# Exponential Random Graph Models (ERGM)

SN & H Workshop, Duke University

Zack W Almquist

May 25, 2017

University of Minnesota

# Preliminaries

# Preliminaries

- This is a rather complete and very fast introduction to ERGMs
- All computation is done using the `statnet` package

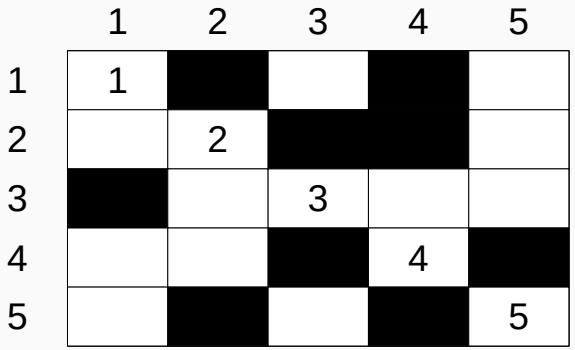
```
install.packages("ergm")
install.packages("network")
install.packages("coda")
```

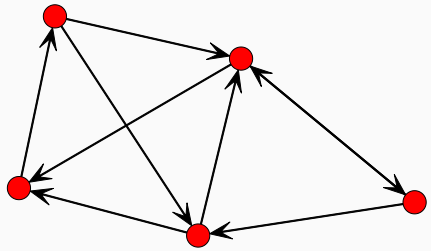
- All R code is provided within the slides for your convience
- These slides attempt to walk a tight rope of enough mathematical detail to be useful and enough intuitive insights to allow for use of ERGMs in your research

# Statistical Models for Social Networks

# Relational Data

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 1 | 0 |
| 3 | 1 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 1 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 0 |



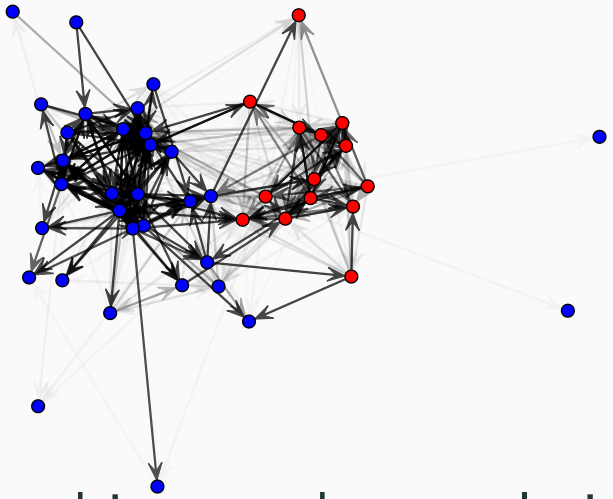



- A collection of entities and a set of measured relations between them
  - Entities: nodes, actors, egos, units, respondents
  - Relations: ties, links, edges
- Relations can be
  - Directed or undirected
  - Valued or dichotomous (binary)

# Core Areas of Statistical Network Analysis

1. Design-based inference: Network inference from sampled data
   - Design: survey and data-gathering procedures.
   - Inference: generaliation of sample data to full network
2. Statistical modeling: evaluation and fitting of network models
   - Testing: evaluation of competing theories of network formation
   - Estimation: evaluation of param in a presumed network model
   - Description: summaries of main network patterns
   - Prediction: prediction of missing or future network relations

# Data Analysis Goals




- How do network features drive our data analysis?
  1. How can we describe features of social relations?
     - E.g., reciprocity/sociability/transitivity: descriptive
  2. How can we identify nodes with similar network roles?
     - E.g., stochastic equivalence : node partitioning
  3. How do we relate the network to covariate information?
     - E.g., homophily: regression modeling

# Common Features of Interest

Beyond nodal and dyadic attributes, many networks exhibit the following features:

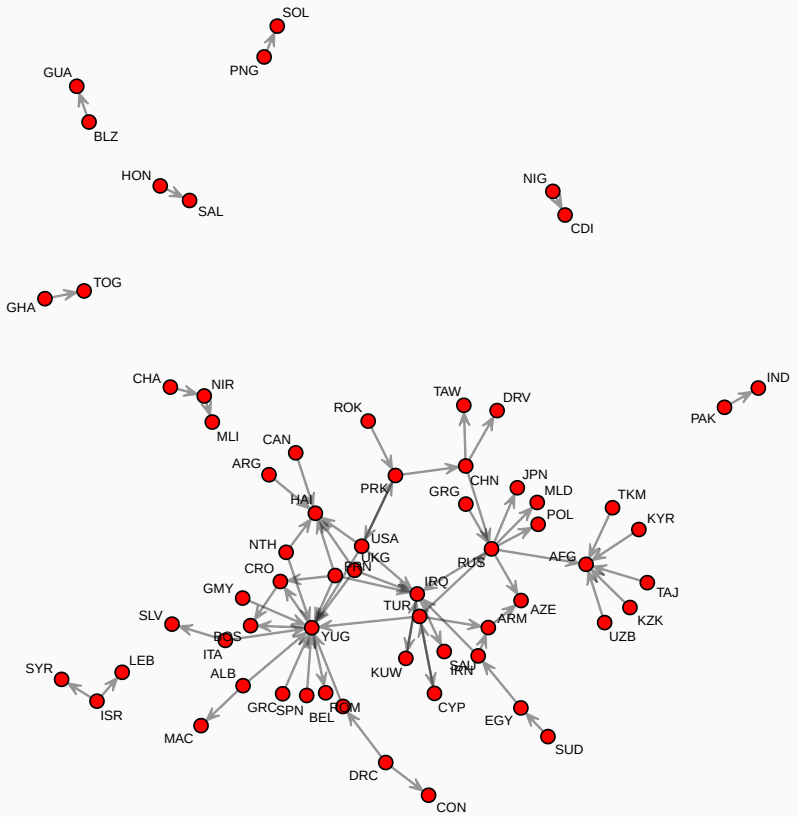
- Reciprocity of ties
- Degree heterogeneity among actors
  - Sociability, Popularity
- Homophily by actor attributes
  - Higher propensity to form ties between actors with similar attr
- Transitivity of relationships
  - Friends of friends have a higher propensity to be friends
- Balance of relationships
  - Liking those who dislike whom you dislike
- Equivalence of nodes
  - Some nodes may have identical/similar patterns of relationships

# Examples!

Now let's look at some examples. . .

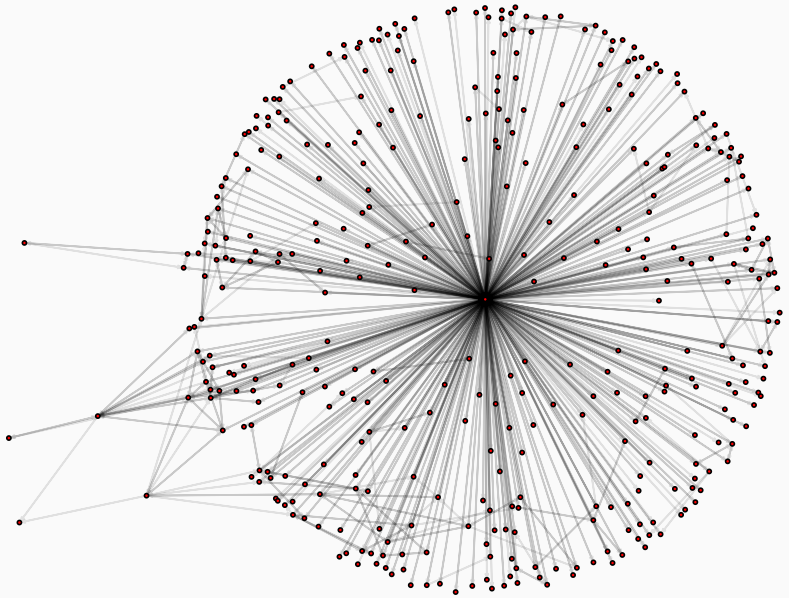
# Militarized Interstate Disputes

**1993 militarized interstate disputes (MIDs)**



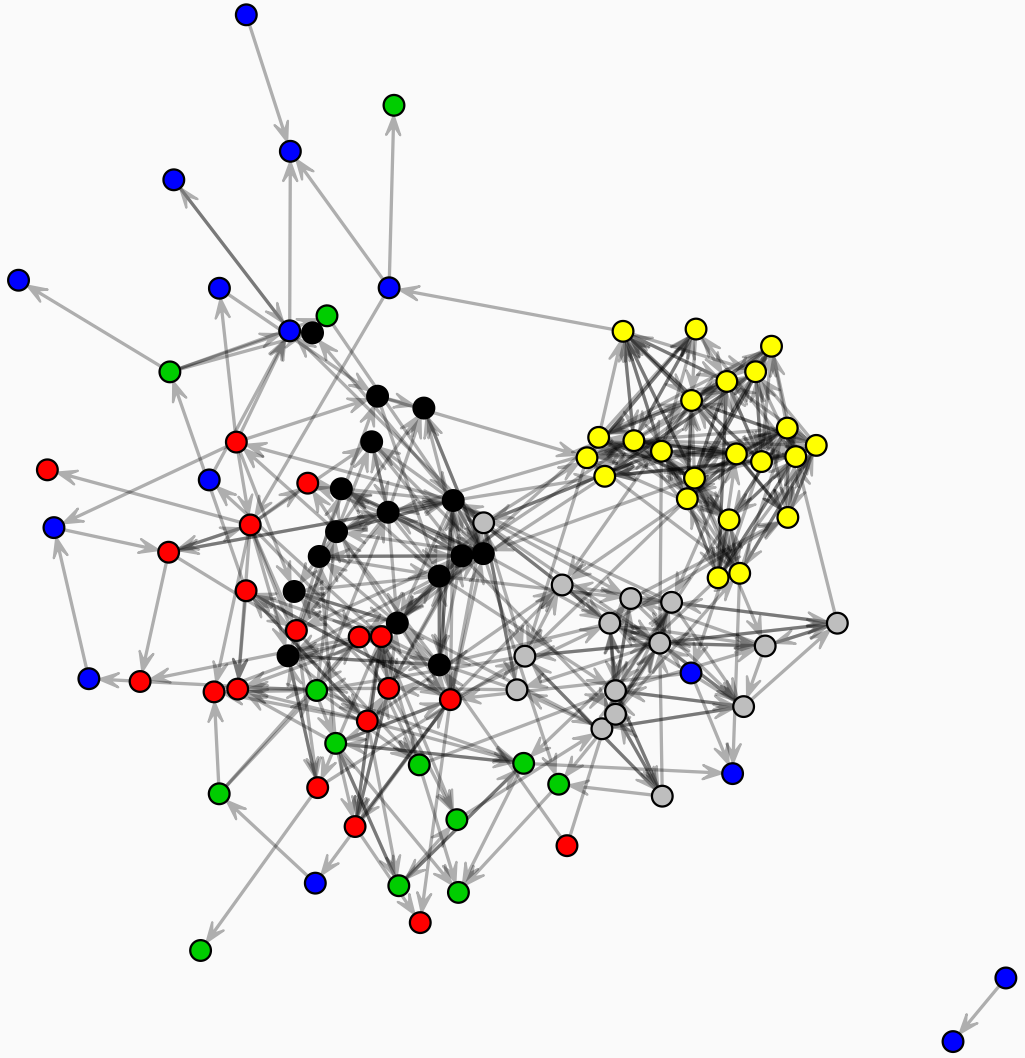

Correlates of War Project

# County-to-County Migration in the US



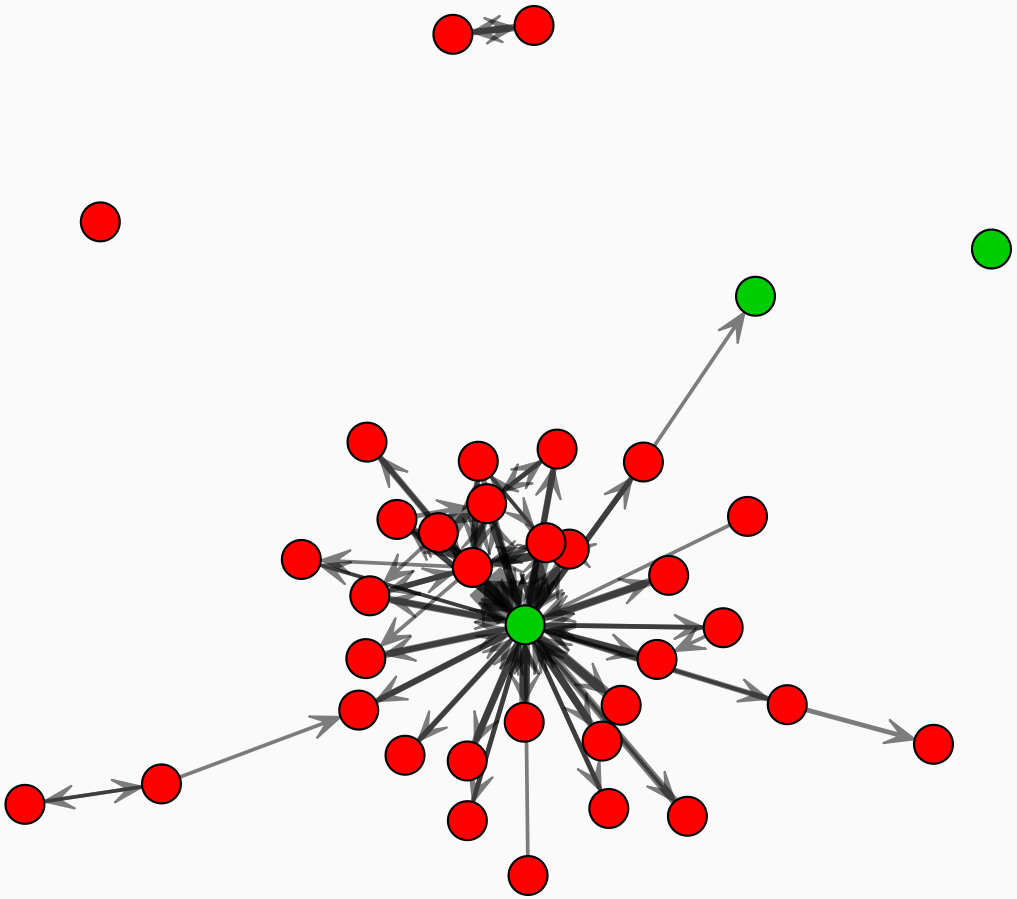

Threshold at 99%

# Faux Desert High (Simulation of an Add Health HS)

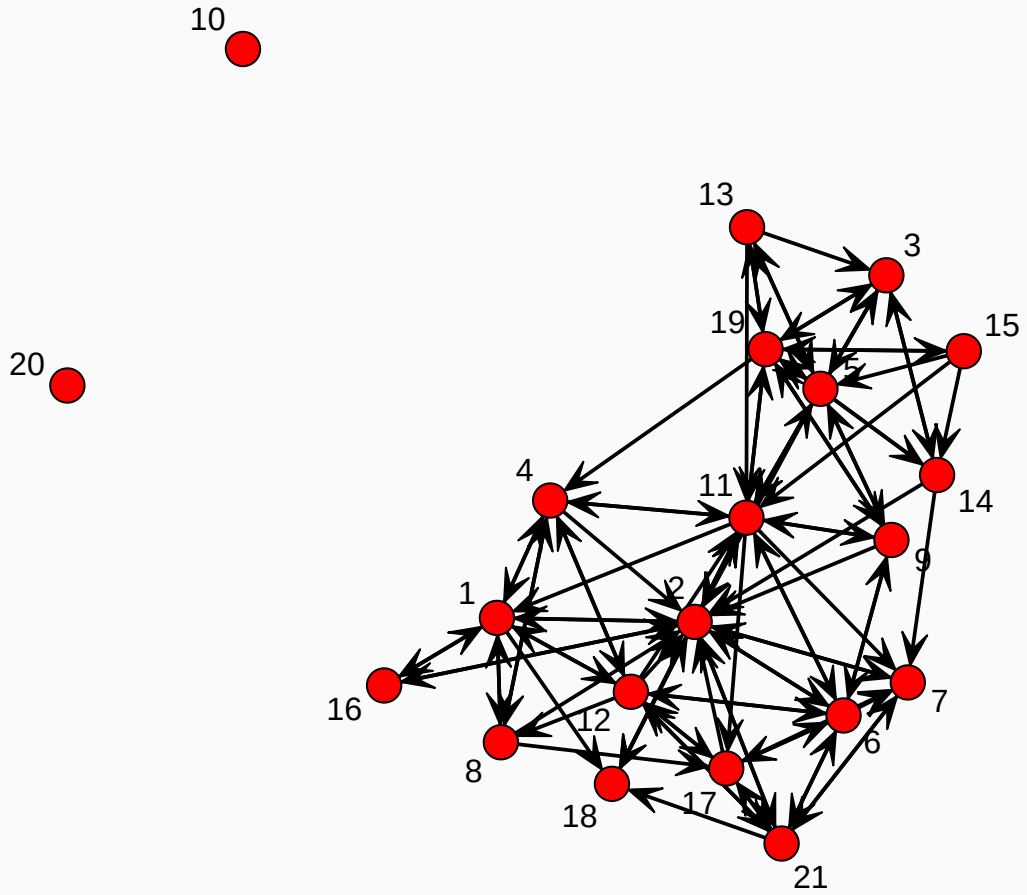


# World Trade Center Radio Communication

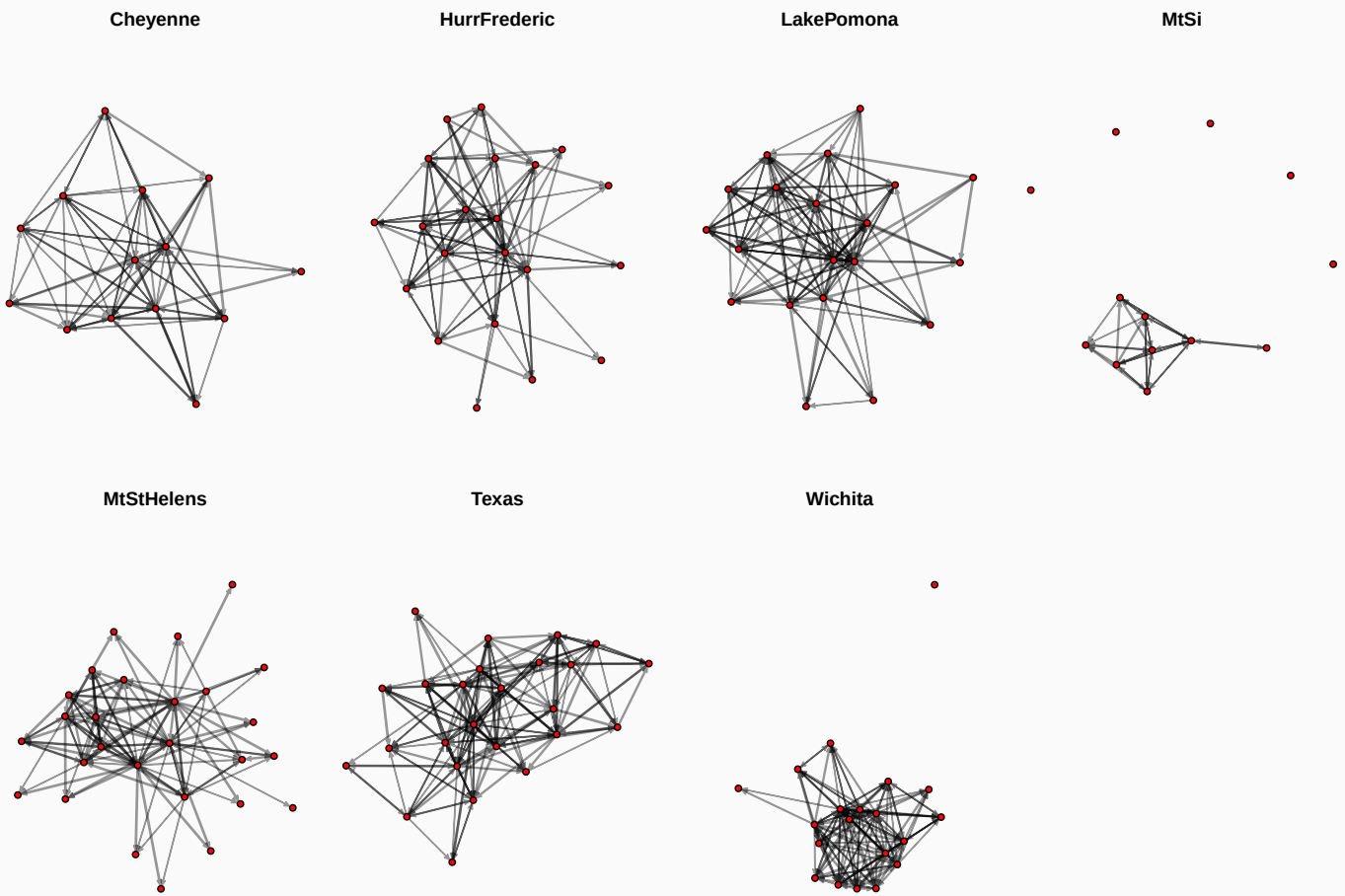
**Data set coded by Butts et al. (2007)**



# David Krackhardt (1987) Perceived Friendships



# SAR EMONs, All Reported Ties, from Drabek et al 1981



# Inferential Goals in the Regression Framework

$y_{ij}$ measures $i \to j$, $x_{ij}$ is a vector of explanatory variables

$$Y = \begin{pmatrix} y_{11} & y_{12} & y_{13} & NA & y_{15} & \cdots \\ y_{21} & y_{22} & y_{23} & y_{24} & y_{25} & \cdots \\ y_{31} & y_{32} & y_{33} & y_{34} & y_{35} & \cdots \\ y_{41} & y_{42} & y_{43} & y_{44} & y_{45} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \end{pmatrix} \quad X = \begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & \cdots \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & \cdots \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & \cdots \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \end{pmatrix}$$

Consider a basic (generalized) linear model

$$y_{ij} \sim \beta^T x_{ij} + e_{ij}$$

A model can provide

- A measure of the association between $X$ and $Y$: $\hat{\beta}$, $se(\hat{\beta})$
- Imputations of missing observations: $\Pr(y_{14}|Y, X)$
- A probabilistic description of network features: $g(\tilde{Y})$, $\tilde{Y} \sim \Pr(\tilde{Y}|Y, X)$

# Statistical Models for Social Networks

- A social network is defined as a set of $n$ entities (e.g., social "actors") and a relationship (e.g., friendship) between each pair of entities

$$Y_{ij} = \begin{cases} 1 & \text{relationship from actor } i \text{ to actor } j \\ 0 & \text{otherwise} \end{cases}$$

- Often $Y := [Y_{ij}]_{n \times n}$ is called a sociomatrix
- And, graphical representation of $Y$ a sociogram
  - Diagonal typically undefined or 0 (i.e., $Y_{ii} = NA$)
  - $Y$ represents a random network with nodes as the actors and edges the relationship
- The basic problem of stochastic modeling is to specify a distribution for $Y$, i.e. $\Pr(Y = y)$

# A Framework for Network Modeling: ERGM

- Let $\mathcal{Y}$ be the sample space of $Y$, e.g. $\{0,1\}^N$
- Any model-class for the multivariate distribution of $Y$ can be parameterized in the form

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

  - $\theta \in \Theta \subset R^q$ $q$-vector of parameters
  - $g(y)$ $q$-vector of network statistics
  - For a "saturated" model-class $q = 2^{|\mathcal{Y}|} - 1$
  - $\kappa(\theta)$ distribution normalizing constant

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

# Classic Probability Models for Networks

# Classics: Bernoulli Graphs

- $Y_{ij}$ are independent but have arbitrary distributions (conditional independence of edges)

$$\Pr(Y = y) = \frac{\exp\left\{\sum_{ij} \theta_{ij} y_{ij}\right\}}{\kappa(\theta)}, \qquad y \in \mathcal{Y}$$

$$g_{ij}(y) = y_{ij}, \quad i, j = 1, \ldots, n \qquad q = N$$

$$\theta_{ij} = \operatorname{logit}(\Pr(Y_{ij} = 1))$$

$$\kappa(\theta) = \prod_{ij} (1 + \exp(\theta_{ij}))$$

- $Y_{ij}$ can depend on dyadic covariates $X_{ij}$

$$\theta_{ij} = X_{ij}\beta$$

# Classics: Holland and Leinhardt's $p_1$ model

- Holland and Leinhardt (1981) proposed a general dyad independence model
- Also, a homogeneous version they refer to as the $p_1$ model

$$\Pr(Y = y) = \frac{\exp\{\rho \sum_{i<j} y_{ij} y_{ji} + \phi y_{++} + \sum_i \alpha_i y_{i+} + \sum_j \beta_j y_{+j}\}}{\kappa(\rho, \alpha, \beta, \phi)}$$

  - Where $\theta = (\rho, \alpha, \beta, \phi)$
    - $\phi$ controls the expected number of edges
    - $\rho$ represent the expected tendency toward reciprocation
    - $\alpha_i$ productivity of node $i$
    - $\beta_j$ attractiveness of node $j$

# Classics: Nodal Markov Statistics

- Frank and Strauss (1986)
  - Motivated by notions of "symmetry" and "homogeneity"
  - Edges in $Y$ that do not share an actor are conditionally independent given the rest of the network
- Analogous to nearest neighbor ideas in spatial statistics
- Degree distribution: $d_k(y) =$ proportion of nodes of degree $k$ in $y$
- $k$-star distribution: $s_k(y) =$ proportion of $k$-stars in graph $y$
- triangels: $t(y) =$ proportion of triangles in the graph y.

# Connecting Logistic Network Regression to Bernoulli Graphs

# Review: Logistic Network Regression

- A simple family of models for predicting unvalued ties
  - Special case of standard logistic regression
  - Dependent variable is a network adjacency matrix
- Model form:

$$\log\left(\frac{\Pr(Y_{ij}=1)}{\Pr(Y_{ij}=0)}\right) = \theta_1 X_{ij1} + \theta_2 X_{ij2} + \cdots + \theta_k X_{ijk} = \theta^T X_{ij}$$

- Where $Y_{ij}$ is the value of the edge from $i$ to $j$ on the dependant relation, $X_{ijk}$ is the value of the $k$th predictor for the $(i,j)$ ordered, and $\theta_1, \ldots, \theta_k$ are coefficients

$$\log(\frac{p}{1-p}) = \text{logit}(p), \text{maps } (0,1) \text{ to } (-\infty, \infty)$$

# Logistic Network Regression in Practice

- Interpretation: Effects are on logit scale
  - Unit change in $i$th covariate multiplies odds by $\exp(\theta_i)$
- We are assuming that edges are independent predictors, covariates
- QAP tests also apply (warning: known tests may not be robust to third variable effects, etc)
- In `statnet`, two ways to perform it
  - `netlogit`, specialized tool in package `sna`
  - `ergm`, generalized ERG function

# Example: Florentine families

- This is a data set of marriage ties among Renaissance Florentine families. The data is originally from Padgett (1994)

```
library(ergm)

data(florentine)
m1 = ergm(flomarriage ~ edges + nodecov("wealth"))
m2 = ergm(flomarriage ~ edges + absdiff("wealth"))
```

# Example: Florentine families

```
Formula 1: flomarriage ~ edges + nodecov("wealth")
Formula 2: flomarriage ~ edges + absdiff("wealth")

                        Estimate Std. Error MCMC % p-value
Model 1: edges            -2.595      0.536      0   0.000
Model 1: nodecov.wealth    0.011      0.005      0   0.026
Model 2: edges            -2.302      0.402      0   0.000
Model 2: absdiff.wealth    0.016      0.006      0   0.013
BIC
Model 1: 112.68 Model 2: 111.53
```

# ERGM

# Moving Beyond the Logistic Case

- The logistic model works for dichotomous data, but is still very limiting
  - No way to model conditional dependence among edges
    - E.g., true triad closure bias, reciprocity
- A more general framework: discrete exponential families
  - Very general way of representing discrete distributions
  - Turns up frequently in statistics, physics, etc.

# The Probability of a Graph (ERGM)

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

- This is the probability of a single graph
- Also, the likelihood function for the general model
- The normalizing constant is summed over all possible graphs

# Interpreting ERGMS: Goal

- To re-express the probability of the graph in terms of the probabilities of an individual tie
- This gives a "local" view of the model
- And some insight into what the $\theta$ coefficients mean

# Interpreting ERGMS: Some Notation

In order to re-express the probability of the graph in terms of the probabilities of a tie, we need to introduce some notation:

- $Y_{ij}^{+} = \{Y \text{ with } Y_{ij} = 1\}$ the graph w/ the $(i,j)$th dyad set to 1
- $Y_{ij}^{-} = \{Y \text{ with } Y_{ij} = 0\}$ the graph w/ the $(i,j)$th dyad set to 0
- $Y_{ij}^{c} = \{Y_{kl} \text{ with } (k,l) \neq (i,j)\}$ all dyads except (i,j)

# Interpreting ERGMS: The conditional probability of a link

This is a simple logical re-expression of $\Pr(Y = y) = \frac{\exp\{\theta^T g(y)\}}{\kappa(\theta, \mathcal{Y})}$

$$\begin{aligned}\Pr(Y_{ij} = 1 | Y_{ij}^c) &= \frac{\Pr(Y = y_{ij}^+)}{\Pr(Y = y_{ij}^+) + \Pr(Y = y_{ij}^-)} \\ &= \frac{\exp\{\theta^T g(y_{ij}^+)\}}{\exp\{\theta^T g(y_{ij}^+)\} + \exp\{\theta^T g(y_{ij}^-)\}}\end{aligned}$$

Note: the $\kappa(\theta)$ has canceled out, but . . . there is an even simpler expression, in terms of the odds

# Interpreting ERGMS: the Conditional log-odds of a link

Reminder: $\text{logit}(p) = \log\left(\frac{p}{(1-p)}\right)$

- Given, $\Pr(Y_{ij} = 1|Y_{ij}^c) = \frac{\exp\{\theta^T g(y_{ij}^+)\}}{\exp\{\theta^T g(y_{ij}^+)\}+\exp\{\theta^T g(y_{ij}^-)\}}$
- Then

$$\log\left\{\frac{\Pr(Y_{ij} = 1|Y_{ij}^c)}{\Pr(Y_{ij} = 0|Y_{ij}^c)}\right\} = \theta^T[g(y_{ij}^+) - g(y_{ij}^-)]$$
$$= \theta^T \delta(y_{ij})$$

Note: $\delta(y_{ij})$ is known as the change statistic

# Interpreting ERGMS: Interpreting $\theta$

- Simple case, The Bernoulli model, where $g(y)$ is just the number of ties in the graph

$$\text{logit}(\Pr(Y_{ij} = 1 | Y_{ij}^c)) = \theta\delta(y)$$

- For the edges term (and dyad independent term), $\delta(y)$ is the change in the statistic when toggling $y_{ij}$ from 0 to 1 is 1, so

$$\log\left(\frac{p}{(1-p)}\right) = \theta$$

and

$$p = \frac{e^{\theta}}{(1+e^{\theta})}$$

# Interpreting ERGMS: Review

- So, the conditional probability of an $(i, j)$ edge is simply the inverse logit of $\theta^T \delta(y_{ij})$
- One idea: to find $\theta$, why not set this up as a logistic network regression problem (i.e., regressing y on $\delta$)?
  - This is an "autologistic regression," and the resulting estimator is known as a *pseudolikelihood* estimator (more on this later)
- Problem: the probability here is only conditional – can use for any one $ij$, but the joint likelihood of $y$ is not generally the product of $\Pr(Y_{ij} = y_{ij} | Y_{ij}^c)$
  - Another view: $y$ appears on both sides – can't regress w/out accounting for the "feedback" (i.e., dependence) among edges
  - Does work iff edges are indepenedent
- Still, useful aid in interpretation!

# MLE and Statistical Inference

# MLE and Statistical Inference

- What is Maximum Likelihood Estimation (MLE)?
- The likelihood function for the general ERGM is:

$$L(\theta) = \Pr(Y = y|\theta) = \frac{\exp(\theta^T g(y, X))}{\kappa(\theta)}$$

- We want to find the value of $\theta$ that maximizes the probability of our data $(Y, X)$
- But the function depends on $\kappa(\theta)$, which makes direct calculation (and thus maximization) difficult (can be done for small networks of size up to 7)

# Fitting ERGs to Data: Computing the MLE

**Even simple models are too complex to get analytical solutions!**

- ERG computations are too difficult to perform directly
  - Simulation used for purely computational purposes (e.g., dealing with the normalizing factors)
- No (effective) way to draw directly from ERG distribution; have to use approximation algorithms
- Primary tool: Markov chain Monte Carlo (MCMC)
  - Iterative method for simulating draws from a given distirbution
  - Algorithm is approximate (although often very, very good)
- What does this mean?
  - MCMC requires more "care and feeding" than simple methods
  - Algorithm can fail, requiring user intervention (rare in classic regression context!)

# Fitting ERGs to Data: MCMC

- **Markov chain**
  - Stochastic process $X_1, X_2, \ldots$ on a state space $S$, such that $\Pr(X_i|X_{i-1}, X_{i-2}, \ldots) = \Pr(X_i|X_{i-1})$ (i.e., only the previous state matters – this is the Markov condition)
- **Monte Carlo procedure**
  - Any procedure which uses randomization to perform a computation, fixed execution time and uncertain output (compare w/ Las Vegas procedures)
- **Markov chain Monte Carlo (MCMC)**
  - Family of procedures using Markov chains to perform computations and/or simulate target distributions; often, these cannot be done any other way

# Fitting ERGs to Data: MCMC

- **Important Example: Metropolis Algorithm**
  - Given $X_i$, draw $X^{'}$ from $q(X_i)$; w/probability min $\left(1, \frac{\Pr(X^{'})}{\Pr(X_i)}\right)$, let $X_{i+1} = X^{'}$, else $X_{i+1} = X_i$. Repeat for $i+1, i+2$, etc.
  - Started w/arbitrary $X_0, X_0, X_1, \ldots, X_n$ converges to $p(X)$ in distribution as $n \to \infty$
  - Requires some constraints on $q$, but is very general - used when we can't sample from target distribution $p$ directly (as when $p$ is an ERG distribution)

# Fitting ERGs to Data: Computing the MLE

- `statnet` employs MCMC methods and MCMC-MLE methods to perform likelihood-based inference
- What happens when you run `ergm`?
  - First guess at $\theta$ done using the MPLE
  - Simulation of $y_1, \ldots, y_n$ based on the initial guess
  - The simulated sample is used to find $\theta$ using MLE
  - Previous two steps are iterated for good measure (since initial estimate is likely off)

# Fitting ERGs to Data: `statnet`

No need to this yourself, software exists to handle this procedure!

- Dedicated statnet package for fitting, simulating models in ERG form
- Basic call structure:

```
ergm(y ~ term1(arg) + term2(arg))
```

- All available terms can be found in:

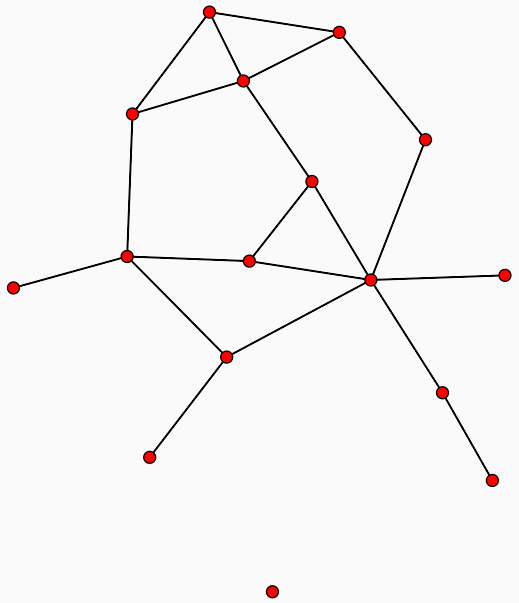
```
help("ergm-terms")
```

- Summary, print and other methods can be used on the "ergm" objects
- `simulate` command can also be used take draws from the fitted model

# Fitting ERGs to Data: `statnet`

Let's do a simple example (remember the florentine network):

```
library(ergm)
data(florentine)
plot(flomarriage)
```



# Fitting ERGs to Data: statnet

```
# Begin with the most basic model: a lone edges term
flom.e <- ergm(flomarriage ~ edges)  # Fit the model
```

```
Evaluating log-likelihood at the estimate.
```

```
flom.e  # Print it
```

```
MLE Coefficients:
 edges
-1.609
```

# Fitting ERGs to Data: statnet

```r
summary(flom.e)  # Get a summary
```

```
==========================
Summary of model fit
==========================

Formula:   flomarriage ~ edges

Iterations:  5 out of 20

Monte Carlo MLE Results:
      Estimate Std. Error MCMC % p-value
edges  -1.6094     0.2449      0  <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.)
```

# Fitting ERGs to Data: statnet

```
# Triangles?
flom.et <- ergm(flomarriage ~ edges + triangle)
```

```
Starting maximum likelihood estimation via MCMLE:
Iteration 1 of at most 20:
The log-likelihood improved by 0.002407
Step length converged once. Increasing MCMC sample size.
Iteration 2 of at most 20:
The log-likelihood improved by 0.0009785
Step length converged twice. Stopping.
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

This model was fit using MCMC.  To examine model diagnostics and check for degeneracy, use the mcmc.diagno
```

# Fitting ERGs to Data: statnet

```
summary(flom.et)
```

```
==========================
Summary of model fit
==========================

Formula:   flomarriage ~ edges + triangle

Iterations:  2 out of 20

Monte Carlo MLE Results:
         Estimate Std. Error MCMC % p-value
edges     -1.6793     0.3553      0  <1e-04 ***
triangle   0.1571     0.5825      0   0.788
---
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: 117.6    (Smaller is better.)
```

# Fitting ERGs to Data: Some Basic Families

- Several familiar and/or famous model families can be fit in `ergm`
- Bernoulli Graph (N,p): edge term only:

```
data(samplike)
ergm(samplike ~ edges)
```

- Bernoulli Graph with covariates:

```
ergm(samplike ~ edges + nodematch("group"))
```

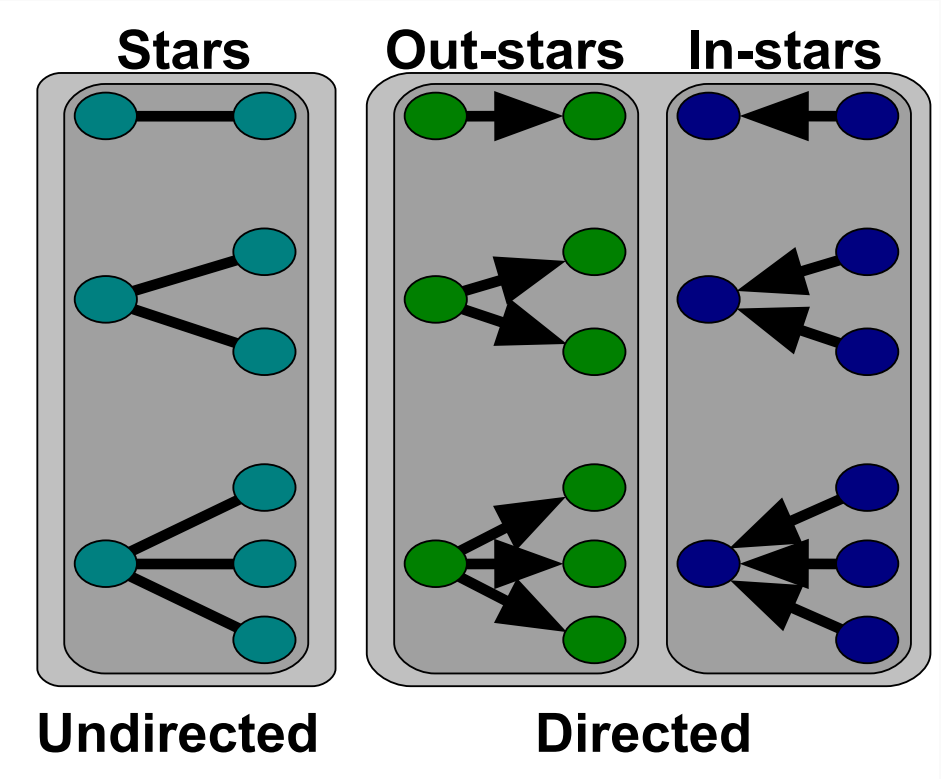
- $p_1$: edge, row-sum, col-sum and mutuality (or any subset thereof)

```
ergm(samplike ~ edges + sender + receiver + mutual)
```

# Fitting ERGs to Data: Beyond Independence

## Star Terms

- **Simple subgraph census terms**
  - k-stars: number of subgraphs isomorphic to $K_{ij}$
  - k-in/out-stars: number of subgraphs isomorphic to orientation of $K_{ij}$
- **Interpretations**
  - Tendency of edges to "stick together" on endpoints ("edge clustering")
  - Fixes moments of the degree distribution
    - 1-star fix mean degree, 2 star fixes variance



# Let's try it!

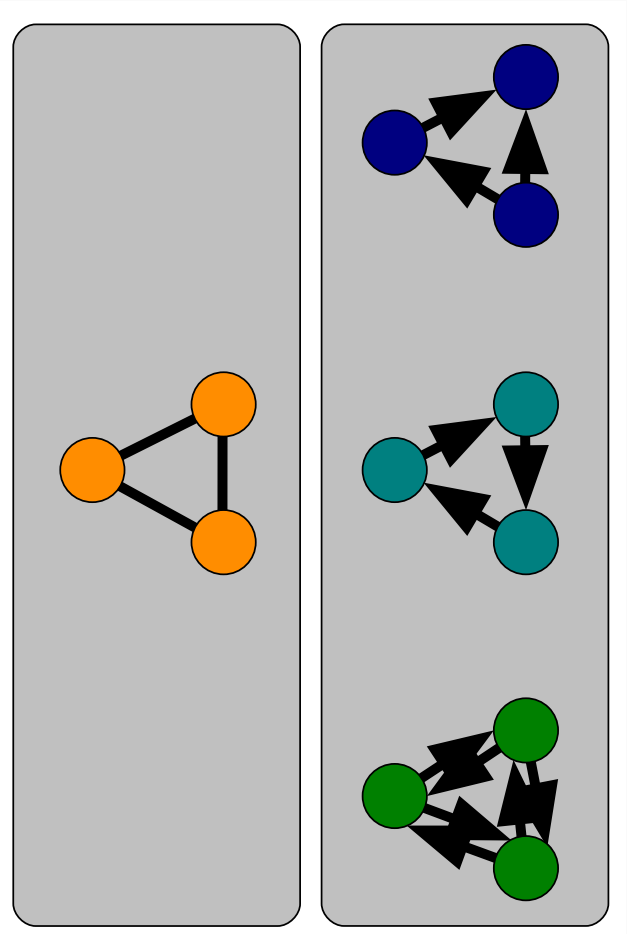
```r
# How about 2-stars?
flom.ets <- ergm(flomarriage ~ edges + kstar(2))
summary(flom.ets)
```

```
==========================
Summary of model fit
==========================

Formula:   flomarriage ~ edges + kstar(2)

Iterations:  2 out of 20

Monte Carlo MLE Results:
       Estimate Std. Error MCMC % p-value
edges  -1.67300    0.82894      0  0.0458 *
kstar2  0.01263    0.16859      0  0.9404
---
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.)
```

# Fitting ERGs to Data: Beyond Independence

## Triad Census Terms

- **Most basic terms for endogenous clustering**
  - Each term counts subgraphs isomorphic to triads of a given type (i.e., elements of the triad census)
  - In practice, triangles, cycles, and transitives most often used
- **Interpretations**
  - Tendencies toward transitive closure, cycles, etc.
  - Transitivity can be an indicator of latent hierarchy
  - Cyclicity can be an indicator of extended reciprocity



# Let's try it!

```r
# Triangles?
flom.et <- ergm(flomarriage ~ edges + triangle)
summary(flom.et)
```

```
==========================
Summary of model fit
==========================

Formula:   flomarriage ~ edges + triangle

Iterations:  2 out of 20

Monte Carlo MLE Results:
         Estimate Std. Error MCMC % p-value
edges     -1.6820     0.3474      0  <1e-04 ***
triangle   0.1677     0.5772      0   0.772
---
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: 117.6    (Smaller is better.)
```

# Model Adequacy and Issues of Degeneracy

# Model Adequacy and Issues of Degeneracy

**Major Problems**

1. Model Degeneracy
2. Quality of the Simulations
3. Model Adquacy Assessment

# Model Degeneracy

Unstable ERGM specifications place almost all probability mass on a small set of graphs that do not resemble the observed graph, and the rest of the graphs in the model space have negligible probability

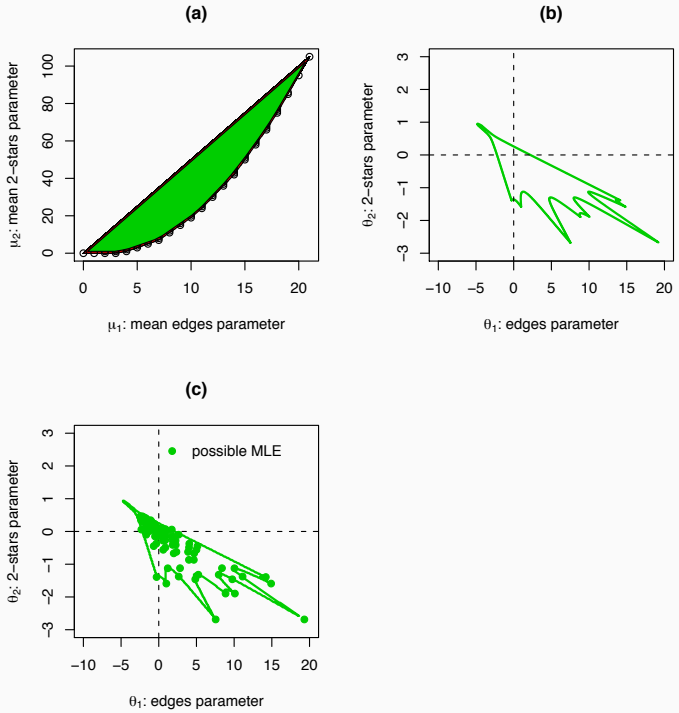
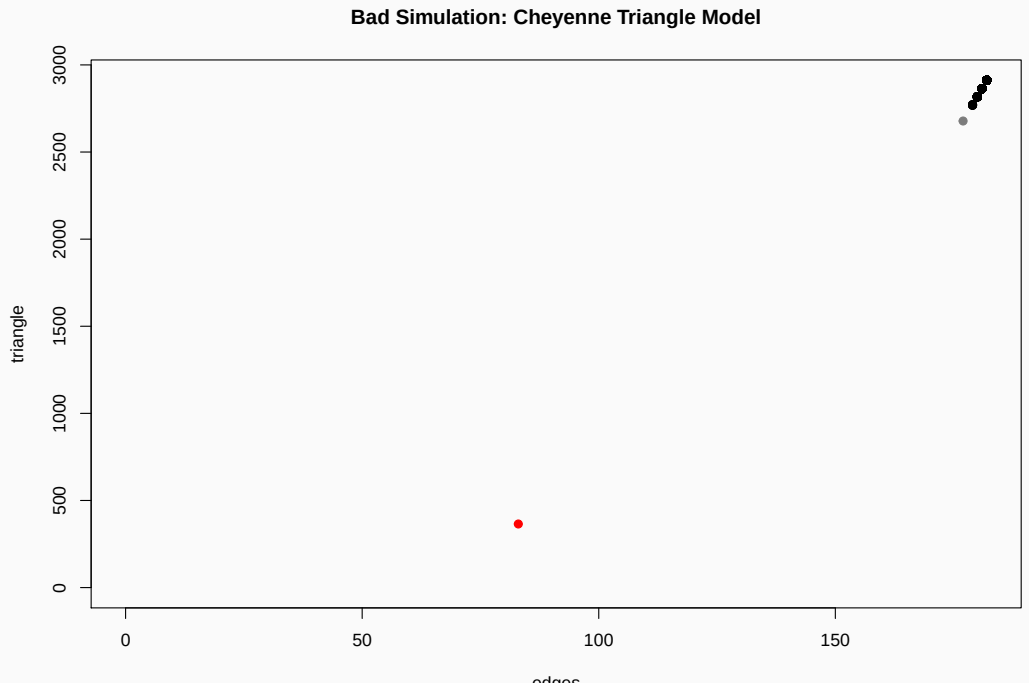



Figure 4: Regions of the mean value and natural parameter spaces.

Mark S. Handcock (2003). "Assessing Degeneracy in Statistical Models of Social Networks." Working Paper no. 39 Center for Statistics and the Social Sciences, University of Washington, Seattle.

# Quality of the Simulations

- **Simulations can fail in several ways**
  1. Insufficient burn-in: starting point still affects results
  2. Insufficient post-burn samples: sample hasn't converged
  3. May be degenerate: almost all graphs are same (usually complete/empty)
  4. Sample does not cover observed graph (problematic for inference)



# Assessing Simulation Quality

## Core MCMC diagnostic tools for ERGM

- In `ergm`, primary tool is `mcmcm.diagnostics()`
  - Requires `coda` library
  - Calculates various diagnostics on MCMC output
    - Correlations and lagged correlations of model statistics
    - Raftery-Lewis convergence diagnostics
  - Basic syntax

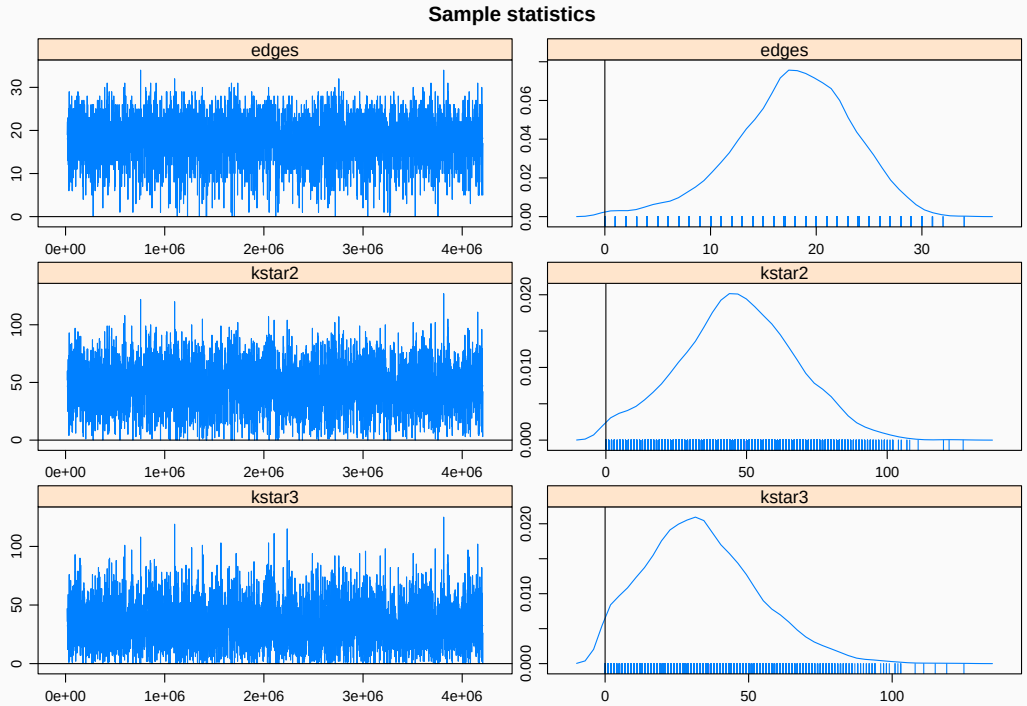
```
fit <- ergm(Y ~ term(1) + term(2))
mcmc.diagnostics(fit)
```

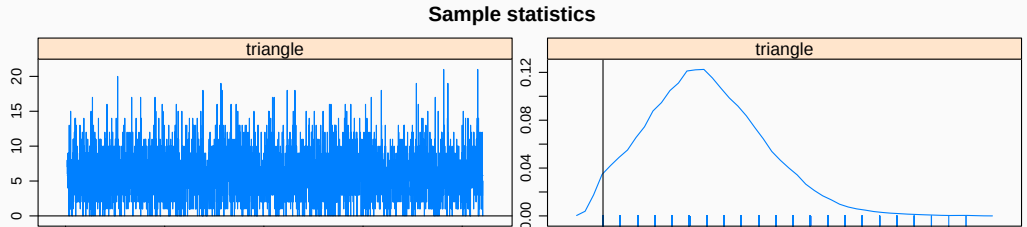
- Can also directly plot statistics vs observed
- `fit$sample` provides normalized simulated stat from an `ergm` object with 0 = observed value

# flobusines w/edges, 2-3 stars, triangles

```
flom.b1 <- ergm(flobusiness ~ edges + kstar(2) + kstar(3) + tria
```

```
mcmc.diagnostics(flom.b1, center = F)
```



# flobusines w/edges, 2-3 stars, triangles i

```
Sample statistics summary:

Iterations = 16384:4209664
Thinning interval = 1024
Number of chains = 1
Sample size per chain = 4096

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

           Mean     SD Naive SE Time-series SE
edges    17.900  5.495  0.08586        0.08586
kstar2   46.930 20.554  0.32115        0.32115
kstar3   33.534 19.569  0.30577        0.30577
triangle  5.962  3.352  0.05237        0.05237

2. Quantiles for each variable:

         2.5%   25% 50% 75% 97.5%
edges       5 15.00  18  22 28.00
kstar2      6 33.75  47  61 87.00
kstar3      1 20.00  32  46 76.62
triangle    0  4.00   6   8 13.00
```

# flobusines w/edges, 2-3 stars, triangles ii

```
Sample statistics cross-correlations:
             edges    kstar2    kstar3  triangle
edges    1.0000000 0.9556761 0.8614547 0.6897236
kstar2   0.9556761 1.0000000 0.9668333 0.8096087
kstar3   0.8614547 0.9668333 1.0000000 0.8390831
triangle 0.6897236 0.8096087 0.8390831 1.0000000

Sample statistics auto-correlation:
Chain 1
                 edges        kstar2       kstar3     triangle
Lag 0     1.0000000000  1.0000000000  1.000000000 1.0000000000
Lag 1024  0.0007063715  0.0018293750 -0.004015708 0.0111763814
Lag 2048  0.0089224102 -0.0022615558 -0.003435802 0.0007435549
Lag 3072  0.0006843674  0.0030467296  0.004152127 0.0178084816
Lag 4096  0.0148756633  0.0176699174  0.019107064 0.0367444101
Lag 5120 -0.0089908404  0.0003101401  0.005446429 0.0067100116

Sample statistics burn-in diagnostic (Geweke):
Chain 1

Fraction in 1st window = 0.1
Fraction in 2nd window = 0.5
```

# flobusines w/edges, 2-3 stars, triangles iii

```
   edges   kstar2   kstar3 triangle
-0.08956  0.17634 -0.03401  0.08136

Individual P-values (lower = worse):
    edges    kstar2    kstar3  triangle
0.9286396 0.8600303 0.9728723 0.9351533
Joint P-value (lower = worse):  0.2430351 .



MCMC diagnostics shown here are from the last round of simulation, prior to computation of final parameter
```

# Strategies for Issues with MCMC Sample

- For burn-in issues, increase `MCMC.burnin` parameter
- For post-burn convergence, incease `MCMC.samplesize`
  - `MCMC.samplesize` increasese final sample
  - `MCMC.interval` increases subsampling interval; useful if chain mixes slowly (i.e.,high autocorrelation and/or slow movement)
- If not of these work, it may be your model!
  - E.g., degeneracy due to runaway clique formation
  - Triangle, star terms especially bad (due to "nesting")
  - Try cuved variants (e.g., `gwesp`)

# Strategies for Issues with MCMC Sample

```
args(control.ergm)
```

```
function (drop = TRUE, init = NULL, init.method = NULL, main.method = c("MCMLE",
    "Robbins-Monro", "Stochastic-Approximation", "Stepping"),
    force.main = FALSE, main.hessian = TRUE, MPLE.max.dyad.types = 1e+06,
    MPLE.samplesize = 50000, MPLE.type = c("glm", "penalized"),
    MCMC.prop.weights = "default", MCMC.prop.args = list(), MCMC.interval = 1024,
    MCMC.burnin = MCMC.interval * 16, MCMC.samplesize = 1024,
    MCMC.effectiveSize = NULL, MCMC.effectiveSize.damp = 10,
    MCMC.effectiveSize.maxruns = 1000, MCMC.effectiveSize.base = 1/2,
    MCMC.effectiveSize.points = 5, MCMC.effectiveSize.order = 1,
    MCMC.return.stats = TRUE, MCMC.runtime.traceplot = FALSE,
    MCMC.init.maxedges = 20000, MCMC.max.maxedges = Inf, MCMC.addto.se = TRUE,
    MCMC.compress = FALSE, MCMC.packagenames = c(), SAN.maxit = 10,
    SAN.burnin.times = 10, SAN.control = control.san(coef = init,
        SAN.prop.weights = MCMC.prop.weights, SAN.prop.args = MCMC.prop.args,
        SAN.init.maxedges = MCMC.init.maxedges, SAN.burnin = MCMC.burnin *
            SAN.burnin.times, SAN.interval = MCMC.interval, SAN.packagenames = MCMC.packagenames,
        MPLE.max.dyad.types = MPLE.max.dyad.types, parallel = parallel,
        parallel.type = parallel.type, parallel.version.check = parallel.version.check),
    MCMLE.termination = c("Hummel", "Hotelling", "precision",
        "none"), MCMLE.maxit = 20, MCMLE.conv.min.pval = 0.5,
    MCMLE.NR.maxit = 100, MCMLE.NR.reltol = sqrt(.Machine$double.eps),
    obs.MCMC.samplesize = MCMC.samplesize, obs.MCMC.interval = MCMC.interval,
    obs.MCMC.burnin = MCMC.burnin, obs.MCMC.burnin.min = obs.MCMC.burnin/10,
    obs.MCMC.prop.weights = MCMC.prop.weights, obs.MCMC.prop.args = MCMC.prop.args,
    MCMLE.check.degeneracy = FALSE, MCMLE.MCMC.precision = 0.005,
    MCMLE.MCMC.max.ESS.frac = 0.1, MCMLE.metric = c("lognormal",
        "logtaylor", "Median.Likelihood", "EF.Likelihood", "naive"),
```

# Model Assessment

- **Model adequacy**:
  - Key idea: Model should reproduce relevant properties of the observed data
- **How does one assess model adequacy? Simulation!**
  - Simulate draws from fitted model
  - Compare observed graph to simulated graphs on measures of interest
  - Verify that observed properties are well-covered by simulated ones (e.g., not in 5% tails)
- **What properties should be considered?**
  - This is application-sepcific – no single uniform answer
  - Start with "in-model" statistics; ERG must get means right, but still verify non-pathological distributions
  - "Out-of-model" statistics can be common low-level properties (e.g., degree, triad census) or theoretically motivated quantities

# Checking Adequacy with `gof`

Basic syntax:

```
got(fit, GOF = ~term1 + term2)
```

- Has `print`, `plot`, and `summary` methods
- Note: still uses MCMC, so check convergence

# Example: flobusiness with edges i

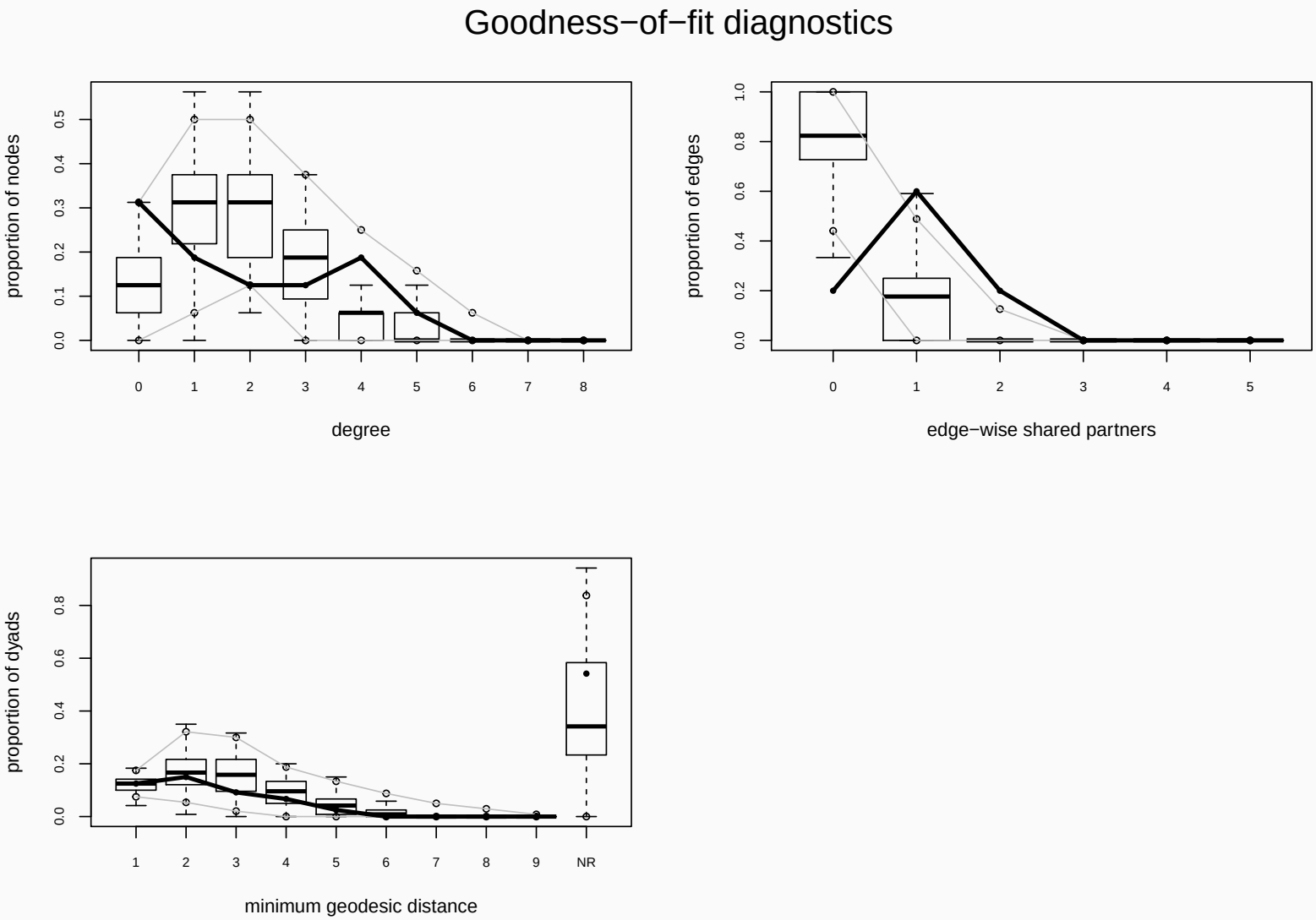
```
flom.b1 <- ergm(flobusiness ~ edges)
flo_gof <- gof(flom.b1)
```

```
Goodness-of-fit for degree

  obs min mean max MC p-value
0   5   0 2.08   8       0.18
1   3   0 4.74   9       0.50
2   2   1 4.76   9       0.18
3   2   0 2.74   6       0.94
4   3   0 1.14   5       0.36
5   1   0 0.46   3       0.66
6   0   0 0.06   1       1.00
7   0   0 0.02   1       1.00

Goodness-of-fit for edgewise shared partner

     obs min  mean max MC p-value
esp0   3   3 12.05  19       0.02
esp1   9   0  2.63  13       0.10
esp2   3   0  0.23   4       0.04
```

# Example: flobusiness with edges ii

```
esp3   0   0  0.01   1       1.00

Goodness-of-fit for minimum geodesic distance

    obs min  mean max MC p-value
1    15   5 14.92  22       1.00
2    18   1 20.66  42       0.88
3    11   0 18.72  38       0.50
4     8   0 11.31  24       0.72
5     3   0  5.49  18       0.84
6     0   0  2.15  12       0.96
7     0   0  0.72   9       1.00
8     0   0  0.31   6       1.00
9     0   0  0.08   3       1.00
10    0   0  0.02   2       1.00
Inf  65   0 45.62 113       0.58
```

# Example: flobusiness with edges



# Common Strategies

- Option 1: Add terms
  - Which features are poorly captured? Is there a term which would add in such effects (ideally minimally)?
- Option 2: Switch terms
  - Can you replace an existing term with a similar one more likely to succeed? (E.g., sociality or degree terms versus k-stars)
- Option 3: Do nothing
  - Is the type of inadequacy a problem for your specific question? Can it be tolerated in this case? How good is the overall fit?

# References and Places for More Information

# References and Places for More Information i

- Statnet wiki: `https://statnet.org/trac`
- JSS Special Issue:
  `https://www.jstatsoft.org/issue/view/v024`
  **Workshop slides are based on the following courses**
- Carter Butt's 2009 Analysis of Social Network Data at UCI
- Mark Handcock's 2011 Statistical Analysis of Networks at UCLA
- Peter Hoff's Statistical Analysis of Social Networks at UW
- Martina Morris' 2016 Statistical Analysis of Social Networks at UW
- Zack Almquist's 2015 Social Network Analysis: Theory and Methods at UMN