Graphical Discovery in Stochastic Actor-Oriented Models for Social Network Analysis

Samantha C. Tyner

2016-09-23

A proposal submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Major: Statistics

Program of Study Committee:

Heike Hofmann, Major Professor
Dianne Cook
Alicia Carriquiry
Cindy Yu
Olga Chyzh

Iowa State University
Ames, Iowa
2016

Contents

1 Literature Review

Social networks have been studied for decades, beginning with a few foundational works, the most well known of which is the 1967 study, “The Small World Problem” by Stanley Milgram (Goldenberg et al. 2010). But in recent years, the study of social networks has grown wildly in popularity due to an increase in the availability of and easy of access to social network data. The digital revolution has led to the creation of social media, linking people from all over the world in a way we never have been before. Now that platforms like Facebook, Twitter, and LinkedIn permeate our world, just about everyone knows what social networks are. In academic circles, collaboration networks are a type of social network that have been extensively studied and can even be a point of pride, like a mathematician’s Erdös number (“The Erdös Number Project” 2016). Social networks are a rich source of knowledge, but the data format does not fit easily within traditional data collection paradigms. Traditionally, data collection involves a set of units of the same, or at least similar, kind, on which observations are made. The storage of traditional data is simple and organized: rows contain variable values collected from units. These units can be people, plants, animals, stocks, objects, fields, and anything else under the sun, but one social network consits of many units, yet on the whole is just one observation. When observing a social network, one observes the possibly very numerous actors (also referred to as vertices or nodes) and the relationships (also referred to as edges or ties) between those actors. One can also collect information on the nodes and the edges separately, such as the age or gender of people and the length of their relationship or how strong it is in a friendship network. Thus, information on the entire network is more difficult to store than traditional data with which statisticians usually work.

This apparent difficulty has not stopped researchers in many different fields from studying social and other types of networks. Sociologists work with human relationship networks of all kinds imaginable, biologists work with protein-protein interaction networks, neurologists use fMRI scans to study biologic neural networks, and the list goes on. These disciplines worked separately for many years, each developing their own measures, softwares, and theories about the fundamental properties of networks. And although statisticians were comparatively late to the party, many statistical models exist for network analysis. Beginning with the classic Erdös-Rényi random graph model and varying in structure, complexity, and application to include longitudinal network data, such as continuous time markov chain models (Goldenberg et al. 2010). The many varying models that exist just for social network analysis are impressive, but I focus my research on one type of continuous time markov chain (CTMC) models, called Stochastic Actor-Oriented Models (SAOMs). A full introduction to the various models that exist for social network analysis is presented in Section 3.3, and a full introduction to the structure and theory of SAOMs is presented in Section 1.2.

1.1 Statistical Models for Social Networks

The literature on statistical models for networks is extensive. In their thorough “Survey of Statistical Models”, Goldenberg et al. (2010) separate these models into two primary classes: static and dynamic. I discuss the several types of models as they relate to stochastic, actor-oriented models, the models of my primary focus, in each of these two categories after a brief introductory section on general network terminology and notation.

1.1.1 Basic Network Terminology and Notation

Formally, a network is defined by a collection of nodes, also referred to as vertices or actors, and the set of ties, also referred to as edges or relationships, between them. Let x denote a network. The network’s collection of nodes, its nodeset, is written $\mathcal{N}$, and its collection of edges, its edgeset, is written. $\mathcal{E}$. Typically, the nodes are numbered so that $\mathcal{N} = \{1, 2, \ldots, n\}$, where n is the total number of nodes in the network. The edgeset is usually described as a set of pairs, written as x_{ij} or $i \rightsquigarrow j$ or (i, j), where $i \neq j \in \mathcal{N}$. In an undirected network, the ordering of i and j does not matter: there is no parent-child relationship, to use a term from graph theory, just a connection of some kind. In a directed graph, however, the order does matter: the tie x_{ij} is not equivalent to the tie x_{ji}. In a simple, undirected graph, the number of possible edges is $\binom{n}{2}$, while in simple, directed graphs it is $n(n-1)$, assuming no self-loops (also called self-ties or simply loops) and only allowing for at most one edge between any two nodes.

In statistical network analysis, an observed network is written as x, while X denotes an unobserved network being treated as a random variable. I assume binary network ties throughout: if the edge between nodes i and j is present, $x_{ij} = 1$, whereas $x_{ij} = 0$ if the edge is not present. If x is undirected, then $x_{ij} = x_{ji} \forall i \neq j \in \mathcal{N}$. If x is directed, then x_{ij} may equal x_{ji}, but this is not required and should not be assumed. Note that the definition of binary edge variables makes the assumption that edges are unweighted and that their cannot be more than one edge between two nodes. It is possible for networks to have weighted edges or multiple ties between nodes, but the models I discuss here, including the stochastic actor-oriented models that are my primary focus, are all for unweighted networks.

A network x can also be expressed as an $n \times n$ matrix of 0s and 1s called the adjacency matrix, denoted A. The ij^{th} entry of this matrix, a_{ij} is 1 if there is an edge between nodes i and j and 0 otherwise. The diagonal entries of this matrix, a_{ii} are structurally 0, as self-ties or self-loops are not allowed as mentioned above.

1.1.2 Static Network Models

The Erdös-Rényi random graph model is widely regarded as the first random graph model (Goldenberg et al. 2010). This model, first introduced in Erdös and Rényi (1959), describes random, undirected networks. Edges x_{ij} are selected at random from all possible edges. The parameter in this model is p, the probability that an edge exists between any two nodes in the network. The number of edges in the network, $e = \sum_{i<j} x_{ij}$, has likelihood

$$f(e|p,n) = p^e(1-p)^{\binom{n}{2}-e}.$$

The properties and asymptotic behavior of this network model are well-established (Goldenberg et al. 2010). Nodes in networks generated using this model all have about the same degree, or number of incident edges, which, in practice, is a very unrealistic property for a network to have. As such, many other models have been devised over the years as a way to better capture the network creation process underlying real-world networks.

In order to better model real-world networks, the exponential random graph family of models (ERGMs) was developed. These are also referred to as p^* models after the first use of the exponential family form in the p_1 model for directed networks of Holland and Leinhardt (1981). This class of models uses structural properties of the network as sufficient statistics in the likelihood. The properties used are different for directed and undirected graphs. Some statistics used for directed networks are the outdegree of the nodes, $x_{i+} = \sum_{j=\neq i} x_{ij}$, the indegree of the nodes, $x_{+i} = \sum_{j\neq i} x_{ji}$, and the number of reciprocal ties of the nodes, $x_{i,recip} = \sum_{j\neq i} x_{ij}x_{ji}$. For undirected networks, however, structures that are considered are the number of triangles in the network, $T(x) = \sum_{i\neq j\neq h} x_{ij}x_{ih}x_{jh}$, or the number of k-stars, $S_k(x) = \sum_i \binom{x_{i+}}{k}$, where $k = 2$ is most commonly chosen. The likelihood for ERGMs is written in terms of the whole network, x. The general form of the likelihood for x is

$$f(x|\boldsymbol{\beta}) = \frac{1}{\psi(\boldsymbol{\beta})} \exp\left(\sum_k \beta_k s_k(x)\right),$$

where β_k are parameters corresponding to K sufficient statistics chosen by the researcher and $\psi(\beta)$ is the normalizing constant. A problem with this model arises when one considers the nested nature of the sufficient statistics. For example, an edge can be contained in a 2-star, which can be contained in a triangle. So, the sufficent statistics can be dependent. Despite this flaw, this type of ERGM has been studied extensively, and many methods for parameter estimation exist, for example in the `R` packages `statnet` and `sna` (R Core Team 2016; Handcock et al. 2008; Butts 2014).

Other models are extensions of the Erdös-Rényi (ER) random graph model, including the preferential attachment model or the small world model, which are among the first network models that consider a network *formation* process over time. Eventually, network models expanded to include dynamic models, which consider changing network states in time.

1.1.3 Dynamic Network Models

Dynamic networks are extremely important because of how realistic they are. Social networks do not form spontaneously: they evolve over time. Ties can be added and deleted, and new nodes can join the network. Modeling the process of network changes over time is more complex but ultimately more useful if done correctly. The work on dynamic network models began with fairly straightforward random graph models that are quasi-dynamic extensions of the classic Erdös-Rényi model.

A model is quasi-dynamic if it models a *static* network via an *underlying* dynamic process. The first is the preferential attachment model of Barabási and Albert (1999). Given n_0 nodes to start, at each time point t a new node is added with $n_t \leq n_0$ ties to the nodels already in the network. The n_t new ties are assigned proportionally based on the degree of each existing node. It is quasi-dynamic because it is usually used to model one scale-free network oberervation. The preferential attachment model is also referred to as the "rich-get-richer" model because it results in a network where there are a few nodes with very high degree.

Another quasi-dynamic model is the small-world model of Watts and Strogatz (1998). Given n nodes to start, each with k edges that form a ring lattice (nodes layed out in a circle and connected to their k closest neigbors), edges are randomly "rewired" with probability p. This results in networks with the small-world property: let L be the average distance between any two nodes in the graph, and if the graph has the small-world property, $L \propto \log(n)$ as n increases (Watts and Strogatz 1998).

Truly dynamic models consider the same network observed at multiple points in time. To indicate a dynamic network, we write $x(t)$ instead of x for the network observation at time t. Dynamic network models can be in discrete or continuous time.

One such model in discrete time is an extension of the ERGM family. It models the transition probability, the probability of moving from the current network $x(t-1)$ to a potential future network, $x(t)$, that differs from $x(t-1)$ by one tie. The form of this probability is similar to the likelihood of the static ERGM model:

$$Pr(x(t)|x(t-1)) = \frac{1}{\psi(\boldsymbol{\beta})} \exp\left\{\sum_k \beta_k s_k(x(t), x(t-1))\right\},$$

where the s_k for $k = 1, \ldots, K$, are structural network statistics, similar to, but not the same as, the network statistics defined for static ERGMs. Some examples of statistics used are, the density of edges of the network, $s_1(x(t), x(t-1))$, or the stability of the network between time $t-1$ and time t. The density of a network is a ratio of the number of edges to the number of nodes in the network at the next time point, $s_1(x(t), x(t-1)) = \frac{1}{n-1}\sum_{i \neq j} x_{ij}(t)$. The stability is a measure of how many changes were made in the network between two time points, relative to the number of nodes, $s_2(x(t), x(t-1)) = \frac{1}{n-1}\sum_{i\neq j}(x_{ij}(t)x_{ij}(t-1) + (1 - x_{ij}(t))(1 - x_{ij}(t-1)))$. The likelihood of the entire network for all its states in *discrete* time is the joint probability of each transition step:

$$Pr(x(1), x(2), \ldots x(T)) = \prod_{t=2}^{T} Pr(x(t)|x(t-1)).$$

The family of dynamic network models in continuous time, of which stochastic, actor-oriented models are a member, are called continuous time Markov Chain (CTMC) models. These models are founded in the theory of continuous time Markov processes. Let $\{X(t), |t \in \mathcal{T}\}$ be a stochastic process in a continuous time interval $\mathcal{T}$ and finite state space $\mathcal{X}$. For any two timepoints $t_a < t_b \in \mathcal{T}$, the future state of the network, $X(t_b)$, depends only on the current state of the network, $X(t_a)$, and not any other previous network state. This is the Markov property, which for CTMCs is written as:

$$Pr(X(t_b) = \tilde{x}|X(t) = x(t), \quad \forall t \leq t_a) = Pr(X(t_b) = \tilde{x}|X(t_a) = x(t_a))$$

where $\tilde{x}$ is a potential future state in $\mathcal{X}$ and $x(t_a)$ is the present, observed state of the network. Assuming this probability relies only on the length of time that passes, $t_b - t_a$, then $X(t)$ has a stationary transition distribution. Then, the transition matrix for the process $X(t)$ is

$$Pr(t_b - t_a) \equiv \Big[Pr(X(t_b) = \tilde{x}|X(t_a) = x(t_a))\Big]_{x,\tilde{x}\in\mathcal{X}}.$$

Model	$\mathbf{q_{ij}(x)} =$	**Brief Description**
Independent arc	$\lambda_{x_{ij}}$	Edges are independent and have equal probability of changing from 0 to 1 and from 1 to 0
Reciprocity	$\lambda_{x_{ij}} + \mu_{x_{ij}} x_{ji}$	Rate of change depends on the presence of reciprocal edge.
Popularity	$\lambda_{x_{ij}} + \pi_{x_{ij}} x_{+j}$	Rate of change is dependent on the indegree of the child node, j
Expansiveness	$\lambda_{x_{ij}} + \pi_{x_{ij}} x_{i+}$	Rate of change is dependent on the outdegree of the parent node, i

Table 1: Some propensity functions to describe the network dynamics in CTMC models.

Write $t_b - t_a = t'$. Then, thanks to the stationarity of $X(t)$, the transition matrix of $X(t)$, $Pr(t')$ is equal to the matrix exponenial $\exp(t'\mathbf{Q})$, where $\mathbf{Q}$ is called the *intensity matrix* in the CTMC literature. The elements of this matrix will be defined in greater detail later, but one should note that the rows of $\mathbf{Q}$ are constructed to always sum to 0, and that each element also determines the probability of changing from one state to the other as a function of time.

For network modelling with CTMCs, the state space $\mathcal{X}$ is the set of all $2^{n(n-1)}$ possible networks with n nodes and directed, binary edges. Let x denote the current state of the network. From this network, there are $n(n-1)$ possible networks that x could become by changing just one edge variable, x_{ij} to its opposite value, $1 - x_{ij}$. Then, let $q_{ij}(x)$ be the propensity for for x_{ij} to become $1 - x_{ij}$ given x. This function $q_{ij}(x)$ "completely specifies the dynamics of the network model" (Goldenberg et al. 2010, 48). There are many forms in this family of models, which differ only in their choice of $q_{ij}(x)$. A list of some fairly simple choices for $q_{ij}(x)$ is provided in Table 1.

Additional definitions of $q_{ij}(x)$ are more complicated. These next set of models rely on two different underlying mechanisms: one that determines which node is given the opportunity to change and one that determines the propensity of change. First, I consider the subset of models with edge-oriented dynamics. Let $x(i \rightsquigarrow j)$ denote the network that differs from x by just one node, x_{ij}, which takes on the value $1 - x_{ij}$ in $x(i \rightsquigarrow j)$. Then, write the probability that node x_{ij} changes to $1 - x_{ij}$ as

$$p_{ij}(x) = \frac{\exp(f(\boldsymbol{\beta}, x(i \rightsquigarrow j)))}{\exp(f(\boldsymbol{\beta}, x)) + \exp(f(\boldsymbol{\beta}, x(i \rightsquigarrow j)))},$$

where $f(\boldsymbol{\beta}, x) = \sum_k \beta_k s_k(x)$ is called the potential or objective function (Goldenberg et al. 2010). The β_k are parameter values associated with the network statistics that are also used in ERGMs. For more definitions of the possible $s_k(x)$, see Table 2. The opportunity for change in this model is controlled by a constant rate parameter, α. The wait time between a change of any edge in the network is exponentially distributed with parameter α. So, the function $q_{ij}(x)$ is defined as $\alpha p_{ij}(x)$.

The next subset of models rely on node-oriented dynamics. These are very similar to the edge-oriented dynamics but the rate parameter and propensity to change are defined with respect to the nodes instead of the edges. Now, each node has its own rate at which it gets an opportunity for change, α_i. Additionally, the objective function is defined for each node, $f_i(\boldsymbol{\beta}, x) = \sum_k \beta_k s_{ik}(x)$. This changes the definition of the statistics used slightly, from global statistics to local statistics with ego node i. Thus, the propensity function becomes $q_{ij}(x) = \alpha_i p_{ij}(x)$.

Finally, stochastic, actor-oriented models belong to the set of CTMC models that combine edge and node dynamics so that the propensity function becomes a hybrid of the prior two: $q_{ij}(x) = \alpha p_{ij}(x)$ where α is a constant rate of edge change, while p_{ij} is the propensity to change edge x_{ij} using the node-oriented objective function $f_i(\boldsymbol{\beta}, x)$. These are described in greater detail in Section 1.2.

1.2 Stochastic Actor-Oriented Models for Longitudinal Social Networks.

A Stochastic Actor-Oriented Model (SAOM) is a model that is changing in time in order to accomodate for observations from the same network made at different points in time and that allows for changes in network structure due to actor-level covariates. These two properties are crucial to understanding networks as they exist naturally. Most social networks, even holding constant the set of actors over time, are ever-changing as relationships decay or grow, and most actors (or nodes) in social networks have inherent properties that could affect how they change their place within the network.

1.2.1 Terminology, Notation, and Mathematical Definition of SAOMs

A longitudinal network is a network consisting of the same set of n nodes that is changing over time, and is observed at M discrete time points, $t_1, \ldots, t_M$. We denote these network observations $x(t_1), \ldots, x(t_M)$. The SAOM assumes that this longitudinal network is embedded within a continuous time markov process (CTMP), call it $X(T)$. This process is almost entirely unobserved. The process $X(T)$ theoretically exists outside of the range of observation, but for simplicity of notation, assume that the beginning of the process, $X(0)$ is equivalent to the first observation $x(t_1)$, while the end of the process $X(\infty)$ is equivalent to the last observation $x(t_M)$. The observations $x(t_1), \ldots, x(t_M)$ are observed states of the process, $x(t_1) \equiv X(0), x(t_2) \equiv X(T_{t_2}), \ldots, x(t_{M-1}) \equiv X(T_{t_{M-1}}), x(t_M) \equiv X(\infty)$, but the time points t_m and T_{t_m} for $m = 2, \ldots M-1$ are not equivalent. The process $X(T)$ is a series of single tie changes, in which one actor at a time is given the opportunity to add or remove one outgoing tie. These opportunities for change can arise at a different rate for each actor, and the overall rate of change, the distribution of the waiting times that *any* actor will be given the opportunity to change is a function of all actors' rates. Additionally, once an actor is given the chance to change a tie, it tries to maximize a sort of utility function based on the current and potential future states of the network. These functions are described in detail in subsections 1.2.2 and 1.2.3.

1.2.2 The Rate Function

For the network x and each actor i in the network, the rate function dictates how often the actor i gets to change its ties, x_{ij}, to other nodes $j \neq i$ in the network. This rate can depend on the time period of observation, some actor-level covariates or some actor-level network statistics. The rate function can be unique to each actor, and is denoted λ_i. The most general form is $\lambda_i(\alpha, \rho, x, m)$, where α is a simple rate of change parameter, ρ is a parameter or a vector of parameters corresponding to one or more covariates, $x \in \mathcal{X}$ is the current state of the network, and m indicates the time point of the current network observation, t_m. The rate function determines how quickly actor i gets an opportunity to change one of its ties, x_{ij} in the time period $t_m \leq T < t_{m+1}$. We assume that the actors i are conditionally independent given their current ties, $x_{i1}, \ldots, x_{in}$. This assumption leads to the rate function for the whole network:

$$\lambda(\alpha, \rho, x, m) = \sum_i \lambda_i(\alpha, \rho, x, m).$$

In order to achieve the memorylessness property of a Markov process, for any time point, T, where $t_m \leq T < t_{m+1}$, the waiting time to the next change opportunity by actor i is exponentially distributed with expected value $(\lambda_i(\alpha, \rho, x, m))^{-1}$. Thus, the waiting time to the next change opportunity by *any* actor in the network is also exponentially distributed with. mean $(\lambda(\alpha, \rho, x, m))^{-1}$, where

$$\lambda(\alpha, \rho, x, m) = \sum_i \lambda_i(\alpha, \rho, x, m)$$

.

There are many possibilities for the rate function, λ_i. The simplest is that it is constant over all actors and all unobserved timepoints between observations $x(t_m)$ and $x(t_{m+1})$, $\lambda_i(\alpha, \rho, x, m)) = \alpha_m$. The rate function can also depend on covariate values, call them $\mathbf{z}_i(t_m)$, of the actors, or structural network elements such as outdegree, or both. For instance, assume $\lambda_i(\alpha, \rho, x, m)) = \lambda_{i1}\lambda_{i2}\lambda_{i3}$, where λ_{i1} is constant over all actors within a time period (t_m, t_{m+1}), λ_{i2} depends on the actor covariates, and λ_{i3} depends on a structural network property for node i. λ_{i1} might be written as α_m. λ_{i2} might be written as

$$\lambda_{i2} = \exp\left(\sum_h \rho_h z_{ih}(t_m)\right),$$

where there are $h = 1, \ldots, H$ actor covariates of interest, each with their own parameter ρ_h. λ_{i3} can be written as a function of the outdegree of node i, denoted x_{i+} with its own parameter α_{H+1}, so that, for example,

$$\lambda_{i3} = \frac{x_{i+}}{n-1}\exp(\alpha_{H+1}) + \left(1 - \frac{x_{i+}}{n-1}\right)\exp(-\alpha_{H+1}).$$

When $H = 0$, this form of λ_{i3} is equivalent to the model proposed by Wasserman (1980), which is one of the first models proposed for modeling dynamic networks as continuous-time Markov processes (T. A. Snijders 2001). Once a change occurs, according to the rate of change for the whole network, $\lambda(\cdot)$, the probability that actor i is the node with the power to change a tie is

$$\frac{\lambda_i(\alpha, \rho, x, m))}{\sum_i \lambda_i(\alpha, \rho, x, m))}$$

.

1.2.3 The Objective Function

Thanks to the conditional dependence assumptions in the model, we can consider the objective function for each node separately, since only one tie from one node is changing at a time. The objective function is written as

$$f_i(\boldsymbol{\beta}, x) = \sum_k \beta_k s_{ik}(x, \mathbf{Z}),$$

for $x \in \mathcal{X}$ and $\mathbf{Z}$ the matrix of covariates. The vector $\boldsymbol{\beta}$ are the parameters of the model with corresponding network and covariate statistics, $s_{ik}(x, \mathbf{Z})$, for $k = 1, \ldots, K$. Given the focal or ego node, i, there are n possible steps for the actor i to take: either one of all current ties $x_{ij} = 1$ will be destroyed, a new tie will be created, or no change will occur.

The parameters, $\boldsymbol{\beta}$, are attached to various actor-level network statistics, $s_{ik}(x)$. There are always at least two parameters, β_1 for the outdegree of a node, and β_2 for the number of reciprocal ties held by a node (T. A. Snijders 2001, 371). There are many possible parameters β to add to the model. They can be split up into two groups: first, the structural effects, which only depend on the structure of the network. The inclusion of these effects has origin in the ERGMs discussed in Section 1.1.2. These effects are written in terms of the edge variables x_{ij}, for $i \neq j$. The second set of effects are the actor-level or covariate effects. These effects also depend on the structure of the network. They are written in terms of x_{ij} but also in terms of the covariates, $\mathbf{Z}$. A table of some possible structural and covariate effects is given in 2.

When node i is given the chance to change a node, we assume that they wish to maximize the value of their objective function $f_i(\boldsymbol{\beta}, x)$ plus a random element, $U_i(x)$, where the $U_i(x)$ are from "the type 1 extreme value distribution (or Gumbel distribution) with mean 0 and scale parameter 1" (T. A. Snijders 2001, 368). This distribution, which is also known as the log-Weibull distribution, has probability distribution function, using μ for the mean parameter and σ for the scale parameter, of

$$f(u|\mu, \sigma) = \frac{1}{\sigma}\exp\left\{-\left(\frac{x-\mu}{\sigma} + e^{-\frac{x-\mu}{\sigma}}\right)\right\}.$$

Using this distribution is convenient because it allows the probablity the actor i chooses to change its tie to actor j in terms of the objective function alone. Let $p_{ij}(\boldsymbol{\beta}, x)$ be this probability. Next, write the network x in its potential future state, where the tie x_{ij} has changed to $1 - x_{ij}$, as $x(i \rightsquigarrow j)$. Then, the probility that the tie x_{ij} changes is

Table 2: Some of the possible effects to be included in the stochastic actor-oriented models in RSiena. There are many more possible effects, but we only consider a select few here. For a complete list, see the RSiena manual (Ripley et al 2016).

Structural Effects	
outdegree	$s_{i1}(x) = \sum_j x_{ij}$
reciprocity	$s_{i2}(x) = \sum_j x_{ij}x_{ji}$
transitive triplets	$s_{i3}(x) = \sum_{j,h} x_{ij}x_{jh}x_{ih}$
Covariate Effects	
covariate-alter	$s_{i4}(x) = \sum_j x_{ij}z_j$
covariate-ego	$s_{i5}(x) = z_i \sum_j x_{ij}$
same covariate	$s_{i6}(x) = \sum_j x_{ij}\mathbb{I}(z_i = z_j)$
jumping transitive triplets	$s_{i7}(x) = \sum_{j \neq h} x_{ij}x_{ih}x_{hj}\mathbb{I}(z_i = z_h \neq z_j)$

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

1.2.4 A SAOM as a CTMC

In order to fit this model definition back into the original context of the CTMC described in Section 1.1.3, it must be written in terms of its intensity matrix, $\mathbf{Q}$. This matrix describes the rate of change between states of the process. For networks, there are a very large number of possible states, $2^{n(n-1)}$, so the intensity matrix is a square matrix of that dimension. But, thanks to the property of SAOMs that the states are allowed to change only one tie at a time, there are only n possible states given the current state, $n-1$ of which are uniquely determined by the node i that is given the opportunity to change. 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 represent the possible states that are one edge different from a given state, and the additional non-zero entry is for the state to remain the same. All other entries in a row are zero because those column states cannot be reached from the row state by just one change as dictated by the SAOM. 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 Q is:

$$q(x^b, x^c) = \begin{cases} q_{ij}(\alpha, \rho, \boldsymbol{\beta}, x^b) = \lambda_i(\alpha, \rho, x^b, m)p_{ij}(\boldsymbol{\beta}, x^b) & \text{if } x^c \in \{x^b(i \rightsquigarrow j) | \text{ any } i \neq j \in \mathcal{N}\} \\ 0 & \text{if } x^c \text{ differs from } x^b \text{ by more than 1 tie} \in \mathcal{N} \\ -\sum_{i \neq j} q_{ij}(\alpha, \rho, \boldsymbol{\beta}, x^b) & \text{if } x^b = x^c \end{cases}$$

Thus, the rate of change between any two states 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.[1] Furthermore, the theory of continuous time Markov chains gives that the matrix of transition probabilities between observation times t_{m-1} and t_m is dependent only on the difference between timepoints, $t_m - t_{m-1}$. Following the same definition for transition probabilities in Section 1.1.3, the matrix of transition probabilities is

$$e^{(t_m - t_{m-1})\mathbf{Q}},$$

where $\mathbf{Q}$ is the matrix defined above and e^X for a real or complex square matrix X is equal to $\sum_{k=0}^{\infty} \frac{1}{k!}X^k$.

1.2.5 Model Fitting for SAOMs

Stochastic actor-oriented models are "too complicated for the calculation of likelihoods or estimators in closed form, but they represent stochastic processes which can be easily simulated" (T. a. B. Snijders, Koskinen, and

[1] Just to be clear, the change is from x_{ij}^b to $x_{ij}^c = 1 - x_{ij}^b$.

Schweinberger 2010, 568). Thus, calculation of the method of moments estimates of parameters in SAOMs is done via Markov Chain Monte Carlo (MCMC) approximation. The algorithm presented here were first presented in T. A. Snijders (2001).

The vector of parameters that need to be estimated is

$$\boldsymbol{\theta} = (\alpha_2, \dots, \alpha_M, \beta_1, \dots, \beta_K).$$

The length of $\boldsymbol{\theta}$ is $L = M - 1 + K$, where M is the number of network observations and K is the number of parameters included in the objective function, $f_i(\boldsymbol{\beta}, x)$. The corresponding sufficient statistics for estimating the rate parameters, α_m, are the number of edges that have changed between $x(t_{m-1})$ and x_{t_m}, $C_2, \dots, C_M$, where $C_m = \sum_{i \neq j} |x_{ij}(t_m) - x_{ij}(t_{m-1})|$. Let The corresponding sufficient statistics for estimating the rate parameters, β_k, are the corresponding values of the node-level statistics, some of which are seen in Table 2, summed over all nodes for each network observation, $S_{2k}, \dots, S_{Mk}$ where $S_{mk} = \sum_i s_{ik}(x(t_m))$ for $m = 2, \dots, M$. Denote the whole vector of sufficient statistics as $\mathbf{S} = \big(C_2, \dots, C_M, S_{2k}, \dots, S_{Mk}\big)$.

The method of moments estimator of $\boldsymbol{\theta}$ is the solution to $E_{\boldsymbol{\theta}}[\mathbf{S}] = \mathbf{s}$ where $\mathbf{s}$ are the observed values of $\mathbf{S}$ in $x(t_2), \dots, x(t_M)$. Following T. A. Snijders (2001), the estimate $\hat{\boldsymbol{\theta}}$ can be separated into the vectors $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\beta}}$, which are the solutions to the system of equations

$$\begin{aligned} E_{\boldsymbol{\alpha}}[C_m | x(t_m)] &= c_m \\ \sum_{m=1}^{M-1} E_{\boldsymbol{\beta}}[S_{mk} | x(t_m)] &= \sum_{m=1}^{M-1} s_{mk}. \end{aligned}$$

The solutions to these moment equations are, unless the model is extremely simple, not able to be calculated explicitly. Because of this, random simulation of networks with the desired distribution can be used in Markov Chain Monte Carlo simulation of the moment estimates. Given a starting value $\boldsymbol{\theta}^{(0)}$, the updating step of the simulation is, for iterations $b = 0, \dots, B$:

$$\boldsymbol{\theta}^{(b+1)} = \boldsymbol{\theta}^{(b)} + a_b D_0^{-1} (\mathbf{S}_b - \mathbf{s})$$

where $\mathbf{S}_b$ is drawn from the distribution of the model under $\boldsymbol{\theta} = \boldsymbol{\theta}^{(b)}$, $\mathbf{s}$ are the observed statistics, D_0 is a positive diagonal matrix, usually the identity, and a_b is called the *gain sequence* and is some sequence of positive values that approach 0 as $b \to \infty$ at about the same rate as b^{-r} for some $0.5 < r < 1$. The method of moments estimator, $\tilde{\boldsymbol{\theta}}$ is then an average of the B iterations, $\tilde{\boldsymbol{\theta}} = \frac{1}{B} \sum_{b=1}^{B} \boldsymbol{\theta}^{(b)}$. It must be the average in order to obtain optimal convergence (T. A. Snijders 2001).

This algorithm is implemented in the R software package `RSiena` for computation of parameter estimates for various SAOMs (Ripley, Boitmanis, and Snijders 2013). This is the software I use for model fitting in 2 and 3.

1.2.5.1 Model Selection and Testing for SAOMs

A likelihood ratio test was also developed in T. a. B. Snijders, Koskinen, and Schweinberger (2010), but it has yet to be implemented in the software RSIENA for parameter estimation of SAOMs. Tests of the elements of $\boldsymbol{\beta}$ are, however, availabile in RSIENA. Both t-tests and Wald-type tests are implemented. A goodness-of-fit test is also implemented, but it only assesses the fit of a model with respect to the "auxiliary statistics of networks [...] that are not explicitly fit by a particular effect" (Ripley et al. 2016, 53). It is this lack of goodness-of-fit testing that led my research down the path of applying visual inference principles and protocols to hypothesis testing for SAOMs.

1.3 Network Visualization

Network visualization, also called network mapping, is a very well-established subfield of network analysis. As networks have such a non-traditional data structure, visualization has always been of the utmost importance to understanding the structre of a network.

1.3.1 Layout Algorithms

The key difficulty with network visualization that does not arise with most other types of data visualization is the lack of a well-defined axis. This is not something one has to think hard about for most data visualizations. If the variables are numerical, histograms, scatterplots, or time series plots are straightforward to construct: one variable on the x-axis, another on the y-axis in 2D Euclidean space. If the variables are categorical, bar charts and mosaic plots can be constructed in this same space. If the data are spatial, there is a well-defined space In pretty much any case, the location and labels of the data and axes can be defined with very little struggle. With network data, however, this is a more difficult problem.

Network visualizations are made by representing nodes with points in 2D Euclidean space, just like one would with any other data set, and then by representing edges by connecting the points with lines if there is an edge between the two nodes. But, because there is no natural placement of the points, a random placement is used, then adjusted iteratively via a layout algorithm, of which there are many kinds. I will focus on the 2D layout algorithms only because I work later with the `ggplot2` package to visualize networks, and this package only has 2D drawing capabilities.

Some layout algorithms were designed to mimic physical systems, drawing the graphs based on the "forces" connecting them. The network's edges act as springs pushing and pulling the nodes in 2D space. Some force-directed layout algorithms are:

- Kamada-Kawai: first introduced in Kamada and Kawai (1989). Has "symmetric drawings, a relatively small number of edge crossings, and almost congruent drawings of isomorphic graphs" (Kamada and Kawai 1989, 15)
- Fruchterman-Reingold: first introduced in Fruchterman and Reingold (1991). Primary advantage is speed over Kamada-Kawai (Fruchterman and Reingold 1991, 1161).
- Spring embedding: first introduced in Eades (1984). Other force-directed layouts are refinements of this original algorithm.
- Target diagram: nodes placed in concentric circles with hig-centrality nodes placed nearer to the center of the circle. First introduced in Brandes, Kenis, and Wagner (2003).

Other layout algorithms depend on the mathematical properties of the network's adjacency matrix or some other function or propterty of the network. Algorithms of this kind are:

- Eigen: node placement is based on the eigenvalues of the adjacency matrix
- Hall: node placement is based on the last two eigenvectors of the Laplacian of the adjacency matrix
- Multidimensional Scaling (MDS): node placement is based on metric multidimensional scaling of a given distance matrix. Distance metric can vary.
- Principal Coordinates: node placement is based on the eigenvalues of a given covariance or correlation matrix.

Some layout algorithms only exists for certain types of networks:

- Reingold-Tilford: for trees
- Sugiyama: for layerd directed acyclic graphs

Finally, some layout methods just place the nodes randomly or in a simple ordering:

- Random: places nodes randomly according to some distribution, usually uniform or some Gaussian distribution.
- Grid: places nodes on a 2D grid
- Circle: places nodes in a circle in numerical order by ID number

These layout algorithms have been provided in several R packages for network visualization. Another important aspect of network visualization is the addition of varirable information into the properties of the points and segments of the network visualization. For example, the size of the point, the width of the line, and the color of these these can all be mapped to the points and segments making up the network visualization. This is discussed further in Section 1.3.3.

The visualization methods outlined above are all for static networks. There has been little work done on how to visualize dynamic networks. The only R package to my knowledge that attempts dynamic network visualization is `ndtv` by Bender-deMoll (2016). I will use this package to help visualize the continuous time Markov chain underlying the SAOM dynamics. The goal is to better understand the network changes, how they change, and better see the differences, but it may turn out to be less effective than looking at side-by-side comparisons of two network observations.

1.3.2 R Packages

There is a multitude of R packages that exist for network analysis, and many, if not most, of them contain some sort of built-in functionality for visualizing networks. The most popular of these is probably the `igraph` package by Csardi and Nepusz (2006). This package is extensive, and contains much more than methods for network visualization. It contains tools for both 2D and 3D visualization of networks. The 2D layouts it contains are `random`, `circle`, `star`, `grid`, `graphopt`, `bipartite`, `fruchterman_reingold`,`kamada_kawai`, `mds`, `grid_fruchterman_reingold`, `lgl`, `reingold_tilford`, `reingold_tilford_circular`, and `sugiyama`.

Another popular package for network analysis is `sna` by Butts (2014). This package was designed specifically for social network analysis (sna), so it also contains much more capabilities for network analysis in addition to visualization. Like `igraph`, `sna` contains both 2D and 3D layout methods. The 2D layout algorithms available in `sna` are `circle`, `circrand`, `eigen`, `fruchtermanreingold`, `geodist`, `hall`, `kamadakawai`, , `mds`, `princoord`, `random`, `rmds`, `segeo`, `seham`, `spring`, `springrepulse`, and `target`.

Research into possible layout algorithms is important, but it ignores some of the things that statisticians usually consider when visualizing data. For instance, since the location of points in 2D space contains no information about the data, how else should this information be visualized? As an example, consider a friendship network of students at a university. Representing this network as simple points and lines leaves a lot of information out. Some information that could be incorporated includes the students' majors, year in school, and whether the students have ties through their classes or their extracurricular activities. In the network visualization, this information can be mapped to color of point, shape of point, and linetype, respectively. Adding this aesthetic information helps to make up for the loss of two dimensions of visual perception and to bring the network visualization into the world of statistical graphics.

1.3.3 The Importance of the `ggplot2` Package

The `gg` of `ggplot2` is for the "grammar of graphics". The grammar of graphics is a well-defined theory for creating statistical graphics described in Wilkinson (1999) and Wickham (2009). In the grammar, a plot has layers, each of which has four distinct pieces: the data and aesthetic mapping, a statistical transformation, a geometric object, and a position adjustment. The aesthetic mapping takes the data and *maps* the variables in the data frame to visual features. Some of these features are horizontal and vertical placement in the plane, size of the geometric object and color of the geometric object. The statistical transformation dictates how to transform the data to the values that create the visual feature. Some `stats` are `identity` (no change in data), `bin`, and `smooth`. The geometric object or `geom` is the tool used to draw a plot layer. Some `geoms` are `point`, `line` and `bar`. Finally, the position adjustment is there to slightly change the position of the visual features in order to better view the data. This is typically only a probelm with discrete data, where overplotting can occur. Some position adjustments are `identity`, `jitter`, and `dodge`.

With the theory well defined and constructed, the `ggplot2` package allows for creation of rich, visually dense plots. The user can combine multiple aesthetic mappings to view four variables at once or view many data sets of similar scale at once. The widespread use of `ggplot2` and the many packages that have built upon

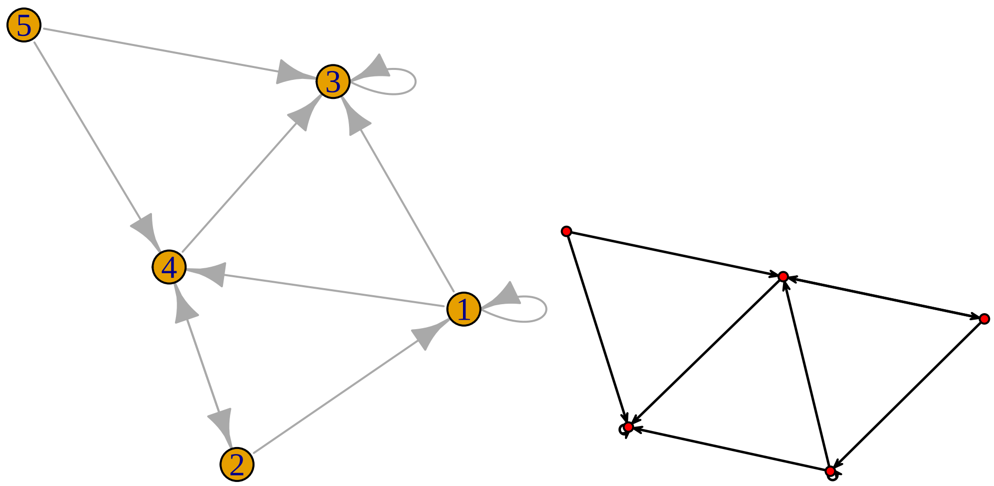

Figure 1: The same random network plotted with the default options in extttigraph (at left) and extttnetwork (at right.

ggplot2 to create visualizations above and beyond what it is capable of by itself make the ggplot2 package an ideal framework on which to build additional methods of network visualization in R.

First, the data structure required in ggplot2 is fairly simple: data frames. Some other network packages contain network data structures unique to them, like the igraph class of data in the igraph packages or the network class of data in the network package. These unique structures come with unqiue syntax that can make customizing visualizations tricky. Additionally, the default visualizations in these packages are not very pleasing to the eye, as is shown with the random graph examples from igraph and network in Figure 1. As I will discuss in 4, network visualization within the ggplot2 framework results in beautiful, easily customizable plots.

I will use network visualization in the ggplot2 framework to graphically explore the SAOMs. By using visual inference, I will learn about the importance of the many possible model parameters in SAOMs and about how they affect the visible structure of the network.

1.4 What is Visual Inference?

Viewing plots of data is an important part of exploratory data analysis (EDA) and of model diagnostics (MD). In EDA, plots guide the analyst to discovering relationships between variables in their data, while in MD, plots help the analyst determine if the model chosen is appropriate. In EDA, the analyst may notice that a covariate is strongly correlated with the dependent variable by drawing a scatterplot, leading the analyst to choose a simple linear model. But in MD, the analyst could later notice a pattern in the residuals plotted against the covariate, indicating that the variance of the dependent variable is not constant across changing values of the covariate. These steps of EDA and MD have become so engrained in statistical practice that they are taught in introductory statistics courses. But, how can we formalized this visual discovery process?

1.4.1 A Formal Definition and Construction

The idea of visual inference was first introduced in Buja et al. (2009). In this seminal work, the authors outline two protocol for visual tests of hypotheses, the "Rorschach" the "lineup". The former allows one "to measure a data analyst's tendency to overinterpret plots in which there is no or only spurious structure," while the latter has the viewer "identify the plot of the real data from among a set of decoys [. . .] under the veil of ignorance" (Buja et al. 2009, 4368–9).

They begin by formalizing the definition of the set of discoverable (i.e. visible) features of a plot as a set of test statistics, denoted $T^{(i)}(\mathbf{y})(i \in I)$. The value $\mathbf{y}$ is the data in the plot, and the set I is the hard-to-define set of all possible visual features one could discover in a plot. Then they consider a general null hypotheses scenario, H_0, from which the data could have arisen. Samples are then taken from this null model and the same plot is made for the samples as was made for the data. These plots are called "null plots" while the other is the "data plot". The idea is that if an "analyst" sees a feature in the data, and also in the null plots, then the data cannot be said to come from a different scenario than H_0.

Generating samples from H_0 is not trivial. The authors provide three types of sampling available for creating the null plots: conditional sampling given a minimally sufficient statistic, parametric bootstrap sampling, and Bayesian posterior predictive sampling. (Buja et al. 2009, 4367). Once the null plots are generated, they are presented to an analyst through the Rorschach and lineup protocols.

In the Rorschach protocal, the analyst looks at a series of plots and describes any features or structures that stand out to them. These plots will all be null plots, but the analyst should not know this. The protocol administrator should also not know whether or not the data plot is in the series of plots. Then, these results are examined by the researcher, who determines what tendency the analyst have to "over-interpret" plot structure.

In the lineup protocol, the analyst looks at M plots that are laid out in a grid. $M-1$ of these plots will be null plots, while one is the data plot. For $M = 20$, the probability of choosing the data plot from among the null plots is 0.05, providing us with an inferentially valid p-value of $\alpha = 0.05$. The lineup protocal has several special features. First, there is no need for pre-specification of the visual feature the analyst should identify. They can simple be asked to pick the most different or most special plot. Second, the analyst can self-administer the lineup once, thereby becoming a data point in their own experiment. Next, it is possible that 2 or more plots can be selected from among the M plots, as ranked data methods can be used for data analysis. Finally, the procedure can have as many repetitions as possible, as long as the analysts are independently selected and have not previously viewed the plot of the data. This can lead to extremely small p-values for inference, with the smallest possible being 0.05^K for K analysts, assuming all K selected the data plot from the lineup. Formally, the p-value of a lineup of size M evaluated by K analysts is

$$Pr(X \geq x) = 1 - Binom_{K,\frac{1}{M}}(x-1)$$

where X is the number of analysts who correctly identify the data plot, x the observed value X for an experiment, and $Binom_{K,\frac{1}{M}}(x)$ is the probability mass function of the binomial distribution with K trials and probability of success $\frac{1}{M}$ evaluated at the observed x. Type I error, the probability that a test rejects H_0 when it is true, is also formally defined as $Pr(X \geq x_\alpha)$, where x_α is the number of observers picking the data plot needed so that $P(X \geq x_\alpha | H_0)$ is less than or equal to the chosen value of α. The type II error, the probability that H_0 is not rejected when it is not true, is then $P(X < x_\alpha)$, where X and x_α are defined as above. Additionally, the power of the test given the true state, either when H_0 is true or when it is not, is the probability that the test rejects H_0. When H_0 is true, the power is $1 - Binom_{K,\frac{1}{M}}(x_\alpha - 1)$. If H_0 is not true, the power depends on the specific true state (alternative hypothesis) chosen (Majumder, Hofmann, and Cook 2013).

The type of plots shown in visual inference will vary based on the context of the research question and null hypothesis of interest. For example, scatterplots can be shown to test for independence of two variables or for clustering; histograms can be shown to test for distribution of a variable; time series plots can be shown to test for trends; residual plots can be shown to test for presence of structure the model misses; and smoothers can be shown to test for differences in trends between groups. All of these examples are discussed in detail in Buja et al. (2009).

Additional detail to consider is the importance of varying skillsets of analysts, and the effectiveness of each analyst at selecting the data lineup. Some analysts, especially when doing experimentation, will be more visually inclined, or more analytically inclined, and these individual differences can affect the success rate of an analyst, and the rate of identification may need to be modified to account for these differences.

1.4.2 Applications of Visual Inference

There have been two distinct areas of application of visual inference since Buja et al. (2009). The first is true application of the methodology, while the second is understanding the methodology via application of the protocols. In both applications usually rely on the Amazon Mechanical Turk service (Amazon 2010) or other similar services to show lineups to many participants from different backgrounds quickly.

In true applications, researchers have one or more alternative hypotheses and corresponding nulls on which they perform visual inference tests to show many participants of different backgrounds the lineups. One such paper, Loy, Follett, and Hofmann (2016), considers the visual inference tests for normality via lineups of Q-Q plots and compares these tests to traditional statistical normality tests. The authors found that visual inference used in this way is a more powerful test for normality than classical tests (Loy, Follett, and Hofmann 2016). In another direct application, Zhao et al. (2013) use visual inference to establish the existence of a structrue in the RNA sequence of soybean plants where different treatments and conditions alter the gene expression. Yet another application is that of Hofmann et al. (2012), in which the authors use visual inference to determine which view of a dataset to present so that the important data properties are communicated most accurately and efficiently.

The second type of application, understanding the methodology through application is the type that I pursue in 3. One such instance of this type of application is Chowdhury et al. (2014), in which visual inference is used to better understand problems that arise when viewing high dimension, low sample size data. A second application is that of Loy and Hofmann (2015) in which the authors use visual inference to determine hierarchical model misspecification. In both of these applications, visual inference is used to discover more about the models or structures under investigation. This is how I intend to use visual inference for SAOMs. By using the lineup protocol, I hope to learn more about the effects of parameter selection on SAOMs, and order to do this, I also need to have tools to visualize the networks simulated from SAOMs.

1.5 Summary

Stochastic actor-oriented models are a rich and interesting set of models because of the complicated nature of statistical network modeling and the variety in choice of parameters available to the researcher. In the next three chapters, my aim is to fully characterize the structure and function of these models. I will do this using the lineup protocol for visual inference, and the R package, `geomnet` that I created as a part of this work.

2 Stochastic Actor-Oriented Models for Longitudinal Network Data: Removing the Blindfold

Abstract: In this chapter, I will "remove the blindfold" from some aspects of modeling social networks with stochastic actor-oriented models. I will follow procedures similar to those in Wickham, Cook, and Hofmann (2015) in order to better understand the latent process of edge creation and destruction that characterizes a SAOM. I will also visualize, compare, and contrast several different networks simulated from a SAOM with a particular set of parameters and parameter values in order to better understand what the distribution of networks under SAOM looks like. Finally, I will compare large numbers of networks using tools like structural principal components analysis in order to determine how differences in model structure and parameter values affect the overall structure of a social network.

2.1 Introduction

Network analysis has earned much interest in fields suchs as sociology or computer science. Many methods of analysis for networks have been derived from researchers in these fields, and the interest in network analysis in these fields has grown exponentially throughout the last two decades. There are now entire journals dedicated to network analysis, including *Network Science* and *Social Networks*. In the field of statistics, however, not much attention has been given to this area. There are several possible reasons for this, the first being that statistical network analysis does not fit easily into the statistician's workflow and way of thinking.

First and foremost, when modelling, statisticians consider the population from which their data were generated. In some social network problems, this is easy, like in the students at a university example. In others, however, it might not be so straightforward, like when mapping the interconnectedness of scientific disciplines (Kolaczyk 2009). Next, statisticians consider the representativeness of their data. Usually, a random sample of the population has been carefully selected for analysis to be representative. But in network analysis, there may not be a well-designed, appropriate sampling procedure. Social media networks have become very popular and very large, so many researchers have interest in studyin them, but how can they be sampled? Additionally, how does one avoid biases in social network sampling, and can these biases be corrected for (Kolaczyk 2009). If one wants to summarize their network data, what statistics should be used? Measure such as like mean and variance are available and presented for most types of statistical, but there is no definition for a "mean" or a "variance" on a network. So, other statistics like average outdegree are often used. But ultimately, these statistics cannot describe the network structure as well as the mean and variance can describe the distribution of a random variable. As was shown in section 3.3 there are many models for network analysis, but once a network is modeled, how can we make predictions for what new networks will be or what new edges will form?

Furthermore, traditional statistical models have some common properties that network models might not always have. First, the models have to be well-grounded in measure and probability theory so that their behavior can be well-understood based on the fundamentals of these areas of study. This may be true in many network models, but they may also be analytically intractable. Second, the models have to be estimable from the data at hand. Again, this is possible in many of the network models, but some may only be estimable with advanced simulation methods that have only recently become available with respect to computing power. Next, a crucial element within statistics is the quantification of uncertainty in estimation. Since there is not a measure of "variance" on a network, how then can the estimation of network model parameters be qualified to include uncertainty? Finally, statisticians require methods to validate goodness-of-fit of their models. In network analysis, however, goodness-of-fit measures are nearly non-existent (Kolaczyk 2009).

The introduction to this chapter will also serve as a brief introduction to the structure of stochastic actor-oriented models (SAOMs) for longitudinal network data. As there is a greater exposition of these models in the literature review, I do not repeat that information here at this time.

2.2 Visualizing the Dynamic Network Process

The basis for SAOMs is a continuous-time Markov chain (CTMC) that remains entirely unobserved with the exception of the finite number of network observations, $T \geq 2$. The first part of this chapter will discuss a to-be-developed procedure to visualize the intermediary steps in the Markov chain that lead to the network transformation from one time point to another. This video (or interactive app ?) will clearly demonstrate the multilevel process the SAOMs are modelling. First, it will show that a node is selected and given the opportunity to change a tie, and then it will show how that node's objective function decides which tie to change, or to not change at all.

2.3 Characterizing a SAOM

My hope is that this demonstration will also raise some important questions in the minds of researchers working with these social network models and others. Primarily, I hope to bring question of the model *distribution* to the forefront. The process being modeled is random, and as such has a theoretical underlying distribution. What does that distribution look like? This is an open question in statistical network analysis. In statistics, the entity being modeled is assumed to have an underlying distribution according to any level complexity of model. Even in situations of analytical intractability, data can be simulated in some way or another in order to give the researcher an idea of what the distribution of the model looks like. These distributions can be viewed with histograms or barcharts, or with contour plots, heat maps, or 3D plots for higher dimensional data. But none of these statistical graphics are appropriate for most network models, including SAOMs. Thus, I will aim to create a way to, given various parameter values, visualize the distribution of a SAOM. One area which I will explore to accomplish this task is principal component analysis for graph data.

Once a distribution is characterized and viewable, the next things people notice are what the average value from this distribution looks like and how spread out the distribution is. Therefore, my next task will be to come up with a way to compute and view the "average" network from a given SAOM, as well as some measure of variability so that researchers can more easily determine whether or not an observed network or set of networks could possibly have come from a given model.

3 Visual Inference for Stochastic Actor Oriented Models

Abstract: In this chapter, I will use the visual inference lineup protocol to determine which aspects of SAOMs are visually discoverable. First, I explore the relationship between the addition of statistically significant parameters to a SAOM and the visual side effects of the inclusion of the additional significant parameter. Using visual inference, I will determine whether the addition of the significant parameter changes the observable structure of networks belonging to the model. Then, I will simulate networks from SAOMs using the multitude of possible parameter values available and I will use visual inference protocol to determine which parameters, when included in the model, make significant changes to the appearance of the networks.

3.1 Introduction

The introduction of this chapter will start with a shortened introduction to the rate function and objective function of a SAOM. I do not include it here for now because these functions are introduced in great detail in Section 1.2.

The R package `RSiena` provides methods for estimation of stochastic actor-oriented models for social network analysis (R Core Team 2016; Ripley, Boitmanis, and Snijders 2013). In this software, there are many possible parameter values that can be included into a model, but no ways of performing model selection. There are, however, some simple t-type and Wald-type tests of parameters that can be used to determine the significance of a parameter included in the objective function. This is crucial to the statistical analysis of the network in order to better understand the mechanism underlying the data. When studying a social network, it is also pertinent to the analysis to visualize the data. There is not, however, a way to know just which parameter values are affecting the network structure seen in the visualization. In this paper, we explore the visible effects of different model parameters, and use the lineup protocal of Buja et al. (2009) to perform visual tests of model parameters.

3.2 Data

The data we use in our first experiment is a small subset of 16 actors in the dynamic friendship network collected by Michell and Amos (1997). This data is made available on the RSiena webpage. It is a study of teenage girls and the changes in their friendship network overtime, and it also includes covariate information on the girls, such as their drinking and smoking behavior or whether or not they participate in school sports. We chose to subset the data to this small network to decrease the cognitive load on our experiment's subjects. The subset contained three waves of data, 16 girls and the relationships between them, and the drinking behavior of each actor at each of the three waves. This specific subset was chosen because it showed somewhat higher connectivity than other subsets, as determined by the adjacency matrix visualization of each wave shown in Figure 2. The first wave of our small network, which is conditioned on in estimation, is shown in Figure 3. The covariate, drinking behavior, is coded in the original data set as an integer from 1-5: 1 means no consumption of alchohol, 2 is consumption once or twice a year, 3 is once a month, 4 is once a week, and 5 alcohol consupmtion more than once a week.

3.3 Models

We fit three models to the data. The first model, which we write as $M1$, is the "null model" because it includes only the abolute minimum number of parameters that should be included in the objective function SAOM: the rates of change between timepoint, the outdegree effect, and the reciprocity effect (see T. A. Snijders 2001, 371). We also fit two "alternative models", which we write as M_2 and M_3. The alternative model M_2 includes one more parameter than the null model, the "jumping transitive triplet"" (JTT), which was determined to be significant by the t-type test in the `RSienaTest` package with a p-value of 0.00282. The parameters in the objective function are tested for significance using t-tests where test statistic is the ratio of the parameter estimate to its standard error. This parameter incorporates the actor covariate, drinking

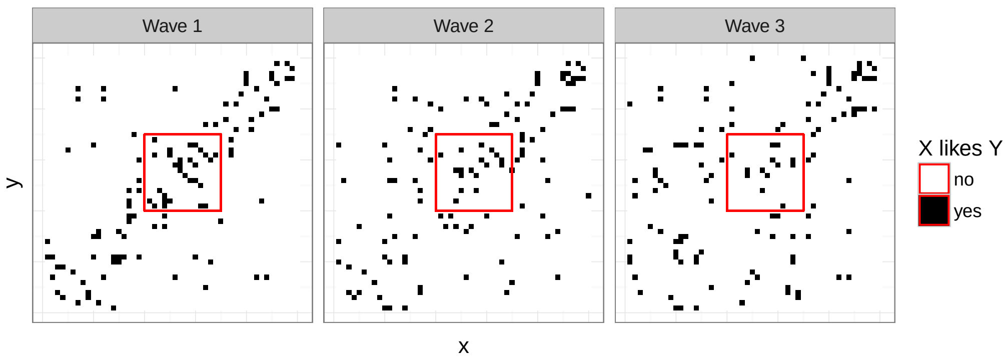

Figure 2: Adjacency matrices describing friendship relations between 50 students in waves 1,2, and 3 of the friendship study of friendsdata. The red squares identify the subset we focus on for our experiment.

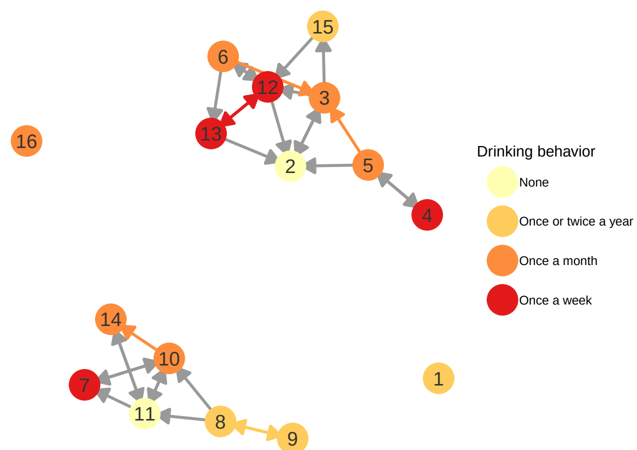

Figure 3: Network of friendships of wave 1 of the subset of students that we will be exploring.

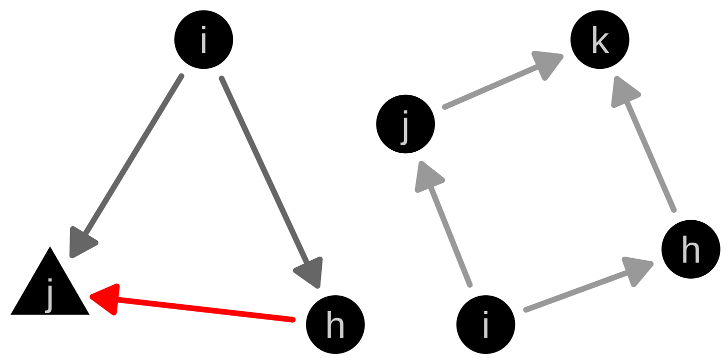

Figure 4: Structural network effects. On the left, a jumping transitive triplet (JTT). On the right, a doubly achieved distance between i and k. At left, a realization of a jumping transitive triplet, where i is the focal actor, j is the target actor, and h is the intermediary. The group of the actors is represented by the shape of the node. At right, doubly achieved distance between actors i and k.

behavior, into the model. The other alternative model, M_3, includes one additional structural parameter, "number of doubly achieved distances two effect", whose significance was determined in the same way, with a p-value less than 0.0001. Finally, each of the three models includes rate parameters, α_1 and α_2, which represent how many opportunities for change each actor gets, on average, when moving from wave 1 to 2 and from wave 2 to 3, respectively.

The objective function for each actor in each model is given below.

$$
\begin{aligned}
f_i^{M_1}(\boldsymbol{\beta}, x) &= \beta_1 s_{i1}(x) + \beta_2 s_{i2}(x) \\
f_i^{M_2}(\boldsymbol{\beta}, x) &= \beta_1 s_{i1}(x) + \beta_2 s_{i2}(x) + \beta_3 s_{i3}(x) \\
f_i^{M_3}(\boldsymbol{\beta}, x) &= \beta_1 s_{i1}(x) + \beta_2 s_{i2}(x) + \beta_4 s_{i4}(x)
\end{aligned}
$$

The form of the statistics, $s_{ik}(x)$ for $k = 1, \ldots, 4$ are given in Table 3. The outdegree parameter, β_1, represents how likely an actor is to change outgoing ties. If the estimate, $\hat{\beta}_1$, is positive, the actor is more likely to create outgoing ties, while a negative estimate leads the actor to deleting outgoing ties. This effect is highly correlated with the reciprocity parameter, β_2. A negative estimate of this parameter implies that the actor is discouraged from reciprocating its incoming ties, while a positive estimate implies that the actor is encouraged to reciprocate all ties. The additional parameter in M_2, β_3, is a the covariate parameter that takes into account the girls' drinking behavior. This jumping transitive triplet effect impacts the transitive closure of actors from different groups. Thus, a positive estimate encourages transitive closure when one of three actors is in a different covariate group than the other two, while a negative estimate discourages closure when one member is from a different group. An example of this type of closure is given in Figure 4. With the directed edges we also distinguish between 'i likes j' and 'j likes i'. Finally, the doubly achieved distances effect is defined by the number of actors to whom actor i is not directly tied, but to which it is connected through two different paths via at least two intermediary actors. This is a structural effect, like the density and reciprocity effects. A positive coefficient value encourages indirect ties, while a negative value discourages the formation of indirect ties in favor of direct ties.

3.3.1 Model Fitting and Simulation

Each model, $M1$, $M2$, and $M3$, was fit to the data 1000 times in order to obtain a distribution of parameter estimates. These distributions are shown in Figure 5. Then, the overall mean of the 1000 estimates for each parameter was calculated and used as the plug-in values for simulating from each of the three models. These values are presented in Table 3.

Table 3: Parameters and estimates of models M_1, M_2, and M_3. Estimates are the mean of 1000 iterations of the model estimates. The lineups that follow are simulated from models using these values.

Effect name	Parameter	Corresponding Statistic	M_1	M_2	M_3
Rate 1 (wave 1 → 2)	α_1	$\sum_{i,j=1 i\neq j}^{n} (x_{ij}(t_2) - x_{ij}(t_1))^2$	4.66	5.18	4.71
Rate 2 (wave 2 → 3)	α_2	$\sum_{i,j=1 i\neq j}^{n} (x_{ij}(t_3) - x_{ij}(t_2))^2$	1.93	2.02	1.98
Outdegree	β_1	$s_{i1}(x) = \sum_{j=1}^{n} x_{ij}$	-3.6	-4.1	-3.59
Reciprocity	β_2	$s_{i2}(x) = \sum_{j=1}^{n} x_{ij}x_{ji}$	4.15	4.28	4.23
Jumping Transitive Triplets	β_3	$s_{i3}(x) = \sum_{\forall j\neq h} x_{ij}x_{ih}x_{hj}\mathbb{I}(v_i = v_h \neq v_j)$	–	3.21	–
# doubly achieved distances	β_4	$s_{i4}(x) = \lvert\{j : x_{ij} = 0, \sum_h x_{ih}x_{hj} \geq 2\}\rvert$	–	–	-7.58

Because the likelihood function for these complicated models is intractable, `RSiena` implements Markov Chain Monte Carlo simulation to obtain method of moments estimates of the parameter values. This fitting procedure was first introduced in T. A. B. Snijders (1996).

The model fitting in `RSiena` is done in three phases. Briefly, in the initial phase, sensitivity of the statistics to the parameter values is determined, then in the second phase, parameter values are fit iteratively. Finally, in the third phase, networks are simulated from the fitted models and the model is checked for convergence (Ripley et al. 2016).

The convergence of a non-rate parameter is determined through the simulated values from phase 3 of the algorithm. The simulations are compared to the observed values of the statistics in the data. The values from the simulations should be fairly close to the observed values, but because the fitting is done stochastically, the deviation from statistics will not be exactly zero. Checking for convergence is based on a t-ratio of the average of these deviations to the standard deviation of these deviations. `RSiena` also performs an overall maximum convergence check by finding the maximum t-ratio value for any linear combination of the observed statistics. According to the RSiena Manual, "convergence is excellent when the overall maximum convergence ratio is less than 0.2", and for the non-rate parameters, the threshold for "reasonable" convergence is set at 0.3 with excellent convergence when the t-ratio is less than or equal to 0.1 in absolute value. (Ripley et al. 2016).

3.4 Using the Lineup Protocol

The additional parameters that we included, β_3 and β_4, were determined to be significant based on a rather simple statistical test, so we wanted to test whether this significance can be detected visually just as simply as the statistical test detects it. If visualizations of simulated networks from two nested models have a much different appearance when placed side-by-side, then the difference in appearance can be attributed to the additional parameter in one. If, however, there is no visually detectable difference, then the additional parameter does not appear to have changed the network structure all that much. Because model selection and diagnostics for network models are less developed areas of the theory, testing network parameters in this visual way could lead to additional methods of model selection for networks.

3.4.1 Lineup Simulation

To create the lineups that we used in our experiment, we used the values given in Table 3 as starting values in simulation of network observations from each of the three models. Each lineup was generated independently of the others. Four different lineup constructions were used: $M1$ v. $M2$, $M2$ v. $M1$, $M1$ v. $M3$, and $M3$ v. $M1$, where $M1$, $M2$, and $M3$ have the objective functions and parameter values given in Table 3. The lineup types are named to indicate which model is the "null" model and which model is the "alternative" model. The first model listed is considered the null model, so $M - 1$ of plots of data simulated from this model are included in the lineup, where M is the total number of plots included in the lineup. Then, one

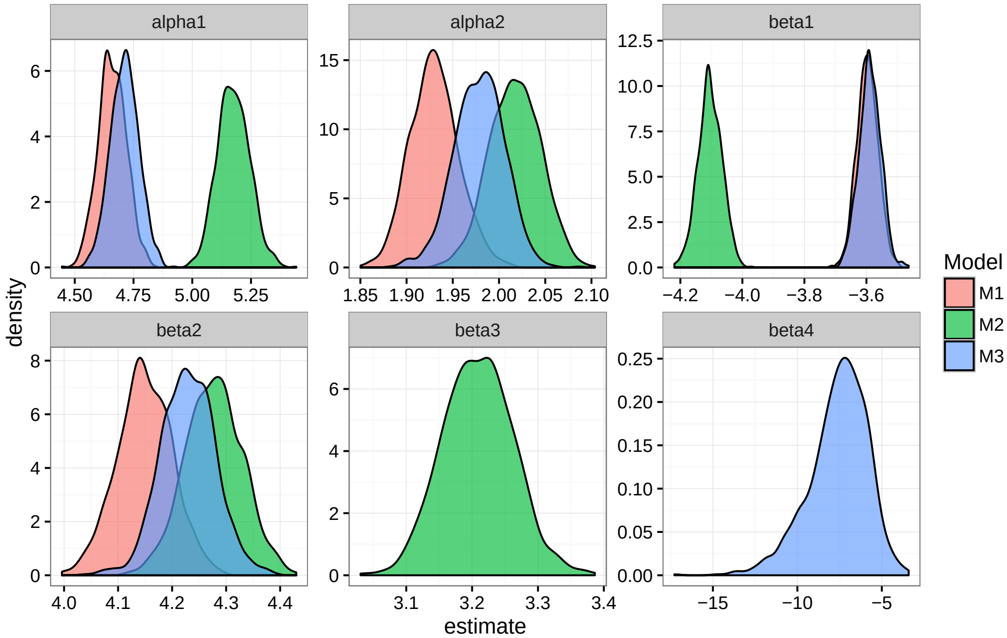

Figure 5: Histogram of the distribution of model parameters based on 1,000 simulation runs. Model parameter β_3 for jumping transitive triplets in model M_2 is significantly different from zero, but its inclusion also leads to significant changes in the other model parameters of model M_1. The parameter β_4 for doubly acheived distances is also significantly different from zero, but has larger variance. The inclusion of β_4 also changes the estimates of the other model parameters, but not as much as the inclusion of β_3.

additional plot from the second model listed, the alternative model, is placed among the $M - 1$ null plots at random. For each lineup simulated, the same default RSiena algorithm was used that was used to generate the fitted models with the exception that the algorithm only simulated from the given set of parameters. Additionally, each lineup and plot within lineup was generated idependently.

We initially constructed lineups of various sizes, $M \in \{3, 6, 9, 12, 16\}$. We ultimately decided to use lineups of size $M = 12$ because these lineups appeared to be the most square, and contained enough plots to make the probability of choosing the correct plot at random fairly low, at $\frac{1}{12} = 0.083$ while not overwhelming the viewer with too many plots to examine at once.

3.4.2 Parameter Estimation from Lineups

After the lineups were created, we re-fit each of our three models to each panel in each lineup of all sizes. Fitting all models to all lineup plots allows us to gauge the ability of the parameter estimates to provide a measure of lineup identification difficulty. For instance, when comparing models 1 and 2 in a lineup, the estimates from fitting model 2 to plots which were simulated from model 2 should be significantly different from the model 2 estimates from plots simulated from model 1. The smaller the difference between these estimates, the harder it should be to identify the different model in the lineup.

We did run into convergence problems when estimating parameter values from the lineup data. In about $\frac{1}{3}$ of cases, the algoritm did not converge, as determined by phase 3 of the fitting process and the convergence values stated in Section 3.3.1. Convergence by the model that was fit to the data and the model from which the data were simulated is sumarized in Figure 6. Additionally, distributions of parameter values for all fitted models according to the fitted model, the model from which the data were simulated, and whether or not the estimation converged are shown in Figure 7. For the estimates of $M1$ that did not converge, the distributions of the estimates were very different from what they should have been, i.e. distributions of $\hat{\beta}_3$ and $\hat{\beta}_4$ should be near 0 when the true model was model $M1$, but this is not the case for instances where the algorithm did not converge.

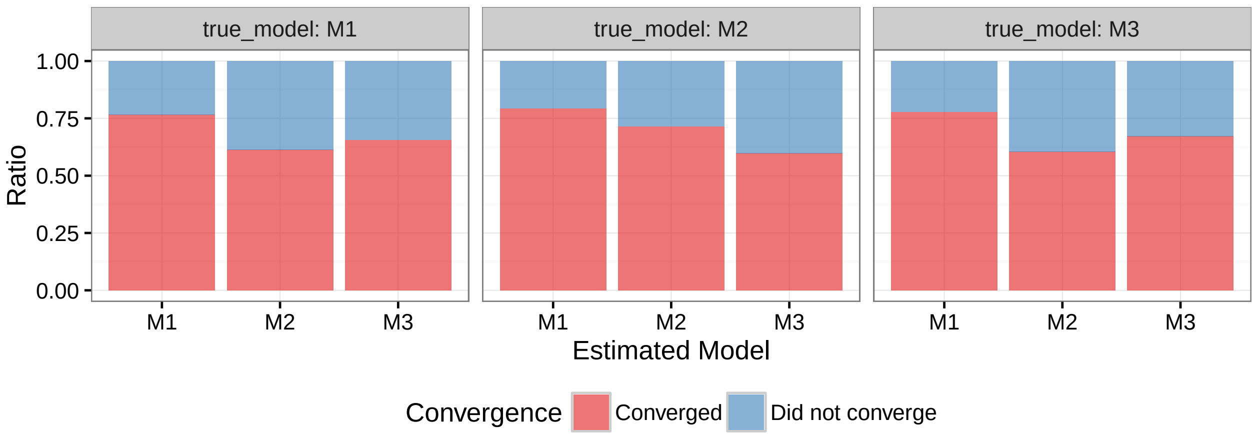

Figure 6: Pattern of convergence. A total of 67.7% of all lineup data converged in 5,000 iterations. The simplest model, M_1 has the highest rate of convergence. For true models M_2 and M_3, data generated from the true model converged at a higher rate than data generated from the other model.

Figure 8 shows an overview of model estimates for each simulated data set. What we expect to see is that estimates do not change much if they are estimated under a model different from the one they are generated from. This is true for all data sets estimated under model M_1 independently of which model they were simulated from (see top row of Figure 8. For data fit under model M_2 we see that parameter β_3 is estimated to be about zero, if the data is generated from models M_1 or M_3. For model M_2, the parameter β_3 is estimated to be significantly different from zero in 53% of cases. This coincides with our expectation.

However, the bottom row of Figure 8 shows that independently of which model data is generated from, β_4 is estimated to be significantly different from zero in a large number of cases (about half of the data generated from model M_2 and more than that in model M_1). While β_4 is a highly significant parameter in model M_3, this questions the way that parameters are fitted and tells us that we are not likely to be able to visually distinguish between data generated from model M_3 and data generated from models M_1 and M_2. We might, however, be able to distinguish between data sets for which β_4 is estimated to be significantly different from zero and those where β_4 is not significant. We hypothesize that the strong influence of the inclusion of β_4 in $M3$ makes the probability of being able to visually distiguish $M1$ from $M3$ extremely low. The β_4 estimate is always significantly greater than zero in fitted models that converged. Thus, β_4 should have been included from the beginning.

3.4.3 Results from A Pilot Study

We performed a pilot study that consisted of an evaluation of a set of 20 lineups by 11 volunteers. For each of the model situations (M_1 vs M_2, M_2 vs M_1, M_1 vs M_3 and M_3 vs M_1) one lineup of size $m = 3, 6, 9, 12$ was used. Overall, the number of data identifications in the lineups was very low (34 out of 220 evaluations). Participants identified the data plot in two to four of the lineups they evaluated. The mode was three data identifications out of twenty per participant.

3.4.3.1 Suitability of M_3 in lineups:

Figure 9 shows barcharts of responses from the pilot study detailing the number of data identifications in each lineup. It is more difficult to identify the data plot in lineups of graphs based on data from models M_1 and M_3 than in lineups of graphs based on data from models M_1 and M_2.

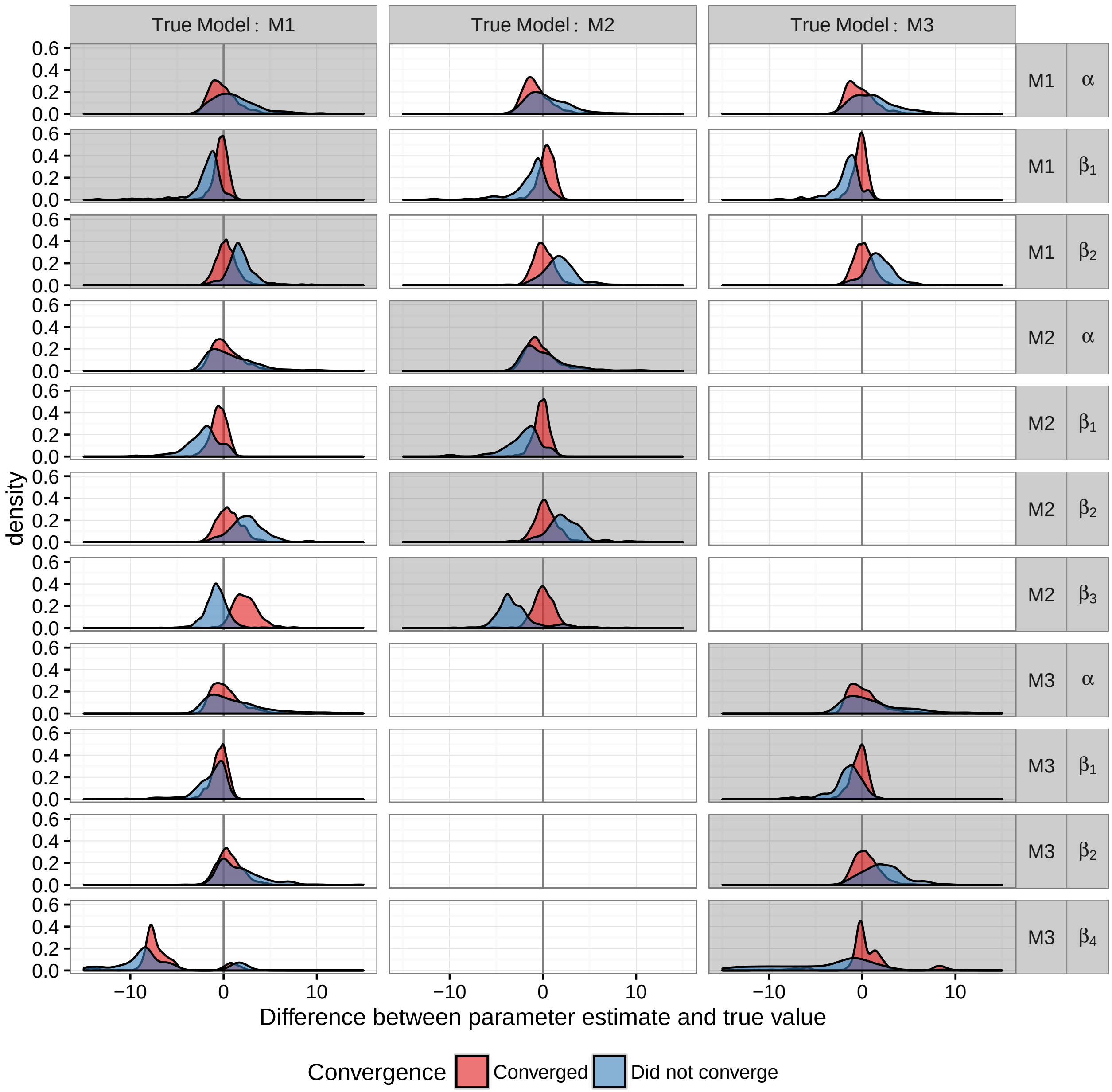

Figure 7: Difference between parameter estimate and true value. Panels with a light grey background show model fits with data sampled from the same model. Densities are drawn for both converged and non-converged data. The red-filled densities should have a mode near zero, idicating that the model converged toward the correct value. For data from model M_1, estimates for parameters β_3 and β_4 converge to a wrong value.

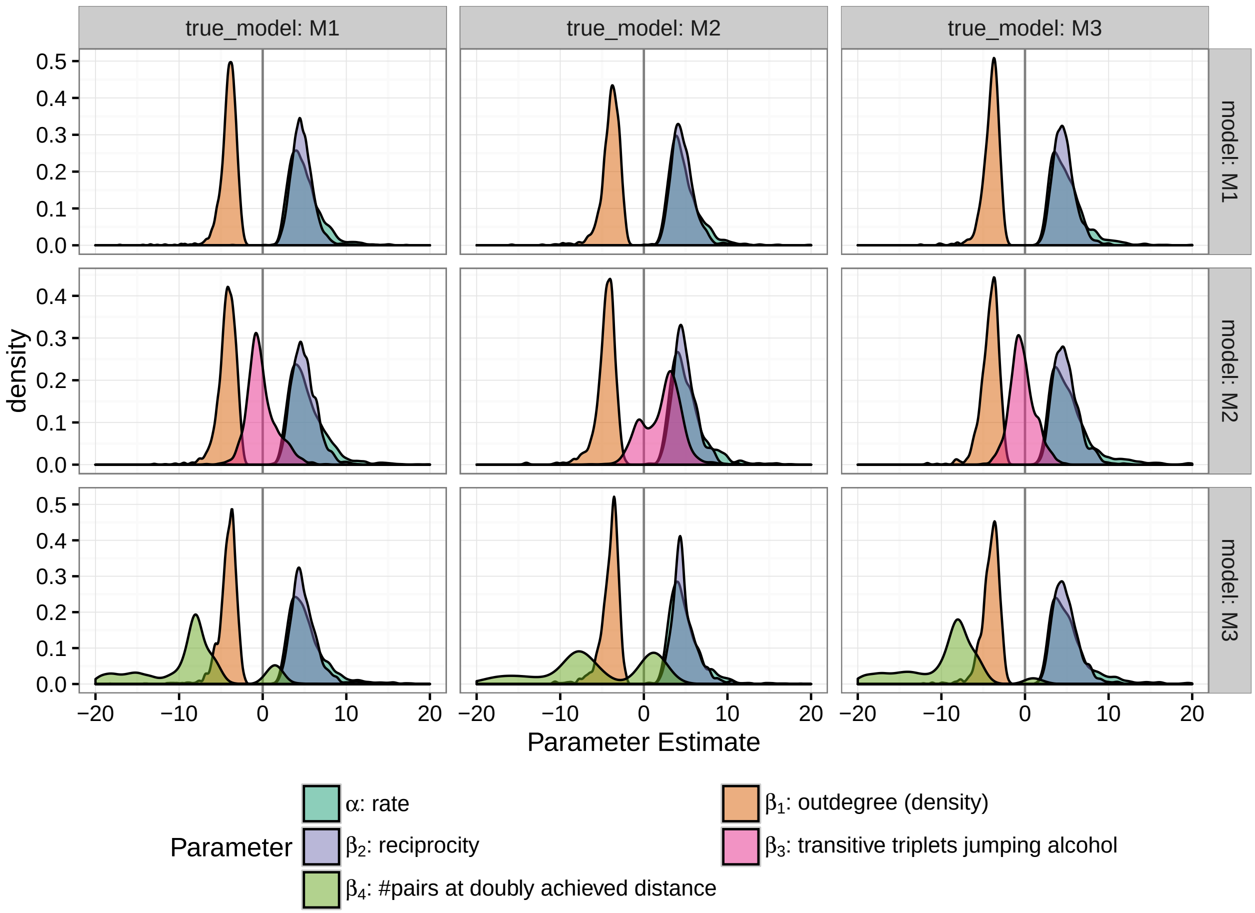

Figure 8: Comparison of model estimates under all three models under investigation.

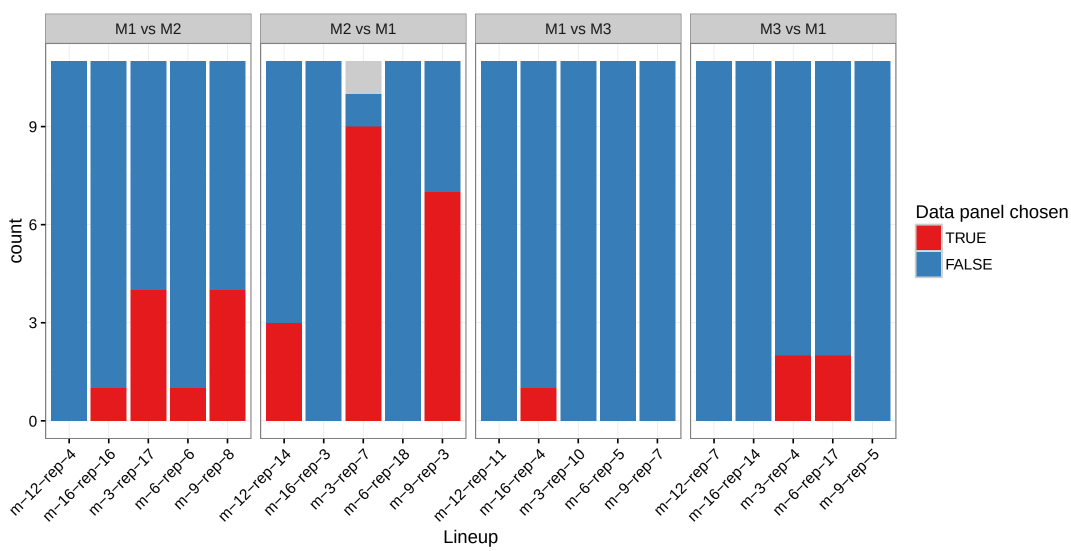

Figure 9: Barchart summarizing the number of responses from the pilot study by model and lineup. Color shows the number of times the data panel was chosen from the lineup. Clearly, the first two sets of lineups (M_1 vs M_2 and M_2 vs M_1) have on average higher number of data identifications.

3.5 Amazon Turk Experiment

3.5.1 Methods

In order to test our hypotheses, we set up an Amazon Mechanical Turk experiment (Amazon 2010) using the lineup protocal of Buja et al. (2009). We presented four types of lineups: We created a total of 25 lineups to show to Amazon Turk users taking our experiment: 10 for M1 v. M2 and 5 each for the other three types of lineups. In each lineup, there were 12 plots shown: 11 of the plots were simulated from the first model (the "null" model) and 1 was simulated from the second model (the "alternative" model). We chose to show lineups of size 12 because we felt that showing more than 12 would be too large of a cognitive load, while showing fewer than 12 would leave too much of the experiment to random chance. The order of the plots within the lineups was randomly assigned. We selected five lineups presented from each group based on the following criteria: first, for M1 v M2 and M1 v M3, we selected the lineups where the additional parameter in the objective function, β_3 and β_4, respectively, had the largest estimated value among the 12 networks presented. There were not many of these in the size 12 lineups, so our other selection criteria was that the estimated parameter value of interest was larger in the alternative panel than in at least half of the other lineups, and that it was close to the estimate from the panel that did have the largest value. For the $M2$ v $M1$ and $M3$ v $M1$ lineups, we first selected lineups of size 12 where the smallest parameter estimate belonged to the alternative model. If more lineups were needed, we next selected those which had estimates *smaller* than at least half of the other panels and were close to the *minimum* estimate in the lineup. For the remaining five plots of type $M1$ v $M2$, we chose the lineups where the alternative plot had the largest number of jumping transitive triplets, i.e. with the highest value of $\sum_i s_{i3}(x)$, appear in the network. We chose this statistic because its value has great effect on both the estimation of the β_3 parameter and on the visual appearance of the plot.

In order to become a subject in our experiment, the Amazon turk user had to first read through some introductory material and prove they could identify the correct plot in two test lineups. Each user was greeted with a brief welcome message and experiment description, seen in Figure 10. Then, a training page about how to identify the correct plot in a lineup, seen in Figure 11 was present. After the training page, two trial plots were presented. In order to complete the experiment, the user had to correctly identify the alternative plot. In the trial they identified a plot, provided reasoning for this choice (most complex, least complex, or other), and provided their confidence level in their choice (Very Uncertain, Uncertain, Neutral, Certain, Very Certain). Two of these trial plots with the correct responses selected are shown in Figure 12 one lineup of type $M1$ v. $M2$ and one of type $M2$ v. $M1$. The training lineups shown to the Turkers were randomly selected from ten lineups that were constructed to be very simple for training purposes. Once the turk user chose the correct plot in the two trial plots, the experiment proceeded with the same interface as the trial plots, with users selecting which plot they thought was most different, why they thought that, and how certain they were of their choice of different plot for 10 lineups chosen at random from the aforementioned pool of 25. The users were also allowed to select multiple plots for their response.

3.5.2 Results

We collected on ten lineups each from 77 Amazon Turk users. A table of demographic information collected on the participants in the experiment is in Table 4.

Because of the random assignment of the 25 plots to users, almost every lineup was seen by a different number of people. Repetition 3 of $M1$ v. $M3$ was seen by 47 different users while repetition 2 of the same type was seen by only 6 different users. The mean number of times seens was 30.8 per lineup and the median was 31. In 15 of the 25 lineups, no user identified the correct plot, and in the remaining 10 lineups, the users correctly identified the alternative plot in the lineup a minimum of 4.35% of the time and a maximum of 64% of the time. A plot showing the number of different plots chosen, how many times they were chosen, and whether or not it was the correct choice is given in Figure 13. Since the turk users were allowed to select multiple plots, the values shown in the histogram are actually the total weights received for each plot in the lineup. If the user selected only one plot, then that plot received a weight of 1, but if the user selected $n \geq 2$ plots, then each plot selected received a weight of $\frac{1}{n}$.

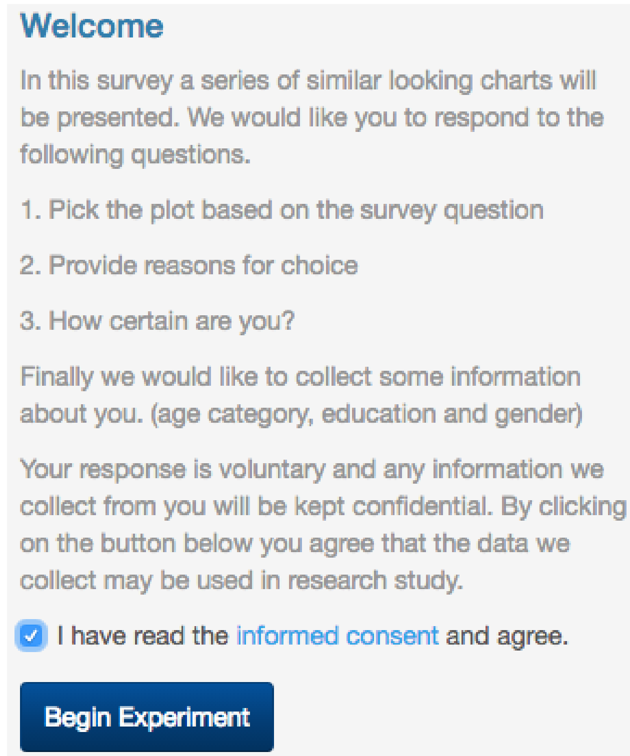

Figure 10: The welcome message shown to the Amazon Turk users.

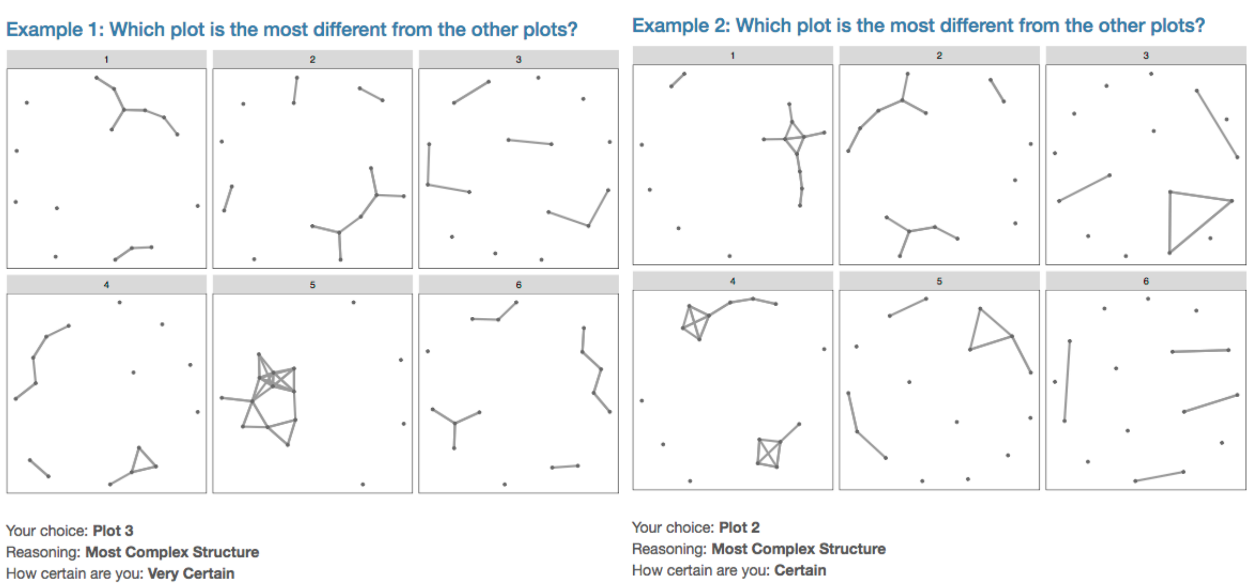

Figure 11: The first training page in the Amazon Turk experiment.

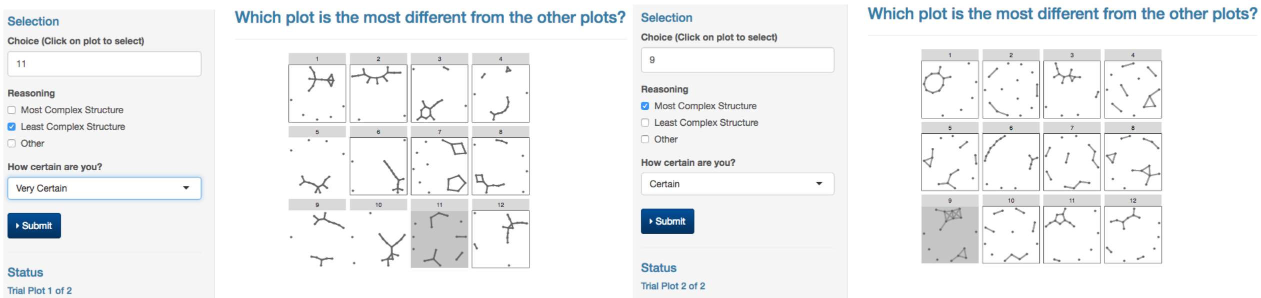

Figure 12: Two of trial plots that users had to correctly answer in order to participate in the Amazon Turk experiment. The lineup on the left is representative of the $M1$ v $M2$ type of lineup, while the lineup on the right is representative of the $M2$ v $M1$ type.

Table 4: Demographic information collected from experiment participants. An asterisk (*) indicates fewer than 5 participants.

Female	32
I choose not to provide this information	*
Male	69

18-25	20
26-30	31
31-35	24
36-40	11
41-45	5
46-50	2
51-55	4
56-60	3
Over 60	3

Graduate Degree	16
High School or Less	12
I choose not to provide this information	*
Some Graduate Courses	*
Some Undergraduate Courses	30
Undergraduate Degree	40

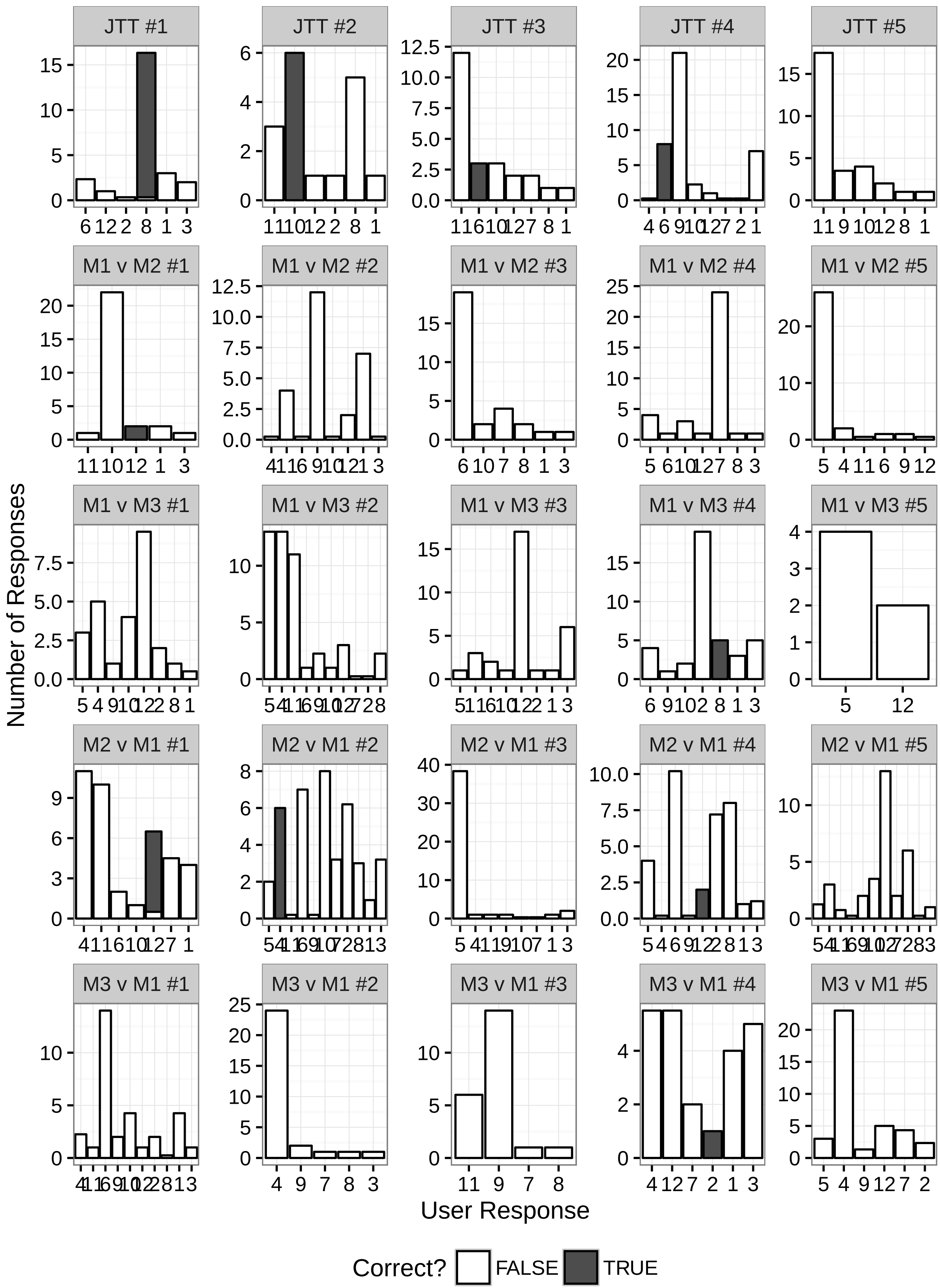

Figure 13: Plots chosen for each of the 25 lineups in the experiment. The x-axis ticks do not have a label because the label is less important than the number of different plots users chose and whether or not they chose the correct plot, shown in the coloring of the bars.

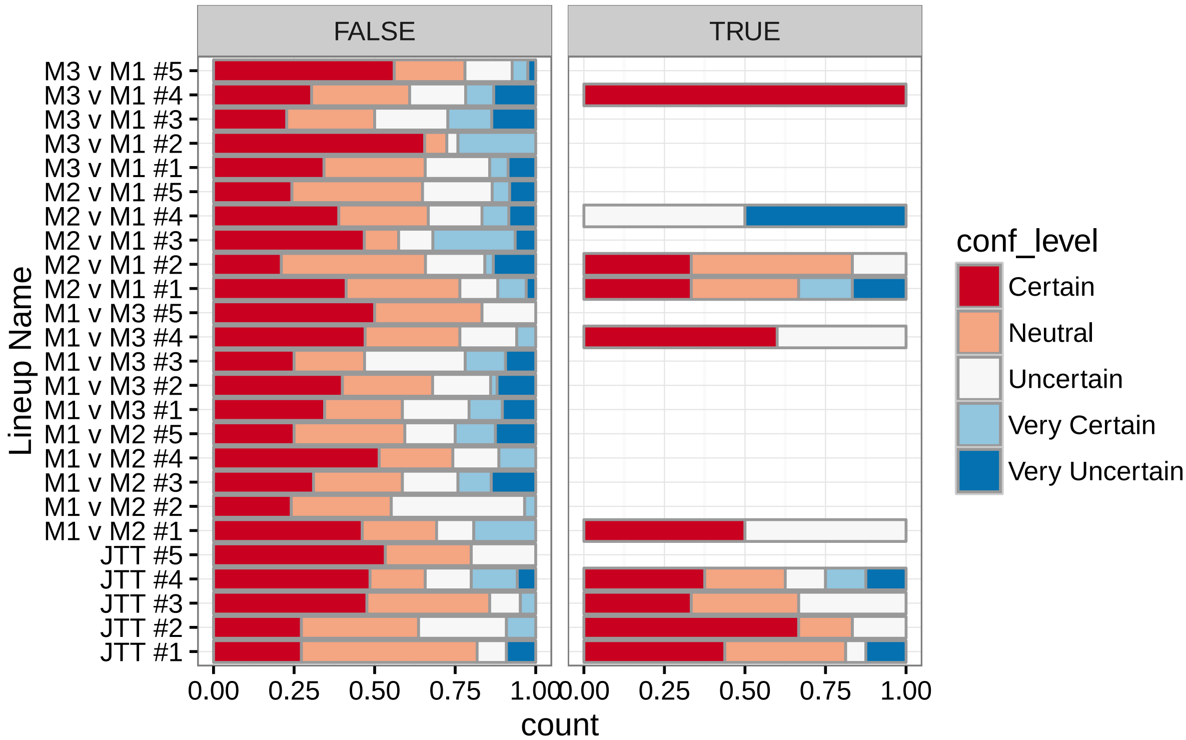

Figure 14: All responses for all lineups, separated by whether the plot selected was the true alternative plot (TRUE) or not (FALSE) and colored by the respondent's uncertainty levels. We see here that most respondents were certain in their answer whether or not they were correct.

This plot shows us the abysmal ability of the Turkers to identify the alternative plot correctly. There are also a few other interesting results from this plot. First, there are several lineups where a majority of participants selected the same wrong alternative plot. Additionally, we see a lot of variation in the number of different plots selected at the alternate plot by the Turkers. At most, there were 10 different plots selected, compared to 2 at the opposite end.

Next, we investigate the confidence of the experiment's participants in selecting the most different plot from the others. Overall, in 40% of the responses, the respondent was certain they were correct. User confidence in the remaining categories, neutral, uncertain, very certain, and very uncertain, was 26.88%, 16.88%, 9.48%, and 6.76%, respectively. These results are summarized by lineup in Figure 14. We also performed a 5-sample test for equality of proportions in order to test the null hypothesis that the proportion of correct responses is the same in all 5 certainty categories. This test resulted in a p-value of 0.5881, so there is no evidence that the certainty of a user's response affects whether or not they choose the correct plot. This provides further evidence that visually detecting differences between networks simulated from different models is an extremely difficult problem.

We next investigate the length of time that users' took to respond to each lineup. The minimum was 3.762 seconds and the maximum was 578.8 seconds (nearly 10 minutes). The median was 13.01 seconds and the mean was 20.32 seconds. First, a 2-sample Kolmogorov-Smirnov test for equality of distribution was done to see if the distribution of times for correct plots chosen is the same as the distribution of times for incorrect plots chosen. This two-sided test resulted in a p-value of 1, so there is no evidence that the distribution of times is different whether or not the response was correct.

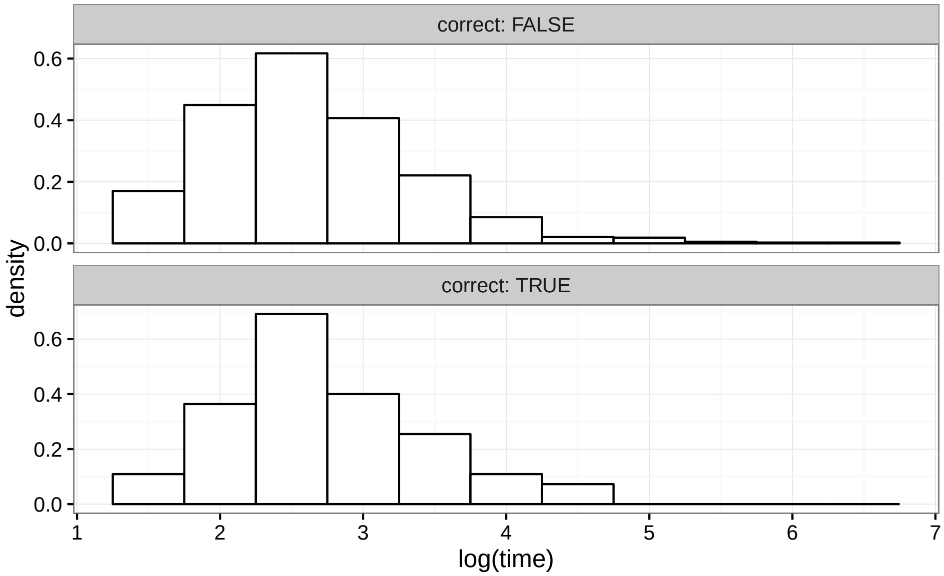

Figure 15: Log of time taken to complete a lineup for all responses, separated by whether or not the response was correct. The distribution is roughly the same for correct and incorrect responses, with more right skew on incorrect responses.

3.5.3 Discussion

This small experiment suggests that detecting a network model difference from just one simulation from one model and 11 from the other model is just about impossible. This result is fairly unsurprising: drawing a single random point from, say, a χ^2_1 distribution and placing it in a lineup with 11 separate draws from a standard normal distribution would likely have similar results. Every once in a while, a chi-square value would appear that would be too large to belong with draws from a standard normal distribution, but values near 1, the mean, would probably not appear that different from some random standard normal draws. This means that we need to develop a different way to compare two network models. We need a way to visualize many samples from a network model in one panel.

We have also discovered that a high value of the JTT statistic leads to correct lineup selection more than a high estimated parameter value. This leads us to question whether statistical significance has any relation to visual detectability at all.

3.6 Future Work

The next part of this paper requires several more experiments to be performed. We will simulate data from many more models, each with different parameters and parameter values included in the model. Then, networks simulated from null models like $M1$ will be placed in lineups with a network simulated from one of the several alternative models we will define. By performing visual inference on SAOMs in this way, we hope to determine which parameters, when included, cause significant change in the visible network structure.

4 Drawing Networks in the `ggplot2` Framework

This next section is a paper I authored with Heike Hofmann (Iowa State University) and François Briatte (European School of Political and Social Sciences) that has been accepted for publication in *The R Journal* subject to revision. I developed the package `geomnet` ("our package") with Dr. Hofmann in 2015. Separately from the implementation of the graph `geom` in our package, Briatte implemented graph visualization in two different approaches: the ggnet2 function in the GGally package and the ggnetwork package. This paper gives an overview of the three different approaches, highlights why ggplot2 is generally well suited for graph visualization, and then discusses the pros and cons of the three methods: geom_net() in geomnet, ggnet2() in GGally, and ggnetwork() in ggnetwork.

5 Other Projects

I have also completed a couple of other significant research projects that I feel have been important in my studies. The first of these is a project on clustering digital images of paintings of the artist Bob Ross. This was originally a project assignment for STAT 503: Exploratory Methods and Data Mining in Spring 2015. I entered a version of the project assignment into Significance Magazine's 2015 Young Statisticians Writing Competition, where it was among the final three choices for the top prize.[2] The other project is a visual exploration of the Trans-Atlantic Slave Trade Database, in which I used the `geomnet` package to better understand the structure and impact of the trans-atlantic slave trade. I submitted this paper to the 2016 Student Paper Competition sponsored by the ASA Section on Statistical Graphics and Computing, where it won top prize for a student paper in Graphics.[3]

6 References

Amazon. 2010. Https://www.mturk.com/mturk/welcome.

Barabási, Albert-László, and Réka Albert. 1999. "Emergence of Scaling in Random Networks." *Science* 286 (5439). American Association for the Advancement of Science: 509–12. doi:10.1126/science.286.5439.509.

Bender-deMoll, Skye. 2016. *Ndtv: Network Dynamic Temporal Visualizations*. https://CRAN.R-project.org/package=ndtv.

Brandes, Ulrik, Patrick Kenis, and Dorothea Wagner. 2003. "Communicating centrality in policy network drawings." *IEEE Transactions on Visualization and Computer Graphics* 9 (2): 241–53.

Buja, Andreas, Dianne Cook, Heike Hofmann, Michael Lawrence, Eun-Kyung Lee, Deborah F. Swayne, and Hadley Wickham. 2009. "Statistical Inference for Exploratory Data Analysis and Model Diagnostics." *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 367 (1906). The Royal Society: 4361–83. doi:10.1098/rsta.2009.0120.

Butts, Carter T. 2014. *Sna: Tools for Social Network Analysis*. https://CRAN.R-project.org/package=sna.

Chowdhury, Niladri Roy, Dianne Cook, Heike Hofmann, Mahbubul Majumder, Eun-Kyung Lee, and Amy L Toth. 2014. "Using visual statistical inference to better understand random class separations in high dimension, low sample size data." *Computational Statistics* 30 (2): 293–316.

Csardi, Gabor, and Tamas Nepusz. 2006. "The Igraph Software Package for Complex Network Research." *InterJournal* Complex Systems: 1695. http://igraph.org.

Eades, Peter. 1984. "A Heuristic for Graph Drawing." *Congressus Numerantium* 42 (11): 149–60.

Erdös, Paul, and Alfréd Rényi. 1959. "On Random Graphs I." *Publicationes Mathematicae* 6: 290–97.

Fruchterman, Thomas M.J., and Edward M. Reingold. 1991. "Graph Drawing by Force-Directed Placement." *Software: Practice and Experience* 21 (11): 1129–64.

Goldenberg, Anna, Alice X. Zheng, Stephen E. Fienberg, and Edoardo M. Airoldi. 2010. "A Survey of Statistical Network Models." *Foundations and Trends in Machine Learning* 2 (2): 129–233. doi:10.1561/2200000005.

Handcock, Mark S., David R. Hunter, Carter T. Butts, Steven M. Goodreau, and Martina Morris. 2008. "Statnet: Software Tools for the Representation, Visualization, Analysis and Simulation of Network Data." *Journal of Statistical Software* 24 (1): 1–11. http://www.jstatsoft.org/v24/i01.

Hofmann, Heike, Lendie Follett, Mahbubul Majumder, and Dianne Cook. 2012. "Graphical Tests for Power Comparison of Competing Designs." *IEEE Transactions on Visualization and Computer Graphics* 18 (12):

[2] See https://www.statslife.org.uk/culture/2553-the-joy-of-clustering.
[3] See http://stat-computing.org/awards/student/winners.html.

2441–8.

Holland, Paul W., and Samuel Leinhardt. 1981. "An Exponential Family of Probability Distributions for Directed Graphs." *Journal of the American Statistical Association* 76 (373). [American Statistical Association, Taylor & Francis, Ltd.]: 33–50. http://www.jstor.org/stable/2287037.

Kamada, Tomihisa, and Satoru Kawai. 1989. "An Algorithm for Drawing General Undirected Graphs." *Information Processing Letters* 31 (1): 7–15.

Kolaczyk, Eric D. 2009. *Statistical Analysis of Network Data: Methods and Models*. Springer.

Loy, Adam, and Heike Hofmann. 2015. "Are You Normal? The Problem of Confounded Residual Structures in Hierarchical Linear Models." *Journal of Computational and Graphical Statistics* 24 (4): 1191–1209.

Loy, Adam, Lendie Follett, and Heike Hofmann. 2016. "Variations of QQ Plots: The Power of Our Eyes!" *The American Statistician* 70 (2): 202–14.

Majumder, Mahbubul, Heike Hofmann, and Dianne Cook. 2013. "Validation of Visual Statistical Inference, Applied to Linear Models." *Journal of the American Statistical Association* 108 (503): 942–56. doi:10.1080/01621459.2013.808157.

Michell, Lynn, and Amanda Amos. 1997. "Girls, pecking order and smoking." *Social Science & Medicine* 44 (12): 1861–9.

R Core Team. 2016. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Ripley, Ruth, Krists Boitmanis, and Tom A.B. Snijders. 2013. *RSiena: Siena - Simulation Investigation for Empirical Network Analysis*. https://CRAN.R-project.org/package=RSiena.

Ripley, Ruth, Tom A.B. Snijders, Zsófia Boda, András Vörös, and Paulina Preciado. 2016. *Manual for RSiena*. University of Oxford: Department of Statistics; Nuffield College; University of Groningen: Department of Sociology.

Snijders, T A B. 1996. "Stochastic actor-oriented models for network change." *Journal of Mathematical Sociology* 21 (1-2): 149–72. http://www.scopus.com/inward/record.url?eid=2-s2.0-0000413317{\&}partnerID=40{\&}md5=d981a59ed505ebc7fe083b42a8b9a179.

Snijders, Tom a. B., Johan Koskinen, and Michael Schweinberger. 2010. "Maximum likelihood estimation for social network dynamics." *The Annals of Applied Statistics* 4 (2): 567–88. doi:10.1214/09-AOAS313.

Snijders, Tom A.B. 2001. "The Statistical Evaluation of Social Network Dynamics." *Sociological Methodology* 31 (1). Blackwell Publishers, Inc.: 361–95. doi:10.1111/0081-1750.00099.

"The Erdös Number Project." 2016. \url{http://wwwp.oakland.edu/enp/}.

Wasserman, Stanley S. 1980. "A Stochastic Model for Directed Graphs with Transition Rates Determined by Reciprocity." *Sociological Methodology* 11. [American Sociological Association, Wiley, Sage Publications, Inc.]: 392–412. http://www.jstor.org/stable/270870.

Watts, D J, and S H Strogatz. 1998. "Collective dynamics of 'small-world' networks." *Nature* 393 (6684): 440–2. doi:10.1038/30918.

Wickham, Hadley. 2009. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. http://ggplot2.org.

Wickham, Hadley, Dianne Cook, and Heike Hofmann. 2015. "Visualizing statistical models: Removing the blindfold." *Statistical Analysis and Data Mining* 8 (4). John Wiley; Sons Inc.: 203–25.

Wilkinson, Leland. 1999. *The Grammar of Graphics*. New York: NY: Springer.

Zhao, Yifan, Dianne Cook, Heike Hofmann, Mahbubul Majumder, and Niladri Roy Chowdhury. 2013. "Mind Reading: Using an Eye-Tracker to See How People are Looking at Lineups." *International Journal of Intelligent Technologies and Applied Statistics* 6 (4): 393–413.