```
library(nimble)
library(coda)
#############################################################
############             Load data               ############
#############################################################

load("outcomedata.Rda")
#### outcomedata.Rda contains the following:
#### n = number of counties (88)
#### y = n*TT x 4 matrix with columns for the outcome variables
#### Pop = n*TT vector of the county populations
#### Year = n*TT vector of the year
#### cen = an n*TT vector of indicators of whether there was censoring in the
treatment count or not
#### bd = an n*TT x 2 matrix of the lower and upper bound of the censoring
interval; if not censoring then both bounds equal the observed value
#### W = n x n adjacency matrix


y.T = y[,1]
y.D = y[,2]
y.C = y[,3]
y.I = y[,4]

TT = length(unique(Year))


### plug in values for censored right now just to compute the expected count
### assume the midpoint of the bounded interval
y.T0 = ceiling(1/2*(bd[,2]+bd[,1]))

n<-nrow(W)
num<-colSums(W)

ET = rep(0,n*TT)
ED = rep(0,n*TT)
EC = rep(0,n*TT)
EI = rep(0,n*TT)
for(t in 1:TT){
  for(i in 1:n){
    ET[((t-1)*n+i)] =  sum(y.T0[which(Year==2014)])/
sum(Pop[which(Year==2014)])*Pop[((t-1)*n+i)]
    ED[((t-1)*n+i)] =  sum(y.D[which(Year==2014)])/
sum(Pop[which(Year==2014)])*Pop[((t-1)*n+i)]
    EC[((t-1)*n+i)] =  sum(y.C[which(Year==2014)])/
sum(Pop[which(Year==2014)])*Pop[((t-1)*n+i)]
    EI[((t-1)*n+i)] =  sum(y.I[which(Year==2014)])/
sum(Pop[which(Year==2014)])*Pop[((t-1)*n+i)]
  }
}



#############################################################
```

```
############         Set up for NIMBLE          ############
#############################################################


adj<-NULL
for(j in 1:n){
  adj<-c(adj,which(W[j,]==1))
}
adj<-as.vector(adj)
num<-as.vector(num)
weights<-1+0*adj


model_code=nimbleCode({
  for (t in T0:T0){
    for (i in 1:n){
    y.D[n*(t-T0)+i] ~ dpois(ED[(n*(t-T0)+i)]*lambdaD[n*(t-T0)+i])
    cen[n*(t-T0)+i] ~ dinterval(y.T[n*(t-T0)+i],bd[n*(t-T0)+i,1:2])
    y.T[n*(t-T0)+i] ~ dpois(ET[(n*(t-T0)+i)]*lambdaT[n*(t-T0)+i])
    y.C[n*(t-T0)+i] ~ dpois(EC[(n*(t-T0)+i)]*lambdaC[n*(t-T0)+i])
    y.I[n*(t-T0)+i] ~ dpois(EI[(n*(t-T0)+i)]*lambdaI[n*(t-T0)+i])
    log(lambdaT[n*(t-T0)+i]) <- inprod(load.mat[1,1:F],U[n*(t-T0)+i,1:F]
+mu[n*(t-T0)+i,1:F]) + VT[(n*(t-T0)+i)]
    log(lambdaD[n*(t-T0)+i]) <- inprod(load.mat[2,1:F],U[n*(t-T0)+i,1:F]
+mu[n*(t-T0)+i,1:F]) + VD[(n*(t-T0)+i)]
    log(lambdaC[n*(t-T0)+i]) <- inprod(load.mat[3,1:F],U[n*(t-T0)+i,1:F]
+mu[n*(t-T0)+i,1:F]) + VC[(n*(t-T0)+i)]
    log(lambdaI[n*(t-T0)+i]) <- inprod(load.mat[4,1:F],U[n*(t-T0)+i,1:F]
+mu[n*(t-T0)+i,1:F]) + VI[(n*(t-T0)+i)]
    VD[n*(t-T0)+i] ~ dnorm(0,tau.VD)
    VT[n*(t-T0)+i] ~ dnorm(0,tau.VT)
    VC[n*(t-T0)+i] ~ dnorm(0,tau.VC)
    VI[n*(t-T0)+i] ~ dnorm(0,tau.VI)
    mu[n*(t-T0)+i,1:F] <- mu0[t-T0+1,1:F]
    }



    # ICAR prior for the spatial factors
    U[(n*(t-T0)+1):(n*(t-T0)+n),1] ~ dcar_normal(adj[], weights[], num[],
tau.U[t-(T0-1),1],zero_mean=1)
    U[(n*(t-T0)+1):(n*(t-T0)+n),2] ~ dcar_normal(adj[], weights[], num[],
tau.U[t-(T0-1),2],zero_mean=1)
    U[(n*(t-T0)+1):(n*(t-T0)+n),3] ~ dcar_normal(adj[], weights[], num[],
tau.U[t-(T0-1),3],zero_mean=1)

    tau.U[t-(T0-1),1] ~ dgamma(0.5,0.5)
    tau.U[t-(T0-1),2] ~ dgamma(0.5,0.5)
    tau.U[t-(T0-1),3] ~ dgamma(0.5,0.5)


  }

for (t in (T0+1):(T0+TT-1)){
```

```
  for (i in 1:n){
    y.D[n*(t-T0)+i] ~ dpois(ED[(n*(t-T0)+i)]*lambdaD[n*(t-T0)+i])
    cen[n*(t-T0)+i] ~ dinterval(y.T[n*(t-T0)+i],bd[n*(t-T0)+i,1:2])
    y.T[n*(t-T0)+i] ~ dpois(ET[(n*(t-T0)+i)]*lambdaT[n*(t-T0)+i])
    y.C[n*(t-T0)+i] ~ dpois(EC[(n*(t-T0)+i)]*lambdaC[n*(t-T0)+i])
    y.I[n*(t-T0)+i] ~ dpois(EI[(n*(t-T0)+i)]*lambdaI[n*(t-T0)+i])
    log(lambdaT[n*(t-T0)+i]) <- inprod(load.mat[1,1:F],U[n*(t-T0)+i,1:F]
+mu[n*(t-T0)+i,1:F]) + VT[(n*(t-T0)+i)]
    log(lambdaD[n*(t-T0)+i]) <- inprod(load.mat[2,1:F],U[n*(t-T0)+i,1:F]
+mu[n*(t-T0)+i,1:F]) + VD[(n*(t-T0)+i)]
    log(lambdaC[n*(t-T0)+i]) <- inprod(load.mat[3,1:F],U[n*(t-T0)+i,1:F]
+mu[n*(t-T0)+i,1:F]) + VC[(n*(t-T0)+i)]
    log(lambdaI[n*(t-T0)+i]) <- inprod(load.mat[4,1:F],U[n*(t-T0)+i,1:F]
+mu[n*(t-T0)+i,1:F]) + VI[(n*(t-T0)+i)]
    VD[n*(t-T0)+i] ~ dnorm(0,tau.VD)
    VT[n*(t-T0)+i] ~ dnorm(0,tau.VT)
    VC[n*(t-T0)+i] ~ dnorm(0,tau.VC)
    VI[n*(t-T0)+i] ~ dnorm(0,tau.VI)
    mu[n*(t-T0)+i,1] <- mu0[t-T0+1,1]+eta[1]*(U[n*(t-(T0+1))+i,1]-mu0[t-T0,1])
    mu[n*(t-T0)+i,2] <- mu0[t-T0+1,2]+eta[2]*(U[n*(t-(T0+1))+i,2]-mu0[t-T0,2])
    mu[n*(t-T0)+i,3] <- mu0[t-T0+1,3]+eta[3]*(U[n*(t-(T0+1))+i,3]-mu0[t-T0,3])

  }


  # ICAR prior for the spatial factors
  U[(n*(t-T0)+1):(n*(t-T0)+n),1] ~ dcar_normal(adj[], weights[], num[],
tau.U[t-(T0-1),1],zero_mean=1)
  U[(n*(t-T0)+1):(n*(t-T0)+n),2] ~ dcar_normal(adj[], weights[], num[],
tau.U[t-(T0-1),2],zero_mean=1)
  U[(n*(t-T0)+1):(n*(t-T0)+n),3] ~ dcar_normal(adj[], weights[], num[],
tau.U[t-(T0-1),3],zero_mean=1)

  tau.U[t-(T0-1),1] ~ dgamma(0.5,0.5)
  tau.U[t-(T0-1),2] ~ dgamma(0.5,0.5)
  tau.U[t-(T0-1),3] ~ dgamma(0.5,0.5)

}
  #Priors

  ### loadings matrix entries
  for(k in 2:K){
    for(f in 1:(k-1)){
      load.mat[k,f] ~ dnorm(0,sd=10)
    }
  }


  for(t in T0:(T0+TT-1)){
    mu0[t-T0+1,1] ~ dflat()
    mu0[t-T0+1,2] ~ dflat()
    mu0[t-T0+1,3] ~ dflat()
  }
```

```
    tau.VD ~ dgamma(0.5,0.5)
    tau.VT ~ dgamma(0.5,0.5)
    tau.VC ~ dgamma(0.5,0.5)
    tau.VI ~ dgamma(0.5,0.5)
    eta[1] ~ dunif(0,1)
    eta[2] ~ dunif(0,1)
    eta[3] ~ dunif(0,1)


    })


############################################################
#########              Call NIMBLE                 ###########
############################################################

mod_constants=list(bd=bd,n=n,TT=TT,T0=min(Year),num=num,adj=adj,weights=weights,K=4,F=3)

mod_data=list(cen=cen,y.D=y.D,y.T=y.T,y.C=y.C,y.I=y.I,ET=ET,ED=ED,EC=EC,EI=EI)


mod_inits=list(U=matrix(0,n*TT,3),load.mat=diag(4)[,1:3],mu0=matrix(0,TT,
3),VD=rep(0,n*TT),VT=rep(0,n*TT),VC=rep(0,n*TT),VI=rep(0,n*TT),y.T=floor(.
5*(bd[,1]+bd[,2])))


# Build the model.
nim_model <- nimbleModel(model_code, mod_constants,mod_data,mod_inits)
compiled_model <- compileNimble(nim_model,resetFunctions = TRUE)




# Set up samplers.
mcmc_conf <- configureMCMC(nim_model,monitors=c("mu0","load.mat","mu",
                                                "U","eta","VD","VT","VC","VI","lambdaT",
= TRUE)

mod_mcmc<-buildMCMC(mcmc_conf)
compiled_mcmc<-compileNimble(mod_mcmc, project = nim_model,resetFunctions =
TRUE)

# Run the model
MCS=500000
st<-Sys.time()
samples=runMCMC(compiled_mcmc,inits=mod_inits,
                  nchains = 1, nburnin=floor(MCS/2),niter =
MCS,samplesAsCodaMCMC = TRUE,thin=50,
                  summary = FALSE, WAIC = FALSE,progressBar=TRUE)
Sys.time()-st

save(samples,file="ModOutput.Rda")

quit()
```