Math 8820 Project 1

Shirong Zhao Shanshan Jia Boyoung Hur *

January 6, 2019

*Zhao: Department of Economics, Clemson University, Clemson, South Carolina 29634–1309, USA; email shironz@g.clemson.edu.
Jia: Department of Mathematical Science, Clemson University, Clemson, South Carolina 29634,USA;email shanshj@g.clemson.edu.
Hur: Department of Mathematical Science, Clemson University, Clemson, South Carolina 29634,USA;email bhur@clemson.edu.

1 Model

In this section, we will set up our model. Let Y_{it} denote the number of populations that got flu in state i and time t, where $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, M$. Since Y_{it} denotes the counts, naturally we assume that

$$Y_{it} \sim Poisson\{\lambda_{it}\} \tag{1}$$

$$\lambda_{it} = \exp(\boldsymbol{X_{it}\beta} + \epsilon_{it}) \tag{2}$$

$$\boldsymbol{X_{it}\beta} = \beta_0 + \sum_{k=1}^{p} \beta_k X_{itk}, \quad \text{where } X_{itk} \text{ is the k-th covariate} \tag{3}$$

Equation (2) includes spacial and temporal correlation in our model through the random effects ϵ_{it}, we model our spatial and temporal dependence using first order autoregressive structures (i.e., AR(1)). Denote $\boldsymbol{\epsilon_t} = (\epsilon_{1t}, \epsilon_{2t}, , \epsilon_{Nt})^T$ and $\boldsymbol{\phi_t} = (\phi_{1t}, \phi_{2t}, , \phi_{Nt})^T$.

$$\boldsymbol{\epsilon_t} = \theta\boldsymbol{\epsilon_{t-1}} + \boldsymbol{\phi_t}, \quad for \quad t = 2, 3, \ldots, M \tag{4}$$

$$\boldsymbol{\epsilon_1} = \boldsymbol{\phi_1} \tag{5}$$

And the spatial random effects $\boldsymbol{\phi_t}$ at time t follows a CAR distribution and are independently and identically distributed.

$$\boldsymbol{\phi_t} \sim CAR(\tau^2; \rho), \quad or \quad \boldsymbol{\phi_t} \sim N(0, \tau^2(D - \rho W)), \quad for \quad t = 1, 2, \ldots, M \tag{6}$$

To complete the model, we specify the following priors

$$\begin{aligned} \boldsymbol{\beta} &\sim N(0, \boldsymbol{R}) \\ \theta &\sim Unif(-1, 1) \\ \rho &\sim Unif(0, 1), \text{ since flu is contagious} \\ \tau^{-2} &\sim Gamma(a_0, b_0) \end{aligned} \tag{7}$$

Then the posterior distribution for $\boldsymbol{\beta}, \boldsymbol{\phi_t}, \theta, \rho, \tau^{-2}$ is

$$\begin{aligned}
P(\boldsymbol{\beta}, \boldsymbol{\phi_t}, \theta, \rho, \tau^{-2}|X, Y) &\propto L(\boldsymbol{\beta}, \boldsymbol{\phi_t}, \theta, \rho, \tau^{-2}|X, Y)\pi_0(\boldsymbol{\beta})\pi_0(\boldsymbol{\phi_t})\pi_0(\theta)\pi_0(\rho)\pi_0(\tau^{-2}) \\
&\propto \prod_{t=1}^{M}\prod_{i=1}^{N} \exp(\boldsymbol{X_{it}\beta} + \epsilon_{it})^{Y_{it}} \exp(-\exp(\boldsymbol{X_{it}\beta} + \epsilon_{it})) \\
&\times \prod_{t=1}^{M} (\tau^{-2})^{N/2} \exp(-\frac{\boldsymbol{\phi_t}^T(D - \rho W)\boldsymbol{\phi_t}}{2\tau^2}) \\
&\times \exp(-\frac{\boldsymbol{\beta}^T \boldsymbol{R}^{-1}\boldsymbol{\beta}}{2}) \\
&\times \frac{1}{2} \\
&\times 1 \\
&\times (\tau^{-2})^{a_0 - 1} \exp(-\tau^{-2} b_0)
\end{aligned} \tag{8}$$

Notice that

$$\begin{aligned}
\boldsymbol{\epsilon_t} &= \theta\boldsymbol{\epsilon_{t-1}} + \boldsymbol{\phi_t} \\
&= \theta(\boldsymbol{\epsilon_{t-2}} + \boldsymbol{\phi_{t-1}}) + \boldsymbol{\phi_t} \\
&= \theta\boldsymbol{\epsilon_{t-2}} + \theta\boldsymbol{\phi_{t-1}} + \boldsymbol{\phi_t} \\
&= \dots \\
&= \theta^{t-1}\boldsymbol{\phi_1} + \theta^{t-2}\boldsymbol{\phi_2} + \dots + \theta\boldsymbol{\phi_{t-1}} + \boldsymbol{\phi_t}
\end{aligned} \tag{9}$$

Therefore if we know $\boldsymbol{\phi_t}$, for t = 1, 2, ..., M, and θ, we will know $\boldsymbol{\epsilon_t}$, for t = 1, 2, ..., M.

After some algebra, it can be shown that the posterior for τ^{-2} is

$$\tau^{-2} \sim Gamma(a_0 + \frac{NM}{2}, b_0 + \frac{\sum_{t=1}^{M} \boldsymbol{\phi_t}^T(D - \rho W)\boldsymbol{\phi_t}}{2})$$

However, we could not recognize the posterior distribution for $\boldsymbol{\beta}, \boldsymbol{\phi_t}, \theta, \rho$. And we will use Metropolis Algorithm or Metropolis-Hastings Algorithm. The proposal distribution for $\boldsymbol{\beta} \sim N(\boldsymbol{\beta}^{(s-1)}, beta.var.prop)$; for $\boldsymbol{\phi_t} \sim N(\boldsymbol{\phi_t}^{(s-1)}, phi.var.prop)$. Since $\theta \in [-1, 1]$; the proposal distribution for θ is reflected random walk, i.e. $\theta \sim Unif(\theta^{(s-1)} - \delta, \theta^{(s-1)} + \delta)$. If $\theta < -1$, we use $-2 - \theta$. If $\theta > 1$, we use $2 - \theta$. And since $\rho \in [0, 1]$, the proposal distribution for ρ is $\log(\frac{\rho}{1-\rho}) \sim N(\log(\frac{\rho^{(s-1)}}{1-\rho^{(s-1)}}), c)$. However, before we run our model, we need to tune the parameters *beta.var.prop*, *phi.var.prop*, δ and c so that the accepting rate for each group of parameters will be around 35%.

2 Simulation

To show our model actually works. We simulate N=36 states with M=5 months data. And we let $X = cbind(1, rnorm(NM))$, $\boldsymbol{\beta} = c(1, 2)$, $\theta = 0.5$, $\rho = 0.5$, $\tau^2 = 2$. And we use the methods developed in Section 1 to run 2×10^5 iterations. Our estimates and plots are shown below.[1]

Table 1: Outcome of Simulations

Parameter	True	Estimate	HPD
β_1	1	0.8771	[0.5559, 1.2049]
β_2	2	1.9180	[1.7353, 2.1031]
τ^2	2	2.3418	[1.2649, 3.3912]
θ	0.5	0.4392	[0.1656, 0.6882]
ρ	0.5	0.7168	[0.2951, 0.9966]

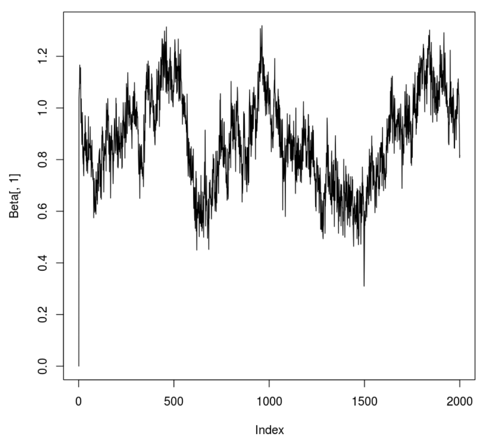

Figure 1: Trace Plot for β_1

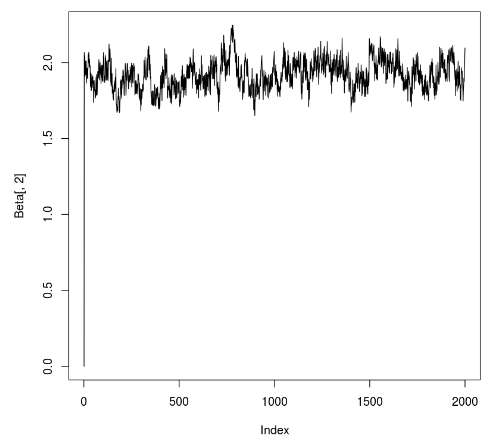

Figure 2: Trace Plot for β_2

[1]For the details of our simulation, please check the attached codes.

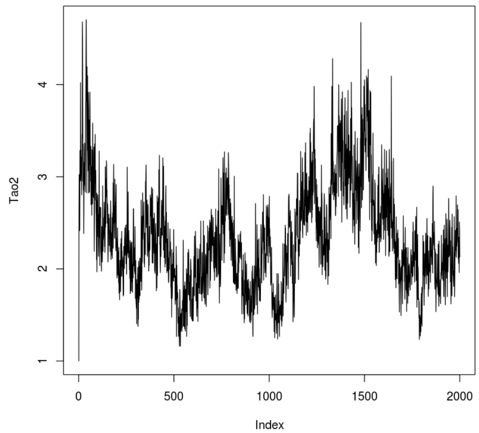

Figure 3: Trace Plot for τ^2

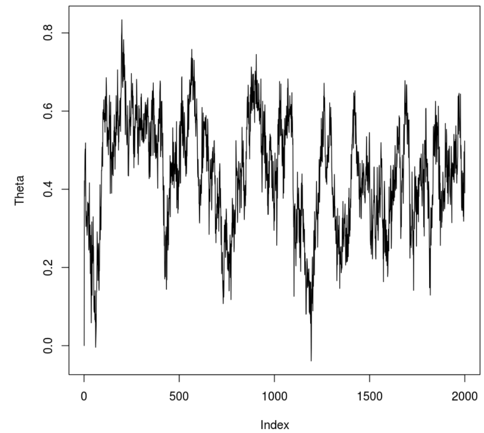

Figure 4: Trace Plot for θ

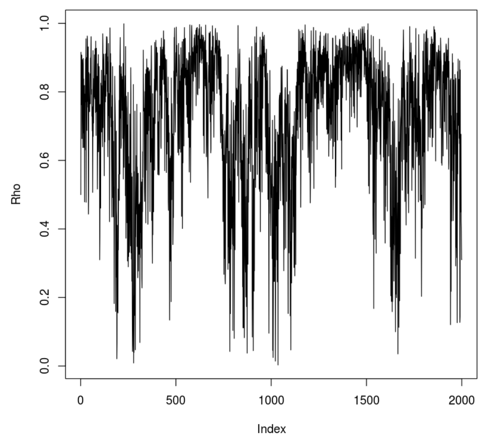

Figure 5: Trace Plot for ρ

Overall, it appears that the chain has converged for all parameters and the HPD covers the true value of parameters. Therefore our model makes sense.

3 Data Application

3.1 Data Description

After gathering data, we have the data for 47 states and time during October 2010 to December 2017. Due to data problems, we did not consider states Hawaii, Alaska, Florida, District of Columbia and Puerto Rico. Specially, we have the data for the following states: Alabama, Arizona, Arkansas, California, Colorado, Connecticut, Delaware, Georgia, Idaho, Illinois, Indiana, Iowa, Kansas, Kentucky, Louisiana, Maine, Maryland, Massachusetts, Michigan, Minnesota, Mississippi, Missouri, Montana, Nebraska, Nevada, New Hampshire, New Jersey, New Mexico, New York, North Carolina, North Dakota, Ohio, Oklahoma, Oregon, Pennsylvania, Rhode Island, South Carolina, South Dakota, Tennessee, Texas, Utah, Vermont, Virginia, Washington, West Virginia, Wisconsin, Wyoming. D

The variables we used for this project are listed in Table 2.

Table 2: Variables Definition

Variables	Definiton
Y_{it}	number of persons who got flu in state i and month t
X_{it0}	constant, equal to 1
X_{it1}	number of population in million in state i and month t
X_{it2}	median income in thousands dollar in state i and month t
X_{it3}	temperature in in Fahrenheit in state i and month t
X_{it4}	precipitation in in Inches in state i and month t
X_{it5}	share of white in state i and month t
X_{it6}	share of African American in state i and month t
X_{it7}	share of American Indian in state i and month t
X_{it8}	share of Asian in state i and month t
X_{it9}	spring dummy variable in state i and month t
X_{it10}	autumn dummy variable in state i and month t
X_{it11}	winter dummy variable in state i and month t

Notice that X_{it9}, X_{it10}, X_{it11} are used to control for seasonality. Specially summer is used as a benchmark.

3.2 Outcome

We use the data from October 2010 to December 2016 to train our model and will use our model to predict flu from January 2017 to December 2017 and compare predicted flu with true flu. We run our MCMC for $2 * 10^5$. The estimated values for the parameters of the model is shown in Table 3.

Table 3: The Estimated Values for the Parameters of the Model

Parameter	Estimate	HPD
β_0	-10.6842	[-11.8981 , -9.6458]
β_1	5.2725	[5.0277 , 5.4664]
β_2	-0.0355	[-0.0384 , -0.0330]
β_3	-0.0313	[-0.0317 , -0.0309]
β_4	0.0020	[1.1637e-04, 0.0040]
β_5	18.3413	[17.3524 , 19.6370]
β_6	23.8373	[22.8659 , 25.1871]
β_7	25.9956	[24.7529 , 27.4812]
β_8	29.4129	[28.1423 , 30.9573]
β_9	0.5034	[0.4849 , 0.5144]
β_{10}	0.1205	[0.1053 , 0.1382]
β_{11}	0.3433	[0.3120 , 0.3728]
τ^2	0.4986	[0.4123 , 0.5847]
θ	0.9356	[0.9309 , 0.9418]
ρ	0.8730	[0.8214 , 0.9176]

Notice that The HPDs for all estimates do not include 0, which means all the estimates are significant from 0. Also $\beta_1 > 0$, meaning that the state with more populations will be more likely to have more persons getting flu. Also $\beta_2 < 0$, meaning that the state with high median income will be more likely to have less persons getting flu. $\beta_3 < 0$ since flu usually happens in cold whether. Also notice that $\beta_9 > \beta_{11} > \beta_{10}$, meaning that flu will be more likely to happen in Spring and Winter and be less likely to happen in Summer and Autumn. Finally, notice that θ and ρ are both positive and very close to 1, showing that there are very high spacial correlation and time correlation.

We then use our model to predict the flu from Jan 2017 to Dec 2017. We use the mean of these estimates to calculate $\lambda_{it} = \exp(\boldsymbol{X_{it}\beta} + \epsilon_{it})$. The mean of true flu and predicted flu for each month is listed in Table 4.

Table 4: Mean Predictions of Persons Getting Flu for Jan 2017 - Dec 2017

Months	True	Estimate
1	2502	969
2	2575	1010
3	3202	1402
4	1054	1031
5	746	523
6	530	232
7	500	310
8	480	244
9	670	284
10	1484	533
11	1716	641
12	3059	869

Clearly our estimates underestimate the true number of populations who get flu. It could be due to some other reasons, for example, the pattern of flu has changed for 2017.

4 Conclusion

In this project, we use a spacial-temporal model to predict the number of flu for each states in U.S. We find that there are strong spacial and time series correlations. There correlations should be considered, otherwise, it will cause biased estimates of coefficients. However, our predictions are not good enough. We need to figure them out in future.

5 Appendix

5.1 A. R code

```
##################################################
#  8820 Introduction to Bayesian Statistics
#   Project 1: Predicts Flu
#   Shirong Zhao Shanshan Jia and Boyoung Hur
#################################################
library(MASS)
library(mvtnorm)
library(coda)
library(Matrix)
library(mnormt)
library(gdata) # use inside command write.fwf
#################################################
#######################  BEGIN Import Cleaned Data ##########################

formatInfo<-read.csv("./formatInfoall.csv")
df <- read.fwf(file="./all.txt", widths=formatInfo$width + 1, skip=1, strip.white=TRUE, na.strings="n.a.")
# V14 is the number of flu for state i and time t


formatInfoW<-read.csv("./formatInfoW.csv")
W<- read.fwf(file="./w.txt", widths=formatInfoW$width + 1, skip=1, strip.white=TRUE, na.strings="n.a.")

######################## End Import Cleaned Data #########################
dim(df)
W = as.matrix(W)
d=rowSums(W[,1:47])
D=diag(d,47,47)

df$V5<-as.numeric(gsub(",", "", df$V5))

Y=df$V14
x0=1
x1=df$V4/100000000 # population in 100 million
x2=df$V5/1000 # income in 1000
x3=df$V6 # temperature
x4=df$V7 # precipation
x5=df$V8/df$V4 # share of white
x6=df$V9/df$V4 # share of African American
x7=df$V10/df$V4 # share of American Indian
x8=df$V11/df$V4 # share of Asian

df$spring = 0  # summer as a benchmark
df$autumn = 0
df$winter = 0

df$spring[which(df$V3==3 | df$V3==4 | df$V3==5)] = 1
df$autumn[which(df$V3==9 | df$V3==10 | df$V3==11)] = 1
df$winter[which(df$V3==12 | df$V3==1 | df$V3==2)] = 1

spring=df$spring
autumn=df$autumn
winter=df$winter

X=cbind(x0,x1,x2,x3,x4,x5,x6,x7,x8,spring,autumn,winter)

# here we only use the data from Oct,2010-Dec,2016 to train the data
```

```
# the data in 2017 will be used to forecast

Y=Y[1:(4089-564)]  # 47*12=564
X=X[1:(4089-564),]

# popu: one hundred million
# income: thousand
# temperature: Degrees Fahrenheit,
# Temperature is in Fahrenheit and Precipitation is in Inches

# First using MLE find the sd for proposal distribution of beta
fit <- glm(Y ~ X[,1:12]-1, family=poisson()) # X[,1] is the intercept
summary(fit)
############################################################################
# Inputs:
# Y = response vector for current data, N*M by 1
# N = the numner of states
# M = the number of months
# p: number of covariates
# X = design matrix for current data N*M by p
# R = prior covariance matrix for beta (i.e. beta~N(0,R))
# a0 = prior parameter for tau2
# b0 = prior parameter for tau2, where tau2^{-1}~gamma(a0,b0)
# beta = initial value of regression coefficients
# tau2  = initial value of precission parameter
# theta: initial value for autoregressive coefficients
# rho: initial value for rho
# beta.var.prop: variance for the beta proposal distribution,(i.e., beta.p~N(beta(s-1), beta.var.prop))
# phi.var.prop: variance for the phi proposal distribution,(i.e., phi.p~N(phi(s-1), phi.var.prop))
# c: variance for the rho proposal distribution


N = 47
M = 75 # from 201010-201612

p = dim(X)[2]
NM = dim(X)[1]

iter = 2e5
thin = 1e2

# a_chol<-chol(DW)
# chol2inv(a_chol)

beta.var.prop=vcov(fit)
#beta.var.prop<-diag(rep(0.00005,p), p, p) # need to specify later and it's better to use var of parameters in poisson regression
# for here I just specify var as 0.1
# we could also consider var.prop<-var(log(Y))*solve(t(X)%*%X)
phi.var.prop=diag(rep(0.00002,N), N, N) # need to consider later
delta=0.010 # used in proposal disttribution for theta,reflected random walk
c=2 # specify the variance of proposal distribution for rho, log(rho.p/(1-rho.p)) follows normal (log(rho/(1-rho)), c)
# or rho.p/(1-rho.p) follows the lognormal(log(rho/(1-rho)), c)


# Specify the priors
R<-rep(10, p)
a0 = 1
b0 = 1
phi = matrix(0, nrow = N, ncol = M)
epsilon = matrix(0, nrow = N, ncol = M)
epsilon<-as.vector(epsilon) # nrow=N*M
```

```
tau2 = 1
theta = 0
rho = 0.5
rho1 = rho/(1-rho)

DW<-D-rho*W
DWI<-solve(DW)
DW<-as.matrix(DW)
DWI<-as.matrix(DWI)

# save the parameters
Beta = matrix(-99, nrow=iter/thin, ncol=p)
beta = rep(0,p)
Beta[1, ] = beta
Tau2 = rep(-99, iter/thin)
Tau2[1] = tau2
Theta = rep(-99, iter/thin)
Theta[1] = theta
Rho = rep(-99, iter/thin)
Rho[1] = rho
Phi = matrix(-99, nrow=iter/thin, ncol=N*M)
Phi[1, ] = as.vector(phi)

acc0 = acc1 = acc2 = acc3 = 0

llik = sum(dpois(Y, exp(X%*%beta + epsilon), log = TRUE))

############################################################################
# Burn in loop

for(i in (thin + 1):iter){


  ## update all beta simultaneously, using Metropolis Algorithm
  beta.p = t(rmvnorm(1, beta, beta.var.prop))
  beta.prior = sum(dnorm(beta, 0, R, log = TRUE))
  llik.p = sum(dpois(Y, exp(X%*%beta.p + epsilon), log = TRUE))
  beta.prior.p = sum(dnorm(beta.p, 0, R, log = TRUE))
  r = exp(llik.p -llik + beta.prior.p - beta.prior)
  Z<-rbinom(1,1,min(r,1))
  if(Z==1){
    beta = beta.p
    llik = llik.p
    acc0 = acc0 + 1
  }



  ## update epsilon and phi (the spacial random effect), using Metropolis Algorithm
  phi.p = matrix(-99, nrow = N, ncol = M)
  epsilon.p = matrix(-99, nrow = N, ncol = M)
  phi.prior = 0
  phi.prior.p = 0
  for (t in 1:M) {
    phi.p[,t] = t(rmvnorm(1, phi[,t], phi.var.prop))
    phi.prior.t = dmvnorm(phi[,t], mean=rep(0,N), sigma=tau2*DWI, log = TRUE) # sigma is
        covariance matrix
    phi.prior = phi.prior + phi.prior.t
    phi.prior.p.t = dmvnorm(phi.p[,t], mean=rep(0,N), sigma=tau2*DWI, log = TRUE)
    phi.prior.p = phi.prior.p + phi.prior.p.t
  }
  epsilon.p[,1] = phi.p[,1]
  for (t in 2:M) {
    epsilon.p[,t] = theta*epsilon.p[,t-1] + phi.p[,t]
  }
```

```
epsilon.p = as.vector(epsilon.p) # change to a vector
llik.p = sum(dpois(Y, exp(X%*%beta + epsilon.p), log = TRUE))
r = exp(llik.p -llik + phi.prior.p - phi.prior) # need to add density of proposal
    distribution
Z<-rbinom(1,1,min(r,1))
if(Z==1){
  phi = phi.p # phi is a matrix
  epsilon = epsilon.p # epsilon is a vector
  llik = llik.p
  acc1 = acc1 + 1
}



## update tau2, using Gibbs sampler
sumt = 0
for (t in 1:M) {
  sumt = sumt + t(phi[,t])%*%DW%*%phi[,t]/2
}
at = a0 + N*M/2
bt = b0 + sumt
tauI2 = rgamma(1,at,bt)
tau2 = 1/tauI2



## update theta, using Metropolis-Hastings Algorithm, reflected random walk
# prior for theta is uniform(-1,1)
theta.p = runif(1, min=theta-delta, max=theta+delta)
if (theta.p < -1){
  theta.p = -2- theta.p
} else if (theta.p > 1){
  theta.p = 2-theta.p}
epsilon.p = matrix(-99, nrow = N, ncol = M)
epsilon.p[,1] = phi[,1]
for (t in 2:M) {
  epsilon.p[,t] = theta.p*epsilon.p[,t-1] + phi[,t]
}
epsilon.p = as.vector(epsilon.p) # change to a vector
llik.p = sum(dpois(Y, exp(X%*%beta + epsilon.p), log = TRUE))
r = exp(llik.p -llik)
Z<-rbinom(1,1,min(r,1))
if(Z==1){
  theta = theta.p
  epsilon = epsilon.p # epsilon is a vector
  llik = llik.p
  acc2 = acc2 + 1
}


## update rho, using Metropolis-Hastings Algorithm, symmetric random walk
# prior for rho is uniform(0,1)
rho1.p= rlnorm(1, meanlog = log(rho/(1-rho)), sdlog = c)
rho.p=rho1.p/(1+rho1.p) # rho1.p=rho.p/(1-rho.p)
DW.p<-D-rho.p*W
DWI.p<-solve(DW.p)
DW.p<-as.matrix(DW.p)
DWI.p<-as.matrix(DWI.p)

llik.rho = 0
llik.rho.p = 0

for (t in 1:M) {
  llik.rho.t = dmvnorm(phi[,t], mean=rep(0,N), sigma=tau2*DWI, log = TRUE) # sigma is
      covariance matrix
```

```
    llik.rho = llik.rho + llik.rho.t
    llik.rho.p.t = dmvnorm(phi[,t], mean=rep(0,N), sigma=tau2*DWI.p, log = TRUE)
    llik.rho.p = llik.rho.p + llik.rho.p.t
  }

  r=exp(llik.rho.p - llik.rho
        + dlnorm(rho1, meanlog = log(rho1.p), sdlog = c, log = TRUE)
        - dlnorm(rho1.p, meanlog = log(rho1), sdlog = c, log = TRUE))
  Z<-rbinom(1,1,min(r,1))
  if(Z==1){
    rho =rho.p
    rho1 = rho1.p
    DWI = DWI.p
    acc3 = acc3 + 1
  }


  ## tuning the necessary parameters
  if(i %% 1000 == 0){
    beta.var.prop<-  beta.var.prop + (acc0/1000 >0.55)*0.75*beta.var.prop - (acc0/1000 < 0.35)*0.75*beta.var.prop
    phi.var.prop<-  phi.var.prop + (acc1/1000 >0.55)*0.75*phi.var.prop - (acc1/1000 < 0.35)*0.75*phi.var.prop
    delta<-  delta + (acc2/1000 >0.55)*0.75*delta - (acc2/1000 < 0.35)*0.75*delta
    c<-c + (acc3/1000 >0.55)*0.75*c - (acc3/1000 < 0.35)*0.75*c
    print(c(acc0,acc1,acc2,acc3))
    acc0<-0
    acc1<-0
    acc2<-0
    acc3<-0

  }
}



# ################################################################################################
# Sampling loop

for(i in (thin + 1):iter){


  ## update all beta simultaneously, using Metropolis Algorithm
  beta.p = t(rmvnorm(1, beta, beta.var.prop))
  beta.prior = sum(dnorm(beta, 0, R, log = TRUE))
  llik.p = sum(dpois(Y, exp(X%*%beta.p + epsilon), log = TRUE))
  beta.prior.p = sum(dnorm(beta.p, 0, R, log = TRUE))
  r = exp(llik.p -llik + beta.prior.p - beta.prior)
  Z<-rbinom(1,1,min(r,1))
  if(Z==1){
    beta = beta.p
    llik = llik.p
    acc0 = acc0 + 1
  }



  ## update epsilon and phi (the spacial random effect), using Metropolis Algorithm
  phi.p = matrix(-99, nrow = N, ncol = M)
  epsilon.p = matrix(-99, nrow = N, ncol = M)
  phi.prior = 0
  phi.prior.p = 0
  for (t in 1:M) {
```

```
    phi.p[,t] = t(rmvnorm(1, phi[,t], phi.var.prop))
    phi.prior.t = dmvnorm(phi[,t], mean=rep(0,N), sigma=tau2*DWI, log = TRUE) # sigma is
        covariance matrix
    phi.prior = phi.prior + phi.prior.t
    phi.prior.p.t = dmvnorm(phi.p[,t], mean=rep(0,N), sigma=tau2*DWI, log = TRUE)
    phi.prior.p = phi.prior.p + phi.prior.p.t
  }
  epsilon.p[,1] = phi.p[,1]
  for (t in 2:M) {
    epsilon.p[,t] = theta*epsilon.p[,t-1] + phi.p[,t]
  }
  epsilon.p = as.vector(epsilon.p) # change to a vector
  llik.p = sum(dpois(Y, exp(X%*%beta + epsilon.p), log = TRUE))
  r = exp(llik.p -llik + phi.prior.p - phi.prior)
  Z<-rbinom(1,1,min(r,1))
  if(Z==1){
    phi = phi.p # phi is a matrix
    epsilon = epsilon.p # epsilon is a vector
    llik = llik.p
    acc1 = acc1 + 1
  }



  ## update tau2, using Gibbs sampler
  sumt = 0
  for (t in 1:M) {
    sumt = sumt + t(phi[,t])%*%DW%*%phi[,t]/2
  }
  at = a0 + N*M/2
  bt = b0 + sumt
  tauI2 = rgamma(1,at,bt)
  tau2 = 1/tauI2



  ## update theta, using Metropolis-Hastings Algorithm, reflected random walk
  # prior for theta is uniform(-1,1)
  theta.p = runif(1, min=theta-delta, max=theta+delta)
  if (theta.p < -1){
    theta.p = -2- theta.p
  } else if (theta.p > 1){
    theta.p = 2-theta.p}
  epsilon.p = matrix(-99, nrow = N, ncol = M)
  epsilon.p[,1] = phi[,1]
  for (t in 2:M) {
    epsilon.p[,t] = theta.p*epsilon.p[,t-1] + phi[,t]
  }
  epsilon.p = as.vector(epsilon.p) # change to a vector
  llik.p = sum(dpois(Y, exp(X%*%beta + epsilon.p), log = TRUE))
  r = exp(llik.p -llik)
  Z<-rbinom(1,1,min(r,1))
  if(Z==1){
    theta = theta.p
    epsilon = epsilon.p # epsilon is a vector
    llik = llik.p
    acc2 = acc2 + 1
  }


  ## update rho, using Metropolis-Hastings Algorithm, symmetric random walk
  # prior for rho is uniform(0,1)
  rho1.p= rlnorm(1, meanlog = log(rho/(1-rho)), sdlog = c)
  rho.p=rho1.p/(1+rho1.p) # rho1.p=rho.p/(1-rho.p)
  DW.p<-D-rho.p*W
```

```
  DWI.p<-solve(DW.p)
  DW.p<-as.matrix(DW.p)
  DWI.p<-as.matrix(DWI.p)

  llik.rho = 0
  llik.rho.p = 0

  for (t in 1:M) {
    llik.rho.t = dmvnorm(phi[,t], mean=rep(0,N), sigma=tau2*DWI, log = TRUE) # sigma is covariance matrix
    llik.rho = llik.rho + llik.rho.t
    llik.rho.p.t = dmvnorm(phi[,t], mean=rep(0,N), sigma=tau2*DWI.p, log = TRUE)
    llik.rho.p = llik.rho.p + llik.rho.p.t
  }

  r=exp(llik.rho.p - llik.rho
        + dlnorm(rho1, meanlog = log(rho1.p), sdlog = c, log = TRUE)
        - dlnorm(rho1.p, meanlog = log(rho1), sdlog = c, log = TRUE))
  Z<-rbinom(1,1,min(r,1))
  if(Z==1){
    rho =rho.p
    rho1 = rho1.p
    DWI = DWI.p
    acc3 = acc3 + 1
  }


  if(i %% thin == 0){
    Beta[i / thin, ] = beta
    Tau2[i / thin] = tau2
    Theta[i / thin] = theta
    Rho[i / thin] = rho
    Phi[i / thin, ] = as.vector(phi)
    print(i)
  }


}


outcome6=cbind(Beta,Tau2,Theta,Rho)
formatInfo6<-write.fwf(x=outcome6, file="./outcome6.txt", formatInfo=TRUE)
write.csv(formatInfo6, "./formatInfo6.csv")


###############################################################################
###  Summarize the estimation outcome


acc0 / (iter-thin)
acc1 / (iter-thin)
acc2 / (iter-thin)
acc3 / (iter-thin)



Beta.mcmc = as.mcmc(Beta)
print(paste0("Estimate Mean of beta:  ", apply(Beta.mcmc, 2, mean)))
HPDinterval(Beta.mcmc)
print(paste0("Effective Sample Size:  ", effectiveSize(Beta.mcmc)))
plot(Beta.mcmc)
autocorr.plot(Beta.mcmc)
plot(Beta[, 1], typ = 'l')
plot(Beta[, 2], typ = 'l')
```

```
Tau2.mcmc = as.mcmc(Tau2)
print(paste0("Estimate Mean of tau2:  ", mean(Tau2.mcmc)))
HPDinterval(Tau2.mcmc)
print(paste0("Effective Sample Size:  ", effectiveSize(Tau2.mcmc)))
plot(Tau2.mcmc)
autocorr.plot(Tau2.mcmc)
plot(Tau2, typ = 'l')



Theta.mcmc = as.mcmc(Theta)
print(paste0("Estimate Mean of theta:  ", mean(Theta.mcmc)))
HPDinterval(Theta.mcmc)
print(paste0("Effective Sample Size:  ", effectiveSize(Theta.mcmc)))
plot(Theta.mcmc)
autocorr.plot(Theta.mcmc)
plot(Theta, typ = 'l')



Rho.mcmc = as.mcmc(Rho)
print(paste0("Estimate Mean of rho:  ", mean(Rho.mcmc)))
HPDinterval(Rho.mcmc)
print(paste0("Effective Sample Size:  ", effectiveSize(Rho.mcmc)))
plot(Rho.mcmc)
autocorr.plot(Rho.mcmc)
plot(Rho, typ = 'l')

proc.time()
```