Visualizations for Diagnosing 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

July 15, 2019

Abstract

Statistical models for networks have been studied for decades, and they have gotten more complex as computing power has grown. Because of their complexity, traditional model diagnostic methods are unavailable. In this paper, we propose new methods for diagnosing network models. We use a three-pronged visualization attack: (a) network visualization of the fitted model, (b) visualizations of collections of models to reveal relationships between parameters, and (c) visualizations of algorithms to gain insights into the underlying network mechanisms. The visual diagnostics we propose serve the same purpose for network models as residual plots for assessing goodness-of-fit in linear models.

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

Contents

1 Introduction

Scholars have been trying to understand relationships between people, or *social networks*, for decades, especially in the social sciences. There are a host of quantitative methods available for analyzing social networks, including statistical models, such as exponential random graph models (ERGM) and latent space models (LSM), that strive to capture the underlying network-generating mechanism (Frank and Strauss, 1986; Hoff et al., 2002). The ERGM and LSM families are examples of static network models, which focus on a single "snapshot" of a network and "global network statistics" such as outdegree distribution (Goldenberg et al., 2010). Other static network model families include the Erdös-Rényi random graph model and the exchangeable graph model (Goldenberg et al., 2010). These static models, however, are not optimal for modeling real networks because they do not account for changes in the network as time passes. In real networks, edges may be created or destroyed, and nodes can be born or die (Goldenberg et al., 2010). Dynamic models for networks, which model networks observed at many points in time, are the more realistic alternative to static network models.

Dynamic network models do not have as rich of a history as static network models, largely due to the historical lack of data and available analytical tools (Goldenberg et al., 2010). With the advent of social media and ever-increasing computational power, however, more and more researchers are turning their attention toward dynamic network models. Some of the classical models, such as the Erdös-Rényi model and the small-world model of Watts and Strogatz (1998) are pseudo-dynamic because they could be applied in a dynamic way but rarely are (Goldenberg et al., 2010). Fully dynamic social network models, however, are more complicated: they consider both the network structure and its changes in time. If we can capture this change mechanism over time, we can truly begin to model the underlying network dynamics that drive social connections. A comprehensive review of dynamic network models is presented in Goldenberg et al. (2010).

We are particularly interested in one truly dynamic family of network models: continuous time Markov chain (CTMC) models, first introduced in Holland and Leinhardt (1977). We chose these models because they correspond to our intuition about how social networks are formed, and they are very flexible and widely applicable. In these models, network

changes are modeled hierarchically at the node level: nodes are given opportunities to change via a rate function, then the node chooses which edge to change according to its utility function. In each step of the hierarchy, many possible effects can be implemented to capture the change mechanism. For example, the rate function could be a simple waiting-time model, or it could depend on a node's indegree or outdegree, if we have reason to believe that the more popular nodes change their ties more frequently. In addition, we can incorporate node covariate information into the hierarchy, such as gender or education level of people in a social network. For instance, the genders of two people could affect whether and how quickly they establish a relationship. People also will likely form more relationships with those who share their education level. Thus, the transition probabilities between two states of a network depend on the network's structure, the characteristics of its nodes, or both (Snijders, 1996). For example, Schaefer et al. (2011) use a CTMC model to learn about the tendency of depressed adolescents to form friendships and found that depressed students tended to avoid creating new friendships over time. The CTMC model family can also be expanded to model node behavior, as in Chyzh (2016), who applies CTMC models to show that more indirect trade relationships in a country decrease its overall respect for human rights. Thus, with CTMC models we can examine assumptions made about social networks.

With the flexibility of CTMC models comes increased complexity and estimation difficulty (Snijders, 2017). Likelihood functions quickly become very complex due to the inherent dependency structure in the data, and therefore computational methods, such as Markov chain Monte Carlo (MCMC), are used to fit models. Methods for maximum likelihood estimation (MLE) for these models were not introduced until Snijders et al. (2010b), and method of moments (MM) estimation is frequently used (Ripley et al., 2016). For CTMC parameter estimation, we use the software SIENA, and its `R` implementation `RSiena` (R Core Team, 2016; Ripley et al., 2016). This software is a considerable contribution to the field of social network analysis, allowing for estimation of structural effects, node covariate effects, as well as modeling both network and node changes in a wide variety of CTMC models. This extensive parameter estimation process is largely hidden from the user, and the primary outputs are parameter estimates. Direct simulation from a given

network model with fixed parameter values is also made easy with `RSiena`. This software opens up a great deal of possibilities for network modeling and simulation.

Since we chose the CTMC models for dynamic social networks, we would like to determine how well they fit data. We chose these models because they match our intuition about how social networks form in reality, and after we fit a model, we will want to assess its goodness-of-fit. In statistical network models generally, the lack of large-sample theory leads to a lack of goodness-of-fit tests (Goldenberg et al., 2010). Most methods for assessing goodness-of-fit for network models, therefore, involve comparing statistics computed on simulated networks to the same statistics computed on the data, for example in Hunter et al. (2008a). More recently, Schweinberger (2012) used score-type tests for forward model selection in the CTMC model context. The `RSiena` software also contains methods for goodness-of-fit based on network statistics, such as outdegree distribution, where the data values are compared to simulated values as in Hunter et al. (2008a) and in Ripley et al. (2016).

We propose new methods for goodness-of-fit and other diagnostics that use visualization as a supplement to computational methods. Network visualization, also called graph drawing, is a prominent subfield of network analysis with a rich history: early drawings of complete graphs date back to 1280 CE (Knuth, 2013). We use this visualization foundation to combine modeling with visualization. Note that we do not propose new methods of visualizing networks, and for a complete review of the field of graph drawing we direct the reader to Tamassia (2013). What is novel in our work is the use of existing visualization methods to diagnose network models with model visualization.

Model visualization or "model-vis" complements traditional numerical model diagnostic tools to enrich our understanding and interpretation of statistical models (Wickham et al., 2015). 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 3. By translating the model visualization work of Wickham et al. (2015) to the dynamic network model space, we provide network researchers with a new suite of tools for visualizing and diagnosing performance of their network models. Though we apply the tools to CTMCs because we find the models intuitive, the methods

5

we use can be applied to any network model, static or dynamic. The visual diagnostics proposed here serve the same purpose for network models as residual plots for assessing goodness-of-fit in linear models.

The rest of this paper is structured as follows. In Section 2, we provide an overview of network data and visualization and give a more detailed description of CTMC models and how we fit them with `RSiena` (Ripley et al., 2016). In Section 3, we introduce the motivating data and CTMC models we assess using model-vis. Then, in Section 4, we explore the first step in model-vis by placing the models in the data space to determine how well the models fit the data. Section 5 follows, in which we visualize collections of models to explore their behavior. Finally, in Section 6, we visualize the underlying algorithms in the CTMC fitting process to gain insight into model fits. We conclude with a discussion in Section 7.

2   Network Models and Visualizations

Network data across fields have similar structure: they consist of 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, such as age or gender, and the type of relationship (friends, coworkers, etc) that an edge represents. Dynamic networks are observed at two or more discrete points in time, where each observation of a network is often called a "wave" of data.

Visualizing Network Data: 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 primary network visualization or "net-vis" methods: the node-link diagram and the adjacency matrix visualization (Fekete, 2009). We demonstrate these with a simple example. Consider the network with five nodes, $\{1, 2, 3, 4, 5\}$, connected by five directed edges: $\{2 \to 4, 3 \to 4, 1 \to 5, 3 \to 5, 5 \to 4\}$ shown in Figure 1.

[Figure 1 about here.]

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 at left in Figure 1. Because there is often no natural placement of the nodes, they are placed in 2D space at random, then adjusted with a layout algorithm. 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. For a full discussion of available node-link layout algorithms, we direct the reader to Gibson et al. (2013). In Figure 1, we use the KK algorithm, and unless otherwise stated, all other node-link diagrams in this paper are laid out using the FR algorithm (Fruchterman and Reingold, 1991).

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

$$A_{ij}(x) = \begin{cases} 1 & \text{if } x_{ij} = 1, \quad i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

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 x_{ij} (the edge $i \to j$) is filled in if $A_{ij} = 1$ and is empty if $A_{ij} = 0$. For undirected networks $\mathbf{A}$ is symmetric, so the adjacency matrix visualization is also symmetric, with each edge drawn 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). Furthermore, 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 visualize our example data and CTMC models.

2.1 Defining CTMC Network Models

The continuous-time Markov chain (CTMC) 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., 2010a)). CTMC models reflect our intuition about social networks: most social networks, even holding constant the set of actors over time, are everchanging as relationships form or dissolve. Actors in social networks also have properties that could 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.

2.1.1 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, $t_1, \ldots, t_M$ with $t_1 < t_2 < \cdots < t_M$. We denote the network observed at timepoint t_m by $x(t_m)$ for $m = 1, \ldots, M$. 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 $x(t_2), \ldots, x(t_{M-1})$. Unlike the first and last observations of the network, $x(t_2), \ldots, x(t_{M-1})$ do not have direct correspondence with steps in the CTMC. Thus, the observations $x(t_2), \ldots, x(t_{M-1})$ are treated as "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 the selected node randomly

changes its ties according to probabilities derived from the *objective function.*

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 $m = 1, \ldots, M - 1$. The rate parameter, α_m, dictates how often an actor i is selected to change one of its ties, x_{ij}, for $j \neq i \in \{1, \ldots, n\}$ in the time period from t_m to t_{m+1}. The CTMC model assumes that the actors i are conditionally independent given their ties $x_{i1}, \ldots, x_{in}$ at the current network state. Let $\tau(i|x_{i1}(m), \ldots, x_{in}(m))$ 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,

$$\tau(i|x_{i1}(m), \ldots, x_{in}(m)) \overset{\text{iid}}{\sim} Exp(\alpha_m) \tag{2}$$

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 (MM) is used with the statistic

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

which is the total number of edge differences between $x(t_m)$ and $x(t_{m+1})$.

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*:

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

where x is the current network state, $\mathbf{Z}$ is a matrix of node covariates, and K is the total number of parameters in the objective function. The parameters of the model, $\beta_1, \dots, \beta_K$ correspond to network statistics, $s_{ik}(x, \mathbf{Z})$, for $k = 1, \dots, K$. To maximize $f_i(\boldsymbol{\beta}, x, \mathbf{Z})$, i will either destroy a tie, create a tie, or make no change.

The parameters of $f_i(\boldsymbol{\beta}, x, \mathbf{Z})$ are either structure or covariate parameters: structure parameters correspond to statistics that are *only* functions of the current network state, x, while covariate parameters are also functions of the covariates, $\mathbf{Z}$. According to Snijders (2001, p. 371), there should be at least two parameters 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 Figure 3 with visualizations of the corresponding effects. In the RSiena software, there are over 80 possible parameters β_k with corresponding statistics $s_{ik}(x, \mathbf{Z})$ to add to the model, the definitions of which are provided in Ripley et al. (2017). We define two additional parameters and their corresponding statistics in Figure 3, and we discuss all parameters more in Section 3.

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 probability $p_{ij}(\boldsymbol{\beta}, x, \mathbf{Z})$. 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 new network. The probability that the tie x_{ij} changes is defined as:

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

These probabilities are used to define the intensity matrix.

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 directed network, denoted $\mathcal{X}$, contains $2^{n(n-1)}$ states: there are two possible states for an edge, $\{0, 1\}$, and $n(n-1)$

10

possible edge relationships. 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 the diagonal entry for the current state.

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

$$
q_{bc} = \begin{cases} q_{ij} = \alpha_m p_{ij}(\boldsymbol{\beta}, x^b) & \text{if } x^c_{ij} = 1 - x^b_{ij} \text{ for some } i \neq j \in \{1, 2, \ldots, n\} \\ -\sum_{i \neq j} q_{ij} & \text{if } x^b = x^c \\ 0 & \text{otherwise} \end{cases}
$$

Thus, 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 $\mathbf{Q}$ completely defines the CTMC model for network change.

2.1.2   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 MM or maximum likelihood (ML) estimation (Ripley et al., 2016). We chose to use MM estimation, so we describe that process here, while more information on ML estimation can be found in Snijders et al. (2010b); 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.

Estimation conditions on the first observed network, $x(t_1)$, then proceeds to estimate model parameters for all future observations. The full SIENA MM estimation algorithm operates in three phases which we briefly summarize here. Full description of the algorithmc can be found in Ripley et al. (2017) and Snijders (2016). The first phase obtains initial estimates of the score functions for use in the second phase. The second phase is the Robbins-Monro algorithm, which results in parameter estimates via iterative parameter updates and subsequent simulation from the CTMC at each set of parameter values. The third phase uses the parameters estimated in phase two to estimate the score functions and covariance matrix of the parameters and carries out convergence checks. In Section 6, we further explore these phases in the SIENA MM algorithm through visualization.

3   Example Data and Models

[Figure 2 about here.]

Example Data: To guide our visual exploration of CTMC models for dynamic networks, we use a canonical example in the social network literature: a subset of the 50 actor data set from the "Teenage Friends and Lifestyle Study" (Michell and Amos, 1997). We chose this data because it is a well-known and easy to understand example. We were all teenagers once, and we know the volatility and fragility of those relationships, as well as why they might change. 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. In the small data set, there are 16 teenage girls sampled over three years, who were asked about their friends and their behaviors, such as how often they smoke cigarettes or drink alcohol (Michell and Amos, 1997). Each girl was asked to identify her friends in the school to form a directed network. We visualize this data with a node-link diagram in Figure 2. Each girl is represented by a node and an arrow is drawn from her to each of the friends she has reported. Sometimes, two girls identify each other as friends, as shown by nodes 8 and 9 in the first sample, on the left in Figure 2. Often, the ties are not reciprocated, as shown by the edges from node 5 to nodes 2 and 3 in the first sample. We also color nodes in Figure 2 according to each girl's self-reported drinking behavior: does not drink, drinks

once or twice per year, drinks once per month, and drinks once per week. 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 the behavioral forces at play.

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. Though our models are very simple, the model-vis techniques we present can easily be applied to much more complex models. The first model of interest, M1, contains the absolute minimum number of parameters in the objective function $f_i(x)$:

$$M1: \quad f_i(x) = \beta_1 s_{i1}(x) + \beta_2 s_{i2}(x), \tag{7}$$

where $s_{i1}(x)$ is the density network statistic and $s_{i2}(x)$ is the reciprocity network statistic for actor i at the current network state x (see Figures 3a & 3b). 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 (`Wald.RSiena()`) provided in the `RSiena` software (Ripley et al., 2016). The model M2 contains an actor-level covariate parameter, and the model M3 contains an additional structural effect in the objective function.

$$M2: \quad f_i(x, z) = \beta_1 s_{i1}(x) + \beta_2 s_{i2}(x) + \beta_3 s_{i3}(x, z) \tag{8}$$

$$M3: \quad f_i(x) = \beta_1 s_{i1}(x) + \beta_2 s_{i2}(x) + \beta_4 s_{i4}(x), \tag{9}$$

[Figure 3 about here.]

where $s_{i3}(x, z)$ and $s_{i4}(x)$ are defined in Figures 3c & 3d. These statistics are called the "number of jumping transitive triplets" (JTT) and the "number of doubly achieved distances two effect" (N22), respectively in Ripley et al. (2017). The JTT effect emphasizes triads formed between actors with *different* covariate values, while the N22 effect emphasizes *indirect* ties between actors. The four model parameters are visualized in Figure 3, where the ties affected by the parameter value are represented by dotted lines. If the estimates of the parameters $\hat{\beta}_1, \ldots, \hat{\beta}_4$ 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. Mean estimates from 1,000 repeated fittings are given in Table 1, which we use when we simulate from the network models.

[Table 1 about here.]

The values in Table 1 show us that all effects are very significant and relatively stable across the different models. Estimates of β_1 are negative in all three models, meaning that creating new outgoing ties is discouraged, while β_2 estimates are positive in all three models, meaning that creating reciprocated ties is encouraged. Thus in the context of teenage girls, we interpret this to mean they do not want to form new friends unless they are becoming friends with a girl who already thinks of them as a friend. The estimate of β_3 is positive, meaning that girls want to be close friends with their friend's friends even if they have different drinking behaviors. And finally, the estimate of β_4 is negative, so indirect relationships are discouraged. What we do not learn from the values in Table 1 is anything about the goodness of fit of each of the models, or dependencies between the parameters and their estimates.

Our goal here is to assess the fit of models M1, M2, and M3, to the data, not to find the best model for this data, which is why we have kept both the data and the models very small. A complete analysis of the network would include all of the available data, including additional covariates related to smoking and other behaviors. The exact model structure and effects we choose are not as important to us as our goal of assessing their impact on model fit. We keep the data and model simple in order to apply the more complicated model visualization tools that we discuss in Sections 4-6.

4   Model in the data space

Placing the model in the data space is crucial to understanding how well the model represents the observed data. 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 to visualize the three data spaces together (Tyner and Hofmann, 2016). The package draws the network using node-link diagrams, and node or edge data are then mapped

14

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 line type of the edges between nodes can be used to represent variables in the edge data. Finally, the networks observed at each time point are placed side-by-side, showing the network evolve over time. Bringing all network information together in `geomnet` allows us to show the entire data space at once.

4.1   Expectation Networks

[Figure 4 about here.]

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 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 do not illuminate the overall structure of the network. How then, can we arrive at an expected network? We propose the following method:

1. Condition on $x(t_1)$ and simulate 1,000 instances of $x(t_2)$ and $x(t_3)$ from the model of interest using previously estimated parameter values (see Table 1). Call these simulations $x^{(1)}(t_m), x^{(2)}(t_m), \dots x^{(1000)}(t_m)$ for $m = 2, 3$.

2. Count the number of times each edge $i \to j$ appeared in one of the 1,000 network simulations. i.e. compute $x_{ij}^{(B)}(t_m) = \sum_{b=1}^{1000} x_{ij}^{(b)}(t_m)$ for $m = 2, 3$.

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. i.e. $\bar{x}_{ij}(t_m) = x_{ij}^{(B)}(t_m)/1000$.

4. Create the *expected network* by including only the edges that appeared in more than 5% of the simulations. i.e. $\bar{x}_{ij}(t_m) := \bar{x}_{ij}(t_m) \cdot \mathbb{I}(\bar{x}_{ij}(t_m) > 0.05)$. We write the expected networks as $\bar{x}(t_2)$ and $\bar{x}(t_3)$.

15

The above method can be generalized and applied to any network model, static or dynamic, from which we can simulate, for any number of simulations from the network. In step 4, we chose 5% because that is the typical significance level in statistical tests, and other values could be chosen. The expected networks from our three models of interest are shown in the top row of Figure 4. Edges that appear with higher frequency are drawn with thicker lines. Edge frequency equal to one indicates that the edge was present in all 1,000 simulations used to create the expected network. On the bottom row of Figure 4, we show the data for comparison: $x(t_1)$ on the left, and $x(t_2)$ on the right. Visualizing the expected network in this way places models M1, M2, and M3, in the data space, and helps assess model fit. For model M1 we can see that the overall structure of the average network is much more similar to $x(t_1)$ than to $x(t_2)$. In $x(t_1)$ and each of the three $\bar{x}(t_2)$, there are two clusters of nodes: $\{2, 3, 4, 5, 6, 12, 13, 15\}$ and $\{7, 8, 9, 10, 11, 14\}$. The expected networks, however, do not resemble $x(t_2)$, which they are supposed to represent. In $x(t_2)$, there is one small, strongly connected cluster and one larger, sparsely connected cluster, with four unconnected nodes. Because the expected networks $\bar{x}(t_2)$ more closely resemble $x(t_1)$ than $x(t_2)$, we take this as evidence that none of the models are a good fit to the data. Model M2, which included the transitive triplet parameter, results in more strongly connected components, as shown by the increased number of edges and the thicker lines between nodes. The lack of fit in all models is clear: not one of the three model expectations show node 16 gaining ties as it does in $x(t_2)$, nor do they show nodes 4, 7, and 15, becoming isolated. In model M2, however, the ties to node 4, 7, and 15, appearless frequently than in model M1 or model M3, and the nodes $\{10, 11, 14\}$ are also more often connected, as they are in wave 2 of data. This suggests that, despite the fact that neither of the three models is a good fit, M2 may be the best fitting model of the three. In a complete data analysis, we can use this information to select more models for analysis that utilize transitivity and covariate effects like β_3, as opposed to structural and indirect effects like β_4.

4.2   Residual Networks

[Figure 5 about here.]

Another way to view our models in the data space is with *residual networks* between

16

the expected network and the data, as shown in Figure 5. We created these networks by computing the residual between $\bar{x}(t_2)$ and $x(t_1)$, $\mathbf{A}(x(t_1)) - \mathbf{A}(\bar{x}(t_2))$, and the residual between $\bar{x}(t_2)$ and $\bar{x}(t_2)$, $\mathbf{A}(x(t_2)) - \mathbf{A}(\bar{x}(t_2))$ for model M1. The lack of model fit is apparent on the right in Figure 5, where the dashed edges represent ties that only show up in the expected network, and the solid edges represent 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 is correct. On the left in Figure 5, we see the residual network between $x(t_1)$ and the expected network. We see mostly dotted edges, meaning that the simulated networks consist primarily of edges in $x(t_1)$, plus some new edges represented by the dashed lines that are not in the data. The unobserved ties are all reciprocating ties in the data, so the model is predicting more reciprocity between actors than there actually is. On the right in Figure 5, we see the residual network between $x(t_2)$ and the expected network. Because the model is simulating $x(t_2)$, we would hope to see very few differences between the expected network and the data. However, there are six solid edges, which are in the data but do not appear at all in the expected network, and there are many dashed edges that appear in the expected network but are not in $x(t_2)$. Although the residual network shows some edges that appear in both the expected network and $x(t_2)$, the majority of edges in the expected network are not in the observed data, and there are several edges in the data missing from the expectation. We therefore 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. As with the expected network, we can apply the residual network visualization other network models to determine how closely the model fits the data.

5 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. To assess model behavior, we employ the second principle of model visualization: "collections are more informative" (Wickham et al., 2015). We rely on three methods for constructing collections of CTMC models. First, we explore the space of all possible models to determine significant model parameters. Second, we vary model settings to uncover relationships between parameters and their effects on the

fitted model. Finally, we fit the model many times to assess the behavior of the underlying simulations. We chose these three collections because they each explore something different about model behavior: the choice of and relationships between parameters in the objective function, and the stochastic effects from simulation.

Exploring the space of all possible models: The `RSiena` manual contains over 80 effects to include in the objective function (Ripley et al., 2017). To select objective function parameters for modeling our example data, we searched through all available parameters to find significant effects. The `RSiena` package includes the function `Wald.Rsiena()` for Wald-type tests of parameter significance (Ripley et al., 2016). 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. 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. We chose models M2 and M3 in addition to model M1 to examine more closely because their added effects had the smallest p-values from the Wald-type tests. This particular type of model exploration is applicable to any network model with an `R` package or other software that allows the user to identify model parameters, such as the `ergm` package, which has well over 100 effects for fitting exponential-family models to networks (Handcock et al., 2018; Hunter et al., 2008b).

Varying model settings: The second method for obtaining collections of models is to vary model settings. We have done this already by choosing three different models M1, M2, M3 to fit to our example data sets. CTMC models, however, are much more flexible than the three simple models we have chosen. Other models could have included different rate functions for each node, where for example the rate of change could depend on drinking behavior if we have reason to believe that girls who drink more change friends more frequently. We could also model the behavior changes, not just the network changes, if we believe that friends influence behavior via peer pressure. This method of gathering a collection of models can be used for any network model which allows the user to vary hyperparameters in the prior and model parameters, such as the hierarchical Bayesian network estimation model in the `R` package `sna` (Butts, 2016).

Fitting the model repeatedly: 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 resulting means and standard deviations of the estimates are given in Table 1, and the distributions of all parameter estimates are shown in Figure 6. Figure 6b shows a **diagnostic pairs plot**, in which we can see that the addition of β_3 to the model results in very different estimates the of the other four parameters, α_1, α_2, β_1, and β_2. In fact, the diagnostic pairs plot reveals that the distributions of α_1 and β_1 for M2 do not overlap at all with the distribution of the same parameters from model M1. 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, the model concludes that girls change friends more frequently. It is not clear, however, that the addition of a parameter to the objective function should affect the estimates of the rate parameters.

On the other hand, the inclusion of the indirect tie parameter, β_4, does not lead to such drastically different estimates of other model parameters. The mean estimate of β_4 is large and negative, which suggests that indirect ties are strongly discouraged from forming. This is fairly intuitive given the context: teenage girls likely want to have direct friendships in tightly knit groups, so indirect relationships are not common. Finally, in Figure 6b, we demonstrate the relationship between each of pair of parameters common to all three models. The strongest correlation within each model is between β_1 and β_2, with absolute correlation greater than 0.90 in all three models. The β_1 parameter is also highly correlated with the β_3 parameter within model M2 ($\rho = $ -0.755), but it is not as highly correlated with the β_4 parameter in model M3 ($\rho = 0.419$). The inherent relationship between effects is a known problem in network analysis (Goldenberg et al., 2010), and the diagnostic pairs plot can be used for any network model where parameters are estimated with MCMC methods.

6   Explore algorithms, not just end result

The third and final model visualization principle, "explore the process of model fitting," is crucial to understanding the CTMC models (Wickham et al., 2015). When fitting models in `RSiena`, several simulation processes are hidden from the user. We refer to one of these hidden pieces in the SIENA MM algorithm as the *microstep process*, also called the ministep process in Snijders (2016). The microstep process is a series of state changes in the CTMC as described in Section 2.1. At each microstep, an ego node is selected to change, and then randomly makes one change in its ties according to to the probabilities p_{ij} defined in Equation 5. The microstep process is a key piece of the model fitting procedure. During modeling, the process proceeds until the network differs from the $x(t_m)$ by exactly C_m, defined in Equation 3. 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 value of C_m. We want to simulate and view these steps in order to better understand the behavior of the CTMC model. By default, the software discards the microsteps, which 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, which is possible in `RSiena` with some small changes to the default settings. We visualize the microsteps to visually explore the algorithms, so that we are visualizing the model fitting process, and not only the model fitting result.

[Figure 7 about here.]

To visualize the underlying process, we animate the microsteps that form the CTMC from $x(t_1)$ to $x(t_2)$ for the example data. The microstep animation, some frames of which are shown in Figure 7, represents one possible path from $x(t_1)$ to $x(t_2)$ 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 a particular microstep, the microstep does not appear. Some frames of the animations are shown in Figure 7, and the full movie, which we created using `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 7 show a tie that is destroyed, while frames 71-84 show a tie that is created. 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 model: 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. The layout of the network also adjusts as edges are removed or added, demonstrating how the overall network structure changes with each of the individual tie changes. Combining the microstreps in this animation makes the CTMC process the focus, demonstrating the underlying change mechanism of the dynamic network according to the model.

With the node-link animation, we can see how the model behaves. Some tie changes appear to alter the overall network structure, as shown by the change in the layout of the nodes, while others seem to be inevitable, such as an added reciprocal tie between two nodes in a large connected group that does not change the layout. Other edges are created in one step and destroyed the next. When watching the animation, we are watching the model in action, which helps us better understand its behavior. This same animation method can be applied to any network model which changes one tie at a time. For example, the Erdös-Rényi random graph model is a series of tie additions, and we could animate the creation of a network from this model. Unlike CTMCs, however, Erdös-Rényi random graph models only create ties (Goldenberg et al., 2010). Regardless of the network model under consideration, animating the node-link diagram to view how the network changes gives insight into the model behavior and the process of model fitting.

[Figure 8 about here.]

We continue to use animation to view the changing structure of the network via adjacency matrix visualizations. Two adjacency matrix visualizations are shown in Figure 8.

At left in Figure 8, we use the node ID to order the rows and columns. This arbitrary ordering does not easily convey the underlying structure of the network, so we use *matrix seriation* to reorder the rows and columns of the adjacency matrix visualization to make the structure of the network more obvious (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, $\mathbf{A}^{cum}_{ij}$, is the total number of times the edge x_{ij} 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:

$$\mathbf{A}^{cum}_{ij} = \sum_{r=0}^{R} X_{ij}(r). \tag{10}$$

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 net-vis. The resulting visualization for the first network observation is shown on the right in Figure 8. Using matrix seriation on the adjacency matrix visualization clearly shows the two distinct connected components in the first network, which are difficult to find in the visualization ordered by node ID at left. We use the seriated matrix in an adjacency matrix animation of the microstep process, which is available at `https://vimeo.com/240092677`. Some frames of the video are shown in Figure 9. In the animation, a square appears or disappears when the edge it represents appears or disappears in the microstep process. This animation, unlike the previous one, shows the microsteps where the chosen node does not make a change. Starting at frame 0 (the first network observation $x(t_1)$), there are two clearly connected components: one in the bottom left, and one in the top right. With each frame, 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. By the end, the bottom left component has become more sparsely connected, while the top right component has shrunk, but remains closely connected. In contrast to the node-link animation, the adjacency matrix animation shows how the algorithm explores the space of all possible networks, because all possible edges are represented in each frame of the adjacency matrix animation. It is also easier to see the connected components of the graph in this animation because we have clearly grouped them with matrix seriation.

As with the node-link animation, the adjacency matrix animation could also be used for other dynamic network models. Because the adjacency matrix emphasizes edges, this animation method is best for large directed networks whose structure is easier to see in an adjacency matrix visualization than in a node-link visualization (Ghoniem et al., 2005). Both animation methods demonstrate model behavior that we would not have seen with traditional visualizations or computations.

[Figure 9 about here.]

7 Discussion

We have introduced four diagnostic visualizations, the expected network, the residual network, the node-link animation, and the adjacency matrix animation, to complement existing numerical diagnostics for network models. These visual diagnostics give insights into the network model fit that purely numerical methods do not. Though we have applied these methods to CTMC models for dynamic social networks, all of the diagnostics presented can be applied to other dynamic network models and most can also be applied to any static network model. By following the guiding principles of model visualization, viewing the model in the data space, visualizing collections of models, and looking at the underlying algorithms, we gained insight into and appreciation for a complicated and flexible family of network models (Wickham et al., 2015). The expected network and the residual network let us view the model in the data space, and we are then able to easily assess model goodness-of-fit. Visualizing the residual network between the data and the expected network also helps assess model fit by clearly showing us exactly where the model fits the network well and where it lacks fit. Exploring collections of models give us the opportunity to learn more about the large number of parameters possible in CTMC models. The diagnostic pairs plot visualizes collections of models, giving us insight into the marginal distribution of parameters and their covariance structure. Finally, we used animation of node-link diagrams and the adjacency matrix visualizations to explore the underlying algorithms and expand our understanding of the model's stochastic behavior. All in all, this three-pronged

23

model-vis attack has uncovered many properties of CTMC models for social networks and shows great potential for application to other network models.

Using existing `R` packages, we have demonstrated how to apply model-vis to one family of models for social network analysis. The models and data we have used are multilayered and multidimensional, making them more complex than common statistical models, such as linear models. As models become more complicated, visualization becomes more important to model diagnosis because increasing complexity makes it harder to adequately summarize model fit only using numerical methods. Even more complex models, such as those for machine learning, can also benefit greatly from model visualization techniques without a lot of additional effort. The visualization methods we have used, such as the node-link diagram, are not new but we have used them in conjuction with model simulation to learn a great deal about our models of interest.

We explored a few simple models from the complicated and multilayered CTMC model family for social networks, but many more than three parameters can be included in the objective function, and more than one actor-level covariate can be included at either level in the hierarchical model. We can also model the rate of change differently for each actor if, for example, we think an actor's covariate affects its rate of change. With increased model complexity, there is more potential for adverse effects, such as the correlation between parameters that we saw in Section 5. Armed with model-vis techniques, researchers can gain a better understanding of these and other adverse effects and can then find ways to alleviate them the model. In addition, extensions of the CTMC model can also model behavior change of the actors in the network, which we haveused as model covariates. Because of its flexibility, the CTMC family of models for dynamic networks has immense potential for additional model visualization applications.

While the data we have used are quite small by network data standards, which favors node-link views, we also discuss adjacency matrix visualization, which scale up to large networks. Additionally, interactive and/or three-dimensional graphics, which we did not explore here, could be used for larger networks. 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` built into `geomnet` (Sievert et al., 2017; Tyner and Hofmann, 2016).

The principles of model-vis provide a solid foundation on how to use visualization to gain insight into more complex models, such as network 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.

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

Butts, C. T. (2016), *sna: Tools for Social Network Analysis*, R package version 2.4.

Chyzh, O. (2016), "Dangerous liaisons: An endogenous model of international trade and human rights," *Journal of Peace Research*, 53, 409–423.

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.

Goldenberg, A., Zheng, A. X., Fienberg, S. E., and Airoldi, E. M. (2010), "A Survey of Statistical Network Models," *Foundations and Trends in Machine Learning*, 2, 129–233.

Handcock, M. S., Hunter, D. R., Butts, C. T., Goodreau, S. M., Krivitsky, P. N., and Morris, M. (2018), *ergm: Fit, Simulate and Diagnose Exponential-Family Models for Networks*, The Statnet Project (`http://www.statnet.org`), R package version 3.9.4.

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.

Hunter, D. R., Goodreau, S. M., and Handcock, M. S. (2008a), "Goodness of fit of social network models," *Journal of the American Statistical Association*, 103, 248–258.

Hunter, D. R., Handcock, M. S., Butts, C. T., Goodreau, S. M., and Morris, M. (2008b), "ergm: A Package to Fit, Simulate and Diagnose Exponential-Family Models for Networks," *Journal of Statistical Software*, 24, 1–29.

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.

R Core Team (2016), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

Ripley, R., Boitmanis, K., Snijders, T. A., and Schoenenberger, F. (2016), *RSiena: Siena - Simulation Investigation for Empirical Network Analysis*, R package version 1.1-304/r304.

Ripley, R. M., Snijders, T. A., Boda, Z., Vörös, 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.

Schaefer, D. R., Kornienko, O., and Fox, A. M. (2011), "Misery Does Not Love Company: Network Selection Mechanisms and Depression Homophily," *American Sociological Review*, 76, 764–785.

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.

Schweinberger, M. (2012), "Statistical modelling of network panel data: Goodness of fit," *British Journal of Mathematical and Statistical Psychology*, 65, 263–281.

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. (2010a), "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.

Snijders, T. A. B., Koskinen, J., and Schweinberger, M. (2010b), "Maximum Likelihood Estimation for Social Network Dynamics," *The Annals of Applied Statistics*, 4, 567–588.

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.

Watts, D. J. and Strogatz, S. H. (1998), "Collective dynamics of 'small-world' networks," *Nature*, 393, 440.

Wickham, H. (2009), *ggplot2: Elegant Graphics for Data Analysis*, useR, Springer.

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.

List of Figures

Node-Link Diagram

Adjacency Matrix Visualization

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.

Figure 2: The small friendship network data we model throughout this paper. Each node represents a teenager, an edge is drawn when a girl identifies another as her friend. By comparing the node-link diagrams we see that teenagers' drinking behavior changes – drinking generally increases; darker nodes indicate more occurrences – and the network becomes less densely connected over time. These are the mechanisms we hope to model with our chosen CTMCs.
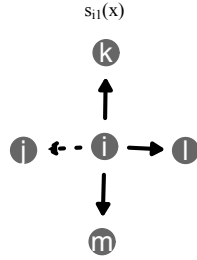
(a) Demonstrating the outdegree effect β_1. $s_{i1}(x) = \sum_j x_{ij}$
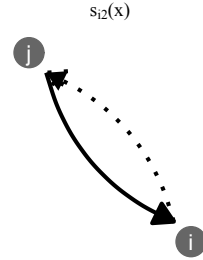
(b) Demonstrating the reciprocity effect β_2. $s_{i2}(x) = \sum_j x_{ij}x_{ji}$
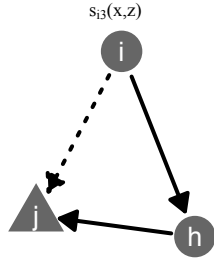
(c) Demonstrating a JTT, where the covariate is represented by the shape of the nodes. $s_{i3}(x, z) = \sum_{j \neq h} x_{ij}x_{ih}x_{hj}\mathbb{I}(z_i = z_h \neq z_j)$
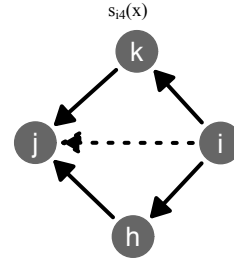
(d) Demonstrating a N22 between actors i and j. $s_{i4}(x) = |\{j : x_{ij} = 0, \sum_h x_{ih}x_{hj} \geq 2\}|$

Figure 3: The network statistics whose corresponding effects are included in the models we fit to the friends data. In each drawing, the dotted tie is encouraged to for if the parameter is positive, and is discouraged otherwise.
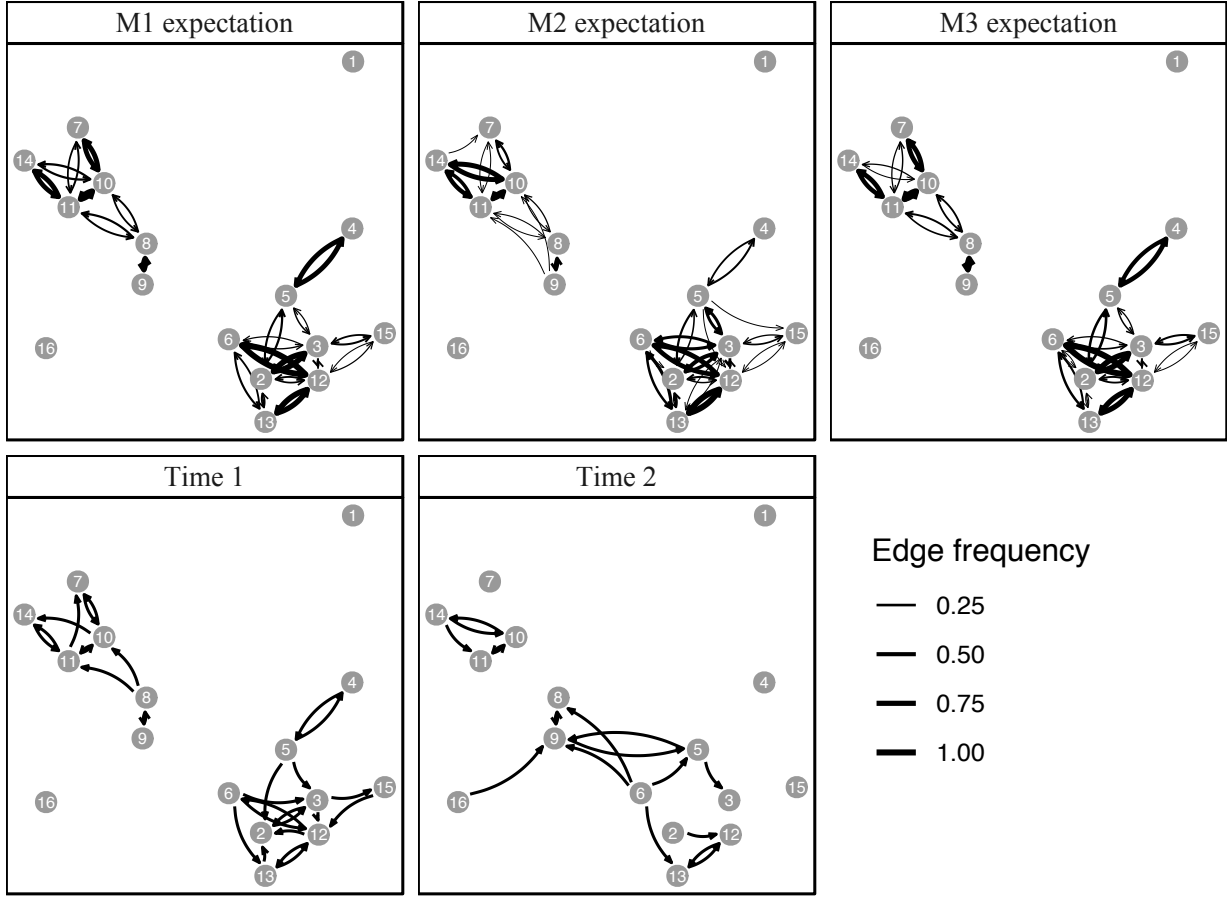
Figure 4: In the bottom left, the network at the first time point $(x(t_1))$, which is conditioned on in the model. In the bottom middle, the network at the second time point $(x(t_2))$. In the top row, the three expected networks for time point 2 $(\bar{x}(t_2))$ according to model M1, M2, M3. The legend shows how edge width corresponds to frequency in the three expected networks.
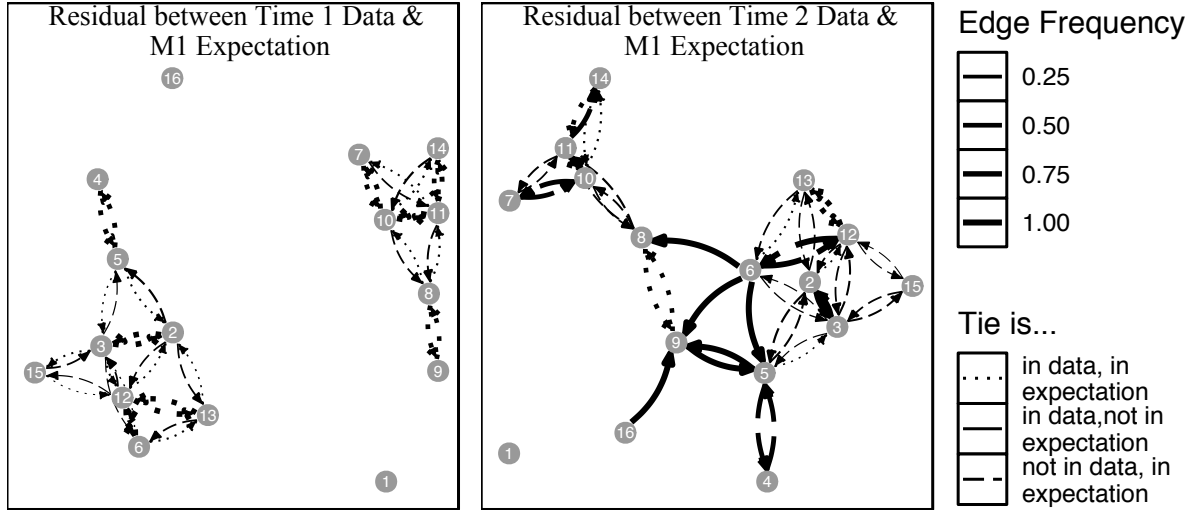
Figure 5: Residual networks between the expected network from model M1 $\bar{x}(t_2)$ and $x(t_1)$ and $x(t_2)$. Some edges appear in both the data and the expected network $\bar{x}(t_2)$ (dotted lines), while others only appear in the data (solid lines) or only in the expectation (dashed lines). Width of dotted and dashed edges represent the frequency of appearances in the 1,000 simulations.
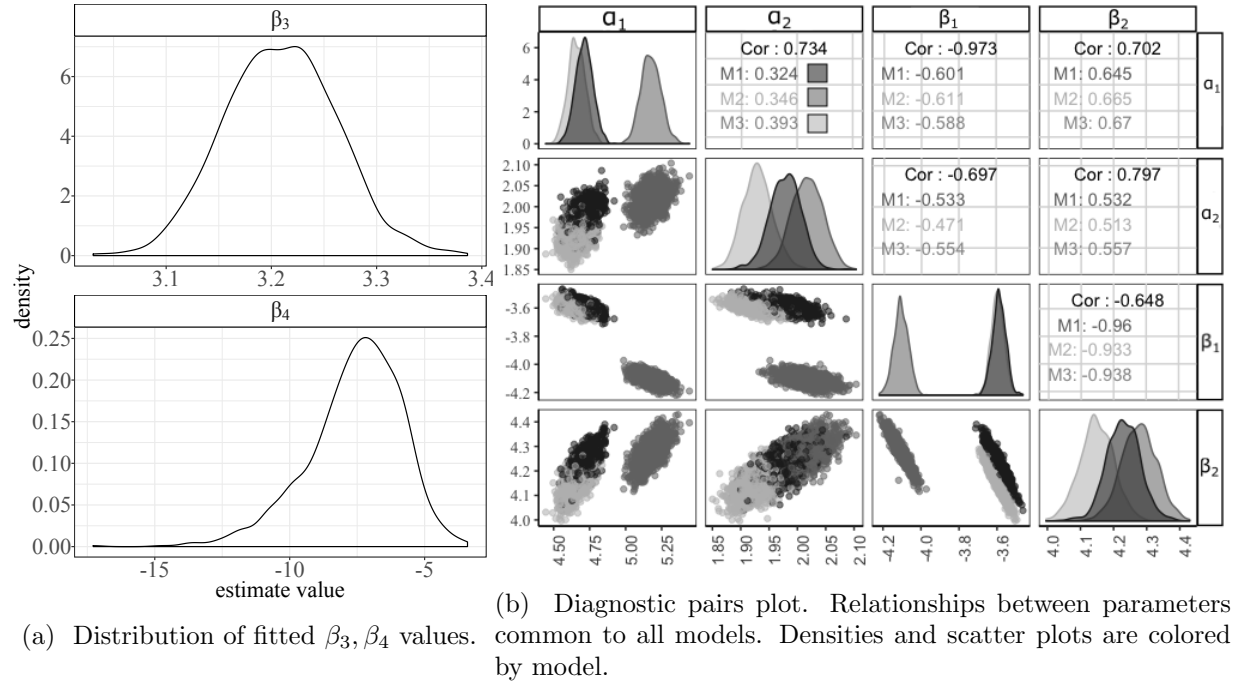
(a) Distribution of fitted β_3, β_4 values.

(b) Diagnostic pairs plot. Relationships between parameters common to all models. Densities and scatter plots are colored by model.

Figure 6: Distribution of fitted parameter values for our three models. The inclusion of β_3 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, while the effect of β_4 is not as noticeable.
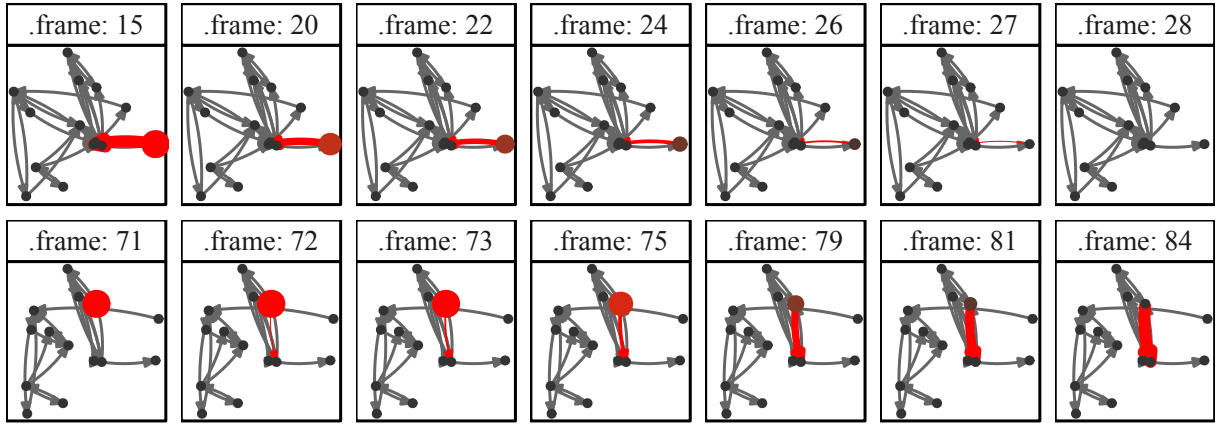
Figure 7: Some frams from the animation at `https://vimeo.com/240089108`. The selected edges and nodes are emphasized by changing size and color, then they fade in or out while the nodes change color and shrink to blend in with the other nodes. Frames 15-28 show edge destruction, while frames 71-84 show edge creation. A spring layout is used (Butts, 2016).
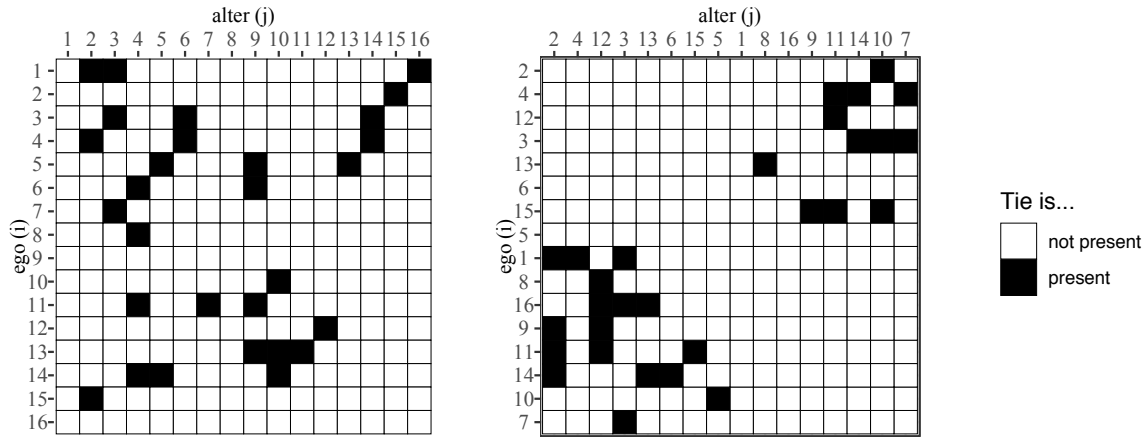
Figure 8: On the left, the first observation of the friendship network represented in adjacency matrix form, ordered by node ID. On the right, the same adjacency matrix is presented after seriation
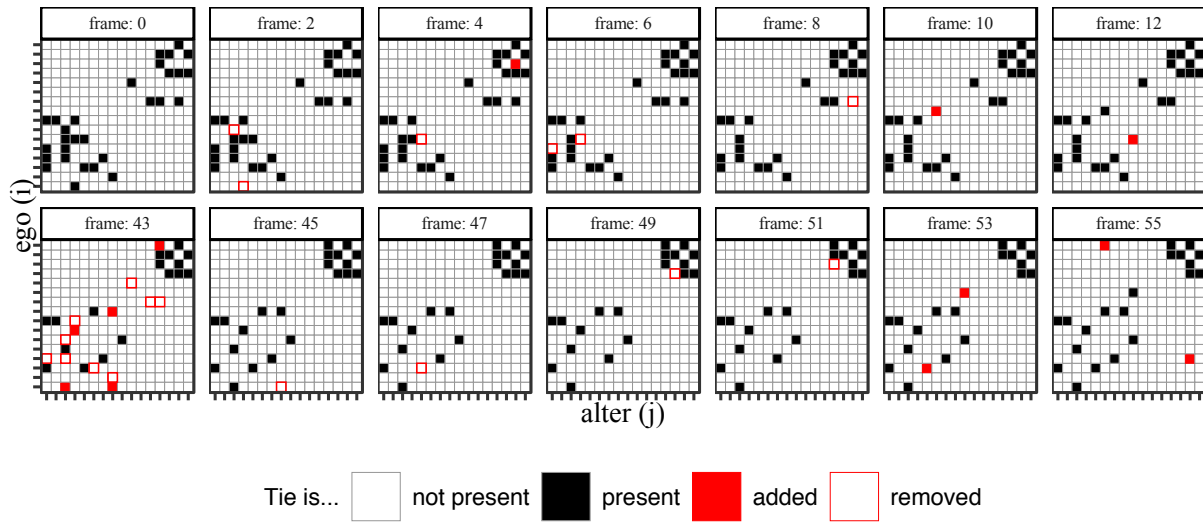
Figure 9: A selection of frames from the adjacency matrix visualization animation for one series of microsteps. The node labels are removed to declutter the figure. Complete video available at https://vimeo.com/240092677.

List of Tables

Effect Name	Parameter	M1	M2	M3
Rate (time $1 \rightarrow 2$)	α_1	4.660 (0.059)	5.176 (0.068)	4.712 (0.060)
Rate (time $2 \rightarrow 3$)	α_2	1.930 (0.026)	2.017 (0.028)	1.979 (0.027)
Outdegree	β_1	-3.597 (0.033)	-4.104 (0.038)	-3.589 (0.035)
Reciprocity	β_2	4.149 (0.050)	4.277 (0.052)	4.230 (0.050)
JTT	β_3	–	3.209 (0.053)	–
N22	β_4	–	–	-7.582 (1.746)

Table 1: The means (standard deviations) of parameter values estimated from 1,000 repeated fittings of M1, M2, and M3 to the example data.