Supplementary material

Mplus-Code

```
Title:  SRM analysis with mean structure;
Data:   File is 'data.csv';
Variable:Names are YAM YAV YMA YMS YVA YVS YSM YSV;
          COUNT ARE YAM YAV YMA YMS YVA YVS YSM YSV;
Analysis: INTEGRATION = MONTECARLO(5000);
Model:
    ! Family Effect
    FE BY YAV YVA@1 YAM@1 YMA@1 YSM@1 YMS@1 YSV@1 YVS@1 (l1-l8);
    ! Actor Effects
    ActorM BY YMA YMS@1;
    ActorF BY YVA YVS@1;
    ActorS BY YSM YSV@1;
    ActorA BY YAM YAV@1;
    ! Partner Effects
    PartnerM BY YAM YSM@1;
    PartnerF BY YAV YSV@1;
    PartnerS BY YMS YVS@1;
    PartnerA BY YMA YVA@1;

    ! Generalized reciprocity
    FE WITH ActorM@0 ActorF@0 ActorS@0 ActorA@0 PartnerM@0 PartnerF@0
            PartnerS@0 PartnerA@0;
    ActorM WITH ActorF@0 ActorS@0 ActorA@0 PartnerF@0 PartnerS@0 PartnerA@0;
    ActorF WITH ActorS@0 ActorA@0 PartnerM@0 PartnerS@0 PartnerA@0;
    ActorS WITH ActorA@0 PartnerM@0 PartnerF@0 PartnerA@0;
    ActorA WITH PartnerM@0 PartnerF@0 PartnerS@0;
    PartnerM WITH PartnerF@0 PartnerS@0 PartnerA@0;
    PartnerF WITH PartnerS@0 PartnerA@0;
    PartnerS WITH PartnerA@0;

    ! Means
    [FE] (Fe);
    [ActorM ActorF ActorS ActorA] (AM AF AS AA);
    [PartnerM PartnerF PartnerS PartnerA] (PM PF PS PA);
    [ yam ](meanyam);
  [ yav ](meanyav);
  [ ysm ](meanyav);
  [ ysv ](meanyam);
  [ yma ](meanyma);
  [ yms ](meanyms);
  [ yva ](meanyms);
  [ yvs ](meanyma);


Model Constraint:
        0 = AM + AF + AS + AA;
        0 = PM + PF + PS + PA;
```

```
        0 = AS + AA - (PM + PF);
        0 = meanyam + meanyav;
        0 = meanyma + meanyms;
```

R-Code

```
##############
# R-packages #
##############

library(lavaan)
library(rjags)
library(MplusAutomation)

#########################
# Data-generating Model #
#########################

data.outcome <- '
# Family effect:
FE =~ 1*H_M_value + 1*H_V_value + 1*M_H_value + 1*M_Z_value
    + 1*V_H_value  + 1*V_Z_value + 1*Z_M_value + 1*Z_V_value

# Actor effects:
A.H =~ 1*H_M_value + 1*H_V_value
A.M =~ 1*M_H_value + 1*M_Z_value
A.V =~ 1*V_H_value + 1*V_Z_value
A.Z =~ 1*Z_M_value + 1*Z_V_value

# Partner effects:
P.H =~ 1*M_H_value + 1*V_H_value
P.M =~ 1*H_M_value + 1*Z_M_value
P.V =~ 1*H_V_value + 1*Z_V_value
P.Z =~ 1*M_Z_value + 1*V_Z_value

# Variance labels
FE  ~~ 0.05*FE
A.V ~~ 0.5*A.V
A.M ~~ 0.5*A.M
A.Z ~~ 0.5*A.Z
A.H ~~ 0.5*A.H

P.V ~~ 0.05*P.V
P.M ~~ 0.05*P.M
P.Z ~~ 0.05*P.Z
P.H ~~ 0.05*P.H

# Generalized reciprocity:
A.H ~~ (-0.001)*P.H
A.M ~~ 0.001*P.M
A.V ~~ 0.001*P.V
```

```
A.Z ~~ 0.001*P.Z


H_M_value ~~ 0*H_M_value
H_V_value ~~ 0*H_V_value
M_H_value ~~ 0*M_H_value
M_Z_value ~~ 0*M_Z_value
V_H_value ~~ 0*V_H_value
V_Z_value ~~ 0*V_Z_value
Z_M_value ~~ 0*Z_M_value
Z_V_value ~~ 0*Z_V_value

H_M_value ~ -0.010*1
H_V_value ~ 0.010*1
M_H_value ~ 0.015*1
M_Z_value ~ -0.015*1
V_H_value ~ -0.015*1
V_Z_value ~ 0.015*1
Z_M_value ~ 0.010*1
Z_V_value ~ -0.010*1

FE  ~ 1*1
A.H ~ -0.1*1
A.M ~ 0.1*1
A.V ~ -0.1*1
A.Z ~ 0.1*1
P.H ~ 0.1*1
P.M ~ 0.1*1
P.V ~ -0.1*1
P.Z ~ -0.1*1
'
# Continuous dyadic scores
test <- simulateData(data.outcome, sample.nobs=N)
# Creating count dyadic scores
for (j in 1:length(test$H_M_value)){
    test$H_M_count[j] <- rpois(1, exp(test$H_M_value[j]))
    test$H_V_count[j] <- rpois(1, exp(test$H_V_value[j]))
    test$M_H_count[j] <- rpois(1, exp(test$M_H_value[j]))
    test$M_Z_count[j] <- rpois(1, exp(test$M_Z_value[j]))
    test$V_H_count[j] <- rpois(1, exp(test$V_H_value[j]))
    test$V_Z_count[j] <- rpois(1, exp(test$V_Z_value[j]))
    test$Z_M_count[j] <- rpois(1, exp(test$Z_M_value[j]))
    test$Z_V_count[j] <- rpois(1, exp(test$Z_V_value[j]))
    }


##########################################################
# The social relations model for count data in rjags #
##########################################################
# Where x_IJ is a dyadic value with I = rater and J = ratee
# T = Target, M = Mother, F = Father, S = Sibling

# glm module
```

```
load.module("glm")

# Model
model_count_blocked_mean <- '
model{
# Likelihood function (count dyadic data)
for (i in 1:N) {
x_MS[i] ~ dpois(mu_RMS[i])
x_SM[i] ~ dpois(mu_RSM[i])
x_MT[i] ~ dpois(mu_RMT[i])
x_TM[i] ~ dpois(mu_RTM[i])
x_FS[i] ~ dpois(mu_RFS[i])
x_SF[i] ~ dpois(mu_RSF[i])
x_FT[i] ~ dpois(mu_RFT[i])
x_TF[i] ~ dpois(mu_RTF[i])
}

# defining the expected value of the dyadic measurements
# as a combination of the actor, partner
# and family effects (and residuals effect)
for (i in 1:N) {
mu_RMS[i] = exp(f_RMS + f_AM[i] + f_PS[i] + f_FE[i] )
mu_RSM[i] = exp(f_RSM + f_AS[i] + f_PM[i] + f_FE[i] )

mu_RMT[i] = exp(f_RMT + f_AM[i] + f_PT[i] + f_FE[i] )
mu_RTM[i] = exp(f_RTM + f_AT[i] + f_PM[i] + f_FE[i] )

mu_RFS[i] = exp(f_RFS + f_AF[i] + f_PS[i] + f_FE[i] )
mu_RSF[i] = exp(f_RSF + f_AS[i] + f_PF[i] + f_FE[i] )

mu_RFT[i] = exp(f_RFT + f_AF[i] + f_PT[i] + f_FE[i] )
mu_RTF[i] = exp(f_RTF + f_AT[i] + f_PF[i] + f_FE[i] )
}

# Distribution family effect
for(i in 1:N){
f_FE[i] ~ dnorm(mu_FE,tau_FE)
}
# Distributions actor and partner effects (bivariate normal) - multi_normal
for(i in 1:N){
f_mother[i,1:2]  ~ dmnorm(mu_M[1,1:2], Tau_M[1:2,1:2])
f_father[i,1:2]  ~ dmnorm(mu_F[1,1:2], Tau_F[1:2,1:2])
f_sibling[i,1:2] ~ dmnorm(mu_S[1,1:2], Tau_S[1:2,1:2])
f_target[i,1:2]  ~ dmnorm(mu_T[1,1:2], Tau_T[1:2,1:2])
}

# Residual intercepts with the accompanying constraints
f_RMS = alpha_adj[1]
f_RSM = alpha_adj[2]

f_RMT = -alpha_adj[1]
f_RTM = -alpha_adj[2]
```

```
f_RFS = -alpha_adj[1]
f_RSF = -alpha_adj[2]

f_RFT = alpha_adj[1]
f_RTF = alpha_adj[2]

for(i in 1:2){
alpha_adj[i] ~ dnorm(0,0.0001)
}

# Making clear that f_role consists of actor and partner effects
for (i in 1:N) {
f_AM[i] = f_mother[i,1]
f_PM[i] = f_mother[i,2]

f_AF[i] = f_father[i,1]
f_PF[i] = f_father[i,2]

f_AS[i] = f_sibling[i,1]
f_PS[i] = f_sibling[i,2]

f_AT[i] = f_target[i,1]
f_PT[i] = f_target[i,2]
}

# Precision matrix Actor - Partner for each role
Sigma_M = inverse(Tau_M)
Sigma_F = inverse(Tau_F)
Sigma_S = inverse(Tau_S)
Sigma_T = inverse(Tau_T)

# Calculation of Actor-Partner correlations
Cor_M = Sigma_M[1,2]/(sqrt(Sigma_M[1,1])*sqrt(Sigma_M[2,2]))
Cor_F = Sigma_F[1,2]/(sqrt(Sigma_F[1,1])*sqrt(Sigma_F[2,2]))
Cor_S = Sigma_S[1,2]/(sqrt(Sigma_S[1,1])*sqrt(Sigma_S[2,2]))
Cor_T = Sigma_T[1,2]/(sqrt(Sigma_T[1,1])*sqrt(Sigma_T[2,2]))

# Precision/Variance Group Effect (Family Effect)
sigma_FE <- 1/tau_FE

# Mean Effects
#  Family Effect
mu_FE ~ dnorm(0,0.0001)
# Mother
mu_M[1,1] = mu_actor[1]
mu_M[1,2] = mu_beta[1]
# Father
mu_F[1,1] = mu_actor[2]
mu_F[1,2] = mu_actor[3] + mu_actor[4] - mu_beta[1]
# Sibling
mu_S[1,1] = mu_actor[3]
mu_S[1,2] = mu_beta[3]
# Target
```

```
mu_T[1,1] = mu_actor[4]
mu_T[1,2] = mu_actor[1] + mu_actor[2] - mu_beta[3]

for (i in 1:4){
mu_actor[i] <- mu_alpha[i] - mean(mu_alpha)
mu_alpha[i] ~ dnorm(0,0.0001)
mu_beta[i] ~ dnorm(0,0.0001)
}

# Prior Precision Family Effect
tau_FE ~ dgamma(.01, .01)

# Prior Precision Actor-Partner effects
Tau_M ~ dwish(R[1:2,1:2], 2)
Tau_F ~ dwish(R[1:2,1:2], 2)
Tau_S ~ dwish(R[1:2,1:2], 2)
Tau_T ~ dwish(R[1:2,1:2], 2)
}'

R = structure(.Data=c(w,0,0,w),.Dim=c(2,2)) # let w vary 0.2/1/5
jags <- jags.model(textConnection(model_count_blocked_mean),
                   data = list( x_MS = x_MS, x_MT = x_MT,
                                x_FS = x_FS, x_FT = x_FT,
                                x_SM = x_SM, x_TM = x_TM,
                                x_SF = x_SF, x_TF = x_TF,
                                N = N, R=R ),
                   n.chains = 3)
# Adaptation iterations
update(jags,1000)
# Parameters of interest
params <- c("sigma_FE",'Sigma_M', 'Sigma_F', 'Sigma_S','Sigma_T',
            "Cor_M","Cor_F",'Cor_S','Cor_T',
            'mu_M','mu_F','mu_T','mu_S','mu_FE','f_RMS','f_RSM',
            'f_RMT','f_RTM','f_RFS','f_RSF','f_RFT','f_RTF')
# 10.000 iterations with thinning factor of 15
samps_count <- coda.samples(jags, params, 10000, thin=15)
# Results
summary(samps_count)
# Highest Posterior Density intervals
HPDinterval(samps_count)

## Diagnostics: Convergence?
# Traceplots
traceplot(samps_count)
plot(samps_count)
# Autocorrelation plots
acfplot(samps_count)
# Gelman plot
gelman.plot(samps_count)

############################################################################
# The social relations model for count data in rjags: group analysis #
############################################################################
```

```
model_count_mean_group <- '
model{
# likelihood function (count dyadic data, depending on family type)
for (i in 1:N) {
x_MS[i] ~ dpois(mu_RMS[i, type[i]])
x_SM[i] ~ dpois(mu_RSM[i, type[i]])
x_MT[i] ~ dpois(mu_RMT[i, type[i]])
x_TM[i] ~ dpois(mu_RTM[i, type[i]])
x_FS[i] ~ dpois(mu_RFS[i, type[i]])
x_SF[i] ~ dpois(mu_RSF[i, type[i]])
x_FT[i] ~ dpois(mu_RFT[i, type[i]])
x_TF[i] ~ dpois(mu_RTF[i, type[i]])
}

# defining the expected value of the dyadic measurements as a
# combination of the actor, partner and family effects (and residuals effect)
for (i in 1:N) {
mu_RMS[i, type[i]] = exp(f_RMS[type[i]] + f_AM[i, type[i]] +
                           f_PS[i, type[i]] + f_FE[i, type[i]])
mu_RSM[i, type[i]] = exp(f_RSM[type[i]] + f_AS[i, type[i]] +
                           f_PM[i, type[i]] + f_FE[i, type[i]])

mu_RMT[i, type[i]] = exp(f_RMT[type[i]] + f_AM[i, type[i]] +
                           f_PT[i, type[i]] + f_FE[i, type[i]])
mu_RTM[i, type[i]] = exp(f_RTM[type[i]] + f_AT[i, type[i]] +
                           f_PM[i, type[i]] + f_FE[i, type[i]])

mu_RFS[i, type[i]] = exp(f_RFS[type[i]] + f_AF[i, type[i]] +
                           f_PS[i, type[i]] + f_FE[i, type[i]])
mu_RSF[i, type[i]] = exp(f_RSF[type[i]] + f_AS[i, type[i]] +
                           f_PF[i, type[i]] + f_FE[i, type[i]])

mu_RFT[i, type[i]] = exp(f_RFT[type[i]] + f_AF[i, type[i]] +
                           f_PT[i, type[i]] + f_FE[i, type[i]])
mu_RTF[i, type[i]] = exp(f_RTF[type[i]] + f_AT[i, type[i]] +
                           f_PF[i, type[i]] + f_FE[i, type[i]])
}

# Distribution group effects
for(i in 1:N){
f_FE[i,1] ~ dnorm(mu_FE,tau_FE)
f_FE[i,2] ~ dnorm(mu_FE2,tau_FE2)
}
# distribution actor and partner effects (bivariate normal) - multi_normal
for(i in 1:N){
f_mother[i,1:2]  ~ dmnorm(mu_M[1,1:2], Tau_M[1:2,1:2])
f_father[i,1:2]  ~ dmnorm(mu_F[1,1:2], Tau_F[1:2,1:2])
f_sibling[i,1:2] ~ dmnorm(mu_S[1,1:2], Tau_S[1:2,1:2])
f_target[i,1:2]  ~ dmnorm(mu_T[1,1:2], Tau_T[1:2,1:2])

f_mother2[i,1:2]  ~ dmnorm(mu_M2[1,1:2], Tau_M2[1:2,1:2])
f_father2[i,1:2]  ~ dmnorm(mu_F2[1,1:2], Tau_F2[1:2,1:2])
f_sibling2[i,1:2] ~ dmnorm(mu_S2[1,1:2], Tau_S2[1:2,1:2])
```

```
f_target2[i,1:2]  ~ dmnorm(mu_T2[1,1:2], Tau_T2[1:2,1:2])
}

f_RMS[1] = alpha_adj[1]
f_RSM[1] = alpha_adj[2]

f_RMT[1] = -alpha_adj[1]
f_RTM[1] = -alpha_adj[2]

f_RFS[1] = -alpha_adj[1]
f_RSF[1] = -alpha_adj[2]

f_RFT[1] = alpha_adj[1]
f_RTF[1] = alpha_adj[2]

f_RMS[2] = alpha_adj2[1]
f_RSM[2] = alpha_adj2[2]

f_RMT[2] = -alpha_adj2[1]
f_RTM[2] = -alpha_adj2[2]

f_RFS[2] = -alpha_adj2[1]
f_RSF[2] = -alpha_adj2[2]

f_RFT[2] = alpha_adj2[1]
f_RTF[2] = alpha_adj2[2]

for(i in 1:2){
alpha_adj[i] ~ dnorm(0,0.0001)
alpha_adj2[i] ~ dnorm(0,0.0001)
}

# Making clear that f_role consists of actor and partner effects
for (i in 1:N) {
f_AM[i,1] = f_mother[i,1]
f_PM[i,1] = f_mother[i,2]
f_AM[i,2] = f_mother2[i,1]
f_PM[i,2] = f_mother2[i,2]

f_AF[i,1] = f_father[i,1]
f_PF[i,1] = f_father[i,2]
f_AF[i,2] = f_father2[i,1]
f_PF[i,2] = f_father2[i,2]

f_AS[i,1] = f_sibling[i,1]
f_PS[i,1] = f_sibling[i,2]
f_AS[i,2] = f_sibling2[i,1]
f_PS[i,2] = f_sibling2[i,2]

f_AT[i,1] = f_target[i,1]
f_PT[i,1] = f_target[i,2]
f_AT[i,2] = f_target2[i,1]
f_PT[i,2] = f_target2[i,2]
```

```
}

# Precision Actor - Partner
Sigma_M = inverse(Tau_M)
Sigma_F = inverse(Tau_F)
Sigma_S = inverse(Tau_S)
Sigma_T = inverse(Tau_T)

Sigma_M2 = inverse(Tau_M2)
Sigma_F2 = inverse(Tau_F2)
Sigma_S2 = inverse(Tau_S2)
Sigma_T2 = inverse(Tau_T2)

# Ratio of variances
ratio_M[1] = Sigma_M[1,1]/Sigma_M2[1,1]
ratio_M[2] = Sigma_M[2,2]/Sigma_M2[2,2]
ratio_F[1] = Sigma_F[1,1]/Sigma_F2[1,1]
ratio_F[2] = Sigma_F[2,2]/Sigma_F2[2,2]
ratio_S[1] = Sigma_S[1,1]/Sigma_S2[1,1]
ratio_S[2] = Sigma_S[2,2]/Sigma_S2[2,2]
ratio_T[1] = Sigma_T[1,1]/Sigma_T2[1,1]
ratio_T[2] = Sigma_T[2,2]/Sigma_T2[2,2]

# Correlations of Actor-Partner
Cor_M = Sigma_M[1,2]/(sqrt(Sigma_M[1,1])*sqrt(Sigma_M[2,2]))
Cor_F = Sigma_F[1,2]/(sqrt(Sigma_F[1,1])*sqrt(Sigma_F[2,2]))
Cor_S = Sigma_S[1,2]/(sqrt(Sigma_S[1,1])*sqrt(Sigma_S[2,2]))
Cor_T = Sigma_T[1,2]/(sqrt(Sigma_T[1,1])*sqrt(Sigma_T[2,2]))

Cor_M2 = Sigma_M2[1,2]/(sqrt(Sigma_M2[1,1])*sqrt(Sigma_M2[2,2]))
Cor_F2 = Sigma_F2[1,2]/(sqrt(Sigma_F2[1,1])*sqrt(Sigma_F2[2,2]))
Cor_S2 = Sigma_S2[1,2]/(sqrt(Sigma_S2[1,1])*sqrt(Sigma_S2[2,2]))
Cor_T2 = Sigma_T2[1,2]/(sqrt(Sigma_T2[1,1])*sqrt(Sigma_T2[2,2]))

# Precision/Variance Group Effect (Family Effect)
sigma_FE <- 1/tau_FE
sigma_FE2 <- 1/tau_FE2
# Ratio variances Family Effect
ratio_FE <- sigma_FE/sigma_FE2

# Mean Effects
# Family Effect
mu_FE ~ dnorm(0,0.0001)
mu_FE2 ~ dnorm(0,0.0001)

# Mother
mu_M[1,1] = mu_actor[1]
mu_M[1,2] = mu_beta[1]
# Father
mu_F[1,1] = mu_actor[2]
mu_F[1,2] = mu_actor[3] + mu_actor[4] - mu_beta[1]
# Sibling
mu_S[1,1] = mu_actor[3]
```

```
mu_S[1,2] = mu_beta[3]
# Target
mu_T[1,1] = mu_actor[4]
mu_T[1,2] = mu_actor[1] + mu_actor[2] - mu_beta[3]

for (i in 1:4){
mu_actor[i] <- mu_alpha[i] - mean(mu_alpha)
mu_alpha[i] ~ dnorm(0,0.0001)
mu_beta[i] ~ dnorm(0,0.0001)
}

# Mother
mu_M2[1,1] = mu_actor2[1]
mu_M2[1,2] = mu_beta2[1]
# Father
mu_F2[1,1] = mu_actor2[2]
mu_F2[1,2] = mu_actor2[3] + mu_actor2[4] - mu_beta2[1]
# Sibling
mu_S2[1,1] = mu_actor2[3]
mu_S2[1,2] = mu_beta2[3]
# Target
mu_T2[1,1] = mu_actor2[4]
mu_T2[1,2] = mu_actor2[1] + mu_actor2[2] - mu_beta2[3]

for (i in 1:4){
mu_actor2[i] <- mu_alpha2[i] - mean(mu_alpha2)
mu_alpha2[i] ~ dnorm(0,0.0001)
mu_beta2[i] ~ dnorm(0,0.0001)
}

# Differences between groups in means
mu_FE_diff = mu_FE - mu_FE2
# Mother
mu_M_diff[1,1] = mu_M[1,1] - mu_M2[1,1]
mu_M_diff[1,2] = mu_M[1,2] - mu_M2[1,2]
# Father
mu_F_diff[1,1] = mu_F[1,1] - mu_F2[1,1]
mu_F_diff[1,2] = mu_F[1,2] - mu_F2[1,2]
# Sibling
mu_S_diff[1,1] = mu_S[1,1] - mu_S2[1,1]
mu_S_diff[1,2] = mu_S[1,2] - mu_S2[1,2]
# Target
mu_T_diff[1,1] = mu_T[1,1] - mu_T2[1,1]
mu_T_diff[1,2] = mu_T[1,2] - mu_T2[1,2]

# Prior Precision Family Effect
tau_FE ~ dgamma(.01, .01)
tau_FE2 ~ dgamma(.01, .01)

# Prior SRM effects
Tau_M ~ dwish(R[1:2,1:2], 2)
Tau_F ~ dwish(R[1:2,1:2], 2)
Tau_S ~ dwish(R[1:2,1:2], 2)
```

```
Tau_T ~ dwish(R[1:2,1:2], 2)
Tau_M2 ~ dwish(R[1:2,1:2], 2)
Tau_F2 ~ dwish(R[1:2,1:2], 2)
Tau_S2 ~ dwish(R[1:2,1:2], 2)
Tau_T2 ~ dwish(R[1:2,1:2], 2)
}'

structure(.Data=c(w,0,0,w),.Dim=c(2,2)) # Let w vary 0.2/1/5
jags <- jags.model(textConnection(model_count_mean_group),
                   data = list(x_MS = x_MS, x_MT = x_MT,
                               x_FS = x_FS, x_FT = x_FT,
                               x_SM = x_SM, x_TM = x_TM,
                               x_SF = x_SF, x_TF = x_TF,
                              type=type,N = N,R=R),
                   n.chains = 3)
# Adaptation iterations
update(jags,1000)

# Parameters of interest
params <- c("sigma_FE",'Sigma_M', 'Sigma_F', 'Sigma_S','Sigma_T',
            "sigma_FE2",'Sigma_M2', 'Sigma_F2', 'Sigma_S2','Sigma_T2',
            "Cor_M","Cor_F",'Cor_S','Cor_T',
            "Cor_M2","Cor_F2",'Cor_S2','Cor_T2',
            'mu_M','mu_F','mu_T','mu_S','mu_FE',
            'mu_M2','mu_F2','mu_T2','mu_S2','mu_FE2',
            'mu_M_diff', 'mu_S_diff', 'mu_T_diff', 'mu_F_diff',
            'ratio_M','ratio_F', 'ratio_S', 'ratio_T',
            'ratio_FE','mu_FE_diff',
            'f_RMS','f_RSM',
            'f_RMT','f_RTM','f_RFS','f_RSF','f_RFT','f_RTF')
# Fitting model with 10.000 iterations and thinning factor of 15
samps_count_group <- coda.samples(jags, params, 10000, thin=15)
# Summary of parameters
summary(samps_count_group)
# Highest Posterior Density intervals
HPDinterval(samps_count_group)

## Diagnostics: Convergence?
# Traceplots
traceplot(samps_count_group)
plot(samps_count_group)
# Autocorrelation plots
acfplot(samps_count_group)
# Gelman plot
gelman.plot(samps_count_group)

##############################################
# Running Mplus model + Reading the results #
##############################################
# Running Mplus model (i.e. an .inp file)
runModels("~/Path_to_Folder_containing_MplusModel")
# Reading Mplus output (i.e. an .out file)
output <- readModels("~/Path_to_Folder_containing_MplusOutput/Mplus.out")
```