonality, and entropy, but also methods of nonlinear time-series analysis based on dynamic systems theory (Fulcher et al., 2013; Henderson and Fulcher, 2021; Kang et al., 2020; Wang et al., 2006). These methods usually involve parametric assumptions, parameter estimation, non-trivial calculations, and approximations, as well as preprocessing tasks such as finding time series components, differencing, and whitening thus presenting drawbacks and computation issues related to the nature of the data, such as the length of the time series.

One of the main goals of this work is to contribute to the feature-based approach in time series analysis. For that, we start proposing an alternative set of UTS features based on complex network concepts.

In the field of network science, we can find a vast set of topological graph features (see Section 2.3.3). Recent literature has shown an exponential growth of scientific works where the topological features of networks have been applied in various ways for the analysis of particular time series and to respond to specific problems, such as description, classification, and clustering of time series. Examples include the automatic classification of sleep stages (Zhu et al., 2014a), characterizing the dynamics of the human heartbeat (Shao, 2010), distinguishing healthy from non-healthy electroencephalographic series (Campanharo and Ramos, 2017), and analyzing seismic signals (Telesca and Lovallo, 2012).

In this section, we establish a new set of time series features, *NetF*, by mapping the time series into the complex networks domain. We establish a procedure for time series mining via *NetF* to address the question of whether time series features based on complex networks are useful to capture the properties of time series. Our procedure, represented in Figure 5.1 comprises the following steps: map the time series into (natural and horizontal) visibility graphs (see Sections 3.1.1.1 and 3.1.1.2) and quantile graphs (see Section 3.1.2.1) using the appropriate mapping methods and compute five specific topological features for each network, thus establishing a vector of 15 features. These features can then be used to mining time series data. In this chapter, we present the results of applying this procedure to explore the inherent characteristics of a large and diversified set of UTS datasets.

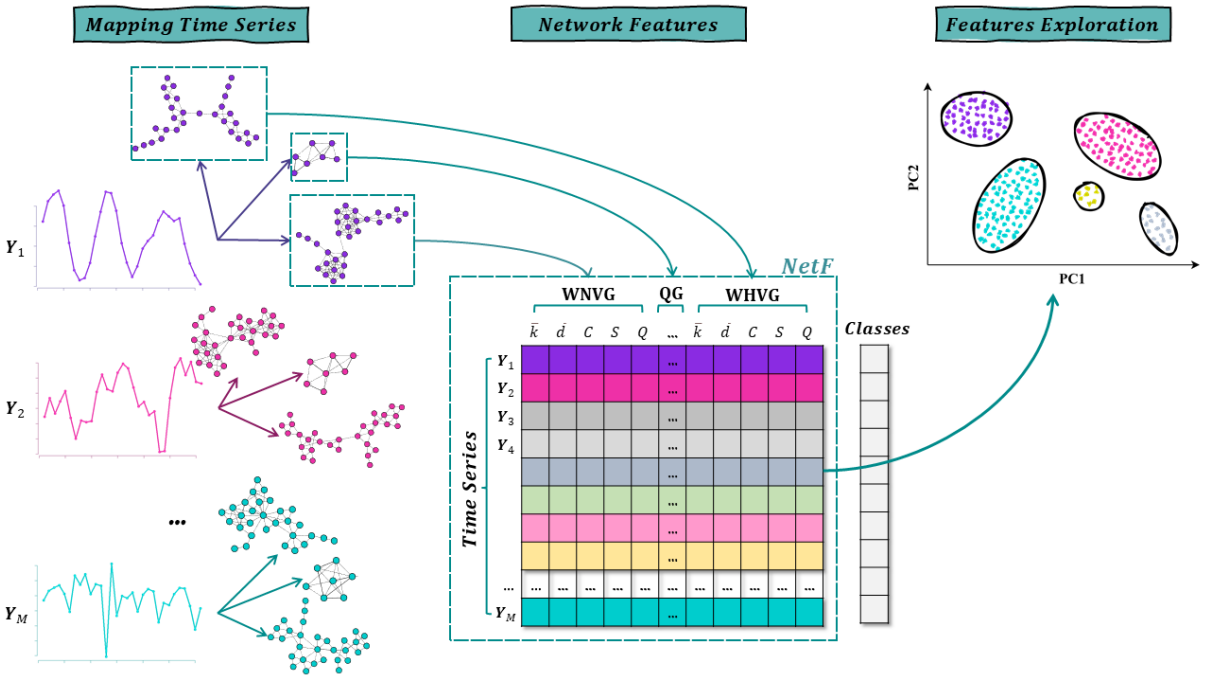


Figure 5.1: Schematic diagram of the network-based features approach to univariate time series mining.

5.1 *NetF*: a Novel Set of Univariate Time Series Features

In this work, we introduce ***NetF*** as an alternative set of features. Our approach differs from those previously mentioned in that we leverage the usage of different complex network mappings to offer a set of time series features based on the topology of those networks. One of the main advantages of this approach comes from the fact that the mapping methods (Section 3.1) do not require typical time series preprocessing tasks, such as decomposing, differencing, or whitening. Moreover, our methodology is applicable to any time series, regardless of its characteristics.

NetF is constituted by 15 different features, as depicted in Figure 5.2. These features correspond to the concatenation of five different topological features, as explained in Section 2.3.3 ($\bar{k}$, the average weighted degree; $\bar{d}$, the average path length; C, the clustering coefficient; S, the number of communities; Q, the modularity), each of them applied to three different mappings of the time series, as explained in Section 3.1.1 (*WNVG*, the weighted natural visibility graph; *WHVG*, the weighted horizontal visibility graph; *QG*, the quantile graph).

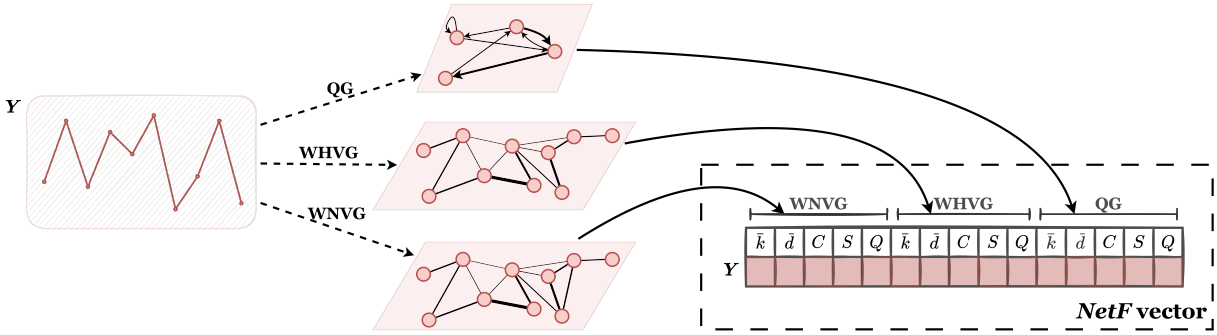


Figure 5.2: Schematic diagram of the *NetF*. A time series $\mathbf{Y}$ is mapped into three complex networks (WNVG, WHVG, and QG) and for each of these networks, five topological features are taken ($\bar{k}$, $\bar{d}$, C, S and Q), resulting in the *NetF* vector containing 15 features.

Our main goal is to provide a varied set of representative features that expose different properties captured by the topology of the mapped networks, providing a rich characterization of the underlying time series.

The topological features themselves were selected so that they represent global features of centrality, distance, community detection, and connectivity, while still being accessible, easy to compute, and widely used in the network analysis domain.

5.2 Implementation Details

To compute the WNVGs we implement the *divide & conquer* algorithm proposed in Lan et al. (2015) and for the WHVGs the algorithm proposed in Luque et al. (2009)[1]. To both we added the weighted version mentioned in Section 3.1.1.5, adding the respective weights to the edges. In this work, we used the inverse of Euclidean distance measure to define the weights, that is: $w_{i,j} = 1/\sqrt{(t_j - t_i)^2 + (Y_j - Y_i)^2}$. For the QGs we chose $\eta = 50$ quantiles, as in Campanharo and Ramos (2016), and we implemented the method described in Section 3.1.2.1 to create the nodes and edges of the networks. We used the sample quantile method, which uses a scheme of linear interpolation of the empirical distribution function (Hyndman and Fan, 1996), to calculate the sample quantiles (nodes) in support of the time series. To save the network structure as a graph structure, we used the `igraph` (Csardi and Nepusz, 2006) package which also allows us to calculate the topological features. The methods and algorithms used by the functions we used to calculate the features were as follows (see Section 2.3.3 for more details):

[1] https://sites.google.com/view/lucaslacasa/research-topics/visibility-graphs#h.27cb86b8e1ba43fd_148

Average Path Length ($\bar{d}$) In this work, we follow an algorithm that does not consider edge weights and uses the breadth-first search algorithm to calculate the shortest paths $d_{i,j}$ between all pairs of vertices, both ways for directed graphs.

Clustering Coefficient (C) The function that we use in this work ignores the edge direction for directed graphs. For this reason, before we calculate C for QGs, which are directed graphs, we first transform them into an undirected graph, where for each pair of nodes that are connected with at least one directed edge the edge is converted to an undirected edge. And then, the C is calculated by the ratio of the total number of closed triangles[2] in the graph to the number of triplets[3].

Number of Communities (S) The function we use in this work calculates densely connected subgraphs via random walks, such that short random walks tend to stay in the same community. See the Walktrap community finding algorithm (Pons and Latapy, 2005) for more details.

Modularity (Q) In relation to some division of nodes into communities, we measure how separated the nodes belonging to the different communities are as follows:

$$Q = \frac{1}{2|E|} \sum_{i,j} \left[w_{i,j} - \frac{k_i k_j}{2|E|} \right] \delta \left(\mathcal{C}_i, \mathcal{C}_j \right),$$

where $|E|$ is the number of edges, $\mathcal{C}_i$ and $\mathcal{C}_j$ the communities of v_i and v_j, respectively, and $\delta(\mathcal{C}_i, \mathcal{C}_j) = 1$ if v_i and v_j belong to the same community ($\mathcal{C}_i = \mathcal{C}_j$) and $\delta(\mathcal{C}_i, \mathcal{C}_j) = 0$ otherwise.

We performed all implementations and computations in R (R Core Team, 2020), version 4.0.3, and a set of packages. However, these results can be replicated using the framework we will present in this work.

We make the source code and dataset used in this chapter available in https://github.com/vanessa-silva/NetF.

5.3 Empirical Evaluation

In this section, we investigate, via synthetic datasets, whether the set of features introduced above is useful for characterizing time series data.

[2] A triangle is a set of three nodes with edges between each pair of nodes.
[3] A triplet is a set of three nodes with at least edges between two pairs of nodes.

To this end, we consider a set of eleven linear and nonlinear time series models, denoted by Univariate Data Generating Processes (UDGP), which present a wide range of characteristics summarized in Table 5.1. A detailed description of the UDGP and computational details are given in Appendix B.1. For each of the UDGPs (Figure 5.3 illustrate one instance of each) in Table 5.1 we generated 100 realizations of length $T = 10000$. Following the steps presented in Figure 5.1, we map each realization into three networks and extract the corresponding topological features. The resulting time series features, organized by mapping, are summarized, mean and standard deviation, in Tables B.1 to B.3. Note that the values have been Min-Max normalized for comparison purposes since the range of the different features varies across models.

Table 5.1: Summary about the univariate data generating process (time series models) of the synthetic data. Parameters, the main characteristic of the datasets, and notation is also included. See Appendix B.1 for more details.

Process	Parameters	Main Property	Notation
White Noise	$\epsilon_t \sim N(0,1)$	Noise effect	`WN`
AR(1)	$\phi_1 \in \{-0.5, 0.5\}$	Smoothness	`AR(1)-0.5` `AR(1)0.5`
AR(2)	$\phi_1 = 1.5, \phi_2 = -0.75$	Pseudo-periodic	`AR(2)`
ARIMA(1,1,0)	$\phi_1 = 0.7$	Stochastic trend	`ARIMA`
ARFIMA(1,0.4,0)	$\phi_1 = 0.5$	Long memory effect	`ARFIMA`
SETAR(1)	$\alpha = 0.5, \beta = -1.8, \gamma = 2,$ $r = -1$	Regime-dependent autocorrelation[4]	`SETAR`
Poisson-HMM	$N = 2, \left[\begin{smallmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{smallmatrix}\right]$ $\lambda \in \{10, 15\}$	State transitions	`HMM`
GARCH(1,1)	$\omega = 10^{-6}, \alpha_1 = 0.1,$ $\beta_1 = 0.8$	Persistent periods of high or low volatility	`GARCH`
EGARCH(1,1)	$\omega = \left(10^{-6} - 0.1\sqrt{2/\pi}\right),$ $\alpha_1 = 0.1, \beta_1 = 0.01, \gamma_1 = 0.3$	Asymmetric effects of positive and negative shock	`EGARCH`
INAR(1)	$\alpha = 0.5, \epsilon_t \sim Po(1)$	Correlated counts	`INAR`

WNVGs (Table B.1 and Figure 5.4a) present the lowest values for the clustering coefficient (C) for ARIMA models. Models producing time series with more than one state (`HMM` and `SETAR`) present a lower average weighted degree but a higher number of communities (S). The later values are comparable to those for `AR(2)` time series, the fact that can be explained by the pseudo-periodic nature of the particular `AR(2)` model entertained here. WHVGs (Table B.2 and Figure 5.4b) present average weighted degree $(\bar{k})$ approximately 0 for `HMM`'s and approximately 1 for `GARCH` and `EGARCH`. This indicates that `HMM` time series have, on average, horizontal visibility for more distant points (in time and/or value), while the opposite is true for heteroskedastic time series. The clustering coefficient (C) is the lowest (approximately 0) for networks obtained from `INAR` time series, indicating that most points have visibility only for the two closest neighbors.

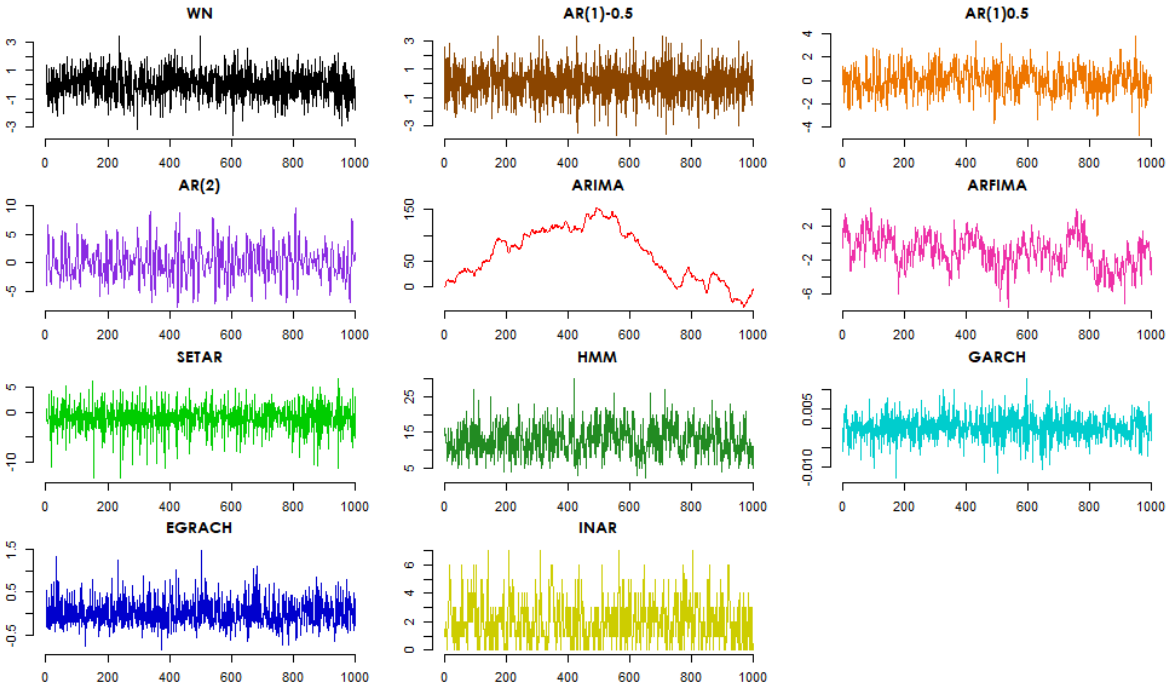
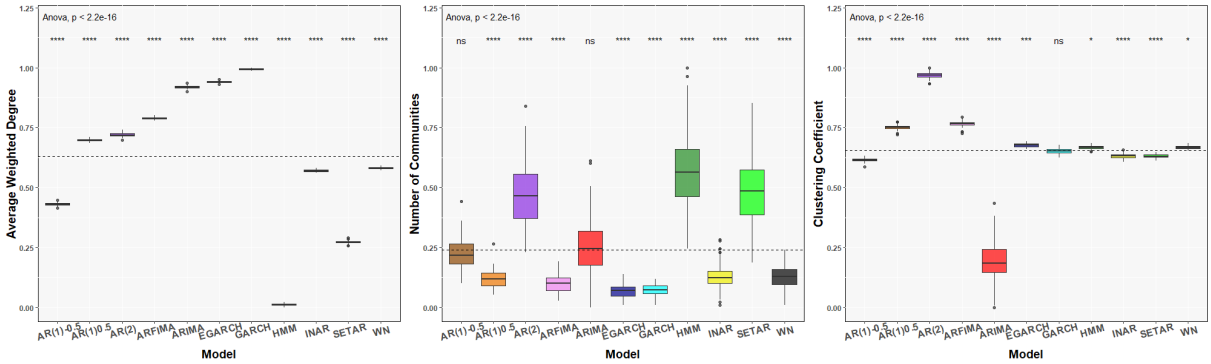


Figure 5.3: Plot of one instance of each simulated time series model.

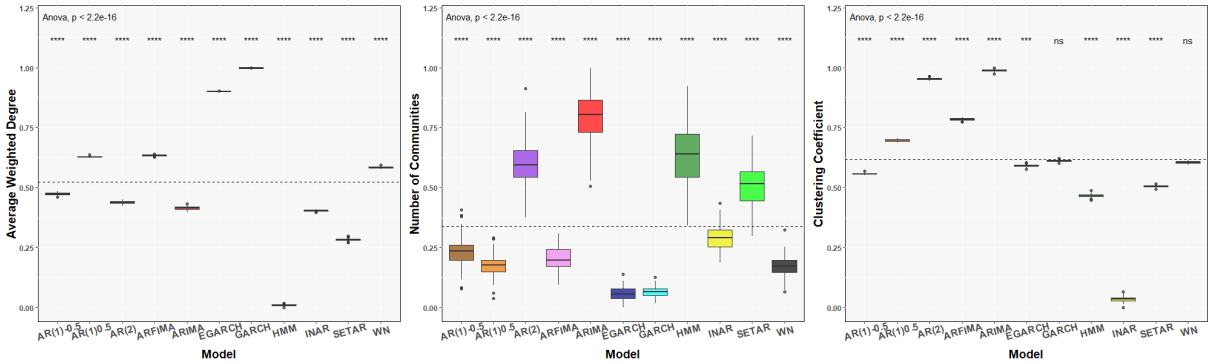
QGs (Table B.3 and Figure 5.5) present high values of average path length, $(\bar{d})$, for `ARIMA`, contrasting with all other UDGP which present low values. On the other hand, the (C) for `ARIMA` presents low values while all other UDGPs present high values.

The next step is to study the feature space to understand which network features capture specific properties of the time series models. Figure 5.6 represents a bi-plot obtained using the 15 features (5 for each mapping method) and with the two principal components (PC) explaining 68.8% of the variance. It is noteworthy that the eleven groups of time series models are clearly identified and arranged in the bi-plot according to their main characteristics. Overall, we can say that the number of communities of VGs, S, are positively correlated among themselves and are negatively correlated with the average weighted degree, $\bar{k}$, of *NetF*. The average path length, $\bar{d}$, of WHVGs and QGs and the clustering coefficient, C, of WHVG are positively correlated, but negatively to the $\bar{d}$, C and Q of WNVG, Q of WHVG and C of QGs. The features that most contribute to the total dimensions formed by the principal component analysis (PCA) are: $\bar{k}$, S, Q and $\bar{d}$ of the QGs, $\bar{k}$ of the WNVGs, and $\bar{k}$, S and Q of the WHVGs (see Figure 5.7).

The (stochastic) trend of the `ARIMA`, in fact, the only non-stationary UDGP in this dataset, is represented by high average path lengths, $\bar{d}$, in WHVG and QG. Discrete states in the data, `HMM`, `SETAR`, `INAR`, are associated with the number of communities, S. The bi-plot further indicates that height average weighted degree, $\bar{k}$, mainly that of the WHVG, represents hetero-skedasticity in the time series, e.g., `GARCH` and `EGARCH`. Cycles, `AR(2)`, are captured by the clustering coefficient, C.



(a) WNVG topological features



(b) WHVG topological features

Figure 5.4: Boxplot (by time series model) of (a) WNVG topological features, and (b) WHVG topological features. The plots also show the results of applying a statistical significance test (ANOVA).

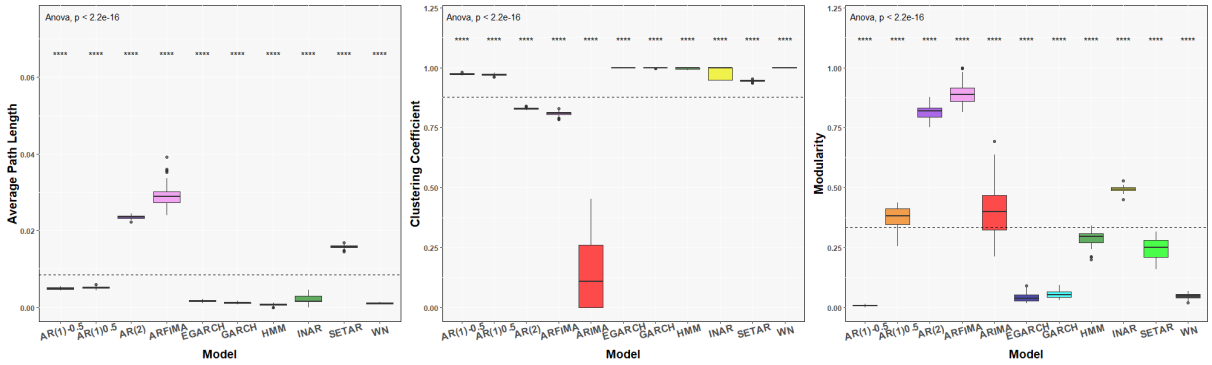


Figure 5.5: Boxplot (by time series model) of 50-QG topological features. The plots also show the results of applying a statistical significance test (ANOVA).

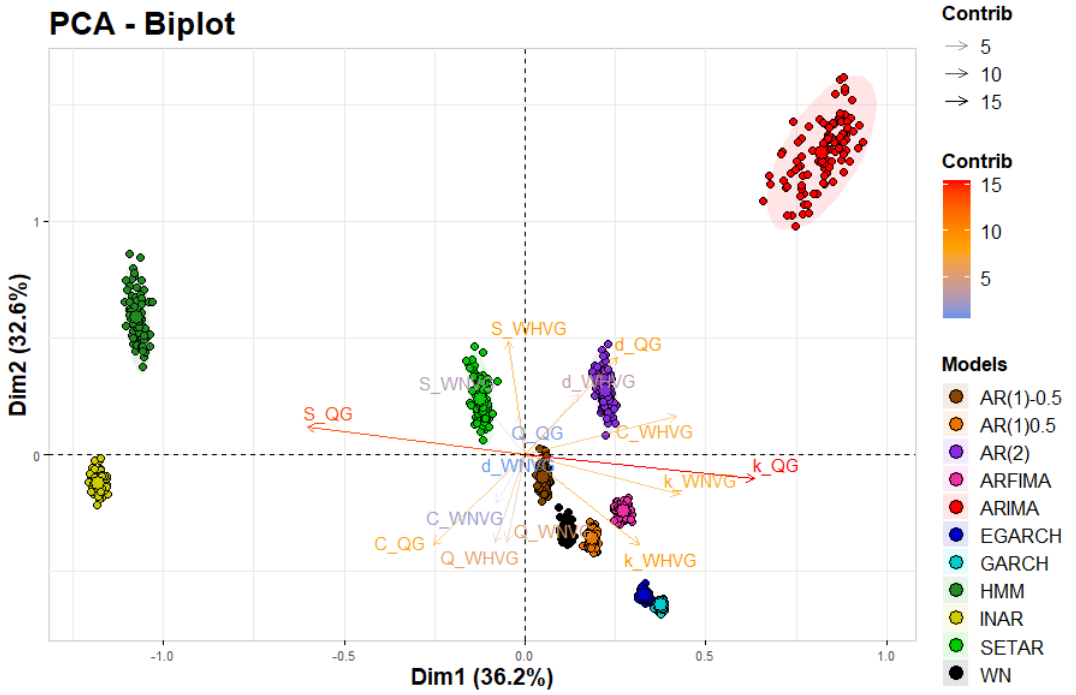


Figure 5.6: Bi-plot of the first two PC's for the synthetic dataset. Each Univariate Data Generating Process (UDGP) is represented by a color and the arrows represent the contribution of the corresponding feature to the PC's: the larger the size, the sharper the color, and the closer to the red the greater the contribution of the feature. Features grouped together are positively correlated while those placed on opposite quadrants are negatively correlated.

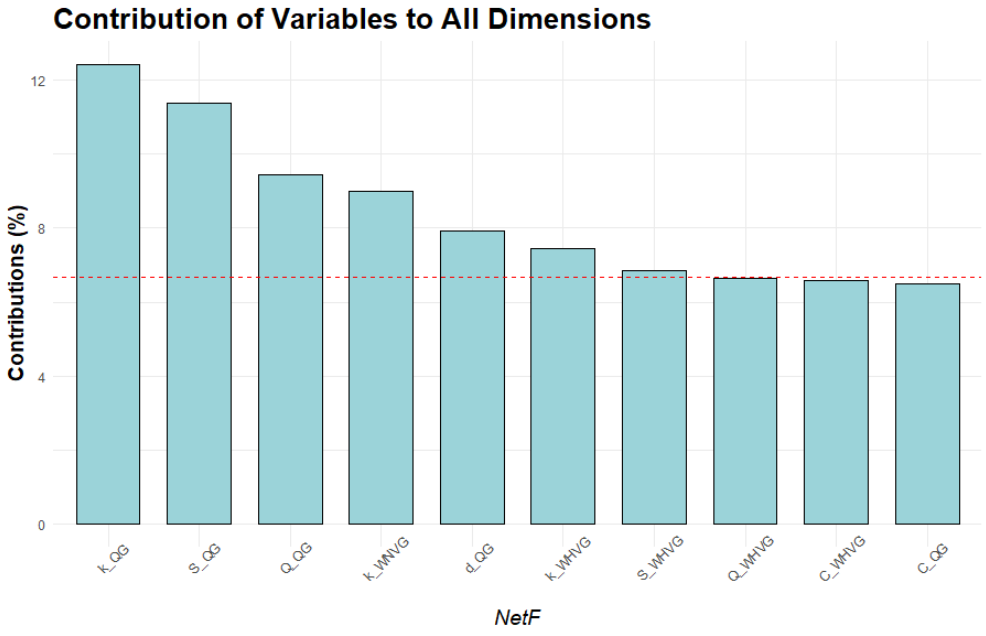


Figure 5.7: Bar plot with contributions of *NetF* features to the total of all 15 principal components formed by the PCA. The red dashed line on the plot indicates the expected average contribution.

5.4 Conclusions

In this chapter we introduce *NetF*, a novel set of 15 time series features, and we explore its ability to characterize univariate time series data. Our methodology relies on mapping the time series into complex networks using three different mapping methods: natural and horizontal visibility and quantile graphs (based on transition probabilities). We then extract five topological features for each mapped network, concatenating them into a single time series feature vector, and we describe in detail how we can do this in practice. We select the average weighted degree, average path length, number of communities, clustering coefficient, and modularity, which measure global characteristics, are simple to compute and interpret in the graph context and are commonly used in network analysis, thus capable of providing useful information about the structure and properties of the underlying systems.

To better understand the potential of our approach, we first perform an empirical evaluation on a synthetic dataset of 3300 networks, grouped in 11 different and specific time series models. Analyzing the weighted visibility (natural and horizontal) and quantile graphs feature space provided by *NetF*, we were able to identify sets of features that distinguish non-stationary from stationary time series, counting from real-valued time series, periodic from non-periodic time series, state time series from non-state time series and heteroskedastic time series. The non-stationarity time series have high values of average path length and low values of clustering coefficients in their QGs, and the opposite happens for the stationary time series. The counting series has the lowest value of the average weighted degree, the highest value of the number of communities in their QGs, and the lowest value of the clustering coefficient in WHVGs, while the opposite happens for the non-counting time series. For state time series the average weighted degree value in their weighted VGs is the lowest and the number of communities is high, the opposite happens for the non-state time series. Heteroskedastic time series are identified with high average weighted degree values of their WHVGs, compared to the other UDGPs.

The results show that *NetF* is able of capturing information regarding specific properties of time series data. In particular, features from different types of networks are able of capturing different information from the time series model (including the same base feature). This shows that different mapping methods translate the data information to the connections of the resulting graph differently.

NetF does not require typical time series preprocessing tasks, such as decomposing, differencing, or whitening. Moreover, our methodology is applicable to any time series, regardless of the nature of the data. The mappings and topological network features considered are global, but it is important to clarify that they do not constitute a "universal" solution. In particular, we found that the weighted versions of the visibility graph mappings used here produce better results than their unweighted versions, as we can see in previous works (Silva, 2018). Formulating a set of general features capable of fully characterizing a time series without knowing both the time series properties and the intended analysis is a difficult and challenging task (Kang et al., 2020).

Chapter 6

Features for Multivariate Time Series

The set of features available for multivariate time series is not as large as in the univariate setting, and is commonly extracted via statistical measures based on computations of causality, such as Granger causality (Runge et al., 2019a), correlation, such as CCF, as well as measures of distance between time series, such as dynamic time warping (Ruiz et al., 2021). Many of the MTS analysis methods focus on methods for dimensionality reduction and feature representation, commonly based on PCA (Granato et al., 2018), for example.

Recent literature (see Section 3.2.2) shows that multiplex versions of visibility graphs applied to MTS can achieve better accuracy and more promising results than conventional mappings into monoplex networks (Eroglu et al., 2018; Lacasa et al., 2015; Silva et al., 2021). However, as far as we are aware, the above-mentioned results do not work directly with inter-layer edges basing the analysis of the networks on topological measurements on the monoplex networks resulting from flattening approaches or through similarity measures on the individual layers of the multiplex networks. We believe that high-level network structures lead to less loss of data information after mapping functions allowing to expand the range of resources available in the literature and to explore more network components, such as inter-layer edges.

In this chapter, we propose to use multilayer network topological measures as features to MTS. In particular, we will make use of intra-layer and inter-layer edges to compute topological features to compare and analyze the serial and cross-dependencies, verifying whether they complement the intra-layer features that are normally used. Based on the underlying concepts of *NetF* (presented in Section 5.1), node centrality, graph distances, clustering, and community, and in the 3 types of MNet subgraphs previously mentioned in Section 2.3.2, we propose a set of MNet topological features based on MNet subgraphs structure which includes: i) common topological features extended to MNets and ii) a new feature constructed for MNets.

We start by introducing the new set of topological features. Next, we analyze this feature set on the new method for mapping time series into MHVG (see Section 4.1). Finally, we propose the *MNetF* as a diversified vector of multidimensional topological features extracted from the proposed multilayer time series network, MHVG and MQG.

6.1 Multilayer Network Topological Features

A common approach to analyzing multilayer networks through features is the direct approach, where the usual topological features of complex networks are extended to the MNet structure and its substructures. In this section, we follow this approach and extend five topological features to the subgraph of MNets. Furthermore, we propose a new feature based on the intra-layer degree and inter-layer degree measures. Figure 6.1 illustrates this process.

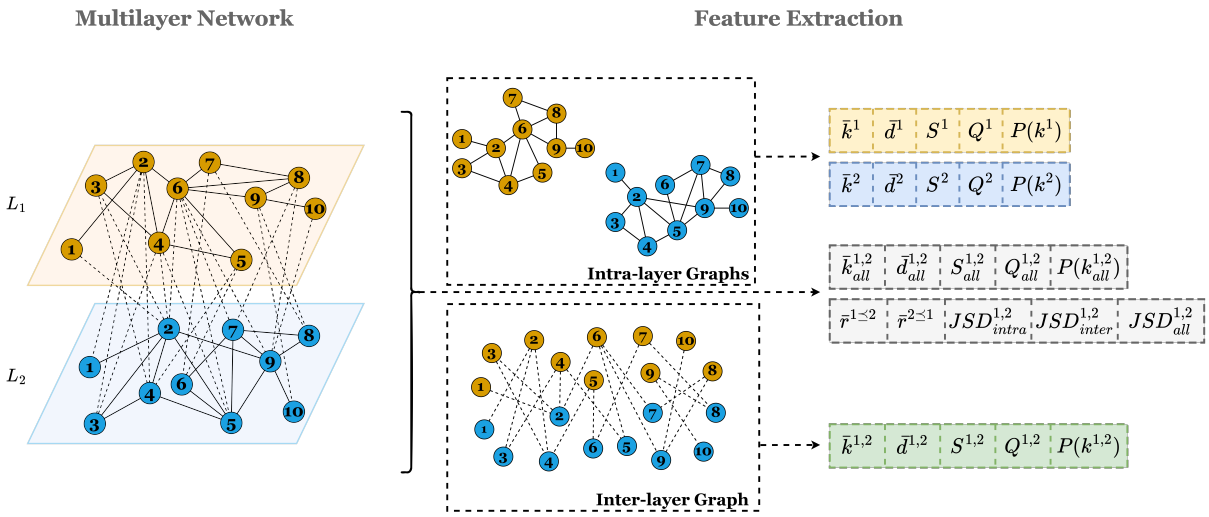


Figure 6.1: Schematic diagram of topological features extraction from MNet. From each subgraph, MNet (intra-layer, inter-layer, and all-layer graphs) are computed global topological features and relational features.

6.1.1 Topological Features Extended to MNets

Common network topological features (see Section 2.3.3) such as node centrality, graph distances, clustering, and community can be naturally extended to an MNet structure and all the subgraphs mentioned in Section 2.3.2. To illustrate, consider local centrality features for single-layer networks such as the *degree* k_i of a node-layer v_i, which represents the number of its adjacent edges. In an MNet, we can compute three variants of node degree for each individual layer α (with $\alpha = 1, \ldots, m$) that measures the number of adjacency edges of a given node v_i^α depending of the connections type, i.e., intra-layer edges, inter-layer edge, or both (intra-layer and inter-layer) edges:

6.1. MULTILAYER NETWORK TOPOLOGICAL FEATURES

- **intra-layer degree:** $k_i^{\alpha} = \sum_j A_{ij}^{\alpha}$
- **inter-layer degree:** $k_i^{\alpha \prec \beta} = \sum_j A_{ij}^{\alpha,\beta}$
- **all-layer degree:** $k_i^{\alpha \preceq \beta} = k_i^{\alpha} + k_i^{\alpha \prec \beta}$

where $\beta \neq \alpha$ and the symbol $\prec$ (and $\preceq$) expresses the inter-layer edges from a "source" layer α (and including intra-layer edges of the "source" layer) to a "destination" layer β. Note that *local* inter-layer and all-layer topological features are asymmetric features, that is, $k_i^{\alpha \prec \beta} \neq k_i^{\beta \prec \alpha}$ and $k_i^{\alpha \preceq \beta} \neq k_i^{\beta \preceq \alpha}$, since the feature is relative to node-layer v_i^{α} or node-layer v_i^{β}.

In general, any common (local) topological feature F_i can be easily extended to *intra-layer features*, F_i^{α}, just computing them over individual layers, to *inter-layer features*, $F_i^{\alpha \prec \beta}$, computing over inter-layer edges, and to *all-layer features*, $F_i^{\alpha \preceq \beta}$, which compute over both intra-layer and inter-layer edges.

An important feature associated with the degree is the ***degree distribution*** $P(k)$ that measures the fraction of nodes in a single-layer network with degree k. In this work, we analyze the three variants of degree distributions, $P(k^{\alpha})$, $P(k^{\alpha \prec \beta})$ and $P(k^{\alpha \preceq \beta})$, in layer $L_{\alpha}, \alpha = 1, \ldots, m$, associated with its intra-layer degree, inter-layer degree and all-layer degree, respectively.

To measure the similarity between pairs of layers in an MNet, we also use the ***Jensen–Shannon divergence*** (JSD) which measures the distance between two distributions. As an example, the JSD between intra-layer degree distributions $P(k^{\alpha})$ and $P(k^{\beta})$, ($JSD_{intra}^{\alpha,\beta}$) is defined as follows:

$$JSD(P(k^{\alpha})||P(k^{\beta})) = \frac{1}{2}KLD(P(k^{\alpha})||Q(k)) + \frac{1}{2}KLD(P(k^{\beta})||Q(k))$$

where $Q(k) = \frac{1}{2}(P(k^{\alpha}) + P(k^{\beta}))$ and KLD is the Kullback–Leibler divergence:

$$KLD(P(k^{\alpha})||Q(k)) = \sum_k P(k^{\alpha}) \log_2 \left(\frac{P(k^{\alpha})}{Q(k)} \right).$$

Similarly, we define JSD for the inter-layer degree distributions ($JSD_{inter}^{\alpha,\beta} = JSD(P(k^{\alpha,\beta})||P(k^{\beta,\alpha}))$) and the all-layer degree distributions ($JSD_{all}^{\alpha,\beta} = JSD(P(k^{\alpha,\beta})||P(k^{\beta,\alpha}))$). Note that JSD is a symmetrical version of the asymmetrical feature KLD. In the remainder of this work, we will refer to similarity measures, such as JSD, as *relational features*.

In addition, we also extend global topological features to MNets. These features involve all (sub)graph elements and therefore are symmetric. As an example, consider the *average degree* $\bar{k}$ which calculates the arithmetic mean of the degree k_i of all nodes in a single-layer network. As before, we can compute three variants of average degree in a MNet as follow,

- **average intra-degree:** $\bar{k}^{\alpha} = \frac{1}{|V_{\alpha}|} \sum_i k_i^{\alpha}$
- **average inter-degree:** $\bar{k}^{\alpha,\beta} = \frac{1}{|V_{\alpha}|+|V_{\beta}|} \left(\sum_i k_i^{\alpha \prec \beta} + \sum_j k_j^{\beta \prec \alpha} \right)$

- **average all-degree:** $\bar{k}_{all}^{\alpha,\beta} = \frac{1}{|V_\alpha|+|V_\beta|}\left(\sum_i k_i^{\alpha \preceq \beta} + \sum_j k_j^{\beta \preceq \alpha}\right)$

In short, we can compute a (global) topological feature $\boldsymbol{F}$ in the subgraphs of the MNet: intra ($\boldsymbol{F}^{\alpha}$), inter ($\boldsymbol{F}^{\alpha,\beta}$), and all-layer graphs ($\boldsymbol{F}_{all}^{\alpha,\beta}$).

Motivated by the *NetF* set of features proposed in Silva et al. (2022), namely based on the concepts of node centrality, graph distances, clustering, and community and the three types of MNet measurements defined above, we propose intra-layer, inter-layer, and all-layer, for each pair of layers, features as follows (see Table 6.1 for the formulation details):

- **Average Degree:** the average intra-degree $\bar{k}^{\alpha}$, average inter-degree $\bar{k}^{\alpha,\beta}$ and average all-degree $\bar{k}_{all}^{\alpha,\beta}$, as formulated above.
- **Average path length:** geodesic distances $d_{i,j}, i \neq j$ between node v_i and v_j corresponding to the length of the shortest paths between them, where the path length is the number of edges in the path. The *average (intra-/inter-/all-)path length* ($\bar{d}^{\alpha}$, $\bar{d}^{\alpha,\beta}$ and $\bar{d}_{all}^{\alpha,\beta}$) is the arithmetic mean of the shortest paths among all pairs of nodes in (intra, inter, and all-layer) graph.
- **Number of communities:** The *number of (intra-/inter-/all-)communities*, (S^{α}, $S^{\alpha,\beta}$ and $S_{all}^{\alpha,\beta}$), is the amount of groups/communities of nodes that are densely connected on the subgraph. These communities are found by performing random walks on the subgraph (intra, inter, and all-layer graph), so that short random walks tend to stay in the same community until the modularity value (defined below) cannot be increased anymore.
- **Modularity:** *(Intra-/Inter-/All-)modularity*, (Q^{α}, $Q^{\alpha,\beta}$ and $Q_{all}^{\alpha,\beta}$), measures how good a specific division of the corresponding subgraph G is into (intra-/inter-/all-)communities.

6.1.2 Ratio Degree: a New MNet Topological Feature

The *ratio degree* is a new topological feature for multilayer graphs, introduced here to relate intra-layer and inter-layer visibility.

For any two different layers L_α and L_β, $\alpha, \beta = 1, \ldots, m$, the ratio degree of node v_i^{α} from layer L_α to layer L_β is defined as

$$r_i^{\alpha \preceq \beta} = \frac{k_i^{\alpha \prec \beta}}{k_i^{\alpha}} \tag{6.1}$$

with $\alpha \neq \beta$ and $i = 1, \ldots, T$. The *average ratio degree*, $\bar{r}^{\alpha \preceq \beta}$, is the arithmetic mean of the ratio degree of the nodes of layer L_α. Note that the ratio degree and the average ratio degree are asymmetric, and thus it is not necessarily true that ${r_i}^{\alpha \preceq \beta} = {r_i}^{\beta \preceq \alpha}$ and that $\bar{r}^{\alpha \preceq \beta} = \bar{r}^{\beta \preceq \alpha}$.

Table 6.1 provides the formulation details of the multilayer topological features studied in this work.

Table 6.1: Summary formulation of topological features of multilayer networks [1].

Feature	Formulation	Note
Average Degree	$\bar{k}^{\alpha} = \frac{1}{N_\alpha}\sum_i k_i^{\alpha}$ $\bar{k}^{\alpha,\beta} = \frac{1}{N_{\alpha,\beta}}\left(\sum_i k_i^{\alpha\prec\beta} + \sum_j k_j^{\beta\prec\alpha}\right)$ $\bar{k}_{all}^{\alpha,\beta} = \frac{1}{N_{\alpha,\beta}}\left(\sum_i k_i^{\alpha\preceq\beta} + \sum_j k_j^{\beta\preceq\alpha}\right)$	
Degree Distribution	$P(k^{\alpha}) = \frac{n_{k^\alpha}}{N_\alpha}$ $P(k^{\alpha\prec\beta}) = \frac{n_{k^{\alpha\prec\beta}}}{N_\alpha}$ $P(k^{\alpha\preceq\beta}) = \frac{n_{k^{\alpha\preceq\beta}}}{N_\alpha}$	n: number of nodes v_i^α with the corresponding degree k
Average Path Length	$\bar{d}^{\alpha} = \frac{1}{N_\alpha(N_\alpha-1)}\sum_{i\neq j} d_{i,j}^{\alpha}$ $\bar{d}^{\alpha,\beta} = \frac{1}{N_{\alpha,\beta}(N_{\alpha,\beta}-1)}\sum_{i\neq j} d_{i,j}^{\alpha,\beta}$ $\bar{d}_{all}^{\alpha,\beta} = \frac{1}{N_{\alpha,\beta}(N_{\alpha,\beta}-1)}\sum_{i\neq j} d_{i,j,all}^{\alpha,\beta}$	$d_{i,j}$: length of the shortest paths between v_i and v_j in the corresponding subgraph
Number of Communities	$S^{\alpha} = \lvert\mathcal{C}^{\alpha}\rvert$ $S^{\alpha,\beta} = \lvert\mathcal{C}^{\alpha,\beta}\rvert$ $S_{all}^{\alpha,\beta} = \lvert\mathcal{C}_{all}^{\alpha,\beta}\rvert$	$\mathcal{C}$: set of communities in corresponding subgraph
Modularity	$Q^{\alpha} = \frac{1}{2\lvert E^\alpha\rvert}\sum_{i,j}\left[B_{i,j} - \frac{k_i k_j}{2\lvert E^\alpha\rvert}\right]\delta\left(\mathcal{C}_i,\mathcal{C}_j\right)$ $Q^{\alpha,\beta} = \frac{1}{2\lvert E^{\alpha,\beta}\rvert}\sum_{i,j}\left[B_{i,j} - \frac{k_i k_j}{2\lvert E^{\alpha,\beta}\rvert}\right]\delta\left(\mathcal{C}_i,\mathcal{C}_j\right)$ $Q_{all}^{\alpha,\beta} = \frac{1}{2\lvert E_{all}^{\alpha,\beta}\rvert}\sum_{i,j}\left[B_{i,j} - \frac{k_i k_j}{2\lvert E_{all}^{\alpha,\beta}\rvert}\right]\delta\left(\mathcal{C}_i,\mathcal{C}_j\right)$	$\boldsymbol{B} = \boldsymbol{A}^{\alpha}$ $\boldsymbol{B} = \begin{bmatrix} \mathbf{0} & \boldsymbol{A}^{\alpha,\beta} \\ \boldsymbol{A}^{\beta,\alpha} & \mathbf{0} \end{bmatrix}$ $\boldsymbol{B} = \begin{bmatrix} \boldsymbol{A}^{\alpha} & \boldsymbol{A}^{\alpha\beta} \\ \boldsymbol{A}^{\beta\alpha} & \boldsymbol{A}^{\beta} \end{bmatrix}$
Jensen–Shannon Divergence	$JSD_{intra}^{\alpha,\beta} = JSD(P(k^{\alpha})\lvert\rvert P(k^{\beta}))$ $JSD_{inter}^{\alpha,\beta} = JSD(P(k^{\alpha\prec\beta})\lvert\rvert P(k^{\beta\prec\alpha}))$ $JSD_{all}^{\alpha,\beta} = JSD(P(k^{\alpha\preceq\beta})\lvert\rvert P(k^{\beta\preceq\alpha}))$	
Average Ratio Degree	$\bar{r}^{\alpha\preceq\beta} = \frac{1}{\lvert N_\alpha\rvert}\sum_i r_i^{\alpha\preceq\beta}$	

[1] Remember that V^α is the set of nodes in layer L_α, we define $N_\alpha = |V^\alpha|$ and $N_{\alpha,\beta} = |V^\alpha| + |V^\beta|$ the number of nodes in the corresponding layer(s).

6.2 Empirical Evaluation of MHVG Features

In this section, we investigate whether the MHVG mapping method proposed in Section 4.1 and the features set described in Table 6.1 are useful for characterizing MTS data by evaluating the performance of the methodology for MTS mining tasks. We use synthetic bivariate time series, generated from bivariate time series models to control for MTS correlation (serial and cross) properties. First, we make some considerations about the implementation of the methodology.

6.2.1 Implementation Details

We briefly describe how we computed our proposed methodology, illustrated in Figure 6.2. For illustrative purposes, we used $m = 2$, but the method is extensible to any value of m.

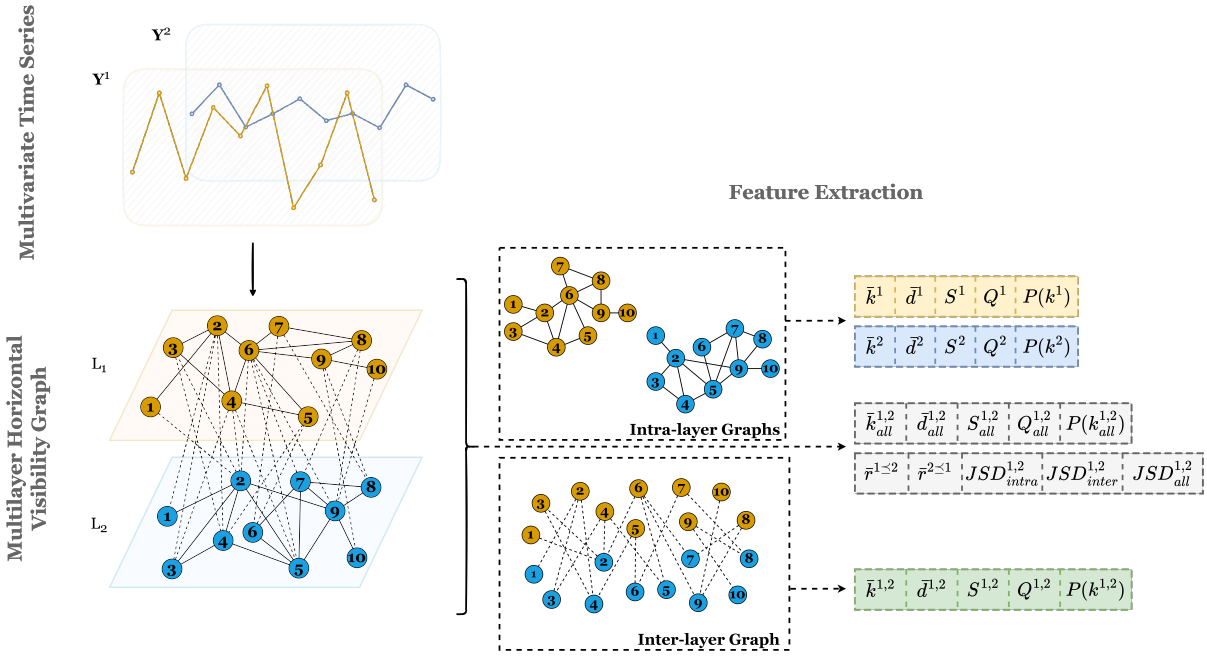


Figure 6.2: Schematic diagram of the multilayer network features set extraction process. A multivariate time series $\boldsymbol{Y}$ is mapped into a multilayer horizontal visibility graph. And for each of its subgraphs (intra-layer, inter-layer, and all-layer graphs) are computer the global topological features ($\bar{k}, \bar{d}, S, Q$ and $P(k)$) and relational features ($\bar{r}$ and JSD).

To map a multivariate time series $\boldsymbol{Y}$ into an MHVG, we follow Algorithm 2. The intra-layer HVGs, $\{G^{\alpha}\}_{\alpha=1}^{m}$, for each time series component, $\{\boldsymbol{Y}_{\alpha}\}, \alpha = 1, \ldots, m$, are created using Algorithm 5 proposed in Luque et al. (2009). The inter-layer edges are added following the mapping method based on cross-horizontal visibility criteria proposed in Section 4.1.1, and using Algorithm 6. Subgraphs corresponding to intra-, inter-, and all-layers, are fixed via corresponding adjacency sub-matrices of the MHVG (see Sections 2.3.2). And the corresponding topological features described above are computed using the methodologies and algorithms described below.

The **average degree ($\bar{k}$)** and **average ratio degree ($\bar{r}$)** are calculated by the arithmetic mean of the degrees k_i and ratio degrees r_i, respectively, of all node v_i in the respective subgraph. In this work, the **average path length ($\bar{d}$)** follows an algorithm that computes the average shortest path length between all pairs of nodes (of respective subgraphs) using a breadth-first search algorithm. To calculate the **number of communities (S)**, the function used makes use of the known "Louvain" algorithm that finds community structures by multi-level optimization of **modularity (Q)** measure (see Blondel et al. (2008) for more details). And the **degree distributions ($P(k)$)** and **Jensen–Shannon divergence (JSD)** are implemented as described above section.

We used C++ and the set of libraries (such as `igraph` and standard libraries) to implement the data structure to store an MNet and compute the functions to extract the topological features.

6.2.2 Synthetic Datasets

We consider a set of six bivariate time series models ($m = 2$), denoted by Multivariate Data Generating Processes (MDGPs), summarized in Table 6.2. These MTS models present a set of particular characteristics in terms of serial and cross-correlation (see Section2.1.2), namely: *white noise* (WN) processes simulate noise effects, one process does not present any kind of correlation, and the other presents a strong contemporaneous correlation; *vector autoregression* (VAR) processes simulate smooth linear data, presenting both serial and cross-correlation; *vector generalized autoregressive conditional heteroskedasticity* (VGARCH) processes simulate nonlinear data with persistent periods of high or low volatility. The parameters of each MDGP are chosen so that the data exhibits a range of serial and cross-correlation properties as described in Table 6.2. A detailed description of the MDGP and their properties as well as computational details are presented in Appendix C.1.

For each MDGP in Table 6.2, we generated 100 instances of length $T = 10000$. As illustrated in Figure 6.2, we map each bivariate time series into an MHVG, highlight the intra-, inter-, and all-layer graphs and extract the corresponding topological features.

To illustrate the procedure, we represent in Figure 6.3 one instance with 300 observations of each MDGP and the corresponding cross-correlation (CCF) plot (first two columns of the plot), the intra-, inter-, and all-layers degree distributions on a semi-logarithmic scale (last three columns of the plot). These degree distributions are computed as the arithmetic mean of the degree distributions of the 100 simulated instances [2]. The plots clearly show that the degree distributions are different across the MDGPs. In fact, Luque et al. (2009) has shown that the intra-layer degree distribution for white noise (uncorrelated data) follows a power law $\left(P(k) = \frac{1}{3}\left(\frac{2}{3}\right)^{k-2}\right)$ and our results indicate that strong serial correlation leads to intra-layer degree distributions that are positively skewed: as illustrated in Appendix C.1, the `sVAR` is the only MDGP that produces data with strong serial correlation. The degree distribution for the inter-layer subgraphs does not have an algebraic close form even in the simplest case of

[2] In Figure C.2 we present the variability of the 100 samples of the degree distributions in the form of boxplots

Table 6.2: Summary about the multivariate data generating processes (bivariate time series models) of the synthetic data. Parameters, the main characteristic of the datasets, and notation is also included. See Appendix C.1 for more details.

MDGP	Parameters	Characteristics	Notation
Independent White Noise	$\boldsymbol{\epsilon}_t \sim N(0,1)$	Noise effect No correlation	iBWN
Correlated White Noise	$\begin{bmatrix}\epsilon_{1,t}\\ \epsilon_{2,t}\end{bmatrix} \sim N\left(0, \begin{bmatrix}1.00 & 0.86\\ 0.86 & 1.50\end{bmatrix}\right)$	Noise effect No serial correlation Cross-correlation	cBWN
Weak VAR(1)	$\boldsymbol{\varphi} = \begin{bmatrix}2.50\\ 0.50\end{bmatrix}, \boldsymbol{\phi} = \begin{bmatrix}0.20 & 0.10\\ 0.02 & 0.10\end{bmatrix}$ $\boldsymbol{\epsilon}_t \sim \begin{bmatrix}1.00 & 0.10\\ 0.10 & 1.50\end{bmatrix}$	Weak correlation (serial and cross)	wVAR
Strong VAR(1)	$\boldsymbol{\varphi} = \begin{bmatrix}0\\ 0\end{bmatrix}, \boldsymbol{\phi} = \begin{bmatrix}0.70 & 0.02\\ 0.30 & 0.80\end{bmatrix}$ $\boldsymbol{\epsilon}_t \sim \begin{bmatrix}1.00 & 0.86\\ 0.86 & 1.50\end{bmatrix}$	Strong correlation (serial and cross, lagged and contemporaneous)	sVAR
Weak VGARCH(1,1)	$\boldsymbol{\omega} = \begin{bmatrix}0.05\\ 0.02\end{bmatrix}, \boldsymbol{\alpha} = \begin{bmatrix}0.10 & 0.00\\ 0.00 & 0.05\end{bmatrix}$ $\boldsymbol{\beta} = \begin{bmatrix}0.85 & 0.00\\ 0.00 & 0.88\end{bmatrix}, \boldsymbol{\epsilon}_t \sim \begin{bmatrix}1.00 & 0.10\\ 0.10 & 1.50\end{bmatrix}$	No serial correlation Weak cross-correlation	wGARCH
Strong VGARCH(1,1)	$\boldsymbol{\omega} = \begin{bmatrix}0.05\\ 0.02\end{bmatrix}, \boldsymbol{\alpha} = \begin{bmatrix}0.10 & 0.00\\ 0.00 & 0.05\end{bmatrix}$ $\boldsymbol{\beta} = \begin{bmatrix}0.85 & 0.00\\ 0.00 & 0.88\end{bmatrix}, \boldsymbol{\epsilon}_t \sim \begin{bmatrix}1.00 & 0.86\\ 0.86 & 1.50\end{bmatrix}$	Strong contemporaneous cross-correlation	sGARCH

two uncorrelated white noises. However, the extensive simulations (see Figure C.2) and the analytical results that we present in Appendix E indicate that the inter-layer degree distribution does not follow the power law $P(k) = \frac{1}{3}\left(\frac{2}{3}\right)^{k-2}$, as illustrated in the first line, the third column of Figure 6.3. The plots also indicate that inter-layer degree distribution depends both on the correlation between the two time series (CCF represented in the second column of the plot in Figure 6.3) and the serial correlation within each time series. Moreover, we note that inter-layer degree distributions for sVAR are positively skewed, for GARCH models, wGARCH and sGARCH, are exponentially shaped while the remaining are approximately linear. Once again, a slower decay of the lagged correlation leads to a longer tail in the degree distribution. Also, the degree distribution curves corresponding to the GARCH models stand out from the others, especially the inter- and all-layer degree distributions. The exponential shape of the inter-layer degree distributions is induced by the heteroscedasticity and volatility clusters in the data which limit cross-horizontal visibility to the nearest neighbors.

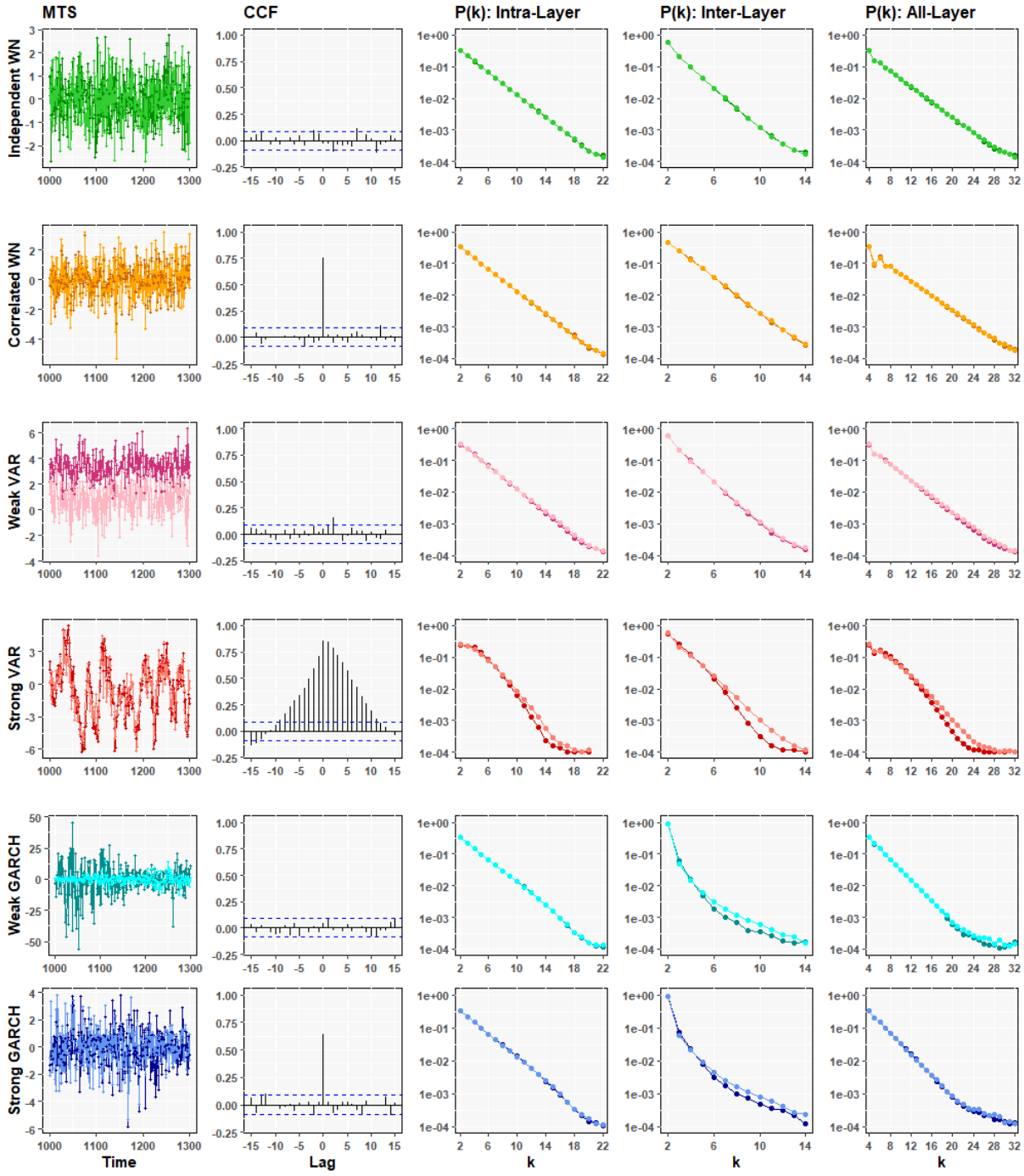


Figure 6.3: Analysis plot of multivariate DGP. The first column shows a subset of timestamps from each MDGP bivariate time series, the second column plots the cross-correlation function between their corresponding time series components, and the last three columns show the intra, inter, and both layers degree distribution plots of the corresponding MHVG's. The plots corresponding to the degree distributions are on a semi-logarithmic scale. Lines of different colors refer to the different MDGP models ($\boldsymbol{Y}_t$), where the darkest colors refer to their first-time series components ($Y_{1,t}$) and the lighter colors to the second components ($Y_{2,t}$).

Table 6.3: Mean values (standard deviation) of the 100 instances of each MDGP for each topological global feature from intra-layer graphs, G^1 and G^2, and inter-layer graph, $G^{1,2}$, resulting from the corresponding MHVGs. The columns of the tables are colored with a gradient based on the mean values: cells with the highest value are coloured red, cells with the lowest value are coloured white, and the remainder with a hue of red colour proportional to its value in the respective column.

MDGP	Average Degree			Average Path Length			Number of Communities			Modularity		
	$\bar{k}^1$	$\bar{k}^2$	$\bar{k}^{1,2}$	$\bar{d}^1$	$\bar{d}^2$	$\bar{d}^{1,2}$	S^1	S^2	$S^{1,2}$	Q^1	Q^2	$Q^{1,2}$
iBWN	0.805 (0.081)	0.756 (0.077)	0.615 (0.033)	0.044 (0.019)	0.049 (0.023)	0.012 (0.024)	0.265 (0.083)	0.312 (0.083)	0.206 (0.064)	0.150 (0.055)	0.207 (0.056)	0.319 (0.093)
cBWN	0.802 (0.083)	0.752 (0.080)	0.940 (0.078)	0.045 (0.022)	0.050 (0.022)	0.007 (0.015)	0.260 (0.084)	0.310 (0.090)	0.126 (0.080)	0.150 (0.051)	0.196 (0.062)	0.181 (0.141)
wVAR	0.790 (0.079)	0.759 (0.090)	0.615 (0.033)	0.058 (0.022)	0.056 (0.025)	0.009 (0.016)	0.342 (0.093)	0.338 (0.100)	0.211 (0.062)	0.287 (0.058)	0.277 (0.062)	0.308 (0.099)
sVAR	0.561 (0.121)	0.601 (0.108)	0.683 (0.092)	0.449 (0.050)	0.328 (0.039)	0.011 (0.025)	0.791 (0.104)	0.700 (0.121)	0.195 (0.069)	0.893 (0.046)	0.857 (0.064)	0.263 (0.103)
wGARCH	0.540 (0.159)	0.554 (0.140)	0.102 (0.135)	0.380 (0.136)	0.325 (0.097)	0.252 (0.188)	0.239 (0.104)	0.282 (0.083)	0.696 (0.212)	0.185 (0.057)	0.207 (0.081)	0.669 (0.210)
sGARCH	0.542 (0.184)	0.505 (0.186)	0.146 (0.174)	0.390 (0.138)	0.385 (0.149)	0.232 (0.217)	0.234 (0.103)	0.300 (0.105)	0.670 (0.220)	0.179 (0.065)	0.212 (0.065)	0.626 (0.240)

The results for all the 21 features introduced in Section 6.1 and all MDGPs, organized by subgraph structure, are summarized, mean (standard deviation), in Tables 6.3 and 6.4. The values have been Min-Max normalized (across models) for comparison purposes since the range of the different features varies across the different MDGPs. The cells in the tables are coloured with a gradient based on the mean values in each column: cells with the highest value are coloured red, cells with the lowest value are coloured white, and the remainder with a hue of red colour proportional to its value in the respective column.

The results indicate that each set of features - intra-layer (first two columns of each feature in Table 6.3), inter-layer (third column of each feature in Table 6.3), all-layer (first four columns of Table 6.4) and relational (last five columns of Table 6.4) - distinguishes two groups of MTS depending on properties pertaining to correlation (serial and cross) and volatility clustering.

Focusing on the characteristic of data heteroscedasticity, we can see that the mapping that result in Cross-HVG is very sensitive to this property. In fact, the visibility criterion is very dependent on the very high and very low values of the data over time, and therefore the heteroscedasticity and correlation between the data impose even more limits on the visibility between values of different variables over time, which is reflected in greater variability of the topological features of the Cross-HVG. This fact can be observed in Figure C.2 where the values of the (intra, inter and all-layer graph) degree distributions corresponding to the 100 samples of each MDGP analyzed here are plotted as boxplots.

Table 6.4: Mean values (standard deviation) for each topological global and relational features from all-layer graphs, $G^{1,2}_{all}$, resulting from MHVGs of MDGP, computed over the 100 instances of each MDGP. The columns of the tables are colored with a gradient based on the mean values: cells with the highest value are coloured red, cells with the lowest value are coloured white, and the remainder with a hue of red colour proportional to its value in the respective column.

MDGP	Average Degree	Average Path L.	Num. of Comm.	Modular.	Average Ratio Deg.		Jensen–Shannon Divergence		
	$\bar{k}^{1,2}_{all}$	$\bar{d}^{1,2}_{all}$	$S^{1,2}_{all}$	$Q^{1,2}_{all}$	$\bar{r}^{1\preceq 2}$	$\bar{r}^{2\preceq 1}$	$JSD^{\alpha,\beta}_{intra}$	$JSD^{\alpha,\beta}_{inter}$	$JSD^{\alpha,\beta}_{all}$
iBWN	0.617 (0.033)	0.042 (0.018)	0.237 (0.107)	0.338 (0.052)	0.586 (0.028)	0.582 (0.037)	0.170 (0.071)	0.034 (0.0355)	0.090 (0.047)
cBWN	0.940 (0.078)	0.051 (0.018)	0.382 (0.087)	0.507 (0.064)	0.942 (0.071)	0.931 (0.084)	0.166 (0.079)	0.061 (0.076)	0.244 (0.236)
wVAR	0.616 (0.033)	0.054 (0.022)	0.305 (0.107)	0.413 (0.049)	0.565 (0.032)	0.571 (0.034)	0.207 (0.074)	0.034 (0.041)	0.096 (0.051)
sVAR	0.682 (0.092)	0.457 (0.04955)	0.457 (0.127)	0.842 (0.05673)	0.574 (0.106)	0.579 (0.091)	0.682 (0.128)	0.120 (0.148)	0.314 (0.225)
wGARCH	0.104 (0.134)	0.297 (0.065)	0.310 (0.108)	0.206 (0.076)	0.103 (0.136)	0.097 (0.127)	0.154 (0.065)	0.085 (0.110)	0.075 (0.065)
sGARCH	0.147 (0.173)	0.388 (0.120)	0.431 (0.118)	0.3954 (0.088)	0.149 (0.179)	0.145 (0.170)	0.149 (0.077)	0.114 (0.161)	0.137 (0.172)

To understand which MNet topological features capture the specific properties of the MTS models, we perform PCA on the feature space. Figure 6.4 represents a bi-plot obtained using the intra-, inter-, all-layer, and relational features, with the two principal components (PC) explaining 83.8% of the variance. The bi-plots resulting from PCA in restricted feature sets are represented in Figure C.3 (Appendix C.1). Overall, we can say that the average degree and average ratio degree, $\bar{k}$ and $\bar{r}$, are positively and negatively correlated, respectively, with the average path length, $\bar{d}$. The community-related features of the intra- and all-layer graphs are positively correlated but less correlated with the community-related features of the inter-layer graphs. The features that most contribute to the first two PCs are the $\bar{k}^{1,2}$, $S^{1,2}$ and $Q^{1,2}$ of the inter-layer graphs, the $\bar{k}^{1,2}_{all}$ of the all-layer graphs, the $\bar{r}^{1\preceq 2}$ and $\bar{r}^{2\preceq 1}$ of the relational layers, and Q^1 and Q^2 of the intra-layer graphs (see Figure C.4 of Appendix C.1).

Figure 6.4 clearly shows four groups of models, GARCH, `sVAR`, `cBWN` and a group constituted by the `wVAR` and the `iBWN` and identifies the topological features that characterize them.

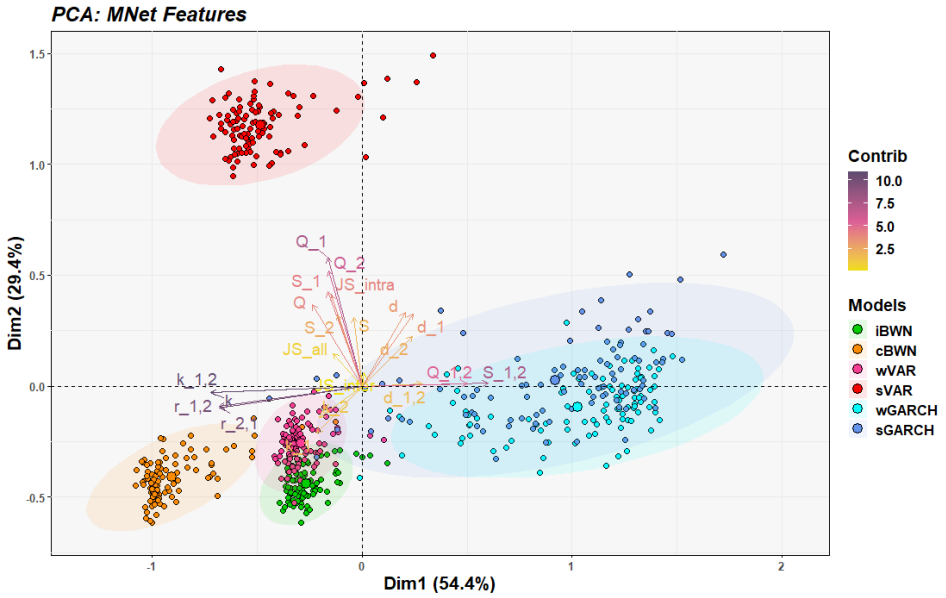


Figure 6.4: Bi-plot of the first two principal components (PC) of principal component analysis for the Data Generating Process (MDGP). Each MDGP is represented by a different color and the arrows represent the contributions of the MNet features to the PCs, the larger the size, sharpness, and closer to the red the greater the contribution of the feature. Features set together are positively correlated and those placed on opposite quadrants are negatively correlated.

The strong ACF and CCF of the `sVAR` are represented by high values for the number of communities and modularity in its intra- and all-layer graphs. Inter-layer graphs present higher values of community-related features for GARCH models. The average path length represents the GARCH models, in particular, the average path length of the all-layer graphs tries to distinguish both `wGARCH` and `sGARCH`. The strong contemporaneous CCF of the `cBWN` is represented by high values of average ratio degree, such as the average degree values of its inter and all-layer graphs. The `iBWN` and `wVAR`, are represented by high values of intra-layer average degree.

The above results indicate that the topological features extracted from MHVG are adequate as a set of MTS features.

6.3 *MNetF*: a Novel Set of Multivariate Time Series Features

Based on the results obtained in the previous section and on the *NetF* concept for univariate time series, we propose a more complete set of global topological features of MNet: the ***MNetF*** feature vector. In general *MNetF* is defined as a vector of global topological features formed by concatenating the vector of MNets features (introduced in 6.1) from the resulting MHVG and MQG of a multivariate time series. It comprises *intra-layer* topological features, *inter-layer* topological features, *all-layer* topological features, and *relational* features which are computed for an MTS for the two different mapped multilayer networks.

6.3. *MNetF*: A NOVEL SET OF MULTIVARIATE TIME SERIES FEATURES

MNetF is constituted by different features corresponding to the concatenation of six different topological features from intra, inter, and all-layer subgraphs, as explained in Section 6.1 ($\bar{k}$, the average degree; $\bar{d}$, the average path length; S, the number of communities; Q, the modularity; $\bar{r}$, the average ration degree; JSD, the Jensen–Shannon divergence), each of them applied to three different mappings of the MTS, as explained in Chapter 4 (*MHVG*, the multilayer horizontal visibility graph; *MQG*, the multilayer quantile graph with $\eta = 50$). To compute *MNetF* we follow the same steps described in Section 6.2.1, with the addition of two new steps: repeat the process described for MQG, and concatenate the two resulting feature vectors.

Our main goal is to provide a varied set of representative features that expose different properties captured by the topology of the mapped MNet, improving the previous results.

Next, we investigate, via synthetic multivariate datasets, whether the set of features introduced to *MNetF* are useful for characterizing multivariate time series data. To this end, we consider the same MDGP analyzed in the previous section: a set of six linear and nonlinear bivariate time series models with particular characteristics in terms of serial and cross-correlation.

We study the feature space with the aim of understanding which of the topological multidimensional features capture specific properties of the time series models. In Figure 6.5 we represent a bi-plot obtained using a total of 42 features (21 for each mapping method) and with the two PC's explaining 84% of the variance. We can see that the clustering of the MTS samples has improved since the clusters seem denser. And we can also observe a better distinction between the `sGARCH` and `wGARCH` models that is more noticeable in a three-dimensional space (see Figure C.5 in Appendix). The introduction of new topological features from different kinds of MNets seems to distinguish other properties of the data. Specifically, if we analyze only the features extracted from MQGs (see the feature space in Figure 6.6), we can say that the set of features of the MQGs makes a better distribution of the samples of the models by the feature space. Therefore, we can say that the MQG topological features are the ones that best distinguish heteroskedastic data with different cross-correlation properties. Since the QG are very good at characterizing the dynamic variations of the data and consequently the heteroscedasticity properties. We can see that the average degree and the number of communities of intra-layer graphs of MQG try to place the WN models in the third quadrant while the same features for inter-layer graphs try to place the VGARCH and VAR models weakly correlated in the second quadrant. If we analyze Figure C.6 which shows the features of *MNetF* that most contribute to the PCs, we can see that they are the features related to the MQGs.

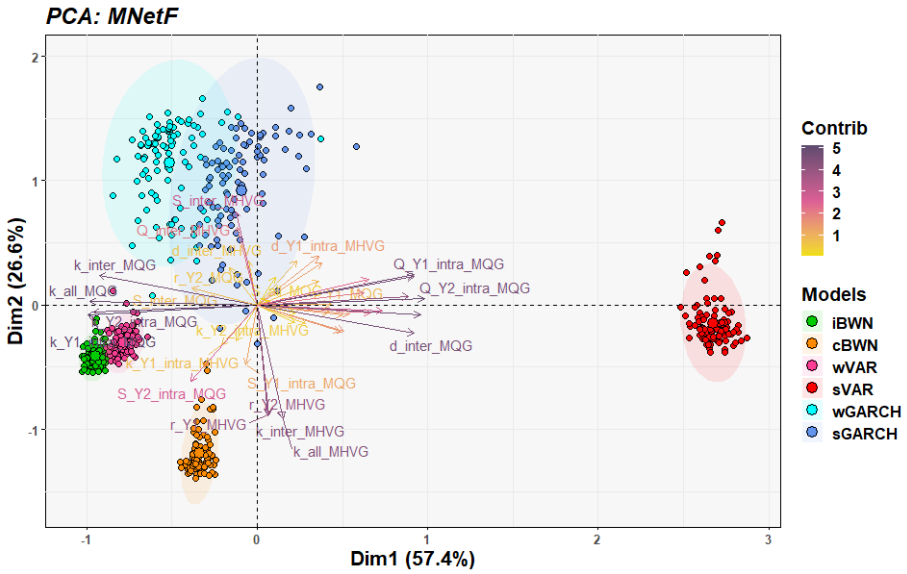


Figure 6.5: Bi-plot of the first two PCs of *MNetF* feature set for the synthetic bivariate dataset. Each Multivariate Data Generating Process (MDGP) is represented by a color and the arrows represent the contribution of the corresponding feature to the PC's: the larger the size, the sharper the color, and the closer to the red the greater the contribution of the feature. Features grouped together are positively correlated while those placed on opposite quadrants are negatively correlated.

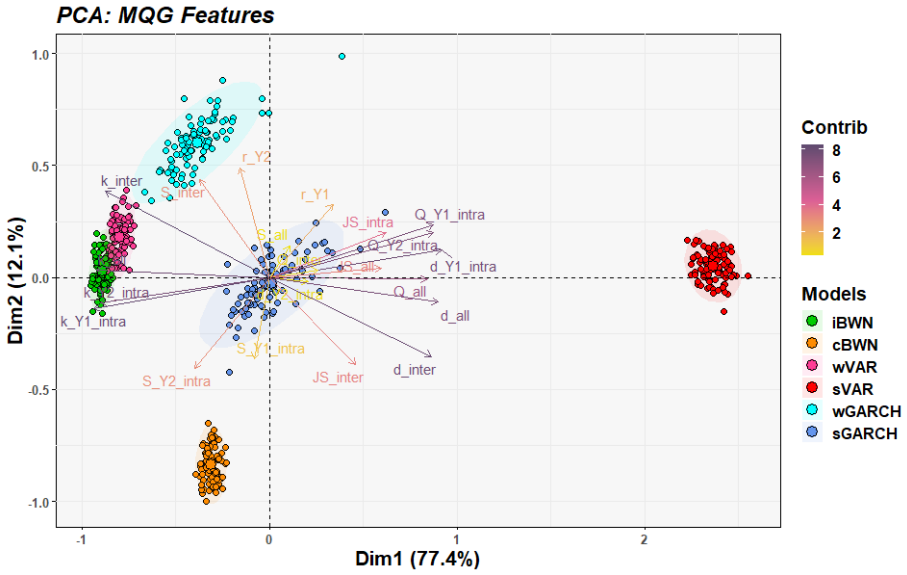


Figure 6.6: Bi-plot of the first two PCs of MQG topological feature set for the synthetic bivariate dataset. Each Multivariate Data Generating Process (MDGP) is represented by a color and the arrows represent the contribution of the corresponding feature to the PC's: the larger the size, the sharper the color, and the closer to the red the greater the contribution of the feature. Features grouped together are positively correlated while those placed on opposite quadrants are negatively correlated.

6.4 Conclusions

To assess the proposed MHVG method, we analyzed a specific set of topological features of multilayer networks. These features are based on concepts of node centrality, graph distances, clustering, communities, and similarity measures. Each feature extracted from all the subgraphs of the resulting multilayer network structure: *intra-layer graphs* (only intra-layer edges), *inter-layer graphs* (only inter-layer edges) and *all-graphs* (with both intra and inter-layer edges).

We perform an empirical evaluation on a set of 600 synthetic bivariate time series, grouped in 6 different and specific statistical models, that result in a dataset of 600 MHVGs. To understand the potential of our proposed mapping method, we first analyze the degree distributions of the intra, inter, and all-layer subgraphs of MHVGs. We were able to identify the specific properties of multivariate time series models, namely, we were able to relate weak and strong cross-correlation with shapes of the inter-layer degree distribution curves and weak and strong autocorrelation with shapes of the intra-layer degree distribution curves. In particular, we were also able to relate the persistence of strong correlations to distributions (that result in positively skewed shape) that have a longer right tail. Adding to the correlation properties (both auto and cross, contemporaneous and lagged), the properties of the statistical models, such as heteroscedasticity and smoothness, resulting in inter and all-layer degree distributions with different shape curves.

We also investigated the global features of the subgraphs (intra, inter, and all-layer). Community-related features from intra and all-layer graph highlight the strongly VAR models, with high and persistent autocorrelation and cross-correlation, as well as with smoothness, and from inter-layer graphs highlight the heteroscedasticity models, both weak and strong VGARCH models. However, the values of average path length from all-layer graphs seem to distinguish the properties of weakly and strongly correlated. The average intra-degree has higher values for independent white noise and weak VAR models, but not distinguishing them.

The new similarity measure proposed in this work, average ratio degree, seems to differentiate well the highly correlated contemporary white noise models, which leads to a similarity in its inter-layer degree and intra-layer degree features. In the context of this work, based on the synthetic models chosen for analysis, the Jensen–Shannon divergence measure is only useful to characterize strong VAR models that present very strong and very persistent correlations, unlike the other models. However, this feature can be quite useful in real contexts, where the different variables of a multivariate time series can follow different dynamic models that will be captured by this feature. We intend to investigate this and other characteristics, as well as other topological features of multilayer networks in our future works.

Similar to what was done for the univariate case, we also propose a much larger set of features, which we call *MNetF*, which result from features of the different types of multivariate networks that we propose in this work: MHVG and MQG. Making use of the proposed set of multivariate time series models to analyze MHVG networks, we apply the proposed methodology to the *MNetF*. The results show that the MQG mappings are sufficient to distinguish the characteristics inherent to each of the models analyzed in this work. Combining all features into a single vector, the *MNetF*, shows slightly worse results for this dataset.

The purpose of this work is not to find a unique feature vector restricted to all types of time series data. In fact, this is very far from being possible. Our goal is to contribute to a diverse set of multidimensional topological features that, in their particularities, can be used together or separately to analyze various types of multivariate temporal data. In particular, it is possible to apply feature selection methods on the proposed *MNetF* vector in order to retain the features that best fit the data under analysis.

To conclude, this chapter proposes a procedure to map multivariate time series into multilayer networks as a mean to obtain a rich set of multivariate time series features that can be used for mining tasks. Multilayer networks encode significantly more information than their isolated single layers since they incorporate correlations between the nodes in different layers and between the statistical properties of layers (time series components). This was proven by the empirical analysis carried out in this chapter, where the introduction of inter-layer graph features and all-layer graph features improve the results obtained with solely intra-layer features.

Chapter 7

Mining Time Series via Complex Networks

In this chapter, we illustrate the usefulness of complex networks based time series features in data mining tasks with a case study regarding time series clustering. We focus on the whole (univariate and multivariate) time series clustering. Within the various possible methodologies, we will focus on feature-based clustering methodologies (Maharaj et al., 2019), where (univariate or multivariate) time series data are represented by a set of descriptive characteristics that are passed as input to the clustering methods. We use the *NetF* and *MNetF* presented, respectively, in Sections 5.1 and 6.3, to clustering analysis of synthetic and real-world and benchmark (univariate and multivariate) datasets.

For purposes of evaluating and comparing the approaches proposed in this work, we use two other sets of conventional benchmark time series features (Henderson and Fulcher, 2021), as feature vectors that represent the UTS (or the individual variables of the MTS, in the multivariate setting). The first is a set of time series statistical features that has been used in a variety of tasks such as clustering (Wang et al., 2006), forecasting (Kang et al., 2017; Talagala et al., 2018) and generation of time series data (Kang et al., 2020). It comprises sixteen features calculated using the `tsfeatures` package (Hyndman et al., 2020) of the R CRAN (R Core Team, 2020), namely, frequency and number and length of seasonal periods, the strength of the trend, "spikiness" of a time series, linearity and curvature, spectral entropy, and measures based on autocorrelation coefficients of the original series, first-differenced series, and second-differenced series. These will be denoted by *tsfeatures* in the remainder of this work. The second, is the denominated canonical feature set (Lubba et al., 2019) (we will be denoted by *catch22*), and has been recently proposed based on a features library from an interdisciplinary time series analysis literature (Fulcher and Jones, 2017) and has been used in time series classification tasks (Lubba et al., 2019). There are twenty two features calculated using the `Rcatch22`[1] package (Henderson,

[1] https://github.com/hendersontrent/Rcatch22

2021) of the R CRAN (R Core Team, 2020), that include properties of the distributions and simple temporal statistics of values in the time series, linear and nonlinear autocorrelation, successive differences, scaling of fluctuations, and others.

We start this chapter by describing the methodology used for the task of clustering time series using feature vectors, which is common to the presented univariate and multivariate clustering problems. Next, we perform an empirical study of the *NetF* features in the context of clustering the UDGP dataset and an empirical study of the *MNetF* for clustering the MDGP dataset. We analyzed the performance of the different combinations of the feature vectors from the WNVG, WHVG, and QG mappings to the *NetF* analysis and of the features vectors from MHVG and MQG to the *MNetF* analysis. We also analyzed the different combinations of feature vectors corresponding to the proposed approaches (*NetF* or *MNetF*) with the two sets of feature vectors previously proposed in the literature (conventional approach), to compare results. Finally, we present a clustering analysis on the benchmark datasets using the different (multilayer) network-based time series feature extraction strategies.

7.1 Clustering Methodology

The overall procedure proposed here for feature-based clustering is represented in Figure 7.1.

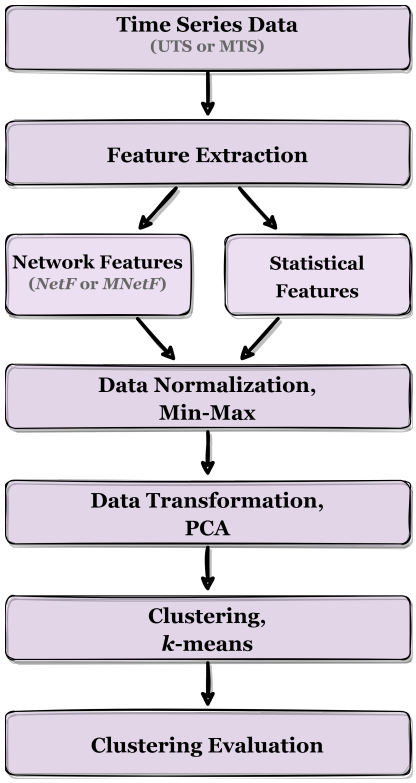


Figure 7.1: Schematic diagram for the (univariate or multivariate) time series clustering analysis procedure.

Given a set of UTS (or MTS) data, we compute the feature vectors (network-based features and statistical features) which are then Min-Max rescaled into the $[0, 1]$ interval and organized in a feature data matrix. Principal Components (PC) are computed (no need of z-score normalization within PCA) and finally a clustering algorithm is applied to the PC's corresponding to 100% of variance. Among several algorithms available for clustering analysis, we opt for *k-means* (Hartigan and Wong, 1979) since it is fast and widely used for clustering. Its main disadvantage is the need to pre-define the number of clusters. This issue will be discussed within each dataset example. The clustering results are assessed using appropriate evaluation metrics: *Average Silhouette* (AS); *Adjusted Rand Index* (ARI) and *Normalized Mutual Information* (NMI) when the ground truth is available. The AS does not need the ground truth, while the ARI and NMI do. The scale of the results for NMI is $[0, 1]$, while for ARI and AS is $[-1, 1]$. A higher value refers to a model with more coherent clusters (or identical partitions to the intrinsic structure), and a lower value refers to a model with less coherent clusters (or random partitions).

7.2 Clustering Synthetic Data

In this section, we perform an empirical study of Clustering of synthetic UTS and MTS datasets using *NetF* and *MNetF*, respectively. *NetF* is tested and compared with two other benchmark feature vectors in the UTS models (UDGP) presented in Table 5.1, and *MNetF* is tested and compared with *catch22* in the synthetic multivariate models (MDGP) presented in Tables 6.2.

7.2.1 UTS Clustering using *NetF*

In this section, we analyze the performance of different combinations of the feature vectors from the WNVG, WHVG, and QG mappings in a clustering of UDGP. We set the number of clusters to $k = 11$, the total of UTS models, and assess the clustering results with the evaluation features. The results are summarized in Table 7.1[2]. Comparing the first three rows of Table 7.1 with the last three rows we can see that joining the features obtained from the two mapping concepts (VGs and QGs) adds information that leads to improvements in the clustering results. In fact, as illustrated in Figure 7.2, clustering based on *NetF* can lead to a perfect attribution of the UTS model samples across the 11 different clusters.

[2]The results are means from 10 repetitions of the clustering analysis. The chosen clustering method (*k*-means) is a stochastic process, so 10 repetitions are adequate. The corresponding standard deviations indicate little or no variation between the repetitions.

Table 7.1: Clustering evaluation metrics for the different clustering analyses resulting from different network-based feature vectors. The values reflect the mean of 10 repetitions of the proposed method for different feature vectors and for the ground truth ($k = 11$). The highest values are highlighted.

Mappings	ARI $[-1,1]$	NMI $[0,1]$	AS $[-1,1]$
WNVG	0.68	0.86	0.51
WHVG	0.83	0.94	0.63
QG	0.64	0.84	0.66
WNVG - WHVG	0.81	0.93	0.57
WNVG - QG	0.84	0.94	0.67
WHVG - QG	0.90	0.96	**0.73**
NetF	**0.92**	**0.97**	0.68

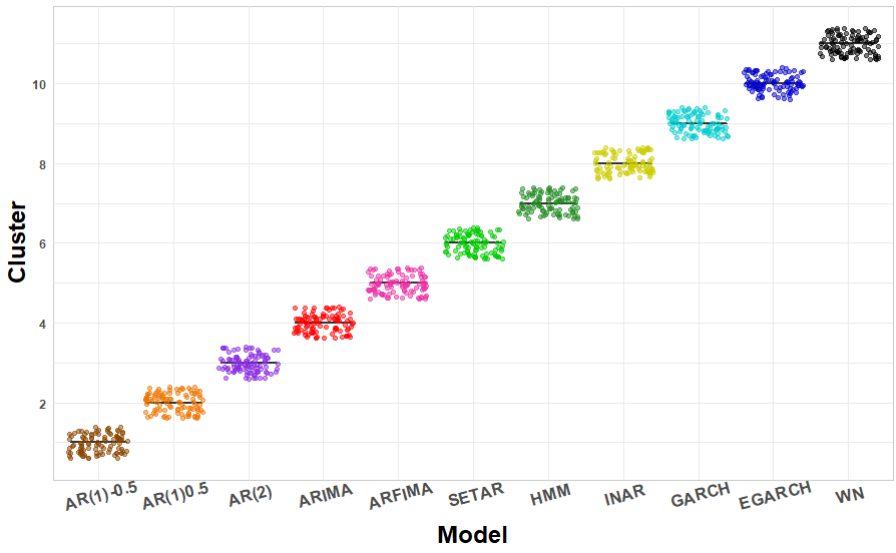


Figure 7.2: Attribution of the samples corresponding to instances of time series models to the different clusters, according to *NetF*. The different models are represented on the horizontal axis and by a unique color. The time series is represented by the colored points according to its model process. The vertical axis represents the cluster number to which a time series is assigned.

These results show that different mapping methods capture different properties from the series, as we analyzed in the previous Section 5.3, translating into a better clustering result of the widely diverse set of UTS models, as we expected. If we analyze only feature vectors corresponding to one network kind, the first three rows of Table 7.1, we note that the WHVGs are the ones that best capture the characteristics of time series models, having high evaluation values, namely, 0.83 for ARI, 0.94 for NMI and 0.63 for AS. The last three lines of Table 7.1 show better results than those obtained using only WHVG features, thus showing that the resulting features of the QGs add information about certain properties of the time series models.

7.2. Clustering Synthetic Data

We also study how these seven sets of features perform in determining the number of clusters k (knowing that the ground truth is $k = 11$) using the ARI, NMI, and AS evaluation metrics, as a reference for choosing the best k. The results for k obtained from features corresponding to only one kind of network, range from 8 to 13 for ARI, from 11 to 14 for NMI, and 3 to 9 for AS. However, when *NetF* is used, we obtain $k = 11$ for ARI and NMI and $k = 10$ for AS.

To validate the results presented here, we performed a comparative analysis with the two vectors of conventional features: *tsfeatures* and *catch22*. We present the clustering results for the ground truth in Table 7.2. The clustering and feature scaling process performed was the same for all experiments. We can verify that the best results fall on the feature vector that includes the topological features proposed in this work. The vector of statistical features *tsfeatures* also had great results, as expected, since it includes several features of conventional time series analysis, and the dataset under analysis is a dataset of systematic time series models. Note that the best result is observed for the vector resulting from the concatenation of *NetF* with *tsfeaures*. Apparently, the vector of *catch22* features makes the results worse.

Table 7.2: Clustering evaluation features for the different clustering analysis resulting from different feature vectors: *NetF*, *tsfeatures*, and *catch22*. The values reflect the mean of 10 repetitions of the proposed method for different feature vectors and for the ground truth ($k = 11$). The highest values are highlighted.

Mappings	**ARI** $[-1, 1]$	**NMI** $[0, 1]$	**AS** $[-1, 1]$
tsfeatures	0.76	0.91	**0.73**
catch22	0.40	0.66	0.39
tsfeatures - catch22	0.56	0.77	0.40
NetF - tsfeatures	**0.91**	**0.97**	**0.73**
NetF - catch22	0.76	0.87	0.43
NetF - tsfeatures - catch22	0.78	0.89	0.48
NetF	**0.92**	**0.97**	**0.68**

We also investigate the performance of *NetF*, *catch22* and *tsfeatures* in the automatic determination of the number of clusters k, using the clustering evaluation metrics, ARI, NMI and AS. For the proposed approach (*NetF*) the results of k were the ones mentioned above, for *tsfeatures* the best values obtained for k were 14 considering ARI and NMI and 9 considering AS; for *catch22* were 8, 13 and 2, for ARI, NMI and AS, respectively. Comparing the results, *NetF* and *tsfeatures* are the ones that best fit the known value of k, with *NetF* being the best, with *catch22* having the worst results, especially when we look at the AS feature as the feature that determines the value of k (which is commonly used). Figure 7.3 shows this results to AS feature evaluation.

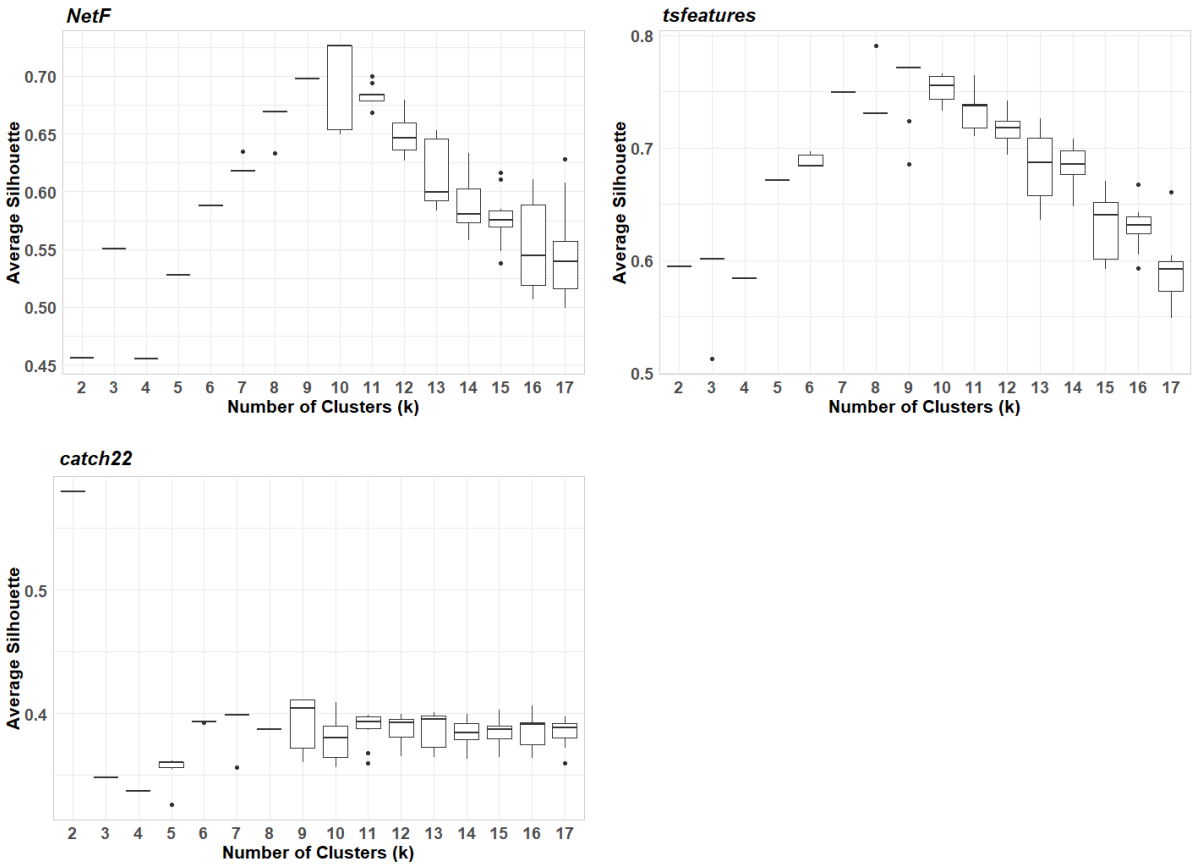


Figure 7.3: Number of clusters, k, for the UDGP, using the silhouette method for 10 repetitions of the clustering analysis using the 3 features vectors: *NetF*, *tsfeatures* and *catch22*. The ground truth for UDGP is $k = 11$.

7.2.2 MTS Clustering using *MNetF*

In this section, we present the clustering analysis results obtained from different feature vectors (from MHVG, MQG and *MNetF*) to the MDGP. We start by analyzing the results of the set of features corresponding to the MHVGs, then those corresponding to the MQGs and we finish with the results for the *MNetF*.

The clustering results obtained using the all 21 features from MHVG and from MQG introduced in Section 6.1 of all MDGPs are summarized, respectively, in Figures 7.4 and 7.5. For the synthetic dataset analyzed here, the MQG mapping obtained the best results in all evaluation features. As illustrated in Figure 7.4, the different evaluation features indicate a different number of clusters for the dataset: ARI indicates $k = 5$ followed by $k = 6$, which is the ground truth value of the MDGP dataset, while the NMI indicates either $k = 6$ or $k = 7$. The AS, using the silhouette method to assess the quality of the clusters indicates $k = 3$. The three clusters are constituted by: the `sVAR` models are one cluster (very strong auto and cross-correlations), the GARCH models are another (heteroskedastic data), and the WN and `wVAR` models are the last cluster (white noise and weakly correlated data), indicating that the MDGPs were clustered

according to correlation (serial and cross) and volatility properties. For the MQG features vector, the results are consistent to ARI and NMI, obtaining $k = 6$ with near-maximum values. While for AS the maximum value of 0.70 indicates $k = 2$ (that is not a good result), but obtain 0.6 for $k = 5$ (that is a great result for the AS feature). The latter joins in the same cluster the `wVAR` and `iWN` models that are dynamically more similar.

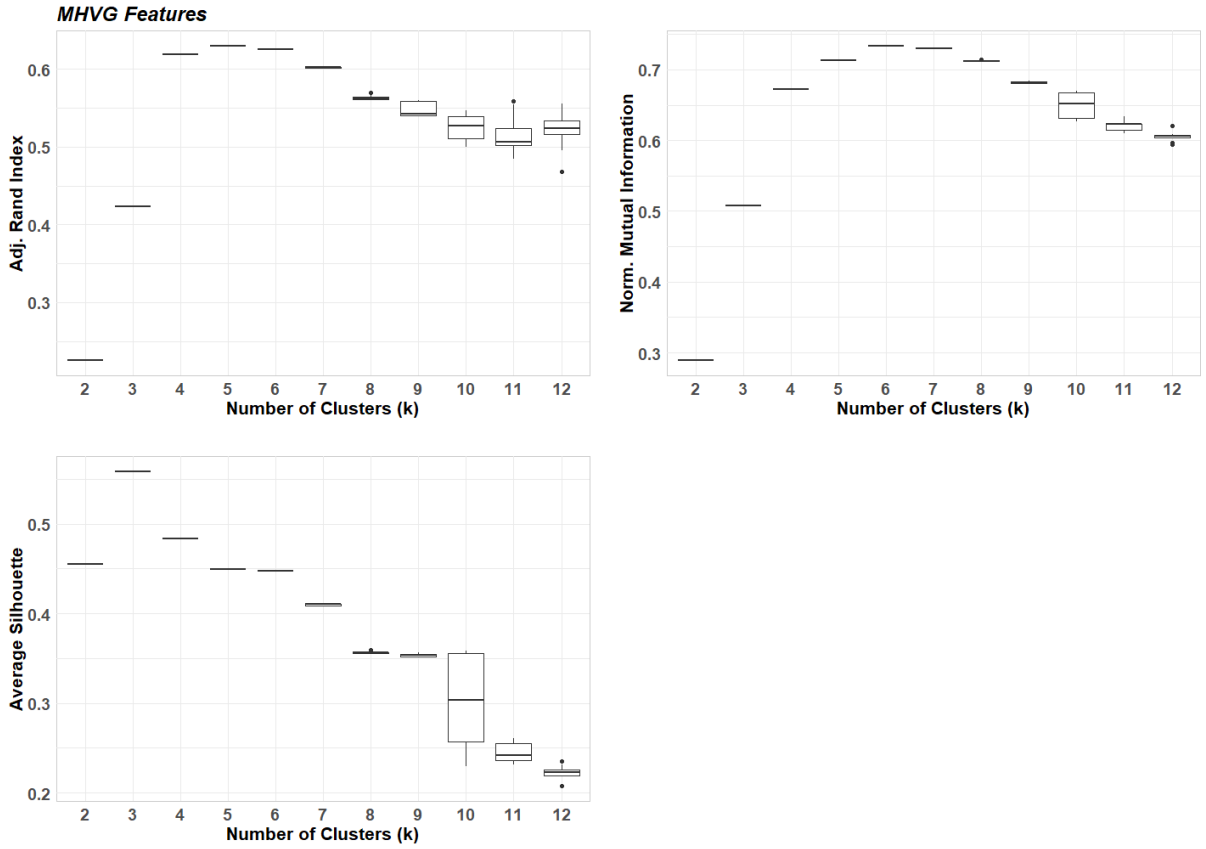


Figure 7.4: Evaluation of clustering results from MHVG features set for the MDGP. The results represent the 10 repetitions of the clustering analysis. The ground truth for MDGP is $k = 6$.

Figure 7.6 represents the results from clustering the 100 instances of the 6 MDGPs, using the 21 MNet features (from MHVG and from MQG) and the k-means algorithm with $k = 6$. The result from MHVG features presents a perfect attribution of `cBWN` and `sVAR` samples across two different clusters (clusters 1 and 3), the attribution of `iBWN` and `wVAR` samples across the same cluster (cluster 2), and a homogeneous attribution of GARCH samples (`wGARCH` and `sGARCH`) across two clusters (4 and 5), and some samples of `cBWN` and `sGARCH` across cluster 6 since $k = 6$ and the GARCH samples are the most disparate in the feature space. While from MQG features the MTS samples are (almost) perfectly attributed by the 6 different clusters, only a few samples of the weakly or no cross-correlated models (`iWN`, `wVAR` and `wGARCH`) are assigned to neighboring clusters.

Performing the clustering exercise considering subsets of the MNet feature set. The results summarized in Table 7.3 indicate that inter-layer edges contain, in fact, information about the MTS, leading to better clustering results (together with the intra-layer edges). We can also analyze that the relational features, which include the new topological features of MNet proposed in this work (average ratio degree), achieve good clustering results when considered alone.

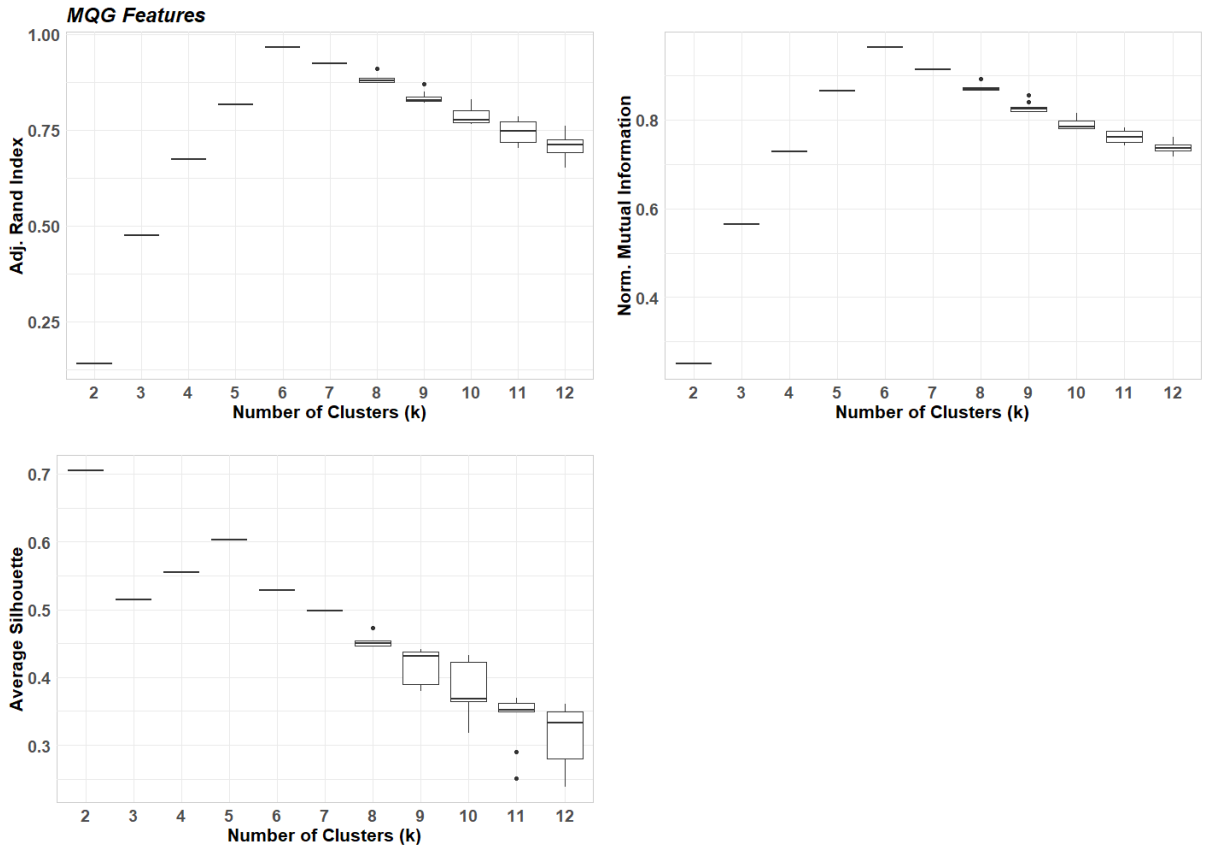
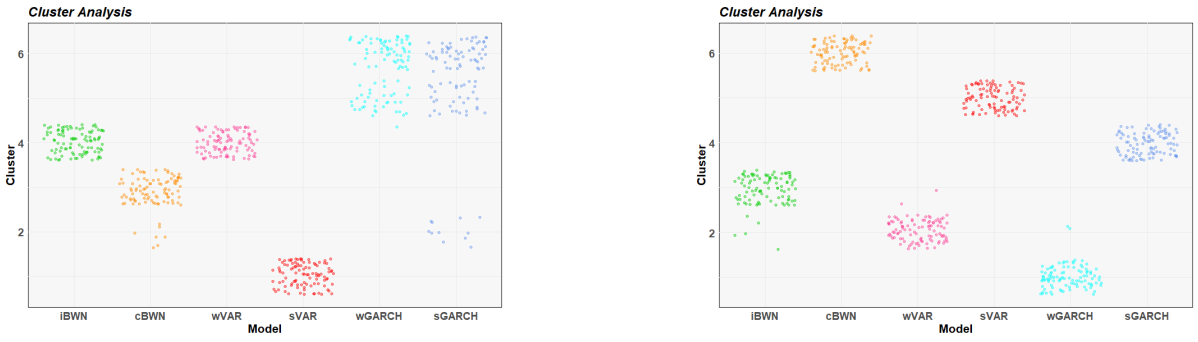


Figure 7.5: Evaluation of clustering results from MQG features set for the MDGP. The results represent the 10 repetitions of the clustering analysis. The ground truth for MDGP is $k = 6$.

Subgraphs with both intra-layer edges and inter-layer edges add information that leads to improvements in the clustering results (compare the last three rows of the Table 7.3 with the first two). The results show that cross-HVG and the contemporaneous quantile graphs (the inter-layer edges from MHVG and MQG, respectively) capture different properties from MTS data, that, as we expected, translate into better clustering results. Also note that the results of ARI and NMI from the set of intra-layer features are good results, because the MDGP under analysis involves the same statistical process for the two components of time series whose properties inherent to each process are also captured by the HVG and QG mapping methods.



(a) MHVG

(b) MQG

Figure 7.6: Attribution of the samples corresponding to instances of MTS models to the different clusters, according to MHVG and MQG feature sets. The different models are represented on the horizontal axis and by a unique color. The bivariate time series samples are represented by the colored points according to their model process. The vertical axis represents the cluster number to which a bivariate time series samples are assigned.

Table 7.3: Clustering evaluation metrics for the different clustering analyses resulting from different MNet feature vectors of MHVG and MQG. The values reflect the mean of 10 repetitions of the proposed method for different feature vectors and for the ground truth ($k = 6$). The highest values are highlighted.

Feature Set	**ARI** $[-1,1]$		**NMI** $[0,1]$		**AS** $[-1,1]$	
	MHVG	**MQG**	**MHVG**	**MQG**	**MHVG**	**MQG**
***Intra*-layer**	0.52	0.52	0.61	0.65	0.29	0.52
***Inter*-layer**	0.29	0.57	0.42	0.68	0.51	**0.83**
***All*-layer**	**0.67**	**0.78**	**0.72**	**0.86**	**0.50**	**0.78**
Relational	0.57	0.71	0.65	0.81	**0.62**	0.55
MNet	**0.63**	**0.96**	**0.71**	**0.96**	0.45	0.53
MNetF	**0.73**		**0.86**		**0.49**	

To finalize our empirical analysis, we apply the *MNetF* in the context of clustering the MDGP. We set the number of clusters to $k = 6$ and we obtain the following conclusion (see last two rows of the Table 7.3): the topological features from MQGs is more accurate, almost perfect when we look at the evaluation features ARI and NMI, with mean value of 0.96, and AS with 0.53, when compared to cross-visibility-based mappings and of *MNetF* that obtain, respectively, 0.63 and 0.73 to ARI feature, 0.71 and 0.86, to NMI feature, and 0.45 and 0.49 to AS. For the dataset studied here (MDGP), the MQG is sufficient to group the different MTS model samples. We can also say that the topological features of MHVG worsen the clustering accuracy when joined with MQG. However, for other datasets, they may be more valuable, such as multidimensional data with periodic characteristics that tend to be well captured by visibility concepts (Silva et al., 2022). We will analyze this in the next section.

7.3 Clustering Benchmark Data

In this section, we perform a Clustering analysis of real-world and benchmark UTS and MTS datasets using the proposed approaches: *NetF* and *MNetF*, respectively. *NetF* is tested and compared in the following univariate datasets: the benchmark empirical UTS datasets from UEA & UCR Time Series Classification Repository (Bagnall et al.), widely used in classification tasks; the M3 competition data from package `Mcomp` (Hyndman, 2018) used for testing the performance of forecasting algorithms; the set "18Pairs" extracted from package `TSclust` (Montero and Vilar, 2014) which represents pairs of time series of different domains; and a new dataset regarding the production of several crops across Brazil (IBGE). *MNetF* is tested and compared in the set of benchmark empirical MTS datasets from UEA & UCR Time Series Classification Repository (Bagnall et al.).

7.3.1 UTS Clustering using *NetF*

In this section, we analyze the performance of the proposed approach to extract features from benchmark UTS datasets via single-layer networks. In the following, we present only some of these datasets (for the sake of simplicity) which are described in Table 7.4, but in Appendix D a brief description and clustering results for the remaining datasets are presented in Tables D.2-D.6. For the benchmark UTS datasets, we assume the labels/classes associated with each dataset to be the ground truth for the clustering, so the clustering evaluation measures ARI and NMI can be used. Additionally, we also analyze a set of observations comprising the production over forty three years of nine agriculture products in 108 meso-regions of Brazil (IBGE). We note that the size of the ElectricDevices dataset, 16575 time series, is different from the total available in the repository, as exactly 62 time series return missing values for the entropy feature of the *tsfeatures* set and so we decided to exclude these series from our analysis.

First, we investigate the performance of *NetF*, *catch22* and *tsfeatures* in the automatic determination of the number of clusters k, using the clustering evaluation metrics, ARI, NMI and AS. The results (see Table D.1) show overall similar values but we note that *NetF* seems to provide a value of k equal to or closer to the ground truth value (when available) more often. For the Production in Brazil data, for which there is no ground truth, values for k are obtained by averaging 10 repetitions of the clustering procedure and using the silhouette method. The results of the 10 repetitions are represented in Figure D.1 and summarized in Table 7.6.

Next, fixing k to the ground truth we perform the clustering procedure. The clustering evaluation metrics, mean over 10 repetitions, are presented in Table 7.5 [3].

[3]The results for the remaining empirical time series datasets of the UEA & UCR Time Series Classification Repository are presented in Tables D.2 to D.4.

Table 7.4: Brief description of the empirical univariate time series datasets.

Dataset	Dataset Size	Time Series Length	Num. of Classes	Source
18Pairs	36	1000	18	Montero and Vilar (2014)
M3 data	3003	[20, 144]	6	Hyndman (2018)
CinC_ECG_torso	1420	1639	4	Bagnall et al.
Cricket_X	780	300	12	Bagnall et al.
ECG5000	5000	140	5	Bagnall et al.
ElectricDevices	16575	96	7	Bagnall et al.
FaceAll	2250	131	14	Bagnall et al.
FordA	4921	500	2	Bagnall et al.
InsectWingbeatSound	2200	256	11	Bagnall et al.
UWaveGestureLibraryAll	4478	945	8	Bagnall et al.
Production in Brazil	108	198	9	IBGE

Table 7.5: Clustering evaluation metrics obtained for the three approaches *NetF*, *tsfeatures* and *catch22*. The values reflect the mean of 10 repetitions of the clustering analysis with number of classes equal to the ground truth (see Table B.2). The values in bold represent the best results.

Dataset	ARI $[-1,1]$			NMI $[0,1]$			AS $[-1,1]$		
	tsf.	*cat.*	*NetF*	*tsf.*	*cat.*	*NetF*	*tsf.*	*cat.*	*NetF*
18Pairs	**0.51**	0.39	0.49	**0.89**	0.86	**0.89**	**0.42**	0.32	0.34
M3 data	**0.14**	0.13	0.13	**0.21**	0.19	0.18	**0.36**	0.22	0.31
CinC_ECG_tors	0.31	0.32	**0.45**	0.37	0.35	**0.52**	0.23	0.19	**0.31**
Cricket_X	0.15	0.15	**0.16**	**0.32**	0.28	0.30	**0.20**	0.16	0.10
ECG5000	0.29	0.28	**0.31**	**0.32**	0.29	0.30	**0.24**	**0.24**	0.16
ElectricDevices	0.20	**0.21**	0.19	**0.30**	0.29	0.29	**0.33**	0.25	0.27
FaceAll	0.15	**0.21**	0.15	0.33	**0.36**	0.29	**0.22**	0.15	0.09
FordA	**0.19**	0.01	0.01	**0.27**	0.01	0.01	**0.53**	0.33	0.29
InsectWingbeat	0.07	**0.21**	0.17	0.18	**0.37**	0.32	**0.19**	0.18	0.11
UWaveGesture	0.17	**0.2**	0.18	0.27	**0.28**	**0.28**	**0.2**	0.19	0.12
Synthetic (UDGP)	0.76	0.40	**0.92**	0.91	0.66	**0.97**	**0.73**	0.39	0.68
Production Brazil	0.09	0.18	**0.30**	0.40	0.55	**0.70**	0.46	0.39	**0.61**

The results indicate that none of the three approaches performs uniformly better than the others. Some interesting comments follow. For the synthetic datasets and 18Pairs, *tsfeatures* and *NetF* perform better than *catch22* in all evaluation criteria. The clusters for ECG5000, ElectricDevices, and UWaveGestureLibraryAll datasets produced by the three approaches fare equally well when assessed by ARI, NM, and AS. The same is true for M3 data and Cricket_X datasets, with slightly lower results. *NetF* approach seems to produce better clusters for CinC_ECG_torso measured according to the three criteria, the *tsfeatures* seems to produce better clusters for FordA, and the *catch22* for FaceAll and InsectWingbeat measured according to the ARI and NMI.

Analyzing the overall results, Tables 7.5, D.2-D.6 we can state that *tsfeatures* and *NetF* approaches present the best ARI and NMI evaluation metrics, while *tsfeatures* achieves by far the best results in the AS. If we consider the UEA & UCR repository classification of the datasets, we note the following: the *NetF* approach presents good results for time series data of the type Image (BeetleFly, FaceFour, MixedShapesRegularTrain, OSULeaf and Symbols), ECG (CinC_ECG_tors and TwoLeadECG) and Sensor (Wafer); the *tsfeatures* performs best for types Simulated (BME, UMD and TwoPatterns), ECG (NonInvasiveFetalECGTho), Image (ShapesAll) and Sensor (SonyAIBORobotSurface and Trace); finally the *catch22* approach presents best results for Spectro (Coffee), Device (HouseTwenty) and Simulated (ShapeletSim) types. In summary *NetF* and *tsfeatures* perform better in data with the same characteristics while *catch22* seems to be more appropriate to capture other characteristics.

Regarding the dataset Production in Brazil, Table 7.6 shows more detail on the clustering results, adding the value k to indicate the number of clusters that was automatically computed. We note that the 4 clusters obtained with *NetF* correspond to 4 types of goods: eggs; energy; gasoline and cattle; hypermarkets, textile, furniture, vehicles and food. Figure 7.7 shows the clustering results for each of the three approaches used. Note that both *tsfeatures* and *catch22* put eggs and textile production in the same cluster, and *tsfeatures* cannot distinguish energy. The *NetF* approach also resulted in the highest AS value, which means that, on average, the consistency of the samples within clusters of data formed using *NetF* is better than that of the clusters resulting from the *tsfeatures* and *catch22* approaches. To illustrate the relevance of the results, Figure 7.8 depicts a representative time series for each cluster.

Table 7.6: Clustering evaluation metrics for the different clustering analysis on Production in Brazil dataset based on *Netf*, *tsfeatures* and *catch22* approaches. The values reflect the mean of 10 repetitions of the proposed method for different feature vectors and for the number of clusters detected according to the average silhouette metric.

Approach	k	**ARI** $[-1,1]$	**NMI** $[0,1]$	**AS** $[-1,1]$
tsfeatures	2	0.09	0.40	0.46
catch22	3	0.18	0.55	0.39
NetF	4	**0.30**	**0.70**	**0.61**

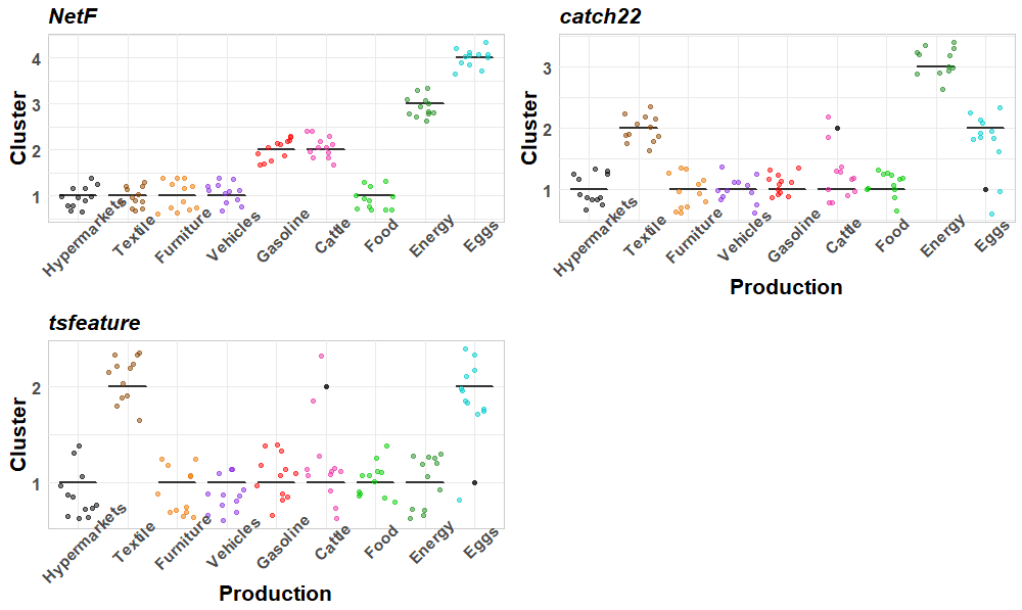


Figure 7.7: Attribution of the Production in Brazil time series to the different clusters, according to each of the feature approaches. The different productions are represented on the horizontal axis and by a unique color. The time series are represented by the colored points according to its production type. The vertical axis represents the cluster number to which a time series is assigned.

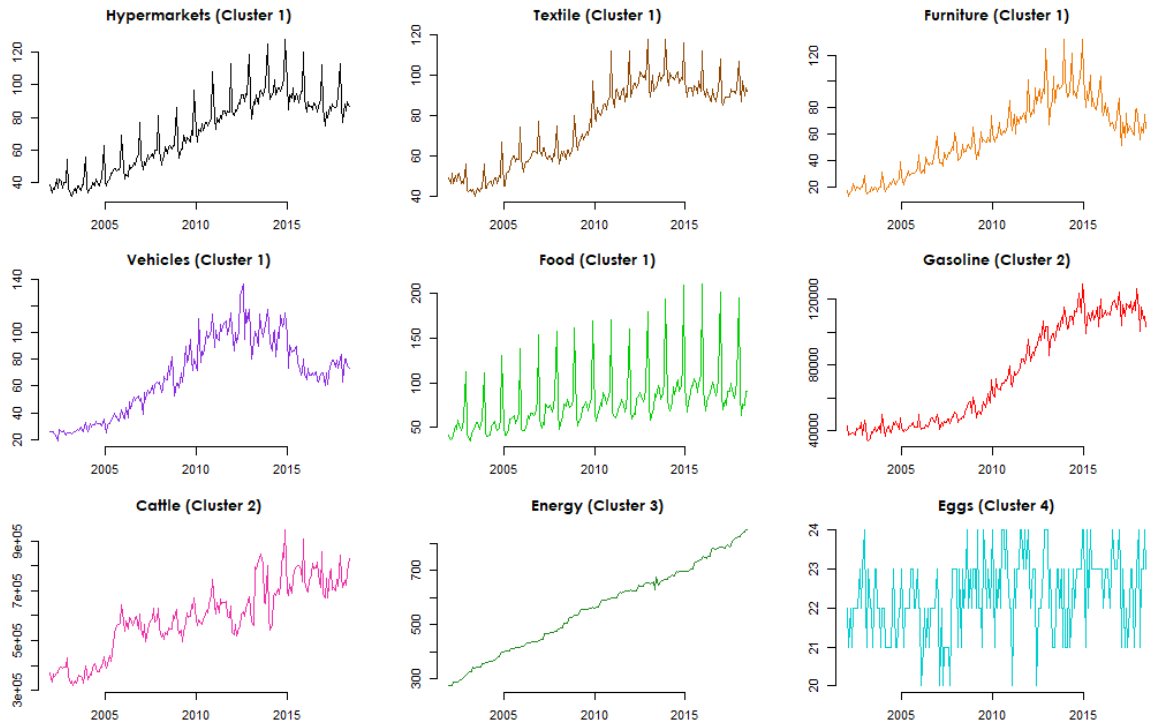


Figure 7.8: Production in Brazil representative of each cluster (indicated in subtitle) obtained using the proposed approach, *NetF*.

7.3.2 MTS Clustering using *MNetF*

In this section, we analyze the performance of the proposed approach to extract features from benchmark MTS via MNets. And we also compare the results with the *catch22* feature vector. In the multivariate setting analyzed in the section, we extract these statistical features, individually, from the time series components. Note that we do not compare with the statistics feature set *tsfeatures*, as it is not computable in most of the datasets, leading to the need for data preprocessing and selection of computable features which is not the purpose of this work.

The datasets in Table 7.7 are MTS data, belonging to the UEA & UCR Time Series Classification Repository (Bagnall et al.). We summarize the description of the dataset, the size of the dataset, its length, the number of classes, and the number of dimensions/components UTS. For the benchmark MTS datasets, we assume the labels/classes associated with each dataset to be the ground truth for the clustering, so the clustering evaluation measures ARI and NMI can be used. The UEA & UCR Time Series Classification Repository has a total of 30 MTS datasets available. However, in this work we present only a subset of these datasets for the reasons that we describe below. Some datasets do not have the same length, making it difficult to compute the *catch22* features since, currently, the R software implementations for *catch22* and their respective auxiliary utility packages are not designed to receive datasets with different lengths as input. Furthermore, the high dimensionality of some datasets made it difficult to compute the *catch22* and MHVG feature vectors, while, as expected, the MQG feature vector was possible to compute. The high dimensionality also led to some memory-related problems when computing the PC's, which were also calculated using R software. For the sake of simplicity and consistency of comparison between all feature vectors (MHVG, MQG, *MNetF* and *catch22*), we decided not to present the datasets with the mentioned problems, since it does not impair the purpose of this work.

For all real-world datasets, we investigated the different feature sets proposed in this work (features from MHVG, MQG, and *MNetF*). Table 7.8 presents the clustering results (using k as the ground truth corresponding to each dataset) for the three clustering evaluation features used in this work. The ARI and NMI measures shows better results of in most cases for the approach based on *MNetF* (9 of the 19 datasets), while the AS measure shows better results for the MHVG feature set (11 of the 19 datasets).

As we expected, some of the datasets are better using the proposed *MNetF* set than just using the MQG feature set, one particular example is the ERing dataset (which describes finger and hand gestures) with only 65 observations on each time series component, which obtains an ARI value of 0.28 for MQG and 0.51 and 0.65 for MHVG and *MNetF*, respectively. The results suggest that the MQG (isolated) are descriptively weaker than the *MNetF* in a dataset with a larger dimension. The datasets in which we obtain higher values of ARI and NMI are datasets of type HAR (Human Activity Recognition). And the datasets with the highest AS value are ECG (eletrocardiograma) and EEG (electroencephalogram) type datasets.

Table 7.7: Brief description of the empirical multivariate time series datasets.

Dataset	Dataset Size	TS Length	Num. of Dimen.	Num. of Classes	Source
Synthetic MDGP (0)	600	10000	2	6	-
PenDigits (1)	10992	8	2	10	Bagnall et al.
LSST (2)	4925	36	6	14	Bagnall et al.
Handwriting (3)	1000	152	3	26	Bagnall et al.
ArticularyWordRecognition (4)	575	144	9	25	Bagnall et al.
SelfRegulationSCP1 (5)	561	896	6	2	Bagnall et al.
EthanolConcentration (6)	524	1751	3	4	Bagnall et al.
FingerMovements (7)	416	50	28	2	Bagnall et al.
SelfRegulationSCP2 (8)	380	1152	7	2	Bagnall et al.
NATOPS (9)	360	51	24	6	Bagnall et al.
Libras (10)	360	45	2	15	Bagnall et al.
RacketSports (11)	303	30	6	4	Bagnall et al.
ERing (12)	300	65	4	6	Bagnall et al.
Epilepsy (13)	275	206	3	4	Bagnall et al.
HandMovementDirection (14)	234	400	10	4	Bagnall et al.
Cricket (15)	180	1197	6	12	Bagnall et al.
BasicMotions (16)	80	100	6	4	Bagnall et al.
AtrialFibrillation (17)	30	640	2	3	Bagnall et al.
StandWalkJump (18)	27	2500	4	3	Bagnall et al.

Then, we perform the clustering procedure for the conventional approach where we use the vector of features corresponding to *catch22* (for each sample of an MTS dataset, we compute *catch22* individually for each of the time series components). We also analyzed the advantage of using *MNetF* over this type of time series problem, joining the features vector of *MNetF* to the vector of *catch22*, in order to verify if the topological features add valuable information to the features already existing in the literature. The results are presented in Table 7.9. The results are balanced for the ARI and NMI features, while for AS the clusters seem to be more defined for the *catch22* feature set. Some datasets seem to get better clustering results when we combine both *MNetF* and *catch22* vectors, these results are directed towards small datasets, i.e. with small time series lengths. This suggests that topological features can be useful for MTS with few observations, which in fact is one of the major problems in conventional time series analysis. The statistical models need a good set of observations to better model time series data, which leads to performance losses on small datasets.

Table 7.8: Evaluation metrics of the resulting multivariate times series clustering from different features vectors, namely, features from *MHVGs*, features from *MQGs*, and features from *MNetF*. The values reflect the mean of 10 repetitions of the clustering analysis with the number of classes equal to the ground truth (see Table 7.7). The values in bold represent the best results.

Dataset	**ARI** $[-1,1]$			**NMI** $[0,1]$			**AS** $[-1,1]$		
	MHVG	*MQG*	*MNetF*	*MHVG*	*MQG*	*MNetF*	*MHVG*	*MQG*	*MNetF*
(0)	0.63	**0.96**	0.73	0.71	**0.96**	0.86	0.45	**0.53**	0.49
(1)	**0.21**	0.04	0.19	**0.35**	0.08	0.29	**0.23**	0.18	0.12
(2)	**0.06**	0.04	**0.06**	0.16	0.12	**0.17**	**0.04**	**0.04**	0.03
(3)	0.02	0.02	**0.03**	0.17	0.18	**0.19**	0.06	**0.08**	0.05
(4)	0.60	0.28	**0.65**	0.79	0.57	**0.81**	**0.09**	0.06	0.08
(5)	0.00	**0.04**	0.03	0.00	**0.03**	0.02	**0.29**	0.24	0.15
(6)	0.00	0.00	0.00	0.00	**0.01**	0.00	**0.10**	0.08	0.07
(7)	0.00	0.00	0.00	0.00	0.00	0.00	**0.09**	**0.09**	0.07
(8)	0.00	0.00	0.00	0.00	0.00	0.00	0.22	**0.24**	0.15
(9)	0.29	0.43	**0.44**	0.42	0.51	**0.54**	0.04	**0.05**	**0.05**
(10)	0.35	0.26	**0.42**	0.59	0.52	**0.65**	0.17	**0.19**	0.17
(11)	0.07	**0.08**	**0.08**	0.12	0.14	**0.17**	**0.05**	0.04	0.04
(12)	0.51	0.28	**0.65**	0.62	0.39	**0.71**	**0.14**	0.09	0.11
(13)	0.40	0.38	**0.42**	0.40	0.40	**0.43**	0.11	**0.13**	0.10
(14)	**0.01**	-0.01	0.00	**0.03**	0.01	0.01	0.03	**0.05**	0.03
(15)	**0.80**	0.51	0.72	**0.89**	0.70	0.85	**0.22**	0.16	0.21
(16)	0.82	0.79	**0.83**	0.82	0.79	**0.87**	**0.14**	0.09	0.11
(17)	-0.04	**0.01**	-0.03	0.05	**0.06**	0.05	0.22	**0.26**	0.18
(18)	**0.07**	0.03	0.02	**0.14**	**0.14**	0.12	**0.19**	0.16	0.17
Mean	0.23	0.18	**0.25**	0.31	0.26	**0.33**	**0.14**	0.12	0.11
Win	5	4	**9**	4	5	**9**	**11**	10	1
Win (%)	26.32	21.05	**47.37**	21.05	26.32	**47.37**	**57.89**	52.63	5.26

Table 7.9: Evaluation metrics of the resulting multivariate times series clustering from different approaches, namely, *MNetF* (proposed approach) and *catch22* (conventional approach). The values reflect the mean of 10 repetitions of the clustering analysis with the number of classes equal to the ground truth (see Table 7.7). The values in bold represent the best results.

Dataset	**ARI** $[-1,1]$			**NMI** $[0,1]$			**AS** $[-1,1]$		
	MNetF	*catch*	*MNetF-catch*	*MNetF*	*catch*	*MNetF-catch*	*MNetF*	*catch*	*MNetF-catch*
(0)	**0.73**	0.28	0.67	**0.86**	0.37	0.77	**0.49**	0.29	0.29
(1)	0.19	0.31	**0.41**	0.29	0.44	**0.50**	0.12	**0.16**	0.12
(2)	**0.06**	0.05	0.05	**0.17**	0.14	0.14	0.03	**0.12**	0.10
(3)	0.03	0.04	**0.06**	0.19	0.23	**0.26**	0.05	**0.08**	0.05
(4)	0.65	**0.88**	0.76	0.81	**0.94**	0.88	0.08	**0.12**	0.09
(5)	0.03	**0.05**	0.03	0.02	**0.04**	0.02	0.15	**0.18**	0.15
(6)	0.00	0.00	0.00	0.00	0.00	0.00	0.07	**0.23**	0.10
(7)	0.00	0.00	0.00	0.00	0.00	0.00	0.07	**0.12**	0.07
(8)	0.00	0.00	0.00	0.00	0.00	0.00	0.15	**0.24**	0.15
(9)	**0.44**	0.28	**0.44**	**0.54**	0.46	**0.54**	0.05	**0.06**	0.05
(10)	0.42	0.44	**0.54**	0.65	0.67	**0.74**	0.17	**0.18**	0.17
(11)	0.08	0.25	**0.30**	0.17	0.34	**0.38**	0.04	**0.08**	0.04
(12)	0.65	0.54	**0.92**	0.71	0.60	**0.92**	0.11	0.11	**0.12**
(13)	**0.42**	0.33	0.36	**0.43**	0.39	0.41	0.10	**0.14**	0.11
(14)	0.00	0.00	0.00	0.01	0.01	0.01	0.03	**0.05**	0.03
(15)	0.72	**0.92**	0.85	0.85	**0.95**	0.92	0.21	**0.26**	0.22
(16)	0.83	**0.97**	0.82	0.87	**0.96**	0.83	0.11	**0.24**	0.15
(17)	-0.03	-0.03	-0.03	**0.05**	0.03	0.04	**0.18**	0.15	0.16
(18)	**0.02**	-0.02	-0.05	**0.12**	0.06	0.03	0.17	**0.18**	0.11
Mean	0.28	0.30	**0.32**	0.35	0.35	**0.39**	**0.16**	**0.16**	0.12
Win	5	4	**6**	6	5	**6**	2	**16**	1
Win (%)	26.31	21.05	**31.58**	31.58	26.31	**31.58**	10.53	**84.21**	5.26

7.4 Conclusions

In this chapter, we presented the results obtained for univariate time series clustering and multivariate time series clustering using the *NetF* and *MNetF* feature vectors, respectively. We also compared the results with conventional feature vectors from time series analysis.

The results presented in this section show that *NetF* is capable of capturing information regarding specific properties of UTS data. And the clustering results show that using the whole feature set (from WNVG, WHVG, and QG) leads to better performance than any possible subset. This showcases how the different features complement each other and how they capture different characteristics of the underlying time series. This is even more evident when we compare the results with more conventional feature vectors.

To further showcase the applicability of *NetF*, we use its feature set for clustering both the previously mentioned synthetic data, as well as a large set of benchmark empirical UTS datasets. The results for the datasets in which ground truth is available indicate that *NetF* yields the highest mean for ARI (0.287) compared to alternative time series features, namely *tsfeatures* and *catch22*, with means of 0.267 and 0.228, respectively. For the NMI metric the results are similar (0.395, 0.397 and 0.358, respectively) and for AS the highest mean was found for *tsfeatures*, 0.332 versus approximately 0.3 for the others. However, the higher values for AS[4] must be viewed in light of the low values of ARI and NMI which indicate an imperfect formation of the clusters. For the production data in Brazil, for which no ground truth is available, *NetF* produces clusters that group production series with different characteristics, namely, time series of counts, marked upward trend, series in the same range of values, and with the seasonal component.

NetF is capable of grouping time series of different domains, such as data from ECGs, images and sensors, as well as identifying different characteristics of the time series using different mapping concepts, which stand out in different topological features. The general characteristics of the data, namely, the size of the dataset, the length of the time series and the number of clusters, do not seem to be influencing the results obtained.

We verified that not all test cases present the best results for *NetF*. However, this is not our purpose. First, *NetF* does not comprise a complete set of topological features, it represents a restricted set of possible graph topological features. And second, we want to show that the use of features from networks can help in time series mining tasks, such as clustering, being an additional feature vector that can complement those already existing in the literature.

[4] samples are very similar inside the cluster and show little similarity inter-cluster

In this chapter, we also conclude that the introduction of inter-layer edges to MTS mapping methods can capture information that is lost during MTS mappings to multiplex networks or single-layer networks. Although alone (considering only the inter-layer edges) do not obtain the best results (for example in MTS clustering tasks), when combined with intra-layer edge structures, the topological features of networks are enriched. Relational topological features also seem to have an important impact on the description of multivariate data, capturing cross-dependencies and similarities between time series components.

Through the results presented, we can conclude that the different sets of features (statistics and topology) used in this work describe different properties of the MTS data, which complement each other in certain types of data. The results also show that the proposed mapping methods can be useful to provide new insights and provide new methodologies for analyzing multivariate data, mainly in contexts of short datasets. Through the datasets used, it is perceived that the topological features seem to be good for analyzing data extracted from human activity recognition methodologies, which are increasingly one of the main sources of data for analysis in a variety of real-world problems.

Statistical features, which are often based on statistical models and methods designed for certain parameters, often become impractical. The approach proposed in this work tries to overcome this limitation, in fact, the topological features of networks used are always computable for any type of dataset and do not need data preprocessing.

Chapter 8

tsmnet: a Framework for Time Series and Feature Extraction

In the course of this work, we have been developing a complete framework, which we call ***tsmnet***, which supports the methodologies and methods proposed in this document. *tsmnet* is an object-oriented application framework written entirely in C++ with the main goal of providing methods for mapping large datasets of univariate and multivariate time series into general and complex structures of multilayer networks as well as methods for extracting topological features from these networks. It implements mapping functions based on visibility graphs and mapping functions of quantile graphs. Both are adapted for univariate and multivariate temporal data and are adapted to run in parallel when mapping is done for large datasets. It also implements several topological features of multilayer networks that are adapted to the substructures of the MNets presented in this work and implements other useful functions. All functions are fully implemented, but there is also a module that allows integration with the `igraph` network library for C, in order to allow the use of other network methods not implemented in the core of *tsmnet*. Thus, another goal of *tsmnet* is to put together useful methods from other packages/frameworks so as not to limit network methods. The *tsmnet* allows us to have complete control of the underlying structures and methods as well as full development freedom.

As we saw in Section 3.3 there are not many software packages available for mapping time series to networks, mainly for multilayer networks. In fact, as far as we know, no other package software is adapted to the more general structure of a MNet. The *tsmnet* is and embodies this structure. In this way, *tsmnet* can be very promising. And although it still only contains mappings based on visibility graphs and quantile graphs, the objective is to implement other mapping methods. *tsmnet* is available at: https://github.com/vanessa-silva/tsmnet.

In this chapter, we present the main functions and methods of *tsmnet*.

8.1 *tsmnet* Definition

In general, *tsmnet* has inheritance and composition hierarchy, as we can see in Figure 8.1 through the dependency graph which reflects relationships between the main class modules. *tsmnet* is divided into the following modules:

components Data structures for multilayer networks, able to handle general multilayer networks structures (Kivelä et al., 2014), that allows any inter-layer edges type, multiple layers, and multiple aspects.

mappings Mapping methods to transform a multivariate time series into multilayer visibility networks and multilayer quantile networks.

metrics Topological multilayer network features (local and global features), adapted to incorporate both intra-layer and inter-layer components.

utils Useful functions, such as prints, adjacency matrix structure, and transformation multilayer network functions.

io Functions to read multivariate time series datasets and functions to write multilayer networks data.

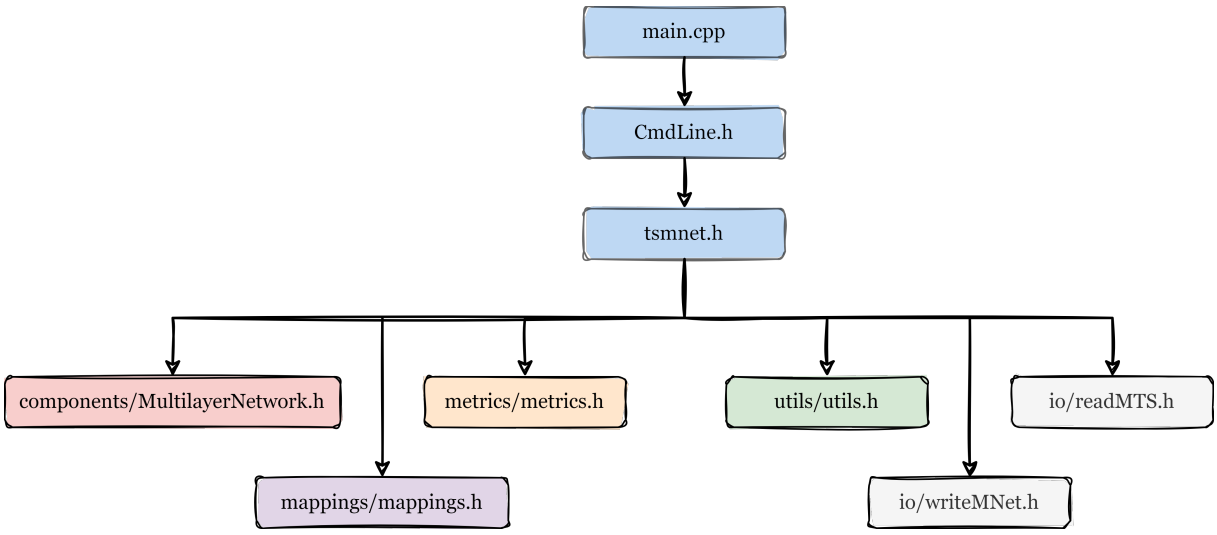


Figure 8.1: Dependency graph of *tsmnet* framework.

In order to better understand the workflow for which *tsmnet* is designed, we present the respective flowchart in Figure 8.2. *tsmnet* receives as input one (or several) sets of temporal data (MTS or UTS), which are read and stored in a data structure. Following is the mapping of the dataset into multilayer networks, as described in Sections 4.1 and 4.2, the multiplex networks version can also be selected. The mapping involves two steps, mapping each individual UTS and mapping the inter-layer edges from pairs of UTS components. After the creation of MNet,

several functionalities can be implemented. Perform an aggregation approach of MNet into a single-layer network and then compute topological features from the transformed network. Compute topological features on the intra and inter-layer MNet structures. These features can be used to create relational (single-layer) networks, where the nodes represent the layers (UTS components) of the MNet and the directed edges represent relationships between layers based on the computed features, next we can compute new features from the created networks. After computing the topological features, it is possible to extract them to a file to be used in other data analysis tools. Finally, MNet, its possible transformations, and its substructures can be written to be a specific format in order to allow the use of external tools, such as tools for network visualization.

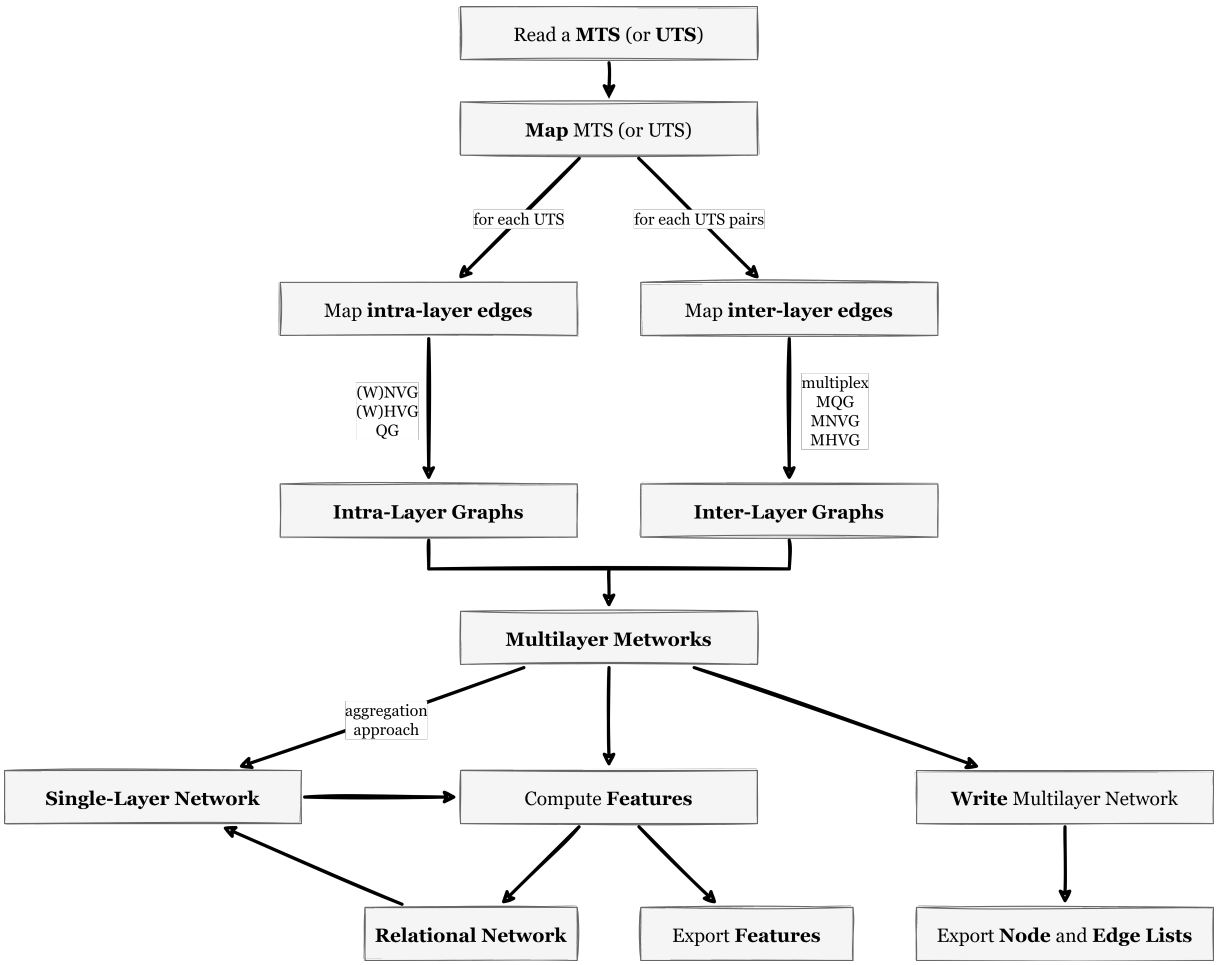


Figure 8.2: Schematic diagram with the possible workflow paths that *tsmnet* allows.

8.2 Implementation Aspects

Apart from the necessary set of standard libraries, the *tsmnet* framework does not depend on any external framework to work. However, we make use of the `igraph` library available for C, as an auxiliary and complementary tool to the framework. The `igraph` is used to model networks, it provides many different network analysis tools, which we can be used to extract information from the MNet and its intra-layer graphs and inter-layer graphs.

The process of mapping methods to transform MTS into MNets normally requires a high number of computations. For this reason, the slowest and most important feature functions are implemented in parallel by multithreading support of `thread` and `mutex` libraries.

Multilayer Networks Data Structure

The ***components*** module is a fundamental module of *tsmnet*. It incorporates the whole data structure developed to store the multilayer network information (nodes and edges). The structure implemented here is defined based on the definitions of Kivelä et al. (2014) and Dickison et al. (2016), and the corresponding libraries `Pymnet` (a Python software library available at http://www.mkivela.com/pymnet/) and `multinet` (Python and R software libraries Magnani et al., 2022).

The general `MultilayerNetwork` data structure follows the general definition of MNets presented in Section 2.3.2, and depends on the basic structures represented by the following components classes (see Figure 8.3 for class diagram):

`Edge` a basic component of MNet that represents an intra or interconnection between two `Node` components belonging to the same `Layer` component or to different `Layer` components, respectively. These components are used to represent intra and interconnections within and between observations (or patterns) of time series components.

`Node` a basic component of MNet that is represented by an `Entity` component and belongs to a `Layer` component in a MNet. It is used to represent a specific timestamp (or sample quantile) of an UTS component.

`Entity` a basic component of MNet that represents a set of `Node` components in a MNet. `Entity` is used to represent the single timestamps or single sample quantiles reference.

`Layer` a basic component of MNet that represents a set of `Node` components and can belong to a `Aspect` in a MNet. It represents a UTS component.

`Aspect` a basic component of MNet that represents a set of `Layer` components in a MNet. It can be used to represent the dimensions (or the feature) of a multidimensional MTS.

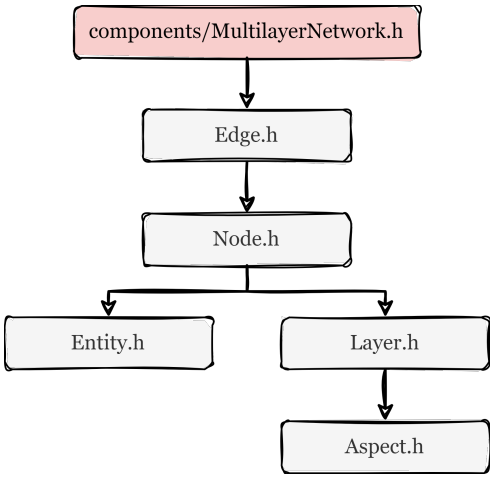


Figure 8.3: Dependency graph of `MultilayerNetwork` class.

Time Series Mapping Functions

The functions that define the mappings used to translate MTS into MNets are the core of *tsmnet*. These functions are implemented in ***mappings*** module where the main class `mappings` manages the call of specific mapping functions, the available functions are the following:

`do_mapping()` Function that creates the unique `Entity` and `Layer` components and calls the respective mapping methods to add the intra-layer edges and inter-layer edges.

`map_NVG()` Maps a UTS component into a (weighted) natural visibility graph. The undirected and directed versions of NVG are internally incorporated into the MNet structure. The function follows different implementation strategies depending on the time series length, in order to be more efficient. Uses the original implementation proposed in (Lacasa et al., 2008) (see Section 3.1.1.1) and the divide and conquer implementation proposed in (Lan et al., 2015).

`map_HVG()` Maps a UTS component into a (weighted) horizontal visibility graph. The undirected and directed versions are internally incorporated into the MNet structure. The function follows different implementation strategies depending on the time series length. Uses the original implementation proposed in (Luque et al., 2009) (see Section 3.1.1.2) and the linear time implementation proposed in (Zhu et al., 2014b).

`map_QG()` Maps a UTS component into a layer using the quantile graph method (see Section 3.1.2.1). The intra-layer edges are directed and weighted, with weight representing the transition frequency between quantile ranges, or the transition probability, if required a Markov matrix. The function uses a scheme of linear interpolation of the empirical distribution function to the quantile computation. If the number of quantiles, η, is not given by the user, is used the heuristic presented in (Campanharo et al., 2018): $\eta \approx 2T^{1/3}$.

`map_interMNVG()` Maps a MTS into a multilayer (weighted) natural visibility graph. The function adds inter-layer edges, between pairs of layers, based on the cross-natural visibility method (see Section 4.1). A divide and conquer strategy (based on (Lan et al., 2015)) and code parallelization is implemented.

`map_interMHVG()` Maps a MTS into a multilayer (weighted) horizontal visibility graph. The function adds inter-layer edges, between pairs of layers, based on the cross-horizontal visibility method (see Section 4.1). A linear strategy and code parallelization is used.

`map_interMQG()` Maps a MTS into multilayer quantile graph. The function adds inter-layer edges, between pairs of layers, based on the sequence of contemporary quantiles method proposed in Section 4.2.

`map_multiplex()` Create a multiplex network adding inter-layer edges between counterpart nodes across the layers. It runs in parallel.

All mapping functions are fully implemented in *tsment* framework. Figure 8.4 presents the corresponding class dependency diagram.

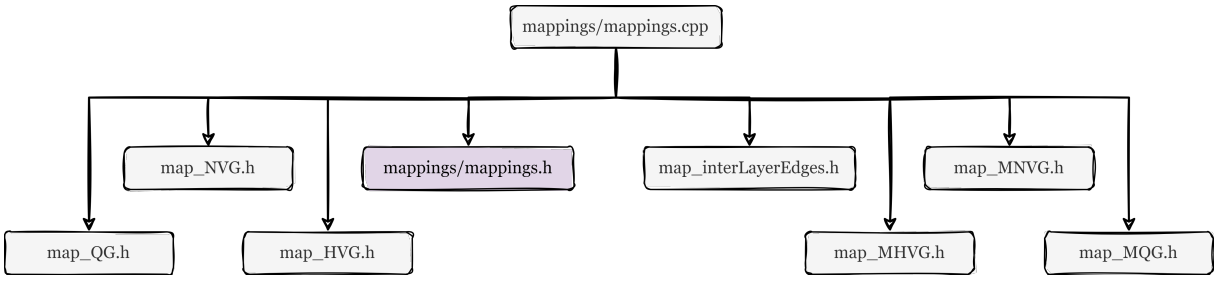


Figure 8.4: Dependency graph of `mappings` class.

Topological Features

tsmnet provides a considerable set of topological features of MNets that incorporate the ***metrics*** module. We divide these features into the following categories (see Figure 8.5):

metrics The `metric` module incorporates a set of auxiliary functions that return sets of local, global, and relational topological features for the intra-layer, inter-layer, all-layer, and whole MNet structures. The concept behind these features is presented in Section 6.1. The auxiliary functions call the respective metric functions depending on the intended feature concept.

basic Set of functions to compute basic features of the MNet: `order`, the number of nodes/entities in a MNet, `size`, the number of edges, and `density`, the graph density.

degree Set of functions to compute the degree centrality measures of the MNet. In this module, *tsmnet* has a diverse choice of features related to the degree (number of connections of a node/entity) in the structure and substructures of MNet. This includes degree vectors, degree sequences, distributions, means, standard deviations, and maximum and minimum values. It also includes strength (weighted degree) features and the new proposed features: average ratio degree (see Section 6.1).

distance Set of functions to compute the topological features related to path lengths in a MNet. It includes the shortest path length, average path length, diameter, and radius. The implementations follow the Dijkstra implementation. *tsmnet* also include the mean jump length feature proposed in (Campanharo and Ramos, 2016).

centrality Set of centrality features, namely, betweenness centrality, closeness centrality, harmonic closeness centrality, and eccentricity. The betweenness centrality is only implemented for the intra-layer graph and the rest of the features use the distance measure that follows the Dijkstra algorithm.

importance Set of features that provide a description, in terms of neighborhood, of the entities in the multilayer network. It measures the importance of the components of a MNet in terms of redundancy and relevance features introduced to multiplex networks (Dickison et al., 2016).

overlap Feature set that measures the overlap edges in a set of layers of MNet (see (Campanharo and Ramos, 2016) for more details).

entropy Designed to compute the entropy of the MNet according to the Von Neumann entropy. However, it is not yet implemented.

similarity Set of features to measure similarities between pairs of layers in an MNet. It includes local similarity, connection similarity, inter-layer mutual information, Kullback–Leibler, and Jensen–Shannon divergences.

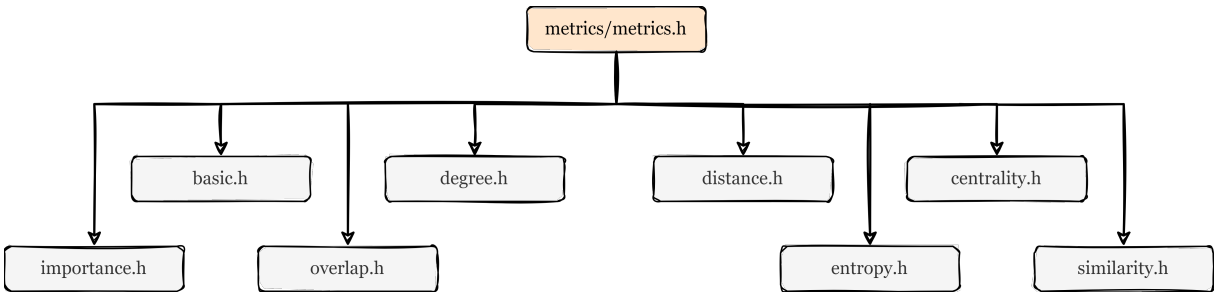


Figure 8.5: Dependency graph of *metrics* module.

Useful Functions

The ***utils*** module serves various additional functionalities which support the remaining *tsmnet* modules. It includes:

math Auxiliary basic mathematical and statistical functions. Among others, it includes auxiliary functions for matrix calculation (`markov_matrix`, `trace_matrix`, `multiplication_matrix`, `transpose_matrix`), data normalization (`normalize_Min_Max`), and quantile calculation (`Quantiles`).

dtw Method to calculate the cost matrix of dynamic time warping between two time series components (`euclideanDistance`, `costMatrixDTW`).

graphMatrix Data structure implementation of a graph (defined as an abstract class) using both adjacency matrix and list.

transform Set of functions that take a MNet and return transformed versions of them. `aggregate` consists of aggregation methods, which merge all nodes corresponding to the same entity into a single node in a new single-layer network (or a new 1-aspect multilayer network, if the initial network has more than 1 aspect), and aggregates the edges between pairs of nodes that belong to all layers of the network or to preselected layers. The aggregation can result in two types of edge weights: the edge weight indicates the sum of weights of aggregated edges, or the edge weight indicates the number of layers where the edge is present. `relational_net` create a new single-layer network with nodes corresponding to each layer of a MNet and weighted directed edges represent a relationship between pairs of layers that are determined by a given topological feature. `supra_adjacency_matrix` consists of a flattening approach to the transformation of a MNet structure into the corresponding supra-adjacency matrix.

igraphFormat Utility functions to convert multilayer network components in to `igraph` structures (`create_igraph`).

igraphMeasures Utility functions that call a set of `igraph` functions that compute (local and global) topological features (`igraph_global_measures` and `igraph_local_measures`).

In Figure 8.6 we display the class dependency diagram.

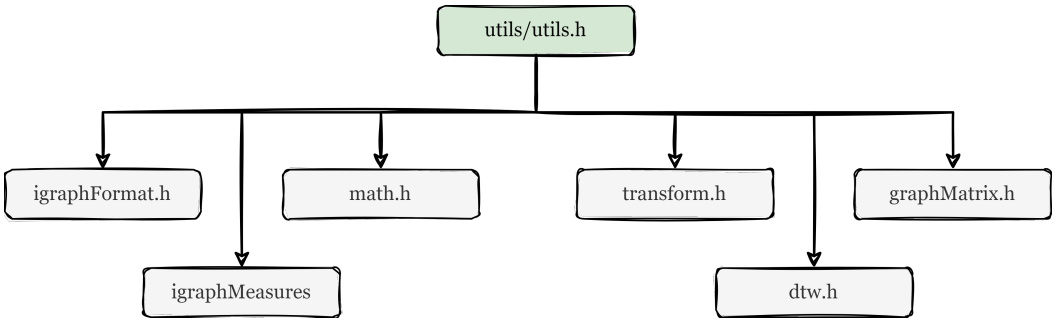


Figure 8.6: Dependency graph of *utils* module.

Input/Output Functions

The *tsmnet* has two functions to read and store multivariate time series data (***readMTS***).

`read_mts_csv()` read multivariate time series data from comma-separated values files (`.csv`),

`read_mts_ts()` read multivariate time series data from a bespoke time series storage format [1] files (`.ts`),

these functions store the MTS data in appropriate data structures. In particular, the last function is useful to read datasets from *The UEA & UCR Time Series Classification Repository* (Bagnall et al.).

The ***writeMNet*** include utility functions to write a multilayer network in various types of appropriated files. It includes the following functions:

- `write_gephi_format`: generates node lists and edge lists, designed to be used in `Gephi` software [2].
- `write_multinet_format`: generate the appropriate file based on the R software package `multinet` terms [3].
- `write_pymnet_format`: generate the appropriate file based on the Python software library `pymnet` terms [4].
- `write_muxviz_format`: generate the appropriate file based on the R software package `muxViz` terms [5].

[1] https://github.com/alan-turing-institute/sktime/blob/main/examples/loading_data.ipynb
[2] https://gephi.org/
[3] https://www.rdocumentation.org/packages/multinet/versions/3.2/topics/multinet.IO
[4] http://www.mkivela.com/pymnet/
[5] https://github.com/manlius/muxViz

8.3 Final Remarks

In this chapter, we present the framework that we have developed throughout this research work. *tsmnet* is a C++ framework aimed at transforming multivariate time series into multilayer network structures, as well as extracting topological features for time series analysis. A constant concern about mapping methods is the run-time for transforming large sets of multivariate series. To minimize this problem, *tsmnet* was designed to use mapping functions with the best time complexity as well as to execute mapping functions in parallel, using multithreading. Our goal with *tsmnet* is to design it so that it is easy to access and use. At this moment, *tsmnet* receives a series of commands as arguments and automatically processes these commands in order to generate an output that can be: multilayer network files, files with topological features for an MTS dataset, or several files of topological features for MTS datasets that are used for feature-based mining tasks, such as clustering and classification. Some of *tsmnet*'s functionality is not completely finalized and some methods can still be improved and optimized. The aim is to improve it over time and extend it as new methods emerge.

Chapter 9

Conclusions and Future Work

Complex networks are universal models capable of describing the most diverse artificial and natural systems. The representation of (univariate and multivariate) time series data via networks provides a new tool for extracting information from these high-dimension datasets. Network methodologies have already been applied to time series analysis in many fields, such as finance, medicine, climate science, and engineering, and to respond to tasks such as forecasting, classification, clustering, and outliers detection. However, this is still a recent area, despite nearly two decades of research work. Time series data present characteristic properties related to temporal dependence, which can be serial and crossed, and which leads to difficulty in understanding and properly analyzing this dataset. Extraction of descriptive features from such data represents a difficult problem from a mathematical, selection, and also computational point of view when we are dealing with large datasets. The objective of this thesis is precisely to provide alternative and innovative methods that can help to solve these problems and to leverage the currently existing methods.

This chapter summarizes the main results obtained, present the main contributions, and discusses limitations and directions for future research.

9.1 Summary

This work overviews the state-of-the-art on time series analysis via network science, describing all main mappings methods with an emphasis on their concepts and algorithmic aspects. With nearly two decades of related research work, there is substantial related published literature.

The core of the work resides in extending the existing methodology to the context of multivariate time series data. From a conceptual point of view, current methods follow two distinct approaches based on the structures of the resulting networks: single-layer networks where each series (or patterns based on observations from multivariate series) is represented by a node in the network and the connections are established based on relationships between the series (or patterns); and multiple layers where each layer represents a time series and each layer can be mapped based on one of the univariate time series mapping methods, separately.

Most existing works focus on the first mapping approach, and the second approach is very recent and focuses only on a specific set of multilayer networks, i.e. the multiplex networks. However, we are interested in the more complete structures of multilayer networks. Such structures allow us to add more information to the data to be mapped, avoiding losses inherent to the mapping. This implies that it is possible to add external connections between the mapped structures (timestamps or patterns/symbols), leaving the criteria with which they are established dependent on the mapping concept used.

One of the main innovations introduced in this work is the creation of two new methods for mapping multivariate time series in multilayer networks based on two new concepts: cross visibility and transition of contemporary symbols. The addition of inter-layer edges based on these concepts allows us to add information from cross-time series data.

The first question is how to perform these mappings? *tsmnet* is a C++ framework that was created with that purpose in mind. It represents a general data structure for multilayer networks capable of storing the set of components that constitute a multilayer network: aspects, layers, entities, nodes, and edges (intra and inter-layers). And it represents an appropriate data structure to map (univariate and multivariate) time series data in multilayer networks. Its main function is to read a set of multivariate time series data and apply the previously provided mapping methods, storing the resulting information in the multilayer data structure in an appropriate way.

The mapping methods developed in this work, as well as the underlying methods (corresponding methods in the univariate context), are implemented in the data structure that generates the mapping methods. Multilayer quantile graphs have no limitations and are easy to implement, requiring only one parameter that dictates the number of quantiles used for the mapping. Cross-visibility graphs, on the other hand, are much more complex at the computational level for very large datasets, since they depend on the length of the time series and the number of components. As this last method is pairwise (the condition of visibility is made between all pairs of layers and it is directional) we parallelize the code in order to execute this mapping faster for a large dataset. In this work, we focus on single-aspect multilayer networks. However, the proposed methods can be extended, naturally, to multiple-aspects. In particular, *tsmnet* already allows this extension so the data structure of multilayer networks was designed according to the more general definition which includes multiple aspects.

The second question is how can I extract (multivariate) time series information through the resulting (multilayer) network structures? We propose a set of topological features to answer this question. We propose *NetF*, a set of topological features of univariate time series networks extracted from three types of univariate mapping methods: (natural and horizontal) visibility graphs and quantile graphs. And we propose a set of topological features of multilayer networks that give rise to the *MNetF* which constitutes a set of topological features of multilayer networks of multivariate time series extracted from two types of multivariate mapping methods: multilayer horizontal visibility graphs and multilayer quantile graphs.

The third question is are topological features appropriate for analyzing properties of temporal data? In order to answer this question, we propose an approach based on topological features that aim at the exploratory analysis of the space of (multilayer) topological features resulting from (multivariate) time series mappings and aims to apply the resulting topological feature vectors to time series mining tasks, which in this work was the clustering analysis.

With all methods, algorithms, and data structures developed, we obtained a methodological framework for the analysis of multivariate and univariate time series datasets.

9.2 Main Contributions

This work makes the following major contributions:

- **Survey and comparison of time series mappings algorithms:** we present a complete literature survey of mapping algorithms (in univariate and multivariate settings), creating a high-level conceptual division and an associated taxonomy and illustrative figure able to distinguish methods, and we present the main experimental results and comparison.
- **A set of network-based univariate time series features:** we introduce *NetF* as an alternative set of features, incorporating several representative topological features of different complex network mappings of the univariate time series. We can connect mapped network features to properties inherent in diversified time series models, showing that *NetF* can be useful to characterize time data. We also show that *NetF* is appropriate to time series mining tasks.
- **New multivariate time series mappings methods:** we propose two novel methods for mapping multivariate time series into complete multilayer network structures. The first is based on a new concept of cross-visibility between pairs of time series components. It is designed to keep the temporal lagged dependencies between timestamps of different variables in the network structure. And the second is based on contemporaneous transition quantiles. It is designed to analyze the contemporaneous variations between the pairs of time series components. We show that the incorporation of inter-layer edges based on these mapping concepts can add additional information to the network structure when

the multiplex structures do not allow connection between timestamps or different symbols from different time series.

- **A set of network-based multivariate time series features:** we propose a set of global topological multilayer network features as a novel set of features for multivariate time series data that comprise intra-layer topological features, inter-layer topological features, all-layer topological features, and relational features. We propose new topological features aimed at relating intra and inter-layer connections between time series components. The features set allows us to analyze and compare the underlying properties of intra-layer and inter-layer edges, and to assess the contribution of the proposed mapping method concerning the multiplex methods in the literature.

- ***tsmnet* framework and its associated algorithms:** we implement a general framework for mapping multivariate time series data into multilayer networks, extracting topological features from these networks (and from their substructures), and able to extract (by writing) the multilayer network generated to external analysis.

9.3 Future Work

Time series analysis via network science approaches can be leveraged to address existing fundamental open issues. Among them, we highlight the following: missing values, unequally spaced data, visualization of high-dimensional data, time series classification, and clustering. Data recorded using electronic devices such as sensors are prone to missing values and conventional imputation methods may cause bias in the data. The work of Donner and Donges (2012) addresses this problem of missing values via a network perspective and, although the results are preliminary, they highlight a path to the solution of this issue. Unequally spaced data collected over time occurs frequently for several reasons, hindering the application of time series analysis methodology. The most common approach to this issue is data aggregation which leads to the loss of information. As far as we know, this problem has not yet been addressed from the point of view of networks. Visualization of high dimensional temporal data is a practical issue to whose solution may contribute strategies to reduce the dimensionality of the data by using suitable mapping methods (for example, the multilayer quantile graphs that we introduce in this work). Mining time large collections of time series data is increasingly accomplished via time series features, e.g. trend, length of time series, the strength of the trend, autocorrelations (Kang et al., 2017). The features extracted from the time series networks may be added to the classical set of time series features to enhance the data characterization. In this work, we show that (single-layer and multilayer) network features can be used to accurately cluster (univariate and multivariate) time series from different models, a relevant and important contribution to time series mining.

We hope that the work presented in this thesis can lead to further research in this captivating area of data science. We present general methods for also general tasks of time series mining. That is, we present a methodological work that can serve as a basis for several branches of research on temporal data. Most of the work presented in this thesis was based on two basic concepts of mapping methods existing in the literature, within a wide list of possibilities, based on global topological features of graphs and commonly used in the area, also within a wide range available, and based on high-level network structures where several approaches can be followed for their analysis. We provide some indications for future work below.

- **Improve mapping methods available in *tsmnet*.** The proposed multilayer visibility methods can be computationally heavy for datasets with many time series components since the methods perform visibility between components in a pair-wise way.
- **Explore other topological features.** In our work, we extended the one of most common topological features of networks to a multilayer setting. However, there is a vast other set of topological features that can be explored and that can even get better discriminatory results from time series analysis.
- **Time series classification problems.** In this work, we focus on time series clustering tasks to analyze our approaches because clustering is an unsupervised learning method, so it was more advantageous for our analysis. However, time series classification problems are in recent years one of the main focuses of time series data mining research, and therefore using topological features in classification models can lead to interesting results.
- **Spatio-temporal datasets.** Data indexed in time and indexed in space is a type of data that has attracted the attention of researchers in recent times. The flexible structure of multilayer networks can enhance the creation of interesting space-time models.

9.4 Final Remarks

More than four years ago, when work on this thesis began, the theme "Time Series Analysis via Network Science" was still very much focused on time series analysis for univariate settings. Over the years, developments around this area have grown a lot and lifted the veil for multidimensional network structures. We tried to be as general as possible, to include as many time series data problems as possible, focusing on the general structures of multilayer networks, which in itself is almost as broad as the main topic addressed here. The concept of multidimensionality ends up uniting both multivariate temporal data and multilayer network structures and, in the end, we feel that everything seems to be directed toward the future of new lines of research in data science.

Researching these tasks was both a difficult and very interesting process, where we studied diversified concepts from major core areas (time series analysis, network science, and data mining), as well as algorithmic aspects, and applied our methods to exploring synthetic and real datasets. We hope that our tools will be useful for the scientific community and practitioners. One of the most rewarding aspects of this process was receiving positive feedback on the work that was published and made available.

Finally, we hope that our work will lead to new research.

Appendix | A

Mapping Algorithms

In this Appendix we present the implementation algorithms that support the mapping methods proposed in this thesis.

We present the implementation algorithms that support the MHVG method. The Algorithm 5 describes our implementation of the HVGs to a given input univariate time series component. And the Algorithm 6 describes our implementation of the Cross-HVG method to map two given input time series components into a Cross-HVG. Note that the time complexity of the algorithm can be improved by computing the two "for" loops in parallel.

We also present the implementation Algorithm 7 that describes our implementation of the QGs to a given input univariate time series component.

Algorithm 5: Horizontal Visibility Graph

Input: A time series, $\boldsymbol{Y}^a$, (***ts***), and a layer, L_a, (***layer***)

Procedure `HVG(`*ts*, *layer*`)`

```
1   T ← ts.size()                                   ▷ The time series length
    for i ← 1 to T do
2       mnet.add_Node(i, layer)                     ▷ Add node-layer v_i^a corresponding
                                                      to timestamp t_i
    end
    for i ← 1 to T − 1 do
3       mnet.add_Edge(i, i+1, layer)                ▷ Add intra-layer edge (v_i^a, v_{i+1}^a)
4       k ← 1
        for j ← 2 to T − i do
5           condition ← ts[i + k]
6           p ← ts[i + j]
            /* Test HVG condition:  Eq. 3.2                                          */
            if condition ≥ ts[i] then
7               break
            else if condition < p then
8               mnet.add_Edge(i, i+j, layer)        ▷ Add intra-layer edge (v_i^a, v_{i+j}^a)
9               k ← j
        end
    end
10  return
```

Algorithm 6: Cross-Horizontal Visibility Graph

Input: Two rescaled time series, $\boldsymbol{Z}^a$ and $\boldsymbol{Z}^b$, ($\boldsymbol{tsA}$, $\boldsymbol{tsB}$), the corresponding layers, L_a and L_b, ($\boldsymbol{layerA}$, $\boldsymbol{layerB}$), and the maximum time series, $\texttt{max}\left(\boldsymbol{Z}^a, \boldsymbol{Z}^b\right)$,($\boldsymbol{tsMax}$)

Procedure CHVG($tsA, tsB, tsMax, layerA, layerB$)

$T \leftarrow tsMax.size()$ ▷ The time series lengths
/* Test cross-horizontal visibility to the right of $Y_{a,i}$ */
for $i \leftarrow 1$ **to** $T-1$ **do**
  $k \leftarrow i+1$
  for $j \leftarrow i+1$ **to** T **do**
    if $j = i+1$ **then**
      mnet.add_Edge(*i, i+1, layerA, layerB*) ▷ Add inter-layer edge (v_i^a, v_{i+1}^b)
    else
      $condition \leftarrow tsMax[k]$
      $p \leftarrow tsB[j]$
      /* Test Cross-HVG condition: Eq. 4.1 */
      if $condition \geq tsA[i]$ **then**
        break
      end
      if $condition < p$ **then**
        mnet.add_Edge(*i, j, layerA, layerB*) ▷ Add inter-layer edge (v_i^a, v_j^b)
        $k \leftarrow j$
      else if $condition < tsMax[j]$ **then**
        $k \leftarrow j$ ▷ Update index k when $tsMax[j] > tsMax[k]$
      end
    end
  end
end
/* Test cross-horizontal visibility to the left of $Y_{a,i}$ */
for $i \leftarrow 2$ **to** T **do**
  $k \leftarrow i-1$
  for $j \leftarrow i-1$ **to** 1 **do**
    if $j = i-1$ **then**
      mnet.add_Edge(*i, i-1, layerA, layerB*) ▷ Add inter-layer edge (v_i^a, v_{i-1}^b)
    else
      $condition \leftarrow tsMax[k]$
      $p \leftarrow tsB[j]$
      /* Test Cross-HVG condition: Eq. 4.1 */
      if $condition \geq tsA[i]$ **then**
        break
      end
      if $condition < p$ **then**
        mnet.add_Edge(*i, j, layerA, layerB*) ▷ Add inter-layer edge (v_i^a, v_j^b)
        $k \leftarrow j$
      else if $condition < tsMax[j]$ **then**
        $k \leftarrow j$ ▷ Update index k when $tsMax[j] > tsMax[k]$
      end
    end
  end
end

Algorithm 7: Quantile graph

Input: A time series, $Y_{a,t}$, (***ts***), a layer, L_a, (***layer***), and a value with the number of quantiles, (***numQ***)

Procedure `QG(`*ts*, *layer*, *numQ*`)`

```
T ← ts.size()                                  ▷ The time series length
probs ← {}                                     ▷ List to store the probabilities
quantiles ← {}                                 ▷ List to store the sample quantiles
for i ← 1 to numQ do
    probs[a] ← i/numQ
end
quantiles ← Quantile(ts, probs)                ▷ Produces sample quantiles using linear
                                                 interpolation of the empirical cdf
for i ← 1 to numQ do
    mnet.add_Node(i, layer)                    ▷ Add node-layer v_i^a corresponding to the
                                                 i-th quantile
end
for i ← 1 to numQ − 1 do
    from_node ← which_geq(quantiles, ts[i])    ▷ Find corresponding quantile of Y_{a,i}
    to_node ← which_geq(quantiles, ts[i+1])    ▷ Find corresponding quantile of Y_{a,i+1}
    edge ← mnet.get_Edge(from_node, to_node, layer)   ▷ Get the intra-layer edge
                                                        between consecutive time
                                                        quantiles if it already exists
    /* If the intra-layer edge does not exist yet                          */
    if !edge then
        mnet.add_Edge(from_node, to_node, layer, 1)   ▷ Add intra-layer edge with
                                                        weight 1
    end
    /* If the intra-layer edge exists                                      */
    else
        w ← mnet.get_weight(edge)              ▷ Get the weight of the edge
        mnet.set_weight(edge, w+1)             ▷ Increase the weight of the edge
    end
    /* Store the current quantile(s) in the quantiles sequence             */
    layer.q_seq[i] ← from_node
    if i == T − 1 then
        layer.q_seq[T] ← to_node
    end
end
return
```

Appendix B

NetF: Evaluation on Univariate Time Series Models

B.1 Univariate Time Series Models

Main reference (Shumway and Stoffer, 2017)

Linear Models

WN The *white noise* process, ϵ_t, is a sequence of i.i.d. random variables with mean 0 and constant variance σ_ϵ^2. It is the simplest time series process that reflects information that is neither directly observable nor predictable. We generated $\epsilon_t \sim N(0,1)$ white noise processes and refer to them as `WN`.

AR(p) We defined a process as an AR process of order p if it satisfies the following equation:

$$Y_t = \sum_{i=1}^{p} \phi_i Y_{t-i} + \epsilon_t, \tag{B.1}$$

where ϵ_t is the white noise and ϕ_i is the autoregressive constant. We used $p \in \{1,2\}$, and parameters $\phi_1 \in \{-0.5, 0.5\}$ to generate AR(1) processes and $\phi_1 = 1.5$ and $\phi_2 = -0.75$ for AR(2) processes. These parameters ensure that the time series present the following characteristics: $\phi_1 = 0.5$ leads to smoother time series than $\phi_1 = -0.5$; and $\phi_1 = 1.5$ and $\phi_2 = -0.75$ generates pseudo-periodic time series. We refer to the three models generated as `AR(1)-0.5`, `AR(1)0.5` and `AR(2)`, respectively.

ARIMA(p, d, q) The *autoregressive integrated moving average* model is a generalization of the ARMA model suitable for modeling non-stationary time series. A process is an ARMA(p, q) process if it satisfies the equation:

$$Y_t = \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t, \tag{B.2}$$

where θ_i is the moving average constant. If a process is a non-stationary time series it can be written as an ARIMA(p, d, q) process if its dth-differences $\nabla^d Y_t = (1 - B)^d Y_t$, $d \in \mathbb{N}$, is a stationary ARMA(p, q) process. So Y_t satisfies the following equation,

$$\left(1 - \sum_{i=1}^{p} \phi_i B^i\right) (1 - B)^d Y_t = \left(1 + \sum_{i=1}^{q} \theta_i B^i\right) \epsilon_t, \tag{B.3}$$

where B represents the backshift operator, $BY_t = Y_{t-1}$. We use $p = 1$, $d = 1$, $q = 0$, and $\phi_1 = 0.7$ to generate ARIMA$(1, 1, 0)$ processes with stochastic trend. We refer to these processes as `ARIMA`.

ARFIMA(p, d, q) *Autoregressive fractionally integrated moving average* model is a generalization of the ARIMA model useful for modeling time series with long range dependence. A process is an ARFIMA(p, d, q) if it satisfies the (Eq B.3) and the difference parameter, d, can take real values. We generate the ARFIMA$(1, 0.4, 0)$ processes that exhibit long memory and refer to them as `ARFIMA`.

Univariate time series are generated from the above UDGP using the following R packages: `timeSeries` (Wuertz et al., 2017a) and `fracdiff` (Maechler et al., 2020).

Nonlinear Models

SETAR(1) The *self-exciting threshold autoregressive* model of order 1 specify the nonlinearity in the conditional mean. It is useful for processes with regime changes that approximate a nonlinear function by piece wise linear functions dependent on the regime (Tong, 2011). This model can be presented by the following equation system,

$$Y_t = \begin{cases} \alpha Y_{t-1} + \epsilon_t \ , & \text{if } Y_{t-1} \leq r \\ \beta Y_{t-1} + \gamma \epsilon_t, & \text{if } Y_{t-1} > r \end{cases}, \tag{B.4}$$

where r represents a real threshold. We used $\alpha = 0.5$, $\beta = -1.8$, $\gamma = 2$, and $r = -1$, generating time series with regimes presenting different autocorrelation properties: in the first, the correlation is positive, while in the second, it alternates between positive and negative values. We refer to this model as `SETAR`.

HMM *Hidden Markov models* are probabilistic models for the joint probability the random variables $(Y_1, \ldots, Y_T, X_1, \ldots, X_T)$ where Y_t is a discrete (or continuous) variable and X_t is a *hidden Markov chain* with a finite number of states. The following conditional independence assumptions hold (Zucchini et al., 2016):

1. $P(X_t \mid X_{t-1}, Y_{t-1}, \ldots, X_1, Y_1) = P(X_t \mid X_{t-1})$,
2. $P(Y_t \mid X_T, Y_T, \ldots, X_1, Y_1) = P(Y_t \mid X_t)$.

We used 2 states and the transition matrix $\begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$. The data are generated from a Poisson distribution with $\lambda = 10$ for the first regime and $\lambda = 15$ for the second. We refer to this model as `HMM`.

The following nonlinear models are based on ARCH models where the mean-corrected asset return is serially uncorrelated but dependent and the dependency can be described by a simple quadratic function of its lagged values (Tsay, 2005). Hereafter, ϵ_t are uncorrelated random variables, z_t represents a white noise with variance 1 and σ_t the standard deviation of ϵ_t, that is $\epsilon_t = \sigma_t z_t$.

GARCH(p, q) The GARCH model is a *generalization* of the ARCH model in which the conditional volatility is a function not only of the squares of past innovations, but also of their own past values (Cryer and Chan, 2008). Thus, ϵ_t is a GARCH(p, q) process if it satisfies the following equation,

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2, \tag{B.5}$$

where $\omega > 0$, $\alpha_i, \beta_i \geq 0$, $\sum_{i=1}^{p} \beta_i + \sum_{i=1}^{q} \alpha_i < 1$. The conditional standard deviation can exhibit persistent periods of high or low volatility because past values of the process are fed back into the present value. We used $p = 1, q = 1, \omega = 10^{-6}$, $\alpha_1 = 0.1$ and $\beta_1 = 0.8$ to generate the GARCH$(1, 1)$ processes and we refer to them as `GARCH`.

EGARCH(p, q) The *exponential* GARCH allows asymmetric effects of positive and negative shocks on volatility (Tsay, 2005). The EGARCH(p, q) model is given by the equation,

$$log(\sigma_t^2) = \omega + \sum_{i=1}^{q} \alpha_i \left| \frac{\epsilon_{t-i}}{\sigma_{t-i}} \right| + \sum_{i=1}^{p} \beta_i log(\sigma_{t-i}^2) + \sum_{i=1}^{q} \gamma_i \frac{\epsilon_{t-i}}{\sigma_{t-i}}, \tag{B.6}$$

where $\omega = \alpha_0 - \alpha_1 \sqrt{\frac{2}{\pi}}$, α_i characterize the volatility clustering phenomena, β_i is the persistence parameter, and γ_i describes the leverage effect. The logged conditional variance allows to relax the positivity constraint of the model coefficients. To this model we choose $p = 1, q = 1, \omega = \left(10^{-6} - 0.1\sqrt{\frac{2}{\pi}}\right)$, $\alpha_1 = 0.1$, $\beta_1 = 0.01$ and $\gamma_1 = 0.3$, and we refer to it as `EGARCH`.

INAR(1) The *integer-valued autoregressive* models have been proposed to model integer-valued time series, in particular, correlated counts (Silva and Oliveira, 2004). These models are based on thinning (random) operations defined on the integers, where the following binomial thinning is the most common: let X be a non-negative integer valued random variable and $0 < \alpha < 1$, then $\alpha * X = \sum_{i=1}^{X} Y_i$ where $\{Y_i\}$ is a sequence of i.i.d. Bernoulli random variables, independent of X. A process is said to be an INAR(1) if it satisfies the equation,

$$Y_t = \alpha * Y_{t-1} + \epsilon_t. \tag{B.7}$$

If the innovation sequence ϵ_t and the initial distribution are Poisson, Y_t is said to be a Poisson INAR(1) process. We used $\alpha = 0.5$ and Poisson arrivals with $\epsilon_t \sim Po(1)$ to generate integer-valued data with autocorrelation decaying at a rate of 0.5. We refer to this model as `INAR`.

Univariate time series from the `HMM` and `GARCH` are simulated using the following R packages: `HMMpa` (Witowski and Foraita, 2014) and `fGarch` (Wuertz et al., 2017b), respectively. The time series generated from the remaining UDGPs are from our own implementation. For reproducibility purposes, the source code and the datasets are made available in https://github.com/vanessa-silva/NetF.

B.2 Feature Evaluation in Synthetic Univariate Time Series

The topological features of WNVGs, WHVGs, and QGs obtained from the 1100 time series models are, respectively, summarized in tables B.1, B.2 and B.3. The table reports the mean and standard deviation (in brackets) of the *Min-Max* normalized features. The columns of the tables are colored with a gradient based on the mean values: cells with the highest value are coloured red, cells with the lowest value are coloured white, and the remainder with a hue of red colour proportional to its value in the respective column.

Weighted Natural Visibility Graphs

Table B.1: Table of mean values of the 100 instances of each UDGP for each topological feature, resulting from WNVGs. The standard deviations are presented in parentheses.

Models	Average Degree ($\bar{k}$)	Average Path Length ($\bar{d}$)	Number of Communities (S)	Clustering Coefficient (C)	Modularity (Q)
`AR(1)-0.5`	0.430 (0.006)	0.457 (0.032)	0.225 (0.063)	0.615 (0.007)	0.900 (0.022)
`AR(1)0.5`	0.698 (0.005)	0.443 (0.037)	0.116 (0.037)	0.751 (0.009)	0.963 (0.010)
`AR(2)`	0.719 (0.008)	0.408 (0.035)	0.472 (0.132)	0.968 (0.012)	0.792 (0.082)
`ARIMA`	0.919 (0.006)	0.211 (0.095)	0.257 (0.115)	0.188 (0.079)	0.367 (0.140)
`ARFIMA`	0.790 (0.005)	0.412 (0.036)	0.099 (0.035)	0.766 (0.012)	0.973 (0.009)
`SETAR`	0.273 (0.006)	0.438 (0.041)	0.491 (0.145)	0.631 (0.007)	0.772 (0.070)
`HMM`	0.012 (0.005)	0.446 (0.039)	0.570 (0.150)	0.667 (0.008)	0.654 (0.093)
`INAR`	0.570 (0.004)	0.452 (0.118)	0.131 (0.052)	0.631 (0.011)	0.956 (0.024)
`GARCH`	0.994 (0.003)	0.433 (0.036)	0.070 (0.022)	0.652 (0.010)	0.991 (0.005)
`EGARCH`	0.940 (0.004)	0.425 (0.032)	0.066 (0.027)	0.677 (0.007)	0.980 (0.004)
`WN`	0.581 (0.005)	0.450 (0.034)	0.128 (0.044)	0.667 (0.007)	0.942 (0.009)

Weighted Horizontal Visibility Graphs

Table B.2: Table of mean values of the 100 instances of each UDGP for each topological feature, resulting from WHVGs. The standard deviations are presented in parentheses.

Models	Average Degree ($\bar{k}$)	Average Path Length ($\bar{d}$)	Number of Communities (S)	Clustering Coefficient (C)	Modularity (Q)
AR(1)−0.5	0.473 (0.005)	0.004 (0.001)	0.230 (0.057)	0.557 (0.004)	0.319 (0.056)
AR(1)0.5	0.628 (0.003)	0.009 (0.001)	0.174 (0.046)	0.697 (0.004)	0.793 (0.031)
AR(2)	0.438 (0.006)	0.022 (0.002)	0.600 (0.092)	0.953 (0.004)	0.397 (0.074)
ARIMA	0.414 (0.007)	0.588 (0.161)	0.794 (0.106)	0.988 (0.005)	0.236 (0.074)
ARFIMA	0.633 (0.003)	0.030 (0.004)	0.204 (0.047)	0.785 (0.004)	0.878 (0.028)
SETAR	0.284 (0.005)	0.003 (0.001)	0.508 (0.086)	0.504 (0.004)	0.354 (0.092)
HMM	0.009 (0.003)	0.013 (0.002)	0.640 (0.116)	0.467 (0.006)	0.385 (0.105)
INAR	0.403 (0.002)	0.034 (0.005)	0.292 (0.056)	0.034 (0.010)	0.927 (0.033)
GARCH	0.998 (0.001)	0.007 (0.002)	0.064 (0.022)	0.611 (0.004)	0.777 (0.015)
EGARCH	0.900 (0.001)	0.004 (0.001)	0.059 (0.026)	0.591 (0.005)	0.747 (0.017)
WN	0.584 (0.004)	0.004 (0.001)	0.170 (0.045)	0.605 (0.004)	0.624 (0.033)

Quantile Graphs

Table B.3: Table of mean values of the 100 instances of each UDGP for each topological feature, resulting from QGs. The standard deviations are presented in parentheses.

Models	Average Degree ($\bar{k}$)	Average Path Length ($\bar{d}$)	Number of Communities (S)	Clustering Coefficient (C)	Modularity (Q)
AR(1)−0.5	1.000 (0.000)	0.005 (0.000)	0.000 (0.000)	0.972 (0.003)	0.008 (0.003)
AR(1)0.5	1.000 (0.000)	0.005 (0.000)	0.027 (0.010)	0.971 (0.003)	0.374 (0.044)
AR(2)	1.000 (0.000)	0.024 (0.000)	0.044 (0.015)	0.829 (0.003)	0.816 (0.028)
ARIMA	1.000 (0.000)	0.943 (0.068)	0.062 (0.022)	0.144 (0.132)	0.398 (0.099)
ARFIMA	1.000 (0.000)	0.029 (0.003)	0.047 (0.017)	0.808 (0.009)	0.890 (0.039)
SETAR	1.000 (0.000)	0.016 (0.000)	0.027 (0.009)	0.946 (0.003)	0.245 (0.039)
HMM	0.276 (0.008)	0.001 (0.000)	0.730 (0.008)	0.998 (0.003)	0.289 (0.027)
INAR	0.000 (0.002)	0.002 (0.001)	0.981 (0.009)	0.984 (0.024)	0.493 (0.010)
GARCH	1.000 (0.000)	0.001 (0.000)	0.031 (0.012)	1.000 (0.001)	0.055 (0.016)
EGARCH	1.000 (0.000)	0.002 (0.000)	0.019 (0.010)	0.999 (0.001)	0.041 (0.016)
WN	1.000 (0.000)	0.001 (0.000)	0.029 (0.011)	1.000 (0.000)	0.047 (0.011)

Appendix C

MNetF: Evaluation on Multivariate Time Series Models

C.1 Multivariate Time Series Models

Main references (Cipra, 2020; Shumway and Stoffer, 2017; Tsay, 2013; Wei, 2019).

Linear Models

BWN The *vector white noise* process, $\boldsymbol{\epsilon}_t$, is a vector of sequences of i.i.d. random variables with mean vector $\mathbf{0}$ and and covariance matrix function $\boldsymbol{\Sigma}$, where $\boldsymbol{\Sigma}$ is an $m \times m$ symmetric positive definite matrix. The components of the white noise process are serially uncorrelated, $\text{corr}(\epsilon_{i,t}, \epsilon_{i,s}) = 0$ for $t \neq s$, but may be contemporaneously correlated, $\text{corr}(\epsilon_{i,t}, \epsilon_{j,t}) \neq 0$. It is the simplest multivariate time series process that reflects information that is neither directly observable nor predictable. We generate two sets of vector white noise processes, one not correlated, that is, are independent, $\boldsymbol{\epsilon}_t \sim N(0, 1)$, and we refer to this as `iBWN`, and the other contemporaneously correlated, $\begin{bmatrix} \epsilon_{1,t} \\ \epsilon_{2,t} \end{bmatrix} \sim N\left(0, \begin{bmatrix} 1.00 & 0.86 \\ 0.86 & 1.50 \end{bmatrix}\right)$, and we refer to this model as `cBWN`.

VAR(1) The *vector autoregression* process is a natural extension of the univariate AR process in which variable values depend linearly on its own previous values and on a stochastic term. We defined a VAR(1) process as a vector AR process of order 1 if it satisfies the following equation:

$$\boldsymbol{Y}_t = \boldsymbol{\varphi} + \boldsymbol{\phi}\boldsymbol{Y}_{t-1} + \boldsymbol{\epsilon}_t, \tag{C.1}$$

where $\boldsymbol{\epsilon}_t$ is the vector white noise, $\boldsymbol{\phi}$ is the vector of autoregressive constants and $\boldsymbol{\varphi}$ is the vector of intercepts. We generate a VAR(1) of 2 components with the following vector of parameters:

$$\begin{bmatrix} Y_{1,t} \\ Y_{2,t} \end{bmatrix} = \begin{bmatrix} \varphi_{1,1} \\ \varphi_{2,1} \end{bmatrix} + \begin{bmatrix} \phi_{1,1} & \phi_{1,2} \\ \phi_{2,1} & \phi_{2,2} \end{bmatrix} \begin{bmatrix} Y_{1,t-1} \\ Y_{2,t-1} \end{bmatrix} + \begin{bmatrix} \epsilon_{1,t} \\ \epsilon_{2,t} \end{bmatrix}, \tag{C.2}$$

where $\boldsymbol{\varphi} = \begin{bmatrix} 2.50 \\ 0.50 \end{bmatrix}$, $\boldsymbol{\phi} = \begin{bmatrix} 0.20 & 0.10 \\ 0.02 & 0.10 \end{bmatrix}$ and $\boldsymbol{\epsilon}_t \sim \begin{bmatrix} 1.00 & 0.10 \\ 0.10 & 1.50 \end{bmatrix}$ to generate weakly correlated VAR(1) processes, and $\boldsymbol{\varphi} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\boldsymbol{\phi} = \begin{bmatrix} 0.70 & 0.02 \\ 0.30 & 0.80 \end{bmatrix}$ and $\boldsymbol{\epsilon}_t \sim \begin{bmatrix} 1.00 & 0.86 \\ 0.86 & 1.50 \end{bmatrix}$ to generate strongly correlated VAR(1) processes. We refer to the two models generated as `wVAR` and `sVAR`, respectively.

Nonlinear Models

VGARCH$(1,1)$ Also *generalized autoregressive conditional heteroskedasticity* (GARCH) models can be generalized to multidimensional settings, extending the principle of univariate conditional heteroscedasticity to mutual volatility. We generate a bivariate GARCH$(1,1)$ model according to the following volatility equation:

$$\boldsymbol{\sigma}_t = \boldsymbol{\omega} + \boldsymbol{\alpha}\boldsymbol{\epsilon}^{\mathbf{2}}{}_{t-1} + \boldsymbol{\beta}\boldsymbol{\sigma}_{t-1}, \tag{C.3}$$

where $\boldsymbol{\sigma}_t$ denotes the volatility in the variables $\boldsymbol{Y}_t$. We generate a VGARCH$(1,1)$ of 2 components with the following vector of parameters:

$$\begin{bmatrix} \sigma_{11,t} \\ \sigma_{22,t} \end{bmatrix} = \begin{bmatrix} \omega_{1,1} \\ \omega_{2,1} \end{bmatrix} + \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} \\ \alpha_{2,1} & \alpha_{2,2} \end{bmatrix} \begin{bmatrix} \epsilon^2_{1,t-1} \\ \epsilon^2_{2,t-1} \end{bmatrix} + \begin{bmatrix} \beta_{1,1} & \beta_{1,2} \\ \beta_{2,1} & \beta_{2,2} \end{bmatrix} \begin{bmatrix} \sigma_{11,t} \\ \sigma_{22,t} \end{bmatrix}, \tag{C.4}$$

where $\boldsymbol{\epsilon}_t \sim \begin{bmatrix} 1.00 & 0.10 \\ 0.10 & 1.50 \end{bmatrix}$ to generate weakly correlated VGARCH$(1,1)$ processes, and $\boldsymbol{\epsilon}_t \sim \begin{bmatrix} 1.00 & 0.86 \\ 0.86 & 1.50 \end{bmatrix}$ to generate strongly correlated VGARCH$(1,1)$ processes. To both processes we use $\boldsymbol{\omega} = \begin{bmatrix} 0.05 \\ 0.02 \end{bmatrix}$, $\boldsymbol{\alpha} = \begin{bmatrix} 0.10 & 0.00 \\ 0.00 & 0.05 \end{bmatrix}$ and $\boldsymbol{\beta} = \begin{bmatrix} 0.85 & 0.00 \\ 0.00 & 0.88 \end{bmatrix}$. We refer to the two models generated as `wVGARCH` and `sVGARCH`, respectively.

The bivariate time series are generated from the above DGP using the following R packages, `lgarch` (Sucarrat, 2015), `mAr` (Barbosa, 2012) and `ccgarch` (Nakatani, 2014).

C.2 Autocorrelation Function Plots

See Figure C.1.

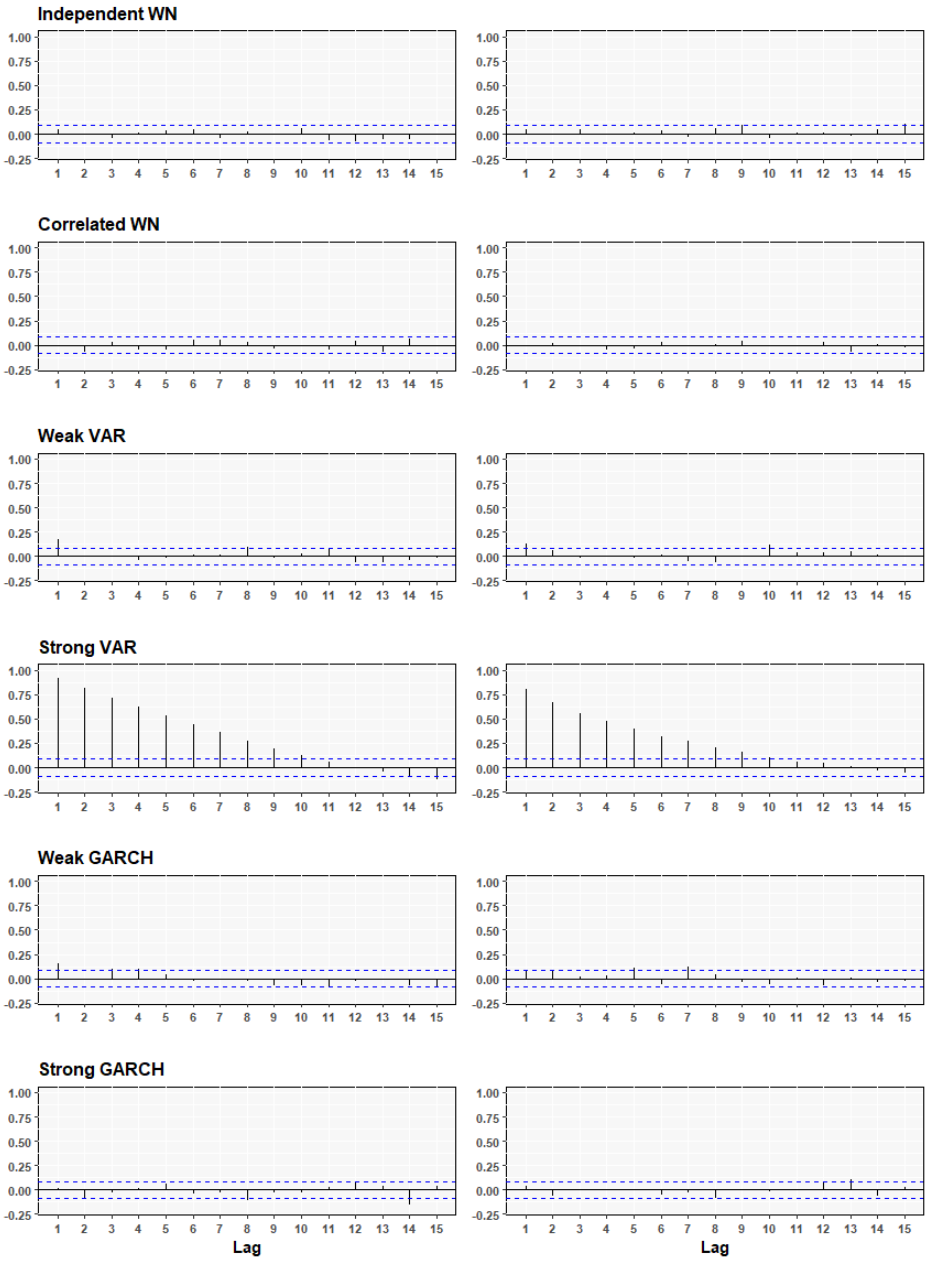


Figure C.1: Plot of the autocorrelation function of an instance of each of the MDGP. The first column refers to the ACF's of the first time series component ($Y_{1,t}$) of each model, while the second column refers to the second component ($Y_{2,t}$).

C.3 Degree Distribution of MHVG Feature Set

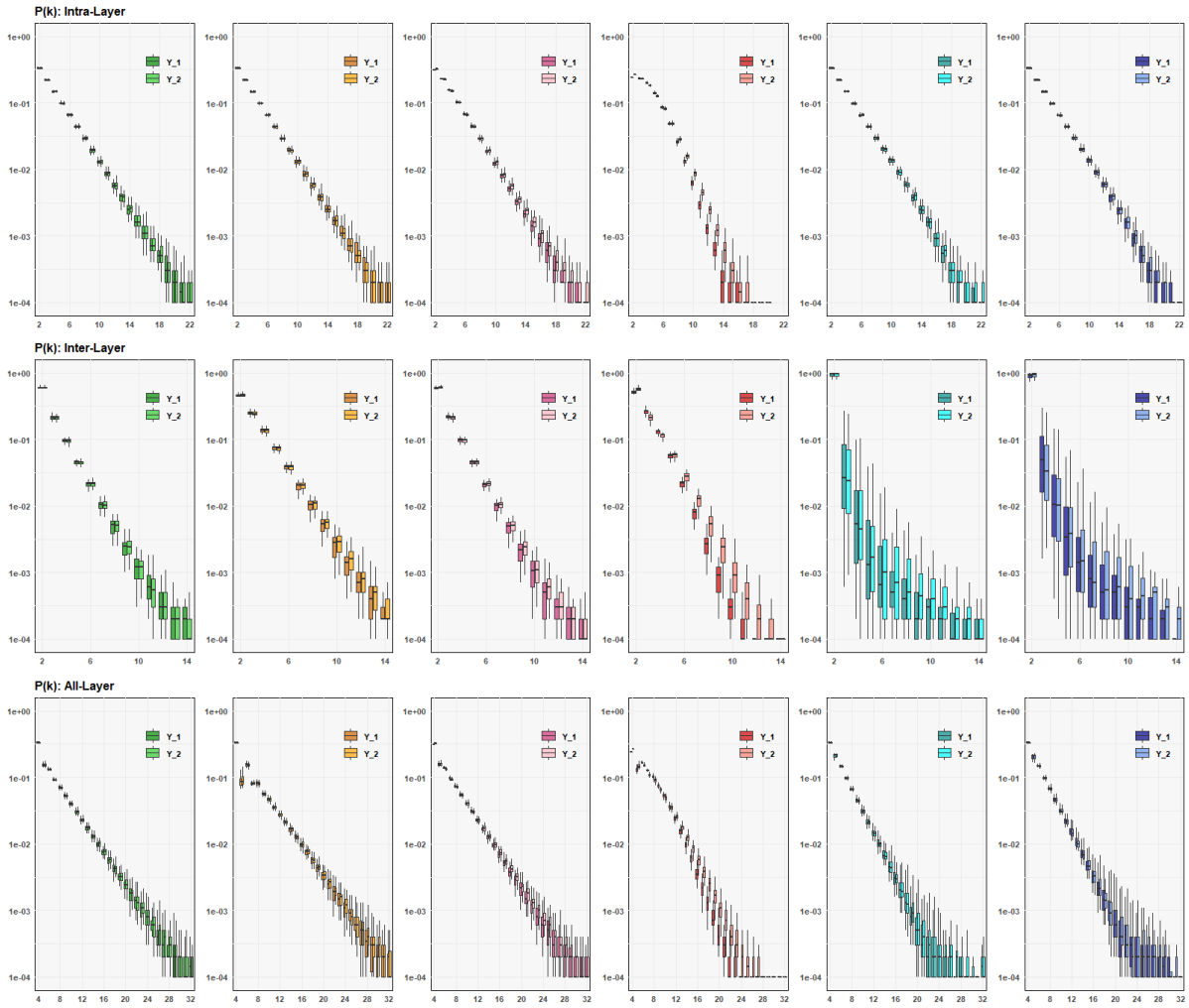
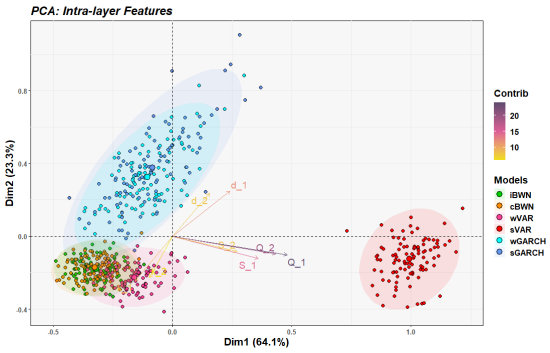
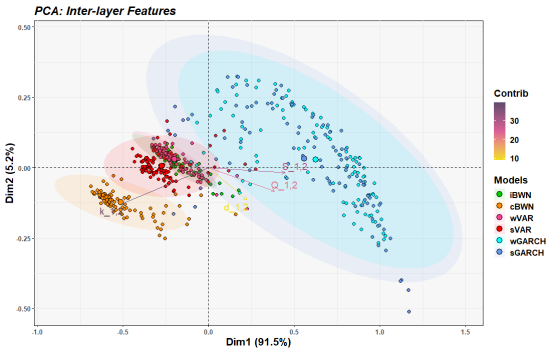


Figure C.2: Variability analysis of the degree distributions of MHVG structures that represents the MDGP's. The first line shows intra-layer degree distribution, the second shows inter-layer degree distribution, and the three line shows the all-layer degree distribution of the MHVGs. The plots are on a semi-logarithmic scale and correspond to the boxplot of the degree distribution values obtained to 100 samples of each MDGP. Different colors refer to each different MDGP ($\boldsymbol{Y}$), where the darkest colors refer to their first components ($\boldsymbol{Y}^1$) and the lighter ones to the second ($\boldsymbol{Y}^2$).

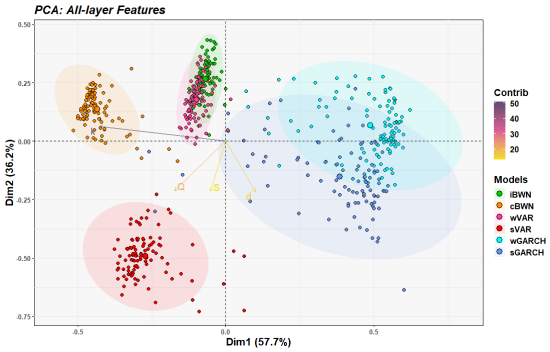
C.4 Principal Component Analysis Results



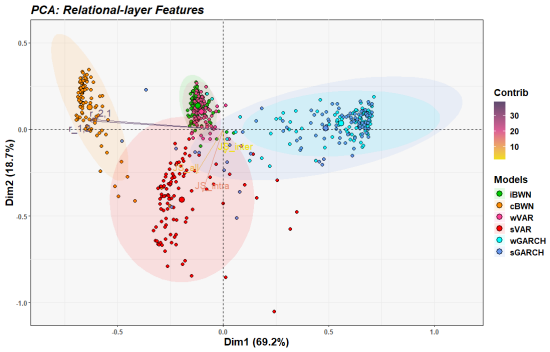
(a) Intra-layer features



(b) Inter-layer features



(c) All-layer features



(d) Relational features

Figure C.3: Bi-plot of the first two principal components (PC) of principal component analysis for the Univariate Data Generating Process (UDGP) using the different feature vectors. Each UDGP is represented by a different color and the arrows represent the contributions of the set of features to the PC's, the larger the size, sharpness, and closer to the red the greater the contribution of the feature.

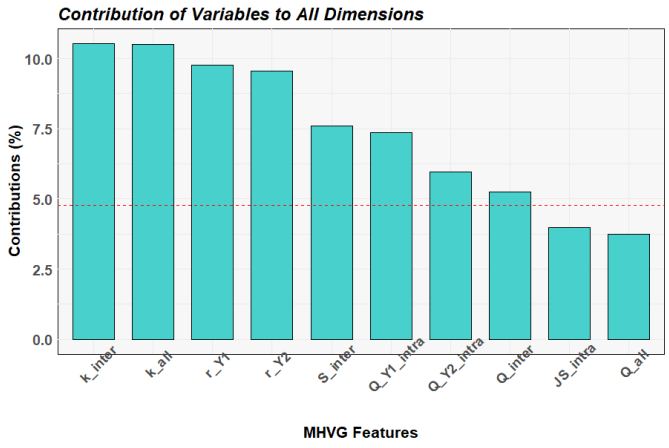
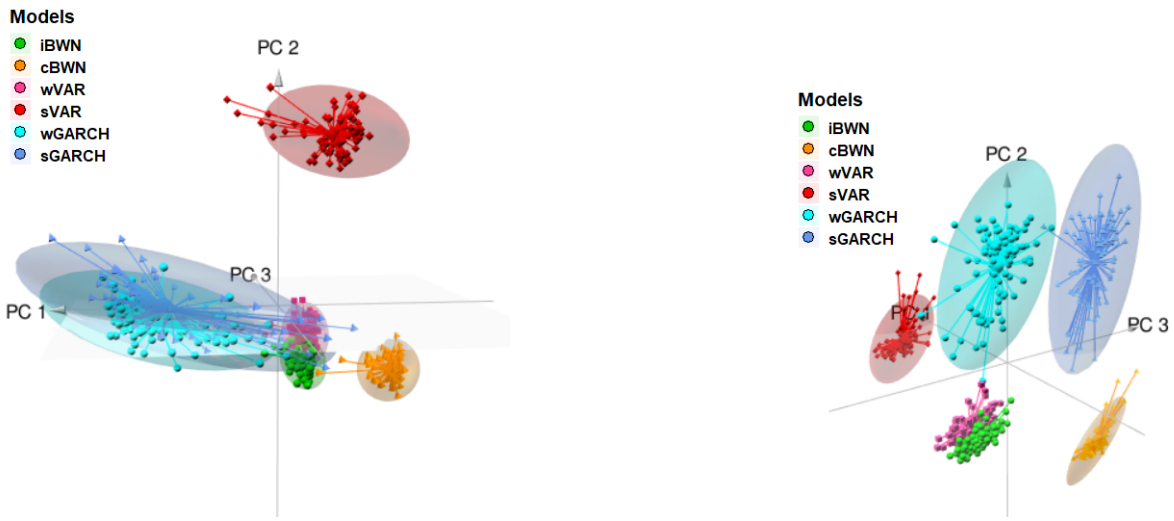


Figure C.4: Bar plot with contributions of MNet features to the total of all 20 principal components formed by the PCA. The red dashed line on the plot indicates the expected average contribution.



(a) MHVG Features

(b) MNetF

Figure C.5: 3D-plot of the first three principal components (PC) of principal component analysis for the Multivariate Data Generating Process (MDGP) using the different feature vectors (MNetF and only to MHVG). Each MDGP is represented by a different color.

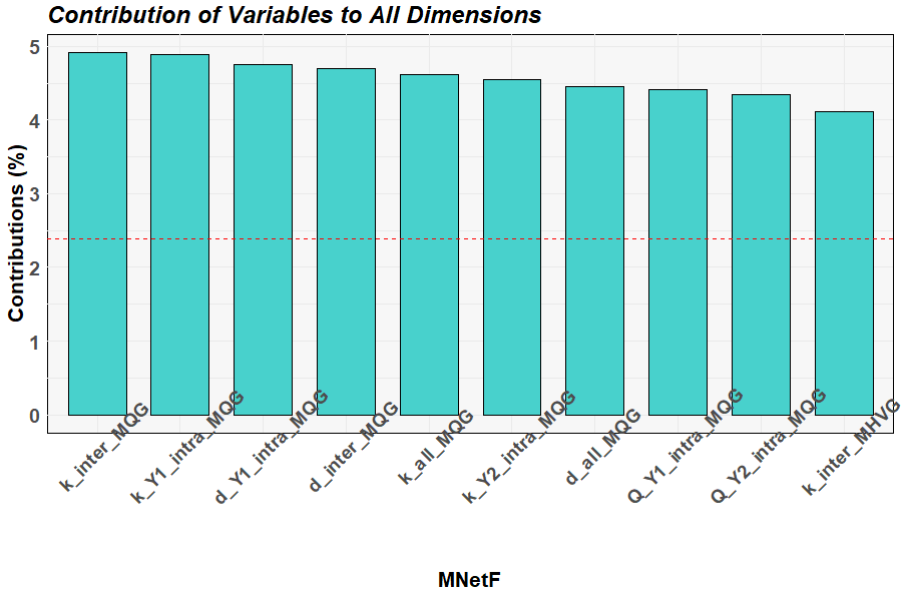


Figure C.6: Bar plot with contributions of MNetF feature set to the total of all 20 principal components formed by the PCA. The red dashed line on the plot indicates the expected average contribution.

Appendix D

NetF: Experimental Evaluation

D.1 Clustering Time Series with *NetF*

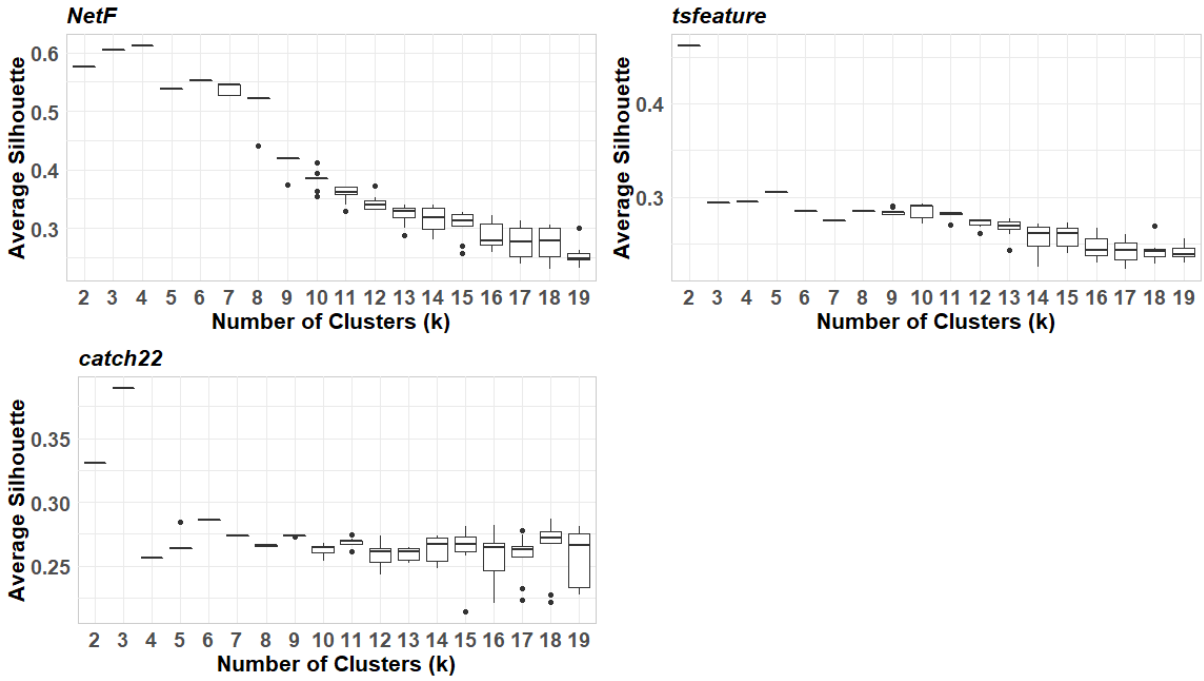


Figure D.1: Number of clusters, k, for the Production in Brazil dataset using the silhouette method for 10 repetitions of the clustering analysis using the 3 features vectors: *NetF*, *catch22* and *tsfeatures*.

Table D.1: Comparison of clustering evaluation using features obtained for the two classical approaches (*tsf.* and *cat.*) and for the proposed approach (*NetF*). The number of clusters, k, is determined automatically by the evaluation metrics.

Dataset	**Num. of Classes**	**Best k**			**ARI** $[-1,1]$			**NMI** $[0,1]$			**AS** $[-1,1]$		
		tsf.	*cat.*	*NetF*	*tsf.*	*cat.*	*NetF*	*tsf.*	*cat.*	*NetF*	*tsf.*	*cat.*	1c | *NetF*
18Pairs	18	21	19	21	**0.51**	0.41	0.39	**0.90**	0.87	0.87	**0.40**	0.37	0.28
M3 data	6	9	9	5	0.14	0.14	0.14	**0.22**	0.21	0.18	**0.31**	0.26	0.26
CinC_ECG_torso	4	4	4	5	0.31	0.32	**0.45**	0.37	0.35	**0.53**	0.23	0.19	**0.31**
Cricket_X	12	9	17	10	0.17	**0.16**	**0.16**	**0.31**	0.30	0.29	**0.20**	0.15	0.10
ECG5000	5	4	2	2	0.35	**0.45**	**0.45**	**0.37**	0.35	0.32	0.25	**0.33**	0.19
ElectricDevices	7	9	3	5	0.21	0.26	**0.28**	0.30	**0.35**	0.32	**0.31**	0.28	0.30
FaceAll	14	23	14	11	0.16	**0.20**	0.15	**0.37**	0.36	0.27	**0.20**	0.15	0.09
FordA	2	4	6	3	**0.21**	0.12	0.13	**0.30**	0.21	0.13	**0.31**	0.14	0.21
InsectWingbeatSound	11	6	11	3	0.09	**0.21**	0.20	0.19	0.36	**0.39**	**0.21**	0.18	0.17
UWaveGestureLibraryAll	8	16	10	6	0.21	0.21	0.21	**0.36**	0.30	0.30	0.19	**0.20**	0.15
Synthetic (DGP)	11	15	8	11	0.79	0.47	**0.92**	0.91	0.69	**0.97**	0.64	0.38	**0.68**

D.2 Clustering Results: UEA & UCR Time Series Datasets

For some sets of benchmark empirical time series, some features of *tsfeatures* and *catch22* approaches return missing values, and some have time series with missing values. We decided not to consider these sets in our clustering analysis as they are just a few. So Tables D.2 to D.6 present the results for 119 sets, out of a total of 129.

Table D.2: Brief description of the empirical time series datasets from UEA & UCR time series repository (Bagnall et al.) and the clustering evaluation metrics obtained for the two conventional approaches (*tsfeatures* and *catch22*) and for the proposed approach (*NetF*). The values reflect the mean of 10 repetitions of the clustering analysis for the ground truth, k. The values in bold represent the best results of the respective evaluation metric comparing the two approaches. M represents the size of dataset, T the time series length and k the number of classes.

Dataset	M	T	k	**ARI** $[-1,1]$			**NMI** $[0,1]$			**AS** $[-1,1]$		
				tsf.	*cat.*	*Net.*	*tsf.*	*cat.*	*Net.*	*tsf.*	*cat.*	*Net.*
ACSF1	200	1460	10	0.22	**0.32**	0.17	0.50	**0.56**	0.40	**0.48**	0.30	0.23
Adiac	781	176	37	**0.21**	**0.21**	0.11	**0.55**	**0.55**	0.43	0.20	**0.26**	0.13
ArrowHead	211	251	3	0.24	0.17	**0.34**	0.27	0.21	**0.31**	0.24	**0.27**	0.15
BME	180	128	3	**0.50**	0.36	0.40	**0.58**	0.36	0.45	**0.55**	0.20	0.23
Beef	60	470	5	**0.09**	0.05	0.04	**0.23**	0.24	0.18	0.33	**0.38**	0.18
BeetleFly	40	512	2	0.55	0.10	**0.63**	0.48	0.11	**0.55**	**0.34**	0.26	0.16
BirdChicken	40	512	2	0.07	0.55	**0.63**	0.10	**0.56**	0.53	**0.35**	0.19	0.29
CBF	930	128	3	0.31	**0.37**	**0.37**	0.34	0.39	**0.40**	**0.22**	0.19	0.13
Car	120	577	4	**0.25**	0.16	0.20	**0.35**	0.23	0.29	0.22	**0.28**	0.18
Chinatown	365	24	2	**0.33**	0.29	-0.05	**0.29**	0.21	0.03	0.23	**0.34**	0.27
ChlorineConcentration	4307	166	3	0.00	0.00	0.00	0.00	0.00	0.00	**0.46**	0.42	0.17
CinCECGTorso	1420	1639	4	0.31	0.32	**0.45**	0.37	0.35	**0.52**	0.23	0.19	**0.31**
Coffee	56	286	2	0.61	**1.00**	0.45	0.54	**1.00**	0.42	**0.32**	0.21	0.13
Computers	500	720	2	0.00	0.00	**0.07**	0.00	0.00	**0.05**	**0.46**	0.17	0.27
CricketX	780	300	12	0.15	0.15	**0.16**	**0.32**	0.28	0.30	**0.20**	0.16	0.10
CricketY	780	300	12	0.13	**0.14**	0.08	0.27	**0.28**	0.21	**0.19**	0.17	0.09
CricketZ	780	300	12	0.15	**0.16**	0.14	**0.33**	0.28	0.27	**0.19**	0.16	0.10
Crop	24000	46	24	**0.19**	0.16	0.09	**0.39**	0.34	0.26	**0.19**	0.18	0.09
DiatomSizeReduction	322	345	4	0.67	0.11	**0.69**	**0.67**	0.26	**0.67**	**0.34**	**0.34**	0.26
DistalPhalanxOutlineAgeG	539	80	3	**0.45**	0.44	0.20	**0.35**	0.34	**0.35**	**0.47**	**0.47**	0.44
DistalPhalanxOutlineCorr	876	80	2	0.00	0.00	**0.01**	0.00	0.00	**0.01**	0.45	0.42	**0.69**
DistalPhalanxTW	539	80	6	0.41	**0.45**	0.28	**0.47**	0.45	0.43	0.31	**0.42**	0.11

Table D.3: (*cont.*) Brief description of the empirical time series datasets from UEA & UCR time series repository (Bagnall et al.) and the clustering evaluation metrics obtained for the two conventional approaches (*tsfeatures* and *catch22*) and for the proposed approach (*NetF*). The values reflect the mean of 10 repetitions of the clustering analysis for the ground truth, k. The values in bold represent the best results of the respective evaluation metric comparing the two approaches. M represents the size of dataset, T the time series length and k the number of classes.

Dataset	M	T	k	**ARI** $[-1,1]$			**NMI** $[0,1]$			**AS** $[-1,1]$		
				tsf.	*cat.*	*Net.*	*tsf.*	*cat.*	*Net.*	*tsf.*	*cat.*	*Net.*
ECG200	200	96	2	**0.25**	0.07	0.03	**0.16**	0.05	0.04	**0.30**	0.19	0.16
ECG5000	5000	140	5	0.29	0.28	**0.31**	**0.32**	0.29	0.30	**0.24**	**0.24**	0.16
ECGFiveDays	884	136	2	**0.02**	0.00	0.00	**0.02**	0.00	0.01	**0.36**	0.30	0.25
ElectricDevices	16575	96	7	0.20	**0.21**	0.19	**0.30**	0.29	0.29	**0.33**	0.25	0.27
EOGHorizontalSignal	724	1250	12	**0.18**	0.16	0.12	**0.38**	0.33	0.27	**0.27**	0.22	0.13
EOGVerticalSignal	724	1250	12	0.10	**0.13**	0.06	0.26	**0.28**	0.17	**0.21**	0.20	0.11
Earthquakes	461	512	2	**0.00**	-0.03	-0.07	0.00	**0.07**	0.04	0.27	0.18	**0.52**
EthanolLevel	1004	1751	4	**0.01**	0.00	**0.01**	0.01	0.00	**0.02**	0.15	**0.27**	0.15
FaceAll	2250	131	14	0.15	**0.21**	0.15	0.33	**0.36**	0.29	**0.22**	0.15	0.09
FaceFour	112	350	4	0.26	0.36	**0.46**	0.32	0.45	**0.54**	**0.39**	0.22	0.23
FacesUCR	2250	131	14	0.14	**0.19**	0.15	0.33	**0.37**	0.28	**0.22**	0.15	0.09
FiftyWords	905	270	50	**0.19**	**0.19**	0.09	**0.57**	0.54	0.44	**0.18**	0.17	0.11
Fish	350	463	7	**0.19**	0.12	0.17	0.29	0.21	**0.30**	0.20	**0.25**	0.13
FordA	4921	500	2	**0.19**	0.01	0.01	**0.27**	0.01	0.01	**0.53**	0.33	0.29
FordB	4446	500	2	**0.27**	0.07	0.02	**0.31**	0.07	0.02	**0.48**	0.29	0.22
FreezerRegularTrain	3000	301	2	0.22	0.27	**0.31**	0.19	0.21	**0.24**	**0.52**	0.50	0.11
FreezerSmallTrain	2878	301	2	0.22	0.27	**0.32**	0.19	0.21	**0.24**	**0.52**	0.50	0.11
Fungi	204	201	18	**0.77**	0.72	0.40	**0.92**	0.90	0.68	**0.48**	0.43	0.13
GestureMidAirD1	338	360	26	0.27	0.25	**0.31**	0.62	0.59	**0.63**	**0.23**	**0.23**	0.15
GestureMidAirD2	338	360	26	0.24	0.25	**0.26**	0.60	0.60	**0.61**	0.21	**0.23**	0.15
GestureMidAirD3	338	360	26	**0.18**	0.14	0.16	**0.52**	0.48	0.50	**0.19**	**0.19**	0.13
GesturePebbleZ1	304	[100, 455]	6	0.15	0.16	**0.23**	0.23	0.22	**0.35**	0.16	0.15	**0.18**
GesturePebbleZ2	304	[100, 455]	6	0.15	0.16	**0.23**	0.23	0.22	**0.35**	0.16	0.15	**0.18**
GunPoint	200	150	2	0.00	0.00	**0.19**	0.06	0.00	**0.30**	**0.70**	0.28	0.27
GunPointAgeSpan	451	150	2	0.01	**0.10**	0.00	0.02	**0.07**	0.00	**0.48**	0.23	0.30
GunPointMaleVersusFema	451	150	2	0.07	0.05	**0.32**	0.13	0.04	**0.37**	**0.48**	0.23	0.30
GunPointOldVersusYoung	451	150	2	0.00	0.08	**0.20**	0.00	0.06	**0.27**	**0.48**	0.23	0.30

Table D.4: (*cont.*) Brief description of the empirical time series datasets from UEA & UCR time series repository (Bagnall et al.) and the clustering evaluation metrics obtained for the two conventional approaches (*tsfeatures* and *catch22*) and for the proposed approach (*NetF*). The values reflect the mean of 10 repetitions of the clustering analysis for the ground truth, k. The values in bold represent the best results of the respective evaluation metric comparing the two approaches. M represents the size of dataset, T the time series length and k the number of classes.

Dataset	M	T	k	**ARI** $[-1,1]$			**NMI** $[0,1]$			**AS** $[-1,1]$		
				tsf.	*cat.*	*Net.*	*tsf.*	*cat.*	*Net.*	*tsf.*	*cat.*	*Net.*
Ham	214	431	2	0.00	0.00	**0.01**	0.00	**0.01**	**0.01**	**0.33**	0.16	0.18
HandOutlines	1370	2709	2	0.05	0.02	**0.06**	0.02	0.00	**0.11**	0.37	0.50	**0.53**
Haptics	463	1092	5	0.04	0.03	**0.07**	0.07	0.07	**0.10**	**0.38**	0.28	0.15
Herring	128	512	2	**0.00**	-0.01	-0.01	0.00	0.00	0.00	**0.22**	0.16	0.17
HouseTwenty	135	3000	2	-0.01	**0.64**	0.10	0.01	**0.61**	0.09	**0.40**	0.21	0.29
InlineSkate	650	1882	7	**0.10**	0.02	0.08	**0.22**	0.07	0.17	**0.29**	0.18	0.16
InsectEPGRegularTrain	311	601	3	0.50	0.43	**0.55**	**0.65**	0.44	0.61	**0.37**	0.22	0.19
InsectEPGSmallTrain	266	601	3	0.50	0.44	**0.52**	**0.65**	0.46	0.61	**0.37**	0.21	0.19
InsectWingbeatSound	2200	256	11	0.07	**0.21**	0.17	0.18	**0.37**	0.32	**0.19**	0.18	0.11
ItalyPowerDemand	1096	24	2	**0.04**	0.01	0.03	**0.05**	0.01	0.03	0.38	**0.40**	0.27
LargeKitchenAppliances	750	720	3	**0.21**	0.06	0.00	**0.23**	0.05	0.01	**0.35**	0.23	0.30
Lightning2	121	637	2	**0.07**	0.02	0.05	**0.14**	0.04	0.07	**0.42**	0.28	0.19
Lightning7	143	319	7	**0.22**	0.21	0.18	**0.39**	**0.39**	0.35	**0.24**	**0.24**	0.14
Mallat	2400	1024	8	**0.70**	0.69	0.53	0.80	**0.83**	0.65	**0.35**	0.32	0.13
Meat	120	448	3	**0.45**	**0.45**	0.17	0.46	**0.63**	0.18	0.29	**0.43**	0.13
MedicalImages	1141	99	10	**0.10**	0.03	0.06	**0.28**	0.19	0.17	**0.31**	0.21	0.17
MiddlePhalanxOutlineAgeG	554	80	3	0.42	**0.43**	0.42	0.39	0.39	0.39	**0.56**	0.49	0.40
MiddlePhalanxOutlineCorr	891	80	2	-0.01	**0.00**	-0.01	**0.01**	0.00	**0.01**	0.47	0.41	**0.71**
MiddlePhalanxTW	553	80	6	0.34	**0.57**	0.24	0.40	**0.43**	0.39	0.28	**0.47**	0.14
MixedShapesRegularTrain	2925	1024	5	0.44	0.23	**0.52**	0.49	0.26	**0.55**	**0.31**	0.20	0.21
MixedShapesSmallTrain	2525	1024	5	0.45	0.23	**0.51**	0.49	0.25	**0.54**	**0.31**	0.20	0.21
MoteStrain	1272	84	2	0.01	0.02	**0.17**	0.01	0.01	**0.17**	**0.48**	0.25	0.20
NonInvasiveFetalECGThor1	3765	750	42	**0.51**	0.27	0.07	**0.76**	0.58	0.30	**0.22**	0.20	0.09
NonInvasiveFetalECGThor2	3765	750	42	**0.54**	0.36	0.11	**0.79**	0.67	0.36	**0.25**	0.23	0.11
OSULeaf	442	427	6	0.29	0.28	**0.49**	0.42	0.38	**0.54**	**0.17**	0.15	0.16
OliveOil	60	570	4	**0.29**	0.18	0.10	**0.36**	0.27	0.17	0.30	**0.32**	0.10

Table D.5: (*cont.*) Brief description of the empirical time series datasets from UEA & UCR time series repository (Bagnall et al.) and the clustering evaluation metrics obtained for the two conventional approaches (*tsfeatures* and *catch22*) and for the proposed approach (*NetF*). The values reflect the mean of 10 repetitions of the clustering analysis for the ground truth, k. The values in bold represent the best results of the respective evaluation metric comparing the two approaches. M represents the size of dataset, T the time series length and k the number of classes.

Dataset	M	T	k	**ARI** $[-1,1]$			**NMI** $[0,1]$			**AS** $[-1,1]$		
				tsf.	*cat.*	*Net.*	*tsf.*	*cat.*	*Net.*	*tsf.*	*cat.*	*Net.*
PLAID	1074	[100, 1344]	11	**0.38**	0.27	0.25	**0.51**	0.41	0.38	**0.31**	0.22	0.18
PhalangesOutlinesCorrect	2658	80	2	**0.01**	**0.01**	0.00	0.00	0.00	0.00	0.33	0.37	**0.70**
Phoneme	2110	1024	39	0.06	0.07	**0.09**	0.29	0.31	**0.34**	0.14	**0.13**	**0.13**
PickupGestureWiimoteZ	100	[29, 361]	10	0.22	0.24	**0.29**	0.50	0.50	**0.55**	0.18	0.18	**0.19**
PigAirwayPressure	312	2000	52	0.11	0.04	**0.13**	0.63	0.57	**0.64**	**0.26**	0.20	0.17
PigArtPressure	312	2000	52	0.40	**0.45**	0.44	0.79	**0.82**	0.81	**0.24**	**0.24**	0.22
PigCVP	312	2000	52	0.13	0.18	**0.20**	0.64	**0.68**	**0.68**	**0.20**	0.14	0.16
Plane	210	144	7	**0.98**	0.86	0.97	**0.98**	0.90	0.97	**0.58**	0.34	0.37
PowerCons	360	144	2	0.13	0.00	**0.25**	0.11	0.00	**0.22**	**0.22**	0.20	**0.22**
ProximalPhalanxOutlineAg	605	80	3	0.55	**0.57**	0.35	0.53	**0.56**	0.45	0.49	0.40	**0.50**
ProximalPhalanxOutlineCo	891	80	2	0.05	**0.06**	0.04	0.07	**0.08**	0.05	0.44	0.49	**0.73**
ProximalPhalanxTW	605	80	6	0.43	**0.47**	0.34	0.56	**0.58**	0.48	0.29	**0.41**	0.15
RefrigerationDevices	750	720	3	0.02	0.02	0.02	0.02	0.02	0.02	**0.41**	0.27	0.27
Rock	70	2844	4	**0.36**	0.16	0.35	**0.51**	0.27	0.49	**0.25**	0.23	0.24
ScreenType	750	720	3	0.03	**0.04**	0.01	0.03	**0.04**	0.01	**0.31**	0.16	0.22
SemgHandGenderCh2	900	1500	2	0.00	0.00	**0.01**	**0.01**	0.00	0.00	0.29	0.23	**0.31**
SemgHandMovementCh2	900	1500	6	0.06	0.04	**0.10**	0.14	0.12	**0.20**	**0.24**	0.23	0.22
SemgHandSubjectCh2	900	1500	5	0.16	**0.26**	0.18	0.25	**0.33**	0.28	0.25	0.24	**0.26**
ShakeGestureWiimoteZ	100	[40, 385]	10	0.54	0.45	**0.56**	0.72	0.67	**0.74**	**0.29**	0.20	0.27
ShapeletSim	200	500	2	0.10	**1.00**	0.85	0.08	**1.00**	0.78	0.18	**0.33**	0.17
ShapesAll	1200	512	60	**0.37**	0.28	0.23	**0.70**	0.65	0.60	**0.26**	0.22	0.13
SmallKitchenAppliances	750	720	3	**0.19**	0.08	0.18	0.19	0.07	**0.20**	0.32	0.25	**0.50**
SonyAIBORobotSurface1	621	70	2	**0.58**	0.42	0.56	**0.53**	0.47	0.47	**0.34**	0.26	0.25
SonyAIBORobotSurface2	980	65	2	**0.55**	0.37	0.00	**0.47**	0.28	0.01	**0.37**	0.22	0.14
StarLightCurves	9236	1024	3	0.49	0.43	**0.65**	**0.57**	0.56	0.53	**0.42**	0.34	0.32
Strawberry	983	235	2	-0.05	-0.02	**0.01**	**0.11**	0.09	0.03	**0.52**	0.35	0.16
SwedishLeaf	1125	128	15	**0.46**	0.31	0.41	**0.68**	0.53	0.60	**0.25**	0.23	0.16
Symbols	1020	398	6	0.69	0.65	**0.77**	0.78	0.79	**0.84**	0.40	**0.49**	0.35
SyntheticControl	600	60	6	0.57	**0.61**	0.43	0.71	**0.74**	0.50	**0.34**	0.20	0.14

Table D.6: (*cont.*) Brief description of the empirical time series datasets from UEA & UCR time series repository (Bagnall et al.) and the clustering evaluation metrics obtained for the two conventional approaches (*tsfeatures* and *catch22*) and for the proposed approach (*NetF*). The values reflect the mean of 10 repetitions of the clustering analysis for the ground truth, k. The values in bold represent the best results of the respective evaluation metric comparing the two approaches. M represents the size of dataset, T the time series length and k the number of classes.

Dataset	M	T	k	**ARI** $[-1,1]$			**NMI** $[0,1]$			**AS** $[-1,1]$		
				tsf.	*cat.*	*Net.*	*tsf.*	*cat.*	*Net.*	*tsf.*	*cat.*	*Net.*
ToeSegmentation1	268	277	2	0.01	0.00	**0.05**	0.01	0.00	**0.05**	0.22	**0.23**	0.20
ToeSegmentation2	166	343	2	0.12	0.07	**0.37**	0.06	0.03	**0.26**	0.30	0.18	**0.37**
Trace	200	275	4	**1.00**	0.73	0.63	**1.00**	0.79	0.70	**0.65**	0.35	0.22
TwoLeadECG	1162	82	2	0.00	0.01	**0.71**	0.00	0.01	**0.61**	**0.60**	0.16	0.19
TwoPatterns	5000	128	4	**0.14**	0.00	0.01	**0.17**	0.00	0.01	**0.16**	0.15	0.11
UMD	180	150	3	**0.48**	0.17	0.35	**0.53**	0.20	0.38	**0.41**	0.24	0.21
UWaveGestureLibraryAll	4478	945	8	0.17	**0.20**	0.18	0.27	**0.28**	**0.28**	**0.20**	0.19	0.12
UWaveGestureLibraryX	4478	315	8	0.18	0.19	**0.23**	0.30	0.29	**0.33**	0.19	**0.20**	0.15
UWaveGestureLibraryY	4478	315	8	**0.22**	0.16	0.14	**0.36**	0.25	0.25	**0.22**	0.20	0.15
Wafer	7164	152	2	-0.02	-0.04	**0.99**	0.00	0.02	**0.96**	**0.56**	0.33	0.51
Wine	111	234	2	**0.03**	-0.01	0.01	**0.03**	0.00	0.01	0.39	**0.43**	0.22
WordSynonyms	905	270	25	**0.14**	0.10	0.05	**0.41**	0.34	0.26	**0.21**	0.18	0.11
Worms	258	900	5	**0.19**	0.14	0.13	**0.24**	0.22	0.22	**0.26**	0.17	0.22
WormsTwoClass	258	900	2	0.04	0.07	**0.12**	0.02	0.04	**0.08**	**0.34**	0.19	0.29
Yoga	3300	426	2	0.00	0.00	0.00	0.00	0.00	0.00	**0.29**	0.20	0.27
Mean	-	-	-	**0.24**	0.22	**0.24**	**0.33**	0.30	0.32	**0.33**	0.26	0.23
Win	-	-	-	46	28	**47**	**48**	28	43	**82**	26	16
Win (%)	-	-	-	38.66	23.53	**39.50**	**40.34**	23.53	36.13	**68.91**	21.85	13.45

Appendix E

An Approximation of Degree Distribution of Cross-HVG under Independence

Let $\boldsymbol{X}_t$ and $\boldsymbol{Y}_t$ two independent bi-infinite time series of i.i.d. random variables with probability density $f(.)$ with the distribution support in a generic interval, i.e., $x, y \in [a, b]$ and consider its associated Cross-HVG. Without loss of generality, we can rescale the distribution support to $[0, 1]$ since the associated graph remains invariant. Following the method presented in the works of Erdős and Rényi (1966); Luque et al. (2009); Zhang and Small (2006) we will find a mathematical expression for degree distribution $P(k)$ of the Cross-HVGs [1].

We select a data $x_0 \in \boldsymbol{X}$ to be the seed. We calculate the probability that an arbitrary data with value x_0 has cross-horizontal visibility of exactly k other data of $\boldsymbol{Y}$. From the definition of Cross-HV presented in Section 4.1.1, whenever a x_0 has cross-horizontal visibility to k data, there are two bounding data ($\max(x_{-j}, y_{-j})$ and $\max(x_i, y_i)$) one on the left side of x_0 and one on the right side, that limit the visibility of x_0 to other data beyond them. Furthermore, the remaining $k-2$ visible data (or even the remaining $k-1$ or k data, since the bounding data can be data of the same random variable as x_0, i.e., x_{-j} or x_i, and therefore are not cross-visible by x_0) are located within the two bounding data. Note that $k = 2$ is the minimum possible degree (see Figure E.1).

We will compute the first terms of the degree distribution of the associated Cross-HVG, namely, $P(k = 2)$ and $P(k = 3)$. Compared to the HVGs of the univariate random time series, the first terms of $P(k)$ of a Cross-HVG are not so trivially calculated due to the cross-horizontal visibility condition that is determined by the maximum of the two random time series. This condition leads to a greater number of possible configurations for the same term $P(k)$ (with $k = 1, 2, \ldots$) and each of the possible configurations can lead to a combination of possible values

[1] Note that the degree distribution of Cross-HVG corresponds to the inter-layer degree distributions $P(k^{\alpha \prec \beta})$ of the MHVG presented in Section 6.1, and that $P(k^{\alpha \prec \beta})$ is asymmetric. For simplicity, in this Appendix, we will just use the notation $P(k)$ to refer to the degree distribution of Cross-HVG.

for the cross-visible and no cross-visible data and for the bounding data that define the value of $P(k)$ term. We will see this in more detail below.

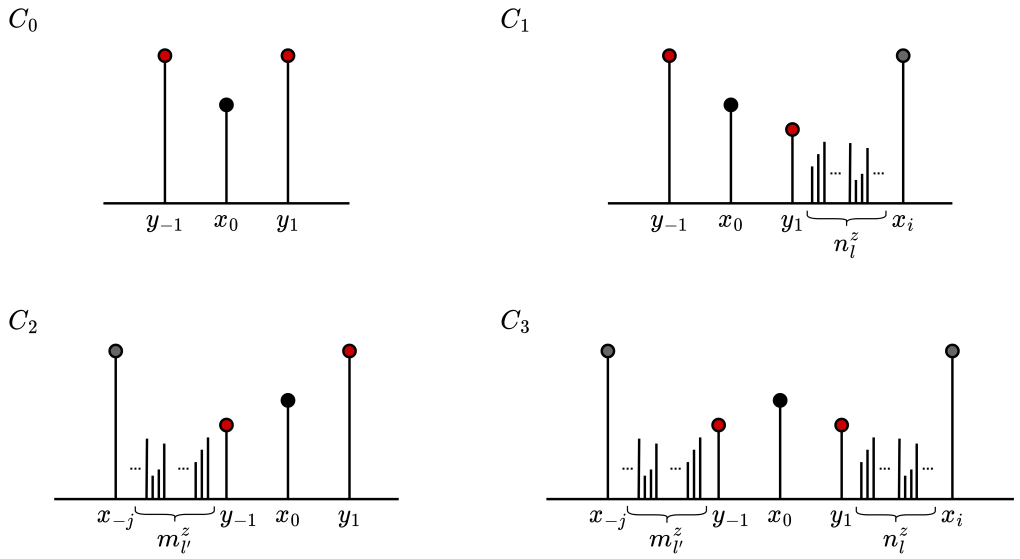


Figure E.1: Set of possible configurations for a seed data x_0 with $k = 2$. The data bars with the red ball represent the (bounding) data cross-visible by x_0 and the data bars with the gray ball represent the bounding data that is not cross-visible by x_0, i.e., $x_i \in \boldsymbol{X}$. Note that the minus sign in the subscript at y_{-t} or x_{-t} indicates that the data is located on the left side of x_0. C_0 represents the trivial configuration where the boundary and cross-visible data are the nearest neighbors of $x0$ and the remaining configurations represent the rarer cases where one (C_1 and C_2) or both (C_3) of the boundary data are not the nearest neighbors and are not cross-visible by x_0. The signs n_l^z and $m_{l'}^z$ indicate the number of hidden data (non cross-visible by x_0).

The probability that x_0 sees $k \geq 2$ is 1 by Cross-HV definition since the Cross-HVG algorithm assures that any data will always have cross-horizontal visibility of its nearest neighbors. For the case $P(k = 2)$, see Figure E.1, we have to impose that there are two boundary data ($\max(x_{-j}, y_{-j})$ and $\max(x_i, y_i)$) and only two cross-visible data (y_{-1} and y_1), this leads to 4 possible configurations (C_0, C_1, C_2 and C_3) which determine the boundary data and which we explain below. To simplicity, from now on we define the maximum time series $\boldsymbol{Z}_t$ such that an arbitrary data have value $z_t = \max(x_t, y_t)$.

C_0: the height of bounding data greater than x_0.

C_1: the height of bounding data greater than x_0, where the left bounding data is z_{-1}, the right bounding data is x_i with $i > 1$ and all y_i are not cross-visible by x_0, and the height of right cross-visible data is smaller that x_0.

C_2: the height of bounding data greater than x_0, where the right bounding data is z_1, the left bounding data is x_{-j} with $j > 1$ and all y_{-j} are not cross-visible by x_0, and the height of left cross-visible data is smaller that x_0.

C_3: the height of bounding data greater than x_0, where the left and right bounding data are x_{-j} and x_i with $j, i > 1$, respectively, and all y_{-j} and y_i are not cross-visible by x_0, and the heights of left and right cross-visible data are smaller that x_0.

Then,

$$\begin{aligned} P(k=2) &= P(C_0) + P(C_1) + P(C_2) + P(C_3) \\ &\equiv p_0 + p_1 + p_2 + p_3. \end{aligned} \tag{E.1}$$

Let's start with the base case C_0 (see Figure E.1),

$$\begin{aligned} p_0 &= Prob(z_{-1}, z_1 \geq x_0) \\ &= \int_0^1 f(x_0)\, dx_0 \int_{x_0}^1 f_{\boldsymbol{Z}}(z_{-1})\, dz_{-1} \int_{x_0}^1 f_{\boldsymbol{Z}}(z_1)\, dz_1. \end{aligned} \tag{E.2}$$

Before we proceed, we highlight some important properties for the calculations. The cumulative probability distribution function $F(x)$ of any probability density $f(x)$ is defined as

$$F(x) = \int_0^x f(x')\, dx', \tag{E.3}$$

where $\frac{d}{dx}F(x) = f(x), F(0) = 0$ and $F(1) = 1$. The relation between f and F holds,

$$f(x)F^{n-1}(x) = \frac{1}{n}\frac{d}{dx}F^n(x). \tag{E.4}$$

By definition, the distribution function of $\boldsymbol{Z}$ (the maximum of two i.i.d random variables) is

$$\begin{aligned} F_{\boldsymbol{Z}}(x) = Prob(z_t \geq x) &= Prob\left(\max(x_t, y_t) \geq x\right) \\ &= Prob(x_t \geq x) Prob(y_t \geq x), \end{aligned} \tag{E.5}$$

using the fact that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independent. However, both $\boldsymbol{X}$ and $\boldsymbol{Y}$ have the same distribution function $F(.)$ and the same density function $f(.)$, this means that

$$F_{\boldsymbol{Z}}(x) = F(x)F(x) = [F(x)]^2. \tag{E.6}$$

The density function of $\boldsymbol{Z}$ can now be found by differentiation,

$$\begin{aligned} f_{\boldsymbol{Z}}(x) = \frac{d}{dx}F_{\boldsymbol{Z}}(x) &= \frac{d}{dx}[F(x)]^2 \\ &= 2F'(x)F(x) \\ &= 2f(x)F(x). \end{aligned} \tag{E.7}$$

Then, using Eqs. E.3, E.4 and E.7, we can rewrite the Eq. E.2:

$$\begin{aligned} p_0 &= \int_0^1 f(x_0)\, dx_0 \int_{x_0}^1 2f(z_{-1})F(z_{-1})\, dz_{-1} \int_{x_0}^1 2f(z_1)F(z_1)\, dz_1 \\ &= \int_0^1 f(x_0)[1 - F^2(x_0)]^2\, dx_0 \\ &= \frac{8}{15}. \end{aligned} \tag{E.8}$$

APPENDIX E. AN APPROXIMATION OF DEGREE DISTRIBUTION OF CROSS-HVG UNDER INDEPENDENCE

We proceed to the remaining configurations. Note that in the configurations C_1, C_2 and C_3 eventually there can be located between the cross-visible data and the no cross-visible bounding data an arbitrary number r of hidden data (i.e., not cross-visible by x_0) $n_1, n_2, \ldots, n_r$, and this fact needs to be taken into account in the probability calculations. However, the geometrical restrictions for this n_l hidden data are not trivial as mentioned earlier. For the sake of simplicity, we define n_l^x if $n_l \in \boldsymbol{X}$, n_l^y if $n_l \in \boldsymbol{Y}$ and n_l^z if $n_l \in \boldsymbol{Z}$. To illustrate this, we take the case C_1 and $r = 2$ which is shown in Figure E.2 with two possible configurations C_1^a and C_1^b. To continue to assure that $k = 2$, in C_1^a and C_1^b the two data n_l^x (bars with yellow ball) are hidden because its belong to the same time series (by definition), the two data n_l^y and y_i (bars with green ball) are hidden data and, consequently, not cross-visible by x_0 because its heights must be less than or equal to some previous maximum data z_j between z_1 (inclusive) and it, restricting the cross-horizontal visibility. For example, the cross-horizontal visibility line of y_i for x_0 is intercepted by n_2^x in C_1^a and by n_1^y in C_1^b. All these possibilities lead to a great combination of possible configurations for the calculations of p_1, p_2 and p_3, and remembering that r can be an arbitrarily large number. Therefore, we decided to calculate a minorant and a majorant value for $P(k = 2)$, being the minorant the value corresponding to p_0 (the base case) and the majorant the total value of $P(k = 2)$ which we calculate below and whose geometric restrictions for the hidden data are limited to the height of x_0, that is, $n_l^z < x_0, l = 1, \ldots, r$ and $m_{l'}^z < x_0, l = 1, \ldots, s$ for C_1, C_2 and C_3. Note that, this restriction allows some configurations that should not be included in the exact calculation of $P(k = 2)$.

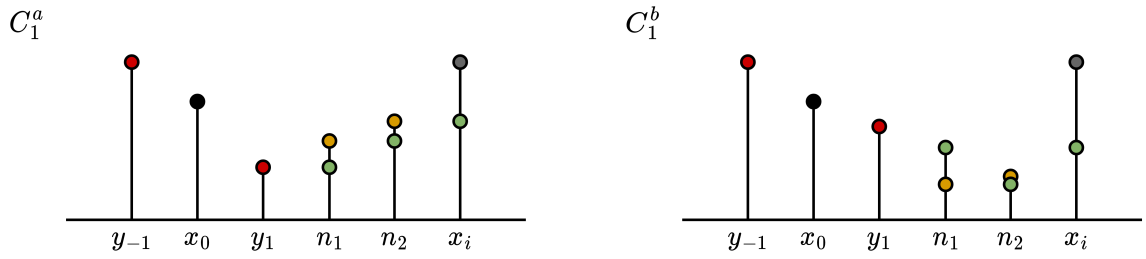


Figure E.2: Two possible example configurations for a seed x_0 with $k = 2$ and two hidden data ($r = 2$). The data bars with the red ball represent the (bounding) data cross-visible by x_0, the data bars with the gray ball represent the bounding data that is not cross-visible by x_0, i.e., $x_i \in \boldsymbol{X}$, the data bars with the yellow ball represent the hidden data that is not cross-visible by x_0 because its belong to the same time series $\boldsymbol{X}$, and the data bars with the green ball represent the hidden data that is not cross-visible by x_0 because there is some previous maximum data z_j that restricts the cross-horizontal visibility between them.

Then,

$$\begin{aligned} p_1 &= Prob\left((z_{-1} \geq x_0) \cap (z_1 < x_0) \cap (x_i \geq x_0) \cap (y_i \leq z_1) \cap (\{n_l^z < x_0\}_{l=1,2,\ldots,r})\right), \\ p_2 &= Prob\left((z_1 \geq x_0) \cap (z_{-1} < x_0) \cap (x_{-j} \geq x_0) \cap (y_{-j} \leq z_1) \cap (\{m_{l'}^z < x_0\}_{l'=1,2,\ldots,s})\right). \end{aligned} \tag{E.9}$$

Note that we need to consider every possible hidden data configuration, that is, C_1 without hidden data, C_1 with a single hidden data, C_1 with two hidden data, and so on, and the same for C_2. Then,

$$\begin{aligned}
p_1 &= \int_0^1 f(x_0)\,dx_0 \int_{x_0}^1 2f(z_{-1})F(z_{-1})\,dz_{-1} \int_0^{x_0} 2f(z_1)F(z_1)\,dz_1 \int_{x_0}^1 f(x_i)\,dx_i \int_0^{z_1} f(y_i)\,dy_i \\
&+ \sum_{r=1}^{\infty} \int_0^1 f(x_0)\,dx_0 \int_{x_0}^1 2f(z_{-1})F(z_{-1})\,dz_{-1} \int_0^{x_0} 2f(z_1)F(z_1)\,dz_1 \int_{x_0}^1 f(x_i)\,dx_i \int_0^{z_1} f(y_i)\,dy_i \\
&\prod_{l=1}^{r} \int_0^{x_0} 2f(n_l^z)F(n_l^z)\,dn_l^z \\
&= \int_0^1 f(x_0)\,dx_0 \int_{x_0}^1 2f(z_{-1})F(z_{-1})\,dz_{-1} \int_0^{x_0} 2f(z_1)F(z_1)\,dz_1 \int_{x_0}^1 f(x_i)\,dx_i \int_0^{z_1} f(y_i)\,dy_i \\
&\sum_{r=0}^{\infty} \left[\int_0^{x_0} 2f(n_l^z)F(n_l^z)\,dn_l^z \right]^r,
\end{aligned} \tag{E.10}$$

where the first term corresponds to the configuration with no hidden data and the second sums up the configurations of r hidden data. Note that p_1 is equal to p_2 since p_1 is symmetric to p_2 (see Figure E.1).

From Eq. E.4 and making e use of the sum of a geometric series we arrive to

$$\begin{aligned}
p_1 &= \int_0^1 \frac{f(x_0)}{[1-F^2(x_0)]}\,dx_0 \int_{x_0}^1 2f(z_{-1})F(z_{-1})\,dz_{-1} \int_0^{x_0} 2f(z_1)F(z_1)\,dz_1 \int_{x_0}^1 f(x_i)\,dx_i \int_0^{z_1} f(y_i)\,dy_i \\
&= \int_0^1 f(x_0) \frac{[1-F^2(x_0)][1-F(x_0)]}{[1-F^2(x_0)]}\,dx_0 \int_0^{x_0} 2f(z_1)F^2(z_1)\,dz_1 \\
&= \frac{2}{3} \int_0^1 f(x_0)[1-F(x_0)]F^3(x_0)\,dx_0 \\
&= \frac{1}{30}.
\end{aligned} \tag{E.11}$$

$$\begin{aligned}
p_3 &= Prob\Big((z_{-1}, z_1 < x_0) \cap (x_i \geq x_0) \cap (y_i \leq z_1) \cap (x_{-j} \geq x_0) \cap (y_{-j} \leq z_{-1}) \cap \\
&\cap (\{n_l^z < x_0\}_{l=1,2,\ldots,r}) \cap (\{m_{l'}^z < x_0\}_{l'=1,2,\ldots,s}) \Big) \\
&= \int_0^1 f(x_0)\,dx_0 \int_0^{x_0} 2f(z_{-1})F(z_{-1})\,dz_{-1} \int_0^{x_0} 2f(z_1)F(z_1)\,dz_1 \int_{x_0}^1 f(x_i)\,dx_i \int_0^{z_1} f(y_i)\,dy_i \\
&\int_{x_0}^1 f(x_{-j})\,dx_{-j} \int_0^{z_{-1}} f(y_{-j})\,dy_{-j} \sum_{r=0}^{\infty} \left[\int_0^{x_0} 2f(n_l^z)F(n_l^z)\,dn_l^z \right]^r \sum_{s=0}^{\infty} \left[\int_0^{x_0} 2f(m_{l'}^z)F(m_{l'}^z)\,dm_{l'}^z \right]^s \\
&= \int_0^1 f(x_0) \frac{[1-F(x_0)]^2}{[1-F^2(x_0)]^2}\,dx_0 \int_0^{x_0} 2f(z_{-1})F^2(z_{-1})\,dz_{-1} \int_0^{x_0} 2f(z_1)F^2(z_1)\,dz_1 \\
&= \frac{4}{9} \int_0^1 f(x_0) \frac{[1-F(x_0)]^2[F^3(x_0)]^2}{[1-F^2(x_0)]^2}\,dx_0 \\
&= \frac{28}{15} - \frac{8}{3}\ln(2).
\end{aligned} \tag{E.12}$$

Finally, we obtain

$$\begin{aligned} P(k=2) &= p_0 + 2p_1 + p_3 \\ &= \frac{37}{15} - \frac{40}{15}\ln(2). \end{aligned} \tag{E.13}$$

Let us proceed by tackling the case $P(k=3)$, that is, the probability that the x_0 has only three cross-visible data. This happens in two ways, when there are 2 cross-visible data on the left side of x_0 and 1 on its right side or there are 2 cross-visible data on the right side of x_0 and 1 on its left side. Figure E.3 shows all possible configurations in which: there are two boundary data ($\max(x_{-j}, y_{-j})$ and $\max(x_i, y_i)$), two (boundary) data cross-visible (y_{-1} and y_2), and a right-hand side inner data cross-visible (y_1), and the same for another way, that is, two boundary data ($\max(x_{-j}, y_{-j})$ and $\max(x_i, y_i)$), two (boundary) data cross-visible (y_{-2} and y_1), and a left-hand side inner data cross-visible (y_{-1}). Note that one or both of the boundary data ($\max(x_{-j}, y_{-j})$ and $\max(x_i, y_i)$) can correspond to the one or both of the (boundary) data cross-visible data (y_{-1} and y_2 or y_{-2} and y_1, respectively), depending on the 6 possible configurations (C_0, C_1, C_2, C_3, C_4 and C_5) shown in Figure E.3.

Thus, in the same way as in case $P(k=2)$ we have

$$\begin{aligned} P(k=3) &= P(C_0) + P(C_1) + P(C_2) + P(C_3) + P(C_4) + P(C_5) \\ &\equiv p_0 + p_1 + p_2 + p_3 + p_4 + p_5. \end{aligned} \tag{E.14}$$

Let's start with the base cases C_0 and C_1, where $p_0 = p_1$ once which are symmetric configurations. So let's proceed with the case C_0. Analyzing Figure E.3, we verify that 2 configurations emerge from C_0, one for the case in which the second cross-visible data by x_0 has value greater than or equal to x_0 ($y_2 \geq x_0$), and therefore is also the boundary data (let's call this configuration C_0^0 with $P(C_0^0) \equiv p_0^0$); and another configuration for the case where the second cross-visible data by x_0 has value smaller than the value of x_0 ($y_2 < x_0$) , and therefore the boundary data is x_2 ($x_2 \geq x_0$), (let's call this configuration C_0^1 with $P(C_0^1) \equiv p_0^1$). Thus, we have

$$p_0 = p_0^0 + p_0^1. \tag{E.15}$$

We calculate first p_0^0 and then p_0^1, using the same properties that we used to calculate the configurations of $P(k=2)$ and properly considering the geometric restrictions for the hidden data.

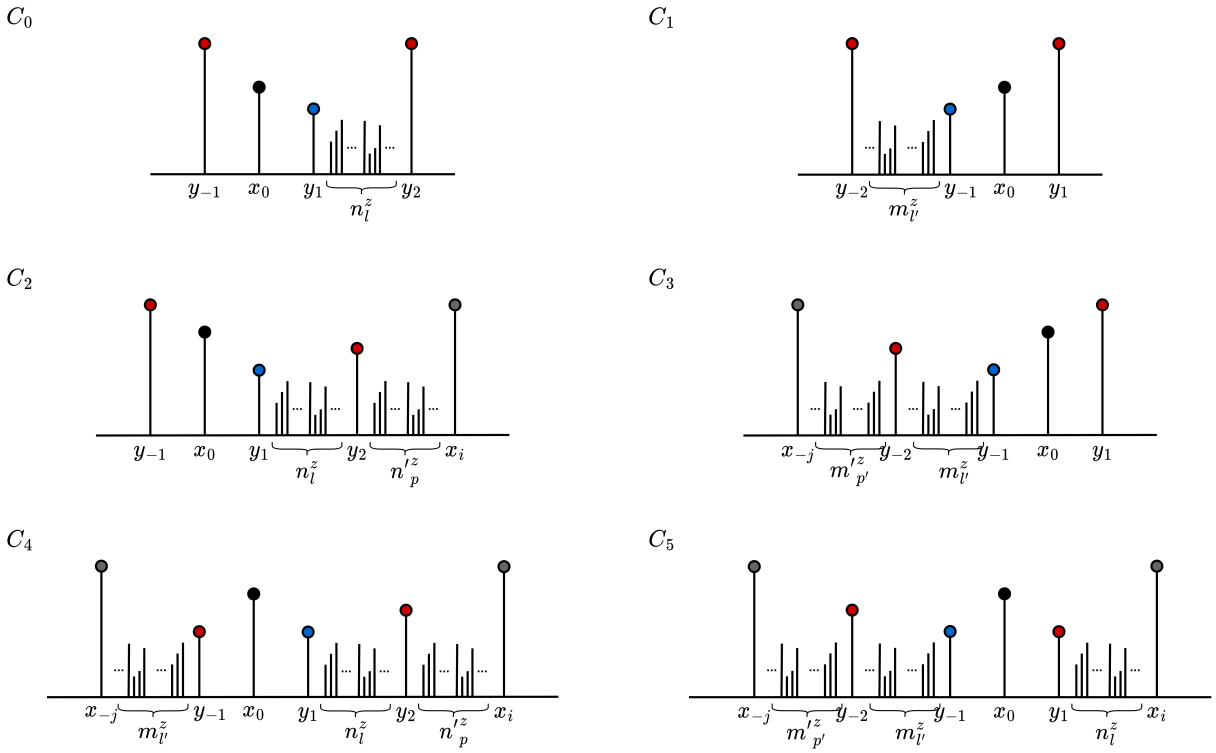


Figure E.3: Set of possible configurations for a seed data x_0 with $k = 3$. The data bars with the red ball represent the (boundary) data cross-visible by x_0, the data bars with the blue ball represent the inner data cross-visible by x_0, and the data bars with the gray ball represent the bounding data that is not cross-visible by x_0, i.e., $x_i \in \boldsymbol{X}_t$. Note that the minus sign in the subscript ate y_{-i} or x_{-i} indicates that the data is located on the left side of x_0. C_0 and C_1 represent the trivial configurations where the boundary and cross-visible data are (y_{-1} and y_2) or (y_{-2} and y_1), respectively, and the remaining configurations represent the rarer cases where one (C_2 and C_3) or both (C_4 and C_5) of the boundary data are not the (boundary) data cross-visible by x_0. The signs $n_l^z, n'^z_p, m_{l'}^z$ and $m'^z_{p'}$ indicate the number of the hidden data (no cross-visible by x_0).

$$
\begin{aligned}
p_0^0 &= Prob\left((z_{-1} \geq x_0) \cap (y_2 \geq x_0) \cap (z_1 < x_0) \cap (\{n_l^z < x_0\}_{l=1,2,\ldots,r})\right) \\
&= \int_0^1 f(x_0)\, dx_0 \int_{x_0}^1 2f(z_{-1})F(z_{-1})\, dz_{-1} \int_{x_0}^1 f(y_2)\, dy_2 \int_0^{x_0} 2f(z_1)F(z_1)\, dz_1 \\
&\sum_{r=0}^{\infty} \left[\int_0^{x_0} 2f(n_l^z)F(n_l^z)\, dn_l^z\right]^r \\
&= \int_0^1 f(x_0) \frac{[1-F^2(x_0)][1-F(x_0)]F^2(x_0)}{[1-F^2(x_0)]}\, dx_0 \\
&= \int_0^1 f(x_0)[F^2(x_0) - F^3(x_0)]\, dx_0 \\
&= \frac{1}{12},
\end{aligned}
\tag{E.16}
$$

$$
\begin{aligned}
p_0^1 &= Prob\left((z_{-1} \geq x_0) \cap (x_2 \geq x_0) \cap (y_2 < x_0) \cap (z_1 < y_2) \cap (\{n_l^z < y_2\}_{l=1,2,\ldots,r})\right) \\
&= \int_0^1 f(x_0)\,dx_0 \int_{x_0}^1 2f(z_{-1})F(z_{-1})\,dz_{-1} \int_{x_0}^1 f(x_2)\,dx_2 \int_0^{x_0} f(y_2)\,dy_2 \int_0^{y_2} 2f(z_1)F(z_1)\,dz_1 \\
&\sum_{r=0}^{\infty} \left[\int_0^{y_2} 2f(n_l^z)F(n_l^z)\,dn_l^z\right]^r \\
&= \int_0^1 f(x_0)[1-F^2(x_0)][1-F(x_0)]\,dx_0 \int_0^{x_0} f(y_2)\frac{F^2(y_2)}{[1-F^2(y_2)]}\,dy_2 \\
&= \frac{1}{2}\int_0^1 f(x_0)[1-F^2(x_0)][1-F(x_0)][-2F(x_0) - \ln(1-F(x_0)) + \ln(1+F(x_0))]\,dx_0 \\
&= \frac{2}{3}\ln(2) - \frac{9}{20}.
\end{aligned}
\tag{E.17}
$$

Finally, we have the value of $P(C_0)$,

$$
p_0 = \frac{2}{3}\ln(2) - \frac{11}{30}. \tag{E.18}
$$

We now proceed to C_2 and C_3, where $p_2 = p_3$ since C_2 and C_3 are symmetric.

$$
\begin{aligned}
p_2 &= Prob\left((z_{-1} \geq x_0) \cap (z_1 < y_2) \cap (\{n_l^z < y_2\}_{l=1,2,\ldots,r}) \cap (x_2 < x_0) \cap (y_2 < x_0) \cap\right. \\
&\left. \cap (x_i \geq x_0) \cap (y_i \leq y_2) \cap (\{n'^z_p < x_0\}_{p=1,2,\ldots,s})\right) \\
&= \int_0^1 f(x_0)\,dx_0 \int_{x_0}^1 2f(z_{-1})F(z_{-1})\,dz_{-1} \int_0^{y_2} 2f(z_1)F(z_1)\,dz_1 \int_0^{x_0} f(x_2)\,dx_2 \int_0^{x_0} f(y_2)\,dy_2 \\
&\int_{x_0}^1 f(x_i)\,dx_i \int_0^{y_2} f(y_i)\,dy_i \sum_{r=0}^{\infty} \left[\int_0^{y_2} 2f(n_l^z)F(n_l^z)\,dn_l^z\right]^r \sum_{s=0}^{\infty} \left[\int_0^{x_0} 2f(n'^z_p)F(n'^z_p)\,dn'^z_p\right]^s \\
&= \int_0^1 f(x_0)\frac{[1-F^2(x_0)][1-F(x_0)]F(x_0)}{[1-F^2(x_0)]}\,dx_0 \int_0^{x_0} f(y_2)\frac{F^3(y_2)}{[1-F^2(y_2)]}\,dy_2 \\
&= \int_0^1 f(x_0)[F(x_0) - F^2(x_0)]\left[\frac{1}{2}[-F^2(x_0) - \ln(1-F^2(x_0))]\right]\,dx_0 \\
&= -\frac{79}{360} + \frac{1}{3}\ln(2).
\end{aligned}
\tag{E.19}
$$

Lastly, we compute C_4 and C_5, with $p_4 = p_5$ (C_4 and C_5 are symmetric).

$$
\begin{aligned}
p_4 &= Prob\Big((z_{-1} < x_0) \cap (x_{-j} \geq x_0) \cap (y_{-j} \leq z_{-1}) \cap (\{m^z_{l'} < x_0\}_{l'=1,2,\dots,q}) \cap (z_1 < y_2) \cap \\
&\cap (x_2 < x_0) \cap (y_2 < x_0) \cap (x_i \geq x_0) \cap (y_i \leq y_2) \cap (\{n^z_l < y_2\}_{l=1,2,\dots,r}) \cap (\{n'^z_p < x_0\}_{p=1,2,\dots,s})\Big) \\
&= \int_0^1 f(x_0)\,dx_0 \int_0^{x_0} 2f(z_{-1})F(z_{-1})\,dz_{-1} \int_0^{y_2} 2f(z_1)F(z_1)\,dz_1 \int_{x_0}^1 f(x_{-j})\,dx_{-j} \int_0^{z_{-1}} f(y_{-j})\,dy_{-j} \\
&\int_0^{x_0} f(x_2)\,dx_2 \int_0^{x_0} f(y_2)\,dy_2 \int_{x_0}^1 f(x_i)\,dx_i \int_0^{y_2} f(y_i)\,dy_i \sum_{r=0}^{\infty} \left[\int_0^{y_2} 2f(n^z_l)F(n^z_l)\,dn^z_l\right]^r \\
&\sum_{s=0}^{\infty} \left[\int_0^{x_0} 2f(n'^z_p)F(n'^z_p)\,dn'^z_p\right]^s \sum_{q=0}^{\infty} \left[\int_0^{x_0} 2f(m^z_{l'})F(m^z_{l'})\,dm^z_{l'}\right]^q \\
&= \frac{2}{3}\int_0^1 f(x_0)\frac{[1-F(x_0)]^2F^4(x_0)}{[1-F^2(x_0)]^2}\,dx_0 \int_0^{x_0} f(y_2)\frac{F^3(y_2)}{[1-F^2(y_2)]}\,dy_2 \\
&= \frac{1}{3}\int_0^1 f(x_0)\frac{[1-F(x_0)]^2F^4(x_0)}{[1-F^2(x_0)]^2}\left[-F^2(x_0) - \ln(1-F^2(x_0))\right]\,dx_0 \\
&= \frac{107}{270} - \frac{\pi^2}{9} + \frac{1}{9}\ln(2) + \frac{4}{3}\ln^2(2).
\end{aligned}
\tag{E.20}
$$

Finally, we obtain

$$
\begin{aligned}
P(k=3) &= 2p_0 + 2p_2 + 2p_4 \\
&= -\frac{41}{108} - \frac{2}{9}\pi^2 + \frac{20}{9}\ln(2) + \frac{8}{3}\ln^2(2).
\end{aligned}
\tag{E.21}
$$

As the values obtained are minorant and majorant from $P(k)$ to $k = 2, 3$ we can write the following

$$
\begin{aligned}
p_0 < P(k=2) &< (p_0 + 2p_1 + p_3) \\
0.5333 \lessapprox P(k=2) &\lessapprox 0.6183,
\end{aligned}
\tag{E.22}
$$

where C_0 is the base case of $P(k=2)$, and

$$
\begin{aligned}
2p_0 < P(k=3) &< (2p_0 + 2p_2 + 2p_4) \\
0.1909 \lessapprox P(k=3) &\lessapprox 0.2487,
\end{aligned}
\tag{E.23}
$$

where C_0 and C_1 is the base case of $P(k=3)$.

In order to validate these results, we performed an experimental study with 1000 repetitions of bivariate white noise time series (independents) of length $T = 10000$. For each bivariate time series, we generate the corresponding Cross-HVG and we obtain the associated degree distributions $P(k)$. A statistical summary of the $P(k=2)$ and $P(k=3)$ values corresponding to the 1000 repetitions can be seen in Table E.1. We can verify that the values are within the limits obtained analytically.

Table E.1: Statistical summary of the $P(k = 2)$ and $P(k = 3)$ values corresponding to the 1000 repetitions of bivariate white noise time series (independents).

		Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
$P(k = 2)$	$\boldsymbol{X}_t$	0.5736	0.5900	0.5996	0.6062	0.6169	0.7086
	$\boldsymbol{Y}_t$	0.5739	0.5896	0.5988	0.6060	0.6150	0.7342
$P(k = 3)$	$\boldsymbol{X}_t$	0.1644	0.2018	0.2133	0.2137	0.2250	0.2571
	$\boldsymbol{Y}_t$	0.1514	0.2023	0.2139	0.2137	0.2258	0.2614

Bibliography

Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering–a decade review. *Information systems*, 53:16–38, 2015.

Mehran Ahmadlou, Hojjat Adeli, and Anahita Adeli. New diagnostic EEG markers of the alzheimer's disease using visibility graph. *Journal of Neural Transmission*, 117(9):1099–1109, 2010.

Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47, 2002.

José María Amigó, Samuel Zambrano, and Miguel AF Sanjuán. True and false forbidden patterns in deterministic and random dynamics. *EPL (Europhysics Letters)*, 79(5):50001, 2007.

Panfeng An, Weixin Si, Sirui Ding, Guotong Xue, and Zhiyong Yuan. A novel EEG sleep staging method for wearable devices based on amplitude-time mapping. In *2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 124–129. IEEE, 2019.

Rodolfo Baggio and Ruggero Sainaghi. Mapping time series into networks as a tool to assess the complex dynamics of tourism systems. *Tourism Management*, 54:23–33, 2016.

A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31:606–660, 2017.

Anthony Bagnall, Jason Lines, William Vickers, and Eamonn Keogh. The UEA & UCR time series classification repository. www.timeseriesclassification.com.

Francisco J Baldán, Daniel Peralta, Yvan Saeys, and José M Benítez. SCMFTS: Scalable and distributed complexity measures and features for univariate and multivariate time series in big data environments. *International Journal of Computational Intelligence Systems*, 14(1):1–15, 2021.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286 (5439):509–512, 1999.

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

Marília Barandas, Duarte Folgado, Letícia Fernandes, Sara Santos, Mariana Abreu, Patrícia Bota, Hui Liu, Tanja Schultz, and Hugo Gamboa. Tsfel: Time series feature extraction library. *SoftwareX*, 11:100456, 2020. doi:10.1016/j.softx.2020.100456.

S. M. Barbosa. *mAr: Multivariate AutoRegressive analysis*, 2012. R package version 1.1-2.

Matteo Barigozzi and Christian Brownlees. Nets: Network estimation for time series. *Journal of Applied Econometrics*, 34(3):347–364, 2019.

Matteo Barigozzi and Marc Hallin. Generalized dynamic factor models and volatilities: recovering the market volatility shocks, 2016.

Matteo Barigozzi and Marc Hallin. A network analysis of the volatility of high dimensional financial series. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 66(3):581–605, 2017.

IV Bezsudnov and AA Snarskii. From the time series to the complex networks: The parametric natural visibility graph. *Physica A: Statistical Mechanics and its Applications*, 414:53–60, 2014.

Filippo Maria Bianchi, Lorenzo Livi, Cesare Alippi, and Robert Jenssen. Multiplex visibility graphs to investigate recurrent neural network dynamics. *Scientific Reports*, 7:44037, 2017.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

Stefano Boccaletti, Ginestra Bianconi, Regino Criado, Charo I Del Genio, Jesús Gómez-Gardenes, Miguel Romance, Irene Sendina-Nadal, Zhen Wang, and Massimiliano Zanin. The structure and dynamics of multilayer networks. *Physics Reports*, 544(1):1–122, 2014.

Giovanni Bonanno, Guido Caldarelli, Fabrizio Lillo, Salvatore Micciche, Nicolas Vandewalle, and Rosario Nunzio Mantegna. Networks of equities in financial markets. *The European Physical Journal B*, 38(2):363–371, 2004.

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 2015.

Elizabeth Bradley and Holger Kantz. Nonlinear time-series analysis revisited. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 25(9):097610, 2015.

Ed Bullmore and Olaf Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10(3):186, 2009.

Shi-Min Cai, Yan-Bo Zhou, Tao Zhou, and Pei-Ling Zhou. Hierarchical organization and disassortative mixing of correlation-based weighted financial networks. *International Journal of Modern Physics C*, 21(03):433–441, 2010.

Andriana SLO Campanharo and Fernando Ramos. Distinguishing different dynamics in electroencephalographic time series through a complex network approach. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, 5(1), 2017.

Andriana SLO Campanharo and Fernando M Ramos. Hurst exponent estimation of self-affine time series using quantile graphs. *Physica A: Statistical Mechanics and its Applications*, 444: 43–48, 2016. doi:10.1016/j.physa.2015.09.094.

Andriana SLO Campanharo, M Irmak Sirer, R Dean Malmgren, Fernando M Ramos, and Luís A Nunes Amaral. Duality between time series and networks. *PLOS ONE*, 6(8):e23378, 2011.

Andriana SLO Campanharo, Erwin Doescher, and Fernando M Ramos. Application of quantile graphs to the automated analysis of EEG signals. *Neural Processing Letters*, 52:5–20, 2018.

Fernando Manuel Campanharo, Andriana SLO. Quantile graphs for the characterization of chaotic dynamics in time series. In *WCCS 2015 – IEEE Third World Conference on Complex Systems*. IEEE, 2016.

Yinhe Cao, Wen-wen Tung, JB Gao, Vladimir A Protopopescu, and Lee M Hively. Detecting dynamical changes in time series using the permutation entropy. *Physical Review E*, 70(4): 046217, 2004.

AK Charakopoulos, TE Karakasidis, PN Papanicolaou, and A Liakopoulos. The application of complex network time series analysis in turbulent heated jets. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 24(2):024408, 2014.

Dong-Rui Chen, Chuang Liu, Yi-Cheng Zhang, and Zi-Ke Zhang. Predicting financial extremes based on weighted visual graph of major stock indices. *Complexity*, 2019, 2019.

Bill Chiu, Eamonn Keogh, and Stefano Lonardi. Probabilistic discovery of time series motifs. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 493–498, 2003.

Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*, 307:72–77, 2018. doi:10.1016/j.neucom.2018.03.067.

Tomas Cipra. *Time series in economics and finance*. Springer, Wiesbaden, Deutschland, 2020. doi:10.1007/978-3-030-46347-2.

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.

Luciano da Fontoura Costa, Osvaldo N Oliveira Jr, Gonzalo Travieso, Francisco Aparecido Rodrigues, Paulino Ribeiro Villas Boas, Lucas Antiqueira, Matheus Palhares Viana, and Luis Enrique Correa Rocha. Analyzing and modeling real-world phenomena with complex networks: A survey of applications. *Advances in Physics*, 60(3):329–412, 2011.

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

Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.

Maria Elena De Giuli, Andrea Flori, Daniela Lazzari, and Alessandro Spelta. Brexit news propagation in financial systems: multidimensional visibility networks for market volatility dynamics. *Quantitative Finance*, 22(5):973–995, 2022.

Mark E Dickison, Matteo Magnani, and Luca Rossi. *Multilayer social networks*. Cambridge University Press, 2016.

Francis X Diebold and Kamil Yılmaz. On the network topology of variance decompositions: Measuring the connectedness of financial firms. *Journal of Econometrics*, 182(1):119–134, 2014.

Henk A Dijkstra, Emilio Hernández-García, Cristina Masoller, and Marcelo Barreiro. *Networks in Climate*. Cambridge University Press, 2019.

Sergii Domanskyi, Carlo Piermarocchi, and George I Mias. Pyiomica: longitudinal omics analysis and trend identification. *Bioinformatics*, 36(7):2306–2307, 2020.

Jonathan F Donges, Hanna CH Schultz, Norbert Marwan, Yong Zou, and Jürgen Kurths. Investigating the topology of interacting networks. *The European Physical Journal B*, 84(4): 635–651, 2011a.

Jonathan F Donges, Jobst Heitzig, Reik V Donner, and Jürgen Kurths. Analytical framework for recurrence network analysis of time series. *Physical Review E*, 85(4):046105, 2012.

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

Jonathan F Donges, Jobst Heitzig, Boyan Beronov, Marc Wiedermann, Jakob Runge, Qing Yi Feng, Liubov Tupikina, Veronika Stolbova, Reik V Donner, Norbert Marwan, et al. Unified functional network and nonlinear time series analysis for complex systems science: The pyunicorn package. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 25(11):113101, 2015.

Jonathan Friedmann Donges, Reik Volker Donner, K Rehfeld, Norbert Marwan, Martin H Trauth, and Jürgen Kurths. Identification of dynamical transitions in marine palaeoclimate records by recurrence network analysis. *Nonlinear Processes in Geophysics*, 18(5):545–562, 2011b.

Reik V Donner and Jonathan F Donges. Visibility graph analysis of geophysical time series: Potentials and possible pitfalls. *Acta Geophysica*, 60(3):589–623, 2012.

Reik V Donner, Yong Zou, Jonathan F Donges, Norbert Marwan, and Jürgen Kurths. Recurrence networks — a novel paradigm for nonlinear time series analysis. *New Journal of Physics*, 12(3): 033025, 2010.

Reik V Donner, Jobst Heitzig, Jonathan F Donges, Yong Zou, Norbert Marwan, and Jurgen Kurths. The geometry of chaotic dynamics—a complex network perspective. *The European Physical Journal B*, 84(4):653–672, 2011a.

Reik V Donner, Michael Small, Jonathan F Donges, Norbert Marwan, Yong Zou, Ruoxi Xiang, and Jürgen Kurths. Recurrence-based time series analysis by means of complex network methods. *International Journal of Bifurcation and Chaos*, 21(04):1019–1046, 2011b.

Reik V Donner, Jonathan F Donges, Yong Zou, and Jan H Feldhoff. Complex network analysis of recurrences. In *Recurrence Quantification Analysis*, pages 101–163. Springer, 2015.

Randal Douc, Eric Moulines, and David Stoffer. *Nonlinear Time Series: Theory, Methods and Applications with R Examples*. Chapman and Hall/CRC, 1 edition, 2014.

JP Eckmann, S Oliffson Kamphorst, and D Ruelle. Recurrence plots of dynamical systems. *Europhysics Letters (EPL)*, 4(9):973—-977, 1987.

Victor M Eguiluz, Dante R Chialvo, Guillermo A Cecchi, Marwan Baliki, and A Vania Apkarian. Scale-free brain functional networks. *Physical Review Letters*, 94(1):018102, 2005.

JB Elsner, TH Jagger, and EA Fogarty. Visibility network of united states hurricanes. *Geophysical Research Letters*, 36(16):L16702, 2009.

Sacha Epskamp and Eiko I Fried. A tutorial on regularized partial correlation networks. *Psychological Methods*, 23(4):617, 2018.

P Erdős and Alfréd Rényi. On the existence of a factor of degree one of a connected random graph. *Acta Math. Acad. Sci. Hungar*, 17:359–368, 1966.

Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

Deniz Eroglu, Norbert Marwan, Martina Stebich, and Jürgen Kurths. Multiplex recurrence networks. *Physical Review E*, 97(1):012312, 2018.

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

Jan H Feldhoff, Reik V Donner, Jonathan F Donges, Norbert Marwan, and Jürgen Kurths. Geometric detection of coupling directions by means of inter-system recurrence networks. *Physics Letters A*, 376(46):3504–3513, 2012.

JH Feldhoff, Reik V Donner, Jonathan F Donges, Norbert Marwan, and Jürgen Kurths. Geometric signature of complex synchronisation scenarios. *EPL (Europhysics Letters)*, 102(3):30007, 2013.

Chen Feng and Bo He. Construction of complex networks from time series based on the cross correlation interval. *Open Physics*, 15(1):253–260, 2017.

Leonardo N. Ferreira. From time series to networks in r with the ts2net package, 2022. doi:10.48550/ARXIV.2208.09660.

Ryan Flanagan, Lucas Lacasa, and Vincenzo Nicosia. On the spectral properties of Feigenbaum graphs. *Journal of Physics A: Mathematical and Theoretical*, 53(2):025702, 2019.

Andrea Flori, Fabio Pammolli, and Alessandro Spelta. Commodity prices co-movements and financial stability: A multidimensional visibility nexus with climate conditions. *Journal of Financial Stability*, 54:100876, 2021.

Miwa Fukino, Yoshito Hirata, and Kazuyuki Aihara. Coarse-graining time series data: Recurrence plot of recurrence plots and its application for music. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 26(2):023116, 2016.

Ben D Fulcher. Feature-based time-series analysis. In *Feature engineering for machine learning and data analytics*, pages 87–116. CRC Press, 2018.

Ben D Fulcher and Nick S Jones. Highly comparative feature-based time-series classification. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3026–3037, 2014.

Ben D Fulcher and Nick S Jones. hctsa: A computational framework for automated time-series phenotyping using massive feature extraction. *Cell systems*, 5(5):527–531, 2017.

Ben D Fulcher, Max A Little, and Nick S Jones. Highly comparative time-series analysis: the empirical structure of time series and their methods. *Journal of the Royal Society Interface*, 10 (83):20130048, 2013.

Ya-Chun Gao, Yong Zeng, and Shi-Min Cai. Influence network in the chinese stock market. *Journal of Statistical Mechanics: Theory and Experiment*, 2015(3):P03017, 2015.

Zhong-Ke Gao, Xin-Wang Zhang, Ning-De Jin, Norbert Marwan, and Jürgen Kurths. Multivariate recurrence network analysis for characterizing horizontal oil-water two-phase flow. *Physical Review E*, 88(3):032910, 2013.

Zhong-Ke Gao, Qing Cai, Yu-Xuan Yang, Wei-Dong Dang, and Shan-Shan Zhang. Multiscale limited penetrable horizontal visibility graph for analyzing nonlinear time series. *Scientific Reports*, 6(1):35622, 2016a.

Zhong-Ke Gao, Yu-Xuan Yang, Qing Cai, Shan-Shan Zhang, and Ning-De Jin. Multivariate weighted recurrence network inference for uncovering oil-water transitional flow behavior in a vertical pipe. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 26(6):063117, 2016b.

Zhong-Ke Gao, Wei Guo, Qing Cai, Chao Ma, Yuan-Bo Zhang, and Jürgen Kurths. Characterization of ssmvep-based EEG signals using multiplex limited penetrable horizontal visibility graph. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(7):073119, 2019a.

Zhong-Ke Gao, Yan-Li Li, Yu-Xuan Yang, and Chao Ma. A recurrence network-based convolutional neural network for fatigue driving detection from EEG. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(11):113126, 2019b.

Zhongke Gao and Ningde Jin. Complex network from time series based on phase space reconstruction. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 19(3):033137, 2009.

Pierre Geurts. Pattern extraction for time series classification. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 115–127. Springer, 2001.

Subir Kumar Ghosh. *Visibility Algorithms in the Plane*. Cambridge university press, 2007.

Daniel Granato, Jânio S Santos, Graziela B Escher, Bruno L Ferreira, and Rubén M Maggio. Use of principal component analysis (pca) and hierarchical cluster analysis (hca) for multivariate association between bioactive compounds and functional properties in foods: A critical perspective. *Trends in Food Science & Technology*, 72:83–90, 2018.

Heng Guo, Jia-Yang Zhang, Yong Zou, and Shu-Guang Guan. Cross and joint ordinal partition transition networks for multivariate time series analysis. *Frontiers of Physics*, 13(5):130508, 2018.

Gregory Gutin, Toufik Mansour, and Simone Severini. A characterization of horizontal visibility graphs and combinatorics on words. *Physica A: Statistical Mechanics and its Applications*, 390 (12):2421–2428, 2011.

J.A. Hartigan and M.A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

Trent Henderson. *Rcatch22: Calculation of 22 CAnonical Time-Series CHaracteristics*, 2021. R package version 0.1.13.

Trent Henderson and Ben D. Fulcher. An empirical evaluation of time-series feature sets, 2021.

Petter Holme and Jari Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, 2012.

FZ Hou, FW Li, J Wang, and FR Yan. Visibility graph analysis of very short-term heart rate variability during sleep. *Physica A: Statistical Mechanics and its Applications*, 458:140–145, 2016.

Xinyu Huang, Dongming Chen, Tao Ren, and Dongqi Wang. A survey of community detection methods in multilayer networks. *Data Mining and Knowledge Discovery*, 35(1):1–45, 2021a.

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, 2021b.

Rob Hyndman. *Mcomp: Data from the M-Competitions*, 2018. R package version 2.8.

Rob Hyndman, Yanfei Kang, Pablo Montero-Manso, Thiyanga Talagala, Earo Wang, Yangzhuoran Yang, and Mitchell O'Hara-Wild. *tsfeatures: Time Series Feature Extraction*, 2020. R package version 1.0.2.

Rob J Hyndman and Yanan Fan. Sample quantiles in statistical packages. *The American Statistician*, 50(4):361–365, 1996. doi:10.1080/00031305.1996.10473566.

Rob J Hyndman, Earo Wang, and Nikolay Laptev. Large-scale unusual time series detection. In *2015 IEEE international conference on data mining workshop (ICDMW)*, pages 1616–1619. IEEE, 2015.

Jacopo Iacovacci and Lucas Lacasa. Sequential motif profile of natural visibility graphs. *Physical Review E*, 94(5):052309, 2016a.

Jacopo Iacovacci and Lucas Lacasa. Sequential visibility-graph motifs. *Physical Review E*, 93(4): 042309, 2016b.

IBGE. Instituto Brasileiro de Geografia e Estatística – IBGE. https://www.ibge.gov.br.

Koji Iwayama, Yoshito Hirata, Hideyuki Suzuki, and Kazuyuki Aihara. Change-point detection with recurrence networks. *Nonlinear Theory and Its Applications, IEICE*, 4(2):160–171, 2013.

Rinku Jacob, KP Harikrishnan, R Misra, and G Ambika. Can recurrence networks show small-world property? *Physics Letters A*, 380(35):2718–2723, 2016.

Rinku Jacob, KP Harikrishnan, R Misra, and G Ambika. Weighted recurrence networks for the analysis of time-series data. *Proceedings of the Royal Society A*, 475(2221):20180256, 2019.

Yanfei Kang, Rob J Hyndman, and Kate Smith-Miles. Visualising forecasting algorithm performance using time series instance spaces. *International Journal of Forecasting*, 33(2): 345–358, 2017.

Yanfei Kang, Rob J Hyndman, and Feng Li. GRATIS: GeneRAting TIme Series with diverse and controllable characteristics. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 2020.

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

L Douglas Kiel and Euel W Elliott. *Chaos Theory in the Social Sciences: Foundations and Applications*. University of Michigan Press, 1996.

Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. Multilayer networks. *Journal of Complex Networks*, 2(3):203–271, 2014.

Marlene Kretschmer, Dim Coumou, Jonathan F Donges, and Jakob Runge. Using causal effect networks to analyze different arctic drivers of midlatitude winter circulation. *Journal of Climate*, 29(11):4069–4081, 2016.

Christopher W Kulp, Jeremy M Chobot, Helena R Freitas, and Gene D Sprechini. Using ordinal partition transition networks to analyze ECG data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 26(7):073114, 2016.

Lucas Lacasa and Raul Toral. Description of stochastic and chaotic series using visibility graphs. *Physical Review E*, 82(3):036120, 2010.

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.

Lucas Lacasa, Bartolo Luque, Jordi Luque, and Juan Carlos Nuno. The visibility graph: A new method for estimating the hurst exponent of fractional brownian motion. *EPL (Europhysics Letters)*, 86(3):30001, 2009.

Lucas Lacasa, Angel Nunez, Édgar Roldán, Juan MR Parrondo, and Bartolo Luque. Time series irreversibility: a visibility graph approach. *The European Physical Journal B*, 85(6):217, 2012.

Lucas Lacasa, Vincenzo Nicosia, and Vito Latora. Network structure of multivariate time series. *Scientific Reports*, 5(1):15508, 2015.

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.

Markku Lanne and Henri Nyberg. Generalized forecast error variance decomposition for linear and nonlinear multivariate models. *Oxford Bulletin of Economics and Statistics*, 78(4):595–603, 2016.

Daoyuan Li, Jessica Lin, Tegawendé François D Assise Bissyande, Jacques Klein, and Yves Le Traon. Extracting statistical graph features for accurate and efficient time series classification. In *21st International Conference on Extending Database Technology (EDBT)*, pages 205–216, Vienna, Austria, 2018. OpenProceedings. doi:10.5441/002/edbt.2018.19.

Xiaodong Li, Reynold Cheng, Kevin Chen-Chuan Chang, Caihua Shan, Chenhao Ma, and Hongtai Cao. On analyzing graphs with motif-paths. *Proceedings of the VLDB Endowment*, 14 (6):1111–1123, 2021. doi:10.14778/3447689.3447714.

Chuang Liu and Wei-Xing Zhou. Superfamily classification of nonstationary time series based on dfa scaling exponents. *Journal of Physics A: Mathematical and Theoretical*, 43(49):495005, 2010.

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.

Lu Liu and Zhiguang Wang. Encoding temporal markov dynamics in graph for visualizing and mining time series. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Ruiyun Liu and Yu Chen. Analysis of stock price motion asymmetry via visibility-graph algorithm. *Frontiers in Physics*, 8:539521, 2020.

Yu Long. Visibility graph network analysis of gold price time series. *Physica A: Statistical Mechanics and its Applications*, 392(16):3374–3384, 2013.

Edward N Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2): 130–141, 1963.

Carl H Lubba, Sarab S Sethi, Philip Knaute, Simon R Schultz, Ben D Fulcher, and Nick S Jones. catch22: Canonical time-series characteristics. *Data Mining and Knowledge Discovery*, 33(6): 1821–1852, 2019.

Bartolo Luque, Lucas Lacasa, Fernando Ballesteros, and Jordi Luque. Horizontal visibility graphs: Exact results for random time series. *Physical Review E*, 80(4):046103, 2009.

Ao Mo Lyapunov. The general problem of motion stability. *Annals of Mathematics Studies*, 17, 1892.

Martin Maechler, Chris Fraley, Friedrich Leisch, Valderio Reisen, Artur Lemonte, and Rob J Hyndman. *fracdiff: Fractionally differenced ARIMA aka ARFIMA(p,d,q) models*, 2020. R package version 1.5-1.

Matteo Magnani, Luca Rossi, Davide Vega, and Obaida Hanteer. *multinet: Analysis and Mining of Multilayer Social Networks*, 2022. R package version 4.1.1.

Elizabeth Ann Maharaj, Pierpaolo D'Urso, and Jorge Caiado. *Time series clustering and classification*. CRC Press, Boca Raton, Florida, 2019.

Fragkiskos D Malliaros and Michalis Vazirgiannis. Clustering and community detection in directed networks: A survey. *Physics reports*, 533(4):95–142, 2013. doi:10.1016/j.physrep.2013.08.002.

Norbert Marwan. A historical review of recurrence plots. *The European Physical Journal Special Topics*, 164(1):3–12, 2008.

Norbert Marwan and Jürgen Kurths. Nonlinear analysis of bivariate data with cross recurrence plots. *Physics Letters A*, 302(5-6):299–307, 2002.

Norbert Marwan, M Carmen Romano, Marco Thiel, and Jürgen Kurths. Recurrence plots for the analysis of complex systems. *Physics Reports*, 438(5-6):237–329, 2007.

Norbert Marwan, Jonathan F Donges, Yong Zou, Reik V Donner, and Jürgen Kurths. Complex network approach for recurrence analysis of time series. *Physics Letters A*, 373(46):4246–4254, 2009.

Michael McCullough, Michael Small, Thomas Stemler, and Herbert Ho-Ching Iu. Time lagged ordinal partition networks for capturing dynamics of continuous dynamical systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 25(5):053101, 2015.

Michael McCullough, Konstantinos Sakellariou, Thomas Stemler, and Michael Small. Counting forbidden patterns in irregularly sampled time series. i. the effects of under-sampling, random depletion, and timing jitter. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 26(12): 123103, 2016.

Michael McCullough, Michael Small, Herbert Ho-Ching Iu, and Thomas Stemler. Multiscale ordinal network analysis of human cardiac dynamics. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 375(2096):20160292, 2017.

Nicolai Meinshausen, Peter Bühlmann, et al. High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34(3):1436–1462, 2006.

Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, Shai Shen-Orr, Inbal Ayzenshtat, Michal Sheffer, and Uri Alon. Superfamilies of evolved and designed networks. *Science*, 303 (5663):1538–1542, 2004.

Pablo Montero and José A. Vilar. TSclust: An R package for time series clustering. *Journal of Statistical Software*, 62(1):1–43, 2014. doi:10.18637/jss.v062.i01.

Pablo Montero-Manso, George Athanasopoulos, Rob J Hyndman, and Thiyanga S Talagala. FFORMA: Feature-based forecast model averaging. *International Journal of Forecasting*, 36(1): 86–92, 2020. doi:10.1016/j.ijforecast.2019.02.011.

Usue Mori, Alexander Mendiburu, and José Antonio Lozano. Distance measures for time series in R: The tsdist package. *R J.*, 8(2):451, 2016.

Stephen Mutua, Changgui Gu, and Huijie Yang. Visibility graph based time series analysis. *PloS one*, 10(11):e0143015, 2015.

Stephen Mutua, Changgui Gu, and Huijie Yang. Visibility graphlet approach to chaotic time series. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 26(5):053107, 2016.

Tomomichi Nakamura, Toshihiro Tanizawa, and Michael Small. Constructing networks from a dynamical system perspective for multivariate nonlinear time series. *Physical Review E*, 93(3): 032323, 2016.

Tomoaki Nakatani. *ccgarch: An R Package for Modelling Multivariate GARCH Models with Conditional Correlations*, 2014. R package version 0.2.3.

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

Mark EJ Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

Angel Nuñez, Lucas Lacasa, Eusebio Valero, Jose Patricio Gómez, and Bartolo Luque. Detecting series periodicity with horizontal visibility graphs. *International Journal of Bifurcation and Chaos*, 22(07):1250160, 2012a.

Angel M Nuñez, Lucas Lacasa, Jose Patricio Gomez, and Bartolo Luque. Visibility algorithms: A short review. *New Frontiers in Graph Theory*, pages 119–152 (Chapter 6), 2012b.

Mitchell O'Hara-Wild, Rob Hyndman, and Earo Wang. *feasts: Feature extraction and statistics for time series*, 2021. R package version 0.2.1.

Stuart Oldham, Ben Fulcher, Linden Parkes, Aurina Arnatkevičiūtė, Chao Suo, and Alex Fornito. Consistency and differences between centrality measures across distinct classes of networks. *PLOS ONE*, 14(7):e0220061, 2019. ISSN: 1932-6203. doi:10.1371/journal.pone.0220061.

Robert L. Peach, Alexis Arnaudon, Julia A. Schmidt, Henry A. Palasciano, Nathan R. Bernier, Kim E. Jelfs, Sophia N. Yaliraki, and Mauricio Barahona. HCGA: Highly comparative graph analysis for network phenotyping. *Patterns*, 2(4):100227, 2021. ISSN: 2666-3899. doi:10.1016/j.patter.2021.100227.

Xin Pei, Jiang Wang, Bin Deng, Xile Wei, and Haitao Yu. WLPVG approach to the analysis of EEG-based functional brain network under manual acupuncture. *Cognitive Neurodynamics*, 8 (5):417–428, 2014.

H Hashem Pesaran and Yongcheol Shin. Generalized impulse response analysis in linear multivariate models. *Economics Letters*, 58(1):17–29, 1998.

Arthur AB Pessa and Haroldo V Ribeiro. Characterizing stochastic time series with ordinal networks. *Physical Review E*, 100(4):042304, 2019.

Aruane M Pineda, Fernando M Ramos, Luiz Eduardo Betting, and Andriana SLO Campanharo. Quantile graphs for EEG-based diagnosis of alzheimer's disease. *Plos one*, 15(6):e0231169, 2020.

Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *International symposium on computer and information sciences*, pages 284–293, Berlin, Heidelberg, 2005. Springer. doi:10.1007/11569596_31.

Mason A Porter. What is... a multilayer network. *Notices of the AMS*, 65(11), 2018.

Meng-Cen Qian, Zhi-Qiang Jiang, and Wei-Xing Zhou. Universal and nonuniversal allometric scaling behaviors in the visibility graphs of world stock market indices. *Journal of Physics A: Mathematical and Theoretical*, 43(33):335002, 2010.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

Martín Gómez Ravetti, Laura C Carpi, Bruna Amin Gonçalves, Alejandro C Frery, and Osvaldo A Rosso. Distinguishing noise from chaos: objective versus subjective criteria using horizontal visibility graph. *PLOS ONE*, 9(9):e108004, 2014.

Henggang Ren, Qianshun Yuan, Sherehe Semba, Tongfeng Weng, Changgui Gu, and Huijie Yang. Pattern interdependent network of cross-correlation in multivariate time series. *Physics Letters A*, 384(30):126781, 2020.

Weikai Ren and Ningde Jin. Sequential limited penetrable visibility-graph motifs. *Nonlinear Dynamics*, 99:2399—–2408, 2020.

M Carmen Romano, Marco Thiel, Jürgen Kurths, and Werner von Bloh. Multivariate recurrence plots. *Physics Letters A*, 330(3-4):214–223, 2004.

Otto E Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5):397–398, 1976.

Yijing Ruan, Reik V Donner, Shuguang Guan, and Yong Zou. Ordinal partition transition network based complexity measures for inferring coupling direction and delay from time series. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(4):043111, 2019.

Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 35(2):401–449, 2021.

Jakob Runge, Vladimir Petoukhov, and Jürgen Kurths. Quantifying the strength and delay of climatic interactions: The ambiguities of cross correlation and a novel measure based on graphical models. *Journal of Climate*, 27(2):720–739, 2014.

Jakob Runge, Vladimir Petoukhov, Jonathan F Donges, Jaroslav Hlinka, Nikola Jajcay, Martin Vejmelka, David Hartman, Norbert Marwan, Milan Paluš, and Jürgen Kurths. Identifying causal gateways and mediators in complex spatio-temporal systems. *Nature Communications*, 6(1):1–10, 2015.

Jakob Runge, Sebastian Bathiany, Erik Bollt, Gustau Camps-Valls, Dim Coumou, Ethan Deyle, Clark Glymour, Marlene Kretschmer, Miguel D Mahecha, Jordi Muñoz-Marí, et al. Inferring causation from time series in earth system sciences. *Nature Communications*, 10(1):1–13, 2019a.

Jakob Runge, Peer Nowack, Marlene Kretschmer, Seth Flaxman, and Dino Sejdinovic. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science Advances*, 5 (11):eaau4996, 2019b.

Konstantinos Sakellariou, Michael McCullough, Thomas Stemler, and Michael Small. Counting forbidden patterns in irregularly sampled time series. ii. reliability in the presence of highly irregular sampling. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 26(12):123104, 2016.

Speranza Sannino, Sebastiano Stramaglia, Lucas Lacasa, and Daniele Marinazzo. Visibility graphs for fMRI data: Multiplex temporal graphs and their modulations across resting-state networks. *Network Neuroscience*, 1(3):208–221, 2017.

Facebook Infrastructure Data Science. Kats. https://facebookresearch.github.io/Kats/.

Ying-Hui Shao, Gao-Feng Gu, Zhi-Qiang Jiang, Wei-Xing Zhou, and Didier Sornette. Comparing the performance of fa, dfa and dma using different synthetic long-range correlated time series. *Scientific Reports*, 2:835, 2012.

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

AH Shirazi, G Reza Jafari, J Davoudi, J Peinke, M Reza Rahimi Tabar, and Muhammad Sahimi. Mapping stochastic processes onto complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(07):1–11, 2009.

Robert H Shumway and David S Stoffer. *Time Series Analysis and its Applications*. 1431-875X. Springer, 4 edition, 2017.

Maria Eduarda Silva and Vera Lúcia Oliveira. Difference equations for the higher-order moments and cumulants of the INAR(1) model. *Journal of Time Series Analysis*, 25(3):317–333, 2004. doi:https://doi.org/10.1111/j.1467-9892.2004.01685.x.

Vanessa Freitas Silva. Time series analysis based on complex networks. Msc thesis, University of Porto, 2018.

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:10.1002/widm.1404.

Vanessa Freitas Silva, Maria Eduarda Silva, Pedro Ribeiro, and Fernando Silva. Novel features for time series analysis: a complex networks approach. *Data Mining and Knowledge Discovery*, 36:1062—-1101, 2022.

Vanessa Freitas Silva, Maria Eduarda Silva, Pedro Ribeiro, and Fernando Silva. MHVG2MTS: Multilayer horizontal visibility graphs for multivariate time series analysis, 2023. doi:10.48550/ARXIV.2301.02333.

Michael Small. Complex networks from time series: Capturing dynamics. In *2013 IEEE International Symposium on Circuits and Systems*, pages 2509–2512. IEEE, 2013.

Michael Small, Jie Zhang, and Xiaoke Xu. Transforming time series into complex networks. In *International Conference on Complex Sciences*, volume 5, pages 2078–2089. Springer, 2009.

Xingjian Song and Fuyuan Xiao. Combining time-series evidence: A complex network model based on a visibility graph and belief entropy. *Applied Intelligence*, pages 1–10, 2022.

Fernanda Strozzi, Karmen Poljansek, Flavio Bono, Eugenio Gutierrez, and Jose'Manuel Zaldivar. Recurrence networks: evolution and robustness. *International Journal of Bifurcation and Chaos*, 21(04):1047–1063, 2011.

Narayan Puthanmadam Subramaniyam and Jari Hyttinen. Dynamics of intracranial electroencephalographic recordings from epilepsy patients using univariate and bivariate recurrence networks. *Physical Review E*, 91(2):022927, 2015.

Genaro Sucarrat. *lgarch: Simulation and Estimation of Log-GARCH Models*, 2015. R package version 0.6-2.

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.

Floris Takens. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980*, pages 366–381. Springer, 1981.

Thiyanga S Talagala, Rob J Hyndman, George Athanasopoulos, et al. Meta-learning how to forecast time series. *Monash Econometrics and Business Statistics Working Papers*, 6:18, 2018.

Luciano Telesca and Michele Lovallo. Analysis of seismic sequences by using the method of visibility graph. *EPL (Europhysics Letters)*, 97(5):1–4, 2012.

Howell Tong. Threshold models in time series analysis—30 years on. *Statistics and its Interface*, 4 (2):107–118, 2011.

Ruey S Tsay. *Analysis of Financial Time Series*, volume 543. John Wiley & Sons, 2005.

Ruey S Tsay. *Multivariate Time Series Analysis: with R and Financial Applications*. John Wiley & Sons, 2013.

AA Tsonis and KL Swanson. Review article "on the origins of decadal climate variability: a network perspective". *Nonlinear Processes in Geophysics*, 19(5):559–568, 2012.

Michele Tumminello, Fabrizio Lillo, and Rosario N Mantegna. Correlation, hierarchies, and networks in financial markets. *Journal of Economic Behavior & Organization*, 75(1):40–58, 2010.

Alessandro Vespignani. Twenty years of network science. *Nature*, 558:528–529, 2018.

Jiang Wang, Chen Yang, Ruofan Wang, Haitao Yu, Yibin Cao, and Jing Liu. Functional brain networks in alzheimer's disease: EEG analysis based on limited penetrable visibility graph and phase space method. *Physica A: Statistical Mechanics and its Applications*, 460:174–187, 2016a.

Minggang Wang and Lixin Tian. From time series to complex networks: The phase space coarse graining. *Physica A: Statistical Mechanics and its Applications*, 461:456–468, 2016.

Minggang Wang, André LM Vilela, Ruijin Du, Longfeng Zhao, Gaogao Dong, Lixin Tian, and H Eugene Stanley. Exact results of the limited penetrable horizontal visibility graph associated to random time series and its application. *Scientific Reports*, 8(1):5130, 2018a.

Minggang Wang, Hua Xu, Lixin Tian, and H Eugene Stanley. Degree distributions and motif profiles of limited penetrable horizontal visibility graphs. *Physica A: Statistical Mechanics and its Applications*, 509:620–634, 2018b.

RongXi Wang, JianMin Gao, ZhiYong Gao, Xu Gao, and HongQuan Jiang. Complex network theory-based condition recognition of electromechanical system in process industry. *Science China Technological Sciences*, 59(4):604–617, 2016b.

Xiaozhe Wang, Kate Smith, and Rob J Hyndman. Characteristic-based clustering for time series data. *Data mining and knowledge Discovery*, 13(3):335–364, 2006.

Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *Nature*, 393(6684):440–442, 1998.

William WS Wei. *Multivariate Time Series Analysis and Applications*. John Wiley & Sons, 2019.

Tongfeng Weng, Jie Zhang, Michael Small, Rui Zheng, and Pan Hui. Memory and betweenness preference in temporal networks induced from time series. *Scientific Reports*, 7(February): 41951, 2017.

Vitali Witowski and Dr. Ronja Foraita. *HMMpa: Analysing accelerometer data using hidden Markov models*, 2014. R package version 1.0.

Diethelm Wuertz, Tobias Setz, and Yohan Chalabi. *timeSeries: Rmetrics - Financial Time Series Objects*, 2017a. R package version 3042.102.

Diethelm Wuertz, Tobias Setz, Yohan Chalabi, Chris Boudt, Pierre Chausse, and Michal Miklovac. *fGarch: Rmetrics - Autoregressive Conditional Heteroskedastic Modelling*, 2017b. R package version 3042.83.

Ruoxi Xiang, Jie Zhang, Xiao-Ke Xu, and Michael Small. Multiscale characterization of recurrence-based phase space networks constructed from time series. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(1):013107, 2012.

Wen-Jie Xie, Rui-Qi Han, and Wei-Xing Zhou. Tetradic motif profiles of horizontal visibility graphs. *Communications in Nonlinear Science and Numerical Simulation*, 72:544–551, 2019.

Yuxuan Xiu, Xinyue Ren, Ting Zhang, Yanyu Chen, Li Jiang, Duo Li, Xingjun Wang, Liang Zhao, and Wai Kin Chan. Time labeled visibility graph for privacy-preserved physiological time series classification. In *2022 7th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, pages 280–284. IEEE, 2022.

Paiheng Xu, Rong Zhang, and Yong Deng. A novel visibility graph transformation of time series into weighted networks. *Chaos, Solitons & Fractals*, 117:201–208, 2018.

Xiaoke Xu, Jie Zhang, and Michael Small. Superfamily phenomena and motifs of networks induced from time series. *Proceedings of the National Academy of Sciences*, 105(50):19601–19605, 2008.

Wanfeng Yan and Edgar van Tuyll van Serooskerken. Forecasting financial extremes: A network degree measure of super-exponential growth. *PLoS One*, 10(9):e0128908, 2015.

Yue Yang and Huijie Yang. Complex network-based time series analysis. *Physica A: Statistical Mechanics and its Applications*, 387(5-6):1381–1386, 2008.

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.

Can-Zhong Yao and Ji-Nan Lin. A visibility graph approach to cny exchange rate networks and characteristic analysis. *Discrete Dynamics in Nature and Society*, 2017, 2017.

Joseph P Zbilut, Alessandro Giuliani, and Charles L Webber Jr. Detecting deterministic signals in exceptionally noisy environments using cross-recurrence quantification. *Physics Letters A*, 246(1-2):122–128, 1998.

Bo Zhang, Jun Wang, and Wen Fang. Volatility behavior of visibility graph emd financial time series from ising interacting system. *Physica A: Statistical Mechanics and its Applications*, 432: 301–314, 2015.

Haicheng Zhang, Daolin Xu, and Yousheng Wu. Predicting catastrophes of non-autonomous networks with visibility graphs and horizontal visibility. *Mechanical Systems and Signal Processing*, 104:494–502, 2018a.

Jei Zhang, Xiaodong Luo, and Michael Small. Detecting chaos in pseudoperiodic time series without embedding. *Physical Review E*, 73(1):016216, 2006.

Jiayang Zhang, Jie Zhou, Ming Tang, Heng Guo, Michael Small, and Yong Zou. Constructing ordinal partition transition networks from multivariate time series. *Scientific Reports*, 7(1):7795, 2017a.

Jie Zhang and Michael Small. Complex network from pseudoperiodic time series: Topology versus dynamics. *Physical Review Letters*, 96(23):238701, 2006.

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, 2017b.

Rong Zhang, Yong Zou, Jie Zhou, Zhong-Ke Gao, and Shuguang Guan. Visibility graph analysis for re-sampled time series from auto-regressive stochastic processes. *Communications in Nonlinear Science and Numerical Simulation*, 42:396–403, 2017c.

Rong Zhang, Baabak Ashuri, Yu Shyr, and Yong Deng. Forecasting construction cost index based on visibility graph: A network approach. *Physica A: Statistical Mechanics and its Applications*, 493:239–252, 2018b.

Yi Zhao, Xiaoyi Peng, and Michael Small. Reciprocal characterization from multivariate time series to multilayer complex networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(1):013137, 2020.

Minzhang Zheng, Sergii Domanskyi, Carlo Piermarocchi, and George I Mias. Visibility graph based temporal community detection with applications in biological time series. *Scientific Reports*, 11(1):1–12, 2021.

Ting-Ting Zhou, Ning-De Jin, Zhong-Ke Gao, and Yue-Bin Luo. Limited penetrable visibility graph for establishing complex network from time series. *Acta Physica Sinica*, 61(3):030506, 2012.

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, 2014a.

Guohun Zhu, Yan Li, and Peng Paul Wen. Epileptic seizure detection in eegs signals using a fast weighted horizontal visibility algorithm. *Computer methods and programs in biomedicine*, 115(2): 64–75, 2014b.

Enyu Zhuang, Michael Small, and Gang Feng. Time series analysis of the developed financial markets' integration using visibility graphs. *Physica A: Statistical Mechanics and its Applications*, 410:483–495, 2014.

Yong Zou, Jobst Heitzig, Reik V Donner, Jonathan F Donges, J Doyne Farmer, Riccardo Meucci, Stefano Euzzor, Norbert Marwan, and Jürgen Kurths. Power-laws in recurrence networks from dynamical systems. *EPL (Europhysics Letters)*, 98(4):48001, 2012.

Yong Zou, Reik V Donner, Norbert Marwan, Jonathan F Donges, and Jürgen Kurths. Complex network approaches to nonlinear time series analysis. *Physics Reports*, 787:1–97, 2019.

Walter Zucchini, Iain L MacDonald, and Roland Langrock. *Hidden Markov Models for Time Series: An Introduction using R*. Chapman and Hall/CRC, 2016.

L Zunino, DG Pérez, MT Martín, A Plastino, M Garavaglia, and OA Rosso. Characterization of gaussian self-similar stochastic processes using wavelet-based informational tools. *Physical Review E*, 75(2):021115, 2007.