Master's Thesis

Exponential Random Graph Models for Signed Networks: Implementation and Application

Department of Statistics
Ludwig-Maximilians-Universität München

Marc Schalberger

Munich, May 28th, 2023

Submitted in partial fulfillment of the requirements for the degree of M. Sc.
Supervised by Dr. Cornelius Fritz

Abstract

Signed networks occur when the edges between actors are either positive or negative. Such networks are frequently found in social studies or political studies, where the actors, e.g. persons or countries, can be friends/allies or enemies. Although this area has been widely researched, especially in the context of the structural balance theory, the implementation of a model that allows inference for this type of network is much more limited. The aim of this thesis is to provide a software extension to the widely used Exponential Random Graph Model (ERGM) that is able to handle a signed network by building on the existing `statnet` packages. In addition, this model should accommodate static cross-sectional networks as well as dynamic networks. Both of these types of networks will be illustrated by means of an empirical application example.

Contents

1 Introduction

Network analysis is a set of methods used to study how entities, such as individuals, groups, or organizations, interact or relate to one another. This involves representing the entities within a system as nodes and the relationships between them as edges. These edges can take on different forms depending on the type of network and the nature of the relationships being studied. These approaches are commonly used in fields such as political and social science, where actors often have mutual interests and dependencies. However, traditional methods often assume independence among actors, which can lead to faulty inference. Network analysis allows for the simultaneous analysis of actors and their relationships, providing a more accurate and complete understanding of complex systems.

This approach has been applied to a wide range of topics, including sexual relations (Morris and Kretzschmar, 1997), friendship relations (Newcomb, 1961), and international relations (Ward and Hoff, 2007).

The most popular and widespread methods in the field of network analysis is the Exponential Random Graph Model (ERGM), which is also referred to as the "p*" class of models in the psychology and sociology literature. (Wasserman and Pattison, 1996)

The basic version of this model is only capable of accommodating binary edges, where a relationship between two actors is either present or absent. However, a lot of extensions have been built on the basis of the ERGM to handle different type of networks, such as valued networks (Robins et al., 1999) or, in this case, signed networks.

Signed Networks are those that differentiate between positive and negative ties (or non-existent). For example, in a social network, a positive edge might represent a friendship between two individuals, while a negative edge might represent a conflict or animosity. This type of network has been of interest to political scientists since the early 1960s (Harary, 1961). Of particular interest is the formation of triads and the associated structural balance theory. Heider (1946) divided triads into balanced, which have later been extended to "strong" and "weak" structural balance (Cartwright and Harary, 1956, Davis, 1967), and unbalanced triads, where balanced triads are considered to be more stable and will therefore last longer than unbalanced triads. Structural balanced theory is discussed in greater detail in Section 3.

Network data with signed relations comes in many forms, from friendships and bullying between children (Huitsing et al., 2012) to alliances and conflicts between criminal groups (Nakamura et al., 2020).

This thesis is structured as follows: In the next Section the statistical theory of Exponential Random Graph Models will be briefly reviewed. In Section 3, the ERGM will

be extended to signed networks. Next, the aforementioned ERGM for signed networks will be implemented in R, and then, in Section 5, a data set is introduced to which the model is applied. Finally, this thesis will be concluded with a discussion about possible extensions and problems.

2 Exponential Random Graph Model (ERGM)

2.1 Motivation

The Exponential Random Graph Model (ERGM) is a statistical model used to analyze network patterns and characteristics. It is used to understand fundamental principles that drive the formation and evolution of networks. ERGMs are used in a wide range of fields, including social science and political science.

The central tenet of ERGMs is the presumption that the probability of a particular network configuration can be expressed as an exponential function of a set of network statistics, selected to capture the essential characteristics of the network. These statistics can be either exogenous or endogenous. Exogenous are those occurring outside the network, such as age, gender, wage of a person in social studies or GDP, and population size of a country in political studies. Endogenous effects occur inside the network and are structural features of the network. For instance, the number of edges, triangles, and degree distribution of the network's nodes are common endogenous network statistics used in ERGMs. The ability to include exogenous, as well as endogenous effects, is one of the main advantages of the ERGM.

The ERGM can be used to test hypotheses about the variables that affect the formation and evolution of the network. For instance, it could be used to investigate whether a network's presence of a specific kind of node is related to a higher probability of connecting to other nodes. In addition to testing hypotheses, ERGMs can be used to predict a network's future behaviour by simulating the network's evolution over time.

Overall, the ERGM is a widely used model in network analysis, as it remains a very powerful yet relatively intuitive model to interpret, capable of modelling both endogenous and exogenous effects simultaneously. There is a large amount of literature on the ERGM, and many of the current extensions build on its fundamental concept. As a result, the ERGM is a great place for researchers to start when looking into network dependencies, particularly if endogenous effects need to be tested.

2.2 Model Formulation

For binary networks Frank and Strauss (1986) introduced the ERGM. As the name suggests, the connections between the nodes in a binary network are binary, with each edge being either "on" or "off". The presence of an edge between two nodes indicates that there is some kind of relationship between the two nodes. For example, in a social network of friendship connections, an edge between two nodes might represent that the two individuals are friends.

In the following, lowercase letters will denote the realised value of a random variable, and capitalised letters indicate that they are stochastic random variables. The observed network $y \in \mathcal{Y}$ consists of n nodes and m edges or dyads $\{y_{ij} : i = 1, \ldots, n; j = 1, \ldots, n\}$ where $y_{ij} = 1$ translates to an edge existing between actors i and j and $y_{ij} = 0$ indicates that there is no edge between the two.

Wasserman and Pattison (1996) formulated a probability distribution of observing network y over all possible networks that could have been observed is given by:

$$\begin{aligned} \mathbb{P}(Y = y) &= \frac{\exp\{\theta' s(y)\}}{\kappa(\theta)} \forall y \in \mathcal{Y} \\ \kappa(\theta) &:= \sum_{\tilde{y} \in \mathcal{Y}} \exp\{\theta' s(\tilde{y})\} \end{aligned} \tag{1}$$

Thus, the probability of the network y is modelled as a log-linear function of endogenous network statistics and exogenous covariates, which are captured in the vector $s(y)$, weighted by the coefficients θ. Note that the normalisation constant $\kappa(\theta)$, which sums over all possible network configurations, is intractable for almost all networks except small and simple ones. Therefore, statistical inference generally relies on an approximation of the likelihood function.

The ERGM requires two main assumptions. Firstly, for two models with the same network statistics $s(y)$ the probability of observing one of the two will also be the same. This means that all relevant variables and effects have been included in the model, and there are no omitted variables that could affect the likelihood of observing the network. Secondly, the observed network exhibits the average network statistics over the networks that could have been observed. This means that the observed network is representative of the underlying distribution of networks and that it is not an outlier or an unusually rare configuration. (Cranmer et al., 2020)

Coefficients θ of a fitted ERGM can be interpreted locally (on edge-level), analogously to a logistic regression as the effect on the odds or log-odds of a tie.

$$\begin{aligned} \text{odds}\left(Y_{ij} = 1 \mid y_{ij}^c\right) &= \frac{\exp\{\theta' s(y^+)\} / \kappa(\theta)}{\exp\{\theta' s(y^0)\} / \kappa(\theta)} = \exp\left\{\theta' \delta^{(ij)}(y)\right\} \\ \delta^{(ij)} &:= s\left(y^+\right) - s\left(y^0\right) \end{aligned} \tag{2}$$

$$\begin{aligned} \text{logit}\left(Y_{ij} = 1 \mid y_{ij}^c\right) &= \log\left(\frac{\exp\{\theta\prime s(y^+)\} / \kappa(\theta)}{\exp\{\theta\prime s(y^0)\} / \kappa(\theta)}\right) = \theta' \delta^{(ij)}(y) \\ \delta^{(ij)} &:= s\left(y^+\right) - s\left(y\right) \end{aligned} \tag{3}$$

In order to interpret the odds of actor i and j sharing a tie ($Y_{ij} = 1$) given the rest of the network (y^c), the probability of the network when there exists a tie between

i and j (y^+) is divided by the probability of the network when there does not exist a tie between the two actors (y^0), given that all else is equal. The formula can be shortened by introducing a new parameter $\delta^{(ij)}$, which is the change in network statistics when y_{ij} changes from 0 to 1 while the rest of the network remains the same (y^c). The specific interpretation of individual estimates is discussed in more detail in Section 5 on the basis of the application example.

2.3 Dynamic Networks

The basic ERGM is used to analyse a static network, which is usually a snapshot of an underlying process that may be in a state of change. However, when used on data that is collected over a period of time, this model has a tendency to hide any variations or dependencies in the parameters being examined that might occur over time.

Because of this limitation of the ERGM, it is crucial to use a Temporal Exponential Random Graph Model or TERGM (Hanneke et al., 2010) to effectively uncover the parameters of networks that change over time.

In the TERGM, time-dependence is incorporated by utilising time-specific statistics. For a network at a specific time point t, represented as $y^{(t)}$, it is assumed to be independent of the network at all previous time points ($y^{(1)}$, $y^{(2)}$, ..., $y^{(t-2)}$), given the network at the immediately preceding time point ($y^{(t-1)}$). This results in the following model for an observed network at a given time t:

$$P\left(Y = y^{(t)} \mid \theta, y^{(t-1)}\right) = \frac{\exp\left\{\theta^\top \mathbf{s}\left(y^{(t)}, y^{(t-1)}\right)\right\}}{\kappa(\theta)}, \quad t = 2, \ldots, T \tag{4}$$

The normalisation constant, $\kappa(\theta)$, remains unchanged and additional statistics based on the previous network have been added to the vector of summary statistics $\mathbf{s}\left(y^{(t)}, y^{(t-1)}\right)$. For the computational implementation, the `tergm` package has been used and will be discussed further in Section 4. Due to the complexity of the model, statistical analysis is usually performed using maximum pseudo-likelihood estimators (MPLE) and a bootstrap-based correction is applied to the confidence intervals to account for the increased complexity. (Desmarais and Cranmer, 2012, 2010)

3 ERGM for Signed Networks

3.1 Model

Consider an undirected signed network with n actors, where each edge is assigned a positive or negative sign. The adjacency matrix for this network is denoted as $y = (y_{ij}) \in \{-1, 0, 1\}^{(n \times n)}$, where $y_{ij} = -1$ indicates a negative edge, $y_{ij} = 1$ indicates a positive edge, and $y_{ij} = 0$ indicates no edge between actor i and j. In the undirected case, $y_{ij} = y_{ji}$, meaning the adjacency matrix must be symmetric. This is not the case for a directed network. Additionally, the network might have self-loops, in which case $y_{ii} \neq 0$. In the following only undirected networks without self-loops will be considered. The set of all possible signed adjacency matrices, $\mathcal{Y}^{\pm}$, denotes the space of all observable signed networks between n actors.

$$\mathbb{P}(Y = y) = \frac{\exp\{\theta' s(y)\}}{\kappa(\theta)} \forall y \in \mathcal{Y}^{\pm} \tag{5}$$

Analogously to the basic ERGM, the probability of a signed network y is modelled using a log-linear function that considers both endogenous network statistics and exogenous covariates. These factors are represented in a vector $s(y)$, and their influence on the probability is determined by the coefficients θ.

Many network statistics are essentially counts of certain features of the network, such as the number of edges or the number of triangles. These statistics can be applied to signed networks as well, but can also be split up to focus on counting only positive or negative edges. As an example, the number of negative ties in a signed network y can be represented by the following equation:

$$s_{edges^-}(y) \coloneqq \sum_{i<j} \mathbb{I}\left(y_{ij} = \text{" } -1\text{"}\right) \tag{6}$$

Where $\mathbb{I}\left(y_{ij} = \text{" } -1\text{"}\right)$ is an indicator function that returns 1 if the tie between i and j is negative and 0 otherwise.

Geometrically weighted statistics, such as the degree counts discussed in Snijders et al. (2006), can also be modified to work with signed network data.

$$s_{gwdegree^+}(y \mid \alpha) \coloneqq \sum_{k=0}^{n-1} e^{-\alpha k} d_k^+(y) \tag{7}$$

Where $d_k^+(y)$ counts the number of nodes with positive degree k, and the parameter α with a value greater than 0 controls the rate at which the weight assigned to each degree decreases as the degree increases. This parameter is referred to as the degree

weighting parameter. When α is set to a large value, nodes with higher degrees will have a greatly reduced impact on the overall statistic, thus combating one of the main issues with ERGMs, degeneracy. (Mukherjee, 2020)

In addition to traditional metrics, statistics related to structural balance can also be analysed. These will be examined further in the next Section.

Analogously to the basic ERGM, the coefficients of a fitted ERGM for signed networks can be interpreted locally (on a dyad-level) as the effect on the odds or log-odds of a tie.

$$\begin{gathered}\operatorname{odds}\left(Y_{ij}=+1 \mid y_{ij}^{c}\right)=\frac{\exp\left\{\theta' h\left(y^{+}\right)\right\}/k}{\exp\left\{\theta' h\left(y^{0}\right)\right\}/k}=\exp\left\{\theta' \delta_{pos}^{(ij)}(y)\right\} \\ \delta_{pos}^{(ij)}:=h\left(y^{+}\right)-h\left(y^{0}\right)\end{gathered} \tag{8}$$

$$\begin{gathered}\operatorname{odds}\left(Y_{ij}=-1 \mid y_{ij}^{c}\right)=\frac{\exp\left\{\theta' h\left(y^{-}\right)\right\}/k}{\exp\left\{\theta' h\left(y^{0}\right)\right\}/k}=\exp\left\{\theta' \delta_{neg}^{(ij)}(y)\right\} \\ \delta_{neg}^{(ij)}:=h\left(y^{-}\right)-h\left(y^{0}\right)\end{gathered} \tag{9}$$

In order to interpret the odds of actor i and j sharing a positive tie ($Y_{ij}=+1$), given the rest of the network (y_{ij}^{c}), the probability of the network when there is a positive tie between i and j (y^{+}) is divided by the probability of the network when there is no tie between i and j (y^{0}). Equivalently, to interpret the odds of actor i and j sharing a negative tie ($Y_{ij}=-1$), the probability of the network when there is a negative tie between i and j (y^{-}) is divided by the probability of the network when there is no tie between i and j (y^{0}), given all else is equal. The formulas can be shortened by introducing a new parameter δ, which is the change in network statistics when Y_{ij} changes form 0 to +1 or 0 to -1 respectively while the rest of the network remains the same.

Hence the relative log-odds of Y_{ij} to be "+1" and "-1" rather than "0" is:

$$\begin{gathered}\operatorname{logit}\left(Y_{ij}=+1 \mid y_{ij}^{c}\right)=\log\left(\frac{\exp\left\{\theta' h\left(y^{+}\right)\right\}/k}{\exp\left\{\theta' h\left(y^{0}\right)\right\}/k}\right)=\theta' \delta_{pos}^{(ij)}(y) \\ \delta_{pos}^{(ij)}=h\left(y^{+}\right)-h\left(y^{0}\right)\end{gathered} \tag{10}$$

$$\begin{gathered}\operatorname{logit}\left(Y_{ij}=-1 \mid y_{ij}^{c}\right)=\log\left(\frac{\exp\left\{\theta' h\left(y^{-}\right)\right\}/k}{\exp\left\{\theta' h\left(y^{0}\right)\right\}/k}\right)=\theta' \delta_{neg}^{(ij)}(y) \\ \delta_{neg}^{(ij)}=h\left(y^{-}\right)-h\left(y^{0}\right)\end{gathered} \tag{11}$$

The function 5 can be adapted to handle dynamic networks in a similar way to how the basic ERGM is extended.

$$\mathbb{P}_{\theta}\left(Y_{t}=y_{t} \mid Y_{t-1}=y_{t-1}\right)=\frac{\exp\left\{\theta^{\top} s\left(y_{t}, y_{t-1}\right)\right\}}{\kappa\left(\theta, y_{t-1}\right)} \forall y_{t} \in \mathcal{Y}^{\pm} \tag{12}$$

3.2 Estimation

In order to estimate the coefficients θ, the likelihood of function 13 is maximised conditional on the initial network y_0:

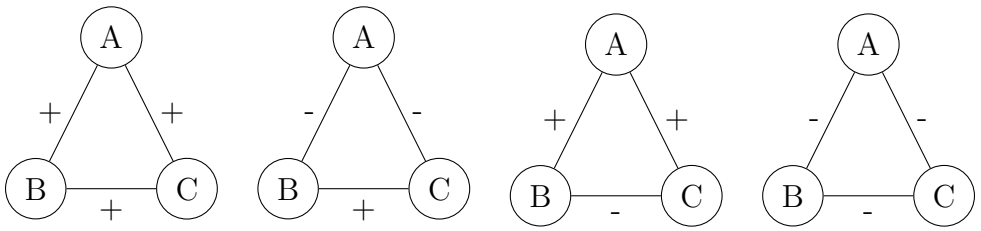
$$\mathcal{L}\left(\theta; y_1, \ldots, y_T\right) = \prod_{t=1}^{T} \frac{\exp\left\{\theta^\top s\left(y_t, y_{t-1}\right)\right\}}{\kappa\left(\theta, y_{t-1}\right)} = \frac{\exp\left\{\theta^\top \left(\sum_{t=1}^{T} s\left(y_t, y_{t-1}\right)\right)\right\}}{\prod_{t=1}^{T} \kappa\left(\theta, y_{t-1}\right)} \tag{13}$$

But as already mentioned, due to the normalisation constant $\kappa(\theta)$, the likelihood is intractable for most models. Therefore the likelihood must be approximated. For this purpose, two methods are most commonly used throughout literature: maximum pseudolikelihood (Strauss and Ikeda, 1990) and Markov Chain Monte Carlo (MCMC) (Geyer and Thompson, 1992).

The MCMC method employs an iterative process and is the default for most software packages. A series of networks sampled from the distribution and parameterised with those parameters that maximised the likelihood using the preceding sample of networks are used to estimate the sum in the denominator of the likelihood function $\kappa(\theta)$. The approximate likelihood function value is optimised iteratively until there is little change. MCMC has several advantages, including the ability to handle a wide range of models and to provide robust estimates, especially for large and sparse networks. However, it can be computationally demanding and may take longer to run, especially for complex models with many parameters. (Cranmer and Desmarais, 2011)

Another method for estimating ERGM coefficients is pseudo-likelihood estimation. This method works by approximating the joint likelihood of the signed outcomes by the product of the conditional probabilities of each signed outcome given the other signed outcomes in the network. This approach allows the use of standard optimisation algorithms to find the parameter values that maximise the pseudo-likelihood function. In other words, the conditional probability of each signed outcome in the network is assessed given the state of the rest of the network, and the product of those probabilities is used to approximate the likelihood (Hunter et al., 2012). Maximising the pseudo-likelihood results in a maximum pseudo-likelihood estimator (MPLE), which, computationally reduces to logistic regression-like approach (Strauss and Ikeda, 1990). The MPLE has some attractive properties, such as being consistent and asymptotically normal under certain conditions. However, it can be sensitive to local optima and may not always provide accurate estimates, especially for large and sparse networks.

Overall, both MCMC sampling and pseudo-likelihood estimation are effective methods for estimating the coefficients of ERGMs. Which method to use depends on the



(a) Balanced (b) Balanced (c) Unbalanced (d) Unbalanced

Figure 1: Balanced and unbalanced triads.

specific characteristics of the network being studied and the goals of the analysis.

3.3 Structural Balance Theory

As mentioned in the introduction structural balance theory is of particular interest when it comes to analysing signed network. Thus, this Section aims to provide a brief overview of the topic.

A key concept in structural balance theory relates to the existence and formation of triads, which is a group of three actors and their relationships with each other. According to Heider (1946) these triads can be classified into balanced and unbalanced triads. Cartwright and Harary (1956) generalised Heider's theory and defined a triad as balanced if all ties are positive ("friend of a friend is a friend") or if only one tie is positive and the other two are negative ("enemy of my enemy is my friend"). The other two configurations are considered unbalanced. These balanced triads are thought to be more stable and therefore occur more often and last longer because the actors have similar goals. Meaning they are either all friends or have a common enemy. The reason unbalanced triads are considered to be unstable is that for the actors it might be more beneficial for one of the edges to change signs. Triads with one negative tie must navigate the conflict between its two "friends" having opposing views. To reach a balanced state, the node should either attempt to convert the negative tie into a positive one or the node will eventually have to take sides and change one of its positive ties to a negative one. Triads where all three actors have negative ties are also unbalanced. In these cases, actors have incentives to change their relationships and form positive connections to gain benefits by working together against a common enemy. An illustration of balanced and unbalanced triads can be found in Figure 1.

4 Implementation

The implementation of the proposed ERGM extension for signed networks is the main focus of this thesis. In this Section, the process of implementing the model in the R programming language will be described in detail, including the necessary data preprocessing, model fitting, and evaluation. The challenges and considerations that arose during the implementation process will also be discussed, as well as the performance and limitations of the resulting model.

The implementation of the proposed model builds on the `statnet` packages, especially the `ergm.multi` package, and utilises a multilayer logic to efficiently and accurately fit the model to signed network data while taking full advantage of the existing `statnet` infrastructure. The data preprocessing step involves ensuring that the input data is in a suitable format for the model, including any necessary formatting or cleaning of the data. The model fitting process involves specifying the desired model and estimating its parameters using maximum likelihood estimation. Finally, the evaluation step involves assessing the fit of the model and examining its output in order to draw meaningful conclusions from the data.

Throughout this Section, detailed examples and code snippets will be provided to illustrate the implementation process and highlight any important considerations or challenges. The performance of the implemented model and its limitations will also be discussed, as well as potential avenues for future work and extension.

The goal of this Section is to provide a transparent and comprehensive description of the implementation of the proposed ERGM extension for signed networks. By providing a thorough and replicable description of the process, others will be able to use and extend this model in their own research.

4.1 Multilayer Networks

Multilayer networks refer to networks in which nodes can be connected not only through traditional edges, but also through additional layers of connections. There are various types of multilayer networks, as described in Kivelä et al. (2014) and Boccaletti et al. (2014). In this thesis, the multiplex networks are of particular interest.

Multiplex networks involve the same set of actors across multiple layers, with each layer representing a different domain of interaction. In this case, between-layer ties connect the same actor across different layers but do not carry any substantive interpretation.

In the past, researchers have examined multilayer network structures using descriptive statistics such as centrality and correlational (Alves et al., 2019, del Río-Chanona

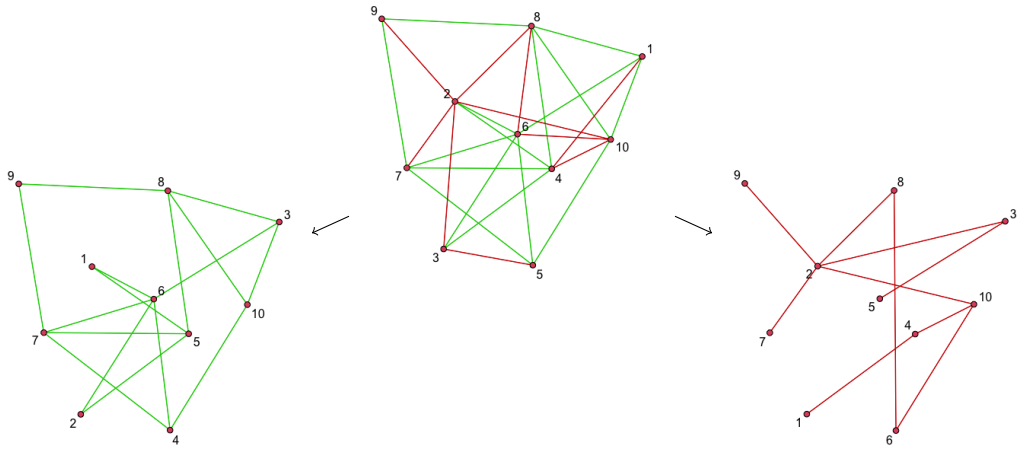


Figure 2: Splitting an undirected signed network into a positive and a negative layer.

et al., 2017, Bonaccorsi et al., 2019). However, there has been a lot of interest in developing statistical approaches for analysing these types of networks, and new methods are being constantly introduced (Kivelä et al., 2014). The multilayer network adaptation of the ERGM has been developed and explained by Chen (2021), and the software adaptation is available in the `ergm.multi` package (Krivitsky et al., 2020).

Signed networks can be effectively represented as multi-layered networks. In this approach, one layer represents positive relationships, while the other layer represents negative relationships. This effectively transforms the signed network into two binary networks, each representing a specific type of relationship. For example, one binary network could represent positive relationships, such as friendships or collaborations, while the other could represent negative relationships, such as conflicts or rivalries.

One necessary constraint, when implementing a signed network as a multilayer network, is that the two layers representing positive and negative edges in a multilayer network cannot have the same edge . This is because an edge cannot be simultaneously positive and negative unless it is a directed edge. In the undirected case it must be ensured that there is no overlap between the edges in the two layers. In other words, an undirected edge cannot connect the same two nodes in both the positive and negative layers.

4.2 Software Implementation in R

The `ergm.multi` package is a part of the `statnet` suite of packages for analyzing and modeling networks in the R programming language. It allows users to fit models to multilayer networks. The `ergm.sign` package is essentially a wrapper package that extends

Formula	Description
`signnet()`	Creates a signed network object.
`summary.static.sign()`	Network attributes of a static signed network.
`summary.dynamic.sign()`	Network attributes of a dynamic signed network.
`plot.static.sign()`	Plot a static signed network.
`plot.dynamic.sign()`	Plot a dynamic signed network.
`sergm()`	Fit an ERGM to a signed network.
`tsergm()`	Fit a TERGM to a signed network.
`count()`	Count network statistics.
`sim_sergm()`	Simulate a signed network based on SERGM.
`sim_tsergm()`	Simulate a signed network based on TSERGM.
`gof_sergm()`	Assess goodness of fit of a SERGM.
`gof_tsergm()`	Assess goodness of fit of a TSERGM.

Table 1: List of Formulas

the functionality of the `statnet` suite to specifically focus on signed networks. It provides a range of functions for calculating various network statistics, generating plots, and conducting inference on signed networks.

Using a wrapper package such as `ergm.sign` has the advantage of benefiting from ongoing improvements and updates to the `statnet` packages, as the wrapper package will incorporate these changes automatically. This can make it easier to keep your analysis up to date and take advantage of new features and improvements as they are developed.

Table 1 provides a summary of functions included in the `ergm.sign` package. Each formula will be discussed in detail in subsequent Sections. In addition, the documentation of the R package can be found in Appendix A.

4.2.1 Create a Signed Network Object

The `ergm.sign` package provides a set of generic functions that can be used to analyse signed network objects using the S3 object-oriented system in R. In order to use these generic functions, an object of class `static.sign` or `dynamic.sign` needs to be created using the function `signnet`. These are special classes defined by the `ergm.sign` package that allow for the representation of signed network objects in R code. Once an object of one of these classes has been created, the generic functions provided by the `ergm.sign` package can be used to perform various operations on the object. The S3 object-oriented system is a way of organising code in R that is based on the use of generic functions and methods. It allows for the definition of classes and methods for those classes, and the use of generic functions to perform operations on objects of those classes in a flexible and extensible way.

	a	b	c
a	0	-1	1
b	0	0	-1
c	0	0	0

Table 2: Adjacency Matrix

From	To	Sign
a	b	-1
a	c	1
b	c	-1

Table 3: Edgelist

The `signnet` function can be used to create a signed network object from either an adjacency matrix or an edgelist. If an adjacency matrix is used as input, it should only contain the values 1, 0, or -1. These values represent the presence or absence of a positive or negative edge between pairs of nodes in the network. If an edgelist is used as input, it should have three columns: "From", "To", and "Sign". The "From" and "To" columns should contain the names or indices of the nodes that the edge connects, and the "Sign" column should contain the value 1 or -1 to indicate whether the edge is positive or negative. Once the input data has been prepared in one of these formats, it can be passed to the `signnet` function along with any other necessary arguments to create a signed network object. In addition to the data, the function requires the user to specify the format of the input data (adjacency matrix or edgelist) and provides two logical arguments for indicating whether the network is directed and whether it includes loops. The default for both of these arguments is `FALSE`, indicating that the network is undirected and does not include loops. It is important to note that if the network is undirected, the adjacency matrix should be either symmetrical or a triangular matrix. This means that the values in the matrix should be the same on both the upper and lower triangles, or that the values on the lower triangle should be empty. This is because an undirected network does not distinguish between "from" and "to" nodes, so the adjacency matrix should reflect this symmetry.

In case of a dynamic network a list of adjacency matrices or edgelists representing the network at different time points must be provided. The format of the input data (adjacency matrix or edgelist) and the other arguments of the `signnet` function (such as whether the network is directed and whether it includes loops) will be the same as for a static network.

The Tables 2 and 3 are two simple examples of what an input for the `signnet` function could look like. Both of these are an undirected network without loops consisting of three actors.

To demonstrate the use of the function, consider a small example. First, a random adjacency matrix is created without loops. In this case, we have an undirected network consisting of 10 actors for the static case, and for the dynamic case, we have an undirected network with 10 actors at four different time points, also without loops.

```
> # Static
> set.seed(1234)
> mat <- matrix(sample(c(-1,0,1),100,replace=TRUE),ncol=10)
> mat[lower.tri(mat)] <- 0
> diag(mat) <- 0

> # Dynamic
> set.seed(1234)
> lis <- c()
> n <- 4
> for (i in c(1:n)) {
+   mat <- matrix(sample(c(-1,0,1),100,replace=TRUE),ncol=10)
+   mat[lower.tri(mat)] <- 0
+   diag(mat) <- 0
+   lis[[i]] <- mat
+ }
```

After the data has been processed, it can now be passed to the function. The arguments directed and loops are superfluous in this case and only for demonstration purposes, as they are already set to `FALSE` by default.

```
> # Static
> static_net <- signnet(mat, directed = F, loops = F, matrix.type = "adjacency")
> class(static_net)
[1] "static.sign"

> # Dynamic
> dynamic_net <- signnet(lis, directed = F, loops = F, matrix.type = "adjacency")
> class(dynamic_net)
[1] "dynamic.sign"
```

Exogenous covariates can be added as vertex attributes to the network using the `cov` argument. The input for this should be a dataframe where the first column contains the names of the vertices, as specified in the `names` argument. If no list is provided for the `names` argument, the column names of the adjacency matrix will be used as names for the vertices. The other columns of the dataframe passed to the `cov` function represent the individual vertex attributes that will be added to the network, however it is not necessary to include all the vertices of the network in the dataframe passed to the `cov` function.

4.2.2 Descriptive Statistics of the Signed Network

The output of a `summary` function for a network includes several attributes that provide information about the structure and characteristics of the network. The "directed" attribute indicates whether the network is directed, meaning that the edges have a specific order and direction, or undirected, meaning that the edges do not have a direction or specific order. The "loops" attribute indicates whether the network contains any edges that connect a node to itself. The "nodes" attribute is the total number of nodes in the network, and the "edges" attribute is the total number of edges. The "+ edges" and "- edges" attributes indicate the number of positive and negative edges, respectively, in the network. The "triads" attribute is the number of triads in the network, which are sets of three nodes that are all connected to each other. The "+ + +," "- - -," "+ + -," and "+ - -" attributes indicate the number of triads with all positive edges, all negative edges, two positive and one negative edge, and two negative and one positive edge, respectively. Finally, the "density" attribute is a measure of how connected or how dense the network is, calculated as the ratio of the number of actual edges (positive or negative) to the number of potential edges.

```
> # Static
> summary.static.sign(static_net)
Network Attributes:
Directed   FALSE
Loops      FALSE
Nodes      10
Edges      32
   + edges 22
   - edges 10
Triads     44
   +++     12
   ---     0
   ++-     24
   +--     8
Density    0.71
```

```
> # Dynamic
> summary.dynamic.sign(dynamic_net)
           Time 1 Time 2 Time 3 Time 4
Directed   FALSE  FALSE  FALSE  FALSE
Loops      FALSE  FALSE  FALSE  FALSE
Nodes      10     10     10     10
Edges      29     28     27     32
   + edges 15     19     16     22
   - edges 14     9      11     10
Triads     27     25     23     44
   +++     4      8      4      12
   ---     2      1      0      0
   ++-     11     11     12     24
   +--     10     5      7      8
Density    0.64   0.62   0.6    0.71
```

Note that the function can also just be called with the generic `summary` function, instead of `summary.static.sign` and `summary.dynamic.sign`, since the class of the object has been set by the `signnet` function to `static.sign` and `dynamic.sign`.

4.2.3 Plot Signed Network

The importance of plotting a network lies in its ability to help in the understanding and analysis of the relationships and connections within a network. Plotting a network allows for the visualisation of the structure and layout of the network, which can aid in the identification of patterns, trends, and clusters.

In addition, it can also be useful for evaluating the performance of a model by comparing the simulated network to the actual network. By visualising both the simulated and actual networks, the structures and layouts can be compared to see how closely they match. If the simulated network is a good fit for the actual network, the two should be visually similar. On the other hand, if the simulated network is not a good fit, there may be significant differences in the structure and layout between the two networks. Comparing the simulated and actual networks in this way can help assess the quality of the model and determine whether it is a good fit for the data.

```
> # Static
> plot.static.sign(static_net, displaylabels = T)
```

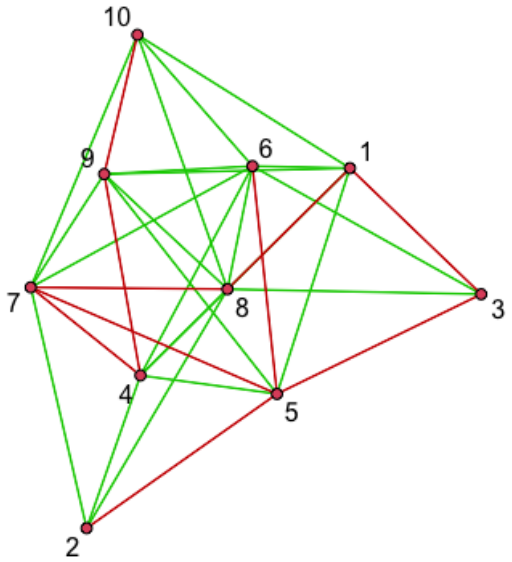


Figure 3: Visualisation of the static network.

```
> # Dynamic
> par(mfrow=c(2,2))
> plot.dynamic.sign(dynamic_net, displaylabels = T)
```

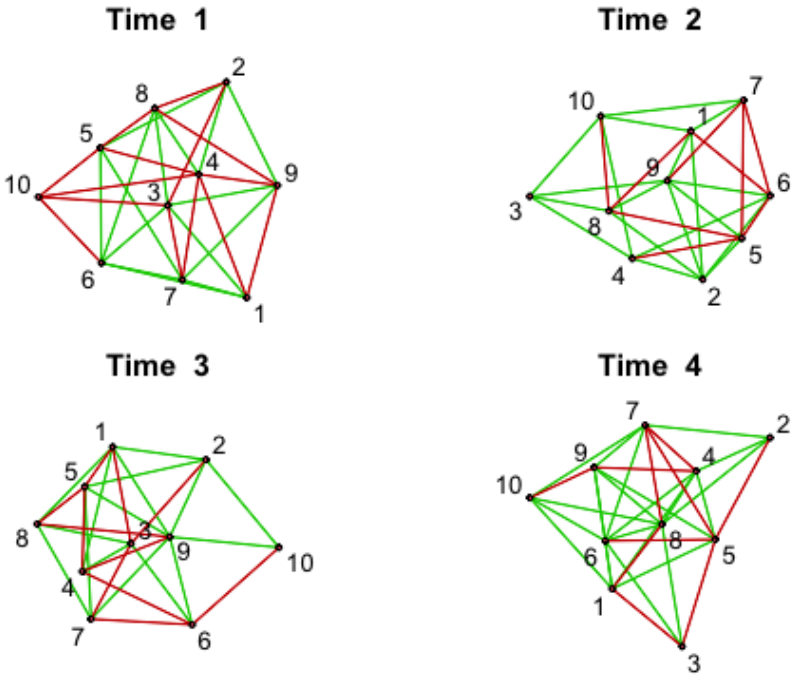


Figure 4: Visualisation of the dynamic networks.

The function takes a signed network object from the `static.sign` or `dynamic.static` class, as well as all the possible arguments from the `plot.network` function from the

network package (Butts, 2008b). By default, the positive and negative ties are coloured green and red, respectively, but these colours can be changed using the argument color_pos and color_neg.

The plot.network function has three additional arguments that allow for the creation of a legend for the vertex.col attribute if it is a vertex attribute. These arguments are vertex.legend, vertex.legend.pos, and vertex.legend.size. By default, vertex.legend is set to FALSE, but when set to TRUE, it will create a legend for the vertex colour. The legend's position can be specified with vertex.legend.pos using one of the following options: "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or "center". Additionally, the size of the legend can be adjusted with the vertex.legend.size argument, which takes a character expansion factor relative to the current par("cex").

To add a title to the dynamic.static plots, use the main argument, which takes a list of strings. If not specified, the plots will be named "Time 1" to "Time T". If no titles are desired, the input should be set to NULL.

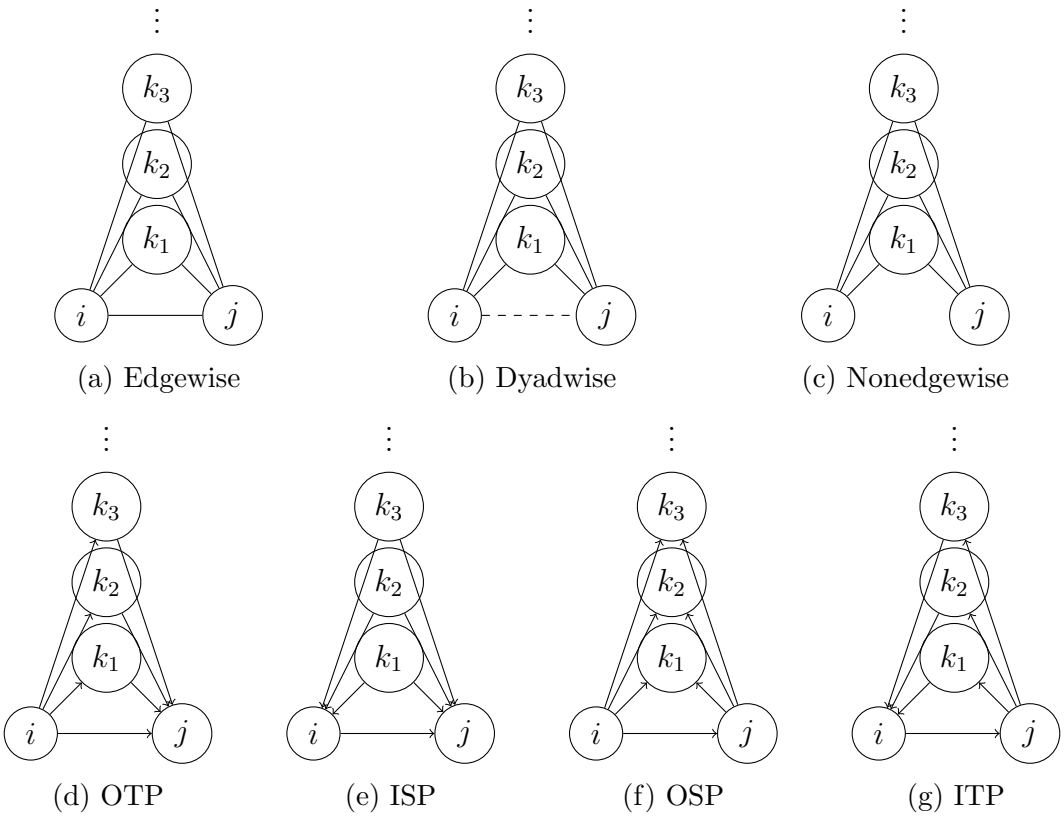
4.2.4 ERGM for Signed Networks

The SERGM function is the adaptation of the ergm function from ergm package for signed networks. The syntax remains the same, with the left-hand side (LHS) of the formula specifying an object of the class static.sign or dynamic.sign. The right-hand side (RHS) of the formula contains a description of the network terms that make up the ERGM, which are combined in an additive manner.

The naming convention for the network terms in SERGM includes the suffixes _pos and _neg, which indicate that the term only relates to positive or negative edges, respectively. If a network term does not have one of these suffixes, it relates to both positive and negative edges.

For example, the term gwdegree_pos(decay = 0.5) indicates that the model includes a term for the geometrically weighted degrees that only relates to positive edges, with a coefficient of 0.5. Whereas, the term edges indicates that the model includes a term for the number of edges in the network, and this term applies to both positive and negative edges.

In addition to the usual ergm terms, some terms have been added specifically for analysing signed networks in relation to structural balance theory. These terms examine the triads in the network, and include edgewise-shared friends/enemies (esf/ese), dyadwise-shared friends/enemies (dsf/dse), and nonedgewise-shared friends/enemies (nesf/nese). For edgewise and dyadwise-shared partners, a base can also be specified using the suf-



(a) Edgewise (b) Dyadwise (c) Nonedgewise

(d) OTP (e) ISP (f) OSP (g) ITP

Figure 5: Different type of triads.

fixes `_pos` and `_neg`. There are also geometrically weighted statistics available for the above-mentioned terms, as shown in the example below.

In an undirected network, there is only one type of shared partner. However, in a directed network, there are five different configurations that can be specified using the `type` argument. The default configuration is the Outgoing Two-path (`OTP`), which is also known as a "transitive shared partner". This configuration corresponds to the case where there is a path from i to k to j in the network, i.e. $i \to k \to j$. The other configurations are the Incoming Two-path (`ITP`) or "cyclical shared partner", which corresponds to the case where there is a path from j to k to i in the network, i.e. $j \to k \to i$; the Reciprocated Two-path (`RTP`), which corresponds to the case where there are bi-directional edges between i, k, and j, i.e. $i \leftrightarrow k \leftrightarrow$ j; the Outgoing Shared Partner (`OSP`), which corresponds to the case where there are directed edges from i to k and from j to k, i.e. $i \to k$, $j \to k$; and the Incoming Shared Partner (`ISP`), which corresponds to the case where there are directed edges from k to i and from k to j, i.e. $k \to i$, $k \to j$. (Butts, 2008a)

```
> set.seed(1234)
> model <- sergm(static_net ~ edges_pos + edges_neg +
+         gwesf_pos(decay=0.5, fix = T))
> model

Call:
"static_net ~ edges_pos + edges_neg + gwesf_pos(decay = 0.5, fix = T)"

Last MCMC sample of size 365 based on:
          edges_pos            edges_neg  gwesf.fixed.0.5_pos
            0.35640             -0.23038              0.05268

Monte Carlo Maximum Likelihood Coefficients:
          edges_pos            edges_neg  gwesf.fixed.0.5_pos
            0.61744             -0.27350             -0.04641
```

4.2.5 TERGM for Signed Networks

Temporal Exponential Random Graph Models (TERGMs) are a statistical approach that are used to analyse networks that change over time. While traditional Exponential Random Graph Models (ERGMs) model a single network at a single point in time, TERGMs typically involve two or more models for the evolution of ties in a network over time, each of these models being an ERGM itself.

There are two types of TERGMs: joint TERGMs and separable TERGMs.

Joint models specify a process for tie dynamics using operators that are dependent within time steps. In other words, a joint TERGM specifies how the ties between nodes in a network at a given time step depends on the ties between those nodes in the previous time step, as well as on other factors such as node attributes and network structure.

In the `tergm` package this is implemented with the `Cross()` + `Change()` models. The `Cross()` operator examines cross-sectional statistics across the network, while the `Change()` operator assesses changes in dyad status. Because the terms in these operators are related, they must be evaluated concurrently during the estimation or simulation process. (Morris et al., 2015)

```
> # Joint TERGM
> set.seed(1234)
> joint <- tsergm(dynamic_net ~
+                   Cross(~ edges_pos + ese_pos(d = 2)) +
+                   Change(~ edges_pos + ese_pos(d = 2)),
+                 estimate = "CMLE",
+                 times = c(0:3))
> joint

Call:
"dynamic_net ~ Cross(~edges_pos + ese_pos(d = 2)) + Change(~edges_pos + "

Last MCMC sample of size 553 based on:
            Cross~edges_pos    Cross~espe.Cross~esp2_pos
                   -0.05737                     -2.27236
           Change~edges_pos  Change~espe.Change~esp2_pos
                    0.02644                     -0.09323

Monte Carlo Maximum Likelihood Coefficients:
            Cross~edges_pos    Cross~espe.Cross~esp2_pos
                   -0.02014                     -2.16367
           Change~edges_pos  Change~espe.Change~esp2_pos
                    0.02922                     -0.11570
```

Separable models specify a process for tie dynamics using operators that are independent within time-step. Separable TERGMs specify how the ties between nodes in a network at a given time step depends only on factors such as node attributes and network structure at that time step, without considering dependencies with other parts of the network. These models are computationally more efficient than joint TERGMs, as they do not need to consider dependencies between different parts of the network. However, they may not be as appropriate in situations where there are dependencies between different parts of the network that need to be taken into account.

In the `Form()` + `Diss()` (or `Persist()`) model, dyads are split into two categories at each timepoint: those that are empty and subject to formation, and those that are tied and subject to dissolution. When a change occurs within a single dyad, it will only affect the `Form()` model statistics or the `Diss()`/`Persist()` model statistics for that time period, but not both. Therefore, the terms in these two operators are unrelated and can

be separately evaluated during the estimation or simulation process. (Morris et al., 2015)

```
> # Separable TERGM
> set.seed(1234)
> separable <- tsergm(dynamic_net ~
+                   Form(~ edges_pos + ese_pos(d = 2)) +
+                   Diss(~ edges_pos + ese_pos(d = 2)),
+                 estimate = "CMLE",
+                 times = c(0:3))
> separable

Call:
"dynamic_net ~ Form(~edges_pos + ese_pos(d = 2)) + Diss(~edges_pos + "

Last MCMC sample of size 627 based on:
         Form~edges_pos  Form~espe.Form~esp2_pos
               -0.08648                  0.02977
         Diss~edges_pos  Diss~espe.Diss~esp2_pos
                0.08395                 51.94752

Monte Carlo Maximum Likelihood Coefficients:
         Form~edges_pos  Form~espe.Form~esp2_pos
               -0.07809                  0.03154
         Diss~edges_pos  Diss~espe.Diss~esp2_pos
                0.14481                 51.94752
```

4.2.6 Count Network Statistics

The `count()` function enables users to easily obtain crucial network statistics from their network data by taking a formula as input, following the syntax of the formula specified for the `sergm` function. Where the LHS of the formula is an object of the class `static.sign` or `dynamic.sign`. Class `dynamic.sign` objects can also use the syntax for the `tsergm` function in addition to the `sergm` syntax. This is particularly beneficial when the statistic in question is not present in the output of `summary.static.sign()` or `summary.dynamic.sign()`.

```
> count(static_net ~ edges_pos + edges_neg + gwesf_pos(decay=0.5, fix = T))
          edges_pos           edges_neg gwesp.fixed.0.5_pos
           22.00000            10.00000            25.61488
```

```
> count(dynamic_net ~ edges_pos + edges_neg + gwesf_pos(decay=0.5, fix = T))
  edges_pos edges_neg gwesp.fixed.0.5_pos
1        15        14           10.180408
2        19         9           20.396142
3        16        11            9.941757
4        22        10           25.614882

> count(dynamic_net ~  Cross(~ edges_pos + ese_pos(d = 2)) +
         Change(~ edges_pos + ese_pos(d = 2)))
 Cross~edges_pos   Cross~esp2_pos Change~edges_pos  Change~esp2_pos
              57                1               67               13
```

Note that the baseline for the `tsergm` syntax is taken as the first timepoint. Hence, the cross-sectional term for positive edges is the sum of positive edges in timepoints 2, 3, and 4.

4.2.7 Simulation

Simulating a fitted ERGM or TERGM for a signed network is a key step in evaluating the fit of the model to the data, as noted in Section 4.2.3. By simulating the SERGM/TSERGM, it is possible to compare the simulated model to the observed data visually, as well as to use various goodness of fit measures based on the simulated model. This can help to determine how well the SERGM/TSERGM captures the patterns in the signed network data.

The simulation is implemented using `sim_sergm` and `sim_tsergm`. The functions take an object as input, which can be a formula or a fitted ERGM or TERGM, respectively, as well as some optional parameters. These optional parameters include `nsim`, which determines the number of simulations to run (default is set to 1), `seed`, which allows for the specification of a seed for the random number generator for reproducibility, and `coef`, which is used to specify the desired coefficients for the model.

```
> sim_static <- sim_sergm(mod, nsim = 10, seed = 123)
> summary(sim_static)
```

```
      Length Class       Mode
 [1,] 5      static.sign list
 [2,] 5      static.sign list
 [3,] 5      static.sign list
 [4,] 5      static.sign list
 [5,] 5      static.sign list
 [6,] 5      static.sign list
 [7,] 5      static.sign list
 [8,] 5      static.sign list
 [9,] 5      static.sign list
[10,] 5      static.sign list
```

Note that each element in the simulated list in the dynamic case is a list of networks with one fewer network than the model, as the model requires an initial network to be specified as a starting point for the simulation process. In the case of the `joint` model, it is based on a dynamic network with four time points, so each of the simulated dynamic networks will have three time points.

```
> sim_dynamic <- sim_tsergm(joint, nsim = 10, seed = 123)
> summary(sim_dynamic)
```

```
      Length Class        Mode
 [1,] 3      dynamic.sign list
 [2,] 3      dynamic.sign list
 [3,] 3      dynamic.sign list
 [4,] 3      dynamic.sign list
 [5,] 3      dynamic.sign list
 [6,] 3      dynamic.sign list
 [7,] 3      dynamic.sign list
 [8,] 3      dynamic.sign list
 [9,] 3      dynamic.sign list
[10,] 3      dynamic.sign list
```

As already mentioned, in addition to the simulation based on a fitted SERGM or TSERGM, the functions `sim_sergm` and `sim_tsergm` also offer the possibility to simulate networks via formulas, which will be shown in Section 4.3.

4.2.8 Goodness-of-Fit

In order to evaluate the model fit to the actual data, so called goodness-of-fit plots are being utilised. The goodness-of-fit (GOF) plots visually display the discrepancy between observed values and simulated values from the model. The observed values are represented by a line in the plot, while the simulated values are depicted through boxplots. Ideally, the line should pass through the center of the boxplot, with the median of the simulated values aligning with the line, indicating a good fit between the observed and simulated values. If the boxplots deviate significantly from the line, it may suggest inadequate model fit and warrant further investigation.

For a static setting, the function `gof_sergm` can be used to evaluate the fit of an ERGM to signed networks. This function simulates the given fitted SERGM model n number of times (default is set to 200) and calculates various network statistics for both the simulated networks and the network the model is based on. These statistics include positive and negative degrees, k-edgewise shared enemies with negative and positive bases ("the enemy of my enemy is my enemy/friend"), and k-edgewise shared friends with negative and positive bases ("the friend of my friend is my enemy/friend").

Note that in this example, due to the small network and simple model, the fit is not expected to be great, as seen in Figure 6 and 8.

```
> set.seed(1234)
> gof_sergm(model)
```

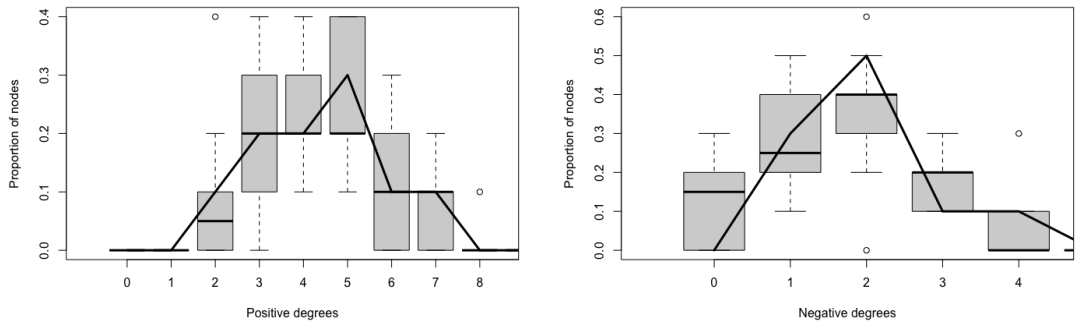


Figure 6: Goodness of fit of the SERGM.

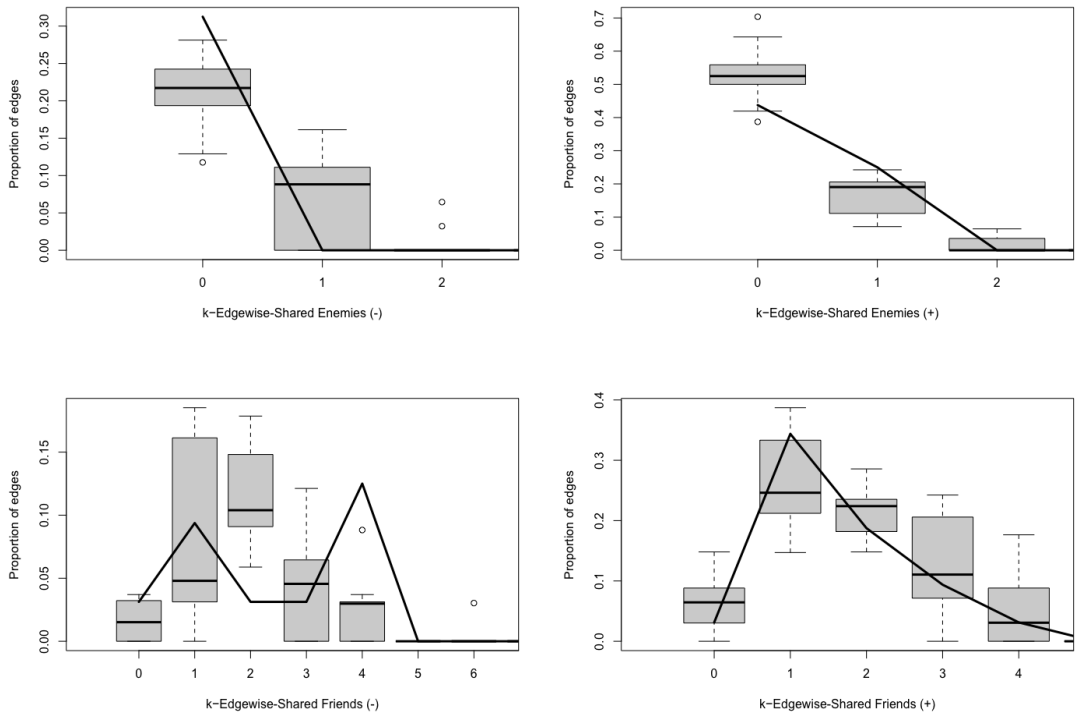


Figure 6: Goodness of fit of the SERGM.

Since the `sergm` and `tsergm` function are based on a MCMC algorithm, another function that can be used to assess the convergence and goodness-of-fit of such MCMC simulations is the `mcmc.diagnostics` function. Which provides diagnostic plots and summaries for a given MCMC object to help determine if the MCMC algorithm has reached stationarity and if the posterior distribution is well-approximated by the simulation.

```
> mcmc.diagnostics(model)
```

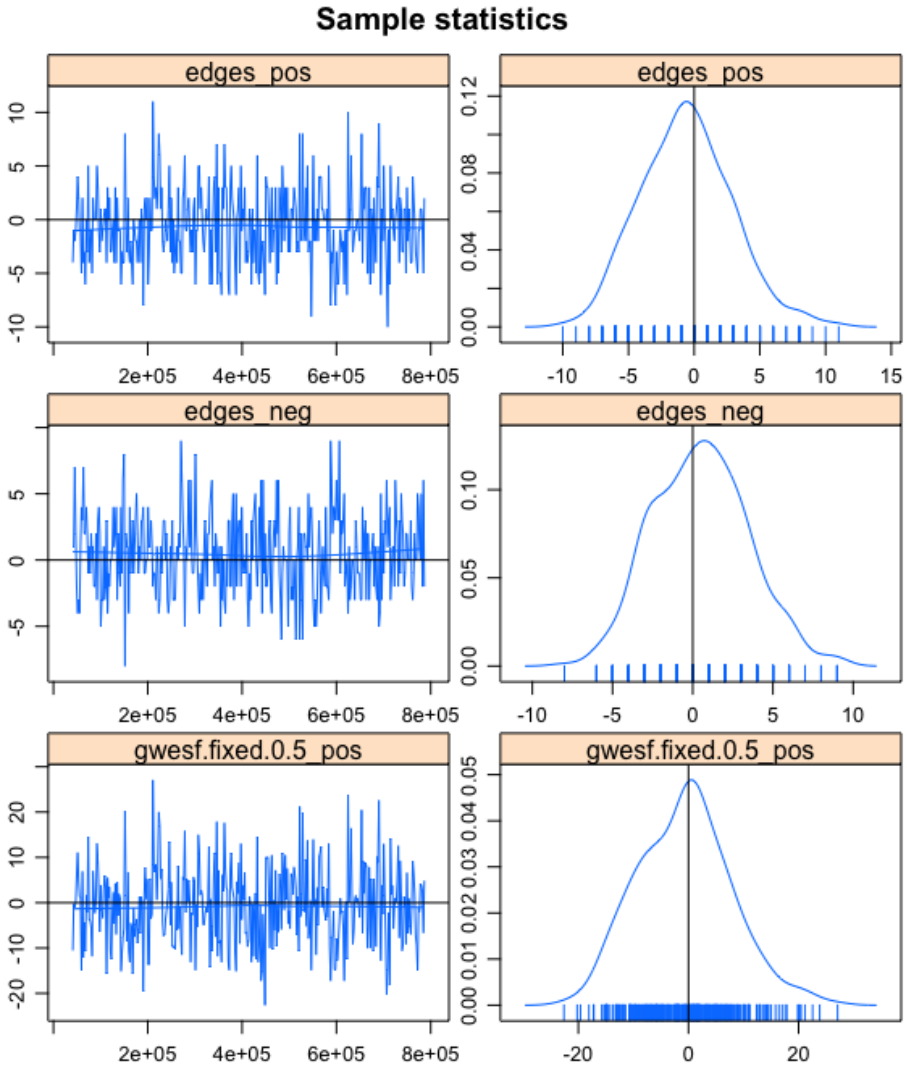


Figure 7: MCMC diagnostics for the SERGM.

In the dynamic case, instead of plotting the proportion of the sufficient statistics of the observed and simulated networks, the average of the sufficient statistics of the observed and simulated networks will be taken. This means that the sum of sufficient statistics is divided by the number of temporal networks, which is one less than the number of time points in the observed network. This has been previously explained in Section 4.2.7.

```
> set.seed(1234)
> gof_tsergm(joint)
```

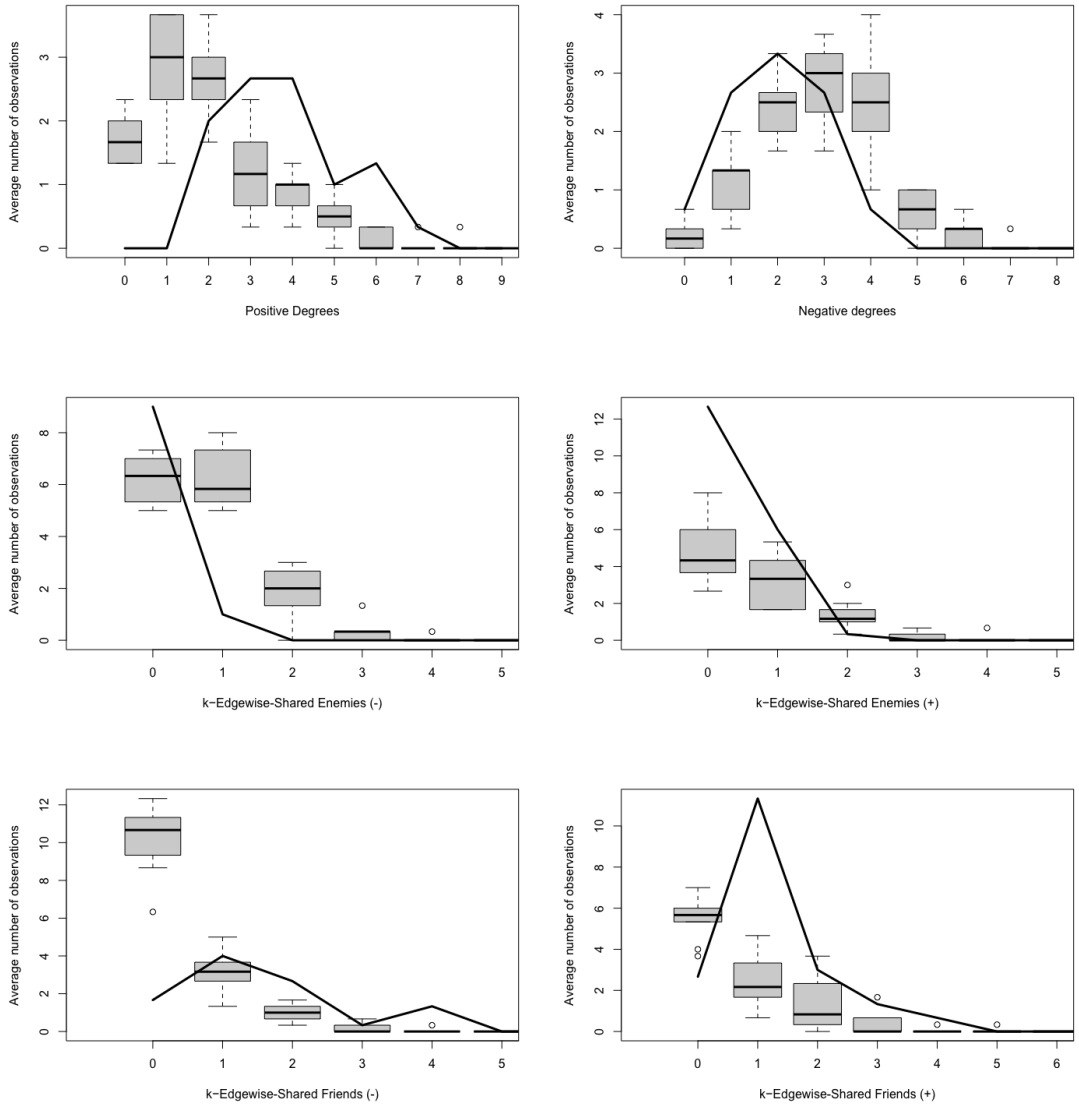


Figure 8: Goodness of fit of the TSERGM.

```
> set.seed(1234)
> mcmc.diagnostics(joint, vars.per.page = 4)
```

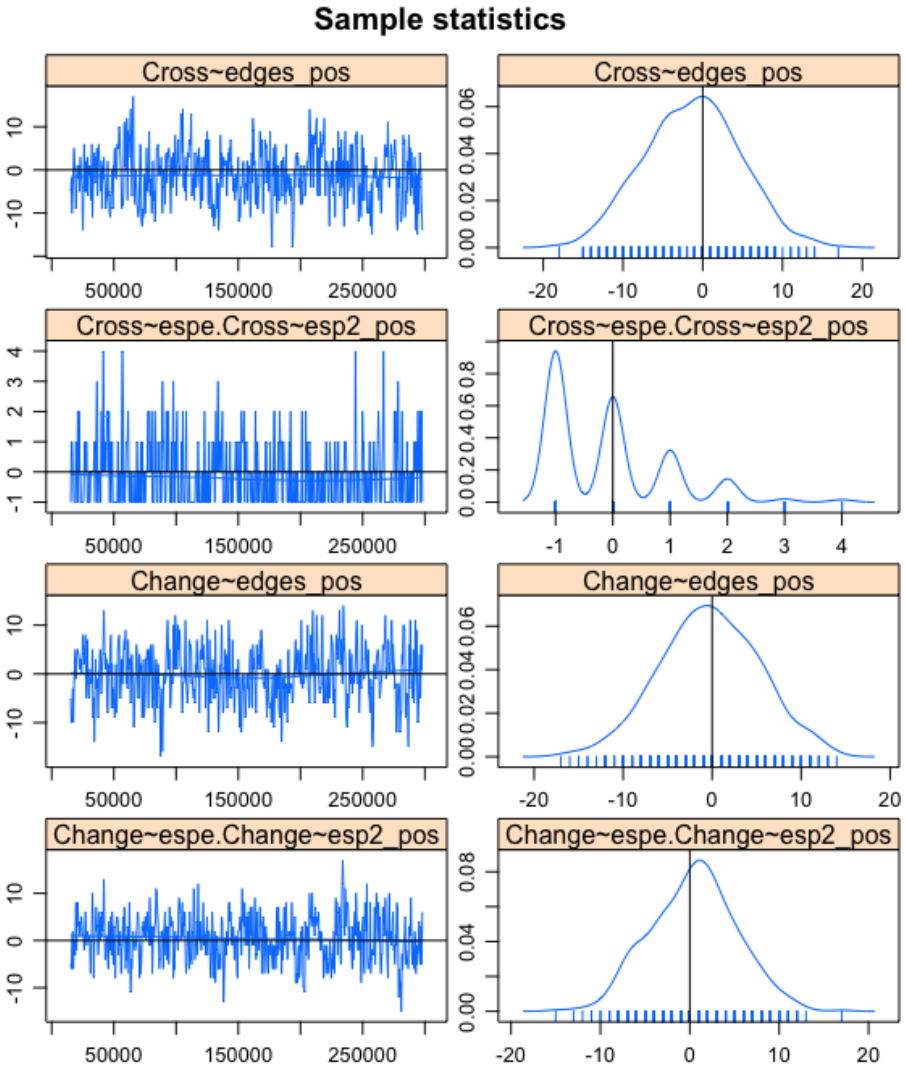


Figure 9: MCMC diagnostics for the TSERGM.

4.3 Simulation Sanity Check

The `sergm` function, and other functions that employ an MCMC algorithm, can be difficult to evaluate for accuracy because it is a "black box", meaning that the internal workings of the algorithm are not directly observable, and it can be challenging to assess how well the algorithm is performing or whether it is producing valid results. To ensure the estimated coefficients are accurate, it is crucial to conduct simulation tests where random networks are generated with set coefficients and then see if the `sergm` function can correctly recover those coefficients.

In the following example, an undirected network with 50 nodes will be created. The coefficients for the statistics of positive edges, negative edges, and geometrically weighted edgewise shared friends with a positive base and a fixed decay parameter of 0.3 ("friend of my friend is my friend") are -3.9, -3, and 1, respectively.

```
> test1 <- sim_sergm(network.initialize(50, directed = F) ~
+                                edges_pos +
+                                edges_neg +
+                                gwesf_pos(decay=0.3, fix = T),
+                               coef = c(-3.9,-3,1),
+                               seed = 1234)

> set.seed(1234)
> sergm(test1 ~  edges_pos + edges_neg + gwesf_pos(decay=0.3, fix = T))

Call:
"test1 ~ edges_pos + edges_neg + gwesf_pos(decay = 0.3, fix = T)"

Last MCMC sample of size 558 based on:
          edges_pos            edges_neg  gwesf.fixed.0.3_pos
            -3.8376              -2.9617               0.7554

Monte Carlo Maximum Likelihood Coefficients:
          edges_pos            edges_neg  gwesf.fixed.0.3_pos
            -3.8366              -2.9477               0.7623


> count(test1 ~  edges_pos + edges_neg + gwesf_pos(decay=0.3, fix = T))
          edges_pos           edges_neg gwesp.fixed.0.3_pos
                 30                  59                   6
```

As can be seen in the example above, the `sergm` function is able to accurately identify the coefficients. It is to be expected that the coefficients will not be exactly the same, so it is up to the researcher to decide if the resulting coefficients are close enough to the actual ones. However, it is important to repeat this process with different statistics to ensure that everything is working correctly.

Therefore, an undirected network with 60 actors will be simulated. The statistics of positive triangles, negative triangles, and geometrically weighted dyadwise shared enemies with a fixed decay parameter of 0.2 will be used with the set coefficients of -2, -2.5, and 1.

```
> test2 <- sim_sergm(network.initialize(60, directed = F) ~
+                                triangles_pos +
+                                triangles_neg +
+                                gwdse(decay = 0.2, fix = T),
+                               coef = c(-2,-2.5,1),
+                               seed = 1234)

> set.seed(1234)
> sergm(test2 ~  triangles_pos +
+         triangles_neg +
+         gwdse(decay = 0.2, fix = T))

Call:
"test2 ~ triangles_pos + triangles_neg + gwdse(decay = 0.2, fix = T)"

Last MCMC sample of size 486 based on:
   triangle_pos     triangle_neg  gwdse.fixed.0.2
         -2.393           -2.430            1.074

Monte Carlo Maximum Likelihood Coefficients:
   triangle_pos     triangle_neg  gwdse.fixed.0.2
         -2.376           -2.363            1.041


> count(test2 ~ triangles_pos +
+         triangles_neg +
+         gwdse(decay=0.2, fix = T))
       triangle_pos        triangle_neg     gwdse.fixed.0.2
             13.000             140.000            1928.098
```

In this example, the `sergm` function also managed to estimate the coefficients relatively accurately. These two examples are not a definitive proof but seem to indicate with a relatively high degree of certainty that the function is able to estimate the coefficients accurately.

It is possible to repeat the identical process for the dynamic case and the `tsergm` function. Given that they both utilise the same framework, it is logical to presume that if the process is successful for the static case, it would also apply to the dynamic case, and therefore it will not be reiterated here.

5 Application

5.1 Data

The dataset that serves as the basis for this application part comes from the Fritz et al. (2021) paper in which a Relational Event Model for Spurious Events (REMSE) is being applied to two examples, one of which pertains to combat events from the Syrian civil war. The raw data for the application part of the paper comes from the Armed Conflict Location & Event Data Project (ACLED).

ACLED collects data on political violence and demonstration events in all countries and territories worldwide by political agents, including governments, rebels, militias, identity groups, political parties, external actors, rioters, protesters, and civilians. ACLED data includes specific information on the location, date, involved actors, fatalities, and other characteristics of the event. (Raleigh and Dowd, 2015)

The exogenous covariates which, in addition to the endogenous effects, play a significant role in the formation and development of the conflict network also come from the paper by Fritz et al. (2021). The exogenous covariates include indicators of the group's ethno-religious identity and external sponsor support, both of which have been shown to reduce the risk of conflict if the two nodes have the same value (Popovic, 2018, Gade et al., 2019). Also included is an indicator of the conflict parties' type, such as rebel groups or state forces, which also affects the likelihood of conflict (Dorff et al., 2020).

The first step will be to clean the dataset. Only violent conflicts will be considered, and events classified as "Strategic developments" will be excluded. Although these events are important for understanding the context, they are not directly related to the violent conflicts of interest. Strategic developments can include arrests of political figures, mass arrests, protests, peace negotiations, hunger strikes, other strikes, recruitment, looting, and property destruction (Raleigh and Dowd, 2015). Furthermore, events where a group is fighting against itself or supporting itself in a fight will be removed. To be included in the analysis, a conflict group must have participated in at least four conflicts for the dynamic analysis and in at least seven conflicts for the static analysis as one of the fighting parties or as a supporting party. Any parties for which exogenous covariates are not available will also be excluded, as the current implementation of the ERGM model cannot handle missing data in vertex attributes. Undefined militas, rioters and rebels such as "Kurdish Ethnic Militia" and "Opposition Rebels" will also be excluded.

After completing the data cleaning process, an adjacency matrix will be created from the data. If two actors fought against each other in a specified year, there will be a -1 in the corresponding cell of the adjacency matrix. Conversely, if the two actors supported

	Static	Dynamic		
	2019	2017	2018	2019
Nodes	34	28	28	28
Edges	72	70	50	45
+ edges	28	19	13	10
- edges	44	51	37	35
Triads	31	64	24	13
+ + +	4	3	1	2
- - -	5	25	13	5
+ + -	10	4	1	0
+ - -	12	32	9	6
Density	0.13	0.19	0.13	0.12

Table 4: Network attributes for the static and dynamic network.

each other in a conflict during that year, there will be a +1 in the same cell. It is worth noting that if there are multiple events between two actors, only the last one will be taken into account. For example, if two actors initially fought against each other but later in the year supported each other, it will be considered a positive edge between the two.

It is notable that since the networks are derived from a conflict dataset, the negative ties will naturally outweigh the positive ones. This imbalance between positive and negative ties may have important implications for the analysis and interpretation of the results. This imbalance can be observed in Figure 10 and 11, as well as Table 4.

For the application part of this thesis, a static network, representing a single year, and a dynamic network, consisting of multiple years, will serve as the foundation of this analysis. The data for ACLED in Syria dates back to 2017, therefore, 2019 is selected for the static network and 2017, 2018, and 2019 are chosen for the dynamic network. However, to use a TERGM, the networks must be of the same size. To achieve this, only actors involved in over four conflicts in all three years are included, resulting in a smaller set of actors in the dynamic network compared to the static network. A summary of the network statistics for the static network as well as the dynamic network is presented in Table 4.

Due to the limitation that all actors have to be present in each timepoint, there is a significant decrease of nodes and edges compared to the static model as can be seen in the Figure 11 and the Table 4. Therefore the number of nodes are the same over the three years. However the networks do appear to decrease in size over time, with fewer edges and lower number of triads, but becoming more connected as reflected by the increasing density over time.

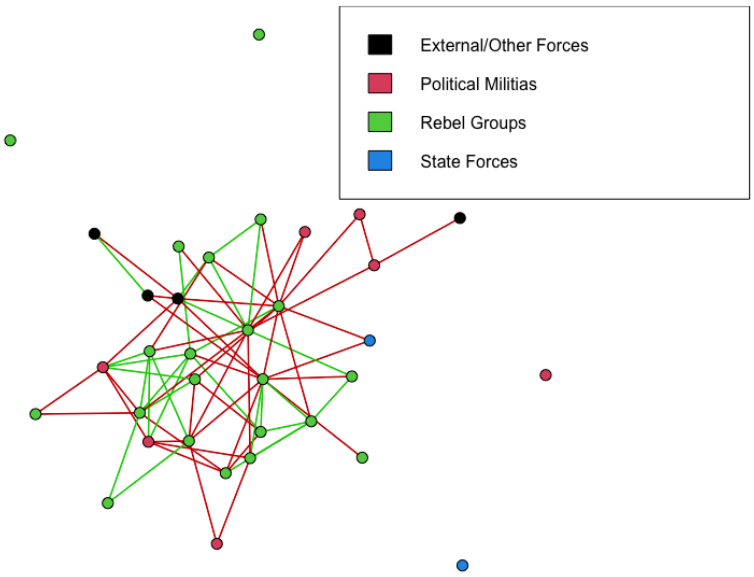


Figure 10: Signed network of the Syrian civil war in 2018.

5.2 Model Specification

The choice of which effects to include in the ERGM model is one of the biggest challenges, as it requires solid theoretical knowledge, particularly when dealing with endogenous effects.

One can choose the endogenous network statistics by either postulating a specific network dependence one wants to account for or by the substantive theory one wants to test. However, some structures are deemed crucial, and it is clear that these should be included in network models. By using a statistic that measures the number of ties in the network, one can account for network density. The inclusion of the number of triangles or geometrically weighted triangles is usually a standard component of an ERGM, in the context of SERGMs and structural balance theory, the inclusion of the four triadic terms is particularly important.

Therefore, in addition to the positive and negative edges, the geometrically weighted edgewise shared friends and enemies with positive and negative bases are included as endogenous effects in the model. Based on the structural balance theory introduced in Section 3.3, the coefficients for $GWESF^+$ and $GWESE^+$ are expected to be positive and statistically significant, as these triads are considered balanced.

Besides the endogenous effects, exogenous effects are essential to model real-life data. It seems reasonable to assume that actors might be more inclined to fight alongside those who share the same sponsor, ethno-religious background, or group type. On the other hand, conflicts might occur more frequently when these conditions are not met.

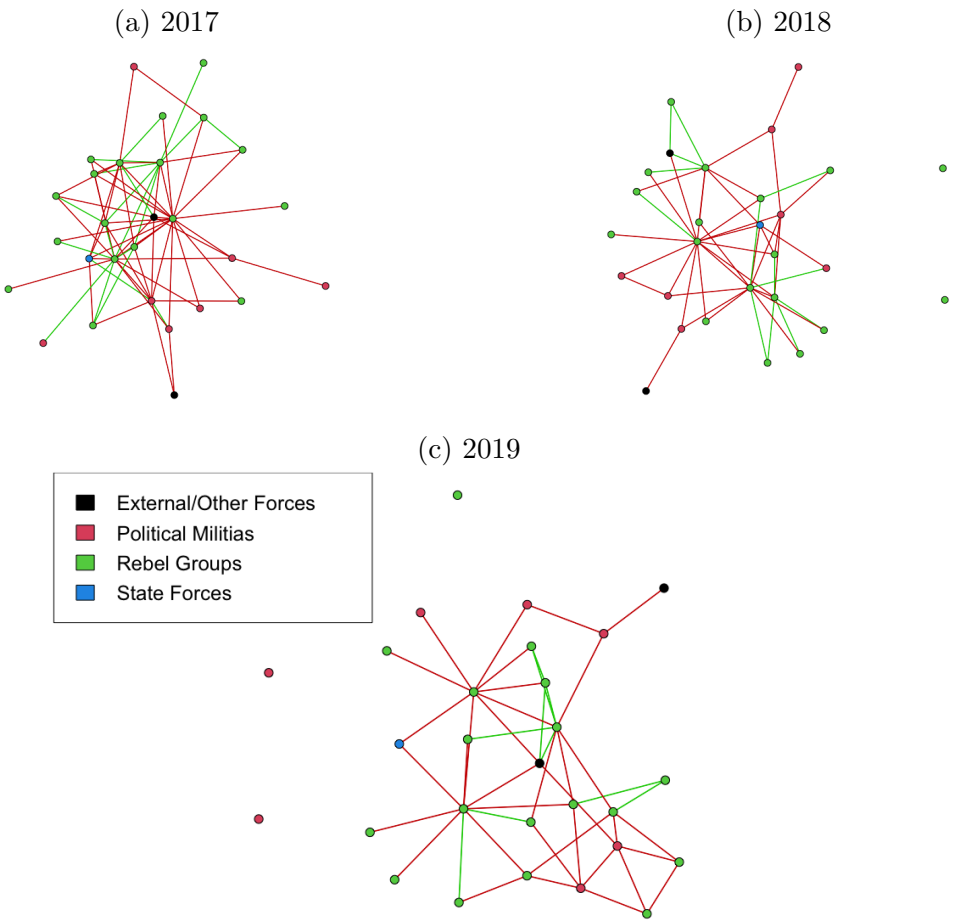


Figure 11: Signed network of Syrian civil war from 2017 to 2019.

These effects can be captured with homophily effects. Homophily is the tendency of an actor to have a tie with another actor if they share similar characteristics. Fellows (2012) showed that even these homophily terms are sensitive to degeneracy in models for random ties and attributes.

For the TSERGM, the same statistics as for the SERGM will be included as cross-sectional statistics and additionally the variables positive and negative edges will be included in the Change operator. Which means that the likelihood of having a positive or negative tie in the current timepoint is affected by the presence or absence of positive and negative ties in the previous timepoint.

5.3 Results

The results of the SERGM and TSERGM are displayed in Table 5 and will be interpreted and discussed in this Section.

The interpretation of the estimated coefficients is analogous to the ones of a logistic

	SERGM		TSERGM	
(Cross-) Statistics	Coef.	CI	Coef.	CI
Edges +	-4.216 ***	[-5.002,-3.43]	-3.460 ***	[-4.505,-2.415]
Edges -	-2.927 ***	[-3.466,-2.388]	-2.158 ***	[-2.656,-1.66]
GWESF + (0.25)	0.554 *	[0.052,1.055]	0.834 **	[0.216,1.452]
GWESF - (0.25)	0.494	[-0.216,1.204]	-0.969	[-2.902,0.963]
GWESE + (0.25)	0.751 *	[0.069,1.432]	0.986 *	[0.217,1.754]
GWESE - (0.25)	0.188	[-0.199,0.575]	0.280 .	[-0.013,0.574]
Common Sponsor +	0.061	[-0.901,1.023]	-0.327	[-1.761,1.106]
Common Sponsor -	-0.367	[-1.276,0.541]	0.191	[-0.562,0.944]
Match Ethno-Religion +	0.208	[-0.58,0.997]	-0.163	[-1.236,0.91]
Match Ethno-Religion -	-1.051 *	[-1.882,-0.219]	0.129	[-0.544,0.803]
Match Type +	1.071 *	[0.208,1.935]	0.809	[-0.253,1.871]
Match Type -	0.827 **	[0.213,1.441]	0.200	[-0.337,0.736]
(Change) Edges +			-1.403 ***	[-1.902,-0.904]
(Change) Edges -			-1.310 ***	[-1.602,-1.017]

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Table 5: Estimated coefficients and confidence intervals of SERGM and TSERGM.

regression. Thus, if a coefficient is positive, the odds of a tie increase; if it is negative, the odds of a tie decrease. For example, in the case of the SERGM, for every increase in the number of positive ties, the log-odds of a positive tie decreases on average additively by 4.216.

$$\begin{aligned} \text{logit}\left(y_{ij} = +1 \mid y_{ij}^{c}\right) &= \theta' \delta_{pos}^{(ij)}(y) \\ &= -4.216 * \text{change in number of positive ties} \\ &= -4.216 * 1 = -4.216 \end{aligned} \tag{14}$$

The corresponding probability is obtained by taking the inverse logit, of θ:

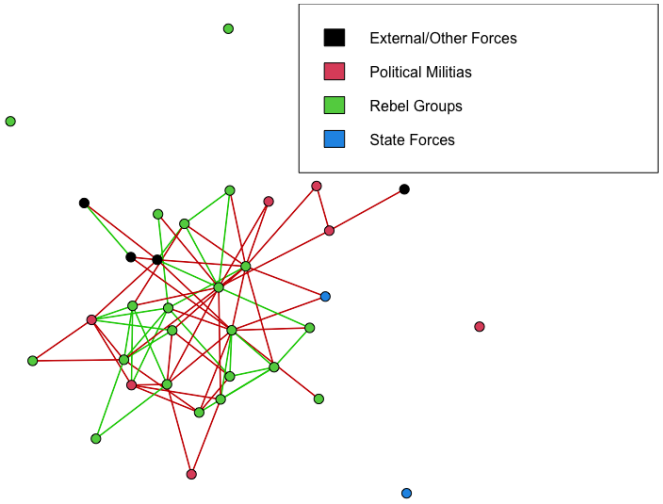
$$\begin{aligned} &= \frac{\exp(-4.216)}{1 + \exp(-4.216)} \\ &\approx 0.015 \end{aligned} \tag{15}$$

The fact that the conflict networks used to estimate both the SERGM and the TSERGM contain more negative ties than positive ties, is reflected in the estimated coefficients of these models. Specifically, the estimated coefficient for the positive edges is smaller than the estimated coefficient for the negative edges in both models. Which implies that the presence of a positive tie reduces the likelihood of observing additional positive ties more than the presence of a negative tie reduces the likelihood of observing

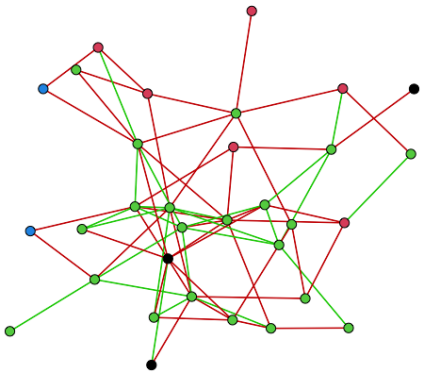
additional negative ties in the network.

As mentioned in Section 5.2, based on structural balance theory, the coefficients $GWESF^{+}$ and $GWESE^{+}$ are expected to be positive and statistically significant. This can be seen in both the SERGM and TSERGM, with both triads that are considered balanced exhibiting positive and significant coefficients, where neither confidence interval includes zero. In contrast, the coefficients for the unbalanced triads are smaller and less significant. In the case of $GWESE^{-}$ in the TSERGM, the coefficient is still positive but less significant, with a confidence interval including zero. This means that in the dynamic network the actors seem to have a tendency of enemies of enemies being enemies. Even though this is contrary to what early structural balance theory stated (Heider, 1946, Cartwright and Harary, 1956), later research by Davis (1967) suggests that this type of triad is only imbalanced in networks with two subsets of nodes, which is not the case in this study. Therefore, the positive and significant coefficient of $GWESE^{-}$ aligns with the predictions of structural balance theory.

Moreover, it is noteworthy to mention that only the SERGM model estimates statistically significant coefficients for some of the exogenous effects. The coefficients for the homophily terms for group type are positive and significant, suggesting that there is a tendency for nodes to form ties, whether positive or negative, with others who are similar to them in terms of their type. The only other significant exogenous effect is the homophily term for ethno-religion for negative ties, which is negative, meaning that groups are less likely to fight against a group having the same ethno-religious background. In the TSERGM, none of the coefficients for any of the exogenous effects exhibit significance. This finding is particularly interesting, as the theory behind conflicts of rebel groups would suggest otherwise (Popovic, 2018, Gade et al., 2019, Dorff et al., 2020).



(a) Observed.



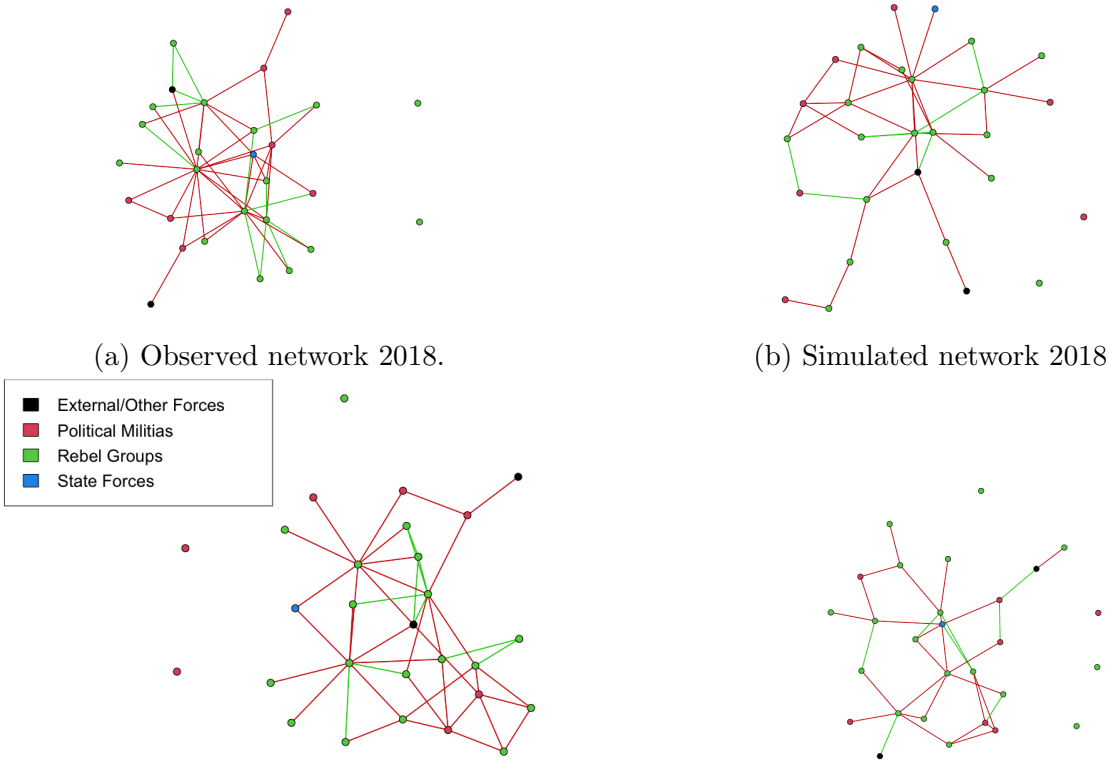
(b) Simulated.

Figure 12: Observed network and simulated network in 2019.

5.4 Model Assessment

As explained in Section 4.2.8, a way to evaluate the model fit of a SERGM or TSERGM is to compare the observed network statistics with those obtained from simulated networks generated from the model. Figures 14 and 15 show the goodness-of-fit assessments of the models by plotting the observed network statistics against 1000 simulated networks for each model. An ideal model fit would be represented by a line passing through the center of the boxplots, indicating that the model generates simulations that closely match the observed values, thus indicating a good fit.

Based on the goodness-of-fit assessments presented in Figures 14 and 15, it appears that both the SERGM and TSERGM provide a good fit to the observed networks. In both



(a) Observed network 2018.

(b) Simulated network 2018.

(c) Observed network 2019.

(d) Simulated network 2019.

Figure 13: Observed networks and simulated networks in 2018 and 2019.

Figures, the line representing the model-predicted values roughly aligns with the center of the boxplots, which represent the observed values. This indicates that the models are able to uncover the underlying network dynamics relatively consistently.

Another way to evaluate the goodness-of-fit of a model is through a qualitative and visual inspection of the simulated network's similarity to the observed network. At first glance, one can see that the simulated network, in Figure 12 and 13, has characteristics similar to those of the observed network, which suggests an appropriate model fit, since networks simulated from the fitted probability distribution will be more likely to resemble the observed network if the model is a good fit to the actual data.

The MCMC diagnostic plots for the SERGM and the TSERGM can be found in the Appendix A and will not be discussed further here.

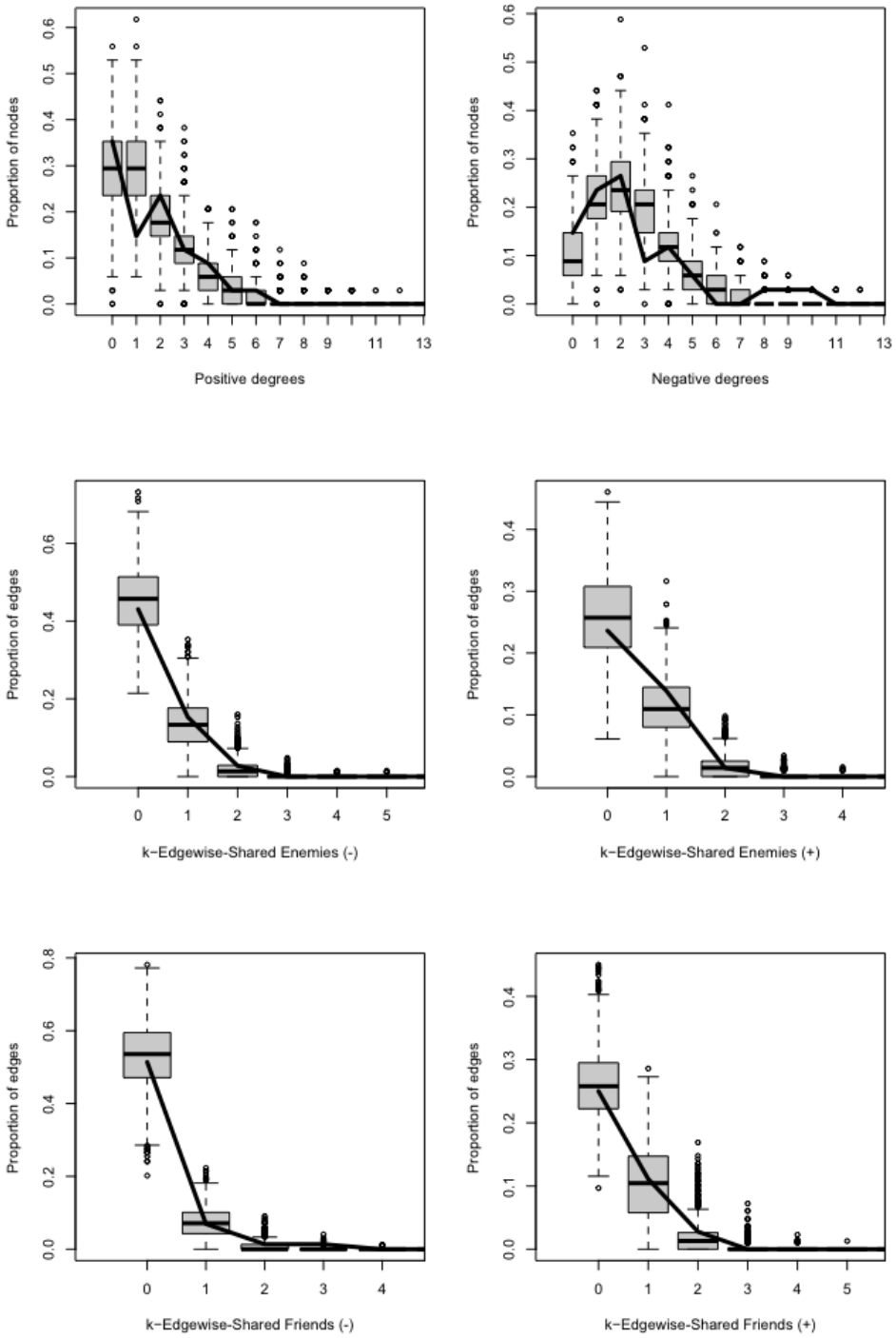


Figure 14: Goodness-of-Fit for SERGM.

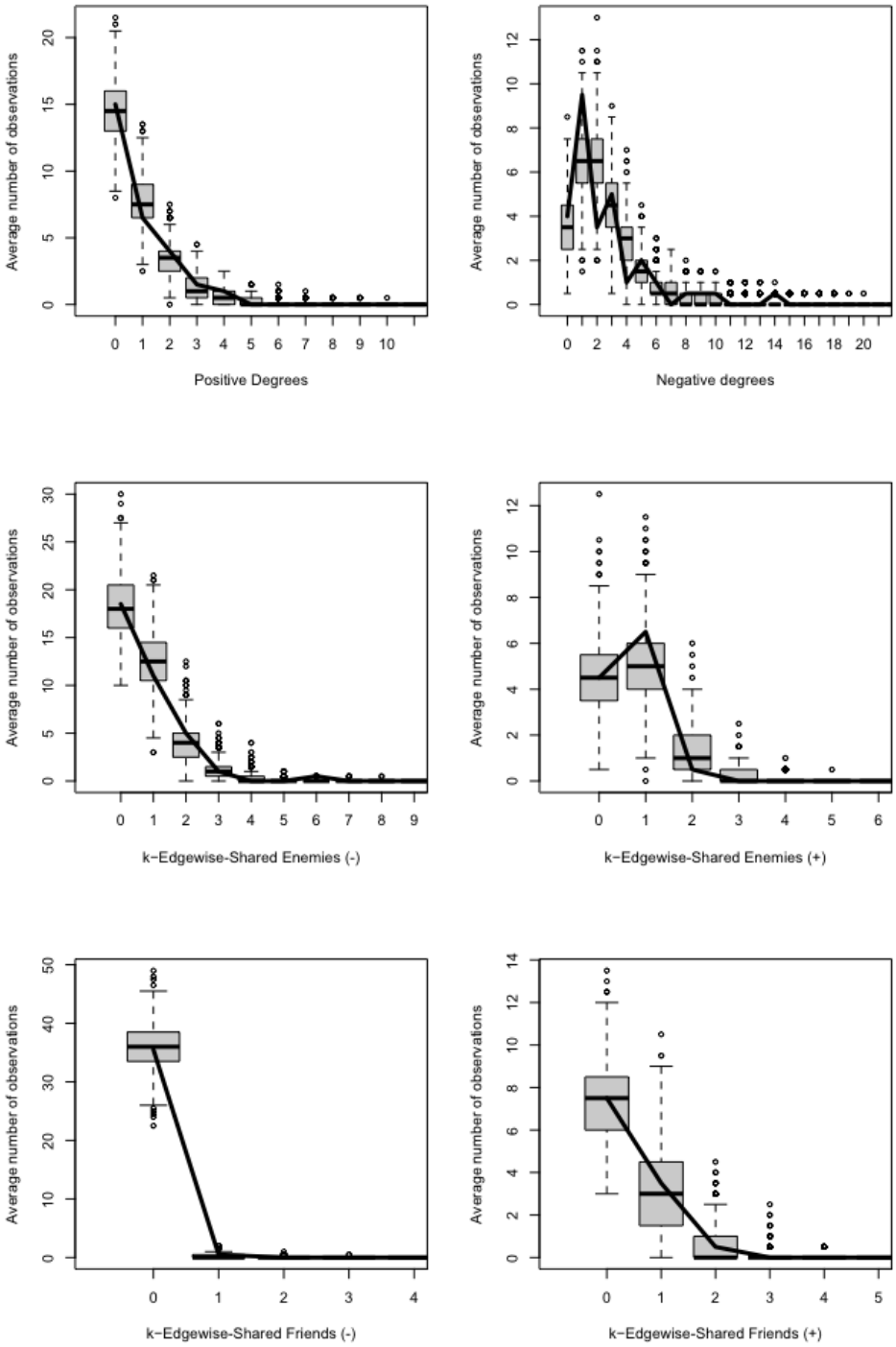


Figure 15: Goodness-of-Fit for TSERGM.

6 Conclusion

The thesis starts with an overview of network analysis and establishes the theoretical framework for the ERGM. It also covers the ERGM extension for signed networks, as well as providing a brief introduction to structural balance theory in connection with the SERGM. The implementation process is then explained, starting with an introduction to multilayer networks and a detailed explanation of each function within the `ergm.sign` package. Finally, the package is utilised to analyse rebel data as a signed network.

The application revealed several noteworthy findings. As predicted by the structural balance theory, the triads thought to be balanced displayed positive, strong, and significant coefficients. Notably, while the TSERGM exhibited insignificant estimation for the exogenous effects, the SERGM did not. One possible explanation could be that the inclusion of positive and negative edges in the STERGM's Change operator may reduce the explanatory power of the exogenous covariates, especially if the past state of the network (i.e. the ties in the previous timepoint) is a stronger predictor of the current state of the network than the exogenous variables.

A software package is never completely finished, but rather a permanent work in progress, either due to bugs that appear or due to improvements being made. This is also the case for the `ergm.sign` package. That being said, as shown in Section 5, it is able to handle real-life data and can be a valuable asset to analyse structural balance theory and other research questions in the context of signed networks.

Some possible extensions to the `ergm.sign` package would be to account for local dependencies as proposed by Schweinberger and Handcock (2015) or the possibility to obtain model-predicted conditional and unconditional tie probabilities for dyads in the given network.

A Appendix

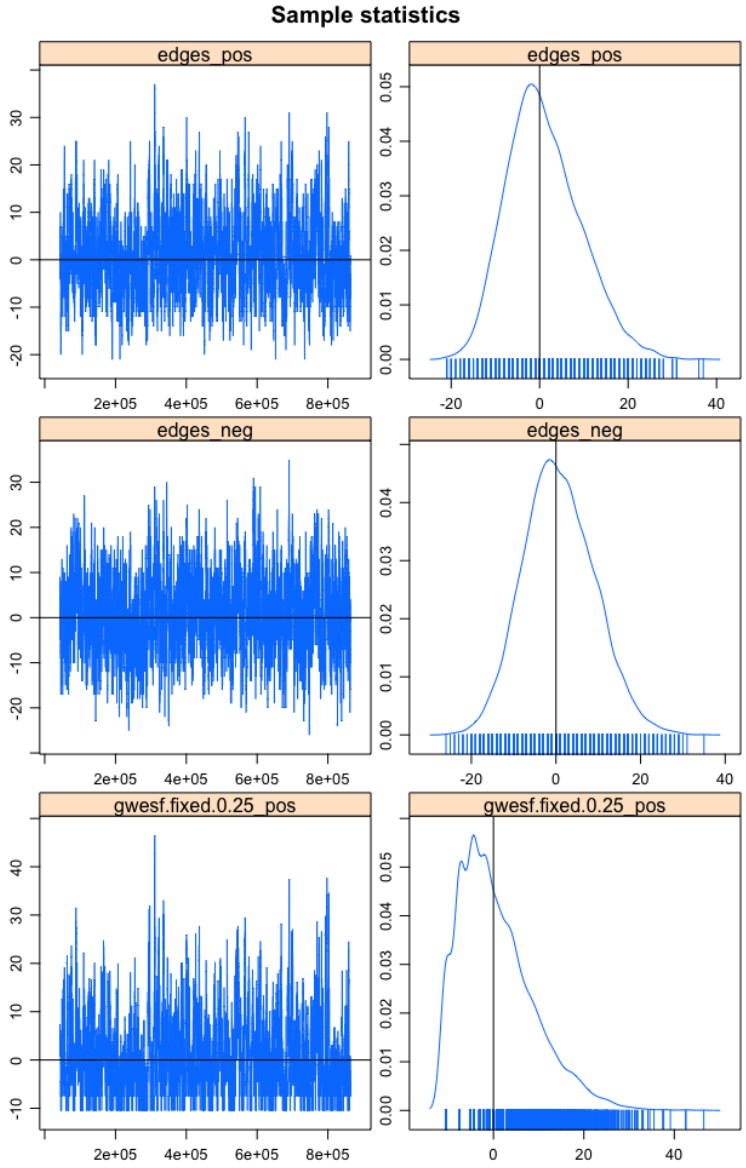


Figure 16: MCMC diagnostics for the SERGM model based on the 2019 network.

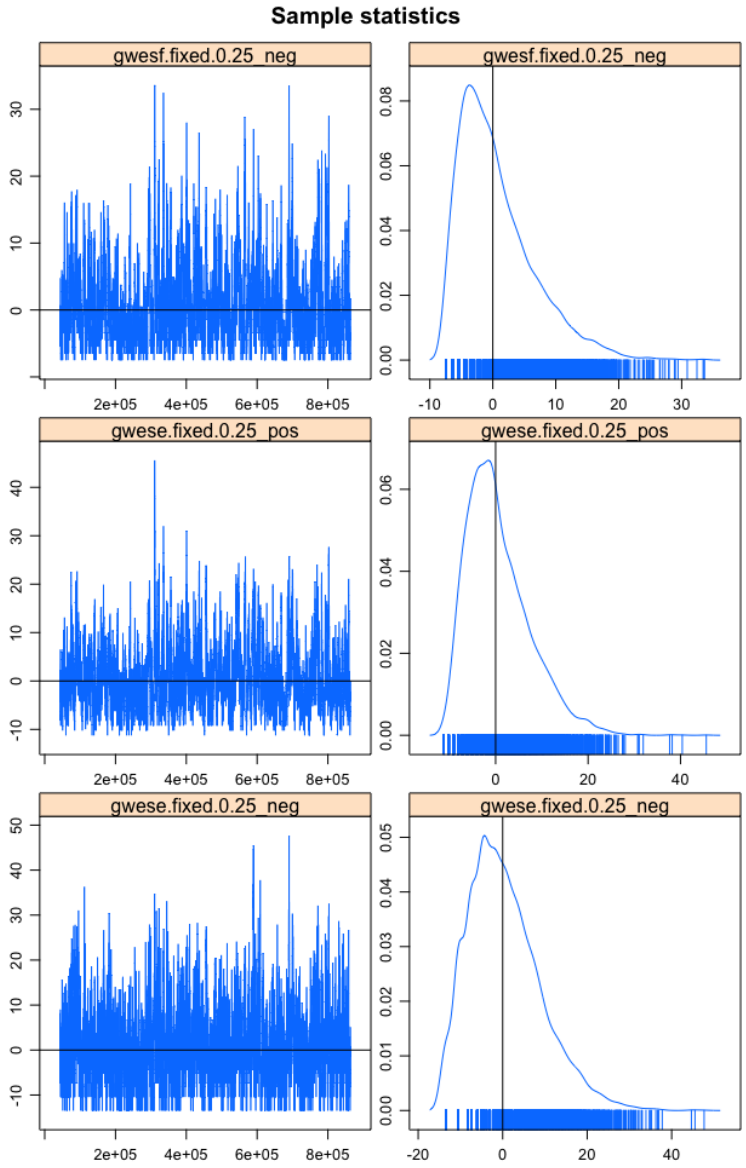


Figure 16: MCMC diagnostics for the SERGM model based on the 2019 network.

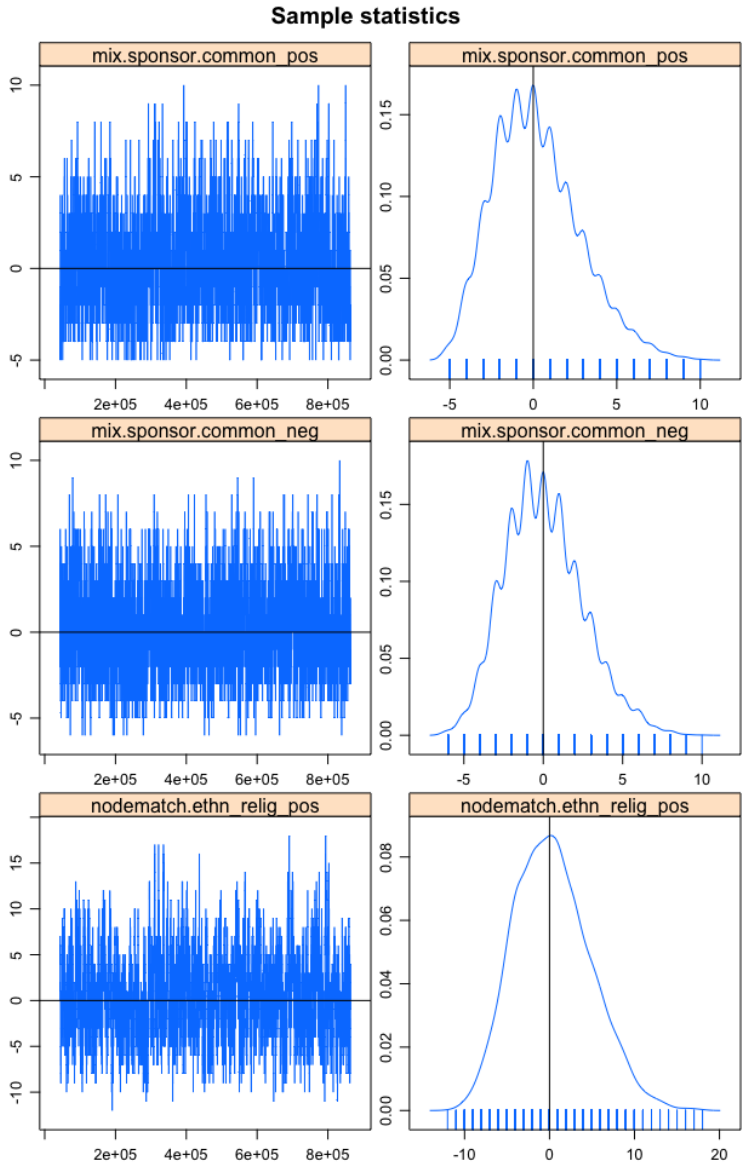


Figure 16: MCMC diagnostics for the SERGM model based on the 2019 network.

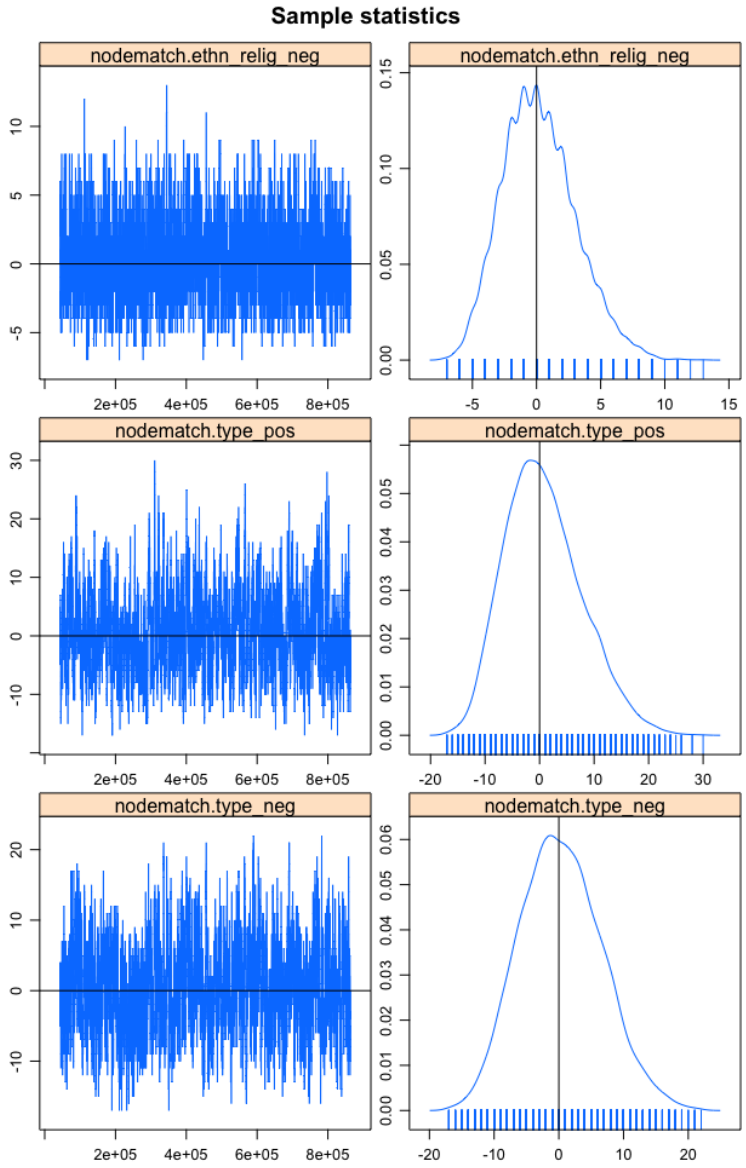


Figure 16: MCMC diagnostics for the SERGM model based on the 2019 network.

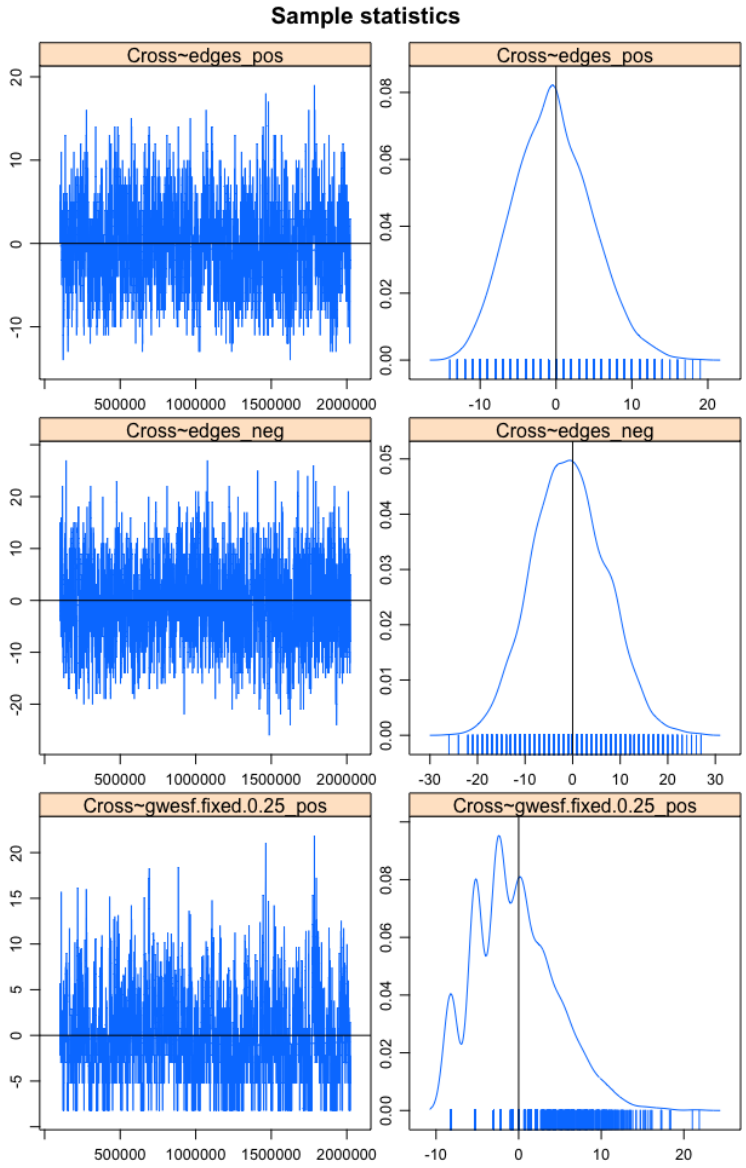


Figure 17: MCMC diagnostics for the TSERGM model based on the 2017-2019 networks.

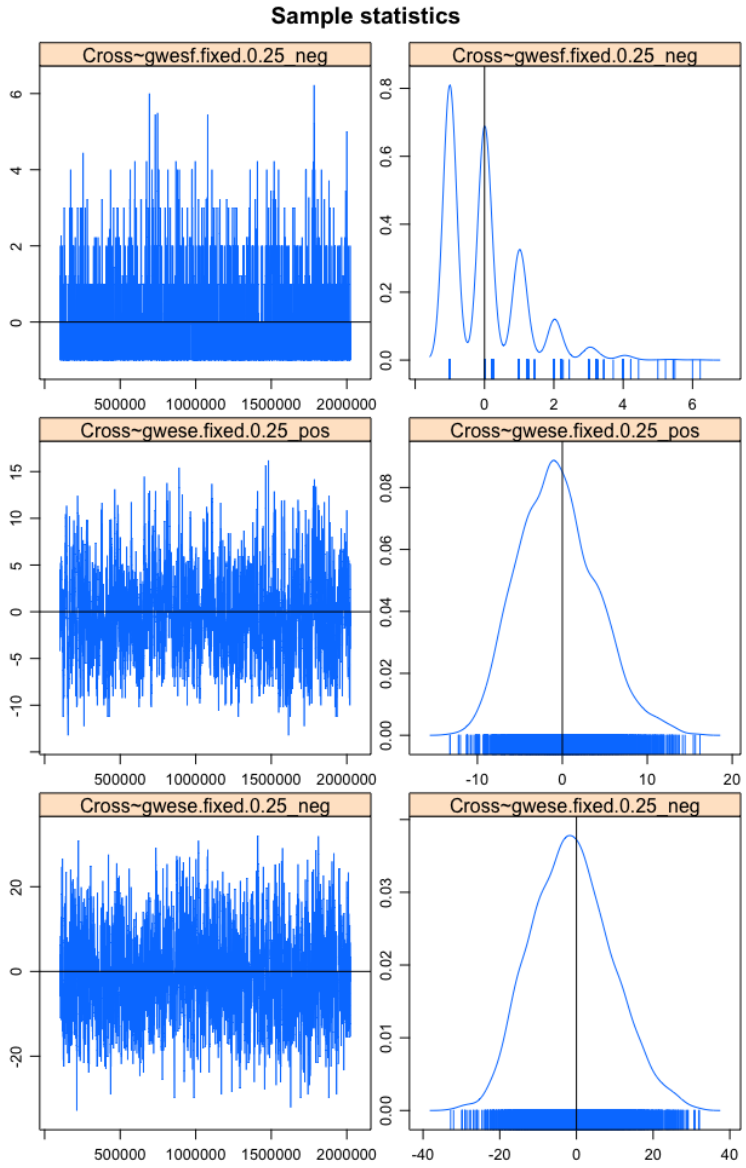


Figure 17: MCMC diagnostics for the TSERGM model based on the 2017-2019 networks.

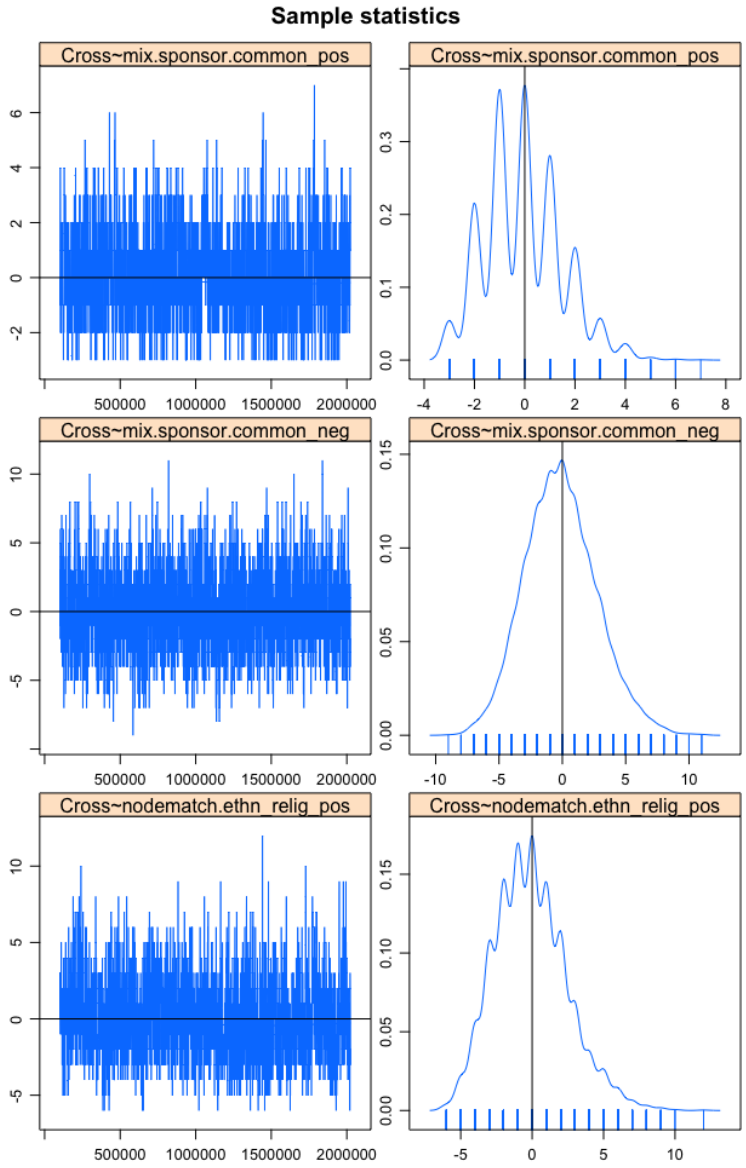


Figure 17: MCMC diagnostics for the TSERGM model based on the 2017-2019 networks.

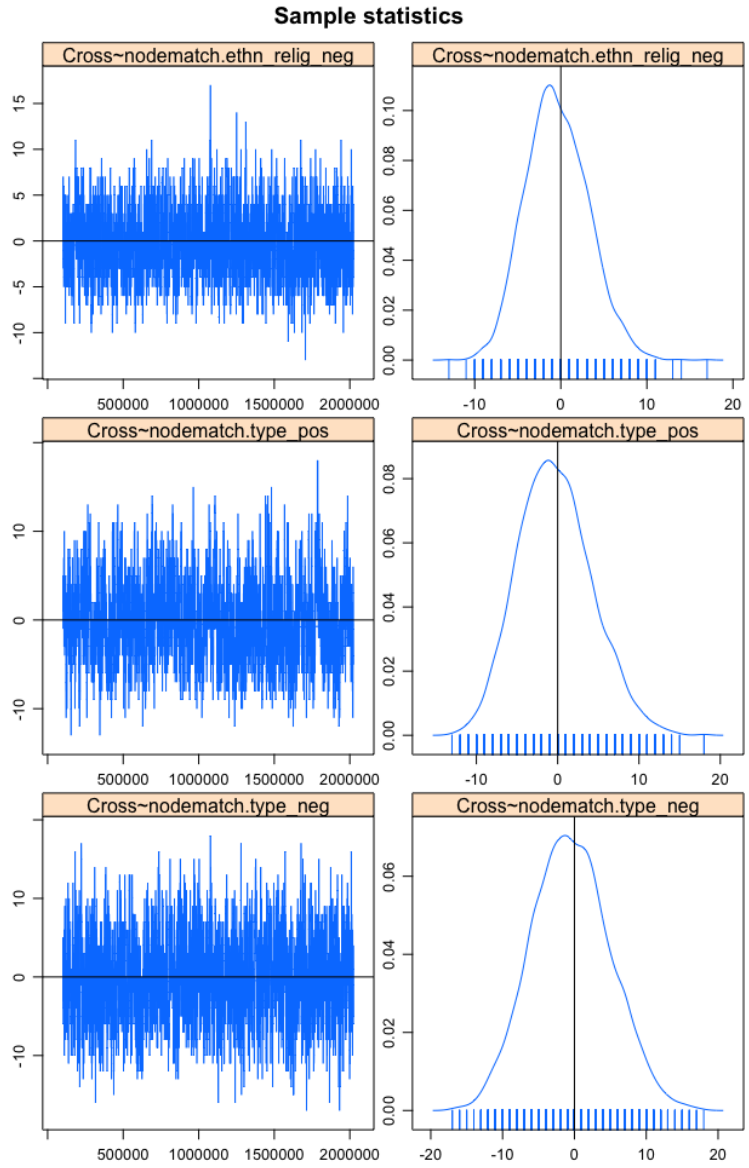


Figure 17: MCMC diagnostics for the TSERGM model based on the 2017-2019 networks.

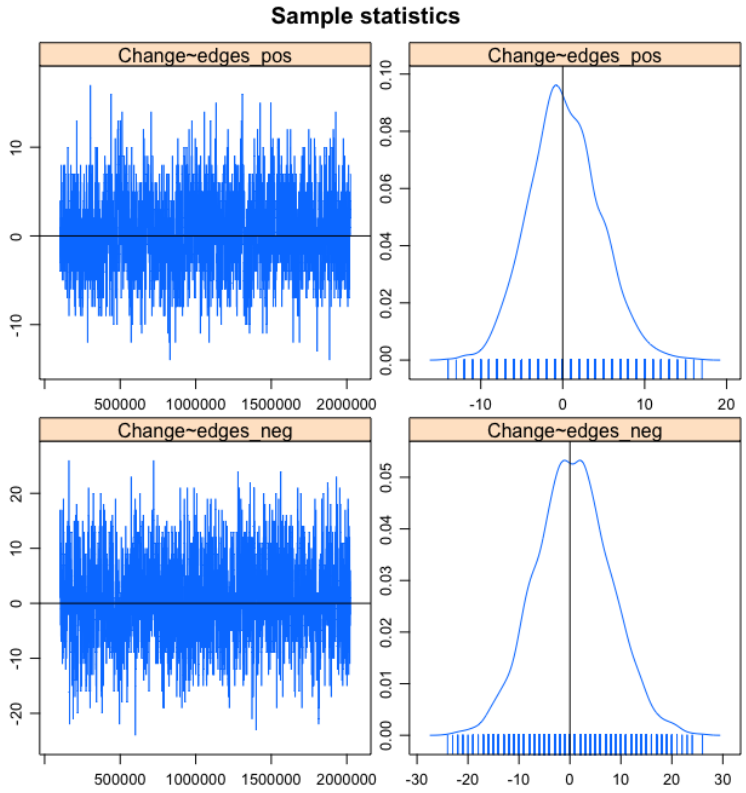


Figure 17: MCMC diagnostics for the TSERGM model based on the 2017-2019 networks.

Package 'ergm.sign'

March 27, 2023

Type Package

Title ERGM for Signed Networks

Version 0.1.0

Author Marc Schalberger

Description This package builds on the ergm.multi package from the statnet packages and makes it possible to fit an ERGM for signed networks.

License use_mit_license()

Encoding UTF-8

LazyData true

LazyLoad yes

RoxygenNote 7.2.1

R topics documented:

`count` *Calculation of Network or Graph Statistics or other Attributes Specified in a Formula*

Description

This function computes summaries of the object on the left-hand side (LHS) of the formula that are specified by its right-hand side (RHS). If a network is given as the LHS and sergm.terms is on the RHS, it computes the sufficient statistics associated with those terms.

Usage

```
count(formula)
```

Arguments

`formula`	A formula having as its LHS a `static.sign` or `dynamic.sign` object to be summarized using a formula.

Value

A vector of statistics specified in the RHS of the formula.

See Also

signnet, sergm.terms

`ergm.sign` *ergm.sign: A Package for Exponential Random Graph Models for Signed Networks*

Description

The ergm.sign package implements the tools to simulate and estimate Signed Exponential Random Graph Models and Temporal Signed Exponential Random Graph Models.

Authors

Marc Schalberger

`gof_sergm` *Conduct Goodness-of-Fit Diagnostics on a Signed Exponential Family Random Graph Model*

Description

This function calculates the goodness-of-fit (GoF) for a fitted SERGM model by simulating new networks using the same model and comparing the statistics of the original network with those of the simulated networks. The statistics used in the comparison are the positive and negative degree distributions, edge-wise shared enemies distributions for positive and negative edges, and edge-wise shared friend distributions for positive and negative edges.

Usage

```
gof_sergm(model, nsim = 200)
```

Arguments

`model`	A fitted SERGM model.
`nsim`	An integer representing the number of simulated networks to generate. Defaults to 200.

Value

Plots 6 diagnostics for the goodness-of-fit of signed exponential family random graph models.

See Also

sergm, gof_tsergm

`gof_tsergm` *Conduct Goodness-of-Fit Diagnostics on a Temporal Signed Exponential Family Random Graph Model*

Description

This function calculates the goodness-of-fit (GoF) for a fitted TSERGM model by simulating new networks using the same model and comparing the average observed statistics of the original network over the timepoints with those of the simulated networks. The statistics used in the comparison are the positive and negative degree distributions, edge-wise shared enemies distributions for positive and negative edges, and edge-wise shared friend distributions for positive and negative edges.

Usage

```
gof_tsergm(model, nsim = 200)
```

Arguments

`model`	A fitted TSERGM model.
`nsim`	An integer representing the number of simulated networks to generate. Defaults to 200.

Value

Plots 6 diagnostics for the goodness-of-fit of temporal signed exponential family random graph models.

See Also

tsergm, gof_sergm

`plot` *Plot Signed Network for Signed Networks*

Description

The function `plot.static.sign` or `plot.dynamic.sign` produces one or multiple simple two-dimensional plot of a dynamic signed network. In addition to the arguments of the `network::plot.network` function, some arguments for the context of signed networks have been added.

Usage

```
plot(
  net,
  time = c(1:length(net)),
  color_pos = "green3",
  color_neg = "red3",
  vertex.col = 2,
  vertex.legend = F,
  vertex.legend.pos = "topleft",
  main = paste("Time ", c(1:length(net))),
  vertex.legend.size = 0.65,
  ...
)

## S3 method for class 'dynamic.sign'
plot(
  net,
  time = c(1:length(net)),
  color_pos = "green3",
  color_neg = "red3",
  vertex.col = 2,
  vertex.legend = F,
  vertex.legend.pos = "topleft",
  main = paste("Time ", c(1:length(net))),
  vertex.legend.size = 0.65,
  ...
)

## S3 method for class 'static.sign'
plot(
  net,
  color_pos = "green3",
  color_neg = "red3",
```

```
  vertex.legend = F,
  vertex.col = 2,
  vertex.legend.pos = "topleft",
  legend.size = 0.65,
  ...
)
```

Arguments

`net`	An object of class `static.sign` or `dynamic.sign` created with `signnet`.
`time`	A vector of integers indicating which timepoints should be plotted.
`color_pos`	Edge color for positive edges (1). Default is set to green.
`color_neg`	Edge color for negative edges (-1). Default is set to red.
`vertex.col`	Choose a vertex attribute as a color for the vertices.
`vertex.legend`	Display vertex legend, if `vertex.col` is a vertex attribute.
`vertex.legend.pos`	Specifying legend's position using one of the following options: "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or "center".
`main`	A list of strings indicating the plot titles. If not specified, the plots will be named "Time 1" to "Time T". If no titles are desired, the input should be set to NULL.
`vertex.legend.size`	Size of the legend, which takes a character expansion factor relative to the current `par("cex")`.
`...`	Additional arguments to plot `network::plot.network`

Value

One or multiple two-dimensional plots.

See Also

signnet

`sanity` *Consistency Checks for Package*

Description

Performs consistency checks for package.

Examples

```
set.seed(1234)
mat <- matrix(sample(c(-1,0,1),100,replace=TRUE),ncol=10)
mat[lower.tri(mat)] <- 0
diag(mat) <- 0
static_net <- signnet(mat, directed = F, loops = F, matrix.type = "adjacency")

# Check that uncombine_network gets the same solution for Layer as manually
set.seed(123)
```

```
model <- sergm(static_net ~ edges_pos + edges_neg + gwesf_pos(decay=0.5, fix = T))

# uncombine_network
uncomb <- uncombine_network(simulate(model, seed = 123), split.vattr = ".LayerName")
neg <- as.sociomatrix(uncomb[[1]])*-1
pos <- as.sociomatrix(uncomb[[2]])
comb <- neg+pos

# manually
net <- as.matrix.network(simulate(model, seed = 123))
n <- ncol(net)
comb2 <- net[1:(n/2),1:(n/2)] + net[(n/2+1):n,(n/2+1):n]*-1

stopifnot(identical(comb, comb2))

# gwespL and gwesp on Layer is the same
class(static_net) <- "network"
MultiLayer <- Layer(static_net, c(`+` = "pos",`-`= "neg"))
set.seed(123)
ergm(MultiLayer ~ gwespL(0.5, fixed = T, Ls.path = c(~`+`,~`+`), L.base = ~ `+`))
summary_formula(MultiLayer ~ gwespL(0.5, fixed = T, Ls.path = c(~`+`,~`+`), L.base = ~ `+`))
set.seed(123)
ergm(MultiLayer ~ L(~ gwesp(0.5, fixed = T) , Ls = ~`+`))
summary_formula(MultiLayer ~ L(~ gwesp(0.5, fixed = T) , Ls = ~`+`))

# impact of L.in_order
set.seed(123)
ergm(MultiLayer ~ gwespL(0.5, fixed = T, Ls.path = c(~`+`,~`-`), L.base = ~ `+`, L.in_order = T))
set.seed(123)
ergm(MultiLayer ~ gwespL(0.5, fixed = T, Ls.path = c(~`+`,~`-`), L.base = ~ `+`, L.in_order = F))
```

`sergm` *Signed Exponential Random Graph Model (SERGM)*

Description

The function `sergm` is used to fit signed exponential-family random graph models (SERGMs). The function can return a maximum pseudo-likelihood estimate, an approximate maximum likelihood estimate based on a Monte Carlo scheme, or an approximate contrastive divergenceestimate based on a similar scheme.

Usage

```
sergm(formula, cons_sim = T, control = control.ergm(), ...)
```

Arguments

`formula`	An R formula object, of the form y ~ <model terms>, where y is a `static.sign` object. For the details on the possible <model terms>, see sergm.terms.
`cons_sim`	Should a constraint exist that an edge can be negative and positive, default is that this is not possible.

`control` A list of control parameters for algorithm tuning, typically constructed with control.ergm(). Its documentation gives the the list of recognized control parameters and their meaning. The more generic utility snctrl() (StatNet ConTRoL) also provides argument completion for the available control functions and limited argument name checking.

Value

An object of class ergm that is a list consisting of coef, sample etc.

See Also

signnet, sergm.terms, tsergm

`sergm.terms` *Terms Used in Signed Exponential Family Random Graph Models*

Description

How to specify network statistics in the ['ergm.sign']ergm.sign] package.

Specifying models

The LHS of the formula needs to specify an object of the class `static.sign` or `dynamic.sign`. Similarly to the definition of terms in **ergm**, the RHS contains a description of the networks terms in an additive manner. As for the naming convention of the currently implemented network terms, the suffix $_{p}os$ indicates that the network term only regard positive edges, while the suffix $_{n}eg$ does the same for negative edges. If there is none of these two suffices in the name of a term, it relates to positive and negative edges at the same time. In addition to the usual ergm terms, some terms have been added specifically for analyzing signed networks in relation to structural balance theory.

Edges

1. `edges_pos`:
   This adds a term counting all positive edges, i.e., where the adjacency matrix network equals 1, in the specified network.
2. `edges_neg`:
   This adds a term counting all negative edges, i.e., where the adjacency matrix network equals -1, in the specified network.
3. `edges`:
   This adds a term counting any type of edges, i.e., where the adjacency matrix network is unequal to 0, in the specified network.

Isolates

1. `isolates_pos`:
   This adds a term counting all actors in the network with no positive edges.
2. `isolates_neg`:
   This adds a term counting all actors in the network with no negative edges.
3. `isolates`:
   This adds a term counting all actors in the network with no edges, be they positive or negative.

Degree

1. `degree_pos(d = c(i:j))`:
   This adds a separate term counting all actors in the networks that have positive degree of i, i+1, ..., j-1, and j.
2. `degree_neg(d = c(i:j))`:
   This adds a separate term counting all actors in the networks that have negative degree of i, i+1, ..., j-1, and j.
3. `degree(d = c(i:j))`:
   This adds a separate term counting all actors in the networks that have degree of i, i+1, ..., j-1, and j.
4. `gwdegree_pos(decay = alpha,fixed = FALSE,attrname = NULL,cutoff = 30)`:
   This adds a term to the model of the geometrically weighted positive degrees. The statistic is equal the sum of actors with a specific number of positive degree, and the number of actors with degree k is weighted by exp(alpha)*(1-(1-exp(-alpha))^k).
5. `gwdegree_neg(decay = alpha,fixed = FALSE,attrname = NULL,cutoff = 30)`:
   This adds a term to the model of the geometrically weighted negative degrees. The statistic is equal the sum of actors with a specific number of negative degree, and the number of actors with degree k is weighted by exp(alpha)*(1-(1-exp(-alpha))^k).
6. `gwdegree(decay = alpha,fixed = FALSE,attrname = NULL,cutoff = 30)`:
   This adds a term to the model of the geometrically weighted degrees. The statistic is equal the sum of actors with a specific number of degree, and the number of actors with degree k is weighted by exp(alpha)*(1-(1-exp(-alpha))^k). The degree of actor n is defined as the number of positive and negative ties actor n has in the network.

Edgewise-shared partners

1. `esf_pos(d,type="OTP")`: This adds a term for the count of positive edgewise-shared friends to the model. The count is the number of friends that each have a positive tie to a common third actor, this value can lie between 0 and n -2 (where n is the number of actors in the network). Relating to the structural balance theory this term translates to clustering according to the 'friends-of-friends-are-friends' mechanism.
2. `esf_neg(d,type="OTP")`: This adds a term for the count of negative edgewise-shared friends to the model. The count is the number of enemies that each have a positive tie to a common third actor, this value can lie between 0 and n -2 (where n is the number of actors in the network). Relating to the structural balance theory this term translates to clustering according to the 'friends-of-enemies-are-friends' mechanism.
3. `ese_pos(d,type="OTP")`: This adds a term for the count of positive edgewise-shared enemies to the model. The count is the number of friends that each have a negative tie to a common third actor, this value can lie between 0 and n -2 (where n is the number of actors in the network). Relating to the structural balance theory this term translates to clustering according to the 'enemies-of-friends-are-enemies' mechanism.
4. `ese_neg(d,type="OTP")`: This adds a term for the count of negative edgewise-shared enemies to the model. The count is the number of enemies that each have a negative tie to a common third actor, this value can lie between 0 and n -2 (where n is the number of actors in the network). Relating to the structural balance theory this term translates to clustering according to the 'enemies-of-enemies-are-enemies' mechanism.
5. `esm_pos(d,type="OTP")`: This adds a term for the count of positive edgewise-shared mixed partners to the model. The count is the number of friends that have a positive and a negative tie to a common third actor, this value can lie between 0 and n -2 (where n is the number of actors

in the network). Relating to the structural balance theory this term translates to clustering according to the 'friends-of-friends-are-enemies' mechanism.

6. `esm_neg(d,type="OTP")`: This adds a term for the count of negative edgewise-shared mixed partners to the model. The count is the number of enemies that have a positive and a negative tie to a common third actor, this value can lie between 0 and n -2 (where n is the number of actors in the network). Relating to the structural balance theory this term translates to clustering according to the 'friends-of-enemies-are-enemies' mechanism.
7. `gwesf_pos(decay = alpha,fixed = FALSE,cutoff = 30,type = "OTP")`: This adds a term to the model of the geometrically weighted positive edgewise-shared friends. The weight is given by exp(alpha)*(1-(1-exp(-alpha))^k).
8. `gwesf_neg(decay = alpha,fixed = FALSE,cutoff = 30,type = "OTP")`: This adds a term to the model of the geometrically weighted negative edgewise-shared friends. The weight is given by exp(alpha)*(1-(1-exp(-alpha))^k).
9. `gwese_pos(decay = alpha,fixed = FALSE,cutoff = 30,type = "OTP")`: This adds a term to the model of the geometrically weighted positive edgewise-shared enemies. The weight is given by exp(alpha)*(1-(1-exp(-alpha))^k).
10. `gwese_neg(decay = alpha,fixed = FALSE,cutoff = 30,type = "OTP")`: This adds a term to the model of the geometrically weighted negative edgewise-shared enemies. The weight is given by exp(alpha)*(1-(1-exp(-alpha))^k).
11. `gwesm_pos(decay = alpha,fixed = FALSE,cutoff = 30,type = "OTP")`: This adds a term to the model of the geometrically weighted positive edgewise-shared mixed partners. The weight is given by exp(alpha)*(1-(1-exp(-alpha))^k).
12. `gwesm_neg(decay = alpha,fixed = FALSE,cutoff = 30,type = "OTP")`: This adds a term to the model of the geometrically weighted negative edgewise-shared mixed partners. The weight is given by exp(alpha)*(1-(1-exp(-alpha))^k).

Dyadwise-shared partners

1. `dsf(d,type="OTP")`: This adds a term for the count of dyadwise-shared friends to the model. The count is the number of pairs of actors (connected or unconnected) that each have a positive tie to a common third actor, this value can lie between 0 and n -2 (where n is the number of actors in the network).
2. `dse(d,type="OTP")`: This adds a term for the count of dyadwise-shared enemies to the model. The count is the number of pairs of actors (connected or unconnected) that each have a negative tie to a common third actor, this value can lie between 0 and n -2 (where n is the number of actors in the network).
3. `dsm(d,type="OTP")`: This adds a term for the count of dyadwise-shared mixed partners to the model. The count is the number of pairs of actors (connected or unconnected) that have a positive and a negative tie to a common third actor, this value can lie between 0 and n -2 (where n is the number of actors in the network).
4. `gwdsf(decay = alpha,fixed = FALSE,cutoff = 30,type = "OTP")`: This adds a term to the model of the geometrically weighted dyadwise-shared friends The weight is given by exp(alpha)*(1-(1-exp(-alpha))^k).
5. `gwdse(decay = alpha,fixed = FALSE,cutoff = 30,type = "OTP")`: This adds a term to the model of the geometrically weighted dyadwise-shared enemies. The weight is given by exp(alpha)*(1-(1-exp(-alpha))^k).
6. `gwdsm(decay = alpha,fixed = FALSE,cutoff = 30,type = "OTP")`: This adds a term to the model of the geometrically weighted dyadwise-shared mixed partners. The weight is given by exp(alpha)*(1-(1-exp(-alpha))^k).

Non-edgewise-shared partners

1. `nesf_pos(d,type="OTP")`: This adds a term for the count of positive non-edgewise-shared friends to the model. The count is the number of pairs of actors that are not friends but each have a positive tie to a common third actor, this value can lie between 0 and n -2 (where n is the number of actors in the network).
2. `nesf_neg(d,type="OTP")`: This adds a term for the count of negative non-edgewise-shared friends to the model. The count is the number of pairs of actors that are not enemies but each have a positive tie to a common third actor, this value can lie between 0 and n -2 (where n is the number of actors in the network).
3. `nese_pos(d,type="OTP")`: This adds a term for the count of positive non-edgewise-shared enemies to the model. The count is the number of pairs of actors that are not friends but each have a negative tie to a common third actor, this value can lie between 0 and n -2 (where n is the number of actors in the network).
4. `nese_neg(d,type="OTP")`: This adds a term for the count of negative non-edgewise-shared enemies to the model. The count is the number of pairs of actors that are not enemies but each have a negative tie to a common third actor, this value can lie between 0 and n -2 (where n is the number of actors in the network).
5. `nesm_pos(d,type="OTP")`: This adds a term for the count of positive non-edgewise-shared mixed partners to the model. The count is the number of pairs of actors that are not friends but are connected through a positive and negative tie to a common third actor, this value can lie between 0 and n -2 (where n is the number of actors in the network).
6. `nesm_neg(d,type="OTP")`: This adds a term for the count of negative non-edgewise-shared mixed partners to the model. The count is the number of pairs of actors that are not enemies but are connected through a positive and negative tie to a common third actor, this value can lie between 0 and n -2 (where n is the number of actors in the network).
7. `gwnesf_pos(decay = alpha,fixed = FALSE,cutoff = 30,type = "OTP")`: This adds a term to the model of the geometrically weighted positive non-edgewise-shared friends. The weight is given by exp(alpha)*(1-(1-exp(-alpha))^k).
8. `gwnesf_neg(decay = alpha,fixed = FALSE,cutoff = 30,type = "OTP")`: This adds a term to the model of the geometrically weighted negative non-edgewise-shared friends. The weight is given by exp(alpha)*(1-(1-exp(-alpha))^k).
9. `gwnese_pos(decay = alpha,fixed = FALSE,cutoff = 30,type = "OTP")`: This adds a term to the model of the geometrically weighted positive non-edgewise-shared enemies. The weight is given by exp(alpha)*(1-(1-exp(-alpha))^k).
10. `gwnese_neg(decay = alpha,fixed = FALSE,cutoff = 30,type = "OTP")`: This adds a term to the model of the geometrically weighted negative non-edgewise-shared enemies. The weight is given by exp(alpha)*(1-(1-exp(-alpha))^k).
11. `gwnesm_pos(decay = alpha,fixed = FALSE,cutoff = 30,type = "OTP")`: This adds a term to the model of the geometrically weighted positive non-edgewise-shared mixed partners. The weight is given by exp(alpha)*(1-(1-exp(-alpha))^k).
12. `gwnesm_neg(decay = alpha,fixed = FALSE,cutoff = 30,type = "OTP")`: This adds a term to the model of the geometrically weighted negative non-edgewise-shared mixed partners. The weight is given by exp(alpha)*(1-(1-exp(-alpha))^k).

`signnet` *Create Signed Network Object*

Description

Turn an adjacency matrix or an edgelist into a static or dynamic signed network

Usage

```
signnet(
  mat,
  directed = F,
  loops = F,
  matrix.type,
  cov = NULL,
  names = NULL,
  ...
)
```

Arguments

`mat`	(List of) Adjacency matrix or edgelist. The adjacency matrix must only consist of 1,0 or -1. The edgelist must consist of 3 columns "From", "To" and "Sign" (1 or -1). For a dynamic network the required input is a list of adjacecny matrices or edgelists.
`directed`	logical; should edges be interpreted as directed?
`loops`	logical; should loops be allowed?
`matrix.type`	Either "adjacency" or "edgelist" indicating what kind of format the input has.
`cov`	Add vertex attributes to the network. The input for this should be a dataframe where the first column contains the names of the vertices and the other columns of the dataframe should represent the individual vertex attributes that will be added to the network, however it is not necessary to include all the vertices of the network in the dataframe.
`names`	Specify vertex names. If no list is provided, the column names of the adjacency matrix will be used as names for the vertices.

Value

A signed network of the class `static.sign` or `dynamic.sign`.

`sim_sergm` *Draw from the Distribution of a Signed Exponential Family Random Graph Model*

Description

`sim_sergm` is used to draw from signed exponential family random network models. See sergm for more information on these models. The method for sergm objects inherits the model, the coefficients, the response attribute, the reference, the constraints, and most simulation parameters from the model fit, unless overridden by passing them explicitly. Unless overridden, the simulation is initialized with either a random draw from near the fitted model saved by `sergm()`.

Usage

```
sim_sergm(object, nsim = 1, seed = NULL, coef = NULL, ...)
```

Arguments

`object`	Either a formula or a fitted sergm. The formula should be of the form y ~ <model terms>, where y is a signed network of the class `static.sign`.
`nsim`	Number of networks to be randomly drawn from the given distribution on the set of all networks, returned by the Metropolis-Hastings algorithm.
`seed`	Seed value (integer) for the random number generator. See set.seed.
`coef`	Vector of parameter values for the model from which the sample is to be drawn.
`...`	Further arguments passed to or used by methods.

Value

A signed network or a list of signed networks (if nsim > 1) of class `static.sign`.

See Also

signnet, sergm, sergm.terms, sim_tsergm

`sim_tsergm` *Draw from the Distribution of a Temporal Signed Exponential Family Random Graph Model*

Description

`sim_tsergm` is used to draw from temporal signed exponential family random network models. See tsergm for more information on these models. The method for tsergm objects inherits the model, the coefficients, the response attribute, the reference, the constraints, and most simulation parameters from the model fit, unless overridden by passing them explicitly. Unless overridden, the simulation is initialized with either a random draw from near the fitted model saved by `tsergm()`.

Usage

```
sim_tsergm(object, nsim = 1, seed = NULL, coef = NULL, ...)
```

Arguments

`object`	Either a formula or a fitted tsergm. The formula should be of the form y ~ <model terms>, where y is a signed network of the class `dynamic.sign`.
`nsim`	Number of networks to be randomly drawn from the given distribution on the set of all networks, returned by the Metropolis-Hastings algorithm.
`seed`	Seed value (integer) for the random number generator. See set.seed.
`coef`	Vector of parameter values for the model from which the sample is to be drawn.
`...`	Further arguments passed to or used by methods.

Value

A dynamic signed network or a list of dynamic signed networks (if nsim > 1) of class `dynamic.sign`.

See Also

signnet, tsergm, sergm.terms, sim_tsergm

`summary` *Network Attributes for Signed Networks*

Description

Print descriptive statistics of a signed network.

Usage

```
summary(net, time = c(1:length(net)))

## S3 method for class 'dynamic.sign'
summary(net, time = c(1:length(net)))

## S3 method for class 'static.sign'
summary(net)
```

Arguments

`net`	A signed network object of class `static.sign` or `dynamic.sign`.
`time`	A list of integers indicating what timepoints should be summarised.

Value

Matrix with network attributes.

See Also

signnet

tribes *Read Highland Tribes*

Description

A network of political alliances and enmities among the 16 Gahuku-Gama sub-tribes of Eastern Central Highlands of New Guinea, documented by Read (1954).

Format

An undirected `static.sign` object with no loops.

Details

This network shows 3 clusters.

Source

http://vlado.fmf.uni-lj.si/pub/networks/data/UciNet/UciData.htm#gama, with corrections from Read (1954).

References

Taken from UCINET IV, which cites the following: Hage P. and Harary F. (1983). Structural models in anthropology. Cambridge: Cambridge University Press. (See p 56-60). Read K. (1954). Cultures of the central highlands, New Guinea. Southwestern Journal of Anthropology, 10, 1-43.

Examples

```
data(tribes)
```

tsergm *Temporal Signed Exponential Random Graph Model (SERGM)*

Description

The function `tergm` is used for finding Temporal ERGMs' (TERGMs) Conditional MLE (CMLE) (Krivitsky and Handcock, 2010) and Equilibrium Generalized Method of Moments Estimator (EGMME) (Krivitsky, 2009).

Usage

```
tsergm(
  formula,
  cons_sim = T,
  control = control.ergm(),
  times = c(0:(length(eval(formula[[2]])) - 1)),
  ...
)
```

Arguments

`formula`	An R formula object, of the form y ~ <model terms>, where y is a `dynamic.sign` object. For the details on the possible <model terms>, see sergm.terms.
`cons_sim`	Should a constraint exist that an edge can be negative and positive, default is that this is not possible.
`control`	A list of control parameters for algorithm tuning, typically constructed with control.ergm(). Its documentation gives the the list of recognized control parameters and their meaning. The more generic utility snctrl() (StatNet ConTRoL) also provides argument completion for the available control functions and limited argument name checking.
`times`	A list of integers specifying which timepoints should be taken into account.

Value

An object of class ergm that is a list consisting of coef, sample etc.

See Also

signnet, sergm.terms, sergm

Index

B Electronic appendix

The source code and datasets used in this thesis are available in the following GitHub repositories:

B.1 `ergm.sign` Package

The R package `ergm.sign` created in the scope of this thesis is available at:
`https://github.com/mschalberger/ergm.sign`

B.2 Replication Files

The replication files for the simulation example from Section 4 and the application in Section 5, as well as the associated datasets, are available at:
`https://github.com/mschalberger/ergm.sign_replication`

References

Alves, L. G., Mangioni, G., Cingolani, I., Rodrigues, F. A., Panzarasa, P. and Moreno, Y. (2019). The nested structural organization of the worldwide trade multi-layer network, *Scientific reports* **9**(1): 1–14.

Boccaletti, S., Bianconi, G., Criado, R., Del Genio, C. I., Gómez-Gardenes, J., Romance, M., Sendina-Nadal, I., Wang, Z. and Zanin, M. (2014). The structure and dynamics of multilayer networks, *Physics reports* **544**(1): 1–122.

Bonaccorsi, G., Riccaboni, M., Fagiolo, G. and Santoni, G. (2019). Country centrality in the international multiplex network, *Applied Network Science* **4**(1): 1–42.

Butts, C. T. (2008a). 4. a relational event framework for social action, *Sociological Methodology* **38**(1): 155–200.

Butts, C. T. (2008b). network: a package for managing relational data in r, *Journal of statistical software* **24**: 1–36.

Cartwright, D. and Harary, F. (1956). Structural balance: a generalization of heider's theory., *Psychological review* **63**(5): 277.

Chen, T. H. Y. (2021). Statistical inference for multilayer networks in political science, *Political Science Research and Methods* **9**(2): 380–397.

Cranmer, S. J. and Desmarais, B. A. (2011). Inferential network analysis with exponential random graph models, *Political analysis* **19**(1): 66–86.

Cranmer, S. J., Desmarais, B. A. and Morgan, J. W. (2020). *Inferential network analysis*, Cambridge University Press.

Davis, J. A. (1967). Clustering and structural balance in graphs, *Human relations* **20**(2): 181–187.

del Río-Chanona, R. M., Grujić, J. and Jeldtoft Jensen, H. (2017). Trends of the world input and output network of global trade, *PloS one* **12**(1): e0170817.

Desmarais, B. A. and Cranmer, S. J. (2010). Consistent confidence intervals for maximum pseudolikelihood estimators, *Proceedings of the Neural Information Processing Systems 2010 Workshop on Computational Social Science and the Wisdom of Crowds*, Citeseer, pp. 1865–76.

Desmarais, B. A. and Cranmer, S. J. (2012). Statistical mechanics of networks: Estimation and uncertainty, *Physica A: Statistical Mechanics and its Applications* **391**(4): 1865–1876.

Dorff, C., Gallop, M. and Minhas, S. (2020). Networks of violence: Predicting conflict in nigeria, *The Journal of Politics* **82**(2): 476–493.

Fellows, I. E. (2012). *Exponential family random network models*, University of California, Los Angeles.

Frank, O. and Strauss, D. (1986). Markov graphs, *Journal of the american Statistical association* **81**(395): 832–842.

Fritz, C., Mehrl, M., Thurner, P. W. and Kauermann, G. (2021). All that glitters is not gold: Modeling relational events with measurement errors, *arXiv preprint arXiv:2109.10348* .

Gade, E. K., Hafez, M. M. and Gabbay, M. (2019). Fratricide in rebel movements: A network analysis of syrian militant infighting, *Journal of Peace Research* **56**(3): 321–335.

Geyer, C. J. and Thompson, E. A. (1992). Constrained monte carlo maximum likelihood for dependent data, *Journal of the Royal Statistical Society: Series B (Methodological)* **54**(3): 657–683.

Hanneke, S., Fu, W. and Xing, E. P. (2010). Discrete temporal models of social networks, *Electronic journal of statistics* **4**: 585–605.

Harary, F. (1961). A structural analysis of the situation in the middle east in 1956, *Journal of Conflict Resolution* **5**(2): 167–178.

Heider, F. (1946). Attitudes and cognitive organization, *The Journal of psychology* **21**(1): 107–112.

Huitsing, G., Van Duijn, M. A., Snijders, T. A., Wang, P., Sainio, M., Salmivalli, C. and Veenstra, R. (2012). Univariate and multivariate models of positive and negative networks: Liking, disliking, and bully–victim relationships, *Social Networks* **34**(4): 645–657.

Hunter, D. R., Krivitsky, P. N. and Schweinberger, M. (2012). Computational statistical methods for social network models, *Journal of Computational and Graphical Statistics* **21**(4): 856–882.

Kivelä, M., Arenas, A., Barthelemy, M., Gleeson, J. P., Moreno, Y. and Porter, M. A. (2014). Multilayer networks, *Journal of complex networks* **2**(3): 203–271.

Krivitsky, P. N., Koehly, L. M. and Marcum, C. S. (2020). Exponential-family random graph models for multi-layer networks, *psychometrika* **85**(3): 630–659.

Morris, M., Handcock, M. S., Butts, C. T., Hunter, D. R., Goodreau, S. M., de Moll, S. B. and Krivitsky, P. N. (2015). Temporal exponential random graph models (tergms) for dynamic network modeling in statnet, Statnet Development Team. http://statnet. org/Workshops/tergm_ tutorial. html.

Morris, M. and Kretzschmar, M. (1997). Concurrent partnerships and the spread of hiv, *Aids* **11**(5): 641–648.

Mukherjee, S. (2020). Degeneracy in sparse ergms with functions of degrees as sufficient statistics.

Nakamura, K., Tita, G. and Krackhardt, D. (2020). Violence in the "balance": A structural analysis of how rivals, allies, and third-parties shape inter-gang violence, *Global Crime* **21**(1): 3–27.

Newcomb, T. M. (1961). The acquaintance process as a prototype of human interaction.

Popovic, M. (2018). Inter-rebel alliances in the shadow of foreign sponsors, *International Interactions* **44**(4): 749–776.

Raleigh, C. and Dowd, C. (2015). Armed conflict location and event data project (acled) codebook, *Find this resource* .

Robins, G., Pattison, P. and Wasserman, S. (1999). Logit models and logistic regressions for social networks: Iii. valued relations, *Psychometrika* **64**(3): 371–394.

Schweinberger, M. and Handcock, M. S. (2015). Local dependence in random graph models: characterization, properties and statistical inference, *Journal of the American Statistical Association* **77**(3): 647.

Snijders, T. A., Pattison, P. E., Robins, G. L. and Handcock, M. S. (2006). New specifications for exponential random graph models, *Sociological methodology* **36**(1): 99–153.

Strauss, D. and Ikeda, M. (1990). Pseudolikelihood estimation for social networks, *Journal of the American statistical association* **85**(409): 204–212.

Ward, M. D. and Hoff, P. D. (2007). Persistent patterns of international commerce, *Journal of Peace Research* **44**(2): 157–175.

Wasserman, S. and Pattison, P. (1996). Logit models and logistic regressions for social networks: I. an introduction to markov graphs andp, *Psychometrika* **61**(3): 401–425.

Declaration of authorship

I hereby declare that the report submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. I am aware that the Thesis in digital form can be examined for the use of unauthorized aid and in order to determine whether the report as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future Theses submitted. Further rights of reproduction and usage, however, are not granted here. This paper was not previously presented to another examination board and has not been published.

Place, Date

Signature