Chapter 6
Statistical Models for Network Graphs

Statistical Analysis of Network Data, with R - Eric D. Kolaczyk

Thu Nguyen

02 July, 2019

Contents

Libraries

```
library(igraph)
library(igraphdata)
library(sand)
```

---

1 Introduction

- **Exponential Random Graph models** ~ standard regression models, in particular *generalized linear models*
- **Stochastic Block models** ~ mixture models, of *classical random graph models*
- **Latent Network models** ~ netwrok-based variant: using *observed* + *unobserved* variables in modeling

---

2 Exponential Random Graph Models

Exponential Random Graphs Models (ERGMs)
$\sim$ Classical Generalized Linear Models (GLMs)

2.1 General Formulation

Let $G = (V, E)$ be random, let $Y_{ij} = Y_{ji} = \mathbb{1}_{\{i,j\} \in E}$, then $\mathbf{Y} = [Y_{ij}]$ is the random adjacency matrix. Let $\mathbf{y} = [y_{ij}]$ be a particular graph, then:

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

- H: configuration, defined to be a set of poosible edges among a subset of vertices in G
- $g_H(\mathbf{y}) = \prod_{y_{ij} \in H}\ y_{ij}$: an indicator of H: 1 if H occurs in $\mathbf{y}$, and 0 otherwise
- $\theta_H \neq 0 \iff Y_{ij}$ are dependent for all pairs of vertices in H
- κ: normalization constant

Recall that a random vector $\mathbf{Z}$ belongs to an *exponential family* if the pdf is:

$$P_\theta\ (\mathbf{Z} = \mathbf{z}) = \exp\left\{ \theta^T\ \mathbf{g}(\mathbf{z}) - \psi(\theta) \right\}$$

where $\theta : p \times 1$ vector of parameters, $\mathbf{g}(\cdot) : p$ dimensional function of $\mathbf{z}$, and $\psi(\theta)$: normalization constant.

To draw *ERGMs* in R, use ergm package, part of the statnet package suite.

```
data(lazega)
A <- get.adjacency(lazega)                              # 1: create adjacency matrix
v.attrs <- get.data.frame(lazega, what = 'vertices')    # 2: arrange into data frame
library(ergm)
lazega.s <- network::as.network(as.matrix(A),           # 3: network object for ergm
                                directed = FALSE)
network::set.vertex.attribute(lazega.s, "Office", v.attrs$Office)
network::set.vertex.attribute(lazega.s, "Practice", v.attrs$Practice)
network::set.vertex.attribute(lazega.s, "Gender", v.attrs$Gender)
network::set.vertex.attribute(lazega.s, "Seniority", v.attrs$Seniority)
```

2.2 Specifying a Model

Given a general formulation of *ERGM*, we can introduce some assumptions to get different models:

1. *Bernoulli random graph*: the probability of any edge between every pair of vertices is **iid**, giving

$$P_\theta\ (\mathbf{Y} = \mathbf{y}) = \frac{1}{\kappa}\ \exp\left\{\sum_{i,j}\ \theta_{ij}\ y_{ij}\right\}$$

2. *Homogeneity*: given the Bernoulli assumption, let $\theta_{ij} = c$ for all i, j, where c is some constant, giving

$$P_\theta\ (\mathbf{Y} = \mathbf{y}) = \frac{1}{\kappa}\ \exp\Big\{\theta\ L(\mathbf{y})\Big\}$$

   where $L(\mathbf{y}) = |E_G|$. Note that this is equivalent to a *Bernoulli random graph* with $p = \frac{\exp(\theta)}{1+\exp(\theta)}$.

In `R`: to specify the model, use `formula()`:

```
my.ergm.bern <- formula(lazega.s ~ edges)
summary(my.ergm.bern)
```

```
## edges
##   115
```

Furthermore, suppose that we want to incorporate statistics of higher-order global network structure such as k-stars $S_k(\mathbf{y})$, and triangles $T(\mathbf{y})$, ... In `R`: in `formula()`, specify `kstar()` and `triangle`:

```
my.ergm <- formula(lazega.s ~ edges + kstar(2) + kstar(3) + triangle)
summary(my.ergm)
```

```
##    edges   kstar2   kstar3 triangle
##      115      926     2681      120
```

Additional statistics such as *Alternating k-star statistic* `altkstar()`, *Geometrically weighted degree count* `gwdegree()`, and *generalization of Triadic structures* `gwesp()`:

```
my.ergm <- formula(lazega.s ~ edges + altkstar(1, fixed = TRUE))
ergm.altkstar <- summary(my.ergm)[2]
my.ergm <- formula(lazega.s ~ edges + gwdegree(1, fixed = TRUE))
ergm.gwdegree <- summary(my.ergm)[2]
my.ergm <- formula(lazega.s ~ edges + gwesp(1, fixed = TRUE))
ergm.gwesp <- summary(my.ergm)[2]
data.frame(Statistics = c('Alternating k-star', 'Geom. weighted Degree', 'Triadic Structure'),
           Values = c(ergm.altkstar, ergm.gwdegree, ergm.gwesp))
```

```
##                          Statistics   Values
## altkstar.1       Alternating k-star 196.0000
## gwdeg.fixed.1 Geom. weighted Degree  79.2000
## gwesp.fixed.1     Triadic Structure 213.1753
```

To measure the total similarity among the vertices in a network, we can look at the statistics:

$$g(\mathbf{y}, \mathbf{x}) = \sum_{1 \le i \le j \le n} y_{ij} \; h(\mathbf{x}_i, \mathbf{x}_j)$$

where h is a symmetric function of choice, and $\mathbf{x}_i$ is a vector of observed attributes:

1. *Main effects*: $h(x_i, x_j) = x_i + x_j$, in 'R': `nodemain()`
2. *Second-order/Homophily effects*: $h(x_i, x_j) = \mathbb{1}_{x_i = x_j}$, in 'R': `nodematch()`

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

2.3 Model Fitting

In general, given *iid* realizations, *ERGMs* are fit using *Maximum Likelihood Estimator*. In R, `ergm()`:

```
set.seed(42)
lazega.ergm.fit <- ergm(lazega.ergm)
anova(lazega.ergm.fit)
```

```
summary(lazega.ergm.fit)
```

```
##
## ==========================
## Summary of model fit
## ==========================
##
## Formula:   lazega.s ~ edges + gwesp(log(3), fixed = TRUE) + nodemain("Seniority") +
##     nodemain("Practice") + match("Practice") + match("Gender") +
##     match("Office")
##
## Iterations:  2 out of 20
##
## Monte Carlo MLE Results:
##                              Estimate Std. Error MCMC % z value Pr(>|z|)
## edges                        -7.00655    0.67114      0 -10.440 < 0.0001 ***
## gwesp.fixed.1.09861228866811  0.59166    0.08554      0   6.917 < 0.0001 ***
## nodecov.Seniority             0.02456    0.00620      0   3.962 < 0.0001 ***
## nodecov.Practice              0.39455    0.10218      0   3.861 0.000113 ***
## nodematch.Practice            0.76966    0.19060      0   4.038 < 0.0001 ***
## nodematch.Gender              0.73767    0.24362      0   3.028 0.002463 **
## nodematch.Office              1.16439    0.18753      0   6.209 < 0.0001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 873.4  on 630  degrees of freedom
##  Residual Deviance: 459.6  on 623  degrees of freedom
##
## AIC: 473.6    BIC: 504.7    (Smaller is better.)
```

Interpretation:

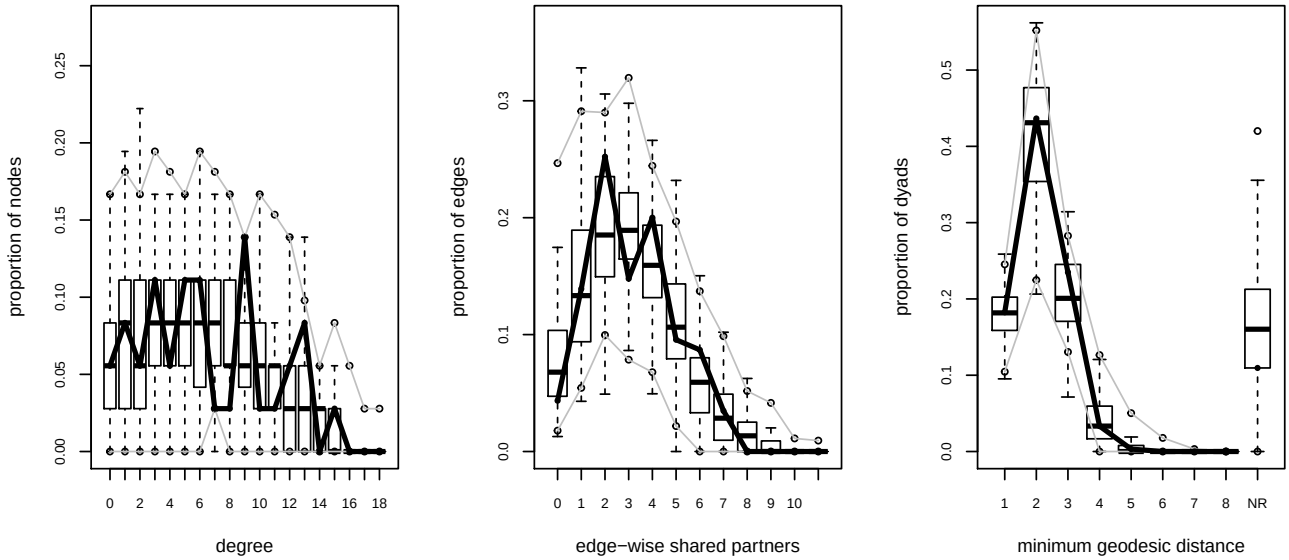
- practicing corporate law, not litigation, increases cooperation by $\exp(.39455) \approx 1.48$, or nearly 50%
- being of the same gender more than doubles the odds: $\exp(.73767) = 2.09$.

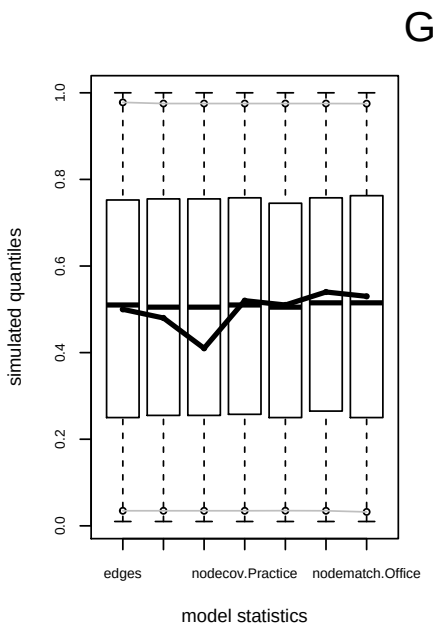
2.4 Goodness-of-Fit of model

To access GOF as fit by `ergm`, in `R`: `gof()` runs the Monte Carlo simulation:

```
gof.lazega.ergm <- gof(lazega.ergm.fit)
```

```
par(mfrow=c(1,3))
plot(gof.lazega.ergm)
```





3 Network Block Models

Network Block Models $\sim$ Classical Mixture Models

3.1 Model Specification

Let $G = (V, E)$, with adjacency matrix $\mathbf{Y} = [Y_{ij}]$. Suppose each vertex $i \in V_G$ belongs to 1 of Q classes $\mathcal{C}_1, \ldots, \mathcal{C}_Q$, and the class label is known: $q = q(i), \forall i$. Conditioned on class labels q, r or vertices i, j, a *block model* is such that each Y_{ij} is *iid* **Bernoulli** with probability π_{ij}. For undirected graph: $\pi_{ij} = \pi_{ji}$, giving

$$P_\theta\ (\mathbf{Y} = \mathbf{y}) = \frac{1}{\kappa}\ \exp\left\{ \sum_{q,r}\ \theta_{qr}\ L_{qr}(\mathbf{y}) \right\}$$

If given Q classes but *unknown* class labels, the model becomes *Stochastic Block Model (SBM)*.

3.2 Model Fitting

In a *non-stochastic block model*, the edge probabilities π_{qr} are estimated using *Maximum Likilihood Estimates*. In R: `mixer()` from the `mixer` package, to specify $\min(Q)$ `qmin`, $\max(Q)$ `qmax`:

```
library(mixer)
set.seed(42)
fblog.sbm <- mixer(as.matrix(get.adjacency(fblog)), qmin = 2, qmax = 15)
```

```
## Mixer: the adjacency matrix has been transformed in a undirected edge list
```

```
fblog.sbm.output <- getModel(fblog.sbm)
names(fblog.sbm.output)
```

```
## [1] "q"         "criterion" "alphas"    "Pis"       "Taus"
```

The criterion above is *Integration Classification Likelihood (ICL)*, similar to AIC and BIC.

```
print(paste0('Fitted model: q = ', fblog.sbm.output$q))
```

```
## [1] "Fitted model: q = 12"
```

```
print(cat('Estimated proportions:', '\n', round(fblog.sbm.output$alphas, 3), '\n'))
```

```
## Estimated proportions:
##  0.151 0.127 0.108 0.057 0.136 0.031 0.124 0.093 0.01 0.021 0.125 0.016
## NULL
```

Thus, *Stochastic Block models* can be used for *graph partitioning*.

```
round(fblog.sbm.output$Taus[, 1:3],6)
```

```
##           [,1]     [,2]     [,3]
##  [1,] 0.999975 0.004911 0.999981
##  [2,] 0.000014 0.000000 0.000001
##  [3,] 0.000010 0.994221 0.000018
##  [4,] 0.000000 0.000000 0.000000
##  [5,] 0.000001 0.000000 0.000000
##  [6,] 0.000000 0.000000 0.000000
##  [7,] 0.000000 0.000868 0.000000
##  [8,] 0.000000 0.000000 0.000000
##  [9,] 0.000000 0.000000 0.000000
## [10,] 0.000000 0.000000 0.000000
## [11,] 0.000000 0.000000 0.000000
## [12,] 0.000000 0.000000 0.000000
```

Interpretation:

- $P(v_1 \in \mathcal{C}_1) = 0.9999747$, or the model labels vertex 1 in class 1
- $P(v_2 \in \mathcal{C}_3) = 0.9942208$, and so on

Entropy of a discrete pmf $\mathbf{p} = (p_1, \ldots, p_Q)$: $H(\mathbf{p}) := -\sum_{q=1}^{Q} p_q \log_2(p_q)$: smaller $H(\mathbf{p})$ indicates distribution is concentrated on fewer classes.

```
my.ent <- function(x) { -sum(x * log(x, 2)) }        # fn to calculate entropy
apply(fblog.sbm.output$Taus[, 1:3], 2, my.ent)
```

```
## [1] 0.0004559176 0.0548083843 0.0003231649
```

```
print(paste('Entropy for if classes are Uniformly dist.:', log(fblog.sbm.output$q, 2)))
```

```
## [1] "Entropy for if classes are Uniformly dist.: 3.58496250072116"
```

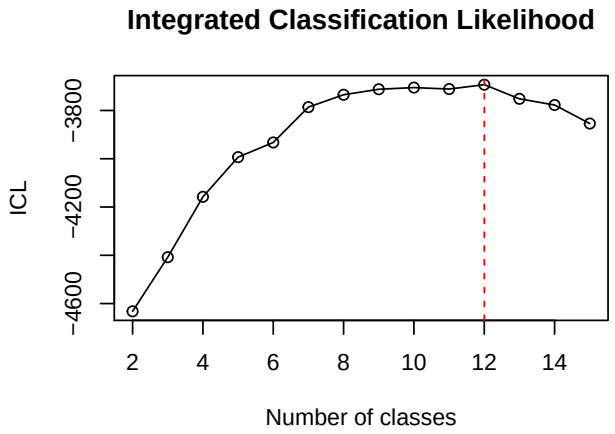
Interpretation: small entropy values are consistent with the `fblog` network: vertices are concentrated on few classes.

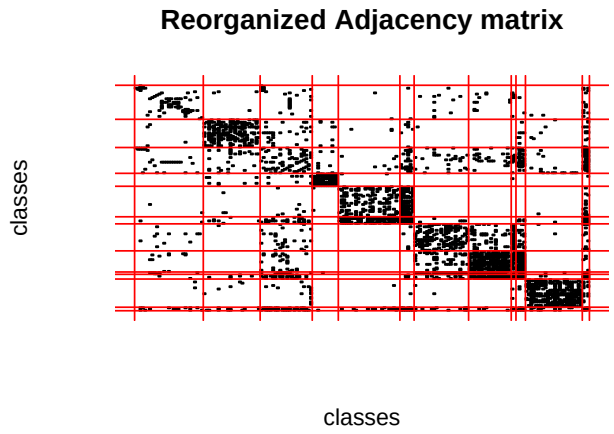
```
summary(apply(fblog.sbm.output$Taus, 2, my.ent))
```

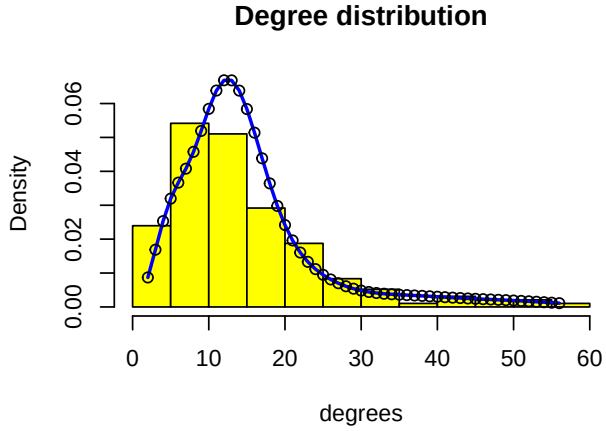
```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000000 0.0000000 0.0000006 0.0414403 0.0029971 1.4130217
```

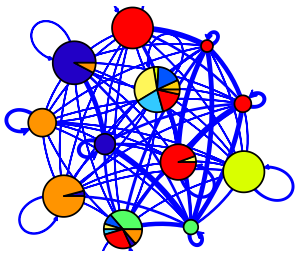
3.3 Goodness-of-Fit

```
plot(fblog.sbm, classes=as.factor(V(fblog)$PolParty))
```









Interpretation:

1. while $Q = 12$ gives $\max(ICL)$, $Q \in \{8, 9, 10, 11, 12\}$ are also reasonable choices
2. from adjacency matrix $\mathbf{Y}$, there are 5 smaller classes and 7 larger classes, whose vertices tend to primarily connect within classes, and with vertices from only some certain other classes
3. degree distribution: blue curve: fitted SBN, vs. yellow: observed distribution.

---

4 Latent Network Models

4.1 General Formulation

Given the absence of any co-variate informtion, it is natural to assume exchangeability of vertices in $G = (V, E)$, with adjacency matrix $\mathbf{Y} = [Y_{ij}]$, giving

$$Y_{ij} = h(\propto, u_i, u_j, \epsilon_{ij}) \tag{1}$$

where $\propto$: constant, u_i: *iid* latent variables, and ϵ_{ij}: *iid* pair-specific effects, and h: symmetric wrt u_i, u_j. An example is *probit model*:

$$\mathbb{P}\big(\ \mathbf{Y}_{ij} = 1 \mid \mathbf{X}_{ij} = \mathbf{x}_{ij}\ \big) = \phi\big(\ \propto + \mathbf{x}_{ij}^T\ \beta + \alpha(u_i, u_j)\ \big) \tag{2}$$

where ϕ is CMF of $\mathbf{Z} \sim \mathcal{N}(0, 1)$. Let $p_{ij} = (2)$, the conditional model for $\mathbf{Y}$ is

$$\mathbb{P}\big(\ \mathbf{Y} = \mathbf{y} \mid \mathbf{X}, u_1, \ldots, u_n\ \big) = \prod_{i<j}\ p_{ij}^{y_{ij}}\ (1 - p_{ij})^{1-y_{ij}}$$

4.2 Specifying the Latent Effects

From (1), the function $\alpha(\cdot, \cdot)$ dictates the effects of the latent variables, in particular:

1. *Latent class models:* analogous to *Stochastic block models* above: $u_i \in \{1, \ldots, Q\}, \alpha(u_i, u_j) = m_{u_i u_j}$ symmetrically
2. *Latent distance model:* under the *principle of homophily*: vertices with more similar characteristics tend to establish an edge, $\alpha(u_i, u_j) = -|u_i - u_j|$ for some distance metric
3. *Eigenmodel:* under the *principles of eigen-analysis*: $\alpha(u_i, u_j) = a_i^T \Lambda u_j$, where u_i: Q-length random vectors, and Λ: $Q \times Q$ diagonal matrix;
    - note that if $\mathbf{U} = [u_1, \ldots, u_Q]$, then $\mathbf{U}\Lambda\mathbf{U}^T$ is analogous to eigen-decomposition of all pairwise latent effects $\alpha(u_i, u_j)$
    - *Eigenmodels* can be thought of as a generalization of both *Latent class* and *Latent distance* models.

4.3 Model Fitting

Given the `lazega` dataset, it is natural to hypothesize that collaboration is driven by:

- similarity of practice $\sim$ a form of *homophily*, or
- similarity of office location $\sim$ a proxy for *distance*

As such, we can compare 3 fitted models with different settings:

1. no pair-specific covariates
2. a covariate for common practice
3. a covariate for shared office location

In R: `eigenmodel` package runs the Monte Carlo Markov Chain simulation to obtain the posterior distributions from the conjugate priors (*Bayesian* approach).

Model 1: no pair-specific covariates:

```
library(eigenmodel)
set.seed(42)
A <- get.adjacency(lazega, sparse = FALSE)
lazega.leig.fit1 <- eigenmodel_mcmc(A, R = 2, S = 11000, burn = 10000)
```

Model 2: a covariate for common practice:

```
# Common practice effects
same.prac.op <- v.attr.lazega$Practice %o% v.attr.lazega$Practice
same.prac <- matrix(as.numeric(same.prac.op %in% c(1,4,9)), 36, 36)
same.prac <- array(same.prac, dim = c(36, 36, 1))
# Fit model
lazega.leig.fit2 <- eigenmodel_mcmc(A, same.prac, R = 2, S = 11000, burn = 10000)
```

Model 3: a covariate for shared office location

```
# Common office effects
same.off.op <- v.attr.lazega$Office %o% v.attr.lazega$Office
same.off <- matrix(as.numeric(same.off.op %in% c(1,4,9)), 36, 36)
same.off <- array(same.off, dim = c(36, 36, 1))
# Fit model
lazega.leig.fit3 <- eigenmodel_mcmc(A, same.off, R = 2, S = 11000, burn = 10000)
```

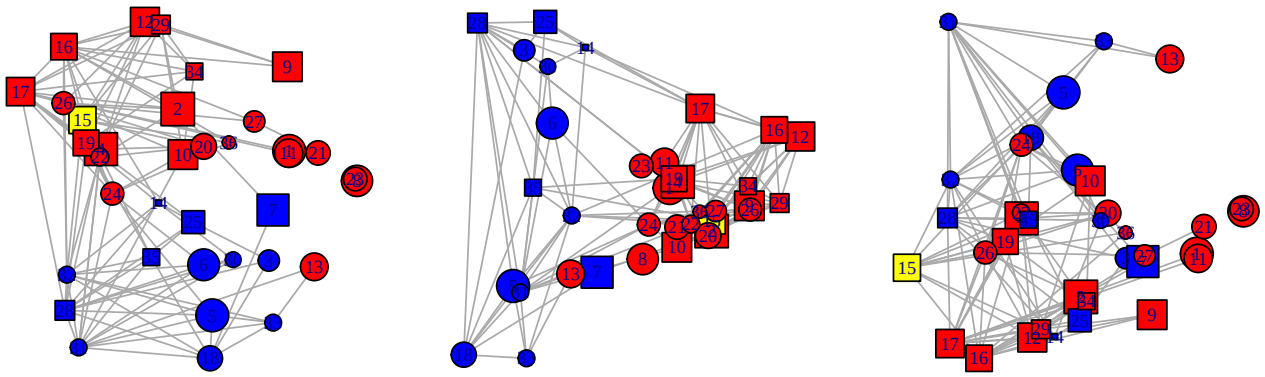
To compare the representation, we extract and plot the eigenvectors for each model, using `eigen()`:

```
lat.sp.1 <- eigen(lazega.leig.fit1$ULU_postmean)$vec[, 1:2]
lat.sp.2 <- eigen(lazega.leig.fit2$ULU_postmean)$vec[, 1:2]
lat.sp.3 <- eigen(lazega.leig.fit3$ULU_postmean)$vec[, 1:2]
```

```
par(mfrow=c(1,3)); par(mar=c(0,1,1,1))
v.colors <- c('red', 'blue', 'yellow')[V(lazega)$Office]
v.shapes <- c('circle', 'square')[V(lazega)$Practice]
v.size <- 3.5*sqrt(V(lazega)$Years)
v.label <- V(lazega)$Seniority
plot(lazega, layout = lat.sp.1, vertex.color = v.colors, vertex.shape = v.shapes,
     vertex.size = v.size, vertex.label = v.label, main = 'Model 1: No covariates')
```

```
plot(lazega, layout = lat.sp.2, vertex.color = v.colors, vertex.shape = v.shapes,
     vertex.size = v.size, vertex.label = v.label, main = 'Model 2: Common practice')
plot(lazega, layout = lat.sp.3, vertex.color = v.colors, vertex.shape = v.shapes,
     vertex.size = v.size, vertex.label = v.label, main = 'Model 3: Shared office')
```

Model 1: No covariates **Model 2: Common practice** **Model 3: Shared office**



Interpretation:

- Models 1 and 2: lawyers are clustered into 2 main groups based on office location
- Model 3: common practice appears to not well distinguish the network structure

```
lambda1 <- apply(lazega.leig.fit1$L_postsamp, 2, mean)
lambda2 <- apply(lazega.leig.fit2$L_postsamp, 2, mean)
lambda3 <- apply(lazega.leig.fit3$L_postsamp, 2, mean)
data.frame(Models = c('No covariate', 'Common practice', 'Common office'),
           lambda.1 = c(lambda1[1], lambda2[1], lambda3[1]),
           lambda.2 = c(lambda1[2], lambda2[2], lambda3[2]))
```

```
##            Models  lambda.1   lambda.2
## 1    No covariate 0.2876963  0.9709850
## 2 Common practice 0.9085345 -0.1531384
## 3   Common office 0.4031347  0.1892327
```

The table is consistent with the plot: in Model 1 and 2, there is 1 value eigenvalue λ that dominates, while in model 3, the 2 eigenvalues do not differ much.

4.4 Goodness-of-Fit

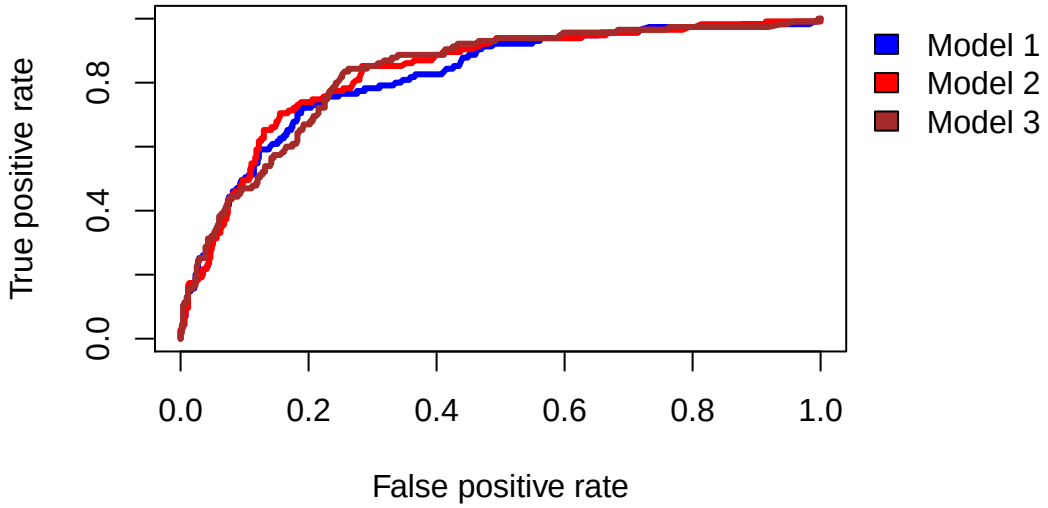
Here, we use cross-validation, with $k = 5$, to access GOF for the 3 different models.

```
perm.index <- sample(1:630)
nfolds <- 5
nmiss <- 630/nfolds
Avec <- A[lower.tri(A)]
Avec.pred1 <- numeric(length(Avec))
Avec.pred2 <- numeric(length(Avec))
Avec.pred3 <- numeric(length(Avec))

for (i in seq(1, nfolds)) {
  # Index of missin values
  miss.index <- seq(((i-1)*nmiss + 1), i*nmiss, 1)
  A.miss.index <- perm.index[miss.index]

  # Fill a new Atemp with NAs
  Avec.temp <- Avec
  Avec.temp[A.miss.index] <- rep('NA', length(A.miss.index))
  Avec.temp <- as.numeric(Avec.temp)
  Atemp <- matrix(0, 36, 36)
  Atemp[lower.tri(Atemp)] <- Avec.temp
  Atemp <- Atemp + t(Atemp)

  # Fit model and predict, model 1
  Y <- Atemp
  model1.fit <- eigenmodel_mcmc(Y, R = 2, S = 11000, burn = 10000)
  model1.pred <- model1.fit$Y_postmean
  model1.pred.vec <- model1.pred[lower.tri(model1.pred)]
  Avec.pred1[A.miss.index] <- model1.pred.vec[A.miss.index]

  # Fit model and predict, model 2
  model2.fit <- eigenmodel_mcmc(Y, same.prac, R = 2, S = 11000, burn = 10000)
  model2.pred <- model2.fit$Y_postmean
  model2.pred.vec <- model2.pred[lower.tri(model2.pred)]
  Avec.pred2[A.miss.index] <- model2.pred.vec[A.miss.index]

  # Fit model and predict, model 3
  model3.fit <- eigenmodel_mcmc(Y, same.off, R = 2, S = 11000, burn = 10000)
  model3.pred <- model3.fit$Y_postmean
  model3.pred.vec <- model3.pred[lower.tri(model3.pred)]
  Avec.pred3[A.miss.index] <- model3.pred.vec[A.miss.index]
}
```

Similar to *Classification* problem, 1 way to evaluate the fitted model is through the ROC curve and the AUC percentage from the `ROCR` package: `prediction()` and `performance()`:

```
par(mar=c(5,5,1,2)); par(oma=c(0, 0, 0, 5))
library(ROCR)
pred1 <- prediction(Avec.pred1, Avec)
perf1 <- performance(pred1, 'tpr', 'fpr')
pred2 <- prediction(Avec.pred2, Avec)
perf2 <- performance(pred2, 'tpr', 'fpr')
pred3 <- prediction(Avec.pred3, Avec)
perf3 <- performance(pred3, 'tpr', 'fpr')
```

```r
plot(perf1, col = 'blue', lwd = 3, legend = 'Model 1')
plot(perf2, col = 'red', lwd = 3, add = TRUE, legend = 'Model 2')
plot(perf3, col = 'brown', lwd = 3, add = TRUE)
legend(par('usr')[2], par('usr')[4], xpd=NA, bty = 'n',
       legend = c('Model 1', 'Model 2', 'Model 3'),
       col = c('blue', 'red', 'brown'), c('blue', 'red', 'brown'))
```




```r
auc.mod1 <- slot(performance(pred1, 'auc'), 'y.values')[[1]]
auc.mod2 <- slot(performance(pred2, 'auc'), 'y.values')[[1]]
auc.mod3 <- slot(performance(pred3, 'auc'), 'y.values')[[1]]
data.frame(Models = c('No covariate', 'Common practice', 'Common office'),
           AUC = c(auc.mod1, auc.mod2, auc.mod3))
```

```
##            Models       AUC
## 1    No covariate 0.8172309
## 2 Common practice 0.8324187
## 3   Common office 0.8284255
```

Comment: all models appear to be comparable in their performance and to perform well: AUC of over 80%.
