UNIVERSITÀ CATTOLICA DEL SACRO CUORE

Interfaculty Economics - Banking, Finance and Insurance Sciences

Master of Science in Statistical and Actuarial Sciences

Data Analytics for Business and Economics

Statistical modelling of longitudinal network data: an analysis of the tech stock market

Student: Andrea CORVI

ID: 4808460

Supervisor: Prof. Stefano PELUSO

Academic Year 2019−2020

Abstract

The core of this work is represented by networks, specifically from a dynamical point of view. Longitudinal networks can be statistically modelled in several ways, most of them developed in the social sciences field. In this essay it is attempted an extension of the use of one of those models, namely the Temporal Exponential Random Graph model,to a difference science field: finance.

The TERGM is here applied to a stock correlation dataset built on the monthly correlations of the daily return of thirteen listed tech companies. The model has been tested both from a predictive and a inferential perspective. It is presented that it is possible to model such networks through the use of a TERGM. The results though can vary depending on the selected period of time. Specifically the main issue, exposed through a rolling origin cross validation methodology, is the consistency of the predictive performance over time. Still, the model has shown satisfactory results over small period of time, such as one or two years, both in terms of prediction and inference and tested on an out-of-sample prediction.

Keywords: Longitudinal Networks; TERGMs; Stock Correlation Networks

Contents

List of Figures

List of Tables

Introduction

The object of this essay are networks. They can be referred to as "a collection of interconnected things", as it is done in the Oxford English Dictionary, or in more technical expression as "a set of points and ties between them" (Tom A. B. Snijders 2006). Although they've been studied and developed in social studies, starting from the 1930s, they can found application in different fields of science, such as economics, computer science and physics.

The purpose of this work is to attempt at extending the use of a statistical model, namely the Temporal Exponential Random Graph mode, beyond the original framework of application which is a sociological study. Specifically the model will be tested on a financial application.

A first chapter will introduce the concept of networks, both static and dynamic ones. Then in the second chapter it will be present an historical and more technical explanation of the model object of this essay, the Temporal Exponential Random Graph model.

In chapter three instead will be addressed the construction of the network. The network will be built from the correlations between the daily returns of stocks belonging to thirteen companies of the tech industry.

On this specific network will be tested the model in chapter 4, testing both its capability at predicting new observations and at explaining the existing ones.

The results will be then discussed in chapter 5.

CHAPTER 1

Longitudinal Network Data

1.1 Introduction

The aim of the chapter is to give a general overview about networks, starting from their definition and mathematical representation, to, then, provide information about the historical background of the modelling approaches of both static and longitudinal networks.

1.2 Networks

1.2.1 Network Definition

Network data have recently raised attention of several researches from different science fields (economics, sociology, computer science, physics, etc.), who have, then, began to focus on their statistical analysis.

Their description and application are the core of the current essay, which requires, to start, to understand what network data actually is.

The Oxford English Dictionary refers to networks as “a collection of interconnected things”; this general description, as well as the concept behind, can be applied to different scenarios, thus leading to a more specific meaning.

An alternative definition is given by Snijders (Tom A. B. Snijders 2006) defines networks as a set of points (or nodes) and the ties between them. The points and the ties can have different meanings depending on the context. For example, social networks can be defined as representations of patterns of relations between actors (Stanley Wasserman and Faust 1994). In social networks, the "nodes" are the people belonging to the network and the object of the study is usually the presence, or not, of a relationship between them, the "ties".

In 2009, Kolaczyk has widened the definition, stating that "*networks are measurements that are either of or from a system conceptualized as a network*". Since networks can be applied to a lot of fields, the descriptions above can refer to social platforms, as well as to protein interactions in biological systems.

1.2.2 Representation of Networks

Since the meaning of “network data” actually changes depending on the context, for the aim of the study, Snijder’s definitions from the statistics field is used. Before describing the representation of networks, the common used by Snijders to talk about network data have to be explained:

- *Nodes:* set of points belonging to a network. They are also known as *actors* in social studies or as *vertices*.

- *Edges:* the ties between the nodes. They are also known as *links*.
- *Graph:* the mathematical structure used to represent such kind of networks. Specifically, a graph $G = (V, E)$ consists of a set V of vertices and a set E of edges, where elements of E are unordered pairs $\{u, v\}$ of distinct vertices $u, v \in V$ (Kolaczyk 2009).
- *Directed Graph:* a graph G for which each edge in E has an ordering to its vertices, so that $\{u, v\}$ is distinct from $\{v, u\}$ (Kolaczyk 2009). It is also known as digraph.
- *Undirected Graph:* a graph G in which each edge in E hasn't an ordering to its vertices so that $\{u, v\}$ is distinct from $\{v, u\}$.

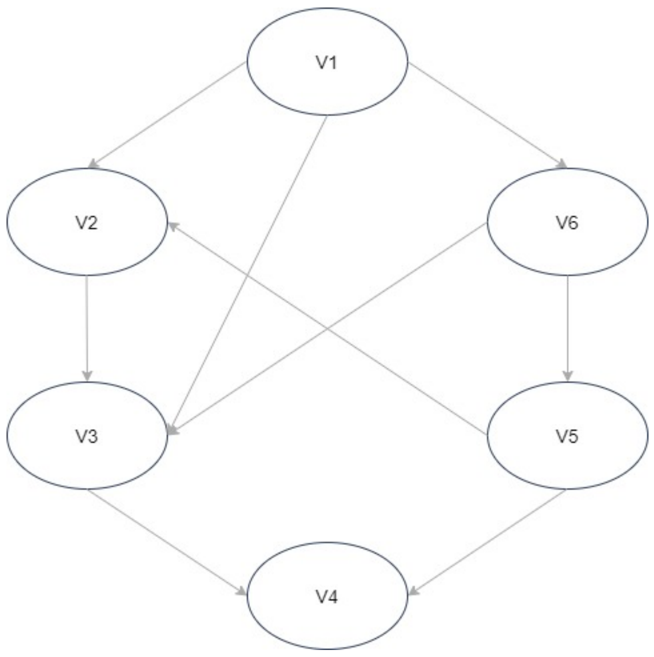
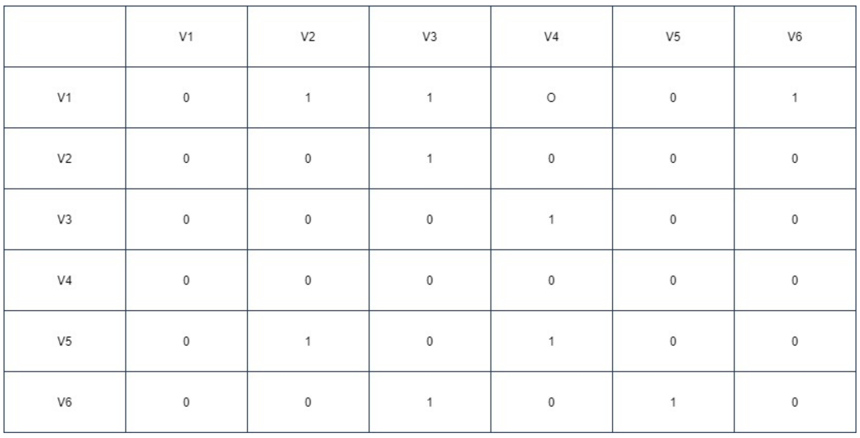


Figure 1.1: An example of graph

To understand graphs, a spatial representation could be considered. Its advantage is the more intuitive perspective given if compared to the mathematical definition described above. Anyway, since precise spatial information are not always available, thus requiring a mathematical estimation of the position and the distance between vertices, non-spatial representations are often preferred. To introduce non-spatial representations, the concept of matrices shall be considered. Two main typologies will be described: the adjacency matrix and the adjacency list.

According to (Miller and Ranum 2011), the adjacency matrix is a two-dimensional matrix $n \times n$, where n corresponds to the numerosity of a given set of nodes N = (1,...,n). In the matrix, rows are numbered from i to n and columns from j to n; thus in the matrix A, $A_{ij} = 1$ if there is a tie, while $A_{ij} = 0$ if there isn't a tie between node i and j. Which lead to the variables a_{ij} as tie variables (Tom A. B. Snijders 2006). Given the graph $G = (V, E)$, the structure of the matrix can be represent as followed by the formula:

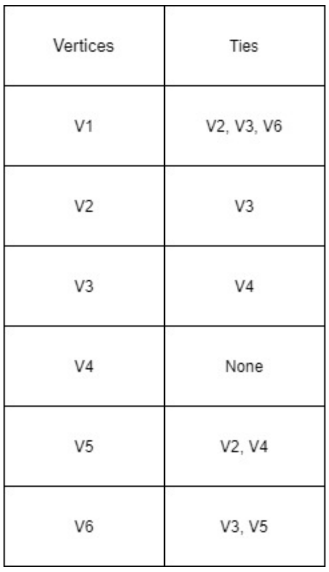
$$A = \left(A_{ij}\right)_{1 \leq i,j \leq n} \quad \text{where} \quad A_{ij} = \begin{cases} 1 & \text{if } a_{ij} \in E \\ 0 & \text{otherwise} \end{cases}$$



	V1	V2	V3	V4	V5	V6
V1	0	1	1	O	0	1
V2	0	0	1	0	0	0
V3	0	0	0	1	0	0
V4	0	0	0	0	0	0
V5	0	1	0	1	0	0
V6	0	0	1	0	1	0

Figure 1.2: Adjacency matrix corresponding to the example graph

On the other hand, the adjacency list represents the graph as a series of lists. A list of all the vertices in the network in. combined with a list of all the connections they have with other vertices (Miller and Ranum 2011).



Vertices	Ties
V1	V2, V3, V6
V2	V3
V3	V4
V4	None
V5	V2, V4
V6	V3, V5

Figure 1.3: Adjacency list corresponding to the example graph

The choice between the adjacency matrix and the adjacency list depends on the nature of the problem considered. The adjacency matrix usually is the most common, as simpler. It is fast in adding new edges and in verify the presence or absence of edges. Though, the speed implies a big effort in terms of memory used, which make it slow in iterate edges and in adding or removing new nodes. These are, instead, the cases in which the adjacency list is preferred, as most efficient when matrices are sparse.

1.2.3 A Brief Summary of Network Modelling Approaches

From an historical perspective, the progress in the understanding about network data comes from the contribution of scholars from different fields.

The first gathering and analysis on network data are dated back to 1930s (Kolaczyk 2009). Specifically, the first application field was social psychology through the studies carried out by Moreno in 1934. Moreno invented the sociogram, which consists in a set of points and lines that represent relationships among people. Its aim was to visualize the relationships between students in a classroom, by analyzing the preferences they had about whom to sit next to. Because of the approached used, its work is considered as one of the first in the field of social network analysis (Moreno 1934).

A further landmark in the history of network models is set by Erdös and Rényi 1959 and by Gilbert 1959. They studied random graphs and described the probability model for network analysis in which all edges have the same probability of occurrence and are independent from one another (Fienberg 2012). Afterwards the study of Stanley Milgram is published in 1967. In his article Milgram described the "Small World" theory, examining short paths between two nodes - people - in social networks (Milgram 1967).

In 1967, then, another progress is possible thanks to the researches of the psychologist Stanley Milgram about the “Small World” theory, which examines the lengths of the social connections that people had in the USA. It was aimed to demonstrate that human connections are a small-world social network in which short paths link people to each other. Based on these groundbreaking models, further extensions have been published in the following years.

One of them is the renowned p_1 model from P. Holland and Leinhardt 1981. In the study an extension of the Erdös-Rényi-Gilbert was provided in order to allow differential attraction, expansiveness and reciprocation. This approach led to a class of $p*$ or Exponential Random Graph Models (ERGM) and also allowed extensions towards multidimensional network and stochastic block-models (P. W. Holland, Laskey, et al. 1983).

Regarding the small world theory, instead, a new approach followed in 1998 thanks to Watts and Strogatz, who developped a random graph generation model that produces graphs with small-world properties (Watts and Strogatz 1998).

More recently, different authors have aimed to combine the stochastic block-model ideas from the 1980s with latent space models, model-based clustering or mixed- membership models to provide generative models that scale in reasonable ways to substantial-sized networks (Goldenberg et al. 2009).

1.3 Longitudinal Networks

1.3.1 Definition

The expression "Longitudinal Networks" refers to dynamic networks whose structure changes over time. They can be viewed as either a single static network that aggregates all edges observed over some time period or as a series of static networks observed in different time points over the entire network observation period (S. Uddin et al. 2013).

Longitudinal networks can be regarded as an intrinsically dynamic phenomenon. Indeed, as time passes, ties, links or vertices can change, being weakened, added or deleted. This is particularly clear in social networks, which are the most common field of study for longitudinal networks, despite the fact that they can be applied to a greater variety of contexts, from sociology to finance.

1.3.2 Historical Summary of Modelling Approaches

The first studies about longitudinal network date back to 1970s and specifically refer to the work of Holland and Leinhardt in 1977. Holland and Leinhardt used a continuos-time Markov chain to model the process by which social structure affects individual behaviour, thus implying that any changes only depend on the immediately precedent time (P. W. Holland and Leinhardt 1977).

A similar approach is followed by Wasserman in 1980s, who provided a longitudinal extension of the log-linear model of Holland and Leinhardt (Stan Wasserman 1987 , Stan Wasserman and Iacobucci 1988).

AAfterwards in 1996 Snijders proposed a different approach called "Stochastic actor- oriented model", in which each node maximizes his own utility function (T. Snijders 1996). More recent trends in the study of longitudinal networks regard the analysis of temporal exponential random graph models (Hanneke et al. 2009) in the discrete time settingand the longitudinal stochastic blockmodel extensions (Matias et al. 2015) Lastly, a different Bayesian approach has been proposed by Durante and Dunson 2014.

Chapter 2

Temporal Exponential Random Graph Models

The current section focuses on Temporal Exponential Random Graph Models (*TERGMs*) since it is the methodology chosen for the empirical application in chapter 3. The model, presented by Hanneke et al. 2009, represents a dynamic extension of the original static Exponential Random Graph Model based on the work by the mathematicians Erdös, Rényi and Gilbert. The aim of the chapter is to first introduce the Exponential Random Graph Models, by providing information about the history of their development and a subsequent description. Then, the Temporal Exponential Random Graph Models are analyzed in depth, to better understand their functioning and how they have been structured starting from the previous ones.

2.1 History of Exponential Random Graph Models

2.1.1 The Erdös-Rényi-Gilbert Model

As mentioned in the previous chapter regarding the historical scenario behind the development of network models, the Hungarian mathematicians Paul Erdös and Alfred Renyi and the American Edgar Gilbert focused their studies in the 1950s on random graphs, thus contributing to the Exponential Random Graph Models (1959) (Goldenberg et al. 2009).

The three researchers conducted in the same years two parallel studies, differentiated by the starting point. Gilbert analyzed the probability to observe connected paths in a random graph. He developed a model $G(N; p$, in which the parameters N and p respectively correspond to the dimensions of the graph (N) and the probability of existence of all the possible edges (p). Erdös and Renyi, instead, structured a similar model, though replacing the parameter p by introducing a new one, which is E. In their model $G(N; E)$, E is equal to the number of desired edges.

- Definition of the Gilbert Model: Given $N \in \mathbb{N}$ and $p \in [0, 1]$, the graph $\mathcal{G}(N, p)$ is defined as the undirected graph with N nodes with p probability of having a tie between two nodes.

- Definition of Erdös-Rényi Model. Given $N \in \mathbb{N}$ and $E \leq \begin{pmatrix} N \\ 2 \end{pmatrix}$, the graph $\mathcal{G}(N, E)$ is defined as the undirected graph with N nodes and E edges.

The two models differ in term of likelihood, as follows:

- The $G(N,p)$ model has a binomial likelihood.Thus the probability of E edges is:

  $$\ell(G(N,p) \text{ has } E \text{ edges } \mid p) = p^E(1-p)^{\binom{N}{2}-E}$$

  and gives the probability p of every tie, and the expected number of edges $p \cdot \binom{N}{2}$.

- the likelihood of the $G(N,E)$ model is an hyper geometric distribution which induces a uniform distribution over the sample space of the possible graphs.

Therefore, while Gilbert's model directly indicates the expected number of edges, considering the probability of each link, Erdös-Renyi's one implies the expected probability of every edge, by imposing a specific number of edges, (Goldenberg et al. 2009)

The connection between the two models is made possible by the formula:

$$E = p\begin{pmatrix} N \\ 2 \end{pmatrix}$$

Because, if N goes to infinity, $G(N,p)$ behaves in the same way as $G(N,E)$, thus producing an equal probability to obtain a specific graph.

Considered the computational advantages produced by the independence of the edges, the $G(N,p)$ model is easier to analyze and, therefore, the most commonly used between the two.

2.1.2 The p1 Model for Social Networks

The second Exponential Random Graph Model considered in the current essay was presented by Holland and Leinhardt in 1981. It focuses on social networks and relationships analysis and is known as p1 Model, as it is meant to be the first viable statistical model for directed graphs (P. Holland and Leinhardt 1981).

The aim of the model is to describe independent dyadic pairings between two actors i and j; since the independence of the dyads, i can tie to j or viceversa, as well as both tie to each other or neither of them.

Each node of the dyad has two parameters αi and βi; the former represents the *expansiveness* and indicates an outgoing edge sent by one of the nodes. The latter is the *popularity*, which means the incoming edge into a node.

Two additional parameters to consider are θ and ρ_{ij} and they refer to the number of ties and tendency toward reciprocation between two vertices v_i and v_i

The model sets the following probabilities for the ties:

- $\log P_{ij}(0,0) = \lambda_{ij}$
- $\log P_{ij}(1,0) = \lambda_{ij} + \alpha_i + \beta_j + \theta$
- $\log P_{ij}(0,1) = \lambda_{ij} + \alpha_j + \beta_i + \theta$
- $\log P_{ij}(1,1) = \lambda_{ij} + \alpha_i + \beta_j + \alpha_j + \beta_i + 2\theta + \rho_i j$

Where λ_{ij} is a normalizing constant which ensures that the probabilities for each dyad (i, j) add to 1.

The likelihood function for the p_1 model is exponential. For the constant reciprocation version, it is equal to:

$$\log \Pr_{p_1}(y) \propto y_{++}\theta + \sum_i y_{i+}\alpha_i + \sum_j y_{+j}\beta_j + \sum_{ij} y_{ij}y_{ji}\rho$$

One of the constraints regarding the model portrayed is the increase of the parameters due to the enlargement of the graph size itself. Which causes a lack of consistent results for the maximum likelihood when n goes to infinity.

2.1.3 Markov Graph - Exponential Random Graph Model

The third model described in the essay is the Markov Graph developed by Frank and Strauss in 1986 as a development of the p1 Model, aimed to overcome the concept of a strict independence of the dyad. In fact, the fundamental idea behind the Markov Graph is the conditional dependence between adjacent edges, thus making a step forward in the understanding of random graphs, by allowing local and global forms of dependence (Frank and Strauss 1986). Markov graphs are undirected and their probability distribution is equal to the formula:

$$\Pr_\theta\{Y = y\} = \exp\left(\sum_{k=1}^{n-1} \theta_k S_k(y) + \tau T(y) + \psi(\theta, \tau)\right) \quad y \in \mathcal{Y}$$

where $\theta := \{\theta_k\}$ and τ are parameters, $\psi(\theta, \tau)$ is the normalizing constant, and the statistics S_k and T are counts of specific structures such as edges, triangles (a non-directed closed triad), and k -stars (a set of k edges sharing a common node) (O'Malley and Onnela 2014).

The generalization of the Markov Graphs led to the current formulation of the Exponential Random Graphs Models (ERGM) or $p*$ models, by Wasserman and Pattinson in 1996. The ERGMs can be applied to both directed and undirected graphs and replace the statistics S and T by an arbitrary one (U).

The ERGMs are generally represented by the formula:

$$\mathcal{P}(N) = \frac{1}{Z(\theta)} \exp\{\theta' \mathbf{u}(N)\}$$

Where, $\theta \in \mathbb{R}^k$, and $\mathbf{u} : \mathcal{N} \to \mathbb{R}^k . Z(\theta)$ is a normalization constant (Hanneke et al. 2009). The function u constitutes the sufficient statistics for the model.

The advantage of the ERGMs is to treat networks as a single multi-variate observation in which the objective relations depend both on exogenous data (covariates) and endogenous processes. This premise allows the model to exploit both internal and external data for prediction purposes in a dynamic framework.

2.2 Temporal Exponential Random Graph Models

2.2.1 Definition

Similarly to the Exponential Random Graph Models (ERGM) previously introduced, the Temporal Exponential Random Graph Models (TERGM) consist in a series of different sub-models. What distinguishes the latter from the former in the shift from a static network and analysis to dynamic ones. For the following dissertation about the exposition of the TERGMs and its explanation, the work of Hanneke et al. 2009 and the bootsrap approach by Leifeld, Skyler Cranmer, et al. 2018 has been considered.

Specifically the model developed by Hanneke, Fu and Xing in 2009 (Hanneke et al. 2009) was used to model a dataset composed by 100 U.S. senators in order to investigate sponsors and cosponsors of different proposals during a specific legislation.

The nature of the networks is useful to understand how TERGMs operate: since they deal with dynamic networks, the observations made are equal to different time points in the evolution of the network itself. To better understand that, it is necessary to make a Markov assumption regarding the progress of the network from one step to the following one. In this case, a Markov assumption means that given the network N represented by an adjacency matrix A at time t, specifically A^t, it is assumed that A^t is independent from $A^1, ..., A^{t-2}$ given A^{t-1} (Hanneke et al. 2009). A different way to write the Markov assumption is the following:

$$\mathcal{P}\left(A^2, A^3, \ldots, A^t \mid A^1\right) = \mathcal{P}\left(A^t \mid A^{t-1}\right)\mathcal{P}\left(A^{t-1} \mid A^{t-2}\right)\cdots\mathcal{P}\left(A^2 \mid A^1\right)$$

Given the Markov assumption, the next step in defining a TERGM is to generalize ERGMs for evolving networks, specifically an additional assumption has to be made:

- $A^t|A^{t-1}$ admits an ERGM representation, meaning that the evolution between each time step can be modelled as a single Exponential Random Graph Model.

In order to define a probability density function (PDF) for the conditional probability $P(A^t|A^{t-1})$ two additional item have to be introduced:

- Function $\mathbf{\Psi}$: defined as $\mathbf{\Psi} : \mathbb{R}_{n\times n} \times \mathbb{R}_{n\times n} \rightarrow \mathbb{R}^k$. This function which is going to be computed over the objective networks has a different meaning, depending on the specific research we are considering. According to Hanneke it can be defined "as as a temporal potential over cliques across two time-adjancent networks" (Hanneke et al. 2009).A clique consists in a subset of nodes of an undirected graph such that each dyad in the clique has adjacent node. A more general definition of the function in given by Desmarais and S.J. Cranmer 2012, stating that it is a vector of network statistics which includes both endogenous and exogenous dependencies (Leifeld, Skyler Cranmer, et al. 2018).

- Vector $\boldsymbol{\theta}$: defined as $\boldsymbol{\theta} \in \mathbb{R}^k$; is the vector containing the value of the parameters of the ERGM and, in the most general case, of the TERGM.

Thus, the conditional PDF of the conditional probability $P(A^t|A^{t-1})$ can be described as:

$$\mathcal{P}\left(A^t \mid A^{t-1}, \boldsymbol{\theta}\right) = \frac{1}{Z\left(\boldsymbol{\theta}, A^{t-1}\right)} \exp\left\{\boldsymbol{\theta}' \boldsymbol{\Psi}\left(A^t, A^{t-1}\right)\right\}$$

Where Z is the normalizing constant, and $\boldsymbol{\theta}'$ indicates the transposed $\boldsymbol{\theta}$ vector. This formulation defines a Temporal Exponential Random Graph Model, specifically modelling the network A from time $t-1$ to time t.

2.2.2 Estimation

The estimation process is based on the exploitation of observed networks $N^1, ..., N^T$ - with $t = 1$,...,T - to compute an estimator $\hat{\boldsymbol{\theta}}$ close to the actual parameter vector $\boldsymbol{\theta}$.

Two options are commonly used in literature to fit, i.e. estimate, such models. The first one is a simulation method based on the use of Markov chain

Monte Carlo maximum likelihood estimation (MCMC-MLE) used by Hanneke Hanneke et al. 2009.

The second one is an approximation method based on the maximum pseudolikelihood estimation (MPLE) proposed by Desmarais and S.J. Cranmer 2012.

Markov chain Monte Carlo maximum likelihood estimation

The MCMC-MLE method uses a sample of networks from a Markov chain to compute an approximation of the normalizing constant Z. Given the formula:

$$\begin{gathered}\mathcal{L}\left(\boldsymbol{\theta}; N^1, N^2, \ldots, N^T\right) = \log \mathcal{P}\left(N^2, N^3, \ldots, N^T \mid N^1, \boldsymbol{\theta}\right) \\ \mathbf{M}(t, \boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\theta}}\left[\Psi\left(\underline{\mathbf{N}}^t, N^{t-1}\right) \mid N^{t-1}\right] \\ \mathbf{C}(t, \boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\theta}}\left[\Psi\left(\underline{\mathbf{N}}^t, N^{t-1}\right) \Psi\left(\underline{\mathbf{N}}^t, N^{t-1}\right)' \mid N^{t-1}\right]\end{gathered}$$

The expected values are computed over the random variable $\underline{\mathbf{N}}^t$ which is the network at time t.

In addition:

$$\nabla \mathcal{L}\left(\boldsymbol{\theta}; N^1, \ldots, N^T\right) = \sum_{t=2}^{T} \left(\Psi\left(N^t, N^{t-1}\right) - M(t, \boldsymbol{\theta})\right)$$

Is the gradient, i.e. the vector containing the partial derivatives of the likelihood function.

$$\nabla^2 \mathcal{L}\left(\boldsymbol{\theta}; N^1, \ldots, N^T\right) = \sum_{t=2}^{T} \left(M(t, \boldsymbol{\theta}) M(t, \boldsymbol{\theta})' - C(t, \boldsymbol{\theta})\right)$$

Is instead the Laplacian, or Laplace operator, given by the sum of second partial derivatives of the function with respect to each independent variable (Hanneke et al. 2009, "Laplace Operator" n.d.).

The algorithm used to compute the estimator $\hat{\boldsymbol{\theta}}$ is based on the work of T. Snijders 2002 and is structured as follows:

The expectations are approximated through Gibbs sampling from the conditional distribution in order to perform an unconstrained optimization similar to the Newton-Raphson method. The Gibbs sampling approximates the expectations, then updates the parameter values according to their growth direction, and, repeat up to the convergence. (Hanneke et al. 2009). It can be represented as:

```
1. Randomly initialize $\boldsymbol{\theta}^{(1)}$
2. For $i = 1$ up until convergence
3.     For $t = 2, 3, \ldots, T$
4.         Sample $\hat{N}^{t,1}_{(i)}, \ldots, \hat{N}^{t,B}_{(i)} \sim \mathcal{P}\left(\underline{N}^t | N^{t-1}, \boldsymbol{\theta}^{(i)}\right)$
5.         $\hat{\mu}^t_{(i)} = \frac{1}{B} \sum_{b=1}^{B} \Psi\left(\hat{N}^{t,b}_{(i)}, N^{t-1}\right)$
6.         $\hat{C}^t_{(i)} = \frac{1}{B} \sum_{b=1}^{B} \boldsymbol{\Psi}\left(\hat{N}^{t,b}_{(i)}, N^{t-1}\right) \boldsymbol{\Psi}\left(\hat{N}^{t,b}_{(i)}, N^{t-1}\right)'$
7.     $\hat{H}_{(i)} = \sum_{t=2}^{T} \left[\hat{\mu}^t_{(i)} \hat{\mu}^{t\prime}_{(i)} - \hat{C}^t_{(i)}\right]$
8.     $\boldsymbol{\theta}^{(i+1)} \leftarrow \boldsymbol{\theta}^{(i)} - \hat{H}^{-1}_{(i)} \sum_{t=2}^{T} \left[\boldsymbol{\Psi}\left(N^t, N^{t-1}\right) - \hat{\mu}^t_{(i)}\right]$
```

Maximum pseudolikelihood estimation

The alternative method is based on maximum pseudolikelihood estimation and has been proposed by Desmarais and Cranmer (Desmarais and S.J. Cranmer 2012).

Instead of approximating the likelihood directly the joint likelihood is replaced with the product over the conditional dyadic tie probabilities given the rest of the network (Leifeld, Skyler Cranmer, et al. 2018).

The conditional probability of the ijth element of the network being 1 can be estimated through the formula:

$$\pi_{ij}(\theta) = \mathrm{P}\left(N_{ij} = 1 \mid N_{-ij}, \theta\right) = \mathrm{logit}^{-1}\left(\sum_{r=1}^{R} \theta_r \delta_r^{(ij)}(N)\right)$$

Where the notation N_{-ij} indicates the entire network N except for the dyad N_{ij}, $\log it^{-1}$ is the inverse logistic function, r refers to any effect included in the ψ vector,and $\delta_r^{(ij)}(N)$ is equal to the change in ψ_r when N_{ij} changes from zero to one(Leifeld, Skyler Cranmer, et al. 2018).

With reference to this conditional probability, in the process of computation of the MPLE, a temporal index is added to π to define it for a network at a given time point. The MPLE is found by maximizing:

$$\arg\max_{\theta} \sum_{t=K+1}^{T} \sum_{\langle ij \rangle} \ln\left[\left(\pi_{ij}^t(\boldsymbol{\theta})\right)^{N_{ij}^t}\left(1 - \pi_{ij}^t(\boldsymbol{\theta})\right)^{1-N_{ij}^t}\right]$$

Strauss and Ikeda (Strauss and Ikeda 1990) proved that the MPLE converges asymp- totically to the MLE, which make it a good estimator as the network size increases.

Furthermore, Desmarais and Cranmer (Desmarais and S.J. Cranmer 2012) presented a bootstrap method to construct consistent confidence intervals (The bootstrap is a resampling technique adopted to estimate statistics on a certain population and works by sampling the population with replacement).

It is particularly important because it allows to build of measures of accuracy - for example confidence intervals - for the estimates. The methodology is structured as follows:

1. A sample of s estimates of θ is taken by drawing s samples of $T-K$ networks from the dynamic series of networks $\left\{N^{K+1}, \ldots, N^{T}\right\}$.
2. $\hat{\boldsymbol{\theta}}$ is then computed for each sample.

The possibility to obtain measure of accuracy can be considered ad advantage in adopting MPLE instead of MCMC-MLE. Comparing the two methods, each technique is best suited depending on the size of data available. The MCMC-MLE is preferable when small data sets have to be analyzed, as it becomes slow as the nodes and/or the time points increase. On the other hand, the MPCLE may be more precise number of time points increases since the bigger the number of observation, the higher its consistency (Leifeld, Skyler Cranmer, et al. 2018).

Chapter 3

Methodology

After an introduction about networks and the history and description of some of the main models, the third chapter is aimed to introduce a new empirical study that has been conducted as a purpose for this essay and that focuses on the Temporal Exponential Random Graph Model.

3.1 Objective & Network Structure

As mentioned in the previous chapters, most network models have been first developed and tested through a social application. As they have an interdisciplinary nature, the research project described in the following part of the essay is meant to test the application of a Temporal Exponential Random Graph Model in a different science field, which, specifically is the financial one.

Since the financial studies include a great variety of topics, it has first been necessary to narrow the range, by identifying a subset to focus on.

This has led to the choice of two main criteria to determine what observation to include: an economic sector and a performance indicator. Thus, a study over the stock market performance of corporates from the technology industry has been conducted.

The reasons behind the choice of these criteria are several. First, it is available a great amount of historical data about big corporations, which is useful considering the dynamic nature of the network that had to be analyzed.

Secondly, considered the characteristics of the sector, it has been evaluated plausible to believe that there could be any possible relations between the financial trends of the main companies in the tech industry. Last, dealing with the changes over time in the tech sector appeared as a challenge in terms of statistical modelling because of its fast-evolving pace and continuous innovation.

Therefore, the study presented in the following paragraphs considers longitudinal networks, which, as previously described, are defined by nodes and edges (the existing relationships between nodes).

3.1.1 Nodes

To structure the longitudinal network object of the study, thirteen big players in the tech sector has been selected as nodes:

- Microsoft Corporation (NASDAQ)
- Alphabet Inc. (NASDAQ)
- Apple Inc. (NASDAQ)
- Amazon.com, Inc. (NASDAQ)

- Cisco Systems Inc. (NASDAQ)
- SAP SE (NYSE)
- STMicroelectronics NV (BIT)
- Siemens AG (DAX)
- Philips (EURONEXT)
- Sony Corporation (TYO)
- Samsung Electronics Co., Ltd. (KRX)
- Fujitsu (TYO)
- Toshiba Corporation (TYO)

All the thirteen companies are listed on -different- stock exchanges and have been chosen in order to cover the technology sector not only by market share but also by geographical positioning; which allowed to take into account also macro-economic data on the companies.

About the companies, their daily stock return has been collected, with reference to a set time period, from January 1st 2005 to August 8th 2020.

All this data was collected through a Python script exploiting the "pandas_datareader" library. The data have, then, been stored in comma separated value file (.csv) having 4059 rows (the daily returns) and 14 columns (the thirteen companies plus a date column indicating the date of the daily returns). The file has, thus, been cleaned by removing missing values, in order to obtain a neat dataset ready to use. The final version of the document file included 3490 observations out of the original 4059 ones.

Reinterpreting this data according to the network perspective, the network that has been structured counted 3490 time points and 13 nodes.

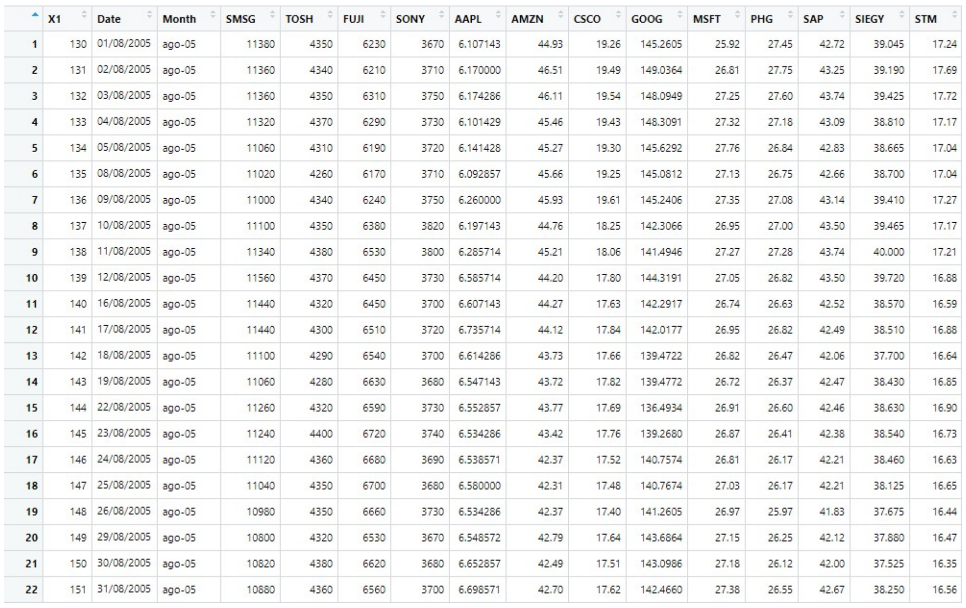
Since 3490 seems a very high number for time points, especially compared to datasets seen in literature, an explanation regarding their decrease in connection with the creation of the edges follows in the next paragraph.

3.1.2 Edges

Given two nodes i and j, the edges of a network, also called ties, indicate whether there is a link between i and j. In directed network the link can be used to study the relation between the nodes.

The edge can have a direction; meaning that the relation from i to j is different than the relation between j and i. Since the relation between the nodes considered in the current essay is symmetric, there is no difference in it being computed from i to j or vice versa. Therefore, the network is undirected.

Starting from the data described in the previous section, the dataset has then been grouped by month, creating 188 sub-datasets, each of them containing the daily return for a specific month. This means that the final longitudinal network has 188 time points, each corresponding to a specific month of a specific year.



	X1	Date	Month	SMSG	TOSH	FUJI	SONY	AAPL	AMZN	CSCO	GOOG	MSFT	PHG	SAP	SIEGY	STM
1	130	01/08/2005	ago-05	11380	4350	6230	3670	6.107143	44.93	19.26	145.2605	25.92	27.45	42.72	39.045	17.24
2	131	02/08/2005	ago-05	11360	4340	6210	3710	6.170000	46.51	19.49	149.0364	26.81	27.75	43.25	39.190	17.69
3	132	03/08/2005	ago-05	11360	4350	6310	3750	6.174286	46.11	19.54	148.0949	27.25	27.60	43.74	39.425	17.72
4	133	04/08/2005	ago-05	11320	4370	6290	3730	6.101429	45.46	19.43	148.3091	27.32	27.18	43.09	38.810	17.17
5	134	05/08/2005	ago-05	11060	4310	6190	3720	6.141428	45.27	19.30	145.6292	27.76	26.84	42.83	38.665	17.04
6	135	08/08/2005	ago-05	11020	4260	6170	3710	6.092857	45.66	19.25	145.0812	27.13	26.75	42.66	38.700	17.04
7	136	09/08/2005	ago-05	11000	4340	6240	3750	6.260000	45.93	19.61	145.2406	27.35	27.08	43.14	39.410	17.27
8	137	10/08/2005	ago-05	11100	4350	6380	3820	6.197143	44.76	18.25	142.3066	26.95	27.00	43.50	39.465	17.17
9	138	11/08/2005	ago-05	11340	4380	6530	3800	6.285714	45.21	18.06	141.4946	27.27	27.28	43.74	40.000	17.21
10	139	12/08/2005	ago-05	11560	4370	6450	3730	6.585714	44.20	17.80	144.3191	27.05	26.82	43.50	39.720	16.88
11	140	16/08/2005	ago-05	11440	4320	6450	3700	6.607143	44.27	17.63	142.2917	26.74	26.63	42.52	38.570	16.59
12	141	17/08/2005	ago-05	11440	4300	6510	3720	6.735714	44.12	17.84	142.0177	26.95	26.82	42.49	38.510	16.88
13	142	18/08/2005	ago-05	11100	4290	6540	3700	6.614286	43.73	17.66	139.4722	26.82	26.47	42.06	37.700	16.64
14	143	19/08/2005	ago-05	11060	4280	6630	3680	6.547143	43.72	17.82	139.4772	26.72	26.37	42.47	38.430	16.85
15	144	22/08/2005	ago-05	11260	4320	6590	3730	6.552857	43.77	17.69	136.4934	26.91	26.60	42.46	38.630	16.90
16	145	23/08/2005	ago-05	11240	4400	6720	3740	6.534286	43.42	17.76	139.2680	26.87	26.41	42.38	38.540	16.73
17	146	24/08/2005	ago-05	11120	4360	6680	3690	6.538571	42.37	17.52	140.7574	26.81	26.17	42.21	38.460	16.63
18	147	25/08/2005	ago-05	11040	4350	6700	3680	6.580000	42.31	17.48	140.7674	27.03	26.17	42.21	38.125	16.65
19	148	26/08/2005	ago-05	10980	4350	6660	3730	6.534286	42.37	17.40	141.2605	26.97	25.97	41.83	37.675	16.44
20	149	29/08/2005	ago-05	10800	4320	6530	3670	6.548572	42.79	17.64	143.6864	27.15	26.25	42.12	37.880	16.47
21	150	30/08/2005	ago-05	10820	4380	6620	3680	6.652857	42.49	17.51	143.0986	27.18	26.12	42.00	37.525	16.35
22	151	31/08/2005	ago-05	10880	4360	6560	3700	6.698571	42.70	17.62	142.4660	27.38	26.55	42.67	38.250	16.56

Figure 3.1: Daily return grouped by month.

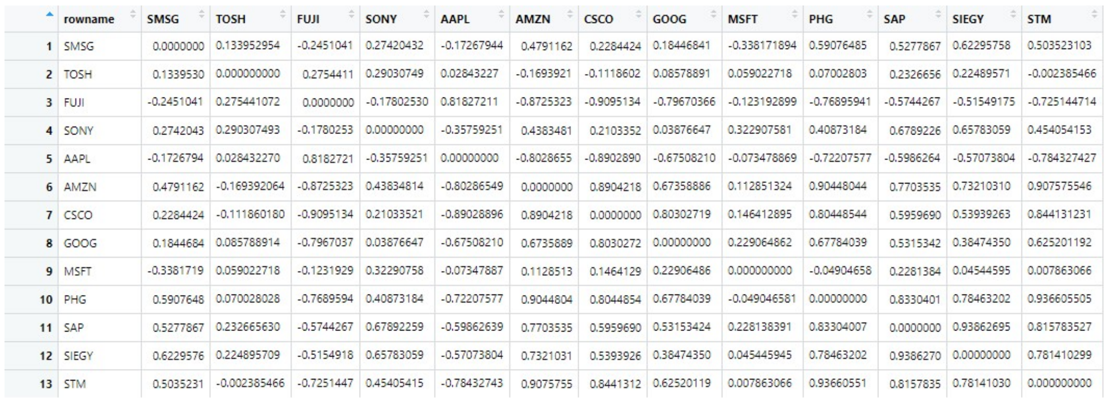
For each of this dataset (Figure 3.1) and for each of the columns (which correspond to the companies) the Pearson correlation coefficient is computed once per each company against all the other companies, once at a time.

The Pearson correlation coefficient is defined as:

$$\rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}$$

where σ_{XY} is the covariance between X and Y and σ_X, σ_Y are the two standard deviations.

The results obtained is the following correlation matrix for each of the 188 sub-datsets:



	rowname	SMSG	TOSH	FUJI	SONY	AAPL	AMZN	CSCO	GOOG	MSFT	PHG	SAP	SIEGY	STM
1	SMSG	0.0000000	0.133952954	-0.2451041	0.27420432	-0.17267944	0.4791162	0.2284424	0.18446841	-0.338171894	0.59076485	0.5277867	0.62295758	0.503523103
2	TOSH	0.1339530	0.000000000	0.2754411	0.29030749	0.02843227	-0.1693921	-0.1118602	0.08578891	0.059022718	0.07002803	0.2326656	0.22489571	-0.002385466
3	FUJI	-0.2451041	0.275441072	0.0000000	-0.17802530	0.81827211	-0.8725323	-0.9095134	-0.79670366	-0.123192899	-0.76895941	-0.5744267	-0.51549175	-0.725144714
4	SONY	0.2742043	0.290307493	-0.1780253	0.00000000	-0.35759251	0.4383481	0.2103352	0.03876647	0.322907581	0.40873184	0.6789226	0.65783059	0.454054153
5	AAPL	-0.1726794	0.028432270	0.8182721	-0.35759251	0.00000000	-0.8028655	-0.8902890	-0.67508210	-0.073478869	-0.72207577	-0.5986264	-0.57073804	-0.784327427
6	AMZN	0.4791162	-0.169392064	-0.8725323	0.43834814	-0.80286549	0.0000000	0.8904218	0.67358886	0.112851324	0.90448044	0.7703535	0.73210310	0.907575546
7	CSCO	0.2284424	-0.111860180	-0.9095134	0.21033521	-0.89028896	0.8904218	0.0000000	0.80302719	0.146412895	0.80448544	0.5959690	0.53939263	0.844131231
8	GOOG	0.1844684	0.085788914	-0.7967037	0.03876647	-0.67508210	0.6735889	0.8030272	0.00000000	0.229064862	0.67784039	0.5315342	0.38474350	0.625201192
9	MSFT	-0.3381719	0.059022718	-0.1231929	0.32290758	-0.07347887	0.1128513	0.1464129	0.22906486	0.000000000	-0.04904658	0.2281384	0.04544595	0.007863066
10	PHG	0.5907648	0.070028028	-0.7689594	0.40873184	-0.72207577	0.9044804	0.8044854	0.67784039	-0.049046581	0.00000000	0.8330401	0.78463202	0.936605505
11	SAP	0.5277867	0.232665630	-0.5744267	0.67892259	-0.59862639	0.7703535	0.5959690	0.53153424	0.228138391	0.83304007	0.0000000	0.93862695	0.815783527
12	SIEGY	0.6229576	0.224895709	-0.5154918	0.65783059	-0.57073804	0.7321031	0.5393926	0.38474350	0.045445945	0.78463202	0.9386270	0.00000000	0.781410299
13	STM	0.5035231	-0.002385466	-0.7251447	0.45405415	-0.78432743	0.9075755	0.8441312	0.62520119	0.007863066	0.93660551	0.8157835	0.78141030	0.000000000

Figure 3.2: Correlation matrix of one month.

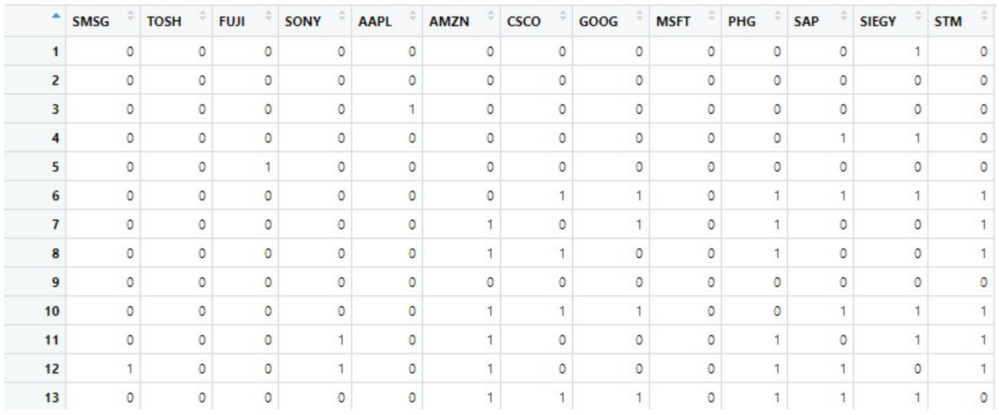
Once obtained the output above, the subsequent step has been to identify edges, given the correlation matrix. To do that, the filtering of the correlation has been used.

Thus by following the decision rule:

$$A_{ij} = \begin{cases} 1 & \text{if correlation} \geq \text{threshold} \quad \mathcal{E} \\ 0 & \text{otherwise} \end{cases}$$

If the monthly correlation between the daily return of two companies is greater than a certain threshold $\mathcal{E}$, then there is a tie and it is indicated as 1 in the matrix. Otherwise if the correlation doesn't reach a certain threshold $\mathcal{E}$, then there isn't an edge and so it is set equal to 0 in the matrix.

By applying the decision rule to each of the 188 sub-datasets, it was possible to obtain 188 adjacency matrix (Figure 3.3) where each of them, on its own, represents either a single static network or, in a collective view, a specific time point of a longitudinal network.



	SMSG	TOSH	FUJI	SONY	AAPL	AMZN	CSCO	GOOG	MSFT	PHG	SAP	SIEGY	STM
1	0	0	0	0	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	1	1	0
5	0	0	1	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	1	0	1	1	1	1
7	0	0	0	0	0	1	0	1	0	1	0	0	1
8	0	0	0	0	0	1	1	0	0	1	0	0	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	1	1	1	0	0	1	1	1
11	0	0	0	1	0	1	0	0	0	1	0	1	1
12	1	0	0	1	0	1	0	0	0	1	1	0	1
13	0	0	0	0	0	1	1	1	0	1	1	1	0

Figure 3.3: Adjacency matrix for one time point.

Given this result, two issues had yet to be addressed: the choice of the threshold $\mathcal{E}$ and the choice of the adjacency matrix to represent the network. In literature is not present an optimal choice for settings similar to one considered. Thus, in this essay, it will not be analyzed just one network corresponding to a specific threshold but four different networks will be built using four different thresholds, specifically:

- $\mathcal{E}_1 = 0.6$
- $\mathcal{E}_2 = 0.7$
- $\mathcal{E}_3 = 0.65$
- $\mathcal{E}_4 = 0.75$

Before performing any analysis, a better performance of the model is expected for higher $\mathcal{E}$ values, since to higher thresholds correspond stronger relationships, specifically only highly correlated companies have a tie.

Regarding the choice of the adjacency matrix, in the first chapter of this work it has been stated that adjacency list performs better in presence of highly sparse matrices. Yet, although the matrices are sparse, in particular the ones corresponding to higher $\mathcal{E}$, it was preferred to use the adjacency matrix framework, because the computational efficiency has been sacrificed in favor of a more intuitive and practical representation, since in this case the starting data are in tabular form, specifically in the dataframe format. As dataframes are represented by rows and columns, same as matrices, the choice of an adjacency matrix has been considered optimal.

3.1.3 Nodal Attributes

In addition to the endogenous effect captured by the filtered correlation, it was decided to exploit the possibility of adding exogenous data, in the form of nodal attributes, to the TERGM. This has made it possible to insert further information on the single nodes in the network. This additional data works then as a covariate in the TERGM.

During the building of the network two attributes were created exploiting the data available for each stock. The new attributes, specifically the Sharpe Ratio and the Beta, were computed for each stock and for each time point using the R package "PerformanceAnalytics".

- Sharpe ratio:

  The Sharpe ratio was developed by Sharpe (Sharpe 1966) in order to measure the performance of an asset against a risk free investment. It is defined as follows:

$$S_a = \frac{E\left[R_a - R_b\right]}{\sigma_a} = \frac{E\left[R_a - R_b\right]}{\sqrt{\operatorname{var}\left[R_a - R_b\right]}}$$

where R_a is the asset return, R_b is the risk-free return which in this case was selected as the return of 10 years the USA Treasury bonds, updated every year.

The higher the Sharpe ratio, the better since when comparing two assets, the one with the higher Sharpe ratio is providing a higher return with the same risk.

- Beta:

  The Beta is a measure of the risk of an asset compared to the market in which it is listed. It is defined as:

$$\beta_{a,b} = \frac{CoV_{a,b}}{\sigma_b} = \frac{\sum\left((R_a - \bar{R}_a)(R_b - \bar{R}_b)\right)}{\sum (R_b - \bar{R}_b)^2}$$

  Where R_a is the asset return and R_b is the benchmark asset return which is the market index in which it is listed. The equation gives the slope of the linear regression of R_a on R_b

Both the covariates have been added to the network as quantitative covariates and as categorical one, by creating four categories based on their quantiles.

To summarise, each longitudinal network has the following structure:

- 13 Nodes
- 2 Nodal attributes
- 188 Time points

In addition to what has already been described, networks can also be evaluated through a density statistic which allows to understand how much the nodes are connected between each other, with respect to the potential connections.

The density is computed as follows:

Potential Connections:

$$PC = \frac{n^*(n-1)}{2}$$

Network Density:

$$\frac{\text{Actual Connections}}{\text{Potential Connections}}$$

The mean density, during the 188 time points, for the four networks is:

- N_1 has density: 0.3919
- N_2 has density: 0.2991
- N_3 has density: 0.3481
- N_4 has density: 0.2459

Thus, this kind of network has a low density, especially for higher correlation thresholds.

Chapter 4

Network Analysis

After an introduction about the methodology considered to develop the study presented through the current essay, the following chapter is aimed to give an overview about the network that has been structured.

Specifically, the analysis of the network is divided in three main parts: to start, a descriptive section introduces the networks evolution over time, also considering a graphical perspective; thus, a modelling part focuses on the implementation of the model; last, the goodness of fit of the model is explained in the third section.

4.1 Network Description

The previous chapter has described the realization of the four different longitudinal networks object of the study. Each of them has been structured through 188 steps, based on different correlation threshold $\mathcal{E}$. The aim of the current section, thus, is to examine in depth how a dynamic network evolves over time, through a graphical representation. Being meant as an example, the description only considers one of the four networks created and, specifically, analyzes two of its time-steps, which have been referred to as t_1 and t_2.

First, the figure 4.1 refers to $t = 1$ and reproduces the network as a graph (a way of representing networks previously introduced in chapter one).

By analyzing the graphical representation, it is possible to deduce that the network is composed by 13 nodes and 9 edges (represented by the orange links). Thus, it is possible to maintain that the 8 linked companies have a relationship, through returns correlation, during the specific time period (month) considered.

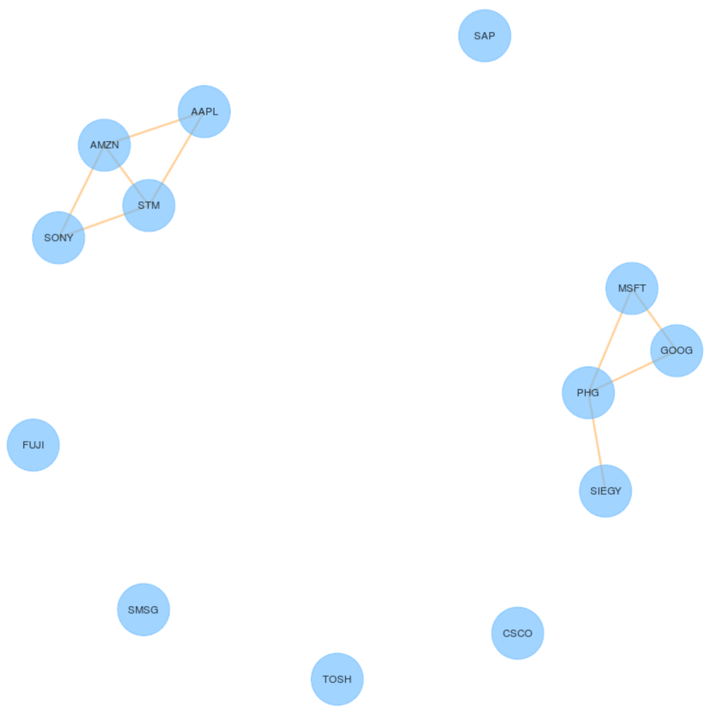


Figure 4.1: Example network at $t = 1$

On the other hand, the figure 4.2 portrays $t = 2$ and shows how relationships - the ties - evolve over time. A meaningful number of ties have, indeed, changed from the first time step to the subsequent one. For the aims of the study, the mutation of ties reproduced by figure 4.2 has, thus, been modelled as shown by the following section.

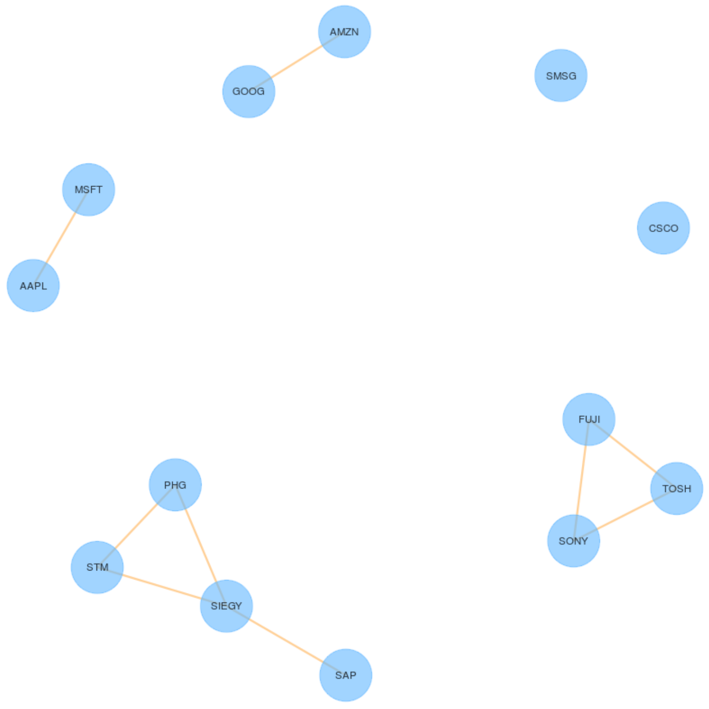


Figure 4.2: Example network at $t = 2$

4.2 Model Presentation

A Temporal Exponential Random Graph Model has been implemented to model the networks described in the study. Specifically, the bootstrap extension proposed by Leifeld, Skyler Cranmer, et al. 2018 has been considered. As shown in chapter two, a TERGM can be portrayed by the following formulation:

$$\mathcal{P}\left(A^t \mid A^{t-1}, \boldsymbol{\theta}\right) = \frac{1}{Z\left(\boldsymbol{\theta}, A^{t-1}\right)} \exp\left\{\boldsymbol{\theta}' \boldsymbol{\Psi}\left(A^t, A^{t-1}\right)\right\}$$

To be able to use this model to perform statistical modelling on real data - i.e. fitting the models – it is necessary to first specify the $\mathbf{\Psi}(A)$ vector of statistics, including exogenous and endogenous dependencies. Given this vector of statistics, the model fitting is performed by estimating the $\boldsymbol{\theta}$ vector of corresponding parameter through maximum pseudolikelihood estimation.

Thus, what defines the specific model are the chosen exogenous and endogenous dependencies. A conspicuous number of them are available and defined in the *ergm* R package (P. N. Krivitsky 2018) and referred as "*terms*". Specifically for an undirected network like the one object of this essay, circa 25 terms are available. This implies an issue which can be seen as a variable selection, since the terms are not proper covariates but still determine the parameters of the model. For the purpose of this essay the terms to choose are the ones that allow the best performance in terms of modelling of the network. Although the goodness of fit is discussed later in the chapter, it is still necessary to first explain the measure used to select the best terms.

The fitting of a statistical model can be evaluated in several ways. The one that has been chosen for the essay is the usage of the goodness of fit (gof) function. The goodness of fit function allows the so structured evaluation:

1. Estimate the parameters from a certain number of time steps.

2. Simulate a certain number of networks starting from the estimated parameters, predicting the network at the following time step.

3. Compare the simulations with the observed network at the target time step.

When the target network that has to be predicted is within the time steps used for the estimation of the parameters, the operation is defined as in-sample prediction. Instead, when the target network is outside of the observation used in the estimation, the operation is an out-of-sample prediction.

Comparing them, the out-of-sample prediction gives the model a stronger validation, as it is performed on observations unseen in the model; therefore, it has been used in the essay.

The goodness of fit function of the R package *btergm* allows the computation of different evaluation metrics,. The receiver operating characteristics (ROC) curve is one of these evaluation metrics: in the ROC curve the dyadic states of the observed network(s) are compared to the dyadic states of the simulated ones.

Given the fact that a binary variable, as in this case the presence or not of a tie, can have to possible outcome: positive and negative, the TPR is computed as the ration between the true positives over the sum between the true positives and the false negatives, meaning positive observations predicted as negative ones.

$$\frac{\text{TP}}{\text{TP} + \text{FN}}$$

The FPR instead is computed as the ration between the false positives (negative observations predicted as positives) and the sum of the true negative observations and the false positives.

$$\frac{\text{FP}}{\text{TN} + \text{FP}}$$

Alongside the ROC curve it is possible to compute the AUC which is the area under the ROC curve. The AUC of different models can be directly compared to select the better model. The higher the better, with values ranging up to 1 which is the best classifier possible.

Thus, in order to obtain the best model possible, the terms of the model object of this essay have been empirically selected by choosing the ones that maximize the AUC of the model in an out-of-sample prediction on data starting from July 2019 to July 2020, using eleven months to estimate the parameters and predicting the network on the twelfth month.

Based on this selection process, the *terms* or network statistics selected for the model are the following:

1. *edges*: network statistic equal to the number of edges in the network.
2. *nodecov("SR")*: nodal quantitative covariate having the value of the Sharpe ratio.
3. *nodecov("Beta")*: nodal quantitative covariate having value corresponding to the beta.
4. *nodecov("SR_factorized")*: nodal categorical covariate created by factorization of the SR vector based on its quantiles.
5. *nodecov("Beta_factorized")*: nodal categorical covariate created by factorization of the Beta vector based on its quantiles.
6. *absdiff("SR")*: network statistic equal to the sum of the absolute difference between the SR value of node i and node j for all edges (i, j).

7. *absdiff("Beta")*: network statistic equal to the sum of the absolute difference between the Beta value of node i and node j for all edges (i, j).

8. *absdiffcat("SR_factorized")*: network statistic equal to the sum of the absolute difference between the SR factorized value of node i and node j for all edges (i, j).

9. *absdiffcat("Beta_factorized")*: network statistic equal to the sum of the absolute difference between the Beta factorized value of node i and node j for all edges (i, j).

10. *timecov()*: time covariate that checks whether there is a time trend in the number of ties over time.

11. *balance*: network statistic equal to the number of triads in the network that are balanced. In the case of an undirected network, as in this one, the balanced triads are those with an even number of ties.

12. *gwesp()*: network statistic equal to the geometrically weighted edgewise shared partner distribution. Which is defined as the number of unordered pairs (i, j) such that i and j have a tie between them and they have exactly k common neighbors (Hunter and M. S. Handcock 2006).

13. *gwdsp()*: network statistic equal to the geometrically weighted dyadwise shared partner distribution.

14. *gwdegree()*: network statistic equal to the geometrically weighted degree distribution. The degree of a node is the number of ties it has to other nodes.

15. *memory()*: memory term that check for the impact of a previous network on the current network. In this case the type is positive autoregression thus, it checks whether previous ties are carried over to the current network.

4.3 Evaluation of the Goodness of Fit

In the previous section, the concepts of goodness of fit and ROC curve have been introduced. To assess the goodness of the model or to validate it, a further topic needs to be addressed. This leads to the analysis of the purpose of the statistical models, as each of them can aim at two different ones: prediction and inference. Every purpose can of them can be assessed in different ways and with different measures.

Predictive Capability

The predictive capability represents the capability of a model to forecast the target variable given a set of observations. The assessment of the goodness of a model in this area is straightforward since different models – as long as they are applied to the same dataset - can be directly compared by looking at some measures such as the AUC of the Roc curve, given the predicted observations.

Usually to assess the predictive capability of the model, if the data available is sufficient, the dataset is randomly splitted into two subsets, which can be referred to as *training* and *test*. The model is fitted on the first and then used to forecast the latter. The issue that can arise with prediction tasks is the risk of overfitting.

Overfitting can happen when the model performs greatly on the training data, i.e. the data used to fit the model, but poorly on new observations. This issue can be addressed by performing model selection based on the results of the model on test data, i.e. new observations not used to fit the model. Thus, it is important to test the model on data not included in the fitting and perform an out-of-sample prediction.

Therefore, the selection process for the *terms* introduced in the previous section is a procedure of validation of the model. It has been chosen by maximizing the ROC AUC of a model fitted on 11 months and used to predict the twelfth one. A good predictive performance in the tweltfth time point is indeed a good performance overall since it is an out-of-sample prediction, on new data. Despite all, there is an issue regarding this validation methodology, which is that it is specific for the time period of 12 months splitted in 11 *training* months and one *test* month. Thus, the model could perform well with the data considered for the *terms* selection and possibly not in the year before the one of the observations included; although the model has been validated in the prediction from July 2019 to July 2020, it still has to be validated in a more general way.

Furthermore, a second issue is related to cyclic events. Indeed one option could be to perform the same prediction for each year in the dataset, but that would not consider the bias present in the process of predicting the same month each year. In fact, it wouldn't validate the model enough to state its goodness of fit for each month and for each year of the data available.

Therefore, in this specific case a different approach is needed. An alternative is the rolling origin technique that has been proposed by Leonard J. Tashman (Tashman 2000); despite being developed for time-series, it can still be applied to longitudinal networks, as they can be considered a time series of single networks at each time point. Taking as example a model that exploits 11 time point to predict the twelfth, as in the previous validation model, this evaluation methodology works as follows:

1. Select the number of time points for the fitting of the model (9).
2. Select the number of time points for the prediction (1, thus the tenth time point will be predicted.
3. Perform fitting and prediction.
4. Shift the starting time point by one (the second time point in the example) keeping the number of time points for the fitting and prediction fixed, so in the example the fitting will be now from time point 2 to time point 10 and the time point 11 will be predicted.
5. *Computation of the Roc AUC*
6. Repeat till the end of the dataset.

The computation of the Roc AUC is not part of the original methodology but has been added in this essay in order to allow the evaluation of the models obtained through the rolling origin technique.

This method has been chosen as, by obtaining consisting performance in terms of Roc AUC for each model, it can be stated that the model is valid in terms of predictive capability, regardless of the training and testing period chosen.

Specifically the technique has been applied three times, using three different training subset dimensions: 11, 23 and 35 months in order to predict the next month network considering a time window of respectively one year, two years and three years.

Given this framework and this methodology, it can be made a comparison of the mean Roc AUC of all the models created through the rolling origin technique.

	Mean Roc AUC 1 year	Mean Roc AUC 2 years	Mean Roc AUC 3 years
Network1 $\mathcal{E}$ = 0.6	0.6203	0.6231	0.6371
Network2 $\mathcal{E}$ = 0.7	0.6257	0.6260	0.6397
Network3 $\mathcal{E}$ = 0.65	0.6227	0.6256	0.6327
Network4 $\mathcal{E}$ = 0.75	0.6369	0.6590	0.6612

Table 4.1: Roc AUC results for the four networks.

Regarding the results table 4.1, some considerations can be made.

First of all, the results don't show optimal values for the AUC. Indeed, values starting from 0.70 could be considered a good result overall. A second consideration is that these values have to be contextualized: in fact, computing the mean of the AUC of all the models created through the rolling origin technique implies re- moving the temporal dependencies of specific time periods. Thus, not obtaining good results in terms of mean AUC doesn't necessarily imply that the observed phenomena can't be predicted by the statistical model considered. It, actually, means that the validation of the model can't be generalized in such way that it performs well even without considering specific time spans. Indeed, the model could be performing well in a certain time windows and this can be understood by looking at the histogram of the vector of AUC results.

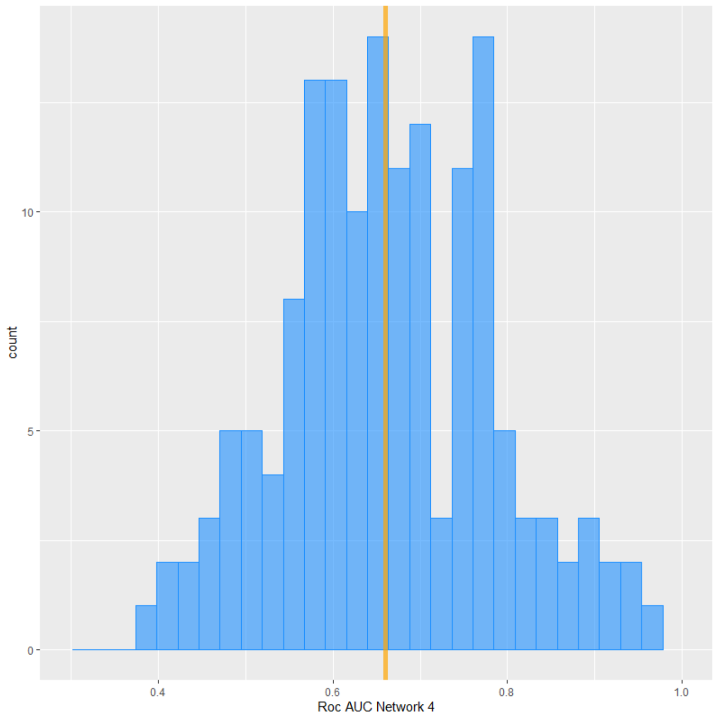


Figure 4.3: Histogram of Roc AUC of Network 4

Taking as example the results of the model based on a three year time window on network 4, from the plot (figure 4.3) it is possible to understand that the model has an overall medium-low performance but still performs well in a certain time period. By recalling that all the models created with the rolling origin methodology perform an out-of-sample prediction, which is a validation technique by itself, it can be stated that the model can, indeed, predict the phenomena in some specific time points but this deduction can't be generalized enough to state that one model could predict the evolution of 15 years of data. The TERGM have, then, to be fine-tuned for the objective time window in order to predict the phenomena since one specific formulation isn't enough for all the time points.

To support the fact that the model can be tuned and used to describe a specific time window, it is possible to consider the time span used for the variable selection process. The time period analyzed starts from July 2019 and goes up to July 2020, where the first eleven months are used to predict the twelfth.

	Model 1
Training Subset	July 2019 - June 2020
Test Subset	July 2020
Roc AUC Network1	0.6778
Roc AUC Network2	0.7568
Roc AUC Network3	0.8033
Roc AUC Network4	0.7849

Table 4.2: Results of the model on a specific time window.

The results shown in the table above (Table 4.2) are promising. They represent a good output overall, because the AUCs computed are based on an out-of-sample prediction. This table, alongside the histogram shown by figure 4.3, support the fact that the phenomenon can be predicted through a Temporal Exponential Random Graph model but the same model can't be generalized for all the time points considered as it needs to be tuned for specific time windows.

Inferential Capability

Given the results obtained in the previous section, it is of interest to analyze the model built on the time window July 2019 - July 2020 also from an inferential point of view to check if that model, which has a good predictive performance in terms of AUC, also describes well the network.

The inferential capability of a model deals with its ability to capture the properties of the data generating process. It means that instead of predicting a specific outcome, it focuses on understanding the relationships between the data, in particular the covariates, allowing then the explanation of phenomena. The objective becomes then to replicate the data because building a statistical model that is able to simulate data close to the observed one means that the model is able to nearly capture the relationship between it.

This aspect, too, can be evaluated through the goodness of fit function of the *btergm* R package. Indeed, the model can be fitted on a dataset and then used to simulate a network at a specific time. By creating numerous simulations of the network and then comparing them, on different statistics, to the observed one, it is possible to understand how close the simulated data is to the real one.

Thus, the evaluation process in this essay is structured as follows:

1. The model is fitted on a training subset.

2. *n* simulations are made of the observed network, not include in the training subset.

3. The simulated networks are then compared to the observed one on the following statistics:

    - *Esp*: edge-wise shared partner distribution.
    - *Dsp*: dyad-wise shared partner distribution.
    - *Deg*: degree distribution.
    - *Kstar*: k-star distribution.
    - *Geodesic*: geodesic distance distribution, i.e. the paths with the minimum number of edges.

To compare the simulated and the observed networks regarding the previously mentioned statistics it is necessary to plot them and check whether the observed network, represented by a line, is within the box plot of the simulated networks and in particular if it is close to the mean and median.

Thus, the goodness of fit of the model can be assessed in a straightforward way from the following plots:

The first plot shows the edgewise shared partner distribution of the four networks, which is defined as *the number of unordered pairs* (i, j) *such that i and j have a tie between them and they have exactly k common neighbors* (Hunter and M. S. Handcock 2006). The statistics are better captured in network 2 and 4 and it can be considered a foreseen result since those two networks have been built with an higher correlation threshold, meaning that the relationship studied is stronger by definition with respect to networks 1 and 3. Specifically in this case it appears that the observation line is especially close to the median for network 4. Regarding the other networks, namely network 1 and 3, the box-plots show that the general trend is captured but not in a sufficiently good measure, especially for network 1.

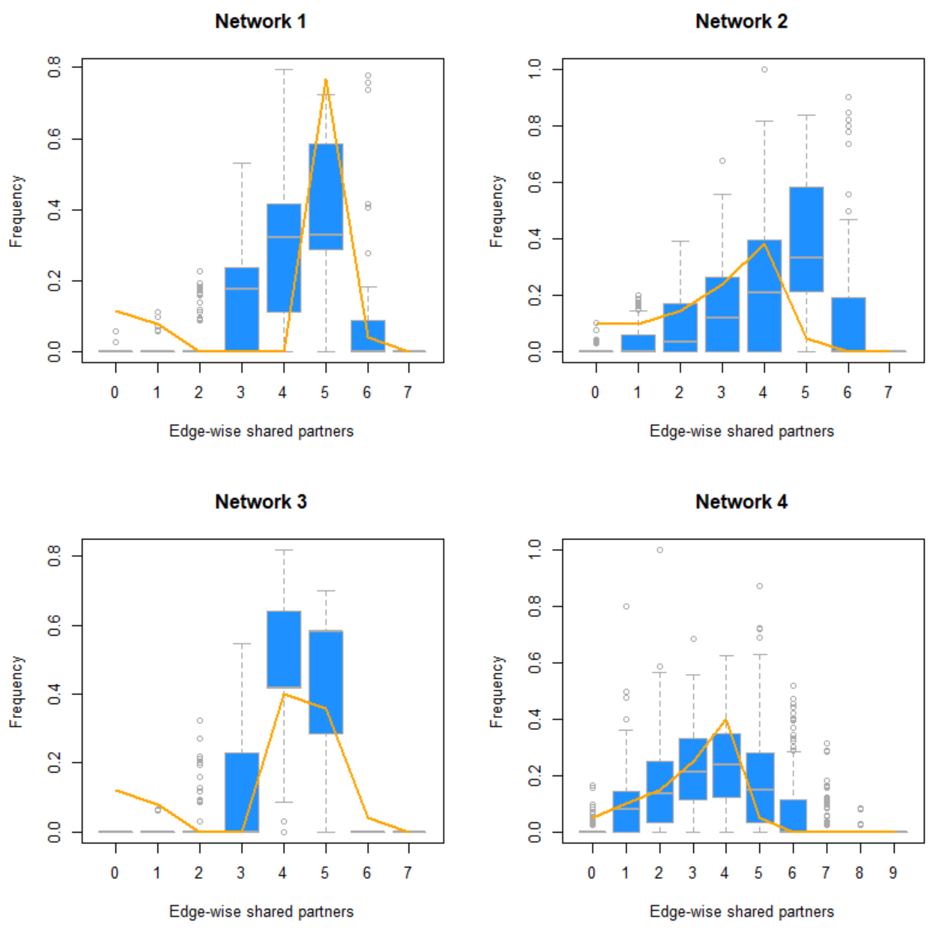


Figure 4.4: Edgewise shared partner distribution for the four networks.

The next plot shows the dyad-wise share partner distribution, which is computed as the edgewise shared partner distribution but in this case the couple i and j are ordered. Since this statistics is really close to the previous one, the outputs are close with the difference that regardless of the network considered, the observation lines are closer to the box-plots medians.

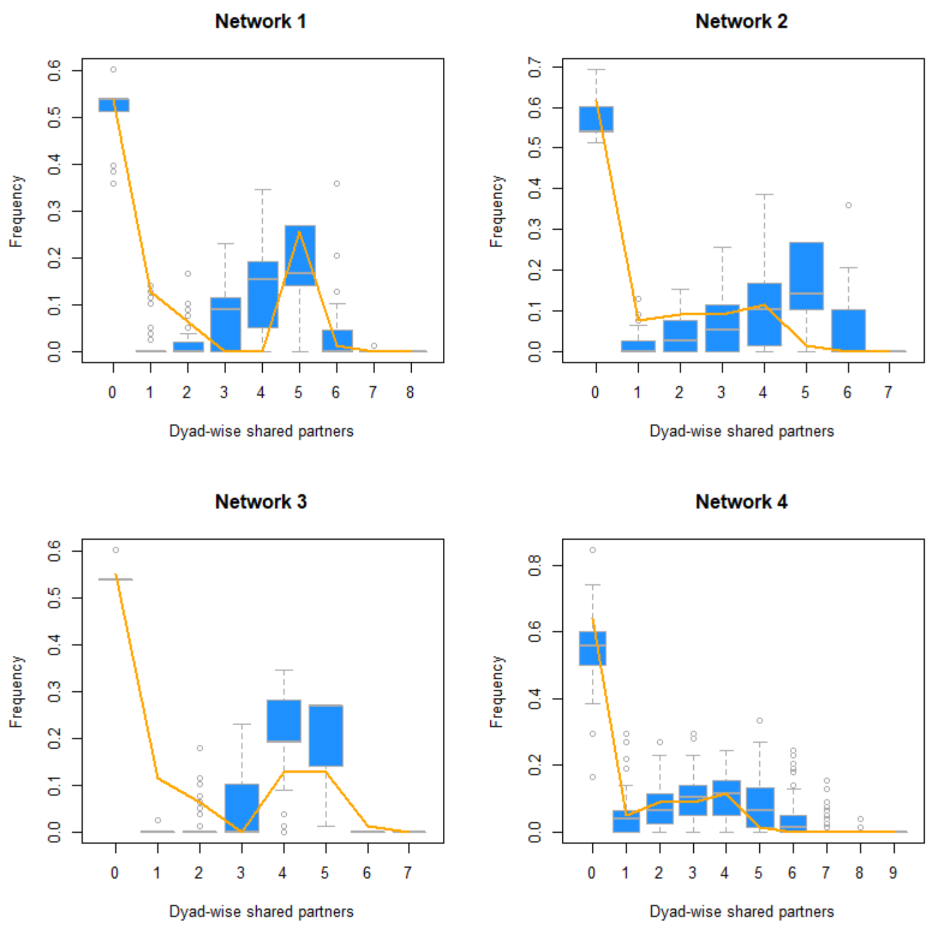


Figure 4.5: Dyadwise shared partner distribution for the four networks.

Given the fact that the degree of a node is the number of ties it has to other nodes, the subsequent degree distribution plots show that this effect is not well captured by the model. Even network 4, although the observation line is not far from the median, the quantiles distribution observed from the box-plots is pretty wide.

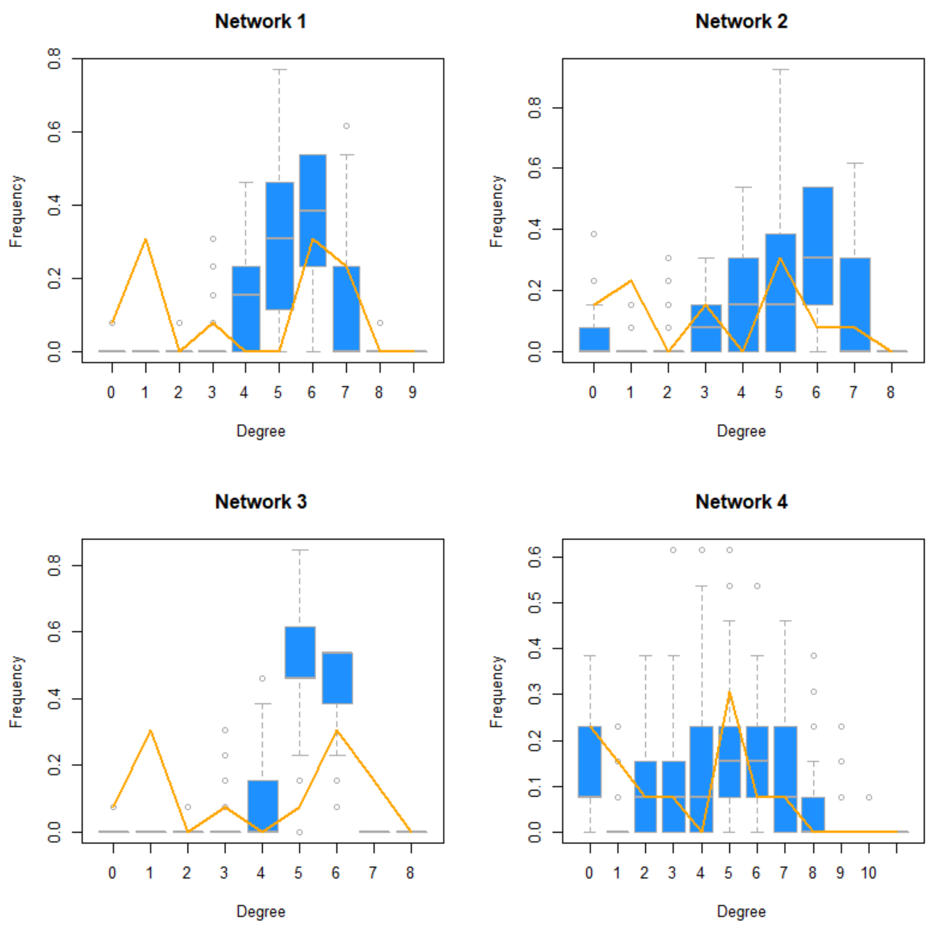


Figure 4.6: Degree distribution for the four networks.

The next plots represent the k -stars distribution. It is recalled that a k-star is a set of k edges sharing a common node (O'Malley and Onnela 2014).

The plots are actually very good for all the networks. This is important since modelling rightly the k-star statistics allows to find some subgroups of the nodes set in which a single node ties together different nodes in an indirect way.

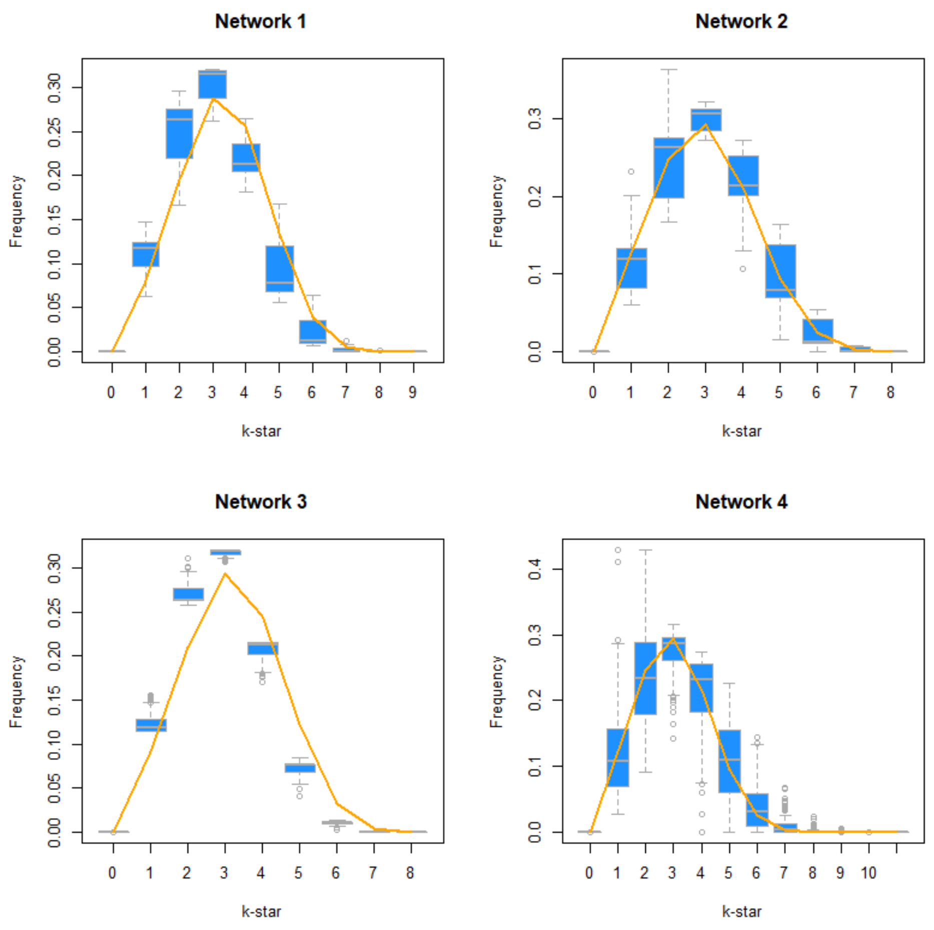


Figure 4.7: k-star distribution for the four networks.

The Geodesic distance distribution plots allows to understand whether paths with the minimum number of edges are modelled correctly in the model.From an analysis of the graphs, it can be understood that, although the general trend it is captured, only for network 4 the median is close to the observation line, i.e. the fit is good.

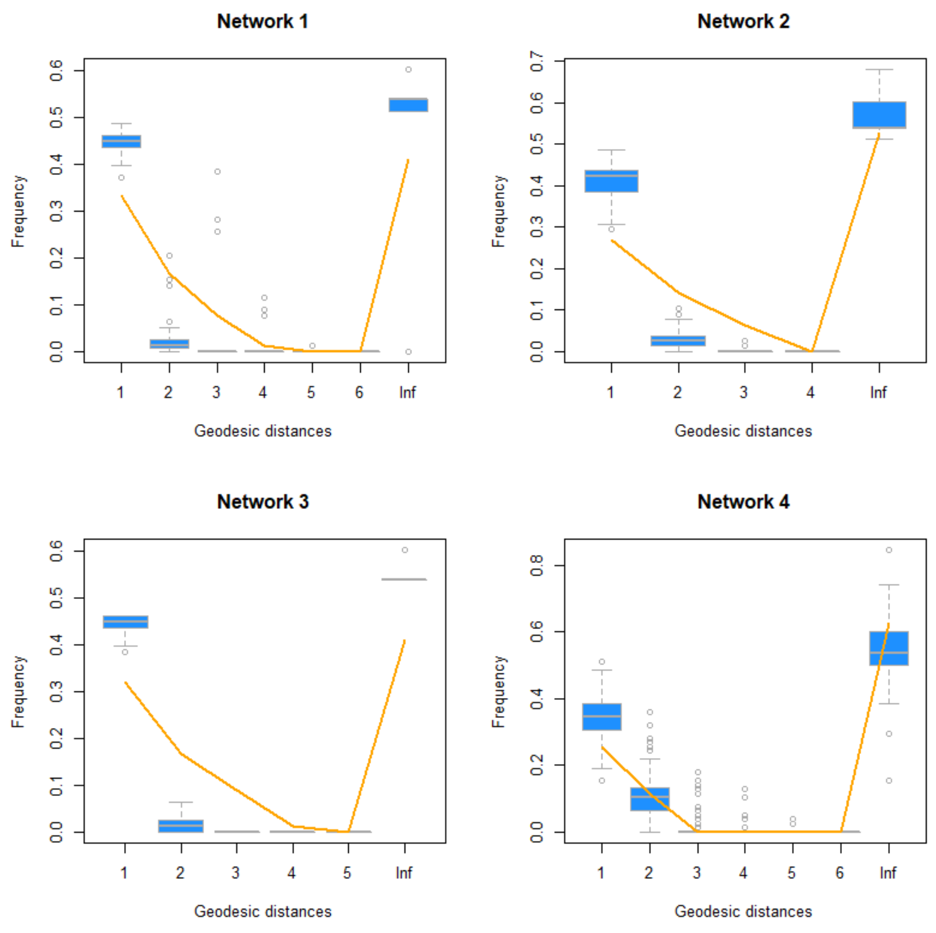


Figure 4.8: Geodesic distance distribution for the four networks.

To summarise, the model, built on twelve months data, has shown some promising results even for inferential purposes. The fit of the simulated networks to the observed one is not perfect and indeed it is not good with respect to certain measures as the degree distribution. Nevertheless the performance in terms of dyad-wise shared partner distribution and k-star distribution is good, non only for the forth network but also for the other ones. This is a good result indeed since Network 4 was built using a very high correlation threshold (0.75) meaning that it contains only strong relationships between the nodes. To be able to model some statistics of the networks with weaker relationships can, thus, be considered a satisfactory achievement for the model portrayed.

Chapter 5

Discussion

As mentioned so far, the main topic concerning the current essay is the study of the Temporal Exponential Random Graph Model (TERGM), proposed by Hanneke in 2009 and, later, extended by the contribution of Leifeld and Skyler in 2018. Belonging to the family of Exponential Random Graph Models, which dates back to the 1950s, the TERGM was developed as a longitudinal – dynamic – extension of the latter, which led to the possibility to study more complex phenomena, by taking into account the temporal dimension and evolution of the networks. Similarly to the studies meant to analyze the ERGM, the TERGM, too, were first tested in a social field, focusing on sociological issues. One of the first examples about that, is its use to model a dataset composed by 100 US senators, in order to investigate sponsors and cosponsors of different proposals during a specific legislation.

Since the main advantage regarding the exploitation of Temporal Exponential Random Graph Models concerns the opportunity to better describe and understand complex phenomena, it is relevant to test their efficiency in several different fields, to check whether their explanatory and predictive efficacy remains consistent.

This is the reason behind the decision to focus, as a purpose to the current essay, on the application of a TERGM to the financial world. Specifically, the aim has been to extend the usage of the Temporal Exponential Random model to describe the relationship between the stock performance of thirteen listed companies belonging to the technology industry, within a fifteen years' time period, starting from 2005, up to 2020.

The technological industry, in fact, can be considered a challenge in terms of statistical modelling, because its main feature is the dynamism and the fast progress pace that has always shown. To be able to understand the relationship between corporates belonging to this world seems, then, appealing and is sustained by the great amount of historical data available about them. Furthermore, the decision to consider financial indicators can be considered a second challenge, since the aleatory nature of this science field. Based on this premise, four networks have been built. To start, data about the daily stock returns of the thirteen tech companies have been collected. Thus, through the analysis of the correlation between the daily returns grouped by month and the consideration of different set thresholds, four different longitudinal networks have been structured. What distinguished them is the varying threshold value E.

Starting from these four networks, a Temporal Exponential Random Graph model has been developed in order to attempt the statistical modelling of the evolving relationships between the companies in order to understand whether there was any connection between the changes in terms of stock performance of the companies included in the network. To test the model, two different paths have been followed: prediction and inference. The first one has tested its predictive capability, which means the ability of the model to predict the connections (ties) between the companies belonging to the network (the nodes) at a certain time. Applying the concept to the financial scenario considered, the predictive task has allowed to understand whether the model is able to predict if the correlation between two stock returns exceeds a set threshold. Having this information would be meaningful and insightful as being able to determine if two stocks in a portfolio are correlated at a certain time can influence subsequent investment decisions. For example, if a high correlation between two stocks belonging to the same portfolio is predicted, it could be optimal to drop the one with the higher cost.

For what concerns the predictive test, the model analyzed has had an overall good performance. Specifically, the model has not had a good performance if tested through a rolling origin technique but, after being tuned and applied to a specific time frame – such as a certain year -, the model has reached satisfactory results.

The results gathered allow to deduce that Temporal Exponential Random Graph Models can be extended and used to study financial phenomena, too, specifically when there is the need to focus on a specific time frame. On the other hand, they may not be as efficient when a prediction is required over a large time span. This may be due to the nature of the stock market itself, which, being highly aleatory, cannot be precisely predicted. Similarly, another possible explanation could be referred to the industry chosen for the analyses conducted: in fact, the technology industry evolves at a fast pace, which could hijack the creation of a model able to perform consistently well in a long time frame. To sum up the results regarding the predictive analysis, the use of a TERGM can be extended to the analysis of correlation between financial indicators, but the model needs to be tuned for the specific time frame considered and cannot be consistently used over a large time span.

The second path followed to test the model regards the inferential dimension, which deals with the capability of the model to describe phenomena. The phenomenon that needed to be described is how close to the real world the structure of the network is. Thus, the inferential analysis has allowed to study some specific statistics, which correspond to characteristics of the network itself and to understand how trustworthy the reproduction of the network can be considered.

The results obtained have proved the model able to portray the correct distribution of single couples of companies (nodes) and common connections.

A conjoint consideration of the outcomes coming from the double method can allow to reach a good understanding of the phenomenon, gathering both information about how a specific status of the network has been reached (inferential analysis) and about the status that it will have at a certain time (predictive analysis).

To conclude, the extension of the use of the Temporal Exponential Random Graph Model beyond the sociological field is actually possible and the performance of the model can vary according to the characteristics of the science field and specific phenomenon considered. To do that, a predictive and inferential analysis need to be simultaneously considered and performed with variations depending on the nature of the observations, of the industry and, generically, of the general field of interest.

Chapter 6

Limits and future developments

The main limit of the current work is that model proposed in this essay has a lack of consistency of the prediction performance. Although it has been proven that on specific time window it has good results in terms of out-of-sample prediction, it is still it doesn't provide constantly satisfactory results over long period of time. This can be due either to two causes.

The first is the inherent evolution of the tech stock market over time, meaning that what can model the data today could not do it with data from ten years ago.

The second is the need of additional data regarding the companies, beyond the information that can be retrieved from the stock data.

Thus, a possible future development could be to search for external data to increase the information available for the network; that could allow for a better consistency of the predictions of the model.

Chapter 7

Conclusions

The analysis performed in this essay has proven that the use of the Temporal Exponential Random Graph model can go beyond the sociological field of application. However this is not valid for every science field and for every phenomenon.

In this work the focus was the financial world and specifically the stock correlation networks. It was shown that it is possible to model such networks through the use of a TERGM. The results though can vary and are not consistent for long period of time, especially in terms of predictive capability. In particular in the case of this essay, the model had to be customized for the specific period of time that had to be modelled. This way it was possible to obtain satisfactory results; however it was not possible to use the same determined model for different time windows, especially for long periods of time.

Thus, the extension of the Temporal Exponential Random Graph model to the financial field is achievable, but the goodness of the model - its performance

- depends on how the model can be tuned and adapted to the specific observed phenomenon.

Appendix A

Appendix

```

### Dataset Building ###

# Libraries Required
import numpy as np
import pandas as pd
import pandas_datareader as web
from datetime import datetime
import matplotlib.pyplot as plt
import seaborn

#select start date for correlation window as well as list of tickers
start = datetime(2005, 1, 1)
symbols_list = ["MSFT","GOOG", "AAPL", "AMZN", "SAP", "STM", "SIEGY",
    "PHG",
              "CSCO", "6758.T", "005930.KS", "6702.T", "6502.T", "^
    IXIC", "^NYA",
                "^GDAXI", "^AEX", "^N225", "^KS11", "FTSEMIB.MI"]

#array to store prices
symbols=[]

#pull price using index for each symbol in list defined above
for ticker in symbols_list:
    r = web.DataReader(ticker, 'yahoo', start)
    # add a symbol column
    r['Symbol'] = ticker
    symbols.append(r)

# concatenate into a dataframe
df = pd.concat(symbols)
df = df.reset_index()
df = df[['Date', 'Close', 'Symbol']]
df.head()
df_pivot = df.pivot('Date','Symbol','Close').reset_index()
df_pivot.head()
df_pivot.to_csv("Data.csv")
```

```

##### Complete Analysis #####

#### Packages required ####
library(readr)
library(corrr)
library(dplyr)
library(tidyr)
library(network)
library(networkDynamic)
library(statnet)
library(htmlwidgets)
library(latticeExtra)
library(tergm)
library(btergm)
library(texreg)
library(lattice)
library(PerformanceAnalytics)
library(xts)
library(ggnet)

# Loading of data which has already been cleaned from missing values
# "Month" column has been added with excel software based on the "
    Date" column
# and the special character "^" from the indexes names has been
    removed
DataG = read_csv("DataG.csv")

#### Covariates ####
```

```r
# Firstly I create the covariates that will be used in the build of the
# network as nodal attributes

cov_data = DataG[,2:22]
Mydata = DataG %>% dplyr::select(-IXIC,-NYA,-GDAXI,-AEX,-N225,-KS11)

cov_data_list = split(cov_data, as.factor(cov_data$Month))

for (i in 1:188){
  a = cov_data_list[[i]]
  cov_data_list[[i]] = a[,-2]
}

betas = matrix(nrow = 13,ncol = 188 )
betas = as.data.frame(betas)

compute_beta = function(x){
  x = xts(x = x[,-1], order.by = as.Date(x$Date))
  a = CAPM.beta(x$SMSG, x$KS11)
  b = CAPM.beta(x$TOSH, x$N225)
  c = CAPM.beta(x$FUJI, x$N225)
  d = CAPM.beta(x$SONY, x$N225)
  e = CAPM.beta(x$AAPL, x$IXIC)
  f = CAPM.beta(x$AMZN, x$IXIC)
  g = CAPM.beta(x$CSCO, x$IXIC)
  h = CAPM.beta(x$GOOG, x$IXIC)
  i = CAPM.beta(x$MSFT, x$IXIC)
  j = CAPM.beta(x$PHG, x$AEX)
  k = CAPM.beta(x$SAP, x$NYA)
  l = CAPM.beta(x$SIEGY, x$GDAXI)
  m = CAPM.beta(x$STM, x$GDAXI)   # TEMPORANEAMENTE
  out = c(a,b,c,d,e,f,g,h,i,j,k,l,m)
}

for (i in 1:188){
  temp = cov_data_list[[i]]
  betas[,i] = compute_beta(temp)
}

rnam = colnames(Mydata)
rnam = rnam[4:16]
rownames(betas) = rnam


# Now I create a dataframe for the sharpe ratios to use it as nodal
    attributes during the building of the network
# Yield_data = DataG %>% dplyr::select(-IXIC,-NYA,-GDAXI,-AEX,-N225,-
    KS11) %>% dplyr::select(-1)
# write.csv(Yield_data, "C:/Users/Yield_data.csv")
# Added year column through excel +  Yield
# Yield values source:
# https://www.macrotrends.net/2016/10-year-treasury-bond-rate-yield-
    chart
Yield_data <- read_csv("Yield_data.csv") %>% dplyr::select(-1)
yield_list <- split(Yield_data, as.factor(Yield_data$Month))

SR = matrix(nrow = 13,ncol = 188 )
SR = as.data.frame(betas)

```

```
compute_sharpe_ratio = function(x){
  x = xts(x = x[,4:17], order.by = as.Date(x$Date))
  a = SharpeRatio(x$SMSG, Rf = x$Yield, p = 0.95, FUN = "StdDev")
  b = SharpeRatio(x$TOSH, Rf = x$Yield, p = 0.95, FUN = "StdDev")
  c = SharpeRatio(x$FUJI, Rf = x$Yield, p = 0.95, FUN = "StdDev")
  d = SharpeRatio(x$SONY, Rf = x$Yield, p = 0.95, FUN = "StdDev")
  e = SharpeRatio(x$AAPL, Rf = x$Yield, p = 0.95, FUN = "StdDev")
  f = SharpeRatio(x$AMZN, Rf = x$Yield, p = 0.95, FUN = "StdDev")
  g = SharpeRatio(x$CSCO, Rf = x$Yield, p = 0.95, FUN = "StdDev")
  h = SharpeRatio(x$GOOG, Rf = x$Yield, p = 0.95, FUN = "StdDev")
  i = SharpeRatio(x$MSFT, Rf = x$Yield, p = 0.95, FUN = "StdDev")
  j = SharpeRatio(x$PHG, Rf = x$Yield, p = 0.95, FUN = "StdDev")
  k = SharpeRatio(x$SAP, Rf = x$Yield, p = 0.95, FUN = "StdDev")
  l = SharpeRatio(x$SIEGY, Rf = x$Yield, p = 0.95, FUN = "StdDev")
  m = SharpeRatio(x$STM, Rf = x$Yield, p = 0.95, FUN = "StdDev")
  out = c(a,b,c,d,e,f,g,h,i,j,k,l,m)
}

for (i in 1:188){
  temp = yield_list[[i]]
  SR[,i] = compute_sharpe_ratio(temp)
}

rnam = colnames(Mydata)
rnam = rnam[4:16]
rownames(SR) = rnam

# Now I set factors for the covariates based on quantiles
library(gtools)

```

```

c = matrix(nrow = 13, ncol = 188)
c = as.data.frame(c)

for (i in 1:188){
  a = SR[,i]
  b = quantcut(a, q = 4)
  b = (as.numeric(b))
  c[,i] = b
}


# for the betas
d = matrix(nrow = 13, ncol = 188)
d = as.data.frame(d)

for (i in 1:188){
  a = betas[,i]
  b = quantcut(a, q = 4)
  b = (as.numeric(b))
  d[,i] = b
}


#### Correlation matrix filtering and network building ####

# Means that there are 188 time points
month_names = as.vector(unique((DataG[,3])))

# All subdataframe list which will be my list of networks
# each rappresented by an adjacency matrix

# I remove the indexes
Mydata = DataG %>% dplyr::select(-IXIC,-NYA,-GDAXI,-AEX,-N225,-KS11)
net_list1 <- split(Mydata, as.factor(Mydata$Month))

# User define function that takes a dataframe as an input a dataframe
# Takes only the columns with the returns and correlates them one vs
    one
# then the correlation is filtered with respect to a treshold. If it
    is upper then the cell is set as 1
# meaning that there is a relation, otherwise it is set = 0
extract_correlation1 = function(x){
  x = x[,4:16]
  x = mutate_all(x, function(x) as.numeric(as.character(x)))
  temp = correlate(x,diagonal = 0)
  temp1 = temp[,-1]
  temp1[temp1 < 0.6] <- 0
  temp1[temp1 >= 0.6] <- 1
  out = temp1
  # out = cbind(temp$rowname, temp1) adds the rowname columns -> with
     it I don't have a square matrix
  return(out)
}

# Now the function is applied to each subdataframe (one per month)
    and eache dataframe is firstly
# converted as a matrix and then as a network object. In the end we
    got a list of adjacency matrices
# the network is directed
```

```
for (i in 1:188){
  net_list1[[i]] = extract_correlation1(net_list1[[i]])
  net_list1[[i]] = as.matrix(net_list1[[i]])
  net_list1[[i]] = network(net_list1[[i]],matrix.type="adjacency",
   directed=FALSE)
  network::set.vertex.attribute(net_list1[[i]], "Beta", as.vector(
   betas[,i])) # Add the betas as nodal
  # attributes
  network::set.vertex.attribute(net_list1[[i]],"SR", as.vector(SR[,i
   ]))
  network::set.vertex.attribute(net_list1[[i]],"SR_factorized", as.
   vector(c[,i]))
  network::set.vertex.attribute(net_list1[[i]],"Beta_factorized", as.
   vector(d[,i]))
  #nam <- paste("CorMat", i, sep = ".")   useful to change the name
   of the single object but a
  #assign(nam, a)                         different loop has to be
   built in order to exploit it
}

# Create now second network with correlation filtering = 0.7 #
net_list2 <- split(DataG, as.factor(DataG$Month))

extract_correlation2 = function(x){
  x = x[,4:16]
  x = mutate_all(x, function(x) as.numeric(as.character(x)))
  temp = correlate(x,diagonal = 1)
  temp1 = temp[,-1]
  temp1[temp1 < 0.7] <- 0
  temp1[temp1 >= 0.7] <- 1
  out = temp1
  # out = cbind(temp$rowname, temp1) adds the rowname columns -> with
     it I don't have a square matrix
  return(out)
}

for (i in 1:188){
  net_list2[[i]] = extract_correlation2(net_list2[[i]])
  net_list2[[i]] = as.matrix(net_list2[[i]])
  net_list2[[i]] = network(net_list2[[i]],matrix.type="adjacency",
    directed=FALSE)
  network::set.vertex.attribute(net_list2[[i]], "Beta", as.vector(
    betas[,i]))
  network::set.vertex.attribute(net_list2[[i]],"SR", as.vector(SR[,i
    ]))
  network::set.vertex.attribute(net_list2[[i]],"SR_factorized", as.
    vector(c[,i]))
  network::set.vertex.attribute(net_list2[[i]],"Beta_factorized", as.
    vector(d[,i]))
}

# Create third network, correlation filtering 0.65 #
net_list3 <- split(DataG, as.factor(DataG$Month))

extract_correlation3 = function(x){
  x = x[,4:16]
  x = mutate_all(x, function(x) as.numeric(as.character(x)))
  temp = correlate(x,diagonal = 1)
  temp1 = temp[,-1]
  temp1[temp1 < 0.65] <- 0
  temp1[temp1 >= 0.65] <- 1
  out = temp1
  # out = cbind(temp$rowname, temp1) adds the rowname columns -> with
     it I don't have a square matrix
  return(out)
}

for (i in 1:188){
  net_list3[[i]] = extract_correlation3(net_list3[[i]])
  net_list3[[i]] = as.matrix(net_list3[[i]])
  net_list3[[i]] = network(net_list3[[i]],matrix.type="adjacency",
    directed=FALSE)
  network::set.vertex.attribute(net_list3[[i]], "Beta", as.vector(
    betas[,i]))
  network::set.vertex.attribute(net_list3[[i]],"SR", as.vector(SR[,i
    ]))
  network::set.vertex.attribute(net_list3[[i]],"SR_factorized", as.
    vector(c[,i]))
  network::set.vertex.attribute(net_list3[[i]],"Beta_factorized", as.
    vector(d[,i]))
}

#Create fourth network, correlation filtered at .75 #
net_list4 <- split(DataG, as.factor(DataG$Month))

extract_correlation4 = function(x){
  x = x[,4:16]
  x = mutate_all(x, function(x) as.numeric(as.character(x)))
  temp = correlate(x,diagonal = 1)
  temp1 = temp[,-1]
```

```r
  temp1[temp1 < 0.75] <- 0
  temp1[temp1 >= 0.75] <- 1
  out = temp1
  # out = cbind(temp$rowname, temp1) adds the rowname columns -> with
     it I don't have a square matrix
  return(out)
}

for (i in 1:188){
  net_list4[[i]] = extract_correlation4(net_list4[[i]])
  net_list4[[i]] = as.matrix(net_list4[[i]])
  net_list4[[i]] = network(net_list4[[i]],matrix.type="adjacency",
    directed=FALSE)
  network::set.vertex.attribute(net_list4[[i]], "Beta", as.vector(
    betas[,i]))
  network::set.vertex.attribute(net_list4[[i]],"SR", as.vector(SR[,i
    ]))
  network::set.vertex.attribute(net_list4[[i]],"SR_factorized", as.
    vector(c[,i]))
  network::set.vertex.attribute(net_list4[[i]],"Beta_factorized", as.
    vector(d[,i]))
}

# Four different network have been built. That will allow to study if
     a different correlation
# threshold has an impact on the modeling and goodness of fit of the
    model


#### Exploratory and Graphical Analysis  ####
```

```
## I create a networkDynamic object to perform some visual analysis (
    plot will be added)

tech_net1 = networkDynamic(network.list = net_list)
tech_net2 = networkDynamic(network.list = net_list2)
# tech_net3 = networkDynamic(network.list = net_list3)
# tech_net4 = networkDynamic(network.list = net_list4)


# Plotting the evolution of the second network
render.d3movie(tech_net2,
plot.par=list(displaylabels=T),
output.mode = 'htmlWidget') # using htmlwidgets package here

# Networks density
n1_dens = list()
n2_dens = list()
n3_dens = list()
n4_dens = list()

for (i in 1:188){
  n1_dens[[i]] = network.density(net_list1[[i]])
  n2_dens[[i]] = network.density(net_list2[[i]])
  n3_dens[[i]] = network.density(net_list3[[i]])
  n4_dens[[i]] = network.density(net_list4[[i]])
}

mean(unlist(n1_dens))
mean(unlist(n2_dens))
mean(unlist(n3_dens))
mean(unlist(n4_dens))

EDA_net1 = net_list2[[50]]
EDA_net2 = net_list2[[51]]

# network plot

ggnet2(EDA_net1, color = "steelblue1", edge.color = "darkorange",
        edge.size = 0.8, alpha = 0.6, size = 20,label.alpha = 0.75,
    edge.alpha = 0.4, label = TRUE, label.size = 3)

ggnet2(EDA_net2, color = "steelblue1", edge.color = "darkorange",
        edge.size = 0.8, alpha = 0.6, size = 20,label.alpha = 0.75,
    edge.alpha = 0.4, label = TRUE, label.size = 3)

#### Analysis with TERGMs ####
### Prediction for last month given last year data.  ###

set.seed(123456)

# THe model is fitted on 11 months data.
net_fit = btergm(net_list4[176:187] ~edges+nodecov("SR")+
                   absdiff("SR")+nodecov("Beta")+nodecov("Beta_
    factorized")+
                   nodecov("SR_factorized")+absdiff("Beta")+
                   absdiffcat("SR_factorized")+absdiffcat("Beta_
    factorized")+
                   timecov(transform = function(t)t)+balance+
                   gwesp(decay=0, fixed=TRUE, cutoff=30)
```

```
                    +gwdsp(decay=0, fixed=TRUE, cutoff=30)
                    +gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=
    NULL)+
                      memory(type = "autoregression"),
                    R = 200, parallel = "multicore", ncpus = 6,
                    verbose = TRUE)


# The goodness of fit is computed using 200 simulation with
    prediction on the twelfth month.
gof_net = gof(net_fit,nsim = 200, target = net_list4[[188]],
                 statistics =c(rocpr,esp, dsp,deg,kstar, geodesic, triad
    .undirected
                 ),
                 coef = coef(net_fit) ,seed = 12, formula = net_list4
    [187:188] ~ edges+
                   nodecov("SR")+absdiff("SR")+nodecov("Beta")+
                   nodecov("Beta_factorized")+nodecov("SR_factorized")+
                   absdiff("Beta")+absdiffcat("SR_factorized")+
                   absdiffcat("Beta_factorized")+timecov(transform =
    function(t)t)+balance
                 +gwesp(decay=0, fixed=TRUE, cutoff=30)
                 +gwdsp(decay=0, fixed=TRUE, cutoff=30)+
                   gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=NULL)
    +
                   memory(type = "autoregression"))

plot(gof_net[[7]], main = "Network 1", col = "dodgerblue", lcol = "
    orange", outline = TRUE, median.col = "orange")
Net_auc = gof_net[[1]]$auc.roc
```

```r
Net_auc

# The model fitting has been then repeated for each network,
    alongside the plot which has to be made
# also for each statistics on which the gof has to evaluated.


# The results need to be validated even through the rolling origin
    methodology.

#### Rolling Origin ####
# In the following loops, for each network the model are fitted
    through the rolling origin
# methodoloy  using 1,2 and 3 years data to predicting the network
    state at one specific time point.
### Network 1 ###

### Annual Basis
Net1_Roc_auc1 = list()
Net1_Pr_auc1 = list()

i = 1
while (i < 178){
  set.seed(123456)
  a = i
  b = i+10
  c = i+11
  net1_fit = btergm(net_list1[a:b] ~edges+nodecov("SR")+
                          absdiff("SR")+nodecov("Beta")+nodecov("Beta_
    factorized")+
                          nodecov("SR_factorized")+absdiff("Beta")+
                          absdiffcat("SR_factorized")+absdiffcat("Beta_
    factorized")+
                          timecov(transform = function(t)t)+balance+
                          gwesp(decay=0, fixed=TRUE, cutoff=30)
                        +gwdsp(decay=0, fixed=TRUE, cutoff=30)
                        +gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=
    NULL)+
                          memory(type = "stability"),
                        R = 100, parallel = "multicore", ncpus = 6,
                        verbose = TRUE)

  gof_net1_fit = gof(net1_fit,nsim = 100, target = net_list1[[c]],
                         statistics =c(rocpr),
                         coef = coef(net1_fit) ,seed = 12, formula = net_
    list1[b:c] ~ edges+
                            nodecov("SR")+absdiff("SR")+nodecov("Beta")+
                            nodecov("Beta_factorized")+nodecov("SR_
    factorized")+
                            absdiff("Beta")+absdiffcat("SR_factorized")+
                            absdiffcat("Beta_factorized")+timecov(
    transform =  function(t)t)+balance
                         +gwesp(decay=0, fixed=TRUE, cutoff=30)
                         +gwdsp(decay=0, fixed=TRUE, cutoff=30)+
                            gwdegree(decay=0, fixed=TRUE, cutoff=30,
    levels=NULL)+
                            memory(type = "stability"))

  Net1_Roc_auc1[[i]] =gof_net1_fit[[1]]$auc.roc
  Net1_Pr_auc1[[i]] = gof_net1_fit[[1]]$auc.pr
```

```
  i = i+1
}



### Two years basis
Net1_Roc_auc2 = list()
Net1_Pr_auc2 = list()

i = 1
while (i < 166){
  set.seed(123456)
  a = i
  b = i+22
  c = i+23
  net1_fit = btergm(net_list1[a:b] ~edges+nodecov("SR")+
                              absdiff("SR")+nodecov("Beta")+nodecov("Beta_
    factorized")+
                              nodecov("SR_factorized")+absdiff("Beta")+
                              absdiffcat("SR_factorized")+absdiffcat("Beta_
    factorized")+
                              timecov(transform = function(t)t)+balance+
                              gwesp(decay=0, fixed=TRUE, cutoff=30)
                            +gwdsp(decay=0, fixed=TRUE, cutoff=30)
                            +gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=
    NULL)+
                              memory(type = "stability"),
                            R = 100, parallel = "multicore", ncpus = 6,
                            verbose = TRUE)

  gof_net1_fit = gof(net1_fit,nsim = 100, target = net_list1[[c]],
                     statistics =c(rocpr),
                     coef = coef(net1_fit) ,seed = 12, formula = net_
    list1[b:c] ~ edges+
                       nodecov("SR")+absdiff("SR")+nodecov("Beta")+
                       nodecov("Beta_factorized")+nodecov("SR_
    factorized")+
                       absdiff("Beta")+absdiffcat("SR_factorized")+
                       absdiffcat("Beta_factorized")+timecov(
    transform =  function(t)t)+balance
                     +gwesp(decay=0, fixed=TRUE, cutoff=30)
                     +gwdsp(decay=0, fixed=TRUE, cutoff=30)+
                       gwdegree(decay=0, fixed=TRUE, cutoff=30,
    levels=NULL)+
                       memory(type = "stability"))

  Net1_Roc_auc2[[i]] =gof_net1_fit[[1]]$auc.roc
  Net1_Pr_auc2[[i]] = gof_net1_fit[[1]]$auc.pr
  i = i+1
}



### Three years basis
Net1_Roc_auc3 = list()
Net1_Pr_auc3 = list()

i = 1
while (i < 154){
  set.seed(123456)
```

```
a = i
b = i+34
c = i+35
net1_fit = btergm(net_list1[a:b] ~edges+nodecov("SR")+
                         absdiff("SR")+nodecov("Beta")+nodecov("Beta_
  factorized")+
                         nodecov("SR_factorized")+absdiff("Beta")+
                         absdiffcat("SR_factorized")+absdiffcat("Beta_
  factorized")+
                         timecov(transform = function(t)t)+balance+
                         gwesp(decay=0, fixed=TRUE, cutoff=30)
                       +gwdsp(decay=0, fixed=TRUE, cutoff=30)
                       +gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=
  NULL)+
                         memory(type = "stability"),
                       R = 100, parallel = "multicore", ncpus = 6,
                       verbose = TRUE)

gof_net1_fit = gof(net1_fit,nsim = 100, target = net_list1[[c]],
                        statistics =c(rocpr),
                        coef = coef(net1_fit) ,seed = 12, formula = net_
  list1[b:c] ~ edges+
                          nodecov("SR")+absdiff("SR")+nodecov("Beta")+
                          nodecov("Beta_factorized")+nodecov("SR_
  factorized")+
                          absdiff("Beta")+absdiffcat("SR_factorized")+
                          absdiffcat("Beta_factorized")+timecov(
  transform =  function(t)t)+balance
                        +gwesp(decay=0, fixed=TRUE, cutoff=30)
                        +gwdsp(decay=0, fixed=TRUE, cutoff=30)+
                       gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=NULL)+
                       memory(type = "stability"))

  Net1_Roc_auc3[[i]] =gof_net1_fit[[1]]$auc.roc
  Net1_Pr_auc3[[i]] = gof_net1_fit[[1]]$auc.pr
  i = i+1
}



### Network 2 ###
Net2_Roc_auc1 = list()
Net2_Pr_auc1 = list()

i = 1
while (i < 178){
  set.seed(123456)
  a = i
  b = i+10
  c = i+11
  net2_fit = btergm(net_list2[a:b] ~edges+nodecov("SR")+
                      absdiff("SR")+nodecov("Beta")+nodecov("Beta_factorized")+
                      nodecov("SR_factorized")+absdiff("Beta")+
                      absdiffcat("SR_factorized")+absdiffcat("Beta_factorized")+
                      timecov(transform = sqrt)+balance+
                      gwesp(decay=0, fixed=TRUE, cutoff=30)
                    +gwdsp(decay=0, fixed=TRUE, cutoff=30)
```

```
                          +gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=
    NULL)+
                            memory(type = "stability"),
                          R = 100, parallel = "multicore", ncpus = 6,
                          verbose = TRUE)

  gof_net2_fit = gof(net2_fit,nsim = 100, target = net_list2[[c]],
                           statistics =c(rocpr),
                           coef = coef(net2_fit) ,seed = 12, formula = net_
    list2[b:c] ~ edges+
                             nodecov("SR")+absdiff("SR")+nodecov("Beta")+
                             nodecov("Beta_factorized")+nodecov("SR_
    factorized")+
                             absdiff("Beta")+absdiffcat("SR_factorized")+
                             absdiffcat("Beta_factorized")+timecov(
    transform =  sqrt)+balance
                           +gwesp(decay=0, fixed=TRUE, cutoff=30)
                           +gwdsp(decay=0, fixed=TRUE, cutoff=30)+
                             gwdegree(decay=0, fixed=TRUE, cutoff=30,
    levels=NULL)+
                             memory(type = "stability"))

  Net2_Roc_auc1[[i]] =gof_net2_fit[[1]]$auc.roc
  Net2_Pr_auc1[[i]] = gof_net2_fit[[1]]$auc.pr
  i = i+1
}


### Two years basis
Net2_Roc_auc2 = list()
```

```
Net2_Pr_auc2 = list()

i = 1
while (i < 166){
  set.seed(123456)
  a = i
  b = i+22
  c = i+23
  net2_fit = btergm(net_list2[a:b] ~edges+nodecov("SR")+
                      absdiff("SR")+nodecov("Beta")+nodecov("Beta_factorized")+
                      nodecov("SR_factorized")+absdiff("Beta")+
                      absdiffcat("SR_factorized")+absdiffcat("Beta_factorized")+
                      timecov(transform = function(t)t)+balance+
                      gwesp(decay=0, fixed=TRUE, cutoff=30)
                    +gwdsp(decay=0, fixed=TRUE, cutoff=30)
                    +gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=NULL)+
                      memory(type = "stability"),
                    R = 100, parallel = "multicore", ncpus = 6,
                    verbose = TRUE)

  gof_net2_fit = gof(net2_fit,nsim = 100, target = net_list2[[c]],
                     statistics =c(rocpr),
                     coef = coef(net2_fit) ,seed = 12, formula = net_list2[b:c] ~ edges+
                       nodecov("SR")+absdiff("SR")+nodecov("Beta")+
                       nodecov("Beta_factorized")+nodecov("SR_factorized")+
                          absdiff("Beta")+absdiffcat("SR_factorized")+
                          absdiffcat("Beta_factorized")+timecov(
    transform =  function(t)t)+balance
                        +gwesp(decay=0, fixed=TRUE, cutoff=30)
                        +gwdsp(decay=0, fixed=TRUE, cutoff=30)+
                          gwdegree(decay=0, fixed=TRUE, cutoff=30,
    levels=NULL)+
                          memory(type = "stability"))

  Net2_Roc_auc2[[i]] =gof_net2_fit[[1]]$auc.roc
  Net2_Pr_auc2[[i]] = gof_net2_fit[[1]]$auc.pr
  i = i+1
}


### Three years basis
Net2_Roc_auc3 = list()
Net2_Pr_auc3 = list()

i = 1
while (i < 154){
  set.seed(123456)
  a = i
  b = i+34
  c = i+35
  net2_fit = btergm(net_list2[a:b] ~edges+nodecov("SR")+
                         absdiff("SR")+nodecov("Beta")+nodecov("Beta_
    factorized")+
                         nodecov("SR_factorized")+absdiff("Beta")+
                         absdiffcat("SR_factorized")+absdiffcat("Beta_
    factorized")+
                         timecov(transform = function(t)t)+balance+
                         gwesp(decay=0, fixed=TRUE, cutoff=30)
                       +gwdsp(decay=0, fixed=TRUE, cutoff=30)
                       +gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=
    NULL)+
                         memory(type = "stability"),
                       R = 100, parallel = "multicore", ncpus = 6,
                       verbose = TRUE)

  gof_net2_fit = gof(net2_fit,nsim = 100, target = net_list2[[c]],
                        statistics =c(rocpr),
                        coef = coef(net2_fit) ,seed = 12, formula = net_
    list2[b:c] ~ edges+
                          nodecov("SR")+absdiff("SR")+nodecov("Beta")+
                          nodecov("Beta_factorized")+nodecov("SR_
    factorized")+
                          absdiff("Beta")+absdiffcat("SR_factorized")+
                          absdiffcat("Beta_factorized")+timecov(
    transform =  function(t)t)+balance
                        +gwesp(decay=0, fixed=TRUE, cutoff=30)
                        +gwdsp(decay=0, fixed=TRUE, cutoff=30)+
                          gwdegree(decay=0, fixed=TRUE, cutoff=30,
    levels=NULL)+
                          memory(type = "stability"))

  Net2_Roc_auc3[[i]] =gof_net2_fit[[1]]$auc.roc
  Net2_Pr_auc3[[i]] = gof_net2_fit[[1]]$auc.pr
  i = i+1
```

```
}

### Network 3 ###

### Annual basis
Net3_Roc_auc1 = list()
Net3_Pr_auc1 = list()

i = 1
while (i < 178){
  set.seed(123456)
  a = i
  b = i+10
  c = i+11
  net3_fit = btergm(net_list3[a:b] ~edges+nodecov("SR")+
                      absdiff("SR")+nodecov("Beta")+nodecov("Beta_factorized")+
                      nodecov("SR_factorized")+absdiff("Beta")+
                      absdiffcat("SR_factorized")+absdiffcat("Beta_factorized")+
                      timecov(transform = function(t)t)+balance+
                      gwesp(decay=0, fixed=TRUE, cutoff=30)
                    +gwdsp(decay=0, fixed=TRUE, cutoff=30)
                    +gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=NULL)+
                      memory(type = "stability"),
                    R = 100, parallel = "multicore", ncpus = 6,
                    verbose = TRUE)

  gof_net3_fit = gof(net3_fit,nsim = 100, target = net_list3[[c]],
                        statistics =c(rocpr),
                        coef = coef(net3_fit) ,seed = 12, formula = net_
    list3[b:c] ~ edges+
                          nodecov("SR")+absdiff("SR")+nodecov("Beta")+
                          nodecov("Beta_factorized")+nodecov("SR_
    factorized")+
                          absdiff("Beta")+absdiffcat("SR_factorized")+
                          absdiffcat("Beta_factorized")+timecov(
    transform =  function(t)t)+balance
                        +gwesp(decay=0, fixed=TRUE, cutoff=30)
                        +gwdsp(decay=0, fixed=TRUE, cutoff=30)+
                          gwdegree(decay=0, fixed=TRUE, cutoff=30,
    levels=NULL)+
                          memory(type = "stability"))

  Net3_Roc_auc1[[i]] =gof_net3_fit[[1]]$auc.roc
  Net3_Pr_auc1[[i]] = gof_net3_fit[[1]]$auc.pr
  i = i+1
}



### Two years basis
Net3_Roc_auc2 = list()
Net3_Pr_auc2 = list()

i = 1
while (i < 166){
  set.seed(123456)
  a = i
```

```
b = i+22
c = i+23
net3_fit = btergm(net_list3[a:b] ~edges+nodecov("SR")+
                      absdiff("SR")+nodecov("Beta")+nodecov("Beta_
  factorized")+
                      nodecov("SR_factorized")+absdiff("Beta")+
                      absdiffcat("SR_factorized")+absdiffcat("Beta_
  factorized")+
                      timecov(transform = function(t)t)+balance+
                      gwesp(decay=0, fixed=TRUE, cutoff=30)
                    +gwdsp(decay=0, fixed=TRUE, cutoff=30)
                    +gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=
  NULL)+
                      memory(type = "stability"),
                    R = 100, parallel = "multicore", ncpus = 6,
                    verbose = TRUE)

gof_net3_fit = gof(net1_fit,nsim = 100, target = net_list3[[c]],
                     statistics =c(rocpr),
                     coef = coef(net3_fit) ,seed = 12, formula = net_
  list3[b:c] ~ edges+
                       nodecov("SR")+absdiff("SR")+nodecov("Beta")+
                       nodecov("Beta_factorized")+nodecov("SR_
  factorized")+
                       absdiff("Beta")+absdiffcat("SR_factorized")+
                       absdiffcat("Beta_factorized")+timecov(
  transform =  function(t)t)+balance
                     +gwesp(decay=0, fixed=TRUE, cutoff=30)
                     +gwdsp(decay=0, fixed=TRUE, cutoff=30)+
                       gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=NULL)+
                       memory(type = "stability"))

  Net3_Roc_auc2[[i]] =gof_net3_fit[[1]]$auc.roc
  Net3_Pr_auc2[[i]] = gof_net3_fit[[1]]$auc.pr
  i = i+1
}



### Three years basis
Net3_Roc_auc3 = list()
Net3_Pr_auc3 = list()

i = 1
while (i < 154){
  set.seed(123456)
  a = i
  b = i+34
  c = i+35
  net3_fit = btergm(net_list3[a:b] ~edges+nodecov("SR")+
                      absdiff("SR")+nodecov("Beta")+nodecov("Beta_factorized")+
                      nodecov("SR_factorized")+absdiff("Beta")+
                      absdiffcat("SR_factorized")+absdiffcat("Beta_factorized")+
                      timecov(transform = function(t)t)+balance+
                      gwesp(decay=0, fixed=TRUE, cutoff=30)
                    +gwdsp(decay=0, fixed=TRUE, cutoff=30)
```

```
                        +gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=
  NULL)+
                          memory(type = "stability"),
                        R = 100, parallel = "multicore", ncpus = 6,
                        verbose = TRUE)

  gof_net3_fit = gof(net3_fit,nsim = 100, target = net_list3[[c]],
                         statistics =c(rocpr),
                         coef = coef(net3_fit) ,seed = 12, formula = net_
  list3[b:c] ~ edges+
                           nodecov("SR")+absdiff("SR")+nodecov("Beta")+
                           nodecov("Beta_factorized")+nodecov("SR_
  factorized")+
                           absdiff("Beta")+absdiffcat("SR_factorized")+
                           absdiffcat("Beta_factorized")+timecov(
  transform =  function(t)t)+balance
                         +gwesp(decay=0, fixed=TRUE, cutoff=30)
                         +gwdsp(decay=0, fixed=TRUE, cutoff=30)+
                           gwdegree(decay=0, fixed=TRUE, cutoff=30,
  levels=NULL)+
                           memory(type = "stability"))

  Net3_Roc_auc3[[i]] =gof_net3_fit[[1]]$auc.roc
  Net3_Pr_auc3[[i]] = gof_net3_fit[[1]]$auc.pr
  i = i+1
}



### Network 4 ###

### Annual basis
Net4_Roc_auc1 = list()
Net4_Pr_auc1 = list()

i = 1
while (i < 178){
  set.seed(123456)
  a = i
  b = i+10
  c = i+11
  net4_fit = btergm(net_list4[a:b] ~edges+nodecov("SR")+
                      absdiff("SR")+nodecov("Beta")+nodecov("Beta_
   factorized")+
                      nodecov("SR_factorized")+absdiff("Beta")+
                      absdiffcat("SR_factorized")+absdiffcat("Beta_
   factorized")+
                      timecov(transform = function(t)t)+balance+
                      gwesp(decay=0, fixed=TRUE, cutoff=30)
                    +gwdsp(decay=0, fixed=TRUE, cutoff=30)
                    +gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=
   NULL)+
                      memory(type = "stability"),
                    R = 100, parallel = "multicore", ncpus = 6,
                    verbose = TRUE)

  gof_net4_fit = gof(net4_fit,nsim = 100, target = net_list4[[c]],
                     statistics =c(rocpr),
                     coef = coef(net4_fit) ,seed = 12, formula = net_
   list4[b:c] ~ edges+
                        nodecov("SR")+absdiff("SR")+nodecov("Beta")+
                        nodecov("Beta_factorized")+nodecov("SR_
  factorized")+
                        absdiff("Beta")+absdiffcat("SR_factorized")+
                        absdiffcat("Beta_factorized")+timecov(
  transform =  function(t)t)+balance
                      +gwesp(decay=0, fixed=TRUE, cutoff=30)
                      +gwdsp(decay=0, fixed=TRUE, cutoff=30)+
                        gwdegree(decay=0, fixed=TRUE, cutoff=30,
  levels=NULL)+
                        memory(type = "stability"))

  Net4_Roc_auc1[[i]] =gof_net4_fit[[1]]$auc.roc
  Net4_Pr_auc1[[i]] = gof_net4_fit[[1]]$auc.pr
  i = i+1
}



### Two years basis
Net4_Roc_auc2 = list()
Net4_Pr_auc2 = list()

i = 1
while (i < 166){
  set.seed(123456)
  a = i
  b = i+22
  c = i+23
  net4_fit = btergm(net_list4[a:b] ~edges+nodecov("SR")+
                              absdiff("SR")+nodecov("Beta")+nodecov("Beta_
factorized")+
                              nodecov("SR_factorized")+absdiff("Beta")+
                              absdiffcat("SR_factorized")+absdiffcat("Beta_
factorized")+
                              timecov(transform = function(t)t)+balance+
                              gwesp(decay=0, fixed=TRUE, cutoff=30)
                            +gwdsp(decay=0, fixed=TRUE, cutoff=30)
                            +gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=
NULL)+
                              memory(type = "stability"),
                            R = 100, parallel = "multicore", ncpus = 6,
                            verbose = TRUE)

gof_net4_fit = gof(net4_fit,nsim = 100, target = net_list4[[c]],
                             statistics =c(rocpr),
                             coef = coef(net4_fit) ,seed = 12, formula = net_
list4[b:c] ~ edges+
                               nodecov("SR")+absdiff("SR")+nodecov("Beta")+
                               nodecov("Beta_factorized")+nodecov("SR_
factorized")+
                               absdiff("Beta")+absdiffcat("SR_factorized")+
                               absdiffcat("Beta_factorized")+timecov(
transform =  function(t)t)+balance
                             +gwesp(decay=0, fixed=TRUE, cutoff=30)
                             +gwdsp(decay=0, fixed=TRUE, cutoff=30)+
                               gwdegree(decay=0, fixed=TRUE, cutoff=30,
levels=NULL)+
                               memory(type = "stability"))

```

```
  Net4_Roc_auc2[[i]] =gof_net4_fit[[1]]$auc.roc
  Net4_Pr_auc2[[i]] = gof_net4_fit[[1]]$auc.pr
  i = i+1
}



### Three years basis
Net4_Roc_auc3 = list()
Net4_Pr_auc3 = list()

i = 1
while (i < 154){
  set.seed(123456)
  a = i
  b = i+34
  c = i+35
  net4_fit = btergm(net_list4[a:b] ~edges+nodecov("SR")+
                          absdiff("SR")+nodecov("Beta")+nodecov("Beta_
   factorized")+
                          nodecov("SR_factorized")+absdiff("Beta")+
                          absdiffcat("SR_factorized")+absdiffcat("Beta_
   factorized")+
                          timecov(transform = function(t)t)+balance+
                          gwesp(decay=0, fixed=TRUE, cutoff=30)
                        +gwdsp(decay=0, fixed=TRUE, cutoff=30)
                        +gwdegree(decay=0, fixed=TRUE, cutoff=30, levels=
   NULL)+
                          memory(type = "stability"),
                        R = 100, parallel = "multicore", ncpus = 6,
                    verbose = TRUE)

  gof_net4_fit = gof(net4_fit,nsim = 100, target = net_list4[[c]],
                     statistics =c(rocpr),
                     coef = coef(net4_fit) ,seed = 12, formula = net_
   list4[b:c] ~ edges+
                       nodecov("SR")+absdiff("SR")+nodecov("Beta")+
                       nodecov("Beta_factorized")+nodecov("SR_
   factorized")+
                       absdiff("Beta")+absdiffcat("SR_factorized")+
                       absdiffcat("Beta_factorized")+timecov(
   transform =  function(t)t)+balance
                     +gwesp(decay=0, fixed=TRUE, cutoff=30)
                     +gwdsp(decay=0, fixed=TRUE, cutoff=30)+
                       gwdegree(decay=0, fixed=TRUE, cutoff=30,
   levels=NULL)+
                       memory(type = "stability"))

  Net4_Roc_auc3[[i]] =gof_net4_fit[[1]]$auc.roc
  Net4_Pr_auc3[[i]] = gof_net4_fit[[1]]$auc.pr
  i = i+1
}



mean(unlist(Net1_Roc_auc1))
mean(unlist(Net1_Roc_auc2))
mean(unlist(Net1_Roc_auc3))

mean(unlist(Net2_Roc_auc1))
mean(unlist(Net2_Roc_auc2))
mean(unlist(Net2_Roc_auc3))

mean(unlist(Net3_Roc_auc1))
mean(unlist(Net3_Roc_auc2))
mean(unlist(Net3_Roc_auc3))

mean(unlist(Net4_Roc_auc1))
mean(unlist(Net4_Roc_auc2))
hist(unlist(Net4_Roc_auc3))

### Histogram of the distributions of the AUC for the Roc curve
# of the fourth network based on three years data
library(ggplot2)
a = as.data.frame(cbind(xaxis, unlist(Net4_Roc_auc3)))
xaxis = seq(1,153,1)
a = as.data.frame(cbind(xaxis, unlist(Net4_Roc_auc3)))

ggplot(a, aes(x= V2)) + geom_histogram(bins = 25, alpha = 0.6, color
   = "dodgerblue", fill = "dodgerblue")+
  geom_vline(aes(xintercept=mean(V2)), size = 2, color = "orange",
   alpha = 0.7)+
  labs(x="Roc AUC Network 4")+
  xlim(c(0,1))
```

Bibliography

Aalen, Odd, Ørnulf Borgan, and Hakon Gjessing (Jan. 2008). *Survival and Event History Analysis: A Process Point of View*. Springer.

Bianchi, Federica, Francesco Bartolucci, Stefano Peluso, and Antonietta Mira (2020). "Longitudinal Networks of Dyadic Relations Using Latent Trajectories: Evidence from the European Interbank Market". *Journal of the Royal Statistical Society* Series C.

Butts, Carter (July 2008). "A relational event framework for social action". *Sociological Methodology* 38, pp. 155–200.

Butts, Carter, Miruna Petrescu-Prahova, and Remy Cross (Apr. 2007). "Responder Communication Networks in the World Trade Center Disaster: Implications for Modeling of Communication Within Emergency Settings". *Mathematical Sociology* 31, pp. 121–147.

Cox, D. R. (1972). "Regression Models and Life-Tables". *Journal of the Royal Statistical Society. Series B (Methodological)* 34.2, pp. 187–220.

Desmarais, B.A. and S.J. Cranmer (2012). "Statistical mechanics of networks: Estimation and uncertainty". *Physica A: Statistical Mechanics and its Applications* 391.4, pp. 1865–1876.

Dosdall, Henrik (Jan. 1972). "Kapferer: Strategy and Transaction in an African Factory", pp. 289–292.

Durante, Daniele and David Dunson (Oct. 2014). "Nonparametric Bayes dynamic modeling of relational data". *Biometrika* 101, pp. 883–898.

Erdös, P. and A. Rényi (1959). "On Random Graphs I". *Publicationes Mathematicae Debrecen* 6, p. 290.

— (1959). "On Random Graphs I". *Publicationes Mathematicae Debrecen* 6, pp. 290–297.

Fienberg, Stephen (Oct. 2012). "A Brief History of Statistical Models for Network Analysis and Open Challenges". *Journal of Computational and Graphical Statistics* 21, pp. 825–839.

Frank, Ove and David Strauss (1986). "Markov Graphs". *Journal of the American Statistical Association* 81.395, pp. 832–842.

Gilbert, E. N. (1959). "Random Graphs". *Annals of Mathematical Statistics*.

Goldenberg, Anna, Alice Zheng, Stephen Fienberg, and Edoardo Airoldi (Dec. 2009). "A Survey of Statistical Network Models". *Foundations and Trends in Machine Learning* 2.

Hanneke, Steve, Wenjie Fu, and Eric Xing (2009). *Discrete Temporal Models of Social Networks*.

He, Yanjun and Peter D. Hoff (2017). *Multiplicative Coevolution Regression Models for Longitudinal Networks and Nodal Attributes*.

Holland, Paul W., Kathryn Blackmond Laskey, and Samuel Leinhardt (1983). "Stochastic blockmodels: First steps". *Social Networks* 5.2, pp. 109–137.

Holland, Paul W. and Samuel Leinhardt (1977). "A dynamic model for social networks". *The Journal of Mathematical Sociology* 5.1, pp. 5–20.

Holland, Paul and Samuel Leinhardt (Mar. 1981). "An Exponential Family of Probability Distributions for Directed Graphs". *Journal of The American Statistical Association* 76, pp. 33–50.

Hunter, David R and Mark S Handcock (2006). "Inference in Curved Exponential Family Models for Networks". *Journal of Computational and Graphical Statistics* 15.3, pp. 565–583.

Ishiguro, Katsuhiko, Tomoharu Iwata, Naonori Ueda, and Joshua Tenenbaum (Jan. 2010). "Dynamic Infinite Relational Model for Time-varying Relational Data Analysis", pp. 919–927.

Kemp, Charles, Joshua Tenenbaum, Thomas Griffiths, Takeshi Yamada, and Naonori Ueda (Jan. 2006). "Learning Systems of Concepts with an Infinite Relational Model". *Cognitive Science* 21.

Kolaczyk, Eric D. (2009). *Statistical Analysis of Network Data: Methods and Models*. 1st. Springer Publishing Company, Incorporated.

Krivitsky, Pavel N (2018). *ergm*.

Krivitsky, Pavel and Mark Handcock (Jan. 2014). "A Separable Model for Dynamic Networks". *Journal of the Royal Statistical Society. Series B, Statistical methodology* 76, pp. 29–46.

"Laplace Operator" (n.d.). *Encyclopedia of Mathematics* ().

Leifeld, Philip and Skyler J. Cranmer (2015). *A Theoretical and Empirical Comparison of the Temporal Exponential Random Graph Model and the Stochastic Actor-Oriented Model*.

Leifeld, Philip, Skyler Cranmer, and Bruce Desmarais (2018). "Temporal Exponential Random Graph Models with btergm: Estimation and Bootstrap Confidence Intervals". *Journal of Statistical Software, Articles* 83.6, pp. 1–36.

Matias, Catherine, Tabea Rebafka, and Fanny Villers (2015). *A semiparametric extension of the stochastic block model for longitudinal networks*.

Milgram, Stanley (1967). "The small world problem". *Psychology Today*.

Miller, Bradley N. and David L. Ranum (2011). *Problem Solving with Algorithms and Data Structures Using Python SECOND EDITION*. 2nd. USA: Franklin, Beedle Associates Inc.

Moreno, Jacob Levi (1934). "Who Shall Survive?" *Nervous and Mental Disease Publishing Company*.

Nowicki, Krzysztof and Tom A. B Snijders (2001). "Estimation and Prediction for Stochastic Blockstructures". *Journal of the American Statistical Association* 96.455, pp. 1077–1087.

O'Malley, James and Jukka-Pekka Onnela (Mar. 2014). "Topics in social network analysis and network science".

Pattison, P. (1996). "LOGIT MODELS AND LOGISTIC REGRESSIONS FOR SOCIAL NETWORKS: I. AN INTRODUCTION TO MARKOV GRAPHS AND p* STANLEY WASSERMAN UNIVERSITY OF ILLINOIS".

Perry, Patrick O. and Patrick J. Wolfe (Mar. 2013). "Point process modelling for directed interaction networks". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 75.5, pp. 821–849.

Raghavan, Prabhakar, Sridhar Rajagopalan, D. Sivakumar, Andrew Tomkins, and Eli Upfal (Mar. 2002). "Stochastic Models for the Web Graph". *In Symposium on Foundations of Computer Science FOCS, IEEE.*

Robins, Garry and Philippa Pattison (Aug. 2001). "Random Graph Models for Temporal Processes in Social Networks". *The Journal of Mathematical Sociology* 25, pp. 5–41.

Sarkar, Purnamrita and Andrew Moore (Dec. 2005). "Dynamic Social Network Analysis using Latent Space Models." Vol. 7.

Sharpe, William F. (1966). "Mutual Fund Performance". *The Journal of Business* 39.1, pp. 119–138.

Snijders, Tom (Apr. 1996). "Stochastic Actor-Oriented Models for Network Change". *Journal of Mathematical Sociology - J MATH SOCIOL* 21, pp. 149–172.

— (Feb. 2001). "The Statistical Evaluation of Social Network Dynamics". *Sociological Methodology* 31.

— (June 2002). "Markov Chain Monte Carlo Estimation of Exponential Random Graph Models". *Journal of Social Structure* 3.

— (Mar. 2017). "Stochastic Actor-Oriented Models for Network Dynamics". *Annual Review of Statistics and Its Application* 4.

Snijders, Tom A. B. (2006). "Statistical Methods for Network Dynamics".

Snijders, Tom, Christian Steglich, and Michael Schweinberger (Jan. 2007). "Modeling the co-evolution of networks and behavior". *Longitudinal Models in the Behavioral and Related Sciences.*

Solomonoff, R. and A. Rapoport (1951). "Connectivity of random nets".

Strauss, David and Michael Ikeda (1990). "Pseudolikelihood Estimation for Social Networks". *Journal of the American Statistical Association* 85.409, pp. 204–212.

Tashman, Leonard J. (2000). "Out-of-sample tests of forecasting accuracy: an analysis and review". *International Journal of Forecasting* 16.4. The M3- Competition, pp. 437–450.

Uddin, S., M. Piraveenan, K. S. K. Chung, and L. Hossain (2013). "Topological analysis of longitudinal networks". *2013 46th Hawaii International Conference on System Sciences*.

Uddin, Shahadat, Nazim Choudhury, Mahendra Piraveenan, and Kon Shing Kenneth Chung (2017). "Exploring Actor-Level Dynamics in Longitudinal Networks: The State of the Art". *Encyclopedia of Social Network Analysis and Mining*. Springer New York.

Vu, Duy, Arthur Asuncion, David Hunter, and Padhraic Smyth (Jan. 2011). "Dynamic Egocentric Models for Citation Networks.", pp. 857–864.

— (May 2012). "Continuous-Time Regression Models for Longitudinal Networks". *Proceedings of the 24th International Conference on Neural Information Processing Systems* 24.

Wasserman, Stan (Mar. 1987). "Conformity of two sociometric relations". *Psychometrika* 52, pp. 3–18.

Wasserman, Stan and Dawn Iacobucci (Feb. 1988). "Sequential social network data". *Psychometrika* 53, pp. 261–282.

Wasserman, Stanley S. (1980). "A Stochastic Model for Directed Graphs with Transition Rates Determined by Reciprocity". *Sociological Methodology* 11, pp. 392–412.

Wasserman, Stanley and Katherine Faust (1994). *Social Network Analysis: Methods and Applications*. Structural Analysis in the Social Sciences. Cambridge University Press.

Watts, D. and S. Strogatz (1998). “Collective dynamics of ’small-world’ networks”. *Nature*.