INTRODUCTION TO EXPONENTIAL RANDOM GRAPH MODELS

By

Bradley Morton, B.S.

A Project Submitted in Partial Fulfillment of the Requirements
for the Degree of

Master of Science
in
Statistics

University of Alaska Fairbanks
May 2021

APPROVED:

Julie McIntyre, Committee Chair

Ronald Barry, Committee Co-Chair

Scott Goddard, Committee Member

Margaret Short, Committee Member

Leah Berman, Department Chair

Department of Mathematics and Statistics

Abstract:

Exponential random graph models (ERGMs) are used for analyzing network data for a variety of applications. Vertices, or nodes, represent entities, and edges, or ties, represent connections between entities. The ERGM model allows for a representation of edges in structures (from lone edges to triangles and cycles) as an exponential family random variable, a known family of distributions with known properties, such as showing statistics to be complete or sufficient by viewing the distribution.

This paper provides an introduction to the topic with both theoretical and applied information, starting with an introduction to the necessary graph theory, graph structures, and theoretical background for fitting models, then moves on to worked examples using the Statnet package.

Introduction

This paper is intended to serve as an introduction to exponential random graph models (ERGMs). The paper begins by defining all of the relevant terms, then describing several characteristics of the models. Next is a discussion on how models are selected and parameter estimates are generated, followed by worked examples and a simulation.

A graph, in the context of graph theory, is defined as a set of vertices, also referred to as points or nodes, with a set of edges that go from vertex to vertex, also referred to as connections or dyads. While in the field of graph theory edges can connect to the same vertex on both sides, this case is not relevant to exponential random graph models.

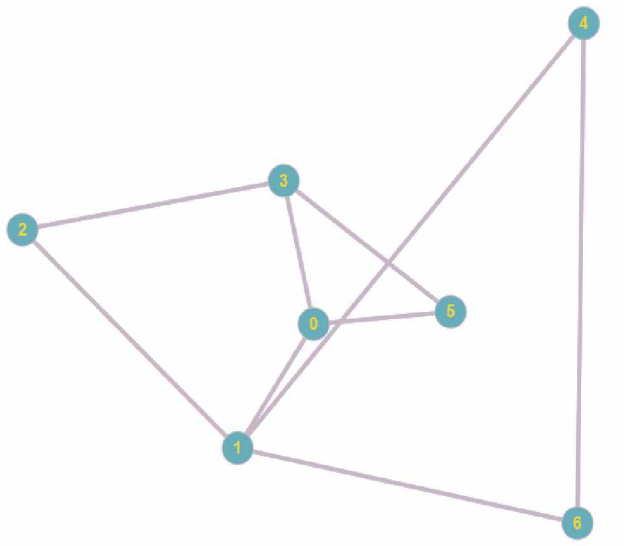
Graphs are stored as either adjacency lists or adjacency matrices. Below is an adjacency matrix, as Table 1, and its corresponding graph is Figure 1. Where a 1 occurs in the matrix, an edge exists between the vertices. Otherwise, there is none. Note that the matrix has diagonal symmetry, as an edge from vertex 1 to vertex 2 is equivalent to an edge from vertex 2 to vertex 1. When using directed edges, where edges have a source and a destination, this symmetry is lost unless all edges are reciprocated, and the sources are listed in the leftmost column and their destinations to the right of them. Note that this indicates that the diagonal of a matrix consists of zeroes; while in other uses of graph theory this can occur, it does not in this context.

It is possible to place numerical weights on the edges in a graph, which allows for models that contain more nuanced information. For instance, applications of ERGMs such as modeling the ties between companies could change from simply using binary states of having done business or not having done business to using numbers such as the total value of transactions between the company, allowing for more nuanced predictions. Such models are, however, beyond the scope of this paper.

Table 1:

Vertex	0	1	2	3	4	5	6
0	0	1	0	1	0	1	0
1	1	0	1	0	1	0	1
2	0	1	0	1	0	0	0
3	1	0	1	0	0	1	0
4	0	1	0	0	0	0	1
5	1	0	0	1	0	0	0
6	0	1	0	0	1	0	0

Figure 1:



Next, definitions of the terms odds and log odds, which will be important to interpreting models that give binary predictions, are provided. Odds refers to the probability of success, where success is an edge occurring and failure is an edge not occurring, divided by the probability of failure as given in Equation 1, where p is the probability of success:

$$\text{odds (p)} = \frac{p}{1-p} \qquad \textbf{Equation 1}$$

Log odds is defined as the natural logarithm of the odds, as shown below in Equation 2. Log odds have the advantage of being addable and subtractable, with a domain of the real number line.

$$\text{Log odds(p)} = ln(odds(p)) \qquad \textbf{Equation 2}$$

Formally speaking, an ERGM is an equation that assigns probabilities to graphs, giving an exponential family random variable, where the vertices are a fixed set and the edges are randomly occurring. The probability mass function looks like

$$p(Y = y) \propto exp(\theta^T z(y)) \qquad \textbf{Equation 3}$$

where y is a particular graph, or set of edges upon the fixed set of vertices,θ is a vector of parameters, and $z(y)$ is some collection of graph statistics for the graph y. The graph statistics are counts of various features that can occur in the model, such as edges and triangles, that are

formally defined shortly. These counts are sufficient statistics for the random variable (Ghafouri, Khasteh, 2020).

In most cases the normalizing constant required for Equation 1 to yield a proper probability mass function is intractable, as the size of the support of the random variable is $2^{(n(n-1)/2)}$ on a set of n vertices. For example, a graph with only 10 vertices would require computations for approximately $3.5 * 10^{13}$ graphs, which, storage issues aside, would take on the order of twenty six thousand years to calculate on a modern computer. As a result, the normalizing constant required to make all of the elements sum to one is only ever calculated in narrow, specific cases, though, due to the finite support and finite value of each element, the normalizing constant will be a finite number. Otherwise other methods are used to get around needing to calculate the normalizing constant.

Ghafouri and Khasteh (2020) presented a survey paper on exponential random graph models. The paper provides a formal definition of ERGMs, discusses methods for estimation of model parameters, and several applications that researchers are using in various fields, followed by information on tools available to perform analysis on network datasets and references to various structures that can be used as the sufficient statistics. A structure refers to a set of edges meeting certain criteria. The following discussion includes many elements of their paper.

The simplest case for a random graph model is when each possible edge has the same probability of occurring, and all edges are independent of each other. This ends up being one of the special cases where calculating the normalizing constant is possible, as the total number of edges in the graph follows a binomial distribution, where each edge has the same, independent probability of occurring. A sample can be taken from a binomial distribution and mapped directly to a graph. This has the advantage of being easily modeled, simulated, and tested, but tends not to work well for real world datasets where interest is in looking at cases where vertices themselves have variables that can be used for model fitting and more complex structures are present.

The number of structures that can be used as sufficient statistics is extensive (Amati, V., Lomi, A., & Mira, A. 2006); what is listed here is only an introduction. Structures and statistics are

computed differently for graphs with directed and undirected edges. The simplest is with undirected edges. First, some notation and a new definition will be developed. The degree of a vertex is the number of edges connected to it. For an undirected graph Y, define Y_{ij} as one if there exists an edge between vertices i and j and as zero otherwise. Note that $Y_{ij} = Y_{ji}$. Note that $Y_{ij}Y_{jk}$ is a one if both of those edges exist and a zero otherwise, for distinct i, j, and k values. The most basic structure is the lone edge. The simplest extension of this is the two path, where two edges share a common vertex, denoted as $Y_{ij}Y_{jk} = 1$, as described above. The next extension of this is a triangle, where $Y_{ij}Y_{jk}Y_{ik} = 1$. In this case, for three vertices i, j, and k, there exists an edge between each pair of these vertices. Longer cycles can be defined similarly, such as in the case of $Y_{ij}Y_{jk}Y_{kl}Y_{li} = 1$. This is a cycle of length four. Two more common statistics are the density, the total number of edges divided by the number of all possible edges, and the triangle percentage, which is the number of triangles that exist divided by the number of possible triangles.

Vertices can have variables themselves that may be described with various statistics. These statistics can be relational, such as the degree of a vertex, defined as the number of edges connecting to a vertex, or defined on the vertices themselves, such as continuous or categorical variables. The simplest such statistic is the mean degree of the vertices. Homophily is a statistic equal to the number of edges that occur between vertices with the same value of the categorical variable. When there are continuous variables present on vertices, the absolute difference between the values for each occurring edge can be used as a statistic, as can the sum of the values. K-stars represent how many vertices particular vertices are connected to, and isolates represent how many vertices of degree zero, which are vertices that have no edges, there are.

Many of these statistics are defined differently with directed graphs. The notation gets slightly more precise here, as $Y_{ij} = 1$ implies an edge extending from i to j. Thus, unlike undirected graphs, $Y_{ij} = 1$ does not imply that $Y_{ji} = 1$. The most basic structure here is the density, the total number of edges divided by the total number of possible edges. Note that the total number of possible edges is twice as large in a directed graph as it is in an undirected graph. Another

variable that can be used is the number of asymmetric relations between vertices, that is to say cases where $Y_{ij} = 1$ or $Y_{ji} = 1$, but $Y_{ij}Y_{ji} = 0$.The accompaniment to this is the number of mutual edges, where $Y_{ij}Y_{ji} = 1$. The activity statistic is defined $Y_{ij}Y_{ik} = 1$,which represents one vertex with an edge headed to two other vertices. This can also be defined with more vertices, such as a case of $Y_{ij}Y_{ik}Y_{il} = 1$. The counterpart to the activity statistic is the popularity statistic, which is defined as $Y_{ji}Y_{ki} = 1$, which represents edges from two vertices reaching the same vertex. Like with activity, this can be extended, such as with the definition $Y_{ji}Y_{ki}Y_{li} = 1$,which represents three vertices each directing a vertex towards a common endpoint. Triangles are defined as before, with $Y_{ij}Y_{jk}Y_{ki} = 1$ defining a triangle, and longer cycles being similarly defined, such as $Y_{ij}Y_{jk}Y_{kl}Y_{li} = 1$ defining a cycle of length four. Triangle percentage is defined the same as in the undirected case. One more statistic is the number of transitivity relations that occur, with a transitivity relation being defined as $Y_{ij}Y_{jk}Y_{ik} = 1$,which represents a source vertex i with two edges leaving, one arriving at k, and the other arriving at j, from which another vertex arrives at k.

Like with undirected graphs, there are also numerous statistics that can be computed on the vertices themselves. To generate a statistic for continuous variables on vertices, the destination value is subtracted from the source value. The number of edges with the same categorical values for each vertex, as with undirected graphs, is defined the same, as are isolates. Mean vertex degree can be implemented with in degree, which refers to the number of edges that end at a vertex, out degree, which refers to the number of edges that leave a vertex, both, or some combination of these as different statistics.

Applications for these models for solving real world problems are very widespread (Ghafouri, Khasteh, 2020). In medical imaging, ERGMs are used for topics such as neural analysis in cases where brain imaging can be represented with networks (Solo V, Poline J-B, Lindquist MA, Simpson SL, Bowman FD, Chung MK, Cassidy B. 2018.). Social models have shown with Romanian teens that isolation can lead to detrimental mental health issues (Baggio, Luisier & Vladescu, 2017). In business, economics, and political science ERGMs have been used to

analyze connections between groups such as donors and groups seeking funding (Gallemore C, Jespersen K. 2016), mapping tourism between locales (Lozano S, Gutiérrez E. 2018), and analyzing the business connections behind black market drug stores (Duxbury SW, Haynie DL. 2018).

Developing the ERGM model

Before describing the details of fitting exponential random graph models, a brief introduction to logistic regression, which has many similarities to some ERGMs, is provided. Logistic regression is a generalized linear model for a binary response, modeling the probability that a response $Y_i = 1$ as a function of a set of covariates and regression coefficients X_i, as given in Equation 4 where each Y_i is given independently.

$$P(Y_i = 1) = \frac{exp(\boldsymbol{\beta}^T X_i)}{1 + exp(\boldsymbol{\beta}^T X_i)} \qquad \textbf{Equation 4}$$

Observe that the domain for $\boldsymbol{\beta}^T X_i$ is the real number line, while the range is (0,1). This allows for the intercept, variables, and weights to take any values while constraining the result of the equation to a valid probability. The regression coefficients for the equation are usually found using the Gauss-Newton algorithm for iteratively reweighted least squares (Strauss, Ikeda, 1990).

Now that the logistic regression model has been developed, the exponential random graph model can be developed as a set of independent logistic regressions. Suppose that for an edge Y_{ij} $P(Y_{ij} = 1) = \frac{exp(\theta)}{1 + exp(\theta)}$. Thus, $P(Y_{ij} = 0) = \frac{1}{1 + exp(\theta)}$. Now the binomial random variable is needed. The binomial random variable represents the probability of getting k successes on n independent trials with equal probability p. The probability mass function of this is $P(K = k) = (n\ choose\ k)\, p^k (1 - p)^{n-k}$. Thus, for the probabilities listed above, the probability of a graph occurring that possess p edges is $P(k\ edges) = \frac{((n\ choose\ 2)\ choose\ k)\ exp(\theta)^k}{(1 + exp(\theta))^{(n\ choose\ 2)}}$. Now consider a particular graph with k edges called Yk. The probability of this graph occurring is the probability of any graph with k edges occurring divided

by the total number of graphs with k edges, as each edge is independent in this case, and thus each such graph is equally likely to occur, as is seen in this expression: $P(Yk) = \frac{P(k\ edges)}{((n\ choose\ 2)\ choose\ k)}$. By substituting in the probability of k edges occurring, the expression can thus be shown as $\frac{P(k\ edges)}{((n\ choose\ 2)\ choose\ k)} = \frac{exp(\theta)^k}{(1+exp(\theta))^{(n\ choose\ 2)}}$. Now, given that the denominator of this term is simply a constant, it is apparent that $\frac{exp(\theta)^k}{(1+exp(\theta))^{(n\ choose\ 2)}} \propto exp(\theta k)$, which is the basic graph model for independent edges occurring with equal probability.

Model Fitting and Interpretation with Software

With ERGMs that lack edge dependence structures logistic regression is still being used, as independent probabilities of edges are still being examined. In many real world situations, however, interest is in models that take into account dependence between edges, often referred to as Markov dependence or Markov models. There are two commonly used ways to fit these models, Maximum Pseudolikelihood Estimation (MPLE) and Markov Chain Monte Carlo (MCMC) approaches (Robins, Pattison, et al. 2006).

MPLE can be used to estimate model parameters when edges are not independent. This is done by conditioning each edge individually on the rest of the data, which allows proceeding with iteratively reweighted least squares as is done with logistic regression. MPLE is easy to implement, and can run quickly when handling nuanced models. However, ignoring the dependence can lead to issues such as inaccurate estimates and standard errors. The properties of MPLEs are not well understood, which makes fitting models with MPLE a risky endeavor (Robins, Pattison, et al. 2006).

MCMC methods are implemented with importance sampling, where the Metropolis algorithm and Metropolis-Hastings algorithm are the most commonly used algorithms. The basic idea behind how MCMC works is that a baseline starting graph, often the empty graph with no vertices, is selected. Then sets of vertices are randomly selected and the state of their relation is randomly changed, which can mean adding or removing a triangle, edge, etc. This is then compared to the previous graph, with the superiority or inferiority of the new graph, measured by

resemblance to the original graph, influencing the probability of it being accepted. The advantage of MCMC algorithms is that they do not require the normalizing constant, just a function providing values proportional to the probabilities. Using these algorithms, an approximation of the probability mass function can be created stochastically, which then allows for parameter estimation. MCMC methods, however, are far more computationally intensive than MPLE estimation methods, making them less suitable for large networks, such as those with over a thousand vertices. Additionally, it is fairly common for graph distributions to be multimodal, which can make traversing the graph in the stochastic step challenging, as importance sampling will likely hover around a local maximum and may not converge. When working with the ergm and statnet packages (Krivitskym Handcock, et al., 2003-2020) degeneracy and invalid results are common when using MCMC approaches.

Degeneracy is an issue that can occur when fitting a model with MCMC. If the model being fit has little to do with the actual dataset, then the simulation step will create graphs that have little resemblance to the original graph. Failure to converge is possible even if the model is a good choice for the data, so a single failure to converge should not be treated as an indication that the model is poor, but attempting numerous times to fit the graph indicates it most likely is. Attempting to fit triangles to the faux.magnolia.high dataset used in the second example was unsuccessful, indicating that triangles are most likely not significant to it (Goodreau, Handcock, Hunter, et al. 2021).

Examples

In this section three network datasets will be examined and inference will be performed on them. The goal of this inference is to understand why connections are formed and in which patterns. Each of the three examples will start with the basic edge only graph model, then examine other features of the data to see if they help understand characteristics of the population.

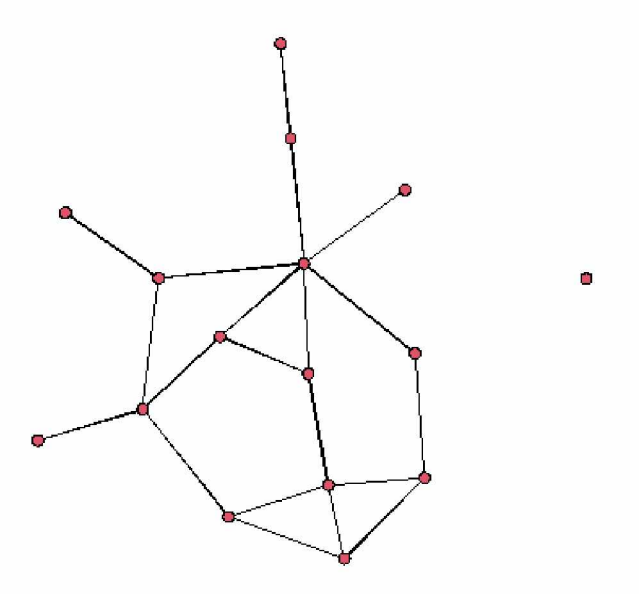
Example 1

Below in Figure 2 is a network taken from the R package statnet (Krivitskym Handcock, et al., 2003-2020) that shows a set of Florentine mercantile families from the Renaissance and the marriages that occurred between them. Each of the vertices represents a family, and each edge

represents a marriage between the two families it connects. To start, the assumed a model where finding the normalizing constant is not challenging, as the number of edges follows a simple binomial distribution given that all edges are predicted to occur with equal probability, and all possible arrangements of a given number of edges within the graph are equally likely. This means that simple maximum likelihood estimation can be done in this particular case.

$$p(Y = y) \propto exp(\theta \ * \ \#\ edges\ in\ y) \qquad \textbf{Equation 5}$$

Figure 2:



When fitting the model with the ergm function in the statnet package, the estimate for the log odds of an edge occurring as -1.6904, which produces an estimated probability of 0.1667. This was expected, given that out of 120 possible edges 20 exist, and thus a ⅙ probability of an edge occurring between two vertices makes sense.

Next, the presence of triangles in the graph will be examined By adding triangles alongside edges to the model, whether or not this type of clustering occurs is being examined. If the coefficient for triangle occurrence is positive, then clustering is likelier than if edges are independent, and if the coefficient is negative then clustering is less likely to occur relative to independent edges. With truly independent edges you would expect some clustering to occur, but a fair number of edges wouldn't be in clusters, and seeing a deviation to either side is

informative about the population. The model used for this is in Equation 6, where θ_1 is the edge density parameter and θ_2 is the triangle parameter.

$$p(Y = y) \propto exp(\theta_1 * \ \#\ edges\ in\ y\ +\ \theta_2 * \ \#triangles\ in\ y) \qquad \textbf{Equation 6}$$

Here the log odds of edges is about the same as last time, but there is a new term for the log odds of a triangle. The conditional log odds of the options available are in Table 2. Now that the model is using edge dependencies, interpretation of the model changes. To interpret the log odds of a possible edge existing, consider every other possible edge to have been determined, and examine what impact this would have on the graph structures. As can be seen in Table 2, the log odds of an edge occurring are dependent on whether or not triangles are formed by the addition of the edge. For instance, the probability of two families forming a marriage increases by about 0.023 if both families already possess a marriage to a third family, and increases by about 0.048 if both families are married to two other families. The p value for triangles, 0.275, is high enough that the effect is not traditionally ($p<0.05$) statistically significant.

Table 2:

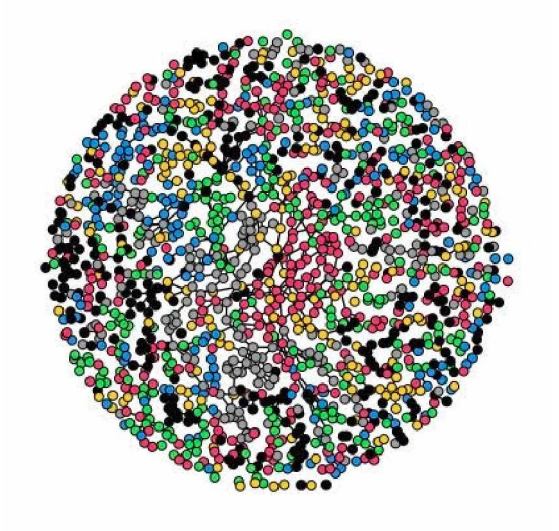
Adding	Log odds	Probability
1 edge, no new triangle	-1.67	0.158
1 edge, 1 new triangle	-1.67 + 0.16 = -1.51	0.181
1 edge, 2 new triangles	-1.67 +0.16*2 = -1.35	0.206

Next, a way to compare model fits is needed. This can be done with a likelihood ratio test, a comparison of AIC (Akaike's Information Criterion), or BIC (Bayes Information Criterion). AIC will be used for simplicity. AIC is a measure of fit based on the natural logarithm of the likelihood function with penalties for additional parameters. Lower values of AIC are better, and a common rule of thumb is that a model is significantly better than another model if the AICis lower by two or more. By this measure, the simpler model fits significantly better (110.1 to 122.1).

Example 2

Next is the faux.magnolia.high dataset from the ergm package (Krivitskym Handcock, et al., 2003-2020). The dataset is a simulation of a friendship network at a fictitious high school and edges representing friendship between students, with vertex variables denoting grade (7-12), sex, and race. Figure 3 shows a plot of the dataset, color coded by grade.

Figure 3:



The first step is fitting the intercept only model to this data, using the same model as described in Equation 5. The log odds of an edge occurring as -6.998, which gives us a probability of occurrence of approximately 0.000913. That is to say, using no other information, the probability that two randomly selected kids will be friends is 0.000913, which is just the proportion of possible edges that exist. Next, the vertex attributes will be examined. Sex, race, and grade are all categorical predictors, and it is examined whether or not sharing values for the predictors influences the probability of two students being friends. The model for this is given in Equation 7, where E is the number of edges that occur in the graph, S is the number of edges that occur between students that are the same sex, G is the number of edges that occur between students that are in the same grade, and R is the number of edges that occur between students of the same race. The θ values are the corresponding coefficients. The results of fitting this model are in Table 3. Observe that matching on any of the three attributes increases the log odds, and thus the probability, of two students being friends. A comparison of the AIC scores shows the model with node attributes has a score that is 2,517 lower than the edge only model, making it clearly a

better fit. Fitting triangles to this dataset was attempted, but resulted in degenerate graphs being generated, and was thus not testable.

$$p(Y = y) \propto exp(\theta_1 E + \theta_2 S + \theta_3 G + \theta_4 R) \qquad \textbf{Equation 7}$$

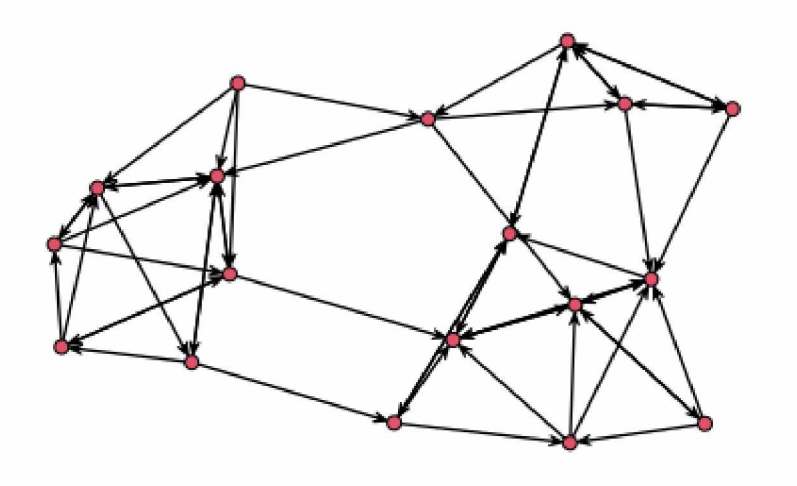
Table 3:

Graph Feature	Log Odds
Edges	-10.01
Sex	0.884
Grade	3.23
Race	1.20

Example 3

In this example a dataset with directed edges will be examined. In 1969 a researcher at a monastery asked each monk to indicate which three to four monks they thought most highly of (Krivitskym Handcock, et al., 2003-2020). The data is in Figure 4.

Figure 4:



The analysis starts with the edge only model, which still uses Equation 5 as its model. This yields a log odds for an edge of -1.496, which indicates that the probability of any particular monk selecting any other monk as one of their favorite three or four other monks is approximately 0.183. Next is the check for whether or not edges tend to be reciprocal. This

model is demonstrated in Equation 8, where E represents the number of edges, M represents the number of mutual edges, and the θ values are coefficients. The model results are listed in Table 4. Thus, the probability of a monk reciprocating being selected as a favorite is approximately 0.536, which is substantially higher than if the monk was not favored in turn. The AIC drops by 21.7, from 293.3 to 271.6, indicating that adding the parameter significantly improves our understanding of the model.

$p(y) \propto exp(\theta_1 E + \theta_2 M)$ **Equation 8**

Table 4:

Graph Structure	Log Odds	Probability
Edges	-2.1574	0.104
Mutual	2.3	0.536

Simulation:

For a simulation, I took the Florentine marriage model from Example 1 with edges only. Two hundred graphs were simulated based on this model. Each graph was fit against edges and triangles, despite triangles not being part of the simulated model, as seen in Equations 9 and 10. The estimated coefficients and p values for each were stored, with the results in Table 5. In Figure 5 is the histogram of estimates for the edge coefficient, and in Figure 6 is the histogram for the estimated triangle coefficient. As seen in Table 5, the estimates for both the edge parameter and the triangle parameter are biased. Despite having over twice the bias of the estimates for the edge parameter, the triangle parameter possesses a much lower mean squared error. While there was a substantial bias for the triangle parameter term, in only three of two hundred cases was the term traditionally statistically significant, while in one hundred and ninety eight cases the edge parameter was.

Simulated Model: $p(Y = y) \propto exp(-1.69 * \#edges\ in\ y)$ **Equation 10**

Tested Model: $p(Y = y) \propto exp(* \#edges\ in\ y + \theta_2 * \#triangles\ in\ y)$ **Equation 11**

Table 5:

	Real Value	Mean Estimate	Bias	Mean Squared Error	Number of times estimate had p< 0.05
Edge	-1.69	-1.506	-0.18	10.29	198

Parameter					
Triangle Parameter	0	-0.38	-0.38	1.88	3

Figure 5:

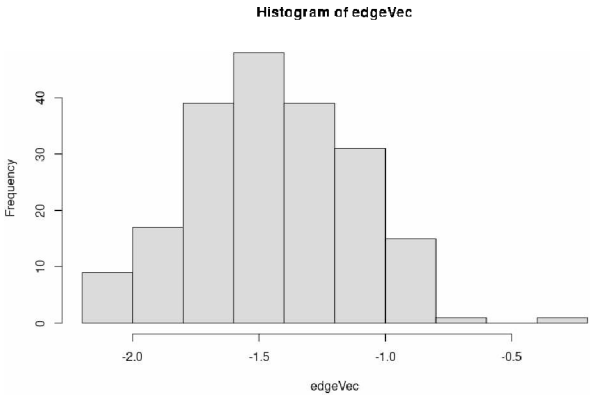
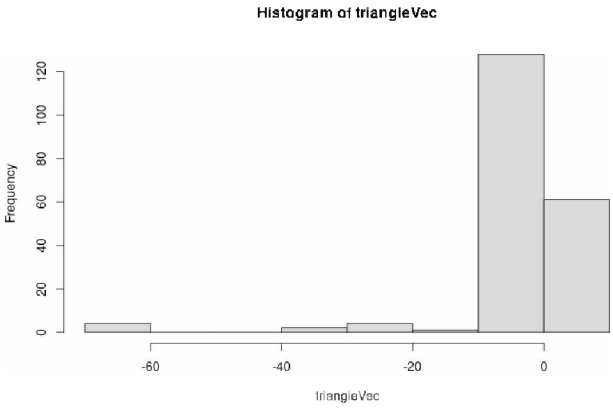


Figure 6:

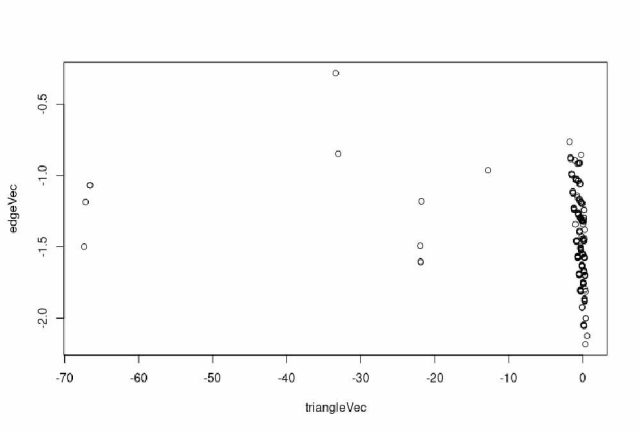


The edge histogram looks fairly close to as expected, though the real parameter is -1.69, and is thus not properly centered. Triangles, however, were expected to have coefficients of zero, which the distribution is not centered on.

Many of the simulations ran failed, largely due to degeneracy, with sixty four failures to generate the two hundred successful samples cases, a failure rate of approximately 0.24. It is possible that failed model fits and how common triangles occur in the simulated data are correlated.

In Figure 7 is the plot of the edge coefficients vs the triangle coefficients. There does not appear to be a relationship between the two, though there are so few data points where the triangle coefficient is less than one that it is hard to tell. Given that the null hypothesis that triangles are not sufficient was only rejected three times, however, there is limited data to look for such an interaction on.

Figure 7:



Conclusions:

The advantage exponential random graph models have over logistic regression is that they can handle dependence structures that logistic regression, which is limited to handling simple binary cases like the first example provided above, can not. This allows for a much more complex and nuanced analysis that can work with more features of datasets, applicable in a vast variety of fields and applications.

Future work:

I would like to familiarize myself with the generalized models that handle weighted edges. Many cases where network data exists are not as simple as a tie or lack thereof, making simpler models inadequate to the dataset, and there are many cases where edge data on networks can be simplified to a binary presence or lack thereof, which removes information from the data that could be used to provide a more nuanced analysis.

Appendix:

Here is the R code and output from the first model in the first example:

```
> model = ergm(flomarriage~edges)
Starting maximum pseudolikelihood estimation (MPLE):
Evaluating the predictor and response matrix.
Maximizing the pseudolikelihood.
Finished MPLE.
Stopping at the initial estimate.
Evaluating log-likelihood at the estimate.
> summary(model)
Call:
ergm(formula = flomarriage ~ edges)

Iterations:  5 out of 20

Monte Carlo MLE Results:
      Estimate Std. Error MCMC % z value Pr(>|z|)
edges  -1.6094     0.2449      0  -6.571   <1e-04 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance: 166.4  on 120  degrees of freedom
 Residual Deviance: 108.1  on 119  degrees of freedom

AIC: 110.1    BIC: 112.9    (Smaller is better.)
```

Here is logistic regression on the same dataset, and getting the same result:

```
> logDat = c(rep(1,20),rep(0,100))
> logModel = glm(logDat~1, family = "binomial")
> summary(logModel)

Call:
glm(formula = logDat ~ 1, family = "binomial")

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.6039  -0.6039  -0.6039  -0.6039   1.8930

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.6094     0.2449  -6.571 5.01e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 108.13  on 119  degrees of freedom
Residual deviance: 108.13  on 119  degrees of freedom
AIC: 110.13

Number of Fisher Scoring iterations: 3
```

And here is the code for the second model fit:

```
> model2=ergm(flomarriage~edges+triangle)
Starting maximum pseudolikelihood estimation (MPLE):
Evaluating the predictor and response matrix.
Maximizing the pseudolikelihood.
Finished MPLE.
Starting Monte Carlo maximum likelihood estimation (MCMLE):
Iteration 1 of at most 20:
Optimizing with step length 1.
The log-likelihood improved by 0.004097.
Step length converged once. Increasing MCMC sample size.
Iteration 2 of at most 20:
Optimizing with step length 1.
The log-likelihood improved by 0.002894.
Step length converged twice. Stopping.
Finished MCMLE.
Evaluating log-likelihood at the estimate. Using 20 bridges: 1 2
This model was fit using MCMC.  To examine model diagnostics and
mcmc.diagnostics() function.
> summary(model2)
Call:
ergm(formula = flomarriage ~ edges + triangle)

Iterations:  2 out of 20

Monte Carlo MLE Results:
         Estimate Std. Error MCMC % z value Pr(>|z|)
edges     -1.6735     0.3596      0  -4.654   <1e-04 ***
triangle   0.1596     0.5802      0   0.275    0.783
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

     Null Deviance: 166.4  on 120  degrees of freedom
 Residual Deviance: 108.1  on 118  degrees of freedom

AIC: 112.1    BIC: 117.7    (Smaller is better.)
```

Here is the code used for the first model fit in the second example:

```
> edge_model = ergm(faux.magnolia.high~edges)
Starting maximum pseudolikelihood estimation (MPLE):
Evaluating the predictor and response matrix.
Maximizing the pseudolikelihood.
Finished MPLE.
Stopping at the initial estimate.
Evaluating log-likelihood at the estimate.
> summary(edge_model)
Call:
ergm(formula = faux.magnolia.high ~ edges)

Iterations:  8 out of 20

Monte Carlo MLE Results:
      Estimate Std. Error MCMC % z value Pr(>|z|)
edges -6.99760    0.03205      0  -218.3   <1e-04 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance: 1478525  on 1066530  degrees of freedom
 Residual Deviance:   15580  on 1066529  degrees of freedom

AIC: 15582    BIC: 15594    (Smaller is better.)
```

And for the second model in the second example:

```
> edge_vertex_match_model = ergm(faux.magnolia.high~edges+nodematch("Sex")+nodematch("Grade")+nodematch("Race"))
Starting maximum pseudolikelihood estimation (MPLE):
Evaluating the predictor and response matrix.
Maximizing the pseudolikelihood.
Finished MPLE.
Stopping at the initial estimate.
Evaluating log-likelihood at the estimate.
> summary(edge_vertex_match_model)
Call:
ergm(formula = faux.magnolia.high ~ edges + nodematch("Sex") +
    nodematch("Grade") + nodematch("Race"))

Iterations:  9 out of 20

Monte Carlo MLE Results:
                Estimate Std. Error MCMC % z value Pr(>|z|)
edges          -10.01277    0.11526      0  -86.87   <1e-04 ***
nodematch.Sex    0.88438    0.07057      0   12.53   <1e-04 ***
nodematch.Grade  3.23105    0.08788      0   36.77   <1e-04 ***
nodematch.Race   1.19646    0.08147      0   14.69   <1e-04 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance: 1478525  on 1066530  degrees of freedom
 Residual Deviance:   13057  on 1066526  degrees of freedom

AIC: 13065    BIC: 13113    (Smaller is better.)
```

Here is the code for the first model for the third example:

```
> monk_edge_model = ergm(samplk3~edges)
Starting maximum pseudolikelihood estimation (MPLE):
Evaluating the predictor and response matrix.
Maximizing the pseudolikelihood.
Finished MPLE.
Stopping at the initial estimate.
Evaluating log-likelihood at the estimate.
> summary(monk_edge_model)
Call:
ergm(formula = samplk3 ~ edges)

Iterations:  5 out of 20

Monte Carlo MLE Results:
      Estimate Std. Error MCMC % z value Pr(>|z|)
edges  -1.4961     0.1478      0  -10.12   <1e-04 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance: 424.2  on 306  degrees of freedom
 Residual Deviance: 291.3  on 305  degrees of freedom

AIC: 293.3    BIC: 297    (Smaller is better.)
```

Here is the code for the second model for the first example:

```
> monk_edge_mutual_model = ergm(samplk3~edges+mutual)
Starting maximum pseudolikelihood estimation (MPLE):
Evaluating the predictor and response matrix.
Maximizing the pseudolikelihood.
Finished MPLE.
Starting Monte Carlo maximum likelihood estimation (MCMLE):
Iteration 1 of at most 20:
Optimizing with step length 1.
The log-likelihood improved by 0.0004352.
Step length converged once. Increasing MCMC sample size.
Iteration 2 of at most 20:
Optimizing with step length 1.
The log-likelihood improved by 0.001283.
Step length converged twice. Stopping.
Finished MCMLE.
Evaluating log-likelihood at the estimate. Using 20 bridges: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 .
This model was fit using MCMC.  To examine model diagnostics and check for degeneracy, use the mcmc.diagnostics() function.
> summary(monk_edge_mutual_model)
Call:
ergm(formula = samplk3 ~ edges + mutual)

Iterations:  2 out of 20

Monte Carlo MLE Results:
       Estimate Std. Error MCMC % z value Pr(>|z|)
edges   -2.1574     0.2158      0  -9.996   <1e-04 ***
mutual   2.3000     0.4732      0   4.860   <1e-04 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

     Null Deviance: 424.2  on 306  degrees of freedom
 Residual Deviance: 267.6  on 304  degrees of freedom

AIC: 271.6    BIC: 279    (Smaller is better.)
```

Here is the code used for the simulation. If running this yourself, note that the software will throw errors that prevent further execution and occasional answers of negative infinity will show up, so more work has to be done than simply running this loop once.

```
mod = ergm(flomarriage~edges)
edgeVec = rep(NA,200)
triangleVec = rep(NA,200)
edgePVals = rep(NA, 200)
triglePVals = rep(NA, 200)
for(i in 197:200)
{
  temp = simulate(mod)
  tempFit = ergm(temp~edges+triangles)
  tempSum = summary(tempFit)
  edgePVals[i] = tempSum$coefs[1,4]
  triglePVals[i] = tempSum$coefs[2,4]
  edgeVec[i]= tempFit$coef[1]
  triangleVec[i]= tempFit$coef[2]
}
t.test(triangleVec)
```

Bibliography:

Amati, V., Lomi, A., & Mira, A. (2018). Social Network Modeling. *Annual Review of Statistics and Its Application, 5(1), 343-369.*
https://www.annualreviews.org/doi/10.1146/annurev-statistics-031017-100746

Strauss, D., & Ikeda, M. (1990). Pseudolikelihood Estimation for Social Networks. *Journal of the American Statistical Association, 58*(409), 204-212.
http://www.stat.cmu.edu/~brian/780/bibliography/02%20ergms/StraussIkeda-JASA-1990.pdf

Goodreau, Handcock, Hunter, et al. A Statnet Tutorial (Downloaded 2021)
http://personal.psu.edu/drh20/papers/v24i09.pdf

Statnet Development Team
(Pavel N. Krivitsky, Mark S. Handcock, David R. Hunter, Carter T. Butts, Chad Klumb, Steven M. Goodreau, and Martina Morris) (2003-2020).
statnet: Software tools for the Statistical Modeling of Network Data.
URL http://statnet.org (Downloaded 2021)

Ghafouri S, Khasteh SH. 2020. A survey on exponential random graph models: an application perspective. PeerJ Computer Science 6:e269 https://doi.org/10.7717/peerj-cs.269

Baggio S, Luisier V, Vladescu C. 2017. Relationships between social networks and mental health: an exponential random graph model approach among Romanian adolescents. Swiss Journal of Psychology 76(1):5

Solo V, Poline J-B, Lindquist MA, Simpson SL, Bowman FD, Chung MK, Cassidy B. 2018. Connectivity in fMRI: blind spots and breakthroughs. IEEE Transactions on Medical Imaging 37:1537-1550

Gallemore C, Jespersen K. 2016. Transnational markets for sustainable development governance: the case of REDD+. World Development 86:79-94

Lozano S, Gutiérrez E. 2018. A complex network analysis of global tourism flows. International Journal of Tourism Research 20(5):588-604

Duxbury SW, Haynie DL. 2018. Building them up, breaking them down: topology, vendor selection patterns, and a digital drug market's robustness to disruption. Social Networks 52:238-250