Advanced Network Analysis

Inferential Network Modeling with the Social Relations Model

Shahryar Minhas [s7minhas.com]

Goals for today

- Basics of network modeling
- Social Relations Model

Goals of statistical network modeling

- How do features drive our data analysis?
  - How can we describe features of social relations?
  - How can we identify nodes with similar network roles?
  - How do we relate the network to covariate information?

What we want to account for when modeling

Many networks exhibit the following features:

- Homophily by actor attributes
  - Higher propensity to form ties between actors with similar attributes
- Degree heterogeneity among actors
  - Sociability, Popularity
- Reciprocity of ties
- Higher order dependencies
  - We'll start to get to this next week

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 is 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., $Pr(Y = y)$

Inferential Goals in the Regression Framework

y_{ij} measures $i \rightarrow j$, 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:

$y_{ij} \approx \beta^T x_{ij} + \epsilon_{ij}$

What do we want this model to provide?:

- A measure of 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: $g(\tilde{Y})$, $\tilde{Y} \approx Pr(\tilde{Y}|Y, X)$

What's Wrong with GLM?

GLM: $y_{ij} \sim \beta^T X_{ij} + e_{ij}$

Networks typically show evidence against independence of $\{e_{ij} : i \neq j\}$

Not accounting for dependence can lead to:

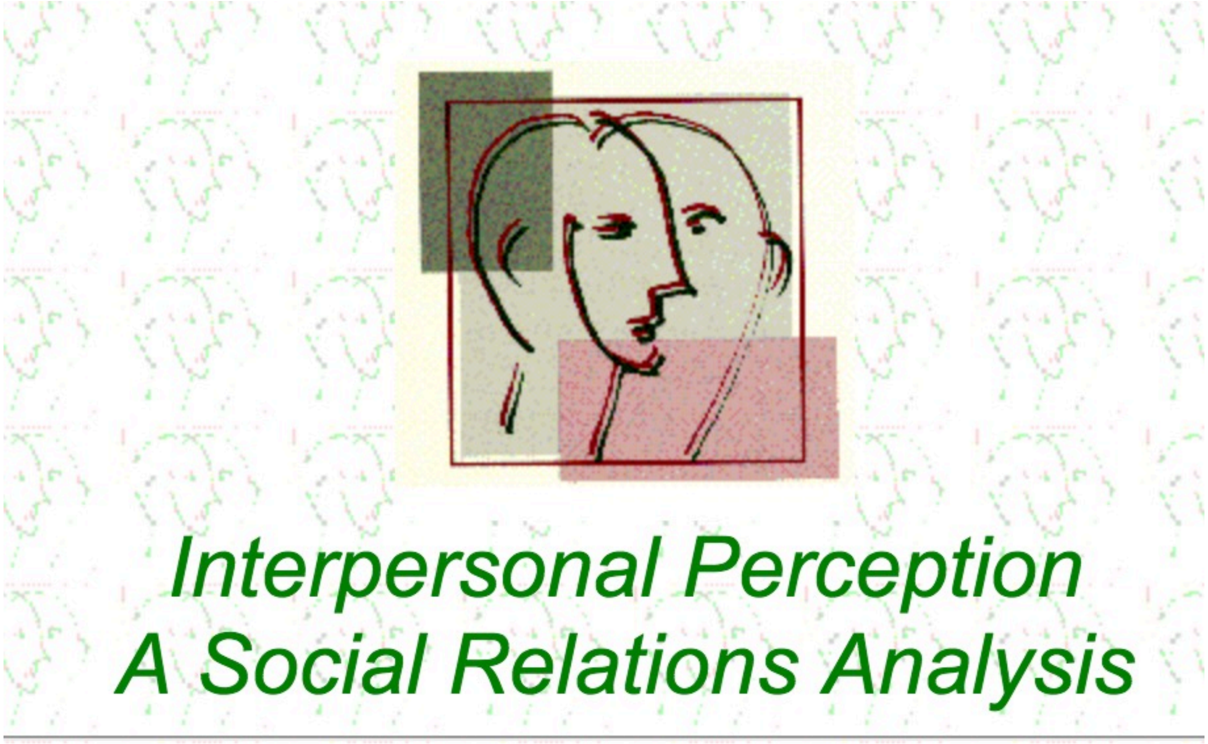
- biased effects estimation
- uncalibrated confidence intervals
- poor predictive performance
- inaccurate description of the event of interest

We've been hearing this concern for decades now:

Thompson & Walker (1982)	Beck et al. (1998)	Snijders (2011)
Frank & Strauss (1986)	Signorino (1999)	Erikson et al. (2014)
Kenny (1996)	Li & Loken (2002)	Aronow et al. (2015)
Krackhardt (1998)	Hoff and Ward (2004)	Minhas et al. (2018)

Social Relations Model

- David Kenny was interested "systematically studying what we think others are like, how we see ourselves and how we think others see us"
- i.e., interpersonal perceptions (ugly cover, good book)



Sender heterogeneity

An actor can induce dependence across its "recievers." Thus values across a row, say $\{y_{ij}, y_{ik}, y_{il}\}$, may be more similar to each other than other values in the adjacency matrix because each of these values has a common sender i.

	i	j	k	l
i	NA	y_{ij}	y_{ik}	y_{il}
j	y_{ji}	NA	y_{jk}	y_{jl}
k	y_{ki}	y_{kj}	NA	y_{kl}
l	y_{li}	y_{lj}	y_{lk}	NA

Receiver heterogeneity

Additionally, values across a column, say $\{y_{ji}, y_{ki}, y_{li}\}$, may be more similar to each other than other values in the adjacency matrix because each of these values has a common receiver i.

	i	j	k	l
i	NA	y_{ij}	y_{ik}	y_{il}
j	y_{ji}	NA	y_{jk}	y_{jl}
k	y_{ki}	y_{kj}	NA	y_{kl}
l	y_{li}	y_{lj}	y_{lk}	NA

Sender-Receiver Covariance

Actors who are more likely to send ties in a network may also be more likely to receive them.

	i	j	k	l
i	NA	y_{ij}	y_{ik}	y_{il}
j	y_{ji}	NA	y_{jk}	y_{jl}
k	y_{ki}	y_{kj}	NA	y_{kl}
l	y_{li}	y_{lj}	y_{lk}	NA

Reciprocity

Values of y_{ij} and y_{ji} may be statistically dependent. Dyads might exhibit high reciprocity because there is a tendency for actors to treat each other similarily, i.e., "respond in kind" to these behaviors.

	i	j	k	l
i	NA	y_{ij}	y_{ik}	y_{il}
j	y_{ji}	NA	y_{jk}	y_{jl}
k	y_{ki}	y_{kj}	NA	y_{kl}
l	y_{li}	y_{lj}	y_{lk}	NA

Lets work through an example

```r
load('preezeObjects/trade_for_ols.rda')
trade[1:3,]
```

```
##   Var1 Var2 trade polity.row polity.col conflicts distance shared_igos
## 2  ARG  AUL 0.058       7.18         10         0    11.72       3.827
## 3  ARG  BEL 0.247       7.18         10         0    11.31       3.917
## 4  ARG  BNG 0.039       7.18          5         0    16.76       3.425
```

We want to fit the following linear model:

```r
form = formula(
  trade ~
  polity.row + polity.col + conflicts + distance + shared_igos
  )
form
```

```
## trade ~ polity.row + polity.col + conflicts + distance + shared_igos
```

Lets hypothesize first

$$
\begin{aligned}
trade_{ij} = & \\
& = \beta_0 + \\
& = \beta_1 \times \text{polity.row}_i + \\
& = \beta_2 \times \text{polity.row}_j + \\
& = \beta_3 \times \text{conflicts}_{ij} + \beta_4 \times \text{distance}_{ij} + \beta_5 \times \text{shared\_igos}_{ij} \\
& = \epsilon_{ij}
\end{aligned}
$$

Lets interpret each β parameter.

Lets estimate the model

```
ols = lm(form, data=trade)
summary(ols)$'coefficients'
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.069      0.172   0.399     0.69
## polity.row       0.022      0.005   4.617     0.00
## polity.col       0.027      0.005   5.759     0.00
## conflicts        0.518      0.091   5.666     0.00
## distance        -0.042      0.006  -7.088     0.00
## shared_igos      0.172      0.043   4.024     0.00
```

Lets go beyond stargazing

- And conduct some residual diagnostics with network dependencies in mind
- We'll need to reorganize the residuals first

```
# pull out errors from trade
trade$olsError = ols$residuals

# construct sociomatrix out of errors
actors = unique(c(trade$Var1, trade$Var2))
n = length(actors)
E = matrix(NA, nrow=n,ncol=n, dimnames=list(actors,actors))
for(ii in 1:nrow(trade)){
  E[trade$Var1[ii], trade$Var2[ii]] = trade$olsError[ii] }
E[1:3,1:3]
```

```
##            ARG        AUL        BEL
## ARG         NA -0.6127548 -0.4568836
## AUL -0.5698027         NA -0.1806427
## BEL -0.4186733 -0.2084223         NA
```

Beyond stargazing ... are errors patternless?

We see strong evidence of structure in the errors that can at least partly be explained by reciprocity

```
cor(c(E), c(t(E)), use='complete.obs')
```

```
## [1] 0.9109391
```

Beyond stargazing ... sender/receiver patterns?

Structure in the errors by sender and receiver heterogeneity:

```
rowErr = apply(E, 1, mean, na.rm=TRUE)
colErr = apply(E, 2, mean, na.rm=TRUE)

sort(rowErr)[1:3]
sort(colErr)[1:3]
```

```
##    ARG    AUL    BEL
## -0.212  0.025 -0.008

##    ARG    AUL    BEL
## -0.203 -0.023 -0.016
```

```
sd(rowErr)
```

```
## [1] 0.4654568
```

```
sd(colErr)
```

```
## [1] 0.4318027
```

Beyond stargazing ... sender/receiver patterns?

Lets visualize these patterns:

```
errDF = melt(E) ; errDF = na.omit(errDF)

errDF$Var1 = factor(errDF$Var1, levels=names(sort(rowErr)))
rowErrorGG = ggplot(errDF, aes(x=Var1,y=value)) +
  geom_boxplot() + geom_jitter(alpha=.3) +
  xlab('') + ylab('Residual by Sender') +
  theme(
    axis.ticks=element_blank(),
    panel.border=element_blank(),
    axis.text.x=element_text(angle=45,hjust=1)
    )
```

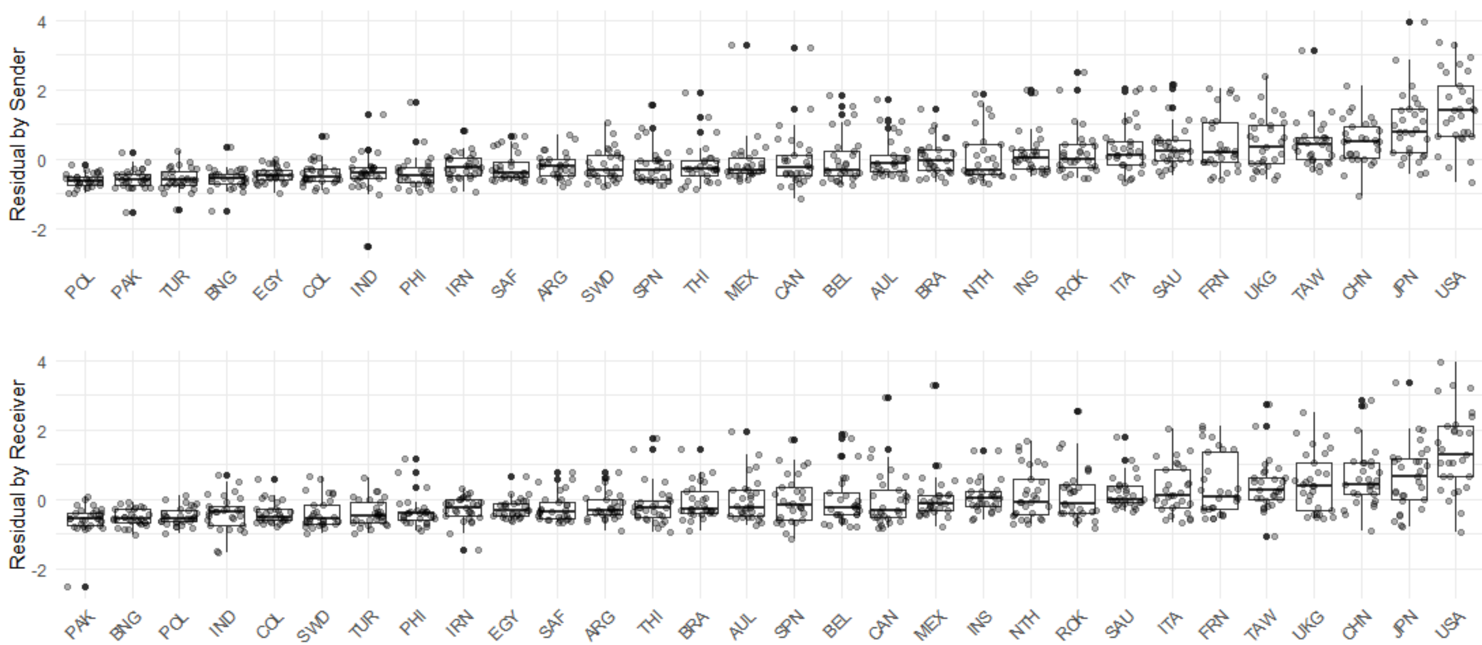
Beyond stargazing … sender/receiver patterns?

Ditto for receiver

```
errDF$Var2 = factor(errDF$Var2, levels=names(sort(colErr)))
colErrorGG = ggplot(errDF, aes(x=Var2,y=value)) +
  geom_boxplot() + geom_jitter(alpha=.3) +
  xlab('') + ylab('Residual by Receiver') +
  theme(
    axis.ticks=element_blank(),
    panel.border=element_blank(),
    axis.text.x=element_text(angle=45,hjust=1)
    )
```

Beyond stargazing ... sender/receiver patterns? YES

```
library(gridExtra)
grid.arrange(rowErrorGG, colErrorGG, nrow=2)
```



Is there always structure in residuals?

- Are the patterns that we've been discovering in the residuals here likely under independence?
- Or put another way what if we had specified the model correctly?

```
# set up some fake dyadic data
simData = expand.grid(letters,letters, stringsAsFactors = FALSE)
# remove i-i observations
simData = simData[simData$Var1 != simData$Var2,]

# add x1
set.seed(6886)
simData$x1 = rnorm(nrow(simData))
# add x2
simData$x2 = rnorm(nrow(simData))

# create a y
simData$y = -1 + 2*simData$x1 + .5*simData$x2 + rnorm(nrow(simData))

# run model
olsSim = lm(y ~ x1 + x2, data=simData)
```

Is there always structure in residuals?

Check for evidence of structure by reciprocity and sender/receiver effects.

What's the point?

GLM: $y_{ij} \sim \beta^T X_{ij} + e_{ij}$ ... when errors are not independent, we get

- biased effects estimation
- uncalibrated confidence intervals

Basically, our parameter estimates for β and particularly the confidence intervals will be wrong ... this is like stargazing on a cloudy day

What about model fit in the context of dependencies?

- So we likely need an approach that accounts for the structure in the residuals … but how do we know we got it right?
- Whenever we evaluate model fit we start by choosing some measures of fit
- In our case we are going to focus on the ability of the model to capture:
  - variation across rows means (out-degrees)
  - variation across column means (in-degrees)
  - correlation within dyads (i.e., reciprocity)

Model fit and dependencies

- Okay, so we have our measures, how do we want to evaluate how well a given model does
- Simulation!
    - We are going to simulate multiple set of predictions from our model
    - Calculate our measures of fit for each set of predictions from the model
    - And compare the simulated values against the observed data

Simulate predictions from a linear model

- We are going to simulate 1000 β for each parameter from our model
- Then multiply the simulated model estimates with the observed data, X
- And end with a matrix of predicted values, one column for each simulation

```
library(MASS)
```

```
betaDraws = MASS::mvrnorm(n = 1000, mu = coef(ols), vcov(ols))
xMatrix = data.matrix(cbind(1, trade[,names(coef(ols))[-1]]))
preds = xMatrix %*% t(betaDraws)
dim(preds)
```

```
## [1]  870 1000
```

Calculating model fit

Lets calculate model fit for one set of predictions. First we need to organize the data.

```
# extract one prediction from simulated model
trade$yhat = preds[,1]

# get actor vector
actors = sort(unique(c(trade$Var1, trade$Var2)))
n=length(actors)

# organize adjacency matrix
yhatMat = matrix(NA, nrow=n, ncol=n, dimnames=list(actors,actors))
for(ii in 1:nrow(trade)){
  yhatMat[trade$Var1[ii],trade$Var2[ii]] = trade$yhat[ii]
}
```

Before proceeding

Lets organize our dv into an adjacency matrix as well using the actors vector that we created

```
# organize dv into adjacency matrix
Y = matrix(NA, nrow=n, ncol=n, dimnames=list(actors,actors))
for(ii in 1:nrow(trade)){
  a1 = trade$Var1[ii]
  a2 = trade$Var2[ii]
  val = trade$trade[ii]
  Y[a1,a2] = val }
```

Calculating model fit

Now lets calculate fit:

- variation across rows means (out-degrees)

```
sd(apply(yhatMat, 1, mean, na.rm=TRUE))
```

```
## [1] 0.1279117
```

```
sd(apply(Y, 1, mean, na.rm=TRUE)) # Y is a sociomatrix of trade$trad
```

```
## [1] 0.4906692
```

Calculating model fit

Now lets calculate fit:

- variation across column means (in-degrees)

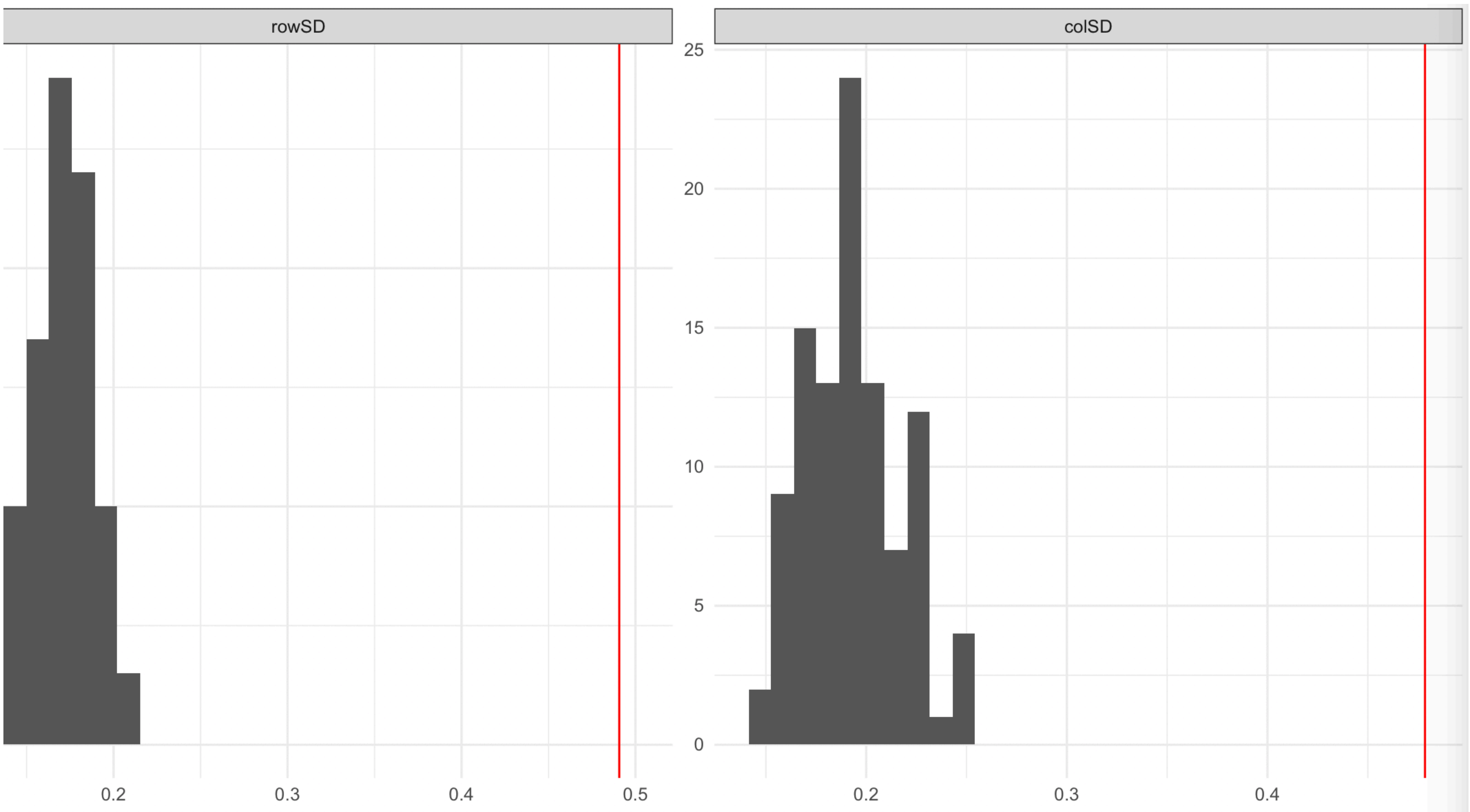
```
sd(apply(yhatMat, 2, mean, na.rm=TRUE))
```

```
## [1] 0.1860768
```

```
sd(apply(Y, 2, mean, na.rm=TRUE))
```

```
## [1] 0.4784861
```

Now lets do this for all predictions



What's the point? II

GLM: $y_{ij} \sim \beta^T X_{ij} + e_{ij}$ ... when errors are not independent, we get

- poor predictive performance
- inaccurate description of the event of interest

Basically, our models can't actually reproduce the observed data

Lets account for the structure.

Conceptual question ...

Why are the errors not independent?

Social Relations Model

- This brings us to the following model (Warner et al. 1979; Li & Loken 2002):

$$
\begin{aligned}
y_{ij} &= \mu + e_{ij} \\
e_{ij} &= a_i + b_j + \epsilon_{ij} \\
\{(a_1, b_1), \ldots, (a_n, b_n)\} &\sim N(0, \Sigma_{ab}) \\
\{(\epsilon_{ij}, \epsilon_{ji}) :\ i \neq j\} &\sim N(0, \Sigma_\epsilon),\ \text{where}
\end{aligned}
$$

$$
\Sigma_{ab} = \begin{pmatrix} \sigma_a^2 & \sigma_{ab} \\ \sigma_{ab} & \sigma_b^2 \end{pmatrix} \qquad \Sigma_\epsilon = \sigma_\epsilon^2 \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}
$$

- μ baseline measure of network activity (for the purpose of regression we turn this into $\beta^T X$)
- e_{ij} residual variation that we will use the SRM to decompose

Social Relations Model: Nodal Effects

$$
\begin{aligned}
y_{ij} &= \mu + e_{ij} \\
e_{ij} &= a_i + b_j + \epsilon_{ij} \\
\{(a_1, b_1), \ldots, (a_n, b_n)\} &\sim N(0, \Sigma_{ab}) \\
\{(\epsilon_{ij}, \epsilon_{ji}) : \ i \neq j\} &\sim N(0, \Sigma_\epsilon), \ \text{where}
\end{aligned}
$$

$$
\Sigma_{ab} = \begin{pmatrix} \sigma_a^2 & \sigma_{ab} \\ \sigma_{ab} & \sigma_b^2 \end{pmatrix} \qquad \Sigma_\epsilon = \sigma_\epsilon^2 \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}
$$

- row/sender effect (a_i) & column/receiver effect (b_j)
- Modeled jointly to account for correlation in how active an actor is in sending and receiving ties

Social Relations Model: Nodal Variance

$$
\begin{aligned}
y_{ij} &= \mu + e_{ij} \\
e_{ij} &= a_i + b_j + \epsilon_{ij} \\
\{(a_1, b_1), \ldots, (a_n, b_n)\} &\sim N(0, \Sigma_{ab}) \\
\{(\epsilon_{ij}, \epsilon_{ji}) : \; i \neq j\} &\sim N(0, \Sigma_\epsilon), \; \text{where}
\end{aligned}
$$

$$
\Sigma_{ab} = \begin{pmatrix} \sigma_a^2 & \sigma_{ab} \\ \sigma_{ab} & \sigma_b^2 \end{pmatrix} \qquad \Sigma_\epsilon = \sigma_\epsilon^2 \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}
$$

- σ_a^2 and σ_b^2 capture heterogeneity in the row and column means
- σ_{ab} describes the linear relationship between these two effects (i.e., whether actors who send [receive] a lot of ties also receive [send] a lot of ties)

Social Relations Model: Dyadic Variance

$$
\begin{aligned}
y_{ij} &= \mu + e_{ij} \\
e_{ij} &= a_i + b_j + \epsilon_{ij} \\
\{(a_1, b_1), \ldots, (a_n, b_n)\} &\sim N(0, \Sigma_{ab}) \\
\{(\epsilon_{ij}, \epsilon_{ji}) : \; i \neq j\} &\sim N(0, \Sigma_\epsilon), \text{ where}
\end{aligned}
$$

$$
\Sigma_{ab} = \begin{pmatrix} \sigma_a^2 & \sigma_{ab} \\ \sigma_{ab} & \sigma_b^2 \end{pmatrix} \qquad \Sigma_\epsilon = \sigma_\epsilon^2 \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}
$$

- ϵ_{ij} captures the within dyad effect
- Second-order dependencies are described by σ_ϵ^2
- Within dyad correlation, aka reciprocity, represented by ρ

What can we do with this?

- Let's model trade using the SR-R-M framework

$$y_{i,j} = \beta_d^T \mathbf{x}_{d,i,j} + \beta_s^T \mathbf{x}_{s,i} + \beta_r^T \mathbf{x}_{r,j} + a_i + b_j + \epsilon_{i,j}$$

Variables we might want to include:

- Polity of i and j

- Number of conflicts from i to j
- Log(Distance) between i and j
- Log Number of common IGOs between i and j

Probit Regression Framework

Hoff 2005; Hoff et al. 2013; Minhas et al. 2018

Threshold model: linking latent Z to Y

- $y_{ij} = 1(z_{ij} > 0)$
  - $z_{ij} = \beta^T x_{ij} + e_{ij}$

Social relations model: inducing network covariance

- $e_{ij} = a_i + b_j + \epsilon_{ij}$
- $\{(a_1, b_1), \ldots, (a_n, b_n)\} \sim N(0, \Sigma_{ab})$
- $\{(\epsilon_{ij}, \epsilon_{ji}) i \neq j\} \sim N(0, \Sigma_\epsilon)$

Estimation:

- MCMC algorithm in which we iteratively sample from the full conditionals of each parameter of interest

Running the model in R

MCMC routine:

Usage

```
ame(Y, Xdyad=NULL, Xrow=NULL, Xcol=NULL, rvar = !(model=="rrl") ,
cvar = TRUE,  dcor = !symmetric, nvar=TRUE, R = 0, model="nrm",
intercept=!is.element(model,c("rrl","ord")),
symmetric=FALSE,
odmax=rep(max(apply(Y>0,1,sum,na.rm=TRUE)),nrow(Y)), seed = 1, nscan =
10000, burn = 500, odens = 25, plot=TRUE, print = TRUE, gof=TRUE)
```

Arguments:

- `Y` an n x n square relational matrix
- `Xdyad` an n x n x pd array of dyadic covariates
- `Xrow` an n x pr array of sender covariates
- `Xcol` an n x pc array of receiver covariates
- `rvar` TRUE/FALSE: fit sender random effects
- `cvar` TRUE/FALSE: fit receiver random effects
- `dcor` TRUE/FALSE: fit dyadic correlation
- `model` one of "nrm", "bin", "ord", "cbin", "frn", "rrl"
- `intercept` TRUE/FALSE: fit with an intercept?
- `symmetric` TRUE/FALSE: are relations directed?
- `nscan` number of iterations of the markov chain
- `burn` burn in for the chain
- `odens` output density
- `R` dimension of multiplicative effects

Inputting nodal covariates

Nodal covariates should be structured as:

- an $n \times p$ matrix of covariates, where n corresponds to number of actors and p covariates
- In the directed case, row and nodal covariates need to be inputted separately into **`Xrow`** and **`Xcol`**

```
Xn[1:10,]
```

```
##          pop      gdp polity
## ARG 3.548755 5.864710   7.18
## AUL 2.895912 6.011414  10.00
## BEL 2.314514 5.370685  10.00
## BNG 4.789989 5.177956   5.00
## BRA 5.070915 6.963597   8.00
## CAN 3.377588 6.531009  10.00
## CHN 7.091101 8.114522  -7.00
## COL 3.652734 5.324862   7.82
## EGY 4.063542 5.371521  -3.55
## FRN 4.082272 7.101956   9.00
```

Inputting dyadic covariates

Dyadic covariates should be structured as:

- an $n \times n \times p$ array of covariates, where p now corresponds to the number of dyadic covariates

```
Xd[1:3,1:3,]
```

```
conflicts
```

```
##     ARG AUL BEL
## ARG  NA   0   0
## AUL   0  NA   0
## BEL   0   0  NA
```

```
distance
```

```
##       ARG   AUL   BEL
## ARG    NA 11.72 11.31
## AUL 11.72    NA 16.71
## BEL 11.31 16.71    NA
```

```
shared_igos
```

```
##      ARG  AUL  BEL
## ARG   NA 3.83 3.92
## AUL 3.83   NA 4.02
## BEL 3.92 4.02   NA
```

Lets first fit a Bayesian linear regression

```
fitOLS = ame(Y=Y,
  Xdyad=Xd, # incorp dyadic covariates
  Xrow=Xn, # incorp sender covariates
  Xcol=Xn, # incorp receiver covariates
  symmetric=FALSE, # tell AME trade is directed
  intercept=TRUE, # add an intercept
  model='nrm', # model type
  rvar=FALSE, # sender random effects (a)
  cvar=FALSE, # receiver random effects (b)
  dcor=FALSE, # dyadic correlation
  R=0, # we'll get to this later
  nscan=10000, burn=5000, odens=25,
  plot=FALSE, print=FALSE, gof=TRUE
  )
```

OLS vs Bayesian linear regression?

```
load('preezeObjects/ameresults.rda')

summary(ols)$'coefficients'
```

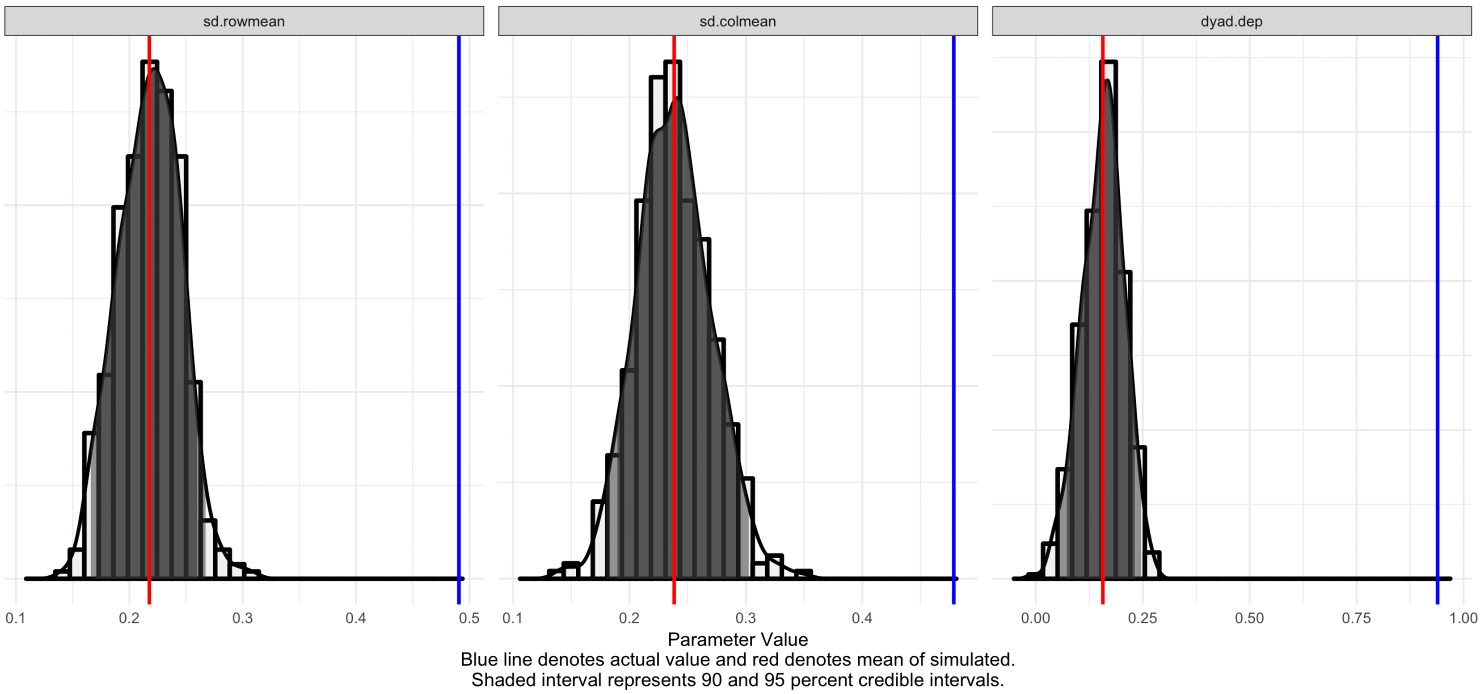
```
##                Estimate  Std. Error    t value     Pr(>|t|)
## (Intercept)  0.06878972 0.172325639  0.3991845 6.898559e-01
## polity.row   0.02196448 0.004757679  4.6166382 4.491947e-06
## polity.col   0.02729898 0.004740507  5.7586619 1.178005e-08
## conflicts    0.51754450 0.091336744  5.6663340 1.987264e-08
## distance    -0.04165499 0.005876931 -7.0878809 2.829161e-12
## shared_igos  0.17240405 0.042842108  4.0241729 6.218859e-05
```

```
summary(fitOLS)
```

```
##
## Regression coefficients:
##                    pmean   psd z-stat p-val
## intercept          0.060 0.175  0.343 0.731
## .row               0.022 0.005  4.219 0.000
## .col               0.027 0.005  5.591 0.000
## conflicts.dyad     0.526 0.090  5.865 0.000
## distance.dyad     -0.041 0.006 -6.837 0.000
## shared_igos.dyad   0.173 0.042  4.154 0.000
```

GOF analysis

```
gofPlot(fitOLS$GOF, symmetric=FALSE)
```



Running SRM model with covariates

```
fitSRM = ame(Y=Y,
             Xdyad=Xd, # incorp dyadic covariates
             Xrow=Xn, # incorp sender covariates
             Xcol=Xn, # incorp receiver covariates
             symmetric=FALSE, # tell AME trade is directed
             intercept=TRUE, # add an intercept
             model='nrm', # model type
             rvar=TRUE, # sender random effects (a)
             cvar=TRUE, # receiver random effects (b)
             dcor=TRUE, # dyadic correlation
             R=0, # we'll get to this later
             nscan=10000, burn=5000, odens=25,
             plot=FALSE, print=FALSE, gof=TRUE
             )
```

objects returned in `fitSRM`

```
names(fitSRM)
```

```
##  [1] "BETA" "VC"   "APM"  "BPM"  "U"    "V"    "UVPM" "EZ"   "YPM"  "GOF"
```

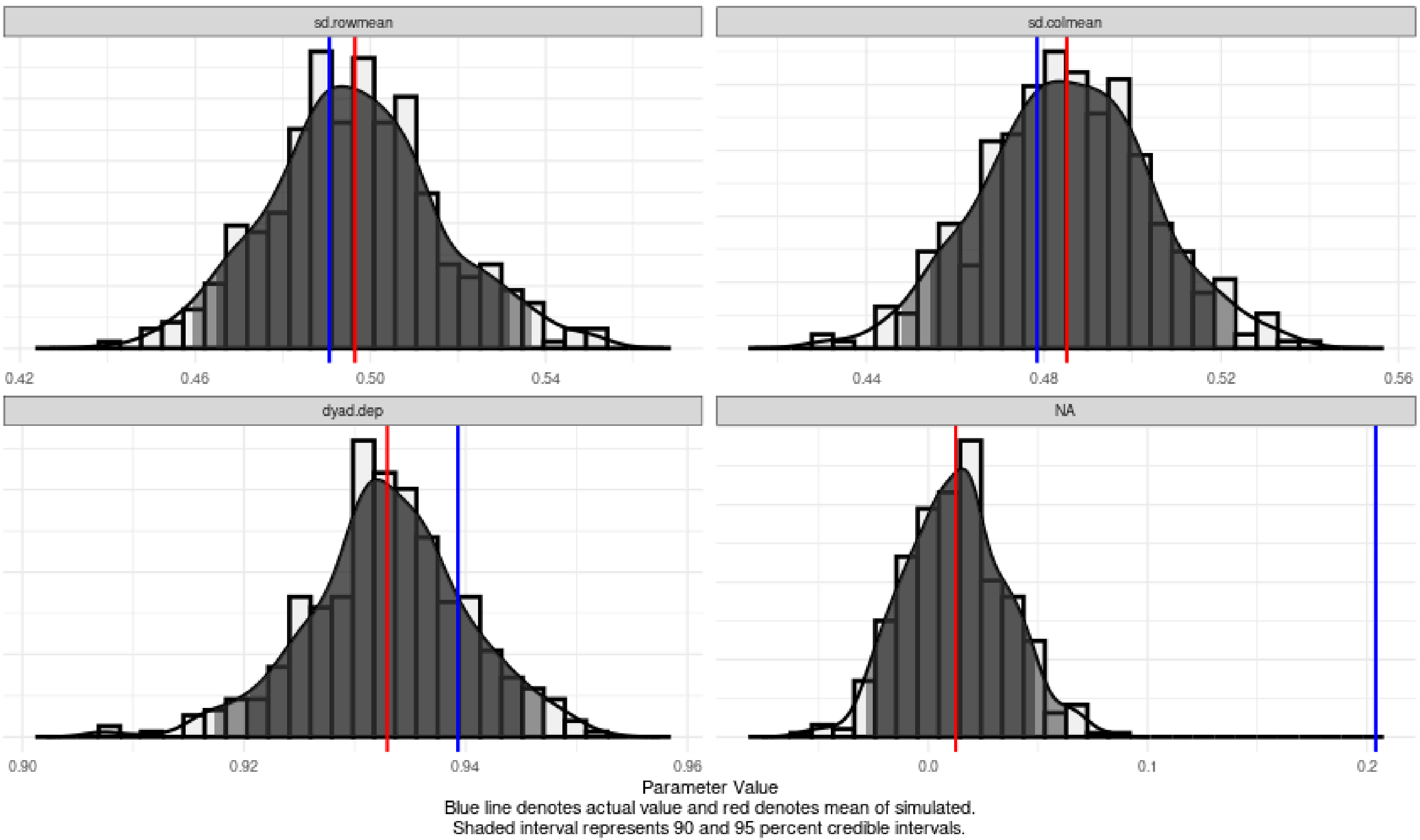
SRM results

```
summary(fitSRM)
```

```
##
## Regression coefficients:
##                      pmean   psd z-stat p-val
## intercept           -2.893 0.579 -4.994 0.000
## .row                 0.007 0.022  0.299 0.765
## .col                 0.014 0.020  0.686 0.493
## conflicts.dyad       0.079 0.038  2.078 0.038
## distance.dyad       -0.037 0.006 -6.324 0.000
## shared_igos.dyad     1.014 0.135  7.525 0.000
##
## Variance parameters:
##     pmean   psd
## va  0.453 0.124
## cab 0.392 0.116
## vb  0.414 0.118
## rho 0.782 0.020
## ve  0.156 0.010
```

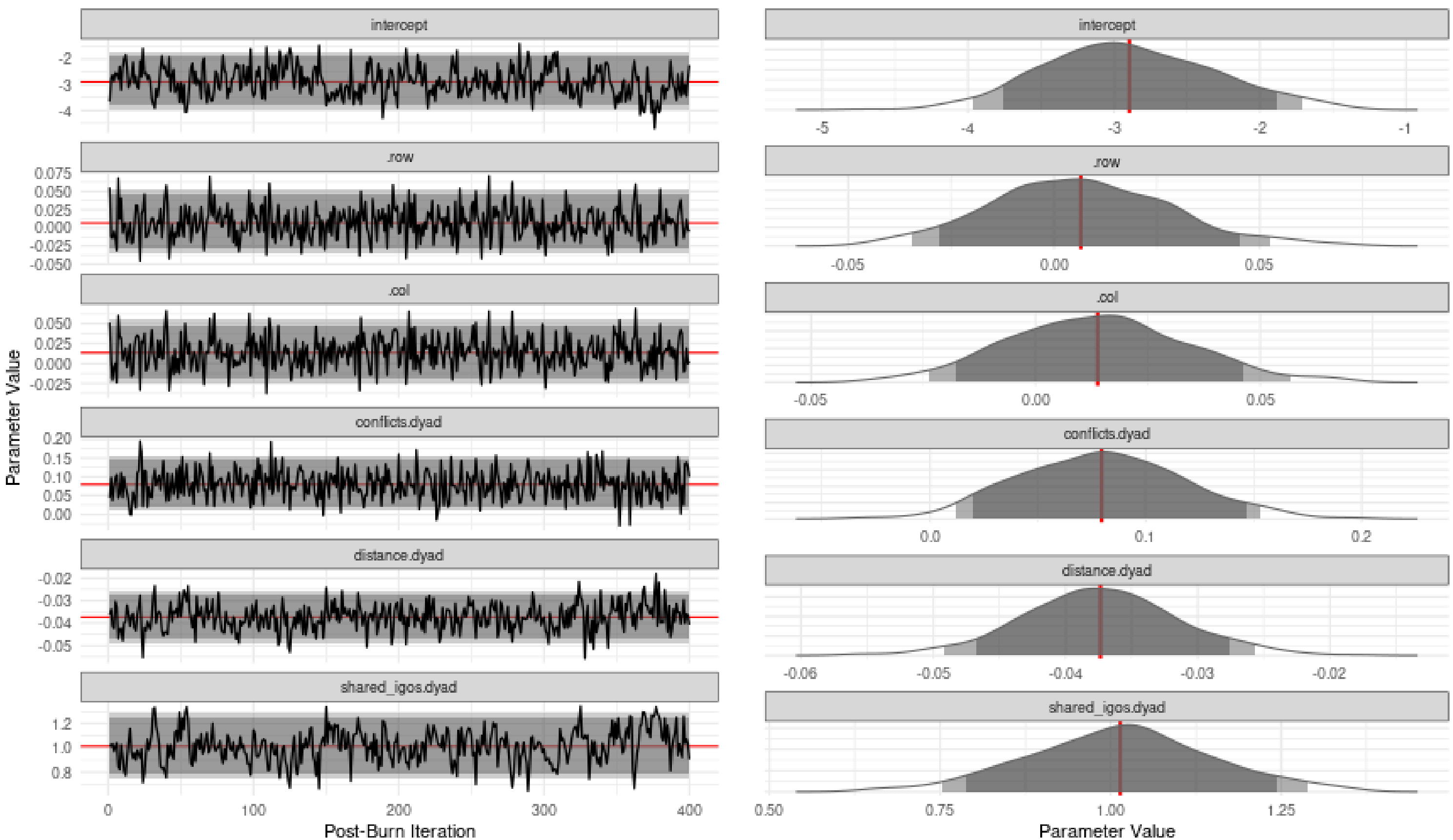
Capturing network features?

```
gofPlot(fitSRM$GOF, symmetric=FALSE)
```



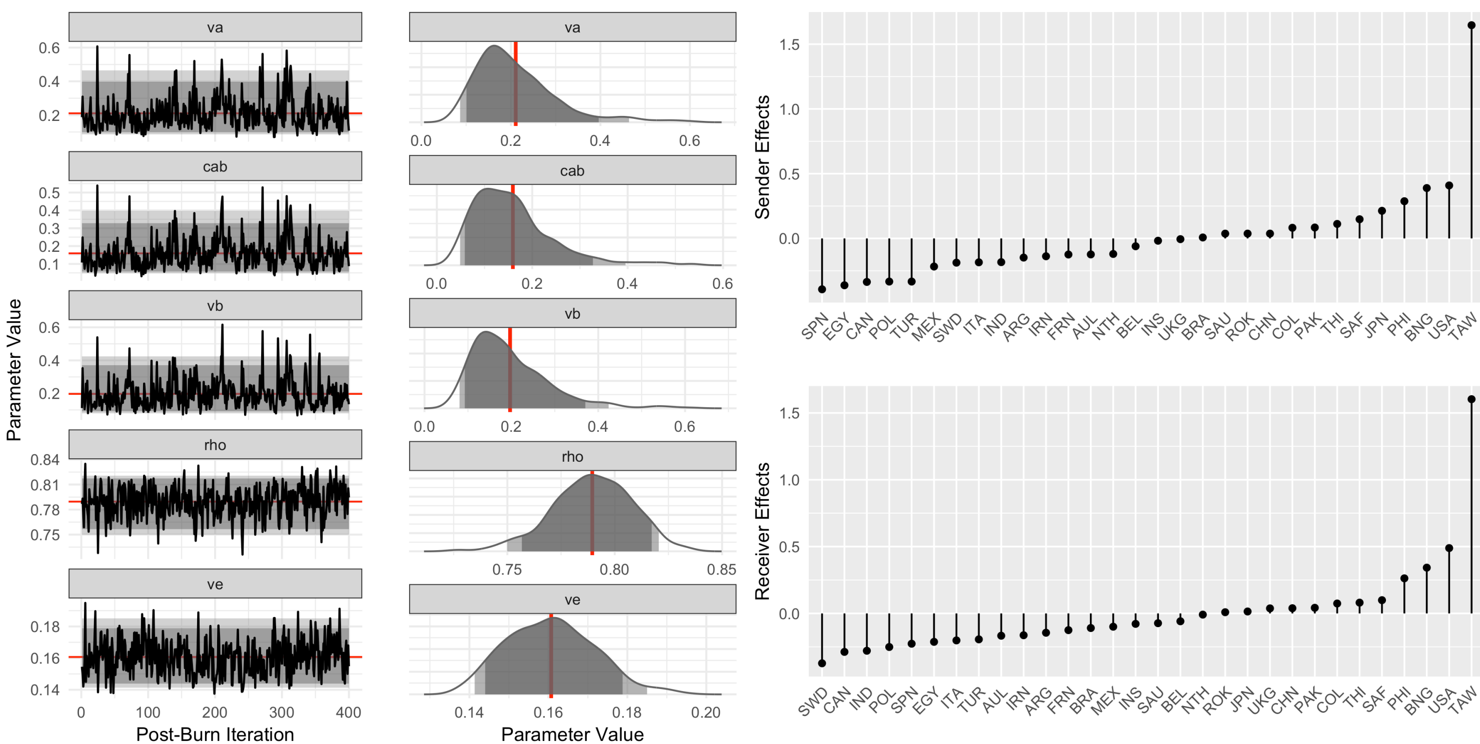
Trace plots

```
paramPlot(fitSRM$BETA)
```

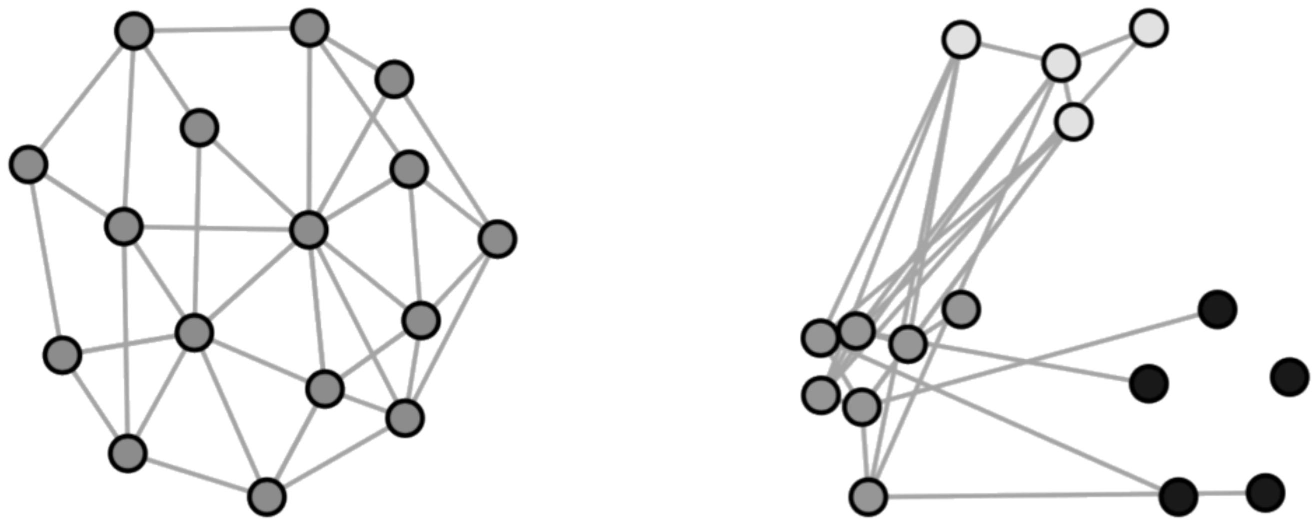


SRM variance parameters

```r
grid.arrange( paramPlot(fitSRM$VC),
  arrangeGrob( abPlot(fitSRM$APM, 'Sender Effects'),
               abPlot(fitSRM$BPM, 'Receiver Effects') ), ncol=2 )
```



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:

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