Statistical Analysis of Network Data

Héctor Corrada Bravo

University of Maryland, College Park, USA

CMSC828O 2019-10-21

Statistical Analysis

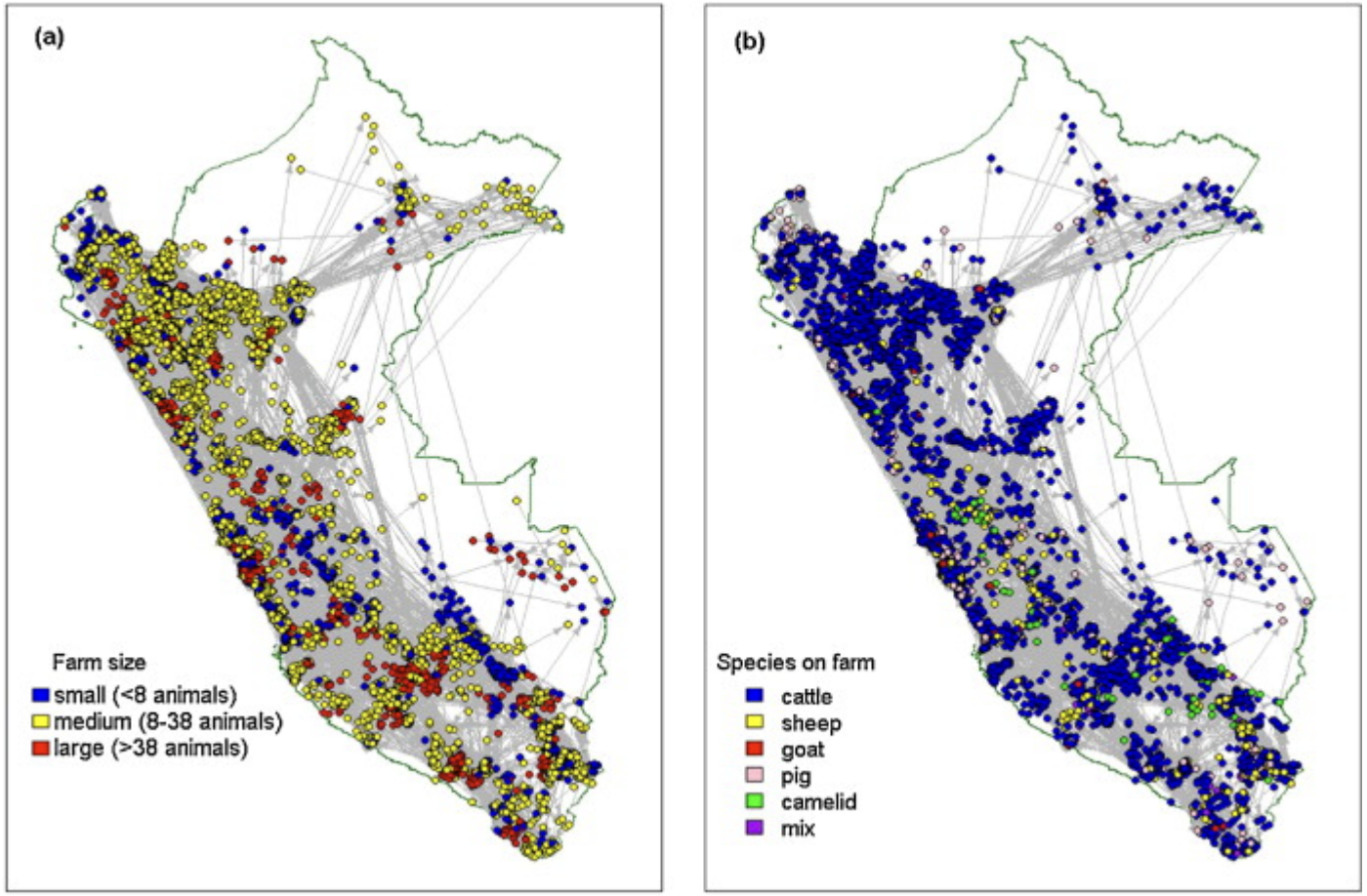
In this next unit we will look at methods that approach network analysis from a statistical inference perspective.

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.

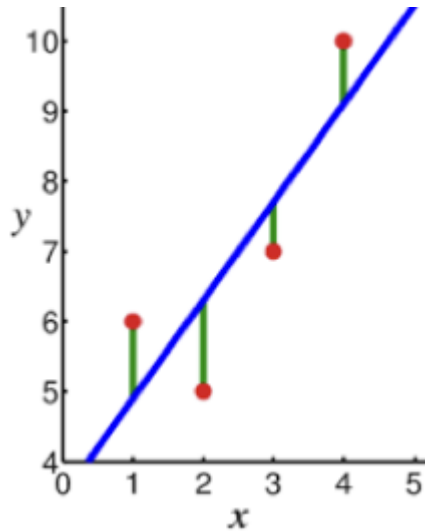
We want to determine $P(Y_{ij} = 1 | Y_{-(ij)} = y_{-(ij)}, x_i, x_j, L(G))$

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)

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

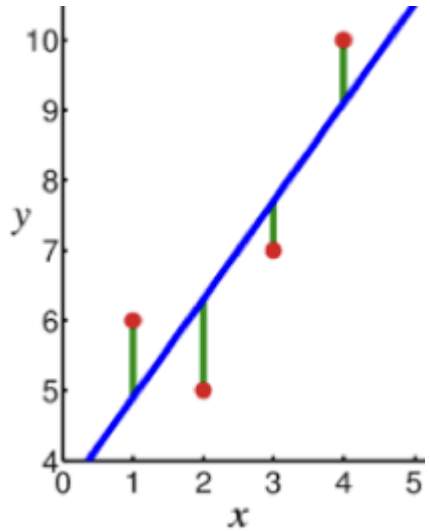


Exponential Random Graph Models

We turn into a probabilistic model as

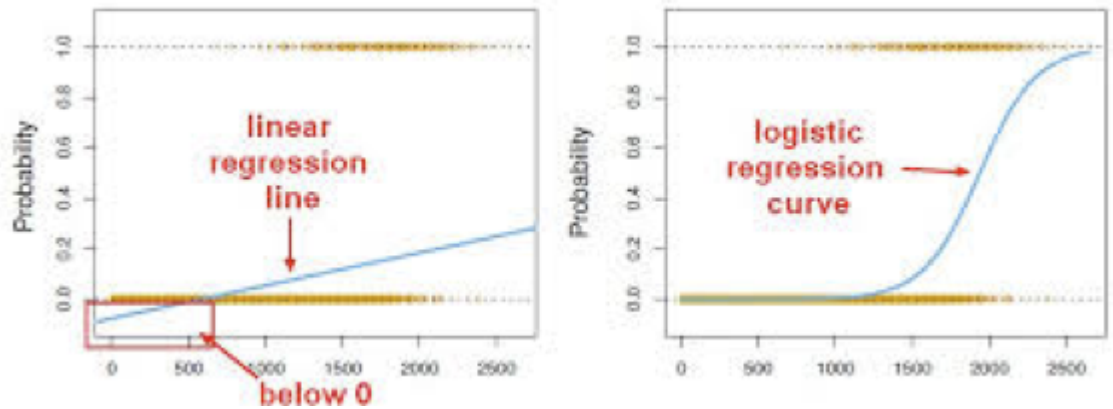
$$Y = \beta' x_i + \epsilon$$

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



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 $P(Y_{ij} = 1|\dots)$ 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)

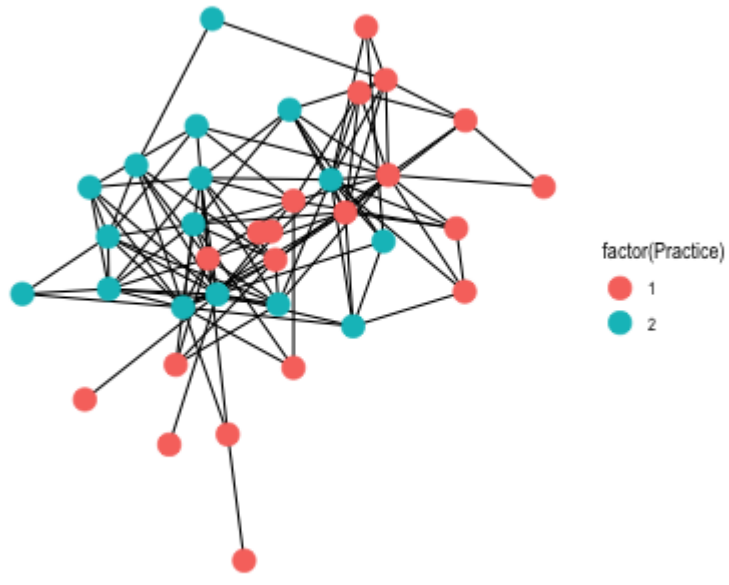
$$\mathcal{L}(\theta; y) = \frac{1}{\kappa} \exp\{\theta L(y)\}$$

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", "V3",
## $ Seniority <int> 1, 2, 3, 4, 5, 6,
## $ Status    <int> 1, 1, 1, 1, 1, 1,
## $ Gender    <int> 1, 1, 1, 1, 1, 1,
## $ Office    <int> 1, 1, 2, 1, 2, 2,
## $ Years     <int> 31, 32, 13, 31, 31, 29, 29, 28, 25, 25, 23, 24, 22, 1,…
## $ Age       <int> 64, 62, 67, 59, 59, 55, 63, 53, 53, 53, 50, 52, 57, 56…
## $ Practice  <int> 1, 2, 1, 2, 1, 1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 2, 2, 1, …
## $ School    <int> 1, 1, 1, 3, 2, 1, 3, 3, 1, 3, 1, 2, 2, 1, 3, 1, 1, 2, …
```



Exponential Random Graph Models

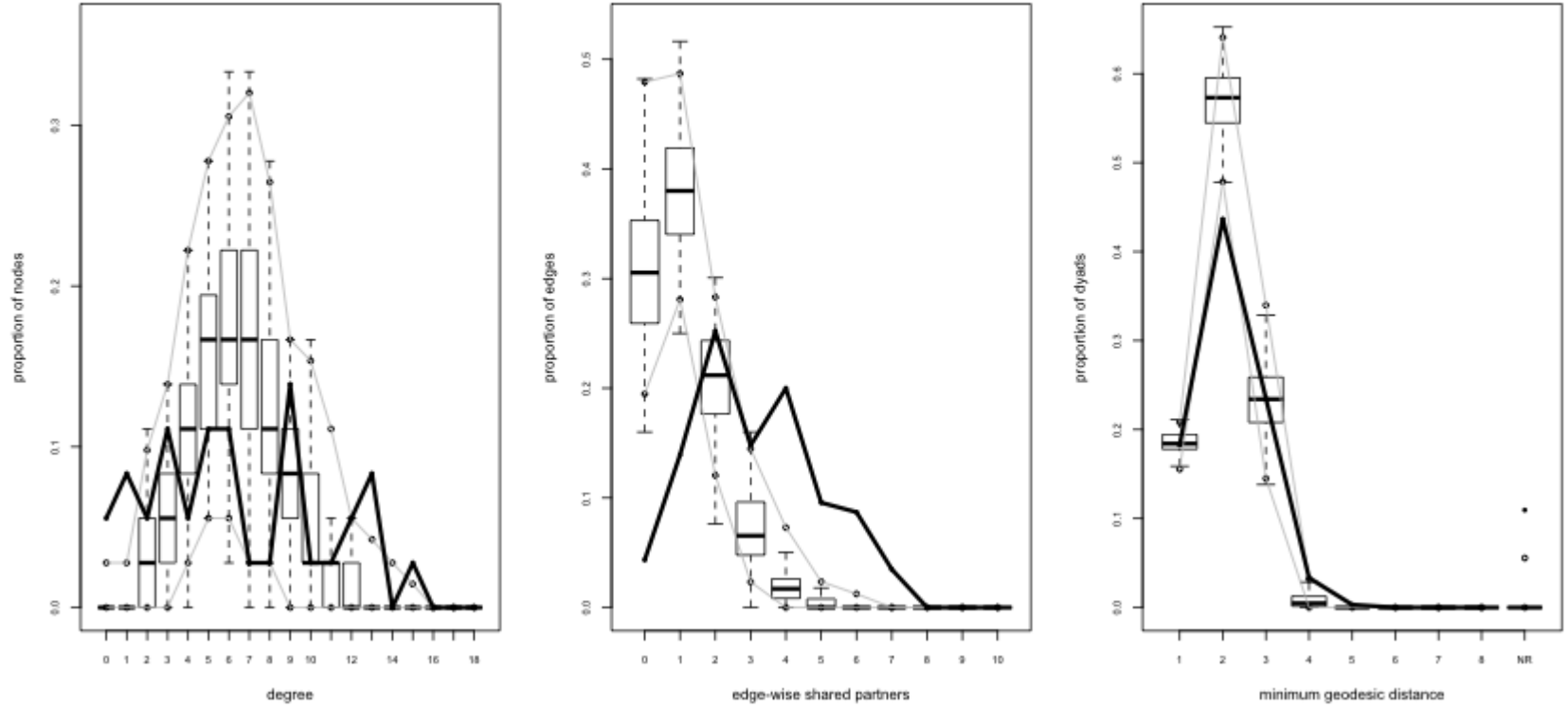
So $\theta = -1.5$

and thus $p = 0.183$

```
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
```

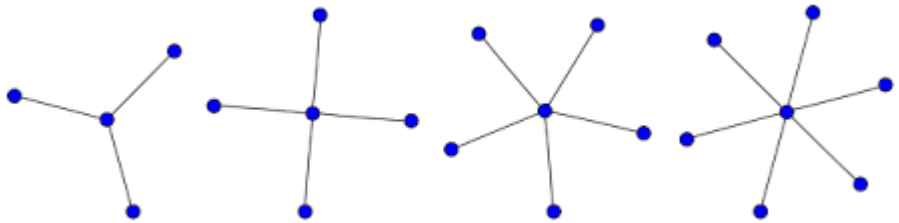
Exponential Random Graph Models

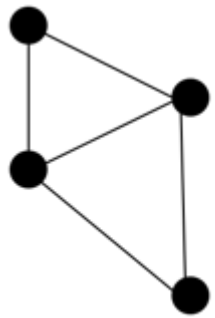


Goodness-of-fit diagnostics

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"))
```

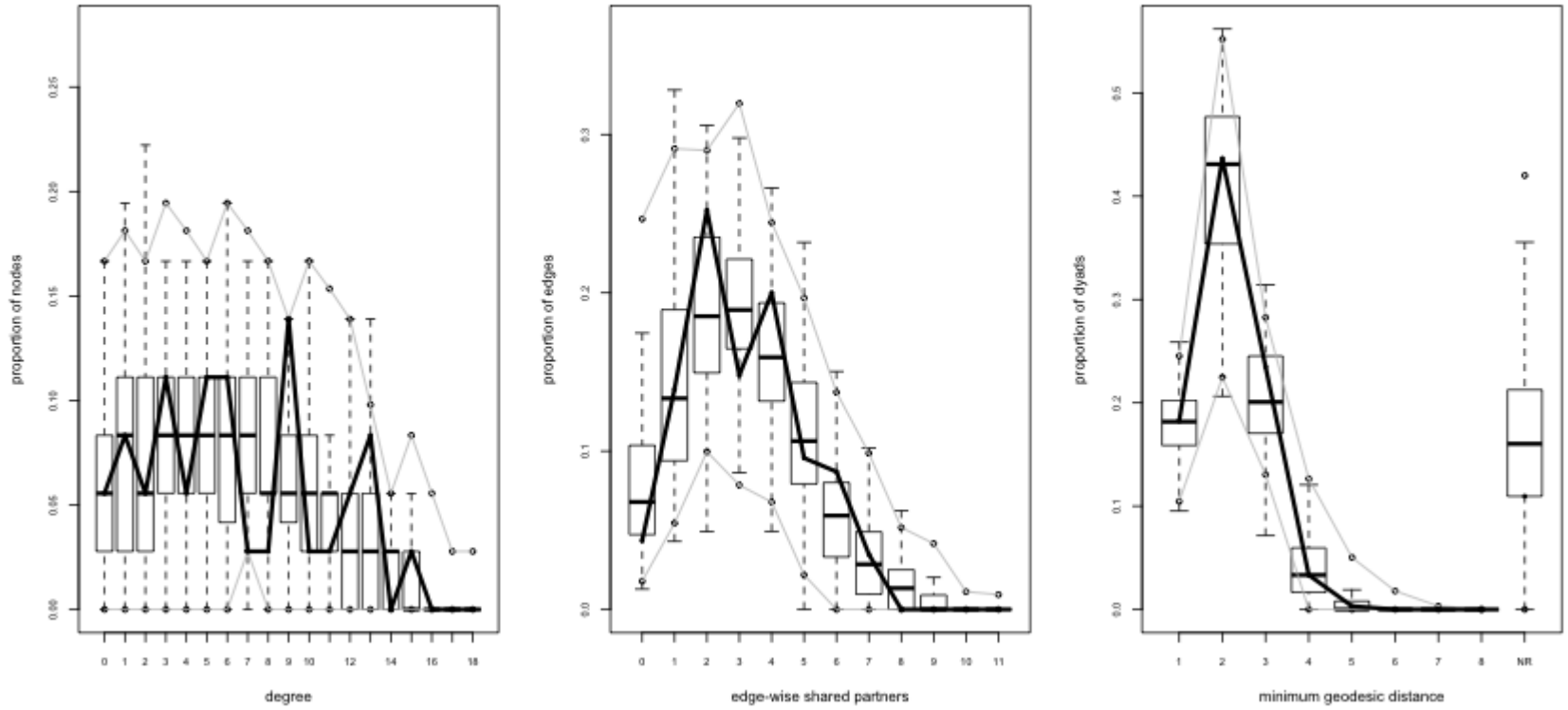
Exponential Random Graph Models

```
## # 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

estimate	std.error	statistic	p.value
-7.0065546	0.6711396	-10.439787	0.0000000
0.5916556	0.0855375	6.916915	0.0000000
0.0245612	0.0061996	3.961761	0.0000744
0.3945545	0.1021836	3.861229	0.0001128
0.7696627	0.1906006	4.038093	0.0000539
0.7376656	0.2436241	3.027885	0.0024627
1.1643929	0.1875340	6.208970	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

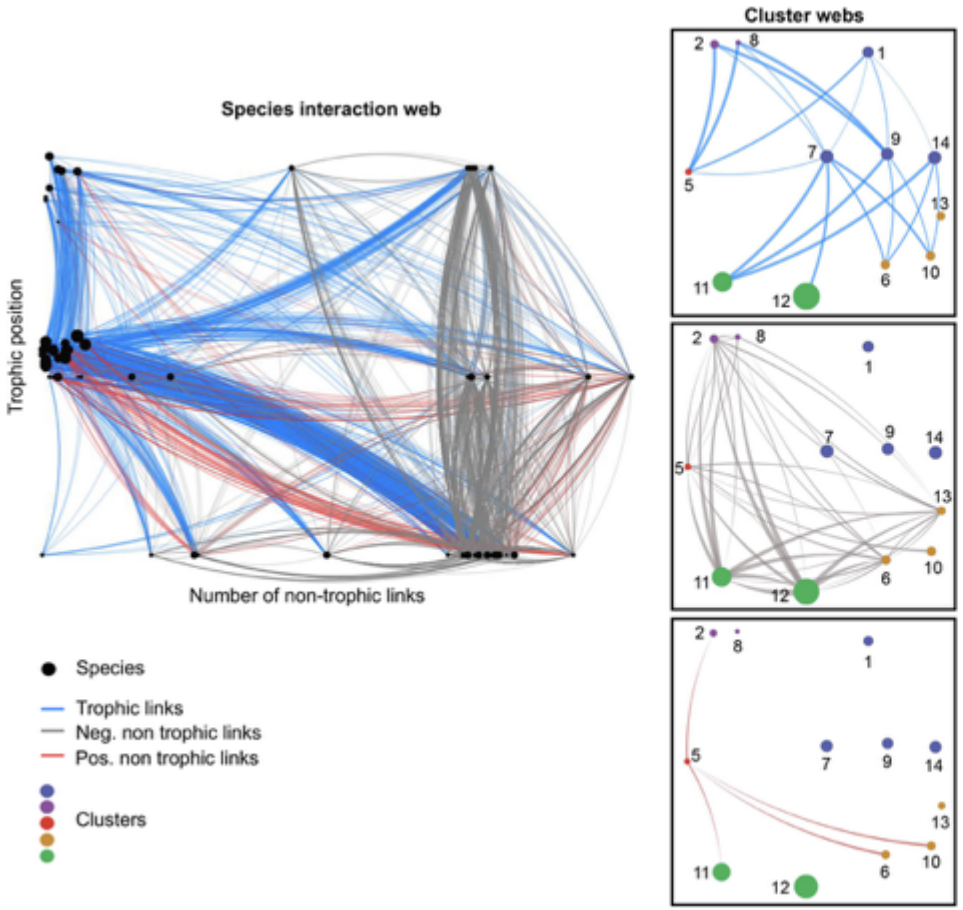
Exponential Random Graph Models

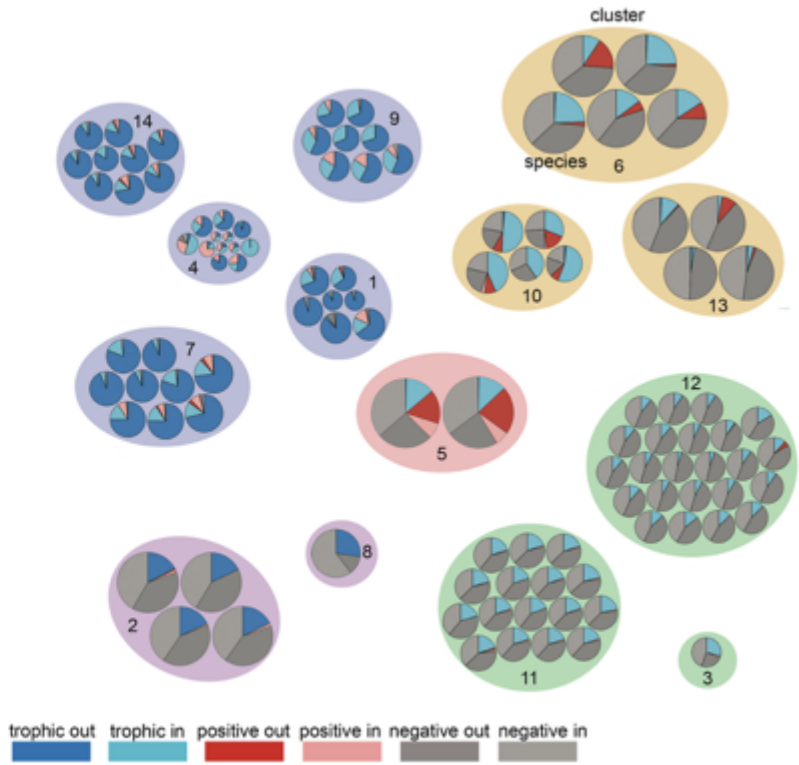
This brings about some complications since it's infeasible to define function over all possible configurations

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

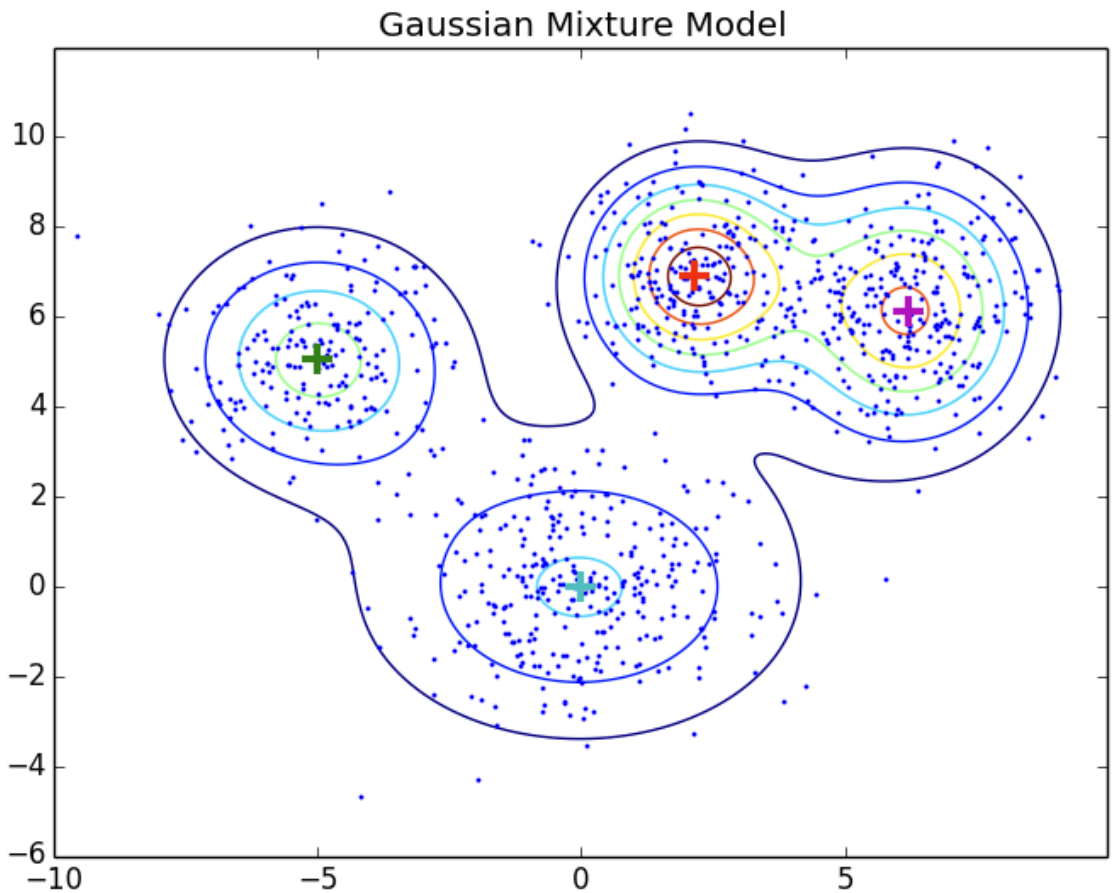
(a model like `g~match(class)` in ERGM)

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:

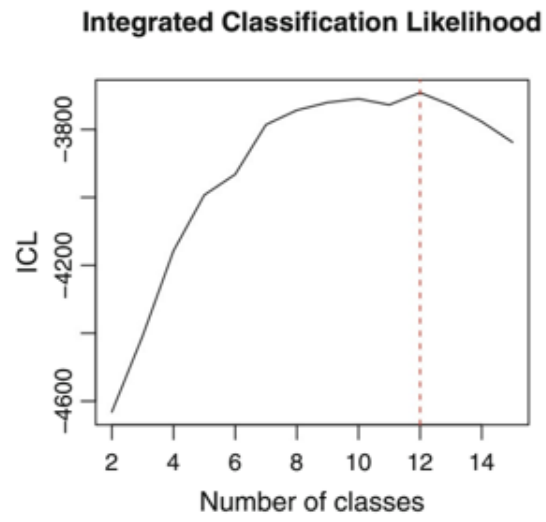
- Initialize parameters θ
- Repeat until "convergence":
  - Compute $\gamma_{iq} = E\{z_{iq}|y;\theta\} = p(z_{iq}|y;\theta)$
  - Maximize likelihood w.r.t. θ plugging in γ_{iq} for z_{iq}.

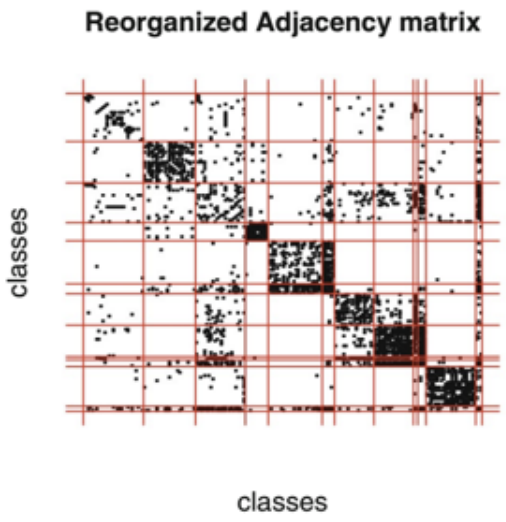
Stochastic Block Models

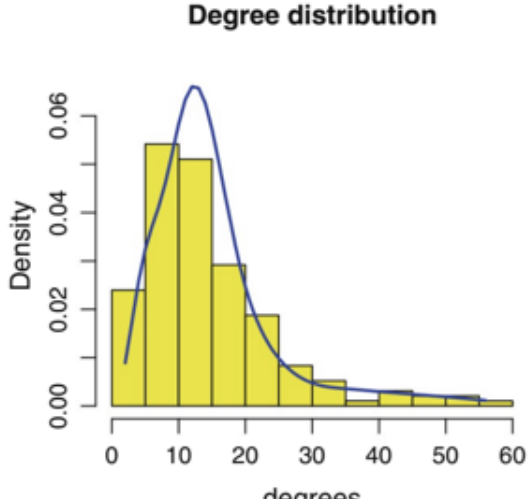
Like similar models need to determine number of classes (clusters) and select using some model selection criterion

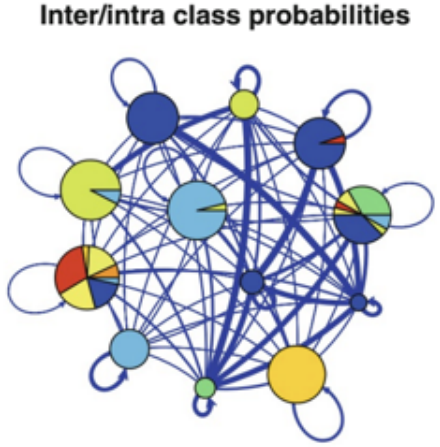
- AIC
- BIC
- Integrated Classification Likelihood

Stochastic Block Models



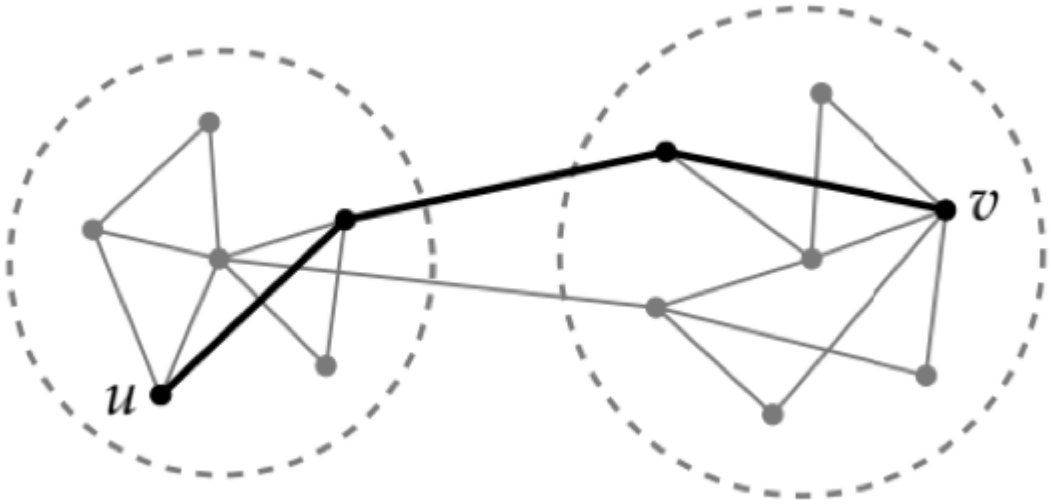






Communities

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



See http://www.pnas.org/content/106/50/21068.full for comparison of these methods

Summary

Slightly different way of thinking probabilistically about networks

Define probabilistic model over network configurations

Parameterize model using network structural properties and vertex properties

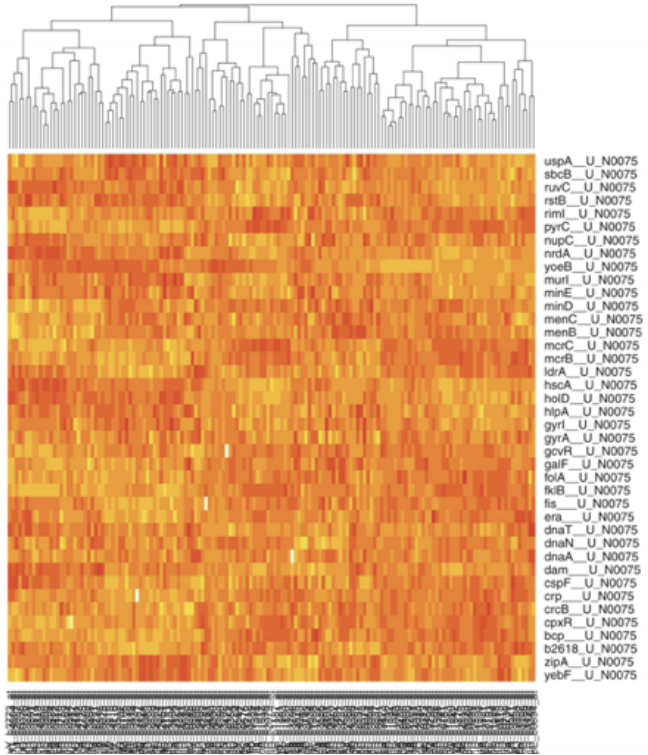
Perform inference/analysis on resulting parameters

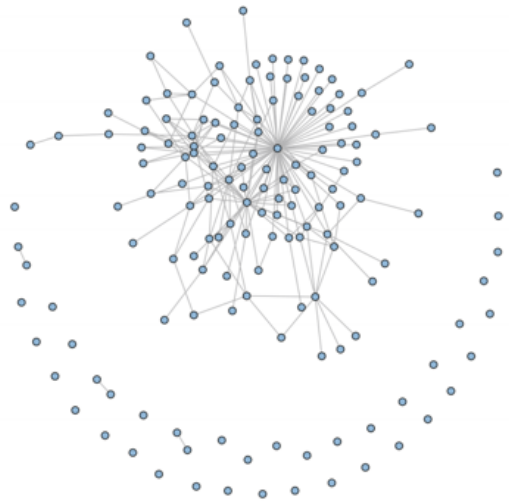
Can also extend classical clustering methodology to this setting

Learning Network Structure

How to find network structure from observational data

Gene Co-expression

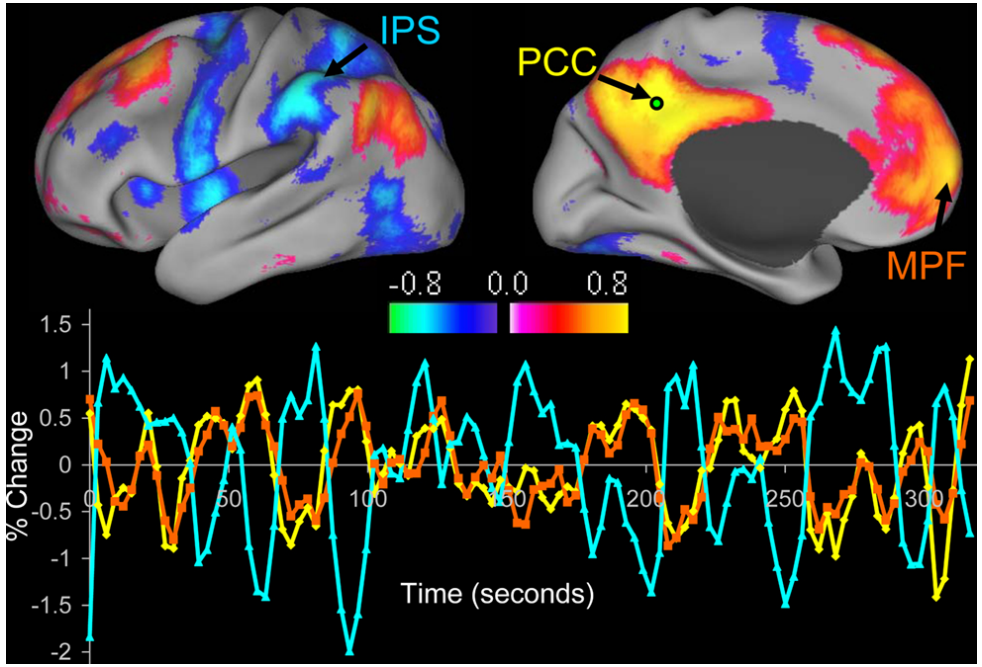




Learning Network Structure

How to find network structure from observational data

Functional Connectivity



Learning Network Structure

Correlation Networks

The simplest approach: compute correlation between observations, if correlation high, add an edge

Learning Network Structure

Assume data $y_i = \{y_{i1}, y_{i2}, \ldots, y_{iT}\}$ (e.g., gene expression of gene i in T different conditions) and y_j

Important quantity 1: the *covariance* of y_i and y_j

$$\sigma_{ij} = \frac{1}{T} \sum_{t=1}^{T} (y_{it} - \bar{y}_i)(y_{jt} - \bar{y}_j)$$

Learning Network Structure

Important quantity 1: the *covariance* of y_i and y_j

$$\sigma_{ij} = \frac{1}{T}\sum_{t=1}^{T}(y_{it} - \bar{y}_i)(y_{jt} - \bar{y}_j)$$

How do y_i and y_j vary around their means?

Learning Network Structure

We can estimate σ_{ij} from data by plugging in the mean of y_i and y_j.

We would notate the estimate as $\hat{\sigma}_{ij}$.

In the following, σ_{ij} often means $\hat{\sigma}_{ij}$, it should follow from context.

Learning Network Structure

We often need to compare quantities across different entities in system, e.g., genes, so we want to remove *scale*

Pearson's Correlation:

$$\rho_{ij} = \frac{\sigma_{ij}}{\sigma_{ii}\sigma_{jj}}$$

With σ_{ii} the standard deviation of y_i:

$$\sigma_{ii} = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(y_{it} - \bar{y}_i)}$$

Learning Network Structure

Note Pearson Correlation is between -1 and 1, it is hard to perform inference on bounded quantities, so one more transformation.

Fisher's transformation

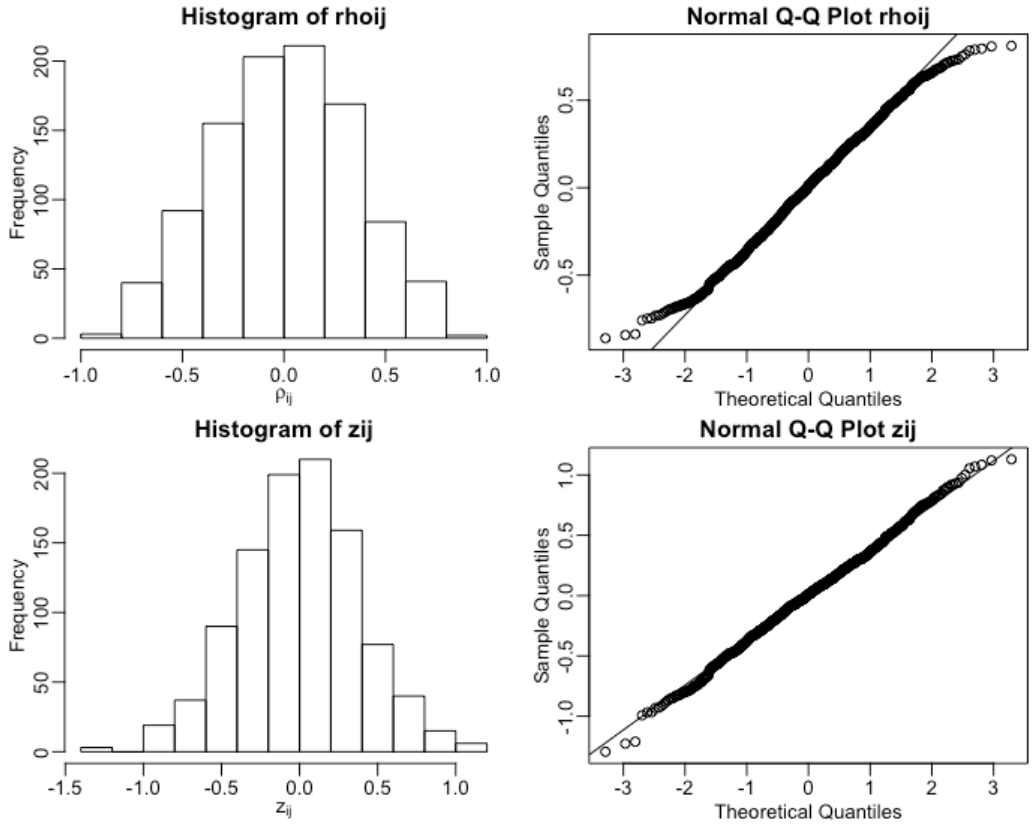
$$zij = \tanh^{-1}(\rho_{ij}) = \frac{1}{2}\log\frac{1+\rho_{ij}}{1-\rho_{ij}}$$

Learning Network Structure

Edge inference: hypothesis test

$$H_0 = \rho_{ij} = 0 \; H_A = \rho_{ij} \neq 0$$

Compute P-value p_{ij} from $N(0, \sqrt{1/(T-3)})$



Learning Network Structure

Perform hypothesis test for every *pair* of entities, i.e., possible edge i j

We would compute P-value for each possible edge

When performing many independent tests, P-values no longer have our intended interpretation

Learning Network Structure

Multiple Hypothesis Testing

	Called Significant	**Not Called Significant**	**Total**
Null True	V	$m_0 - V$	m_0
Altern. True	S	$m_1 - S$	m_1
Total	R	$m - R$	m

Note: m total tests

Learning Network Structure

Error rates

Family-wise error rate (FWER): the probability of at least one Type I error (false positive) $\mathrm{FWER} = \Pr(V \geq 1)$

We use Bonferroni procedure to control FWER.

If testing at level α (e.g., $\alpha = 0.05$), only include egdes for which P-value $p_{ij} \leq \alpha/m$

Learning Network Structure

Error rates

False Discovery Rate (FDR): rate that false discoveries occur
$\mathrm{FDR} = \mathbf{E}(V/R; R > 0)Pr(R > 0)$

We use Benjamini-Hochberg procedure to control FDR.

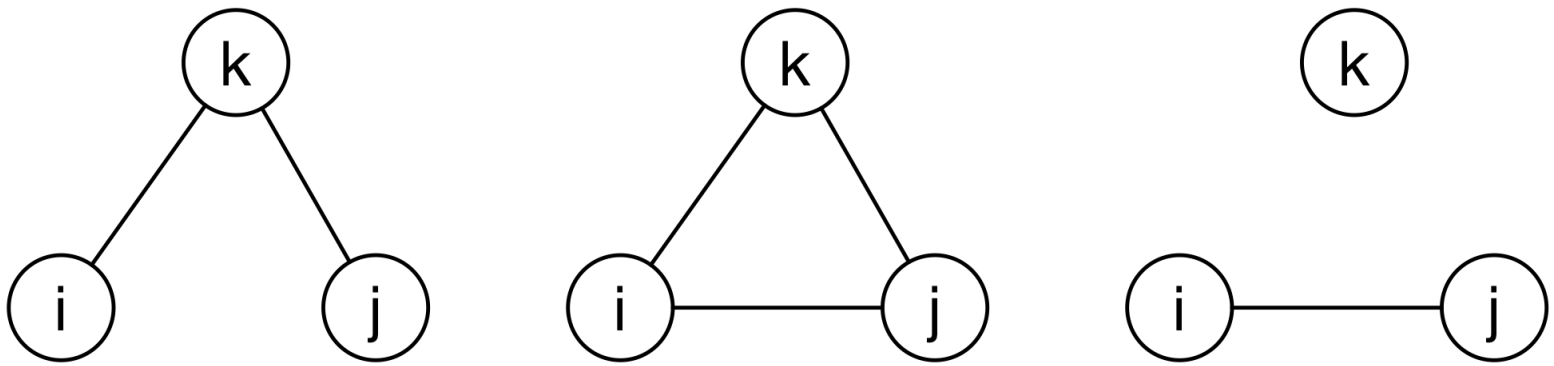
Construct list of edges at FDR level β (e.g. $\beta = 0.1$) if $p_{(k)} \leq \frac{k}{m}\beta$, where $p_{(k)}$ is the p-value for the k-th largest p-value.

Note: there are other more precise FDR controlling procedures (esp. q-values)

Learning Network Structure

The problem with Pearson's correlation

Consider the following networks, where absence of edge corresponds to true *conditional independence* between vertices in graph



In all three of these, Pearson's correlation test with ρ_{ij} is likely statistically significant.

Learning Network Structure

Let's extend the way we think about the situation. First consider covariance *matrix* for i, j and k

$$\Sigma = \begin{pmatrix} \sigma_{ii}^2 & \sigma_{ij} & \sigma_{ik} \\ \sigma_{ji} & \sigma_{jj}^2 & \sigma_{jk} \\ \sigma_{ki} & \sigma_{kj} & \sigma_{kk}^2 \end{pmatrix}$$

Learning Network Structure

We can then think about the covariance of i and j *conditioned* on k

$$\Sigma_{ij|k} = \begin{pmatrix} \sigma_{ii}^2 & \sigma_{ij} \\ \sigma_{ji} & \sigma_{jj}^2 \end{pmatrix} - \sigma_{kk}^{-2} \begin{pmatrix} \sigma_{ik}^2 & \sigma_{ik}\sigma_{jk} \\ \sigma_{ik}\sigma_{jk} & \sigma_{jk}^2 \end{pmatrix}$$

How do y_i and y_j co-vary around their *conditional* means $E(y_i|y_k)$ and $E(y_j|y_k)$

Learning Network Structure

Partial correlation networks

This leads to the concept of partial correlation (which we can derive from the conditional covariance)

$$\rho_{ij|k} = \frac{\rho_{ij} - \rho_{ik}\rho_{jk}}{\sqrt{(1-\rho_{ik}^2)}\sqrt{(1-\rho_{jk}^2)}}$$

Learning Network Structure

Partial correlation networks

What's the test now? No edge if i and j are conditionally independent (there is some k such that $\rho_{ij|k} = 0$)

Formally:

$$H_0 : \ \rho_{ij|k} = 0 \text{ for some } k \in V_{\setminus \{i,j\}}$$

$$H_A : \ \rho_{ij|k} \neq 0 \text{ for all } k$$

Learning Network Structure

Partial correlation networks

To determine edge $i \sim j$ compute P-value p_{ij} as

$$p_{ij} = \max\{p_{ij|k} : \ k \in V_{\setminus\{i,j\}}\}$$

where $p_{ij|k}$ is a P-value computed from (transformed) partial correlation $\rho_{ij|k}$

Use multiple testing correction as before

Learning Network Structure

Problems with partial correlation networks

For every edge, must compute partial correlation wrt. every other vertex

Compound hypothesis tests like the above are harder to control for multiple testing (i.e., correction mentioned above is not quite right)

The dependence structure they represent is unclear

Learning Network Structure

Here we turn to a very powerful abstraction, thinking of graphs as a way of describing the *joint* distribution of gene expression measurements (Probabilistic Graphical Models).

Graphical Models

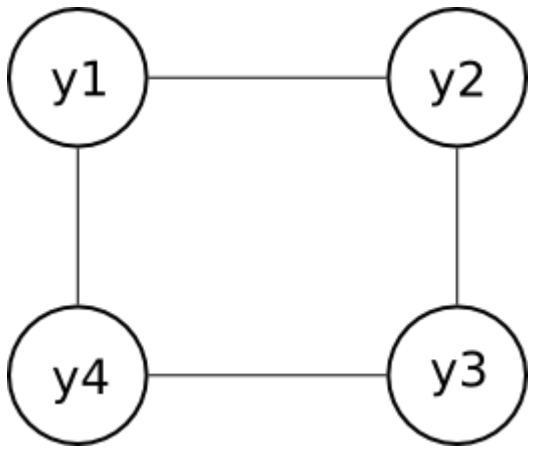
Consider each *complete vector* of expression measurements at each time $\mathbf{y}$

Suppose some *conditional independence* properties hold for some variables in $\mathbf{y}$,

- Example: variable y_2 and y_3 are independent given *remaining* variables in $\mathbf{y}$

Graphical Models

We can encode these conditional independence properties in a graph.



Graphical Models

Hammersley-Clifford theorem: all probability distributions that satisfy conditional independence properties in a graph can be written as

$$P(\mathbf{y}) = \frac{1}{Z}\exp\{\sum_{c \in C} f_c(\mathbf{y}_c)\}$$

C is the set of all *cliques* in a graph, c a specific clique and $\mathbf{y_c}$ the variables in the clique.

Graphical Models

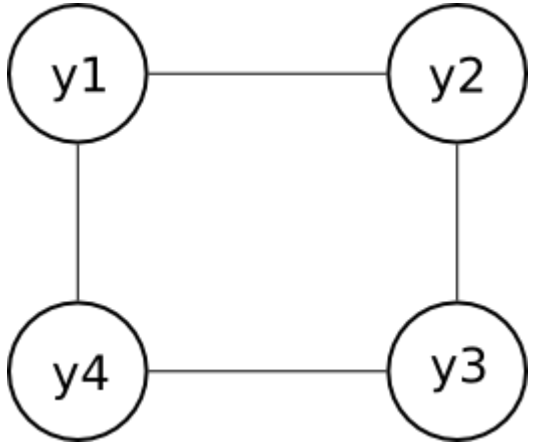
The probability distribution is determined by the choice of potential functions f_c. Example:

1. $f_c(y_i) = -\frac{1}{2}\tau_{ii}y_i^2$
2. $f_c(\{y_i, y_j\}) = -\frac{1}{2}\tau_{ij}y_i y_j$
3. $f_c(\mathbf{y_c}) = 0$ for $|y_c| \geq 3$

Graphical Gaussian Models

Define matrix Σ^{-1} as

1. $\Sigma^{-1}_{ij} = \tau_{ij}$ if there is an edge between y_i and y_j
2. $\Sigma^{-1}_{ij} = 0$ otherwise



$$\Sigma^{-1} = \begin{pmatrix} \tau_{11}^2 & \tau_{12} & 0 & \tau_{14} \\ \tau_{12} & \tau_{22}^2 & \tau_{23} & 0 \\ 0 & \tau_{23} & \tau_{33}^2 & \tau_{34} \\ \tau_{14} & 0 & \tau_{34} & \tau_{44}^2 \end{pmatrix}$$

Graphical Gaussian Models

With this in place, we can say that $\mathbf{y}$ is distributed as *multivariate* normal distribution $N(0, \Sigma)$.

Connection to partial correlation: We can think about distribution of y_i and y_j conditioned on the rest of the graph $V_{\setminus\{i,j\}}$ and the (partial) correlation of y_i and y_j under this distribution

$$\rho_{ij|V_{\setminus\{i,j\}}} = -\frac{\tau_{ij}}{\tau_{ii}\tau_{jj}}$$

Sparse Inverse Covariance

With this framework in place we can now think of network structure inference.

Main idea: given draws from multivariate distribution $\mathbf{y}$ (i.e., expression vector at each time point),

Estimate a *sparse* inverse correlation matrix, get graph from the pattern of 0's in the estimated matrix

Banerjee, et al. ICML 2006, JMLR 2008, Friedman Biostatistics 2007

Sparse Inverse Covariance

Maximum Likelihood estimate of inverse covariance is given by solution to

$$\max_{X \succ 0} \log \det X - (SX)$$

S is the estimated sample covariance matrix

$$S = \sum_{t=1}^{T} \mathbf{y_t} \mathbf{y_t}'$$

(Yuck)

Sparse Inverse Covariance

We can induce zeros in the solution using a penalized likelihood estimate

$$\max_{X \succ 0} \log \det X - (SX) - \lambda \|X\|_1$$

where

$$\|X\|_1 = \sum_{ij} X_{ij}$$

(Yuckier)

Sparse Inverse Covariance

Block-coordinate ascent

Solve by maximizing one column of matrix at a time (edges for each variable, e.g., x_{12} below)

$$\min_{\beta} \frac{1}{2} \|X_{11}^{1/2}\beta - z\|^2 + \lambda\|\beta\|_1$$

with $z = W_{11}^{1/2} s_{12}$ and

$$X = \begin{pmatrix} X_{11} & x_{12} \\ x_{12} & x_{22} \end{pmatrix} \qquad S = \begin{pmatrix} S_{11} & s_{12} \\ s_{12} & s_{22} \end{pmatrix}$$

Sparse Inverse Covariance

Block-coordinate ascent

Solve by maximizing one column of matrix at a time (edges for each variable, e.g., x_{12} below)

$$\min_{\beta} \frac{1}{2} \|X_{11}^{1/2}\beta - z\|^2 + \lambda\|\beta\|_1$$

Solution is then $x_{12} = W_{11}\beta$

(This is l1-regularized least squares, easy to solve, not yucky at all)

Sparse Inverse Covariance

Block-coordinate ascent

Solve by maximizing one column of matrix at a time (edges for each variable, e.g., x_{12} below)

$$\min_{\beta} \frac{1}{2} \|X_{11}^{1/2}\beta - z\|^2 + \lambda\|\beta\|_1$$

Iterate over columns of matrix until β converges (or even better, until objective function converges)

Summary

Using Gaussian Graphical Model representation

- multivariate normal probability over a sparse graph
- take resulting graph as e.g., *gene network*

Use sparsity-inducing regularization (l1-norm)

Block-coordinate ascent method leads to l1-regularized regression at each step

- Can use efficient coordinate descent (soft-thresholding) to solve regression problem