Exponential Random Graph Models

Algorithms and Explanations

Megan Chan[1] Mengzi Guo[2] Amanda Sugiharto[3]
Niloufar Izadinia[4] Noshir Contractor[1]

[1] Northwestern University
[2] University of California, Berkeley
[3] Bain and Company
[4] University of Southern California

Northwestern University

SONIC
advancing the science of networks in communities

Objectives

The purpose of this slide deck is to provide a *technical* deep-dive into the inner workings of the statistical estimation procedures for Exponential Random Graph Models (ERGMs), which is a complement to other presentations which focus on how ERGMs can be applied to test multi-theoretical, multi-level (MTML) theories of communication networks (Contractor et al., 2006; Monge et al., 2003).

Prerequisite knowledge of statistics, probability, and linear algebra are assumed. Knowledge of statistical simulation and mathematical optimization are optional, but could contribute to a clearer understanding of the material.

Overview

Overview

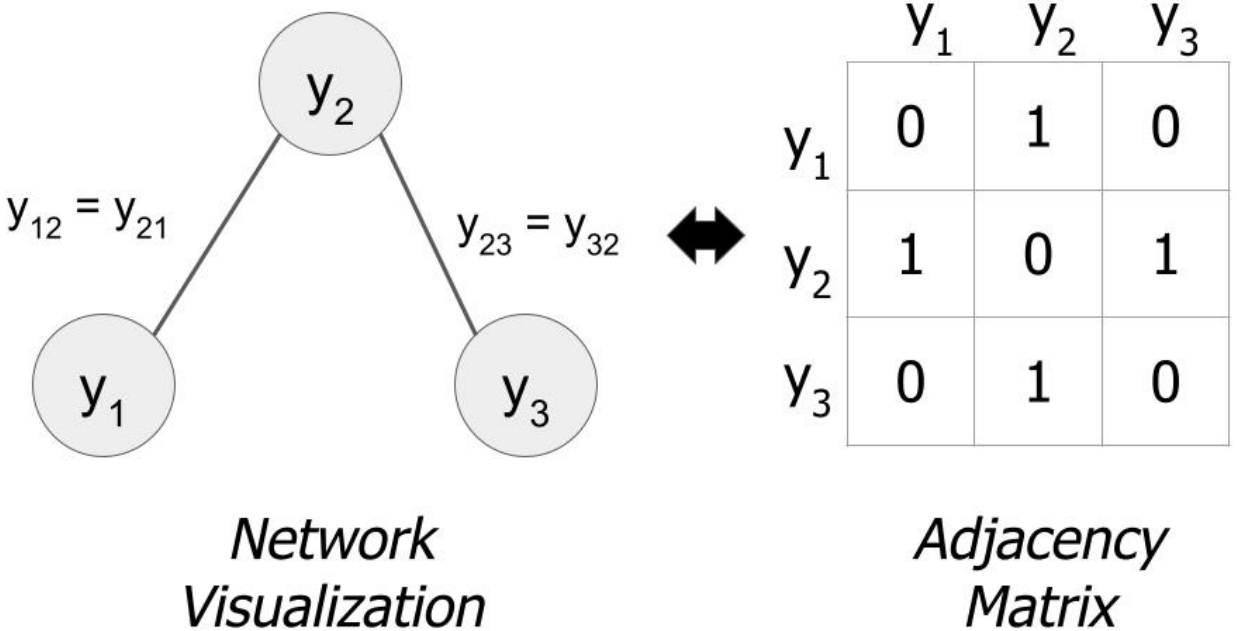
Background on ERGM

The Exponential Random Graph Model (ERGM) aims to determine the likelihood of ties given specified endogenous and exogenous (eg. nodal) attributes of a network.

The model uses the (complete) observed network and takes the number of vertices as being fixed. Through the generation of random networks, we can find coefficients of the model that are most likely to generate the observed network. These coefficients indicate the prevalence of the specified structural signatures in the generation of ties in the network.

ERGM Terminology

A network (or graph) is represented by its adjacency matrix, or the collection of edges connecting pairs of vertices.



	y_1	y_2	y_3
y_1	0	1	0
y_2	1	0	1
y_3	0	1	0

Figure: Undirected Graph

Network Foundations

A graph can be undirected or directed.

- Undirected: represents ties shared by two nodes, with $\binom{n}{2} = \frac{n(n-1)}{2}$ potential edges.
- Directed: represents unilateral relationships, with $n(n-1)$ potential edges.

n represents the number of nodes.

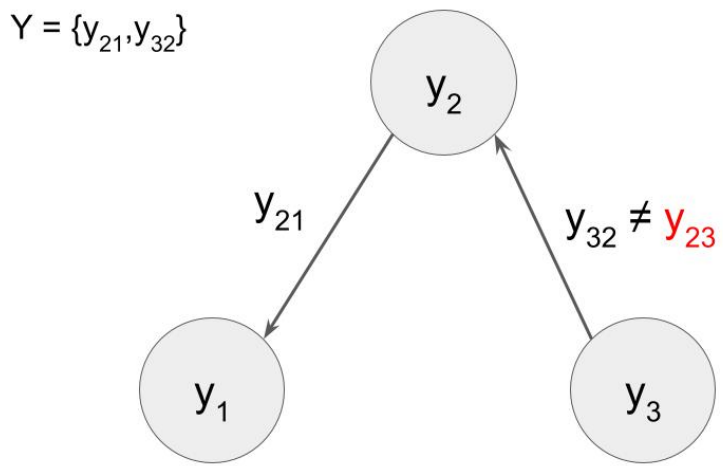
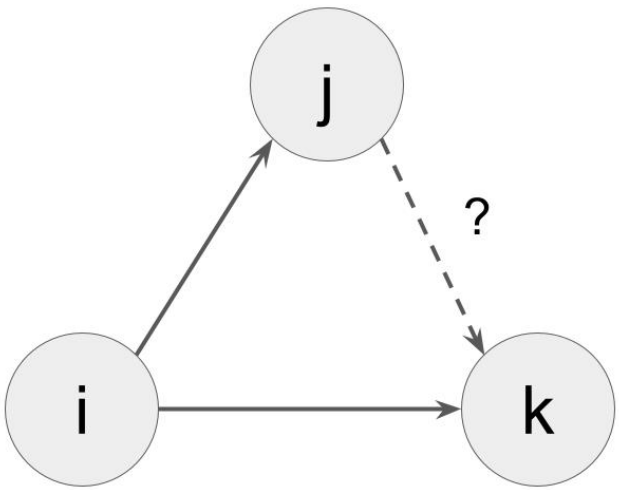


Figure: Directed Graph

Dyadic Independence

This is also known as the p1 and p2 models, where it assumes all dyads Y_{ij} are mutually independent, but this is not realistic for social networks.



Markov Dependence

Assumes conditional dependence only for ties between any two pairs of nodes (i,j) and (h,k) that have at least one node in common.

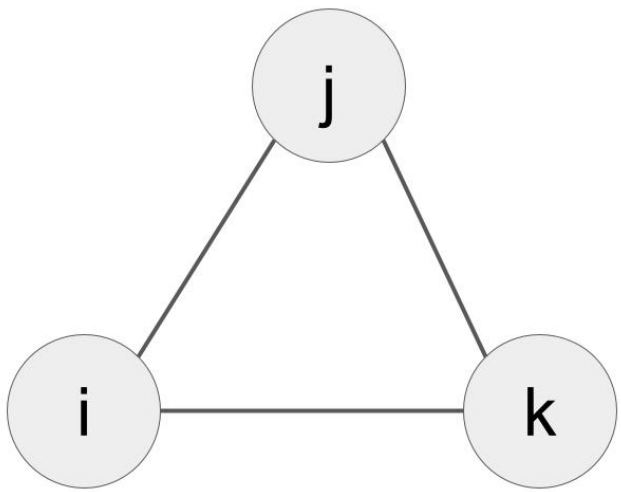


Figure: Markov Dependence

Partial Conditional Dependence

This is a relaxation of the Markov dependence assumption. In addition to the Markov condition, this dependence model also assumes conditional dependence between any two disjoint pair of nodes, if ties are observed between the two nodes within each pair.

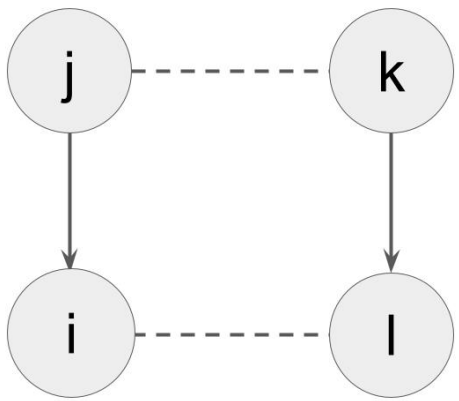


Figure: 4-cycle

Connecting Dependence to Network Structure

Presupposed dependence assumptions generate dependence graphs across a collection of nodes. Each connection in the dependence graph represents a possible structural signature in an ERGM model (Frank and Strauss, 1986).

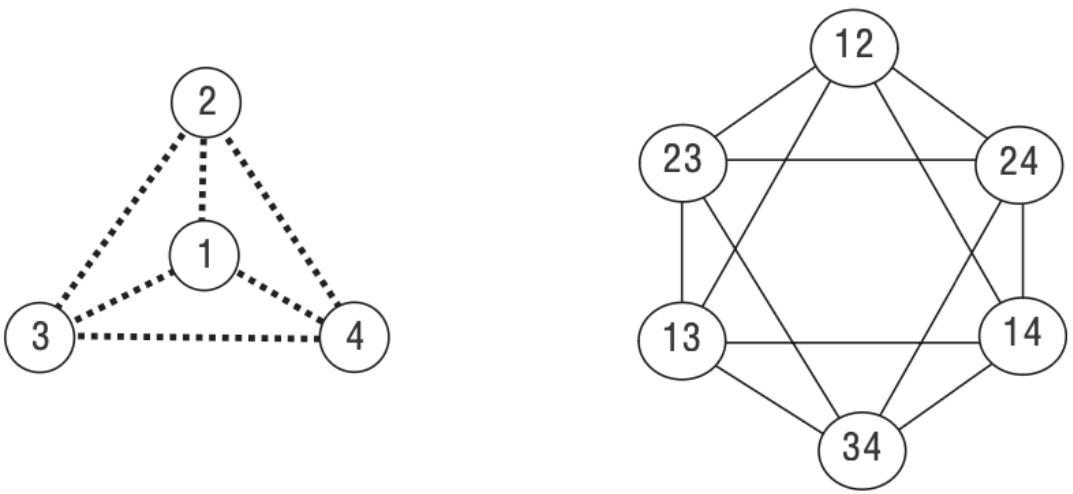


Figure 7.2. Tie-variables of (a) four-node graph and (b) associated Markov dependence graph.

Lusher et al., 2013

ERGM Foundations

Any graph can be decomposed into these structural signatures, where the count of each subgraph is a **sufficient statistic** in an ERGM model (Besag, 1974; Frank and Strauss, 1986).

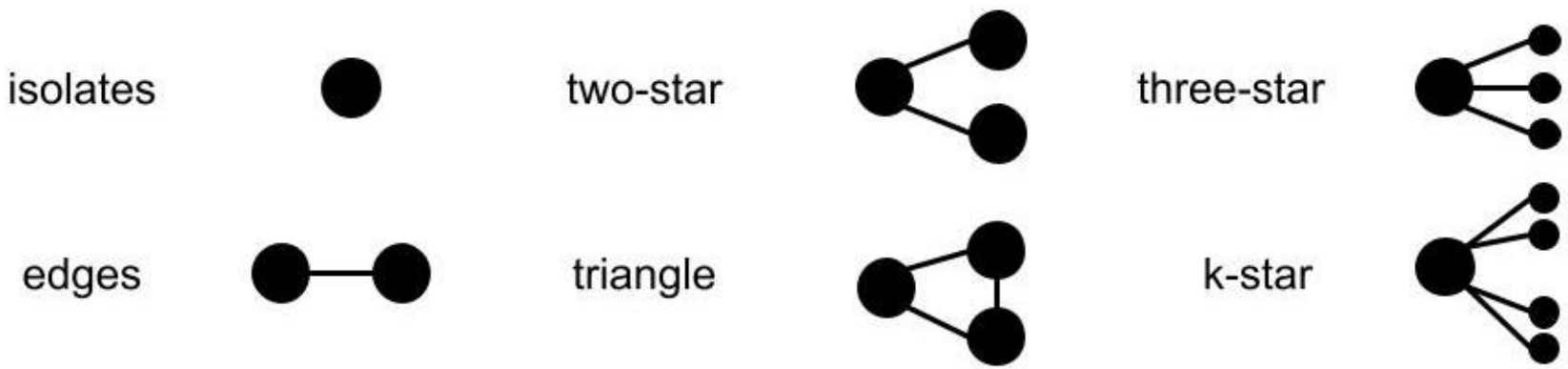
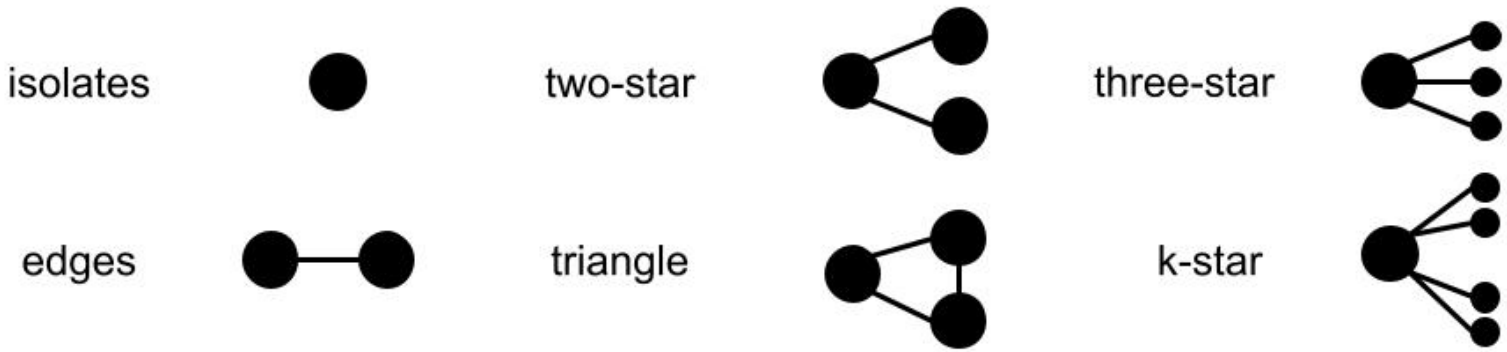
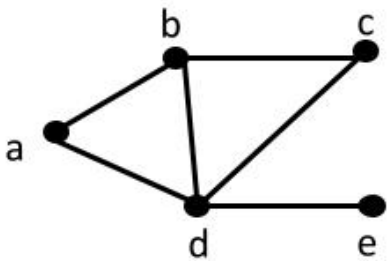


Figure: ERGM Structural Signatures for an Undirected Graph

ERGM Foundations



Example:



Edges: 6

a b c d e

2-Star: 1+3+1+6+0=11

3-Star: 0+1+0+4+0=5

4-Star: 0+0+0+1+0=1

Triangle: 2

ERGM Models

The goal of ERGM is to model the likelihood of structural signatures occurring compared to random chance. In addition to structural predictors, there can be additional node and dyad-level covariates.

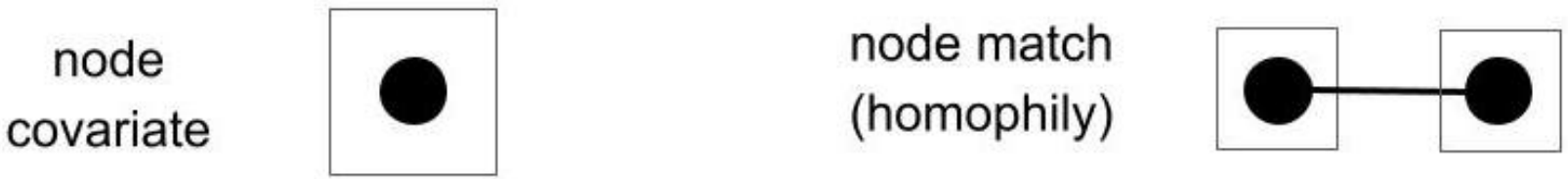


Figure: Examples of node and dyadic covariates

Connecting Structure to Probability

Recall that counts of structural signatures are sufficient statistics in an ERGM model. This is proven using the Hammersley-Clifford theorem (Frank and Strauss, 1986). This allows us to parameterize a probability distribution in the exponential family class.

ERGM Formulation

The general formula for ERGM is given by:

$$P(Y = y \mid \theta) = \frac{\exp(\theta^T g(Y))}{\kappa(\theta)}$$

- Y is the observed network
- $g(Y)$ is the vector of all known statistics (eg. structural signatures, node covariates)
- θ is the vector of model coefficients
- $\kappa(\theta)$ is the normalizing constant

ERGM Formulation

The general formula for ERGM is given by:

$$\underbrace{P(Y = y \mid \theta)}_{\text{Conditional Probability}} = \underbrace{\frac{\exp(\theta^T g(Y))}{\kappa(\theta)}}_{\text{Probability distribution across signatures}}$$

- Y is the observed network
- $g(Y)$ is the vector of all known statistics (eg. structural signatures, node covariates)
- θ is the vector of model coefficients
- $\kappa(\theta)$ is the normalizing constant

Overview

Defining the ERGM Estimation Objective

$$P(Y = y \mid \theta) = \frac{\exp(\theta^T g(Y))}{\kappa(\theta)}$$

Our goal is to find the coefficient or parameter vector θ such that the expectation of the vector of known statistics $g(Y)$ is equivalent to the observed statistics g_{obs} in the empirical graph: $E_\theta(g(X)) = g_{obs}$.

Unfortunately, this parameter vector cannot be solved analytically due to the intractable nature of the expectation and covariance matrix.

The Challenge of Achieving this Objective

We'll proceed using undirected networks without loss of generality.

There are $2^{n(n-1)/2}$ possible undirected networks that can be generated from a set of n nodes, making it nearly impossible to traverse the entire probability space of most graphs, even those with a small number of nodes. Thus, we want to take a more principled approach to estimating θ.

A graph with 15 nodes has 4×10^{31} *possible combinations, more than the number of stars in the universe:* 4×10^{22}

Estimation Techniques for Markov Dependence Model

Pseudolikelihood	MCMC MLE (Importance Sampling)	MCMC MLE (Robbins-Monro)	Bayesian Estimation
1986: Frank & Strauss	1992: Geyer & Thompson	2002 Snijders	2008: Koskinen
• Analytical solution • Computationally inexpensive • Unknown statistical properties • Potentially underestimated standard errors	• Simulation-based approach • Improved reliability • Implemented in major software packages (R statnet, PNet)	• Simulation-based approach • Approaches for handling degeneracy • Implemented in major software packages (R statnet, PNet)	• Simulation-based approach • Can be used for snowball sampling and missing data • Not implemented in major software packages

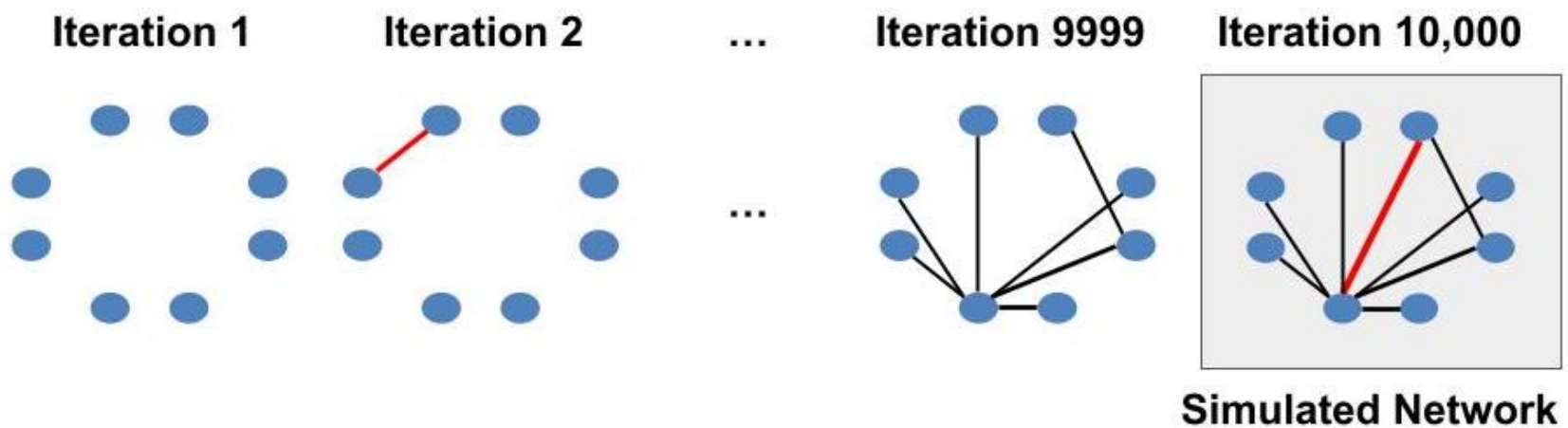
Estimation Techniques for Markov Dependence Model

Pseudolikelihood	MCMC MLE (Importance Sampling)	MCMC MLE (Robbins-Monro)	Bayesian Estimation
1986: Frank & Strauss	1992: Geyer & Thompson	2002: Snijders	2008: Koskinen
• Analytical solution • Computationally inexpensive • Unknown statistical properties • Potentially underestimated standard errors	• Simulation-based approach • Improved reliability • Implemented in major software packages (R, PNet)	• Simulation-based approach • Approaches for handling degeneracy • Implemented in major software packages (R, PNet)	• Simulation-based approach • Can be used for snowball sampling and missing data • Not implemented in major software packages

Estimation Approach - Part 1

Definition (Markov Chain Monte Carlo - MCMC)

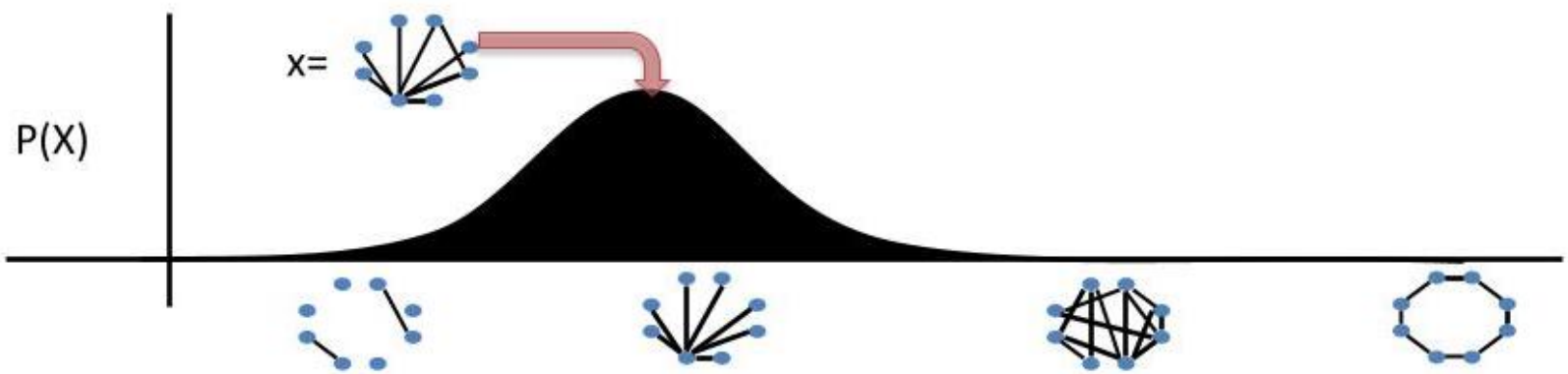
An MCMC method is used to simulate networks which can be used to estimate our parameters. The concept is to iteratively change the vector θ to generate graphs until they resemble the observed graph y_{obs}. Once we achieve a stationary distribution, we use the last graph in the sequence as our sample draw.



Estimation Approach - Part 2

Definition (Maximum Likelihood Estimation)

A Maximum Likelihood Estimate (MLE) provides an estimate of a desired parameter from a target distribution. In our case, we want to find the vector θ that makes the probability $P_\theta(y_{obs})$ as large as possible.



These networks were simulated using MCMC!

Northwestern University

SONIC
advancing the science of networks in communities

MCMC Graph Simulation

How do we simulate a graph?

There are two widely-used algorithms for generating a graph, given a set number of nodes and a parameter estimate θ.

Gibbs Sampling

- Sequential
- Simpler settings
- Conditional distribution

Metropolis Hastings

- Non-consecutive
- Complex settings
- Joint distribution

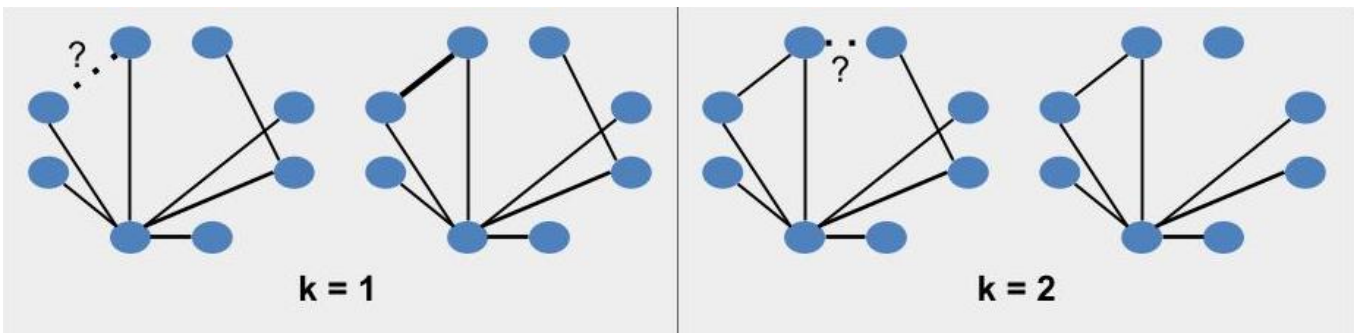
MCMC Graph Simulation - Option 1

Definition (Gibbs Sampling)

In each iteration k, an edge of the graph $Y^{(k)}$ is selected one at a time. The selected edge is set to one or zero in the following iteration $Y_{ij}^{(k+1)}$ according to the conditional probabilities:

$$P_\theta(Y_{ij}^{(k+1)} = a \mid Y^{(k)} = y^{(k)}) = P_\theta(Y_{ij} = a \mid Y_{hk} = y_{hk}^{(k)})$$

$$\text{for all } (h,k) \neq (i,j), a = 0, 1$$



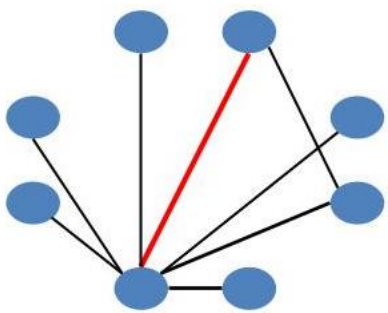
(continued on next slide)

MCMC Graph Simulation - Option 1

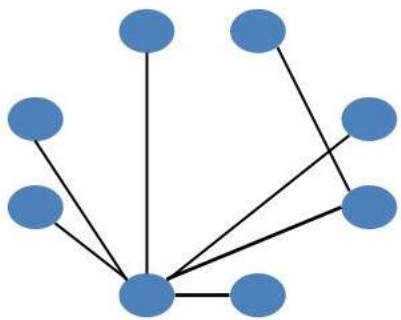
Definition (Gibbs Sampling)

The probability can be calculated by defining two adjacency matrices $y^{(ij1)}$ and $y^{(ij0)}$, where the (i,j) element is equal to 1 or 0, respectively, and all other elements are as they are in $y^{(k)}$ (recall $Y^{(k)} = y^{(k)}$). Then this is the conditional distribution:

$$logit\left[P_\theta(Y_{ij} = 1 | Y_{hk} = y_{hk})\right] = \theta^T\left[g(y^{(ij1)}) - g(y^{(ij0)})\right]$$



$y^{(ij1)}$



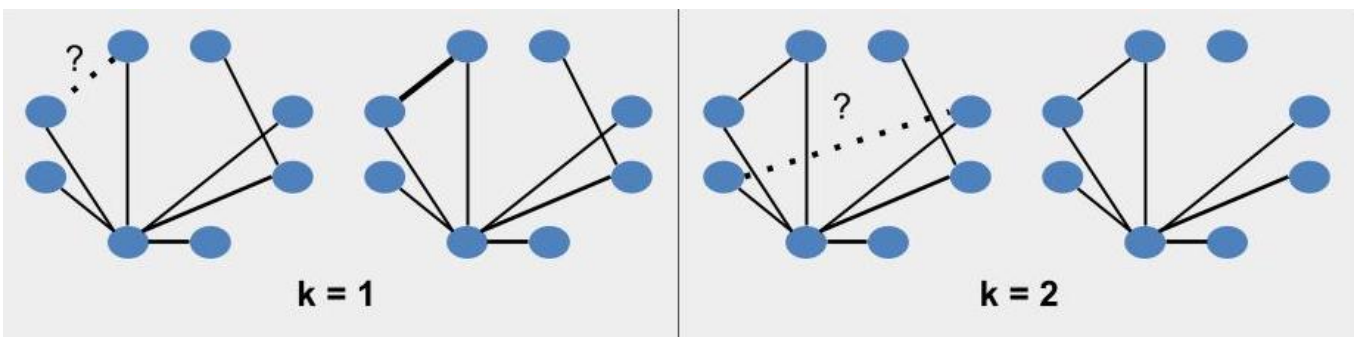
$y^{(ij0)}$

MCMC Graph Simulation - Option 2

Definition (Metropolis-Hastings Sampling)

Define a graph Y^* such that for a random pair of nodes (i,j), $Y^*_{ij} = 1 - Y^{(k)}_{ij}$. In each iteration, $Y^{(k+1)}$ is updated to Y^* with probability $\min\left\{1\ ,\ \frac{P_\theta(Y^*)}{P_\theta(Y)}\right\}$, and $Y^{(k+1)} = Y^{(k)}$ otherwise.

$$\frac{P_\theta(Y^*)}{P_\theta(Y^{(k)})} = \exp(\theta^T[u(Y^*) - u(Y^{(k)})])$$



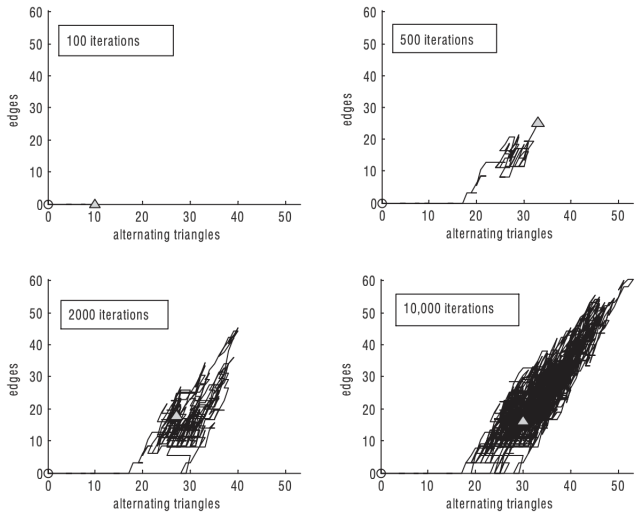
MCMC Graph Simulation - Which Option?

This application of Metropolis-Hastings differs from Gibbs in that it utilizes the previous graph iteration and thus changes the graph more frequently than Gibbs, resulting in higher efficiency (most of the time).

Thus, Metropolis-Hastings is the default option for generating graphs for our MCMC MLE estimation procedure.

MCMC Burn-in

The stochastic approximation algorithm draws from graphs simulated through MCMC. To ensure that the model approximates the desired distribution, simulations must burn-in, or run for a specified number of iterations before sampling a graph.

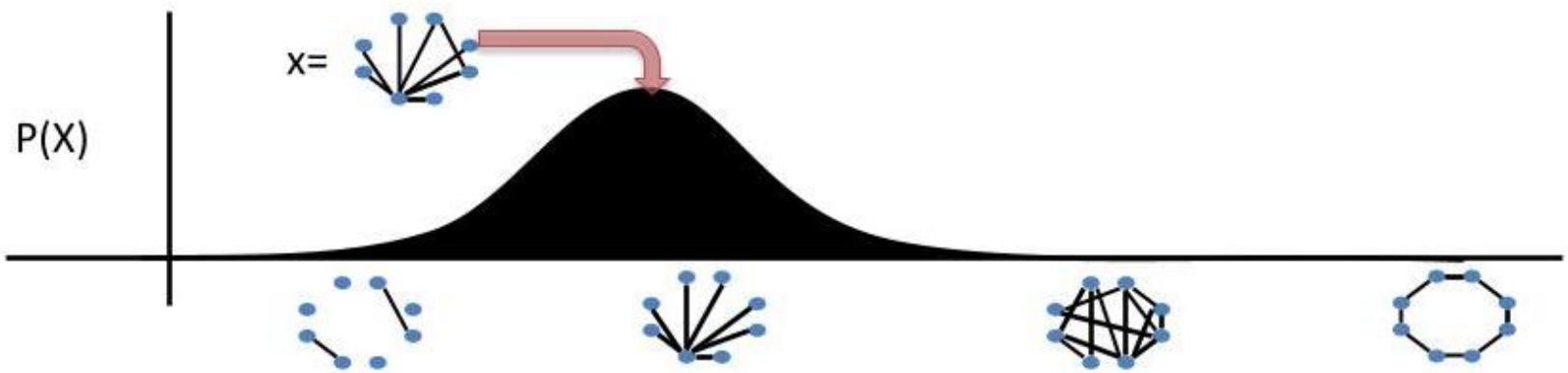


Lusher et al., 2013

Sufficient burn-in is an empirical approach to achieve asymptotic convergence to a target distribution.

Maximum Likelihood Estimation

We want to choose θ such that the expected value of our statistics is equal to the observed statistic: $E_\theta[g(Y)] = g_0$, where $g_0 = g(y_{obs})$ is the observed vector of statistics.



Maximum Likelihood Estimation

We want to choose θ such that the expected value of our statistics is equal to the observed statistic: $E_\theta[g(Y)] = g_0$, where $g_0 = g(y_{obs})$ is the observed vector of statistics.

- This θ is equivalent to the maximum likelihood estimate, which solves: $\max_\theta \left\{P(Y = y_{obs} \mid \theta)\right\} = \max_\theta \left\{ \log(P_\theta(y_{obs}))\right\}$
- This can be seen in the standard calculation:

$$
\begin{aligned}
0 = \frac{\delta}{\delta\theta}\left[\log(P_\theta(y_{obs}))\right] &= \frac{\delta}{\delta\theta}\left[\log\left(\frac{\exp(\theta^T g(y_{obs}))}{\kappa(\theta)}\right)\right] \\
&= g(y_{obs}) - \frac{\delta}{\delta\theta}\log\left(\sum_{y\in Y}\exp(\theta_1 g_1(y) + \cdots + \theta_p g_p(y))\right) \\
&= g(y_{obs}) - \sum_{y\in Y} g(y)P_\theta(y) \Leftrightarrow g_0 - E_\theta[g(Y)]
\end{aligned}
$$

(Hogg and Craig, 2012)

Maximum Likelihood Estimation

We need to *approximate* maximum likelihood estimates due to the intractability of our model.

1. We start by "guessing" a value of θ and generate a sample of graphs using simulation techniques.
2. Then, we compare our simulated graphs to the actual observed graph and use this information to improve our "guess" of θ.
3. The process in step (2) is repeated until the simulated graphs closely represent the observed graph.

Overview

Estimation Techniques for Markov Dependence Model

Pseudolikelihood	**MCMC MLE (Importance Sampling)**	MCMC MLE (Robbins-Monro)	Bayesian Estimation
1986: Frank & Strauss	1992: Geyer & Thompson	2002: Snijders	2008: Koskinen
• Analytical solution	• Simulation-based approach	• Simulation-based approach	• Simulation-based approach
• Computationally inexpensive	• Improved reliability	• Approaches for handling degeneracy	• Can be used for snowball sampling and missing data
• Unknown statistical properties	• Implemented in major software packages (R, PNet)	• Implemented in major software packages (R, PNet)	• Not implemented in major software packages
• Potentially underestimated standard errors			

Importance Sampling Overview

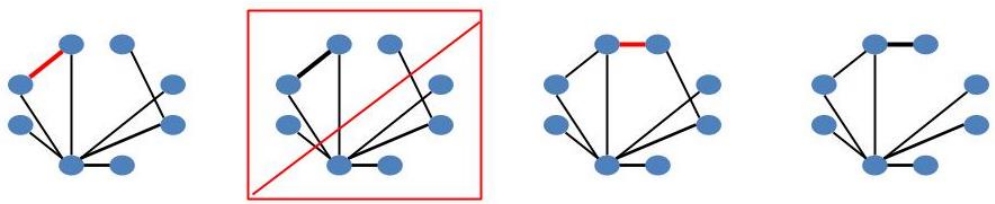
The Geyer-Thompson algorithm starts with simulating a large number of graphs from a provisional θ.

It treats this sample of graphs as the population of all graph possibilities. The estimate is corrected by a weighting term or likelihood ratio $w^{(m)}$.

- $\sum_{m=1}^{M} w^{(m)} g(y^{(m)}) \approx E_\theta(g(Y))$
- $w^{(m)} = \frac{\exp(\theta_1 - \tilde{\theta}_1) g_1(y^{(m)}) + \cdots + (\theta_p - \tilde{\theta}_p) g_p(y^{(m)})}{\sum_{k=1}^{M} \exp(\theta_1 - \tilde{\theta}_1) g_1(y^{(k)}) + \cdots + (\theta_p - \tilde{\theta}_p) g_p(y^{(k)})}$
- m is the graph iteration number
- p is the parameter index

Importance Sampling Overview

Using Gibbs or Metropolis Hastings, graphs are generated in a Markov chain. Samples are selected from the chain, with a thinning approach taken to reduce sample autocorrelation (k steps are discarded between sample points).



The parameter θ is approximated with a Fisher-scoring approach, where θ is updated iteratively until errors are sufficiently small.

Estimation Techniques for Markov Dependence Model

Pseudolikelihood	MCMC MLE (Importance Sampling)	**MCMC MLE (Robbins-Monro)**	Bayesian Estimation
1986: Frank & Strauss	1992: Geyer & Thompson	2002: Snijders	2008: Koskinen
• Analytical solution • Computationally inexpensive • Unknown statistical properties • Potentially underestimated standard errors	• Simulation-based approach • Improved reliability • Implemented in major software packages (R, PNet)	• Simulation-based approach • Approaches for handling degeneracy • Implemented in major software packages (R, PNet)	• Simulation-based approach • Can be used for snowball sampling and missing data • Not implemented in major software packages

Stochastic Approximation Algorithm Overview

Snijders' (2002) algorithm progresses through three main phases: initialization, estimation, and convergence.

1. Initialization involves choosing a starting graph $Y^{(0)}$ to begin the algorithm and computing an approximate covariance matrix for the estimation algorithm.
2. Estimation proceeds by iteratively generating new graphs $Y^{(m)}$ until termination conditions are met.
3. Convergence is the final step used for validating the estimator produced in phase 2 and computing approximate standard errors of the estimates.

Phase 1: Initialization

Basics

- The initial random graph is generated with a pre-determined number of nodes by assuming each edge Y_{ij} is determined independently with a probability 0.5 for the values 0 (not existing) and 1 (existing).
- The initial value of the parameter for the algorithm is $\theta^{(1)}$, which can be randomly picked.
- The observed network statistics is the vector g_0.
- The log of the normalization term is $\phi(\theta)$. $\log(k(\theta)) = -\log(\frac{1}{k(\theta)}) = \phi(\theta)$.

Phase 1: Initialization

Initial Graph Sample

Generate $N_1 = 7 + 3p$ (p denotes the number of parameters) independent networks $Y(n)$ according to the initial parameter $\theta^{(1)}$ using Gibbs or Metropolis updating steps, where $Y(n)$ represents a simulated network after n iterations.

- Rather than using edges, efficiency can be increased by updating larger groups of edges such as dyads or triplets.
- n represents the burn-in and should be adjusted based on ν, the number of nodes in the graph.

Phase 1: Initialization

Graph Inversion

To improve model performance for bimodal distributions and to reduce variance, this algorithm proposes an inversion at each step with a small probability such as 0.01. The inversion is then governed by another probability $p_r(y)$.

- In an inversion, a matrix $Y^{(k)}$ is replaced by its complement $Y^C = \mathbf{1} - y$, where $(\mathbf{1} - y)_{ij} = 1 - y_{ij}$ for all (i,j).
- Gibbs: $p_r(y) = \frac{\exp(\theta^T g(\mathbf{1}-y))}{\exp(\theta^T g(\mathbf{1}-y))+\exp(\theta^T g(y))}$
- Metropolis: $p_r(y) = \min\left\{1, \exp(\theta^T[g(\mathbf{1}-y) - g(y)])\right\}$

Phase 1: Initialization

Statistic Estimation

Let $P(n)$ represent the inversion probability of a graph: $P(n) = p_r(Y(n))$.

Calculate the sample mean of $g(Y)$ based on N_1 networks by the following formula:

$$\overline{g} = \frac{1}{N_1} \sum_{n=1}^{N_1} \Big(P(n) g[\mathbf{1} - Y(n)] + [1 - P(n)] g(Y(n)) \Big) \qquad (1)$$

Phase 1: Initialization

Covariance Matrix

Estimate the covariance matrix of $g(Y)$, which can be represented $D = D(\theta^{(1)}) = cov_{\theta^{(1)}}(g(Y))$.

$$D = \frac{1}{N_1} \sum_{n=1}^{N_1} \Big(P(n) g\big(\mathbf{1} - Y(n)\big)^T g\big(\mathbf{1} - Y(n)\big) + \big(1 - P(n)\big) g\big(Y(n)\big)^T g\big(Y(n)\big) - \overline{g}^T \overline{g} \Big)$$

Make a diagonal matrix from matrix D: $D_0 = diag(D)$

D is an approximation of the Hessian of $k(\theta)$ (Snijders et al., 2002)

Northwestern University

SONIC advancing the science of networks in communities

Phase 1: Initialization

Initialization Summary

What have we accomplished so far?

1. Generated an initial sample of graphs
2. Reviewed the graph inversion approach
3. Generated a covariance matrix approximation for Phase 2 using (1) and (2)

Phase 2: Estimation

Computing Expectation and Covariance

- Our MLE $\hat{\theta}$ can be asymptotically linked to $g(y)$ via the covariance matrix as follows:

$$\text{cov}_\theta(\theta) = \left(\text{cov}(g(Y))\right)^{-1}$$

- The gradient of the normalization term $\phi(\theta)$ is equal to the expectation of $g(Y)$, and the Hessian of the normalization term $\phi(\theta)$ is equal to the covariance of $g(Y)$.

$$\frac{\delta\phi(\theta)}{\delta\theta} = E[g(Y)] \text{ and } \frac{\delta^2\phi(\theta)}{\delta\theta^2} = \text{cov}(g(Y))$$

These equalities hold due to exponential family and under certain regularity conditions.

Phase 2: Estimation

Robbins-Monro Fundamentals

- The cov($g(Y)$) is estimated by D[1]. In the following algorithm, it is more efficient and still optimal to use the positive-definite diagonal matrix D_0 to estimate θ.
- To approximate the value of θ above, we would like to use the Newton-Raphson algorithm with iteration step:

$$\theta^{(n+1)} = \theta^{(n)} - \frac{\frac{\delta\phi(\theta)}{\delta\theta}(\theta^{(n)})}{\frac{\delta^2\phi(\theta)}{\delta\theta^2}(\theta^{(n)})}$$

- Since the derivatives are not computable, we instead approximate the second term iteratively over multiple subphases directed by differing step sizes.

[1] See slide 44

Phase 2: Estimation

Parameter Update

Within a subphase, each iteration generates a graph $Y(n)$ according to current $\theta^{(n)}$ and after each iteration, we update parameter θ according to the formula:
$\theta^{(n+1)} = \theta^{(n)} - a_n D_0^{-1} Z(n)$

- a_n is the step size, a sequence of positive numbers converging to 0. Within a subphase, a_n is a constant. We halve a_n when entering a new subphase. It's advisable to use an initial value $0.001 \leq a_1 \leq 0.1$, where a_1 is smaller if the starting $\theta^{(1)}$ is known to be close to the solution.
- $Z(n)$ is the expectation of statistics centered at the observed network in the n^{th} realization, i.e., $Z(n) = E[g(Y(n))] - g_0$, which is estimated by: $Z(n) = P(n)g(1 - Y(n)) + (1 - P(n))g(Y(n)) - g_0$

Phase 2: Estimation

Stopping Conditions

- At the end of each subphase, the average of $\theta^{(n)}$ is used as the new value of $\theta^{(n)}$.
- The average of $\theta^{(n)}$ in the last subphase is the final estimate of θ.
- The number of subphase iteration steps are bounded by terms $N_{2k}^{-} = 2^{4(k-1)/3}(7+p)$ and $N_{2k}^{+} = N_{2k}^{-} + 200$ for subphase k and total number of nodes p.
- The stopping condition within the bounds is for the sum of successive products $Z_k(n+1)Z_k(n) < 0$. This can also be interpreted as the trajectory of generated statistics crossing the observed statistics.
(i.e. $Z_k(Y^{(n-1)}) > Z_k(y_{obs})$ and $Z_k(Y^{(n)}) < Z_k(y_{obs})$).

Phase 2: Estimation

Summary

What have we accomplished in the estimation phase? We generated a maximum likelihood estimate of the parameter vector $\hat{\theta}$.

Phase 3: Convergence Validation

Graph Simulation

- Make 1000 realizations of a network $(Y^{(1)}, ..., Y^{(1000)})$ using the estimated parameter from Phase 2.
- Compare $\overline{g}_\theta$ to g_0, where g_0 is the statistics in the observed network and $\overline{g}_\theta = \frac{1}{1000} \sum_{n=1}^{1000} g(Y^{(n)})$.

Phase 3: Convergence Validation

Convergence Statistic

- Compute the convergence statistic $C = \frac{\bar{g}_\theta - g_{obs}}{SD_\theta(g_k(y^{(m)}))}$.
- If $|C| < 0.1$, then the model can be considered converged and therefore valid.
- If $|C| > 0.1$, then the estimated parameter θ is too far from the MLE and the estimation should be repeated, with the last obtained estimate used as the initial value for the parameter. Additionally, increasing burn-in can improve the convergence of the model.

Phase 3: Convergence Validation

Covariance Estimation

- Estimate $cov(g(Y))$ by

$$D = \frac{1}{1000} \sum_{n=1}^{1000} \Big(P(n) g\big(\mathbf{1} - Y(n)\big)^T g\big(\mathbf{1} - Y(n)\big) + \big(1 - P(n)\big) g\big(Y(n)\big)^T g\big(Y(n)\big) - \overline{g}^T \overline{g} \Big)$$

This equation is the same as the estimate used in Phase 1.

- Estimate the covariance matrix of the parameters by the formula: $cov(\theta) = (cov(g(Y)))^{-1}$.

Stochastic Approximation Algorithm Overview

We walked through the three main phases of Snijders' (2002) algorithm: initialization, estimation, and convergence.

1. Initialization involves choosing a starting graph $Y^{(0)}$ to begin the algorithm and computing an approximate covariance matrix for the estimation algorithm.
2. Estimation proceeds by iteratively generating new graphs $Y^{(m)}$ until termination conditions are met.
3. Convergence is the final step used for validating the estimator produced in phase 2 and computing approximate standard errors of the estimates.

Final Steps

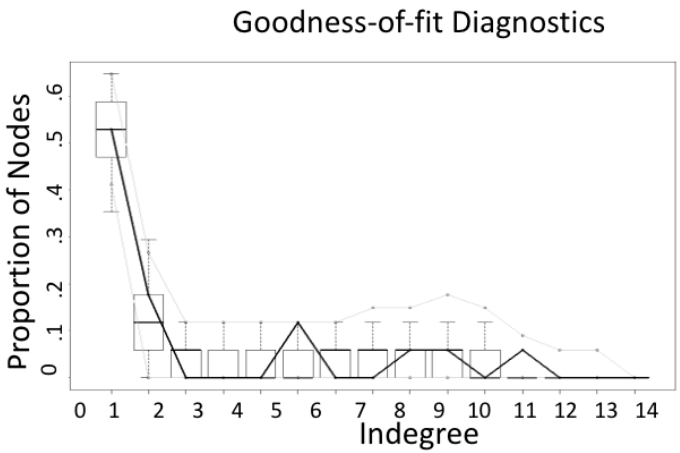
Goodness of Fit

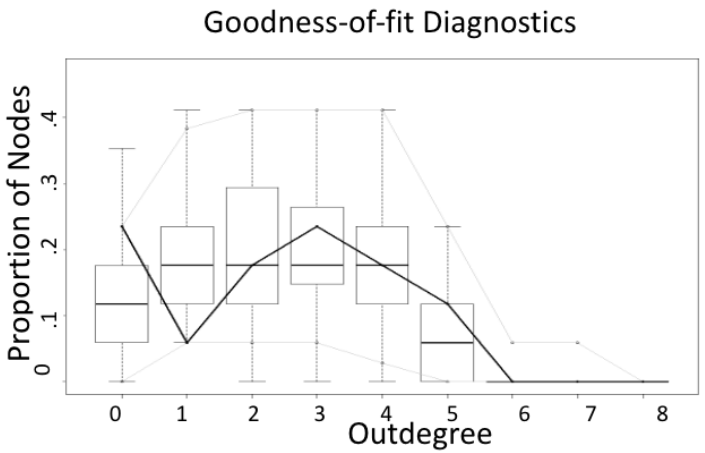
After completing the algorithm, we should proceed by confirming the model's goodness of fit (GOF). Any valid network statistics can be included in the GOF test. For example:

- All statistics in the model such as edge, triangle, stars for undirected networks; arc, in-stars, out-stars, transitive triad, etc. for directed networks
- Any statistics not included in the model but of interest
- Other global statistics, such as geodesic distance distribution, degree distribution, if supported by the software

Final Steps

Goodness of Fit





- Black dots and lines are observed network; Grey dots and lines denote simulated networks
- If the model is a good fit, black line should fall between or close enough to the grey lines

Final Steps

Model Tuning

- If model was a poor fit, consider adding or removing parameters (if theoretically sensible)
- Remember...a converged model is one that isn't going to improve with more iterations. So to improve fit, parameters must be altered
- Be careful to add parameters that have a clear substantive interpretation!

References


Snijders, T. The statistical evaluation of social network dynamics. *Sociological Methodology*. **31**, 361-395 (2001)


Geyer, C. & Thompson, E. Constrained Monte Carlo maximum likelihood for dependent data. *Journal Of The Royal Statistical Society: Series B (Methodological)*. **54**, 657-683 (1992)


Monge, P., Contractor, N. & Contractor, P. Theories of communication networks. (Oxford University Press, USA,2003)

References

Contractor, N., Wasserman, S. & Faust, K. Testing multitheoretical, multilevel hypotheses about organizational networks: An analytic framework and empirical example. *Academy Of Management Review*. **31**, 681-703 (2006)

Frank, O. & Strauss, D. Markov graphs. *Journal Of The American Statistical Association*. **81**, 832-842 (1986)

Besag, J. Spatial interaction and the statistical analysis of lattice systems. *Journal Of The Royal Statistical Society: Series B (Methodological)*. **36**, 192-225 (1974)

References

- Lusher, D., Koskinen, J. & Robins, G. Exponential random graph models for social networks: Theory, methods, and applications. (Cambridge University Press,2013)
- Hogg, R. & Craig, A. Introduction to mathematical statistics.(7th edition). *Englewood Hills, New Jersey*. (2012)
- Snijders, T. & Others Markov chain Monte Carlo estimation of exponential random graph models. *Journal Of Social Structure*. **3**, 1-40 (2002)