Implementing the Social Relations and AMEN model in Stan

Adam M. Lauretig

3/31/2020

Introduction

In a recent review article, Peter Hoff discussed the Additive and Multiplicative Effects Network (AMEN) model, a Bayesian hierarchical model, developed by Minhas, Hoff, and Ward for modeling network data. This model can be understood as a hierarchical model with a more structured covariance, to account for the network dependence between observations. While there is currently an implementation in R, I wanted to explore building this model up in Stan, a probabilistic programming language, and taking advantage of variational inference to fit models more quickly.

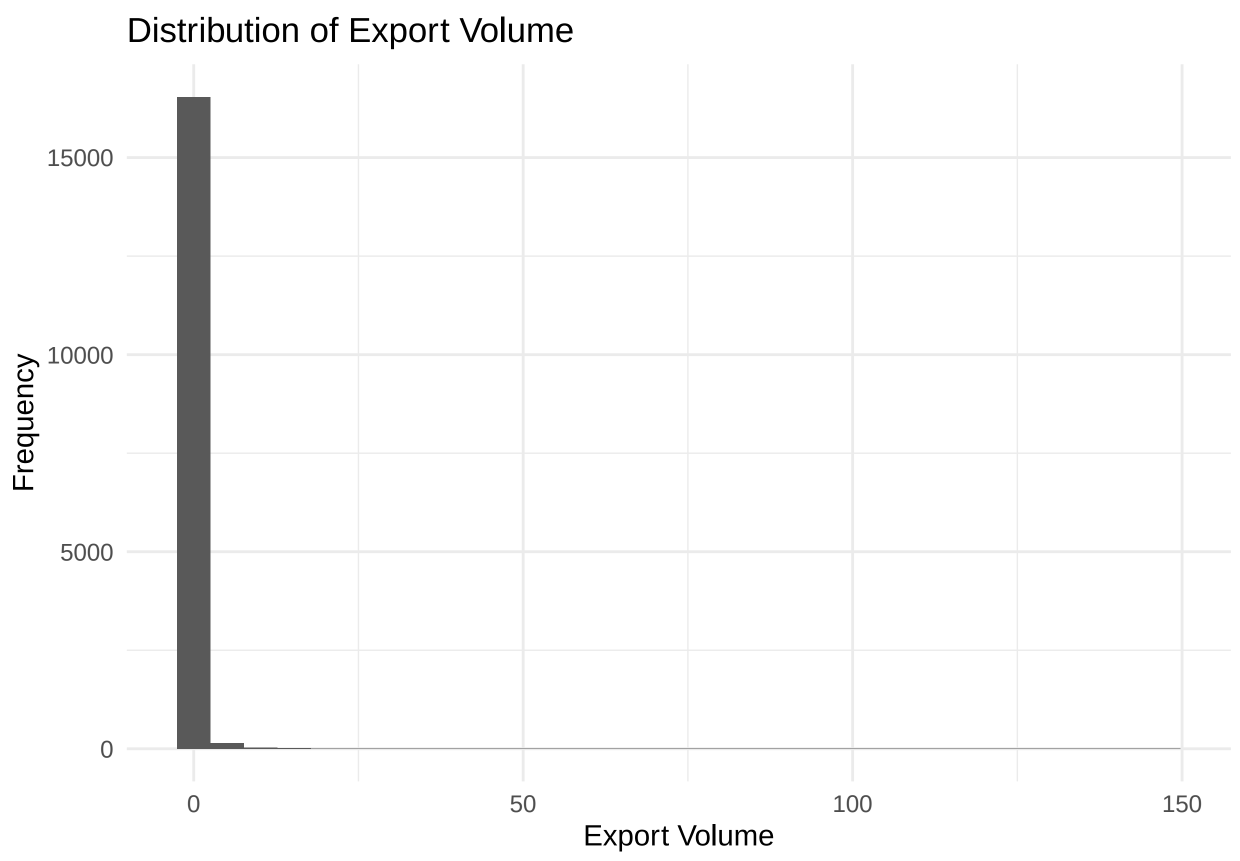
These models were all fit in stan using the `vb()` function, which meant they were fit quickly. Stan is fast, and provides a variety of diagnostic tools, though final work should probably use dynamic HMC/NUTS sampling.

In the remaining sections, I gather some example data, and a fixed effects model, a social relations model, and an AMEN model to trade data from the 1990s.

Dataset

I use a dataset included in the `amen` package to demonstrate model-building: exports from one country to another (in billions of dollars), averaged over the 1990s. I convert the sociomatrix to a real-valued edgelist, with indicators for sending and receiving countries, dyads, and the direction of the dyad. Plotting the data, we see the long-tailed distribution often characteristic of network data.

```r
library(ggplot2)
library(amen)
library(rstan)


# A function to convert a matrix of pairs to an edgelist
# Assumes that rows == columns, but will check

matrix_to_edgelist <- function(sociomatrix_to_convert){

  if(nrow(sociomatrix_to_convert) != ncol(sociomatrix_to_convert)){
    stop("nrows != ncols")
  }

  all_nodes <- expand.grid(
    list(rownames(sociomatrix_to_convert), colnames(sociomatrix_to_convert)))
  all_nodes[, 1] <- as.numeric(all_nodes[, 1])
  all_nodes[, 2] <- as.numeric(all_nodes[, 2])
  all_nodes <- all_nodes[all_nodes[, 1] != all_nodes[, 2], ]

  lookup_table <- data.frame(
    node_names = rownames(sociomatrix_to_convert),
    idx = as.numeric(factor(rownames(sociomatrix_to_convert))))
  obs_values <- which(sociomatrix_to_convert != 0, arr.ind = TRUE)
  obs_values <- cbind(obs_values, sociomatrix_to_convert[obs_values])

   edgelist <- merge(
    all_nodes, obs_values, by.x = c("Var1", "Var2"),
    by.y = c("row", "col"), all.x = TRUE)

  edgelist$V3 <- ifelse(is.na(edgelist$V3), 0, edgelist$V3)
  edgelist <- cbind(
    edgelist,
    rep(seq.int(from = 1, to = sum(edgelist[, 1] < edgelist[, 2])), 2))
  edgelist <- cbind(edgelist, ifelse(edgelist[, 1] < edgelist[, 2], 1, 2))
  colnames(edgelist) <- c("sender", "receiver", "y", "dyad_id", "sr_indicator")
  # for dyad list include a (1, 2) indicator for send/receive
  # if s < r -> 1, else 2

  return(list(edgelist = edgelist, lookup_table = lookup_table,
       n_nodes = max(edgelist[, 1]), n_dyads = max(edgelist[, 4]),
       N = nrow(edgelist)))

}

data_for_stan <- matrix_to_edgelist(IR90s$dyadvars[, , 2]) # trade data

y_df <- data.frame(y = data_for_stan$edgelist$y)
ggplot(data = y_df, aes(x = y)) + geom_histogram() + theme_minimal() + labs(x = "Export Vol
```


Distribution of Export Volume
Frequency
15000
10000
5000
0
0
50
100
150
Export Volume

Fixed Effects Model

I begin with a simple fixed-effects model, where the outcome, $y_{i,j}$ is the exports from sender$_i$ to receiver$_j$.

$$y_{i,j} \sim \mathcal{N}(\alpha + \beta_{\text{sender}_i} + \beta_{\text{receiver}_j}, 1)$$
$$\alpha \sim \mathcal{N}(0, 5)$$

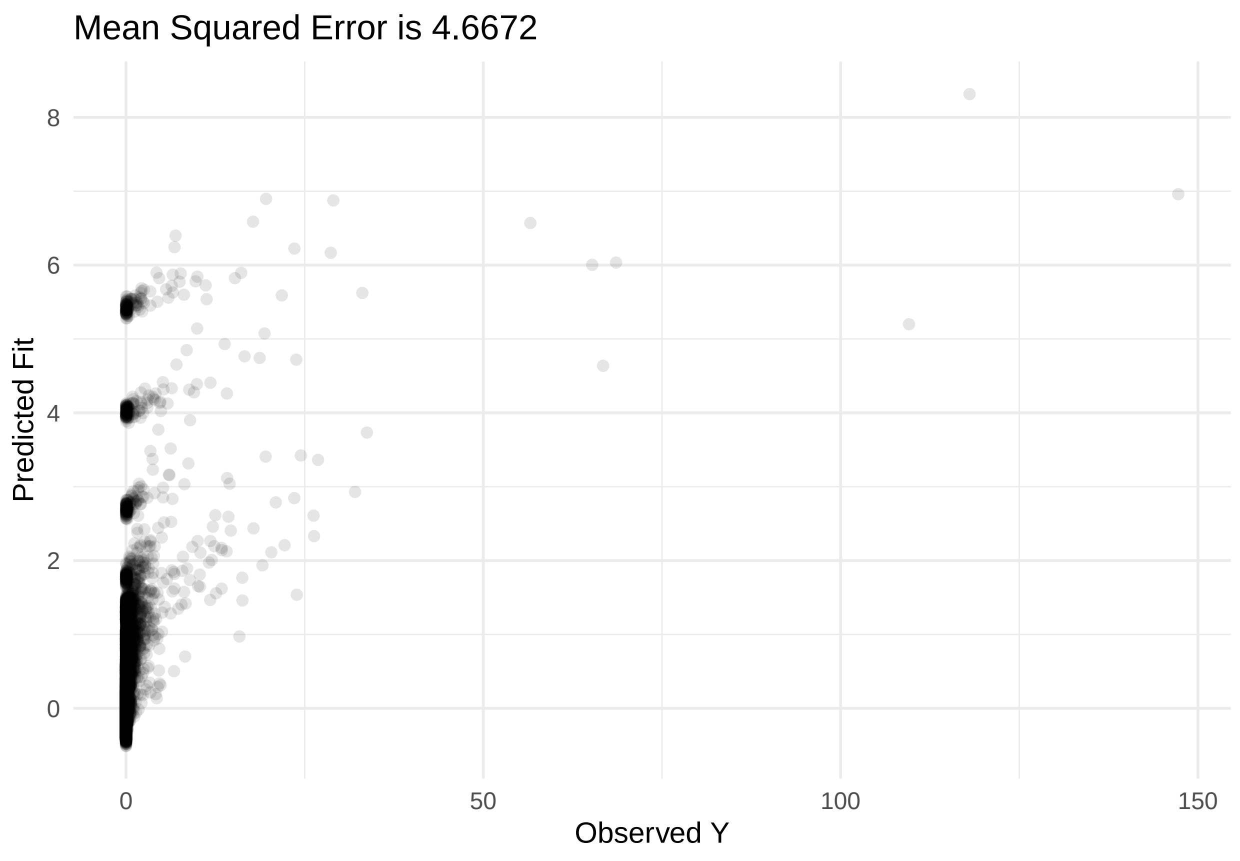
$$\beta_{\text{sender}_i} \sim \mathcal{N}(0, 5)$$

$$\beta_{\text{receiver}_j} \sim \mathcal{N}(0, 5)$$

```
data{
  int n_nodes ;
  int n_dyads ;
  int N; //total obs. should be n_dyads * 2
  int sender_id[n_dyads * 2] ; //indexing for sender
  int receiver_id[n_dyads * 2] ; //indexing for receivers
  real Y[n_dyads * 2] ; // outcome variable

}
parameters{
  real intercept ;
  vector[n_nodes] sender_beta; //fixed effects
  vector[n_nodes] receiver_beta;  //fixed effects
}

model{
  intercept ~ normal(0, 5) ;
  sender_beta ~ normal(0, 5) ;
  receiver_beta ~ normal(0, 5) ;

  for(n in 1:N){
  Y[n] ~ normal(intercept +
        sender_beta[sender_id[n]] + receiver_beta[receiver_id[n]], 1 );
  }
}
generated quantities{
  real Y_sim[N] ;
  for(n in 1:N){
    Y_sim[n] = normal_rng(intercept +
        sender_beta[sender_id[n]] + receiver_beta[receiver_id[n]], 1) ;
  }

}
```

Examining the posterior predictive check from the fixed effects model, we see a model with some clear fit problems, for example, in the scatterplot, the axes are different by an order of magnitude. In this model, the point predictions are the posterior means of outcomes simulated from the fitted model.

```r
library(bayesplot)
color_scheme_set("red")

m0_params <- extract(m0)
to_visualize <- data.frame(y = y_df$y, y_sim = colMeans(m0_params$Y_sim))
mse_0 <- mean((to_visualize$y - to_visualize$y_sim)^2)


#ppc_dens_overlay(y = y_df$y, yrep = m0_params$Y_sim)
ggplot(data = to_visualize, aes(x = y, y = y_sim)) +
  geom_point(alpha = .1) +
  theme_minimal() +
  labs(x = "Observed Y", y = "Predicted Fit", title = paste0("Mean Squared Error is ", rou
```



Social Relations Model

To improve model fit, we can incorporate covariance between sender and receiver. In concrete terms, this means that we assume that US exports covary with US imports. To model this, we turn to the *social relations model*, where the outcome, $y_{i,j}$ is the exports from sender$_i$ to receiver$_j$, however, we add additional structure to the covariances.

$$y_{i,j} \sim \mathcal{N}(\alpha + \beta_{\text{sender}_i} + \beta_{\text{receiver}_j}, 1)$$
$$\alpha \sim \mathcal{N}(0, 5)$$

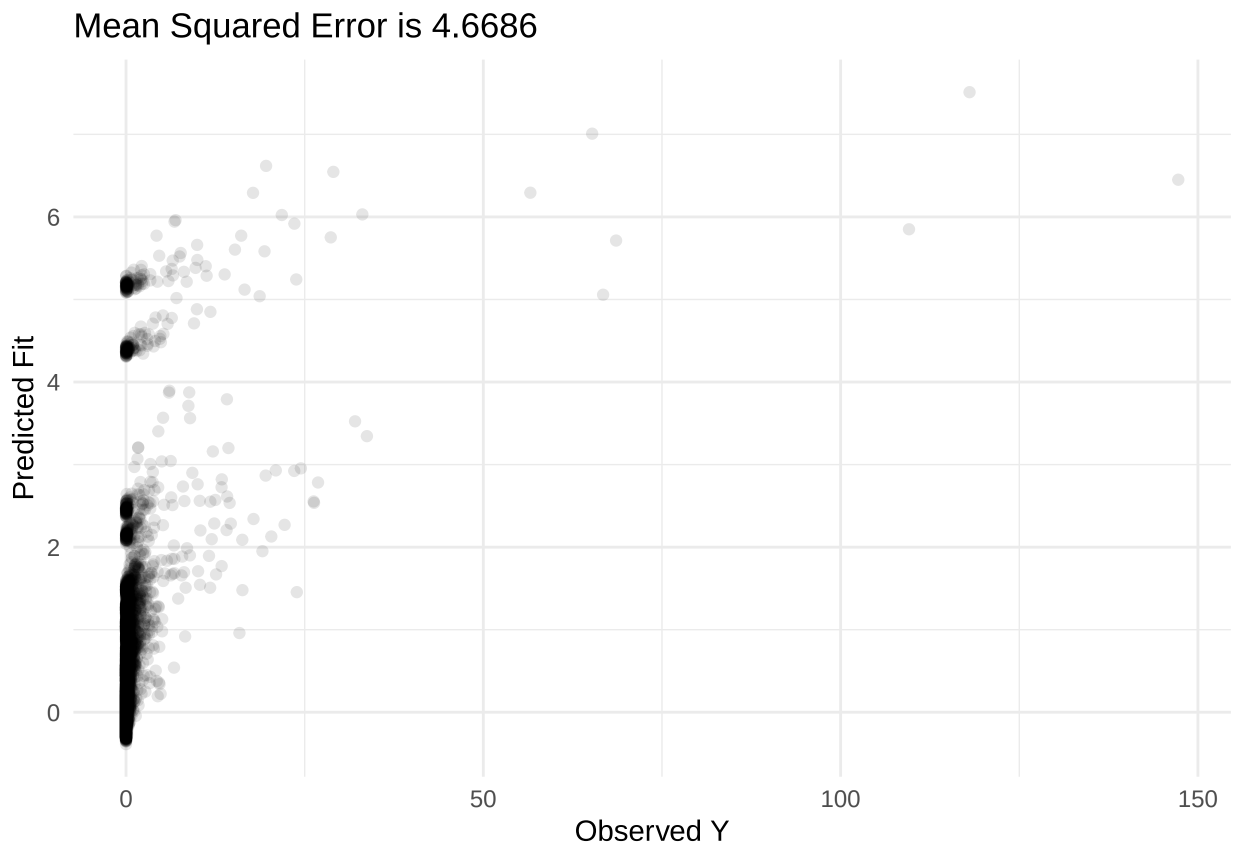
$$\boldsymbol{\beta}_{1\times 2} \sim \mathcal{MVN}(\mathbf{0}, \boldsymbol{\Sigma})$$
$$\boldsymbol{\Sigma} = \boldsymbol{\sigma}\boldsymbol{\Omega}_{\text{cholesky}}$$
$$\boldsymbol{\sigma} \sim \text{Gam}(1, 1)$$
$$\boldsymbol{\Omega}_{\text{cholesky}} \sim \mathcal{LKJ}(5)$$

In Stan, this is written as follows, with the non-centered parameterization used for the Multivariate normal:

```
data{
  int n_nodes ;
  int n_dyads ;
  int N; //total obs. should be n_dyads * 2
  int sender_id[n_dyads * 2] ;
  int receiver_id[n_dyads * 2] ;
  real Y[n_dyads * 2] ;

}
parameters{
  real intercept ;
  cholesky_factor_corr[2] corr_nodes; // correlation matrix
  vector<lower=0>[2] sigma_nodes; // sd
  matrix[2, n_nodes] z_nodes ; // for node non-centered parameterization
}
transformed parameters{
    matrix[n_nodes, 2] mean_nodes;
    mean_nodes = (diag_pre_multiply(
      sigma_nodes, corr_nodes) * z_nodes)'; // non-centered parameterization
}
model{
  intercept ~ normal(0, 5) ;
  to_vector(z_nodes) ~ normal(0, 1) ;
  corr_nodes ~ lkj_corr_cholesky(5) ;
  sigma_nodes ~ gamma(1, 1) ;

  for(n in 1:N){
  Y[n] ~ normal(intercept +
        mean_nodes[sender_id[n], 1] + mean_nodes[receiver_id[n], 2], 1 );
  }
}
generated quantities{
  real Y_sim[N] ;
  for(n in 1:N){
    Y_sim[n] = normal_rng(intercept +
        mean_nodes[sender_id[n], 1] + mean_nodes[receiver_id[n], 2], 1) ;
  }

}
```

Examining fit from this model, we see that fit is largely the same, likely because despite allowing for co-variance between sender and receiver effects, we are still imposing a strong additive relationship:

```r
library(bayesplot)
color_scheme_set("red")

m1_params <- extract(m1)
to_visualize1 <- data.frame(y = y_df$y, y_sim = colMeans(m1_params$Y_sim))
mse_1 <- mean((to_visualize1$y - to_visualize1$y_sim)^2)


#ppc_dens_overlay(y = y_df$y, yrep = m1_params$Y_sim)
ggplot(data = to_visualize1, aes(x = y, y = y_sim)) +
  geom_point(alpha = .1) +
  theme_minimal() +
  labs(x = "Observed Y", y = "Predicted Fit", title = paste0("Mean Squared Error is ", rou
```



Additive and Multiplicative Effects Network Model

To relax the assumption of additivity in the parameters, Peter Hoff suggests the Additive and Multiplicative Effects Network (AMEN) Model. This adds a latent multiplicative effect to capture a higher-order network representation. This effect, for a given observation $y_{i,j}$ is modeled by the inner product $\boldsymbol{u}_{\text{sender}_i} \cdot \boldsymbol{v}^\top_{\text{receiver}_j}$, where $\boldsymbol{u}$ and $\boldsymbol{v}$ both have $1 \times K$ dimensions. These are concatenated together, and their covariance is modeled with a $2K \times 2K$ covariance matrix.

$$y_{i,j} \sim \mathcal{N}(\alpha + \beta_{\text{sender}_i} + \beta_{\text{receiver}_j} + (\boldsymbol{u}_{\text{sender}_i} \cdot \boldsymbol{v}^{\top}_{\text{receiver}_j}), 1)$$
$$\alpha \sim \mathcal{N}(0, 5)$$

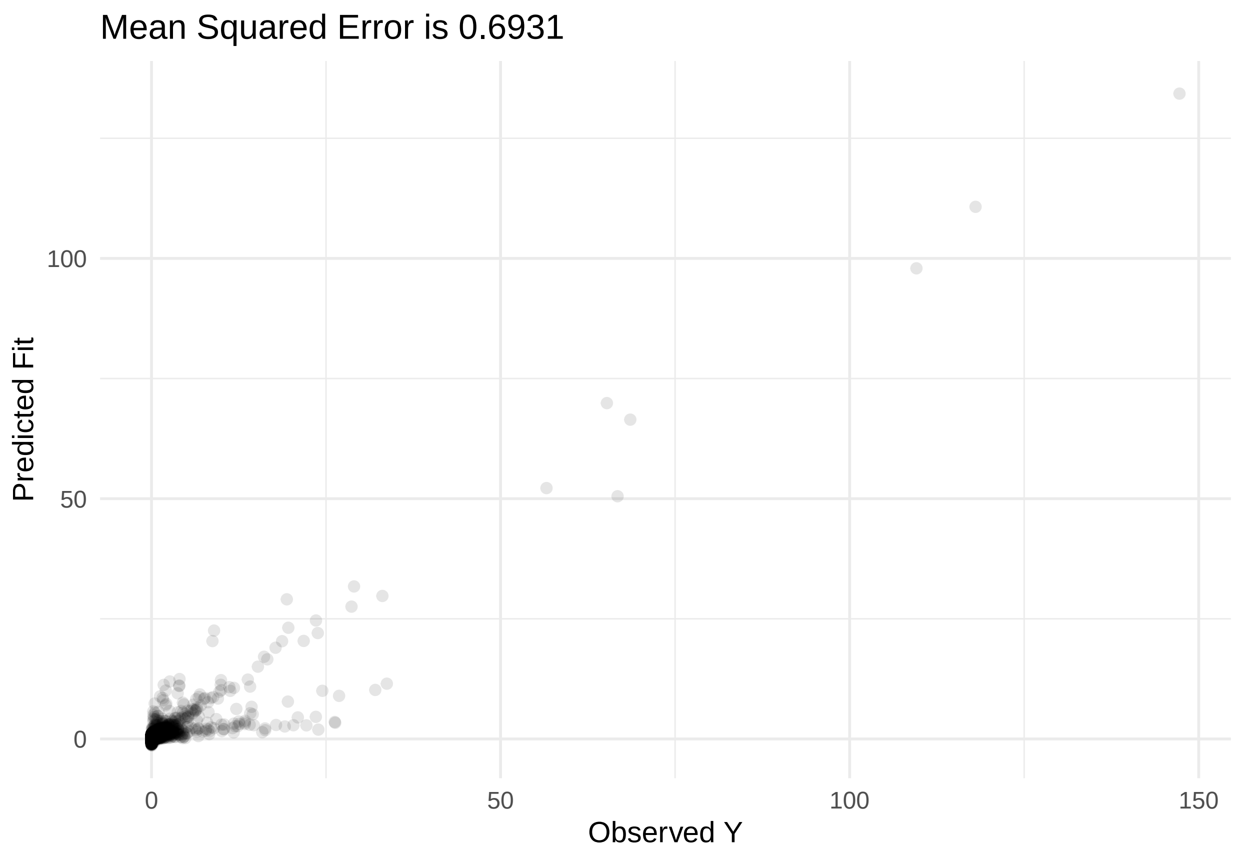
$$\boldsymbol{\beta}_{1\times 2} \sim \mathcal{MVN}(\mathbf{0}, \boldsymbol{\Sigma})$$
$$\boldsymbol{\Sigma} = \boldsymbol{\sigma}\boldsymbol{\Omega}_{\text{cholesky}}$$
$$\boldsymbol{\sigma} \sim \text{Gam}(1, 1)$$
$$\boldsymbol{\Omega}_{\text{cholesky}} \sim \mathcal{LKJ}(5)$$
$$c(\boldsymbol{u}_{\text{sender}_i}, \boldsymbol{v}^{\top}_{\text{receiver}_j}) \sim \mathcal{MVN}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{u},\boldsymbol{v}})$$
$$\boldsymbol{\Sigma}_{\boldsymbol{u},\boldsymbol{v}} = \boldsymbol{\sigma}_{\boldsymbol{u},\boldsymbol{v}}\boldsymbol{\Omega}_{\boldsymbol{u},\boldsymbol{v}_{\text{cholesky}}}$$
$$\boldsymbol{\sigma}_{\boldsymbol{u},\boldsymbol{v}} \sim \text{Gam}(1, 1)$$
$$\boldsymbol{\Omega}_{\boldsymbol{u},\boldsymbol{v}_{\text{cholesky}}} \sim \mathcal{LKJ}(5)$$

In stan, this is written as follows, with non-centered parameterizations for all multivariate normals:

```
data{
  int n_nodes ;
  int n_dyads ;
  int N; //total obs. should be n_dyads * 2
  int sender_id[n_dyads * 2] ;
  int receiver_id[n_dyads * 2] ;
  int K ; // number of latent dimensions
  real Y[n_dyads * 2] ;

}
parameters{
  real intercept ;
  cholesky_factor_corr[2] corr_nodes; // correlation matrix for additive noteeffects
  vector<lower=0>[2] sigma_nodes; // sd w/in nodes
  matrix[2, n_nodes] z_nodes ; // for node non-centered parameterization
  cholesky_factor_corr[K * 2] corr_multi_effects ; // correlation matrix for multiplicative
  vector<lower=0>[K * 2] sigma_multi_effects ; // sd
  matrix[K * 2, n_nodes] z_multi_effects; // Multi-effect non-centered term

}
transformed parameters{
    matrix[n_nodes, 2] mean_nodes; // node parameter mean
    matrix[n_nodes, K * 2] mean_multi_effects ; // multi-effect mean

    mean_nodes = (diag_pre_multiply(
      sigma_nodes, corr_nodes) * z_nodes)'; // sd *correlation
    mean_multi_effects = (diag_pre_multiply(
      sigma_multi_effects, corr_multi_effects) * z_multi_effects)'; // sd *correlation

}
model{
  intercept ~ normal(0, 5) ;

  //node terms
  to_vector(z_nodes) ~ normal(0, 1) ;
  corr_nodes ~ lkj_corr_cholesky(5) ;
  sigma_nodes ~ gamma(1, 1) ;

  // multi-effect terms
  to_vector(z_multi_effects) ~ normal(0, 1) ;
  corr_multi_effects ~ lkj_corr_cholesky(5) ;
  sigma_multi_effects ~ gamma(1, 1) ;

  for(n in 1:N){
  Y[n] ~ normal(intercept +
        mean_nodes[sender_id[n], 1] + mean_nodes[receiver_id[n], 2] +
        mean_multi_effects[sender_id[n], 1:K] *
        (mean_multi_effects[receiver_id[n], (K+1):(K*2)])',
        1 );
  }
}
```

```
generated quantities{
  real Y_sim[N] ;
  for(n in 1:N){
    Y_sim[n] = normal_rng(intercept +
        mean_nodes[sender_id[n], 1] + mean_nodes[receiver_id[n], 2] +
        mean_multi_effects[sender_id[n], 1:K] *
        (mean_multi_effects[receiver_id[n], (K+1):(K*2)])', 1) ;
  }

}
```

```r
library(bayesplot)
color_scheme_set("red")

m3_params <- extract(m3)
to_visualize3 <- data.frame(y = y_df$y, y_sim = colMeans(m3_params$Y_sim))
mse_3 <- mean((to_visualize3$y - to_visualize3$y_sim)^2)


#ppc_dens_overlay(y = y_df$y, yrep = m3_params$Y_sim)
ggplot(data = to_visualize3, aes(x = y, y = y_sim)) +
  geom_point(alpha = .1) +
  theme_minimal() +
  labs(x = "Observed Y", y = "Predicted Fit", title = paste0("Mean Squared Error is ", rou
```



We see that the AMEN model can much more effectively model the outcome variable, with a much lower mean squared error, and that the scale of the predicted outcomes is much closer to the actual outcomes.

Discussion

We have seen that the multiplicative effects model does a far better job of modeling the distribution of y than either the fixed effects model, or the social relations model. That the model fit changes so dramatically suggests the importance of taking network structure seriously in modeling. One further extension would involve incorporating covariates into the model, in order to take advantage of additional estimation when estimating model fit.