# HLSVD

*Woo Min Kim*

*5/5/2018*

## Hierarchical Latent SVD Model

As an extended idea of latent distance models (LDM), the latent eigenmodel was introduced to capture both homophily and stochastic equivalence among nodes. The main difference between LDMs and eigenmodels is the choice of the function dependent on the latent space. More specifically, eigenmodels use $a(u_i, u_j) = u_i^T \Lambda u_j$ instead of mere physical distance function. Besides, all latent class, distance and eigenmodels only deal with the symmetric function $d$ or $a$. However; the function also can be asymmetric if there exist sender and receiver effects are different. It is so-called SVD model, and the model specification is as follows:

$$\text{logit}P(Y_{ij} = 1) = \beta^T X_{ij} + a_i + b_j + u_i^T v_j + \epsilon_{ij}$$

where $u$ and $v$ are vectors of latent nodal attributes, and $a_i$ and $b_i$ are terms for nodal heterogeneity.

### MCMC Derivation

$$\begin{aligned}
\mathbf{Y}_{ijk} &= \mathbb{1}_{[\mathbf{Z}_{ijk}>0]} \\
\mathbf{Z}_{ijk} &= \sum_{p=0}^{P} \beta_{pk}^T \mathbf{X}_{ijpk} + a_{ik} + b_{jk} + u_{ik}^T v_{jk} + \epsilon_{ijk} \\
(a_{ik}, b_{ik}) &\overset{iid}{\sim} MVN_2((0,0), \mathbf{\Sigma}_{ab}) \\
(u_{ik}, v_{ik}) &\overset{iid}{\sim} MVN_4((0,0,0,0), \mathbf{\Sigma}_{uv}) \\
(\epsilon_{ijk}, \epsilon_{jik}) &\overset{iid}{\sim} MVN_2\Big( \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho_k \\ \rho_k & 1 \end{pmatrix} \Big) \\
\beta_p &\sim N(\mu_p, \sigma_p^2) \\
\mu_p &\sim N(\lambda, \tau^2) \\
\sigma_p^2 &\sim IG(\nu_0, S_0) \\
\mathbf{\Sigma}_{ab} &\sim IWishart(\nu_{ab}, S_{ab}^{-1}) \\
\mathbf{\Sigma}_{uv} &\sim IWishart(\nu_{uv}, S_{uv}^{-1})
\end{aligned}$$

Update $Z_{ijk}, Z_{jik}$:

$$(\mathbf{Z}_{ijk}, \mathbf{Z}_{jik}) \overset{iid}{\sim} MVN_2\Big( \begin{pmatrix} \sum_{p=0}^{P} \mathbf{X}_{ijpk}\beta_{pk} + a_{ik} + b_{jk} + u_{ik}^T v_{jk} \\ \sum_{p=0}^{P} \mathbf{X}_{ijpk}\beta_{pk} + a_{jk} + b_{ik} + u_{jk}^T v_{ik} \end{pmatrix}, \begin{pmatrix} 1 & \rho_k \\ \rho_k & 1 \end{pmatrix} \Big)$$

Let $\eta_{ijk} = \sum_{p=0}^{P} \mathbf{X}_{ijpk}\beta_{pk} + a_{ik} + b_{jk} + u_{ik}^T v_{jk}$

$$\mathbf{Z}_{ijk}|\mathbf{Z}_{jik} \sim N(\eta_{ijk} + \rho_k \cdot (\mathbf{Z}_{jik} - \eta_{jik}), 1 - \rho_k^2)$$

$$\begin{aligned}\Pr(\mathbf{Z}_{ijk}|\mathbf{Z}_{jik}, \mathbf{Y}_{ijk} = 1, \cdots) &\propto \Pr(\mathbf{Y}_{ijk} = 1|\cdots) \cdot \Pr(\mathbf{Z}_{ijk}|\mathbf{Z}_{jik}) \\ &= \begin{cases} \mathbb{1}_{[\mathbf{Z}_{ijk}>1]} \cdot dN(\mathbf{Z}_{ijk}; \eta_{ijk} + \rho_k \cdot (\mathbf{Z}_{jik} - \eta_{jik}), 1 - \rho_k^2) \text{ if } Y_{ijk} = 1 \\ \mathbb{1}_{[\mathbf{Z}_{ijk}<1]} \cdot dN(\mathbf{Z}_{ijk}; \eta_{ijk} + \rho_k \cdot (\mathbf{Z}_{jik} - \eta_{jik}), 1 - \rho_k^2) \text{ if } Y_{ijk} = 0 \end{cases}\end{aligned}$$

Update $\beta_k$:

Let $\bar{\mathbf{Z}}_{ijk} = \mathbf{Z}_{ijk} - a_{ik} - b_{jk} - u_{ik}^T v_{jk}$

$$\begin{aligned}\Pr(\bar{\mathbf{Z}}_{ijk}) &\propto \prod_{i \neq j}^{n_k} \exp\left( -\frac{1}{2} \begin{pmatrix} \bar{\mathbf{Z}}_{ijk} - \sum_p \mathbf{X}_{ijpk}\beta_{pk} \\ \bar{\mathbf{Z}}_{jik} - \sum_p \mathbf{X}_{jipk}\beta_{pk} \end{pmatrix}^T \mathbf{\Sigma}_\rho^{-1} \begin{pmatrix} \bar{\mathbf{Z}}_{ijk} - \sum_p \mathbf{X}_{ijpk}\beta_{pk} \\ \bar{\mathbf{Z}}_{jik} - \sum_p \mathbf{X}_{jipk}\beta_{pk} \end{pmatrix} \right) \\ \therefore \Pr(\bar{\mathbf{Z}}_{ijk}|\bar{\mathbf{Z}}_{jik}) &\sim N(\sum_p \mathbf{X}_{ijpk}\beta_{pk} + \rho_k(\bar{\mathbf{Z}}_{jik} - \sum_p \mathbf{X}_{jipk}\beta_{pk}), 1 - \rho_k^2) \\ &\propto \prod_{i \neq j} \exp\left( -\frac{1}{2}\Big(\bar{Z}_{ijk} - \rho_k \sum_p \mathbf{X}_{jipk})\beta_{pk} - \rho_k(\bar{Z}_{jik} - X_{jik}\beta_k)\Big)^2 \cdot (1 - \rho_k^2)^{-1} \right) \\ \therefore \Pr(\beta_k|\mathbf{Z}_k) &\propto \prod_{i \neq j} \exp\left( -\frac{1}{2}\Big((X_{ijk} - \rho_k \sum_p \mathbf{X}_{jipk})\beta_{pk} - \bar{Z}_{ijk} - \rho_k \bar{Z}_{jik}\Big)^2 \cdot (1 - \rho_k^2)^{-1} \right) \cdot \mathrm{dN}(\beta_k; \mu, \mathbf{\Sigma}_\beta) \\ &\propto \exp\left( -\frac{1}{2}(1 - \rho_k^2)^{-1}\Big(\mathbf{\Theta}_\beta \vec{\beta_k} - \mathrm{vec}(\mathbf{Z}_k) + \rho_k \mathrm{vec}(\mathbf{Z}_k^T)\Big)^T \Big(\mathbf{\Theta}_\beta \vec{\beta_k} - \mathrm{vec}(\mathbf{Z}_k) + \rho_k \mathrm{vec}(\mathbf{Z}_k^T)\Big) \right) \\ &\quad \times \exp\left( -\frac{1}{2}(\vec{\beta_k} - \vec{\mu})^T \mathbf{\Sigma}_\beta^{-1}(\vec{\beta_k} - \vec{\mu}) \right)\end{aligned}$$

where $\mathbf{\Theta}_\beta$ is a vector length of $n(n-1)$, and,

$$\mathbf{\Theta}_\beta = \Bigg( \begin{pmatrix} | & \cdots & | \\ \mathrm{vec}(\mathbf{X}_{1k}) & \cdots & \mathrm{vec}(\mathbf{X}_{Pk}) \\ | & \cdots & | \end{pmatrix} - \rho_k \begin{pmatrix} | & \cdots & | \\ \mathrm{vec}(\mathbf{X}_{1k}^T) & \cdots & \mathrm{vec}(\mathbf{X}_{Pk}^T) \\ | & \cdots & | \end{pmatrix} \Bigg)$$

Thus, we have $\mathbf{V}_n = \mathbf{A}_n^{-1}$ where $\mathbf{A}_n = \mathbf{A}_0 + \mathbf{A}_1$, and $\mathbf{A}_0 = \mathbf{\Sigma}_\beta^{-1}$, $\mathbf{A}_1 = (1 - \rho_k^2)^{-1}\mathbf{\Theta}_\beta^T \mathbf{\Theta}_\beta$

$\mu_n = \mathbf{V}_n m_n$ where $m_n = m_0 + m_1$, and $m_0 = \mathbf{\Sigma}_\beta^{-1}\mu$ and $m_1 = (1 - \rho_k^2)^{-1}\mathbf{\Theta}_\beta^T\Big(\mathrm{vec}(\mathbf{Z}_k) - \rho_k \mathrm{vec}(\mathbf{Z}_k^T)\Big)$

Update $a_i, b_i$:

Let $\hat{\mathbf{Z}}_{ijk} = \mathbf{Z}_{ijk} - \sum_p \mathbf{X}_{ijpk}\beta_{pk} - u_{ik}^T v_{jk}$

$$\Pr\left( \begin{pmatrix} a_{ik} \\ b_{ik} \end{pmatrix} | \begin{pmatrix} \hat{Z}_{ijk} \\ \hat{Z}_{jik} \end{pmatrix}, \dots \right) \propto$$

$$\prod_{i \neq j}^{n_k} \exp\left( -\frac{1}{2} \begin{pmatrix} \hat{Z}_{ijk} - a_{ik} - b_{jk} \\ \hat{Z}_{jik} - a_{jk} - b_{ik} \end{pmatrix}^T \Sigma_\rho^{-1} \begin{pmatrix} \hat{Z}_{ijk} - a_{ik} - b_{jk} \\ \hat{Z}_{jik} - a_{jk} - b_{ik} \end{pmatrix} \right) \cdot \mathrm{dMVN}_2\left( \begin{pmatrix} a_{ik} \\ b_{ik} \end{pmatrix}; \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \Sigma_{ab} \right)$$

$$\therefore \Pr(\hat{\mathbf{Z}}_{ijk} | \hat{\mathbf{Z}}_{jik}) \sim N(a_{ik} - (-b_{jk} - \rho_k(\hat{\mathbf{Z}}_{jik} - a_{jk} - b_{ik})), 1 - \rho_k^2)$$

$$\therefore \Pr(a_{ik} | \mathbf{Z}_k) \propto \prod_{j \neq i} \exp\left( -\frac{1}{2}(1-\rho_k^2)^{-1}(\hat{\mathbf{Z}}_{ijk} - a_{ik} - b_{jk} - \rho_k(\hat{\mathbf{Z}}_{jik} - a_{jk} - b_{ik}))^2 \right)$$

$$\cdot \mathrm{dN}(a_{ik} | b_{ik} \ ; \ \mu_{a|b} = \mathbf{\Sigma}_{ab[1,2]} \mathbf{\Sigma}_{ab[2,2]}^{-1} b_{ik}, \ \sigma_{a|b}^2 = \mathbf{\Sigma}_{ab[1,1]} - \mathbf{\Sigma}_{ab[1,2]} \mathbf{\Sigma}_{ab[2,2]}^{-1} \mathbf{\Sigma}_{ab[2,1]})$$

$$\propto \exp\left( -\frac{1}{2}(1-\rho_k)^{-1} \sum_{j \neq i} (a_{ik} - \theta_j)^T (a_{ik} - \theta_j) \right) \cdot \exp\left( -\frac{1}{2}(a_{ik} - \mu_{a|b})^2 \sigma_{a|b}^{-2} \right)$$

where $\theta_j = -b_{jk} - \rho_k(\hat{\mathbf{Z}}_{jik} - a_{jk} - b_{ik}) + \hat{\mathbf{Z}}_{ijk}$ and $i$ is fixed. Then, we can easily figure out its posterior mean and variance.

$\mathbf{V}_n = \mathbf{A}_n^{-1}$ where $\mathbf{A}_n = \mathbf{A}_0 + \mathbf{A}_1$, and $\mathbf{A}_0 = \sigma_{a|b}^{-2}, \mathbf{A}_1 = (n_k - 1)(1 - \rho_k^2)^{-1}$

$\mu_n = \mathbf{V}_n m_n$ where $m_n = m_0 + m_1$, and $m_0 = \sigma_{a|b}^{-2} \mu_{a|b}$ and $m_1 = (1 - \rho_k^2)^{-1} \sum_{j \neq i} \theta_j$

Likewise, we can update $b_{ik}$.

Update $u_i, v_i$:

Let $\tilde{Z}_{ijk} = Z_{ijk} - X_{ijk}\beta_k - a_{ik} - b_{jk}$

$$p\left( \begin{pmatrix} u_{ik} \\ v_{ik} \end{pmatrix} | \begin{pmatrix} \tilde{Z}_{ijk} \\ \tilde{Z}_{jik} \end{pmatrix}, \dots \right) \propto$$

$$\prod_{i \neq j}^{n_k} \exp\left( -\frac{1}{2} \begin{pmatrix} \tilde{Z}_{ijk} - u_{ik}^T v_{jk} \\ \tilde{Z}_{jik} - u_{ik}^T v_{jk} \end{pmatrix}^T \Sigma_\rho^{-1} \begin{pmatrix} \tilde{Z}_{ijk} - u_{ik}^T v_{jk} \\ \tilde{Z}_{jik} - u_{ik}^T v_{jk} \end{pmatrix} \right) \cdot \mathrm{dMVN}_4\left( \begin{pmatrix} u_{ik} \\ v_{ik} \end{pmatrix}; \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \Sigma_{uvk} \right)$$

$$\therefore \Pr(\tilde{Z}_{ijk} | \tilde{Z}_{jik}) \sim \mathrm{N}(u_{ik}^T v_{jk} + \rho_k(\tilde{Z}_{jik} - u_{jk}^T v_{ik}), 1 - \rho_k^2)$$

$$\therefore \Pr(u_{ik} | \tilde{Z}_k) \propto \prod_{j \neq i} \exp\left( -\frac{1}{2}(1-\rho_k^2)^{-1}(\tilde{Z}_{ijk} - u_{ik}^T v_{jk} - \rho_k(\tilde{Z}_{jik} - u_{jk}^T v_{ik}))^2 \right)$$

$$\mathrm{dMVN}_2(u_{ik} | v_{ik} \ ; \ \mu_{u|v} = \Sigma_{uv[1,2]} \Sigma_{uv[2,2]}^{-1} v_{ik}, \ \Sigma_{u|v} = \Sigma_{uv[1,1]} - \Sigma_{uv[1,2]} \Sigma_{uv[2,2]}^{-1} \Sigma_{uv[2,1]})$$

$$\propto \exp\left( -\frac{1}{2}(1-\rho_k^2)^{-1}(\mathbf{\Theta}_V u_{ik} - \vec{\mu}_u)^T (\mathbf{\Theta}_v u_{ik} - \vec{\mu}_u) \right) \cdot \exp\left( -\frac{1}{2}(u_{ik} - \mu_{u|v})^T \Sigma_{u|v} (u_{ik} - \mu_{u|v}) \right)$$

$$\text{Where } \mathbf{\Theta}_V = \begin{pmatrix} | & | \\ \vec{v}_{jk[-i,1]} & \vec{v}_{jk[-i,2]} \\ | & | \end{pmatrix} \text{ ,and } \vec{\mu}_u = \begin{pmatrix} | \\ \tilde{Z}_{i,\cdot,k} - \rho_k(\tilde{Z}_{\cdot,i,k} - u_{\cdot}^T v_i) \\ | \end{pmatrix}$$

$$\text{, and dimensions are } (n_k - 1) \times 2 \text{and } (n_k - 1) \times 1, \text{respectively.}$$

Then, we have $\mathbf{V}_n = \mathbf{A}_n^{-1}$ where $\mathbf{A}_n = \mathbf{A}_0 + \mathbf{A}_1$, and $\mathbf{A}_0 = \Sigma_{u|v}^{-1}, \mathbf{A}_1 = (1 - \rho_k^2)^{-1} \mathbf{\Theta}_V^T \mathbf{\Theta}_V$

$\mu_n = \mathbf{V}_n m_n$ where $m_n = m_0 + m_1$, and $m_0 = \Sigma_{u|v}^{-1}\mu_{u|v}$ and $m_1 = (1-\rho_k^2)^{-1}\mathbf{\Theta}_V^T\vec{\mu}_u$.

Update $\rho_k$: Let $\tilde{\epsilon}_{ijk} = Z_{ijk} - X_{ijk}\beta_k - a_{ik} - b_{jk} - u_{ik}^T v_{jk}$

$$p\left(\rho_k | \begin{pmatrix}\tilde{\epsilon}_{ijk}\\ \tilde{\epsilon}_{jik}\end{pmatrix}, \dots\right) \propto \prod_{i\neq j}^{n_k} \exp\left(-\frac{1}{2}\begin{pmatrix}\tilde{\epsilon}_{ijk}\\ \tilde{\epsilon}_{jik}\end{pmatrix}^T \Sigma_\rho^{-1} \begin{pmatrix}\tilde{\epsilon}_{ijk}\\ \tilde{\epsilon}_{jik}\end{pmatrix}\right) \cdot \text{dUnif}(\rho_k; -1, 1)$$

To update $\rho_k$, Metropolis-Hastings is needed due to non-conjugacy.

Update $\mu$:

$$\begin{aligned} p(\mu|\vec{Y}, \dots) &\propto \prod_{k=1}^{K} \text{dNorm}(\beta_k; \mu, \sigma) \cdot \text{dNorm}(\mu; \lambda, \tau) \\ &\sim \text{N}(\mu; \frac{\lambda/\tau + \sum_k \beta_k/\sigma^2}{(1\tau + K/\sigma^2)}, (1/\tau + K/\sigma^2)^{-1}) \end{aligned}$$

Update $\sigma^2$:

$$\begin{aligned} p(\sigma^2|\vec{Y}, \dots) &\propto \prod_{k=1}^{K} \text{dNorm}(\beta_k; \mu, \sigma) \cdot \text{dIG}(\sigma^2; \nu_0, S_0) \\ &\sim \text{IG}(\sigma^2; \frac{\nu_0 + K}{2}, \frac{\nu_0 \cdot S_0 + \sum_k (\beta_k - \mu)^2}{2}) \end{aligned}$$

Update $\Sigma_{ab}$:

$$\begin{aligned} p(\Sigma_{abk}|\vec{Y}, \dots) &\propto \prod_{i=1}^{n_k} \text{dMVN}\left(\begin{pmatrix}a_{ik}\\ b_{ik}\end{pmatrix}; \vec{0}, \Sigma_{ab}\right) \cdot \text{dIWishart}(\Sigma_{ab}; \nu_{ab}, S_{abk}^{-1}) \\ &\sim \text{IWishart}(\Sigma_{abk}; \nu_{ab} + n_k, [S_{ab} + \sum_{i=1}^{n_k} \begin{pmatrix}a_{ik}\\ b_{ik}\end{pmatrix}\begin{pmatrix}a_{ik}\\ b_{ik}\end{pmatrix}^T]^{-1}) \end{aligned}$$

Update $\Sigma_{uv}$:

$$\begin{aligned} p(\Sigma_{uvk}|\vec{Y}, \dots) &\propto \prod_{i=1}^{n_k} \text{dMVN}\left(\begin{pmatrix}u_{ik}\\ v_{ik}\end{pmatrix}; \vec{0}, \Sigma_{uv}\right) \cdot \text{dIWishart}(\Sigma_{uvk}; \nu_{uv}, S_{uv}^{-1}) \\ &\sim \text{IWishart}(\Sigma_{uvk}; \nu_{uv} + n_k, [S_{uv} + \sum_{i=1}^{n_k} \begin{pmatrix}u_{ik}\\ v_{ik}\end{pmatrix}\begin{pmatrix}u_{ik}\\ v_{ik}\end{pmatrix}^T]^{-1}) \end{aligned}$$

## Coding (Temporary)

For the following function, two main arguments are required: data and edge_covariate. Both arguments should follow a list format. For now, the argument, edge_covariate, takes only one edge-specific covariate.

```r
hsvd = function(data, edge_covariate, dyad_dep = T, num_iter = 1000, verbose = T, hier = T){

  suppressMessages(require(tmvtnorm))
  suppressMessages(require(mvtnorm))
  suppressMessages(require(truncnorm))
  suppressMessages(require(MCMCpack))

  iternum = num_iter
  Y = data
  COV = edge_covariate
  K = length(Y)
  nk = unlist(lapply(Y, nrow))

   # Create a list to collect MCMC samples
    Chain = list("beta" = list(),
                 "rho" = matrix(nrow=iternum, ncol=K),
                 "sigma_ab" = list(), "sigma_uv" = list(),
                 "U" = list(), "V" = list(),
                 "a" = list(), "b" = list(),
                 "z" = list(), "mu" = matrix(nrow=iternum, ncol=2),
                 "s2" = matrix(nrow=iternum, ncol=2))

    for (i in 1:K){
      Chain$U[[i]] = Chain$V[[i]] = list()
      for (j in 1:iternum){
        Chain$U[[i]][[j]] = Chain$V[[i]][[j]] = list()
      }
      Chain$beta[[i]] = matrix(nrow=iternum,ncol=2)
      Chain$a[[i]] = Chain$b[[i]] = matrix(nrow=iternum, ncol=nk[i])
    }

    # Initial points for parameters
    {
      beta0 = rep(0,K)
      beta1 = rep(0,K)
      beta = cbind(beta0,beta1)
      mu0 = mu = 0
      s02 = s2 = 1
      # sigma_uv
      sigma_uv =  matrix(0.5,4,4)
      diag(sigma_uv) = 1

      # Random Initial points for U and V
      U = V = list()
      for (i in 1:K){
        U[[i]] = as.matrix(rmvnorm(nk[i],c(0,0),sigma_uv[1:2,1:2]),ncol=2)
        V[[i]] = as.matrix(rmvnorm(nk[i],c(0,0),sigma_uv[3:4,3:4]),ncol=2)
      }


      # Create Starting points for z, a, b
      z = a = b = list()
```

```
    # sigma_ab
    sigma_ab = matrix(c(1,0.5,0.5,1),2,2)

    # fill in z,a,b with random numbers
    for (k in 1:K){
      z[[k]] = matrix(rnorm(nk[k]*nk[k]), ncol=nk[k],nrow=nk[k])
      diag(z[[k]]) = NA
      a[[k]] = b[[k]] = numeric(nk[k])
    }

    #initial for rho, beta0, beta1
    if (dyad_dep == T){
      rho = rep(0.5,K)
    } else rho = rep(0,K)
  }


#MCMC

ptm <- proc.time()
  for (k in 1:K){
    diag(Y[[k]]) = 0
    diag(Cov1[[k]]) = 0
    diag(z[[k]]) = 0
  }

  for (sim in (1):(iternum)){

    for (k in 1:K){
      index = which(diag(nk[k]) == 1)

      ### Update Z ###
      etas = matrix(cbind(1,c(Cov1[[k]])) %*% beta[k,] +
                      rowSums(U[[k]][rep(seq_len(nrow(U[[k]])), nk[k]),] *
                                V[[k]][rep(seq_len(nrow(V[[k]])), each=nk[k]),]) +
                      rep(a[[k]], nk[k]) +
                      rep(b[[k]], each=nk[k]),
                    nk[k],nk[k])

      diag(etas) = 0

      m = c(etas) + rho[k]*( c(t(z[[k]])) - c(t(etas)))

      z[[k]] = matrix(rtruncnorm(1,
                                 a = ifelse(c(Y[[k]]), 0, -Inf),
                                 b = ifelse(c(Y[[k]]), Inf, 0),
                                 mean = m, sd = sqrt(1-rho[k]^2)),
                      nrow = nk[k], ncol = nk[k])

      ### Update beta0, beta1 ###
      sigma_beta = matrix(c(s02,0,0,s2),2,2)

      zbar = matrix(
```

```
    c(z[[k]])-
      rep(a[[k]], nk[k]) -
      rep(b[[k]], each=nk[k]) -
      rowSums(U[[k]][rep(seq_len(nrow(U[[k]])), nk[k]),] *
                V[[k]][rep(seq_len(nrow(V[[k]])), each=nk[k]),]),
    nk[k],nk[k])

  diag(zbar) = 0
  rp = 1 / (1-rho[k]^2)
  X = (cbind(1,c(Cov1[[k]])) - rho[k]*(cbind(1, c(t(Cov1[[k]])))))[-index,]

  V_beta = solve( solve(sigma_beta) + rp*crossprod(X))

  mu_beta = V_beta %*% ( solve(sigma_beta) %*% c(mu0,mu)
                         + rp * crossprod(X, c(zbar)[-index] - rho[k] * c(t(zbar))[-index]))

  beta[k,] = rmvnorm(1, mu_beta, V_beta)

  ### Update a, b ###
  zhat = matrix( c(z[[k]]) -
                   c(cbind(1, c(Cov1[[k]]))%*%beta[k,]) -
                   rowSums(U[[k]][rep(seq_len(nrow(U[[k]])), nk[k]),] *
                             V[[k]][rep(seq_len(nrow(V[[k]])), each=nk[k]),]),
                 nk[k], nk[k])

  S11 = sigma_ab[1,1]; S22 = sigma_ab[2,2]; S12 = S21 = sigma_ab[1,2]

  sigma2_a = S11 - S12%*% solve(S22) %*% S21
  sigma2_b = S22 - S21%*% solve(S11) %*% S12

  for (i in sample(1:nk[k])){
    theta_ai = sum(zhat[i,-i] - rho[k]*(zhat[-i,i] - a[[k]][-i] - b[[k]][i]) - b[[k]][-i])

    m_ab = S12%*%solve(S22)%*%b[[k]][i]

    A0 = 1/sigma2_a; A1 = (nk[k]-1) * rp
    m0 = 1/sigma2_a*m_ab; m1 = rp * theta_ai

    mu_ai = 1/(A0+A1)*(m0+m1)
    s2_ai = 1/(A0+A1)
    a[[k]][i] = rnorm(1, mu_ai, sqrt(s2_ai))


    theta_bi = sum(zhat[-i,i] - rho[k]*(zhat[i,-i] - b[[k]][-i] - a[[k]][i]) - a[[k]][-i])
    m_ba = S21%*%solve(S11)%*%a[[k]][i]
    A0 = 1/sigma2_b; A1 = (nk[k]-1) * rp
    m0 = 1/sigma2_b*m_ba; m1 = rp * theta_bi

    mu_bi = 1/(A0+A1)*(m0+m1)
    s2_bi = 1/(A0+A1)
    b[[k]][i] = rnorm(1, mu_bi, sqrt(s2_bi))
  }
```

```
### Update U, V ###

ztilde = matrix( c(z[[k]]) -
                   cbind(1, c(Cov1[[k]]))%*%beta[k,] -
                   rep(a[[k]], nk[k]) - rep(b[[k]], each=nk[k]),
                 nk[k], nk[k])

diag(ztilde) = 0

S11 = sigma_uv[1:2,1:2]; S22 = sigma_uv[3:4,3:4]; S12 = sigma_uv[1:2,3:4]; S21 = sigma_uv[3:4,1

sigma2_u = S11 - S12%*% solve(S22) %*% S21
sigma2_v = S22 - S21%*% solve(S11) %*% S12

for (i in sample(1:nk[k])){
  Theta_V = V[[k]][-i,]
  mu_u = ztilde[i,-i] - c(rho[k]*(ztilde[-i,i] - U[[k]][-i,] %*% V[[k]][i,]))

  mu_uv = S12%*%solve(S22)%*%V[[k]][i,]

  A0 = solve(sigma2_u); A1 = rp*crossprod(Theta_V)

  m0 = solve(sigma2_u)%*%mu_uv; m1 = rp*crossprod(Theta_V,mu_u)

  mu_ui = solve(A0+A1)%*%(m0+m1)

  U[[k]][i,] = mvrnorm(1,mu_ui,solve(A0+A1))

  Theta_U = U[[k]][-i,]
  mu_v = ztilde[-i,i] - c(rho[k]*(ztilde[i,-i] - V[[k]][-i,] %*% U[[k]][i,]))

  mu_vu = S21%*%solve(S11)%*%U[[k]][i,]

  A0 = solve(sigma2_v); A1 = rp*crossprod(Theta_U)
  m0 = solve(sigma2_v)%*%mu_vu; m1 = rp*crossprod(Theta_U,mu_v)
  mu_vi = solve(A0+A1)%*%(m0+m1)

  V[[k]][i,] = mvrnorm(1,mu_vi,solve(A0+A1))
}



### Update rho ###
if (dyad_dep == T){
etilde = matrix(c(z[[k]]) -
                  cbind(1, c(Cov1[[k]]))%*%beta[k,] -
                  rep(a[[k]], nk[k]) - rep(b[[k]], each=nk[k]) -
                  rowSums(U[[k]][rep(seq_len(nrow(U[[k]])), nk[k]),] *
                          V[[k]][rep(seq_len(nrow(V[[k]])), each=nk[k]),]),
                nk[k],nk[k])
diag(etilde) = 0
EM<-cbind(etilde[upper.tri(etilde)],t(etilde)[upper.tri(etilde)] )
emcp<-sum(EM[,1]*EM[,2])
```

```
    emss<-sum(EM^2)
    m<- nrow(EM)

    sr = 2 * (1-cor(etilde)[1,2]^2)/sqrt(m)

    rho_p = rho[k] + sr * qnorm( runif(1,pnorm( (-1-rho[k])/sr), pnorm( (1-rho[k])/sr)))

    rd <- (-.5*(m*log(1-rho_p^2)+(emss-2*rho_p*emcp)/(1-rho_p^2)))-
      (-.5*(m*log(1-rho[k]^2 )+(emss-2*rho[k]*emcp )/(1-rho[k]^2 )))+
      ( (-.5*log(1-rho_p^2)) - (-.5*log(1-rho[k]^2)) )


    if (log(runif(1)) < rd) rho[k] = rho_p
    }

    Chain$rho[sim,k] = rho[k]
    Chain$beta[[k]][sim,] = beta[k,]
    Chain$U[[k]][[sim]] = U[[k]]
    Chain$V[[k]][[sim]] = V[[k]]
    Chain$a[[k]][sim,] = a[[k]]
    Chain$b[[k]][sim,] = b[[k]]
  }

  if (hier == T){
    ### Update Sigma_ab ###
    Sab0 = matrix(c(1,0,0,1),ncol=2); nuab0 = 2+2
    S_theta_ab = matrix(rowSums(apply(do.call(rbind,Map(cbind,a,b)) ,1,tcrossprod)),2,2)
    sigma_ab = riwish(v = nuab0 + sum(nk) , S = Sab0 + S_theta_ab)

    ### Update Sigma_uv ###
    Suv0 = matrix(c(rep(5,16)),ncol=4); diag(Suv0) = 10; nuuv0 = 4+2
    S_theta_uv = matrix(rowSums(apply(do.call(rbind,Map(cbind,U,V)),1,tcrossprod)),4,4)
    sigma_uv = riwish(v =nuuv0 + sum(nk), S= Suv0 + S_theta_uv)

    ### Update mu0 (mean of beta0) ###
    s00 = 1; mu00 = 0
    s02_n= (1/s00 + (K / s02))^(-1)
    mu0_n= (mu00 / s00 + sum(beta[,1]) / s02) * s02_n
    mu0 = rnorm(1, mu0_n, sqrt(s02_n))

    ### Update mu (mean of beta1) ###
    s10 = 1; mu10 = 0
    s2_n= (1/s10 + (K / s2))^(-1)
    mu_n= (mu10 / s10 + sum(beta[,2]) / s2) * s2_n
    mu = rnorm(1, mu_n, sqrt(s2_n))

    ### Update S0 (var of beta0) ###
    c = 5; d = 0.25  # Hyperparameters

    nu0_n = c + K
    ss0_n = 1/nu0_n * (c * d + (K-1)*var(beta[,1]) + K/(K+1)*(mean(beta[,1])-mu0)^2)
    s02 = 1/ rgamma(1, nu0_n/2, nu0_n*ss0_n/2)
```

```
        ### Update S1 (var of beta1) ###
        nu_n = c + K
        ss_n = 1/nu_n * (c * d + (K-1)*var(beta[,2]) + K/(K+1)*(mean(beta[,2])-mu)^2)
        s2 = 1/ rgamma(1, nu_n/2, nu_n*ss_n/2)

        Chain$sigma_ab[[sim]] = sigma_ab
        Chain$sigma_uv[[sim]] = sigma_uv
        Chain$mu[sim,] = cbind(mu0,mu)
        Chain$s2[sim,] = cbind(s02,s2)
      }


      if (sim %% 100 == 0 & verbose == T) {
        ptm2 = (proc.time() - ptm)[3]
        cat(sim,"/",iternum,"   (time:", ptm2, ")",
            "     (estimated time left:", ptm2*(iternum-sim)/100, ") \n")
        ptm = proc.time()
      }
    }
    return(Chain)
}
```

## Test (with simulated data)

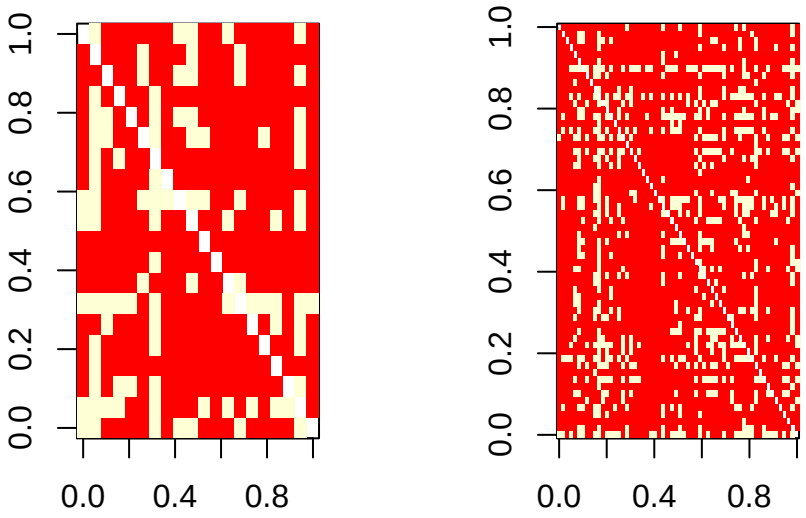


Figure 1: Two Simulated Adjacency Matrice

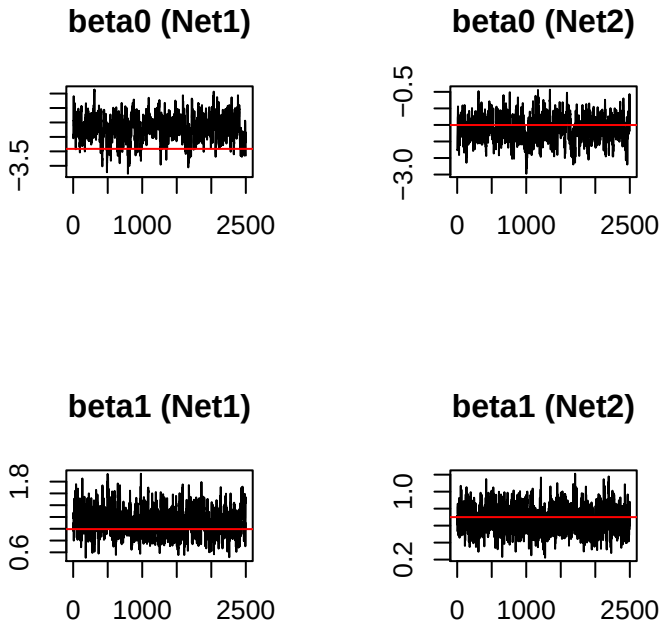


Figure 2: Trace Plot for BETAs for Scenario 1

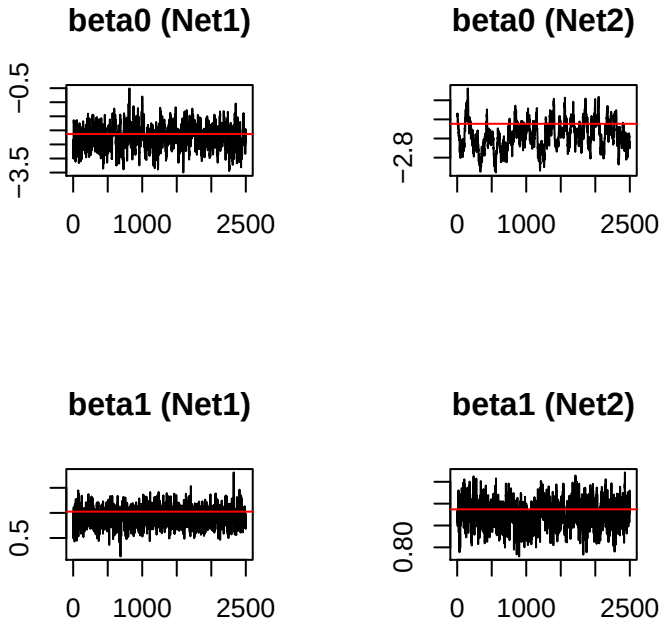


Figure 3: Trace Plot for BETAs for Scenario 3

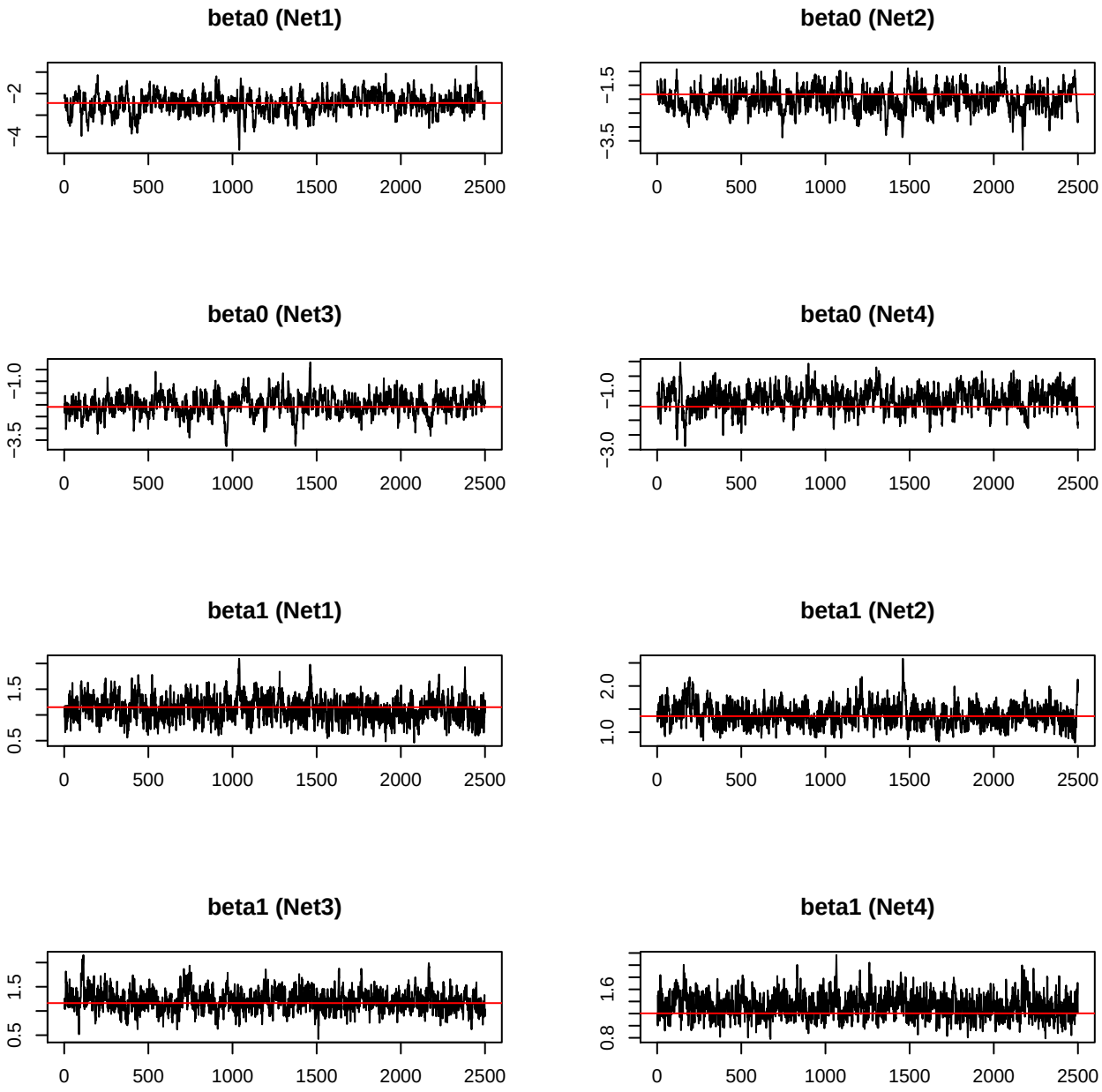


Figure 4: Trace Plot for BETAs for Scenario 4

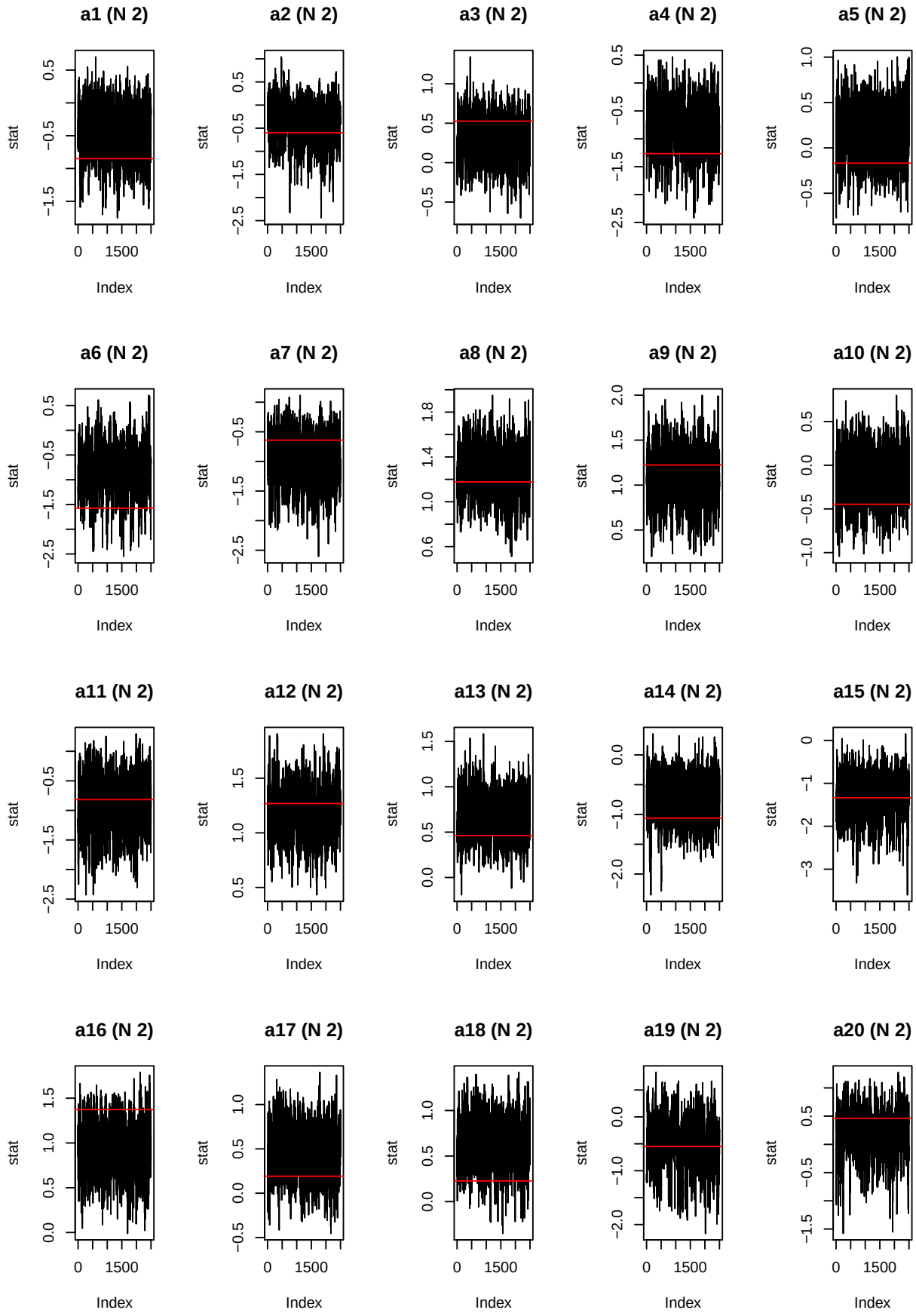


Figure 5: Trace Plot for Row Effect for Scenario 3

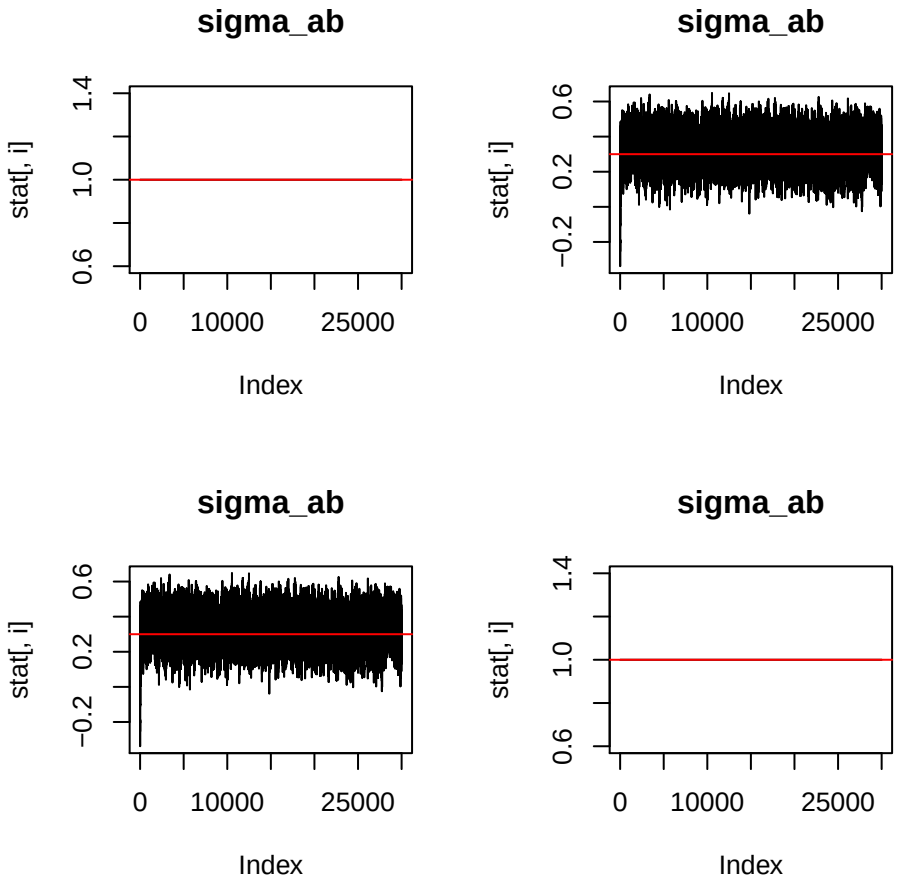


Figure 6: Trace Plot for Correlation Matrix for Row Effect for Scenario 3

30,000 posterior samples were gathered and trace plots of some of parameters are provided above. As can be seen, the trace plots well captures the true values (red horizontal line) of most of parameters. In order to explore the effects of (1) the number of networks and (2) the size of each network in the data set, five difference scenarios were considered.

1. Two networks with sizes (20, 20).
2. Two networks with sizes (20, 60).
3. Two networks with sizes (20, 120).
4. Four networks with sizes of 20s.
5. Eight networks with sizes of 20s.

Naturally, the size of a network affect the estimations of parameters; the larger size, the smaller variance in estimation. This can be checked by comparing Scenarios (1, 2 and 3).

With the given simulated data, the all covariate effects, $\beta_1$, have great mixing with their trace plots; however, the intercept term, $\beta_0$, have relatively bad mixings with large networks. The MCMC effective size of $\beta_0$s in Scenarios 1, 2 and 3 are (217.68, 255,93), (260.78, 68.98), and (244.58, 48.01), respectively. This happens because the variance of the posterior distribution is $[\mathbf{\Sigma}_\beta^{-1} + (1-\rho_k^2)^{-1}\mathbf{\Theta}_\beta^T\mathbf{\Theta}_\beta]^{-1}$. Since $\mathbf{\Theta}_\beta = \left( \begin{pmatrix} | & \cdots & | \\ \text{vec}(\mathbf{X}_{1k}) & \cdots & \text{vec}(\mathbf{X}_{Pk}) \\ | & \cdots & | \end{pmatrix} - \rho_k \begin{pmatrix} | & \cdots & | \\ \text{vec}(\mathbf{X}_{1k}^T) & \cdots & \text{vec}(\mathbf{X}_{Pk}^T) \\ | & \cdots & | \end{pmatrix} \right)$, $\mathbf{\Theta}_\beta^T\mathbf{\Theta}_\beta$ increases by far if the size of a network increases; thereby making the variance of the posterior distribution small. Thus, every jump of Gibbs sampler for $\beta$s becomes tiny.

The effect of the number of networks in a dataset can be explored by comparing Secenarios 1, 4 and 5; however, as long as the size of the networks are identical, differences in the traceplots could not be found.

Furthermore, it is quite reasonable to expect shrinkage effect by the hierarchical structure. Consider $\beta_{0k}$ where $k$ indicates the network-wise index, then we can come up with a grand mean of $\beta_{0k}$'s by setting up a prior distribution of $beta_{0k}$s with the grand mean as a hyperparameter. Then, the estimates of $\beta_{0k}$ will be shifted into the hyperparameter, so compared to the individual network analysis, the hierarchical network analysis would have more stable estimation. More specifically, if there exist a network which is very small, the parameter estimation can have huge standard error; however, the hierarchical structure stablize the estimations of the small network by sharing information of all networks together. In order to check this shrinkage effects from hierarchical modeling, an individual network analysis and a hierarchical network analysis will be compared with real data.

The data set was borrowed from Pitts and Spillance (2009) and it is provided by R *HLSM* library. The respsonse network is advice seeking activities among instructors in 15 different schools and one edge-wise binary covariate is used: it takes 1 if instructors $i$ and $j$ teach the same grade in the school, or it takes 0.

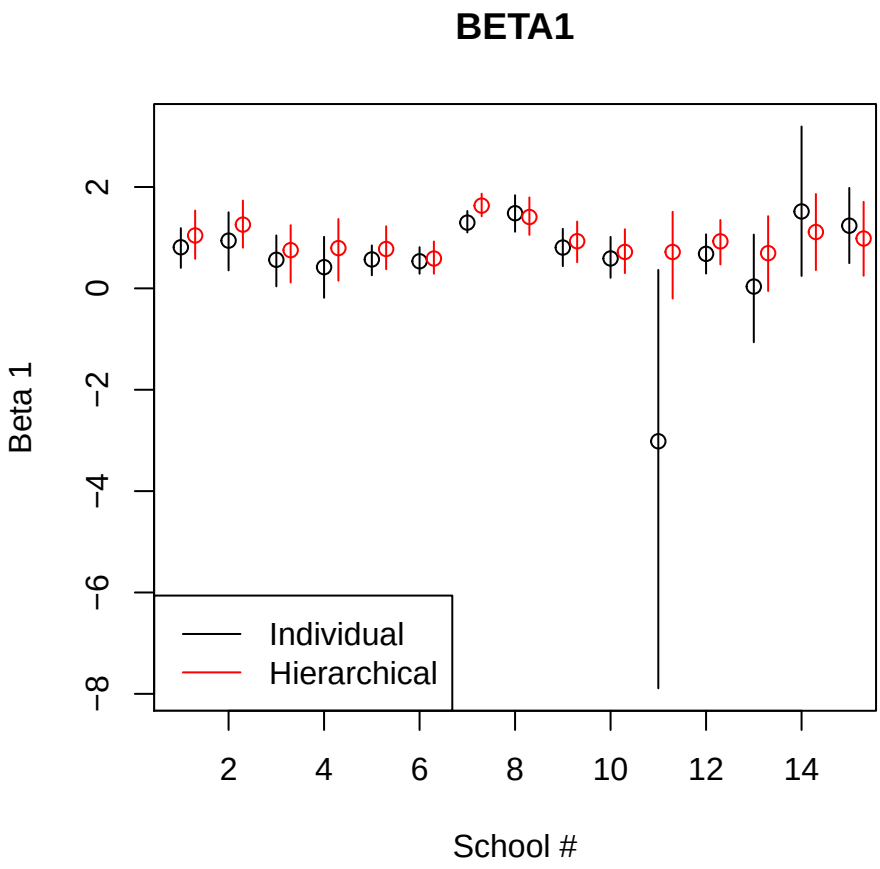


Figure 7: Comparison with Real Data for Individual and Hierarchical Modeling

For individual network analysis, the R library, Additive Multiplicative Effect Model (AMEN) by Hoff et. al. (2015) is used. In the plot above, individual and hierarchical network analysis for covariate effects are compared: the black and red lines are 95 percentiles of the estimates for individual and hierarchical network analysis, respectively. As can be seen, the notable difference between two analyises are school 11. In the data, school 11 has the smallest size and its estimation of covariate effect has large variability without hierarchical structure; however, with the hierarchical structure, the variabilty of the estimate decreased notably, and the estimate of the mean shifted toward the grand mean of the covariate effect implying obvious shrinkage effect.

For more clear effects, I also simulated data with various size of networks.

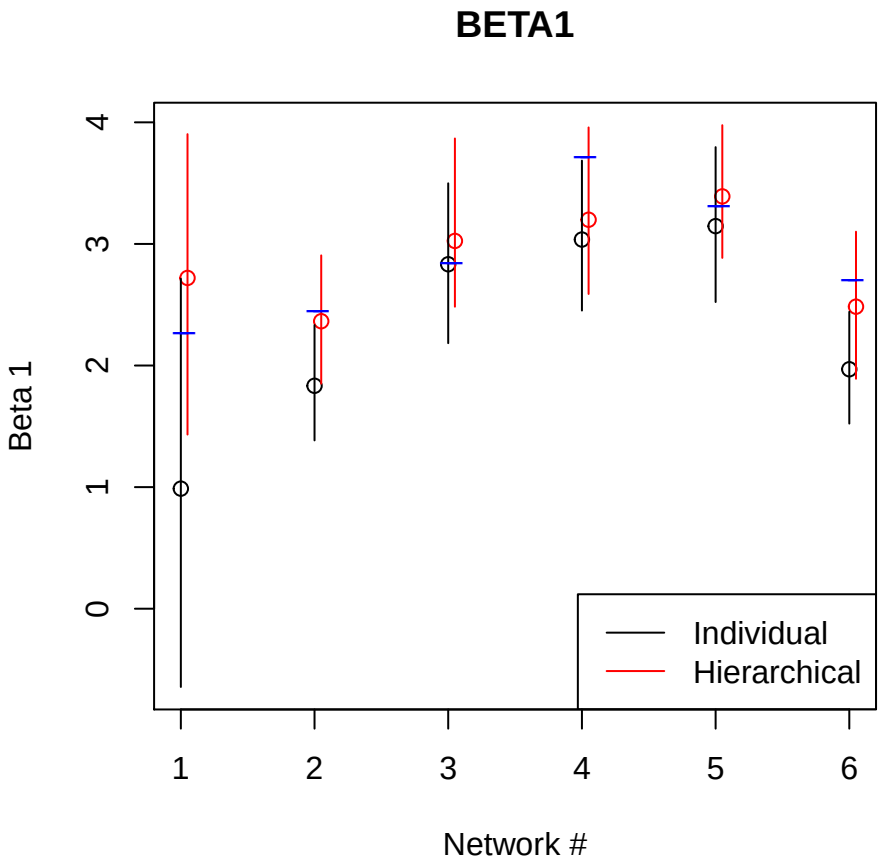


Figure 8: Comparison with Simulated Data for Individual and Hierarchical Modeling

The simulated dataset has 6 networks: Network 1 is the smallest one with 5 nodes, and the other 5 networks have 20 nodes. As discussed previously, we can expect shrinkage effect by hierarchical modeling and smaller variance in parameter estimation for Network 1. The plot above displays 95 percentiles of $\beta_1$s for each network and the blue dashed lines are the true value. Given the assumption that there truely exists hierarchical structure, the individual network analysis not only misses some of the true $\beta_1$s, but also larger variance for small network. Furthermore, given that all mean posterior estimates of networks in hierarchical model have much more similar values than those in individual model, we can easily conclude that the shrinkage exists.

## Discussion

The hierarhical network modeling allows one obtain more stable parameter estimations by adding the assumption that the parameters for each network are generated by some hyperparameters. This assumption is plausible because the networks in a dataset may behave similarly each other. For example, in educational research settings, networks of multiple schools are usually collected, and it is reasonable to consider the schools are somehow conneted each other and the actors of each school behave similarly. And hierarchical network analysis allows one make the connection of those networks.

For further research, the computation speed should be optimized by parallel computing or vectorization. Moreover, with dyadic dependence, $\rho$, every Gibbs sampler simulation was done by fixing the other element. This may lead to bad mixing in MCMC with high autocorrelation. So it would be much faster and nicer in mixing if the MCMC can be done with the joint distribution itself.