

Visual Diagnostics of Dynamic Social Network Models

Sam Tyner

Department of Statistics and Statistical Laboratory, Iowa State University
and

Heike Hofmann

Department of Statistics and Statistical Laboratory, Iowa State University

November 20, 2018

Abstract

Statistical models for networks have been studied for decades, and they have gotten more complex as computing power has grown. Some network models, such as those for dynamic social networks, are analytically intractable and therefore require extensive simulation procedures for model fitting. In this paper, we use model visualization techniques introduced in Wickham et al (2015) to better understand some of these models and their behavior. We use a three-pronged visualization attack: (a) network visualizations to view the *model*, (b) visualizations of collections of models to interpret parameters, and (c) visualizations of fitting algorithms to gain insights into the underlying mechanisms. With the help of static and dynamic visualizations, we bring the hidden algorithmic fitting processes into the foreground, with the goal of a better understanding and higher accessibility of statistical models for social network analysts.

Keywords: social network analysis, dynamic networks, network mapping, animation, statistical graphics

Contents

1	Introduction & Background	3
1.1	Networks and their Visualizations	4
1.2	Defining CTMC Network Models	6
1.2.1	Definitions, Terminology, and Notation	6
1.2.2	Fitting Models to Data	10
2	Example Data and Models	11
3	Model in the data space	15
4	Collections of models	19
5	Explore algorithms, not just end result	23
6	Discussion	27

1 Introduction & Background

As members of the human family, one of the hardest things we try to do is understand our relationships with one another, whether as friends, family, or colleagues. These relationships, or *social networks*, have been studied for decades, in every format from self-help books to publications in academic journals. There are also statistical models, such as exponential random graph models (ERGM) and latent space models (LSM), that strive to capture the underlying relationship-generating mechanism (Frank and Strauss, 1986; Hoff et al., 2002). These static models, however, are not optimal for modeling “real-world” social networks because they do not account for changes in the network as time passes. *Dynamic social networks*, which are observed at many points in time, are the alternative to static networks.

Models for dynamic social networks, however, are more complicated than those for static networks. These models must account for both the structure and the change in time. But if we can capture the network change mechanisms in time,

One family of models for dynamic social networks are continuous time Markov chain (CTMC) models, first introduced for networks in Holland and Leinhardt (1977). These models represent a hierarchical approach to network modeling: the changes happen at the smallest possible level, and subsequent choices are conditioned on the past. The transition probabilities between two adjacent states of a network depend on the structure of the network, the characteristics of the nodes, or both (Snijders, 1996). This means that node-level covariate effects, in addition to common ERGM effects such as reciprocity or transitivity, can be incorporated into the model. With these models, we can examine assumptions made about social networks, i.e. that people with common interests are more likely to become friends. For example, we may assume that teenagers who drink heavily are more likely to be friends with other teenagers who drink heavily. The CTMC family of models allows us to incorporate this behavioral information on shared interests into the modeling process through a particular subfamily of CTMC models known as stochastic actor-oriented models (SAOMs) (Snijders, 1996).

With the added flexibility of CTMC models comes increased estimation difficulty (Snijders, 2017). Likelihood functions quickly become very complex due to the inherent de-

pendency structure in the data, and therefore computational methods, such as Markov chain Monte Carlo (MCMC), are used to fit models. For CTMC parameter estimation, we use the software SIENA, and its R implementation **RSiena** by Ripley et al. (2016a). This software is a considerable contribution to the field of social network analysis, but the extensive parameter estimation process is largely hidden from the user. To better understand CTMCs and the fitting process in **RSiena**, we use model visualization, or model-vis (Wickham et al., 2015), to examine the underlying methods, hidden structures, and simulation results. Ultimately, we aim to help researchers better understand the structure, behavior, fit, and results of CTMC models for social network data.

In the remainder of this section, we briefly introduce the structure of network data and two different network visualization (net-vis) techniques. Then, we carefully define the hierarchical model structure of a CTMC. Finally, we briefly outline the CTMC fitting process in **RSiena** that we use throughout this paper (Snijders, 2016).

1.1 Networks and their Visualizations

Network data across fields have similar structure. There are always units of observation and connections between those units. In a social network, the units of observations, called *nodes* or *actors*, are people, while the connections, called *edges* or *ties*, are their relationships with each other. Other network data include inherent variables of interest on the nodes and edges themselves, e.g. someone’s age, gender, and political affiliation, and the type of relationship (friendship, coworkers, etc) that an edge represents.

Visualizing Network Data: Network visualization, also called graph drawing, is a prominent subfield of network analysis. For a review of graph drawing, see Tamassia (2013). Visualizing network data is inherently challenging due to the structure of the data itself. Most data visualizations rely on well-defined axes inherited from the data, such as Cartesian coordinates or spatial locations, but network data typically do not have such built-in structure.

The lack of a native coordinate system has given rise two very different primary net-vis methods: the node-link diagram and the adjacency matrix visualization (Knuth, 2013; Fekete, 2009). We demonstrate these with a simple example. Consider the network with five

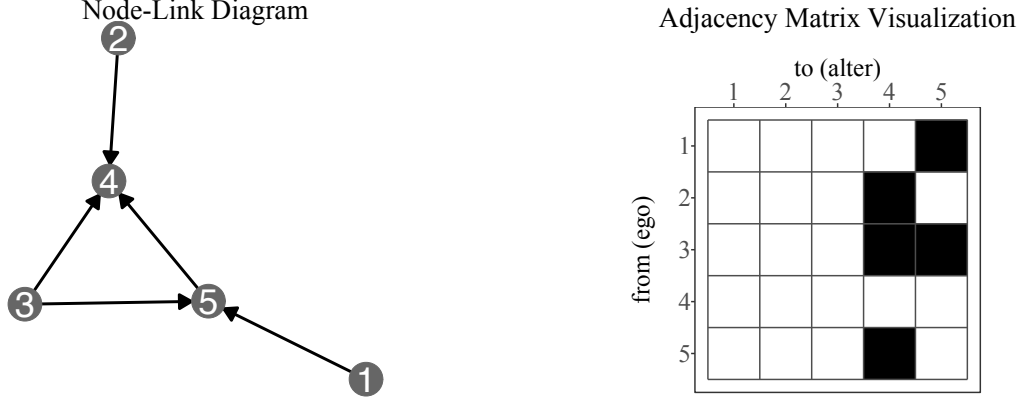


Figure 1: On the left, a node-link diagram of our toy network, with nodes placed using the Kamada-Kawai algorithm. On the right, the corresponding adjacency matrix visualization.

nodes, $\{1, 2, 3, 4, 5\}$, connected by five directed edges: $\{2 \rightarrow 4, 3 \rightarrow 4, 1 \rightarrow 5, 3 \rightarrow 5, 5 \rightarrow 4\}$ shown in Figure 1.

The first net-vis method, the node-link diagram, represents nodes with points in two dimensions and then represents edges by connecting the points with lines. These lines can also have arrows on them indicating the direction of the edge for directed networks, as shown in Figure 1. Because there is often no natural placement of the nodes, they are placed in 2D at random, then adjusted with one of the many layout algorithms available (Gibson et al., 2013). Some commonly used algorithms, such as the Kamada-Kawai (KK) layout (Kamada and Kawai, 1989) and the Fruchterman-Reingold (FR) layout (Fruchterman and Reingold, 1991), are designed to mimic physical systems, drawing the graphs based on the “forces” connecting them. In Figure 1, we use the KK algorithm, and unless otherwise stated, all other node-link diagrams in this paper were drawn with the KK layout algorithm.

The second method for net-vis uses the adjacency matrix of the network. The adjacency matrix of a network, \mathbf{A} is defined as:

$$A_{ij} = \begin{cases} 1 & \text{if an edge exists } i \rightarrow j, \quad i \neq j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The adjacency matrix visualization for our toy example is also shown in Figure 1. This visualization shows the network as a grid, with each row representing the “from” node and each column representing the “to” node of an edge. The grid space for edge $i \rightarrow j$ is filled in if $A_{ij} = 1$ and is empty if $A_{ij} = 0$. For undirected networks, \mathbf{A} and the grid visualization

are symmetric, with each edge represented twice.

Each net-vis method has its own advantages and disadvantages. For instance, *paths* between two nodes in a network are easier to determine with node-link diagrams than with adjacency matrix visualizations (Ghoniem et al., 2005). In node-link diagrams, node information can be incorporated into the visualization by coloring or changing the shape of the points, and edge-level information can be incorporated by coloring the lines, or changing their thickness. Visualizing a node variable in an adjacency matrix visualization is not as simple, because this type of net-vis features the edges. An adjacency matrix visualization can, however, be useful when the network is very complex, dense, or large (Ghoniem et al., 2005). Ultimately, there is no one “correct” way to visualize network information, and we will be using both the node-link and adjacency matrix net-vis methods throughout this paper to explore our example data and CTMC models.

1.2 Defining CTMC Network Models

The CTMC network model family is for analyzing *dynamic* networks, so we can use them to study the change of a network in time while also examining the structural and node covariate effects. To define CTMCs for our purposes, we follow the general hierarchical definition outlined in Snijders (2017) and elsewhere (see e.g. Snijders (2001) and Snijders et al. (2010)). The hierarchy reflects our intuition about social networks as they exist naturally. Most social networks, even holding constant the set of actors over time, are ever-changing as relationships form or disappear. Actors in social networks also have inherent properties that could affect how they change their role within the network, and existing ties may affect formation or dissolution of other ties. In this section, we define the data structure, the hierarchical model, and intensity matrix of the CTMC.

1.2.1 Definitions, Terminology, and Notation

A *dynamic network* consists of a fixed set of n nodes, that is changing over time, and is observed at M discrete time points, t_1, \dots, t_M with $t_1 < t_2 < \dots < t_M$. We denote the network observation at timepoint t_m by $x(t_m)$ for $m = 1, \dots, M$. The model assumes that this longitudinal network of discrete observations is embedded within a CTMC, which

we denote by $X(T)$. The CTMC is almost entirely unobserved: we assume the first and last network observations are the starting and ending points of the CTMC, i.e. $x(t_1) \equiv X(0)$ and $x(t_M) \equiv X(\infty)$. Nearly all other steps in the chain are unobserved, with the exception of $x(t_2), \dots, x(t_{M-1})$. Unlike the first and last observations of the network, these observations do not have direct correspondence with steps in the CTMC. Thus, the observations $x(t_2), \dots, x(t_{M-1})$ are considered to be “snapshots” of the network at some point between two steps in the CTMC.

The CTMC process $X(T)$ is a series of single tie changes that happen according to the model’s *rate function*, the first piece of the hierarchy. One actor at a time is “selected” to make a change, then in the second piece of the hierarchy, it randomly changes its ties according to probabilities derived from the *objective function*.

The Rate Function: In general, the *rate function*, $\rho(x, \mathbf{z}, \boldsymbol{\alpha})$, is a function of the network at its current state x , covariates of interest, \mathbf{z} , and some parameters, $\boldsymbol{\alpha}$. With this general formulation, each node can have a different rate of change. Throughout this paper, we assume a simple rate function, $\rho(x, \mathbf{z}, \boldsymbol{\alpha}) \equiv \alpha_m$ that is constant across all actors between time t_m and t_{m+1} . With this simple rate function, we have $M - 1$ rate parameters, one for each transition from $x(t_m)$ to $x(t_{m+1})$ for $m = 1, \dots, M - 1$. The rate parameter, α_m dictates how often an actor i “gets an opportunity” to change one of its ties, x_{ij} , for $j \neq i \in \{1, \dots, n\}$ in the time period from t_m to t_{m+1} for $m = 1, \dots, M - 1$. The CTMC model assumes that the actors i are conditionally independent given their ties, x_{i1}, \dots, x_{in} at the current network state. Let $\tau(i|x, m)$ be the wait time until actor i gets to the opportunity to change its ties. For any time point T , where $t_m \leq T < t_{m+1}$, the waiting time to actor i ’s next change follows an exponential distribution:

$$\tau(i|x, m)|x_{i1}(m), \dots, x_{in}(m) \stackrel{\text{iid}}{\sim} \text{Exp}(\alpha_m) \quad (2)$$

with expected value α_m^{-1} . From Equation 2, we can derive the waiting time to the next change opportunity by *any* actor in the network, $\tau(x)$. The value $\tau(x)$ is also exponentially distributed, and has expected value $(n\alpha_m)^{-1}$. The estimation of α_m in **RSiena** is simple: method of moments (MoM) is used with the statistic

$$C_m = \sum_i \sum_j |x_{ij}(t_{m+1}) - x_{ij}(t_m)| \quad (3)$$

which is the total number of changes from $x(t_m)$ to $x(t_{m+1})$.

The Objective Function Because of the conditional independence assumptions given in Equation 2, the objective function can be written separately for each node. Suppose node i is selected to change at the next step in the CTMC. This node, called the *ego* node, has the potential to interact with all other nodes in the network, $j \neq i$. These nodes j , are referred to as *alter* nodes, and will not change any of their ties at the next step in the CTMC. The ego node i in the current network state x , prioritizes making changes that maximize its *objective function*:

$$f_i(\boldsymbol{\beta}, x, \mathbf{Z}) = \sum_{k=1}^K \beta_k s_{ik}(x, \mathbf{Z}), \quad (4)$$

where x is the current network state, \mathbf{Z} are node covariates, and K is the total number of parameters in the objective function. The parameters of the model, β_1, \dots, β_K correspond to network statistics, $s_{ik}(x, \mathbf{Z})$, for $k = 1, \dots, K$. To maximize f_i , i will either destroy a tie where $x_{ij} = 1$, create a tie where $x_{ij} = 0$, or make no change.

The parameters of $f_i(\boldsymbol{\beta}, x, \mathbf{Z})$ are either structural or covariate parameters. The structural parameters correspond to statistics that are *only* functions of the current network state, x , while the covariate parameters are also functions of the covariates, \mathbf{Z} . According to Snijders (2001, p. 371), there should be at least two parameters included in the objective function: β_1 for the outdegree of nodes, and β_2 for the reciprocity of nodes. The former measures the propensity of nodes with a lot of outgoing ties to form more outgoing ties (the “rich get richer” effect), while the latter measures the tendency of outgoing ties to be returned within a network. The statistics corresponding to these two parameters, $s_{i1}(x)$ and $s_{i2}(x)$ are given in Table 1. In the **RSiena** software, there are over 80 possible $\boldsymbol{\beta}$ parameters with corresponding statistics $s_{ik}(x, \mathbf{Z})$ to add to the model. The definition of the statistics for all possible parameters are provided in Ripley et al. (2017), and two additional parameters and their corresponding statistics are discussed in Section 2 and shown in Figure 4.

In the CTMC, when actor i can make a change, it chooses which tie x_{i1}, \dots, x_{in} to change at random according to the probabilities $p_{ij}(\boldsymbol{\beta}, x, \mathbf{Z})$. These probabilities are defined via the objective function. Let $x(i \rightsquigarrow j)$ be the network identical to network x with the exception of tie x_{ij} , which is equal to $1 - x_{ij}$; i.e. the presence or absence of node x_{ij} is flipped in the

Structural Effects

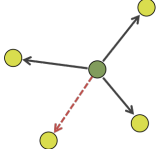

Name	Statistic	Figure
outdegree	$s_{i1}(x) = \sum_j x_{ij}$	
reciprocity	$s_{i2}(x) = \sum_j x_{ij}x_{ji}$	

Table 1: Examples of structural effects included in the objective function. The darker nodes in the Figure column are the ego nodes (i), and all others are alters (j). The dotted lines are ties the ego node is en(dis)couraged to make when the parameter corresponding to each effect is positive (negative).

new network. The probability that the tie x_{ij} changes is defined as:

$$p_{ij}(\boldsymbol{\beta}, x, \mathbf{Z}) = \frac{\exp \{f_i(\boldsymbol{\beta}, x(i \rightsquigarrow j), \mathbf{Z})\}}{\sum_{h \neq i} \exp \{f_i(\boldsymbol{\beta}, x(i \rightsquigarrow h), \mathbf{Z})\}} \quad (5)$$

We explore these probabilities further in Section 5.

Intensity Matrix: The probabilities p_{ij} combined with the rate parameters α_m fully characterize the underlying CTMC that models network change. In the CTMC literature, see for instance Yin and Zhang (2010), chains are characterized by their *intensity* matrix \mathbf{Q} . This matrix describes the rate of change between two states of the CTMC process, and the rows of this matrix always add to zero. The state space for a network, denoted \mathcal{X} , is set which contains $2^{n(n-1)}$ states: two possible states for an edge, $\{0, 1\}$, and $n(n-1)$ edge relationships because the network is directed and we exclude self-ties. The intensity matrix is then a square matrix of dimension $2^{n(n-1)}$. Only one tie changes at a time in the CTMC, resulting in $n(n-1)$ reachable states from the current network state. Thus, the intensity matrix \mathbf{Q} is very sparse, with only $n(n-1) + 1$ non-zero entries in each row. Note that $n(n-1)$ of these entries represent the possible states that are one edge different from a given state, while the additional non-zero entry is for the state to remain unchanged.

The two pieces of the model hierarchy each contribute to the entries of the intensity matrix to describe the rate of change between two network states. The entries of \mathbf{Q} are defined as follows: let $b \neq c \in \{1, 2, \dots, 2^{n(n-1)}\}$ be indices of two different possible states

of the network, $x^b, x^c \in \mathcal{X}$. Then the bc^{th} entry of Q is:

$$q_{bc} = \begin{cases} q_{ij} = \alpha_m p_{ij}(\boldsymbol{\beta}, x^b) & \text{if } \sum_{i,j} |x_{ij}^c - x_{ij}^b| = 1 \\ -\sum_{i \neq j} q_{ij} & \text{if } x^b = x^c \\ 0 & \text{otherwise} \end{cases}$$

The rate of change between any two states, x^b and x^c , that differ by only one tie x_{ij} , is the product of the rate at which actor i gets to change a tie and the probability that the tie that will change is the tie to node j . The intensity matrix \mathbf{Q} fully defines the model for network change, and is the foundation required for parameter estimation.

1.2.2 Fitting Models to Data

To fit a CTMC to dynamic network data, we apply the **RSiena** software, which uses simulation methods to estimate parameter values using either MoM or maximum likelihood (ML) estimation (Ripley et al., 2016a). We chose to use MoM estimation, so we describe that process here, while more information on ML estimation can be found in Snijders (2016). The underlying SIENA software uses a Robbins-Monro algorithm (see Robbins and Monro (1951)) to estimate the solution of the moment equation

$$E_{\theta} S = s_{obs} \tag{6}$$

where $\theta = (\boldsymbol{\alpha}, \boldsymbol{\beta})$ is the vector of rate and objective function parameters, and s_{obs} is the observed vector of the corresponding statistics summed over all network observations. The full SIENA MoM estimation algorithm operates in three phases, as described in Ripley et al. (2017) and Snijders (2016), and we briefly summarize these phases here. The first phase obtains initial estimates of the score functions for use in the second phase. The second phase is the Robbins-Monro algorithm, and results in parameter estimates through iterative updates and simulation from the CTMC at the iterations of the parameter values. The third phase uses the parameters from phase two to estimate the score functions and covariance matrix of the parameter estimates and carries out convergence checks. In Section 5, we further explore these phases in the SIENA MoM algorithm through visualization, bringing them out of their “black box” and into the light.

The rest of the paper is structured as follows. In Section 2, we introduce the example data and models we use for model-vis. Then, in Section 3, we explore the first step in model-vis by placing the models in the data space. Section 4 follows, in which we explore collections of models. Finally, in Section 5, we visualize the underlying algorithms in the CTMC fitting process. We conclude with a discussion in Section 6.

2 Example Data and Models

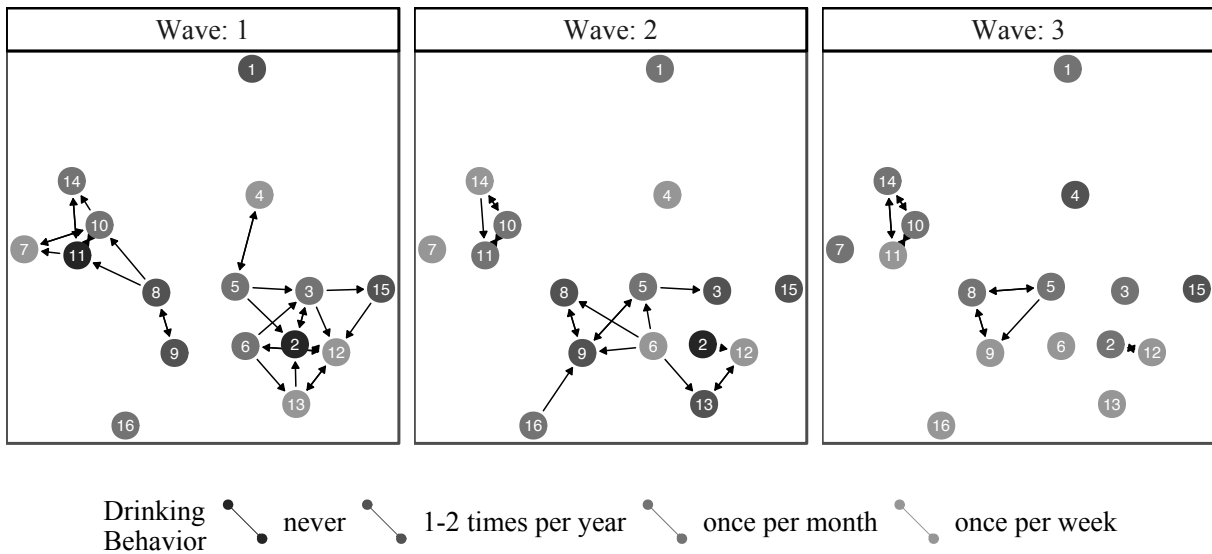


Figure 2: The small friendship network data we model throughout this paper. The teenagers’ drinking behavior changes and the network becomes less densely connected over time. These are the mechanisms we hope to model with our chosen CTMCs.

Example Data: To guide our visual exploration of CTMC models for dynamic networks, we use two example data sources. The first is a subset of the 50 actor dataset from the “Teenage Friends and Lifestyle Study” that is provided on the **RSiena** webpage. These data come from Michell and Amos (1997), and we chose to only work with a subset of the data to make node-link diagrams easier to see and to make any changes in the network more noticeable. We visualize this data with a node-link diagram in Figure 2. The actor covariate of interest, drinking behavior, has four values: (1) does not drink, (2) drinks once or twice a year, (3) drinks once a month, and (4) drinks once a week. In Figure 2, the

nodes are colored according to the drinking behavior of that student. We can see that, as time passes, the students seem to drink more and become increasingly isolated into smaller groups. An analysis of this type of data with a CTMC model should capture these dynamics in a way that allows the researcher to draw conclusions about the nature of the network and behavioral forces at play.

The second data example we use is a much more recent dataset on collaboration in the United States Senate during the 111th through 114th Congresses, shown in Figure 3. These sessions of congress correspond to the years of Barack Obama’s presidency, from 2009-2016. (Details of how this data can be downloaded are provided by Francois Briatte at <https://github.com/briatte/congress>.) In this network, ties are directed from senator i to senator j when senator i cosponsors the bill that senator j authored. There are hundreds of ties between senators when they are connected in this way, so we simplify the network by computing a single value for each pair of senators called the *weighted propensity to cosponsor* (WPC). This value is defined in Gross et al. (2008) as the total number of times senator i cosponsors senator j ’s bill, weighted by the number of total cosponsors:

$$WPC_{ij} = \frac{\sum_{k=1}^{n_j} \frac{Y_{ij(k)}}{c_{j(k)}}}{\sum_{k=1}^{n_j} \frac{1}{c_{j(k)}}} \quad (7)$$

where n_j is the number of bills in a congressional session authored by senator j , $c_{j(k)}$ is the number of cosponsors on senator j ’s k^{th} bill, where $k \in \{1, \dots, n_j\}$, and $Y_{ij(k)}$ is a binary variable that is 1 if senator i cosponsored senator j ’s k^{th} bill, and is 0 otherwise. This measure ranges in value from 0 to 1, where $WPC_{ij} = 1$ if senator i is a cosponsor on every one of senator j ’s bills and $WPC_{ij} = 0$ if senator i is never a cosponsor any of senator j ’s bills.

Because CTMC models require binary edges, we focus only on strong collaborations, where WPC is high. For a senate collaboration network x , edges are defined as

$$x_{ij} = \begin{cases} 1 & \text{if } WPC_{ij} > 0.25 \\ 0 & \text{if } WPC_{ij} \leq 0.25. \end{cases} \quad (8)$$

The node-link diagram for the four senates during the Obama administration are shown in Figure 3. We can see that Senate 111 is much more densely connected than the other three,

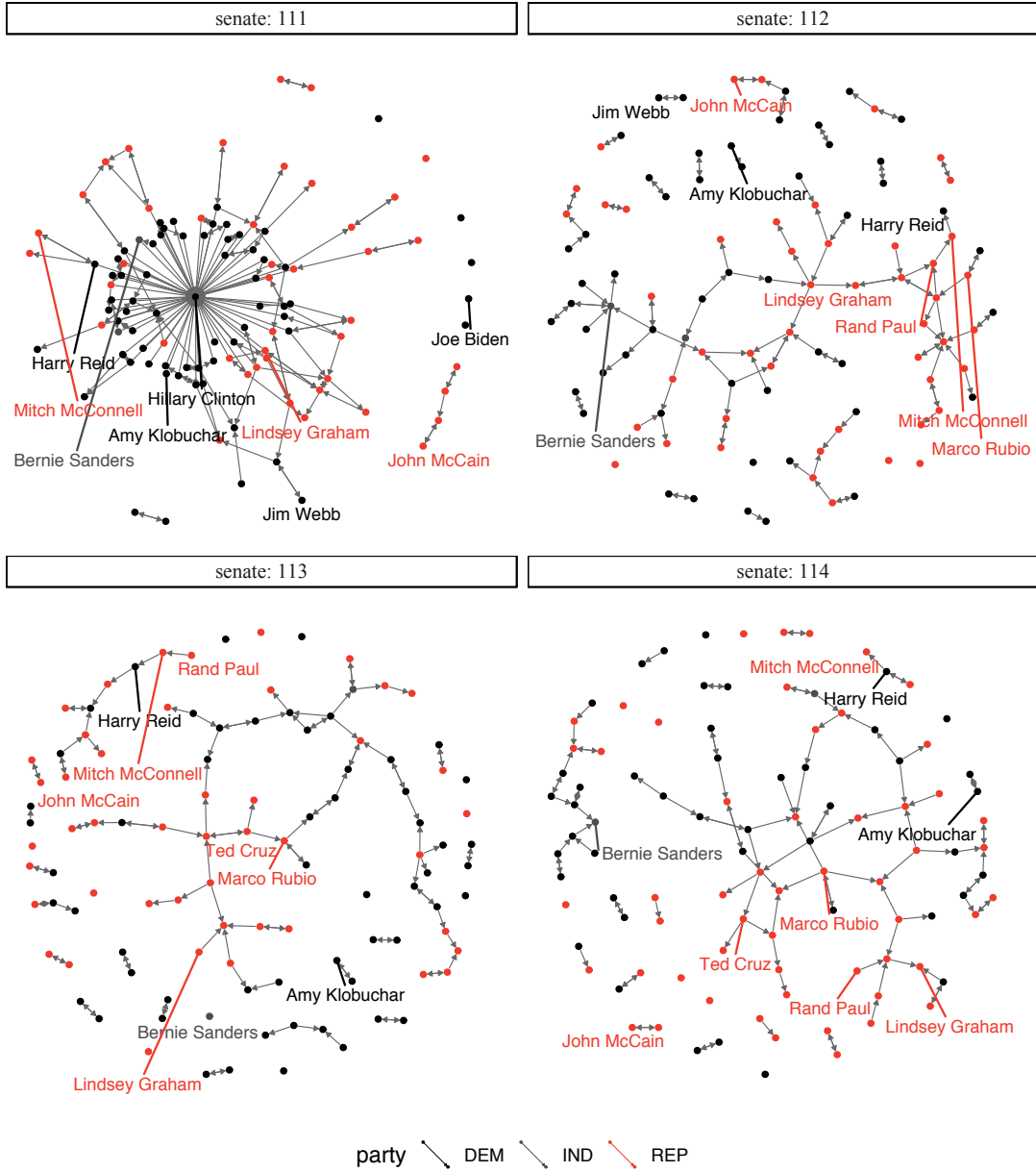


Figure 3: The collaboration network in the four senates during the Obama years, 2009-2016. Edges are shown only if the WPC between two senators is greater than 0.25. We use the FR layout algorithm here.

which have one large, sparsely connected component with a few smaller connected groups throughout. Some prominent names are shown to demonstrate their place in the network. As with the friendship example, analyzing the senate data with a CTMC model should capture the underlying dynamics, giving insight into the impact that the actor covairates



(a) Realization of a JTT, where the covariate is represented by the shape of the nodes. The dotted tie is encouraged if β_3 is positive.

$$s_{i3}(x, z) = \sum_{j \neq h} x_{ij} x_{ih} x_{hj} \mathbb{I}(z_i = z_h \neq z_j)$$

(b) Realization of a N22 between actors i and j . The dotted tie is encouraged if β_4 has a positive value. $s_{i4}(x) = |\{j : x_{ij} =$

$$0, \sum_h x_{ih} x_{hj} \geq 2\}|$$

Figure 4: The additional network effects included in the models we fit to the friends data. On the left, a JTT between i and j , who have different covariate values, and on the right, a N22 between i and j , who are indirectly connected by nodes k, h .

have on the senate's collaboration structure. By using data sets from very different contexts, we hope to assess some differences in model performance.

Models: To our example data, we fit three different models. Each model uses a simple rate function, α_m , and an objective function with two or three parameters. The first model, M1, contains the absolute minimum number of parameters in the objective function $f_i(x)$:

$$M1 : \quad f_i(x) = \beta_1 s_{i1}(x) + \beta_2 s_{i2}(x), \quad (9)$$

where s_{i1} is the density network statistic and s_{i2} is the reciprocity network statistic for actor i at the current network state x (see Table 1). The second and third models, M2 and M3, contain one additional parameter each in the objective function. We chose the additional parameters because they were significant (p -value < 0.05) according to a Wald-type test provided in the **RSiena** software (Ripley et al., 2016b). The model M2 contains an actor-level covariate parameter, and the model M3 contains an additional structural effect in the objective function.

$$M2 : \quad f_i(x, z) = \beta_1 s_{i1}(x) + \beta_2 s_{i2}(x) + \beta_3 s_{i3}(x, z) \quad (10)$$

$$M3 : \quad f_i(x) = \beta_1 s_{i1}(x) + \beta_2 s_{i2}(x) + \beta_4 s_{i4}(x), \quad (11)$$

where $s_{i3}(x, z) = \sum_{j \neq h} x_{ij} x_{ih} x_{hj} \mathbb{I}(z_i = z_h \neq z_j)$, and $s_{i4}(x) = |\{j : x_{ij} = 0, \sum_h x_{ih} x_{hj} \geq$

2}]. These statistics are known as the *number of jumping transitive triplets* (JTT) and the *number of doubly achieved distances two effect* (N22), respectively. The JTT effect emphasizes triads formed between actors with *different* covariate values, while the N22 effect emphasizes *indirect* ties between actors. The covariate for the friendship data is the drinking behavior, and is the number of bills authored for the senate data. These additional effects are visualized in Figure 4a and Figure 4b, where the ties affected by the parameter value are represented by dotted lines. If the estimates of these parameters $\hat{\beta}_1, \dots, \hat{\beta}_4$ are positive, the model encourages the tie represented by the dotted line to form, while the opposite is true if the parameter estimate is negative.

3 Model in the data space

Model visualization complements traditional model diagnostic tools such as R^2 values and residual plots to enrich our understanding and interpretation of statistical models. Each of the three different meanings of the word “model”, the model *family*, the model *form*, and the *fitted* model (see Wickham et al., 2015, pg. 2), can benefit from model visualization (model-vis). We apply the three principles of model-vis: view the model in the data space, visualize collections of models, and explore the process of fitting the model, to the models and data introduced in Section 2. Since we have already decided on our model family (CTMCs), we shift our focus to the fitted model and the model form. Specifically, we use model-vis to learn more about how the *model form* affects the *fitted model*.

First, we put models in the data space to assess the model fit. We define the *data space* for our models as the combined data structures of the network: the node, edge, and time information. We use the R package `geomnet` (Tyner and Hofmann, 2016) to visualize the three data spaces together. The package lays out the network using node-link diagrams, and the network’s node or edge data are then mapped to different visual features. The color, size, and shape of the points can be used to represent variables in the node data, while the color, linewidth, and linetype of the edges between nodes can be used to represent variables in the edge data. To view temporal changes, we place the network at different timepoints side-by-side to see the evolution. By pulling all of this information together using `geomnet`, we can show the entire data space at once.

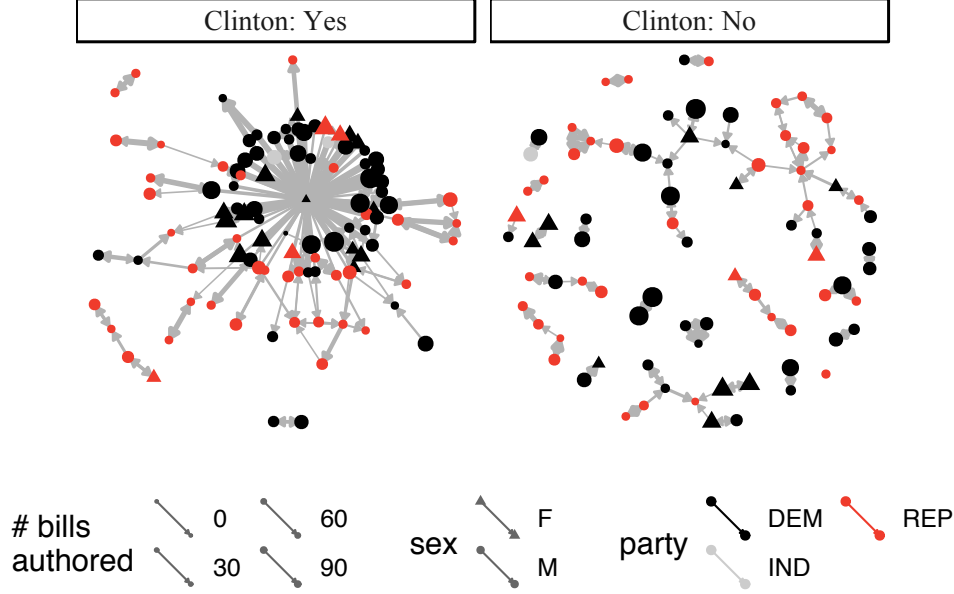


Figure 5: The 111th Senate at two times: while Clinton was in the senate in 2009 (left) and after she departed (right). We map sex, party, and bills authored to the shape, color, and size of the nodes, respectively. We also map WPC to the edge width.

To demonstrate the functionality of `geomnet`, we visualize the connections in the 111th U.S. Senate at two different timepoints: when Hillary Clinton was in the senate, and after she left to become Secretary of State. Clinton was only in the 111th senate from January 3-20, 2009, in the middle of her second term as the junior senator from New York. In that time, she authored two bills and was a cosponsor on 17. There are many senators who cosponsored both of her bills, giving their edges to her a WPC of 1. So with Clinton included in the node-link diagram in Figure 5, the senate looks much more collaborative than it does without her. We then map our potential model covariates to visual features: number of bills authored to size of the node, gender to shape, and party to color. In addition, we visualize the strength of the tie by mapping the WPC value between the two senators to the linewidth. By viewing node information, edge information, and time simultaneously, we have learned that Senator Clinton greatly affected the structure of collaboration within the senate.

To view our models in the data space, we first simulate from the fitted model. No single network instance simulated from a CTMC will fully represent the model, so instead we look at an *expectation* from the network model. We frequently rely on expected values

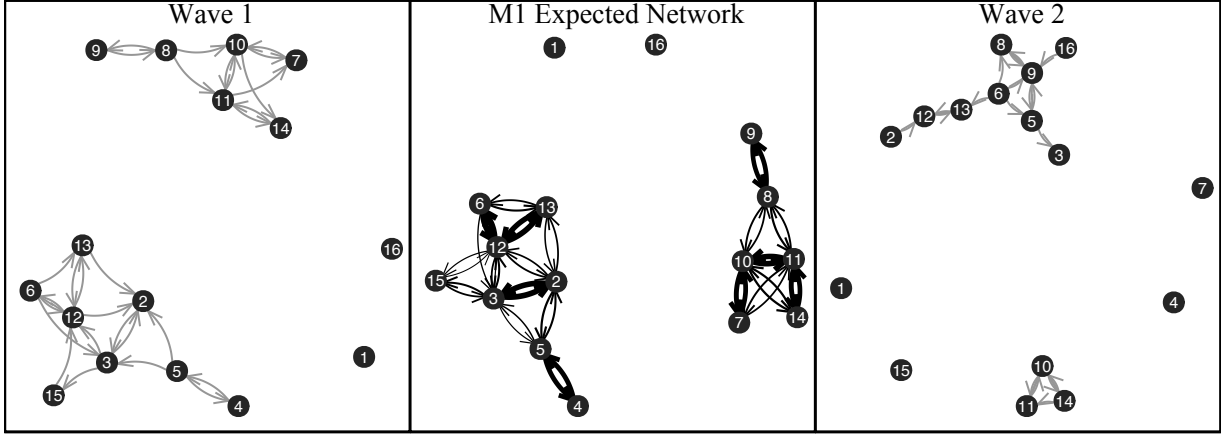


Figure 6: On the left, the first wave of observed data, conditioned on in the model. On the right, the second wave of observed data. In the middle, a summary network from the first model fit to the data. This summary network represents 1,000 simulations of wave 2 using the values from the simple fitted model M1.

in statistics, but network models, especially those as complex as our CTMC models, lack an expected network measure. We could talk about expected values of parameters, but the parameters can be hard to interpret and cannot teach us much about the overall structure of the network. How then, can we arrive at an “expected” network? We answer this question with visualization. To construct an expected network, we

1. simulate 1,000 wave 2 and wave 3 instances of our small friendship network from model M1 using estimated parameter values (see Table 2) that have previously been estimated. Then, we
2. count the number of times each edge $i \rightarrow j$ appeared in one of the 1,000 network simulations. Then, we
3. construct one weighted summary network for time points 2 and 3. Each summary network has edge weights equal to the proportion of the simulated networks in which the edge appeared. Finally, we
4. create the *expected network* by including only the edges that appeared in more than 5% of the simulations.

The resulting expected network is in the middle panel of Figure 6. Edges with higher weights are darker, thicker lines. On either side of the average network in Figure 6, we show the data for comparison: wave 1 on the left, and wave 2 on the right. We can see that the overall structure of the average network is much more similar to wave 1 than to wave 2. In both wave 1 and the expected network, the edges form two distinct clusters of nodes: $\{2, 3, 4, 5, 6, 12, 13, 15\}$ and $\{7, 8, 9, 10, 11, 14\}$. The expected network, however, is supposed to represent the second wave of data, on the right in Figure 6. The expected network does not resemble wave 2, which has one small, strongly connected cluster and one larger, sparsely connected cluster, with four unconnected nodes. Because the expected network does not resemble wave 2, we take this as evidence that the simple model, M1, does not capture the change in network structure from the first to the second wave of data. Visualizing the expected network in this way places the model M1 in the data space, and helps us assess model fit.

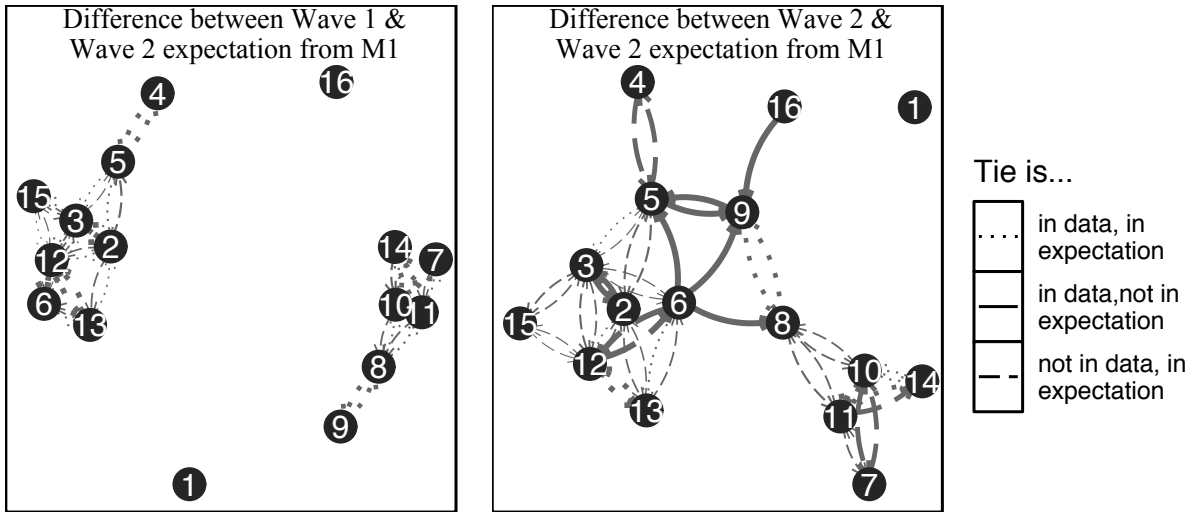


Figure 7: Network differences between the expected network from M1 and Wave 1 and Wave 2 of the data. Some edges appear in both the data and the expected Wave 2, while others only appear in the data or only in the expectation. Width of edges represent the proportion of appearances in the 1,000 simulations. The FR layout is used.

Another way to view our models in the data space is with *difference networks* between the expected network and the data, as shown in Figure 7. We created these networks by computing the difference between the adjacency matrices for the expected network and

wave 1, and for wave 2 and the expected network. The lack of model fit becomes even more clear in Figure 7, with the broken edges representing ties that only show up in the expected network, and the solid edges representing ties in the data that the model fails to capture. Dotted edges are in both the expectation and the data and represent where the model “got it right”. Again, the widths of the edges demonstrate the frequency of the ties in the simulations. On the left in Figure 7, we see the difference between the first wave of data and the expected network. We only see dotted and broken edges, meaning that the simulated networks consist primarily of edges in wave 1, plus some new edges that are not in the first wave of data. The unobserved ties are all reciprocating edges in the data, so the model predicts more reciprocity between actors than there actually is. On the right in Figure 7, we see the difference between the second wave of data and the expected network. Because the model is simulating the second wave of data, we would hope to see very few differences between the expected network and wave 2. However, there are six solid edges, which are edges in the data that do not appear at all in the expected network, and there are many broken edges that appear in the expected network but not in the data. Although there are some edges that appear in both the expected network and wave 2, the majority of edges in the expected network are not in wave 2, and there are several edges in the data missing from the expectation. We conclude that the model M1 is a poor fit to our example data, which we have seen very clearly by viewing the model in the data space.

4 Collections of models

Now that we have assessed model fit, we want to learn about model behavior in order to choose better fitting models in the future. This is the second principle of model visualization: “collections [of models] are more informative” (Wickham et al., 2015). We examine the behavior of the CTMCs by visualizing collections of them, which we construct in four ways. First, we explore the space of all possible models to determine the most significant model parameters. Second, we vary model settings to uncover relationships between parameters and their effects on the fitted model. Next, we fit the same model to the same data many times to assess the behavior of the underlying simulations. Finally, we fit the same model to different data to learn about fitted parameter values under different condi-

tions. We chose these four collections because they each explore something different about model behavior: the choice of parameters in the objective function, the effects of underlying simulations, and the resulting fitted values under different circumstances.

Exploring the space of all possible models: The `RSiena` manual contains over 80 effects to include in the objective function. To select objective function parameters for modeling our example data, we searched through all effects available to find *significant* effects. The `RSiena` package includes the function `Wald.Rsiena` for Wald-type tests of parameter significance. We start by including the outdegree and reciprocity parameters in the model then add one effect to the objective function, fit the model to the example data, test the added effect for significance, and repeat for all possible model parameters. We performed this procedure, which we used to pick M2 and M3, for both the small friendship and the senate collaboration data. A visualization of the significant effects for the two example data sets are shown in Figure 13 in the Appendix. Using this one-at-a-time testing method results in many possible significant parameters of different type and magnitude, demonstrating the flexibility of these models.

Varying model settings: The second method for viewing collections of models is varying model settings. We have done this already by choosing three different models M1, M2, M3 to fit to our example data sets. To explore the behavior of these three models, we simulate from them. Using the estimated parameter values in Table 2, we simulated 1,000 observations of waves 2 and 3 of the friendship data from each of the three models. From the simulations, we created an expected network for wave 2 from models M1, M2, and M3 using the same method described in Section 3. These expected networks are shown in Figure 14 in the Appendix, and we see that all models are failing to capture the structure of the second wave, even though the individual parameters were all significant.

Fitting the same model to the same data: The simulations required to fit the CTMC network models lead to inherent variability in the fitted values. To examine this variability, we fit models M1, M2, and M3 to the friendship network 1,000 times each. The distributions of the fitted model parameters are shown in Figure 8. Clearly, the inclusion of the JTT parameter, β_3 , results in very different estimates of the other four parameters in all models, α_1 , α_2 , β_1 , and β_2 . On the other hand, the inclusion of the indirect tie parameter,

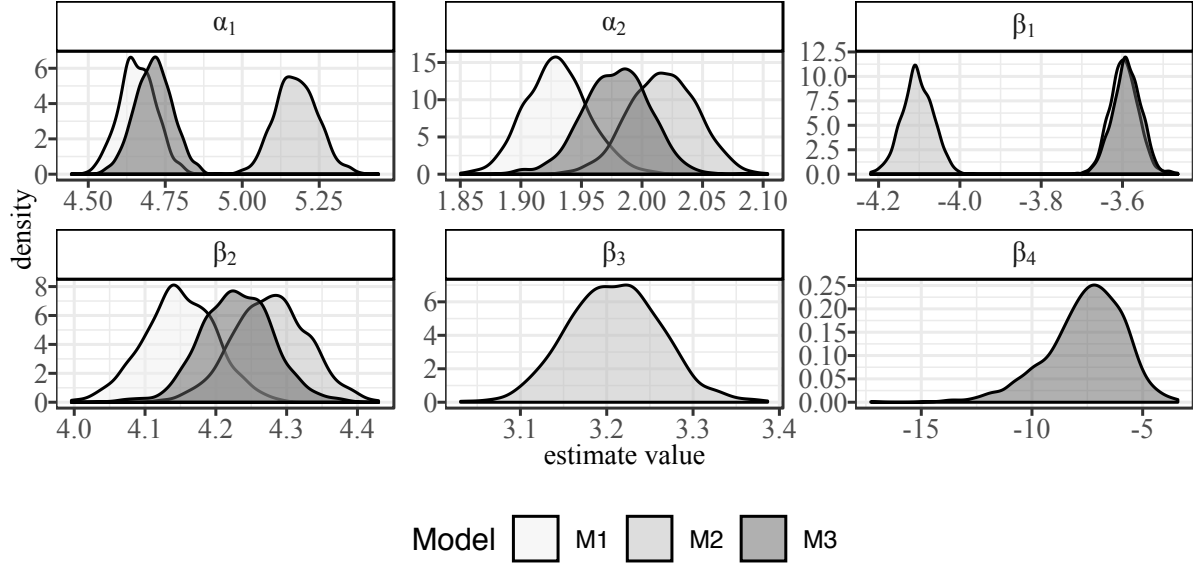


Figure 8: Distribution of fitted parameter values for our three models. The inclusion of β_3 or β_4 clearly has an effect on the distributions of the rate parameters, α_1 and α_2 , and on the other objective function parameters β_1, β_2 .

β_4 , does not lead to such drastically different estimates of other model parameters. When β_3 is included, its estimate is positive, meaning that friendships between two girls with different drinking behaviors tend to form when there is third girl who is already friends with the other two. The inclusion of β_3 also leads to increases in the estimates of α_1, α_2 . Thus, when modeling the transitive triplet behavior in this way, we may conclude that girls change friends more frequently. It is not clear, however, that the addition of a parameter to the objective function *should* effect the estimates of the rate parameters. This strong and possibly problematic relationship between rate parameter estimates and objective function parameters is further shown in Figure 16 in the Appendix. When visualizing the results of the same models fit many times to the same data, we can see potential model behavior problems right away.

Fitting the same model to different data: Finally, we view collections of models by fitting the same models to different data to understand how the model behaves in different contexts. As we did for the friendship data, we estimated the parameters for M1, M2, and M3, when fit to the senate data (see Figure 3) 100 times each. The means and standard deviations of the repeated parameter estimates for all six model-data combinations are given

	Friendship Data			Senate Data		
	M1	M2	M3	M1	M2	M3
α_1	4.660 (0.059)	5.176 (0.068)	4.712 (0.060)	3.344 (0.016)	3.349 (0.016)	3.340 (0.016)
α_2	1.930 (0.026)	2.017 (0.028)	1.979 (0.027)	2.480 (0.017)	2.487 (0.015)	2.483 (0.014)
α_3	—	—	—	2.221 (0.017)	2.227 (0.017)	2.224 (0.016)
β_1	-3.597 (0.033)	-4.104 (0.038)	-3.589 (0.035)	-4.979 (0.027)	-4.993 (0.025)	-4.987 (0.021)
β_2	4.149 (0.050)	4.277 (0.052)	4.230 (0.050)	4.954 (0.046)	4.974 (0.040)	4.970 (0.035)
β_3	—	3.209 (0.053)	—	—	-1.175 (0.789)	—
β_4	—	—	-7.582 (1.746)	—	—	-1.048 (0.486)

Table 2: The means (std. dev.) of parameter values estimated from repeated fittings of $M1, M2, M3$ to the two example data sets.

in Table 2, while corresponding density plots of the objective function parameters are given in Figure 15 in the Appendix. Looking at the estimates for both data sets, a few patterns emerge. First, we can see the same relationship between the outdegree parameter, β_1 and the reciprocity parameter, β_2 in both data sets and across all three models. The estimates of β_1 are all negative and range from about -5 and -3, while the estimates of β_2 are all positive and range from about four to five. This suggests that in both data sets, nodes prefer outgoing ties that are reciprocated to those that are not. When we consider the context, this seems obvious: the students want to have close mutual friendships, and senators want mutual support for their bills. We are also concerned, however, with the extremely high correlation between the estimates of β_1, β_2 . In all six model-data combinations, the absolute correlation between $\hat{\beta}_1, \hat{\beta}_2$ is greater than 0.93. After seeing the high correlation between parameter estimates in very different data contexts, we might question the guideline that both parameters need to be included in the objective function.

In Figure 8, we saw that the inclusion of β_3 for the friendship data has a noticeable effect on the other parameters in the model. In Figure 15, however, we see that this effect is not present in the senate data. Furthermore, the estimates of β_4 for the senate data are near zero, suggesting that this effect, which considers indirect ties, is not informative for the senate data. In contrast, for the friendship data the mean estimate of β_4 is about -7.5, which suggests that indirect ties are *strongly* discouraged from forming. Again, this is

fairly intuitive given the context: teenage girls likely want to have direct friendships with each other. Senators, on the other hand, have to wheel and deal to pass their legislation, and indirect relationships may be beneficial. By examining the results of models fitted to different data, we better understand the interpretation of the objective function parameters, and we have learned of some potential parameter correlation problems affecting the fit of the models. All in all, we have used four different ways of viewing collections of models, which have given us insight into the behavior of CTMC network models.

5 Explore algorithms, not just end result

The last model visualization principle, “explore the process of model fitting,” is crucial to understanding our models (Wickham et al., 2015). When fitting models in **RSiena**, several simulation steps are hidden from the user. One of the hidden steps in the SIENA MoM algorithm is the *microstep* process, which is an observed instance of the CTMC described in Section 1.2. At each microstep, an *ego* node is selected to change, and the chosen node randomly makes one change in its ties according to the probabilities p_{ij} defined in Equation 5. Between two network observations $x(t_m)$ and $x(t_{m+1})$, there can be dozens, hundreds, or even thousands of microsteps, depending on the size of the network and the number of tie changes between two network observations. We want to simulate and view these in-between steps in order to better understand the behavior of the underlying CTMC. Hiding the microsteps is practical and efficient if the goal is to fit a model to data, obtain parameter estimates, and draw conclusions or make predictions. But in order to understand *how* the models are fit, we need to extract and explore the different steps of the process. We use these steps to visually explore the algorithms, so that we are visualizing the model fitting process, and not just the model fitting result.

To visualize the underlying algorithm, we animate the microsteps that form the CTMC from wave 1 to wave 2 of the small friendship network (see Figure 2). The microstep animation, some frames of which are shown in Figure 9, represent one possible path from wave 1 to wave 2 according to the model M1. We show each change in the network as follows: we first increase the size and change the color of the ego node to focus attention to its ties. Then, the changing tie either fades in or fades out. If there are no changes at

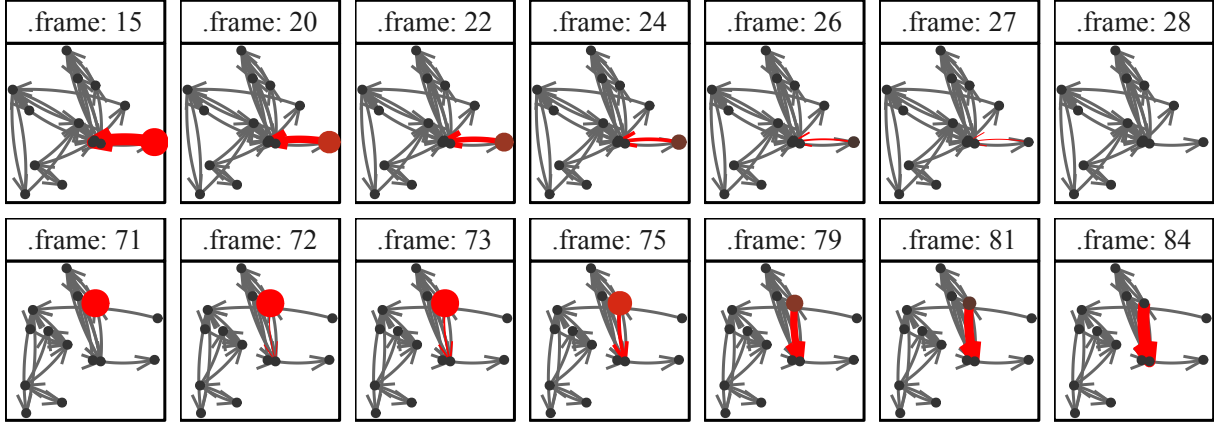


Figure 9: A selection of images in the microstep animation. The selected edges and nodes are emphasized by changing the size and the color, then they fade in or out, while the nodes change color and shrink to blend in with the other nodes. Frames 15-28 show a an edge disappearing, while frames 71-84 show an edge appearing. A spring layout is used.

a particular microstep, the microstep does not appear. Some frames of the animations are shown in Figure 9, and the full movie, created with `tweenr` and `gganimate` can be viewed at <https://vimeo.com/240089108> (Pedersen, 2016; Robinson, 2016). Movies similar to this animation were used to visualize the changes of dynamic networks in Moody et al. (2005). Frames 15-28 of Figure 9 show a deleted tie, while frames 71-84 show an added tie. For both changes, the ego node is emphasized with color and size. When a tie is removed, it is first emphasized with the same color and size as the node. As the animation proceeds, the tie shrinks and disappears as the ego node shrinks and changes color to match the others. If a tie is being added, it appears red from nothing, grows large for emphasis, then changes color and size to match the rest of the edges. In this network animation, we show the modeled steps of the *unobserved* CTMC process between two observed networks. We see each part of the hierarchical come into play: first, the selection of the ego node via the rate function, then the result of the actor maximizing its objective function by either deleting or adding a node. Finally, the layout of the network transitions as edges are removed or added, demonstrating how the overall network structure changes with each of the individual tie changes. Combining the tie changes in this animation makes the CTMC process the focus

and demonstrates the underlying change mechanism of the dynamic network according to the chosen model.

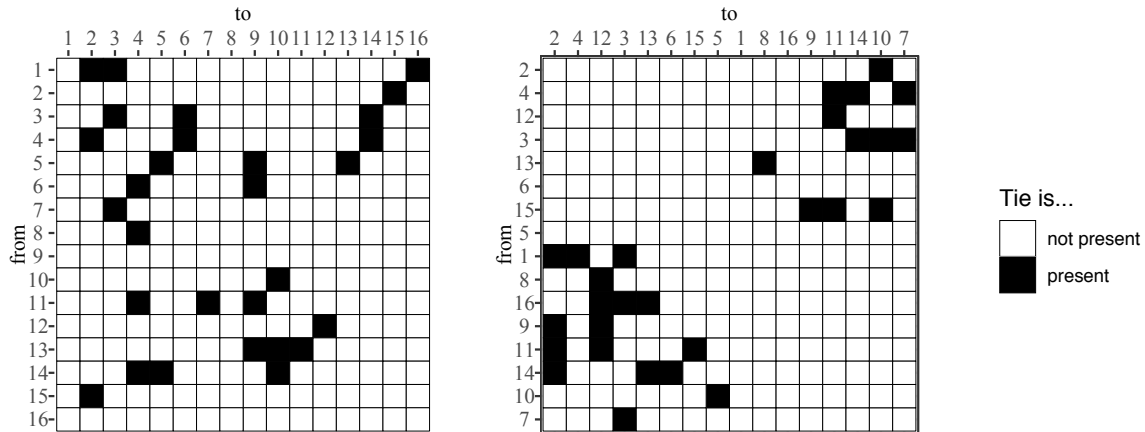


Figure 10: On the left, the starting friendship network represented in adjacency matrix form, ordered by node ID. On the right, the same adjacency matrix is presented after seriation

We continue to use animation to view the changing structure of the network via adjacency matrix visualizations. In the first adjacency matrix visualization in Figure 10, we use the node ID to order the rows and columns. This arbitrary ordering does not easily convey the underlying structure of the network. We use *matrix seriation* to reorder the rows and columns of the adjacency matrix visualization so that the structure of the network is apparent (Liiv, 2010). To seriate the matrix, we first construct a cumulative adjacency matrix, \mathbf{A}^{cum} , for the series of microsteps in the CTMC simulated from $x(t_m)$ to $x(t_{m+1})$. A single entry in the cumulative adjacency matrix, \mathbf{A}_{ij}^{cum} , is the total number of times the edge $i \rightarrow j$ appears in the network from the initial observation, $x(t_m) \equiv X(0)$ to the last microstep, $X(R)$, where R is the total number of microsteps taken:

$$\mathbf{A}_{ij}^{cum} = \sum_{r=0}^R X_{ij}(r). \quad (12)$$

We then perform a principal component analysis (PCA) on \mathbf{A}^{cum} , and use the values of the first principal component to order the vertices on the x and y axes for the adjacency matrix animation. The resulting visualization for the first wave of data is shown in Figure 10. Using PCA on \mathbf{A}^{cum} to order the rows and columns of the adjacency matrix visualization clearly shows the two distinct connected components in the first wave of the network, which are

difficult to find in the visualization ordered by node ID. We continue to use the seriated matrix in another animation of the microstep process, which is available at <https://vimeo.com/240092677>. A few frames of this video are shown in Figure 11. In the animation, a square appears or disappears as that edge appears or disappears in the microstep process. This visualization also shows the microsteps where the nodes do not make a change. From start to finish, we see the network evolve: sometimes ties are removed right after they are created, and sometimes several ties are removed or created in a row. The adjacency matrix animation shows the algorithm exploring the data space. In contrast to the previous animation, all possible edges are represented in the adjacency matrix visualization. By viewing this animation, we gain a better understanding of the underlying dynamics of this model.

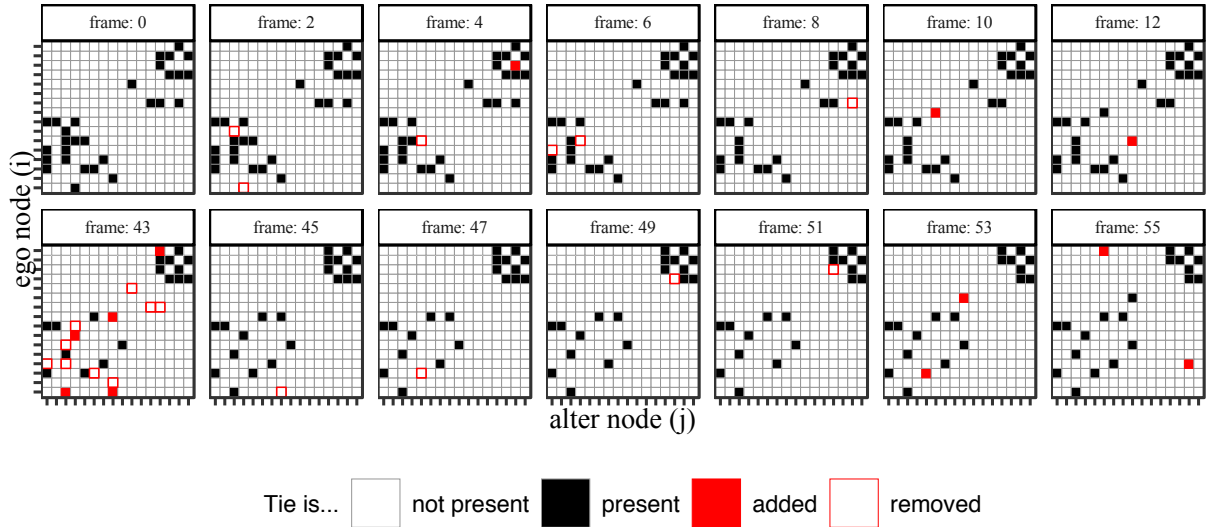


Figure 11: A selection of frames from the adjacency matrix visualization animation for one series of microsteps. The node labels are removed to declutter the figure. Starting at frame 0 (the data), there are two clearly connected components: one in the bottom left, and one in the top right. By the end, the bottom left component has become more sparse, while the top right component has shrunk, but remains closely connected.

We also illuminate the microstep process by visualizing the theoretical and observed transition probabilities for the first microstep in the process. These probabilities are defined in Equation 5. We only do the first step of many because each microstep is dependent on the

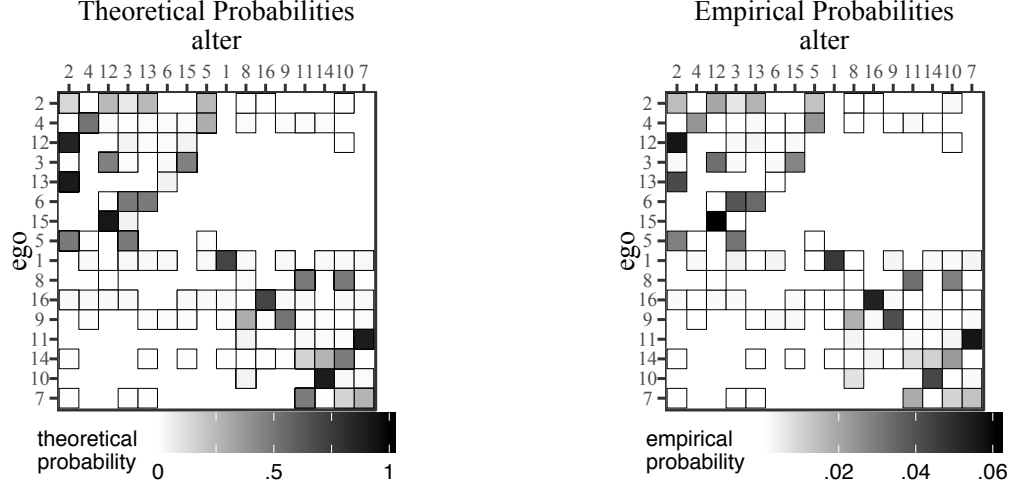


Figure 12: Adjacency matrix visualizations showing the theoretical (left) and empirical (right) transition probabilities for the first microstep taken in 1,000 simulations. The ego node is on the y-axis, and the alter node is on the x-axis. There are many ties with empirical probability zero.

previous network state. There are 1,000 transitions to examine: the first microstep taken in each one of the simulations. We use another seriated adjacency matrix visualization to show theoretical and empirical transition probabilities of the first microstep in Figure 12 for each tie that was the first change in a simulation. The figure is noticeably sparse: of the $16 \cdot 16 = 256$ possible choices for the CTMC to make, only 103, or about 43%, are made in the 1,000 simulated chains. This leaves a large area of data space unexplored by the first step in the model fitting process. We present additional evidence for lack of coverage in Figure 17 the Appendix. The Figures 12 and 17 show that the model M1 and the the corresponding fitting process do not explore the data space enough to adequately capture the network change mechanism, a finding we were only able to discern thanks to model visualization techniques.

6 Discussion

We have used traditional graphics and net-vis methods to explore a set of continuous time markov chain models for dynamic social network data in new ways. The guiding principles of model-vis: viewing the model in the data space, visualizing collections of models, and looking at the underlying algorithms, gave us knowledge and appreciation for

these complicated and flexible models. When we look at the model in the data space by viewing the expected network simulated from a model, we are able to easily see the lack of model fit. When visualizing collections of models, we uncover concerning relationships between parameters. When we explore the algorithms, we gain a better understanding of the model and its stochastic behavior. Our three-pronged model-vis attack has uncovered many properties of these models.

We have only just begun to scratch the surface of these multi-layered models for social networks. The `RSiena` software is incredibly powerful, and we can fit much more flexible hierarchical models than we have examined here. If, for example, we think the network structure or an actor covariate affects the rate of change of the network, we can incorporate that belief into the rate function. In addition, more than one actor-level covariate can be included in the model, and many more than three parameters can be included in the objective function. As we have shown, the rate and objective function parameters are often highly correlated, so researchers can now use model-vis techniques to gain a better understanding of these correlations and hopefully find ways to account for their effects in the model. In addition, extensions of the model can predict behavior change of the actors in the network, a capability we did not explore here.

Furthermore, the data sets we have used are relatively small by network data standards, with only 16 and 100 nodes. Thus, some of the visualizations we have presented may not scale up to larger networks. Future network model-vis research could also include interactive and/or three-dimensional graphics, which we did not explore here. Interactivity is the state-of-the-art in data visualization, and network visualization is just starting to catch on (see e.g. Kairam et al., 2015). Wickham et al. (2015) also relied on interactive tools like `GGobi` (Swayne et al., 2003) for interactivity, and for networks, there is some interactive support with the package `plotly` (Sievert et al., 2017) built into `geomnet`.

We hope that the principles of model-vis continue to be used for learning more about the fit and behavior of complicated statistical models. Our model-vis application has uncovered many properties of CTMC models for dynamic social networks, and potential abounds for application of model-vis in other areas of network analysis.

Appendix: Additional model visualizations

Section 4

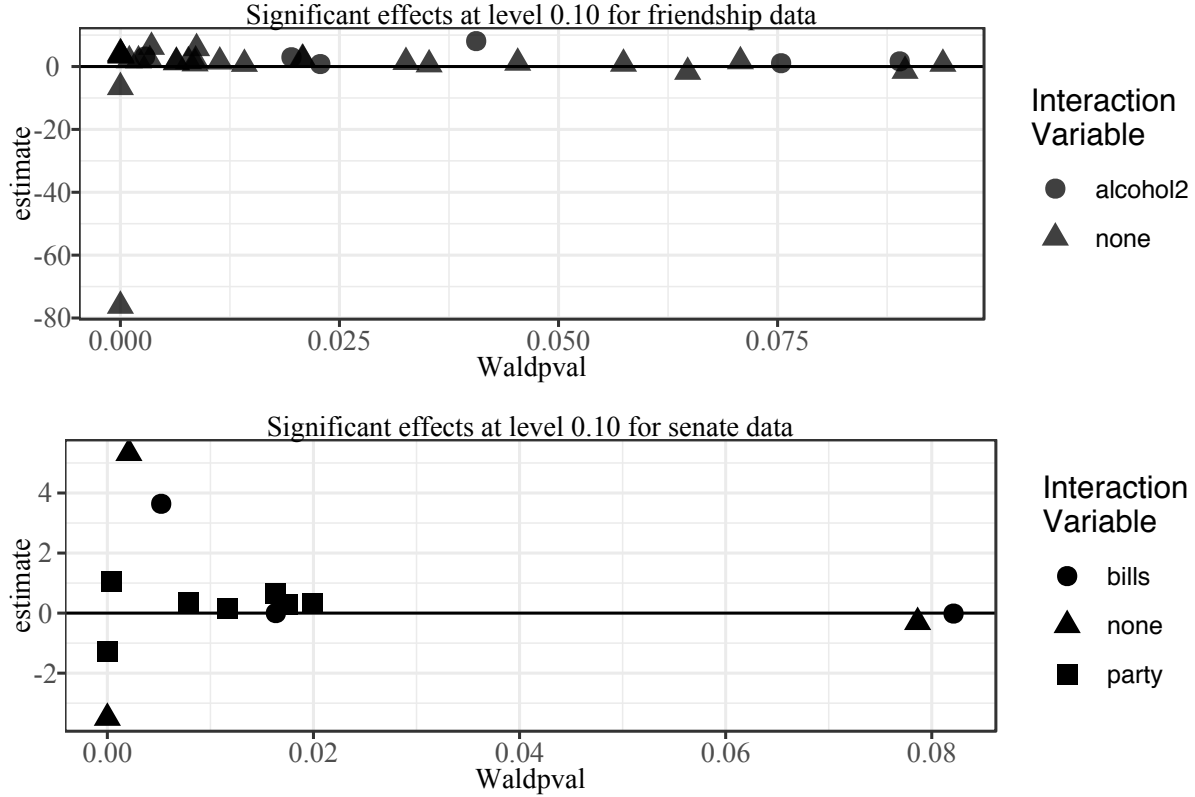


Figure 13: Significant effects for the two data sets, at a significance level of 0.10 or lower as calculated by the Wald-type test available in the SIENA software.

In Figure 13, we see for the friendship data and the senate data that most of the significant effects have absolute value less than ten. In addition, the p -values for the effects from the friendship data are more spread out than the p -values for the senate data, which are concentrated at about 0.02 or less. This suggests that larger data sets tend to result generally in smaller p -values, which is consistent with the construction of Wald-type tests.

To create the expected network visualization shown in Figure 14, we follow the same procedure as in Section 3, counting occurrences of each realized edge in the simulations. The resulting summary network has weighted edges representing the proportion of edge appearances in all 1,000 simulations of wave 2. As in Figure 6, edges only appear in the expected networks if they appear more than 5% of the time in the simulations. In Figure 14,

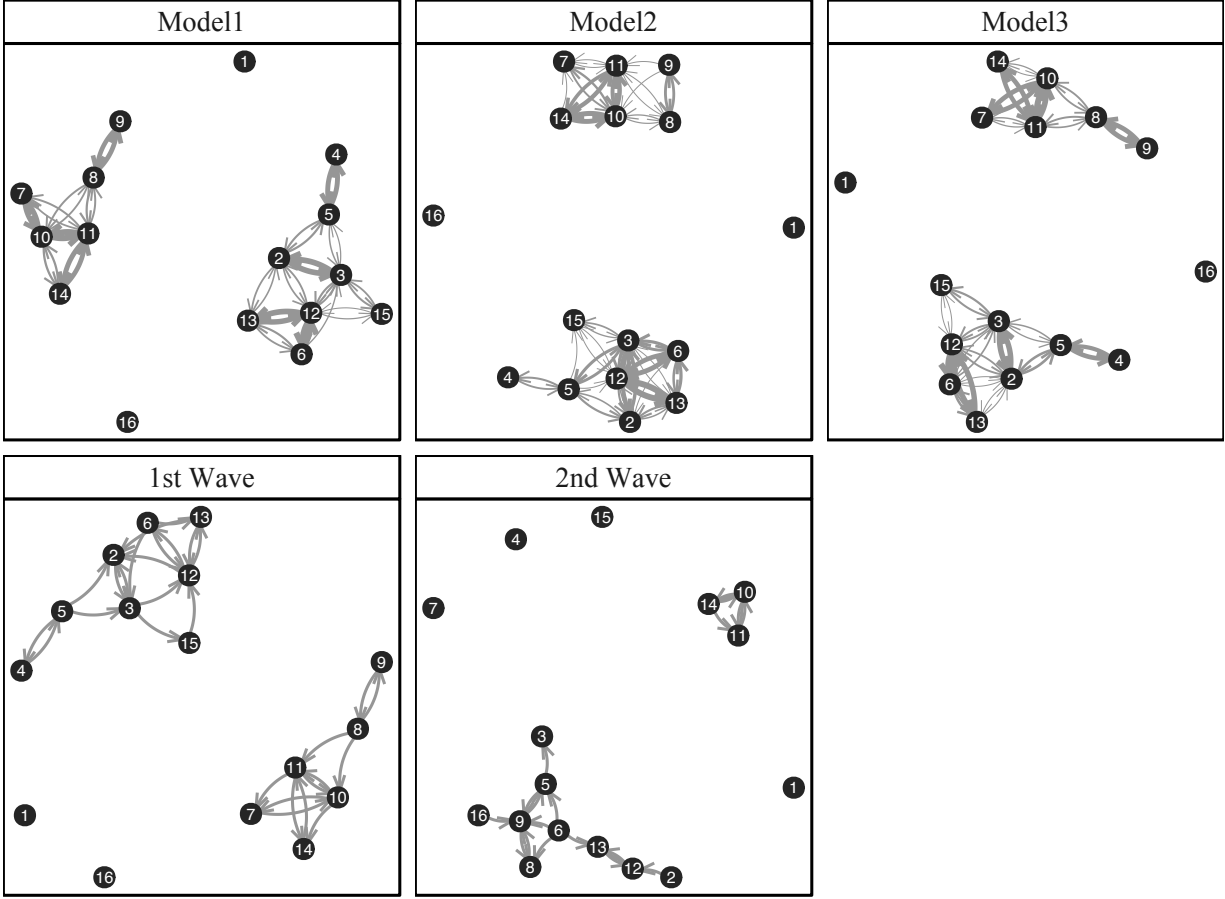


Figure 14: The node-link diagrams from the three expected networks that we calculated are in the top row, and the true wave 1 and wave 2 data are shown in the bottom row above. There is some difference between the three models, but overall, these three models struggle to capture the structure in the true second wave of data.

we show the expected network from the three models we fit and the first and second waves of data. Comparing the three averages to waves 1 and 2, we see that they have very similar structure to wave 1 but that model M2, which included the transitive triplet parameter, results in more strongly connected components. The lack of fit is clear: none of the three average networks show node 16 gaining ties as it does in wave two, nor do they show nodes 4 and 7 becoming isolated. In model M2, however, the ties to node 7 appear much weaker than in model M1 or model M3, and the nodes $\{10, 11, 14\}$ are also strongly connected as

they are in wave 2 of data, suggesting that M2 may be the best model of the three.

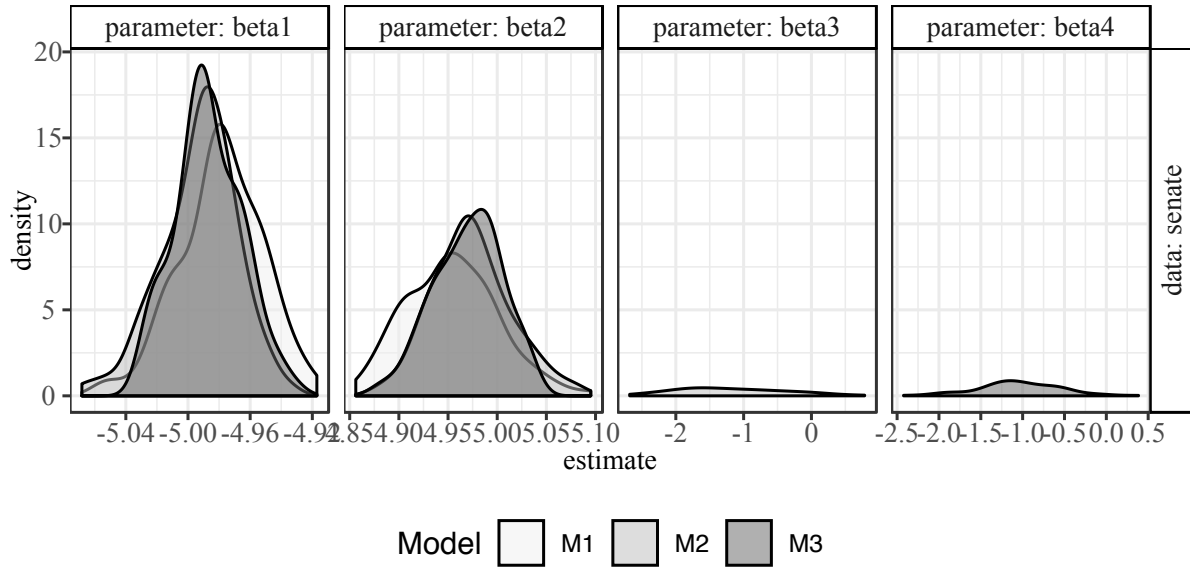


Figure 15: Density plots of objective function parameter estimates from repeatedly fitting models M1, M2, and M3 to the example data.

The density plots in Figure 15 show the distribution of the parameter estimates for the objective function parameters in M1, M2, and M3 the two example data sets we used. The most notable difference is in the values of β_1, β_2 . In the senate data, the inclusion of β_3 has no effect on the estimates of β_1, β_2 , while the opposite is true for the friendship data.

In Figure 16, we examine correlations between each of pair of parameters within each model and overall. The strongest correlation within each model is between β_1 and β_2 , with absolute correlation between them greater than 0.90 in all three models. The β_1 parameter is also highly correlated with the β_3 parameter within model M2, but it is not as highly correlated with the β_4 parameter in model M3. It might therefore be advisable to consider only models that either allow β_1 or β_2 but not both. Looking at the high correlation with α , we might switch to a model without β_1 .

Section 5

Finally, we combine 1,000 simulations from model M1 into a visualization that displays the entire microstep process in Figure 17. To make this visualization, we first assign each

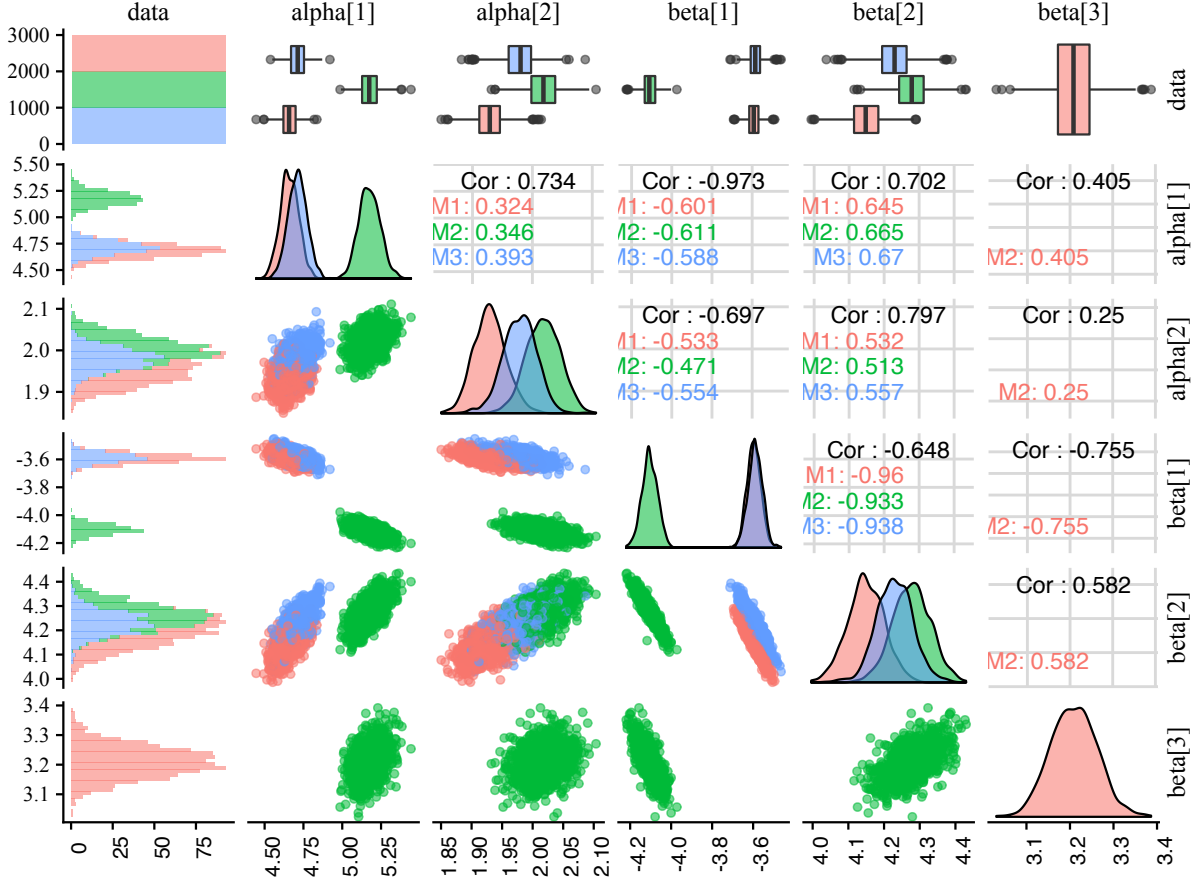


Figure 16: A matrix of plots demonstrating the strong correlations between parameter estimates. The strongest correlation within each model is between β_1 and β_2 .

possible edge in the network an ID number so that we can keep track of it throughout all microsteps and all simulations. Then, we count up the number of times each edge appears in the network throughout the microstep process for each of the 1,000 simulations. We also count the number of times an edge occurs at each step number. Since the number of microsteps in the process varies, the number of times an edge occurs decreases as the step number increases. Next, we compute a proportion, which we call the occurrence percentage, which is the number of times the edge was in the network in step 1, step 2, etc. divided by 1,000. Finally, we visualize all this information together in Figure 17. In this plot, we see that all possible edges appear at least once at some point in the microsteps of at least one simulated process. We also see, however, that the process struggles to focus in on the edges in the second wave of the data. Ideally, we would like to see more occurrences the

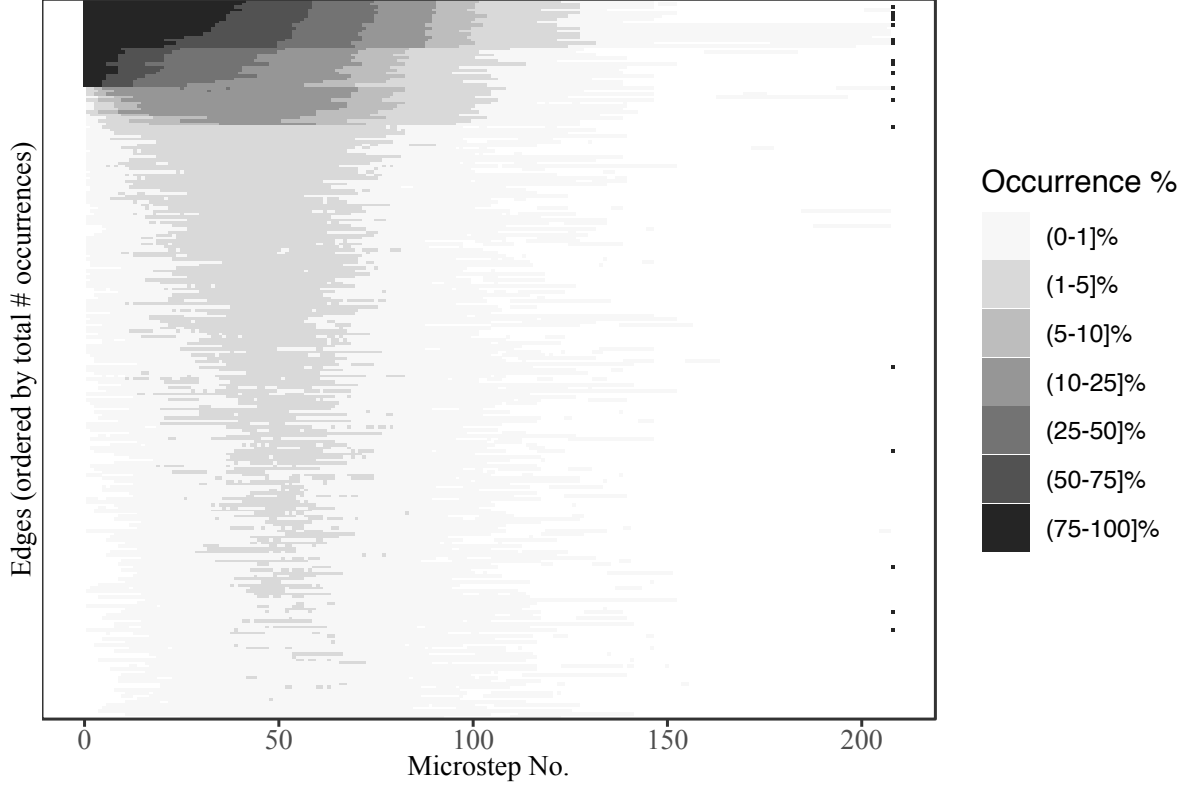


Figure 17: Visualizing all microsteps taken in 1,000 simulations from the model M1. The occurrence percent is split up into groups to correspond with its distribution: only about 10% of the possible edges appear more than 10% of the time in the 1,000 simulations, while about 60% appear less than 1% of the time. The first wave network is shown at microstep 0, and the second wave of the network is shown as the last microstep for comparison. We see that it is rare for a microstep process to last longer than 150 steps, and also that the edges that appear past the 150th step tend to be in either the first wave or the second wave.

edges in wave 2 that are not in wave 1. But, about half of the edges in wave two are in the bottom half of the figure, which means they do not appear as much as they would if the model was actually capturing the mechanisms of tie change in the network. More of the darkest areas of the figure should belong to ties in wave 2, but those are often the lightest. This solidifies what we found in Figure 12: the fitting process does not explore the data space enough to adequately capture the network change mechanism.

Online supplemental material: Much of the code used to create the visualizations in this paper has been packaged and made available online at <https://github.com/>

sctyner/netvizinf and <https://github.com/sctyner/geomnet> (also on CRAN). Additional, non-packaged code can be found at <http://bit.ly/ctmc-model-viz>.

References

- Fekete, J.-D. (2009), “Visualizing Networks using Adjacency Matrices: Progresses and Challenges,” in *11th IEEE International Conference on Computer-Aided Design and Computer Graphics*, pp. 636–638.
- Frank, O. and Strauss, D. (1986), “Markov Graphs,” *Journal of the American Statistical Association*, 832–842.
- Fruchterman, T. M. and Reingold, E. M. (1991), “Graph Drawing by Force-Directed Placement,” *Software: Practice and Experience*, 21, 1129–1164.
- Ghoniem, M., Fekete, J.-D., and Castagliola, P. (2005), “On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis,” *Information Visualization*, 4, 114.
- Gibson, H., Faith, J., and Vickers, P. (2013), “A survey of two-dimensional graph layout techniques for information visualisation,” *Information Visualization*, 12, 324–357.
- Gross, J. H., Kirkland, J. H., and Shalizi, C. R. (2008), “Cosponsorship in the U.S. Senate: A Multilevel Two-Mode Approach to Detecting Subtle Social Predictors of Legislative Support,” *Unpublished Manuscript*, http://www.latinodecisions.com/files/4013/3840/2978/Gross-Kirkland-Shalizi_Multilevel-Cosponsorship_PolAnlys-submission.pdf Last accessed: 15 Oct. 2018.
- Hoff, P. D., Raftery, A. E., and Handcock, M. S. (2002), “Latent Space Approaches to Social Network Analysis,” *Journal of the American Statistical Association*, 97, 1090–98.
- Holland, P. W. and Leinhardt, S. (1977), “A dynamic model for social networks,” *The Journal of Mathematical Sociology*, 5, 5–20.

- Kairam, S., Riche, N. H., Drucker, S., Fernandez, R., and Heer, J. (2015), “Refinery: Visual Exploration of Large, Heterogeneous Networks through Associative Browsing,” *Computer Graphics Forum (Proc. EuroVis)*, 34.
- Kamada, T. and Kawai, S. (1989), “An Algorithm for Drawing General Undirected Graphs,” *Information Processing Letters*, 31, 7–15.
- Knuth, D. E. (2013), *Combinatorics: Ancient and Modern*, Oxford University Press, chap. Two thousand years of combinatorics.
- Liiv, I. (2010), “Seriation and Matrix Reordering Methods: An Historical Overview,” *Statistical Analysis and Data Mining*, 3, 70–91.
- Michell, L. and Amos, A. (1997), “Girls, pecking order and smoking,” *Social Science & Medicine*, 44, 1861–1869.
- Moody, J., McFarland, D., and Bender-deMoll, S. (2005), “Dynamic Network Visualization,” *American Journal of Sociology*, 110, 1206–1241.
- Pedersen, T. L. (2016), *tweenr: Interpolate Data for Smooth Animations*, R package version 0.1.5.
- Ripley, R., Boitmanis, K., Snijders, T. A., and Schoenenberger, F. (2016a), *RSiena: Siena - Simulation Investigation for Empirical Network Analysis*, R package version 1.1-304/r304.
- (2016b), *RSienaTest: Siena - Simulation Investigation for Empirical Network Analysis*, R package version 1.1-294.
- Ripley, R. M., Snijders, T. A., Boda, Z., Vrs, A., and Preciado, P. (2017), *Manual for RSiena*, University of Oxford: Department of Statistics; Nuffield College; University of Groningen: Department of Sociology, https://www.stats.ox.ac.uk/~snijders/siena/RSiena_Manual.pdf.
- Robbins, H. and Monro, S. (1951), “A stochastic approximation method,” *The Annals of Mathematical Statistics*, 22, 400–407.

- Robinson, D. (2016), *gganimate: Create easy animations with ggplot2*, R package version 0.1.0.9000.
- Schloerke, B., Crowley, J., Cook, D., Hofmann, H., Wickham, H., Briatte, F., Marbach, M., and Thoen, E. (2016), *GGally: Extension to ggplot2*, R package version 1.3.0.
- Sievert, C., Parmer, C., Hocking, T., Chamberlain, S., Ram, K., Corvellec, M., and Despouy, P. (2017), *plotly: Create Interactive Web Graphics via 'plotly.js'*, R package version 4.7.1.
- Snijders, T. A. (2016), *Siena Algorithms*, University of Oxford: Department of Statistics, https://www.stats.ox.ac.uk/~snijders/siena/Siena_algorithms.pdf.
- Snijders, T. A., van de Bunt, G. G., and Steglich, C. E. (2010), “Introduction to stochastic actor-based models for network dynamics,” *Social Networks*, 32, 44 – 60.
- Snijders, T. A. B. (1996), “Stochastic actor-oriented models for network change,” *Journal of Mathematical Sociology*, 21, 149–172.
- (2001), “The Statistical Evaluation of Social Network Dynamics,” *Sociological Methodology*, 31, 361–395.
- (2017), “Stochastic Actor-Oriented Models for Network Dynamics,” *Annual Review of Statistics and Its Application*, 4, 346–363.
- Swayne, D., Temple Lang, D., Buja, A., and Cook, D. (2003), “GGobi: Evolving from XGobi into an Extensible Framework for Interactive Data Visualization,” *Computational Statistics & Data Analysis*, 43, 423–444, see also <http://www.ggobi.org>.
- Tamassia, R. (ed.) (2013), *Handbook of Graph Drawing and Visualization*, CRC Press.
- Tyner, S. and Hofmann, H. (2016), *geomnet: Network Visualization in the 'ggplot2' Framework*, R package version 0.2.0.9001.
- Wickham, H., Cook, D., and Hofmann, H. (2015), “Visualizing statistical models: Removing the blindfold,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 8, 203–225.

Yin, G. G. and Zhang, Q. (2010), *Continuous-Time Markov Chains and Applications: A Two-Time-Scale Approach*, New York: Springer, 2nd ed.