Statistical Analysis of Network Data

Héctor Corrada Bravo

University of Maryland, College Park, USA

CMSC828O 2018-10-25

Statistical Analysis

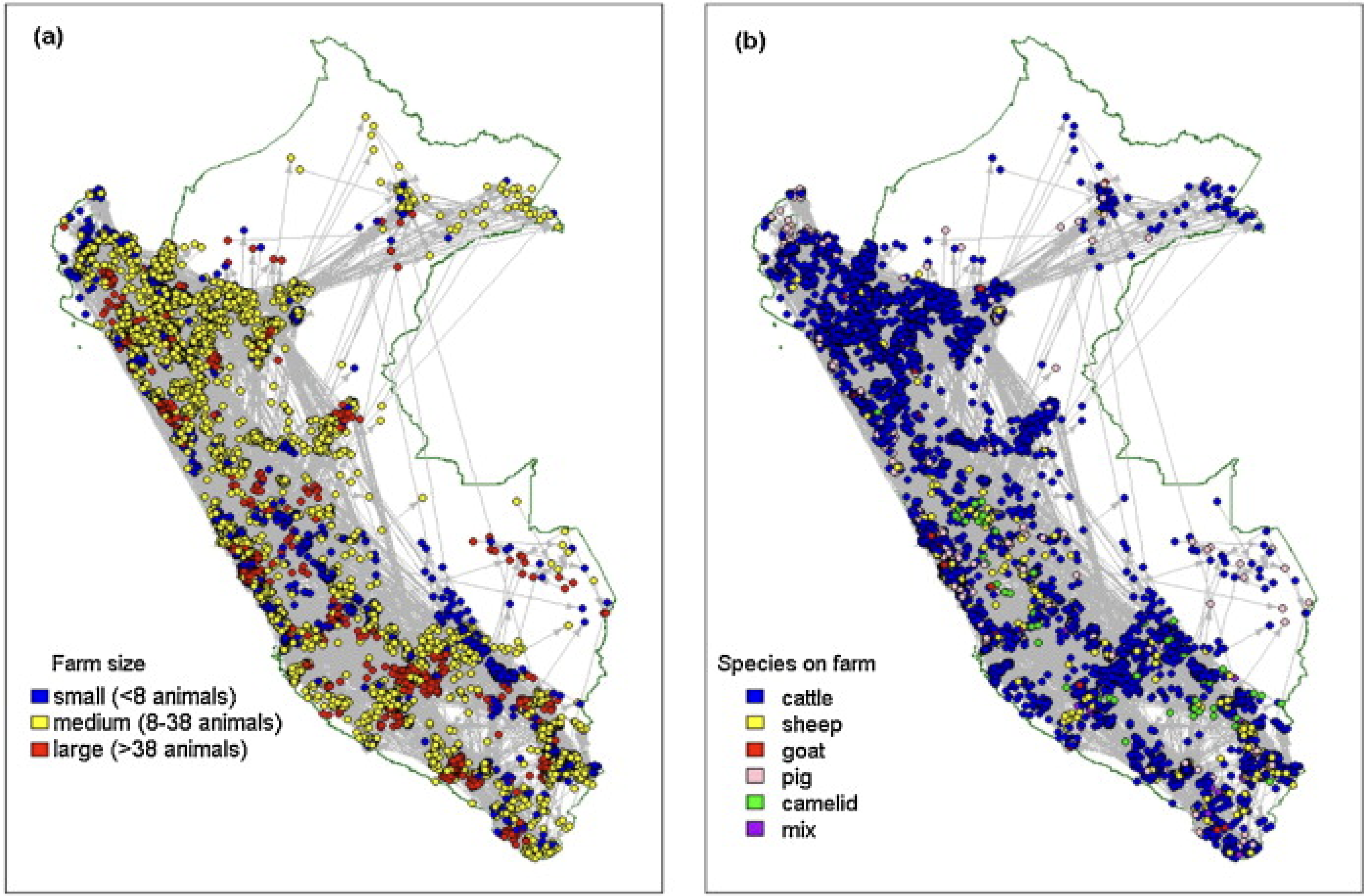
In this next unit we will look at methods that approach network analysis from a statistical inference perspective.

Statistical Analysis

In particular we will look at three statistical inference and learning tasks over networks

- Analyzing edges between vertices as a stochastic process over which we can make statistical inferences
- Constructing networks from observational data
- Analyzing a process (e.g., diffusion) over a network in a statistical manner

Spatial effects in ecological networks



https://doi.org/10.1016/j.prevetmed.2014.01.013

Exponential Random Graph Models

In ER random graph model edge probabilities were independent of vertex characterisitics.

Now assume vertices have measured attributes.

Question: what is the effect of these attributes in network formation, specifically in edge occurence.

Exponential Random Graph Models

Denote Y as adjacency matrix of graph G over n elements

Denote X as matrix of vertex attributes.

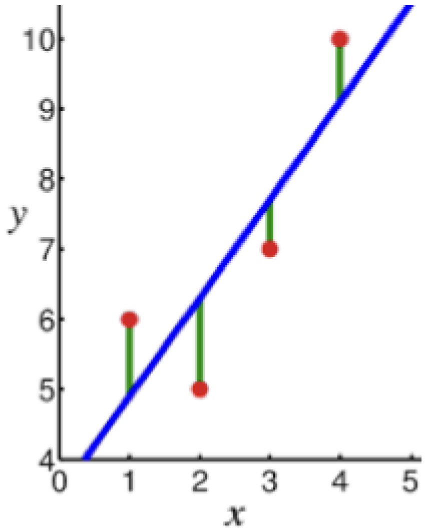
We want to determine $P(Y_{ij} = 1 | Y_{-(ij)} = y_{-(ij)}, x_i, x_j, L(G))$

where $L(G)$ is a measure of structure of graph G and $y_{-(ij)}$ is the *configuration* of edges other than edge $i \sim j$

Exponential Random Graph Models

We can motivate ERGM model from regression (where outcome Y is continuous)

$$E[Y_i|x_i] = \sum_{j=1}^{p} \beta_j x_{ij} = \beta' x_i$$

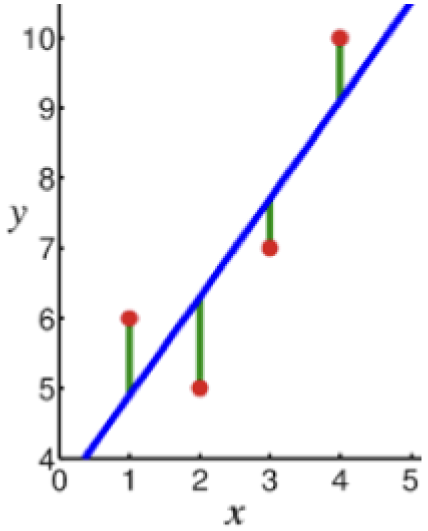


Exponential Random Graph Models

We turn into a probabilistic model as

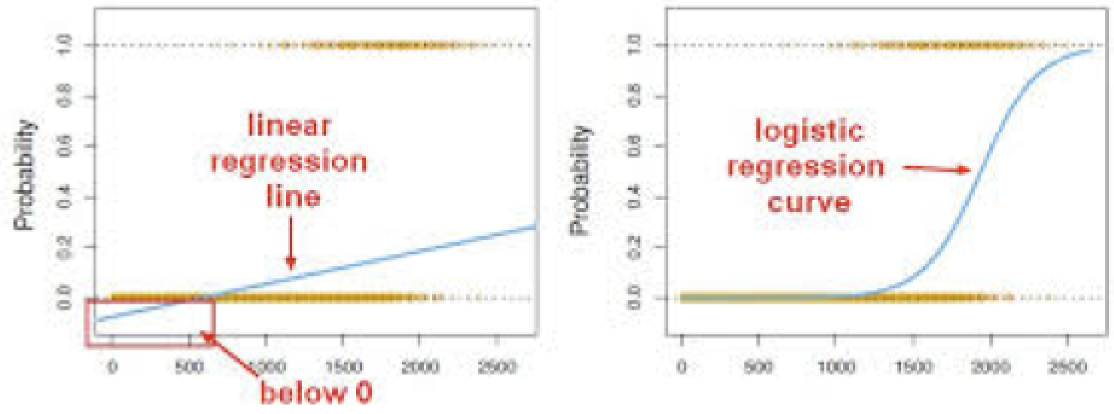
$$Y = \beta' x_i + \epsilon$$

$$\epsilon \sim N(0, \sigma^2)$$



Exponential Random Graph Models

For *binary* outcome Y we use *logistic regression*



$$\log \frac{P(Y_i = 1|x_i)}{1 - P(Y_i = 1|x_i)} = \beta' x_i$$

Which corresponds to a Bernoulli model of $P(Y_i = 1|x_i)$.

Exponential Random Graph Models

The outcome of interest in the ERGM model is the *presence* of edge $y_{ij} = 1$

.

Use a Bernoulli model with y_{ij} as the outcome.

With vertex attributes and graph structural measure as predictors.

Exponential Random Graph Models

Model 1: the ER model

$$\log \frac{P(Y_{ij} = 1 | Y_{-(ij)} = y_{-(ij)})}{P(Y_{ij} = 0 | Y_{-(ij)} = y_{-(ij)})} = \theta$$

Thinking of logistic regression: model is a *constant*, independent of rest of graph structure, independent of vertex attributes

Exponential Random Graph Models

To fit models we need a *likelihood*, i.e., probability of observed graph, given parameters (in this case θ)

Exponential Random Graph Models

To fit models we need a *likelihood*, i.e., probability of observed graph, given parameters (in this case θ)

Write $P(Y_{ij} = 1|\ldots)$ as p, then likelihood is given by

$$\mathcal{L}(\theta; y) = \prod_{ij} p^{y_{ij}} (1-p)^{(1-y_{ij})}$$

Exponential Random Graph Models

(Exercise)

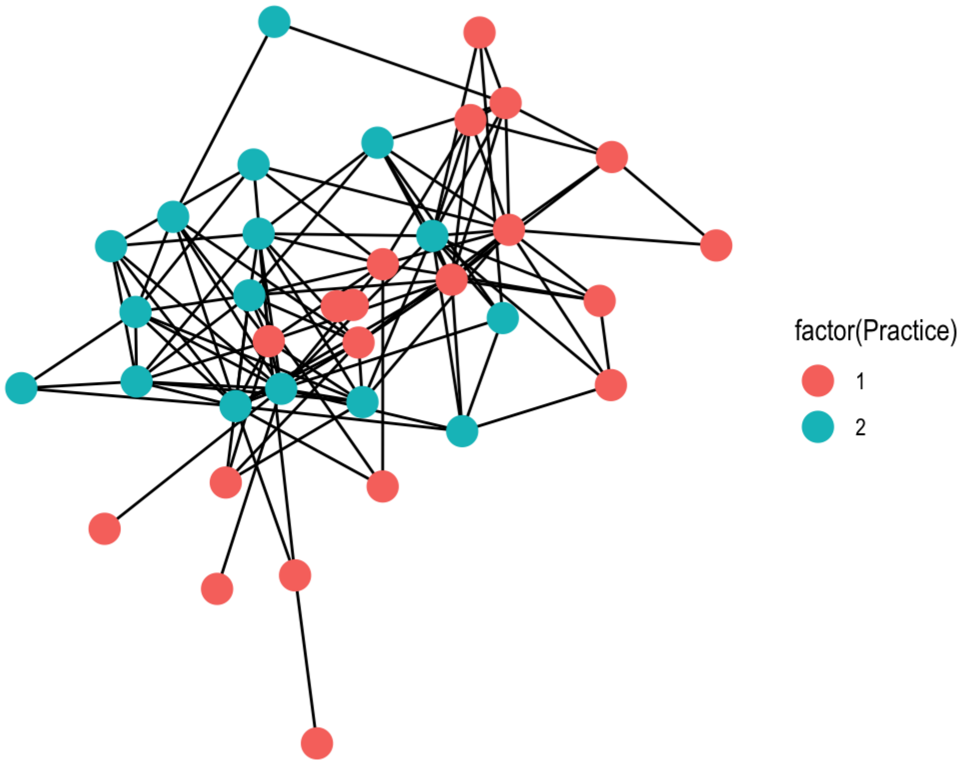
$$\mathcal{L}(\theta; y) = \frac{1}{\kappa}\exp\{\theta L(y)\}$$

where $L(y)$ is the number of edges in the graph.

This is the formulation given in reading!

Exponential Random Graph Models

```
## Observations: 36
## Variables: 9
## $ name       <chr> "V1", "V2",
## $ Seniority  <int> 1, 2, 3, 4,
## $ Status     <int> 1, 1, 1, 1,
## $ Gender     <int> 1, 1, 1, 1,
## $ Office     <int> 1, 1, 2, 1,
## $ Years      <int> 31, 32, 13,
## $ Age        <int> 64, 62, 67,
## $ Practice   <int> 1, 2, 1, 2,
## $ School     <int> 1, 1, 1, 3, 2, 1, 3, 3, 1, 3, 1, 2, 2, 1, 3, 1, 1, 2...
```



Exponential Random Graph Models

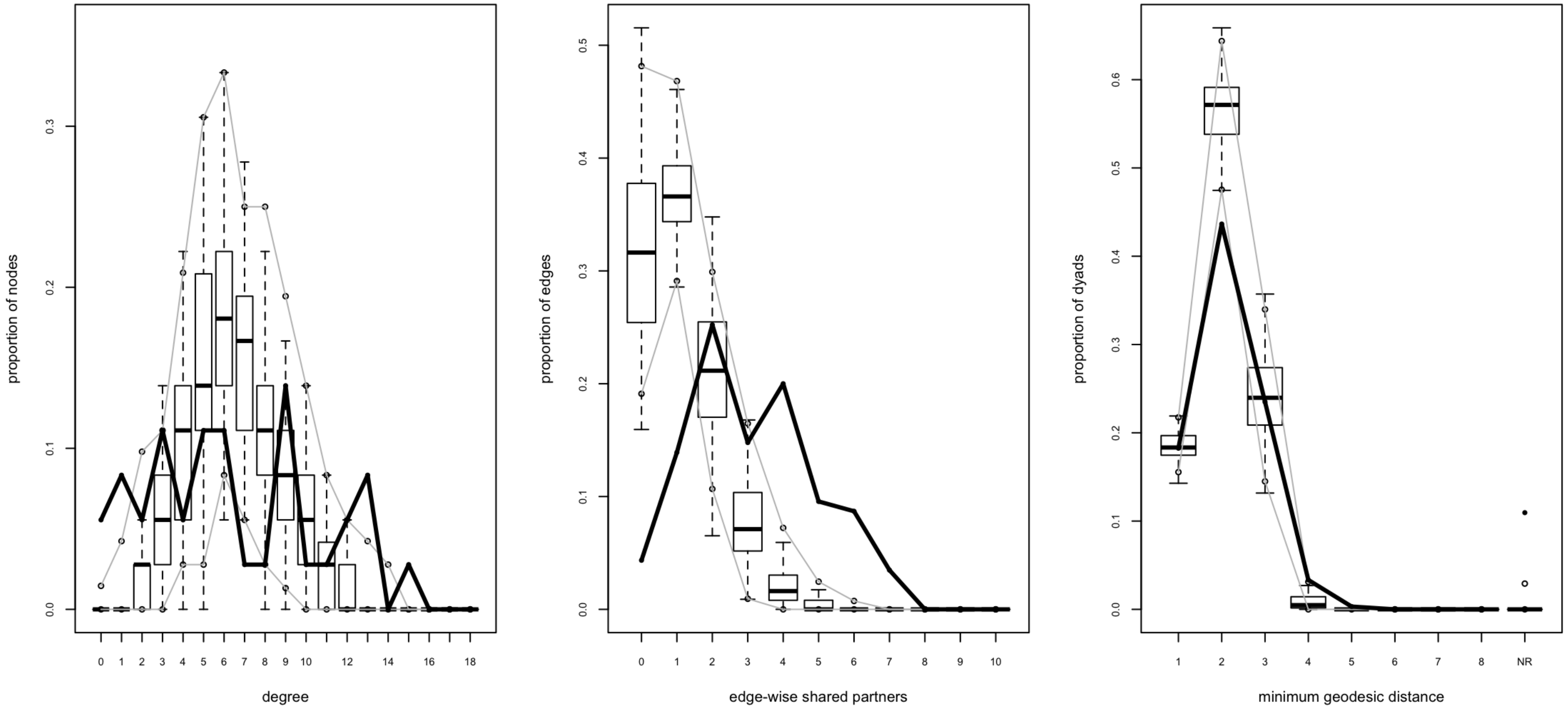
```r
library(ergm)
A <- get.adjacency(lazega)
lazega.s <- network::as.network(as.matrix(A), directed=FALSE)
ergm.bern.fit <- ergm(lazega.s ~ edges)
ergm.bern.fit
```

```
##
## MLE Coefficients:
##  edges
## -1.499
```

So $\theta = -1.5$

and thus $p = 0.183$

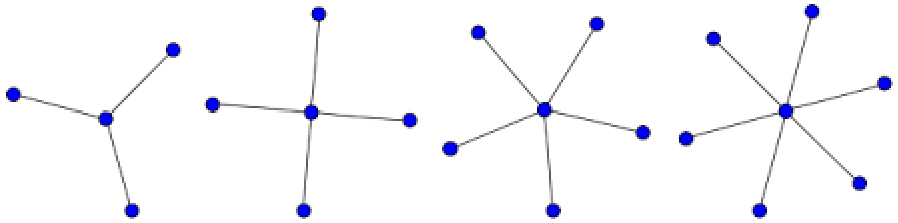
Exponential Random Graph Models



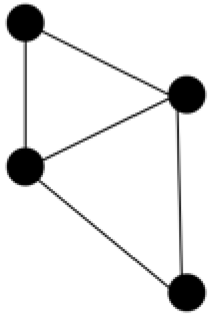
Goodness-of-fit diagnostics

Exponential Random Graph Models

The ER model is not appropriate, let's extend with more graph statistics.



$S_k(y)$: number of k-stars



$T_k(y)$: number of k-triangles

Exponential Random Graph Models

In practice, instead of adding terms for structural statistics at all values of k, they are combined into a single term

For example *alternating k-star counts*

$$\mathrm{AKS}_\lambda(y) = \sum_{k=2}^{N_v-1} (-1)^k \frac{S_k(y)}{\lambda^{k-2}}$$

λ is a parameter that controls decay of influence of larger k terms. Treat as a hyper-parameter of model

Exponential Random Graph Models

Another example is *geometrically weighted degree count*

$$\mathrm{GWD}_\gamma(y) = \sum_{d=0}^{N_v-1} e^{-\gamma d} N_d(y)$$

There is a good amount of literature on definitions and properties of suitable terms to summarize graph structure in these models

Exponential Random Graph Models

In addition we want to adjust edge probabilities based on vertex attributes

For edge $i \sim j$, i may have attribute that increases degree (e.g., seniority)

Or, i and j have attributes that *together* increase edge probability (e.g., spatial distance in an ecological network)

Exponential Random Graph Models

We can add attribute terms to the ERGM model accordingly. E.g.,

- Main effects: $h(x_i, x_j) = x_i + x_j$
- Categorical interaction (match): $h(x_i, x_j) = I(x_i == x_j)$
- Numeric interaction: $h(x_i, x_j) = (x_i - x_j)^2$

Exponential Random Graph Models

A full ERGM model for this data:

```
lazega.ergm <- formula(lazega.s ~ edges + gwesp(log(3), fixed=TRUE) +
  nodemain("Seniority") +
  nodemain("Practice") +
  match("Practice") +
  match("Gender") +
  match("Office"))
```

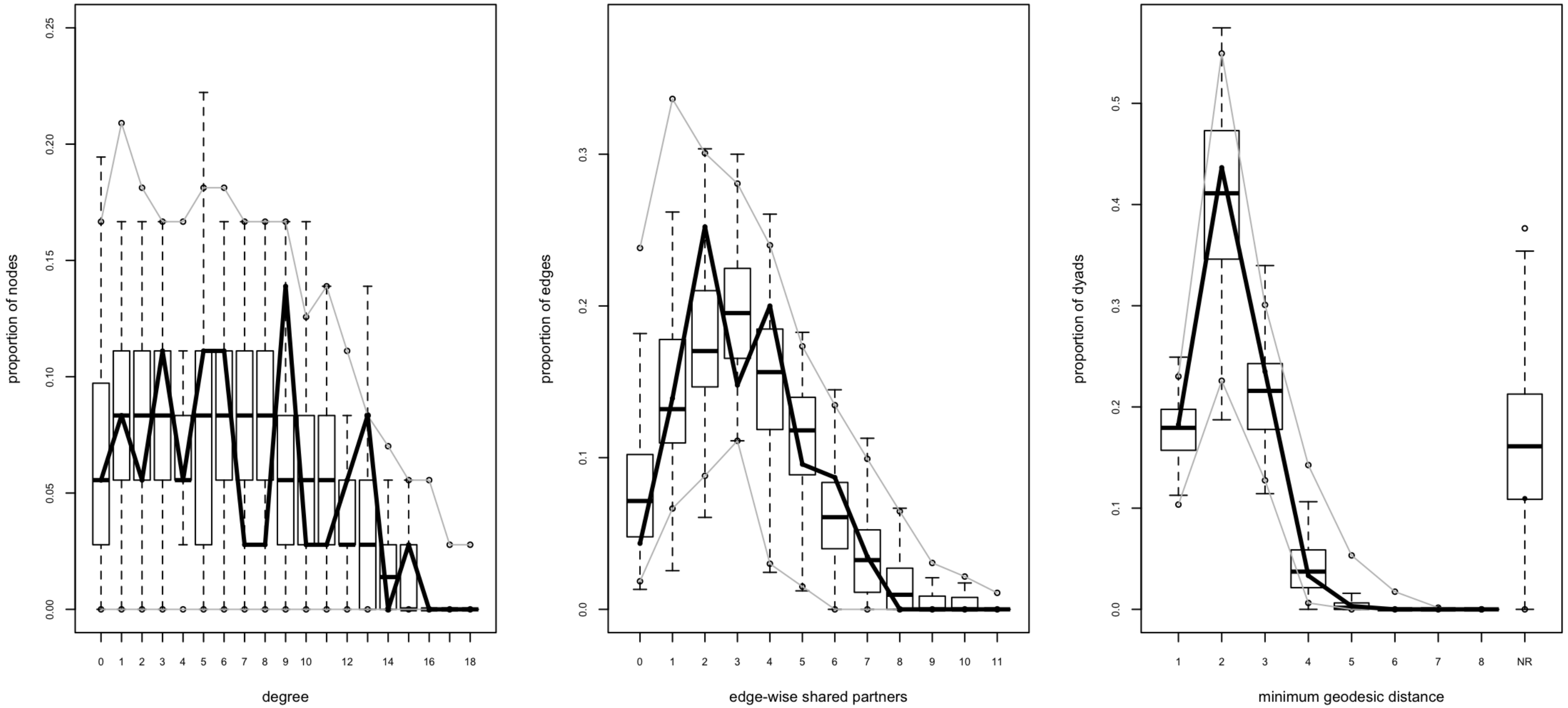
Exponential Random Graph Models

```
## # A tibble: 1 x 5
##   independence iterations logLik   AIC   BIC
##   <lgl>             <int>  <dbl> <dbl> <dbl>
## 1 FALSE                 2  -230.  474.  505.
```

Exponential Random Graph Models

term	estimate	std.error	mcmc.error	p.value
edges	-7.0126525	0.7030685	0	0.0000000
gwesp.fixed.1.09861228866811	0.5906030	0.0866729	0	0.0000000
nodecov.Seniority	0.0247161	0.0063578	0	0.0001013
nodecov.Practice	0.3960366	0.1042741	0	0.0001458
nodematch.Practice	0.7694967	0.1961173	0	0.0000872
nodematch.Gender	0.7374215	0.2463944	0	0.0027639
nodematch.Office	1.1654418	0.1905383	0	0.0000000

Exponential Random Graph Models



Goodness-of-fit diagnostics

Exponential Random Graph Models

A few more points:

The general formulation for ERGM is

$$P_\theta(Y = y) = \frac{1}{\kappa}\exp\left\{\sum_H g_H(y)\right\}$$

where H represents possible *configurations* of possible edges among a subset of vertices in graph

$g_H(y) = \prod_{y_{ij} \in H} y_{ij} = 1$ if configuration H occurs in graph

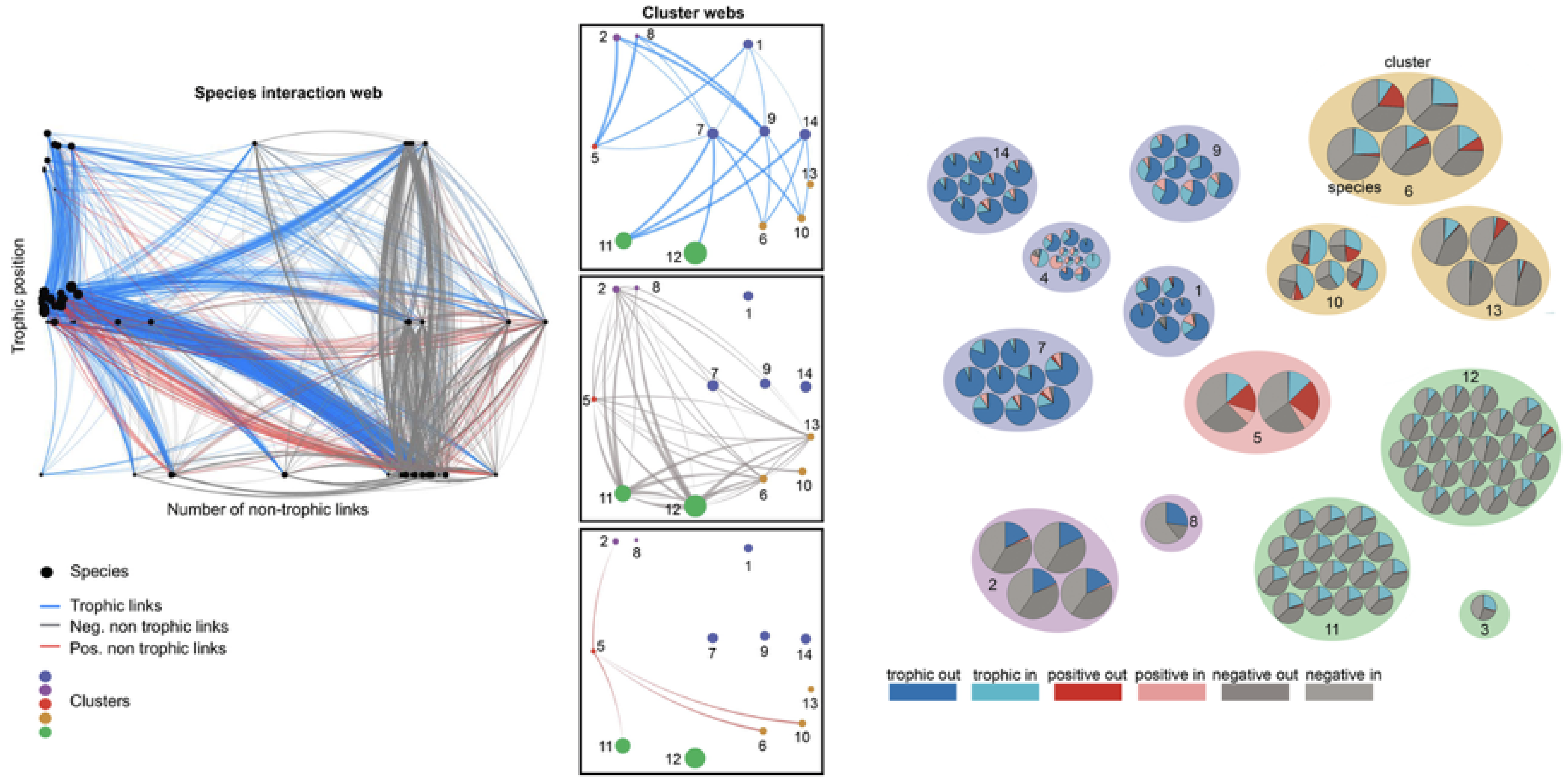
Exponential Random Graph Models

This brings about some complications since it's infeasible to define function over all possible configurations

Instead, collapse configurations into groups based on certain properties, and count the number of times these properties are satisfied in graph

Even then, computing normalization term κ is also infeasible, therefore use sampling methods (MCMC) for estimation

Stochastic Block Models



https://doi.org/10.1371/journal.pbio.1002527

Stochastic Block Models

Method to cluster vertices in graph

Assume that each vertex belongs to one of Q classes

Then probability of edge $i \sim j$ depends on class/cluster membership of vertices i and j

Stochastic Block Models

Clustered ERGM model

If we knew vertex classes, e.g., i belongs to class q and j belongs to class r

$$\log \frac{P(Y_{ij} = 1 | Y_{-(ij)} = y_{-(ij)})}{P(Y_{ij} = 0 | Y_{-(ij)} = y_{-(ij)})} = \theta_{qr}$$

Stochastic Block Models

Clustered ERGM model

Likelihood is then

$$\mathcal{L}(\theta; y) = \frac{1}{\kappa}\exp\{\sum_{qr} \theta_{qr} L_{qr}(y)\}$$

with $L_{qr}(y)$ the number of edges $i \sim j$ where i in class q and j in class r

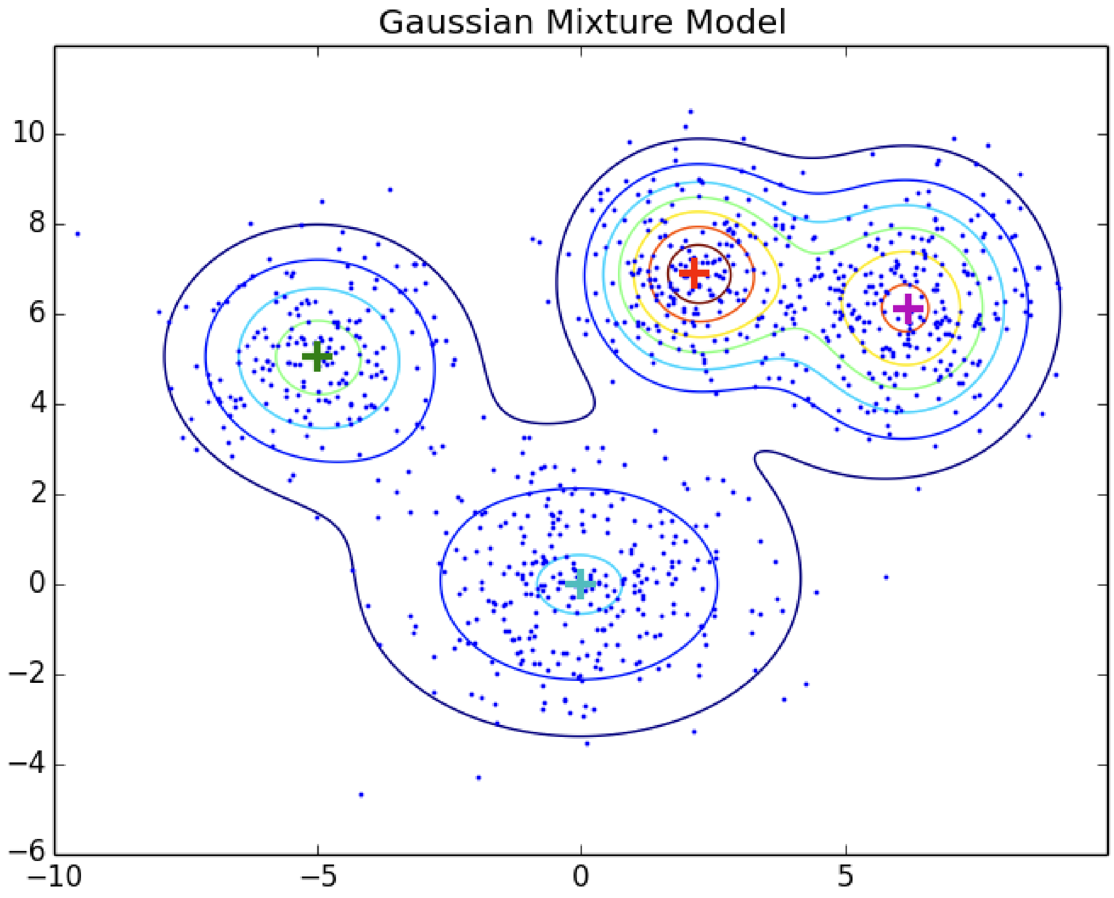
(a model like `g~match(class)` in ERGM)

Stochastic Block Models

However, suppose we don't know vertex class assignments...

SBM is a probabilistic method where we maximize likelihood of this model, assuming class assignments are unobserved

Stochastic Block Models



Stochastic Block Models

- Y_{ij} edge $i \sim j$ (binary)
- z_{iq} indicator for vertex i class q (binary)
- α_q prior for class q $p(z_{iq} = 1) = \alpha_q$
- π_{qr}: probability of edge $i \sim j$ where i in class q and j in class r

Stochastic Block Models

With this we can write again a likelihood

$$\mathcal{L}(\theta; y, z) = \sum_i \sum_q z_{iq} \log \alpha_q + \frac{1}{2} \sum_{i \neq j} \sum_{q \neq r} z_{iq} z_{jr} b(y_{ij}; \pi_{qr})$$

with $b(y, \pi) = \pi^y (1 - \pi)^{(1-y)}$

Stochastic Block Models

Like similar models (e.g., Gaussian mixture model, Latent Dirichlet Allocation) can't optimize this directly

Instead EM algorithm used:

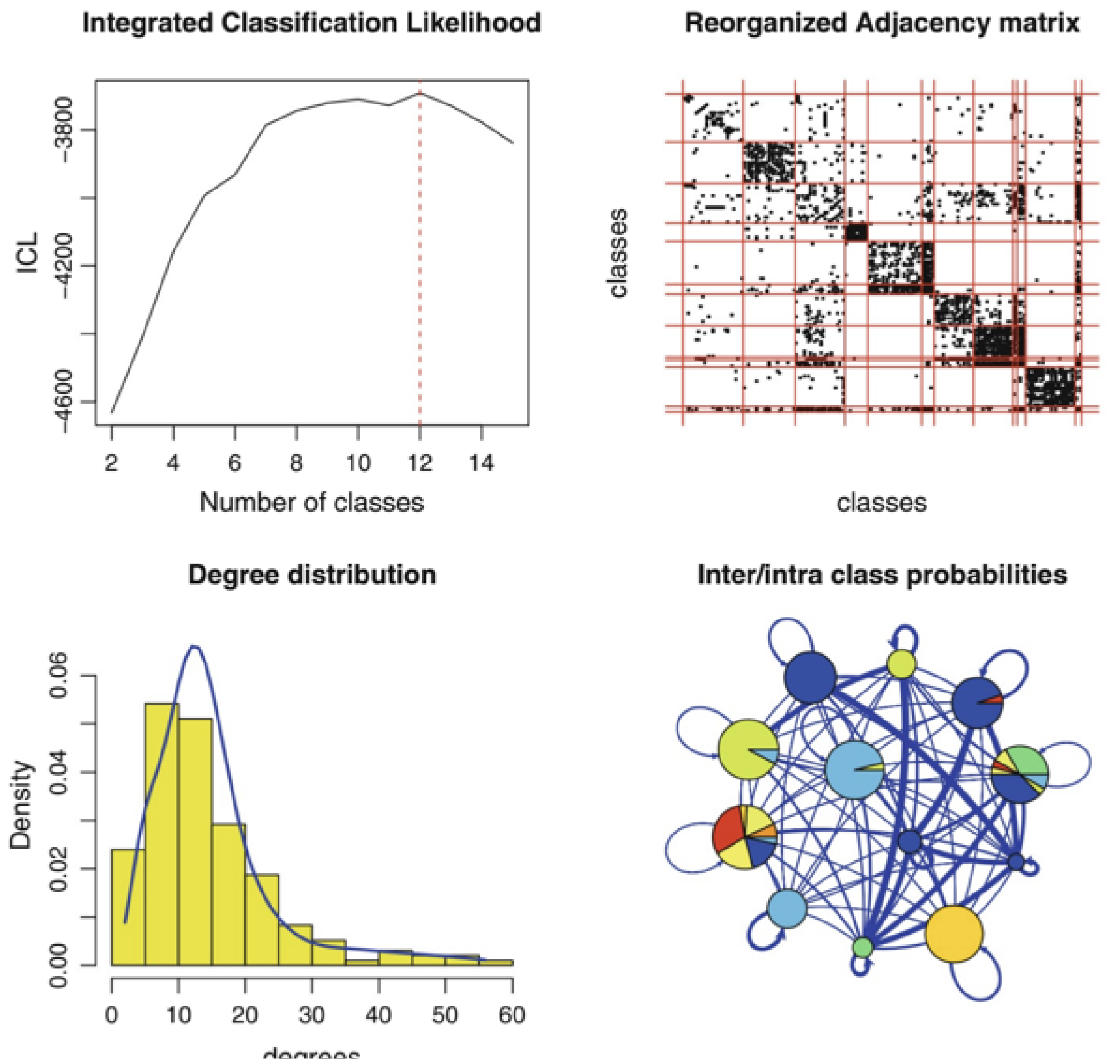
- Initialize parameters θ
- Repeat until "convergence":
  - Compute $\gamma_{iq} = E\{z_{iq}|y;\theta\} = p(z_{iq}|y;\theta)$
  - Maximize likelihood w.r.t. θ plugging in γ_{iq} for z_{iq}.

Stochastic Block Models

Like similar models need to determine number of classes (clusters) and select using some model selection criterion

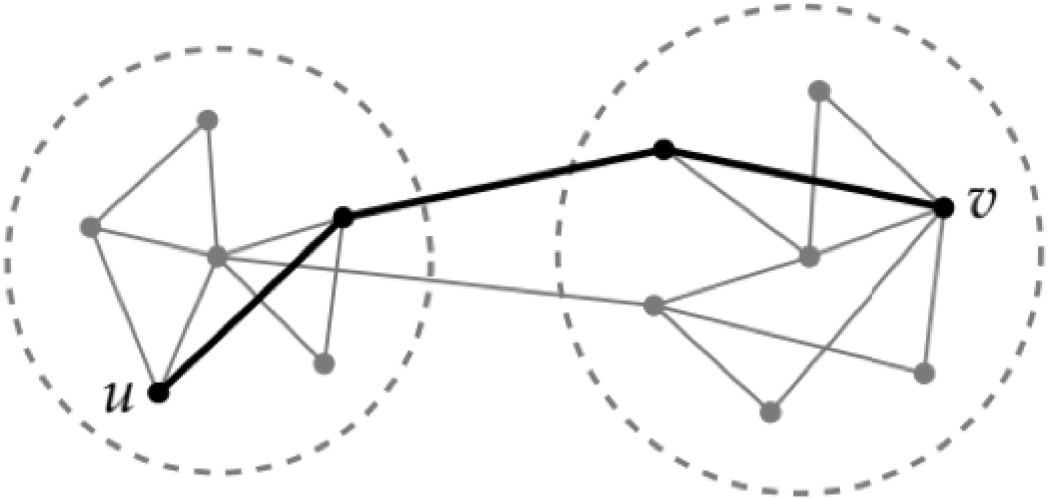
- AIC
- BIC
- Integrated Classification Likelihood

Stochastic Block Models



Communities

If we think of class as *community* we can see relationship with non-probabilistically community finding methods (e.g., Newman-Girvan)



Summary

Slightly different way of thinking probabilistically about networks

Define probabilistic model over network configurations

Parameterize model using network structural properties and vertex properties

Perform inference/analysis on resulting parameters

Can also extend classical clustering methodology to this setting