Computational Techniques in Latent Network Models

Lily Chou, Ben Draves, Nathan Josephs, Kelly Kung

December 12, 2018

1 Introduction

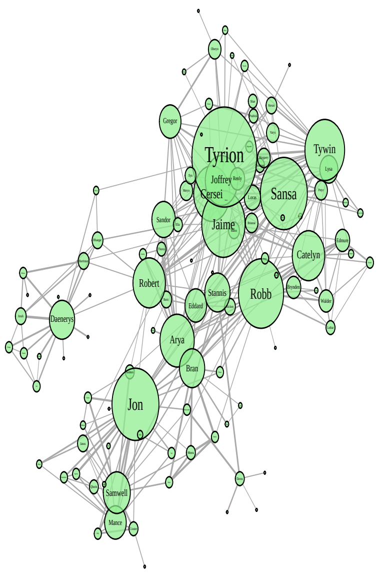
Game of Thrones is a popular HBO TV series adapted from George R.R. Martin's best-selling book series *A Song of Ice and Fire*. The medieval fantasy epic describes the stories of powerful families - kings and queens, knights and renegades, liars and honest men - playing a deadly game for control of the Seven Kingdoms of Westeros and to sit atop the Iron Throne. Conspiracy and deception, power and exile, blood and tears run through the plot, sewing together characters with various backgrounds including royals and peasants, as well as ice zombies and dragons. As the plot develops with each book release, readers wonder where the storyline leads. Within the Seven Kingdoms, enemies become friends and vice-versa, all the while winter spreads as the battle of ice and far draws nearer. For many characters, it is clear who is good and who is evil, but such moral assignments are skewed by the readers' biases. Here, we propose to put aside our personal feelings and let the data decide.

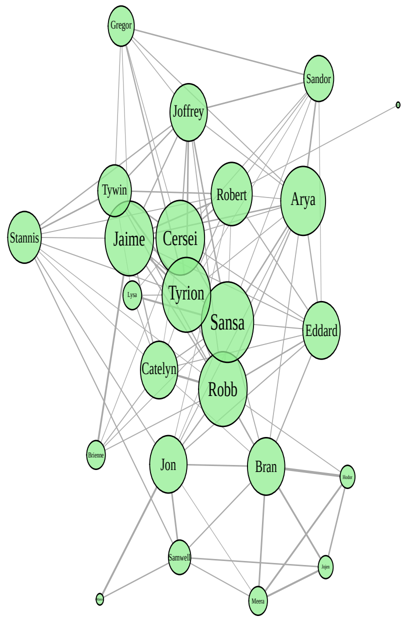
After discovering a dataset on the exchanges between the characters from the third book, *A Storm of Swords*, we start to wonder if, and what, information we can extract. In particular, how can we make inferences on character being good or bad. To address this, we turn to research of Peter Hoff on latent network models. In Section 2, we show how the relationships between characters from the book naturally arise as a network. Before fitting a model, we explain the latent network model framework in Section 3. We then present two methods for fitting our model in Section 4 and then compare the results of the two methods in Section 5. Finally, in Section 6 we end with a discussion on the implications of our findings, as well as possible future work. We provide our code in the Appendix.

2 Data

Due to its global fame, *Game of Thrones* has been studied in many different contexts, especially in network analysis. Therefore, there are many readily available datasets. In our project, we use the dataset from Beveridge and Shan 2016, which contains information about characters' interactions in the third book of the series. In this case, an *interaction* occurs if the characters' names appear within fifteen words of one another. This could mean that the characters interacted with each other, conversed with each other, or were generally mentioned together by another means. There is also a column that contains the number of times each pair interacts with one another. Using this dataset, we constructed a weighted network using the number of interactions as weights. Here, the nodes represent the characters and the edges represent the interactions. We use an adjacency matrix, A, to represent the network, where the $a_{i,j}$ element represents the number of times the characters interacted with each other. Note that this means if $a_{i.j} = 0$, there are no recorded interactions between character i and j based on how an *interaction* is defined. Although the original dataset is intended as a directed network, we treat it as an undirected network in order to simplify our models.

After transforming the dataset, our network G contains $N_V(G) = 107$ nodes and $N_E(G) = 352$ edges which means it is quite sparse since it only contains approximately 6.20% of $\binom{N_V(G)}{2} = 5,671$ possible edges. Figure 2 shows the network described. In order to account for the sparsity of our network, we consider a subnetwork which only contains pairs of characters with at least 100 interactions. We chose a cutoff of 100 interactions because we want to focus our analysis on only the main characters. Looking at the distribution of the weighted degree, we see that 77.6% of the characters had fewer than 100 interactions. Therefore, it





makes sense to use this cutoff to limit our analysis to only the main characters. By doing so, our new network G' contains $N_V(G') = 24$ nodes and $N_E(G') = 102$ edges. Here, we see that the network now contains 38.2% of 276 possible edges, which is a more appropriate level of density for our analysis. Everything that follows is done on this subnetwork G'. Figure 2 shows this subnetwork G', and indeed, we recognize the main characters remain in our network.

3 Models

3.1 Model Formulation

A modern and popular class of network models is the *Latent Network Model*. This model, first introduced by Hoff, Raftery, and Handcock 2002, associates each vertex $v \in V$ with a latent variable $Z_v \in \mathcal{V}$. Using these variables, the probability of an E_{ij} being in the network is given by $p_{ij} = \kappa(Z_i, Z_j)$ where $\kappa : \mathcal{V} \times \mathcal{V} \to [0, 1]$. Following this work, the authors extend these models to the mixed-model, regression framework where nodal attribute vectors x_V are used in modeling the probability of an edge presense (Krivitsky et al. 2009). For simplicity we do not consider nodal attributes in this work and instead focus our inference on the latent variables associated with each network. Formally, we model the presence of an edge given our latent variables as

$$\text{logit } \mathbb{P}(Y_{ij} = 1 | Z) = ||Z_i - Z_j|| + \epsilon_{ij}$$

where

$$Z_i \overset{ind}{\sim} \sum_{k=1}^{G} \lambda_k \text{MVN}_d(\mu_k, \sigma_k^2 I_d)$$

Putting priors over μ_k and σ_k^2, as well as introducing the latent variable K representing the group from which Z is drawn, we have the following model formulation:

$$\begin{aligned}
Y_{ij}|Z_i, Z_j &\overset{ind}{\sim} \text{Bern}\Big[\text{logit}^{-1}\big(\|Z_i - Z_j\|\big)\Big] \\
Z_i|K_i = k_i &\overset{ind}{\sim} MVN(\mu_{k_i}, \sigma^2_{k_i} I_d) \\
K &\overset{iid}{\sim} \text{Multinoulli}\big(G, \lambda\big) \\
\lambda_k &\overset{iid}{\sim} \frac{1}{G} \\
\mu_k &\overset{iid}{\sim} \text{MVN}_d(0, I_d) \\
\sigma^2_k &\overset{iid}{\sim} \text{Inv}\chi^2_1
\end{aligned}$$

With these, we can write the complete likelihood which we will use in both our fitting procedures. We let $\theta = (\mu, \sigma^2, K, \lambda)$ denote our nuisance parameters. Then, we have

$$\begin{aligned}
\mathcal{L}(Z, \theta; Y) &= \prod_{i<j} \mathbb{P}(Y_{ij}|Z_i, Z_j)\mathbb{P}(Z_i|K_i, \mu_{k_i}, \sigma^2_{k_i})\mathbb{P}(Z_j|K_j, \mu_{k_j}, \sigma^2_j) \\
&\quad \times \mathbb{P}(K_i|\lambda_i)\mathbb{P}(\lambda_i)\mathbb{P}(\mu_{k_i})\mathbb{P}(\sigma^2_{k_i})\mathbb{P}(K_j)\mathbb{P}(\mu_{k_j})\mathbb{P}(\sigma^2_{k_j}) \\
&= \prod_{i<j} \Big(\text{logit}^{-1}\big(\|Z_i - Z_j\|\big)\Big)^{Y_{ij}} \Big(1 - \text{logit}^{-1}\big(\|Z_i - Z_j\|\big)\Big)^{1-Y_{ij}} \\
&\quad \times f_{MVN_d}(\mu_{k_i}, \sigma^2_{k_i} I_d) \times \lambda_i \times \frac{1}{G} \times f_{MVN_d}(0, I_d) \times f_{\text{Inv}\chi^2_2} \\
&\quad \times f_{MVN_d}(\mu_{k_j}, \sigma^2_{k_j} I_d) \times \lambda_j \times \frac{1}{G} \times f_{MVN_d}(0, I_d) \times f_{\text{Inv}\chi^2_2} \\
&\propto \prod_{i<j} \Big(\text{logit}^{-1}\big(\|Z_i - Z_j\|\big)\Big)^{Y_{ij}} \Big(1 - \text{logit}^{-1}\big(\|Z_i - Z_j\|\big)\Big)^{1-Y_{ij}} \\
&\quad \times \frac{1}{(\sigma^2_{k_i})^{1/2}} \exp\Big\{-\frac{1}{2\sigma^2_{k_i}}(Z_i - \mu_{k_i})^T(Z_i - \mu_{k_i})\Big\} \; \frac{1}{(\sigma^2_{k_j})^{1/2}} \exp\Big\{-\frac{1}{2\sigma^2_{k_j}}(Z_j - \mu_{k_j})^T(Z_j - \mu_{k_j})\Big\} \\
&\quad \times \exp\Big\{-\frac{1}{2}\mu^T_{k_i}\mu_{k_i}\Big\} \exp\Big\{-\frac{1}{2}\mu^T_{k_j}\mu_{k_j}\Big\} \times \frac{1}{(\sigma^2_{k_i})^2} \exp\Big\{-\frac{1}{\sigma^2_{k_i}}\Big\} \frac{1}{(\sigma^2_{k_j})^2} \exp\Big\{-\frac{1}{\sigma^2_{k_j}}\Big\} \times \lambda_i \times \lambda_j
\end{aligned}$$

4 Computational Methods

4.1 EM

One method for finding the latent variables Z_i for each node is the Expectation-Maximization (EM) algorithm. Unfortunately, the likelihood given in Section 3.1 cannot easily be handled with EM. In particular, finding $\mathbb{E}_{Z_i, Z_j|Y,\theta}\Big[l(Z, \theta; Y)\Big]$ is difficult. One solution is to sample and use Monte Carlo methods to estimate the expectation in the E-step and then proceed with the M-step. This is the so-called Monte Carlo EM (MCEM) method.

Instead, we simplify our model to make EM more analytically tractable. In Section 4.2, we fit the full model using MCMC. Here, we fit two simplified models: an unweighted model where Y indicates the presence of an edge and a weighted model where now Y represents the number of interactions between characters.

4.1.1 Unweighted Model

We first fit an unweighted model on the presence of an edge in the network, which is defined by:

$$\begin{aligned}
Y_{ij}|p_{ij} &\overset{ind}{\sim} \text{Bern}(p_{ij}) \\
p_{ij} &\overset{iid}{\sim} \text{Beta}(\alpha, \beta)
\end{aligned}$$

where we define $p_{ij} \equiv 2 - 2 * \text{logit}^{-1}(d_{ij})$ where d_{ij} are latent distances defined in Section 3.1 and are calculated by $d_{ij} = logit(1 - \frac{p_{ij}}{2})$. We define p_{ij} in this manner to ensure that an infinite distance results in zero probability of interaction and a distance of zero results in a probability of one. Then the likelihoods are

$$\begin{aligned}
\mathcal{L}(p, \alpha, \beta; Y) &= \prod_{i<j} p_{ij}^{Y_{ij}} \big(1 - p_{ij}\big)^{1-Y_{ij}} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p_{ij}^{\alpha-1} \big(1 - p_{ij}\big)^{\beta-1} \\
l(p, \alpha, \beta; Y) &= \sum_{i<j} Y_{ij} \log\Big(\frac{p_{ij}}{1 - p_{ij}}\Big) + \log(1 - p_{ij}) \\
&\qquad + \log\Gamma(\alpha+\beta) - \log\Gamma(\alpha) - \log\Gamma(\beta) \\
&\qquad + (\alpha - 1)\log p_{ij} + (\beta - 1)\log(1 - p_{ij})
\end{aligned}$$

For the E-step, we observe that the beta distribution is conjugate to the binomial and since p_{ij} only depends on the data through Y_{ij}, we have the following likelihood. Note we just suppress our parameters $\theta = \{\alpha, \beta\}$.

$$\begin{aligned}
p_{ij} | Y_{ij}, \theta &\propto \text{Bern}(p_{ij}) \times \text{Beta}(\alpha, \beta) \\
&= \text{Beta}\Big(\alpha + Y_{ij}, \beta + 1 - Y_{ij}\Big)
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\pi_{ij} &\equiv \mathbb{E}_{p_{ij}|Y_{ij},\theta}\big[\log p_{ij}\big] = \Psi\Big(\alpha + Y_{ij}\Big) - \Psi\Big(\alpha + \beta + 1\Big) \\
\eta_{ij} &\equiv \mathbb{E}_{p_{ij}|Y_{ij},\theta}\big[\log(1 - p_{ij})\big] = \Psi\Big(\beta + 1 - Y_{ij}\Big) - \Psi\Big(\alpha + \beta + 1\Big)
\end{aligned}$$

where Ψ is the digamma function. Thus

$$\begin{aligned}
Q\big(\theta; \theta^{(t)}\big) &\equiv \mathbb{E}_{p|Y,\theta^{(t)}}\Big[l(p; Y, \theta)\Big] \\
&= \sum_{i<j} (Y_{ij} + \alpha - 1)\mathbb{E}_{p_{ij}|Y_{ij},\theta^{(t)}}[\log p_{ij}] + (\beta - Y_{ij})\mathbb{E}_{p_{ij}|Y_{ij},\theta^{(t)}}\big[\log(1 - p_{ij})\big] \\
&\qquad + \log\Gamma(\alpha+\beta) - \log\Gamma(\alpha) - \log\Gamma(\beta) \qquad \text{(E)}
\end{aligned}$$

For the M-step, we simply take the partial derivatives of $Q\big(\theta; \theta^{(t)}\big)$ with respect to α and β. Note that we will need an approximate solution in both cases since the digamma function prevents us from finding an analytic solution. For this, we use the Newton-Raphson Method.

$$\begin{aligned}
\frac{\partial Q\big(\theta; \theta^{(t)}\big)}{\partial \alpha^{(t)}} &= \sum_{i<j} \mathbb{E}_{p_{ij}|Y_{ij},\theta^{(t)}}\big[\log p_{ij}\big] + \Psi\Big(\alpha^{(t+1)} + \beta^{(t)}\Big) - \Psi(\alpha^{(t+1)}) = 0 \\
&\implies \Psi\Big(\alpha^{(t+1)} + \beta^{(t)}\Big) - \Psi(\alpha^{(t+1)}) = -\frac{\sum_{i<n} \mathbb{E}_{p_{ij}|Y_{ij},\theta^{(t)}}\big[\log p_{ij}\big]}{\binom{n}{2}} \qquad \text{(MU1)} \\
\frac{\partial Q\big(\theta; \theta^{(t)}\big)}{\partial \beta^{(t)}} &= \sum_{i<j} \mathbb{E}_{p_{ij}|Y_{ij},\theta^{(t)}}\big[\log(1 - p_{ij})\big] + \Psi\Big(\alpha^{(t+1)} + \beta^{(t+1)}\Big) - \Psi(\beta^{(t+1)}) = 0 \\
&\implies \Psi\Big(\alpha^{(t+1)} + \beta^{(t+1)}\Big) - \Psi(\beta^{(t+1)}) = -\frac{\sum_{i<n} \mathbb{E}_{p_{ij}|Y_{ij},\theta^{(t)}}\big[\log(1 - p_{ij})\big]}{\binom{n}{2}} \qquad \text{(MU2)}
\end{aligned}$$

For the Newton-Raphson Method, we first define

$$
\begin{aligned}
g(\alpha) &= \Psi\big(\alpha + \beta^{(t)}\big) - \Psi(\alpha) + \sum_{i<j} \frac{\mathbb{E}_{p_{ij}|Y_{ij},\theta^{(t)}}\big[\log p_{ij}\big]}{\binom{n}{2}} \\
&\implies g'(\alpha) = \Psi'\big(\alpha + \beta^{(t)}\big) - \Psi'(\alpha) \\
g(\beta) &= \Psi\big(\alpha^{(t+1)} + \beta\big) - \Psi(\beta) + \sum_{i<j} \frac{\mathbb{E}_{p_{ij}|Y_{ij},\theta^{(t)}}\big[\log(1-p_{ij})\big]}{\binom{n}{2}} \\
&\implies g'(\beta) = \Psi'\big(\alpha^{(t+1)} + \beta\big) - \Psi'(\beta)
\end{aligned}
$$

Using these equations, we can find the updates of α and β:

$$
\alpha^{(t+1)} = \alpha^{(t)} - \frac{\Psi\big(\alpha^{(t)} + \beta^{(t)}\big) - \Psi(\alpha^{(t)}) + \sum_{i<j} \frac{\mathbb{E}_{p_{ij}|Y_{ij},\theta^{(t)}}\big[\log p_{ij}\big]}{\binom{n}{2}}}{\Psi'\big(\alpha^{(t)} + \beta^{(t)}\big) - \Psi'(\alpha^{(t)})} \tag{α_U}
$$

$$
\beta^{(t+1)} = \beta^{(t)} - \frac{\Psi\big(\alpha^{(t+1)} + \beta^{(t)}\big) - \Psi(\beta^{(t)}) + \sum_{i<j} \frac{\mathbb{E}_{p_{ij}|Y_{ij},\theta^{(t)}}\big[\log(1-p_{ij})\big]}{\binom{n}{2}}}{\Psi'\big(\alpha^{(t+1)} + \beta^{(t)}\big) - \Psi'(\beta^{(t)})} \tag{β_U}
$$

Algorithm 1 shows the algorithm for the Unweighted Model.

Algorithm 1: EM for simplified latent network unweighted model

```
1 LNM EM ($G, tol$);
  Input  : Graph $G$
           Tolerance $tol$
  Output: Nuisance Parameters $\alpha^*$, $\beta^*$
           Latent Probability Estimates $\hat{p}$
           Latent Distance Estimates $\hat{d}$
2 Initialize $Q^{(0)}$ repeat
3     E: calculate $\pi^{(t)}$, $\eta^{(t)}$;
4     M: update $\alpha^{(t+1)}$ using ($\alpha_U$);
5     update $\beta^{(t+1)}$ using ($\beta_U$);
6     calculate $Q(\theta, \theta^{(t+1)})$
7 until $|\frac{Q(\theta^{(t+1)},\theta^{(t)}) - Q(\theta^{(t)},\theta^{(t)})}{Q(\theta^{(t)},\theta^{(t)})}| < tol$;
8 return $\alpha^*$, $\beta^*$, $\hat{p} = e^{\pi^*}$, $\hat{d} = logit^{-1}(1 - \frac{e^{\pi^*}}{2})$; where $\alpha^*, \beta^*, \pi^*$ are converged values
```

4.1.2 Weighted Model

From the unweighted model, we were able to find convergent values for the latent distances for pairs of nodes in the network. However, because the unweighted model only depends on the existence of an edge between the nodes, there are only two possible values for these distances. Since we cannot infer anything from these, we turn to the weighted model for the latent distances where the weights correspond to the number of interactions between the characters.

In this weighted model, we fit the following model:

$$
\begin{aligned}
Y_{ij}|\lambda_{ij} &\overset{ind}{\sim} \text{Pois}(\lambda_{ij}) \\
\lambda_{ij} &\overset{iid}{\sim} \text{Gamma}(\alpha, \beta)
\end{aligned}
$$

where now, $\lambda_{ij} = \frac{1}{d_{ij}}$ and d_{ij} is defined as above and are calculated by $d_{ij} = \frac{1}{\lambda_{ij}}$. Again, we define λ_{ij} in this manner in order to have smaller distances result in higher mean number of interactions and greater distances result in lower mean number of interactions. Thus the likelihood is:

$$\begin{aligned}
\mathcal{L}(\lambda, \alpha, \beta; Y) &= \prod_{i<j} \frac{\lambda_{ij}^{Y_{ij}} e^{-\lambda_{ij}}}{Y_{ij}!} \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda_{ij}^{\alpha-1} e^{-\beta\lambda_{ij}} \\
l(\lambda, \alpha, \beta; Y) &= \sum_{i<j} \log \lambda_{ij} \big(Y_{ij} + \alpha - 1) - \lambda_{ij}(1+\beta) - \log(Y_{ij}!) + \alpha \log(\beta) - \log \Gamma(\alpha)
\end{aligned}$$

Since λ_{ij} only depend on Y_{ij} from the data and since the Poisson and Gamma are conjugate, we know that

$$\begin{aligned}
\lambda_{ij} | Y_{ij}, \theta &\propto Pois(\lambda_{ij}) \times Gamma(\alpha, \beta) \\
&= Gamma(\alpha + Y_{ij}, \beta + 1)
\end{aligned}$$

Note we suppress our parameters $\theta = \{\alpha, \beta\}$. From here, we define the following quantities:

$$\begin{aligned}
\pi_{ij} &\equiv \mathbb{E}_{\lambda_{ij}|Y_{ij},\theta}\big[\lambda_{ij}\big] = \frac{\alpha + Y_{ij}}{1+\beta} \\
\eta_{ij} &\equiv \mathbb{E}_{\lambda_{ij}|Y_{ij},\theta}\big[\log \lambda_{ij}\big] = \log(1+\beta) + \Psi(\alpha + Y_{ij})
\end{aligned}$$

and we have

$$\begin{aligned}
Q(\theta; \theta^{(t)}) &\equiv \mathbb{E}_{\lambda|Y,\theta^{(t)}}\left[l(\lambda; Y, \theta)\right] \\
&= \sum_{i<j} (Y_{ij} + \alpha - 1) \mathbb{E}_{\lambda_{ij}|Y_{ij},\theta^{(t)}}\big[\log \lambda_{ij}\big] \\
&\quad - (1+\beta)\mathbb{E}_{\lambda_{ij}|Y_{ij},\theta^{(t)}}\big[\lambda_{ij}\big] - \log(Y_{ij}!) + \alpha \log(\beta) - \log \Gamma(\alpha)
\end{aligned}$$

For the M-step, we take partial derivatives of $Q(\theta; \theta^{(t)})$ with respect to our parameters, and here, we see that we can update β directly, but again, we have to use Newton Raphson for the α.

$$\begin{aligned}
\frac{\partial Q(\theta; \theta^{(t)})}{\partial \beta^{(t)}} &= -\sum_{i<j} \mathbb{E}_{\lambda_{ij}|Y_{ij},\theta^{(t)}}\big[\lambda_{ij}\big] + \frac{\alpha^{(t+1)}}{\beta^{(t+1)}} = 0 \\
&\implies \beta^{(t+1)} = \frac{\alpha^{(t+1)}}{\frac{\sum_{i<j} \mathbb{E}_{\lambda_{ij}|Y_{ij},\theta^{(t)}}\big[\lambda_{ij}\big]}{\binom{n}{2}}} \qquad (\beta_W) \\
\frac{\partial Q(\theta; \theta^{(t)})}{\partial \alpha^{(t)}} &= \sum_{i<j} \mathbb{E}_{\lambda_{ij}|Y_{ij},\theta^{(t)}}\big[\log \lambda_{ij}\big] + \log(\beta^{(t+1)}) - \Psi(\alpha^{(t+1)}) = 0 \\
&\implies \Psi(\alpha^{(t+1)}) = \frac{\sum_{i<j} \mathbb{E}_{\lambda_{ij}|Y_{ij},\theta^{(t)}}\big[\log \lambda_{ij}\big] + \binom{n}{2}\log(\beta)}{\binom{n}{2}} \qquad \text{(MW1)}
\end{aligned}$$

For the Newton Raphson step, we define

$$\begin{aligned}
g(\alpha) &= \sum_{i<j} \mathbb{E}_{\lambda_{ij}|Y_{ij},\theta^{(t)}}\big[\log \lambda_{ij}\big] + \binom{n}{2} \log \beta^{(t)} - \binom{n}{2}\Psi(\alpha) \\
&\implies g'(\alpha) = -\binom{n}{2}\Psi'(\alpha)
\end{aligned}$$

Thus we have

$$\alpha^{(t+1)} = \alpha^{(t)} - \frac{\sum_{i<j} \mathbb{E}_{\lambda_{ij}|Y_{ij},\theta^{(t)}}\big[\log \lambda_{ij}\big] + \binom{n}{2}\log \beta^{(t)} - \binom{n}{2}\Psi(\alpha)}{-\binom{n}{2}\Psi'(\alpha)}$$

$$= \alpha^{(t)} - \frac{\Psi(\alpha^{(t)}) - \sum_{i<j} \frac{\mathbb{E}_{\lambda_{ij}|Y_{ij},\theta^{(t)}}\big[\log \lambda_{ij}\big]}{\binom{n}{2}} - \log \beta^{(t)}}{\Psi'(\alpha^{(t)})} \qquad (\alpha_W)$$

Algorithm 2 shows the algorithm for the Weighted Model.

Algorithm 2: EM for simplified latent network weighted model

```
1 LNM EM ($(G, tol)$);
  Input  : Graph $G$
           Tolerance $tol$
  Output: Nuisance Parameters $\alpha^*$, $\beta^*$
           Latent Mean Estimates $\hat{\lambda}$
           Latent Distance Estimates $\hat{d}$
2 Initialize $Q^{(0)}$ repeat
3     E: calculate $\pi^{(t)}$, $\eta^{(t)}$;
4     M: update $\beta^{(t+1)}$ using $(\beta_W)$;
5     update $\alpha^{(t+1)}$ using $(\alpha_W)$;
6     calculate $Q(\theta, \theta^{(t+1)})$
7 until $|\frac{Q(\theta^{(t+1)},\theta^{(t)}) - Q(\theta^{(t)},\theta^{(t)})}{Q(\theta^{(t)},\theta^{(t)})}| < tol$;
8 return $\alpha^*$, $\beta^*$, $\hat{\lambda} = \pi^*$, $\hat{d} = \frac{1}{\pi^*}$; where $\alpha^*, \beta^*, \pi^*$ are converged values
```

4.2 MCMC

4.2.1 Full Conditionals

Next, we turn to find the full conditional of each parameter in this likelihood function. At each iteration t of the Gibbs sampler, we condition on the current parameter vector $\theta^{(t)}$ and sample each parameter in stepwise fashion. We develop the full conditionals below.

First, note that all nodes i from group $K = k$ have latent variables $Z_i \overset{iid}{\sim} N(\mu_k, \sigma_k^2)$. Hence the group parameters μ_k and σ_k^2 can be sampled by considering only the nodes i from group k.

$$f_{\mu_k|\theta^{(t)},Y}(\mu_k|\theta^{(t)}, Y) \propto \prod_{k_i=k} \exp\left\{-\frac{1}{2\sigma_{k_i}^2}(Z_i - \mu_{k_i})^T(Z_i - \mu_{k_i})\right\} \exp\left\{-\frac{1}{2}\mu_{k_i}^T\mu_{k_i}\right\}$$

$$\propto \exp\left\{\sum_{i=1}^{N_v} \mathbb{I}\{k_i = k\}\Big[-\frac{1}{2\sigma_{k_i}^2}(Z_i^T Z_i - 2Z_i^T\mu_{k_i} + \mu_{k_i}^T\mu_{k_i}) - \frac{1}{2}\mu_{k_i}^T\mu_{k_i}\Big]\right\}$$

$$\propto \exp\left\{\sum_{i=1}^{N_v} \mathbb{I}\{k_i = k\}\Big[-\frac{Z_i^T Z_i}{2\sigma_{k_i}^t} + \frac{Z_i^T\mu_{k_i}}{\sigma_{k_i}^2} - \left(\frac{1}{2\sigma_{k_i}^2} + \frac{1}{\sigma_{k_i}^2}\right)\mu_{k_i}^T\mu_{k_i}\Big]\right\}$$

$$\propto \exp\left\{\sum_{i=1}^{N_v} \mathbb{I}\{k_i = k\}\Big[-\frac{(\sigma_{k_i}^2 + 1)}{2\sigma_{k_i}^t}\left(\mu_{k_i} - \frac{Z_i}{(\sigma_{k_i}^2 + 1)}\right)^T\left(\mu_{k_i} - \frac{Z_i}{(\sigma_{k_i}^2 + 1)}\right)\Big]\right\}$$

Thus for all $k \in \{1, \ldots, G\}$, we arrive at the following distribution for $\mu_k|\theta^{(t)}, Y$:

$$\mu_k|\theta^{(t)}, Y \sim f_{MVN_d}\left(\sum_{i=1}^{N_v} \mathbb{I}\{k_i = k\} \frac{Z_i^{(t)}}{(\sigma_{k_i}^2)^{(t)} + 1}, \sum_{i=1}^{N_v} \mathbb{I}\{k_i = k\} \frac{(\sigma_{k_i}^2)^{(t)}}{(\sigma_{k_i}^2)^{(t)} + 1} I_d\right)$$

Now, we turn to finding the full conditional for the latent variance parameters $\sigma_{k_i}^2$.

$$\begin{aligned}
f_{\sigma_{k_i}^2|\theta,Y}(\sigma_{k_i}^2|\theta, Y) &\propto \prod_{K_i=k} {\sigma_{K_i}^2}^{-\frac{1}{2}} \exp\left\{\frac{1}{2\sigma_{k_i}}(Z_i - \mu_{k_i})^T(Z_i - \mu_{k_i})\right\} (\sigma_{k_i}^2)^{-\frac{c}{2}-1} \exp\left\{-\frac{1}{2\sigma_{k_i}^2}\right\} \\
&\propto \prod_{K_i=k} (\sigma_{k_i}^2)^{(-\frac{c}{2}-\frac{1}{2})-1} \exp\left\{-\frac{1}{2\sigma_{k_i}^2}\big((Z_i - \mu_{k_i})^T(Z_i - \mu_{k_i}) + 1\big)\right\} \\
&\propto (\sigma_{k_i}^2)^{\left((\frac{-c-1}{2}ng - ng + 1) - 1\right)} \exp\left\{-\frac{1}{2\sigma_{k_i}^2} \sum_{K_i=k} \big((Z_i - \mu_{k_i})^T(Z_i - \mu_{k_i}) + 1\big)\right\}
\end{aligned}$$

Let $n_g = \sum \mathbb{I}_{\{k_i=K\}}$ and $SS_g + n_g = \sum_{K_i=k}\big((Z_i - \mu_{k_i})^T(Z_i - \mu_{k_i}) + 1\big)$. Then by using the posterior distributions

$$\begin{aligned}
\nu_{post} &= (c+1)n_g + 2(n_g - 1) \\
\tau_{post}^2 &= \frac{SS_g + n_g}{(c+1)n_g + 2(n_g - 1)}
\end{aligned}$$

we arrive at the following distribution for $\sigma_k^2|\theta^{(t)}, Y$:

$$\sigma_k^2|\theta^{(t)}, Y \sim \text{Inv}\Gamma(\frac{c}{2}, \frac{1}{2}) \overset{D}{=} \tau^2\ \nu\ \text{Inv}\chi_c^2$$

Next, we find the conditional probabilities for group assignment:

$$\mathbb{P}(K_i = k|\theta, Y) \propto \lambda_k f_{MVN_d(\mu_k, \sigma_k^2)}(Z_i)$$

Since K is Multinoulli, we arrive at the following probability by recognizing they must normalize to unity:

$$\begin{aligned}
\mathbb{P}(K_i = k|\theta, Y) &= \frac{\lambda_k f_{MVN_d(\mu_k,\sigma_k^2)}(Z_i)}{\sum_{g=1}^G \lambda_g f_{MVN_d(\mu_g,\sigma_g^2)}(Z_i)} \\
&= \frac{f_{MVN_d(\mu_k,\sigma_k^2)}(Z_i)}{\sum_{g=1}^G f_{MVN_d(\mu_g,\sigma_g^2)}(Z_i)} \qquad (\lambda^{(t)})
\end{aligned}$$

Finally, we look at the conditional for the latent variable Z_i:

$$f_{Z_i|\theta,Y}(Z_i|\theta, Y) \propto \prod_{j\neq i} \Big(\text{logit}^{-1}\big(\|Z_i - Z_j\|\big)\Big)^{Y_{ij}} \Big(1 - \text{logit}^{-1}\big(\|Z_i - Z_j\|\big)\Big)^{1-Y_{ij}} \exp\left\{-\frac{1}{2\sigma_{k_i}^2}(Z_i - \mu_{k_i})^T(Z_i - \mu_{k_i})\right\}$$

Since this is not a distribution we know how to sample from, we use a Metropolis-Hastings step. Here, we choose a symmetric proposal, which simplifies the subsequent rejection ratio:

$$\begin{aligned}
q(Z_*|\theta^{(t)}, Y) &\sim MVN_d(0, I_d) \\
R(Z^*, Z^{(t)}) &= \frac{f_{Z|\theta,Y}(Z^*|\theta^{(t)}, Y) q(Z^{(t)}|\theta^{(t)}, Y)}{f_{Z|\theta,Y}(Z^{(t)}|\theta^{(t)}, Y) q(Z^*|\theta^{(t)}, Y)} \\
&= \frac{f_{Z|\theta,Y}(Z^*|\theta^{(t)}, Y)}{f_{Z|\theta,Y}(Z^{(t)}|\theta^{(t)}, Y)}
\end{aligned}$$

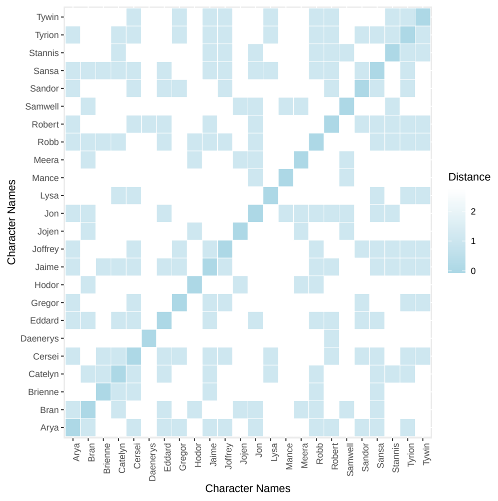
Algorithm 3: Gibbs sampler for latent network model

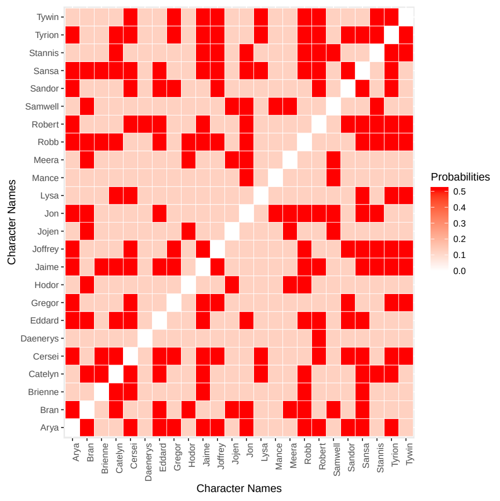
```
LNM MCMC $(G, N_k, d, ns)$;
Input  : Graph $G$
         Number of groups $N_k$
         Dimension of Latent Variable $d$
         Number of samples $ns$
Output: Posterior $p(Z|Y,\theta)$
Initialize parameters $\mu^{(0)}, {\sigma^2}^{(0)}, \lambda^{(0)}, K^{(0)}, Z^{(0)}$;
for $t = 2, \dots, ns$ do
    for $k = 1, \dots, N_k$ do
        sample $\mu_k|\theta^{(t)}, Y \sim MVN_d\left(\sum_{i=1}^{N_v} \mathbb{I}\{k_i = k\} \frac{Z_i^{(t-1)}}{(\sigma^2_{k_i})^{(t-1)}+1}, \sum_{i=1}^{N_v} \mathbb{I}\{k_i = k\} \frac{(\sigma^2_{k_i})^{(t-1)}}{(\sigma^2_{k_i})^{(t-1)}+1} I_d\right)$;
    end
    for $k = 1, \dots, N_k$ do
        sample $\sigma^2_k|\theta^{(t)}, Y \sim \left(1 + \sum_{i=1}^{N_v} \mathbb{I}\{k_i = k\}(Z_i^{(t-1)} - \mu_k^{(t)})^T (Z_i^{(t-1)} - \mu_k^{(t)})\right) \mathrm{Inv}\chi^2_{1+d\sum_{i=1}^{N_V} \mathbb{I}\{k_i = k\}}$ ;
    end
    for $i = 1, \dots, N_v$ do
        sample $K_i \sim$ Multinoulli$(G, \lambda^{(t)})$;
    end
    for $i = 1, \dots, N_v$ do
        sample $Z_i^* \sim MVN_d(0, I_d)$;
        $R(Z_i^*, Z_i^{(t)}) = \min\left(1, \frac{f_{Z|\theta,Y}(Z_i^*|\theta^{(t)}, Y, Z_{[-1]})}{f_{Z|\theta,Y}(Z_i^{(t)}|\theta^{(t)}, Z_{[-1]})}\right)$ ;
        sample $U \sim \mathcal{U}(0,1)$;
        if $U \leq R(Z_i^*, Z_i^{(t)})$ then
            $Z_i^{(t+1)} = Z_i^*$;
            else
                $Z_i^{*(t+1)} = Z_i^{(t)}$
            end
        end
    end
end
```

5 Results

5.1 EM

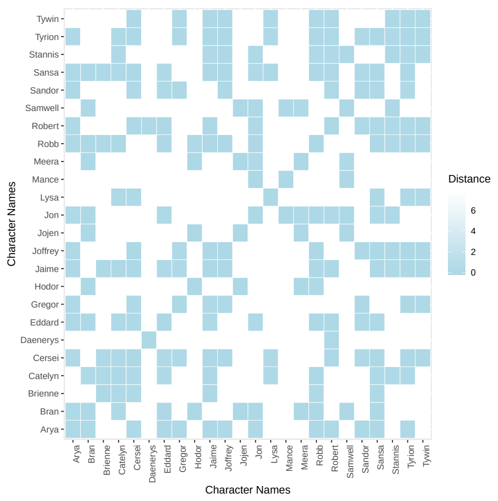
We see that using the EM algorithm for the unweighted model is a good initial attempt in modeling the interactions between characters. From the algorithm, we obtained estimates of probabilities of edges and latent distances between characters, and using these values, we were able to identify some closely related characters. Figure 5.1 shows heat maps that depicts the modeled relationships for pairs of characters.

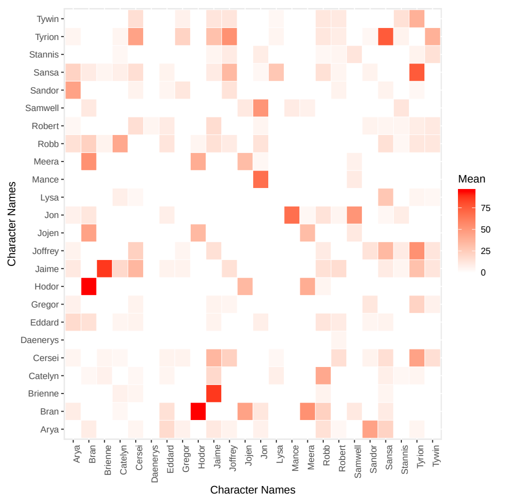




The first heat map is constructed using the latent distances and the darker the color, the closer the characters are. The second heat map is constructed using the probabilities of edges between characters, and again, the darker the color, the higher the probability of an edge between the characters. From these heat maps, we do see that some close relationships, like Bran and Hodor, are picked up in both cases. However, it is important to note that in this unweighted model, there is a one-to-one relationship between the latent distances and the presence of an edge and so the EM algorithm only provides estimates of probabilities and distances for when there is an edge and when there isn't an edge between the characters. Thus, we see that these results can be improved.

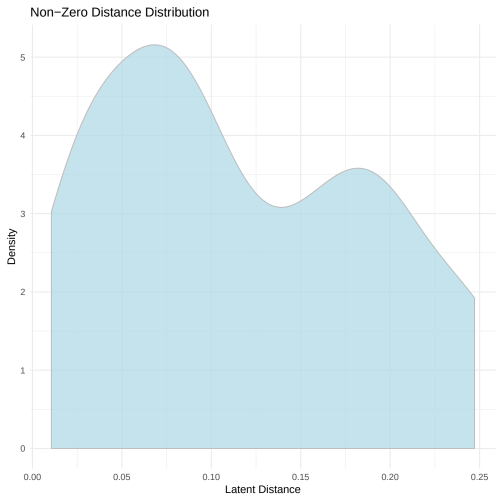
1. Need to incorperate edges weights for better performance
2. Heatmaps showing better fit that the unweighted model

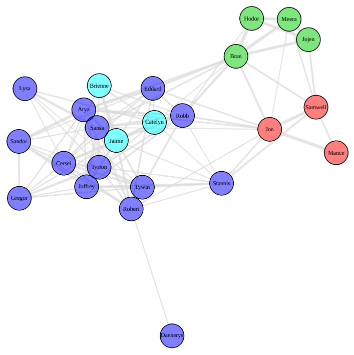




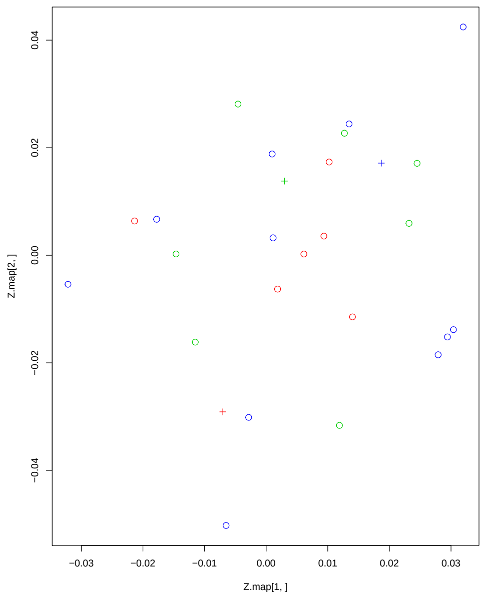
1. We see that the density of the distances show bimodal behavior

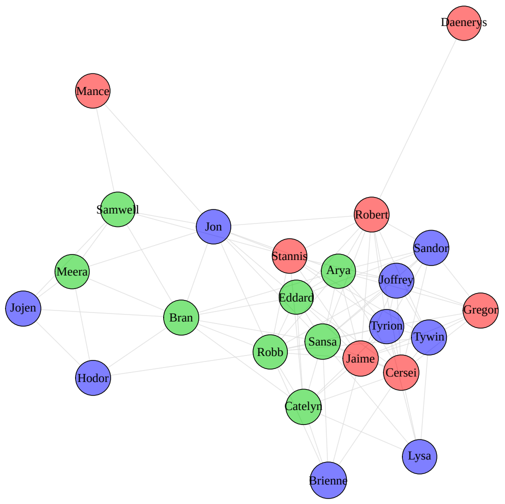
2. Spectral clustering detects nontrivial clusters in the network





5.2 MCMC





6 Conclusion

It is clear from this work that latent network models are a useful tool for drawing inferences about a network. In this project, we demonstrate their utility despite the fact that we simplify the model when using EM and neglect block updates with MCMC. We suggest fully following the model development of Hoff for optimal results. Though EM yielded sensible clusters, we are essentially just performing spectral clustering on the observed Laplacian since the weighted EM model is highly parametrized and cannot estimate vertex level latent variables. On the other hand, our MCMC model uses the dichotomized network where an edge is simply any number of interactions, which clearly is a suppression of valuable information. We believe that fitting a Poisson model for the weighted adjacency matrix using MCMC would yield optimal results. Moreover, block updates are probably necessary as our current implementation suffers from ignoring the latent group dependencies. These changes are obvious improvements that should be made.

Additionally, there is a lot of future work that can be accomplished in this area. *A Storm of Swords* is an illustrative choice of books as it is the third and midpoint of all releases, so characters have developed complex relationships. It would be interesting to perform the same analysis across each book and map a character's group status across time. Moreover, there are opportunities to consider the networks from each book as a dynamic network, which is a popular area of research. We hope this project is a nice foundation for future inference on *Game of Thrones* data.

7 References

References

Beveridge, Andrew and Jie Shan (2016). "Network of thrones". In: *Math Horizons* 23.4, pp. 18–22.

Hoff, Peter D, Adrian E Raftery, and Mark S Handcock (2002). "Latent Space Approaches to Social Network Analysis". In: *Journal of the American Statistical Association* 97.460, pp. 1090–1098. DOI: 10.1198/016214502388618906.

Krivitsky, Pavel N. et al. (2009). "Representing degree distributions, clustering, and homophily in social networks with latent cluster random effects models". In: *Social Networks* 31.3, pp. 204 –213. ISSN: 0378-8733.

Appendix

A Estimation Maximization Code

```r
#----------------------------------------
#
#               EM.LNM.U
#
#----------------------------------------

#pij -> dij
logit <- function(p) log(p/(1 - p))
distance.u <- function(p) logit(1 - p/2)

#newton raphson
nr.u <- function(x, y, s, nE, tol = 10e-4){
  #x is parameter to update
  #y is other parameter in beta dist
  #pi or eta
  #nE is number of edges
  repeat{
    x.new <- x - ((digamma(x + y) -  digamma(x) +  sum(s)/nE) /(trigamma(x
        + y) - trigamma(x))) #nr update
    if(abs((x.new - x)/x)<tol) break #convergence check
    x <- x.new #update new parameter est
  }
  return(x.new)
}

#EM
LNM.EM.U <- function(A, tol = 10e-4, no.iters = 1000){

  #An unweighted adj matrix

  #create Y vector ------------------------------------
  Y <- as.numeric(A[upper.tri(A)])
  nE <- length(Y)

  #Initialize parameter values -------------------------
  alpha <- 1 #Prior beta values from uniform
  beta <- 1 #Prior beta values from uniform
  Q <- 0 #initialize Q
  iter <- 1

  #iterate until convergence
  repeat{

    #E Step
    pi <- digamma(alpha + Y) - digamma(alpha + beta + 1) # log(p_ij)
    eta <- digamma(beta + 1 - Y) - digamma(alpha + beta + 1) # log(1-p_ij)

    #M Step
    alpha <- nr.u(alpha, beta, pi, nE)
    beta <-  nr.u(beta, alpha, eta, nE)

    #Check convergence
    Q.new <- sum((Y + alpha - 1)*pi
                 + (beta - Y)*eta
                 + log(gamma(alpha + beta)) - log(gamma(alpha)) - log(
                    gamma(beta))
                 )

    if(iter > no.iters || (abs((Q.new - Q) / Q) < tol)) break
    Q <- Q.new
    iter <- iter + 1
  }

  list(alpha = alpha, beta = beta, pi = pi, eta = eta, d = distance.u(exp(
     pi)), no.iter = iter)
}

#----------------------------------------
#
#               EM.LNM.W
#
#----------------------------------------

#pij -> dij
distance.w <- function(l) 1/l

#newton raphson
nr.w <- function(alpha, beta, eta, nE, tol = 10e-4){
  #lapha is parameter to update
  #beta, eta is other parameters in Gamma
  #nE is number of edges
  repeat{
    alpha.new <- alpha - (digamma(alpha) - sum(eta)/nE - log(beta))/(
       trigamma(alpha)) #nr update
    if(abs((alpha.new - alpha)/alpha)<tol) break #convergence check
    alpha <- alpha.new #update new parameter est
  }
  return(alpha.new)
}

#EM
LNM.EM.W <- function(W, tol = 10e-4, no.iters = 1000){

  #An unweighted adj matrix

  #create Y vector -------------------------------------
  Y <- as.numeric(W[upper.tri(W)])
  nE <- length(Y)

  #Initialize parameter values -------------------------
  alpha <- 1 #Prior beta values from uniform
  beta <- 1 #Prior beta values from uniform
  Q <- 0 #initialize Q
  iter <- 1

  #iterate until convergence
  repeat{

    #E Step
    pi <- (alpha + Y)/(beta + 1) # lambda_ij
    eta <- log(1 +beta) + digamma(alpha + Y) # log(lambda_ij)

    #M Step
    beta <-  (nE * alpha) / (sum(pi))
    alpha <- nr.w(alpha, beta, eta, nE)


    #Check convergence
    Q.new <- sum((Y + alpha - 1)*eta
                 -(1 + beta)*pi
                 - log(factorial(Y))
                 + alpha * log(beta) - log(gamma(alpha))
    )

    if(iter > no.iters || (abs((Q.new - Q) / Q) < tol)) break
    Q <- Q.new
    iter <- iter + 1
  }
  list(alpha = alpha, beta = beta, pi = pi, d = distance.w(pi), no.iter =
     iter)
}
```

B Markov Chain Monte Carlo Code

```
LOGEPS <- log(.Machine$double.eps / 2)

lse <- function (x) {
  m <- max(x); x <- x - m
  m + log(sum(exp(x[x > LOGEPS])))
}

lse2 <- function (x, y) {
  m <- pmax(x, y); d <- -abs(x - y)
  ifelse(d < LOGEPS, m, m + log(1 + exp(d))) }

rlcat <- function (n, l) {
  l <- Reduce(lse2, l, accumulate = TRUE) # "cumlse"
  l <- l - l[length(l)] # normalize
  findInterval(log(runif(n)), l) + 1
}

LNM.MCMC <- function(G, Nk = 2, d = 2, ns = 10000) {
  library(MASS)
  library(invgamma)

  Nv <- nrow(G)

  # Initialize
  mu <- array(dim = c(Nk, d, ns)); mu[, , 1] <- 0
  sigma <- matrix(nrow = Nk, ncol = ns); sigma[, 1] <- 1
  lambda <- matrix(nrow = Nk, ncol = ns); lambda[, 1] <- -log(Nk)
  K <- matrix(nrow = Nv, ncol = ns); K[, 1] <- sample.int(Nk, Nv, replace
    = TRUE)
  Z <- array(dim = c(Nv, d, ns)); Z[, , 1] <- 0

  # Updates
  for (t in 2:ns) {

    # Mu
    for (k in 1:Nk) {
      Zg <- apply(matrix(Z[K[, t - 1] == k, , t - 1], ncol = d), 2, mean)
      Zg <- ifelse(is.na(Zg), 0, Zg) # in case no one is in group k
      ng <- sum(K[, t - 1] == k)
      mu[k, , t] <- mvrnorm(n = 1
                          , mu = ( ng * Zg ) / ( ng + sigma[k, t - 1] )
                          , Sigma = ( (sigma[k, t -1 ]) /  (ng + sigma[k
                              , t -1]) ) * diag(d))
    }

    # Sigma
    for (k in 1:Nk) {
      ng <- sum(K[, t - 1] == k)
      SS_Zg <- sum(apply(matrix(Z[K[, t - 1] == k, , t - 1] - mu[k, , t],
          ncol = d), 2, crossprod))
      sigma[k, t] <- (1 + SS_Zg) * rinvchisq(1, 1 + ng*d)
    }
    # Group K
    for (i in 1:Nv) {
      lambda_g <- sapply(1:Nk, FUN = function(k) {
        C <- chol(sigma[k, t] * diag(d))
        y <- backsolve(C, (Z[i, , t - 1] - mu[k, , t]), transpose = TRUE)
        - (1/2) * log(2 * pi) - sum(log(diag(C))) - sum(y^2) / 2
      })

      lambda[, t] <- lambda_g - lse(lambda_g)
      K[i, t] <- rlcat(1, lambda[, t])
    }

    # Latent variable Z
    Z[, , t] <- Z[, , t - 1]
    for (i in 1:Nv) {

      Zstar <- mvrnorm(1, mu = rep(0, d), Sigma = diag(d))

      C <- chol(sigma[K[i, t - 1], t] * diag(d))
      y <- backsolve(C, (Z[i, , t - 1] - mu[K[i, t], , t]), transpose =
         TRUE)
      y.star <- backsolve(C, (Zstar - mu[K[i, t], , t]), transpose = TRUE)
      logR <- (-(1/2) * log(2 * pi) - sum(log(diag(C)))- sum(y^2) / 2) - #
          p(Z*)
        (-(1/2) * log(2 * pi) - sum(log(diag(C))) - sum(y^2) / 2)         #
            p(Z^(t-1))

      if (logR >= 0 || log(runif(1)) < logR) Z[i, , t] <- Zstar
    }
  }
  return(list(mu = mu, sigma = sigma, lambda = lambda, K = K, Z = Z))
}
```