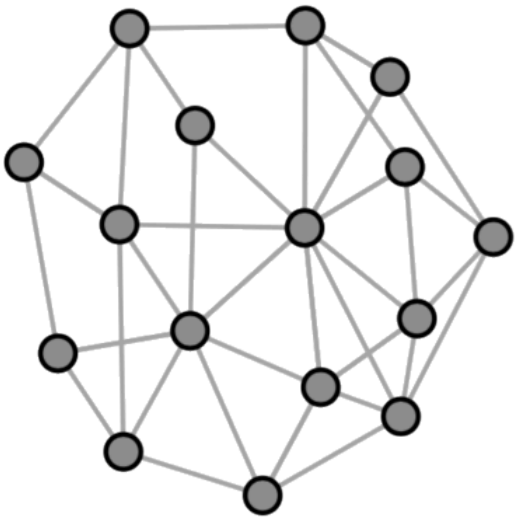
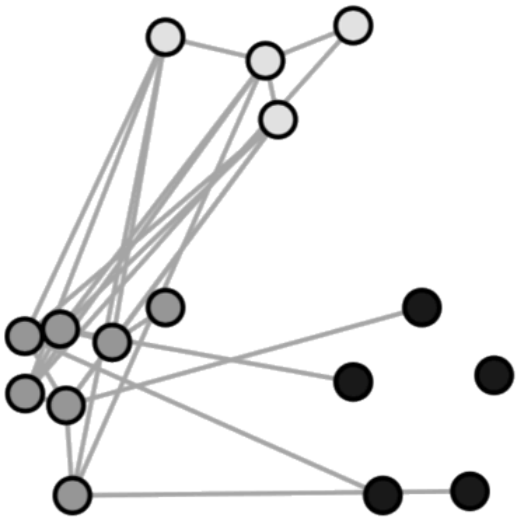
Advanced Network Analysis

Latent Distance Model

Shahryar Minhas [s7minhas.com]

What are we missing?





- **Homophily**: "birds of a feather flock together"
- **Stochastic equivalence**: nothing as pithy to say here, but this model focuses on identifying actors with similar roles

Now we'll start to build on what we have so far and find an expression for γ:

$$y_{ij} \approx \beta^T X_{ij} + a_i + b_j + \gamma(u_i, v_j)$$

Network dependencies

Dependence among the ties in network data is the defining feature that makes networks so interesting and challenging to handle inferentially.

Possible Approaches:

- Latent variable models: This is what we've been doing this week with the SRM, SBM, and today LDM. Goal of these models is to use a lightly supervised approach to fit out the dependencies.

- ERGMs/SAOMS: Next week, you'll go over ERGMs, which try to do something similar at the *graph level* by adding in explict terms for transitivity and reciprocity.

Key differences are:

- Does not require the detailed theory to specify an ERGM.
- Will not exhibit degeneracy issues.
- Drawback: Cant directly test the effect of transitivity at the graph level.

Homophily

Dependence A recurrent finding across network domains and disciplines is that nodes that are alike on salient dimensions are more likely to interact.

Examples:

- Individuals of the same race are more likely to be friends.
- Legislators of similar ideological leaning are more likely to collaborate.
- Countries with similar governing systems are more likely to form military alliances.
- People of the same gender are more likely to start businesses together.

The LDM offers a method to analyze and account for homophily along multiple dimensions, without measuring any of the homophilous attributes.

Basic setup

- Let each node be represented by a k dimensional vector, $\mathbf{z} = \{z_1, z_2, \ldots, z_k\}$ of "latent" attributes.
- Define d_{ij} as the Euclidean distance between the latent attributes of node i and node j.

$$d_{ij} = \sqrt{\sum_{h=1}^{k} \left(z_h^{(i)} - z_h^{(j)} \right)^2}$$

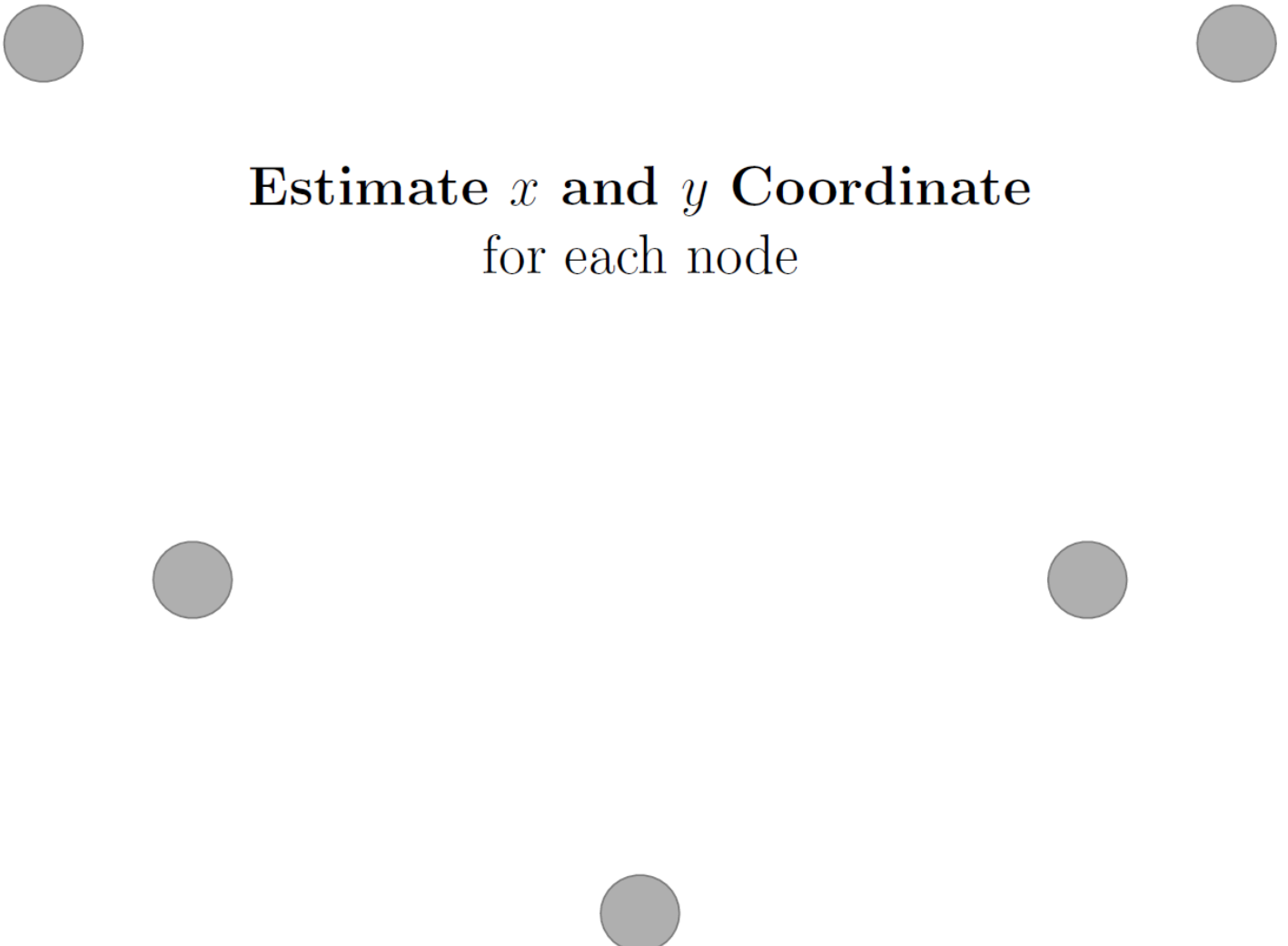
- Then the probability of an edge from i to j is

$$p_{ij} = \text{logit}^{-1} \left(\beta_0 - d_{ij} \right)$$

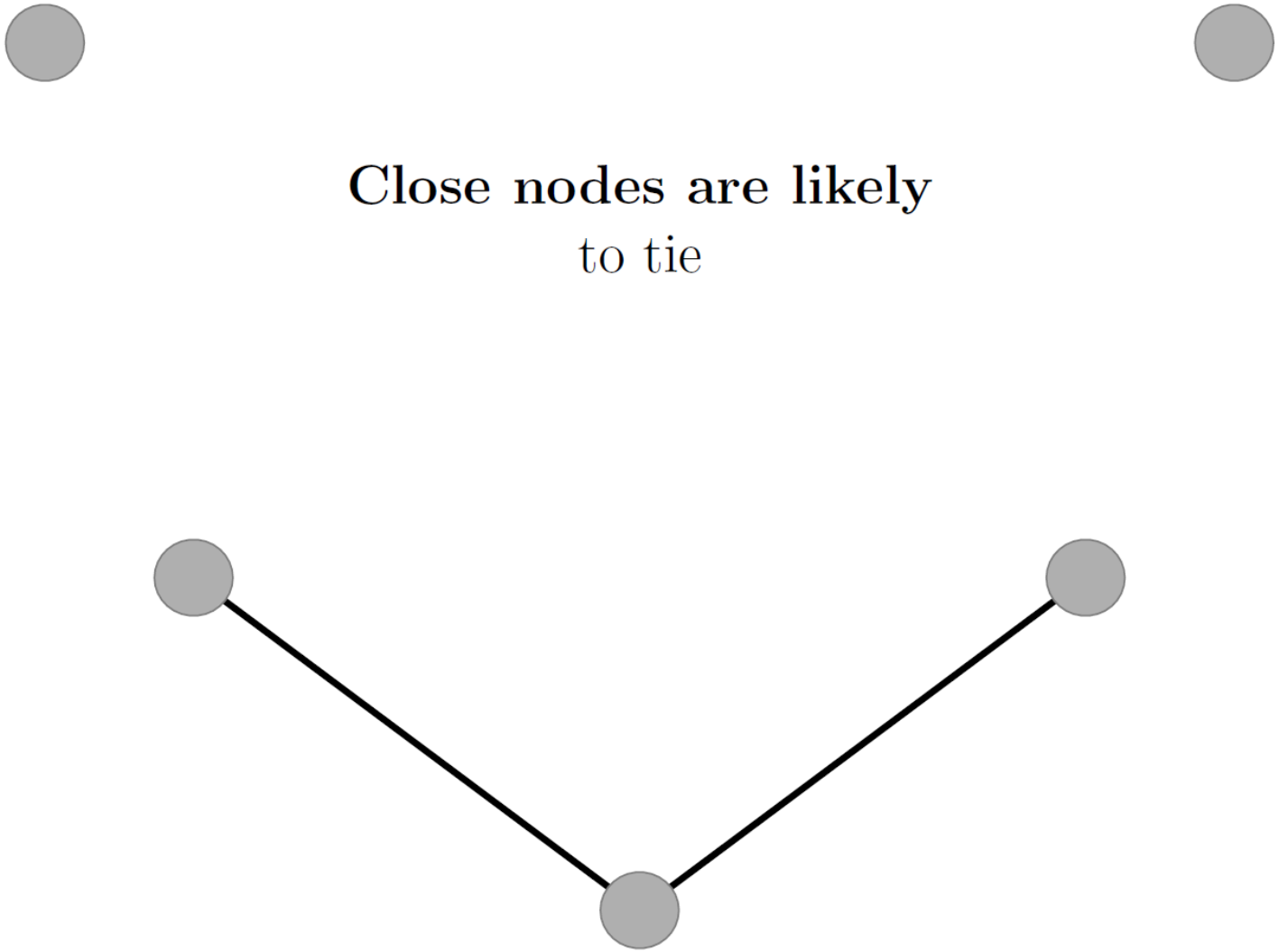
  where $\text{logit}^{-1}(x) = \frac{1}{1+\exp(-x)}$
- β_0 controls overall density of the network and the $\mathbf{z}$ models who connects to whom.

Key Innovation: attributes are inferred through estimation.

Step 1

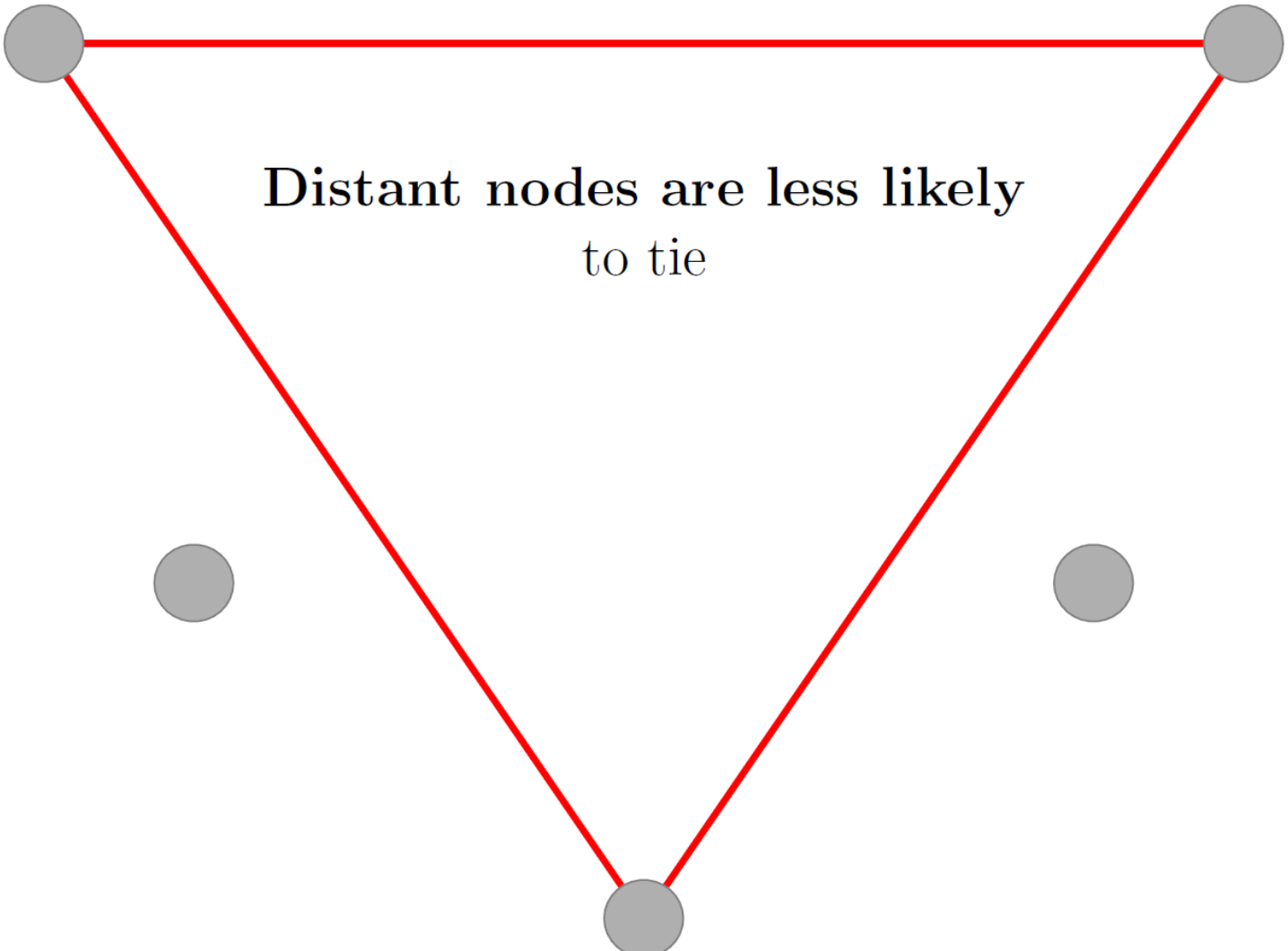


Step 2



Close nodes are likely
to tie

Step 3



Common extensions

Adding in covariates

$$p_{ij} = \text{logit}^{-1}\left(\beta_0 + \beta_1 x_{ij} - d_{ij}\right)$$

- x_{ij} is measured and not inferred
- β_1 gives the change in log odds (in the binary case)

Accounting for node-level sociality

$$p_{ij} = \text{logit}^{-1}\left(\beta_0 + \beta_1 x_{ij} + \alpha_i + \eta_j - d_{ij}\right)$$

- α_i is the `sender effect' of`(i)
- η_j is the `reciever effect' of`(j)

Can be accomodated to examine various distribution types

- **Normal**: $\beta_0 + \beta_1 x_{ij} + \alpha_i + \eta_j - d_{ij}$
- **Poisson**: $\lambda_{ij} = \exp\left[\beta_0 + \beta_1 x_{ij} + \alpha_i + \eta_j - d_{ij}\right]$

Beyond homophily

Homophily constitutes the motivating process for the LSM. However, it can accommodate many other network properties.

Implicit Transitivity: The structure of the latent space model implies transitivity, due to the triangle inequality... $d_{ij} < d_{ik} + d_{jk}$. If i is likely to tie to k and j is likely to tie to k then i is likely to tie to j.

Symmetry and Reciprocity: If p_{ij} is high because d_{ij} is low, then p_{ji} will also be high.

Preferential Attachment: Latent space can represent tendency towards popularity by placing more popular nodes at central positions.

Latent distance model

(Hoff et al. 2002; Krivitsky et al. 2009; Sewell & Chen 2015)

Each node i has an unknown latent position

$$\mathbf{u}_i \in \mathbb{R}^k$$

The probability of a tie from i to j depends on the distance between them

$$Pr(Y_{ij} = 1|\mathbf{u}_i, \mathbf{u}_j) = \alpha - |\mathbf{u}_i - \mathbf{u}_j|$$

- Nodes nearby one another are more likely to have a tie, and will likely have similar ties to others

Software packages:

- CRAN: `latentnet` (Krivitsky et al. 2015)
- CRAN: `VBLPCM` (Salter-Townshend 2015)

Latent Distance Model: Estimation Process

1. Model Setup:
   - Each node i has a latent position $\mathbf{u}_i \in \mathbb{R}^k$
   - Distance between nodes: $d_{ij} = |\mathbf{u}_i - \mathbf{u}_j|$
   - Probability of tie: $P(Y_{ij} = 1|\mathbf{u}_i, \mathbf{u}_j) = \text{logit}^{-1}(\alpha - d_{ij})$
   - α controls overall network density

Likelihood Function

The likelihood of observing the network given latent positions:

$$L(\mathbf{u}, \alpha | Y) = \prod_{i<j} P(Y_{ij} = 1 | \mathbf{u}_i, \mathbf{u}_j)^{Y_{ij}} (1 - P(Y_{ij} = 1 | \mathbf{u}_i, \mathbf{u}_j))^{1-Y_{ij}}$$

Where:

- $\mathbf{u}$ is the set of all latent positions
- Y is the observed network

Interpreting the Likelihood Function in Distance Models

The likelihood function is trying to answer the question:

"How probable is our observed network given our current guess about node positions and overall connectivity?"

Intuitive breakdown:

1. For each pair of nodes:
    - If they're connected, how likely is that given their distance?
    - If they're not connected, how likely is that given their distance?
2. Multiply all these probabilities together.
3. The result tells us how well our current node positions explain the observed network.
4. We want to find node positions that make our observed network as likely as possible.
5. This process helps us uncover hidden spatial relationships between nodes that best explain the network structure we see.

Estimation Approaches: Maximum Likelihood Estimation (MLE)

Objective: Maximize $L(\mathbf{u}, \alpha|Y)$ or equivalently, $\log L(\mathbf{u}, \alpha|Y)$

Process:

1. Define the log-likelihood function:

$$\log L(\mathbf{u}, \alpha|Y) = \sum_{i<j} [Y_{ij} \log(\text{logit}^{-1}(\alpha - |\mathbf{u}_i - \mathbf{u}_j|)) + (1 - Y_{ij}) \log(1 - \text{logit}^{-1}(\alpha - |\mathbf{u}_i - \mathbf{u}_j|))]$$

1. Use optimization algorithms (e.g., gradient descent, Newton-Raphson) to find $\mathbf{u}$ and α that maximize this function

2. Compute gradients: $\frac{\partial \log L}{\partial \mathbf{u}_i} = \sum_{j \neq i} (Y_{ij} - \text{logit}^{-1}(\alpha - |\mathbf{u}_i - \mathbf{u}_j|)) \frac{\mathbf{u}_i - \mathbf{u}_j}{|\mathbf{u}_i - \mathbf{u}_j|}$
   $\frac{\partial \log L}{\partial \alpha} = \sum_{i<j} (Y_{ij} - \text{logit}^{-1}(\alpha - |\mathbf{u}_i - \mathbf{u}_j|))$

Estimation Approaches: Maximum Likelihood Estimation (MLE) Cont'd

Challenges:

- High-dimensional: $O(nk)$ parameters for n nodes in k dimensions
- Non-convex: Multiple local optima possible
- Computational complexity: $O(n^2)$ operations per iteration

Strategies:

- Multiple random initializations to avoid local optima
- Use of stochastic gradient descent for large networks
- Dimensionality reduction techniques as preprocessing

Estimation Approaches: Bayesian Estimation

1. Specify priors:
    - For latent positions: $p(\mathbf{u})$, often multivariate normal
    - For intercept: $p(\alpha)$, often normal or uniform
2. Define the posterior distribution: $p(\mathbf{u}, \alpha|Y) \propto L(\mathbf{u}, \alpha|Y) \cdot p(\mathbf{u}) \cdot p(\alpha)$
3. Use MCMC methods to sample from this posterior:
    - Metropolis-Hastings algorithm
    - Gibbs sampling (if conditional distributions are available)
    - Hamiltonian Monte Carlo for more efficient sampling in high dimensions

Estimation Approaches: Bayesian Estimation Cont'd

Advantages:

- Provides uncertainty quantification for estimates
- Can incorporate prior knowledge
- Handles missing data and model extensions more naturally

Challenges:

- Computationally intensive
- Requires careful tuning of MCMC algorithms
- Assessing convergence can be non-trivial

Estimation Process: MCMC in Detail

1. Initialization:
   - Generate random starting values for $\mathbf{u}$ and α
   - Often use MLE estimates or spectral embedding as starting points
2. For each iteration t: a. Update $\mathbf{u}$:
   - For each node i:
     - Propose new position $\mathbf{u}_i^* \sim q(\mathbf{u}_i^* | \mathbf{u}_i^{(t-1)})$ (e.g., $\mathbf{u}_i^* = \mathbf{u}_i^{(t-1)} + \epsilon$, where $\epsilon \sim N(0, \sigma^2 I)$)
     - Compute acceptance ratio: $r = \frac{p(\mathbf{u}_i^* | Y, \mathbf{u}_{-i}^{(t-1)}, \alpha^{(t-1)})}{p(\mathbf{u}_i^{(t-1)} | Y, \mathbf{u}_{-i}^{(t-1)}, \alpha^{(t-1)})} \cdot \frac{q(\mathbf{u}_i^{(t-1)} | \mathbf{u}_i^*)}{q(\mathbf{u}_i^* | \mathbf{u}_i^{(t-1)})}$
     - Accept $\mathbf{u}_i^*$ with probability $\min(1, r)$

   b. Update α:
   - Propose new $\alpha^* \sim q(\alpha^* | \alpha^{(t-1)})$
   - Compute acceptance ratio similar to above
   - Accept α^* with probability $\min(1, r)$

Estimation Process: MCMC in Detail Cont'd

1. Convergence assessment:
    - Monitor trace plots of parameters
    - Gelman-Rubin statistic for multiple chains
2. Post-processing:
    - Discard burn-in period
    - Thin samples if necessary
    - Compute posterior means, credible intervals, etc.

Challenges:

- Tuning proposal distributions for efficient mixing
- Dealing with label switching (identifiability issues)
- High autocorrelation in samples for some parameters

Model Extensions

1. Covariates: $P(Y_{ij} = 1 | \mathbf{u}_i, \mathbf{u}_j, \mathbf{x}_{ij}) = \text{logit}^{-1}(\alpha + \beta^\top \mathbf{x}_{ij} - d_{ij})$

2. Node-specific effects: $P(Y_{ij} = 1 | \mathbf{u}_i, \mathbf{u}_j) = \text{logit}^{-1}(\alpha + a_i + b_j - d_{ij})$

    - a_i: sender effect for node i
    - b_j: receiver effect for node j

3. Different distance metrics: e.g., Euclidean, Manhattan, Mahalanobis

Challenges and Considerations

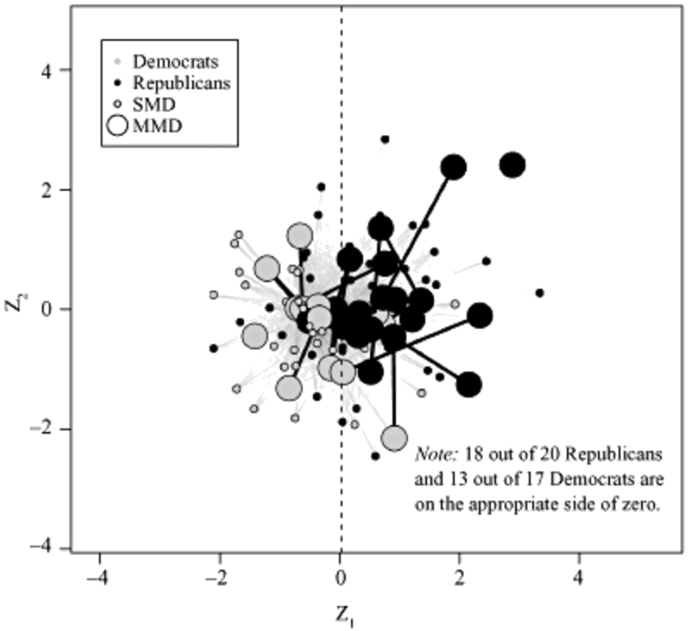
- Identifiability: Latent positions are unique up to rotation, reflection, and translation
- Dimensionality: Choosing appropriate k (typically 2 or 3 for visualization)
- Computational complexity: Scales poorly with network size
- Model comparison: Use information criteria (AIC, BIC) or cross-validation

Interpreting Results

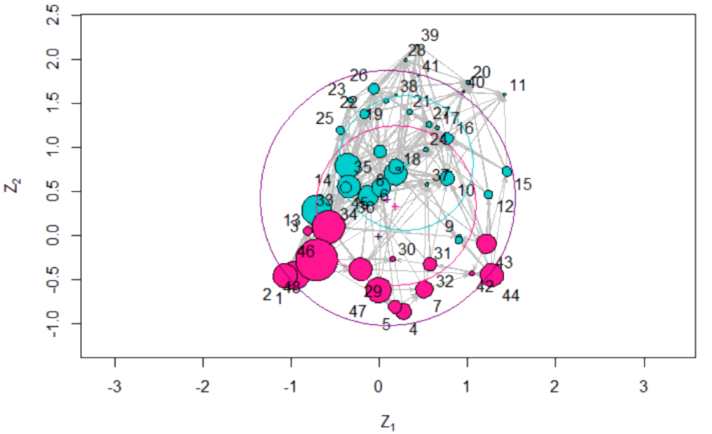
- Visualize latent space: Plot nodes in 2D or 3D space
- Examine distances: Interpret proximity in latent space as similarity
- Predictive checks: Compare observed network statistics to those from model-generated networks

LDM for low dim representations of homophily

Kirkland (2012): North Carolina Legislators



Kuh et al. (2015): Discerning prey and predators from food web



Apply LDM to trade

- We're going to use the `latentnet` package to run our first latent distance model
- First step is to format our data:

```
library(latentnet)
library(intergraph)

# first step lets create network object
load('tradeExampleData.rda')
yGraph = igraph::graph.adjacency(Y,
  mode='directed',
  weighted=TRUE,
  diag=FALSE
  )

# need to convert to network graph object,
# while preserving weighted edge structure
yNet = intergraph::asNetwork(yGraph)
list.edge.attributes(yNet)
```

```
## Warning: `graph.adjacency()` was deprecated in igraph 2.0.0.
## ℹ Please use `graph_from_adjacency_matrix()` instead.
## This warning is displayed once every 8 hours.
```

Run LDM

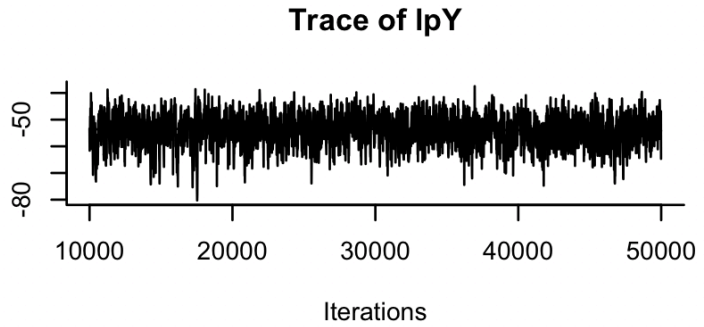
- The `latentnet` package belongs to the `statnet` suite, so it has tons and tons of documentation and tutorials
- The vignette by Krivitsky & Handcock (2008) is a great place to start.

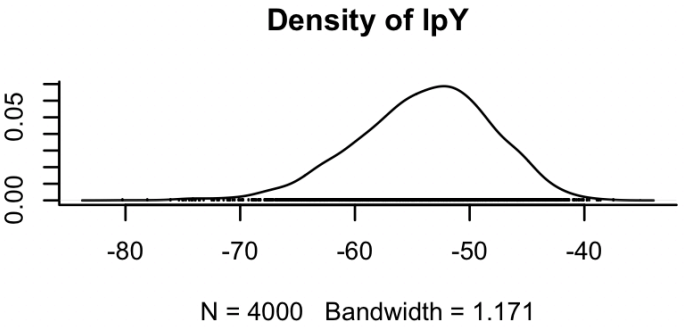
```
y.var<-sd(c(Y), na.rm=TRUE)
lsEucl = ergmm(
  yNet ~ euclidean(d=2),
  family='normal',
  fam.par=list(
    prior.var=4*sd(c(Y), na.rm=TRUE),
    prior.var.df=2 # certainty of the prior, higher more certain
    ) )
```

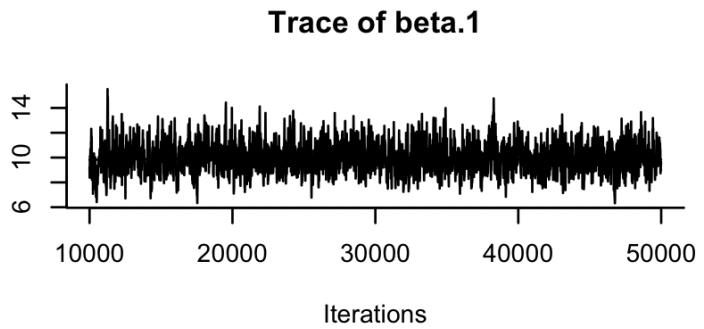
LDM MCMC Convergence

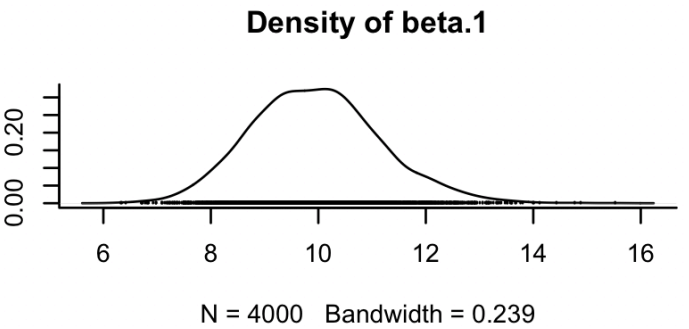
Similar to the `ergm` package, we will want to evaluate convergence:

```
mcmc.diagnostics(lsEucl)
```









Trace of Z.1.1

Density of Z.1.1

Pulling out nodal positions

From the LDM, we can pull out the positions of actor in a euclidean latent space:

```
zPos = summary(lsEucl)$'pmean'$Z
head(zPos)
```

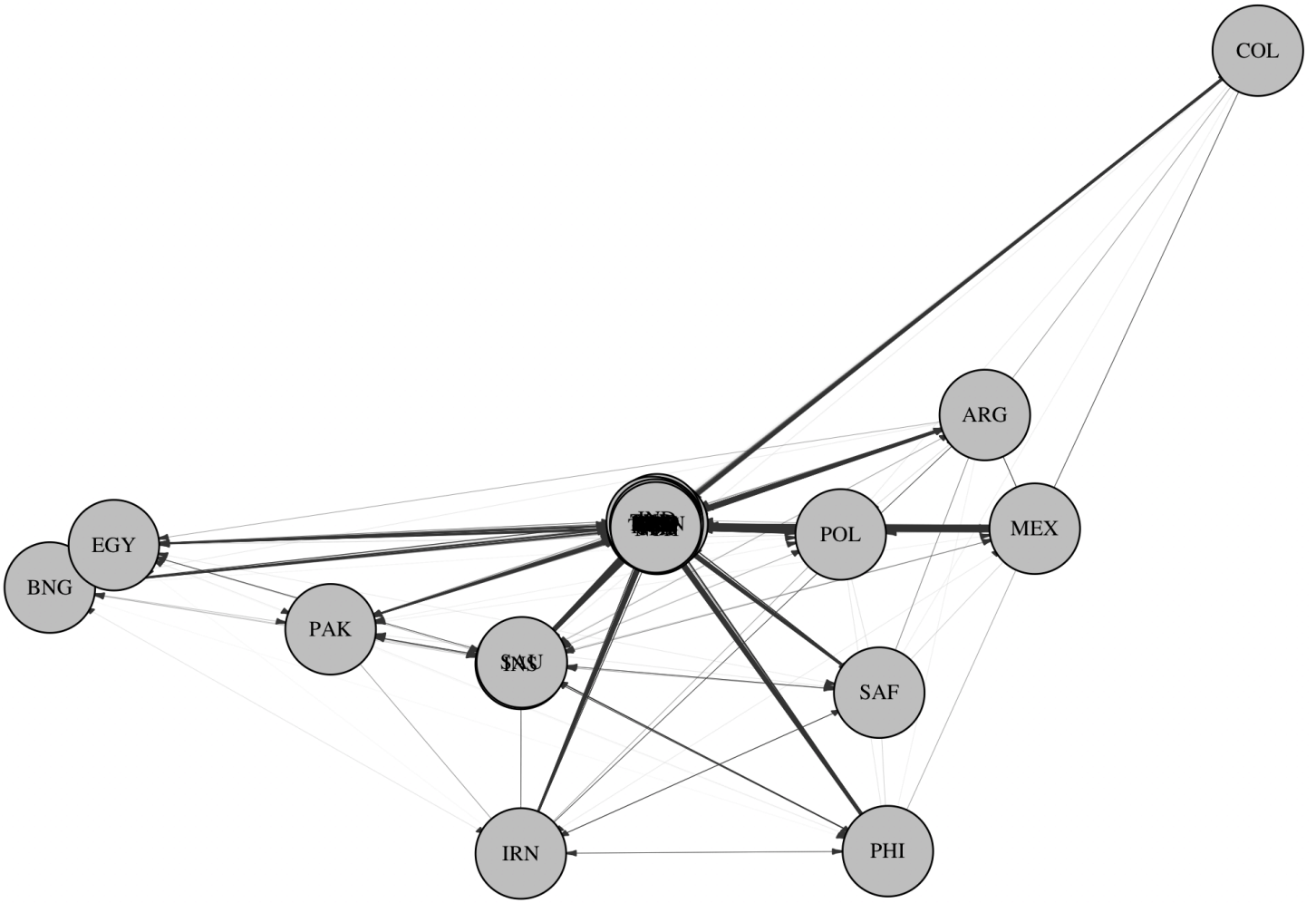
```
##               [,1]        [,2]
## [1,]  0.137887152  0.06491966
## [2,]  0.004919287  0.01391376
## [3,]  0.005987010  0.01375857
## [4,] -0.238257197 -0.01536262
## [5,]  0.005789981  0.01425025
## [6,]  0.005802527  0.01382764
```

Lets visualize the results

To visualize the results, we'll just use `igraph` and for the layout will provide the estimated positions of actors from the LDM:

```
plot(yGraph,
  layout=zPos,
  vertex.color='grey',
  vertex.label.color='black',
  vertex.size=V(yGraph)$size,
  vertex.label.cex =.75,
  edge.color='grey20',
  edge.width=E(yGraph)$weight,
  edge.arrow.size=.2,
  asp=FALSE
  )
```

Lets visualize the results

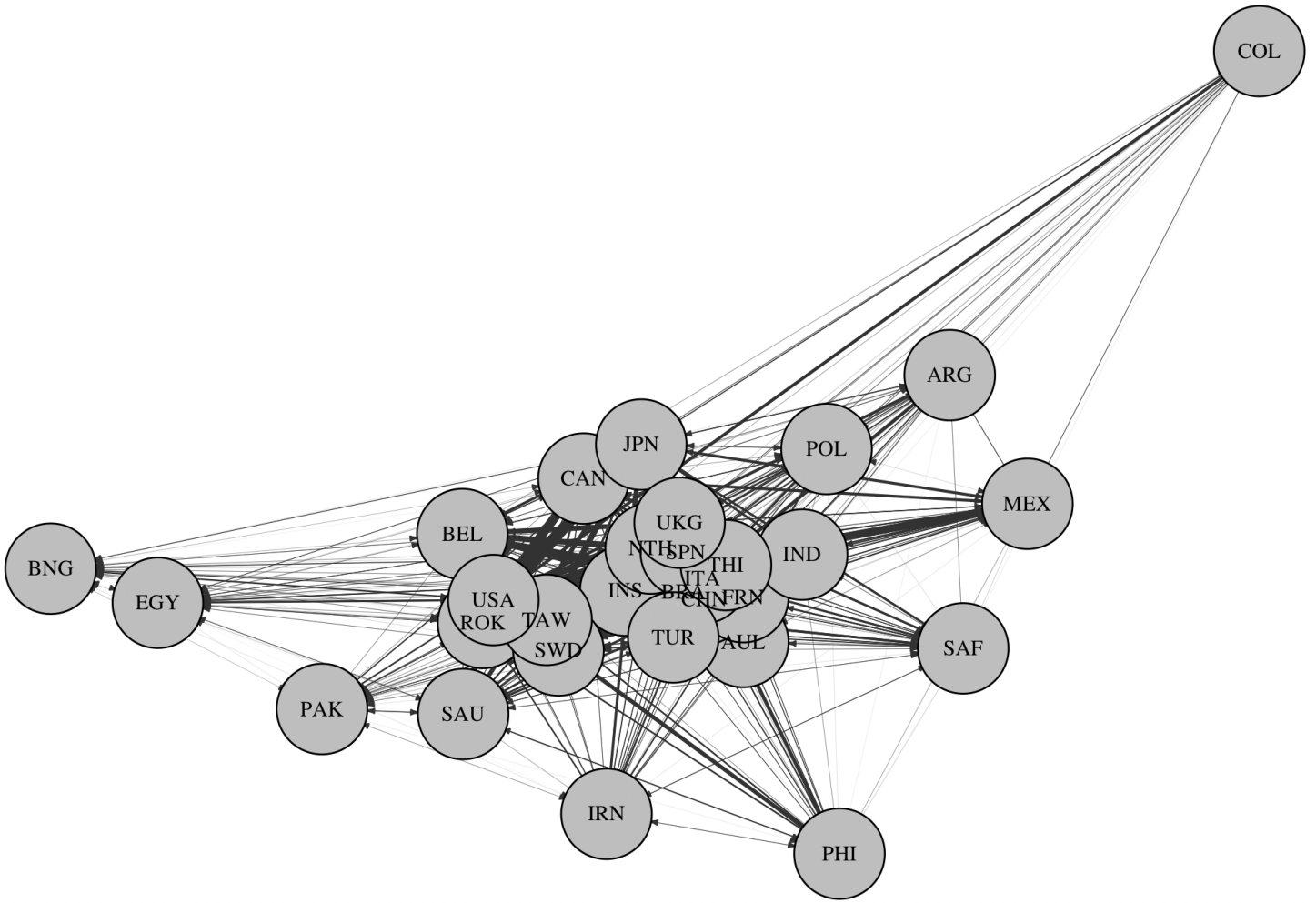


Lets jitter

Lets jitter the points slightly (don't ever actually do this):

```
zPosJitter = zPos+matrix(rnorm(length(zPos),0,.02),ncol=2)
plot(yGraph,
  layout=zPosJitter,
  vertex.color='grey',
  vertex.label.color='black',
  vertex.size=V(yGraph)$size,
  vertex.label.cex =.75,
  edge.color='grey20',
  edge.width=E(yGraph)$weight,
  edge.arrow.size=.2,
  asp=FALSE
  )
```

Lets jitter


COL
ARG
JPN
POL
CAN
MEX
BEL
UKG
NTH
SPN
IND
THI
BNG
ITA
INS
FRN
EGY
USA
ROK
TAW
TUR
SAF
SWD
PAK
SAU
IRN
PHI