**THEORETICAL BACKGROUND**

In many problems it is helpful and even necessary to study the relationships between entities of a system since we find them interconnected forming networks in which some small changes on an element can propagate and affect the entire system. To do so we use graph, and more precisely networks, theories which represent each individuals or actors by nodes and the connection or relationships as edges between the nodes.

We find a large list of examples in which it is necessary to apply network theory on different and wide subjects as social science, finance, biology or climatology.

Studying such networks can reveal lots of information like how an event can trigger a topological change of the entire network, how entities of the network can depend on each other’s states or share similar properties and organize themselves into groups or which individuals are the main agents that lead the evolution of the network.

Classically these analysis have been carried in the simplest way assuming static (non-varying) networks, however this is usually a simplification since in most of the cases the relationships between agents in a network evolve over time as the nodes’ states change. The internet, biological processes or the economy are some examples of networks evolving over time, shrinking or creating new links between the nodes.

Although there is wide literature and methods on modelling static networks it has not been until recent years that interest has grown on studying dynamic or time-varying networks ([1], [3], [5], [6], [7], [8], [11]).

Usually, only time series measurements, such as microarray, stock price, etc., of the activity of the nodal entities, but not their linkage status, are available. The goal is to recover the latent time-varying networks with temporal resolution up to every single time point based on time series measurements.

To be able to understand the current state of the research on the field and the strengths, weakness and challenges of the techniques proposed this work is going to analyze several algorithms, focusing mainly on TESLA (from the acronym TESLLOR, which stands for temporally smoothed *l*1*-*regularized logistic regression), built to recover the structure of time-varying networks over a fixed set of nodes from their time series nodal attributes [1].

# network main properties

Depending on the problem under consideration, it may be convenient to study the network from different points of view, focusing on relationships between the entities (edge-centric), one or more entities (vertex-centric) or the topological properties of the network as a whole (graph-centric) [5].

When studying edge-centric networks some important measures include transitivity (similar to the mathematical property: if *u* connected to *v* and *v* connected to *w*, then *u* connected to *w*) and reciprocity (when we find loops in a network with 2 edges length, so pairs of nodes that have edges running in both directions).

Meanwhile in vertex-centric networks centrality measures are used to quantify the relevance of the different nodes in a network. The most basic example is the degree of centrality of the nodes: the number of edges attached to them. Nodes with high degree are called hubs and obviously play an important role in the system.

Regarding graph-centric approach we are able to analyze the general shape by studying the clusters in which the vertices organize themselves, many techniques like the well-known hierarchical clustering are useful in this task showing important properties of the network like its tendency to follow an assortative or dissortative mixing pattern.

All these properties [9] give us important information to understand the behavior of the system.

# dynamic networks

Latest years the science of networks developed three important aspects that define this field nowadays: it is used to model real-world problems, it frequently assumes the networks are not static and it aims to understand networks not just as topological objects but also as the framework upon distributed dynamical systems are built [3].

Then studying networks as dynamical systems is the best way to understand evolving underlying processes in complex systems that can trigger critic events reshaping the network and changing its properties, one example of this is how a disease spreads: the structure of a network through which a contagious agent is transmitted can have a dramatic impact on outcomes at the level of entire populations [3].

# learning networks

Looking at their topology we find that networks are organized in two main classes: scale-free networks and random graphs.

Scale-free network term was coined to describe the networks that show a power law degree distribution (distribution of the connections over the whole graph) after Barabasi mapped a portion of the World Wide Web [2] finding that it followed this distribution of the links and that some nodes had many more links than others (hubs). Then the most notable characteristic in scale-free networks is that it is relatively common to find vertices with a much higher degree (links) than the average. They also have other important features like robustness due to its hierarchical organization or clustering coefficient that may help to understand networks following this distribution.

A random graph is obtained by starting with a set of isolated vertices and adding successive edges between them at random. The aim of the study in this field is to determine at what stage a particular property of the graph is likely to arise. Different random graph models produce different [probability distributions](https://en.wikipedia.org/wiki/Probability_distribution) on graphs [4].

## Exponential Random Graph Model

Based on this last one, the exponential random graph model (ERGM) has been extensively used to model static networks and also used as a base for two of the first algorithms proposed to study dynamic networks, the temporal exponential random graph model (tERGM) and the hidden tERGM for modelling a sequence of node attribute [7]. When tERGM assume that the sequence of networks is available, htERGM explores the possible dependencies of unobserved rewiring networks and leads to the algorithm that can reconstruct such networks from a snapshots’ sequence of nodal attributes.

Although this algorithms overcame approaches that recover single time-invariant networks they are not the most useful ones since they depend on unobserved network variables being unable to compute likelihood ratios, needing inference algorithms specifically for each problem.

## Hidden Markov Dynamic Bayesian Network

Another classical approach are the Bayesian networks, widely used to describe biological systems, risk analysis and even financial networks.

Bayesian networks are graphs whose edges represent conditional dependencies and nodes are random variables in a Bayesian sense (observable quantities, latent variables, unknown parameters or hypotheses) and are associated with probability functions that takes as input the node’s parent variables and gives the probability of the variable represented by the node.

Its standard assumption is ‘stationarity’, and therefore, several research efforts have been recently proposed to relax this restriction. However, those methods suffer from three challenges: long running time, low accuracy and reliance on parameter settings. [11] propose a non-stationary DBN model by extending each hidden node of Hidden Markov Model into a DBN (called HMDBN), which properly handles the underlying time-evolving networks resulting in a promising experimental evaluation of the method, demonstrating more stably high prediction accuracy and significantly improved computation efficiency (even with no prior knowledge and parameter settings) on both synthetic and real biological data.

Although this probabilistic model is more complex than TESLA algorithm it could be a good method to implement in order to compare results and performance.

## Bayesian non-parametric model

In this line there was proposed a Bayesian non-parametric model including time-varying predictors in dynamic network inference precisely for financial studies [6].

This model computes edge specific predictors where the link probabilities () are estimated via a logistic regression, with a baseline process quantifying the overall propensity to form links in the network across time, are vectors containing the latent coordinates favoring a higher link probability when units *i* and *j* have latent coordinates in the same direction and is a P-dimensional vector of time-varying edge-specific predictors for units *i* and *j* at time t and are the corresponding dynamic coefficients. This allows the proximity between units *i* and *j* at time *t* to depend on predictors in a manner that varies smoothly with time.

The main issue regarding this method is that it assumes time-constant smoothness while in finance and other network applications we expect smoothness to vary over time, however it had some promising results when analyzing the global stock market network in the 2008 crisis and this makes it a good candidate to test in this project.

## Temporally smoothed l1-regularized logistic regression

TESLA represents an extension of the lasso-style sparse structure recovery technique and is based on a key assumption that temporally adjacent networks are likely not to be dramatically different from each other in topology and therefore are more likely to share common edges than temporally distant networks.

Building on the *l*1-regularized logistic regression algorithm for estimating single sparse networks [10] it was developed a regression regularization scheme that connects multiple time-specific network inference functions via a first-order edge smoothness function that encourages edge retention between time-adjacent networks. An important property of this idea is that it fully integrates all available samples of the entire time series in a single inference procedure, what means an advantage in contrast of htERGM algorithm.

TESLA estimates , the correlation (or dependency strength) matrix between the nodes using a time-series of the observed stated of the nodes **xt**.

where,

Where the graph structure is given by the locations of the nonzero elements of the parameter vectors . Components of the vectors are indexed by distinct pairs of nodes and a component *j* of the vector is nonzero if and only if the corresponding edge (*i, j*) ∈ Eτ. denotes the observed states of all nodes but node *i* and is the log conditional likelihood of state under a logistic regression model.

First term of the algorithm are p logistic regressions, one for each node with respect the rest of them ( are the states of all nodes but node I in the dth sample in time epoch t).

The second term is a lasso regularization that introduces sparsity and consistency in neighborhood selection, as showed in [10], converging to the true graph structure. Also a sparse graph effectively limits the degree of freedom of the model, which makes structure recovery possible given a small sample size [8].

Since in this problem we are estimating dynamic networks (so more than one single graph), and, as assumed before, temporary adjacent networks are going to be similar one to each other, we need to introduce the third term in the equation that penalizes the discrepancy between time-adjacent parameters .

Summarizing, the first term is the main one while the last two terms are the penalties () that are introduced to enforce sparsity and smoothness.

## TESLA updated

It is important to note that in this specific method we are assuming that are a piecewise constant function with abrupt changes in parameters (jumps).

However in some cases the changes in parameters are continuous, so further work [8] was carried out to adapt this algorithm including a weighting term that turns in continuous functions instead of the penalty in the structural changes () that assumes small “jumps” between time-adjacent values.

Where the weights are defined by

And is a symmetric nonnegative kernel function.

# financial application

When studying dynamic networks we aim to solve real world problems, so it is usual to look at meta-networks, this is multi-link (many types of links), multi-mode (many types of nodes) and multi-level (nodes can represent subnetworks) networks.

This added to the fact that not many work has been done to recover dynamic networks on finance and the promising algorithms reviewed on this paper sum up enough motivations to build an interesting topic: testing the performance of these different methods in the financial area to find the model that best defines it and extending the previous work done in this subject (analyze the behavior of financial networks against critic events [6]) by looking at meta-networks instead of simple networks.

It makes sense to model the financial market as a dynamic network: Network since its elements are linked and their behaviours can affect the whole system and dynamic to model the highly changing character of the stock market, that difficulties the ability to model it as a static system. In addition modelling a system as an evolving network gives the potential to study patterns or events’ effects that affect the network.

Network structure

When analysing a network we have to focus on different levels, going from the overall network as whole (in order to understand its distribution and measure the cohesion of the network), and later focusing on the individuals by measuring their centralities (to identify the central nodes of the system that support the network and study movements and interactions).

To study the **topology** of the network first we need to find a suitable layout to build the network. Many options are available, after testing and studying the designed layouts existing in R there were chosen some of them in order to try different approaches:

Frutcherman – Reingold layout: based on a directed force algorithm which takes the nodes as charged particles that tend to repel while the edges are treated as springs with an [attraction force or constant???] equals to their weights (TESLA output). The algorithm iterates until it converges to a minimum energy state (equilibrium). Force directed algorithms seem an ideal choice in this case since heavily connected individuals will tend to group while keeping unrelated nodes separated what is a very intuitive way to understand their interactions. The only counterpart of this family of algorithms is that the minimum energy state which it converges is a local minimum, this means that the results don’t need to be the best fit. This equilibrium state will depend on the initial coordinates assigned to the nodes before starting to iterate. A combined application of different algorithms is helpful to solve this problem, for example, using the Kamada–Kawai algorithm (also a directed force algorithm) to quickly generate a reasonable initial layout and then the Fruchterman–Reingold algorithm to improve the placement of neighbouring nodes.

Multidimensional scaling layout: This performs the classic MDS algorithm that uses the chosen distance metric to compute nodes similarities and then plot them in a 2D space keeping these distances and showing the edges. This way one can directly identify similar stocks and compare with the results given by the force directed algorithms. In addition it has the potential to be a useful layout to study flows and contagion effects.

Random layout fixed over time: Keeping the nodes fixed over time is the best way to focus on the evolution of the relationships in the system. Also this one seems to be a good candidate to study flows and compare with MDS.

TODO: layout clustered by sectors

After defining its topology one should study different general aspects of the graph to start characterizing the networks

- Shortest path: In [graph theory](https://en.wikipedia.org/wiki/Graph_theory), the shortest path problem is the problem of finding a [path](https://en.wikipedia.org/wiki/Path_(graph_theory)) between two [vertices](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) (or nodes) in a [graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)) such that the sum of the [weights](https://en.wikipedia.org/wiki/Glossary_of_graph_theory#Weighted_graphs_and_networks) of its constituent edges is minimized. Several algorithms have been introduced to solve this problem, considering that some of the weights of the edges may be negative the suitable one of this case is the [Bellman–Ford algorithm](https://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm) \*[ref]\*

- Connectivity: Is one of the basic concepts of [graph theory](https://en.wikipedia.org/wiki/Graph_theory): it searches for the minimum number of elements (nodes or links) that should be removed to disconnect the remai**n**ing nodes from each other. It is closely related to the theory of [network flow](https://en.wikipedia.org/wiki/Flow_network) problems, one can directly see the more connected a graph is the more information will flow through it. Also it can be used as an important measure of its robustness as a network. The definition of connectivity is quite intuitive, an [undirected graph](https://en.wikipedia.org/wiki/Undirected_graph) G, two vertices are called connected if G contains a [path](https://en.wikipedia.org/wiki/Path_(graph_theory)) from u to v. Otherwise, they are called disconnected.

A [directed graph](https://en.wikipedia.org/wiki/Directed_graph) is called weakly connected if replacing all of its directed edges with undirected edges produces a connected (undirected) graph. It is connected if it contains a directed path from x to y or a directed path from y to x for every pair of vertices x, y. It is strongly connected or strong if it contains a directed path from u to v and a directed path from *v* to *u* for every pair of vertices *u, v*.

The node connectivity of a graph is less than or equal to its edge connectivity. Both are less than or equal to the minimum Total Degree of the graph, (deleting all neighbours of a vertex of minimum degree will disconnect that vertex from the graph)

- Net Robustness: the ability to withstand failures and [perturbations](https://en.wiktionary.org/wiki/perturbation) is a critical attribute of many [complex systems](https://en.wikipedia.org/wiki/Complex_system). In [economics](https://en.wikipedia.org/wiki/Economics), network robustness principles can help our understanding of the stability and risks of banking systems \*[ref]\* But in our specific case we can use the engineering approach that uses it as a metric to evaluate the resilience of the infrastructure network that represent the stock market. \*[ref]\*

The focus of robustness in complex networks is the response of the network to the [removal of nodes](https://en.wikipedia.org/wiki/Node_deletion) or links. The mathematical model of such a process can be thought of as an inverse percolation process. [Percolation theory](https://en.wikipedia.org/wiki/Percolation_theory) models the process of randomly placing pebbles on an n-dimensional lattice with probability p, and predicts the sudden formation of a single large cluster at a critical probability \*[ref]\*

Measures of centrality

Information about the relative importance of nodes and edges in a graph is obtained through [centrality](https://en.wikipedia.org/wiki/Centrality) measures. This importance of a node can have different interpretations leading to different definitions of centrality. The two main interpretations that have been proposed: importance as related to a kind of flow across the network \*[ref]\* or as involved in the cohesiveness of the network. \*[ref]\* [ Borgatti, Stephen P. (2005). "Centrality and Network Flow".Social Networks. Elsevier.**27**: 55–71.[*doi*](https://en.wikipedia.org/wiki/Digital_object_identifier):[*10.1016/j.socnet.2004.11.008*](https://dx.doi.org/10.1016%2Fj.socnet.2004.11.008).]

This question is addressed by centrality measures. Applications include identifying the most influential people in a [social network](https://en.wikipedia.org/wiki/Social_network), key infrastructure nodes in the [Internet](https://en.wikipedia.org/wiki/Internet) or [urban networks](https://en.wikipedia.org/wiki/Urban_network), and [super-spreaders](https://en.wikipedia.org/wiki/Super-spreader) of disease. Centrality concepts were first developed in [social network analysis](https://en.wikipedia.org/wiki/Social_network_analysis), and many of the terms used to measure centrality reflect their [sociological](https://en.wikipedia.org/wiki/Sociology) origin \*[ref]\*, however this concepts (popularity, gregariousness individuals) give a very intuitive description of the stock market as a complex system. This measures also helps to understand the topology/behaviour and dynamics of the studied system (central nodes provide the structure of the market maintaining it joined) and make possible to identify disease spreaders.

* Degree centrality: Counts the number of walks of length one, so the number of first degree connections a node has. The degree can be interpreted in terms of the immediate risk of a node for catching whatever is flowing through the network (flow studies) or cohesiveness measure.

One important characteristic of the recovered networks is that they are directed given that the adjacency matrices computed by the TESLA algorithm returns the relation of each one of the nodes (independently) with the rest on a single inference procedure. This results on not (necessary) reciprocal interactions. In this case it is necessary then to study out (connections of the computed node with the rest) and in (connections of the other computed nodes relating this one) degrees of the nodes. Following a classic approach on Social Networks studies, when ties are associated to some positive aspects such as friendship or collaboration, Out Degree is interpreted as gregariousness (number of computed connections that emerge from the studied node when applying TESLA to find its relation with the rest) and In Degree popularity (hubs).

* + - Taking this into account In Degree of the nodes will be used as a measure to identify relevant stocks (hubs) which influence the rest. Also we can use this directed links to study crash or recover spreads from hubs.
    - The definition of centrality on the node level can be extended to the whole graph, in which case we are speaking of graph centralization [FORMULA]
* Eigenvector Centrality: (also called eigencentrality) is a measure of the influence of a [node](https://en.wikipedia.org/wiki/Node_(networking)) in a [network](https://en.wikipedia.org/wiki/Network_(mathematics)). It assigns relative scores to all nodes in the network based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. [FORMULA]
* Closeness Centrality: In connected [graphs](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)) there is a natural distance metric between all pairs of nodes, defined by the length of their [shortest paths](https://en.wikipedia.org/wiki/Shortest_path_problem). The farness of a node x is defined as the sum of its distances from all other nodes, y, and its closeness was defined by \*[ref: Bavelas]\* as the [reciprocal](https://en.wikipedia.org/wiki/Multiplicative_inverse) of the farness.

Thus, the more central a node is the lower its total distance from all other nodes. Note that taking distances from or to all other nodes is irrelevant in undirected graphs, whereas in directed graphs distances to a node are considered a more meaningful measure of centrality, as in general a node has little control over its incoming links.

When a graph is not [strongly connected](https://en.wikipedia.org/wiki/Strongly_connected_component) (what is our case), a widespread idea is that of using the sum of reciprocal of distances, instead of the reciprocal of the sum of distances, with the convention {\displaystyle 1/\infty =0}1/inf=0:

* Betweeness Centrality: quantifies the number of times a node acts as a bridge along the shortest path between two other nodes. It was created as a measure to quantify the control of a human on the communication between other humans (stocks) in a social network \*[ref: Freeman]\* In his conception, nodes that have a high probability to occur on a random [shortest path](https://en.wikipedia.org/wiki/Shortest_path_problem) between two randomly chosen vertices have higher betweenness:

\*[ref: Brandes]\*

where {\displaystyle \sigma \_{st}} is total number of shortest paths from node s to node t and is the number of those paths that pass through v{\displaystyle v}. The betweenness may be normalised by dividing through the number of pairs of vertices not including v, which for [directed graphs](https://en.wikipedia.org/wiki/Digraph_(mathematics)) is (n-1)(n-2).

Dynamic measures

Since in this case we want to model a dynamic network, apart from analysing the evolution of previous parameters over time to try to find some correlation with the stock market behaviour, there are more recent metrics introduced precisely to study similar problems:

* Crash and contagion study: Gain and Loss studies

Signal-flow graph is based on the concept that the state of a node can be modelled by the previous state of the other vertices that point to it. Given this a node which In Degree is 0 is considered as an independent variable. We write a function for each node that processes the signals that it receives.

One variable I find interesting to add to the models is the previous state of the node itself, given that in this case we are studying dynamic networks it is consequent to consider the state of a node dependant on the previous state (in the signal-flow graphs loops are considered, not here, in these cases the same state of the node we are modelling is added to the function).

\*[ref]\*: Dorf, Richard C.; Bishop,, Robert H. (2001). "Chap 2.-1: Introduction". [*Modern Control Systems*](https://www.site.uottawa.ca/~rhabash/ELG4152LN02.pdf) (PDF). Prentice Hall. p. 2. [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [0-13-030660-6](https://en.wikipedia.org/wiki/Special:BookSources/0-13-030660-6)???

* Distribution of hubs over time, trying to model their behaviour given different parameters (comments later on the analysis part)

Software used:

Matlab: Used to compute TESLA algorithm, which is built and ready to retrieve online. This script has been modified according to our needs, simplifying it, what makes it easier a further implementation of the whole ‘preprocessing + computation + analysis’ pipeline in R to generate a function that automatically recovers dynamic networks and analyses them (TESLA algorithm just finds the dependency strengths).

R: Used in the exploration/preprocessing stage and graph plotting and analysis thanks (mainly) to the packages *igraph, network, sna and ggplot2*.

Data retrieving and preprocessing

As said before, TESLA estimates the dependency strength between nodes using time-series of the observed states of the nodes, hence the data used in this work correspond to time series of the stock values for the most important companies forming the British index: FTSE100.

The FTSE100 consists of the largest 100 UK companies by full market value, given that the problem here is that the companies forming the FTSE100 change quarterly resulting in not homogenous data for large enough intervals, to solve this potential problem, only data from the stocks taking part during all the time of the interval studied are retrieved (when data available during the whole interval). In a similar way, we noticed a couple of cases in which not all stocks had valuation probably corresponding to a vacation, this is solved by performing inner join (using only observations with values for all stocks).

First, and to get started data from 2009 and 2010 was taken from 85 assets that took part on the FTSE100 index on the mentioned years.

In order to study and analyse the impact of critical events, data from January of 2015 until the end of July 2016 was gathered in order to study the effects of the Brexit on the Stock Market. In this case instead of focusing on the FTSE100 data from companies which Market Capitalisation is over 1B. This seems to be more representative of the British Economy since companies from the FTSE 100 are usually international companies, but smaller companies (like the ones that form the FTSE 250) usually are national companies. After some cleaning of the data to remove stocks which values were unreliable (missing values, many consecutive days no change) there remained 186 companies.

Gathering all these data hasn’t been trivial, since the ideal is gathering data by Market Capitalisation, (so we get the strongest companies, considered to be the most influential in the Market given some threeshold), the best way to automatize this process is to connect yahoo finance through R and iterate over the tickers of all the stocks in the British market. First, tickers from all stock markets were retrieved and then filtered by country. This way we have the chance to study (if enough time) the behaviour of countries related to UK (Germany, France, USA…).

After getting the data and cleaning it some preprocessing needs to be done in order to extract variables with analysis potential. Also, and given the assumption that stocks with similar trend are likely to be influenced by the same underlying effect, it is needed to get the daily returns. In addition to this, just using the stock values it is very unlikely to find links between companies with extremely different stock values, even if their trends are similar.

TESLA algorithm

Once the data is prepared, TESLA will recover the structure of the network given chosen sparsity and smoothness parameters (tests were done, just need to make a plot with degree/weights for different sp and sm to assess its performance)

First attempts it didn’t work at all, after simplifying some useless (for this problem) features it started working.

After exploring and plotting first results, I found out that the parameters that related the nodes with themselves (*i,i*) corresponded to extremely high values (in this very case we are not interested in loops on the same node and it obscures the possible existing interactions with other stocks, since sparsity has been enforced in TESLA). However given the problem formulation on the TESLA paper [ref] we expect these values to be 0. In order to fulfil this condition some further modification of the algorithm was necessary.

**EXPERIMENTAL RESULTS**

The analysis performed to these networks focus on studying how the topology of the system changes on different situations, finding and studying the effects of the most important agents, understanding how the relations between these stocks change over time and finally its flow and its capacity of spread or recover from a contagion.

General topology and behaviour

When plotting the network using the Fruchterman – Reingold algorithm the stocks follow a Star Layout, in which there are always a few hubs (that vary over time) with a high In Degree (degree encoded in the size of the ndoes) \*[figures]\* (almost maximum in most cases) that are usually located in the core and are supposed to lead the market behaviour, while the majority of the stocks have a minimum value, showing them as weak stocks whose behaviour is supposed to follow the Market’s core behaviour. An advantage of this star topology is the simplicity of adding additional nodes as well as its reliability since because if one node or its connection breaks it doesn’t affect the other nodes and their connections. On the other hand the hub represents a single point of failure, the less hubs the system have the less robust the system is \*[ref]\*

On the other hand, the Out Degree is usually similar for all the stocks. This is expected, since the Out Weights are TESLA results: it computes each stock *x* on one individual optimization problem that recover the interactions that *x* has simultaneously with the rest of the stocks, and enforcing the same sparsity on all optimization problems. This makes one expect homogeneous results for all stocks. Being that homogeneous makes it an outstanding candidate to use it as a cohesion measure for the network \*[figure]\*. In the plot we can observe how the average Out Degree (or the Out Degree of the Network) changes over time, showing how the connectivity of the network decreases and increases in different periods. Given this measure one thing we could try is to find if it is related with some market variables as the value of the British index, volatility, or other possible macroeconomic variables.

After this analysis of In and Out Degree we can conclude that this is a small – world Network \*[ref]\*, type of [graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)) in which most nodes are not neighbours of one another, but most nodes can be reached from every other node by a small number of hops or steps.

Flow study

To study how the network faces critical events and reacts when some loss (or gain) is spread over the network a time evolving Flow Network has been built. Here the green nodes represent stocks with average positive return over that period while red ones has negative average return. The different sizes of nodes show the (scaled) Total Degree of the nodes, making easy to recognize high risk elements. In these graphs is possible to observe how different areas of the network activates over time as the negative or positive trend expands. This way the colour of the links correspond to the trend of the source node.

\*\*\*To study the possibility of given companies to be potential spreaders, we can compute for each stock the proportion of out – linked nodes that have the same state than this source had in the previous time stamp. DISCUSS POSSIBLE APPROACH: weighting more individuals that change, not so much the ones that have the same state than the source but keeping it from the previous state. TAKE INTO ACCOUNT THE WEIGHTS (INTERACTIONS) SIGNS???

\*\*\*\*Or the probability of a node to change state given the In – links (trend and weight, and maybe also weight of the sources: Market Capitalisation?)

sum(weightsIn(j)\*GainLoss(j)) (normalize??) j represent every node but the one we are studying

(compare with historical context each of the cases: specific hubs and so)

If we could model and predict accurately enough (error, boxplots…) the gain or loss of a stock given the incoming weights of its connected nodes this could be extremely helpful for companies to prevent situations and react before to minimise the loss or maximise the gains. \*\*\*try different approaches: maybe with the averaged weight, or taking into account other factors as the previous Gain/Loss value of the node i, or its value on the stock market, or weighting the weights by the companies’ Market Capitalisation (MC) of the source nodes: sum(weightsIn(j)\*GainLoss(j)\*MC(j))/sum(MC(j)) And maybe multiplying or using MC(i).

Node’s centrality Measures

Compare degree, eigencentrality, closeness and betweenness

One analysis we can do is study if there is any relationship between the evolution of networks’ hubs (or predict prob of turning hub given its previous weights in(norm?), sector, values, gain/loss: Decision trees?: Input=previous net info dataframe(nodes values, edges values, sectors, gain/loss, company values, avg(stockValue)…). Output=hub? Gain/loss?-> mejor, es un valor directo del stock market, los hubs son calculado por TESLA

Intra and inter sector connectivity

In order to study how the importance of different sectors change over time sector degrees and weights were computed.

Also to know how much a sector is isolated or interacts with the market we should study its intra (between stocks within same sector) and interconnectivity with other sectors by measuring the proportion of connections a given sector have between its stocks and with different sectors.

The bigger the intraconnectivity compared to the interconnectivity the more isolated a given sector will be while the bigger the interconnectivity the more active it is, leading the Market.

**CONCLUSIONS**

* + Comparison with other works
  + Further work (next month, next 10 years)
  + Conclusions

also comparing with the results given using the correlation matrix values instead of the TESLA interaction matrix values: same tasks, measure of error from both of them and, modelling trends in the market, use it as performance comparison??