UiA University of Agder

Master

Exploration of stationary time-series' network properties

Visibility- and Horizontal visibility graph of Gaussian white noise, $AR\ (1)$, $MA\ (1)$ and $ARMA\ (1,1)$

FARNOOSH FARHANGIAN

GRY MERETHE NERJORDET

SUPERVISOR

Jochen Jungeilges

University of Agder, 2019

School of Business and Law

Department of Economics and Finance

PREFACE

This thesis is the culmination of a five-year long economics study at the university in Agder. In our studies, both of us have always opted for the more number-oriented electives. The choice of financial economics as our specialization in the master program was therefore a natural one. We are however not typical student straight out of high school. Farnoosh has a bachelor's degree in physics and Gry has a bachelor's degree in both graphics and game design.

Visibility graphs is therefore a topic that suited us both, but in very different ways. Graph and network theory are both closely interlinked with physics and the ability to visually explore graphs enhance the understanding of its properties. This has resulted in a thesis with a different approach and appearance compared to other studies in this subject. We have often opted to use visual representations as an alternative to tables.
Another motivating factor was the fact that this topic was completely new. We both enjoy the accumulation of knowledge and appreciate a challenge. Looking back to January, it is almost unbelievable that we did not know the basics of the topic in which we now have written a master thesis.

We express our gratitude to our supervisor, Jochen Jungeilges, for challenging us with this topic, believing that we were up to this task, and having the nerve of letting the thesis evolve naturally as our understanding of the topic grew. We would not have been able to arrive at our final results without his input and guidance. In addition, he supplied us with the R-script which generate a time-series' adjacency matrix based on the horizontal visibility algorithm. We extended this script to include the visibility graph algorithm as well, and it was essential script in our work.

We also wish to express our gratitude towards our families who, in the last five months, have been both understanding and forgiving when we embarked on, and finished, this rather all-consuming task.

ABSTRACT

We have recorded the network properties of stochastic processes' white noise, $MA(1)$, $AR(1)$ and $ARMA(1,1)$ by means of visibility- and horizontal visibility algorithms. The sample series of the stochastic processes were artificially generated and used in simulations with multiple repetitions. We examined six global network properties: mean degree, normalized degree centrality, degree distribution, transitivity, assortativity and average geodesic path to discover how they reflect the structure of the series. These properties were also used to disclose change in behavior due to variation in the series length. Our contribution to the literature is a thorough recording of basic stochastic processes' network statistics, and the interpretation of these related to time-series. To our knowledge, this is the first time a documentation of this magnitude has been attempted. The thesis result will disclose that the combined network statistics mentioned above are able to recognize a Gaussian white noise process, both by visibility- and horizontal visibility graphs. It is, in addition, possible to distinguish between the processes different autocorrelation coefficients by means of the visibility graph.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1 INTRODUCTION

The accumulation of data gathered in the form of time-series have grown exponentially in the latest decade. With internet of things and smart speakers listening in on conversations in almost every households, the amount of data collected appear limitless. But the value of this data does not lie in its sheer volume, it's in what it can communicate about behaviors, trends, relationships and dependencies. The traditional and often preferred method of studying time-series is by the means of time-series analysis. Time-series analysis maps the change of a variable over time and determines the relationship between variables.

Analyzing real-life time-series, however, may be problematic with this kind of analysis. They often violate stationarity, which is an important assumption in time-series analysis. A series is stationary when it has a constant variance and mean over time, a trait that seldom appear naturally. The problem with stationarity is an old one, and it has been proven that using time-series analysis on non-stationary series can give spurious results. The common solution to the reoccurring stationarity problem is to transform the data until the series is stationary and then perform the analysis. The issue with these transformations is that the result from the analysis are of the transformed data, not the data which we were intending to study. The task of relating the results of the analysis to the actual process under scrutiny can be challenging.

In recent years the idea of converting time-series to networks, bypassing the whole problem with stationarity altogether, has bloomed. This is achieved by mapping a time-series into a graph by an algorithm. Before we move further on, we need to clarify the use of the word graph. This is often used as a common descriptor for any graphical representation of a dataset. In this context though, we are discussing the mathematical graph. In his introductory book on graph theory Wilson (2010, p. 8) describe a graph in exactly the same way as Newman (2010, p. 10) describes a network in his introduction to networks. A network (and a graph) is a visualization of relationships through dots (referred to as nodes) connected by lines (called edges). Thus, the term graph and network are interchangeable in this text.

This mapping of time-series will also result in some information loss, but it will keep most of the time-series' original properties. The loss will therefore be less than the one faced when transforming to achieve stationarity. The theory of complex networks is well-developed and have the potential to provides as much information as a time-series analysis. One of the main focus in this active field of science is the expectation that structural properties of the underlying data generating process allows for discrimination between different types of processes.

When we were familiarizing ourselves with this topic, we were surprised by the lack of interest in the basic network properties of the stochastic processes. Most articles focused on one measurement, typically the cumulative degree distribution, and tried to find characteristics which separates stochastic processes from chaos. And we were wondering: how are they supposed to find differences with no clear idea about the baseline? Namely the network behavior of the stochastic processes. We chose to start at the other end of the chase of finding a method to identity chaos, by investigating and documenting the network behavior of stochastic processes.

Our intention with this exploratory exercise is to generate some benchmarks from which it would eventually be possible to differentiate between processes.

There are multiple methods which can transform a time-series into a network. We chose to use the visibility algorithm and the horizontal visibility algorithm. The conversion involves the use of the visibility criterion which determine the connections based on the time-series different realizations. The horizontal visibility graph is a subgraph of the visibility graph and has a stricter visibility criterion. The mapping itself is achieved by an algorithm which generates an adjacency matrix. This matrix records the connection, edges, between each pair of nodes. The number of edges connected to a node is denoted as the node's degree.

The visibility- and horizontal visibility graphs are linked directly to the time domain and this makes it possible to identify which time-series properties the different network statistics are related to. These algorithms also result in fast conversion which is a factor when dealing with larger sample sizes.

As graph theory offers a vast number of measurements, we chose to narrow this down to the global measurements with less of a social network orientation. The chosen statistics were mean degree, normalized degree centrality, the properties of degree distribution, transitivity, assortativity and averages shortest paths length. All statistics where calculated for both visibility graph and its subgraph.

Our focus in this study is different from what has been done previously as we decided to study four different types of stochastic processes. The stochastic processes used are Gaussian white noise, autoregressive (AR), moving average (MA) and a combination of both autoregressive and moving average, an $ARMA$, processes, the last three are of order one.

To narrow the scope of our work we chose two research questions. Firstly, to explore how our elected global network statistics behave with different sample lengths, if the network is generated from a Gaussian white noise process. By recording these behaviors, we hope to generate a baseline from which, it would be possible to recognize a white noise process, independent of the time-series' stationarity. Secondly, to identify how a change in autocorrelation coefficients of $AR(1)$, $MA(1)$ and $ARMA(1,1)$ represent itself in the corresponding networks properties. This will be done with white noise as a benchmark using results and the insights from the first analysis to enhance the latter. Our purpose of this study is to be able to distinguish between these three processes by means of their network statistics, again independent of stationarity.

All time-series in this thesis are artificially generated in the free software R, and no real-life time-series was used. Our research strategy involves numerical experiments. A basic experimental run takes the following form: a sample series from a known $ARMA$ type data generating processes is represented as a network. Such a representation is achieved via a transformation of the respective sample series. The resulting network structure is then captured and summarized the selected network statistics. Thereby an element in the $ARMA$ family (a specific stochastic process) is associated with a point in the space of network characteristics. By replicating such a

basic run many times, we hope to identify the network properties of the underlying stochastic process in a reliable fashion. For this approach to be viable, the algorithm facilitating the transformation from the sample series into a network has to be both intuitive and fast. We also created a reference set consisting of typical realizations from the different processes. These were constructed such that each series had the same error term and network layout, thus the only difference was the data generating process'. This made it possible to visually compare the graphs and was used to enhance our understanding of the pattern revealed in the multiple repetition simulations. By fixing the graphs layout we lose the ability to visually recognize communities, especially with large sample sizes. Communities are groups of well-connected nodes which are linked to external nodes by a few of its interconnected nodes. This was a sacrifice we were willing to make as we have minimal use of communities in our thesis.

This study will describe and interpret the network behaviors of selected stochastic processes' in an extensive matter. The behavior can be used to identify stochastic processes, permitting the development of tests and tools. Such a comprehensive analysis has not, to our knowledge, been attempted before and may be the starting point to a different approach toward distinguishing amongst different types of data generating processes.

The structure of the sections is organized as follows: the next section is a summary of the evolution and use of networks with focus on time-series based networks. The development of visibility- and horizontal visibility graphs will be emphasized. Next in section three, we will present the stochastic processes studied, followed by a description of our simulation strategy with selected results. Section four describes the mapping of time series as visibility- and horizontal visibility graphs. It also presents the network statistics with examples. Next, in section five we finally present and interpret our findings both individual and compared to each other. The final sections include a discussion, section six, followed by a conclusion in section seven.

2 EVOLUTION AND APPLICATION OF TIME-SERIE-BASED NETWORKS

The idea of visually presenting relationships through the means of a graph is not new. It was used in games in ancient Egypt, to visualized family trees in the middle ages and as a categorization tool in medieval literature. Even though this visualization method had been readily available for thousands of years, the father of graph theory (the study of connection between things), Leonard Euler, did not actually use this in his 1736 papers. It did not take long however before his colleagues, like Vandermonde, Listing and Hamilton, used graph drawing in combination with graph theory with great success (Kruja, Marks, Blair, & Waters, 2002).

Graph theory is now an established branch of mathematics. But the idea of being able to characterize a time-series' underlying process by the transformation to a complex network, however, is new. It has only been around for about a decade. The possibility of using graph theory to analyze data with problematic properties regarding to regular time-series analysis, makes this notion widely attractive. It is still considered very much as an active field of science where most of the efforts are focused on the use of complex networks to distinguish between a stochastic process and a chaotic system.

As mentioned above, we try to understand the network properties associated with linear stochastic processes (with white noise as a special case). The visibility graph and its subgraph are our preferred methods of converting time-series to networks. The cleverly named visibility algorithm generates a graph where the nodes are the value of each of the time-series data points (realizations). The connections (edges) depend on the angle of a line between each node as well as the value of nodes in between. If the time-series is plotted as a bar plot, the connection criterion states simply that if the columns can see each other, they are considered to be linked. This algorithm converts a time-series directly from the time domain resulting into a network that inherits the time-series structure and is unaffected by rescaling and transformations (Lacasa, Luque, Ballesteros, Luque, & Nuño, 2008). Another property of the visibility graph discovered by Lacasa, Luque, Luque, and Nuño (2009) was the dependency between the degree distribution and the Hurst parameter. This is in many cases used to differentiate stochastic processes from chaos. They then proposed the possibility of estimating the Hurst exponent by the means of visibility graphs and thereby be able to distinguish chaos from noise. Li et al. (2016) extended this theory to short time-series by concluding that the Hurst exponent can be estimated by combining the visibility graph with maximum likelihood equation and the KS statistics.

The horizontal visibility algorithm has stricter demands for when the nodes are connected. Thus, creates a simple and analytically solvable subgraph of the visibility algorithm. Networks generated by the horizontal visibility algorithm are also unaffected by rescaling and transformations. Likewise, it is showed that, independent of the probability distribution of the time-series, the degree distribution will have the same exponential functional form (Luque, Lacasa, Ballesteros, & Luque, 2009). There have also been suggested, as with the visibility graph, that the exponential degree distribution frontier can be used to distinguish between correlation stochastic, uncorrelated stochastic and chaotic processes. This is done by considering the exponential degree distribution as $P(k) \sim e^{-\lambda k}$, where k is corresponding to the node degree and λ is a positive parameter used for characterizing the process (Lacasa & Toral, 2010; Luque et al., 2009). Ravetti,

Carpi, Gonçalves, Frery, and Rosso (2014) discovered that for a white noise process, λ is computed to be equal to $\ln(\frac{3}{2})$, which will be more for correlated stochastic processes and less for chaotic series. But they also discouraged the use of the exponential degree distribution as a general rule for distinguish between processes, because they identified several cases where this hypothesis did not hold. The use of this scaling factor as a method to distinguish stochastic from chaotic dynamics where further advised against both in visibility and horizontal visibility graphs by Zhang, Zou, Zhou, Gao, and Guan (2017). They stated that this could not be treated as a general law for separating these dynamics, so the hunt for such a law, and the discovery of these graphs properties continues.

We mentioned earlier that there is a lack of publications which documented the network statistics of different stochastic processes. There are however a few who have embarked on this task. When Luque et al. (2009) presented the horizontal visibility graph they also showed exact results for random time series in three different network statistics. Four network properties of autoregressive process of both first and second orders with time delay was studied by Zhang et al. (2017) . The generalized autoregressive conditional heteroscedasticity process, GARCH, was studied in a similar way by Segberg and Skoglund (2017) where they documented five different network statistics.

The method of using graphs to analyze time-series are still very much in development, but visibility and horizontal visibility have already been successfully applied in multiple studies. Zhuang, Small, and Feng (2014) discovered that the node degree reflected historical incidents which affected time-series from the developed financial markets. The series cycles were also linked to the graph's communities in which the density corresponded to the significance of the cycles. This was also the case in the analysis of natural gas price in North America. They discovered that large degree nodes which were linked with significant events and communities reflected the time-series' cycles (Sun, Wang, & Gao, 2016).

Yang, Qu, and Chang (2015) discovered that they could use visibility graphs to investigate the relationship among parties in financing. It was shown that this result matches the current situation and the default tendencies among the parties. They thereby proved that the network analysis will provide the same results as traditional analysis.

An interesting study linking the coal price index and coal mining accident successfully generated a warning for coal miners. This was based on economic indices by the use of a directed and weighted network (Huang et al., 2016). Yu (2013) used the visiblity graph to analyse the gold price time-series which revealed that the time-series was indeed a long-range dependent fractial series.

Even though this text focuses on the visibility- and the horizontal visibility graphs, there are a plethora of alternative transformation methods in the literature. The parametric natural visibility graph is another subgraph of the visibility graph where the additional parameter view angle is introduced, examining other dynamic properties than the original visibility graph (Bezsudnov & Snarskii, 2014).

Zhang and Small (2006) created a complex undirected cycle network from pseudo periodic time-series which were able to distinguish differences in the time-series' properties. The networks nodes corresponded to a cycle in the time-series and were connected with similar cycles. A phase space-based algorithm was used by Xu, Zhang, and Small (2008) to sort chaotic and random noise into different super families, thus being able to identify and differentiate these. Another methods which use a recurrence matrix instead of the traditional adjacency matrix were purposed by Marwan, Donges, Zou, Donner, and Kurths (2009). The recurrence matrix is generated from the recurrences in phase space and the resulting graph showed potential for detecting dynamic transitions. The authors also discovered that in an unweighted and undirected network the adjacency and the recurrence matrix coincide. Network created using the recurrence matrix also shows the relationship between the topological properties of the network and its underlying dynamic systems (Donner, Zou, Donges, Marwan, & Kurths, 2010). This method is not derived directly from the time domain though, it can be a challenge to interpret.

It has also been shown that the transformation of a time-series to a network can be reversed opening the possibility of not just using network theory on time-series, but also using time-series analysis on networks. Such a reversibility would only be possible if the transformed network inherits characteristics from the time-series it is generated from (Campanharo, Sirer, Malmgren, Ramos, & Amaral, 2011).

3 STOCHASTIC PROSESSES AND THEIR SIMULATIONS

We mentioned in the introduction that we narrowed our exploration of linear stochastic processes to, $AR(1)$, $MA(1)$ and $ARMA(1,1)$ with the addition of the special case a white noise. One might claim that these are the obvious start in the effort to determine the network properties of stochastic processes.

3.1 White noise process

White noise is a homoscedastic, stationary process of random variables with no visible structure and has the following properties:

$$E(y_t) = \mu \tag{1}$$

$$VAR(y_t) = \sigma^2 \tag{2}$$

$$\gamma_{t-r} = \begin{cases} \sigma^2 & if \quad t = r \\ 0 & if \quad t \neq r \end{cases} \tag{3}$$

Where γ denotes the autocovariance parameter. The process has a constant mean and variance and zero covariance between observations of different lags and is identified as an uncorrelated process. An uncorrelated process has no autocorrelation between different observations, and it makes the process unpredictable.

We will use the Gaussian white noise process in our thesis. It is generated from a Gaussian distributed random variable and creates an identically distributed zero white noise process. This is a normally distributed process with a zero mean and a density function defined as:

$$f(y) = \frac{1}{\sqrt{2\pi}}\, e^{-(y-\mu)^2 / 2\sigma^2} \tag{4}$$

A typical plot display of a realization from a Gaussian white noise process will show perfect fluctuation of observations around a constant mean of zero, as is shown in the figure below. This mean reversion property could be used to determine the stationarity of the process. Even though the observations fluctuate in the range from plus to minus three, the variance throughout the series is constant and equal to one. The correlogram in the figure is depicting the realizations' auto correlation parameter. It shows zero autocorrelation parameters for all the lags except the lag of zero as expected. Lag zero will always be one because it shows the autocorrelation parameter of a value with itself. The value from all other lags lies in the Bartlett's band which means their results are insignificant (Brooks, 2008, p. 209).

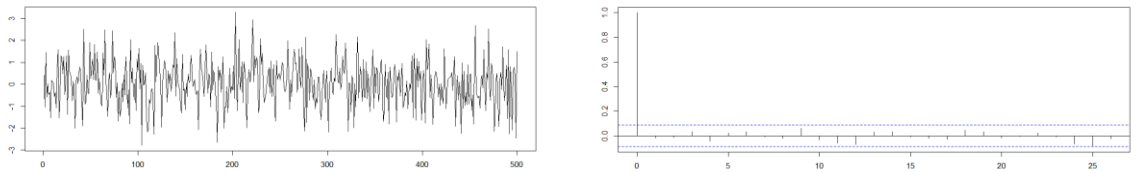


Figure 1: Typical realization of Gaussian white noise process

Line chart to the left, correlogram to the right of a typical realization with a sample size of 500
The blue line in the correlogram indicates the Bartlett's band.

3.2 Moving average process

A moving average, (MA) process can be defined as a linear combination of white noise processes and a process of order q, $MA(q)$, is written as:

$$y_t = \mu + \theta(L)u_t \tag{5}$$

which is expressing the dependence of the variable y_t on the current and previous values of a white noise term. The white noise terms are indicating the shocks in the process where:

$$\theta(L) = 1 + \theta_1 L - \theta_2 L^2 + \cdots + \theta_q L^q \tag{6}$$

The order of q is the number of lags of the white noise terms that impacts the value of the y_t.

A moving average process is a stationary process, where the autocovariance and autocorrelation for the lags larger than q are zero. The strengths of shocks effects are constructed by the autocorrelation function. An autocorrelation function is defined as $\tau_s = \frac{\gamma_s}{\gamma_0}$ where γ_s is the autocovariance at lag s, and γ_0 is interpreted as the autocovariance at lag zero (Brooks, 2008, pp. 211-214).

According to Verbeek (2004, p. 259) the length of the memory of the process could also be constructed by the autocorrelation function. The value of the autocorrelation for lags greater than q is zero, and this implies that there will only be significant effect from shocks of current and past q periods on the value of y_t. For instant, for an $MA(1)$ there will be only significance effect of shocks at time t and $t-1$. This is illustrated in the figure below where we have presented a realization of a $MA(1)$ process. The correlogram displays that the dependent variable does not only get effect from shocks of the current period, but also the shock of the previous period. This is consistent with the definition of an $MA(q)$ process.

We limit the exploration of this thesis to a moving average process of order one, $MA(1)$. When we manipulate the process' autocorrelation coefficient, we are adjusting the previous value of white noises' effect on the current value.

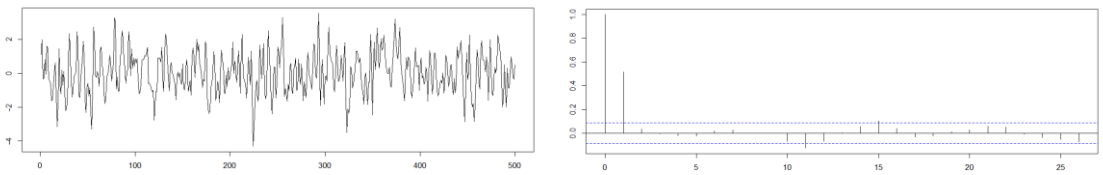


Figure 2: Typical realization of a moving average process of order one

Line chart to the left, correlogram to the right of a typical realization with a sample size of 500
The blue line in the correlogram indicates the Bartlett's band. The vertical line outside the bend at tick one indicates that the moving average process is at order one.

3.3 Autoregressive process

A stochastic autoregressive process, (AR) is defined as a process where the current value of the variable y depends on its' previous values and a stochastic error term, u_t. An autoregressive process of order p, $AR(p)$, is written as:

$$\varphi(L)y_t = \mu + u_t \quad (7)$$

Where,

$$\varphi(L) = 1 - \varphi_1 L - \varphi_2 L^2 - \cdots - \varphi_p L^p \quad (8)$$

and p is the number of the lags. The expected value of the y_t is expressed as the constant term μ, and u_t which is an unpredictable component expressing a white noise process.

An autoregressive process indicates that the current value of an observation is correlated with its previous values. The correlation between the current value of an observation and a given lag could be measured by the partial autocorrelation parameter as well as the autocorrelation parameter. The only difference is that the partial autocorrelation function is removing the effects from the intermediate lags while measuring the autocorrelation parameters, while the autocorrelation function is considering the intermediate effects too. The correlations between the current observation and it lags up to p are significantly non-zero, while for lags larger than p, there will be no significant correlations. This could be shown by the partial autocorrelation function, where there is non-zero partial autocorrelation parameter up to lag p, and zero thereafter. Unlike the partial autocorrelation function, the autocorrelation function of an autoregressive process will show an exponential decay as lags get larger, it will not cut off to zero for lags larger than p. A moving average process will show similar pattern in the partial autocorrelation function (Brooks, 2008, pp. 215-225). How fast the autocorrelation parameters of an $AR(p)$ decrease in an autoregressive process is dependent on the correlation parameter φ. A large φ means that, it takes longer for the series to get back to its mean after a shock is appearing (Verbeek, 2004, p. 260).

Stationarity is a desired property of an autoregressive process. According to Verbeek (2004, p. 258), a process is defined as a strictly stationary process, if any changes of the time origin do not affect the properties of the process. This implies that the distribution of the process stays unchanged. While a weakly stationary process, also called a covariance stationary process, is referred to a process where its mean, variance and covariance are not dependent on time. The world's decomposition theorem states that any stationary process where $|\varphi| < 1$, could be expressed as combination of a deterministic and a stochastic part. Based on this theorem, a stationary autoregressive process of order p with a mean of zero, could be represented as an infinite order moving average model.

The mentioned relationship between the correlation parameter of an $AR(p)$ and the decrease of its autocorrelation parameters is shown in the illustration below of simulated $AR(1)$ process', their autocorrelation functions and partial autocorrelation functions. The plotted data shows more mean reverting property for the correlation parameter of 0.3 (left) than for $\varphi = 0.8$ (right). On the other hand, the autocorrelation function of the parameter of $\varphi = 0.8$ decreases smoother. Which argues that a larger parameter leads to higher dependencies and makes the

process less mean reverting. This is because it takes a while for the autocorrelation function to converge zero. Partial autocorrelation parameters converge to zero for lags larger than p. This is what is shown in our example and as illustrated, no matter what parameter an $AR(1)$ process have, the partial autocorrelation parameters for lags larger than one, lie inside the Bartlett's band and are therefore not significant.

Our explorations are limited to a stationary autocorrelated process of order one, $AR(1)$. When we manipulate the process' autocorrelation coefficient we are adjusting the previous values effect on the current value. We keep the value of the autocorrelation coefficient in an interval between plus and minus one to ensure stationarity.

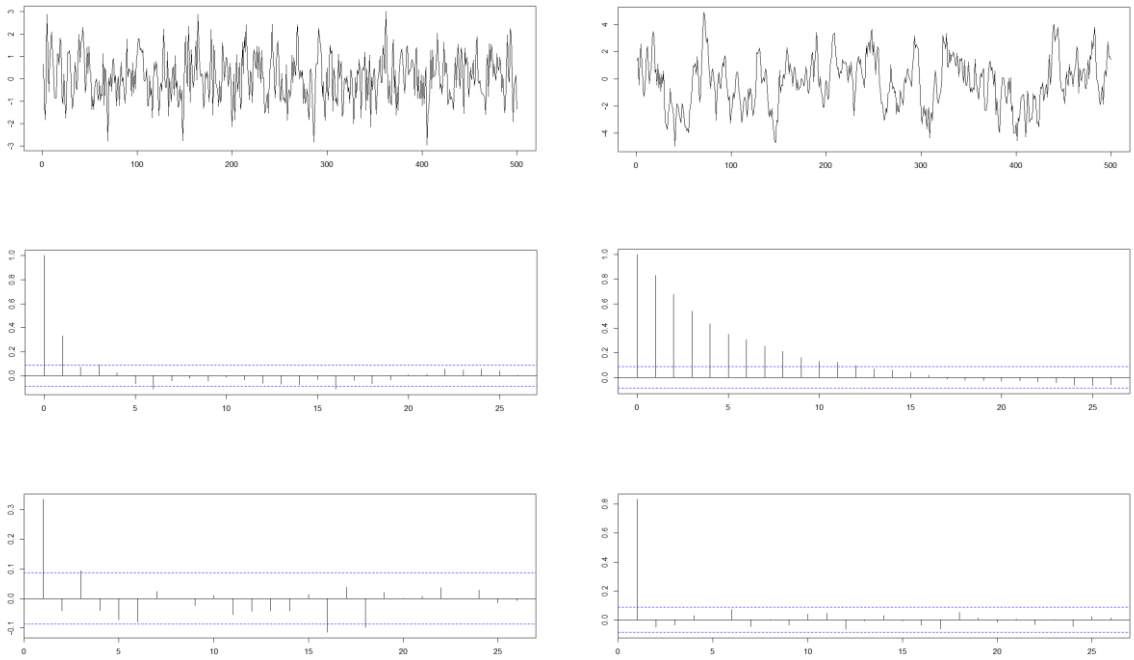


Figure 3: Typical realization of an autoregressive process of order one

From top to bottom: line chart, correlogram of the auto correlation function and correlogram of the partial auto correlation function.
Autocorrelation parameter equal to 0.3 to the left and 0.8 to the right of a typical realization with a sample size of 500.
The blue line in the correlograms indicates the Bartlett's band. The vertical line outside the bend at tick one in the partial autocorrelation indicates that the autoregressive process is at order one.

3.4 Autoregressive moving average process

An autoregressive moving average process, $(ARMA)$ is where the current value of series y depends not only on its previous values but also on the current and previous values of the shocks in the system. It means that a combination of an autoregressive process of order p and a moving average process of order q results in an $ARMA$ process of order p and q, $ARMA(p,q)$.

As an $ARMA(p,q)$ process is a combination of an $MA(q)$ and $AR(p)$, it gets the characteristics of both processes. As discussed, the autocorrelation function of an $MA(q)$ cuts off to zero for lags larger than q, while it decreased geometrically for an $AR(p)$. And the partial autocorrelation function of an $AR(p)$ cuts off to zero for lags larger than p,while it declines geometrically for an $MA(q)$. Therefore, both the autocorrelation function and partial autocorrelation function of an $ARMA(p,q)$ will decline geometrically as lags increases. The autocorrelation function of an $ARMA(p,q)$ will show a combination characteristics of both a moving average and an autoregressive process for the first q lags, but its characteristics will be identical to an $AR(p)$ for lags larger than q (Brooks, 2008, pp. 223-224). Which means the autocorrelation parameters for lags larger than q are identically equal to those of an $AR(p)$. The same could be said about the partial autocorrelation function. The partial autocorrelation parameters for lags larger than p, are equal to those of an $MA(q)$.

A stationary autoregressive process of order p could be written as an $MA(\infty)$. On the other hand, an $MA(q)$ could be written as an infinite order of autoregressive processes, $AR(\infty)$, if the moving average process is invertible, or with another word has unit roots. Verbeek indicates that if the conditions of invertibility and stationarity of a moving average and an autoregressive process is fulfilled, then it will be only "a matter of parsimony" choosing an $MA(q)$, $AR(p)$ or an $ARMA(p,q)$ process. Which means that one could choose a process which is more convenient with the purpose of study. If there is a matter of prediction, it will be more suitable to choose an autoregressive process. While for determining the variances and covariances in a series, it will be more convenient to choose a moving average process (Verbeek, 2004, p. 263).

The data generated for an $ARMA(1,1)$ process shows different behaviors between a highly positive parameter and a lower one as shown in the illustration on the next page. The difference is because of the autoregressive part of the process, since it has been shown that the behaviors in a moving average process is the same for different parameters. Both the autocorrelation and the partial autocorrelation functions shows a combination behavior of the both processes as has been explained further above. For instant, the autocorrelation parameters defined by the first two lags of the autocorrelation function are defining both processes, while the rest are identical with an $AR(p)$. The same holds for the partial autocorrelation function. The first partial autocorrelation parameter is defined for both processes, and the rest are identical for an $MA(q)$.

When we explored the properties of an $ARMA$ process we limited it to order one of both MA and AR processes. In addition, we choose to fix the autocorrelation coefficient of the moving average process. The effect from previous shocks are therefore constant in these experiments. The autocorrelation coefficient of the autoregressive process was adjusted as described above.

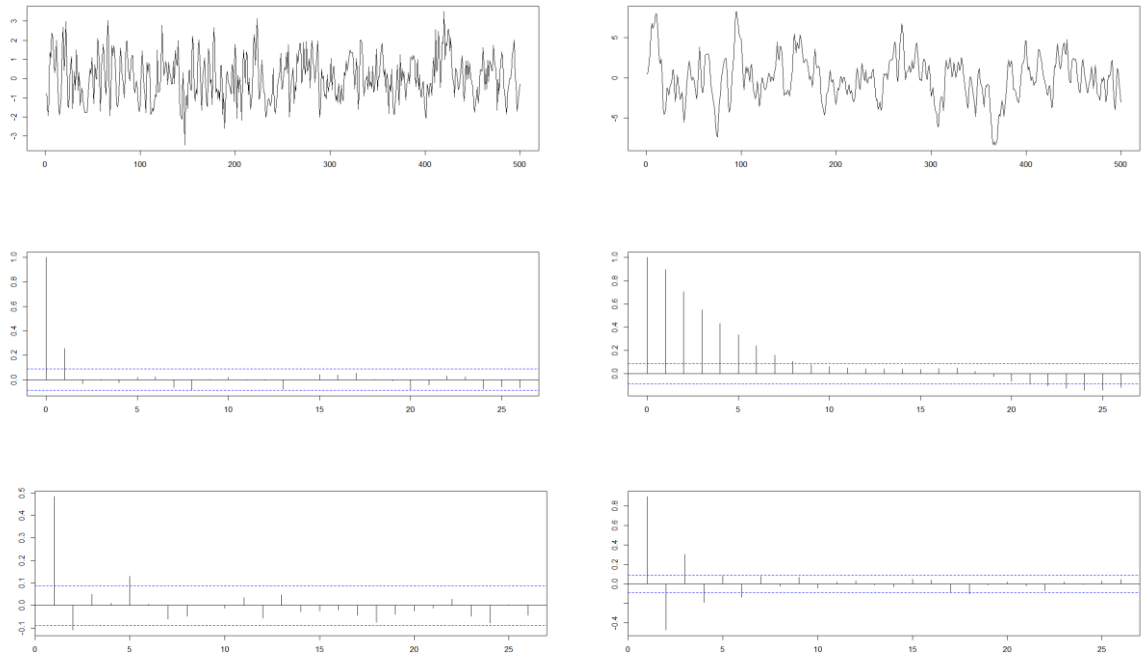


Figure 4: Typical realization of an autoregressive moving average process

From top to bottom: line chart, correlogram of the auto correlation function and correlogram of the partial auto correlation function. Sample size of 500.

Correlation parameter equal to 0.3 for both $\mathrm{AR}(1)$ *and* $\mathrm{MA}(1)$ *to the left and 0.8 for both* $\mathrm{AR}(1)$ *and* $\mathrm{MA}(1)$ *to the right.*

The blue line in the correlograms indicates the Bartlett's band.

3.5 Simulation of sample series of stochastic processes

We need to generate artificial sample series by means of the mentioned stochastic processes to be able to explore their network properties and we used the programming tool R for this purpose (our programming codes are added to the appendix).

Our simulation strategy involved both an example run and a data run. The data run had 100 replications of each parameter summarizing the result in a boxplot diagram. This run used the random number generator which produces a different data series each time, ensuring the estimations validity. The example run does not have any replications, but results in outputs regarded as a "typical" realization of the different sample series. This run has a set seed values which ensures that the errors are identical in each case. This guarantees that the only change will be due to the data generating process. The example run has a set sample size at 1000 datapoints. All graph plots will be forced into the same graph-layout making it possible to detect how the edge pattern change in the different stochastic processes. The data run will be used to examine the general network properties while the example run will be used to visually display a representation of each type and their differences.

White noise

The simulations consist of a series of random numbers generated by a process using a Gaussian distribution, as described above, with a mean of zero and a standard deviation of one. The sample size, or length, of the series are varied from 100 to 3000 and is regarded as the simulation's parameter.

The descriptive statistics and line chart from the example realization are presented below. This, as in all the different simulations, will be the data used throughout the thesis as examples of a typical realization. The values of the mean and the standard deviation are close to the zero and one, and the line chart displays the expected mean reversion property.

	Mean	Std	Max	Min
WN	-0.09	1.01	2.79	-3.30

Table 1: Descriptive statistics of the example realization of white noise

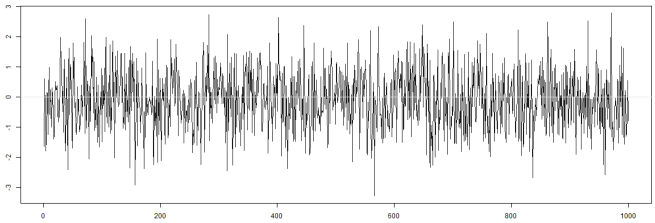


Figure 5: Line chart of the example realization of white noise

Moving average process of order one

The simulations consist of series generated by an $MA(1)$ process, as described above, where the autocorrelation coefficient is varied from -0.9 to 0.9. Length of the sample series is kept constant at 1000 data points.

The descriptive statistics and line chart from selected parameters from the example realizations are presented below. From the descriptive statistics we can see that the $MA(1)$ with the parameter of zero is indeed equal to the statistics of white noise in table 1. This is expected when the previous error terms effect is nulled out and we are using the same current error terms in all example simulations. Another notable observation evident in both the statistics and in the line charts, is that the $MA(1)'s$ with a negative autocorrelation coefficient have values and traits closer to white noise than $MA(1)'s$ with positive parameters, especially when examining the higher values. Complete presentation of all the example line charts can be found in the appendix.

	Mean	Std	Max	Min
-0.9	-0.01	1.36	4.30	-3.93
-0.8	-0.02	1.30	4.05	-3.73
-0.7	-0.03	1.24	3.81	-3.63
-0.6	-0.04	1.18	3.56	-3.58
-0.5	-0.05	1.13	3.31	-3.54
-0.4	-0.06	1.09	3.09	-3.49
-0.3	-0.07	1.06	2.97	-3.44
-0.2	-0.07	1.03	2.85	-3.39
-0.1	-0.08	1.02	2.73	-3.34
0	-0.09	1.01	2.79	-3.30
0.1	-0.10	1.02	2.91	-3.25
0.2	-0.11	1.03	3.03	-3.20
0.3	-0.12	1.05	3.15	-3.15
0.4	-0.13	1.09	3.26	-3.20
0.5	-0.14	1.13	3.38	-3.33
0.6	-0.15	1.18	3.50	-3.46
0.7	-0.16	1.23	3.62	-3.59
0.8	-0.17	1.29	3.77	-3.73
0.9	-0.18	1.36	3.94	-3.92

Table 2: Descriptive statistics of the example realization of $MA(1)$

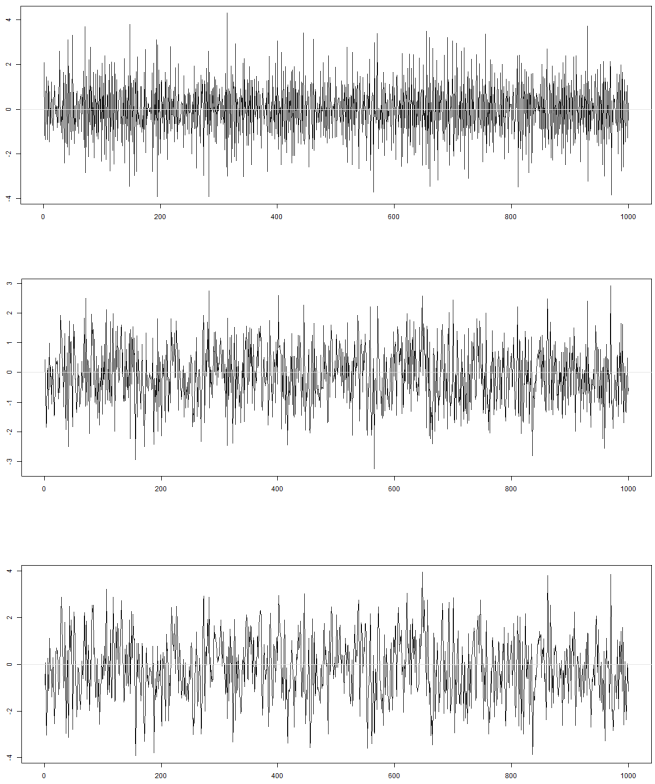


Figure 6: Line chart of the example realization of $MA(1)$ Autocorrelation coefficients from the top: -0.9, 0.1 & 0.9

Autoregressive process of order one

The simulations consist of series generated by an autoregressive process of order one, as described above, where the parameters are varied from -0.9 to 0.9. Sample size is kept constant at 1000 data points.

The descriptive statistics and line chart from selected autocorrelation coefficient from the example realization are presented below. In addition to the mentioned control where the zero parameter statistics is equal to the white noises, we can observe that, contradictory to the observations of the $MA(1)$, changes in the $AR(1)$'s autocorrelation coefficient results in the generation of vastly different sample series. The difference between the $AR(1)$ and white noise is palpable both when the parameter is large in both the positive and the negative end of the scale. Complete presentation of all the example line charts can be found in the appendix.

	Mean	Std	Max	Min
-0.9	-0.04	2.11	6.60	-6.63
-0.8	-0.05	1.60	4.66	-4.98
-0.7	-0.05	1.37	3.86	-4.35
-0.6	-0.05	1.24	3.39	-4.00
-0.5	-0.06	1.15	3.21	-3.61
-0.4	-0.06	1.10	3.07	-3.47
-0.3	-0.07	1.06	2.97	-3.43
-0.2	-0.07	1.03	2.86	-3.39
-0.1	-0.08	1.02	2.73	-3.34
0	-0.09	1.01	2.79	-3.30
0.1	-0.10	1.02	2.90	-3.25
0.2	-0.11	1.03	2.98	-3.22
0.3	-0.13	1.06	3.03	-3.21
0.4	-0.15	1.10	3.28	-3.25
0.5	-0.18	1.16	3.57	-3.53
0.6	-0.22	1.27	3.91	-4.05
0.7	-0.29	1.43	4.29	-4.70
0.8	-0.42	1.73	4.79	-5.36
0.9	-0.82	2.45	7.01	-5.94

Table 3: Descriptive statistics of the example realization of $AR(1)$

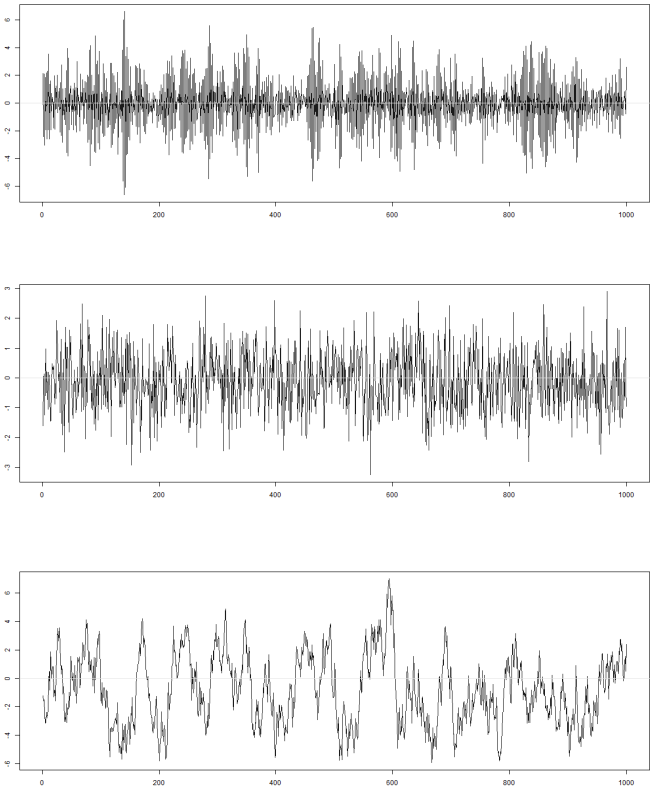


Figure 7: Line chart of the example realization of $AR(1)$

Autocorrelation coefficients from the top: -0.9, 0.1 & 0.9

Autoregressive moving average process

The simulations consist of sample series generated by an $ARMA(1,1)$ process as described above. The autocorrelation coefficient for the $MA(1)$ process is fixed at 0.1 and the autocorrelation coefficient for the $AR(1)$ process varies from -0.9 to 0.9. Sample sample size is kept constant at 1000 data points.

The descriptive statistics and line chart from selected autocorrelation coefficient from the example realization are presented below. The realization from the $ARMA(1,1)$ process is very similar to the $AR(1)$ process, but the presence of the $MA(1)$ can be detected when examining the descriptive statistics. As we see, the $ARMA(1,1)$s values in the summary statistics not equal to white noise when the autocorrelation coefficient of the $AR(1)$ is zero. Which is natural since the autocorrelation coefficient of the $MA(1)$ process is constant. Complete presentation of all the example line charts can be found in the appendix.

	Mean	Std	Max	Min
-0.9	-0.05	1.93	5.93	-6.03
-0.8	-0.05	1.48	4.19	-4.51
-0.7	-0.05	1.28	3.57	-4.07
-0.6	-0.06	1.17	3.27	-3.72
-0.5	-0.06	1.10	3.05	-3.47
-0.4	-0.07	1.06	2.95	-3.43
-0.3	-0.08	1.03	2.85	-3.39
-0.2	-0.08	1.02	2.73	-3.34
-0.1	-0.09	1.01	2.79	-3.30
0	-0.10	1.02	2.91	-3.25
0.1	-0.11	1.03	3.00	-3.21
0.2	-0.12	1.05	3.07	-3.18
0.3	-0.14	1.09	3.22	-3.18
0.4	-0.17	1.15	3.49	-3.36
0.5	-0.20	1.23	3.80	-3.79
0.6	-0.24	1.34	4.16	-4.35
0.7	-0.31	1.53	4.56	-5.05
0.8	-0.46	1.87	5.23	-5.75
0.9	-0.90	2.68	7.69	-6.42

Table 4: Descriptive statistics of the example realization of $ARMA(1,1)$

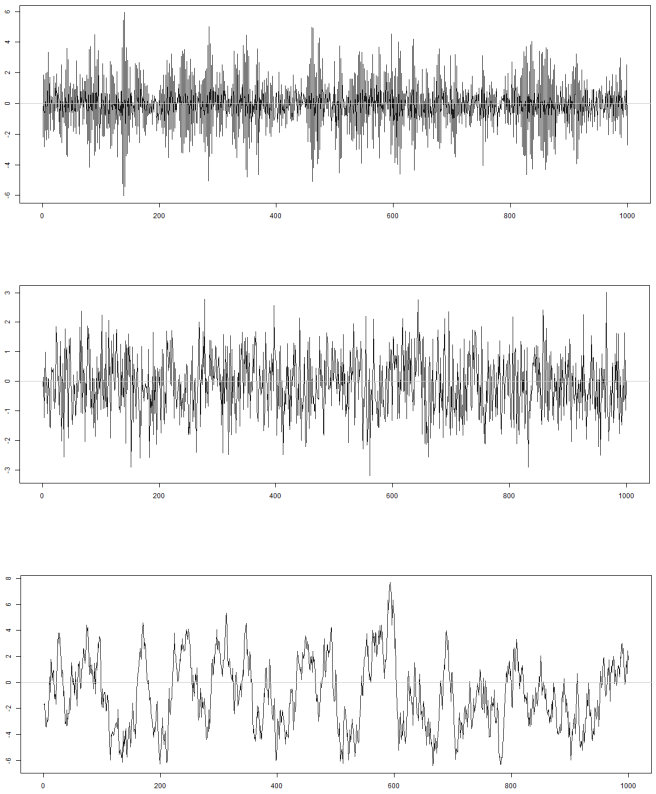


Figure 8: Line chart of the example realization of $ARMA(1,1)$
Autocorrelation coefficients from the top:
$AR(1)$: -0.9 & $MA(1)$: 0.1
AR(1): 0.1 & MA(1): 0.1
AR(1): 0.9 & $MA(1)$: 0.1

4 CREATION AND ANALYSIS OF VISIBILITY GRAPHS

4.1 Visibility graph

The visibility graph algorithm was, as mentioned earlier, proposed by Lacasa et al. (2008) as a mapping between time-series and network graph. Their study shows that several properties of the time-series will be transformed to the generated network, and these could in turn be used to analyze some of time-series properties. The visibility algorithm creates a node for each measure in the time-series such that the number of nodes in the network always will be equal to the length of the original series. These nodes are connected, they share an edge, if they "see" each other. To determine this Lacasa et al. established a geometric criterion called the visibility criteria. The criteria proposed that two arbitrary values $(t_a,\ y_a)$ and (t_b, y_b) are considered as connected, if and only if, there are no intermediate data values like (t_c, y_c) that is deviating from visibility criteria below:

$$y_c < y_b + (y_a - y_b)\frac{t_b - t_c}{t_b - t_a} \tag{9}$$

Thus, the visibility only exists if the connection-line of visibility between any two nodes does not cross the intermediate nodes' connection-line. In other words, if the steepness of the connection-line between two nodes is less than the steepness of intermediate nodes connection-lines with one of the nodes in question, it will fulfil the visibility criteria. The resulting network can be represented mathematically as an adjacency matrix. The matrix is constructed by mapping the connections between each pair of the nodes and is either binary or weighted. Each elements of the binary adjacency matrix' input is either zero or one and is constructed such that

$$A_{ij} = \begin{cases} 1 & if\ nodes\ i\ and\ j\ are\ connected \\ 0 & if\ there\ is\ no\ conenction\ between\ nodes\ i\ and\ j \end{cases} \tag{10}$$

Newman (2010, p. 111) pointed out that, if there are no self-edges in the network, the nodes are not connected to themselves, the diagonal elements of the adjacency matrix will be zero.

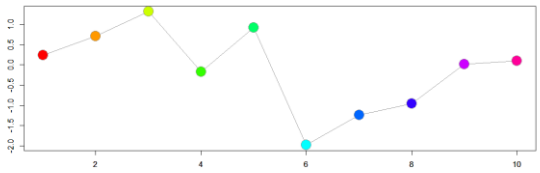
When using a binary adjacency matrix some of the time-series information, as its structural properties, will be lost. Therefore, this mapping is considered as irreversible – the constructed network cannot be converted back to a time-series and still have the same structural properties. The weighted adjacency matrix' input can take on any value but are usually positive. This is an additional set of information from the time-series received by the network expressing the strength of the connections in a network. Lacasa et al. (2008) suggested using the slope of visibility lines as weights, thus making the visibility graph reversible. The purpose of this paper is to examine properties of different processes; thus, we will use binary adjacency matrix, disregarding any information lost in the mapping.

Lacasa et al. (2008) presented three properties of the visibility graphs in their study. The first property is that every node in the visibility graph is at least connected to its nearest neighbor(s), thus there will never be loose nodes (unconnected nodes) in the graph. All of the nodes will be connected to each other either directly or via the intermediate node. The second property is that the edges are defined as undirected. Based on this property there will be no difference between

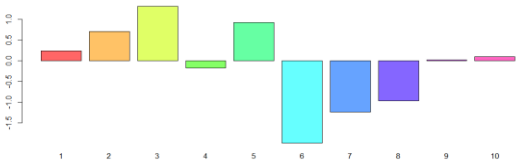
the in-coming and out-going edges to a node. When the connections in a visibility graph are undirected the network will have a symmetric adjacency matrix with n^2 elements, where n is the number of the nodes in a network (Newman, 2010, pp. 110-113). And the last property is that, there will be no difference in the visibility graphs if the horizontal or vertical axes are rescaled or time-series is translated.

The transformation from time-series to visibility graph is rather straight forward and is explained with the help of a small example located on the next page. Time-series are usually presented as line charts as in figure a. To visually demonstrate which realizations that fulfil the visibility criterion we need to convert this chart to a bar plot, b, and then adjust the bar plot such that all realizations are positive, c. This transformation is totally unproblematic since visibility graphs are scale free. The adjusted bar plot can then be used to examine the realizations visibility of each other. The connection exists when we can draw a connection-line from a column to another without it crossing a third column. If the value of realization 2 (orange) was a little higher for example, it would break the visibility line between the nodes 1 (red) and 3 (yellow) and they would no longer be connected. By examining realization 5 (green) we also observe how the steepness of the visibility-lines change: the realizations are connected if the steepness of the connection-line is less than the steepness of connection-lines linking intermediate realizations to that same node. Figure d is a cleaner visualization of the connections shown in c. The straight-line show that all realizations are connected to their immediate neighbor(s). Some realizations have additional connections represented as the curves below and above the straight line. These connections are then transferred to the adjacency matrix, figure e, who, as explained earlier, is a binary matrix where connections have a value of 1. The diagonal (grey) consists only of zeros because no node is connected to itself. The values directly before and after the diagonal is one, which is a result of all the realizations being connected to its immediate neighbor. Additional connections will be marked as one in the matrix, and since there is no direction, the matrix is symmetric. Realization 10 (pink) is at the end of our series, thus it does only have one neighbor, 9 (purple), and this connection is represented in the adjacency matrix $A_{9,10}$ and $A_{10,9}$. In addition, realization 10 have a connection to realization 5 at $A_{5,10}$ and $A_{10,5}$. The other values in both column and row 10 is equal to zero – no other connections. Finally, from the adjacency matrix we can create the finished visibility network in figure f. Each realization is represented as a node, and each connection converted to an edge. Note that all six images in the illustration represent the same information about the data set: how the value of the realizations relate to each other.

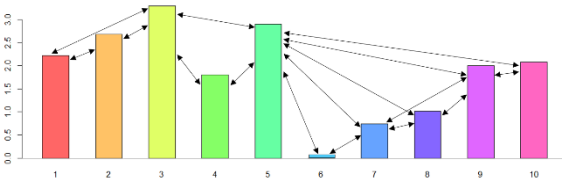
a)



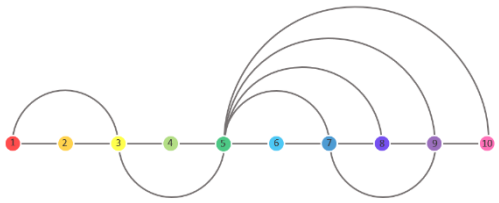
b)



c)



d)



e)

	1	2	3	4	5	6	7	8	9	10
1	0	1	1	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0	0
3	1	1	0	1	1	0	0	0	0	0
4	0	0	1	0	1	0	0	0	0	0
5	0	0	1	1	0	1	1	1	1	1
6	0	0	0	0	1	0	1	0	0	0
7	0	0	0	0	1	1	0	1	1	0
8	0	0	0	0	1	0	1	0	1	0
9	0	0	0	0	1	0	1	1	0	1
10	0	0	0	0	1	0	0	0	1	0

f)

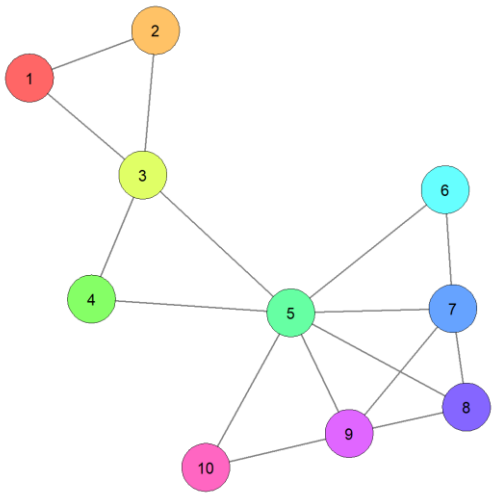


Figure 9: From time-series to visibility graph - a small example

The illustration consists of six images: from the top left:

- *a) Line chart displaying the 10 first realizations of an $AR(2)$*
- *b) Bar plot of the data in a*
- *c) Adjusted bar plot of the data in a where there is an added constant equal to lowest realization in the sample series*
  Connection-lines illustrate which pair of realizations fulfills the visibility criterion
- *d) Alternative illustration of how the realizations are connected.*
- *e) The adjacency matrix*
- *f) The visibility graph*

The illustrations are color-coded where each realization is assigned a color; this color will follow the realization on its transformation from a value in a series to a node in the network. As an example: the first observation in the time-series, which color is red, corresponds to the red column in the box plots, the red column & row in the adjacency matrix and the red node in the network.

The color-code is also consistent in both the example for visibility graph, and horizontal visibility graph

4.2 Horizontal visibility graph

A modification of the visibility algorithm called the horizontal visibility algorithm where proposed by Luque et al. (2009). This algorithm is generating the horizontal visibility graph, a subgraph of the visibility graph, which also can be used as a tool to distinguish chaotic series from random ones. The horizontal visibility algorithm requires a horizontal visibility line connecting two nodes without any intersection by intermediate nodes.

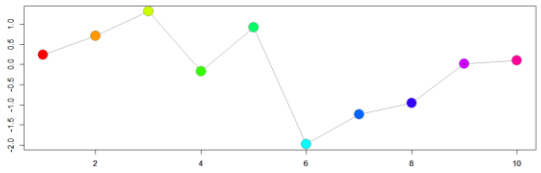
$$x_i , x_j > x_n \quad \forall n , \; i < n < j \tag{11}$$

The two nodes x_i and x_j, are considered as connected in a horizontal visibility graph, if and only if they both are higher than any other nodes standing in between them. Lacasa et al. (2009) defined the horizontal visibility algorithm as both reversible and irreversible. As mentioned, a visibility graph is determined by the adjacency matrix, which could be either a binary or weighted matrix. The horizontal visibility graph could easily be made reversible by using the differences in values as weights in the adjacency matrix.

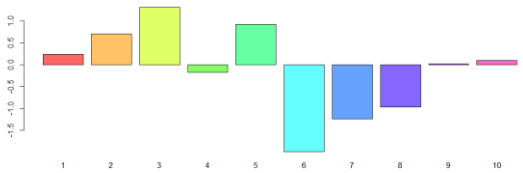
The horizontal visibility graph has some additional properties in regard to the visibility graph. Firstly, the horizontal visibility algorithm can generate both directed and undirected graphs. In a directed graph the edges have direction and therefore one should distinguish between the in-coming and out-going edge connected to each node. The directionality of the edges makes, as mentioned above, an asymmetric adjacency matrix. Secondly, the visibility criterion in the horizontal visibility graph is more restrictive than the general visibility criteria. This results in a graph with few connections which makes the horizontal visibility algorithm faster and the resulting graph easier to analyze (Luque et al., 2009).

The transformation from time-series to horizontal visibility graph is even more straight forward than the visibility graph and is also explained with the help of a small example located on the next page. The data set in the example are the same as in the previous, thus we can compare the two networks and note their differences. Figures a, b as well as the adjustments in c have been explained above. The connection between two realizations in figure c exist if we can draw a horizontal connection-line from a column to another without it crossing a third column. If the value of realization 7 (light blue) had a larger value than the realization of 8 (dark blue) for example, it would break the horizontal visibility line between 8 and 5 (green) and they would no longer be connected. A cleaner representation of the connections can be observed in figure d, which is explained above. Here it's easy to observe that the horizontal visibility algorithm generates fewer connections than the visibility graphs in the previous example. Both connection between $1\&3$ and $7\&9$ in the visibility graph doesn't exists in the horizontal visibility graph. By this algorithm the values of realization 2 and 8 are large enough to prevent visibility. Connections will also in the case of horizontal visibility graph be transferred to an adjacency matrix, figure e. As in the previous example and have the same properties but, as discussed, fewer values of 1. The final result, the horizontal visibility network in figure f is generated from the adjacency matrix. It has several similarities with the visibility graph, as is expected since it's a subgraph, and the differences are the two missing connections we already mentioned between $1\&3$ and $7\&9$.

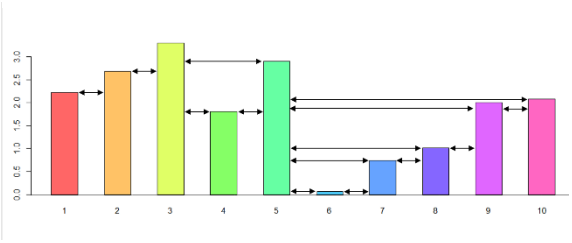
a)



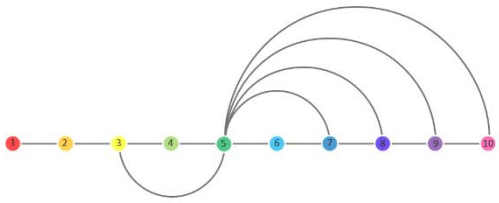
b)



c)



d)



e)

	1	2	3	4	5	6	7	8	9	10
1	0	1	0	0	0	0	0	0	0	0
2	1	0	1	0	0	0	0	0	0	0
3	0	1	0	1	1	0	0	0	0	0
4	0	0	1	0	1	0	0	0	0	0
5	0	0	1	1	0	1	1	1	1	1
6	0	0	0	0	1	0	1	0	0	0
7	0	0	0	0	1	1	0	1	0	0
8	0	0	0	0	1	0	1	0	1	0
9	0	0	0	0	1	0	0	1	0	1
10	0	0	0	0	1	0	0	0	1	0

f)

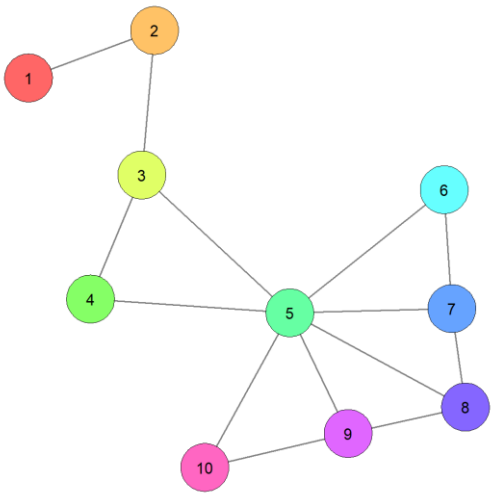


Figure 10: From time-series to horizontal visibility grap - a small example

The illustration consists of six images: from the top left:

- *a) Line chart displaying the 10 first realizations of an $AR(2)$*
- *b) Bar plot of the data in a*
- *c) Adjusted bar plot of the data in a where there is an added constant equal to lowest realization in the sample series*
  Connection-lines illustrate which pair of realizations fulfils the visibility criterion
- *d) Alternative illustration of how the realizations are connected.*
- *e) The adjacency matrix*
- *f) The horizontal visibility graph*

The illustrations are color-coded where each realization is assigned a color; this color will follow the realization on its transformation from a value in a series to a node in the network. As an example: the first observation in the time-series, which color is red, corresponds to the red column in the box plots, the red column & row in the adjacency matrix and the red node in the network.
The color-code is also consistent in both the example for visibility graph, and horizontal visibility graph

4.3 Network statistics

There are several different network metrics and measurements. In this thesis we are running multiple simulations of time-series with a high number of observations. With this type of data sets, looking at local values for each and every node would not be a sound strategy. We have therefore chosen to focus on some of the common global network measurements which provide important information about the properties of the networks. Global network properties are either describing the network as a whole or are a calculated mean value from its local values.

Degree

Newman (2010, p. 133) defined the degree of a node as the total number of its connected edges. An edge is what we earlier mentioned as the connection-line between vertices. In our examples above, the first node (red) have a degree of two in the visibility graph, but only a degree of one in the horizontal visibility graph.

The number of degrees is dependent on whether the network is directed or undirected. The edges in an undirected network have no orientation. Therefore, there will be no difference between the incoming and outgoing edges to a node. Considering the adjacency matrix of an undirected network, A_{ij}, the nodes degree could be written as:

$$k_{\mathrm{i}} = \sum_{i=1}^{n} A_{\mathrm{ij}} \tag{12}$$

In a directed network on the other hand there is two degrees to be considered for each node, the in-degree and out-degree. The in-degree of a node is computed as the number of the nodes that are pointing to the node in question. The out-degree is computed as the number of other nodes the node in question is pointing to. Considering the adjacency matrix of a directed network, the incoming degree and the outgoing degree could be computed as:

$$k_{\mathrm{i}}^{\mathrm{in}} = \sum_{i=1}^{n} A_{\mathrm{ij}} \tag{13}$$

$$k_{\mathrm{j}}^{\mathrm{out}} = \sum_{i=1}^{n} A_{\mathrm{ij}} \tag{14}$$

In our experience, a nodes degree in a time-series correspond to the extremeness of the value in the original time-series, relative to the values in its immediate vicinity. As in our examples above, node five has a degree of seven in both visibility and horizontal visibility graphs. By examining observations connected to node five, we discover that the closest are significantly lower in value. This is more of a general rule in visibility graph than with the horizontal visibility graph, where the effect will be lessened by the horizontal visibility criterion.

We are not the first to make this observation. In their article examining the financial market by the means of visibility graphs Zhuang stated that “The measure of degree allows us to find those important incidents that influence market integration, for example the 2008 financial crisis” (Zhuang et al., 2014).

Mean degree

A node degree is calculated for each individual node and is as such a local network statistic. Since our paper is built on the global statistics measures of a network, we are going to use the mean of the nodes' degrees instead of the degrees of each individual node in the network. The arithmetic mean of the nodes degrees in an undirected network is defined as

$$c = \frac{1}{n}\sum_{i=1}^{n} k_i \tag{15}$$

And measures for the mean degree in a directed network, the mean in-degree and mean out-degree, are constructed as

$$c_{in} = \frac{1}{n}\sum_{i=1}^{n} k_i^{in} \tag{16}$$

$$c_{out} = \frac{1}{n}\sum_{i=1}^{n} k_j^{out} \tag{17}$$

The mean degree in our examples, which are undirected networks, are respectively 3.2 for the visibility graph and 2.8 for the horizontal visibility graph, illustrating the properties of fewer degrees in the horizontal visibility graph.

Cumulative degree distribution

The probability that a node in a network has degree k is referred to as the degree distribution $p(k)$. It measures the fraction of the nodes having a given degree, k, and gives the frequency distribution of the node degrees in the network. The distribution has a tail of high degree nodes, which means that the fraction of the nodes having a small degree is higher than the fraction of having a higher degree in a network. Thus, the distribution of the node degrees is considered to be right-skewed. The degree distribution can be computed for both directed and undirected networks (Newman, 2010, pp. 243-246).

One of the interesting features of a degree distribution is whether it is power law distributed or not. According to E. J. Newman (2004), a quantity is considered to be power laws if the probability of having a given value like k differs inversely as a power of that value. Thus, a power law degree distribution is constructed as $p_k = Ck^{-\alpha}$. α is the scaling parameter with the range $2 < \alpha < 3$ and C is a fixed constant used while dealing with a normalized degree distribution. A power law distribution could be visualized by a histogram of the degree distribution as well as by constructing its cumulative degree function. The cumulative distribution function measures the fraction of nodes having a degree greater or equal to k, $P(K \geq k)$ (Newman, 2010, pp. 247-256). According to Newman if a degree distribution is power laws with a scaling parameter of α, then its cumulative distribution function will also follow a power law distribution with a scaling parameter of $\alpha - 1$.

Networks with degree distributions following a power law distribution are called scale-free networks (Newman, 2010, p. 249). Barabási and Albert (1999) proposed the scale-free networks, the networks where their local connectivity distributions are free of scale, and where the possibility of a new node, A , entering the network joining an existence node, B, is dependent of the degree of node B. A scale-free network in contrast to a random network contains some high

degree nodes as well as nodes with lower number of connections (Barabási & Bonabeau, 2003). A new node added to the system will have high tendency to join a high degree node rather than a node with a lower number of connections, which is referred to as the rich get richer (Barabási & Albert, 1999).

To identify whether a given data set follows a power law distribution we use a goodness-of-fit test. The test hypothesis is constructed as:

$$H_0 : p(k) \approx k^{-\alpha} \quad the\ data\ set\ is\ drawn\ from\ a\ power\ law\ distribution \tag{18}$$

$$H_1 : p(k) \not\approx k^{-\alpha} \quad the\ data\ set\ is\ not\ drawn\ from\ a\ power\ law\ distribution \tag{19}$$

The hypothesis is tested based on the distance between the distribution of the given data set and the hypothesized model, which is assumed to a power law distributed model, and the Kolmogorov-Smirnov or so-called KS statistic is used as a measure of the maximum value of the absolute distance between those two models. The KS test statistic is constructed as:

$$D = \max_{x \geq x_{min}} |S(x) - P(x)| \tag{20}$$

where $S(x)$ is defining the cumulative distribution function of the given data, and $P(x)$ is expressing the cumulative degree distribution of the power law distributed fitted model (Clauset, Shalizi, & Newman, 2009).

According to Clauset et al. (2009), this distance has to be compared to a comparable synthetic distance. The synthetic distance is calculated from the distance between the empirical data set and a synthetic data set, which is similar to the empirical data below x_{min}, and is power laws thereafter. The p-value of the test is defining the probability of the synthetic distance being larger than the empirical distance between the original data set and the power law distributed fitted model. The conclusion of the test could be drawn from comparing the p-value and the considered significance level of the test. If the p-value is less or equal to the significance level, the conclusion will be to reject the null hypothesis and that the empirical data set is not following a power law distribution. Otherwise, we fail to reject the null hypothesis and the evidence will be supporting that the empirical data set is following a power law distribution.

The tests p-value in our examples are 0.4 in the visibility graph and 0.16 in the horizontal visibility graph. Both values are larger than the significance level of 5%, giving evidence for the null hypothesis and we fail to reject this. Both cumulative degree distributions follow a power law distribution (are power laws) and our network examples are scale free.

Normalized degree centrality

Centrality is a network property which is used to measure how influential a node or a network is. There are different types of network centrality, some of them are focusing on the centrality of a given node and some on the centrality of a network as a whole. One of these measurements is degree centrality. This is a local measure and is using the degree of a node as a representation of

the node's influence. A high degree node is considered to be more central, thus more influential, than a node with lower degree. The local measurement is converted to a global one by:

$$C(G) = \sum_v \left(\max_w c_w - c_v \right) \tag{21}$$

Where c_v is the local degree centrality and c_w is the local theoretical max. The global degree centrality is normalized by dividing the value by the global theoretical max. This is the highest centrality score possible attained by a network with the same number of nodes (Csardi, 2019).
Our examples above have a normalized degree centrality of 0.47 for the visibility graph and 0.52 in the horizontal visibility graph. Both graphs have a global theoretical max of 81. In these small examples we can easily see that there is one node, node five, which has a far larger node degree than the rest. This very central node has seven edges in both graphs. If we then look at mean degree in both cases, we realize that the difference between the degrees of the most central node and the mean degree are less in the visibility graph than in the horizontal visibility graph. This in return explain why the visibility graphs degree centrality is less than the horizontal visibility graphs – a network is less central when the nodes have more or less the same number of edges which in a time-series relates to few extreme values. In a more general sense, the measurement of normalized degree centrality can be used to determine if there are extreme values in a time-series.

Transitivity

Considering two nodes A and B which both are connected to node C. Newman (2003) is defining the transitivity as the mean probability that nodes A and B are themselves connected. In other words, transitivity is the probability that two randomly chosen nodes, which both are connected with a third node, themselves are connected thus creating a triple. The transitivity of an undirected network can be defined as the fraction of closed paths of length two in the network (Newman, 2010, p. 201).

Transitivity have both a local and a global measurement. The global transitivity of a network is constructed as:

$$C = \frac{(number\ of\ closed\ paths\ of\ length\ two)}{(number\ of\ paths\ of\ length\ two)} \tag{22}$$

Where C will be a parameter between 0 and one. $C = 1$ shows the existence of perfect transitivity and $C = 0$ refers to cases where there are no triples in the network. Existence of perfect transitivity requires all nodes being connected to each other. Since the probability of having a perfect transitivity is so small, it is considered to be a useless concept in network. Perfect transitivity, a clustering parameter of zero is also highly unlikely. Therefore, a partial transitivity is considered to be a "very useful" concept in network.

Local transitivity is defined for a single node. It controls the flows between the nodes' neighbors and is assumed to be strongly correlated with the between-ness centrality of a network, a measurement of how influential a given node is.

$$c = \frac{(number\ of\ pairs\ of\ neighbors\ of\ i\ that\ are\ connected)}{(number\ of\ pairs\ of\ neighbors\ of\ i)} \tag{23}$$

Newman (2010, pp. 203 - 204) expressed that small groups of nodes are expected to have a higher local clustering parameter. It is because of the existence of a smaller number of pairs of neighbors for each node in a small group, which makes the denominator of the equation above small. On the other hand, there will be more pairs of neighbors for nodes in a large group which makes the denominator large and the local transitivity small. It means the local transitivity is negatively dependent of nodes' degree in a network (Lacasa & Toral, 2010; Ravetti et al., 2014).

The transitivity values in out examples are 0.51 (visibility graph), and 0.41 (horizontal visibility graph). Two nodes, both connected to a third have a 51% chance of themselves being connected in the visibility graph and a 41% chance in the horizontal visibility graph. If we examine the examples, we do discover that the number of triples is larger in the visibility graph.

Transitivity levels in the horizontal visibility graph are particularly interesting because it indicates that two nodes are separated by one or more intermediate nodes with a lower value. A typical case will be the relationship between node three, four and five in the example. These three nodes are parts of a triple, and if we examine the line chart, we see that the values of three and five are larger than two. When the transitivity value is large it indicates that the original time-series often have dips in the value followed by a correction. In financial time-series this can be used to indicate investment opportunities. This interpretation cannot be extended to the visibility graph because this algorithm allows for connection through angles, this allows for multiple events that can produce triplets.

Assortativity

Assortativity or homophily is measuring the tendency of the nodes being connected with nodes of the same pattern. According to Newman (2010, pp. 221 - 222), a network is considered to be assortative, or have assortative mixing, if there is a significant number of connections between nodes of similar patterns. The meaning of the term similar pattern is dependent on type of network. For example, in a social network, nodes could be representing peoples which could be classified based on their gender, languages, ethnicity and any other characteristics which could also be a "scalar quantity" like age or income. So, if a significant fraction of the people being connected with whom they are sharing similar characteristics, the network is considered to have assortative mixing. On the other hand, we could have disassortative mixing in a network when most of the connections are between the dissimilar nodes.

Newman (2002) is introducing assortative mixing by degree as the tendency of high degree nodes being connected to other nodes of high degree and visa verse. Therefore, in an assortative mixing by degree network, the most influential nodes will be connected together and become even more important, and this core of high degree nodes will be surrounded by lower degree node. On the other hand, in the disassortative mixing by degree networks, the high degree nodes prefer to

connect to the lower degree nodes and so there will be no clumps in the network. Newman (2010, p. 230) is assuming that networks with disassortative mixing will have a "star-like" feature and are more uniform. Newman (2002) showed that removal of high degree nodes of disassortative mixing networks will differentiate them more than when the network is assortative mixing. it also showed that many of the social networks tend to be assortative mixing while technological and biological networks are disassortative.

The assortativity by degree is constructed as:

$$r = \frac{\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})\, k_i k_j}{\sum_{ij}(k_{ij}\delta_{ij} - \frac{k_i k_j}{2m})\, k_i k_j} \tag{24}$$

where r is called the assortativity parameter and is the Pearson correlation parameter between degrees of connected nodes. A_{ij} in the equation above is a given element of the adjacency matrix, which will be one when nodes i and j are connected and zero otherwise. k_i and k_j are the degrees of two random nodes, i and j, and δ_{ij} is the Kronecker delta which is considered to be one if connected nodes are similar and in the same class, $i = j$ and zero otherwise. An assortative mix of zero is defined for random graphs, where there is randomness in placing the edges between the nodes (Newman, 2002).

In our examples the visibility graph has as assortativity of -0.32, and the horizontal visibility graph of -0.28. Both are negative, have disassortative mixing, indicating that the nodes which are connected have different degree. The probability of a node being connected with another node with a different degree is larger in the visibility graph than in the horizontal visibility graph. By examining the examples, we can see that the visibility graphs nodes have a larger variation in degrees than the horizontal visibility graph does. Assortativity can be used to identify cycles in the original time-series. If we take the values five to nine in our example and repeat this pattern, we have a cyclic series. The repeated value five will always be connected and they will have the same high number of degrees creating clumps as described by Newman and a high positive assortative mix.

Geodesic path

The geodesic path or the average shortest-path length, is the shortest path from one node to another (Newman, 2003). A path is defined as a route connecting two given nodes passing through other nodes standing between them. The length of the geodesic path will be computed either from the number of the edges passing from a node to another or from the number of the nodes standing in between those two give nodes. Mao and Zhang (2013) implies that there is at least one path connecting any two nodes in a network, which is consistent by the property of a visibility graph being connected. But there could be more than one geodesic path between any two nodes (Newman, 2002).

Mao and Zhang (2013) define the average shortest-path length of a given graph G as the average number of edges in the shortest paths between all possible pairs of nodes, which could be constructed for both directed and undirected graphs.

The average shortest-path length of an undirected graph is defined as:

$$ASPL_G = \frac{\sum_{i,j}^{N} dist(v_i, v_j)}{\sum_{i,j}^{N} N(N-1)} \quad (25)$$

where $dist(v_i, v_j)$ is the value of the shortest-path length between all possible pairs of nodes, and the $N(N-1)$ indicates the number of paths in graph. Newman (2010, p. 139) implies the geodesic paths are self-avoiding and do not intersect themselves, which also indicates that there will not be geodesic path between a node and itself. Therefore if $v_i = v_j$, so $dist(v_i, v_j) = 0$.

The shortest path can easily be visualized in our examples. In the visibility graph, the shortest path between one and eight are trough the nodes three and five which have a path length of three. This is not the only path from one to eight however, we can create multiple paths maybe with a detour around two and seven, but we cannot find one that is shorter than three. The average shortest path here is 1.87 which indicates that the path length between any node in the network is less than two. In the horizontal visibility graph, the shortest path between one and eight are through nodes two, three and five and have a length of four. This path is longer because the horizontal visibility algorithm generates graphs with fewer degrees, thus the path must be longer. The average shortest path is 2.07 which is naturally larger than value in the visibility graph. Average shortest path depends both on a networks number of degrees and their centrality.

5 DISCOVERIES

5.1 Variations in network properties due to change in length of the sample series

When working with global network statistics, it is easy to assume that some of these may depend on the original time-series' length. Does average geodesic path for example get exponentially larger with sample size? What about the other network statistics? To gain further insight on how sample size effects the networks properties we decided to record this relationship in series generated from the white noise process. White noise process was the obvious choice because of its lack of parameters. We are also using white noise as a benchmark in our further investigation, and the thorough recording of its network properties will enhance the interpretation of our results.

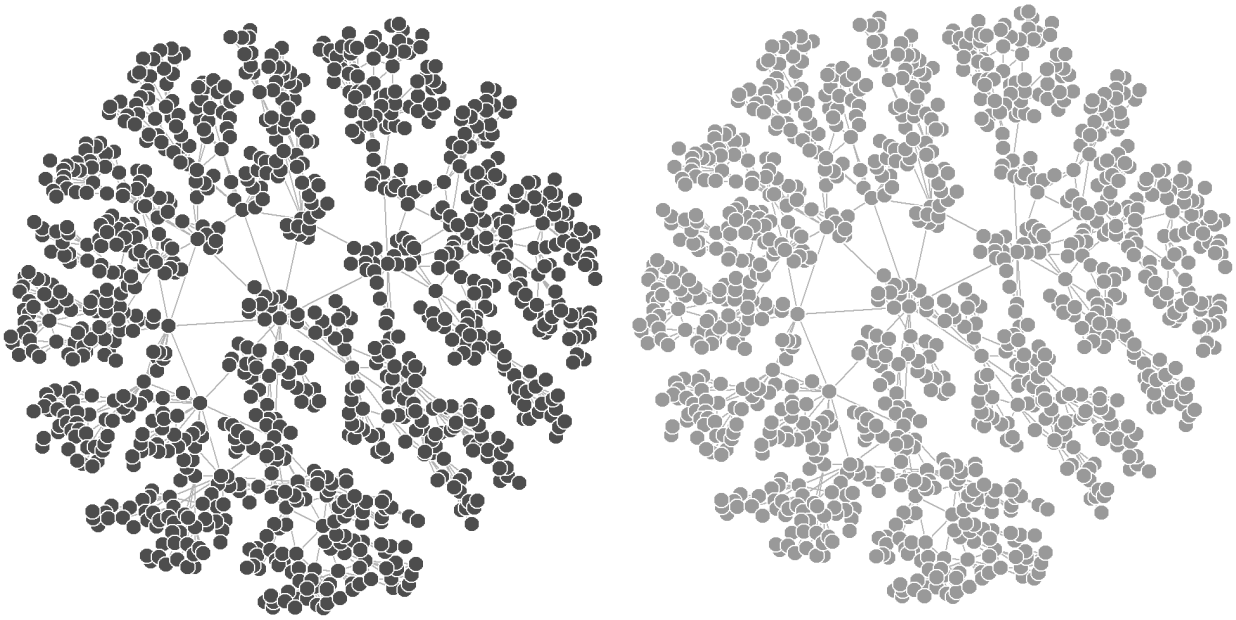


Figure 11: Network from an example realization of the white noise process

Visibility graph on the left (dark grey) and horizontal visibility graph (light grey) on the right

An example of how a time-series generated from a white noise process can present itself as a network is illustrated above. These networks are generated from the typical realizations which we presented earlier in the text, and this will be the case for every network example throughout. The additional typical realizations and their corresponding networks can be found in the appendix.

The visibility graph (on the left) and the horizontal visibility graph (on the right) both have a length of 1000 observations and have as mentioned earlier been forced in the same layout. This makes it difficult to immediately spot the difference between these two. But, as in every five-error game, the differences are there if you only have the patience and look close enough. In this illustration one of the discrepancies can be located at the far-right side where the visibility graph has three visible edges, but the horizontal visibility graph only has two. The reason why we choose to force an identical layout on every graph is because it will make it possible to study the changes in how the nodes are connected dependent on the underlying generating process.

We can observe that these graphs have small clusters, communities, connected in a counterclockwise fashion with very few edges crossing into the center. This is what we would expect in a graph generated form a time-series with regular fluctuations around a center value.

To map the relationship between sample size and the different properties of the network, we generated white noise series of different length. Starting with a length of 100 and ending at 3000 by an interval of 100. We used 100 simulations of each sample size. The results are presented on the next page, visibility graph on the left and horizontal visibility graph on the right. We can clearly see that in all cases, except one, sample size has large impact when it is small, but the effect is diminishing with the increase in length of the sample. We also note that the spread within each simulation tend to reduce as the sample size increases. In addition, we observe that the properties behave very similarly in both types of graphs. Mean degree, normalized degree centrality, transitivity and assortativity all show signs of converging toward a value as the sample size increases. In the case of mean degree, the visibility graph has a larger spread then the horizontal visibility graph which is a result of the latter's more restrictive visibility criterion. A natural result when mean degree stabilizes with sample size, is that the normalized degree centrality will converge toward zero. The theoretical maximum value used when normalizing the degree centrality will increase with the length of the sample. As the measurement of degree centrality is partly dependent on degree, its value will mirror the behavior of mean degree. Since any number divided on an increasingly large number will converge toward zero, the normalized degree centrality will converge to zero as a result of the increasing sample length. The local value of transitivity is negatively dependent on the local degree, and the opposite behavior in the two respective global measurements where expected. When this indeed was the case in our experiment, it affirmed our theory that the relationship also extends to the global measurements.

One of the measurements which does not clearly converge toward a value is, as we prophesied earlier, the average geodesic path. But even this measurement does level out such that changes due to an expanding sample size decrease. We are not the first to discover this however as Sun et al. (2016) states in their article:

> *"We studied the change of the average shortest path length with the increase of the number of nodes in visibility graph network. L is the average shortest path length, N is the number of nodes. If there is linear relationship between L and logarithm of N, then we have L = a + b ln N …" (Sun et al., 2016)*

We do however show that this also is the case for the horizontal visibility graph. This is the only measurement where the spread doesn't reduce as samples increase.

The one network property which clearly differentiate itself from the others in this explorative exercise is the p-values from the test regarding power law distribution. This value is significant for all sample sizes and for both graphs leading us to theorize that networks generated from a white noise process always have a degree distribution which is following power law distribution, thus always are scale free.

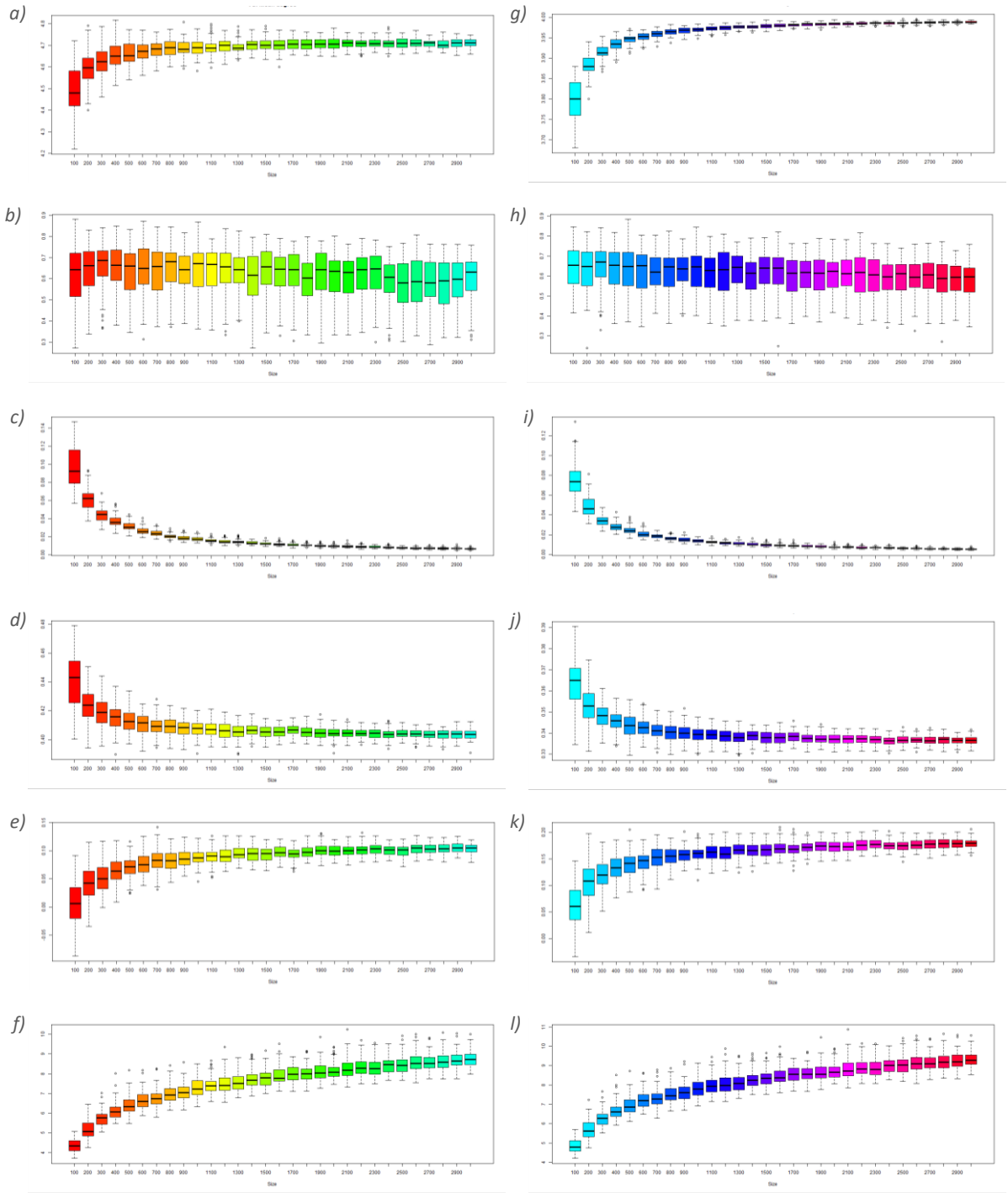


Figure 12: Change in network properties due to the length of sample series

From the top to bottom, visibility graph (VG) on the left side, horizontal visibility graph (HVG) on the right:

- *a) Mean degree, VG*
- *b) P-value, H_0 : cumulative degree distribution follows power law, VG*
- *c) Normalized degree centrality, VG*
- *d) Transitivity, VG*
- *e) Assortativity, VG*
- *f) Average geodesic path, VG*
- *g) Mean degree, HVG*
- *h) P-value, H_0 : cumulative degree distribution follows power law, HVG*
- *i) Normalized degree centrality, HVG*
- *j) Transitivity, HVG*
- *k) Assortativity, HVG*
- *l) Average geodesic path, HVG*

there are 100 simulation for each sample series length, ranging from 100 to 3000 by an interval of 100

5.2 Identifying white noise processes through network statistic

Each of our recorded relationships, except geodesic path and p-values for power laws test, have a reduction in spread as sample size increases. The relationships can in addition, except the p-values, be nicely fitted to different functions. The recorded p-values are always relevant for all tested sample sizes and do not show any signs of a trend, thus we can assume that this will always be the case with a white noise process.

With these characteristics, we propose that it is possible to use network properties to identify white noise series regardless of its stationarity status. This may be achieved both with the visibility graph and the horizontal visibility graph. The horizontal visibility graph is often much preferred because it is fast, easy and analytically solvable. In our results this sub graph actually has a smaller spread, thus will be our recommended algorithm for this identification method.

We suggest a six-part test where the test object must fulfill all six criteria to be identified as a white noise process. First of all, the series must have a degree distribution that is power laws. Then its values for mean degree, normalized degree centrality, transitivity, assortativity and average geodesic path must fit within a confidence interval given by the length of the series.

We choose to use the upper- and lower boundaries as our confidence interval since it accurately reflects the observed changes in spread. For example, a 5% confidence interval would in most cases be far too narrow at small sample sizes and far too wide at the larger ones.

We find the fitted functions for both upper and lower bounds of the statistics by examining two different relation possibilities. At first, we consider the possible significant dependencies of each of the statistics on the logarithm of the sample size. Then we examined whether there is a significant relationship between the logarithm of the statistics and the logarithm of the sample size. These two options could be examined for all of the statistics with exception of assortativity. Assortativity is measured on a scale from -1 to $+1$ and the natural logarithm of minus values, as well as zero, do not exist. Sun et al. (2016) inspired this approach with their proposal of a linear relationship between the average shortest path length and logarithm of the series' length.

We examined the adequacy of the relationships in both options by the use of a linear regression and a partial two-sides Student's t-test of its parameters. The null hypothesis under a Student's t-test is identifying the non-significance of the parameter; that there is no significant relationship between the dependent and independent variables. The alternative hypothesis is assuming a significantly non-zero parameter, indicating an adequate relationship. The test statistics is defined as $t = \frac{\hat{\beta}}{\widehat{se_\beta}}$, where $\hat{\beta}$ is the estimated value of the regression parameter, and $\widehat{se_\beta}$ is the estimated OLS standard error of β. Under the null hypothesis the test statistic will follow a Student's t-distribution with $n-1$ degrees of freedom, (n equal to the number of the observations).

If the probability that the test statistic is larger than its 'empirical value, $p(t > t_{emp})$, is less than the chosen significance level, the test will provide significant evidence for the alternative

hypothesis. This indicates that the relationship between the dependent and independent variables is valid and significant (Hill, Griffiths, & Lim, 2011, pp. 95-113).

We used a 5% significance level in our tests. The results indicated mostly a significant relationship both between the statistics and logarithm of the sample size and between logarithm of the statistics and logarithm of the sample size. While in some cases, there are no significant relationships between either form of the statistics mentioned above and the logarithm of the sample's length. This is the case for transitivity's lower bounds of both graphs and for mean degree's upper bound of the visibility graph. To be able to create these boundaries we investigated the statistics of the other quartiles, establishing that they all had a similar shape. We fitted the best fitting function from the other quartiles to the problematic bound, which resulted in a non-significant relationship when tested. These bounds are therefore not valid and further study is required.

We choose the best fitting function for the statistics with adequate relationship by the use of adjusted R-squared. The chosen fitted functions is the one with the highest adjusted R-squared. The resulting functions plotted with our realizations are displayed below and the regression analysis are available in the appendix.

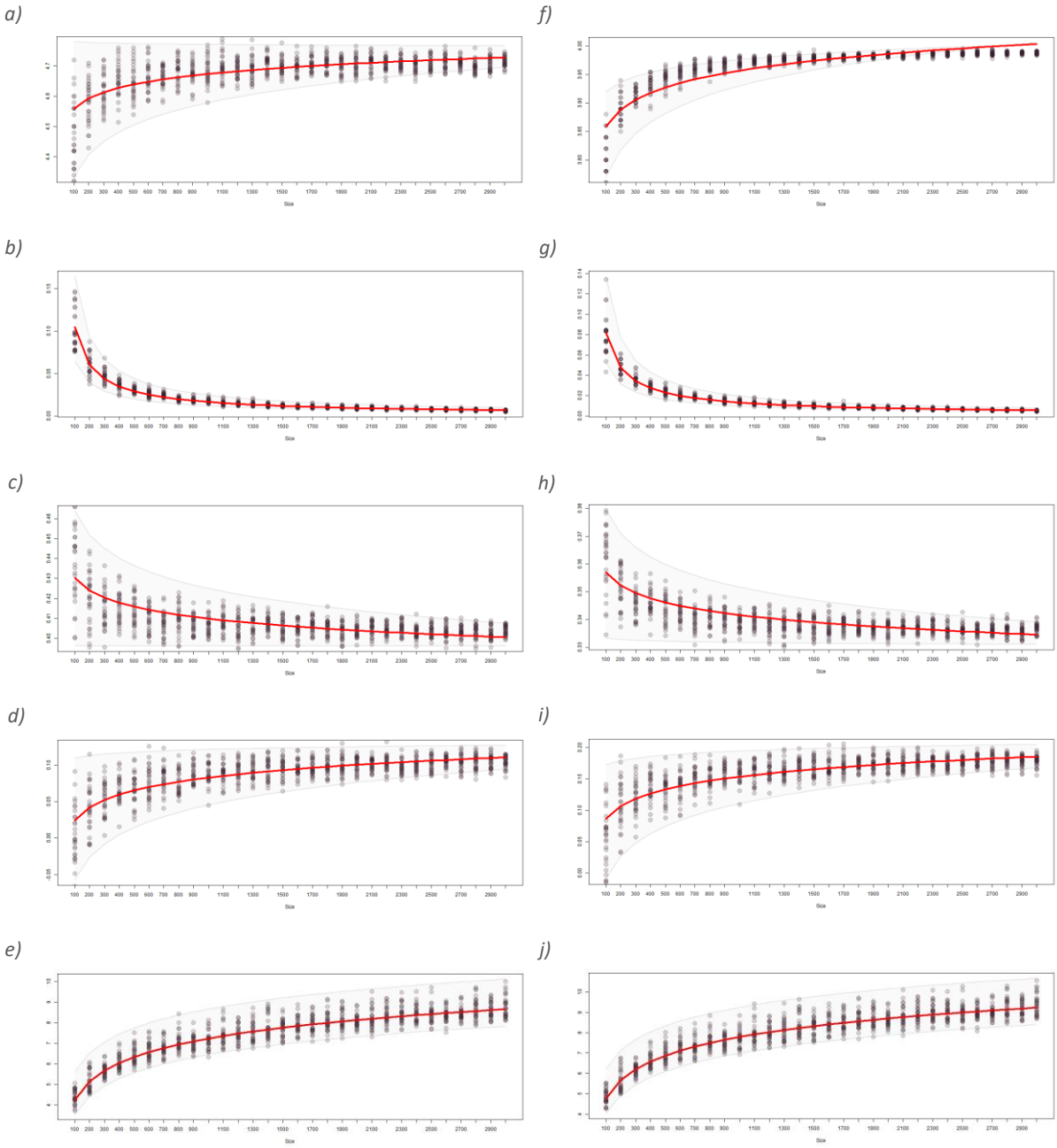


Figure 13: Change in network properties due to length of sample series, fitted to functions

From the top to bottom, visibility graph (VG) on the left side, horizontal visibility graph (HVG) on the right:

- *a) Mean degree, VG*
- *b) Normalized degree centrality, VG*
- *c) Transitivity, VG*
- *d) Assortativity, VG*
- *e) Average geodesic path, VG*
- *f) Mean degree, HVG*
- *g) Normalized degree centrality, HVG*
- *h) Transitivity, HVG*
- *i) Assortativity, HVG*
- *j) Average geodesic path, HVG*

Upper and lower boundaries marked with grey lines, while median is plotted as a red line. The results from our simulations are plotted as dots. There are 100 simulated realization.
The length of the sample series ranging from 100 to 3000 by an interval of 100.

On the basis of the results above we propose a six-part test which can determine if a time-series' data generating process is Gaussian white noise. The hypothesis will be constructed as follows:

$$H_0 : the\ data\ set\ is\ generated\ from\ a\ Gaussian\ white\ noise\ process \quad (26)$$

$$H_1 : the\ data\ set\ is\ not\ generated\ from\ a\ Gaussian\ white\ noise\ process \quad (27)$$

where the criteria to reject the null hypothesis will consist of six sub-tests. All six sub-tests have to fail to reject the null – all the networks statistics needs to be within our confidence bands.

1. $H_0 : p - value > 0{,}05$
   $H_1 : p - value \leq 0{,}05$
2. $H_0 : y_l \leq y_{assortativity} \leq y_u$
   $H_1 : \ y_{assortativity} < y_l \ or \ y_{assortativity} > y_u$
3. $H_0 : y_l \leq y_{average\ geodesic\ path} \leq y_u$
   $H_1 : \ y_{average\ geodesic\ path} < y_l \ or \ y_{average\ geodesic\ path} > y_u$
4. $H_0 : y_l \leq y_{degree\ centrality} \leq y_u$
   $H_1 : \ y_{degree\ centrality} < y_l \ or \ y_{degree\ centrality} > y_u$
5. $H_0 : y_l \leq y_{mean\ degree} \leq y_u$
   $H_1 : y_{mean\ degree} < \ y_l \ or \ y_{mean\ degree} > y_u$
6. $H_0 : y_l \leq y_{transivity} \leq y_u$
   $H_1 : \ y_{transitivity} < y_l \ or \ y_{transitivity} > y_u$

The functions for the different values for upper and lower bound are presented in the table below.

Network statistic	Boundary	Visibility graph	Horizontal visibility graph
Assortativity	Upper	$y_u = 0.0052 * ln(x) + 0.0855$	$y_u = 0.0096ln(x) + 0.1281$
	Lower	$y_l = 0.0445 * ln(x) - 0.262$	$y_l = 0.0521 * ln(x) - 0.2492$
Geodesic path	Upper	$y_u = 1.3265ln(x) - 0.4985$	$y_u = 1.3064 * ln(x) + 0.2167$
	Lower	$y_l = 1.2586ln(x) - 2.2446$	$y_l = 1.2845 * ln(x) - 1.8815$
Degree centrality	Upper	$y_u = 7.1956 * x^{-0.82}$	$y_u = 6.6267 * x^{-0.842}$
	Lower	$y_l = 1.8246 * x^{-0.725}$	$y_l = 1.452 * x^{-0.723}$
Mean degree	Upper	$y_u = -0.004 * ln(x) + 4.7987$	$y_u = 0.0245ln * (x) + 3.8074$
	Lower	$y_l = 0.1064 * ln(x) + 3.8437$	$y_l = 0.0697ln * (x) + 3.4486$
Transitivity	Upper	$y_u = 0.5527 * x^{-0.038}$	$y_u = 0.4418 * x^{-0.033}$
	Lower	$y_l = 0.3961 * x^{-6E-04}$	$y_l = 0.3364 * x^{-0.002}$

Table 5: Test for Gaussian white noise - upper and lower boundaries

The test is performed on a time-series by first transforming it to either a visibility- or horizontal visibility graphs. Then the mentioned statistics must be calculated. The boundaries for each statistic are found by using the length of the time-series. Then the statistics can be compared to the boundaries. If the statistics fits within all of the boundaries – we fail to reject the null and the time-series generating process is a Gaussian white noise.

5.3 Variation in network properties due to change in autocorrelation coefficient

A change in autocorrelation coefficient(s) of an $AR(1)$, $MA(1)$ or $ARMA(1,1)$ implies different statistical properties of the sample series. We demonstrated this in our description of stochastic processes, and it can be further examined in the appendix. We will study how the value of the (true but unknown) parameter of the stochastic process would be reflected in the properties of the associated network.

In the exploration on how a stochastic processes' autocorrelation coefficient influences the visibility- and the horizontal visibility graph, we examined changes in the visual representations of the time-series as a network as well as the network statistics. This was achieved as mentioned earlier with a data run and an example run, both runs had a sample size of 1000. The value of the parameter is varied from -0.9 to 0.9 by an interval of 0.1. In the examination of the $ARMA(1,1)$ process, the autocorrelation coefficient of the moving average process was fixed at 0.1, and the autocorrelation coefficient of the autoregressive process was varied as described above.

The data run had 100 simulations for each parameter, and the example run used a set seed and a forced layout on all network realizations. White noise is included in both runs as a benchmark. When the autocorrelation coefficients of an $AR(1)$ and $MA(1)$ are equal to zero their values should coincide with the results from a white noise process. This will not be the case with the $ARMA(1,1)$ because of the effects from the fixed autocorrelation coefficient of the $MA(1)$.

$\boldsymbol{MA(1)}$

The visual network representations of time-series generated from a moving average process of order one is illustrated below, with visibility graph to the left and horizontal visibility graph to the right. We have included the parameters -0.9, 0.1 and 0.9, top to bottom, to present the extremes, but network representations of all parameters are available in the appendix.

Again, it is difficult to find the differences between the visibility- and the horizontal visibility graphs, but the differences are always present. What is quite noticeable however is that there are many differences observed in the graphs generated by processes of different autocorrelation coefficients. The edges do clearly not form the same pattern, and the number of visible edges seems to increase with the value of the parameter. When we inspect the line plot of the same series presented in an earlier section (also available in the appendix) we do detect subtle differences in the series which are reflected in the generated networks.

Neither of the networks share edge-pattern with the network created from a white noise process, illustrated in the previous section. But when we take a closer look at the network generated from an $MA(1)$ process with a parameter of zero in the appendix, we do find the similarity which we would expect in the edge-patterns. This similarity arises purely because we use the same set of errors in all our example simulations and should not be considered as a common rule.

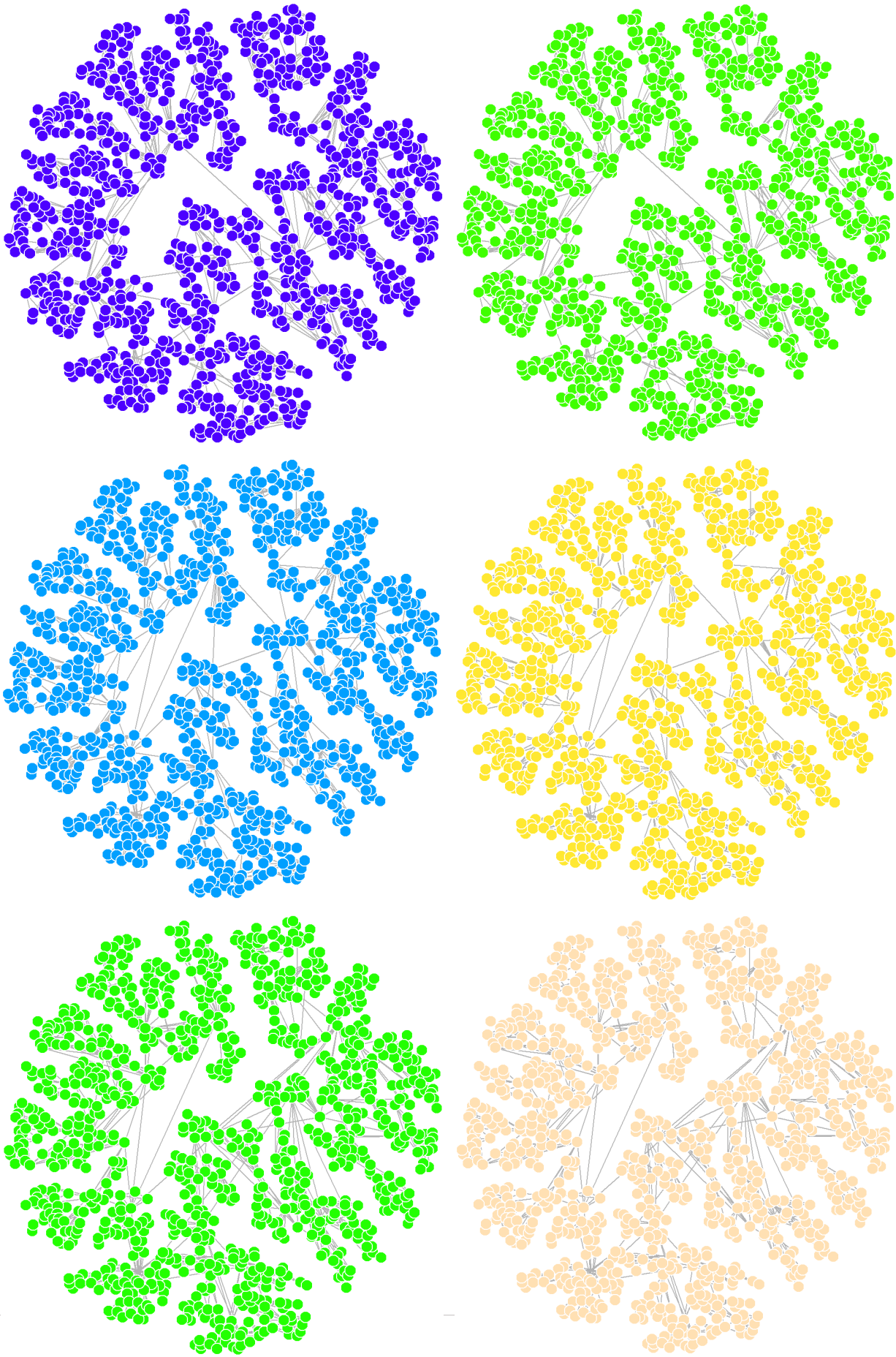


Figure 14: Selected network from example realization of $MA(1)$

Figures are placed on the previous page, from top to bottom:

a) Visibility graph – -0.9
b) Visibility graph – 0.1
c) Visibility graph – 0.9
d) Horizontal visibility graph – -0.9
e) Horizontal visibility graph – 0.1
f) Horizontal visibility graph – 0.9

The networks are color-coded: the network and the box in the boxplot representing the same process and parameter share a color. The numbers relate to the process's autocorrelation coefficient

The outcomes from our explorations with multiple simulations are presented below. White noise is used as a benchmark and its result is placed to the left of the plot and colored grey. The plot then continues with results from an $MA(1)$. Starting with an autocorrelation coefficient of -0.9 followed by increasingly larger parameters until a parameter of 0.9.

We can see that the different parameters do have an effect on the network statistics. In this case, in opposite of the study by sample size in the previous section, the results from the visibility graph are often notably different from the results from its subgraph. Another initial observation consists of how similar the results from a white noise and an $MA(1)$ with the autocorrelation coefficient of zero are. Most of the outputs only tend to differentiate from white noise when the autocorrelation coefficient is large and positive.

Both visibility- and horizontal visibility graphs are inclined to have a low normalized degree centralization. The visibility graphs' values are similar to white noise as long as the autocorrelation coefficients are negative and then decrease as the value of the parameters increase. While in the horizontal visibility graphs values are larger than white noise for negative parameters and smaller for positive. The values for the visibility graph are higher than the ones for the horizontal visibility graph as we expected. When we take a closer look at the example network above, we do see that there are fewer edges across the center when the autocorrelation coefficients are high, indicating less degree centrality. Low degree centrality indicates few extreme values in the series, and we can confirm that this is the case by examining the example line chart presented previously, which are also available in the appendix. An $MA(1)$ process with a high negative autocorrelation coefficient have more extreme values than this process with a high positive parameter. The extreme values are easier to detect by the horizontal visibility algorithm, which is why degree centrality is higher than white noise for negative parameters in this case. This is not reflected in the results from the visibility graph.

The opposite is the case of assortativity, where the horizontal visibility graphs' values are similar to white noise for negative autocorrelation coefficients and then increase as the parameter increases. The visibility graph values are u shaped with values higher than white noise at either end of the parameters scale. All results are positive thus indicating assortative mixing. The values of horizontal visibility graph are higher than the ones for the visibility graph as expected. There is a 22% chance that nodes in the horizontal visibility graph generated from a process with very high positive autocorrelation coefficient are connected with nodes of the same degree. This indicates that a large number of nodes share a similar degree, which suggest that there is

something like a repeating pattern in this series. We cannot however confirm this by examining the line plot or the network realizations.

The p-values are all significant, even though the values are declining with larger parameters they are still above the 5% significance level. Thus, every tested autocorrelation coefficient result in a network which is power laws.

The results regarding transitivity take a form quite different from the rest with an s-shape both in visibility- and horizontal visibility graphs. The outcomes from the negative autocorrelation coefficients are closer to the white noise than the positive in both cases. By examining the horizontal visibility graph for large positive parameters, there is a 37.5% chance that two realizations in the original series are separated by one, or more, realizations of lower value. This is an increase of ca 5% from the values for a large negative autocorrelation coefficient and are supported by the associated line plots. The values from the visibility graph are higher than ones from the horizontal visibility graph which is expected due to the differences in their algorithms.

The exploration of mean degree resulted in a very different pattern in the results from the visibility- and horizontal visibility graphs. The visibility graph has an s-shape while the horizontal visibility graph is displaying results very similar to white noise. This result is expected due to the differences in their algorithms. As we expected all the values from the horizontal visibility graph are lower than the minimum of the results from the visibility graph. If we examine the different line plots, we discover that they differ mainly the frequency of which the values change and the presence of extreme values. This will not affect the results from the horizontal visibility graph since the algorithm do not allow for angles and all the frequencies are rather high. We suggest that mean degree from a horizontal visibility graph created from a series with high frequency will have similar results as white noise. As the frequency decrease, the value of mean degree will also be reduced. The frequency affects the results from the visibility graph differently because of its algorithm. A high frequency series will result in more visibility lines crossing each other, which is denying visibility, and thus the resulting mean degree will be lower than white noise. A lower frequency will allow for more visibility thus the mean degree will be higher than white noise.

The average geodesic paths are both increasing with the value of the autocorrelation coefficient, which may seem like an error, especially in the case of the visibility graph, where the mean degree also is increasing with the parameter. This however is also connected to the degree centrality, which is declining with the parameter. The increase in mean degree cannot compensate for the lack of centrality, thus the result in both graphs are expected.

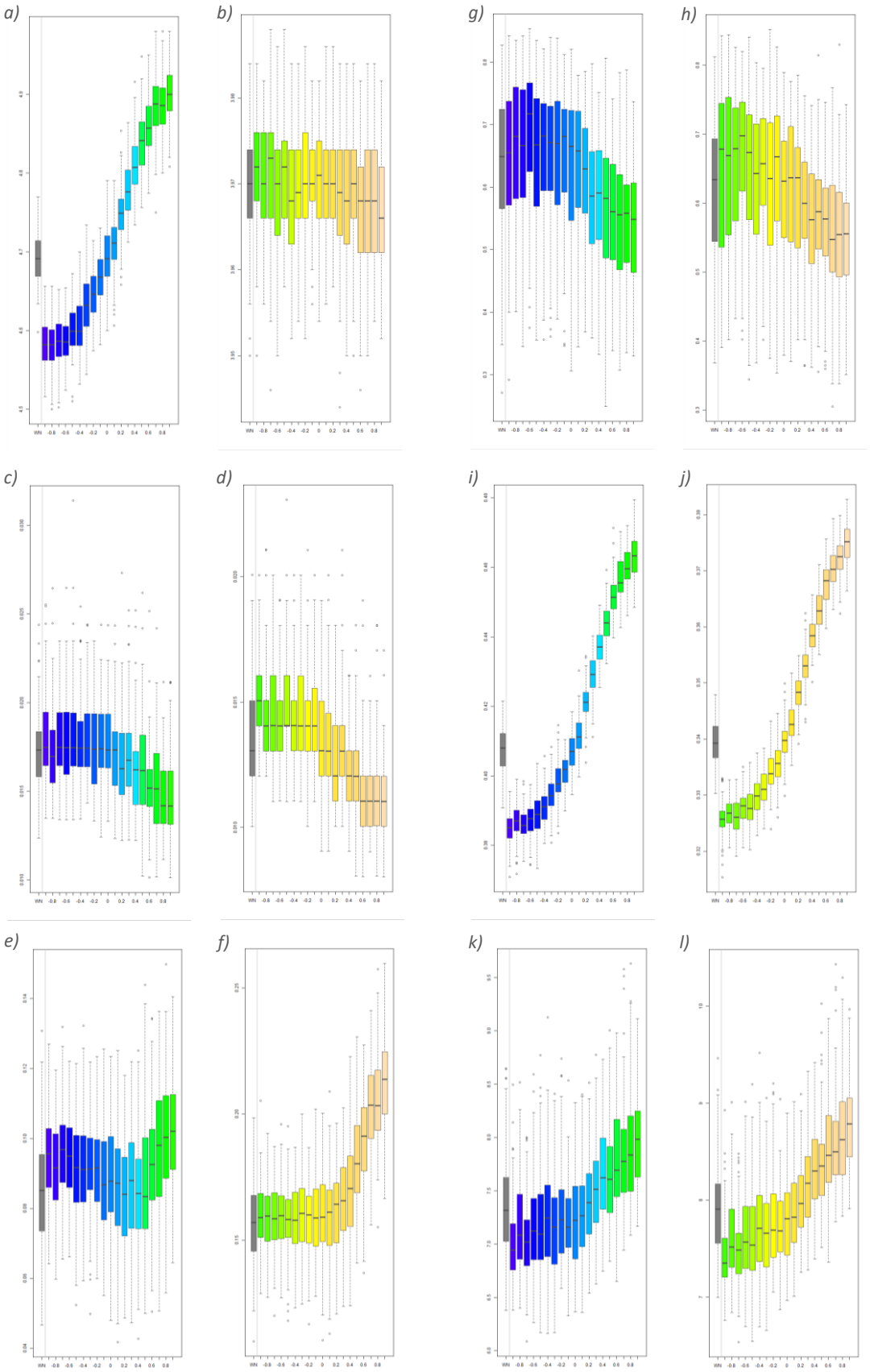

a)
b)
g)
h)
c)
d)
i)
j)
e)
f)
k)
l)

Figure 15: Change in network properties due to autocorrelation coefficient, $MA(1)$

The figures are placed on the previous page: from the top to bottom, visibility graph on the left side, horizontal visibility graph on the right:

a) Mean degree, VG
b) Mean degree, HVG
c) Normalized degree centrality, VG
d) Normalized degree centrality, HVG
e) Assortativity, VG
f) Assortativity, HVG
g) P-value, H_0: cumulative degree distribution follows power law, VG
h) P-value, H_0: cumulative degree distribution follows power law, HVG
i) Transitivity, VG
j) Transitivity, HVG
k) Average geodesic path, VG
l) Average geodesic path, HVG

There are 100 simulation for each process with different autocorrelation coefficient, ranging from -0.9 to 0.9 by an interval of 0.1.
The sample size of the series is 1000. The first box plot in the box plot (grey) is the benchmark value of white noise

$\boldsymbol{AR(1)}$

As with the moving average process we have included chosen networks with the autocorrelation coefficients of $-0.9, 0.1$ and 0.9 below. The full presentations of these can also be found in the appendix. Because the edge pattern is very different from white noise, it may be easier to find differences between the visibility- and the horizontal visibility graphs in this display, but they are still subtle.

The differences between the networks generated by processes with different parameters however are anything but subtle. In both the extreme autocorrelation coefficients there are no edges crossing the center of the network, but the number of visible edges increases quite a bit in both. The line plot of the same series is vastly different which is reflected in the large variance in edge-pattern. By comparing the $MA(1)$ networks with the $AR(1)$s, we realize that if both have an autocorrelation coefficient of $0,1$, their edge-patterns will be very similar. As in the case with the $MA(1)$, the $AR(1)$ with an autocorrelation coefficient of zero (located in the appendix) has an edge-pattern very similar to white noise.

The outcome of our multiple simulation exploration is presented at page 51, using the same layout as in $MA(1)$, and it is apparent that the value of autocorrelation coefficients changes the network statistics. In addition, we can see that the changes are different from the ones in the case of $MA(1)$. This is of course expected since they are totally different data generating processes, which is apparent the line plots, especially with extreme values of the autocorrelation coefficient.

The shapes of the visibility- and the horizontal visibility graphs are often similar, with either a concave or a convex shape where the zero parameter marks either maximum or minimum. If the network is constructed from an $AR(1)$ with an autocorrelation coefficient of zero, the value of all of the statistics will be similar to the benchmark white noise in both graphs.

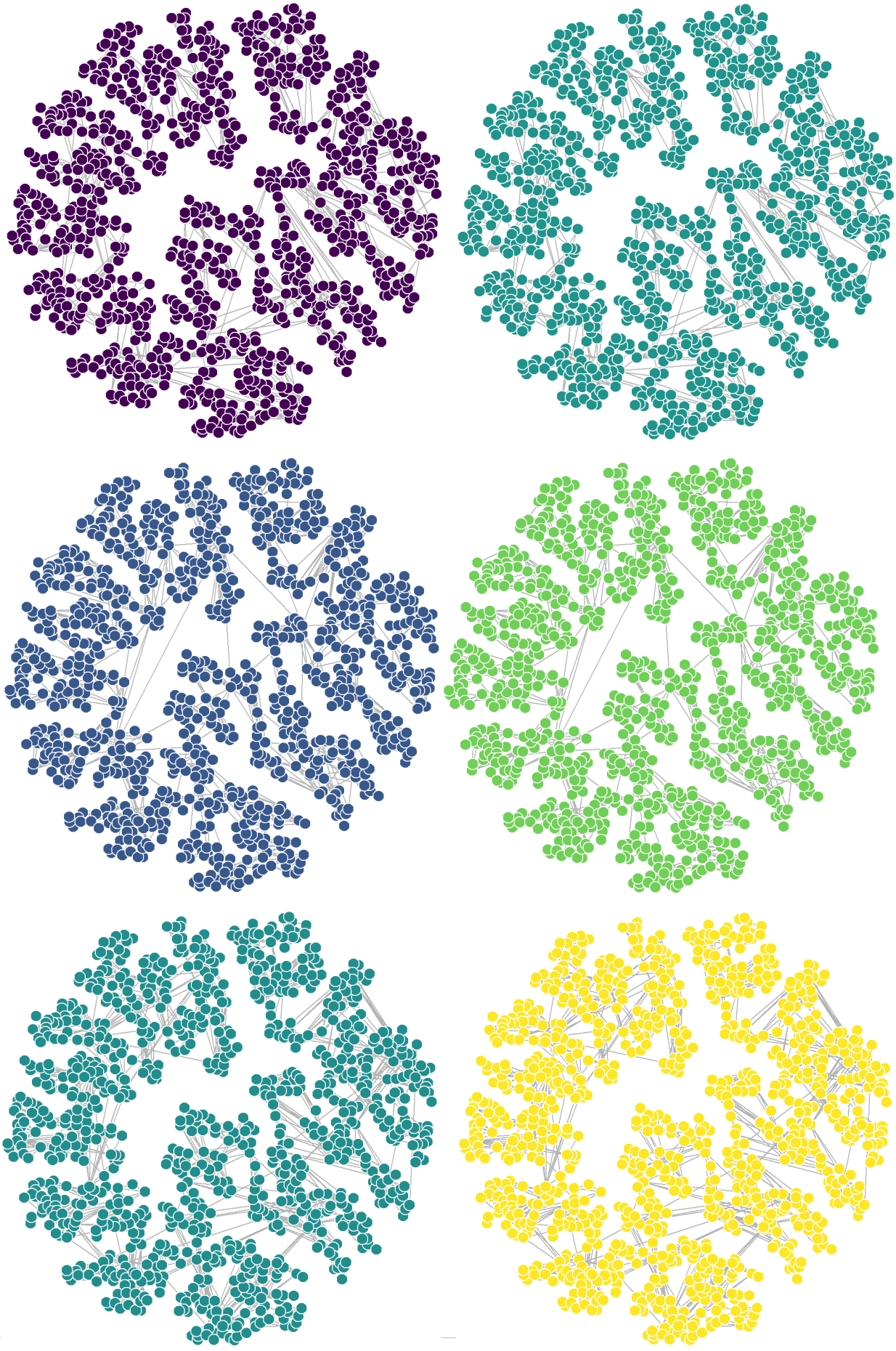


Figure 16: Selected network from example realization of $AR(1)$

Figures are placed on the previous page, from top to bottom:

a) Visibility graph – -0.9
b) Visibility graph – 0.1
c) Visibility graph – 0.9
d) Horizontal visibility graph – -0.9
e) Horizontal visibility graph – 0.1
f) Horizontal visibility graph – 0.9

The numbers relate to the process's autocorrelation coefficientThe networks are color-coded: the network and the box in the boxplot representing the same process and parameter share a color.

Visibility- and horizontal visibility graphs both have low normalized degree centralization with a maximum value equal to white noise. This coincides with our discoveries when we examined the networks commenting on the lack of edges through its center. The horizontal visibility graphs values are lower than the values from the visibility graphs which is expected as it has fewer edges. In addition, we observe that the large positive autocorrelation coefficient effects the horizontal visibility graph more than the visibility graph. The extreme values of the autocorrelation coefficient in an $AR(1)$ generate vastly different series. Where the large negative values have a high frequency with distinct sections of different amplitude, while the large positive values tend to wander similarly to a random walk. Both of these are resulting in fewer extreme values as reflected in the values of the degree centralization.

Both associativity and the p-values share a concave form with degree centrality. The p-values are significant for both graphs and indicating power laws for all autocorrelation coefficients. The results regarding assortativity are very close to zero in both cases, where the extreme high parameters actually indicate disassortative mixing. The values of the horizontal visibility graph are as expected due to its algorithm, larger than the visibility graphs values. Low assortativity suggest that there are no cycles in the series, which is confirmed by examining the line chars.

The visibility graphs' transitivity results have a similar shape as the results from our $MA(1)$ experiment, but the s-shape is less distinct in this case. But the horizontal visibility graph has a different shape. The changing amplitude with high negative autocorrelation coefficients results in a higher transitivity for the horizontal visibility than for a high frequency series with constant amplitude. The reason is because it is more likely that two realizations are separated by one, or more realizations, of a lower value in the case with different amplitudes. The probability of having triples is even higher when the autocorrelation coefficient of the series is high. This series may be described as "hills and valleys" which is the perfect environment for generating triples by the horizontal visibility algorithm.

When we examine the results of mean degree we realize that the shape of both boxplots are different from the results for $MA(1)$. The visibility graph has an increasing number of degrees as the autocorrelation coefficient increases, while the horizontal visibility graph has a concave shape where the effect of a high parameter is larger than the effect of a lower one. The change in amplitude in the lower value parameter, and the reduction in frequency in the higher amplitudes, both impacts the results as we discussed in the previous section. As in the results from the $MA(1)$, the result of average geodesic path is a direct consequence of the degree centrality, which here has a convex shape. The relationship is negatively correlated; thus, the convex shape of the average geodesic paths is as expected.

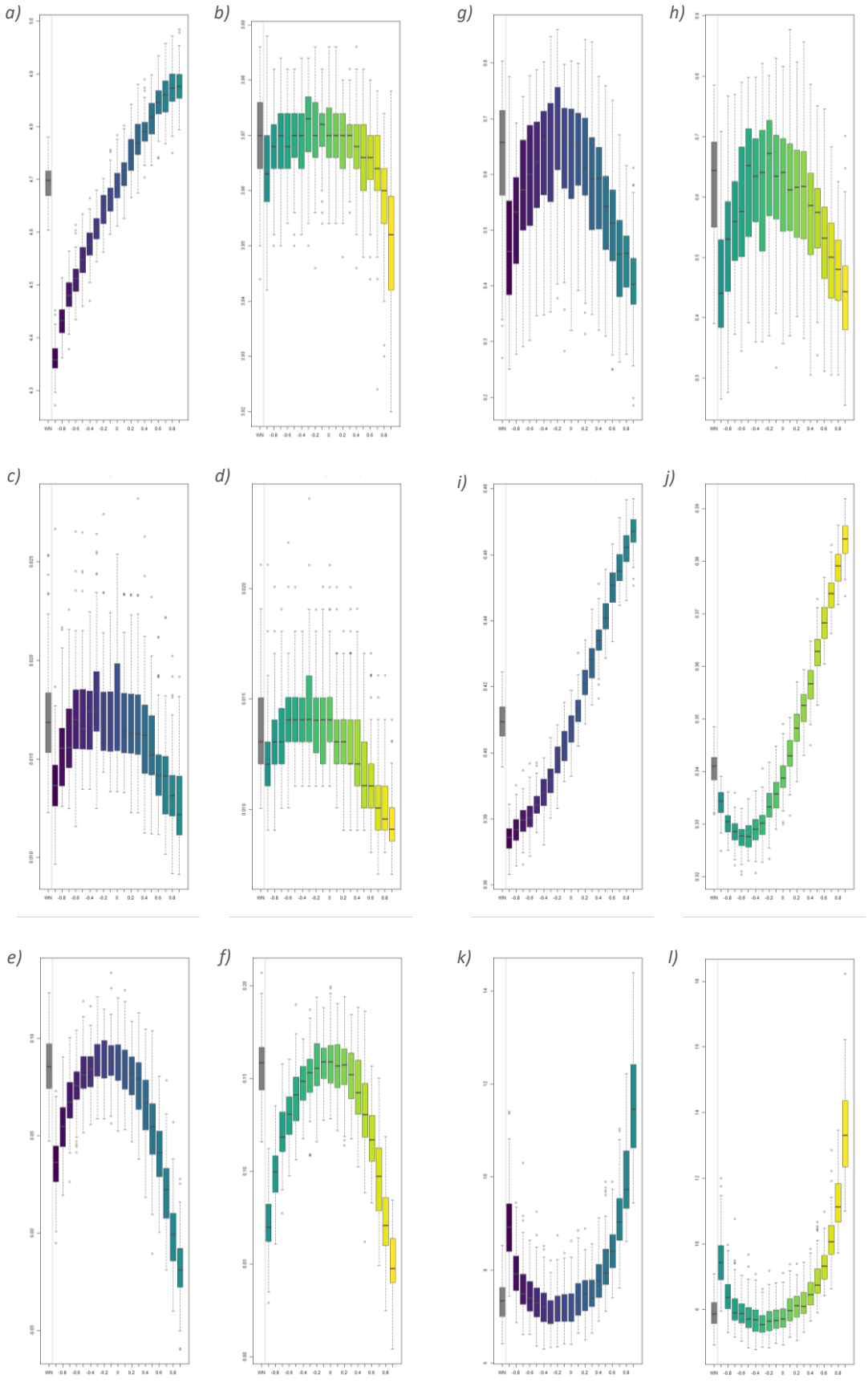

a)
b)
g)
h)
c)
d)
i)
j)
e)
f)
k)
l)

Figure 17: Change in network properties due to autocorrelation coefficient, $AR(1)$

The figure is placed on the previous page: from the top to bottom, visibility graph on the left side, horizontal visibility graph on the right:

a) Mean degree, VG
b) Mean degree, HVG
c) Normalized degree centrality, VG
d) Normalized degree centrality, HVG
e) Assortativity, VG
f) Assortativity, HVG
g) P-value, H_0: cumulative degree distribution follows power law, VG
h) P-value, H_0: cumulative degree distribution follows power law, HVG
i) Transitivity, VG
j) Transitivity, HVG
k) Average geodesic path, VG
l) Average geodesic path, HVG

There are 100 simulation for each autocorrelation coefficient, ranging from -0.9 to 0.9 by an interval of 0.1. The sample size of the series is 1000. The first box plot in the box plot (grey) is the benchmark value of white noise

$\boldsymbol{ARMA(1,1)}$

To explore the changes that the autocorrelation coefficients of both $AR(1)$ and $MA(1)$ implies on the network statistics together, we added $ARMA(1,1)$ to our experiments. We chose to vary the autocorrelation coefficient of $AR(1)$ and keep the $MA(1)$s autocorrelation coefficient fixed at 0.1. The resulting boxplots have, in addition to the box representing white noise, another grey box which represents the values from the $MA(1)$ with an autocorrelation coefficient of 0.1.

When we compare the network realizations from the $ARMA(1,1)$ we immediately realize that they share a lot of the edge patterns from $AR(1)$. One notable difference is that the edge straight across the center of the graph of $AR(1)$ with an autocorrelation coefficient of 0.1 not is present in the case of $ARMA(1,1)$ with the same parameter.

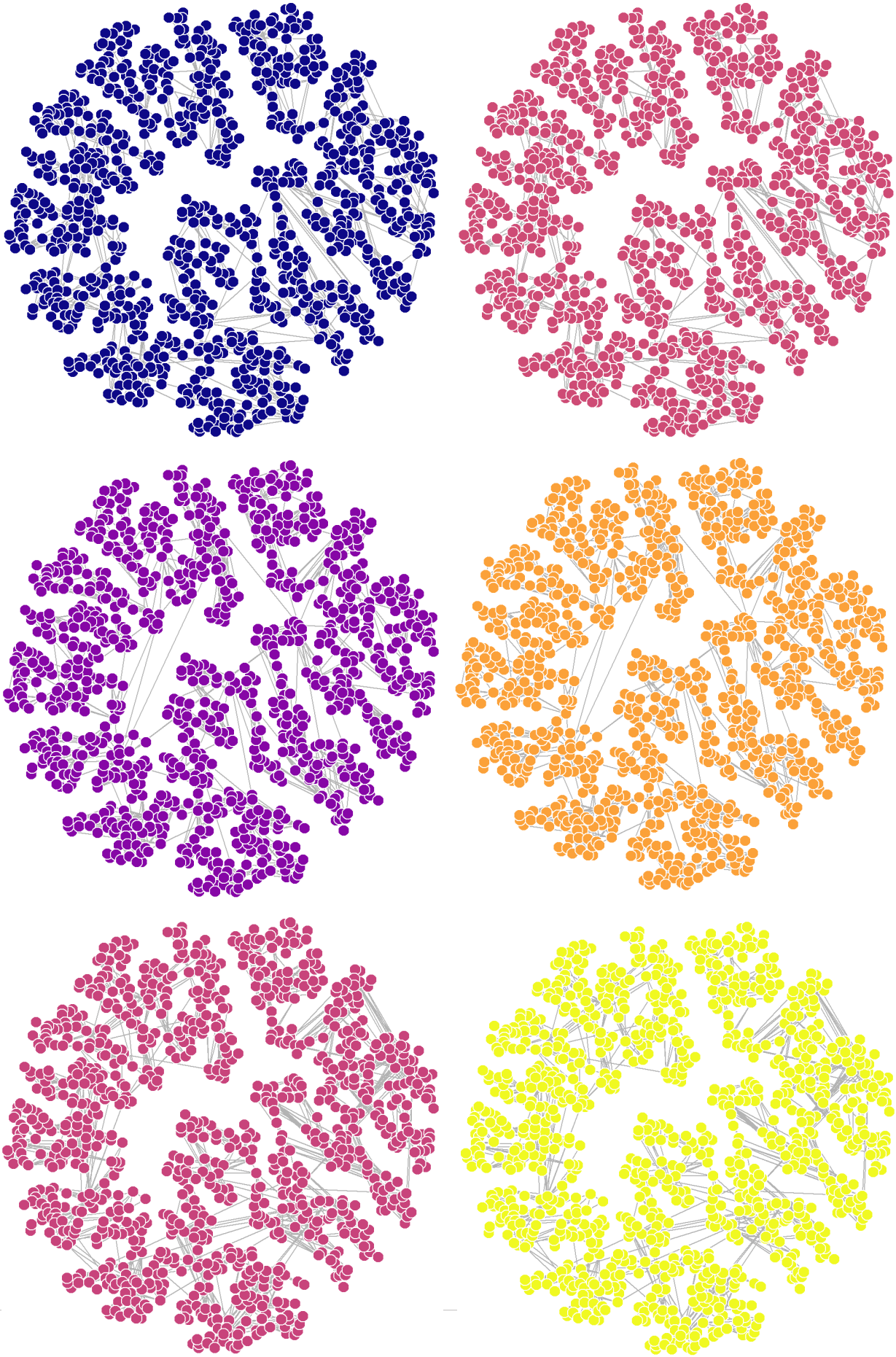
From the experiment with multiple simulations the similarities with the $AR(1)$ continues, outcome presented at page 54. The only observable change made by $MA(1)$ is that it increases the effect that the large positive parameters have on the normalized degree centrality. We did observe that one of the edges in the center of both networks was indeed missing for $ARMA(1,1)$ realizations and a decrease in degree centralization was therefore expected. As a consequence of this, the results from average geodesic path are also different for $AR(1)$ with an even steeper increase of length with large positive autocorrelation coefficient.

Figure 18: Selected network from example realization of $ARMA(1,1)$

Figures are placed on the previous page, from top to bottom:

a) Visibility graph – -0.9
b) Visibility graph – 0.1
c) Visibility graph – 0.9
d) Horizontal visibility graph – -0.9
e) Horizontal visibility graph – 0.1
f) Horizontal visibility graph – 0.9

The numbers relate to the process's autocorrelation coefficient
The networks are color-coded: the network and the box in the boxplot representing the same process and parameter share a color.



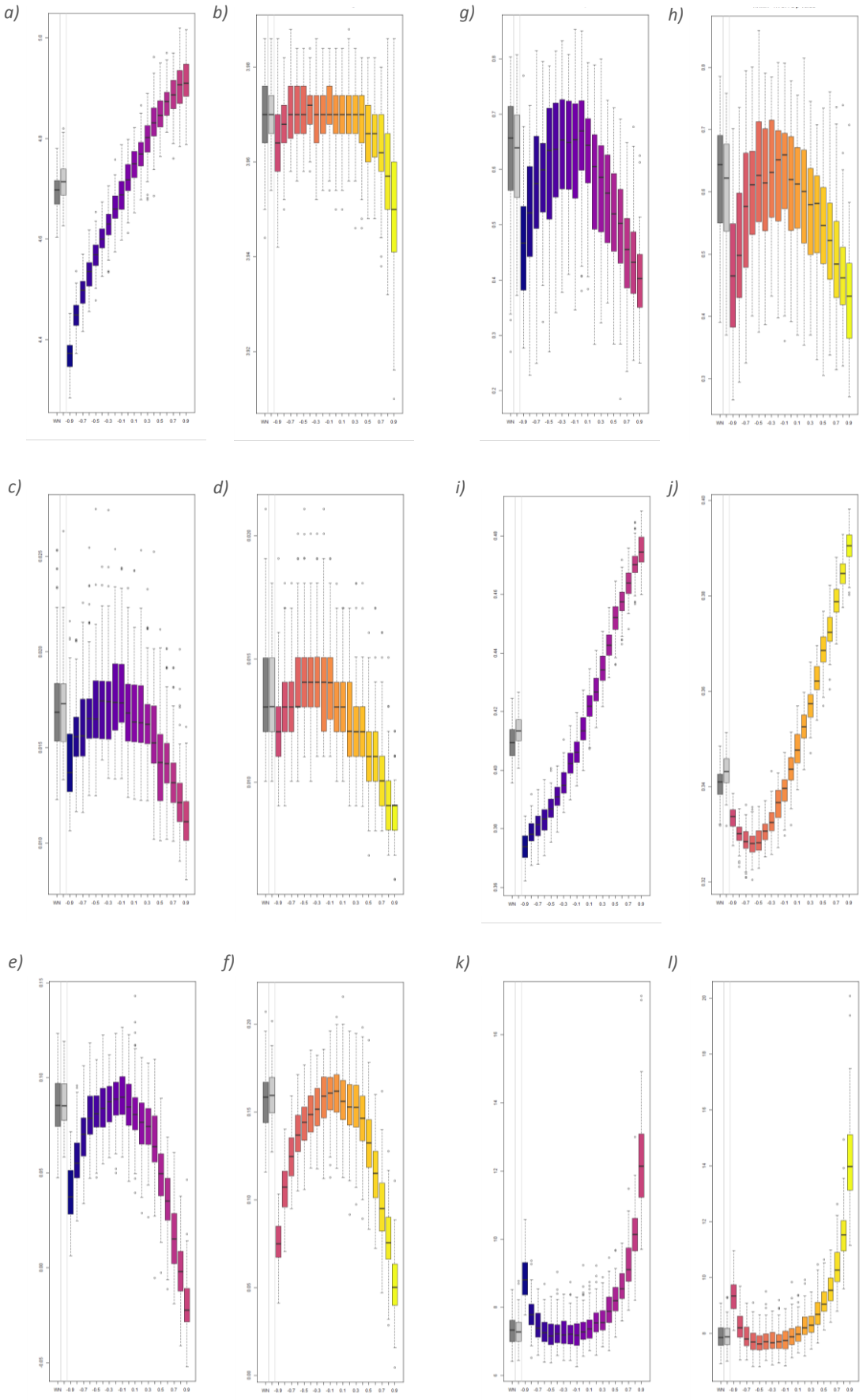

a)
b)
g)
h)
c)
d)
i)
j)
e)
f)
k)
l)

Figure 19: Change in network properties due to autocorrelation coefficient, $ARMA(1)$

The figure is placed on the previous page: from the top to bottom, visibility graph on the left side, horizontal visibility graph on the right:

m) Mean degree, VG
n) Mean degree, HVG
o) Normalized degree centrality, VG
p) Normalized degree centrality, HVG
q) Assortativity, VG
r) Assortativity, HVG
s) P-value, H_0: cumulative degree distribution follows power law, VG
t) P-value, H_0: cumulative degree distribution follows power law, HVG
u) Transitivity, VG
v) Transitivity, HVG
w) Average geodesic path, VG
x) Average geodesic path, HVG

There are 100 simulation for each parameter, ranging from -0.9 to 0.9 by an interval of 0.1.
The sample size of the series is 1000. The first box plot in the box plot (grey) is the benchmark value of white noise

5.4 Estimation of parameters on the basis of network statistics

Some of our results from the visibility graphs presented above have properties that can be used to distinguish between time-series parameters without any need to determine stationarity status. Both measures of mean degree and transitivity generates unique values for each parameter. Therefore, it is possible to compare these results to the statistics for any time-series generated from an $MA(1)$, $AR(1)$ or $ARMA(1,1)$ and identify its parameter. This is only the cases for visibility graph though, as the values in its subgraph do not share this property.

As in the previous case we used the upper and lower quartile to create the upper and lower band of our confidence interval. We chose in this case to use some of the self-starting functions in R because of the special properties our result exhibited.

The S-shape in the transitivity results was fitted to the Boltzmann model.

$$y = c + \frac{d-c}{1+e^{b*(x-e)}} \tag{28}$$

Which is described as an asymptotic five-parameter logistic model. The resulting parameter for each of the boundaries is presented below, as is the display of the fitted functions with our realizations.

	MA(1)		AR(1)		ARMA(1,1)	
Boltz.	*Upper*	*Lower*	*Upper*	*Lower*	*Upper*	*Lower*
b	-3.464	-3.710	2.506	-2.421	-2.412	-2.337
c	0.396	0.372	0.373	0.358	0.378	0.355
d	0.489	0.458	0.484	0.472	0.504	0.479
e	0.297	0.295	0.032	0.302	0.171	0.211

Table 6: Parameters used in the classification of autocorrelation coefficient – transitivity

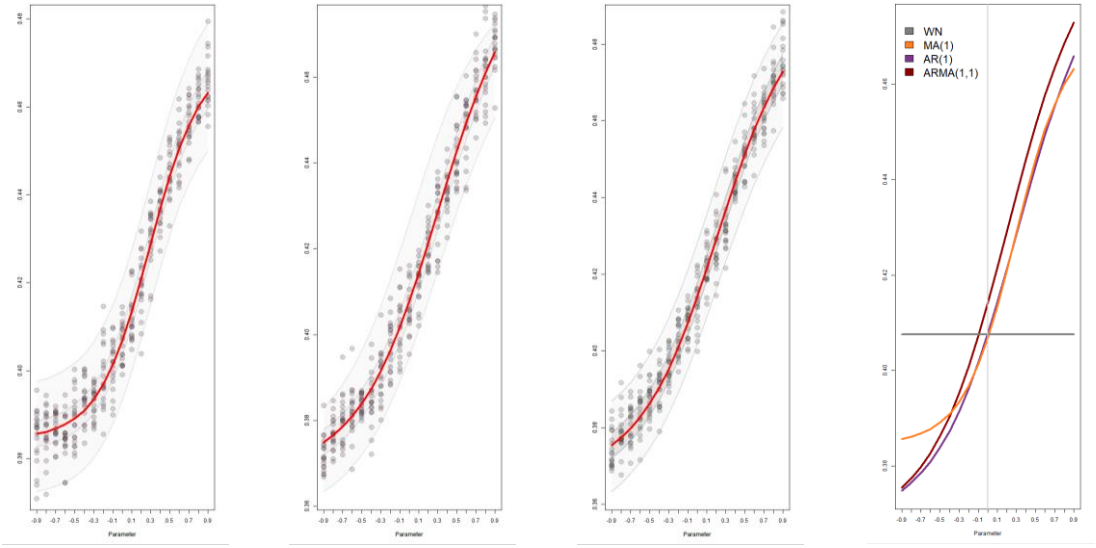


Figure 20: Relationship between transitivity and autocorrelation coefficient fitted to functions

From left to right: $MA(1)$, $AR(1)$, $ARMA(1,1)$ and a comparison with the reference value of white noise
The value of white noise is equal to the median of white noise for sample size 1000 from the previous experiment with sample size

Mean degree result had two distinctive shapes. The S-shape was fitted to the Boltzmann model (28) the concave shapes was fitted to the Asymptotic regression model:

$$y = Asym + (R0 - Asym) * e^{(-e^{lrc} * x)} \quad (29)$$

The resulting parameter for each of the boundaries is presented below, as is the display of the fitted functions with our realizations.

	MA			AR(1,1)		ARMA(1,1)	
Boltz	*Upper*	*Lower*	*Asymp.*	*Upper*	*Lower*	*Upper*	*Lower*
b	-3.736	-4.153	Asym	5.374	4.970	5.377	5.002
c	4.630	4.504	R0	4.785	4.614	4.814	4.629
d	4.988	4.815	lrc	-0.745	-0.330	-0.642	-0.380
e	0.045	0.182					

Table 7: Parameters used in the classification of autocorrelation coefficient - mean degree

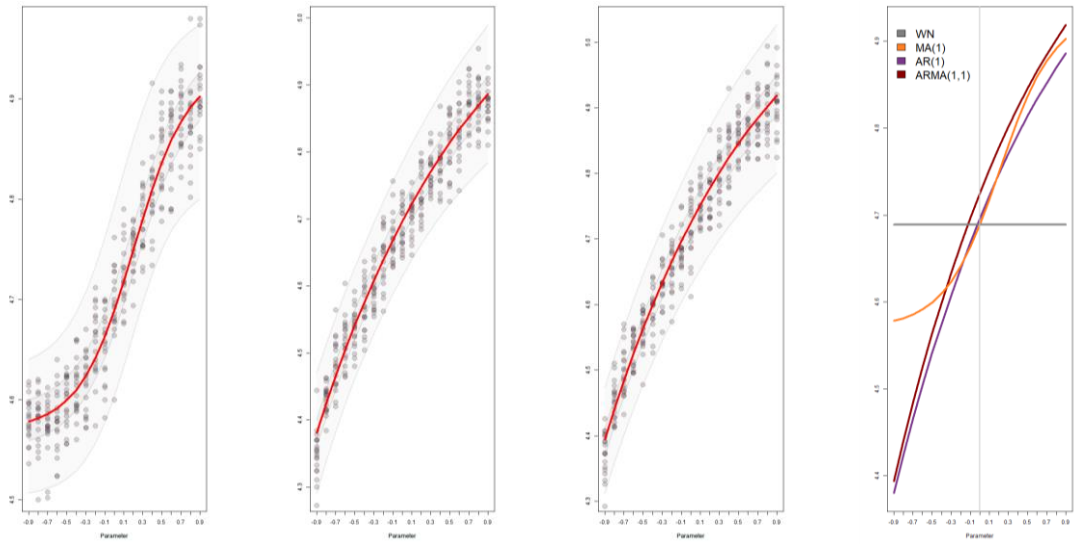


Figure 21: Relationship between mean degree and autocorrelation coefficient fitted to functions

From left to right: $MA(1), AR(1), ARMA(1,1)$ and a comparison with the reference value of white noise.
The value of white noise is equal to the median of white noise for sample size 1000 from the previous experiment with sample size

We were tempted to suggest the use of this results in a test similar to our six-part white noise test, as a three-part test where the parameters was p-value, transitivity and mean degree. The principle would be the same, the tested series values have to lay within the upper and lower bands to be identified as either an $AR(1)$, $MA(1)$ or $ARMA(1,1)$ with a determined parameter. When we realized how close these values are of each other however, we knew that they were in practice inseparable by the means of these results alone. We propose therefore this as a classification tool where we can determine the parameter of any $MA(1), AR(1)$ or $ARMA(1,1)$, but it cannot distinguish between these.

6 DISCUSSION

During our exploration we made two significant discoveries. First of all, we revealed a significant relationship between white noises' network statistics and the length of sample size. Some of these relationships were already established as previously mentioned by Luque et al. (2009). They stated that a horizontal visibility graph will have a mean degree of four for large sample sizes which is reflected in our results as well. The degree distribution in our results were always power laws, consistent with the finding of Luque et al. Another result which was consistent with Luque's findings was the relationship between the average shortest path length and logarithm of sample size for graphs generated from the horizontal visibility algorithm. This relationship was extended to the visibility graph by Sun et al. (2016)

We discovered however that every relationship is valid for both visibility- and horizontal visibility graph. This was revealed by the simulations for white noise of different sample sized up to a sample size of 3000. Our results were used to fit functions which could in theory predict the further development with larger sample sizes. As mentioned above we were unable to fit functions to some of our boundaries. The values were to erratic and we needed to run the simulations multiple times to gain a better understanding on how these behave. We also noticed problems with the functions that we have fitted, even those with a very large goodness of fit measurements. When we extend the sample size, the boundaries do not behave as we would expect, some of them even change from upper to lower bounds. This indicates that our functions do not capture the true nature of our data and the topic should be revisited. Despite of these we could purpose a test from which a white noise series could be identified for series with a sample size up to 3000. This test could be applied for both stationary and non-stationary time-series.

Secondly, we documented the change in network statistics due to alterations in the auto correlation functions of $AR(1), MA(1)$ and $ARMA(1)$. In this case though, we could not distinguish between those three processes from the output, the results were to similar. What we did realize was that we could use the relationships between the parameters of the studied processes and two of the network statistics, mean degree and transitivity, to determine the parameters of the mentioned processes. Thus, creating a tool for identifying the parameter of an $AR(1), MA(1)$ or $ARMA(1,1)$ which is independent of the original time-series stationarity status.

A similar result as ours was discovered by Segberg and Skoglund (2017) in their exploration of $GARCH$. They were able to distinguish between $GARCH$ parameters on the basis of average shortest path length and mean degree. These relationships only hold for the networks generated by the visibility graph algorithm. Zhang et al. (2017) are as far as we know the only ones who have explored the network statistics' behavior of the auto regressive processes of orders one and two. They however used a different approach and considered the effect from re-sampling time delay. Their findings are therefore not comparable to ours due to our use of a fixed sample size

Our results were obtained by the use of a fixed sample size of 1000 and we cannot speculate in how these relationships would change due to changes in the length of the time-series.

All of our data have been artificially generated in R, which may not be completely reliable to generate the desired series for our investigations. We did not however have any surprising results which may have been caused by errors in the generation of data. Since we are novices in the field of network theory, we may have missed such realizations. Every simulation has been performed with 100 repetitions of each variable, and this might not be enough to gain valid results. As we can see in our different boxplots, the values of white noise and a process with zero auto correlation coefficient are close, but not identical. This difference is insignificant though, taking the samples nature into consideration.

Although there are plenty of issues, but we believe that our findings are a contribution towards the identification of stochastic processes through networks.

We encourage the further development and testing of our work. Our discoveries should be tested against real data to enhance their validity before further use. We also suggest investigating the relationship between sample size and network properties for the auto regressive and moving average processes in addition to their combination, of different auto correlation coefficients.

7 CONCLUSION

We have discovered a test which can determine a white noise process in addition to a classification tool which can determine the parameter of moving average-, autoregressive- and autoregressive moving average processes. Neither of them is dependent on the series stationarity. In addition, we gained insight into how a networks statistic rely back to its original time-series, thus aiding the interpretation of different network properties when used with time-series.

We realize that the use of our findings may be limited, they are un-tested and have specific sample size requirements, but we do look at this as a first step towards recording stochastic series. We encourage the continuation of our work by testing our purposed tool and explore the sample size effect further both in white noise but also in the other processes examined in this thesis.

REFERENCES

Barabási, A.-L., & Albert, R. (1999). Emergence of Scaling in Random Networks. *Science, 286*(5439), 509-512. Retrieved from https://science.sciencemag.org/content/sci/286/5439/509.full.pdf. doi:10.1126/science.286.5439.509

Barabási, A.-L., & Bonabeau, E. (2003). Scale-Free Networks. *Scientific American, 288*(5), 60-69. Retrieved from http://www.jstor.org/stable/26060284.

Bezsudnov, I. V., & Snarskii, A. A. (2014). From the time series to the complex networks: The parametric natural visibility graph. *Physica A: Statistical Mechanics and its Applications, 414*, 53-60. Retrieved from http://www.sciencedirect.com/science/article/pii/S0378437114005676. doi:https://doi.org/10.1016/j.physa.2014.07.002

Brooks, C. (2008). *Introductory Econometrics for Finance*. The United States of America, New York: Cambridge University Press.

Campanharo, A. S. L. O., Sirer, M. I., Malmgren, R. D., Ramos, F. M., & Amaral, L. A. N. (2011). Duality between Time Series and Networks. *PLOS ONE, 6*(8), e23378. Retrieved from https://doi.org/10.1371/journal.pone.0023378. doi:10.1371/journal.pone.0023378

Clauset, A., Shalizi, C. R., & Newman, M. E. J. (2009). Power-Law Distributions in Empirical Data. *SIAM Review, 51*(4), 661-703. doi:10.1137/070710111

Csardi, G. (2019). Centralization. Retrieved from https://www.rdocumentation.org/packages/igraph/versions/0.7.1/topics/centralization

Donner, R. V., Zou, Y., Donges, J. F., Marwan, N., & Kurths, J. (2010). Recurrence networks—a novel paradigm for nonlinear time series analysis. *New Journal of Physics, 12*(3), 033025. Retrieved from http://dx.doi.org/10.1088/1367-2630/12/3/033025. doi:10.1088/1367-2630/12/3/033025

E. J. Newman, M. (2004). *Power Laws, Pareto Distributions and Zipf's Law* (Vol. 46).

Hill, R. C., Griffiths, W. E., & Lim, G. C. (2011). *Principles of Econometrics*. The United States of America: John Wiley & Sons, Inc.

Huang, Y., Cheng, W., Luo, S., Luo, Y., Ma, C., & He, T. (2016). Features of the Asynchronous Correlation between the China Coal Price Index and Coal Mining Accidental Deaths. *PLOS ONE, 11*(11), e0167198. Retrieved from https://doi.org/10.1371/journal.pone.0167198. doi:10.1371/journal.pone.0167198

Kruja, E., Marks, J., Blair, A., & Waters, R. (2002). A short note on the history of graph drawing. In (Vol. 2265, pp. 272-286).

Lacasa, L., Luque, B., Ballesteros, F., Luque, J., & Nuño, J. C. (2008). From time series to complex networks: The visibility graph. *Proceedings of the National Academy of Sciences, 105*(13), 4972. Retrieved from http://www.pnas.org/content/105/13/4972.abstract. doi:10.1073/pnas.0709247105

Lacasa, L., Luque, B., Luque, J., & Nuño, J. C. (2009). The visibility graph: A new method for estimating the Hurst exponent of fractional Brownian motion. *EPL (Europhysics Letters), 86*(3), 30001. Retrieved from http://dx.doi.org/10.1209/0295-5075/86/30001. doi:10.1209/0295-5075/86/30001

Lacasa, L., & Toral, R. (2010). *Description of stochastic and chaotic series using visibility graphs* (Vol. 82).

Li, R., Wang, J., Yu, H., Deng, B., Wei, X., & Chen, Y. (2016). Fractal analysis of the short time series in a visibility graph method. *Physica A: Statistical Mechanics and its Applications, 450*, 531-540. Retrieved from http://www.sciencedirect.com/science/article/pii/S0378437115010997. doi:https://doi.org/10.1016/j.physa.2015.12.071

Luque, B., Lacasa, L., Ballesteros, F., & Luque, J. (2009). Horizontal visibility graphs: Exact results for random time series. *Physical Review E, 80*(4), 046103. Retrieved from https://link.aps.org/doi/10.1103/PhysRevE.80.046103. doi:10.1103/PhysRevE.80.046103

Marwan, N., Donges, J. F., Zou, Y., Donner, R. V., & Kurths, J. (2009). Complex network approach for recurrence analysis of time series. *Physics Letters A, 373*(46), 4246-4254. Retrieved from http://www.sciencedirect.com/science/article/pii/S0375960109011852. doi:https://doi.org/10.1016/j.physleta.2009.09.042

Newman, M. E. J. (2002). Assortative Mixing in Networks. *Physical Review Letters, 89*(20), 208701. Retrieved from https://link.aps.org/doi/10.1103/PhysRevLett.89.208701. doi:10.1103/PhysRevLett.89.208701

Newman, M. E. J. (2003). The Structure and Function of Complex Networks. *SIAM Review, 45*(2), 167-256. Retrieved from https://epubs.siam.org/doi/abs/10.1137/S003614450342480. doi:10.1137/s003614450342480

Newman, M. E. J. (2010). *Networks : an introduction*. Oxford: Oxford University Press.

Ravetti, M. G., Carpi, L. C., Gonçalves, B. A., Frery, A. C., & Rosso, O. A. (2014). Distinguishing Noise from Chaos: Objective versus Subjective Criteria Using Horizontal Visibility Graph. *PLOS ONE, 9*(9), e108004. Retrieved from https://doi.org/10.1371/journal.pone.0108004. doi:10.1371/journal.pone.0108004

Segberg, E., & Skoglund, S. (2017). *Visibility Graph Analysis of Real-Life and GARCH-Simulated Financial Time-Series.* University of Agder, Kristiansand, Norway.

Sun, M., Wang, Y., & Gao, C. (2016). Visibility graph network analysis of natural gas price: The case of North American market. *Physica A: Statistical Mechanics and its Applications, 462*, 1-11. Retrieved from http://www.sciencedirect.com/science/article/pii/S0378437116303193. doi:https://doi.org/10.1016/j.physa.2016.06.051

Verbeek, M. (2004). *A Guide to Modern Econometrics*. England: John Wiley & Sons Inc.

Wilson, R. J. (2010). *Introduction to Graph Theory* (5th ed. ed.): United Kingdom: Pearson Education M.U.A.

Xu, X., Zhang, J., & Small, M. (2008). Superfamily phenomena and motifs of networks induced from time series. *Proceedings of the National Academy of Sciences, 105*(50), 19601-19605. Retrieved from https://www.pnas.org/content/pnas/105/50/19601.full.pdf. doi:10.1073/pnas.0806082105

Yang, J., Qu, Z., & Chang, H. (2015). Investigation on Law and Economics Based on Complex Network and Time Series Analysis. *PLOS ONE, 10*(6), e0127001. Retrieved from https://doi.org/10.1371/journal.pone.0127001. doi:10.1371/journal.pone.0127001

Yu, L. (2013). Visibility graph network analysis of gold price time series. *Physica a-Statistical Mechanics and Its Applications, 392*(16), 3374-3384. Retrieved from <Go to ISI>://WOS:000320292500009. doi:10.1016/j.physa.2013.03.063

Zhang, J., & Small, M. (2006). Complex Network from Pseudoperiodic Time Series: Topology versus Dynamics. *Physical Review Letters, 96*(23), 238701. Retrieved from https://link.aps.org/doi/10.1103/PhysRevLett.96.238701. doi:10.1103/PhysRevLett.96.238701

Zhang, R., Zou, Y., Zhou, J., Gao, Z.-K., & Guan, S. (2017). Visibility graph analysis for re-sampled time series from auto-regressive stochastic processes. *Communications in Nonlinear Science and Numerical Simulation, 42*, 396-403. Retrieved from http://www.sciencedirect.com/science/article/pii/S1007570416301332. doi:https://doi.org/10.1016/j.cnsns.2016.04.031

Zhuang, E., Small, M., & Feng, G. (2014). Time series analysis of the developed financial markets' integration using visibility graphs. *Physica A: Statistical Mechanics and its Applications, 410*, 483-495. Retrieved from http://www.sciencedirect.com/science/article/pii/S0378437114004397. doi:https://doi.org/10.1016/j.physa.2014.05.058

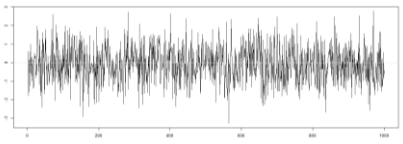
APPENDICES

A: Line plot of realizations generated by an $MA(1)$ with different parameters
B: Line plot of realizations generate by an $AR(1)$ with different parameters
C: Line plot of realizations generated by an $ARMA(1,1)$ with different parameters for AR*
D: Graph of realizations generated by an $MA(1)$ with different parameters
E: Graph of realizations generate by an $AR(1)$ with different parameters
F: Graph of realizations generated by an $ARMA(1,1)$ with different parameters for AR*
G: OLS regression analysis
H: R-code
I: Reflection notes

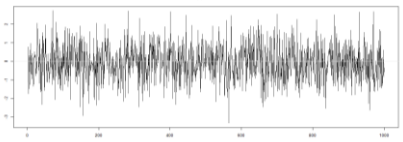
A: Line plot of realizations generate by an $\boldsymbol{MA(1)}$ with different parameters

The processes' autocorrelation coefficient is shortened to "coef" in the table

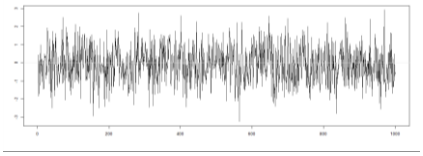
Coef=0:



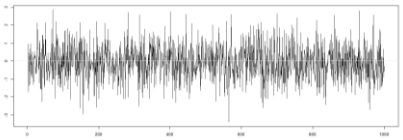
Coef=-0.1:



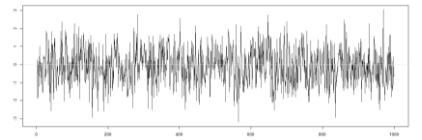
Coef=0,1:



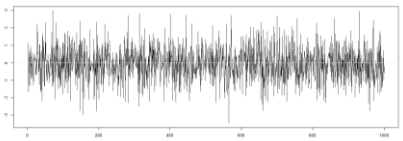
Coef=-0.2:



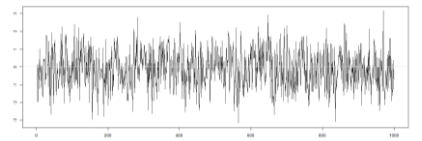
Coef= 0.2:



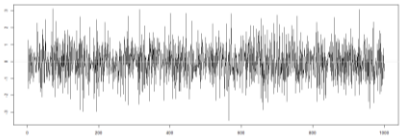
Coef=-0.3:



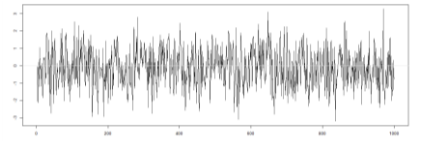
Coef= 0.3:



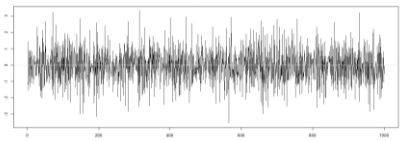
Coef=-0.4:



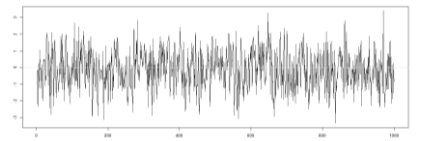
Coef= 0.4:



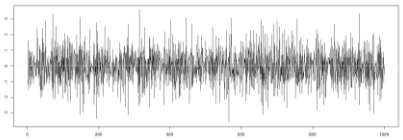
Coef=-0.5:



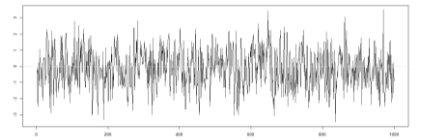
Coef= 0.5:



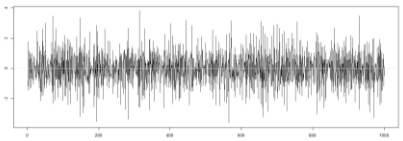
Coef=-0.6:



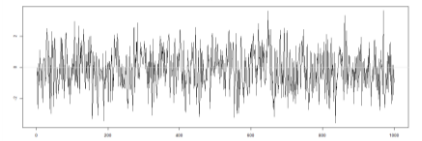
Coef= 0.6:



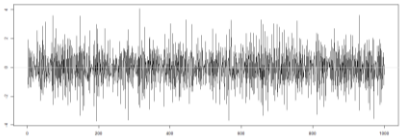
Coef=-0.7:



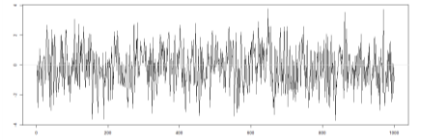
Coef= 0.7:



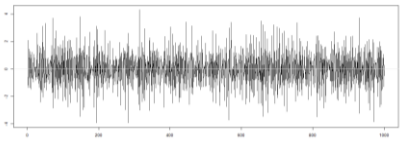
Coef=-0.8:



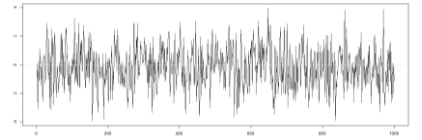
Coef= 0.8:



Coef=-0.9:



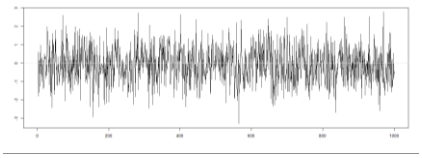
Coef= 0.9:



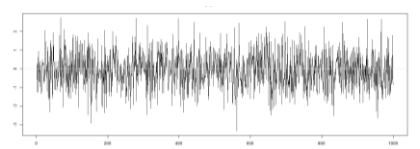
B: Line plot of realizations generate by an $\boldsymbol{AR(1)}$ with different parameters:

The processes' autocorrelation coefficient is shortened to "coef" in the table

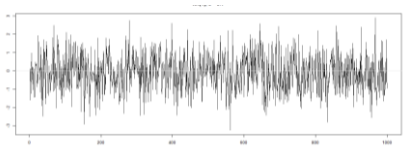
Coef=0:



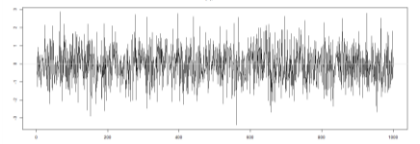
Coef=-0.1:



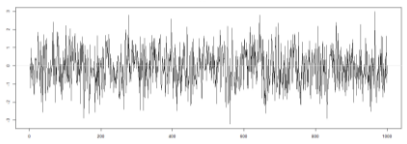
Coef= 0.1:



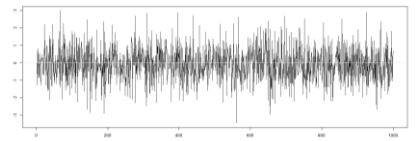
Coef=-0.2:



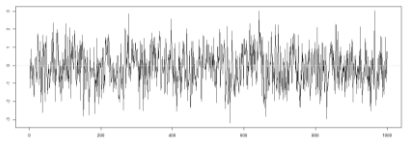
Coef= 0.2:



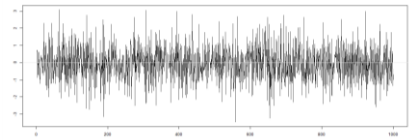
Coef=-0.3:



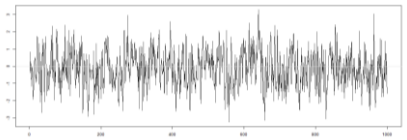
Coef= 0.3:



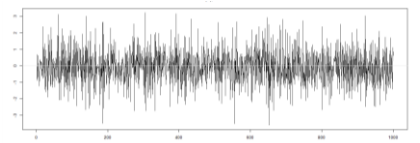
Coef=-0.4:



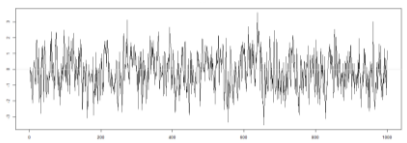
Coef= 0.4:



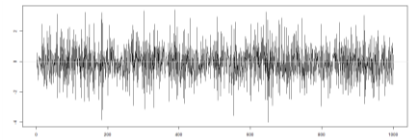
Coef=-0.5:



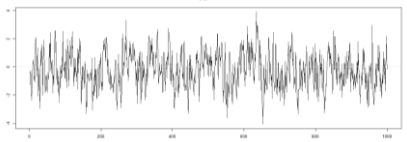
Coef= 0.5:



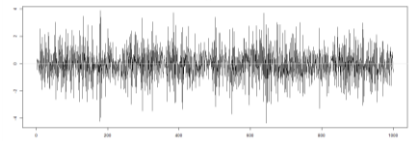
Coef=-0.6:



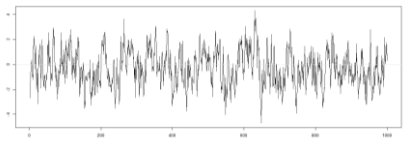
Coef= 0.6:



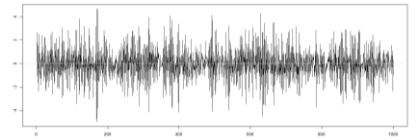
Coef=-0.7:



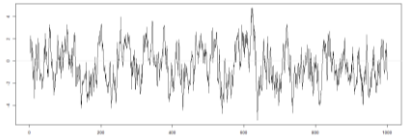
Coef= 0.7:



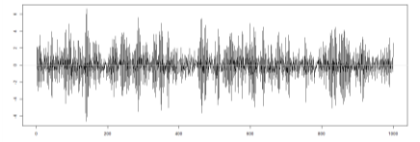
Coef=-0,8:



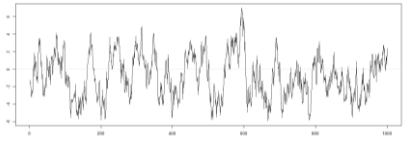
Coef= 0.8:



Coef=-0.9:



Coef=0.9:

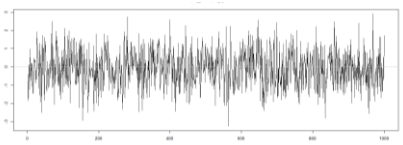


C: Line plot of realizations generate by an $\boldsymbol{ARMA(1,1)}$with different parameters for $\boldsymbol{AR}*$

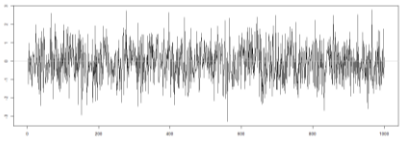
The processes' autocorrelation coefficient is shortened to "coef" in the table

**Autocorrelation coefficient of $MA(1)$ is constant and set at 0.1*

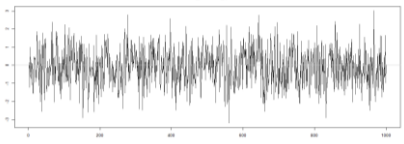
Coef=0:



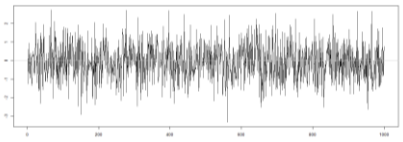
Coef=-0.1:



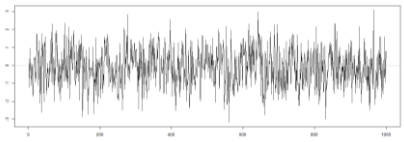
Coef= 0.1:



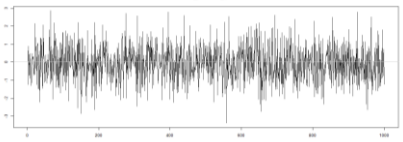
Coef=-0.2:



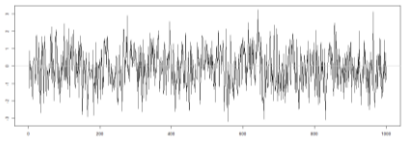
Coef= 0.2:



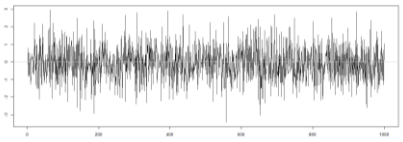
Coef=-0.3:



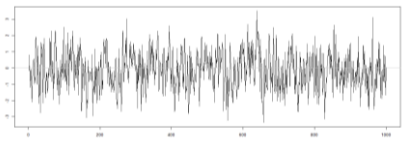
Coef= 0.3:



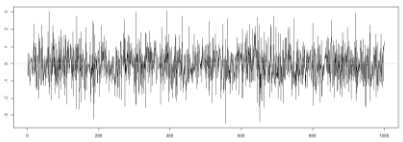
Coef=-0.4:



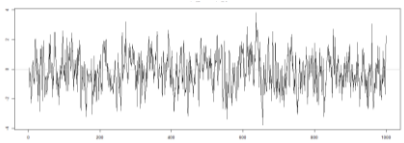
Coef= 0.4:



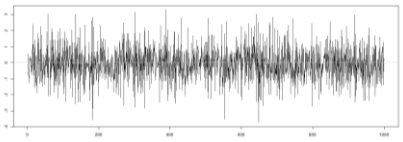
Coef=-0.5:



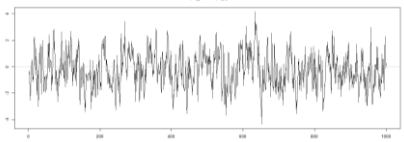
Coef= 0.5:



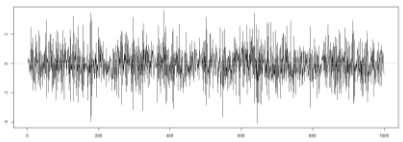
Coef=-0.6:



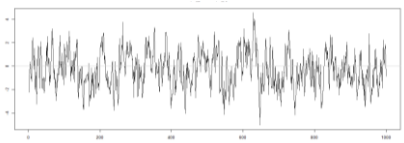
Coef= 0.6:



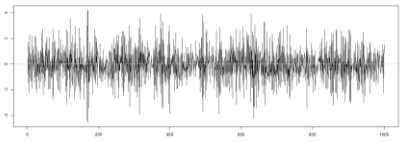
Coef=-0.7:



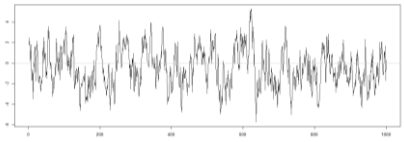
Coef= 0.7:



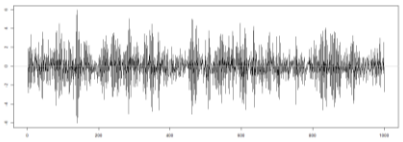
Coef=-0.8:



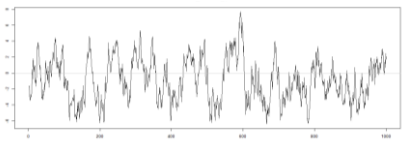
Coef= 0.8:



Coef=-0.9:

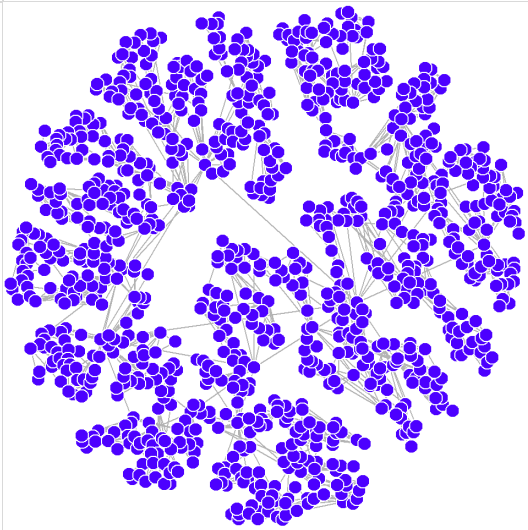
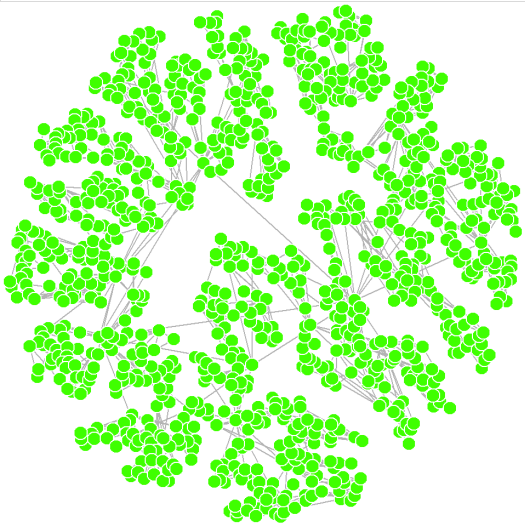
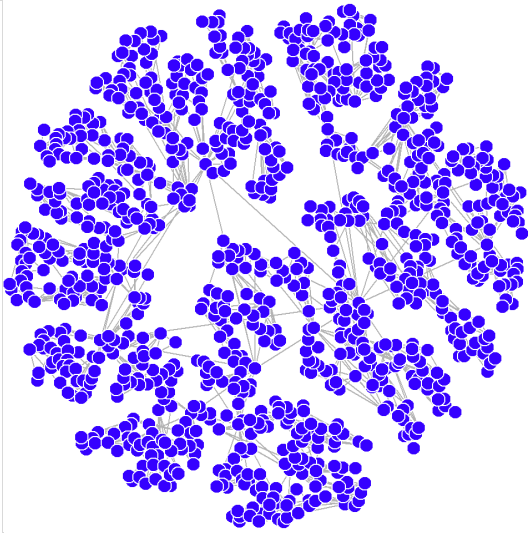
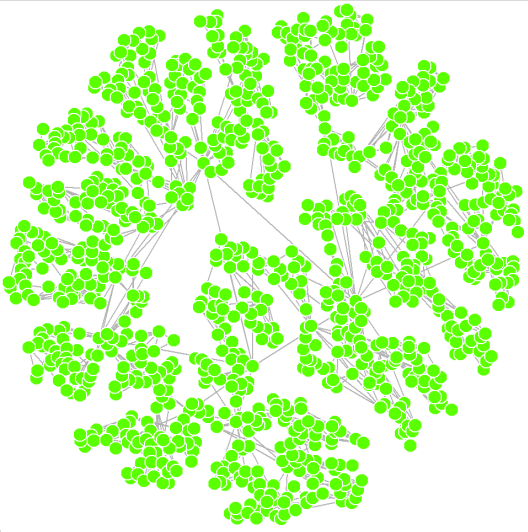
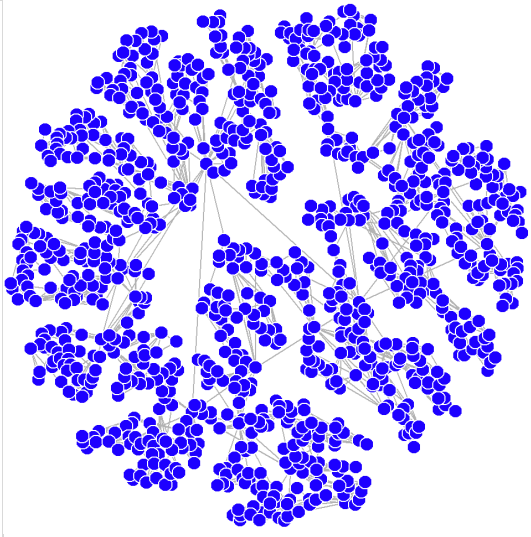
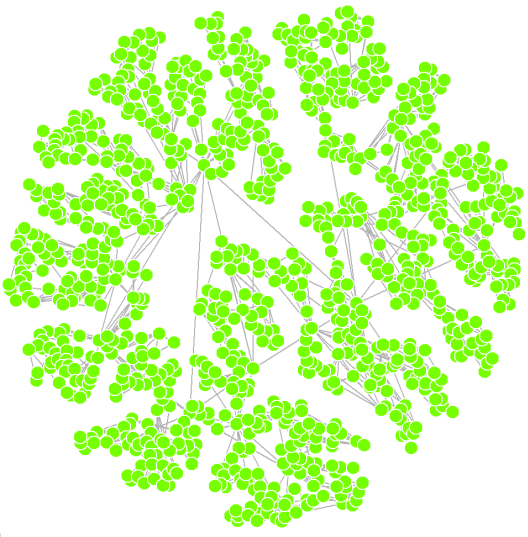


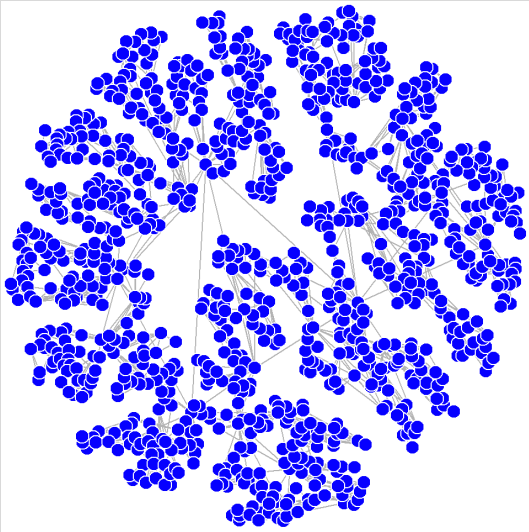
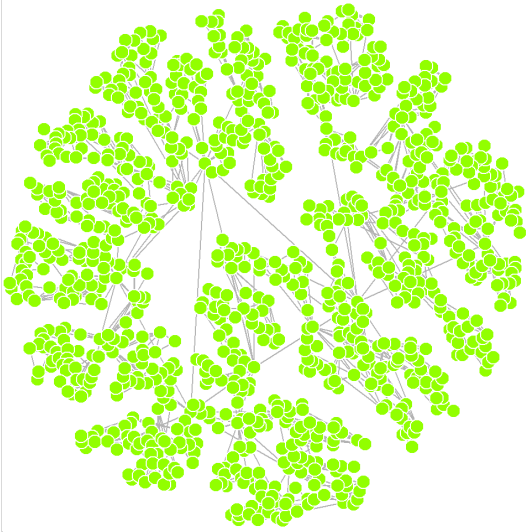
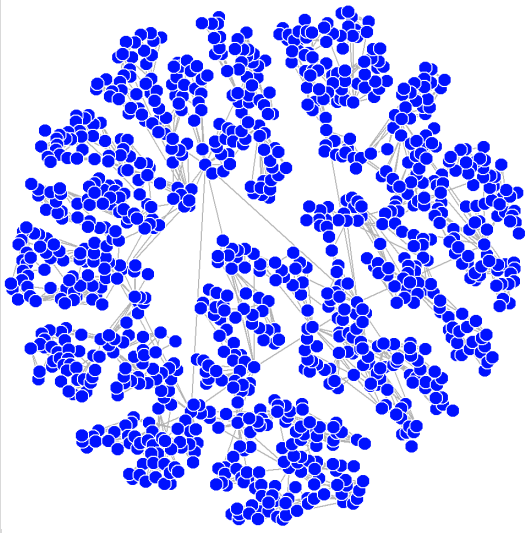
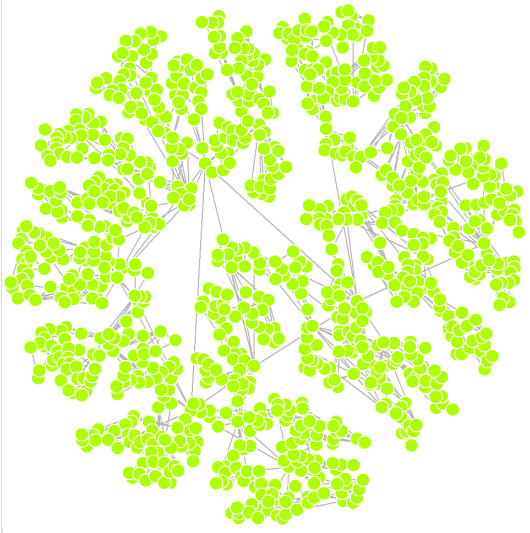
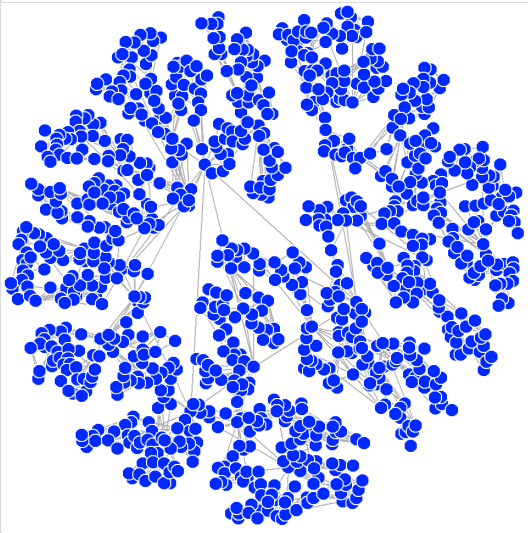
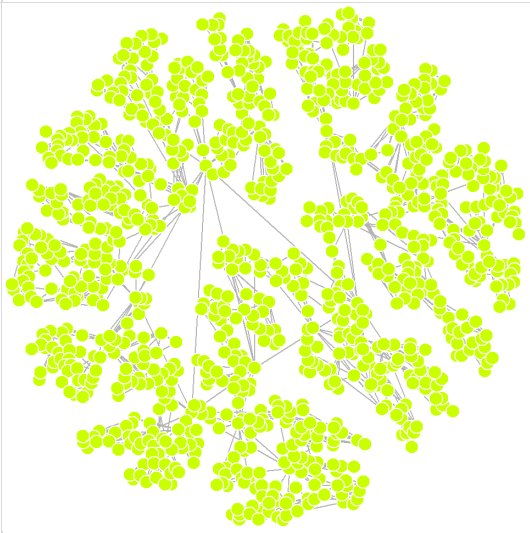
Coef=0.9:

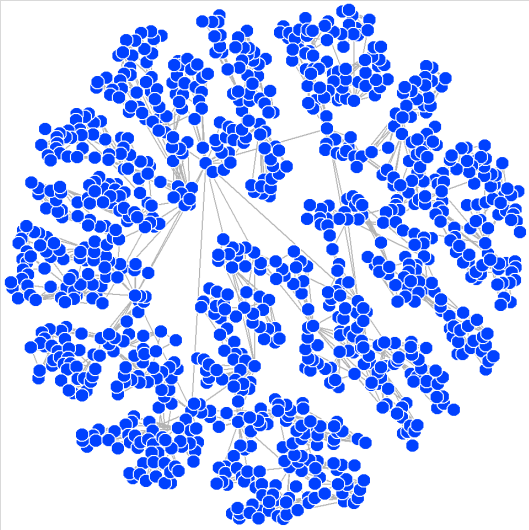
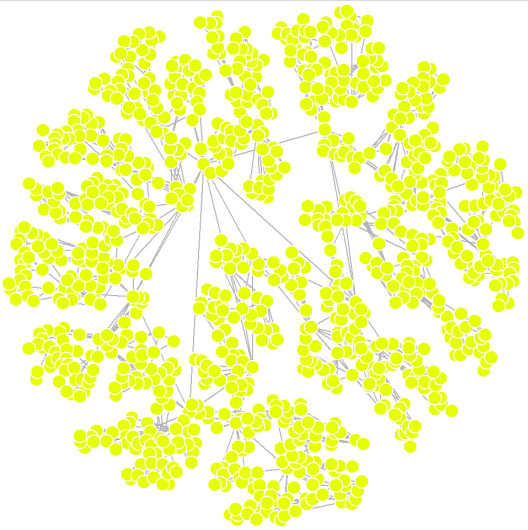
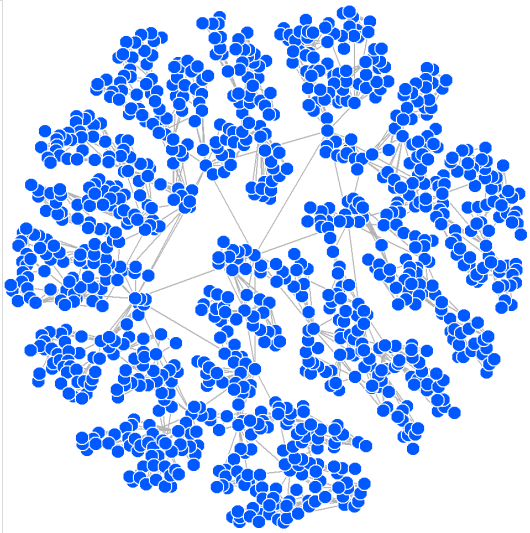
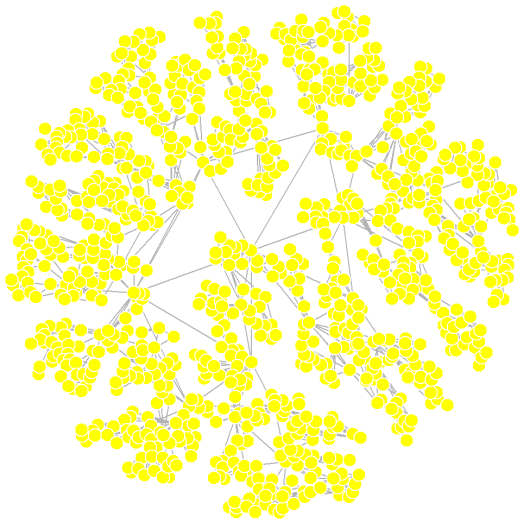
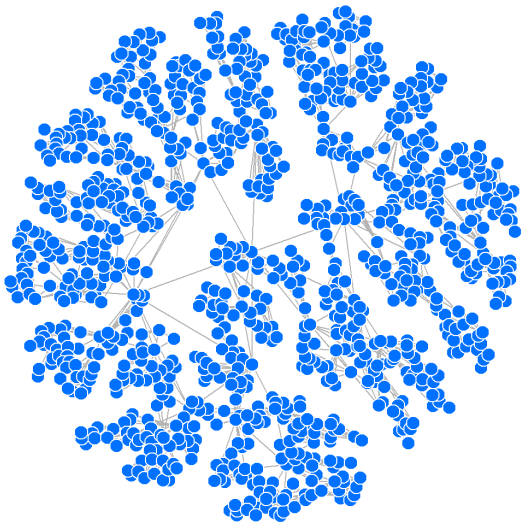
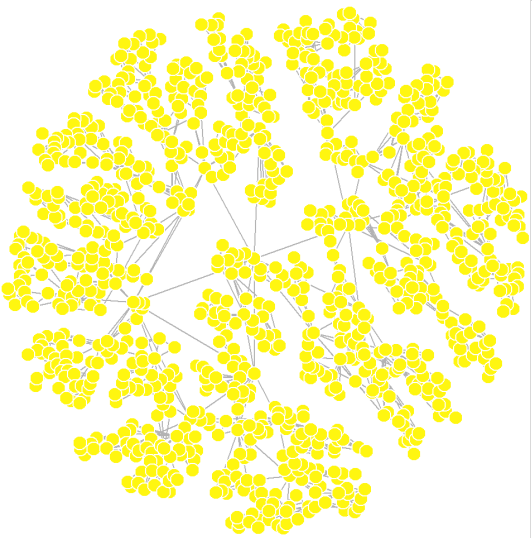


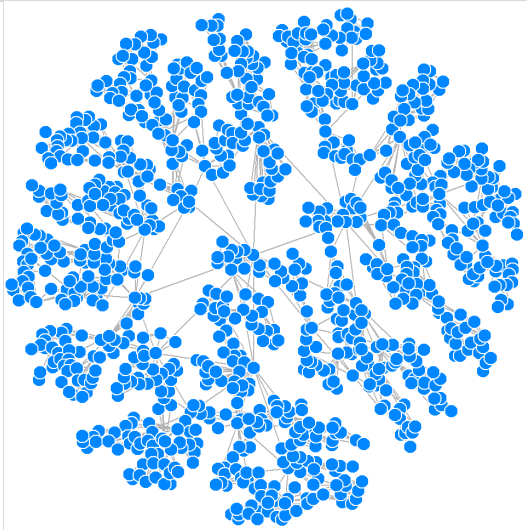
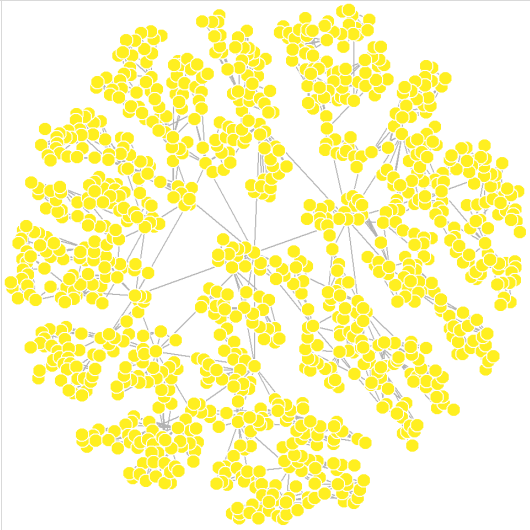
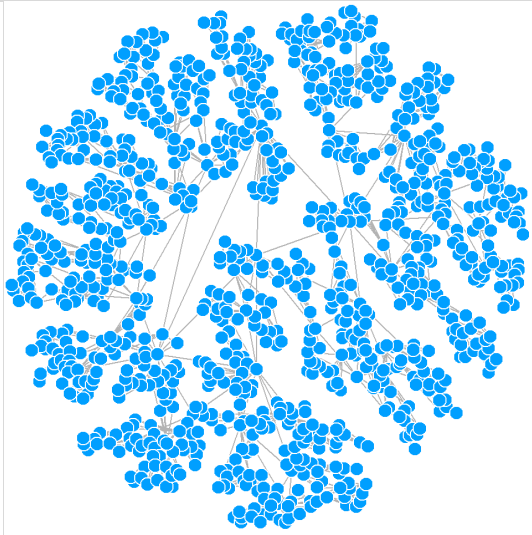
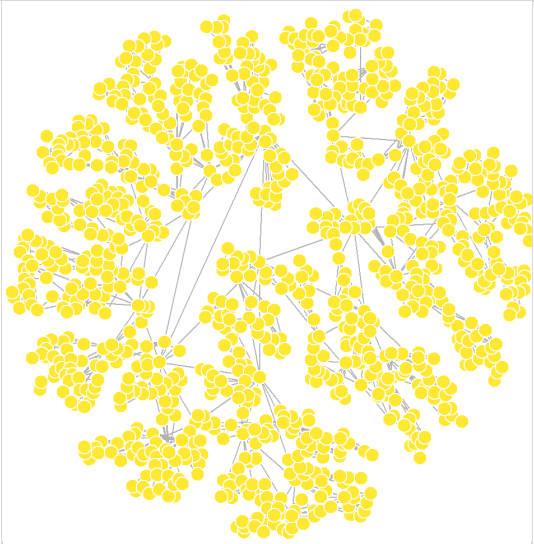
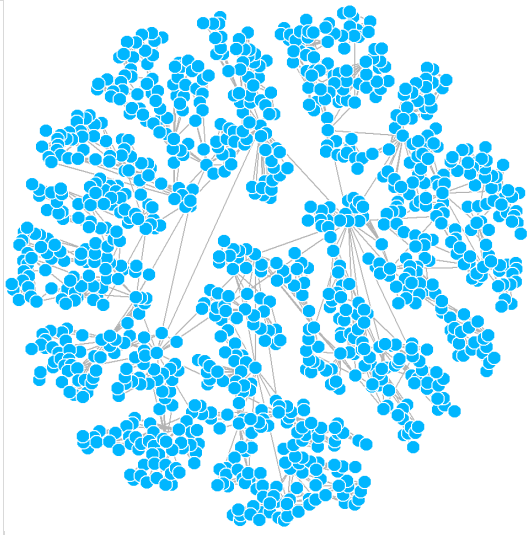
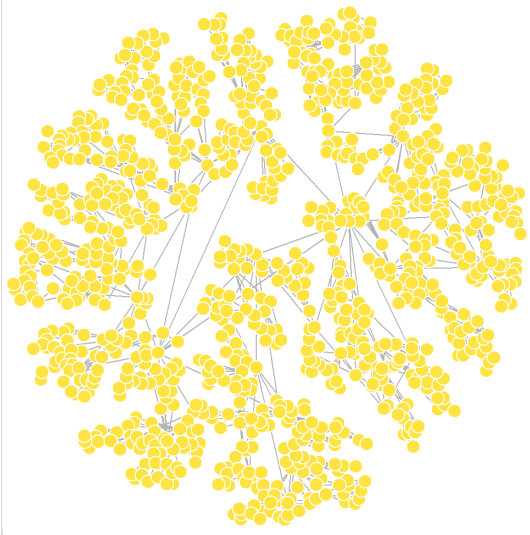
D: Graphs of realizations generated by an $MA(1)$ with different parameters

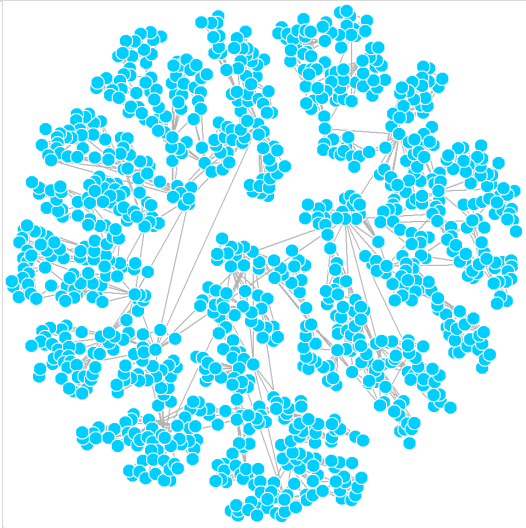
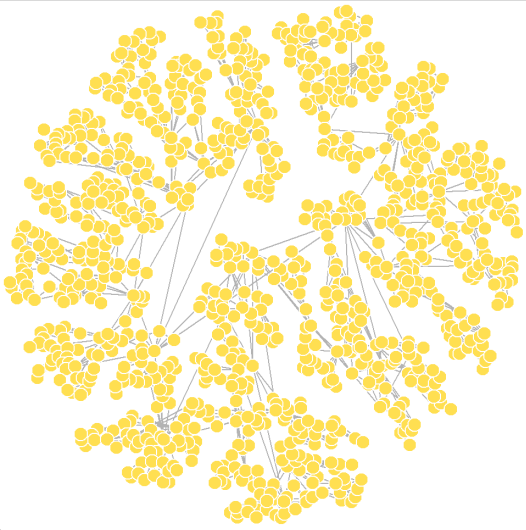
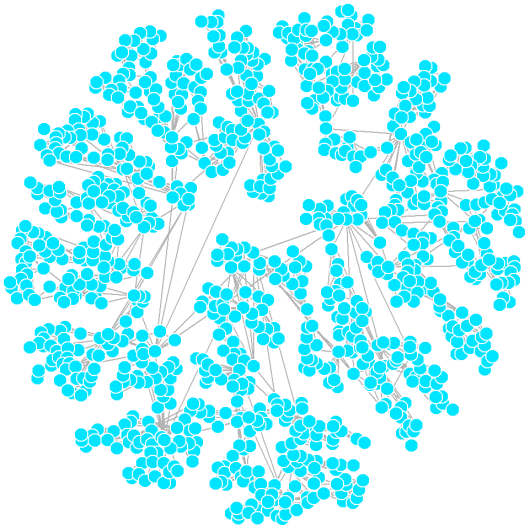
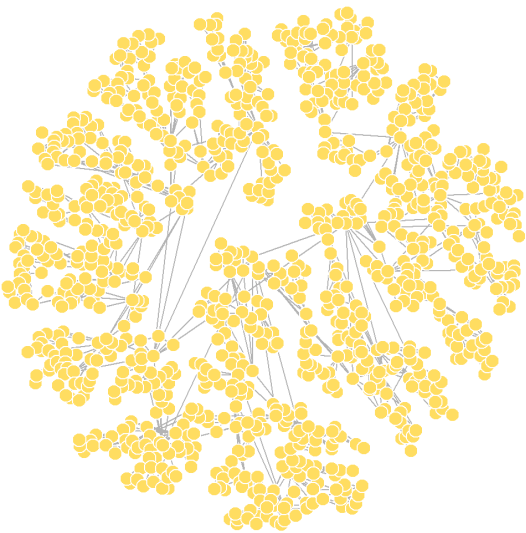
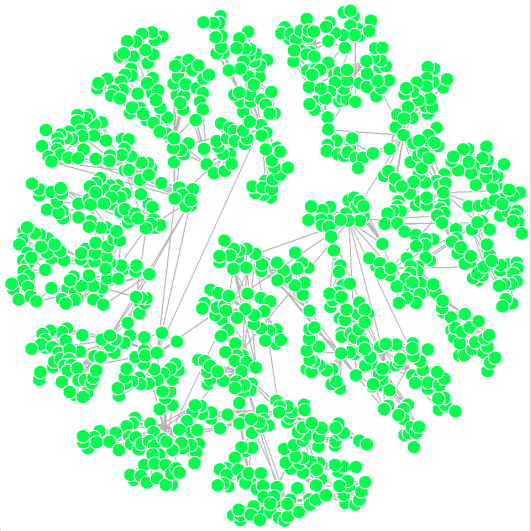
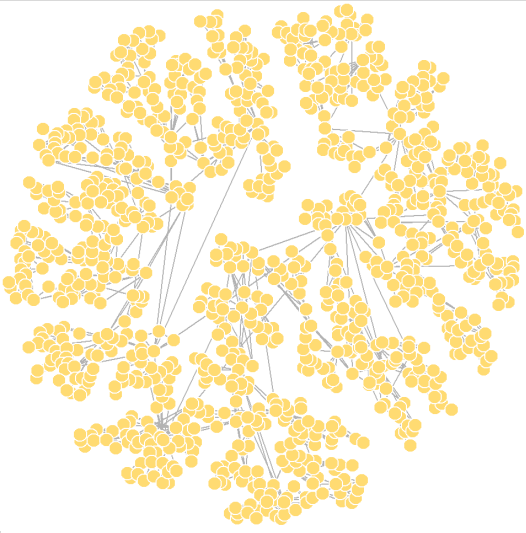
The processes' autocorrelation coefficient is shortened to "coef" in the table

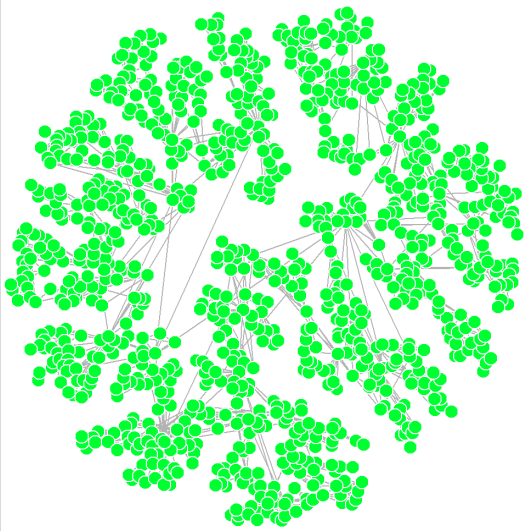
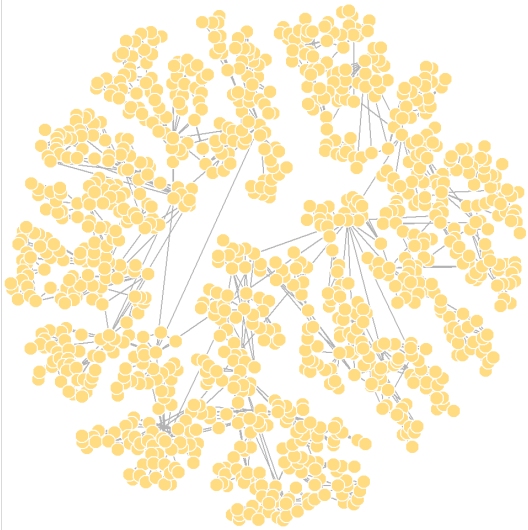
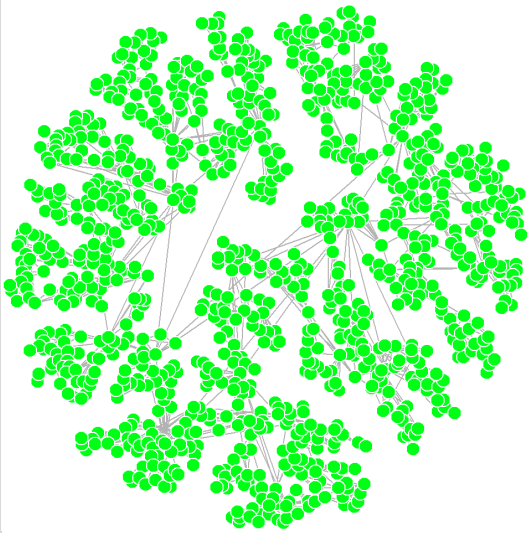
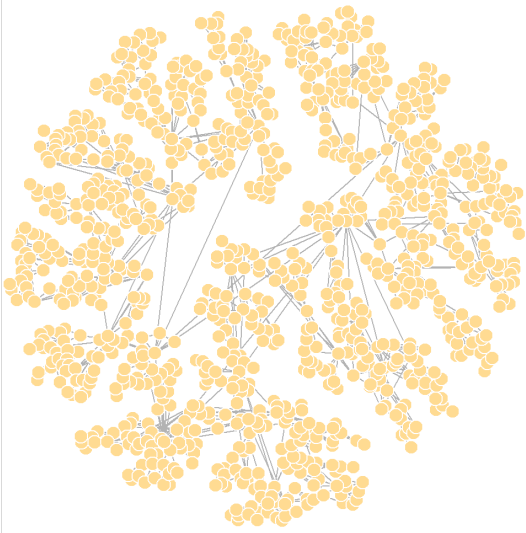
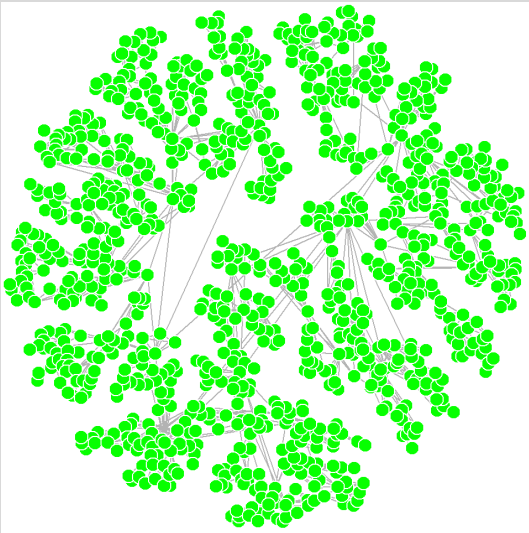
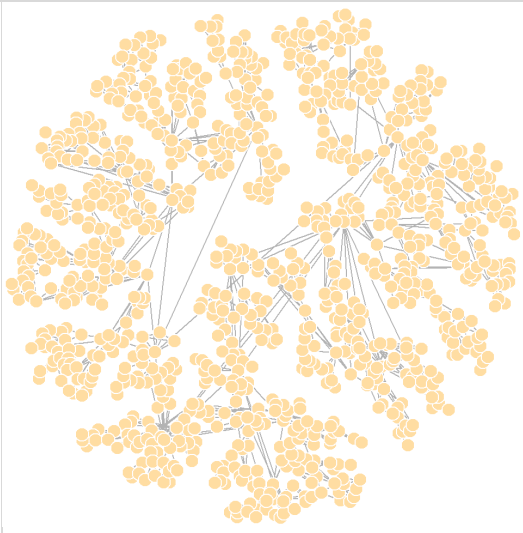
	Visibility graph	*Horizontal visibility graph*
Coef=-0.9:		
Coef=-0.8:		
Coef=-0.7:		

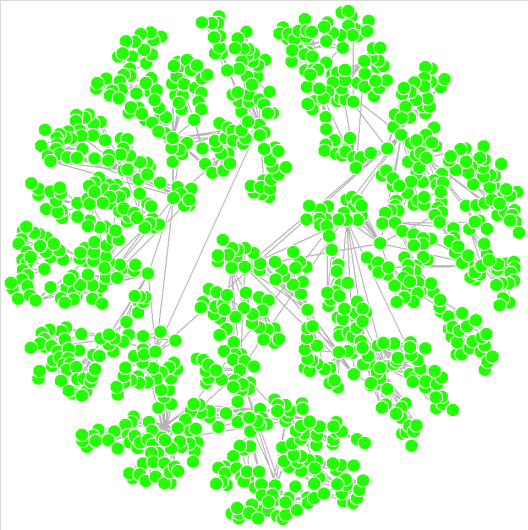
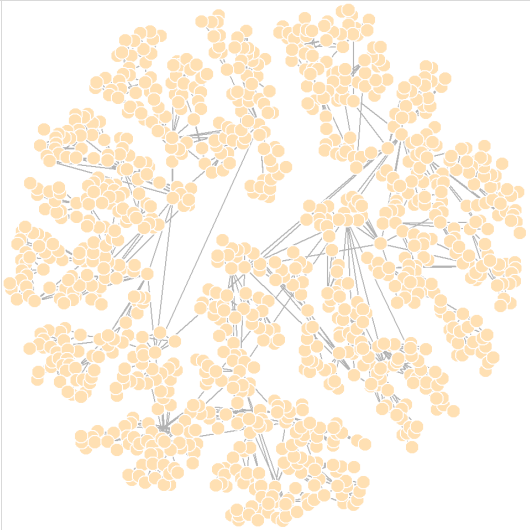
	Visibility graph	*Horizontal visibility graph*
Coef=-0.6:		
Coef=-0.5::		
Coef=-0.4:		

	Visibility graph	*Horizontal visibility graph*
Coef=-0.3:		
Coef=-0.2:		
Coef=-0.1:		

	Visibility graph	*Horizontal visibility graph*
Coef=0:		
Coef= 0.1:		
Coef= 0.2:		

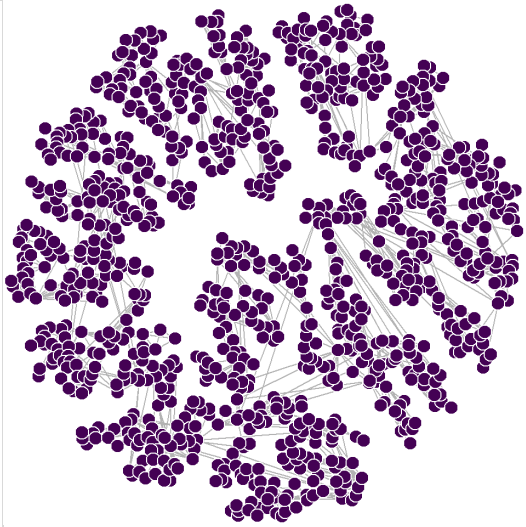
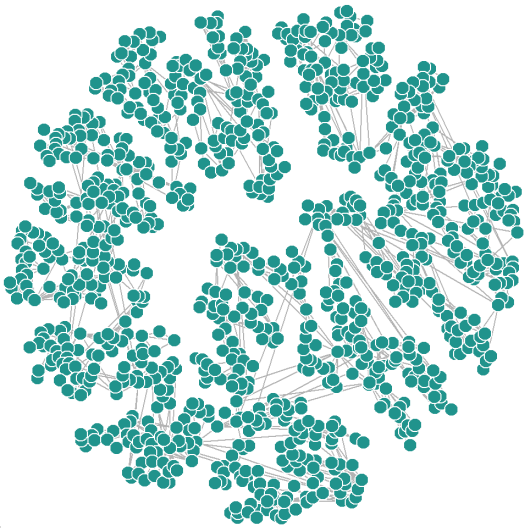
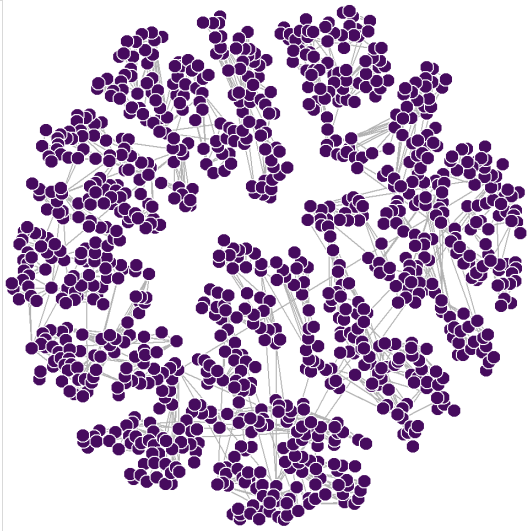
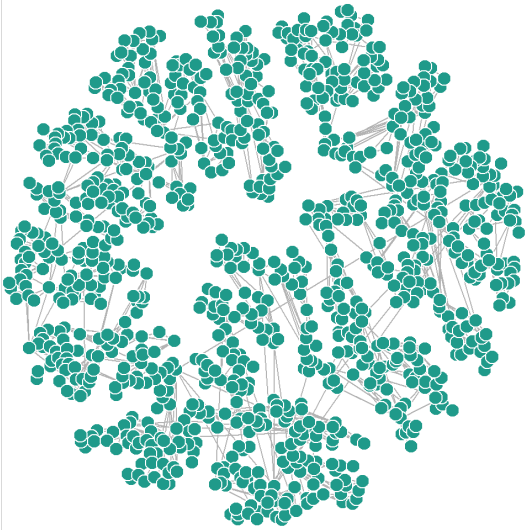
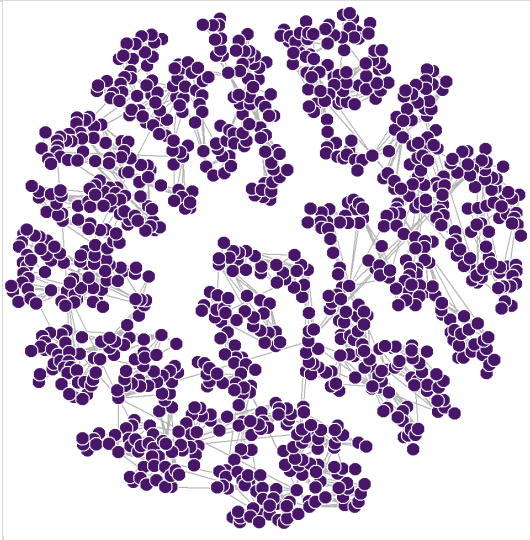
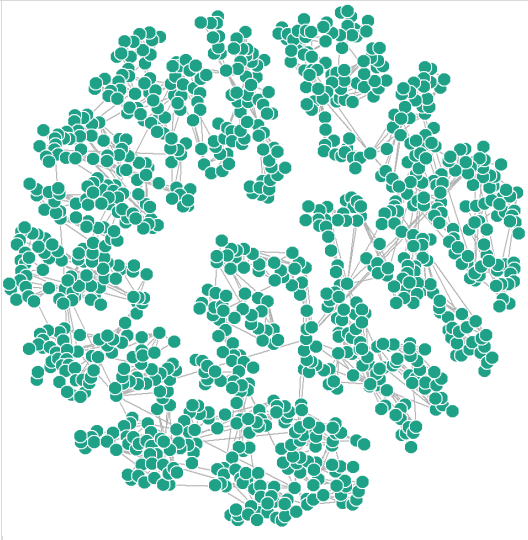
	Visibility graph	*Horizontal visibility graph*
Coef= 0.3:		
Coef= 0.4:		
Coef= 0.5:		

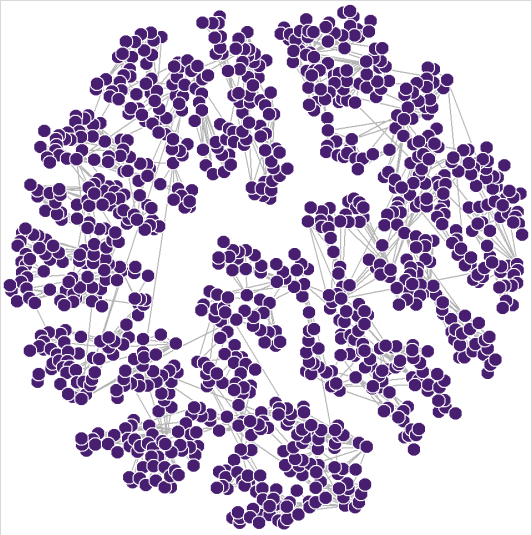
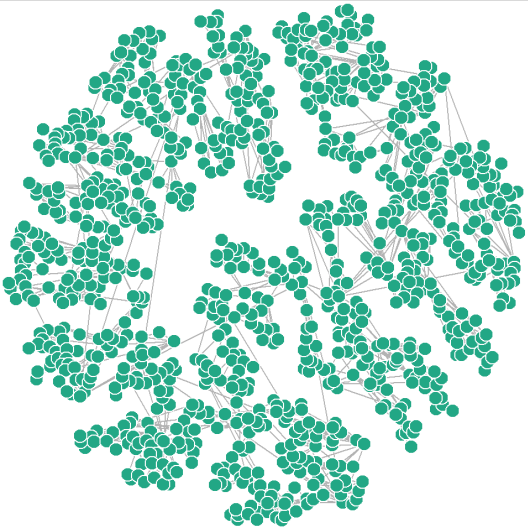
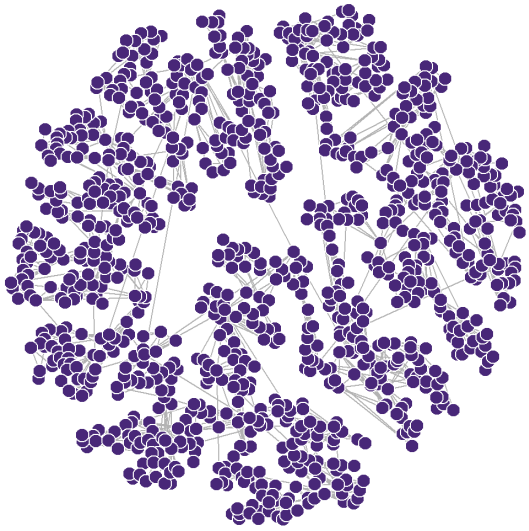
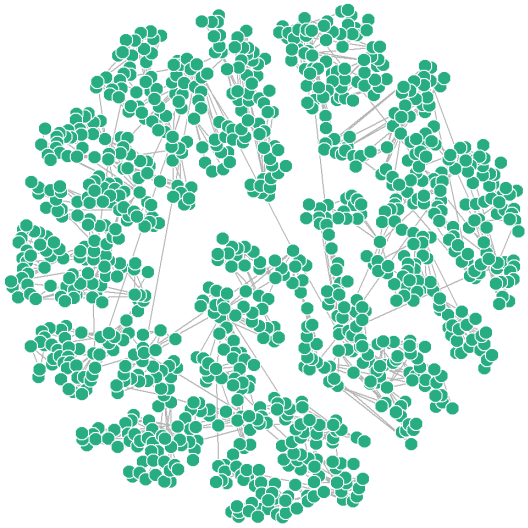
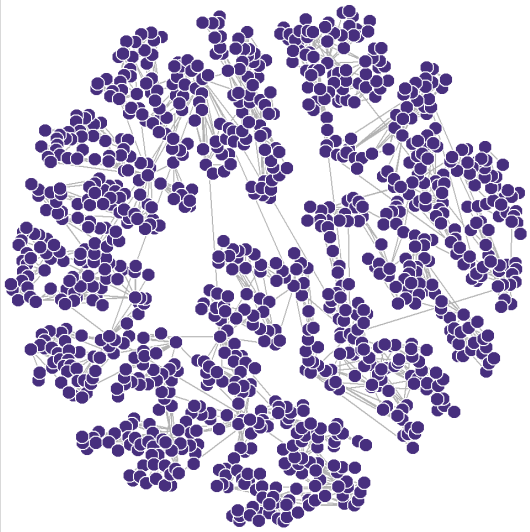
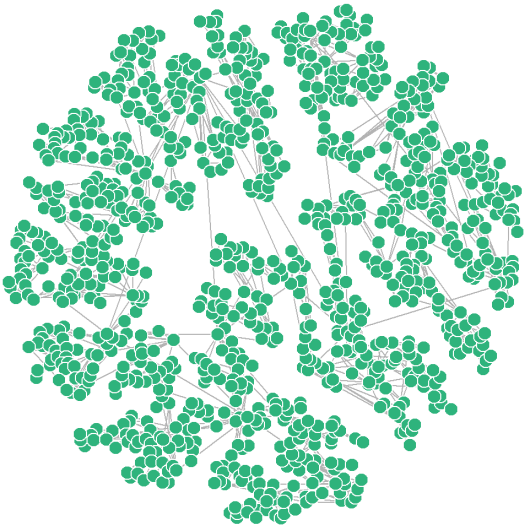
	Visibility graph	*Horizontal visibility graph*
Coef= 0.6:		
Coef= 0.7:		
Coef= 0.8:		

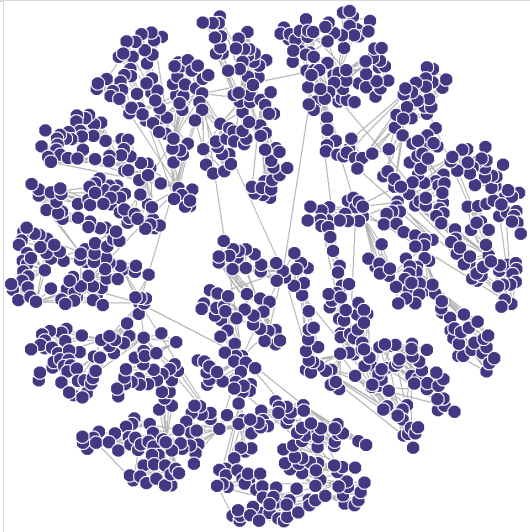
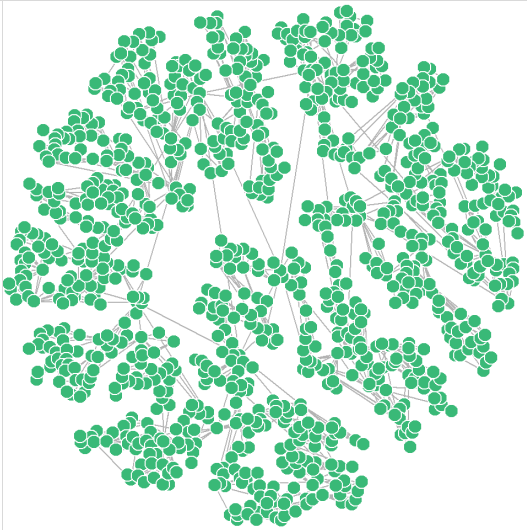
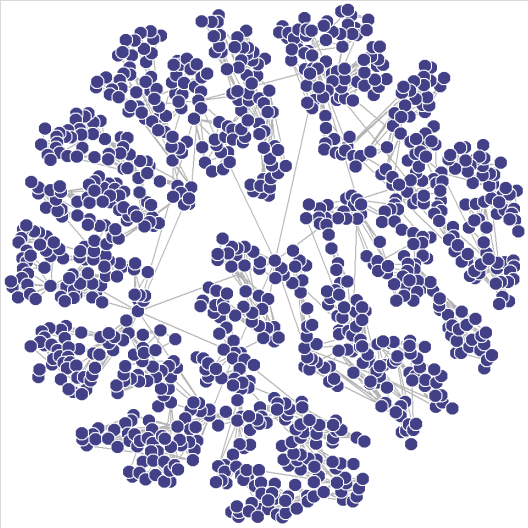
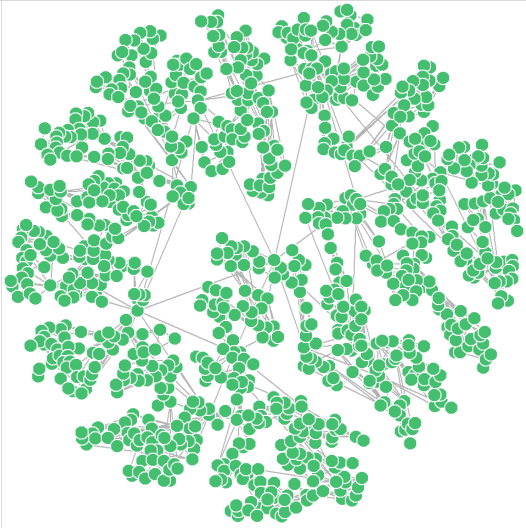
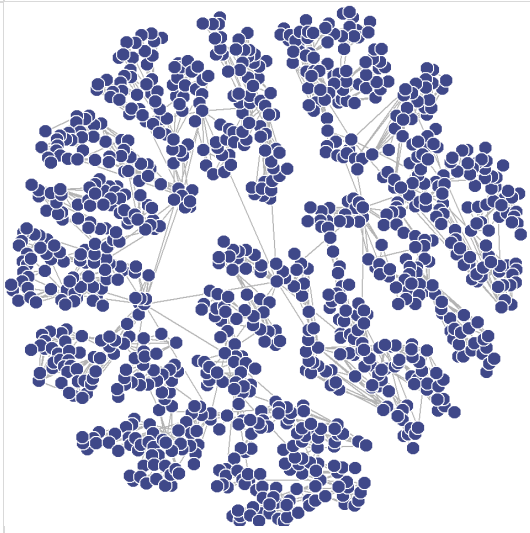
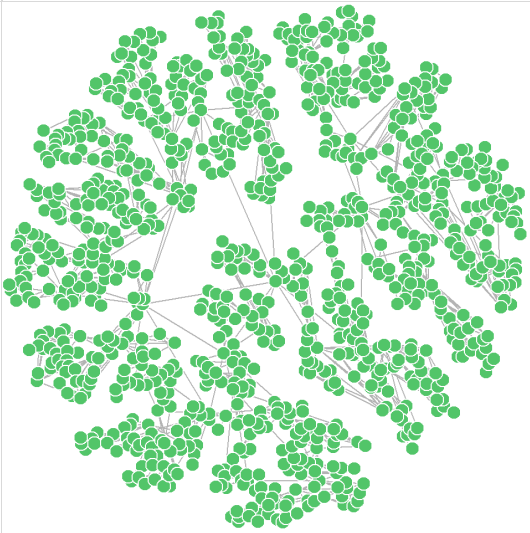
	Visibility graph	*Horizontal visibility graph*
Coef= 0.9:		

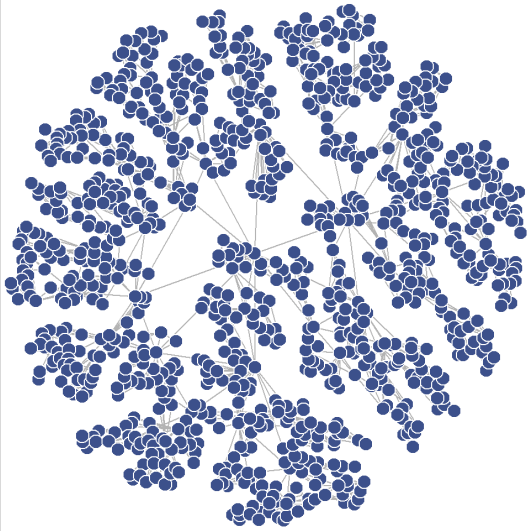
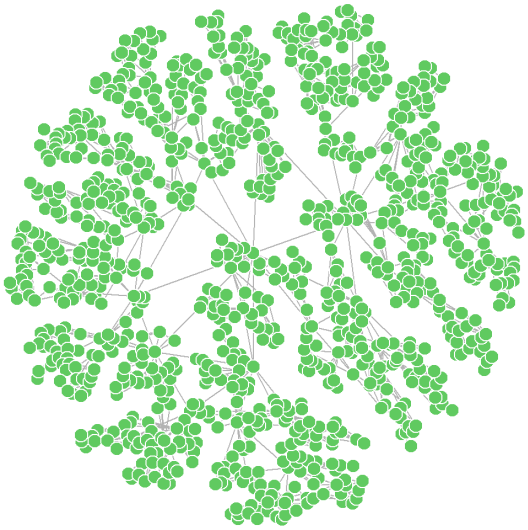
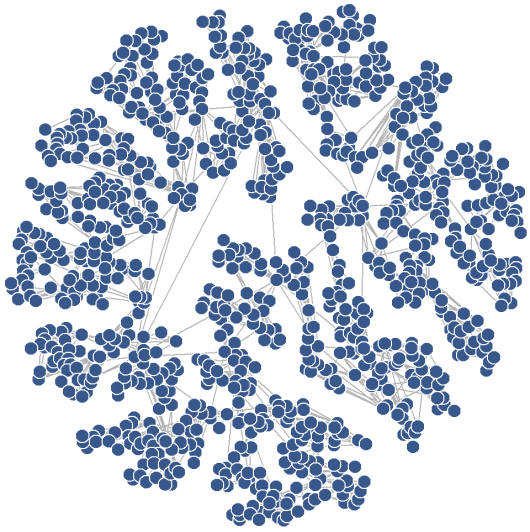
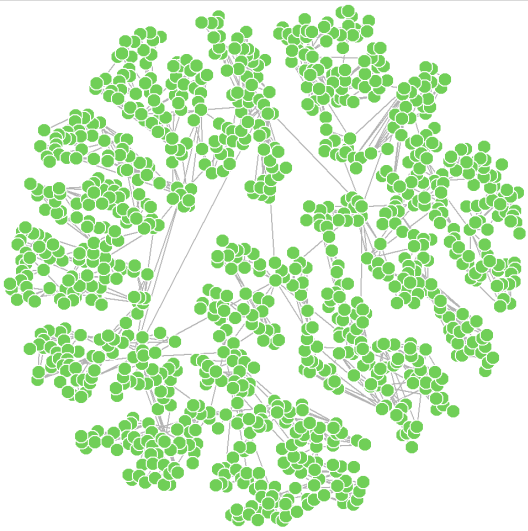
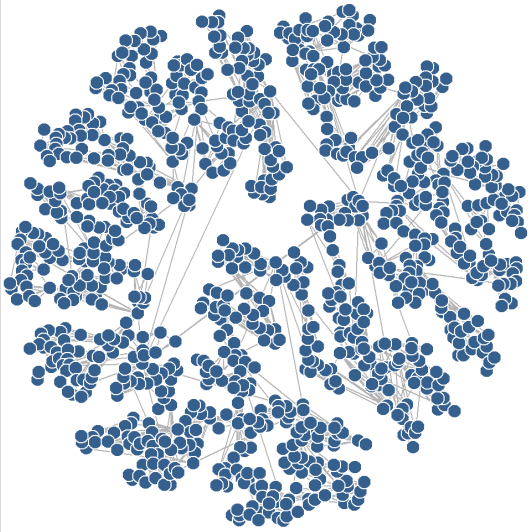
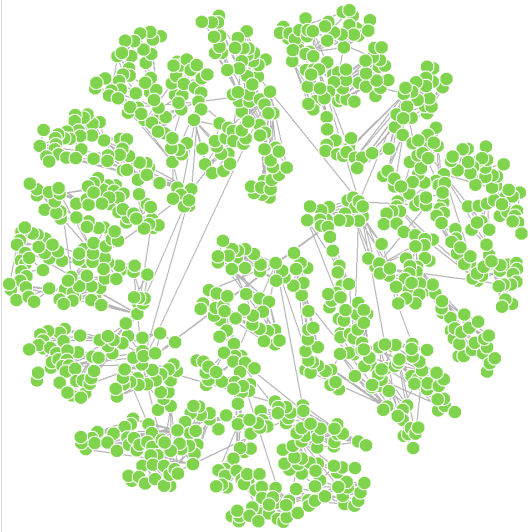
E: Graphs of realizations generate by an $\boldsymbol{AR(1)}$ with different parameters

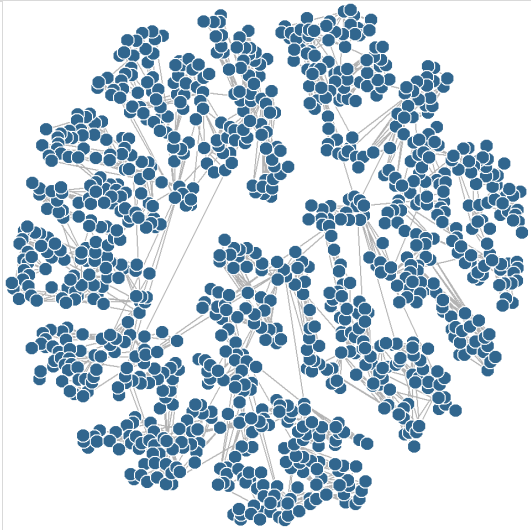
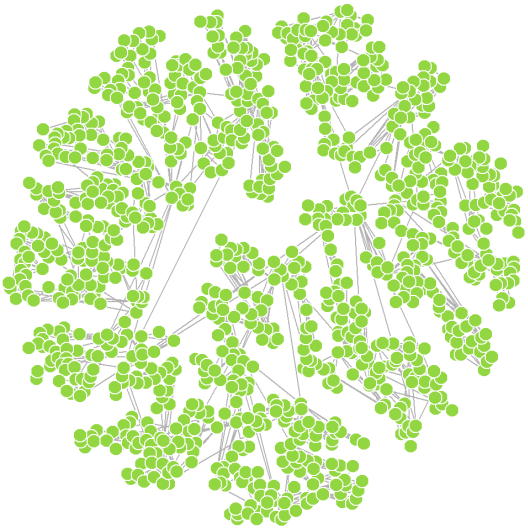
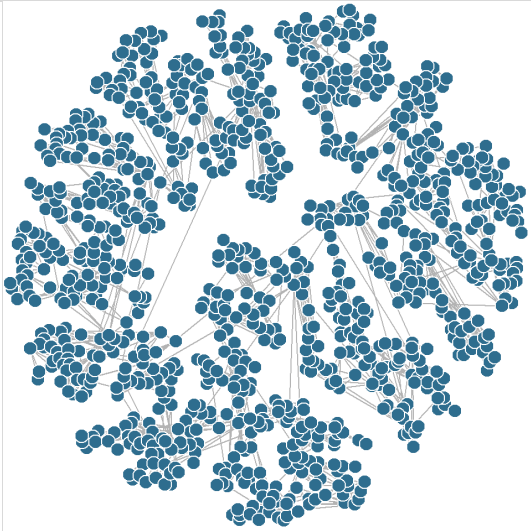
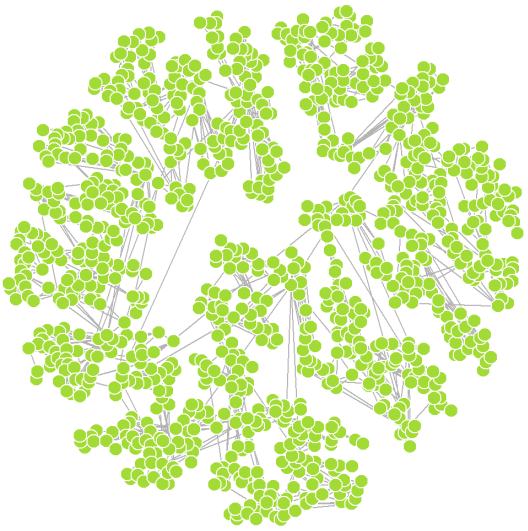
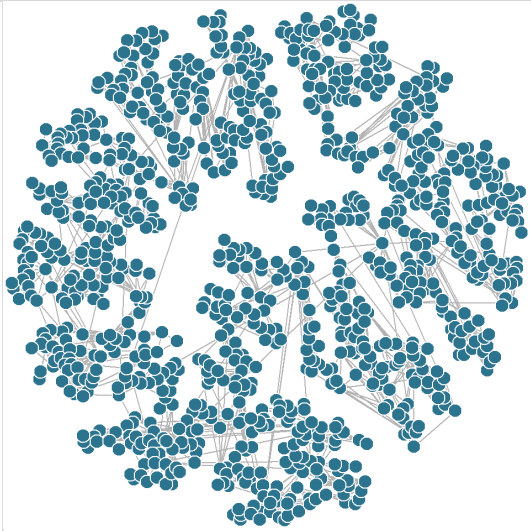
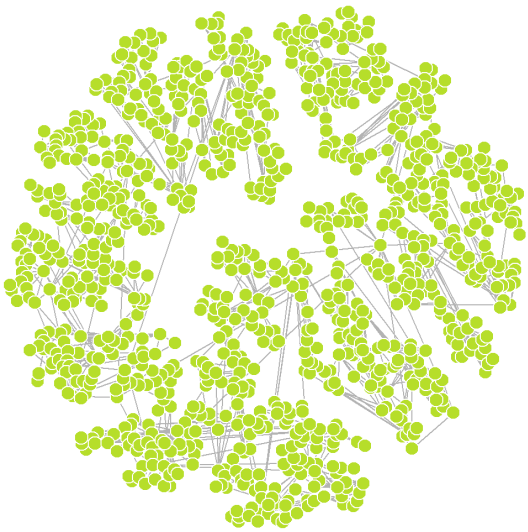
The processes' autocorrelation coefficient is shortened to "coef" in the table

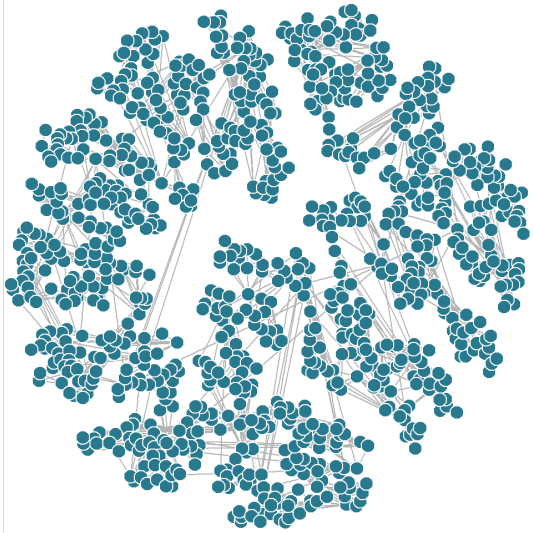
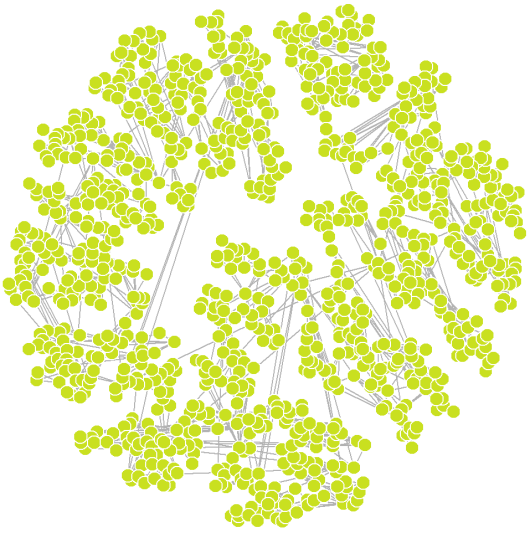
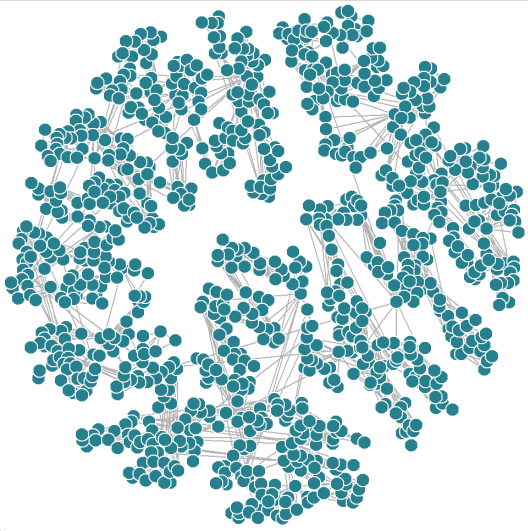
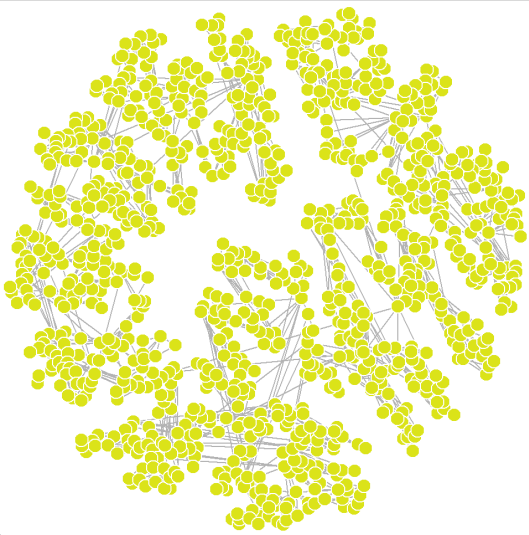
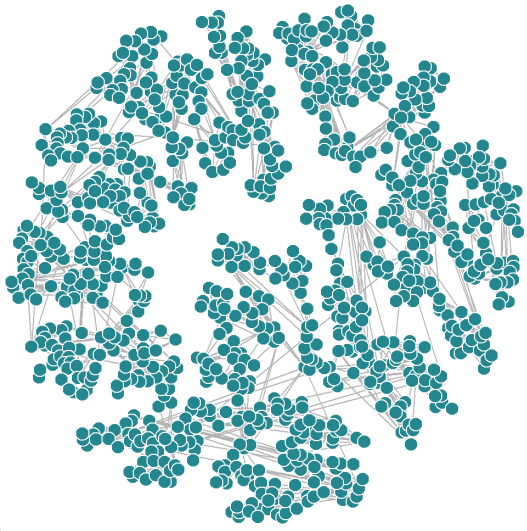
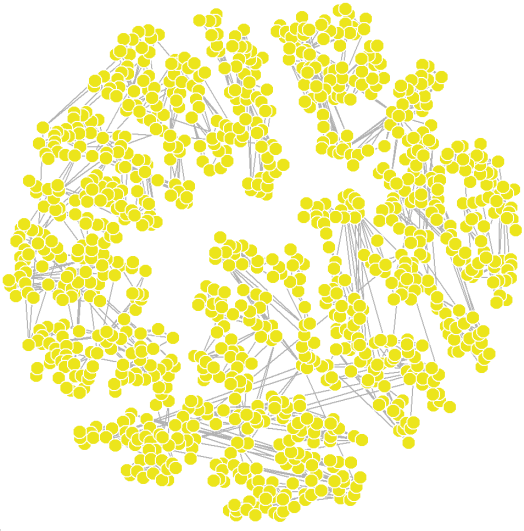
	Visibility graph	*Horizontal visibility graph*
Coef=-0.9:		
Coef=-0.8:		
Coef=-0.7:		

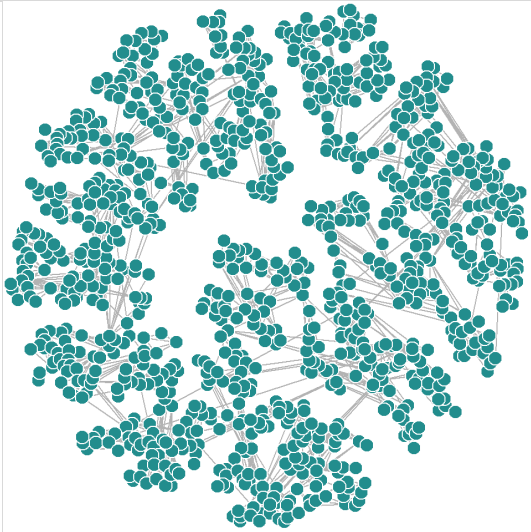
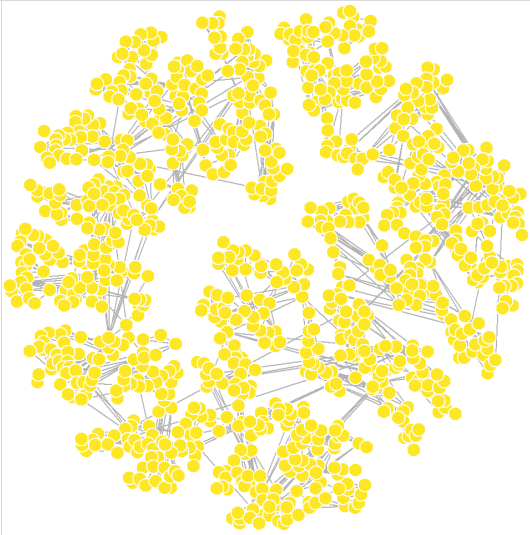
	Visibility graph	*Horizontal visibility graph*
Coef=-0.6:		
Coef=-0.5::		
Coef=-0.4:		

	Visibility graph	*Horizontal visibility graph*
Coef=-0.3:		
Coef=-0.2:		
Coef=-0.1:		

	Visibility graph	*Horizontal visibility graph*
Coef=0:		
Coef= 0.1:		
Coef= 0.2:		

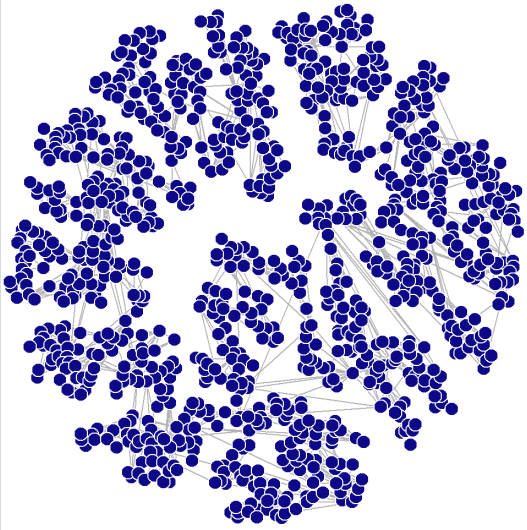
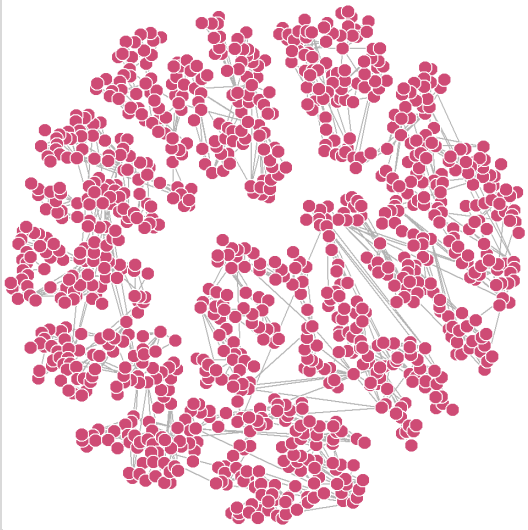
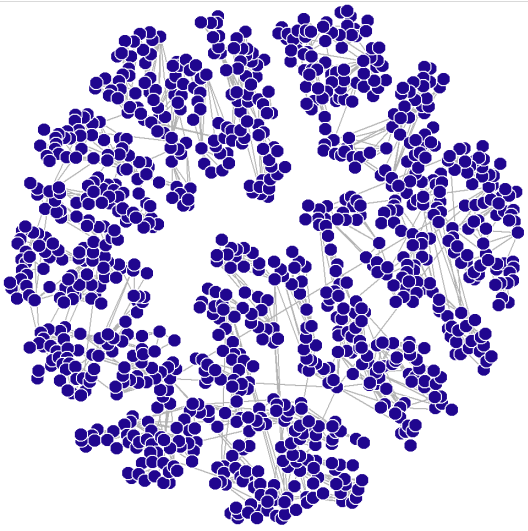
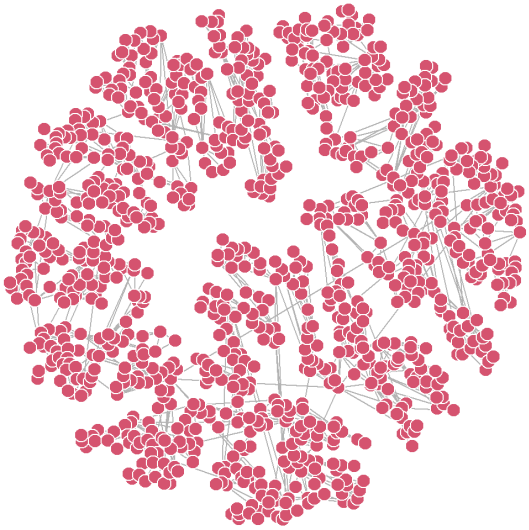
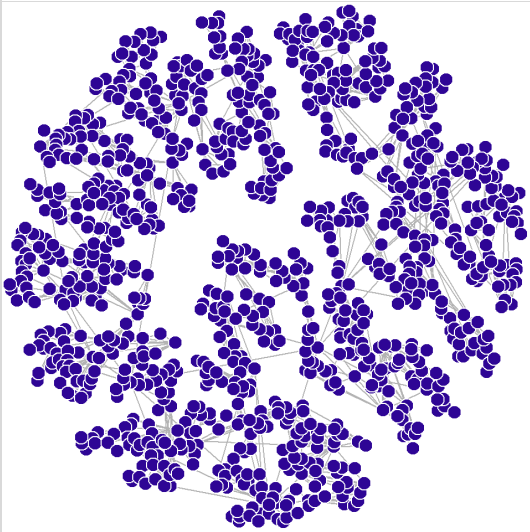
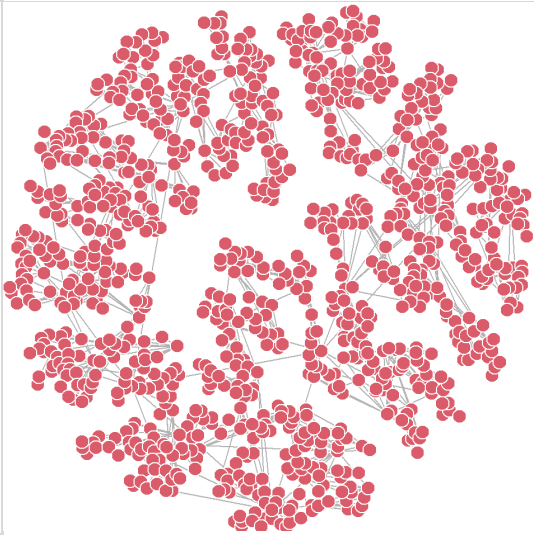
	Visibility graph	*Horizontal visibility graph*
Coef= 0.3:		
Coef= 0.4:		
Coef= 0.5:		

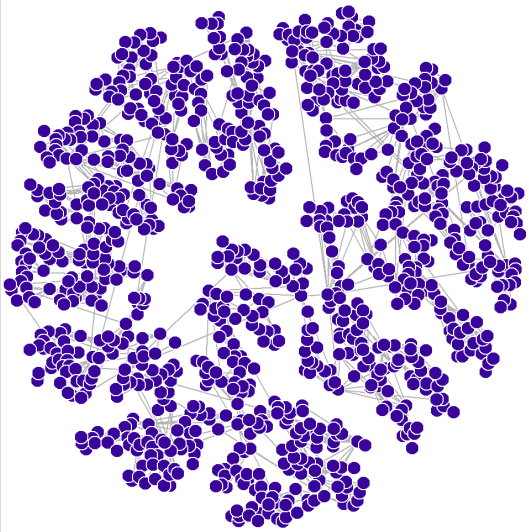
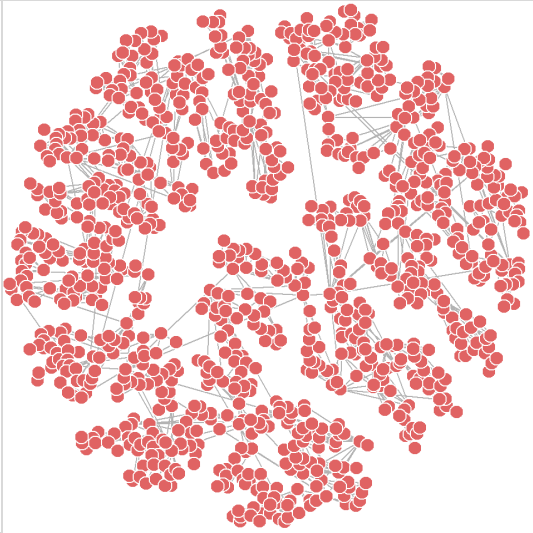
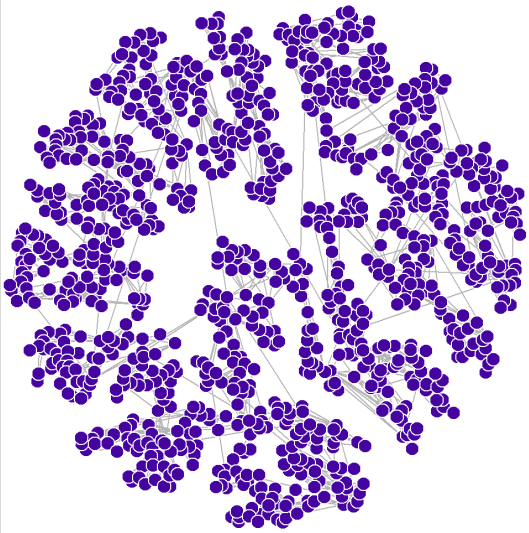
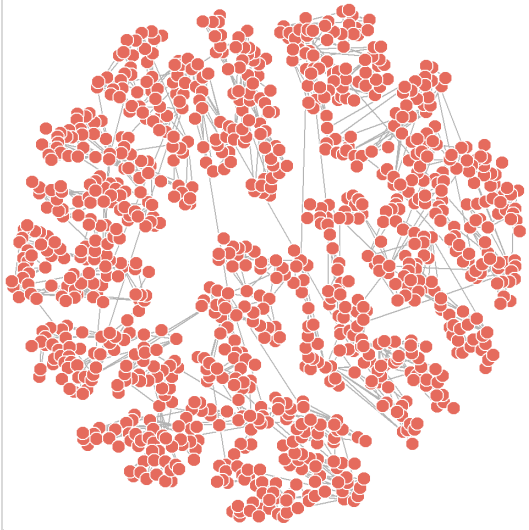
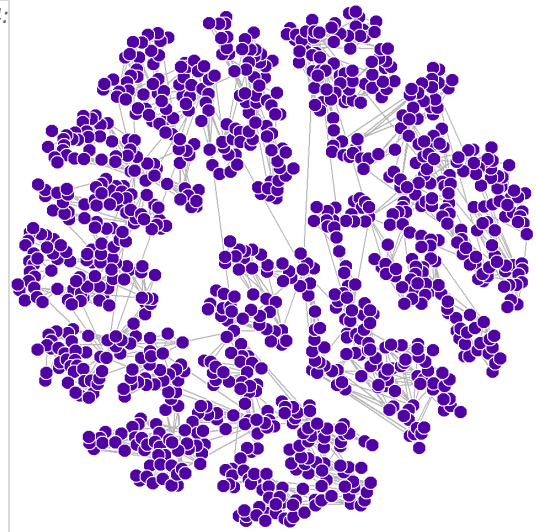
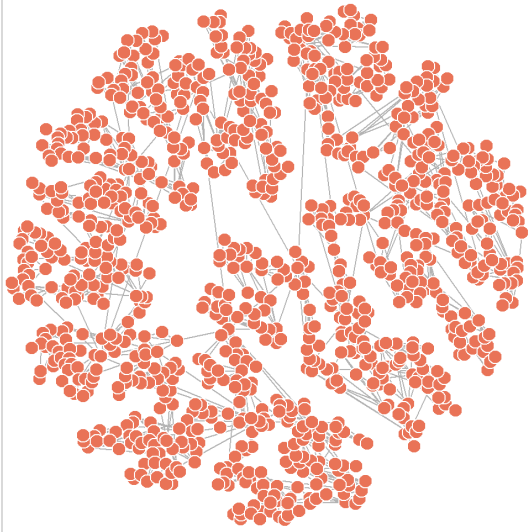
	Visibility graph	*Horizontal visibility graph*
Coef= 0.6:		
Coef= 0.7:		
Coef= 0.8:		

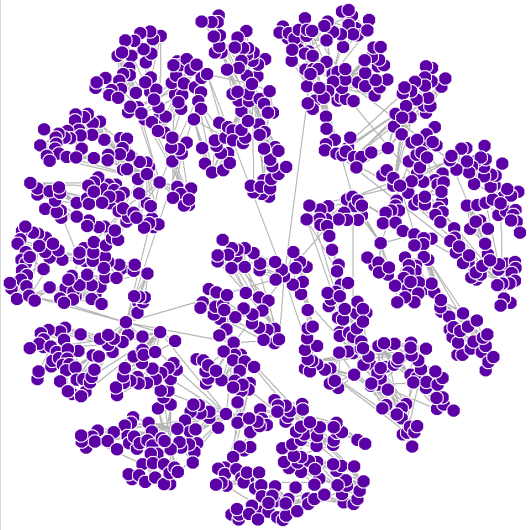
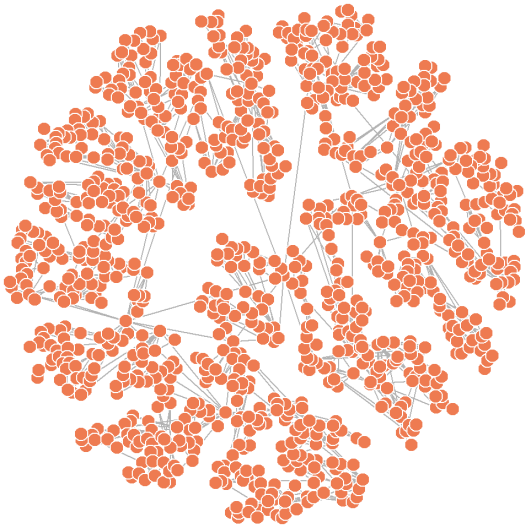
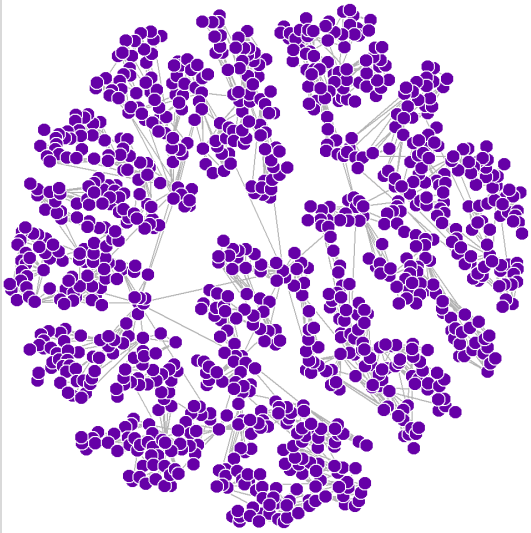
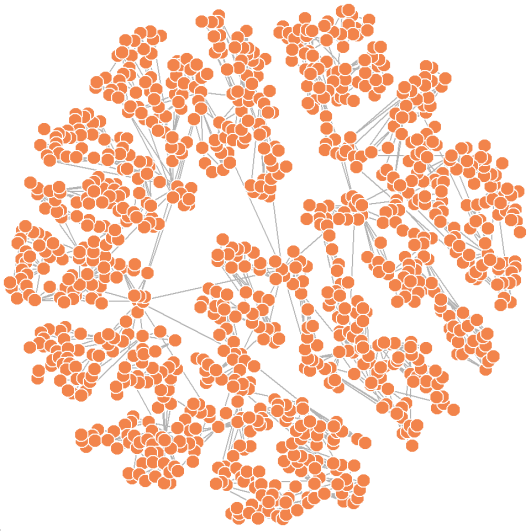
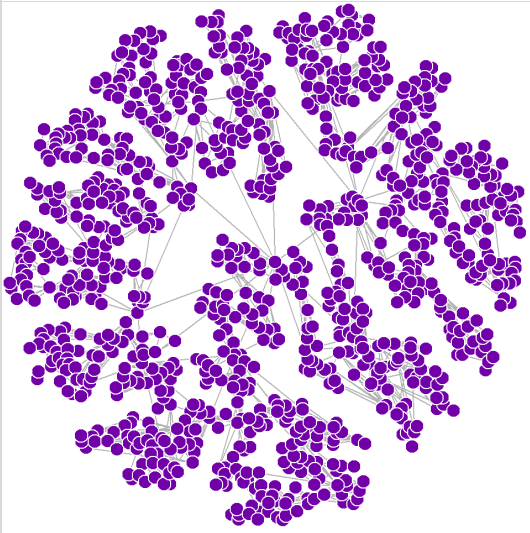
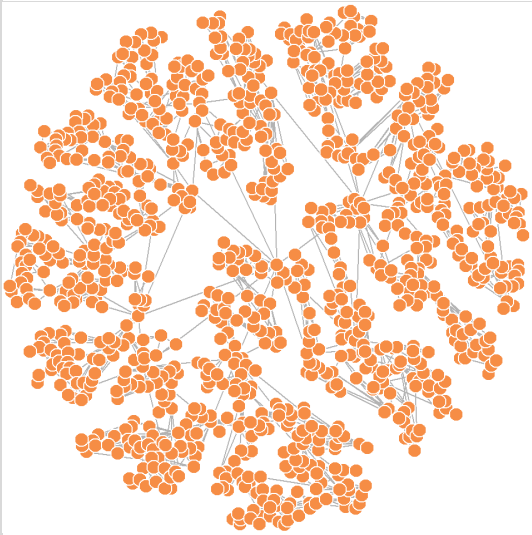
	Visibility graph	*Horizontal visibility graph*
Coef= 0.9:		

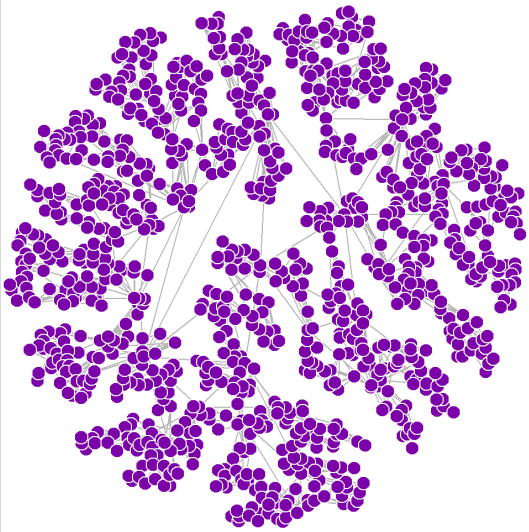
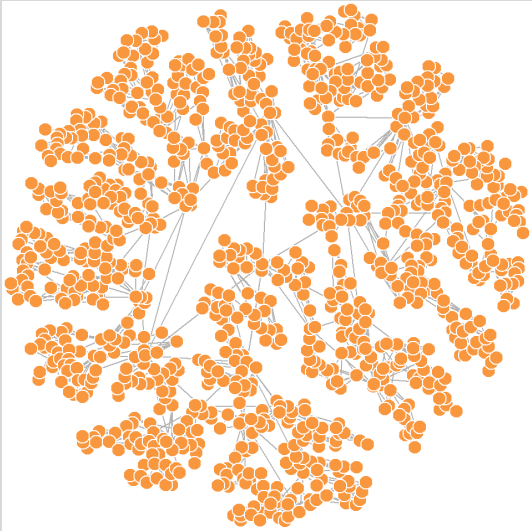
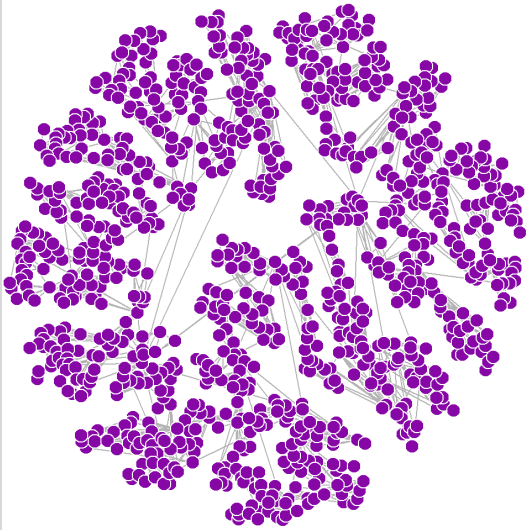
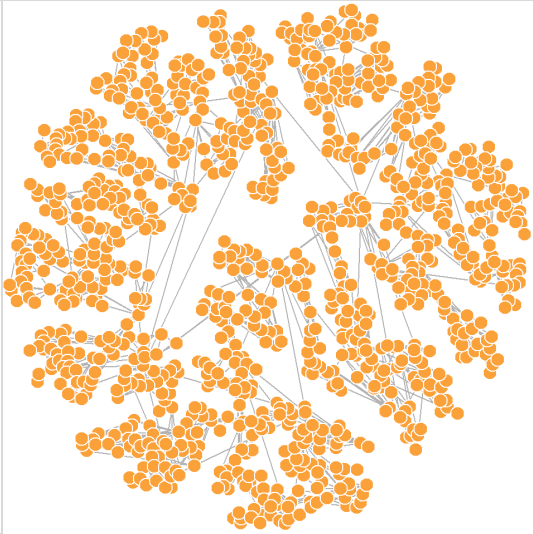
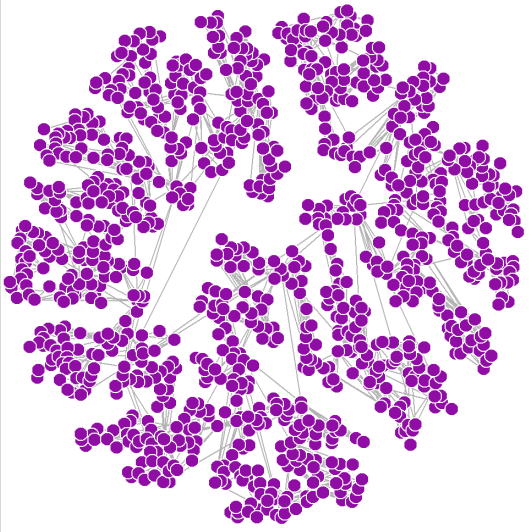
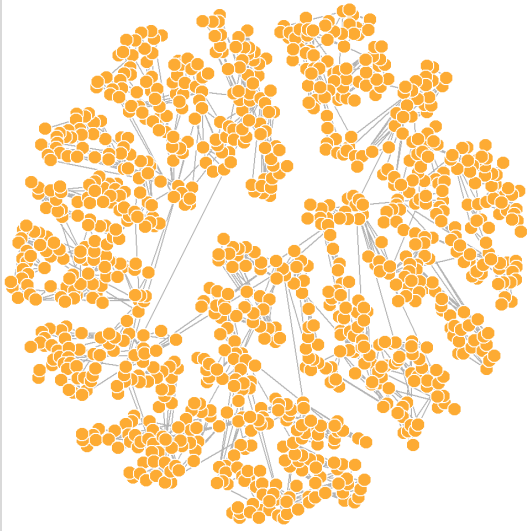
F: : Line plot of realizations generate by an $\boldsymbol{ARMA(1,1)}$ with different parameters for $\boldsymbol{AR}$ $*$

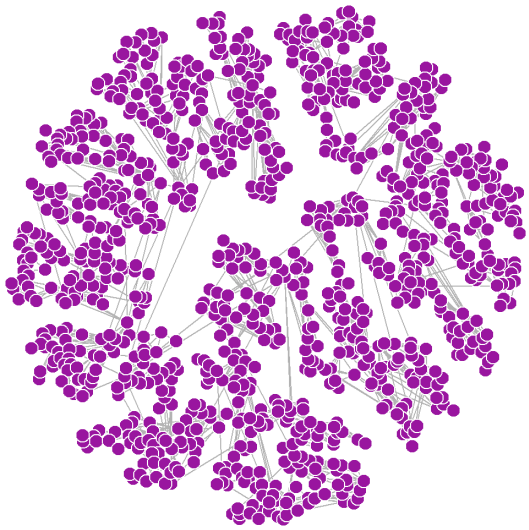
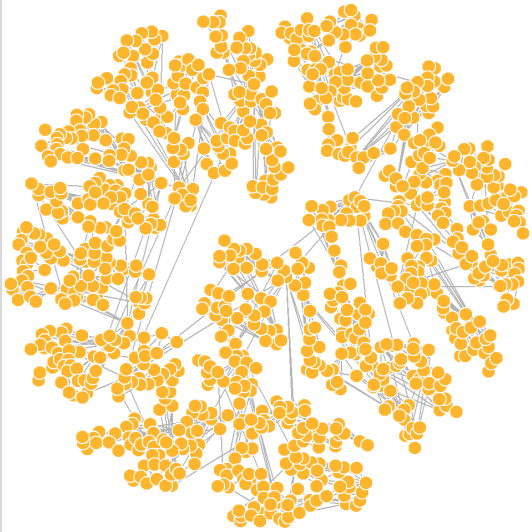
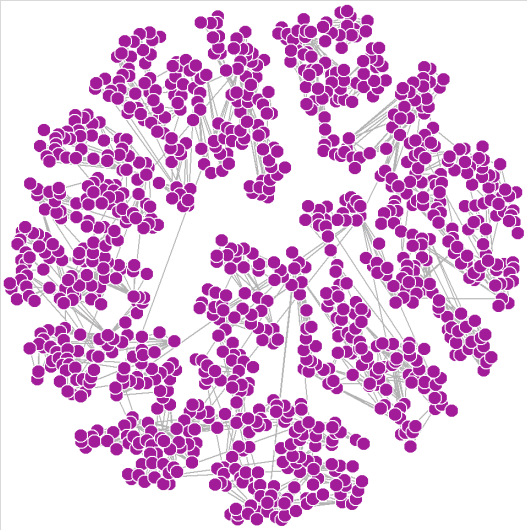
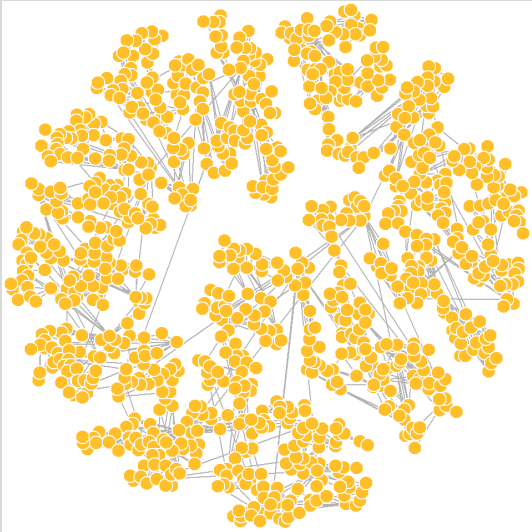
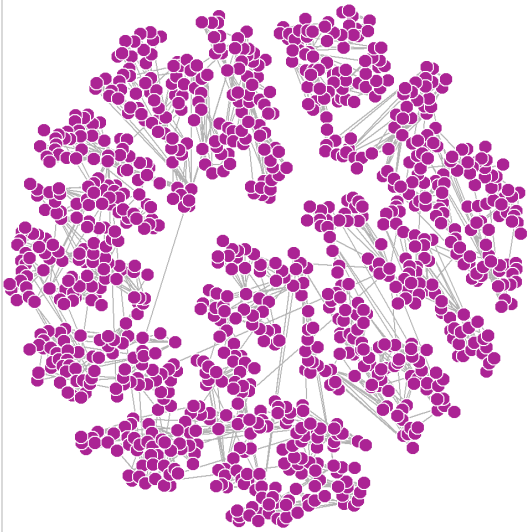
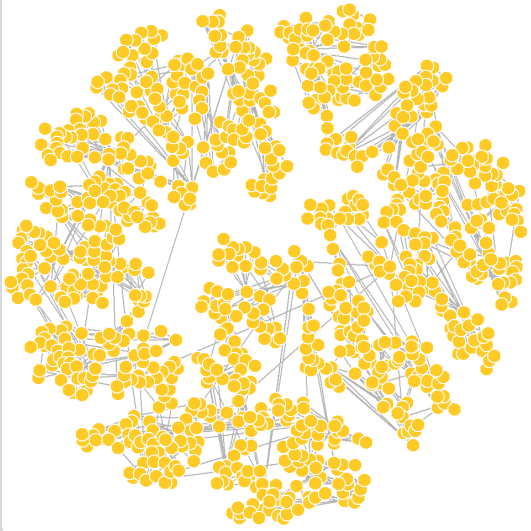
The processes' autocorrelation coefficient is shortened to "coef" in the table

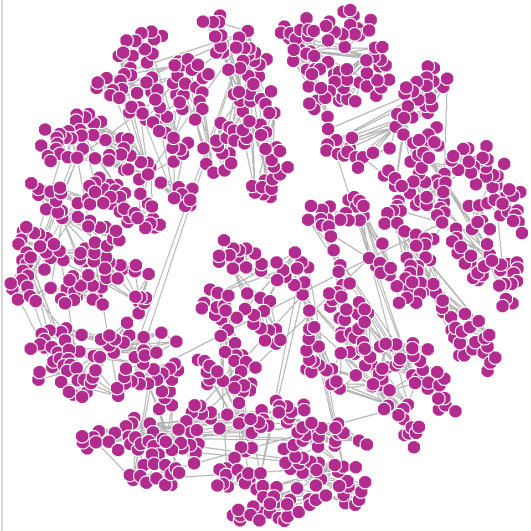
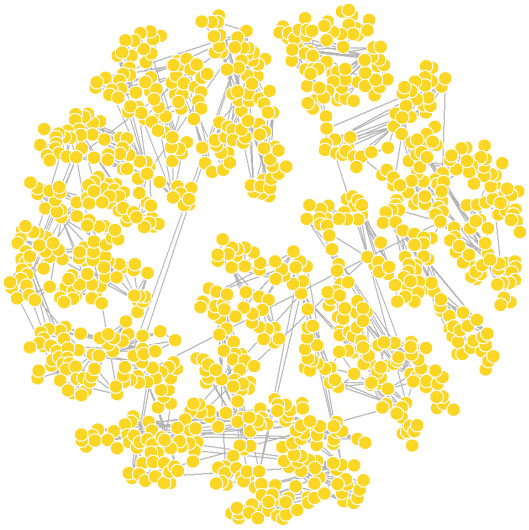
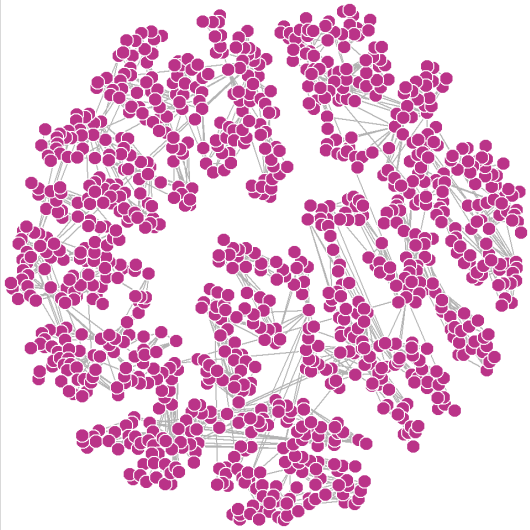
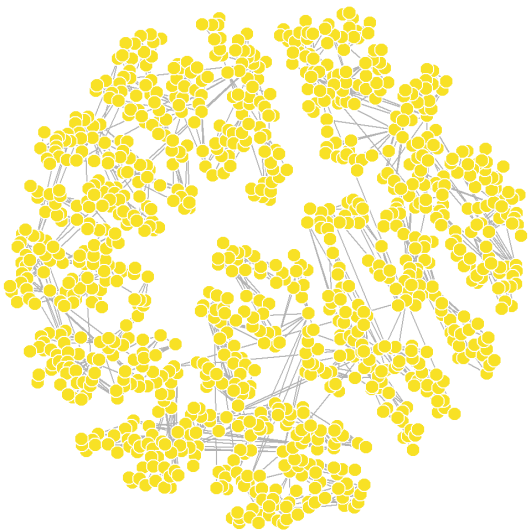
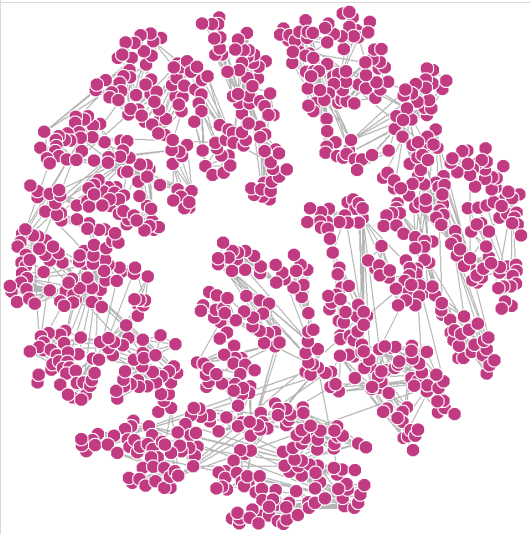
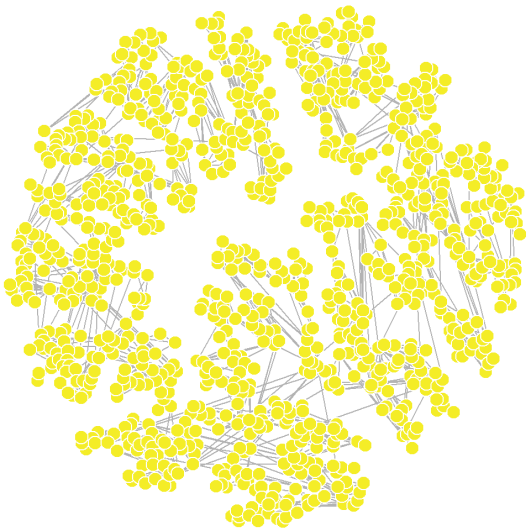
	Visibility graph	*Horizontal visibility graph*
Coef=-0.9:		
Coef=-0.8:		
Coef=-0.7:		

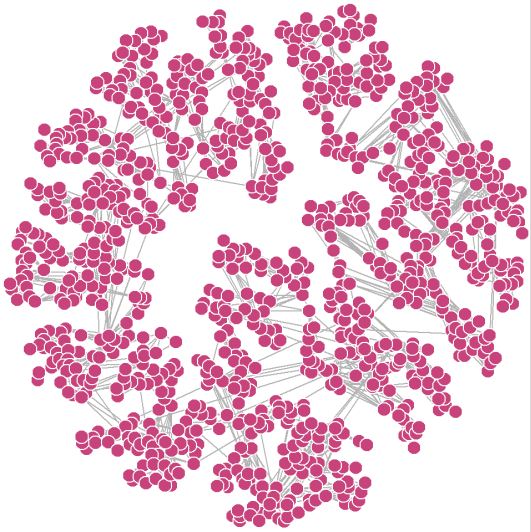
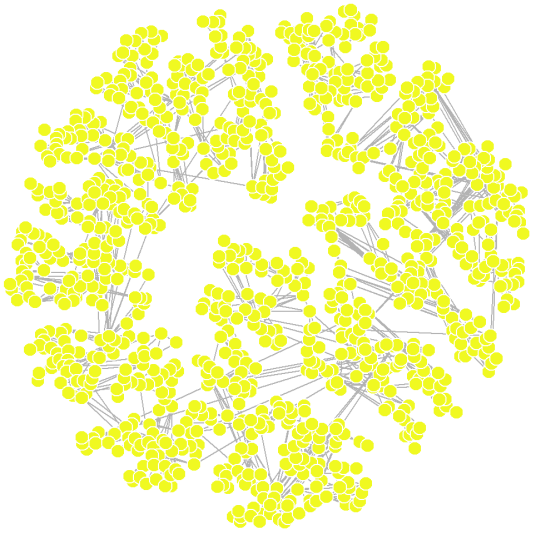
	Visibility graph	*Horizontal visibility graph*
Coef=-0.6:		
Coef=-0.5:		
Coef=-0.4:		

	Visibility graph	*Horizontal visibility graph*
Coef=-0.3:		
Coef=-0.2:		
Coef=-0.1:		

	Visibility graph	*Horizontal visibility graph*
Coef=0:		
Coef= 0.1:		
Coef= 0.2:		

	Visibility graph	*Horizontal visibility graph*
Coef= 0.3:		
Coef= 0.4:		
Coef= 0.5:		

	Visibility graph	*Horizontal visibility graph*
Coef= 0.6:		
Coef= 0.7:		
Coef= 0.8:		

	Visibility graph	*Horizontal visibility graph*
Coef= 0.9:		

*Autocorrelation coefficient for $MA(1)$ is constant and set at 0.1

G: OLS regression analysis

Statistics of both visibility- and horizontal visibility graphs converted from a white noise process

Average shortest path length – Visibility graph:

Lower boundary: $ASPL = -2.24 + 1.26\, ln(N)$

Regression Statistics								
Multiple R	0.9955							
R Square	0.9911							
Adjusted R Square	0.9908							
Standard Error	0.1033							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	33.2258	33.2258	3112.7769	0.0000			
Residual	28	0.2989	0.0107					
Total	29	33.5247						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	-2.2446	0.1611	-13.9298	0.0000	-2.5747	-1.9145	-2.5747	-1.9145
ln(N)	1.2586	0.0226	55.7923	0.0000	1.2124	1.3048	1.2124	1.3048

Upper boundary: $ASPL = -0.5 + 1.33\, ln(N)$

Regression Statistics								
Multiple R	0.9705							
R Square	0.9419							
Adjusted R Square	0.9398							
Standard Error	0.2853							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	36.9069	36.9069	453.5774	0.0000			
Residual	28	2.2783	0.0814					
Total	29	39.1852						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	-0.4985	0.4449	-1.1205	0.2720	-1.4098	0.4128	-1.4098	0.4128
ln(N)	1.3265	0.0623	21.2974	0.0000	1.1989	1.4541	1.1989	1.4541

Average shortest path length (HVG)

Lower boundary: $ASPL = -1.9 + 1.28\, ln(N)$

Regression Statistics								
Multiple R	0.9954							
R Square	0.9909							
Adjusted R Square	0.9906							
Standard Error	0.1066							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	34.6042	34.6042	3046.1287	0.0000			
Residual	28	0.3181	0.0114					
Total	29	34.9223						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	-1.8815	0.1662	-11.3180	0.0000	-2.2220	-1.5409	-2.2220	-1.5409
ln(N)	1.2845	0.0233	55.1917	0.0000	1.2368	1.3321	1.2368	1.3321

Upper boundary: $ASPL = 0.22 + 1.31\, ln(N)$

Regression Statistics								
Multiple R	0.9726							
R Square	0.9460							
Adjusted R Square	0.9441							
Standard Error	0.2702							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	35.7980	35.7980	490.4540	0.0000			
Residual	28	2.0437	0.0730					
Total	29	37.8417						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	0.2167	0.4214	0.5142	0.6111	-0.6464	1.0798	-0.6464	1.0798
ln(N)	1.3064	0.0590	22.1462	0.0000	1.1856	1.4273	1.1856	1.4273

Assortativity (VG):

Lower boundary: $ASS = -0.26 + 0.045\ ln(N)$

Regression Statistics								
Multiple R	0.9687							
R Square	0.9384							
Adjusted R Square	0.9362							
Standard Error	0.0099							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	0.0415	0.0415	426.4266	0.0000			
Residual	28	0.0027	0.0001					
Total	29	0.0442						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	-0.2620	0.0154	-17.0381	0.0000	-0.2935	-0.2305	-0.2935	-0.2305
ln(N)	0.0445	0.0022	20.6501	0.0000	0.0400	0.0489	0.0400	0.0489

Upper boundary: $ASS = 0.086 + 0.005\ ln(N)$

Regression Statistics								
Multiple R	0.5573							
R Square	0.3105							
Adjusted R Square	0.2859							
Standard Error	0.0067							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	0.0006	0.0006	12.6110	0.0014			
Residual	28	0.0013	0.0000					
Total	29	0.0018						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	0.0855	0.0105	8.1519	0.0000	0.0640	0.1069	0.0640	0.1069
ln(N)	0.0052	0.0015	3.5512	0.0014	0.0022	0.0082	0.0022	0.0082

Assortativity (HVG)

Lower boundary: $ASS = -0.25 + 0.052\ ln(N)$

Regression Statistics								
Multiple R	0.9753							
R Square	0.9513							
Adjusted R Square	0.9495							
Standard Error	0.0102							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	0.0569	0.0569	546.6081	0.0000			
Residual	28	0.0029	0.0001					
Total	29	0.0598						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	-0.2492	0.0159	-15.6673	0.0000	-0.2818	-0.2166	-0.2818	-0.2166
ln(N)	0.0521	0.0022	23.3797	0.0000	0.0475	0.0566	0.0475	0.0566

Upper boundary: $ASS = 0.13 + 0.01\ ln(N)$

Regression Statistics								
Multiple R	0.72212138							
R Square	0.52145929							
Adjusted R Square	0.50436855							
Standard Error	0.00796364							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	0.00193501	0.00193501	30.5112185	6.6418E-06			
Residual	28	0.00177575	6.342E-05					
Total	29	0.00371076						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	0.12812089	0.01242066	10.3151415	4.8306E-11	0.10267832	0.15356347	0.10267832	0.15356347
ln(N)	0.00960508	0.00173889	5.52369609	6.6418E-06	0.00604313	0.01316702	0.00604313	0.01316702

Mean degree (VG)

Lower boundary: $D = 3.84 + 0.106\, ln(N)$

Regression Statistics								
Multiple R	0.944							
R Square	0.891							
Adjusted R Square	0.887							
Standard Error	0.032							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	0.237296	0.237296	229.753731	0.000000			
Residual	28	0.028919	0.001033					
Total	29	0.266215						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	3.844	0.0501	76.6837	0.0000	3.7410	3.9464	3.7410	3.9464
ln(N)	0.106	0.0070	15.1576	0.0000	0.0920	0.1207	0.0920	0.1207

Upper boundary: $D = 4.799 - 0.004\, ln(N)$
(not an adequate result!)

Regression Statistics								
Multiple R	0.181							
R Square	0.033							
Adjusted R Square	-0.002							
Standard Error	0.019							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	0.000	0.000	0.947	0.339			
Residual	28	0.010	0.000					
Total	29	0.010						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	4.799	0.030	161.583	0.000	4.738	4.859	4.738	4.859
ln(N)	-0.004	0.004	-0.973	0.339	-0.013	0.004	-0.013	0.004

Mean degree (HVG):

Lower boundary: $D = 3.45 + 0.07ln(N)$

Regression Statistics								
Multiple R	0.9261							
R Square	0.8576							
Adjusted R Square	0.8525							
Standard Error	0.0246							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	0.1018	0.1018	168.6236	0.0000			
Residual	28	0.0169	0.0006					
Total	29	0.1187						
	Coefficients	*tandard Erro*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	3.4486	0.0383	89.9958	0.0000	3.3701	3.5271	3.3701	3.5271
ln(N)	0.0697	0.0054	12.9855	0.0000	0.0587	0.0807	0.0587	0.0807

Upper boundary: $D = 3.81 + 0.025\ ln(N)$

Regression Statistics								
Multiple R	0.9024							
R Square	0.8144							
Adjusted R Square	0.8078							
Standard Error	0.0101							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	0.0126	0.0126	122.8636	0.0000			
Residual	28	0.0029	0.0001					
Total	29	0.0154						
	Coefficients	*tandard Erro*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	3.8074	0.0158	241.4957	0.0000	3.7751	3.8397	3.7751	3.8397
ln(N)	0.0245	0.0022	11.0844	0.0000	0.0199	0.0290	0.0199	0.0290

Transitivity (VG):

Lower boundary: $Ln(TRA) = -0.93 - 0.0006\ ln(N)$
(not an adequate result)

Regression Statistics								
Multiple R	0.0766							
R Square	0.0059							
Adjusted R Square	-0.0296							
Standard Error	0.0065							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	0.0000	0.0000	0.1652	0.6875			
Residual	28	0.0012	0.0000					
Total	29	0.0012						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	-0.9260	0.0102	-91.1155	0.0000	-0.9468	-0.9052	-0.9468	-0.9052
ln(N)	-0.0006	0.0014	-0.4065	0.6875	-0.0035	0.0023	-0.0035	0.0023

Upper boundary: $Ln(TRA) = -0.59 - 0.04\ ln(N)$

Regression Statistics								
Multiple R	0.9600							
R Square	0.9216							
Adjusted R Square	0.9188							
Standard Error	0.0097							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	0.0308	0.0308	329.0316	0.0000			
Residual	28	0.0026	0.0001					
Total	29	0.0334						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	-0.5929	0.0151	-39.2818	0.0000	-0.6238	-0.5619	-0.6238	-0.5619
ln(N)	-0.0383	0.0021	-18.1392	0.0000	-0.0427	-0.0340	-0.0427	-0.0340

Transitivity (HVG):

Lower boundary: $Ln(TRA) = -1.09 - 0.002\ ln(N)$
(not an adequate result)

Regression Statistics								
Multiple R	0.4409							
R Square	0.1944							
Adjusted R Square	0.1656							
Standard Error	0.0037							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	0.0001	0.0001	6.7573	0.0147			
Residual	28	0.0004	0.0000					
Total	29	0.0005						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	-1.0894	0.0057	-191.0905	0.0000	-1.1011	-1.0778	-1.1011	-1.0778
ln(N)	-0.0021	0.0008	-2.5995	0.0147	-0.0037	-0.0004	-0.0037	-0.0004

Upper boundary: $Ln(TRA) = -0.82 - 0.034\ ln(N)$

Regression Statistics								
Multiple R	0.9496							
R Square	0.9018							
Adjusted R Square	0.8983							
Standard Error	0.0096							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	0.0235	0.0235	257.1999	0.0000			
Residual	28	0.0026	0.0001					
Total	29	0.0261						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	-0.8168	0.0149	-54.7772	0.0000	-0.8474	-0.7863	-0.8474	-0.7863
ln(N)	-0.0335	0.0021	-16.0375	0.0000	-0.0378	-0.0292	-0.0378	-0.0292

Normalized degree centrality (VG)

Lower boundary: $Ln(DC) = 0.6 - 0.72\ ln(N)$

Regression Statistics								
Multiple R	0.9947							
R Square	0.9894							
Adjusted R Square	0.9890							
Standard Error	0.0650							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	11.0151	11.0151	2603.5569	0.0000			
Residual	28	0.1185	0.0042					
Total	29	11.1336						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	0.6014	0.1014	5.9277	0.0000	0.3936	0.8092	0.3936	0.8092
ln(N)	-0.7247	0.0142	-51.0251	0.0000	-0.7538	-0.6956	-0.7538	-0.6956

Upper boundary: $Ln(DC) = 1.97 - 0.82\ ln(N)$

Regression Statistics								
Multiple R	0.9954							
R Square	0.9908							
Adjusted R Square	0.9904							
Standard Error	0.0685							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	14.1129	14.1129	3004.0655	0.0000			
Residual	28	0.1315	0.0047					
Total	29	14.2445						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	1.9735	0.1069	18.4606	0.0000	1.7545	2.1925	1.7545	2.1925
ln(N)	-0.8203	0.0150	-54.8094	0.0000	-0.8509	-0.7896	-0.8509	-0.7896

Normalized degree centrality (HVG)

Lower boundary: $Ln(DC) = 0.37 - 0.72\ ln(N)$

Regression Statistics								
Multiple R	0.9952							
R Square	0.9904							
Adjusted R Square	0.9900							
Standard Error	0.0616							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	10.9607	10.9607	2885.6824	0.0000			
Residual	28	0.1064	0.0038					
Total	29	11.0670						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	0.3730	0.0961	3.8802	0.0006	0.1761	0.5699	0.1761	0.5699
ln(N)	-0.7229	0.0135	-53.7185	0.0000	-0.7505	-0.6953	-0.7505	-0.6953

Upper boundary: $Ln(DC) = 1.89 - 0.84\ ln(N)$

Regression Statistics								
Multiple R	0.9935							
R Square	0.9871							
Adjusted R Square	0.9867							
Standard Error	0.0832							
Observations	30							
ANOVA								
	df	*SS*	*MS*	*F*	*Significance F*			
Regression	1	14.8655	14.8655	2145.6388	0.0000			
Residual	28	0.1940	0.0069					
Total	29	15.0594						
	Coefficients	*Standard Error*	*t Stat*	*P-value*	*Lower 95%*	*Upper 95%*	*Lower 95.0%*	*Upper 95.0%*
Intercept	1.8911	0.1298	14.5670	0.0000	1.6252	2.1570	1.6252	2.1570
ln(N)	-0.8419	0.0182	-46.3210	0.0000	-0.8791	-0.8046	-0.8791	-0.8046

H: R- code

White noise process – different sample sizes

```
rm(list = ls(all=TRUE) )
 setwd(dirname(rstudioapi::getActiveDocumentContext()$path))
 dev.off()

 library(igraph)
 source("VG_HVG.r")

 #_____________________________________________________________________
 # Generate data,transform to network & calculate statistics
 #---------------------------------------------------------------------
 # Variables & storage:

 size=seq(100,3000,by=100)
 sim_num=100

 name_WN=c("White noice")
 name_data=c(paste("WN",size))
 name_sim=c(paste("sim",seq(1,sim_num,1)))
 name_nstat=c("VG: Average geodistic path","HVG: Average geodistic path",
              "VG: Assortativity","HVG: Assortativity",
              "VG: Transitivity","HVG: Transitivity",
              "VG: Mean degree","HVG: Mean degree",
              "VG: Degree centrality","HVG: Degree centrality",
              "VG: PL p-value", "HVG: PL p-value")
 name_dstat=c("Mean","Standard deviation","Maximum","Minimum")


 col_rainbow=rainbow(2*length(size))
 col_VG=col_rainbow[-(1:length(size))]
 col_HVG=rev(col_rainbow)
 col_HVG=col_HVG[-(1:length(size))]
 col_HVG=rev(col_HVG)
 col_graph=c("grey80","dodgerblue4","grey80")


 DATA_WN=vector("list",length(size))
 DSTAT_WN=vector("list",length(size))
 NSTAT_WN=vector("list",length(size))

 for (j in 1:length(size)) {
   #--------------------------------------------------------------------
   # Generate White noice
   data_WN=vector()

   dstat_WN=matrix(nrow = sim_num,ncol = length(name_dstat))
   colnames(dstat_WN)=name_dstat
   rownames(dstat_WN)=name_sim

   nstat_WN=matrix(ncol = length(name_nstat),nrow = sim_num)
   colnames(nstat_WN)=name_nstat
   rownames(nstat_WN)=name_sim

   for (i in 1:sim_num) {
     #.....................................................
     # Generate data:
     wn=rnorm(size[j],mean=0,sd=1)
```

```
     wn=as.vector(wn)
     data_WN=cbind(data_WN,wn)

     dstat_WN[i,1]=mean(wn)
     dstat_WN[i,2]=sd(wn)
     dstat_WN[i,3]=max(wn)
     dstat_WN[i,4]=min(wn)

     #.....................................................
     # Generate network:
     imatVG_WN=visi(wn)
     imatHVG_WN=visiH(wn)
     netVG_WN=graph_from_adjacency_matrix(imatVG_WN)
     netHVG_WN=graph_from_adjacency_matrix(imatHVG_WN)
     #.....................................................
     # Calculate network statistics:
     degDisVG_WN=vector()
     degDisHVG_WN=vector()
     #.....................................................
     # Average geodistic path:
     nstat_WN[i,1]=mean_distance(netVG_WN, directed = FALSE, unconnected =
FALSE)
     nstat_WN[i,2]=mean_distance(netHVG_WN, directed = FALSE, unconnected =
FALSE)
     #.....................................................
     # Assortativity:
     nstat_WN[i,3]=assortativity.degree(netVG_WN, directed=FALSE)
     nstat_WN[i,4]=assortativity.degree(netHVG_WN, directed=FALSE)
     #.....................................................
     # Global transitivity:
     nstat_WN[i,5]=transitivity(netVG_WN, type="global", vids = NULL,
                                weights = NULL, isolates = "zero" )
     nstat_WN[i,6]=transitivity(netHVG_WN, type="global", vids = NULL,
                                weights = NULL, isolates = "zero" )
     #.....................................................
     # Mean degree:
     degVG=degree(netVG_WN,v=V(netVG_WN),mode = "in") # change mode
betweeen: "in","out","all","total"
     nstat_WN[i,7]=mean(degVG)

     degHVG=degree(netHVG_WN,v=V(netHVG_WN),mode = "in") # change mode
betweeen: "in","out","all","total"
     nstat_WN[i,8]=mean(degHVG)
     #.....................................................
     # Cumulative degree distribution:
     degDisVG_WN=degree.distribution(netVG_WN, cumulative = TRUE, mode="in"
)
     degDisHVG_WN=degree.distribution(netHVG_WN, cumulative = TRUE,
mode="in" )
     #.....................................................
     # Degreee centrality
     VG_centDeg=centr_degree(netVG_WN, mode = "in", loops = FALSE,
normalized = TRUE)
     nstat_WN[i,9]=VG_centDeg[[2]]

     HVG_centDeg=centr_degree(netHVG_WN, mode = "in", loops = FALSE,
normalized = TRUE)
     nstat_WN[i,10]=HVG_centDeg[[2]]
     #.....................................................
     # Test for Power Law distribution:
```

```
     VG_PowLaw=fit_power_law(degDisVG_WN, xmin=NULL, force.continuous =
TRUE)
     nstat_WN[i,11]=VG_PowLaw[[6]]

     HVG_PowLaw=fit_power_law(degDisHVG_WN, xmin=NULL, force.continuous =
TRUE)
     nstat_WN[i,12]=HVG_PowLaw[[6]]
   }

   colnames(data_WN)=name_sim
   DATA_WN[[j]]=data_WN
   DSTAT_WN[[j]]=dstat_WN
   NSTAT_WN[[j]]=nstat_WN

   write.table(data_WN, file=paste("WN_Data",j,".txt"), col.names=TRUE)
   write.table(dstat_WN,file =paste( "WN_Descriptive stat",j,".txt"),
col.names = TRUE)
   write.table(nstat_WN, file=paste("WN_Network stat",j,".txt"),col.names =
TRUE)
 }

#_____________________________________________________________________
# Generate visualization:
# statistics as boxplot and quartiles as line-chars
#---------------------------------------------------------------------

name_quartiles=c("lower q","median","upper q")
quartiles=vector("list",length(name_nstat))
SSR=matrix(nrow=length(name_nstat),ncol =length(name_quartiles))
colnames(SSR)=name_quartiles
rownames(SSR)=name_nstat

for (i in 1:length(name_nstat)) {
  input=matrix(ncol=length(size),nrow = sim_num)
  med_quantile=matrix(nrow=5,ncol=length(size))
  colnames(input)=c(paste(name_nstat[i],"size:",size))
  for (j in 1:length(size)) {
    stat=NSTAT_WN[[j]]
    input[,j]=stat[,i]
    med=quantile(input[,j])
    med_quantile[,j]=med
  }
  write.table(input,file=paste("WN_Network stat
collected",i,".txt"),col.names = TRUE)
  png(file=paste("WN_Boxplot",i,".png"),width = 1200,res = 72)
  boxplot(input, main=name_nstat[i], xlab="Size", xaxt="n",col=col_VG)
  axis(1,at=1:length(size),labels = size)
  dev.off()
  png(file=paste("WN_Boxplot2",i,".png"),width = 1200,res = 72)
  boxplot(input, main=name_nstat[i], xlab="Size", xaxt="n",col=col_HVG)
  axis(1,at=1:length(size),labels = size)
  dev.off()

  med_quantile=t(med_quantile)
  write.table(med_quantile, file=paste("Quantiles",i,".txt"))
  med_quantile=med_quantile[,-c(1,5)]
  colnames(med_quantile)=name_quartiles

  png(file=paste("WN_Graph",i,".png"),width = 1200,res = 72)
  matplot(med_quantile, type="l", col=col_graph, main=name_nstat[i],
          xlab="Size", ylab="",xaxt="n", lwd=2, lty=1)
```

```
  polygon(c((seq(1,length(size))),rev(seq(1,length(size)))),
          c(med_quantile[,3],rev(med_quantile[,2])),
          col="grey98",border=NA)
  polygon(c((seq(1,length(size))),rev(seq(1,length(size)))),
          c(med_quantile[,2],rev(med_quantile[,1])),
          col = "grey98",border=NA)
  par(new=TRUE)
  matplot(med_quantile, type="l", col=col_graph,
          xlab="", ylab="",xaxt="n", lwd=c(1,2,1), lty=1)
  axis(1,at=1:length(size),labels = size)
  par(new=FALSE)
  dev.off()

}
save.image(file="WN_workspace.RData")
```

$MA(1), AR(1)$ and $ARMA(1,1)$ prosesses – change in autocorrelation coefficient(s)
White noise used as benchmark

```
rm(list = ls(all=TRUE) )
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))


library(igraph)
library(viridis)
source("VG_HVG.r")

#_____________________________________________________________________
# Generate data,transform to network & calculate statistics
#---------------------------------------------------------------------
# Variables:
size=1000
sim_num=1
p=1
q=1
coef_p=seq(-0.9,0.9,by=0.1)
coef_q_one=seq(-0.9,0.9,0.1) # change in MA coefficient when running
coef_q=0.1  # constant MA coefficient when running multipe simulations

name_AR=paste("AR(1), c =",coef_p)
name_MA=paste("MA(1), c =",coef_q)
name_MA_one=paste("MA(1), c =",coef_q_one)
name_ARMA=paste("ARMA, c_p =",coef_p,", c_q=",coef_q)
name_WN=c("White noice")
name_nstat=c("VG: Average geodistic path","HVG: Average geodistic path",
             "VG: Assortativity","HVG: Assortativity",
             "VG: Transitivity","HVG: Transitivity",
             "VG: Mean degree","HVG: Mean degree",
             "VG: Degree centrality","HVG: Degree centrality",
             "VG: PL p-value", "HVG: PL p-value")
name_cstat=c("Deg","Close","Betw","Hub")
name_dstat=c("Mean","Standard deviation","Maximum","Minimum")

col_boxP_WN="grey50"
col_boxP_MA="grey80"
col_MA= topo.colors((length(coef_q_one))*2)
col_AR=viridis((length(coef_p))*2)
col_ARMA=plasma((length(coef_p))*2)
color_VG_MA=col_MA
color_HVG_MA=col_MA[-(1:19)]
color_VG_AR=col_AR
color_HVG_AR=col_AR[-(1:19)]
color_VG_ARMA=col_ARMA
color_HVG_ARMA=col_ARMA[-(1:19)]

if(sim_num<=1){

  seed=runif(1,101,199)

  #-------------------------------------------------------------------
  # White noice - reference
  #...................................................
  # Generate & display data:
  #
  data_WN=vector()
  set.seed(seed)
```

```
  wn=rnorm(size,mean=0,sd=1)
  wn=as.vector(wn)
  data_WN=wn
  write.table(data_WN,file = "data_WN.txt")

  dstat_WN=vector(length = length(name_dstat))
  names(dstat_WN)=name_dstat
  dstat_WN[1]=mean(wn)
  dstat_WN[2]=sd(wn)
  dstat_WN[3]=max(wn)
  dstat_WN[4]=min(wn)
  write.table(dstat_WN, file = "dstat_WN.txt")

  png(file=paste("WN","Graph",".png"), width=1200,res =72 )
  plot(data_WN, type = "l",col="grey10",main=name_WN,ylab = "",xlab = "")
  abline(h=0,col="grey90")
  dev.off()

  adj=min(data_WN)
  adj_WN=data_WN-adj
  png(file=paste("WN","adjBarplot",".png"),width=1200,res=72)
  barplot(adj_WN, main = "Adjusted barplot: WN", col = "black",
          names.arg = c(seq(1,size)))
  dev.off()
  png(file=paste("WN","Auto correlation
function",".png"),width=1200,res=72)
  acf(data_WN)
  dev.off()
  png(file=paste("WN","Partial auto correlation",".png"),width=1200,res=72)
  pacf(data_WN)
  dev.off()
  #.....................................................
  # Generate & display network:
  #
  imatVG_WN=visi(data_WN)
  imatHVG_WN=visiH(data_WN)
  netVG_WN=graph_from_adjacency_matrix(imatVG_WN)
  netHVG_WN=graph_from_adjacency_matrix(imatHVG_WN)
  png(file=paste("WN","VG",".png"),width = 1200,height = 1200,res=72)
  Q=qgraph(imatVG_WN, color="grey30", edge.color="grey70", vsize=2,
esize=1, labels=FALSE,
           border.color=c(rep("white",size)))
  title("Visibility graph: WN",line=3)
  dev.off()
  png(file=paste("WN","HVG",".png"),width = 1200,height = 1200,res=72)
  qgraph(imatHVG_WN,layout=Q$layout,color="grey60" ,edge.color="grey70",
vsize=2, esize=1, labels=FALSE,
         border.color=c(rep("white",size)))
  title("Horizontal visibility graph : WN", line=3)
  dev.off()

  png(file=paste("WN","Degree frequency VG",".png"), height=1200,res=72)
  degree_function(imatVG_WN,2)
  dev.off()
  png(file=paste("WN","Degree frequency HVG",".png"), height=1200,res=72)
  degree_function(imatHVG_WN,2)
  dev.off()
  #.....................................................
  # Calculate network statistics:
  #
  nstat_WN=vector(length = length(name_nstat))
```

```
  names(nstat_WN)=name_nstat

  cstatVG_WN=matrix(nrow=size, ncol =length(name_cstat))
  colnames(cstatVG_WN)=name_cstat
  cstatHVG_WN=matrix(nrow=size,ncol = length(name_cstat))
  colnames(cstatHVG_WN)=name_cstat
  #.............................................................
  # Average geodistic path:
  nstat_WN[1]=mean_distance(netVG_WN, directed = FALSE, unconnected =
FALSE)
  nstat_WN[2]=mean_distance(netHVG_WN, directed = FALSE, unconnected =
FALSE)
  #.............................................................
  # Assortativity:
  nstat_WN[3]=assortativity.degree(netVG_WN, directed=FALSE)
  nstat_WN[2]=mean_distance(netHVG_WN, directed = FALSE, unconnected =
FALSE)
  #.............................................................
  # Assortativity:
  nstat_WN[3]=assortativity.degree(netVG_WN, directed=FALSE)
  nstat_WN[4]=assortativity.degree(netHVG_WN, directed=FALSE)
  #.............................................................
  # Global transitivity:
  nstat_WN[5]=transitivity(netVG_WN, type="global", vids = NULL,
                           weights = NULL, isolates = "zero" )
  nstat_WN[6]=transitivity(netHVG_WN, type="global", vids = NULL,
                           weights = NULL, isolates = "zero" )
  #.............................................................
  # Mean degree:
  degVG=degree(netVG_WN,v=V(netVG_WN),mode = "in")
  nstat_WN[7]=mean(degVG)

  degHVG=degree(netHVG_WN,v=V(netHVG_WN),mode = "in")
  nstat_WN[8]=mean(degHVG)
  #.............................................................
  # Cumulative degree distribution:
  degDisVG_WN=degree.distribution(netVG_WN, cumulative = TRUE, mode="in" )
  degDisHVG_WN=degree.distribution(netHVG_WN, cumulative = TRUE, mode="in"
)
  #.............................................................
  # Degreee centrality
  VG_centDeg=centr_degree(netVG_WN, mode = "in", loops = FALSE, normalized
= TRUE)
  nstat_WN[9]=VG_centDeg[[2]]

  HVG_centDeg=centr_degree(netHVG_WN, mode = "in", loops = FALSE,
normalized = TRUE)
  nstat_WN[10]=HVG_centDeg[[2]]
  #.............................................................
  # Test for Power Law distribution:
  VG_PowLaw=fit_power_law(degDisVG_WN, xmin=NULL, force.continuous = TRUE)
  nstat_WN[11]=VG_PowLaw[[6]]

  HVG_PowLaw=fit_power_law(degDisHVG_WN, xmin=NULL, force.continuous =
TRUE)
  nstat_WN[12]=HVG_PowLaw[[6]]
  #.............................................................
  # Centrality: degree, closeness, betweenness & hub
  par(mfrow=c(4,2))
  cent_VG=centrality(netVG_WN)
  cstatVG_WN[,1]=cent_VG[[2]]
```

```
  cstatVG_WN[,2]=cent_VG[[3]]
  cstatVG_WN[,3]=cent_VG[[4]]
  hub_vg=hub.score(netVG_WN,scale = TRUE,weights = NULL)
  cstatVG_WN[,4]=hub_vg[[1]]

  cent_HVG=centrality(netHVG_WN)
  cstatHVG_WN[,1]=cent_HVG[[2]]
  cstatHVG_WN[,2]=cent_HVG[[3]]
  cstatHVG_WN[,3]=cent_HVG[[4]]
  hub_hvg=hub.score(netHVG_WN,scale = TRUE,weights = NULL)
  cstatHVG_WN[,4]=hub_hvg[[1]]

  #.............................................................
  # Share of geodesics having a given length
  spathVG=cent_VG[7]
  geodshare_VG=table(spathVG)
  png(file=paste("WN","VG-Geodistics",".png"),width=1200,res=72)
  col_bar=colorRampPalette (c(col_boxP_WN,"white"))
  col_barVG=col_bar(length(geodshare_VG))
  barplot(geodshare_VG, col = col_barVG, ylab="Number of geodistics",xlab =
"Geodistic distance",
          main="VG: Share of geodistics having a given length, WN")
  dev.off()

  spathHVG=cent_HVG[7]
  geodshare_HVG=table(spathHVG)
  png(file=paste("WN","HVG - Geodistics",".png"),width=1200,res=72)
  col_bar=colorRampPalette (c(col_boxP_WN,"white"))
  col_barHVG=col_bar(length(geodshare_HVG))
  col_barHVG=paste(col_barHVG,"50",sep="")
  barplot(geodshare_HVG, col =col_barHVG, ylab="Number of geodistics",xlab
= "Geodistic distance",
          main="HVG: Share of geodistics having a given length, WN")
  dev.off()

  write.table(nstat_WN, file = "nstat_WN.txt", col.names = TRUE)
  write.table(cstatVG_WN, file = "cstat_WN_VG.txt", col.names = TRUE)
  write.table(cstatHVG_WN, file = "cstat_WN_HVG.txt", col.names = TRUE)
  #-------------------------------------------------------------------
  # MAs:
  #.....................................................
  # Generate & display data:
  #
  data_MA=vector()

  nstat_MA=matrix(nrow=length(coef_q_one),ncol=length(name_nstat))
  colnames(nstat_MA)=name_nstat
  rownames(nstat_MA)=name_MA_one

  dstat_MA=matrix(nrow=length(coef_q_one),ncol = length(name_dstat))
  colnames(dstat_MA)=name_dstat
  rownames(dstat_MA)=name_MA_one

  cstatVG_MA=matrix(nrow=size, ncol =length(name_cstat))
  colnames(cstatVG_MA)=name_cstat
  cstatHVG_MA=matrix(nrow=size,ncol = length(name_cstat))
  colnames(cstatHVG_MA)=name_cstat

  for (i in 1:length(coef_q_one)) {
    #.................................................
    # Generate & display data:
```

```
    set.seed(seed)
    ma=arima.sim(n=size, model=list(ma=coef_q_one[i]))
    ma=as.vector(ma)
    data_MA=cbind(data_MA,ma)

    dstat_MA[i,1]=mean(ma)
    dstat_MA[i,2]=sd(ma)
    dstat_MA[i,3]=max(ma)
    dstat_MA[i,4]=min(ma)

    png(file=paste(name_MA_one[i],"- Graph.png"), width = 1200,res = 72)
    plot(ma, type = "l",col="grey10", main=name_MA_one[i],ylab = "",xlab =
"")
    abline(h=0,col="grey90")
    dev.off()

    adj=min(ma)
    adj_MA=ma-adj
    png(file=paste(name_MA_one[i],"- adjBarplot.png"), width = 1200,res =
72)
    barplot(adj_MA, main = paste("Adjusted barplot:",name_MA_one[i]), col =
"black",
            names.arg = c(seq(1,size)))
    dev.off()

    png(file=paste(name_MA_one[i],"- Auto correlation function.png"), width
= 1200,res = 72)
    acf(ma)
    dev.off()

    png(file=paste(name_MA_one[i],"- Partial auto correlation
function.png"), width = 1200,res = 72)
    pacf(ma)
    dev.off()
    #.....................................................
    # Generate & display network:
    imatVG_MA=visi(ma)
    imatHVG_MA=visiH(ma)
    netVG_MA=graph_from_adjacency_matrix(imatVG_MA)
    netHVG_MA=graph_from_adjacency_matrix(imatHVG_MA)

    png(file=paste(name_MA_one[i],"- VG.png"),width = 1200,height =
1200,res=72)
    qgraph(imatVG_MA,layout=Q$layout,color=color_VG_MA[i],
edge.color="grey70", vsize=2, esize=1, labels=FALSE,
           border.color=c(rep("white",size)))
    title(paste("Visibility graph:", name_MA_one[i]),line = 3)
    dev.off()

    png(file=paste(name_MA_one[i],"- HVG.png"),width = 1200,height =
1200,res=72)
    qgraph(imatHVG_MA,layout=Q$layout,color=color_HVG_MA[i],
edge.color="grey70", vsize=2, esize=1, labels=FALSE,
           border.color=c(rep("white",size)))
    title(paste("Horizontal visibility graph:", name_MA_one[i]),line = 3)
    dev.off()

    png(file=paste(name_MA_one[i],"- Degree frequency VG.png"),
height=1200,res=72)
    degree_function(imatVG_MA,2)
    dev.off()
```

```
    png(file=paste(name_MA_one[i],"- Degree frequency HVG.png"),
height=1200,res=72)
    degree_function(imatHVG_MA,2)
    dev.off()
    #.....................................................
    # Calculate network statistics:
    #.....................................................
    # Average geodistic path:
    nstat_MA[i,1]=mean_distance(netVG_MA, directed = FALSE, unconnected =
FALSE)
    nstat_MA[i,2]=mean_distance(netHVG_MA, directed = FALSE, unconnected =
FALSE)
    #.....................................................
    # Assortativity:
    nstat_MA[i,3]=assortativity.degree(netVG_MA, directed=FALSE)
    nstat_MA[i,4]=assortativity.degree(netHVG_MA, directed=FALSE)
    #.....................................................
    # Global transitivity:
    nstat_MA[i,5]=transitivity(netVG_MA, type="global", vids = NULL,
                               weights = NULL, isolates = "zero" )
    nstat_MA[i,6]=transitivity(netHVG_MA, type="global", vids = NULL,
                               weights = NULL, isolates = "zero" )
    #.....................................................
    # Mean degree:
    degVG=degree(netVG_MA,v=V(netVG_MA),mode = "in") # change mode
betweeen: "in","out","all","total"
    nstat_MA[i,7]=mean(degVG)

    degHVG=degree(netHVG_MA,v=V(netHVG_MA),mode = "in") # change mode
betweeen: "in","out","all","total"
    nstat_MA[i,8]=mean(degHVG)
    #.....................................................
    # Cumulative degree distribution:
    degDisVG=degree.distribution(netVG_MA, cumulative = TRUE, mode="in" )
    degDisHVG=degree.distribution(netHVG_MA, cumulative = TRUE, mode="in" )
    #.....................................................
    # Degreee centrality
    VG_centDeg=centr_degree(netVG_MA, mode = "in", loops = FALSE,
normalized = TRUE)
    nstat_MA[i,9]=VG_centDeg[[2]]

    HVG_centDeg=centr_degree(netHVG_MA, mode = "in", loops = FALSE,
normalized = TRUE)
    nstat_MA[i,10]=HVG_centDeg[[2]]
    #.....................................................
    # Test for Power Law distribution:
    VG_PowLaw=fit_power_law(degDisVG, xmin=NULL, force.continuous = TRUE)
    nstat_MA[i,11]=VG_PowLaw[[6]]

    HVG_PowLaw=fit_power_law(degDisHVG, xmin=NULL, force.continuous = TRUE)
    nstat_MA[i,12]=HVG_PowLaw[[6]]

    #.....................................................
    # Centrality: degree, closeness, betweenness & hub
    cent_VG=centrality(netVG_MA)
    cstatVG_MA[,1]=cent_VG[[2]]
    cstatVG_MA[,2]=cent_VG[[3]]
    cstatVG_MA[,3]=cent_VG[[4]]
    hub_vg=hub.score(netVG_MA,scale = TRUE,weights = NULL)
    cstatVG_MA[,4]=hub_vg[[1]]
```

```
    for (k in 1:4) {
      pl=cbind(cstatVG_MA[,k],cstatVG_WN[,k])
      matplot(pl, type = "l", col =c(colrs2[k],"grey60"),
ylab="Value",xlab="Node",
              main=paste("VG: Centrality,", name_cstat[k],"MA+WN, c
=",coef_q[i]))
    }

    cent_HVG=centrality(netHVG_MA)
    cstatHVG_MA[,1]=cent_HVG[[2]]
    cstatHVG_MA[,2]=cent_HVG[[3]]
    cstatHVG_MA[,3]=cent_HVG[[4]]
    hub_hvg=hub.score(netHVG_MA,scale = TRUE,weights = NULL)
    cstatHVG_MA[,4]=hub_hvg[[1]]
    for (k in 1:4) {
      pl=cbind(cstatHVG_MA[,k],cstatHVG_WN[,k])
      matplot(pl, type = "l", col =c(colrs2[4+k],"grey60"),
ylab="Value",xlab="node" ,
              main=paste("HVG: Centrality,", name_cstat[k],"MA+WN, c
=",coef_q[i]))
    }

    #.............................................................
    # Share of geodesics having a given length
    spathVG=cent_VG[7]
    geodshare_VG=table(spathVG)
    col_bar=colorRampPalette (c(color_VG_MA[i],"white"))
    col_barVG=col_bar(length(geodshare_VG))
    png(file=paste(name_MA_one[i],"- VG-Geodestic.png"),width=1200,res=72)
    barplot(geodshare_VG, col = col_barVG, ylab="Number of geodistics",xlab
= "Geodistic distance",
            main=paste("VG: Share of geodistics having a given length,
MA(1), c =",coef_q_one[i]))
    dev.off()

    spathHVG=cent_HVG[7]
    geodshare_HVG=table(spathHVG)
    col_bar=colorRampPalette (c(color_HVG_MA[i],"white"))
    col_barHVG=col_bar(length(geodshare_HVG))
    png(file=paste(name_MA_one[i],"- HVG -
Geodistics.png"),width=1200,res=72)
    barplot(geodshare_HVG, col =col_barHVG, ylab="Number of
geodistics",xlab = "Geodistic distance",
            main=paste("HVG: Share of geodistics having a given length,
MA(1), c =",coef_q_one[i]))
    dev.off()
  }
  colnames(data_MA)=name_MA_one

  write.table(data_MA,file="data_MA.txt")
  write.table(dstat_MA,file="dstat_MA.txt")
  write.table(nstat_MA, file = "nstat_MA.txt", col.names = TRUE)
  write.table(cstatVG_MA, file = "cstat_MA_VG.txt", col.names = TRUE)
  write.table(cstatHVG_MA, file = "cstat_MA_HVG.txt", col.names = TRUE)


  #-----------------------------------------------------------------
  # ARs with different coefficient:

  data_AR=vector()
```

```
  nstat_AR=matrix(nrow=length(coef_p),ncol=length(name_nstat))
  colnames(nstat_AR)=name_nstat
  rownames(nstat_AR)=name_AR

  dstat_AR=matrix(nrow=length(coef_p),ncol = length(name_dstat))
  colnames(dstat_AR)=name_dstat
  rownames(dstat_AR)=name_AR

  cstatVG_AR=matrix(nrow=size, ncol =length(name_cstat))
  colnames(cstatVG_AR)=name_cstat
  cstatHVG_AR=matrix(nrow=size,ncol = length(name_cstat))
  colnames(cstatHVG_AR)=name_cstat

  for (i in 1:length(coef_p)) {
    #.....................................................
    # Generate & display data:
    set.seed(seed)
    ar=arima.sim(n=size, model=list(ar=coef_p[i]))
    ar=as.vector(ar)
    data_AR=cbind(data_AR,ar)

    dstat_AR[i,1]=mean(ar)
    dstat_AR[i,2]=sd(ar)
    dstat_AR[i,3]=max(ar)
    dstat_AR[i,4]=min(ar)

    png(file=paste(name_AR[i],"Graph",".png"), width=1200,res =72 )
    plot(ar, type = "l",col="grey10", main=name_AR[i],ylab = "",xlab = "")
    abline(h=0,col="grey90")
    dev.off()

    adj=min(ar)
    adj_AR=ar-adj
    png(file=paste(name_AR[i],"adjBarplot",".png"), width=1200,res =72 )
    barplot(adj_AR, main = paste("Adjusted barplot:",name_AR[i]), col
="black",
            names.arg = c(seq(1,size)))
    dev.off()

    png(file=paste(name_AR[i],"Auto correlation function",".png"),
width=1200,res =72 )
    acf(ar)
    dev.off()

    png(file=paste(name_AR[i],"Patrial auto correlation function",".png"),
width=1200,res =72 )
    pacf(ar)
    dev.off()
    #.....................................................
    # Generate & display network:
    imatVG_AR=visi(ar)
    imatHVG_AR=visiH(ar)
    netVG_AR=graph_from_adjacency_matrix(imatVG_AR)
    netHVG_AR=graph_from_adjacency_matrix(imatHVG_AR)

    png(file=paste(name_AR[i],"VG",".png"),width = 1200,height =
1200,res=72)
    qgraph(imatVG_AR, layout=Q$layout,color=color_VG_AR[i],
edge.color="grey70", vsize=2, esize=1, labels=FALSE,
           border.color=c(rep("white",size)))
    title(paste("Visibility graph:", name_AR[i]),line = 3)
```

```
    dev.off()

    png(file=paste(name_AR[i],"HVG",".png"),width = 1200,height =
1200,res=72)
    qgraph(imatHVG_AR, layout=Q$layout, color=color_HVG_AR[i],
edge.color="grey70", vsize=2, esize=1, labels=FALSE,
           border.color=c(rep("white",size)))
    title(paste("Horizontal visibility graph",name_AR[i]),line=3)
    dev.off()

    png(file=paste(name_AR[i],"Degree frequency VG",".png"),
height=1200,res=72)
    degree_function(imatVG_AR,2)
    dev.off()
    png(file=paste(name_AR[i],"Degree frequency HVG",".png"),
height=1200,res=72)
    degree_function(imatHVG_AR,2)
    dev.off()
    #.....................................................
    # Calculate network statistics:
    #.....................................................
    # Average geodistic path:
    nstat_AR[i,1]=mean_distance(netVG_AR, directed = FALSE, unconnected =
FALSE)
    nstat_AR[i,2]=mean_distance(netHVG_AR, directed = FALSE, unconnected =
FALSE)
    #.....................................................
    # Assortativity:
    nstat_AR[i,3]=assortativity.degree(netVG_AR, directed=FALSE)
    nstat_AR[i,4]=assortativity.degree(netHVG_AR, directed=FALSE)
    #.....................................................
    # Global transitivity:
    nstat_AR[i,5]=transitivity(netVG_AR, type="global", vids = NULL,
                               weights = NULL, isolates = "zero" )
    nstat_AR[i,6]=transitivity(netHVG_AR, type="global", vids = NULL,
                               weights = NULL, isolates = "zero" )
    #.....................................................
    # Mean degree:
    degVG=degree(netVG_AR,v=V(netVG_AR),mode = "in")
    nstat_AR[i,7]=mean(degVG)

    degHVG=degree(netHVG_AR,v=V(netHVG_AR),mode = "in")
    nstat_AR[i,8]=mean(degHVG)
    #.....................................................
    # Cumulative degree distribution:
    degDisVG=degree.distribution(netVG_AR, cumulative = TRUE, mode="in" )
    degDisHVG=degree.distribution(netHVG_AR, cumulative = TRUE, mode="in" )
    #.....................................................
    # Degreee centrality
    VG_centDeg=centr_degree(netVG_AR, mode = "in", loops = FALSE,
normalized = TRUE)
    nstat_AR[i,9]=VG_centDeg[[2]]

    HVG_centDeg=centr_degree(netHVG_AR, mode = "in", loops = FALSE,
normalized = TRUE)
    nstat_AR[i,10]=HVG_centDeg[[2]]
    #.....................................................
    # Test for Power Law distribution:
    VG_PowLaw=fit_power_law(degDisVG, xmin=NULL, force.continuous = TRUE)
    nstat_AR[i,11]=VG_PowLaw[[6]]
```

```
    HVG_PowLaw=fit_power_law(degDisHVG, xmin=NULL, force.continuous = TRUE)
    nstat_AR[i,12]=HVG_PowLaw[[6]]
    #.............................................................
    # Centrality: degree, closeness, betweenness & hub
    par(mfrow=c(2,2))
    cent_VG=centrality(netVG_AR)
    cstatVG_AR[,1]=cent_VG[[2]]
    cstatVG_AR[,2]=cent_VG[[3]]
    cstatVG_AR[,3]=cent_VG[[4]]
    hub_vg=hub.score(netVG_AR,scale = TRUE,weights = NULL)
    cstatVG_AR[,4]=hub_vg[[1]]
    for (k in 1:4) {
      pl=cbind(cstatVG_AR[,k],cstatVG_WN[,k])
      matplot(pl, type = "l", col =c(colrs2[k],"grey60"),
ylab="Value",xlab="Node",
              main=paste("VG: Centrality,", name_cstat[k],"AR+WN, c
=",coef_p[i]))
    }

    cent_HVG=centrality(netHVG_AR)
    cstatHVG_AR[,1]=cent_HVG[[2]]
    cstatHVG_AR[,2]=cent_HVG[[3]]
    cstatHVG_AR[,3]=cent_HVG[[4]]
    hub_hvg=hub.score(netHVG_AR,scale = TRUE,weights = NULL)
    cstatHVG_AR[,4]=hub_hvg[[1]]
    for (k in 1:4) {
      pl=cbind(cstatHVG_AR[,k],cstatHVG_WN[,k])
      matplot(pl, type = "l", col =c(colrs2[4+k],"grey60"),
ylab="Value",xlab="node" ,
              main=paste("HVG: Centrality,", name_cstat[k],"AR+WN, c
=",coef_p[i]))
    }
    #.............................................................
    # Share of geodesics having a given length
    par(mfrow=c(2,1))
    spathVG=cent_VG[7]
    geodshare_VG=table(spathVG)
    col_bar=colorRampPalette (c(color_VG_AR[i],"white"))
    col_barVG=col_bar(length(geodshare_VG))
    png(file=paste(name_AR[i],"VG-Geodistics",".png"),width=1200,res=72)
    barplot(geodshare_VG, col = col_barVG, ylab="Number of geodistics",xlab
= "Geodistic distance",
            main=paste("VG: Share of geodistics having a given length,
AR(1), c=",coef_p[i]))
    dev.off()

    spathHVG=cent_HVG[7]
    geodshare_HVG=table(spathHVG)
    col_bar=colorRampPalette (c(color_HVG_AR[i],"white"))
    col_barVG=col_bar(length(geodshare_HVG))
    png(file=paste(name_AR[i],"HVG-Geodistics",".png"),width=1200,res=72)
    barplot(geodshare_HVG, col = col_barHVG, ylab="Number of
geodistics",xlab = "Geodistic distance",
            main=paste("HVG: Share of geodistics having a given length,
AR(1), c=",coef_p[i]))
    dev.off()
  }
  colnames(data_AR)=name_AR

  write.table(data_AR,file="data_AR.txt")
  write.table(dstat_AR,file="dstat_AR.txt")
```

```
  write.table(nstat_AR, file = "nstat_AR.txt", col.names = TRUE)
  write.table(cstatVG_AR, file="cstat_AR_VG.txt",col.names=TRUE)
  write.table(cstatHVG_AR, file="cstat_AR_HVG.txt",col.names=TRUE)

  #-----------------------------------------------------------------
  # ARMAs with different p coefficient and constant q

  data_ARMA=vector()

  nstat_ARMA=matrix(nrow=length(coef_p),ncol=length(name_nstat))
  colnames(nstat_ARMA)=name_nstat
  rownames(nstat_ARMA)=name_ARMA

  dstat_ARMA=matrix(nrow=length(coef_p),ncol = length(name_dstat))
  colnames(dstat_ARMA)=name_dstat
  rownames(dstat_ARMA)=name_ARMA

  cstatVG_ARMA=matrix(nrow=size, ncol =length(name_cstat))
  colnames(cstatVG_ARMA)=name_cstat
  cstatHVG_ARMA=matrix(nrow=size,ncol = length(name_cstat))
  colnames(cstatHVG_ARMA)=name_cstat

  for (i in 1:length(coef_p)) {
    #.....................................................
    # Generate & display data:
    set.seed(seed)
    arma=arima.sim(n=size, model=list(ar=coef_p[i],ma=coef_q))
    arma=as.vector(arma)
    data_ARMA=cbind(data_ARMA,arma)

    dstat_ARMA[i,1]=mean(arma)
    dstat_ARMA[i,2]=sd(arma)
    dstat_ARMA[i,3]=max(arma)
    dstat_ARMA[i,4]=min(arma)

    png(file=paste(name_ARMA[i],"Graph",".png"), width=1200,res =72 )
    plot(arma, type = "l",col="grey10", main=name_ARMA[i],ylab = "",xlab =
"")
    abline(h=0,col="grey80")
    dev.off()

    adj=min(arma)
    adj_ARMA=arma-adj
    png(file=paste(name_ARMA[i],"adjBarplot",".png"), width=1200,res =72 )
    barplot(adj_ARMA, main = paste("Adjusted barplot:",name_ARMA[i]), col =
"black",
            names.arg = c(seq(1,size)))
    dev.off()
    png(file=paste(name_ARMA[i],"Auto correlation function",".png"),
width=1200,res =72 )
    acf(arma)
    dev.off()
    png(file=paste(name_ARMA[i],"Partial auto correlation
function",".png"), width=1200,res =72 )
    pacf(arma)
    dev.off()
    #.....................................................
    # Generate & display network:
    imatVG_ARMA=visi(arma)
    imatHVG_ARMA=visiH(arma)
    netVG_ARMA=graph_from_adjacency_matrix(imatVG_ARMA)
```

```
    netHVG_ARMA=graph_from_adjacency_matrix(imatHVG_ARMA)

    png(file=paste(name_ARMA[i],"VG",".png"),width = 1200,height =
1200,res=72)
    qgraph(imatVG_ARMA, layout=Q$layout,color=color_VG_ARMA[i],
edge.color="grey70", vsize=2, esize=1, labels=FALSE,
           border.color=c(rep("white",size)))
    title(paste("Visibility graph:", name_ARMA[i]),line = 3)
    dev.off()

    png(file=paste(name_ARMA[i],"HVG",".png"),width = 1200,height =
1200,res=72)
    qgraph(imatHVG_ARMA, layout=Q$layout, color=color_HVG_ARMA[i],
edge.color="grey70", vsize=2, esize=1, labels=FALSE,
           border.color=c(rep("white",size)))
    title(paste("Horizontal visibility graph",name_ARMA[i]),line=3)
    dev.off()

    png(file=paste(name_ARMA[i],"Degree frequency VG",".png"),
height=1200,res=72)
    degree_function(imatVG_ARMA,2)
    dev.off()
    png(file=paste(name_ARMA[i],"Degree frequency HVG",".png"),
height=1200,res=72)
    degree_function(imatHVG_ARMA,2)
    dev.off()
    #.....................................................
    # Calculate network statistics:
    #.....................................................
    # Average geodistic path:
    nstat_ARMA[i,1]=mean_distance(netVG_ARMA, directed = FALSE, unconnected
= FALSE)
    nstat_ARMA[i,2]=mean_distance(netHVG_ARMA, directed = FALSE,
unconnected = FALSE)
    #.....................................................
    # Assortativity:
    nstat_ARMA[i,3]=assortativity.degree(netVG_ARMA, directed=FALSE)
    nstat_ARMA[i,4]=assortativity.degree(netHVG_ARMA, directed=FALSE)
    #.....................................................
    # Global transitivity:
    nstat_ARMA[i,5]=transitivity(netVG_ARMA, type="global", vids = NULL,
                                 weights = NULL, isolates = "zero" )
    nstat_ARMA[i,6]=transitivity(netHVG_ARMA, type="global", vids = NULL,
                                 weights = NULL, isolates = "zero" )
    #.....................................................
    # Mean degree:
    degVG=degree(netVG_ARMA,v=V(netVG_ARMA),mode = "in") # change mode
betweeen: "in","out","all","total"
    nstat_ARMA[i,7]=mean(degVG)

    degHVG=degree(netHVG_ARMA,v=V(netHVG_ARMA),mode = "in") # change mode
betweeen: "in","out","all","total"
    nstat_ARMA[i,8]=mean(degHVG)
    #.....................................................
    # Cumulative degree distribution:
    degDisVG=degree.distribution(netVG_ARMA, cumulative = TRUE, mode="in" )
    degDisHVG=degree.distribution(netHVG_ARMA, cumulative = TRUE, mode="in"
)
    #.....................................................
    # Degreee centrality
```

```
    VG_centDeg=centr_degree(netVG_ARMA, mode = "in", loops = FALSE,
normalized = TRUE)
    nstat_ARMA[i,9]=VG_centDeg[[2]]

    HVG_centDeg=centr_degree(netHVG_ARMA, mode = "in", loops = FALSE,
normalized = TRUE)
    nstat_ARMA[i,10]=HVG_centDeg[[2]]
    #.............................................................
    # Test for Power Law distribution:
    VG_PowLaw=fit_power_law(degDisVG, xmin=NULL, force.continuous = TRUE)
    nstat_ARMA[i,11]=VG_PowLaw[[6]]

    HVG_PowLaw=fit_power_law(degDisHVG, xmin=NULL, force.continuous = TRUE)
    nstat_ARMA[i,12]=HVG_PowLaw[[6]]
    #.............................................................
    # Centrality: degree, closeness, betweenness & hub
    cent_VG=centrality(netVG_ARMA)
    cstatVG_ARMA[,1]=cent_VG[[2]]
    cstatVG_ARMA[,2]=cent_VG[[3]]
    cstatVG_ARMA[,3]=cent_VG[[4]]
    hub_vg=hub.score(netVG_ARMA,scale = TRUE,weights = NULL)
    cstatVG_ARMA[,4]=hub_vg[[1]]
    for (k in 1:4) {
      pl=cbind(cstatVG_ARMA[,k],cstatVG_WN[,k])
      matplot(pl, type = "l", col =c(colrs2[k],"grey60"),
ylab="Value",xlab="Node",
              main=paste("VG: Centrality,", name_cstat[k],"ARMA+WN, p_c
=",coef_p[i]))
    }

    cent_HVG=centrality(netHVG_ARMA)
    cstatHVG_ARMA[,1]=cent_HVG[[2]]
    cstatHVG_ARMA[,2]=cent_HVG[[3]]
    cstatHVG_ARMA[,3]=cent_HVG[[4]]
    hub_hvg=hub.score(netHVG_ARMA,scale = TRUE,weights = NULL)
    cstatHVG_ARMA[,4]=hub_hvg[[1]]
    for (k in 1:4) {
      pl=cbind(cstatHVG_ARMA[,k],cstatHVG_WN[,k])
      matplot(pl, type = "l", col =c(colrs2[4+k],"grey60"),
ylab="Value",xlab="node" ,
              main=paste("HVG: Centrality,", name_cstat[k],"ARMA+WN, c_p
=",coef_p[i]))
    }

    #.............................................................
    # Share of geodesics having a given length
    spathVG=cent_VG[7]
    geodshare_VG=table(spathVG)
    col_bar=colorRampPalette (c(color_VG_ARMA[i],"white"))
    col_barVG=col_bar(length(geodshare_VG))
    png(file=paste(name_ARMA[i],"VG-Geodistics",".png"),width=1200,res=72)
    barplot(geodshare_VG, col = col_barVG, ylab="Number of geodistics",xlab
= "Geodistic distance",
            main=paste("VG: Share of geodistics having a given length,
ARMA(1), c_p=",coef_p[i]))
    dev.off()

    spathHVG=cent_HVG[7]
    geodshare_HVG=table(spathHVG)
    col_bar=colorRampPalette (c(color_HVG_ARMA[i],"white"))
    col_barVG=col_bar(length(geodshare_VG))
```

```
    png(file=paste(name_ARMA[i],"HVG-Geodistics",".png"),width=1200,res=72)
    barplot(geodshare_HVG, col = col_barHVG, ylab="Number of
geodistics",xlab = "Geodistic distance",
            main=paste("HVG: Share of geodistics having a given length,
ARMA(1), c_p=",coef_p[i]))
    dev.off()
  }
  colnames(data_ARMA)=name_ARMA

  write.table(data_ARMA,file="data_ARMA.txt")
  write.table(dstat_ARMA,file="dstat_ARMA.txt")
  write.table(nstat_ARMA, file = "nstat_ARMA.txt", col.names = TRUE)
  write.table(cstatVG_ARMA, file="cstat_ARMA_VG.txt",col.names=TRUE)
  write.table(cstatHVG_ARMA, file="cstat_ARMA_HVG.txt",col.names=TRUE)

}else{
  #-----------------------------------------------------------------
  # White noice - reference
  data_WN=vector()

  dstat_WN=matrix(nrow = sim_num,ncol = length(name_dstat))
  colnames(dstat_WN)=name_dstat

  nstat_WN=matrix(ncol = length(name_nstat),nrow = sim_num)
  colnames(nstat_WN)=name_nstat

  for (i in 1:sim_num) {
    #.....................................................
    # Generate data:
    wn=rnorm(size,mean=0,sd=1)
    wn=as.vector(wn)
    data_WN=cbind(data_WN,wn)

    dstat_WN[i,1]=mean(wn)
    dstat_WN[i,2]=sd(wn)
    dstat_WN[i,3]=max(wn)
    dstat_WN[i,4]=min(wn)
    #.....................................................
    # Generate network:
    imatVG_WN=visi(wn)
    imatHVG_WN=visiH(wn)
    netVG_WN=graph_from_adjacency_matrix(imatVG_WN)
    netHVG_WN=graph_from_adjacency_matrix(imatHVG_WN)
    #.....................................................
    # Calculate network statistics:
    #.....................................................
    # Average geodistic path:
    nstat_WN[i,1]=mean_distance(netVG_WN, directed = FALSE, unconnected =
FALSE)
    nstat_WN[i,2]=mean_distance(netHVG_WN, directed = FALSE, unconnected =
FALSE)
    #.....................................................
    # Assortativity:
    nstat_WN[i,3]=assortativity.degree(netVG_WN, directed=FALSE)
    nstat_WN[i,4]=assortativity.degree(netHVG_WN, directed=FALSE)
    #.....................................................
    # Global transitivity:
    nstat_WN[i,5]=transitivity(netVG_WN, type="global", vids = NULL,
                               weights = NULL, isolates = "zero" )
    nstat_WN[i,6]=transitivity(netHVG_WN, type="global", vids = NULL,
                               weights = NULL, isolates = "zero" )
```

```
    #.....................................................
    # Mean degree:
    degVG=degree(netVG_WN,v=V(netVG_WN),mode = "in")
    nstat_WN[i,7]=mean(degVG)

    degHVG=degree(netHVG_WN,v=V(netHVG_WN),mode = "in")
    nstat_WN[i,8]=mean(degHVG)
    #.....................................................
    # Cumulative degree distribution:
    degDisVG_WN=degree.distribution(netVG_WN, cumulative = TRUE, mode="in"
)
    degDisHVG_WN=degree.distribution(netHVG_WN, cumulative = TRUE,
mode="in" )
    #.....................................................
    # Degreee centrality
    VG_centDeg=centr_degree(netVG_WN, mode = "in", loops = FALSE,
normalized = TRUE)
    nstat_WN[i,9]=VG_centDeg[[2]]

    HVG_centDeg=centr_degree(netHVG_WN, mode = "in", loops = FALSE,
normalized = TRUE)
    nstat_WN[i,10]=HVG_centDeg[[2]]
    #.....................................................
    # Test for Power Law distribution:
    VG_PowLaw=fit_power_law(degDisVG_WN, xmin=NULL, force.continuous =
TRUE)
    nstat_WN[i,11]=VG_PowLaw[[6]]

    HVG_PowLaw=fit_power_law(degDisHVG_WN, xmin=NULL, force.continuous =
TRUE)
    nstat_WN[i,12]=HVG_PowLaw[[6]]
  }

  write.table(data_WN, file="SIM data_WN.txt", col.names=TRUE)
  write.table(dstat_WN,file = "SIM dstat_WN.txt", col.names = TRUE)
  write.table(nstat_WN, file="SIM nstat_WN.txt",col.names = TRUE)
  #-------------------------------------------------------------------
  # MA
  data_MA=vector()

  dstat_MA=matrix(nrow = sim_num,ncol = length(name_dstat))
  colnames(dstat_MA)=name_dstat

  nstat_MA=matrix(ncol = length(name_nstat),nrow = sim_num)
  colnames(nstat_MA)=name_nstat

  for (i in 1:sim_num) {
    #.....................................................
    # Generate data:
    ma=arima.sim(n=size, model=list(ma=coef_q))
    ma=as.vector(ma)
    data_MA=cbind(data_MA,ma)

    dstat_MA[i,1]=mean(ma)
    dstat_MA[i,2]=sd(ma)
    dstat_MA[i,3]=max(ma)
    dstat_MA[i,4]=min(ma)
    #.....................................................
    # Generate network:
    imatVG_MA=visi(ma)
    imatHVG_MA=visiH(ma)
```

```
    netVG_MA=graph_from_adjacency_matrix(imatVG_MA)
    netHVG_MA=graph_from_adjacency_matrix(imatHVG_MA)
    #.....................................................
    # Calculate network statistics:
    #.....................................................
    # Average geodistic path:
    nstat_MA[i,1]=mean_distance(netVG_MA, directed = FALSE, unconnected =
FALSE)
    nstat_MA[i,2]=mean_distance(netHVG_MA, directed = FALSE, unconnected =
FALSE)
    #.....................................................
    # Assortativity:
    nstat_MA[i,3]=assortativity.degree(netVG_MA, directed=FALSE)
    nstat_MA[i,4]=assortativity.degree(netHVG_MA, directed=FALSE)
    #.....................................................
    # Global transitivity:
    nstat_MA[i,5]=transitivity(netVG_MA, type="global", vids = NULL,
                               weights = NULL, isolates = "zero" )
    nstat_MA[i,6]=transitivity(netHVG_MA, type="global", vids = NULL,
                               weights = NULL, isolates = "zero" )
    #.....................................................
    # Mean degree:
    degVG=degree(netVG_MA,v=V(netVG_MA),mode = "in")
    nstat_MA[i,7]=mean(degVG)

    degHVG=degree(netHVG_MA,v=V(netHVG_MA),mode = "in")
    nstat_MA[i,8]=mean(degHVG)
    #.....................................................
    # Cumulative degree distribution:
    degDisVG_MA=degree.distribution(netVG_MA, cumulative = TRUE, mode="in"
)
    degDisHVG_MA=degree.distribution(netHVG_MA, cumulative = TRUE,
mode="in" )
    #.....................................................
    # Degreee centrality
    VG_centDeg=centr_degree(netVG_MA, mode = "in", loops = FALSE,
normalized = TRUE)
    nstat_MA[i,9]=VG_centDeg[[2]]

    HVG_centDeg=centr_degree(netHVG_MA, mode = "in", loops = FALSE,
normalized = TRUE)
    nstat_MA[i,10]=HVG_centDeg[[2]]
    #.....................................................
    # Test for Power Law distribution:
    VG_PowLaw=fit_power_law(degDisVG_MA, xmin=NULL, force.continuous =
TRUE)
    nstat_MA[i,11]=VG_PowLaw[[6]]

    HVG_PowLaw=fit_power_law(degDisHVG_MA, xmin=NULL, force.continuous =
TRUE)
    nstat_MA[i,12]=HVG_PowLaw[[6]]

  }

  write.table(data_MA, file="SIM data_MA.txt", col.names=TRUE)
  write.table(dstat_MA,file = "SIM dstat_MA.txt", col.names = TRUE)
  write.table(nstat_MA, file="SIM nstat_MA.txt",col.names = TRUE)
  #-------------------------------------------------------------------
  # ARs with different coefficient:

  nstat_AR=matrix(ncol=length(name_nstat),nrow=sim_num)
```

```
  colnames(nstat_AR)=name_nstat
  all_data_AR=vector("list",length(coef_p))
  all_dstat_AR=vector("list", length(coef_p))
  all_nstat_AR=vector("list",length(coef_p))

  for (j in 1:length(coef_p)) {
    data_AR=vector()

    dstat_AR=matrix(nrow = sim_num, ncol = length(name_dstat))
    colnames(dstat_AR)=name_dstat

    for (i in 1:sim_num) {
      #.....................................................
      # Generate & display data:
      ar=arima.sim(n=size, model=list(ar=coef_p[j]))
      ar=as.vector(ar)
      data_AR=cbind(data_AR,ar)

      dstat_AR[i,1]=mean(ar)
      dstat_AR[i,2]=sd(ar)
      dstat_AR[i,3]=max(ar)
      dstat_AR[i,4]=min(ar)

      #.....................................................
      # Generate network:
      imatVG_AR=visi(ar)
      imatHVG_AR=visiH(ar)
      netVG_AR=graph_from_adjacency_matrix(imatVG_AR)
      netHVG_AR=graph_from_adjacency_matrix(imatHVG_AR)
      #.....................................................
      # Calculate network statistics:
      #.....................................................
      # Average geodistic path:
      nstat_AR[i,1]=mean_distance(netVG_AR, directed = FALSE, unconnected =
FALSE)
      nstat_AR[i,2]=mean_distance(netHVG_AR, directed = FALSE, unconnected
= FALSE)
      #.....................................................
      # Assortativity:
      nstat_AR[i,3]=assortativity.degree(netVG_AR, directed=FALSE)
      nstat_AR[i,4]=assortativity.degree(netHVG_AR, directed=FALSE)
      #.....................................................
      # Global transitivity:
      nstat_AR[i,5]=transitivity(netVG_AR, type="global", vids = NULL,
                                 weights = NULL, isolates = "zero" )
      nstat_AR[i,6]=transitivity(netHVG_AR, type="global", vids = NULL,
                                 weights = NULL, isolates = "zero" )
      #.....................................................
      # Mean degree:
      degVG=degree(netVG_AR,v=V(netVG_AR),mode = "in")
      nstat_AR[i,7]=mean(degVG)

      degHVG=degree(netHVG_AR,v=V(netHVG_AR),mode = "in")
      nstat_AR[i,8]=mean(degHVG)
      #.....................................................
      # Cumulative degree distribution:
      degDisVG=degree.distribution(netVG_AR, cumulative = TRUE, mode="in" )
      #degDisVG_AR[[i]]=degDisVG

      degDisHVG=degree.distribution(netHVG_AR, cumulative = TRUE, mode="in"
)
```

```
      #degDisHVG_AR[[i]]=degDisHVG
      #.....................................................
      # Degreee centrality
      VG_centDeg=centr_degree(netVG_AR, mode = "in", loops = FALSE,
normalized = TRUE)
      nstat_AR[i,9]=VG_centDeg[[2]]

      HVG_centDeg=centr_degree(netHVG_AR, mode = "in", loops = FALSE,
normalized = TRUE)
      nstat_AR[i,10]=HVG_centDeg[[2]]
      #.....................................................
      # Test for Power Law distribution:
      VG_PowLaw=fit_power_law(degDisVG, xmin=NULL, force.continuous = TRUE)
      nstat_AR[i,11]=VG_PowLaw[[6]]

      HVG_PowLaw=fit_power_law(degDisHVG, xmin=NULL, force.continuous =
TRUE)
      nstat_AR[i,12]=HVG_PowLaw[[6]]
    }
    write.table(data_AR, file=paste("SIM data",name_AR[j] , ".txt"),
col.names=TRUE)
    write.table(dstat_AR,file = paste("SIM dstat",name_AR[j] , ".txt"),
col.names = TRUE)
    write.table(nstat_AR, file=paste("SIM nstat",name_AR[j] ,
".txt"),col.names = TRUE)

    all_data_AR[[j]]=data_AR
    all_dstat_AR[[j]]=dstat_AR
    all_nstat_AR[[j]]=nstat_AR
  }
  #-------------------------------------------------------------------
  # ARMAs with different coefficients:

  nstat_ARMA=matrix(ncol=length(name_nstat),nrow=sim_num)
  colnames(nstat_ARMA)=name_nstat
  all_data_ARMA=vector("list",length(coef_p))
  all_dstat_ARMA=vector("list", length(coef_p))
  all_nstat_ARMA=vector("list",length(coef_p))

  for (j in 1:length(coef_p)) {
    data_ARMA=vector()

    dstat_ARMA=matrix(nrow = sim_num, ncol = length(name_dstat))
    colnames(dstat_ARMA)=name_dstat

    for (i in 1:sim_num) {

      #.....................................................
      # Generate & display data:
      arma=arima.sim(n=size, model=list(ar=coef_p[j],ma=coef_q))
      arma=as.vector(arma)
      data_ARMA=cbind(data_ARMA,arma)

      dstat_ARMA[i,1]=mean(arma)
      dstat_ARMA[i,2]=sd(arma)
      dstat_ARMA[i,3]=max(arma)
      dstat_ARMA[i,4]=min(arma)

      #.....................................................
      # Generate network:
      imatVG_ARMA=visi(arma)
```

```
      imatHVG_ARMA=visiH(arma)
      netVG_ARMA=graph_from_adjacency_matrix(imatVG_ARMA)
      netHVG_ARMA=graph_from_adjacency_matrix(imatHVG_ARMA)
      #.....................................................
      # Calculate network statistics:
      #.....................................................
      # Average geodistic path:
      nstat_ARMA[i,1]=mean_distance(netVG_ARMA, directed = FALSE,
unconnected = FALSE)
      nstat_ARMA[i,2]=mean_distance(netHVG_ARMA, directed = FALSE,
unconnected = FALSE)
      #.....................................................
      # Assortativity:
      nstat_ARMA[i,3]=assortativity.degree(netVG_ARMA, directed=FALSE)
      nstat_ARMA[i,4]=assortativity.degree(netHVG_ARMA, directed=FALSE)
      #.....................................................
      # Global transitivity:
      nstat_ARMA[i,5]=transitivity(netVG_ARMA, type="global", vids = NULL,
                                   weights = NULL, isolates = "zero" )
      nstat_ARMA[i,6]=transitivity(netHVG_ARMA, type="global", vids = NULL,
                                   weights = NULL, isolates = "zero" )
      #.....................................................
      # Mean degree:
      degVG=degree(netVG_ARMA,v=V(netVG_ARMA),mode = "in")
      nstat_ARMA[i,7]=mean(degVG)

      degHVG=degree(netHVG_ARMA,v=V(netHVG_ARMA),mode = "in")
      nstat_ARMA[i,8]=mean(degHVG)
      #.....................................................
      # Cumulative degree distribution:
      degDisVG=degree.distribution(netVG_ARMA, cumulative = TRUE, mode="in"
)
      #degDisVG_ARMA[[i]]=degDisVG

      degDisHVG=degree.distribution(netHVG_ARMA, cumulative = TRUE,
mode="in" )
      #degDisHVG_ARMA[[i]]=degDisHVG
      #.....................................................
      # Degreee centrality
      VG_centDeg=centr_degree(netVG_ARMA, mode = "in", loops = FALSE,
normalized = TRUE)
      nstat_ARMA[i,9]=VG_centDeg[[2]]

      HVG_centDeg=centr_degree(netHVG_ARMA, mode = "in", loops = FALSE,
normalized = TRUE)
      nstat_ARMA[i,10]=HVG_centDeg[[2]]
      #.....................................................
      # Test for Power Law distribution:
      VG_PowLaw=fit_power_law(degDisVG, xmin=NULL, force.continuous = TRUE)
      nstat_ARMA[i,11]=VG_PowLaw[[6]]

      HVG_PowLaw=fit_power_law(degDisHVG, xmin=NULL, force.continuous =
TRUE)
      nstat_ARMA[i,12]=HVG_PowLaw[[6]]
    }
    write.table(data_ARMA, file=paste("SIM data",name_ARMA[j] , ".txt"),
col.names=TRUE)
    write.table(dstat_ARMA,file = paste("SIM dstat",name_ARMA[j] , ".txt"),
col.names = TRUE)
    write.table(nstat_ARMA, file=paste("SIM nstat",name_ARMA[j] ,
".txt"),col.names = TRUE)
```

```
    all_data_ARMA[[j]]=data_ARMA
    all_dstat_ARMA[[j]]=dstat_ARMA
    all_nstat_ARMA[[j]]=nstat_ARMA
  }

  #-----------------------------------------------------------------
  # Compare results form AR with White noice:

  name_quant=c("0%","25%","50%","75%","100%")
  name_display_AR=c("WN",coef_p)

  quant_AR=matrix(ncol=length(name_quant),nrow=length(name_display_AR))
  colnames(quant_AR)=name_quant
  rownames(quant_AR)=name_display_AR

  colBXP_VG=c("grey50", color_VG_AR)
  colBXP_HVG=c("grey50", color_HVG_AR)

  for (j in 1:length(name_nstat)) {
    box_AR=nstat_WN[,j]
    for (i in 1:length(coef_p)) {
      input=all_nstat_AR[[i]]
      box_AR=cbind(box_AR,input[,j])
    }
    for (k in 1:ncol(box_AR)) {
      qu=quantile(box_AR[,k])
      quant_AR[k,]=qu
    }
    colnames(box_AR)= name_display_AR
    png(file=paste("SIM boxplot AR VG",j,".png"),height =1200,res=72)
    boxplot(box_AR, main=paste("AR -",name_nstat[j]),col=colBXP_VG,
border="grey30")
    abline(v=1.5, col="grey70")
    dev.off()

    colnames(box_AR)= name_display_AR
    png(file=paste("SIM boxplot AR HVG",j,".png"),height =1200,res=72)
    boxplot(box_AR, main=paste("AR -",name_nstat[j]),col=colBXP_HVG,
border="grey30")
    abline(v=1.5, col="grey70")
    dev.off()

    colnames(box_AR)=paste(name_display_AR,name_nstat[j])
    write.table(box_AR,file=paste("Boxplot data AR",j,".txt"))
    write.table(quant_AR,file=paste("Quantiles_AR",j,".txt"))

  }

  #-----------------------------------------------------------------
  # Compare results form ARMA with White noice and MA

  name_display_ARMA=c("WN","MA",coef_p)

  quant_ARMA=matrix(ncol=length(name_quant),nrow=length(name_display_ARMA))
  colnames(quant_ARMA)=name_quant
  rownames(quant_ARMA)=name_display_ARMA

  colBXP_VG=c("grey50","grey80", color_VG_ARMA)
  colBXP_HVG=c("grey50","grey80", color_HVG_ARMA)
```

```
  for (j in 1:length(name_nstat)) {
    boxARMA=nstat_WN[,j]
    boxARMA=cbind(boxARMA,nstat_MA[,j])
    for (i in 1:length(coef_p)) {
      input=all_nstat_ARMA[[i]]
      boxARMA=cbind(boxARMA,input[,j])
    }
    for (k in 1:ncol(boxARMA)) {
      qu=quantile(boxARMA[,k])
      quant_ARMA[k,]=qu
    }
    colnames(boxARMA)= name_display_ARMA
    png(file=paste("SIM boxplot ARMA VG",j,".png"),height =1200,res=72)
    boxplot(boxARMA, main=paste("ARMA -
",name_nstat[j]),col=colBXP_VG,border="grey20")
    abline(v=1.5, col="grey70")
    abline(v=2.5,col="grey80")
    dev.off()

    png(file=paste("SIM boxplot ARMA HVG",j,".png"),height =1200,res=72)
    boxplot(boxARMA, main=paste("ARMA -
",name_nstat[j]),col=colBXP_HVG,border="grey20")
    abline(v=1.5, col="grey70")
    abline(v=2.5,col="grey80")
    dev.off()

    colnames(boxARMA)= paste(name_display_ARMA,name_nstat[j])
    write.table(boxARMA,file=paste("Boxplot data ARMA",j,".txt"))
    write.table(quant_ARMA,file=paste("Quantiles_ARMA",j,".txt"))
  }

}


save.image(file="ARMA.RData")
```

I: Reflection notes

Farnoosh Farhangian

The main focus of this thesis is about use of the complex network theory to analysis different stochastic processes. It is about using a better and more consistent tool for analyzing time series, even if they are of the problematic natures. A good example of series of the problematic nature is non stationary time series. These series are challenging to deal with if using normal analysis tools, and the results provided from them are not as reliable as the results from stationary series. While using network theory could provide us with reliable results for any series no matter of its stationarity status. The network theory is about converting timeseries into network. A network, which is also called as a graph is constructed from some nodes that are the realizations in the converted time series and edges which are the connected lines between nodes. The construction of the networks is determined by different algorithms, and we decided to focus on two mainly used algorithms, the visibility and horizontal visibility algorithms. These algorithms are results in to different graphs, visibility and horizontal visibility graphs, which a horizontal visibility graph is considered as a subgraph of the visibility graph. The visibility criterion supposes that two realizations are visible to each other if their values are higher than the value of the intermediate realizations, but the horizontal visibility algorithm is more restrictive and states that two realizations are considered to be connected, if they are horizontally visible to each other without being intersected by any intermediate realizations.

We wanted to study the behavior of different stochastic process under some specific circumstances. For this purpose, we used hundreds of simulations of stochastic processes as Gaussian white noise, autoregressive of order one, moving average of order one and a combination of two latest ones, an $ARMA(1,1)$. We tried to find a pattern that could help us to identify different processes only by studying their networks' properties. We decided to focus on some of the global properties of a network as its' mean degree, degree centrality, average shortest path length, the correlation coefficient between degrees of the nodes, assortativity and the probability of having triples, transitivity. First, we studied a Gaussian white noise process and its networks' statistics behaviors as the length of the sample size changes. The results provided us with enough information about most of the upper and lower boundaries of the statistics for each and every sample size. We were able to fit equations to those boundaries and propose a test for identifying a white noise process. Then we used white noise as the benchmark process and reviewed the network statistics of the other three processes as their parameters vary. We discovered similar results for both $AR(1), MA(1)$ and $ARMA(1,1)$, thus we were not able to distinguish those series based on the behavior of their networks' statistics. But, the results from two of statistics, transitivity and mean degree were interesting. We could actually identify the autocorrelation coefficients of those series by only knowing the length of their series and some of their networks properties as transitivity and mean degree.

This thesis was both challenging and interesting to me. It was challenging since it was new and I did not know anything about it, and it was interesting since learning something new is always interesting and enjoyable. It was a good opportunity to use the previous knowledges both achieved in the last five years at University of Adger and also the prior to that. we need to thank

our supervisor for believing us and providing us with an interesting topic and also helping us in the whole way of creating this thesis.

The network theory has been used to study different concepts in the world-wide. There are several areas which this theory is applied to which could be related to internationalization. One of these areas is finance. It could be used to analysis different financing time series, and as Yang et al. (2015) suggested, the visibility graph could be used in investigating the relationship among parties in financing. As mentioned previously in both the thesis and in this letter, nodes in a network are defining the realizations in the time series and most central nodes are reflecting most important realizations. In a financial timeseries, the most central nodes will be reflecting the most important historical events. This was implied by Zhuang et al. (2014), as well as that the market integration could be measured by use of the visibility algorithm. As we were further discussed in the chapter of "Evaluations and applications of time series-based network", the network theory has been used to identify the dependency between coal price index and coal mining accident. On the other hand, our proposed test could be used to identify a white noise process. Since an efficient market will have the same nature as a white noise process, this test could be used to identify the efficient markets. Of course, that this test needs further investigations and must be tested on real life data before taking any conclusions.

The graph theory is a new approach in analyzing time series even if they are of problematic nature and have some non-linear properties. This theory and the methods behind it could be considered as an innovation in the field of time series analysis. even if it needs more examine to find out its disadvantages as well as it advantages, still it will be innovative to apply it to understand different time series.

It is not easy to relate this analysis tool to responsibility, since it is only a tool as like as the other tools available for analyzing time series. One could think about moral use of this tool and try to define what is meant by using an analyzing tool ethically. Maybe making advantages of the provided results and also timeseries properties captured by this analysis tool, is an irresponsible behavior. Something that could be related to any analysis tools. But the graph theory is an untested theory and needs further examining and it could not be used without considering its limitations and problems. So, it would be irresponsible to introduce it as a great analyzing tool, though been aware that it needs more investigation before taking any conclusions from.

Gry Nerjordet

Our thesis is an exploratory exercise where we record the behavior time-series has as networks. The idea of analyzing time-series with the mature graph theory has become popular in the last decade since it doesn't depend on a series stationarity. Stationarity is one of the main assumptions in time-series analysis, call for a constant mean and variance, properties which seldom is observed in real life data.

We choose to focus on the stochastic processes white noise, autoregressive, moving average and autoregressive moving average, the latter three of order one. For our examination purposes where all four processes artificially generated in the free software R, no real-life time-series was used. The study was performed by running multiple repetitions of the same series with the intent of determine typical behaviors. We also created a reference set consisting of typical realizations from the different processes. These where constructed such that each series had the same error term and network layout, thus the only difference where the data generating processes and their parameters. The reference set was used to enhance our understanding of the pattern revealed in the multiple repetition simulations.

There are multiple methods which can transform a time-series into a network. We chose to use the visibility algorithm and the horizontal visibility algorithm. The conversion involves of the use of a visibility criterion which determine the connection, based on the time-series' different measurements. The horizontal visibility graph is a subgraph of the visibility graph with a stricter visibility criterion. Since the graphs are linked directly to the time domain which makes it possible to identify which time-series properties the different network statistics relate to. These algorithms also result in fast conversion which was a factor when dealing with larger sizes. Since graph theory offer a vast number of measurements, we chose to narrow this down to the global measurement which not was very social network oriented. The chosen statistics were mean degree, normalized degree centrality, the properties of degree distribution, transitivity, assortativity and averages shortest paths length. All statistics where calculated for both visibility graph and its subgraph.

Our study resulted in two findings. Firstly, we discovered that the relationship between white noise and the time-series length, and where able to fit the results to functions. As a result, we purposed a test which could identify white noise processes without the concern of stationarity. Secondly, we discovered that some of the network statistics, mean degree and transitivity had unique results for each of the correlation coefficient in $AR(1), MA(1)$ and $ARMA(1,1)$. Unfortunately, these where not distinguishable from each other, and we could not use our result to generate a test which identified the mentioned processes. Instead we developed a tool which could, without regarding stationarity, identify the processes correlation coefficient, but not the process it selves.

None of the achievements mentioned above would be possible without the knowledge accumulated during my studies at the university of Agder. The foundation built by topics as Quantitative Financial Economics, Econometrics for Finance, Research Methods in Business and Advanced Econometrics for Finance. The practical introduction to R in Computational Finance and Portfolio Management. The presence of engaging lecturers and excellent supervisor. All three

have been crucial in the creation of this thesis. The ability to learn and to learn fast has also been important when working with this thesis as the world of graph theory was a completely new topic.

As our thesis was an exploratory one, we have discovered tests and tools, the results are international by nature. There is no need to adapt this to accommodate other languages and cultures. The development of an alternative method to analyze time-series is by an international effort where multiple scientific communities contributes. Such a method may result in enterprises' enhanced understanding of global market and thus result in their increased involvement, but this is just speculations. The analysis of time-series through networks is as a science in its infancy and must have years to grow and mature before it enters the enterprises board meetings as a convincing decision-making tool.

We regard our thesis as a building block in the emerging development of an alternative to time-series analysis and are therefore at the forefront of a complexly new way processing this kind of data. The properties we have discovered have not been documented before and can now be used into the development of this new and emerging field of science. Much of the development in improving on time-series analyses is done behind closed doors and will never be made available to the public. The reason being that the ability to correctly identify the underlying process of time-series is a significant competitive advantage. The decision to deny the public contribution to an emerging science is a two-edged sword. On one hand the enterprise may have a short-lived competitive advantage which will surely evaporate when the competitors or the global scientific communities catches up. On the other side by sharing the development it may propel the method forward and making it more reliable, but its usage will immediately be known and used by competitors. The choice between the enterprise profits and to contribute in the development of a new analysis is not an obvious one and will never have an exact answer. We choose to share our findings and invite others to test, use and enhance our results. Another aspect of responsibility is on how an emerging alternative to time-series is used. Whit an MBA I am trusted to perform analysis, interpret and present the results to colleagues which don't have the same expertise as myself. The use of a new and untested method, like time-series analysis through graph theory, and not informing about the contradictions and incompleteness currently residing in this analysis will be reckless.