A Simulation Study Using Random Graph Models to Fit Social Networks

Levi Lee

April 05, 2017

Submitted to the
Department of Mathematics and Statistics
of Amherst College
in partial fulfillment of the requirements
for the degree of
Bachelor of Arts with honors

Faculty Advisor: Amy Wagaman

Acknowledgements

This thesis would not have been possible without the support of so many people. First and foremost, I would like to thank everyone currently from the Mathematics and Statistics departments along with some previous faculty as well. In particular, Professor Xiaofei (Susan) Wang and Professor Eunice Kim were some of the first people that really got me heavily involved with R and R-Studio through the coursework they provided. I would never imagine that one day, I would be doing something so advanced with such software. I would also like to thank my fellow stats majors for always providing such amazing help when I get stuck. My gratitude goes to Stephany Flores-Ramos '17, Melody Owen '17, Jordan Browning '17, and Muling Si '17 for being such wonderful peers in Advanced Data Analysis and for always bearing with me. Seeing how wonderfully they all code really inspired me to improve my own skills as well. Special thanks also goes to Ningyue (Christina) Wang '16 for all the senior advice when I was a junior. To all the students, faculty, and staff who knew I was working on this project, thank you for the constant moral support. Those who also had a hand in helping with this project, even if it was just moral support, are truly a cut above. I am especially grateful to Roy Andrews '80 from the Writing Center, Caleb Ki '17, and Azka Javaid '17 who were kind enough to read over my thesis and provide valuable feedback. Special thanks to Nguyen (Johnson) Tran, a childhood friend from back home in Florida, who was kind enough to gift me a wonderful textbook on statistics, which proved to be very useful in my research. I want to dedicate this next paragraph to my family. Without their love and support, I would not be at Amherst today. Since the day I started school, my parents have done everything in their power to make sure the only thing I ever really need to focus on was my education. Nothing could ever amount to the sacrifices they have made. I now realize that being able to focus solely on academics was a privilege that not everyone had. My penultimate thanks goes to Professor Nicholas Horton, whom I have had the pleasure of having a professor for two courses in a row. He is a professor, mentor, and coach who really has a sense of the direction statistics is taking, and is really preparing us all to "think with data." Finally, I want to give my thanks to Professor Amy Wagaman, who has been my "three-times advisor" for Mathematics, Statistics, and my Mathematics Thesis. She has been an amazing advior in all regards, and I cannot thank her enough for her patience, understanding, and dedication to my learning. Working on such a project has really been challenging, but was definitely the highlight of my final year in college. It is thanks to her that I was able to explore the facinating world of networks and pursue graduate study in this field. I hope that this project has made her proud.

Table of Contents

List of Tables

List of Figures

Abstract

Networks see usage in across many disciplines. This work focuses on social networks, a category of networks which tend to exhibit some characteristics unique to themselves. Is it possible to model social networks despite their unique properties? To answer this question, we begin with an overview of networks, including network statistics, and graph models. I look at the Erdős-Rényi and Watts-Strogatz models as well as exponential random graph models (ERGMs). We conduct a simulation study where we apply various graph models to an observed network. Here, we choose to analyze a component of the Facebook network. We compare the network statistics of this Facebook network with those calculated from graphs of parable magnitude simulated from their respective graph models. We conclude that models such as the Erdős-Rényi and Watts-Strogatz models do not accurately fit the Facebook network for these network statistics. ERGMs that only use combinations of edge, triangle, and k-star parameters also do not accurately model this Facebook network as well. However, the results suggest that adding other parameters can significantly improve simulation results.

Introduction

Generally speaking, a *network* is a collection of inter-related objects. This broad definition allows networks to be used in a variety of disciplines. We focus on *social networks*, in which these objects are either individuals or groups (though these objects need not be human), called *actors* and the relationships among them are called *ties*. For people, these ties can be friendships, collaborations, and email exchanges, among other things. Social networks have been a growing field of study since the 1930s. Even before the computer age, people have observed unique relationships among individuals (Kolaczyk 2009 and Newman 2010). One famous example is Milgram's letter experiments. Travers & Milgram (1967) considered 296 individuals in Nebraska and Massachusetts who were instructed to construct chains of correspondence via letter until the correspondence reached a target person in Boston. Out of the 64 letter chains that succeeded, they discovered that, overall, these chains did not require that many people. Surprisingly, the median number of people the letters had to go through was approximately six, giving rise to the popular term "six degrees of separation."

This phenomenon is not true for all networks; in fact, many social networks exhibit characteristics not seen in other types of networks. Would it be possible then, for us to model social networks despite phenomena such as the "six degrees of separation?" Models for networks certainly exist, but how accurate are they? It turns out that through a simulation study, we can hope to find an answer (or at least be pointed in the right direction). However, before heading straight to a simulation study, we must be careful in our approach. Exactly how do we mathematically define a network? What does a model for a network consist of? How would we measure accuracy for such models? We begin with some mathematical background before describing the components of the simulation study and we will see examples of social networks in subsequent sections.

0.1 Basic Terminology and Definitions[1]

In order to precisely define what we mean by a network, we borrow much of the mathematical terminology for networks and network statistics from graph theory. More formally, a network is a collection of vertices and edges. It is a *graph*, that is, a structure denoted $G = (V, E)$ such that E is the set of edges and V is the set of vertices. The *order* of a graph G is defined to be the number of vertices in G, denoted $N_V = |V|$. Similarly, the *size* of G is the number of edges in G, denoted $N_E = |E|$.

Each edge connects two vertices, and so for any two vertices $i, j \in V$, if i and j have a connection then the ordered pair $\{i, j\} = e \in E$. In this case, we say i is *adjacent* to j and that i and j are *neighbors*. That is, $\{i, j\}$ is the same as $\{j, i\}$. Such a graph is called *undirected* (or *non-directed*). Graphs that have ordering to their edges, that is, where $\{i, j\} \in E$ is different from $\{j, i\} \in E$, are called *directed graphs* (or *digraphs*), and the edges in a directed graph are called *directed edges* (or *arcs*). We will focus on graphs whose edges are undirected.

We can also represent a network with just the connections that are present (or not present) between vertices by using an *adjacency matrix*. Given a graph $G = (V, E)$, the adjacency matrix $\mathbf{A}$ is an N_V x N_V matrix where

$$A_{ij} = \begin{cases} 1 & \text{if } \{i, j\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

Matrix $\mathbf{A}$ is symmetric if G is a undirected graph, but this may not necessarily be the case if G is a directed graph. This is because $\{i, j\} \in E$ does not necessarily guarantee that $\{j, i\} \in E$ for directed graphs.

While the adjacency matrix $\mathbf{A}$ is a data structure that fully characterizes a graph, for large N_V, size and reading time on the part of the computer can become an issue very quickly, especially in the context of working with data structures and particular algorithms. For a graph with N_V vertices, the corresponding adjacency matrix will have N_V^2 elements. Thus, we can also consider an alternative: an *edge list* is a two-column list of all the edges and their corresponding vertices present in a graph. More often than not, a graph will not have an edge between every pair of vertices, so it is more often the case that using an edge list will mean we are dealing with strictly less than N_V^2 elements. Larger networks, as we will soon see, are more often than not represented with an edge list.

[1]The terms and definitions listed here are certainly not an exhaustive list. This section follows largely from Kolaczyk's *Statistical Analysis of Network Data*. See Kolaczyk (2009) for more details.

To describe how to move about in a network, we define a *walk* in graph G to be a sequence that begins and ends with two vertices and alternates between vertices and edges. The sequence $(a_0, e_1, a_1, e_2, ..., v_{l-1}, e_l, v_l)$ denotes one path from vertex a_0 to vertex a_l, where for all $e_i \in E$, e_i connects points a_{i-1} and $a_i \in V$. Using this notation, we define the *length* of the walk to be the value l. More refined than a walk, a *trail* is defined to be a walk in which the sequence does not traverse through the same edge more than once. And even more refined still is a *path* in which a walk does not travel through the same vertex twice.

A vertex j is *reachable* from another vertex i is there is a walk from i to j. G is *connected* if all vertices in G are is reachable by some other vertex in G. A *component* is a subgraph that is maximally connected. The length of the shortest path(s) between any two vertices is defined to be the *distance* (or *geodesic distance*) between vertices. This distance is infinite if no path between two particular vertices exist, that is, they are located on different components. The *diameter* is the longest (geodesic) distance achieved in a graph. In a digraph, a *directed walk* is analogous to a walk, but uses arcs to get from one vertex to another.

A graph is *complete* if every vertex has an edge with all other vertices in the graph. An undirected graph is *simple* if it has no *loops* and *multi-edges*, meaning vertices do not have an edge that points to itself and any pair of vertices do not have more than one edge between them. Edges on a simple graph are called *proper edges*. A *clique* is a graph or subgraph that is complete. A graph is *regular* if each vertex has the same *degree*. That is, each vertex has the same number of edges connecting to it. Degree is usually denoted k, and a particular vertex with a degree k is called a *k-star*.

Connectedness and simpleness are very important characteristics of networks as many of the descriptive statistics that we will see can only be applied graphs with such properties. In fact, our focus will also be limited to simple and connected graphs.

A graph can contain other information or data, called *attributes*, beyond just the collection of vertices and edges. Incorporating these attributes, otherwise known as *decorating* a graph, can occur in both the vertices, edges, and even in the entire graph itself. For example, it is possible to assign *edge weights* to each edge. That is, we assign an edge weight w_e to $e = \{i, j\} \in E$. A *weighted graph* is a graph whose edges include weights that are nonzero and between 0 and 1. Edge and vertex attributes can be *discrete* or *continuous*. An example of a discrete attribute is a binary case of positive or negative relationship between pairs of vertices. An example of a continuous attribute would be the strength (or weight) of the relationship between pairs.

0.2 Example 1: Zachary's Karate Club

One famous (or infamous) social network, called "the karate club of Zachary," was observed by anthropologist Zachary (1977). The vertices in this network indicate the 34 individual actors (that is, members) of the karate club while the edges indicate the 78 social ties among pairs of the members. Figure 1 provides a visual summary of the karate network data. As shown by the colors of the vertices, the club, at this particular moment in time, is being divided into two factions centering around two vertices as labeled by letters–the master and primary disciple of the dojo.

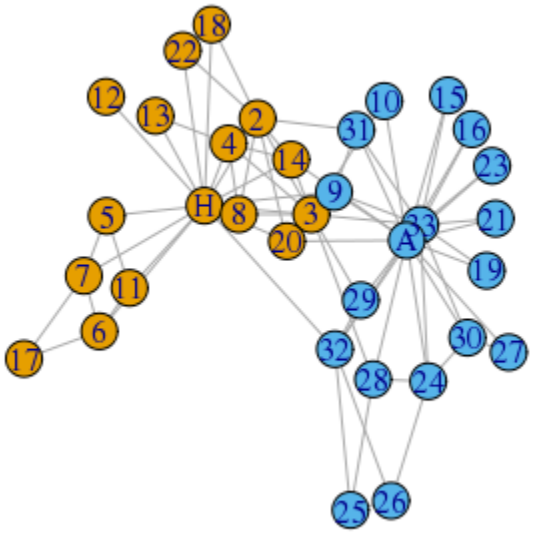

Figure 1: Zachary's Karate Club ($N_V = 34$, $N_E = 78$)

It is possible to display more than just the actors and their relationships. We can provide more information by assigning other visual cues–such as vertex shapes to determine gender, vertex size to indicate belt rank, and edge weights to determine the strength of the social tie–to each member.

0.3 Example 2: Lazega's Law Firm

Consider a network of actors that reflects the relational data collected by Lazega & Pattison (1999) of a New England law firm consisting of over 70 lawyers to study cooperation in organizations. This network has 36 actors (lawyers) and 115 ties (collaborations). In Figure 2, each lawyer is individually labeled with smaller numbers

indicating seniority. Shapes for each lawyer indicate the type of practice: litigation (circles) and corporate (squares). The shapes' sizes indicate the relative number of years of having been in with the law firm. Color is used to indicate office location (red, blue, and yellow). The graph below provides a visual summary of the lawyer network data.

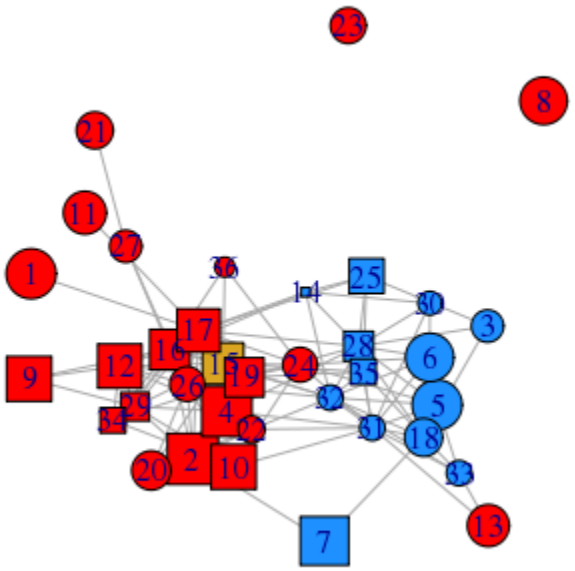

Figure 2: Lazega's Law Firm ($N_V = 36$, $N_E = 115$)

We can also continue adding layers of data to this visual as well. For example, if the appropriate data was recorded, it is possible quantify and/or qualify the collaborations between the lawyers. We can apply weights to give an idea of how frequently any pair of lawyers interacted with each other in this resource exchange; we can even apply line types (solid, dashed, double-dashed, etc.) to indicate how this collaboration took place: was it done in person, through the phone, through email, or some combination of the three?

0.4 Modeling Real-World Networks

As noted earlier, one topic of study with practical applications is using models to "model" observed networks. These are more formally known as *graph models* and for now, we can think of them as ways to emulate graphs. Which models can generate graphs that are visually and structurally similar to the social networks that we

observed? In reality, there are many graph models to choose from. With a network chosen, one approach is through a simulation study in which we can generate graphs from different graph models and compare *descriptive network statistics* between the simulated graphs and the observed graphs to see how accurate these graph models are.

Being able to model observed networks would benefit many fields. By accurately predicting how networks form, it may be possible to understand how friendships among people form or even how information is spread across people. More generally, we can improve traffic flow, provide better product suggestions to online shoppers, and help prevent the spread of diseases.

In this thesis, we provide a general overview about network statistics and graph models, followed by a simulation study that tests different graph models and see how well they "fit" a target social network of our choosing. In Chapter 1, we focus on various network statistics. These statistics have direct (but sometimes non-trivial) analogs to the measures of spread and central tendency seen in introductory Statistics. For example, we will see that *centrality*, a measure of importance of vertices in graphs, is similar to the arithmetic mean for a set. In Chapter 2, we focus on describing various graph models, such as the Erdős-Rényi and Watts-Strogatz models and several different exponential random graph models (ERGMs). These graph models will be used to fit a social network of our choosing. We will be using an undirected, simple, and connected component of the Facebook network, which consists of 4039 vertices and 88234 edges. Chapter 3 describes our simulation study and results. We compare the network statistics of our simulated graphs to those of the observed network. We conclude with Chapter 4, which describes some of the limitations in this study and potential future work.

Chapter 1

Descriptive Network Statistics

Descriptive statistics for networks are somewhat analogous to those seen in elementary statistics. In fact, we have seen a few already in the section of basic terminology in our Introduction. We describe a few below and explain them in the context of Zachary's karate club and Lazega's lawyer firm.[1] We will use these network statistics to characterize our component of the Facebook network as well as the networks generated by the graph models described in the next chapter. As a result, we then have a way to compare different networks, which is similar to how elementary descriptive statistics allow us to compare across different data sets.

1.1 Transitivity

Transitivity, otherwise known as the *clustering coefficient*, is the probability that adjacent vertices of a particular vertex are connected. In other words, this value provides a sense of how well-connected the graph is. A higher transitivity implies a graph having many vertices in proximity with each other with edges connecting all of them. Different kinds of transitivity exist and the choice of which one to observe depends largely on whether one wishes to talk about the network globally or locally (Kolaczyk 2009). Both are explained below, but for the sake of choosing one for our analyses, we will focus on the *local average transitivity*.

Globally speaking, this value is the ratio of connected triples to all possible triangles in the graph. We define a *triangle* (or *triad*) as subgraphs with three vertices; if these vertices are connected by two edges, it is a *connected triple*. Formulaically, the

[1] Just like the basic terminology for networks, the list of network statistics seen here is not exhaustive. For more details, please refer to Kolaczyk (2009), Newman (2010) and Csárdi & Nepusz (2006).

clustering coefficient, denoted with uppercase C, is

$$C = \frac{(\text{number of triangles}) \times 3}{\text{number of connected triples}}.$$

The factor of 3 represents the fact that a triangle is counted three times when we consider the number of triples in the network.

The local transitivity is calculated for each vertex in the graph and is the ratio of triangles connected to a particular vertex to the connected triples centered on the vertex. For a particular vertex i, the local transitivity is denoted as

$$C_i = \frac{(\text{number of pairs of neighbors of } i \text{ that are connected})}{(\text{number of pairs of neighbors of } i)}.$$

The local average transitivity then, is the average of the local transitivity values for each vertex in the graph.

In the Zachary's karate club and Lazega's law firm examples, the local average transitivity values are: 0.588 and 0.487, respectively, and the global transitivity values are 0.256 and 0.389, respectively. This means that while the lawyer network has a greater amount of clustering than the karate network overall, locally speaking, there are more vertices in the karate network that form many triangles and connected triples compared to the lawyer network. However, social networks in general tend to have higher-than-average transitivity values compared to networks of other types.

1.2 Average Path Length

Average path length (or *average geodesic distance*) is the average of the shortest paths of all distinct pairs of vertices in the network. This is also a commonly used statistic for social networks since average path lengths tend to be much less than what we would expect in these networks than if the edges were placed between vertices randomly (Kolaczyk 2009).

The karate network has an average path length of 2.408 while the lawyer network has an average path length of 2.144. This means that in both cases, any two people in both networks are separated by an average of a little over two people. With Milgram's experiment and the concept of six degrees of separation in mind, we can tell that these are reasonable values indicative of social networks, which are known to have small average path lengths regardless of the number of individuals in the network (Newman 2010).

1.3 Diameter

The *diameter* of a graph is the longest of all the shortest paths between distinct pairs of vertices. This value gives our graph a notion of distance. Unlike average path length, however, diameter can be unpredictable at times since this value can be an outlier compared to other path lengths in the network (Kolaczyk 2009).

The diameters of the karate and lawyer networks are 13 and 5, respectively. In the law firm of 36 people, people are, maximally, only as far as five people away from each other, but for the karate network, even in a group of 34, there exist two club members that do not have a direct relationship with each other and are separated by 12 other members (that is, 13 ties in between).

1.4 Centrality

Centrality assigns a measure of "importance" to each vertex in the network. It is similar to the measures of central tendency seen in elementary statistics. People have proposed different types of centrality, some of which are discussed below. We assume our graph G is undirected (Kolaczyk 2009).

Vertex degree (or *degree centrality*) is perhaps the most common type of centrality. It is not surprising that vertices with higher vertex degrees are considered to be more central to the network than those with lower vertex degrees. In fact, one common characteristic among real-world networks is that their degree distributions tend to have an unusually high number of such vertices. These graphs follow a *power law distribution*. This property indicates that quite a few vertices with particularly high degrees occur fairly frequently (compared to if the degree distribution was a Poisson, for example). The mean of the vertex degrees of all the vertices of the graph yields the *mean* (or *average*) *degree*, often denoted with lowercase c.

Closeness centrality measures how close a vertex is to other vertices based on the inverse of the total distance of the vertex from all others. Closeness centrality is computed as:

$$c_{Cl}(i) = \frac{1}{\sum_{j \in V} d(i,j)},$$

where $dist(i,j)$ is the geodesic distance between the vertices $i, j \in V$. To compare across graphs and with other centrality measures, this measure is often normalized to lie in the interval $[0,1]$ by multiplying by a factor of $N_V - 1$.

Betweenness centrality measures are aimed at summarizing the extent to which a vertex is located between other pairs of vertices. The most commonly used version of betweenness centrality is

$$c_B(i) = \sum_{g \neq h \neq i \in V} \frac{\sigma(g,h|i)}{\sigma(g,h)},$$

where $\sigma(g,h|i)$ is the total number of shortest paths between g and h that pass through i, and $\sigma(g,h) = \sum_{i \in V} \sigma(g,h|i)$. If the shortest paths are unique, $c_B(i)$ just counts the number of shortest paths going through i. This centrality can be normalized by dividing by a factor of $\frac{(N_V-1)(N_V-2)}{2}$.

Eigenvector centrality is based on the idea of "status," "prestige," or "rank;" the more central the neighbors of a vertex are, the more central that vertex itself is. One definition of such a centrality measure is

$$c_{Ei}(i) = \alpha \sum_{\{i,j\} \in E} c_{Ei}(u).$$

The vector $c_{Ei} = (c_{Ei}(1), ..., c_{Ei}(N_V))^T$ is the solution to the eigenvalue problem $\mathbf{A}\mathbf{c}_{Ei} = \alpha^{-1}\mathbf{c}_{Ei}$, where $\mathbf{A}$ is the adjacency matrix for network graph G.

Centrality measures, like the local average transitivity, provide a value for each vertex in the graph. Because of this, we can obtain an overall centrality measure for the graph as a whole if we average the values for each vertex.

Table 1.1: Average centrality measures for the karate and lawyer networks

Network Statistic	Zachary's Karate Club	Lazega's Law Firm
Avg. Degree	4.588	6.389
Avg. Closeness Centrality	0.005	0.007
Avg. Betweenness Centrality	26.194	17.833
Avg. Eigenvector Centrality	0.377	0.458

Table 1.1 above shows the average centrality measures for both the karate and lawyer networks. Because each type of centrality is measuring importance of vertices differently, it comes as no surprise that all of the average centrality values within one network will be different from each other. These average values, however, do not give any indication of which, or how many, vertices in the network are important. Additionally, the use of an average means that the value can be subjected to some

very drastic outliers that could potentially skew its value. However, from Table 1.1, we see that lawyers in the law firm tend to have higher degrees, closeness centrality, and eigenvector centrality on average than the karate club. This means that, on average the lawyers in this firm have a lot of ties in the firm, are very well connected, and have a relatively high rank amongst their peers when compared to those in the karate club. The high betweenness centrality in the karate club indicates that there are a few members (in particular, the dojo's master and the disciple) who tend to have connections with members that do not have connections with each other. These centrality measures make sense since those in the company are probably more likely to know one another than those in the karate club who may only be on close terms with the master.

We will use these seven network statistics listed above in our simulation study. By calculating these values from our observed network, it is possible for us to compare our network with graphs simulated from our graph models.

Chapter 2

Graph Models

A *graph model* takes in fixed parameters that generate a graph that can vary in structure with each iteration. Equivalently, it is also possible to consider a model for a graph as a collection, or *ensemble*,

$$\{\mathbb{P}_\theta(G), G \in \mathcal{G} : \theta \in \Theta\}$$

in which G is a collection or ensemble of possible graphs, P_θ is a *probability distribution* on G (that is, a function that assigns a value for very possible graph that G can be, whose total sums to 1), and θ is a vector of parameters that describe the graphs that G can be, ranging over possible parameters in Θ (Kolaczyk 2009, Newman 2010).[1]

2.1 Erdős-Rényi Model

The *Erdős-Rényi model* (also known as the *Erdős-Rényi-Gilbert model*) is one of the most studied graph models.[2] It is also one of the simplest, as it takes only two parameters. The model $G(N_V, N_E)$, first suggested by Gilbert (1959), takes in N_V, the number of nodes, and N_E the number of edges. Erdős and Rényi (1959, 1960, 1964) considered the model of the form $G(N_V, p)$, where instead of using the number

[1] Many of the explanations and derivations for the Erdős-Rényi and Watts-Strogatz networks here follow from Newman (2010). Much of the explanation about exponential random graph models (ERGMs) follows largely from C. Butts et al. (2015), D. R. Hunter, Handcock, Butts, Goodreau, & Morris (2008), and Jackson (2013).

[2] Many other names for this model exist, such as *Bernoulli model* (or *Bernoulli random graph*) and the *Poisson random graph* due to the properties of its degree distribution, as we will see. In fact, Paul Erdős and Alfréd Rényi were not the first to study or discover this model. According to Newman (2010), the earliest known study of this model is by Ray Solomonoff and Anatol Rapoport in 1951.

of edges, the probability of an edge forming between any pairs of nodes is fixed. Our focus is on the latter model. It is clear that, on average, the $G(N_V, p)$ model would comprise of many more networks than the $G(N_V, N_E)$ model, as the number edges is not fixed. This allows us to consider a larger number of networks with similar network statistics seen in Chapter 1. However, while unlikely, it is possible to obtain a graph with no edges or all possible edges from the $G(N_V, p)$ model (Newman 2010).

Figure 2.1: Examples of graphs generated from the Erdős-Rényi model. Left and Right: $N_V = 10$, $p = 0.25$.

In the $G(N_V, p)$ model, graphs constructed according to this model could potentially look every different from each other. This goes back to the idea that a graph model can be thought of as a probability distribution over an ensemble of networks, and all networks in this particular ensemble have equal probability of being chosen. We immediately see a potential drawback in graphs generated from this model: it places no significance on structures that we may see in our observed social networks–such as having high clustering, cliques, and connected components. However, because of the random placement of the edges across the vertices, values such as average path length are often quite small, even for large graphs (Newman 2010). This is one of many things that characterize social networks and is a primary the motivation for using this model for our analysis (even as a starting point).

The Erdős-Rényi model has other nice properties, some of which we discuss below. Even with only two parameters N_V and p on hand, we can still create formulas that

allow us to calculate various network statistics, such as average degree and clustering coefficient, and describe the model's degree distribution.

As indicated earlier, the $G(N_V, p)$ model is the collection of simple graphs with exactly N_V vertices, meaning that a particular simple graph g with exactly N_V vertices has probability

$$P(G = g) = p^{N_E}(1-p)^{\binom{N_V}{2}-N_E}$$

of being picked. For precisely N_E edges, there are $\binom{\binom{N_V}{2}}{N_E}$ ways to arrange the N_E edges among the $\binom{N_V}{2}$ possible edges. Thus, the total probability of a random graph G with N_E edges and N_V vertices is

$$P(G) = \binom{\binom{N_V}{2}}{N_E} p^{N_E}(1-p)^{\binom{N_V}{2}-N_E}$$

This, however, is simply a binomial distribution, where we have some probability of success (p), two possible outcomes (edge formation or no edge formation), a finite number of trials ($\binom{N_V}{2}$ distinct edges), and $\binom{\binom{N_V}{2}}{N_E}$ different ways in which the outcomes can be arranged. Using this, the mean value of N_E for the model is then a weighted average. It is the sum of the products of every possible number of edges N_E and the total probability that a graph with N_V vertices and N_E edges appears, $P(N_E)$. However, because we know the probability of an edge forming, the mean number of edges would equal to the product of the total number of possible vertices $\binom{N_V}{2}$ and the probability p. This makes sense as we can expect that $100p$ percent of the possible edges in the graph to actually have edges. Thus, the mean number of edges $\langle N_E \rangle$ for a random graph G is

$$\langle N_E \rangle = \sum_{N_E=0}^{\binom{N_V}{2}} N_E P(N_E) = \binom{N_V}{2} p$$

For a graph with exactly N_E edges, it is easy to see that the mean degree is $\frac{2N_E}{N_V}$. The factor of 2 allows an edge to be counted as part of the degree for each of the pair vertices that it connects. By taking a similar weighted average, where we sum the products of the average degree of a graph with N_E edges with the total probability of a graph with N_E edges being chosen, we can get the average degree, denoted c or $\langle N_V \rangle$, for this model. Using the formula for the mean number of edges above, we can

simplify this weighted average as

$$\begin{aligned} c &= \langle N_V \rangle \\ &= \sum_{N_E=0}^{\binom{N_V}{2}} \frac{2N_E}{N_V} P(N_E) \\ &= \frac{2}{N_V} \binom{N_V}{2} p \\ &= (N_V - 1)p \end{aligned}$$

The final expression in this equation makes sense; for any vertex on the random graph, we would expect $100p$ percent of the other $N_V - 1$ vertices to be connected to it (Newman 2010).

We can also describe the degree distribution of the $G(N_V, p)$ model. We show it is a binomial distribution, and that it actually converges to the Poisson distribution as the number of vertices N_V increases infinitely. Observe that the probability, p_k, of a vertex connected to k other vertices (without loops) is

$$p_k = \binom{N_V - 1}{k} p^k (1-p)^{N_V - 1 - k},$$

which is a binomial distribution (Newman 2010).

Consider the equation for the mean number of edges: $c = (N_V - 1)p$. Rewriting this for p gives, $p = \frac{c}{N_V - 1}$. This tells us that as the number of nodes increases, the probability p will decrease to 0 indefinitely. Using, this, we can rewrite the $(1-p)^{N_V - 1 - k}$ factor of the equation for p_k above as the solution to the limit as $N_V \to \infty$, as shown below.

$$\begin{aligned} (1-p)^{N_V - 1 - k} &= e^{ln((1-p)^{N_V - 1 - k})} \\ &= e^{N_V - 1 - k} ln(1 - \frac{c}{N_V - 1}) \\ &\simeq e^{-(N_V - 1 - k)(\frac{c}{N_V - 1})} \\ &\simeq e^{-c}, \end{aligned}$$

where we have expanded the natural logarithm as a Taylor series. These approximations become more exact as $n \to \infty$ (Newman 2010).

Similarly, we can take the limit as $n \to \infty$ for the $\binom{N_V-1}{k}$ factor of the p_k probability.

$$\binom{N_V-1}{k} = \frac{(N_V-1)!}{(N_V-1-k)!k!} = \frac{(N_V-1)^k}{k!}.$$

Combining these two equations and $p = \frac{c}{N_V-1}$, the probability p_k becomes

$$\begin{aligned} p_k &= \binom{N_V-1}{k} p^k (1-p)^{N_V-1-k} \\ &= \frac{(N_V-1)^k}{k!} \left(\frac{c}{N_V-1}\right)^k e^{-c} \\ &= e^{-c}\frac{c^k}{k!} \end{aligned}$$

as $N_V \to \infty$. This equation is simply the Poisson distribution (Newman 2010).

To use this model in our simulation study, we simply take the number of nodes, N_V, and the probability of a link forming, which equals the number of observed edges divided by the number of possible edges, that is, $\frac{N_E}{\binom{N_V}{2}}$. While we have discussed issues of using the Erdős-Rényi model earlier–such as clustering–the random placement of nodes serves as a good starting point. The idea is to see if we can find a model that does at least better than the random assignment of edges to the vertices.

2.2 Watts-Strogatz Model

Watts & Strogatz (1998) noticed that many networks in real life have high levels of clustering and only require short average path lengths between nodes with high degree. In the Erdős-Rényi model, simulated graphs of parable magnitude tend to have smaller-than-expected clustering coefficients. As we had seen in Milgram's experiment, networks among people are not quite has disconnected as we had once thought. If any two people in the world are, on average, ever seperated by six people, there is some notion of a "small-world." In the Watts-Strogatz model, we start with several parameters: N_V vertices, which are arranged in a circular fashion, which we will call the "circular model", the number of beginning neighbors for each node, r, and the rewiring probability, p, of an edge being moved to another pair of vertices. A variant of this model, which we will use instead, sets the rewiring probability to be zero and utilizes a shortcut probability that added edges randomly to the starting circular model that connect vertices that did not have an edge before (Newman 2010).

In a circular model with 10 nodes, each having two neighbors, the clustering

coefficient is relatively high at exactly 0.5. Diameter and average path length are nontrivial, and can be rather high as well, at 3 and 1.667, respectively.

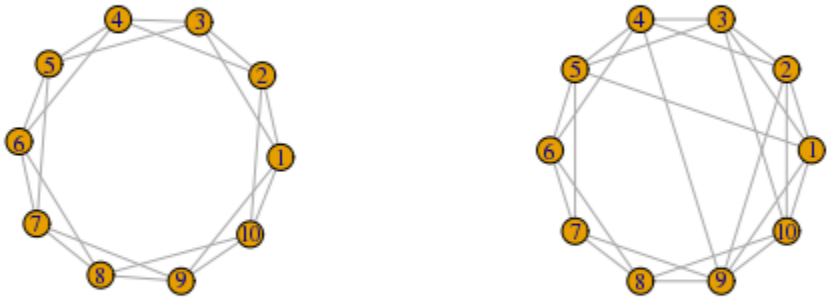

Figure 2.2: Examples of graphs generated from the Watts-Strogatz model. Both: $N_V = 10$, $r = 2$. Left: $N_E = 20$. Right: $N_E = 24$.

By adding edges (instead of simply rewiring), the diameter and average path length tend to decrease. This makes sense because this rewiring creates random shortcuts in the network framework. In the right-hand graph of Figure 3.2, which is the left-hand graph after adding 4 random edges, we see that the diameter and average path length have decreased to 2, and 1.467, respectively.

Like the Erdős-Rényi model, the Watts-Strogatz model also has very nice properties. Not only is it able to achieve a high clustering coefficient due to the inherent circular structure–which was definitely a drawback seen in the previous model–but adding in shortcuts also allow for average path lengths to be small as well. Other properties, such as degree distributions and counts of triangle and triples, and clustering coefficients, are shown below.

Given a circular model with N_V vertices and number of starting neighbors r, a triangle on the model is represented by traveling across two edges in the same direction from a starting vertex, followed by one edge back to that vertex. With r neighbors, there are $\binom{\frac{r}{2}}{2} = \frac{1}{4}N_V r\left(\frac{1}{2}r - 1\right)$ ways to choose the two starting edges, and then N_V ways to take the last step back, making for a total of $\frac{1}{2}N_V r(r-1)$ triangles. The number of triples for each vertex is $\binom{r}{2} = \frac{1}{2}r(r-1)$ since any two

unique edges from a vertex will connect three vertices. This, the total number of triples is $\binom{r}{2} = \frac{1}{2}N_V r(r-1)$.

Combining these results, we have the clustering coefficient of the circular model.

$$\begin{aligned} C &= \frac{(\text{number of pairs of neighbors of } i \text{ that are connected})}{(\text{number of pairs of neighbors of } i)} \\ &= \frac{\frac{1}{4}N_V r\left(\frac{1}{2}r-1\right)\times 3}{\frac{1}{2}N_V r(r-1)} \\ &= \frac{3(r-2)}{4(r-1)}. \end{aligned}$$

There is a minimum clustering coefficient of 0 if $r = 2$ and a maximum of $\frac{3}{4}$ as $r \to \infty$. Note that this value is also independent of N_V (Newman 2010).

The starting circular model has $\frac{1}{2}N_V r$ non-shortcut edges. It is possible to add in shortcut edges to pairs of vertices that have yet to be connected with some probability p. Thus, on average, we add approximately $\frac{1}{2}N_V rp$ shortcuts–which mean there are $N_V rp$ ends created by the shortcuts. The probability p_s that a number of shortcuts, s, is attached to any one vertex has mean rp follows a Poisson distribution

$$p_s = e^{-rp}\frac{(rp)^s}{s!}.$$

Using the fact that the total degree of a vertex k is the sum of the starting neighbors r and the number of starting shortcuts s, that is $k = r + s$, we can combine this with the equation above to get the degree distribution for the Watts-Strogaz model:

$$p_k = e^{-rp}\frac{(rp)^{k-r}}{k-r},$$

for $k \geq r$ and $p_k = 0$ if $k < r$ (Newman 2010).

While it can be difficult to precisely calculate the clustering coefficient, we can calculate a limit for the number of triangles and triples. The number of triangles in the circular model remain unchanged, and the number of triangles created by the shortcuts is negligible compared to the number that the circle originally contained for large graphs. This is because the number of paths of length two that are completed by shortcuts to form triangles remains constant for large N_V. The argument is similar for new triangles formed from two or three shortcuts as well. Thus, the leading term for the number of triangles in the Watts-Strogatz model is $\frac{1}{4}N_V r\left(\frac{1}{2}r-1\right)$, which is equal to that of the circular model. For the number of triples, we consider the $\frac{1}{2}N_V r(r-1)$ triples from the circular structure itself, the $\frac{1}{2}N_V rp$ shortcuts on average for each

vertex times r neighbors times 2 ends for each shortcut equals $N_V r^2 p$ number of triples created by a shortcut combining with an edge in the circle, and the $\frac{1}{2}r^2p^2$ expected number of connected triples centered at a vertex times N_V vertices equals $\frac{1}{2}N_V r^2 p^2$. Thus,

$$\begin{aligned} C &= \frac{(\text{number of triangles}) \times 3}{\text{number of connected triples}} \\ &= \frac{\frac{1}{4}N_V r\left(\frac{1}{2}r - 1\right) \times 3}{\frac{1}{2}N_V r(r-1) + N_V r^2 p + \frac{1}{2}N_V r^2 p^2} \\ &= \frac{3(r-2)}{4(r-1) + 8rp + 4rp^2}. \end{aligned}$$

When $p = 0$, the clustering coefficient is equal to that of the circular model. And as p increases, the clustering coefficient will decrease. It reaches a minimum value of $C = \frac{3(r-2)}{(4c-1)}$ when $p = 1$ (Newman 2010).

Calculating other parameters, such as average path length, become much more difficult. In fact, no exact expressions for average path length have been discovered; only approximations. One special property of path lengths in this model, however, is that they *scale* with the model parameters. This means that path length is calculated in terms of some function that does not depend on any parameters. For more information on scaling, see Newman (2010).

While it is entirely possible that a network could potentially have some kind of rewiring process, it is less likely that this is true in the context of social networks. In other words, ties between people do not usually disappear and randomly reappear between others. However, we can still make use of the possibility that a social network can start with a circular model with some initial number of neighbors. New edges to create shortcuts can be added until we get a graph of similar order and size.

In the case where we start with circular model of $r = 1$ neighbor, however, this is no different from the Erdős-Rényi model. This is because the circular model with only one neighbor is very similar to a random assignment of edges (but with the restriction that each node only has a degree of 1). Additionally, depending on the number of edges that need to be added to the network, the starting circle may not have a large effect on the overall structure of the graph generated.

2.3 Exponential Random Graph Models (ERGMs)

Exponential random graph models (ERGMs) are a class of models that can also be used to generate probability distributions for a set of random graphs. ERGMs have

enormous flexibility in construction since they allow any combination of parameters from a given data set to be used in constructing the model. We are not only able to simulate additional graphs from the underlying probability distribution like in the Erdős-Rényi and Watts-Strogatz models, but we are also able to obtain maximum-likelihood estimates of the model for the data set and conduct goodness-of-fit tests for model assessment (Hunter et al. 2008a).

The general form for an ERGM is as follows:

$$P_{\theta,\mathcal{Y}}(\mathbf{Y} = \mathbf{y}) = \frac{exp(\theta^T \mathbf{g(y)})}{\kappa(\theta, \mathcal{Y})}, \mathbf{y} \in \mathcal{Y},$$

where $\mathbf{Y}$ is the random variable representing the adjacency matrix of a graph and $\mathbf{y}$ is the particular adjacency matrix we observe. $\mathbf{g(y)}$ is the vector of model statistics for $\mathbf{y}$, θ is the vector of coefficients for those statistics, and $\kappa(\theta, \mathcal{Y})$ is the quantity in the numerator summed over all possible networks. The $\theta^T \mathbf{g(y)}$ is a polynomial of coefficients θ and statistics $g(y)$. This statistics could be things such as the number of edges or triangles of a graph in question. The θ parameters are interpreted as the log odds of an individual tie conditional on all the others (Hunter et al. 2008a, Butts et al. 2015, Jackson 2013).

The denominator

$$\kappa(\theta, \mathcal{Y}) = \sum_{z \in \mathcal{Y}} exp(\theta^T \mathbf{g(y)})$$

serves as a normalizing factor so that $P_{\theta,\mathcal{Y}}(\mathbf{Y} = \mathbf{y})$ is a proper probability distribution (that is, the values that are output are between 0 and 1). It is also possible to consider actual graphs themselves instead of their adjacency matrices. This means we can replace $\mathbf{y}$ with an observed graph g, $\mathbf{Y}$ with a random graph G, and $\mathcal{Y}$ with the ensemble of graphs $\mathcal{G}$.

As an example, consider a graph g with the following statistics: $L(g)$ is the number of edges in the graph, $T(g)$ is the number of triangles, and $K(g)$ is the number of k-stars.

Using an ERGM to fit g would result in the following probability distribution:

$$P_{\theta,\mathcal{G}}(g) = \frac{exp(\theta_L L(g) + \theta_T T(g) + \theta_K K(g))}{\sum_{g' \in \mathcal{G}} exp(\theta_L L(g) + \theta_T T(g) + \theta_K K(g))}, g \in \mathcal{G}.$$

The statistics used to build an ERGM are either *dyad independent* or *dyad dependent* terms (*dyad* meaning *edge*). Dyad independent terms (such as edges between nodes, $L(g)$) imply no dependence between the edges of other nodes. Thus, the presence or

absence of an edge between vertices does not affect and is not affected by the presence and absence of other edges. On the other hand, dyad dependent terms (such as triangles, $T(g)$ and k-stars $K(g)$) do imply dependence among edges in the graph; the absence of a link does affect the triangle and k-star counts. In order to calculate the θ parameters, ERGMs implement an estimation algorithm. If the model only contains dyad independent terms, the algorithm is simply a maximum log-likelihood estimation, in which we find the value that maximizes the log-probability model. However, if the ERGM contains dyad dependent terms, it can be impossible to obtain a closed form of the probability model and estimate the parameters by maximizing the log-likelihood function. Instead, we turn to a new method of estimation, called *Markov chain Monte Carlo (MCMC)*. In short, this means we estimate the parameters via simulation rather than precise calculation (Butts et al. 2015). See Gilks, Richardson, & Spiegelhalter (1995) for more information about MCMC.

2.3.1 Deriving the Erdős-Rényi Model from ERGMs

The Erdős-Rényi and Watts-Strogatz models are actually special cases of ERGMs. Here, we will show that the Erdős-Rényi model can be derived from an ERGM that only takes into acount the number of edges in a graph. Suppose we have a particular graph g and the only statistic we have is $L(G)$, the number of edges in g. Our probability model is thus

$$P_{\theta,\mathcal{G}}(g) = \frac{exp(\theta_L L(g))}{\sum_{g' \in \mathcal{G}} exp(\theta_L L(g'))}, g \in \mathcal{G}.$$

Consider the probability distribution for a particular graph g with N_E edges again (from the Erdős-Rényi model). Using the fact that $N_E = L(g)$, taking the equation as

a power of base e, we get the following:

$$\begin{aligned}
P(g) &= p^{N_E}(1-p)^{\left(\binom{N_V}{2}-N_E\right)} \\
&= p^{L(g)}(1-p)^{\left(\frac{N_V(N_V-1)}{2}-L(g)\right)} \\
&= \left(\frac{p}{1-p}\right)^{L(g)}(1-p)^{\frac{N_V(N_V-1)}{2}} \\
&= exp\left(L(g)log\left(\frac{p}{1-p}\right) - \frac{N_V(N_V-1)}{2}log\left(\frac{p}{1-p}\right)\right) \\
&= exp(\theta_L L(g) - c) \\
&= \frac{exp(\theta_L L(g))}{exp(c)},
\end{aligned}$$

where $c = \frac{N_V(N_V-1)}{2}ln\left(\frac{p}{1-p}\right)$ is the normalizing constant $\kappa(\theta, \mathcal{Y})$ seen in the general form of an ERGM and exactly the denominator $\sum_{g' \in \mathcal{G}} exp(\theta_L L(g'))$ seen in the ERGM that fits a graph g with only the edge statistic $L(g)$ (Hunter et al. 2008a & Jackson 2013).

Chapter 3

Simulation Study

We will use a subset of the Facebook network for our analysis. We fit graph models to the observed network and assess each model's accuracy by comparing the network statistics of the observed network to the graphs generated by the models.

3.1 Description of the Dataset

This Facebook network was compiled by Stanford University and was accessed through the Stanford Large Network Data Collection (SNAP). For more information, please see Jure Leskovec & Krevl (2014). Our subset of the Facebook network was presented as an edge list. The social network is one giant component composed of 4039 vertices and 88234 edges, which represent 4039 anonymous users and the 88234 connections between them, respectively.[1] This network is simple and undirected, and we believe that the network consists of established, mutual friendships. The first column of values in Table 3.1 below describes some characteristics of this network. We use the `igraph`, `sna`, `network`, `ergm`, and `statnet` packages in this study.[2]

[1]Some of our network statistics that we calculated in R did not match those shown in the SNAP website. For example, we calculated the diameter and transitivity of this Facebook component to be 17 and 0.617, respectively. However, the SNAP website reports the diameter and "average clustering coefficient" to be 8 and 0.6055, respectively. However, other values, such as vertex and edge count, as well as the number of triangles, are equal. While SNAP's algorithm used to calculate their network statistics is unclear, for the purposes of this simulation study, we will be using the values that we calculated from the Facebook component to compare with those generated from our graph models.

[2]See Csárdi & Nepusz (2006), C. T. Butts & others (2008), Butts (2015), Butts (2008), Handcock et al. (2017), D. R. Hunter et al. (2008), Handcock et al. (2016), Handcock, Hunter, Butts, Goodreau, & Morris (2008), and Bojanowski (2015) for more details about these packages.

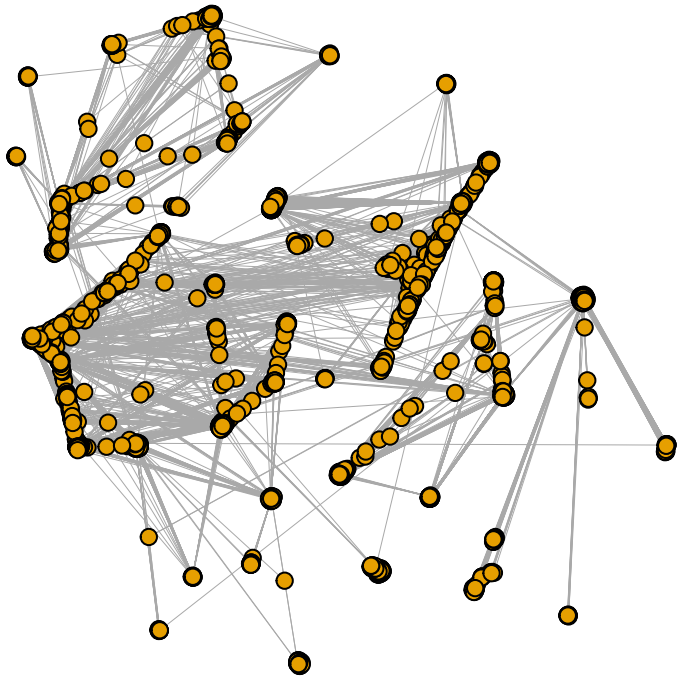

Figure 3.1: Overview of our component of the Facebook network

3.2 Generating Random Graphs

For each model, we create graphs of parable magnitude using some of the information calculated from the our Facebook network. We compare selected statistical network statistics of the Facebook network with that of the graphs we generate to see if the observed network and the simulated graphs have similar characteristics. We explain the process of generating one random graph for each model in the sections below. We simulate 1000 random graphs for each model and record the statistics such as transitivity, diameter, and average centrality values. Because we have 1000 values (one for each random graph for each model), we could potentially see a distribution of these values or even just take the average of these values. This means that for statistics such as *average* degree, we were interested in recording an average of the average values. The first column of Section A shows the network statistics we were interested in and the average statistics recorded under each model along with the standard deviation.

3.2.1 Using the Erdős-Rényi Model

Recall that the Erdős-Rényi model takes in only two parameters, which are the number of vertices, N_V, and the fixed probability of edge forming between any two different vertices, p. The number of vertices is already given as 4039, but to find the probability, we estimate this by taking the number of observed edges and dividing it by the number of possible edges. For a graph G, this estimated probability, $\hat{p}$, is

$$\hat{p} = \frac{N_E}{\binom{N_V}{2}},$$

where N_E is the number of edges in G and N_V is the number of vertices in G. Our estimated probability $\hat{p}$ for this Facebook network is then $\frac{88234}{\binom{4039}{2}}$, or approximately 0.011 Thus, in order to simulate one graph from this model, we must look at every possible edge among the 4039 vertices and determine if an edge will form based on the given probability.

3.2.2 Using the Watts-Strogatz Model

Recall that the Watts-Strogatz model takes in the following parameters: the number of vertices N_V, the number of starting neighbors r, and the probability of an edge to be rewired p. However, instead of rewiring our edges with a fixed probability, we

will instead randomly add edges until we have approximately the same number as our observed network. To do this, we start with a lattice with 4039 vertices and assign a number of edges to the vertices equal to the smallest degree seen in our Facebook network (which we find to be 1). This is our starting number of neighbors for each vertex. In this situation, $p = 0$. We then randomly add edges equal to the difference between the our starting lattice and the observed network. That is, we add $88234 - 4039 = 84195$ edges to our lattice. Finally, we simplify our simulated graph to eliminate multi-edges and loops. Since there are $\binom{4039}{2}$ edges to choose from, intuitively, we can see that having many multi-edges or loops will be infrequent. This makes up one simulated graph from the model.

3.2.3 Using ERGMs

We consider four different ERGMs–which we label as ERGM 1a, ERGM 2a, ERGM 2b, and ERGM 3a. ERGM 1a takes in only the most basic parameter: edges. In Chapter 2, we have already shown its relationship to the Erdős-Rényi model. ERGM 2a and 2b both take in two parameters: edges and triangles for 2a, and edges and k-stars (of size 3) for 2b. Lastly, ERGM 3a, takes in all three parameters: edges, triangles, and k-stars (of size 3). Similar to linear regression, adding in more parameters will allow the ERGM to fit more closely to the observed network but at the cost of greater complexity. We do not consider models that only consist of just the triangle parameter or just the k-star parameter. This is because information about k-stars and triangles are based off on some knowledge about edges anyway. After fitting ERGMs with the desired parameters (via maximum likelihood or MCMC), we begin simulating random graphs from each model. For each random graph, we calculate the networks statistics of interest to us.

Table 3.1: Comparisons of network statistics among graphs simulated from the Erdős-Rényi and Watts-Strogatz models and the observed network

Network Statistic	Observed	Erdős-Rényi	Watts-Strogatz
Transitivity	0.617	0.0108 ± 0.0001	0.0107 ± 9.135e-05
Average Path Length	4.338	2.606 ± 0.002	2.6093 ± 0.0002
Diameter	17	3.96 ± 0.21	3.95 ± 0.22
Avg. Degree	43.691	43.69 ± 0.14	43.45 ± 0.01
Avg. Betweenness Centrality	2072.642	3242 ± 4	3249.2 ± 0.4
Avg. Closeness Centrality	8.881e-08	9.507e-05 ± 7.230e-08	9.494e-05 ± 7.319e-09
Avg. Eigenvector Centrality	0.040	0.620 ± 0.022	0.6235 ± 0.0227

Table 3.2: Comparisons of network statistics among graphs simulated from ERGMs (1a: edges and 2a: edges and triangles) and the observed network

Network Statistic	Observed	ERGM1a	ERGM2a
Transitivity	0.617	0.3696 ± 0.0012	0.4823 ± 0.0020
Average Path Length	4.338	2.885 ± 0.004	3.052 ± 0.0086
Diameter	17	5.216 ± 0.412	6.098 ± 0.035
Avg. Degree	43.691	43.66 ± 0.052	44.54 ± 0.03
Avg. Betweenness Centrality	2072.642	3805 ± 9	4140 ± 18
Avg. Closeness Centrality	8.881e-08	8.286e-05 ± 8.353e-06	6.008e-05 ± 1.514e-05
Avg. Eigenvector Centrality	0.040	0.0417 ± 0.0008	0.0410 ± 3.577e-05

Table 3.3: Comparisons of network statistics among graphs simulated from ERGMs (2b: edges and k-stars(3) and 3a: edges, triangles, and k-stars(3)) and the observed network

Network Statistic	Observed	ERGM2b	ERGM3a
Transitivity	0.617	0.3787 ± 0.0013	0.4891 ± 0.0020
Average Path Length	4.338	2.851 ± 0.003	3.0562 ± 0.0079
Diameter	17	5.095 ± 0.293	6.161 ± 0.3677
Avg. Degree	43.691	44.06 ± 0.057	44.52 ± 0.034
Avg. Betweenness Centrality	2072.642	3736 ± 6	4148 ± 16
Avg. Closeness Centrality	8.881e-08	8.558e-05 ± 6.025e-06	5.875e-05 ± 1.5035e-05
Avg. Eigenvector Centrality	0.040	0.0392 ± 0.0002	0.0410 ± 3.451e-05

Based on Section A, we see that the estimated average statistics for the Erdős-Rényi and Watts-Strogatz models are very similar to each other but are drastically different from many of the observed values of our Facebook network. This means that while the graphs generated from each model look very similar to each other, they look drastically different from our observed Facebook network. This comes as no surprise since the methods used to construct graphs under both models are very likely to generate almost the same number of vertices. We even see this in our simulation study; in the fourth row of Section A, the average degree estimate for both models was within one degree of the other. Additionally, in both constructions, we are adding the edges randomly, so they are almost always spread evenly throughout the 4039 vertices. This means that additional features of the observed network, such as large clusters or degrees with high centralities, are likely to be missed by graphs generated from these two models. This is why the transitivity, average path length, and diameter for both models are smaller than what was actually observed.

Similarly, due to the random disperal of edges throughout the network, there is little control in deciding which vertices and how many are deemd to be important, making the average centrality measures for the random graphs much larger than those seen in the observed model. The only statistic whose value was actually close to, if not equal to, the observed value was the average degree. However, this makes sense since we fixed the probability p of a link forming in the Erdős-Rényi model (which

of is directly proportional to the average degree) and because we ensured that every graph generated from the Watts-Strogatz model has about the same number of edges as the observed network. Thus, we have shown through this simulation study that the Erdős-Rényi and Watts-Strogatz models are not very accurate in modeling our Facebook network for these particular network statistics.

As Section A and Section A suggest, it appears that the random graphs generated from ERGMs provide very good estimates of average degree and average eigenvector centrality. Estimates of other network statistics, however, are quite different from what is actually observed. Not surprisingly, many of the average network statistics of the random graphs generated by the ERGMs get closer to the observed value as the number of parameters increase. This includes an overall increase in transitivity, average path length, and diameter and an overall decrease of average closeness centrality as the number of parameters increase from one to three. Average betweenness centrality, however, continue to drift farther away from the observed value as the number of parameters increase.

The fact that ERGM 2a (edges and triangles) has such similar values to ERGM 3a (edges, triangles, and k-stars of size 3) in spite the difference in the number of parameters suggests that the k-star parameter may be redundant. This is also further suggested by the fact that ERGM 2b (edges and k-stars of size 3) and ERGM 1a (edges) are also very similar in values as well. While this simulation study suggests that the right combinations of parameters could potentially allow for an ERGM to accurately model an observed network, we conclude that the four ERGMs used in this study do not do a very good job of modeling our Facebook network for these particular network statistics as well.

Chapter 4

Conclusion

In this thesis, we have provided a brief overview of networks, including network statistics and some graph models. We also applied certain graph models–such as the Erdős-Rényi and Watts-Strogatz models along with various ERGMs with different combinations of parameters–and fitted them to our component of the Facebook network. By letting the graph models take in certain parameters of our Facebook network, we generated graphs of parable magnitude. We compared the network statistics between these generated graphs and our target network to measure each model's accuracy. This simulation study suggested that that none of the models that we examined did a good job of modeling the Facebook network for our chosen network statistics.

However, this simulation study only scratches the surface of what network science has to offer. Firstly, we can simply change our approach with Erdős-Rényi, Watts-Strogatz, and exponential random graph models we have so far. In applying the Erdős-Rényi model, we had used the $G(N_V, p)$ model. For the same of comparison, we could also explore how well the $G(N_V, N_E)$ model fits the Facebook network. Our use of the Watts-Strogatz model began with a circular lattice with only one neighbor. Instead of specifying a probability p of a shortcut forming, we simply added "enough" edges so that they are similar to the observed Facebook network. While the methodology is logical, the results of the simulation study become very similar to that of the Erdős-Rényi model and we ultimately lose the characteristic of high transitivity that the Watts-Strogatz model is known for. Future applications of the Watts-Strogatz model should include a logical methodology to make better use of the circular model and shortcut probability p. ERGMs seemed to have the most promise–as we increase the number of parameters, we see that the estimates of many network statistics tend to approach the observed value. This suggests that trying other ERGMs with more parameters and different combinations of them could potentially improve the results

of this study. See C. Butts et al. (2015) for a list of possible ERGM parameters to choose from.

Additonally, we can bring in other social networks into this study. Could underlying undirected graphs of directed networks be a better fit for the models here? We could potentially analyze a different dataset, such as Twitter, in which we look at the underlying graph of directed followers instead of mutal friendships. Could network size be a factor as well? The Facebook component we are currently observing could perhaps be too small and cause issues of variance. It is also possible for us to test other network statistics. Without providng the definitions, we can consider values such as *reciprocity*, *assortativity*, or even other kinds of centrality measures. See Newman (2010) for more details.

It is also possible for us to choose other models to test our current Facebook network on. In fact, many other graphs models exist beyond what was presented here. Because social networks exhibit certain characteristics different from other kinds of networks, it comes as no surprise that there have been other models that are specifically tailored to its study. See Toivonen et al. (2009) for a study of *network evolution models* and *nodal attribute models*, two specific types of social network models. Also, see Jurij Leskovec, Chakrabarti, Kleinberg, & Faloutsos (2005) and Jure Leskovec, Chakrabarti, Kleinberg, Faloutsos, & Ghahramani (2010) about more recent developments in network science. They introduce models called *Kronecker graphs* that have the benefit of only requiring a few parameters to fit to a target network in linear time.

A successful simulation study has many implications and can benefit not only statistics and network science, but in just about all other disciplines as well.

Appendix A

R/R-Markdown Code

This first appendix includes all of the R chunks of code used throughout the document (using the `include = FALSE` chunk tag) to help with readability and/or setup. Some code chunks were shown in the main body of the chapters for clarity and illustration.

In the main Rmd file:

This is the setup file for the template.

```
# This chunk ensures that the acstats package is
# installed and loaded. This acstats package includes
# the template files for the thesis and also two functions
# used for labeling and referencing
if(!require(devtools))
  install.packages("devtools", repos = "http://cran.rstudio.com")
if(!require(acstats)){
  library(devtools)
  devtools::install_github("Amherst-Statistics/acstats")
}
library(acstats)
```

The following are the packages used throughout this work. In particular, the `igraph`, `network`, `statnet`, and `ergm` packages were heavily used throughout all the chapters.

```
library(sand)
library(igraph)
library(network)
library(sna)
library(statnet)
library(ergm)
library(xtable)
```

```
options(xtable.comment = FALSE)
options(digits = 4)
```

In Introduction

We use the `karate` and `lawyer` data sets as examples to illustrate our discussion of networks.

```
data(karate)
data(lazega)
```

The following code chunk generates a visual representing Zachary's karate club network. See Figure 1.

```
plot(karate)
```

The following code chunk generates a visual representing Lazega's law firm network. See Figure 2.

```
# Office location indicated by color.
colbar <- c("red", "dodgerblue", "goldenrod")
v.colors <- colbar[V(lazega)$Office]
# Type of practice indicated by vertex shape.
v.shapes <- c("circle", "square")[V(lazega)$Practice]
# Vertex size proportional to years with firm.
v.size <- 3.5*sqrt(V(lazega)$Years)
# Label vertices according to seniority.
v.label <- V(lazega)$Seniority
# Reproducible layout.
set.seed(42)
l <- layout.fruchterman.reingold(lazega)
plot(lazega, layout=l, vertex.color=v.colors,
     vertex.shape=v.shapes, vertex.size=v.size,
     vertex.label=v.label)
```

In Chapter 1

Below are the centrality calculations for the karate club and law firm networks. Results are shown in Table 1.1.

```
mean(igraph::degree(karate))
#average degree centrality: 4.588235
mean(igraph::closeness(karate))
#average closeness centrality: 0.005450958
```

```
mean(igraph::betweenness(karate))
#average betweenness centrality: 26.19363
mean(igraph::eigen_centrality(karate)$vector)
#average eigenvector centrality 0.3772814

mean(igraph::degree(lazega))
#average degree centrality: 6.388889
mean(igraph::closeness(lazega))
#average closeness centrality: 0.00670515
mean(igraph::betweenness(lazega))
#average betweenness centrality: 17.83333
mean(igraph::eigen_centrality(lazega)$vector)
#average eigenvector centrality 0.4578719
```

In Chapter 2

The code chunk below generates two `igraph` objects from the Erdős-Rényi model with specified parameters–the number of vertices, N_V, is 10 and probability p of edge formation is 0.25. See Figure 2.1.

```
set.seed(499)

g1.er <- erdos.renyi.game(n = 10, p = 0.25)
g2.er <- erdos.renyi.game(n = 10, p = 0.25)
```

The following code plots the two graphs simulated from the Erdős-Rényi model.

```
par(mfrow=c(1,2))
plot(g1.er, vertex.size=20, vertex.label.cex = 0.75)
plot(g2.er, vertex.size=20, vertex.label.cex = 0.75)
```

This generates two `igraph` objects from the Watts-Strogatz model with specified parameters–the number of vertices, N_V, is 10, number of neighbors, r, is 2. The left graph has $N_E = 20$ edges the right graph as $N_E = 24$ edges (that is, 4 edges were randomly added to the circular model). See Figure 2.2.

```
set.seed(499)

g1.ws <- watts.strogatz.game(dim = 1, size = 10, nei = 2, p = 0)

g2.ws <- watts.strogatz.game(dim = 1, size = 10, nei = 2, p = 0)
# the effect of adding one edge to the circular arrangement
randomedgepairs <- sample(1:10, 8, replace=TRUE)
g2.ws <- simplify(add_edges(g2.ws, randomedgepairs))
```

```
diameter(g1.ws)
average.path.length(g1.ws)
transitivity(g1.ws, type = "localaverage")
diameter(g2.ws)
average.path.length(g2.ws)
transitivity(g2.ws, type = "localaverage")
```

The following code plots the two graphs simulated from the Watts-Strogatz model.

```
par(mfrow=c(1,2))
plot(g1.ws, vertex.size = 20, vertex.label.cex = 0.75)
plot(g2.ws, vertex.size = 20, vertex.label.cex = 0.75,
     layout=layout_in_circle)
```

In Chapter 3

We load in our Facebook data set, that is, our observed network. It is a simple, undiected, connected component of the entire Facebook network in the form of an edge list. We convert to an `igraph` object in order to calculate certain network statistics.

```
setwd("~/STAT495-Lee/LeeThesis/data")

#edge list
facebookcombined <- read.table(gzfile("facebook_combined.txt.gz"),
                               header = F)

#igraph object
fbel <- graph.data.frame(facebookcombined)
```

Below are some values that characterize the observed Facebook network, including some descriptive network statistics.

```
length(unique(c(facebookcombined$V2, facebookcombined$V1)));
vcount(fbel)
#number of vertices: 4039
nrow(facebookcombined); ecount(fbel)
#number of edges: 88234
sum(count_triangles(fbel))/3
#number of unique triangles (up to ordering): 1612010

transitivity(fbel, type="localaverage") #0.6170038
diameter(fbel) #17
average.path.length(fbel) #4.337744
```

```
mean(igraph::degree(fbel))
#average degree centrality: 43.69101
mean(igraph::betweenness(fbel))
#average betweenness centrality: 2072.642
mean(igraph::closeness(fbel))
#average closeness centrality: 8.881448e-08
mean(igraph::eigen_centrality(fbel)$vector)
#average eigenvector centrality 0.04047316
```

The code below plots the entire component of the Facebook that SNAP has provided. See Figure 3.1.

```
plot(fbel,
     edge.arrow.size = 0,
     edge.width = 0.05,
     vertex.label = NA,
     vertex.size = 5)
```

The following code describes the data simulation process using the Erdős-Rényi model. We create seven empty vectors and fill the i^{th} term of each vector by running through a for loop that calculates seven network statisitcs for the i^{th} random graph generated from the model. See results in .

```
set.seed(499)

numsim <- 1000

g.er.fbel.ecount <- rep(NA, numsim)
g.er.fbel.coef <- rep(NA, numsim)
g.er.fbel.apl <- rep(NA, numsim)
g.er.fbel.dia <- rep(NA, numsim)
g.er.fbel.avgdeg <- rep(NA, numsim)
g.er.fbel.avgbtwcen <- rep(NA, numsim)
g.er.fbel.avgclocen <- rep(NA, numsim)
g.er.fbel.avgeigveccen <- rep(NA, numsim)

#probability used in for-loop below
p = ecount(fbel)/choose(vcount(fbel), 2)

for (i in 1:numsim) {
  # n = number of vertices, p = probability of a link
  #p is caluclated as number of edges over number of possible edges
  #p is then (4039 choose 2) as calculated above
  #igraph object
```

```
  g.er.fbel <- erdos.renyi.game(n = vcount(fbel), p)

  #observe the network statistics for this simulated graph
  g.er.fbel.ecount[i] <-
    ecount(g.er.fbel)
  g.er.fbel.coef[i] <-
    transitivity(g.er.fbel, type="localaverage")
  g.er.fbel.apl[i] <-
    average.path.length(g.er.fbel)
  g.er.fbel.dia[i] <-
    diameter(g.er.fbel)
  g.er.fbel.avgdeg[i] <-
    mean(igraph::degree(g.er.fbel))
  g.er.fbel.avgbtwcen[i] <-
    mean(igraph::betweenness(g.er.fbel))
  g.er.fbel.avgclocen[i] <-
    mean(igraph::closeness(g.er.fbel))
  g.er.fbel.avgeigveccen[i] <-
    mean(igraph::eigen_centrality(g.er.fbel)$vector)
}

#save values as dataframe
g.er.fbel <-
  as.data.frame(cbind(g.er.fbel.ecount, g.er.fbel.coef,
                      g.er.fbel.apl, g.er.fbel.dia,
                      g.er.fbel.avgdeg,
                      g.er.fbel.avgbtwcen,
                      g.er.fbel.avgclocen,
                      g.er.fbel.avgeigveccen))

mean(g.er.fbel.coef) #0.01082603
sd(g.er.fbel.coef) #0.000101139
confint(g.er.fbel.coef)

mean(g.er.fbel.apl) #2.605739
sd(g.er.fbel.apl) #0.001976112

mean(g.er.fbel.dia) #3.956
sd(g.er.fbel.dia) #0.2051977

mean(g.er.fbel.avgdeg) #43.69406
sd(g.er.fbel.avgdeg) #0.143238

mean(g.er.fbel.avgbtwcen) #3241.987
```

```
sd(g.er.fbel.avgbtwcen) #3.989771

mean(g.er.fbel.avgclocen) #9.507038e-05
sd(g.er.fbel.avgclocen) #7.229988e-08

mean(g.er.fbel.avgeigveccen) #0.6202335
sd(g.er.fbel.avgeigveccen) #0.02240826
```

The following code describes the data simulation process using the Watts-Strogatz model. This process is analogous to the process described above. See results in .

```
set.seed(499)

numsim <- 1000

g.ws.fbel.ecount <- rep(NA,numsim)
g.ws.fbel.coef <- rep(NA, numsim)
g.ws.fbel.apl <- rep(NA, numsim)
g.ws.fbel.dia <- rep(NA, numsim)
g.ws.fbel.avgdeg <- rep(NA, numsim)
g.ws.fbel.avgbtwcen <- rep(NA, numsim)
g.ws.fbel.avgclocen <- rep(NA, numsim)
g.ws.fbel.avgeigveccen <- rep(NA, numsim)

for (i in 1:numsim) {
  #create a lattice with the same number of vertices as 'fbel'
  #let the starting number of neighbors be
  #equal to the minimum vertex degree of 'fbel'
  g.ws.fbel <- watts.strogatz.game(dim = 1, size = vcount(fbel),
                                   nei = min(degree(fbg)), p = 0)

  #generate list of random vertex values to add edges to lattice
  #each pair in the list represents one random edge
  randomedgepairs <- sample(1:vcount(fbel),
                            2*(ecount(fbel)-vcount(fbel)),
                            replace=TRUE)
  g.ws.fbel.pre <- add_edges(g.ws.fbel, randomedgepairs)


  #igraph object
  g.ws.fbel <- simplify(g.ws.fbel.pre)

  #observe the network statistics for this simulated graph
  g.ws.fbel.ecount[i] <-
```

```
    ecount(g.ws.fbel)
  g.ws.fbel.coef[i] <-
    transitivity(g.ws.fbel, type="localaverage")
  g.ws.fbel.apl[i] <-
    average.path.length(g.ws.fbel)
  g.ws.fbel.dia[i] <-
    diameter(g.ws.fbel)
  g.ws.fbel.avgdeg[i] <-
    mean(igraph::degree(g.ws.fbel))
  g.ws.fbel.avgbtwcen[i] <-
    mean(igraph::betweenness(g.ws.fbel))
  g.ws.fbel.avgclocen[i] <-
    mean(igraph::closeness(g.ws.fbel))
  g.ws.fbel.avgeigveccen[i] <-
    mean(igraph::eigen_centrality(g.ws.fbel)$vector)
}

#save values as dataframe
g.ws.fbel <-
  as.data.frame(cbind(g.ws.fbel.ecount, g.ws.fbel.coef,
                      g.ws.fbel.apl, g.ws.fbel.dia,
                      g.ws.fbel.avgdeg,
                      g.ws.fbel.avgbtwcen,
                      g.ws.fbel.avgclocen,
                      g.ws.fbel.avgeigveccen))

mean(g.ws.fbel.coef)#0.01073166
sd(g.ws.fbel.coef) #9.135127e-05

mean(g.ws.fbel.apl) #2.609323
sd(g.ws.fbel.apl) #0.000193598

mean(g.ws.fbel.dia) #3.95
sd(g.ws.fbel.dia) #0.218054

mean(g.ws.fbel.avgdeg) #43.44555
sd(g.ws.fbel.avgdeg) #0.01105388

mean(g.ws.fbel.avgbtwcen) #3249.223
sd(g.ws.fbel.avgbtwcen) # 0.3908744

mean(g.ws.fbel.avgclocen) #9.493807e-05
sd(g.ws.fbel.avgclocen) #7.319204e-09
```

```
mean(g.ws.fbel.avgeigveccen) #0.6234877
sd(g.ws.fbel.avgeigveccen) #0.02269783
```

The following code descibes how to fit graphs using ERGMs and simulating from the models that have been constructed. We have four models with different combinations of parameters, which we denote 1a (edges), 2a (edges and triangles), 2b (edges and k-stars of size 3), and 3a (edges, triangles, and k-stars of size 3). See results in and .

```
# needs an object of class network
# uses `fbg`, a `network` object

## one parameter: edges
# this should be the same as Erdos-Renyi
# set the seed for reproducible analysis
set.seed(499)

# model 1a
g.ergm1a.fbg <- ergm(fbg ~ edges)

numsim = 1000
set.seed(499)

g.ergm1a.fbg.sim.ecount <- rep(NA, numsim)
g.ergm1a.fbg.sim.coef <- rep(NA, numsim)
g.ergm1a.fbg.sim.apl <- rep(NA, numsim)
g.ergm1a.fbg.sim.dia <- rep(NA, numsim)
g.ergm1a.fbg.sim.avgdeg <- rep(NA, numsim)
g.ergm1a.fbg.sim.avgbtwcen <- rep(NA, numsim)
g.ergm1a.fbg.sim.avgclocen <- rep(NA, numsim)
g.ergm1a.fbg.sim.avgeigveccen <- rep(NA, numsim)

for (i in 1:numsim) {
  #simulate graphs from ergms model one at a time
  #convert to igraph object
  g.ergm.fbg.sim.convert <- asIgraph(simulate(g.ergm1a.fbg, nsim = 1))

  #observe the network statistics for this simulated graph
  g.ergm1a.fbg.sim.ecount[i] <- ecount(g.ergm.fbg.sim.convert)
  g.ergm1a.fbg.sim.coef[i] <-
    transitivity(g.ergm.fbg.sim.convert, type="localaverage")
  g.ergm1a.fbg.sim.apl[i] <-
    average.path.length(g.ergm.fbg.sim.convert)
  g.ergm1a.fbg.sim.dia[i] <- diameter(g.ergm.fbg.sim.convert)
  g.ergm1a.fbg.sim.avgdeg[i] <-
```

```
    mean(igraph::degree(g.ergm.fbg.sim.convert))
  g.ergm1a.fbg.sim.avgbtwcen[i] <-
    mean(igraph::betweenness(g.ergm.fbg.sim.convert))
  g.ergm1a.fbg.sim.avgclocen[i] <-
    mean(igraph::closeness(g.ergm.fbg.sim.convert))
  g.ergm1a.fbg.sim.avgeigveccen[i] <-
    mean(igraph::eigen_centrality(g.ergm.fbg.sim.convert)$vector)
}

g.ergm1a.fbg.sim.df <-
  as.data.frame(cbind(g.ergm1a.fbg.sim.ecount, g.ergm1a.fbg.sim.coef,
                      g.ergm1a.fbg.sim.apl, g.ergm1a.fbg.sim.dia,
                      g.ergm1a.fbg.sim.avgdeg,
                      g.ergm1a.fbg.sim.avgbtwcen,
                      g.ergm1a.fbg.sim.avgclocen,
                      g.ergm1a.fbg.sim.avgeigveccen))

mean(g.ergm1a.fbg.sim.coef) #0.3696413
sd(g.ergm1a.fbg.sim.coef) #0.001246206

mean(g.ergm1a.fbg.sim.apl) #2.885094
sd(g.ergm1a.fbg.sim.apl) #0.004316369

mean(g.ergm1a.fbg.sim.dia) #5.216
sd(g.ergm1a.fbg.sim.dia) #0.4117202

mean(g.ergm1a.fbg.sim.avgdeg) #43.66728
sd(g.ergm1a.fbg.sim.avgdeg) #0.05240351

mean(g.ergm1a.fbg.sim.avgbtwcen) #3805.712
sd(g.ergm1a.fbg.sim.avgbtwcen) #8.757621

mean(g.ergm1a.fbg.sim.avgclocen) #8.285989e-05
sd(g.ergm1a.fbg.sim.avgclocen) #8.353233e-06

mean(g.ergm1a.fbg.sim.avgeigveccen) #0.04179492
sd(g.ergm1a.fbg.sim.avgeigveccen) #0.0008307558


## two parameters: edges and triangles
# set the seed for reproducible analysis
set.seed(499)

# model 2a
```

```
# WARNINGS: takes time (20 iterations)
g.ergm2a.fbg <- ergm(fbg ~ edges + triangle)

numsim = 1000
set.seed(499)

g.ergm2a.fbg.sim.ecount <- rep(NA, numsim)
g.ergm2a.fbg.sim.coef <- rep(NA, numsim)
g.ergm2a.fbg.sim.apl <- rep(NA, numsim)
g.ergm2a.fbg.sim.dia <- rep(NA, numsim)
g.ergm2a.fbg.sim.avgdeg <- rep(NA, numsim)
g.ergm2a.fbg.sim.avgbtwcen <- rep(NA, numsim)
g.ergm2a.fbg.sim.avgclocen <- rep(NA, numsim)
g.ergm2a.fbg.sim.avgeigveccen <- rep(NA, numsim)

for (i in 1:numsim) {
  #simulate graphs from ergms model one at a time
  #convert to igraph object
  g.ergm.fbg.sim.convert <- asIgraph(simulate(g.ergm2a.fbg, nsim = 1))

  #observe the network statistics for this simulated graph
  g.ergm2a.fbg.sim.ecount[i] <-
    ecount(g.ergm.fbg.sim.convert)
  g.ergm2a.fbg.sim.coef[i] <-
    transitivity(g.ergm.fbg.sim.convert, type="localaverage")
  g.ergm2a.fbg.sim.apl[i] <-
    average.path.length(g.ergm.fbg.sim.convert)
  g.ergm2a.fbg.sim.dia[i] <-
    diameter(g.ergm.fbg.sim.convert)
  g.ergm2a.fbg.sim.avgdeg[i] <-
    mean(igraph::degree(g.ergm.fbg.sim.convert))
  g.ergm2a.fbg.sim.avgbtwcen[i] <-
    mean(igraph::betweenness(g.ergm.fbg.sim.convert))
  g.ergm2a.fbg.sim.avgclocen[i] <-
    mean(igraph::closeness(g.ergm.fbg.sim.convert))
  g.ergm2a.fbg.sim.avgeigveccen[i] <-
    mean(igraph::eigen_centrality(g.ergm.fbg.sim.convert)$vector)
}

g.ergm2a.fbg.sim.df <-
  as.data.frame(cbind(g.ergm2a.fbg.sim.ecount, g.ergm2a.fbg.sim.coef,
                      g.ergm2a.fbg.sim.apl, g.ergm2a.fbg.sim.dia,
                      g.ergm2a.fbg.sim.avgdeg,
                      g.ergm2a.fbg.sim.avgbtwcen,
```

```
                        g.ergm2a.fbg.sim.avgclocen,
                        g.ergm2a.fbg.sim.avgeigveccen))

mean(g.ergm2a.fbg.sim.coef) #0.4823232
sd(g.ergm2a.fbg.sim.coef) #0.001965885

mean(g.ergm2a.fbg.sim.apl) #3.052008
sd(g.ergm2a.fbg.sim.apl) #0.008639291

mean(g.ergm2a.fbg.sim.dia) #6.098
sd(g.ergm2a.fbg.sim.dia) #0.3008097

mean(g.ergm2a.fbg.sim.avgdeg) #44.54161
sd(g.ergm2a.fbg.sim.avgdeg) #0.03491666

mean(g.ergm2a.fbg.sim.avgbtwcen) #4140.19
sd(g.ergm2a.fbg.sim.avgbtwcen) #17.65641

mean(g.ergm2a.fbg.sim.avgclocen) #6.007678e-05
sd(g.ergm2a.fbg.sim.avgclocen) #1.513646e-05

mean(g.ergm2a.fbg.sim.avgeigveccen) #0.04103567
sd(g.ergm2a.fbg.sim.avgeigveccen) #3.5777786e-05


## two parameters: edges and k-stars
# WARNINGS: takes time (18 iterations)
# set the seed for reproducible analysis
set.seed(499)

# model 2b
g.ergm2b.fbg <- ergm(fbg ~ edges + kstar(3))

numsim = 1000
set.seed(499)

g.ergm2b.fbg.sim.ecount <- rep(NA, numsim)
g.ergm2b.fbg.sim.coef <- rep(NA, numsim)
g.ergm2b.fbg.sim.apl <- rep(NA, numsim)
g.ergm2b.fbg.sim.dia <- rep(NA, numsim)
g.ergm2b.fbg.sim.avgdeg <- rep(NA, numsim)
g.ergm2b.fbg.sim.avgbtwcen <- rep(NA, numsim)
g.ergm2b.fbg.sim.avgclocen <- rep(NA, numsim)
g.ergm2b.fbg.sim.avgeigveccen <- rep(NA, numsim)
```

```
for (i in 1:numsim) {
  #simulate graphs from ergms model one at a time
  #convert to igraph object
  g.ergm.fbg.sim.convert <- asIgraph(simulate(g.ergm2b.fbg, nsim = 1))

  #observe the network statistics for this simulated graph
  g.ergm2b.fbg.sim.ecount[i] <-
    ecount(g.ergm.fbg.sim.convert)
  g.ergm2b.fbg.sim.coef[i] <-
    transitivity(g.ergm.fbg.sim.convert, type="localaverage")
  g.ergm2b.fbg.sim.apl[i] <-
    average.path.length(g.ergm.fbg.sim.convert)
  g.ergm2b.fbg.sim.dia[i] <-
    diameter(g.ergm.fbg.sim.convert)
  g.ergm2b.fbg.sim.avgdeg[i] <-
    mean(igraph::degree(g.ergm.fbg.sim.convert))
  g.ergm2b.fbg.sim.avgbtwcen[i] <-
    mean(igraph::betweenness(g.ergm.fbg.sim.convert))
  g.ergm2b.fbg.sim.avgclocen[i] <-
    mean(igraph::closeness(g.ergm.fbg.sim.convert))
  g.ergm2b.fbg.sim.avgeigveccen[i] <-
    mean(igraph::eigen_centrality(g.ergm.fbg.sim.convert)$vector)
}

g.ergm2b.fbg.sim.df <-
  as.data.frame(cbind(g.ergm2b.fbg.sim.ecount, g.ergm2b.fbg.sim.coef,
                      g.ergm2b.fbg.sim.apl, g.ergm2b.fbg.sim.dia,
                      g.ergm2b.fbg.sim.avgdeg,
                      g.ergm2b.fbg.sim.avgbtwcen,
                      g.ergm2b.fbg.sim.avgclocen,
                      g.ergm2b.fbg.sim.avgeigveccen))

mean(g.ergm2b.fbg.sim.coef) #0.3786565
sd(g.ergm2b.fbg.sim.coef) #0.001279854

mean(g.ergm2b.fbg.sim.apl) #2.850693
sd(g.ergm2b.fbg.sim.apl) #0.003085883

mean(g.ergm2b.fbg.sim.dia) #5.095
sd(g.ergm2b.fbg.sim.dia) #0.2933617

mean(g.ergm2b.fbg.sim.avgdeg) #44.0645
sd(g.ergm2b.fbg.sim.avgdeg) #0.05715313
```

```
mean(g.ergm2b.fbg.sim.avgbtwcen) #3736.415
sd(g.ergm2b.fbg.sim.avgbtwcen) #6.259627

mean(g.ergm2b.fbg.sim.avgclocen) #8.558362e-05
sd(g.ergm2b.fbg.sim.avgclocen) #6.024968e-06

mean(g.ergm2b.fbg.sim.avgeigveccen) #0.03923439
sd(g.ergm2b.fbg.sim.avgeigveccen) #0.0002047637


## three parameters: edges and triangles and k-stars
# set the seed for reproducible analysis
set.seed(499)

# model 3a
g.ergm3a.fbg <- ergm(fbg ~ edges + triangle + kstar(3))

numsim = 1000
set.seed(499)

g.ergm3a.fbg.sim.ecount <- rep(NA, numsim)
g.ergm3a.fbg.sim.coef <- rep(NA, numsim)
g.ergm3a.fbg.sim.apl <- rep(NA, numsim)
g.ergm3a.fbg.sim.dia <- rep(NA, numsim)
g.ergm3a.fbg.sim.avgdeg <- rep(NA, numsim)
g.ergm3a.fbg.sim.avgbtwcen <- rep(NA, numsim)
g.ergm3a.fbg.sim.avgclocen <- rep(NA, numsim)
g.ergm3a.fbg.sim.avgeigveccen <- rep(NA, numsim)

for (i in 1:numsim) {
  #simulate graphs from ergms model one at a time
  #convert to igraph object
  g.ergm.fbg.sim.convert <- asIgraph(simulate(g.ergm3a.fbg, nsim = 1))

  #observe the network statistics for this simulated graph
  g.ergm3a.fbg.sim.ecount[i] <-
    ecount(g.ergm.fbg.sim.convert)
  g.ergm3a.fbg.sim.coef[i] <-
    transitivity(g.ergm.fbg.sim.convert, type="localaverage")
  g.ergm3a.fbg.sim.apl[i] <-
    average.path.length(g.ergm.fbg.sim.convert)
  g.ergm3a.fbg.sim.dia[i] <-
    diameter(g.ergm.fbg.sim.convert)
  g.ergm3a.fbg.sim.avgdeg[i] <-
```

```
    mean(igraph::degree(g.ergm.fbg.sim.convert))
  g.ergm3a.fbg.sim.avgbtwcen[i] <-
    mean(igraph::betweenness(g.ergm.fbg.sim.convert))
  g.ergm3a.fbg.sim.avgclocen[i] <-
    mean(igraph::closeness(g.ergm.fbg.sim.convert))
  g.ergm3a.fbg.sim.avgeigveccen[i] <-
    mean(igraph::eigen_centrality(g.ergm.fbg.sim.convert)$vector)
}

g.ergm3a.fbg.sim.df <-
  as.data.frame(cbind(g.ergm3a.fbg.sim.ecount, g.ergm3a.fbg.sim.coef,
                      g.ergm3a.fbg.sim.apl, g.ergm3a.fbg.sim.dia,
                      g.ergm3a.fbg.sim.avgdeg,
                      g.ergm3a.fbg.sim.avgbtwcen,
                      g.ergm3a.fbg.sim.avgclocen,
                      g.ergm3a.fbg.sim.avgeigveccen))

mean(g.ergm3a.fbg.sim.coef) #0.4890789
sd(g.ergm3a.fbg.sim.coef) #0.001991055

mean(g.ergm3a.fbg.sim.apl) #3.056179
sd(g.ergm3a.fbg.sim.apl) #0.007902389

mean(g.ergm3a.fbg.sim.dia) #6.161
sd(g.ergm3a.fbg.sim.dia) #0.3677149

mean(g.ergm3a.fbg.sim.avgdeg) #44.51779
sd(g.ergm3a.fbg.sim.avgdeg) #0.03363215

mean(g.ergm3a.fbg.sim.avgbtwcen) #4148.404
sd(g.ergm3a.fbg.sim.avgbtwcen) #16.0654

mean(g.ergm3a.fbg.sim.avgclocen) #5.875202e-05
sd(g.ergm3a.fbg.sim.avgclocen) #1.503473e-05

mean(g.ergm3a.fbg.sim.avgeigveccen) #0.0410138
sd(g.ergm3a.fbg.sim.avgeigveccen) #3.450909e-05
```

In Conclusion

None

Appendix B

Revisions

Six typographical errors were corrected throughout the thesis. Major changes among include turning some letters into in-line mathematical symbols (e.g. "a graph G" versus "a graph G"). Other changes include: incorrect word choice (e.g. "edges" instead of "vertices") and grammatical errors.

There were also three footnote errors that did not display the intended information in the Introduction, Chapter 1, and Chapter 2.

Two edits to the headers of various R-code chunks were made so that the code chunk does not actually run when the file itself is being knit.

There was a problem were some sources referenced in the text did not appear in the bibliography; they were added back in.

References

Baumer, B., Kaplan, D., & Horton, N. (2017). *Modern Data Science With R: With Digital Download*. Taylor & Francis. Retrieved from `https://books.google.com/books?id=Gv1nvgAACAAJ`

Bojanowski, M. (2015). *Intergraph: Coercion routines for network data objects*. Retrieved from `http://mbojan.github.io/intergraph`

Butts, C. T. (2008). Network: A package for managing relational data in r. *Journal of Statistical Software*, *24*(2). Retrieved from `http://www.jstatsoft.org/v24/i02/paper`

Butts, C. T. (2015). *Network: Classes for relational data*. The Statnet Project (`http://statnet.org`). Retrieved from `http://CRAN.R-project.org/package=network`

Butts, C. T., & others. (2008). Social network analysis with sna. *Journal of Statistical Software*, *24*(6), 1–51.

Butts, C., Hunter, D., Handcock, M. S., Morris, M., Krivtisky, P. N., Almquist, Z., ... Bender de-Moll, S. (2015, June). Introduction to Exponential-family Random Graph (ERG or p*) modeling with ergm. Retrieved from `https://statnet.org/trac/raw-attachment/wiki/Sunbelt2015/ergm_tutorial.pdf`

Csárdi, G., & Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal*, *Complex Systems*, 1695. Retrieved from `http://igraph.org`

Erdős, P., & Rényi, A. (1959). On random graphs, i. *Publicationes Mathematicae (Debrecen)*, *6*, 290–297.

Erdős, P., & Rényi, A. (1960). On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, *5*(1), 17–60.

Erdős, P., & Rényi, A. (1964). On the strength of connectedness of a random graph. *Acta Mathematica Academiae Scientiarum Hungarica*, *12*(1-2), 261–267.

Gilbert, E. N. (1959). Random graphs. *The Annals of Mathematical Statistics*, *30*(4), 1141–1144.

Gilks, W. R., Richardson, S., & Spiegelhalter, D. (1995). *Markov chain monte carlo*

in practice. CRC press.

Handcock, M. S., Hunter, D. R., Butts, C. T., Goodreau, S. M., & Morris, M. (2008). Statnet: Software tools for the representation, visualization, analysis and simulation of network data. *Journal of Statistical Software*, *24*(1), 1–11. Retrieved from `http://www.jstatsoft.org/v24/i01`

Handcock, M. S., Hunter, D. R., Butts, C. T., Goodreau, S. M., Krivitsky, P. N., & Morris, M. (2017). *Ergm: Fit, simulate and diagnose exponential-family models for networks*. The Statnet Project (`http://www.statnet.org`). Retrieved from `https://CRAN.R-project.org/package=ergm`

Handcock, M. S., Hunter, D. R., Butts, C. T., Goodreau, S. M., Krivitsky, P. N., Bender-deMoll, S., & Morris, M. (2016). *Statnet: Software tools for the statistical analysis of network data*. The Statnet Project (`http://www.statnet.org`). Retrieved from `CRAN.R-project.org/package=statnet`

Hunter, D. R., Handcock, M. S., Butts, C. T., Goodreau, S. M., & Morris, M. (2008). Ergm: A Package to Fit, Simulate and Diagnose Exponential-Family Models for Networks. *J Stat Softw*, *24*(3), nihpa54860. Retrieved from `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2743438/`

Hunter, D. R., Handcock, M. S., Butts, C. T., Goodreau, S. M., Morris, & Martina. (2008). Ergm: A package to fit, simulate and diagnose exponential-family models for networks. *Journal of Statistical Software*, *24*(3), 1–29.

Jackson, M. O. (2013). Social and economic networks: Models and analysis. Stanford University; Coursera Course Lecture.

Kolaczyk, E. D. (2009). *Statistical Analysis of Network Data: Methods and Models*. Springer Science & Business Media.

Kolaczyk, E. D., & Csárdi, G. (2014). *Statistical Analysis of Network Data with R*. Springer.

Lazega, E., & Pattison, P. E. (1999). Multiplexity, generalized exchange and cooperation in organizations: A case study. *Social Networks*, *21*(1), 67–90.

Leskovec, J., & Krevl, A. (2014, June). SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`.

Leskovec, J., Chakrabarti, D., Kleinberg, J., & Faloutsos, C. (2005). Realistic, Mathematically Tractable Graph Generation and Evolution, Using Kronecker Multiplication. In *Knowledge Discovery in Databases: PKDD 2005* (pp. 133–145). Springer, Berlin, Heidelberg. Retrieved from `http://link.springer.com/chapter/10.1007/11564126_17`

Leskovec, J., Chakrabarti, D., Kleinberg, J., Faloutsos, C., & Ghahramani, Z. (2010). Kronecker Graphs: An Approach to Modeling Networks. *Journal of Machine Learning Research*, *11*(Feb), 985–1042. Retrieved from `http://www.jmlr.org/`

papers/v11/leskovec10a.html

Newman, M. (2010). *Networks: An Introduction*. OUP Oxford.

Toivonen, R., Kovanen, L., Kivelä, M., Onnela, J.-P., Saramäki, J., & Kaski, K. (2009). A comparative study of social network models: Network evolution models and nodal attribute models. *Social Networks*, *31*(4), 240–254. http://doi.org/10.1016/j.socnet.2009.06.004

Travers, J., & Milgram, S. (1967). The small world problem. *Phychology Today*, *1*, 61–67.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of "small-world" networks. *Nature*, *393*(6684), 440–442. http://doi.org/10.1038/30918

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, *33*(4), 452–473. http://doi.org/10.1086/jar.33.4.3629752