season 2

**#R code-Intrinsic CAR-semiparametric POISS version linear**

install.packages(pkgs = c("BRugs"), type="source")

library('devtools')

rm(list=ls())

library(rgdal)

library(raster)

library(spdep)

library(igraph)

library(mclcar)

library(mvtnorm)

library(coda)

library(BRugs)

library(rjags)

library(R2OpenBUGS)

library(dclone)

library(stats)

library(nlme)

library(MASS)

library(LRs)

library(lattice)

####### a spatial generalized semiparametric model with a ICAR spatial effect

###------------------------- Essential function -----------------------###

### define space domain based on Minnesota shape file

G.shape = readOGR(dsn=path.expand("C:\\Users\\Elahe\\Desktop\\Miss Hosseini\\shp\_bdry\_counties\_in\_minnesota"),layer="mn\_county\_boundaries")

minnesota <- G.shape[G.shape$CTYONLY\_ID<88,]

minnesota$CTY\_NAME

#plot(G.shape)

nb\_q <- poly2nb(minnesota)

ccMat=nb2mat(nb\_q,style='B') ### weight matrix

#is.symmetric.matrix(ccMat)

num = as.vector(rowSums(ccMat))

##---- define adjacency matrix to use in OpenBUGS

n.site <- dim(ccMat)[1] # n.site: number of areas

n.edge <- sum(ccMat)/2 # n.edge: number of unique pairs

SEind1 <- SEind2 <- 0

matmy <- ccMat

for(i in 1:(n.site-1)){

for(j in (i+1):n.site){

if (ccMat[i,j]>0) {SEind1<-c(SEind1,i)

SEind2<-c(SEind2,j)

matmy[i,j]<-matmy[j,i]<-length(SEind1)-1

}

}

}

SEind1 <- SEind1[-1] # edges sorted by row of upper triangle of the adj. matrix

SEind2 <- SEind2[-1] # SEind1[k]=i and SEind2[k]=j => kth edge is edge ij

#dput(SEind1,"SEind1.txt")

#dput(SEind2,"SEind2.txt")

#dput(ccMat, "W.txt")

# create adjacency information needed for WinBUGS

mkAdj <- function(W){

n <- nrow(W)

adj <- 0

for(i in 1:n){

for(j in 1:n){

if(W[i,j]==1){adj<-append(adj,j)

}

}

}

adj <- adj[-1]

return(adj)

}

adj = mkAdj(ccMat)

###--------------------------- Simulation

set.seed(123456)

M = diag(num)

C = ccMat/num

n = nrow(C)

rho = 1

x = runif(n, -3, 3)

m <- 1 + 2\*x - 3\*sin(x)

data.sim <- function(n=87, tau.phi){

Phi <- rnorm(n)

Phi.Mean <- c()

for (i in 1:n){

Phi.Mean[i] <- rho\*(sum(C[i,-i]\*Phi[-i])/num[i])

}

Phi.Sig <- 1/(tau.phi\*num)

Phi <- rnorm(n, Phi.Mean, Phi.Sig)

E = rep(30, 87)

eta = m + Phi

lambda = exp(eta)

y = rpois(n, lambda = lambda)

return(list(O=y, eta = eta ,E=E, lambda = lambda, u = Phi))

}rep=1

y.rep = eta.rep = lambda.rep = u = l.rep = eta.hat.ebrep = matrix (NA, ncol = rep, nrow = n)

Beta.dc=matrix(NA, ncol=rep, nrow = 2)

sigma.dc=prec.dc=c()

############################## REP for all #####################################

for (q in 1:rep) {

dt <- data.sim(n=87, tau.phi = 0.5)

y.rep[,q] = dt$O

eta.rep[,q] = dt$eta

lambda.rep[,q] = dt$lambda

u[,q] = dt$u

################ Fitting SGLMM with proper CAR using HB method

model <- function(){

# =====

# Likelihood

for(k in 1:K){

for (i in 1 : N) {

O[i,k] ~ dpois(mu[i,k])

log(mu[i,k]) <- log(E[i]) + inprod(X[i,],beta[1,]) + S[k,i]

# Area-specific relative risk

# RR[k,i] <- exp(S[k,i])

}

}

# Proper CAR prior distribution for spatial random effects:

for(k in 1:K){

S[k,1:N] ~ car.normal(adj[], weights[], num[], prec)

}

for(l in 1:sumNumNeigh) {

weights[l] <- 1

}

# Other priors:

# alpha ~ dnorm(0, 0.0001)

for (j in 1:np) {

beta[1,j] ~ dnorm(0, 0.0001)

}

# prior on precision

prec ~ dgamma(0.05, 0.05)

sigma <- sqrt(1/prec) # standard deviation

#S.mean <- sum(S[])

}

#=============================================

#Data

N = length(y.rep[,q])

O = y.rep[,q]

E = dt$E

#X <- bs(m, knots=quantile(m, c(0.25,.5,.75)), degree=3) # By LR

X <- x # Simple Linear Regression

X <- model.matrix(~X)

np = ncol(X)

num = num

sumNumNeigh = sum(num)

######---------------

#adjacency matrix

adj = c(3, 4, 7,

3, 5,

1, 2, 5, 7,

1, 6, 7, 14,

2, 3, 7, 8, 9,

4, 11, 14, 21, 24,

1, 3, 4, 5, 9, 12, 14, 17, 18,

5, 9, 12, 13, 15, 16,

5, 7, 8, 12, 13,

11,

6, 10,

7, 8, 9, 16, 18, 20,

8, 9,

4, 6, 7, 17, 21,

8, 16, 19, 20,

8, 12, 15, 20,

7, 14, 18, 21, 22, 23, 28, 29,

7, 12, 17, 20, 22,

15, 20, 25, 26,

12, 15, 16, 18, 19, 22, 25,

6, 14, 17, 23, 24, 27, 30, 31,

17, 18, 20, 25, 28,

17, 21, 29, 30,

6, 21, 27,

19, 20, 22, 26, 28, 32, 33,

19, 25, 32, 34,

21, 24, 31, 39, 40,

17, 22, 25, 29, 33, 37,

17, 23, 28, 30, 35, 37,

21, 23, 29, 31, 35, 39, 42,

21, 27, 30, 39, 40,

25, 26, 33, 34, 36, 38,

25, 28, 32, 36, 37, 38,

26, 32, 36, 41,

29, 30, 37, 42,

32, 33, 34, 38, 41, 43,

28, 29, 33, 35, 38, 42, 44, 45, 47,

32, 33, 36, 37, 43, 44,

27, 30, 31, 40, 42, 46,

27, 31, 39, 46, 49,

34, 36, 43, 48,

30, 35, 37, 39, 45, 46, 50,

36, 38, 41, 44, 48, 51,

37, 38, 43, 47, 51, 57,

37, 42, 47, 50, 53, 54,

39, 40, 42, 49, 50, 52,

37, 44, 45, 53, 57,

41, 43, 51, 55,

40, 46, 52, 56,

42, 45, 46, 52, 54, 56, 58,

43, 44, 48, 55, 57,

46, 49, 50, 56,

45, 47, 54, 57, 59,

45, 50, 53, 58, 59,

48, 51, 57, 60, 62, 63,

49, 50, 52, 58, 61, 65,

44, 47, 51, 53, 55, 59, 60, 66, 67,

50, 54, 56, 59, 64, 65,

53, 54, 57, 58, 64, 67,

55, 57, 63, 66, 71, 72,

56, 65, 68, 76, 77,

55, 63, 70,

55, 60, 62, 70, 71,

58, 59, 65, 67, 69, 74,

56, 58, 61, 64, 74, 75, 76,

57, 60, 67, 69, 72, 78,

57, 59, 64, 66, 69,

61, 73, 77,

64, 66, 67, 74, 78, 82, 84,

62, 63, 71, 79,

60, 63, 70, 72, 79, 80,

60, 66, 71, 78, 80, 81,

68, 77, 83, 85,

64, 65, 69, 75, 84, 86,

65, 74, 76, 86, 87,

61, 65, 75, 77, 87,

61, 68, 73, 76, 85, 87,

66, 69, 72, 81, 82,

70, 71, 80,

71, 72, 79, 81,

72, 78, 80, 82,

69, 78, 81, 84,

73, 85,

69, 74, 82, 86,

73, 77, 83, 87,

74, 75, 84, 87,

75, 76, 77, 85, 86

)

#K <- 1

#Y <- matrix(c(O), N, K, byrow=F)

#parameters.hb <- c("beta", "prec", "sigma")

#data.hb <- list(O=Y, X=as.matrix(X), E=E, N=N, adj=adj, num=num,

# sumNumNeigh=sumNumNeigh, K=1, np=np)

#initial values:

#inits.hb <- list(list(beta=matrix(rep(0, np),1,np), prec=1),

# list(beta=matrix(rep(0, np),1,np), prec=1))

#modelout.hb <- bugs.fit(data=data.hb, parameters.hb, model,

# inits=inits.hb, n.chains=2, n.iter=3000, n.thin=5, n.burnin=1000,

# program="openbugs", debug=FALSE)

#print(paste("Gelman"))

#print(max((gelman.diag(modelout.hb))$psrf[,1]))

#stats.hb <- summary(modelout.hb,quantiles=c(0.005,0.01,0.025,0.05,0.5,0.95,0.975,0.99,0.995))

#lambda.hb <- as.numeric(lambdamax.diag(modelout.hb))

#prec.hb = stats.hb$statistics[np+1,1]

#sigma.hb = stats.hb$statistics[np+2,1]

#Beta.hb = stats.hb$statistics[1:np,1]

#==================================================

#DC approach:

dat2 <- list(O=dcdim(O), X=X, E=E, N=N, adj=adj, num=num,

sumNumNeigh=sumNumNeigh, np=np, K=1)

parameters.dc <- c("beta", "prec", "sigma")

inits.dc <- list(list(beta=matrix(rep(0, np),1,np), prec=1),

list(beta=matrix(rep(0, np),1,np), prec=1))

K1 <- c(0) #number of clones

dcmod <-dc.fit(dat2, parameters.dc, model, inits.dc, n.iter=5000,

n.burnin=2000, n.thin=10, n.chains=2, n.clones=c(5), multiply=c("K"),

unchanged=c("N", "X", "np", "sumNumNeigh", "E", "num", "adj"),

flavour="bugs", program="openbugs", debug=FALSE)

dcd <- dcdiag(dcmod)

r2 <- dcd[1,4]

mse <- dcd[1,3]

r.hat <- dcd[1,5]

#lambda.ratio <- dcd[1,2]/lambda.hb

stats.dc = summary(dcmod)

sigma.dc[q] = stats.dc$statistics[np+2,1]

prec.dc[q] = stats.dc$statistics[np+1,1]

Beta.dc[,q] = stats.dc$statistics[1:np,1]

#=====================prediction part is deleted==================================================

X.Beta = X%\*%Beta.dc[,q]

##########################################################

eta0=c()

for(i in 1: nrow(l.rep)){

eta0[i]<-eta.rep[which.max(y.rep[i,])]

}

sigma = (sigma.dc \* num)\*(diag(rep(1,87))) # Sigma Matrix

P<- matrix(0, ncol=nrow(eta.rep), nrow= nrow(eta.rep))

dim(P)

for(j in 1:ncol(P)){

for(i in 1:nrow(P)){

if(i==j){ P[i,j]<-(1/exp(eta0[i]))}

}

P

}

Z<- diag(ncol(sigma))

R<- Z%\*%sigma%\*%t(Z)+P

#..######### tolid eta hat EB marhale1

#for(q in 1: ncol(l.rep)){

for(i in 1: nrow(l.rep)){

l.rep[i,q]<- (y.rep[i,q] - exp(eta0[i])+eta0[i]%\*%exp(eta0[i]))/exp(eta0[i])

# }

}

l.rep

#==

dim(y.rep)

#for(q in 1: ncol(eta.hat.ebrep)){

for(i in 1:nrow(eta.hat.ebrep)){

eta.hat.ebrep[i,q]<- X.Beta[i]+ t(Z[i,])%\*%sigma%\*%t(Z)%\*%solve(R)%\*%(l.rep[,q]-X.Beta)

#}

}

eta.hat.ebrep

}

###########################eta.rep vs eta.hat.ebrep #################################

mspe<- function(etahat, etahat.eb){

mspe<- c(rep(0, nrow(eta.rep)))

for(i in 1:nrow(etahat)){

for(j in 1: ncol(etahat))

mspe[i]<-sum(etahat[i,j]-etahat.eb[i,j])^2/ncol(etahat)

}

return(mspe)

}

MSE.LR<- mspe(eta.rep,eta.hat.ebrep)

boxplot(MSE.LR)

boxplot(MSE.LR[which(MSE.LR<1)])

save(MSE.LR, file = "MSE-LR.RData")

save(eta.rep, file = "eta-LR-rep.RData")

save(eta.hat.ebrep, file = "eta-LR-eb.RData")

save(sigma.dc, file = "sigma-dc.LR.RData")

save(y.rep, file = "y-rep-LR.RData")

save(X.Beta, file = "X-Beta-LR.RData")

save(X, file = "X-LR.RData")

save(Beta.dc, file = "Beta-LR.RData")

stat.linear<-stats.dc

# **R code-Intrinsic CAR-semiparametric BIN version With Spline**

rm(list=ls())

library(rgdal)

library(raster)

library(spdep)

library(igraph)

library(mclcar)

library(mvtnorm)

library(coda)

library(BRugs)

library(rjags)

library(R2OpenBUGS)

library(dclone)

library(stats)

library(nlme)

library(MASS)

library(splines)

library(lattice)

####### a spatial generalized semiparametric model with a ICAR spatial effect

###------------------------- Essential function -----------------------###

### define space domain based on Minnesota shape file

G.shape = readOGR(dsn=path.expand("C:\\Users\\Elahe\\Desktop\\Miss Hosseini\\shp\_bdry\_counties\_in\_minnesota"),layer="mn\_county\_boundaries")

minnesota <- G.shape[G.shape$CTYONLY\_ID<88,]

minnesota$CTY\_NAME

plot(G.shape)

nb\_q <- poly2nb(minnesota)

ccMat=nb2mat(nb\_q,style='B') ### weight matrix

#is.symmetric.matrix(ccMat)

num = as.vector(rowSums(ccMat))

##---- define adjacency matrix to use in OpenBUGS

n.site <- dim(ccMat)[1] # n.site: number of areas

n.edge <- sum(ccMat)/2 # n.edge: number of unique pairs

SEind1 <- SEind2 <- 0

matmy <- ccMat

for(i in 1:(n.site-1)){

for(j in (i+1):n.site){

if (ccMat[i,j]>0) {SEind1<-c(SEind1,i)

SEind2<-c(SEind2,j)

matmy[i,j]<-matmy[j,i]<-length(SEind1)-1

}

}

}

SEind1 <- SEind1[-1] # edges sorted by row of upper triangle of the adj. matrix

SEind2 <- SEind2[-1] # SEind1[k]=i and SEind2[k]=j => kth edge is edge ij

dput(SEind1,"SEind1.txt")

dput(SEind2,"SEind2.txt")

dput(ccMat, "W.txt")

# create adjacency information needed for WinBUGS

mkAdj <- function(W){

n <- nrow(W)

adj <- 0

for(i in 1:n){

for(j in 1:n){

if(W[i,j]==1){adj<-append(adj,j)

}

}

}

adj <- adj[-1]

return(adj)

}

adj = mkAdj(ccMat)

###--------------------------- Simulation

set.seed(123456)

M = diag(num)

C = ccMat/num

n = nrow(C)

rho = 1

x = runif(n, -5, 5)

# m <- 1 + 2\*x - 3\*sin(x)

m = 0.1\*exp(x)+5\*sin(x)

data.sim <- function(n=87, tau.phi){

Phi <- rnorm(n)

Phi.Mean <- c()

for (i in 1:n){

Phi.Mean[i] <- rho\*(sum(C[i,-i]\*Phi[-i])/num[i])

}

Phi.Sig <- 1/(tau.phi\*num)

Phi <- rnorm(n, Phi.Mean, Phi.Sig)

E = rep(30, 87)

eta = m + Phi

p = exp(eta)/(1+exp(eta))

y = rbinom(n, E, prob = p)

return(list(O=y, eta = eta ,E = E, p = p, u = Phi))

}

rep=1

y.rep = eta.rep = p.rep = u = l.rep = eta.hat.ebrep = matrix (NA, ncol = rep, nrow = n)

############################## REP for all###################################

for (q in 1:rep) {

dt <- data.sim(n=87, tau.phi = 0.5)

y.rep[,q] = dt$O

eta.rep[,q] = dt$eta

p.rep[,q] = dt$p

u[,q] = dt$u

################ Fitting SGLMM with proper CAR using HB method

model <- function(){

# Likelihood

for(k in 1:K){

for (i in 1 : N) {

O[i,k] ~ dbin(mu[i,k], E[i])

mu[i,k] <- exp(inprod(X[i,],beta[1,]) + S[k,i])/(1+exp(inprod(X[i,],beta[1,]) + S[k,i]))

# Area-specific relative risk

# RR[k,i] <- exp(S[k,i])

}

}

# Proper CAR prior distribution for spatial random effects:

for(k in 1:K){

S[k,1:N] ~ car.normal(adj[], weights[], num[], prec)

}

for(l in 1:sumNumNeigh) {

weights[l] <- 1

}

# Other priors:

# alpha ~ dnorm(0, 0.0001)

for (j in 1:np) {

beta[1,j] ~ dnorm(0, 0.0001)

}

# prior on precision

prec ~ dgamma(0.05, 0.05)

sigma <- sqrt(1/prec) # standard deviation

#S.mean <- sum(S[])

}

#=============================================

#Data

N = length(y.rep[,q])

O = y.rep[,q]

E = dt$E

X <- bs(m, knots=quantile(m, c(0.25,.5,.75)), degree=3) # By Spline

#X <- x # Simple Linear Regression

X <- model.matrix(~X)

np = ncol(X)

num = num

sumNumNeigh = sum(num)

#######adjacency matrix

adj = c(3, 4, 7,

3, 5,

1, 2, 5, 7,

1, 6, 7, 14,

2, 3, 7, 8, 9,

4, 11, 14, 21, 24,

1, 3, 4, 5, 9, 12, 14, 17, 18,

5, 9, 12, 13, 15, 16,

5, 7, 8, 12, 13,

11,

6, 10,

7, 8, 9, 16, 18, 20,

8, 9,

4, 6, 7, 17, 21,

8, 16, 19, 20,

8, 12, 15, 20,

7, 14, 18, 21, 22, 23, 28, 29,

7, 12, 17, 20, 22,

15, 20, 25, 26,

12, 15, 16, 18, 19, 22, 25,

6, 14, 17, 23, 24, 27, 30, 31,

17, 18, 20, 25, 28,

17, 21, 29, 30,

6, 21, 27,

19, 20, 22, 26, 28, 32, 33,

19, 25, 32, 34,

21, 24, 31, 39, 40,

17, 22, 25, 29, 33, 37,

17, 23, 28, 30, 35, 37,

21, 23, 29, 31, 35, 39, 42,

21, 27, 30, 39, 40,

25, 26, 33, 34, 36, 38,

25, 28, 32, 36, 37, 38,

26, 32, 36, 41,

29, 30, 37, 42,

32, 33, 34, 38, 41, 43,

28, 29, 33, 35, 38, 42, 44, 45, 47,

32, 33, 36, 37, 43, 44,

27, 30, 31, 40, 42, 46,

27, 31, 39, 46, 49,

34, 36, 43, 48,

30, 35, 37, 39, 45, 46, 50,

36, 38, 41, 44, 48, 51,

37, 38, 43, 47, 51, 57,

37, 42, 47, 50, 53, 54,

39, 40, 42, 49, 50, 52,

37, 44, 45, 53, 57,

41, 43, 51, 55,

40, 46, 52, 56,

42, 45, 46, 52, 54, 56, 58,

43, 44, 48, 55, 57,

46, 49, 50, 56,

45, 47, 54, 57, 59,

45, 50, 53, 58, 59,

48, 51, 57, 60, 62, 63,

49, 50, 52, 58, 61, 65,

44, 47, 51, 53, 55, 59, 60, 66, 67,

50, 54, 56, 59, 64, 65,

53, 54, 57, 58, 64, 67,

55, 57, 63, 66, 71, 72,

56, 65, 68, 76, 77,

55, 63, 70,

55, 60, 62, 70, 71,

58, 59, 65, 67, 69, 74,

56, 58, 61, 64, 74, 75, 76,

57, 60, 67, 69, 72, 78,

57, 59, 64, 66, 69,

61, 73, 77,

64, 66, 67, 74, 78, 82, 84,

62, 63, 71, 79,

60, 63, 70, 72, 79, 80,

60, 66, 71, 78, 80, 81,

68, 77, 83, 85,

64, 65, 69, 75, 84, 86,

65, 74, 76, 86, 87,

61, 65, 75, 77, 87,

61, 68, 73, 76, 85, 87,

66, 69, 72, 81, 82,

70, 71, 80,

71, 72, 79, 81,

72, 78, 80, 82,

69, 78, 81, 84,

73, 85,

69, 74, 82, 86,

73, 77, 83, 87,

74, 75, 84, 87,

75, 76, 77, 85, 86

)

#K <- 1

#Y <- matrix(c(O), N, K, byrow=F)

#parameters.hb <- c("beta", "prec", "sigma")

#data.hb <- list(O=Y, X=as.matrix(X), E=E, N=N, adj=adj, num=num,

# sumNumNeigh=sumNumNeigh, K=1, np=np)

#initial values:

#inits.hb <- list(list(beta=matrix(rep(0, np),1,np), prec=1),

# list(beta=matrix(rep(0, np),1,np), prec=1))

#modelout.hb <- bugs.fit(data=data.hb, parameters.hb, model,

# inits=inits.hb, n.chains=2, n.iter=3000, n.thin=5, n.burnin=1000,

# program="openbugs", debug=FALSE)

#print(paste("Gelman"))

#print(max((gelman.diag(modelout.hb))$psrf[,1]))

#stats.hb <- summary(modelout.hb,quantiles=c(0.005,0.01,0.025,0.05,0.5,0.95,0.975,0.99,0.995))

#lambda.hb <- as.numeric(lambdamax.diag(modelout.hb))

#prec.hb = stats.hb$statistics[np+1,1]

#sigma.hb = stats.hb$statistics[np+2,1]

#Beta.hb = stats.hb$statistics[1:np,1]

#==================================================

#DC approach:

dat2 <- list(O=dcdim(O), X=X, E=E, N=N, adj=adj, num=num,

sumNumNeigh=sumNumNeigh, np=np, K=1)

parameters.dc <- c("beta", "prec", "sigma")

inits.dc <- list(list(beta=matrix(rep(0, np),1,np), prec=1),

list(beta=matrix(rep(0, np),1,np), prec=1))

K1 <- c(0) #number of clones

dcmod <-dc.fit(dat2, parameters.dc, model, inits.dc, n.iter=8000,

n.burnin=3000, n.thin=10, n.chains=2, n.clones=c(20), multiply=c("K"),

unchanged=c("N", "X", "np", "sumNumNeigh", "E", "num", "adj"),

flavour="bugs", program="openbugs", debug=F)

dcd <- dcdiag(dcmod)

r2 <- dcd[1,4]

mse <- dcd[1,3]

r.hat <- dcd[1,5]

#lambda.ratio <- dcd[1,2]/lambda.hb

stats.dc = summary(dcmod)

sigma.dc = stats.dc$statistics[np+2,1]

prec.dc = stats.dc$statistics[np+1,1]

Beta.dc = stats.dc$statistics[1:np,1]

#=====================prediction part is deleted====================================

X.Beta = X%\*%Beta.dc

##########################################################

p0=c()

for(i in 1: nrow(p.rep)){

p0[i]<-p.rep[i,which.max(y.rep[i,])]

}

sigma = (1/prec.dc \* num)\*(diag(rep(1,87))) # Sigma Matrix

P<- matrix(0, ncol=nrow(p.rep), nrow= nrow(p.rep))

dim(P)

for(j in 1:ncol(P)){

for(i in 1:nrow(P)){

if(i==j){ P[i,j]<-(1/n\*p0[i]%\*%(1-p0[i]))}

}

# P

}

Z<- diag(ncol(sigma))

R<- Z%\*%sigma%\*%t(Z)+P

#..........................................................

for(i in 1: nrow(l.rep)){

# for(q in 1:ncol(l.rep)){

l.rep[i,q]<- (y.rep[i,q] - n\*p0[i] + log(p0[i]/(1-p0[i]))\*(n\*p0[i]%\*%(1-p0[i])))/n\*p0[i]%\*%(1-p0[i])

# }

# l.rep

}

######### tolid eta hat EB marhale1#==

#for(q in 1: ncol(eta.hat.ebrep)){

for(i in 1:nrow(eta.hat.ebrep)){

eta.hat.ebrep[i,q]<- X.Beta[i]+ t(Z[i,])%\*%sigma%\*%t(Z)%\*%solve(R)%\*%(l.rep[,q]-X.Beta)

# }

}

#eta.hat.ebrep

}

########################################################

mspe<- function(etahat, etahat.eb){

mspe<- c(rep(0, nrow(u)))

for(i in 1:nrow(etahat)){

for(j in 1: ncol(etahat))

mspe[i]<-sum(etahat[i,j]-etahat.eb[i,j])^2/ncol(etahat)

}

return(mspe)

}

MSE.Spline.Bin<- mspe(eta.rep,eta.hat.ebrep)

boxplot(MSE.Spline.Bin)

save(MSE.Spline.Bin, file = "MSE-Spline-Bin.RData")

save(eta.rep, file = "eta-Spline-Bin.RData")

save(eta.hat.ebrep, file = "eta-Spline-Bin.RData")

save(X.Beta, file = "XBeta-Spline-Bin.RData")

save(sigma, file = "Sigma-Spline-Bin.RData")

save(y.rep, file = "y-rep-Spline-Bin.RData")

save(p.rep, file = "p-rep-Spline-Bin.RData")

######### tolid eta hat EB marhale1#==

#for(q in 1: ncol(eta.hat.ebrep)){

for(i in 1:nrow(eta.hat.ebrep)){

eta.hat.ebrep[i,q]<- X.Beta[i]+ t(Z[i,])%\*%sigma%\*%t(Z)%\*%solve(R)%\*%(l.rep[,q]-X.Beta)

# }

}

#eta.hat.ebrep

}

########################################################

mspe<- function(etahat, etahat.eb){

mspe<- c(rep(0, nrow(u)))

for(i in 1:nrow(etahat)){

for(j in 1: ncol(etahat))

mspe[i]<-sum(etahat[i,j]-etahat.eb[i,j])^2/ncol(etahat)

}

return(mspe)

}

MSE.Spline.Bin<- mspe(eta.rep,eta.hat.ebrep)

boxplot(MSE.Spline.Bin)

save(MSE.Spline.Bin, file = "MSE-Spline-Bin.RData")

save(eta.rep, file = "eta-Spline-Bin.RData")

save(eta.hat.ebrep, file = "eta-Spline-Bin.RData")

save(X.Beta, file = "XBeta-Spline-Bin.RData")

save(sigma, file = "Sigma-Spline-Bin.RData")

save(y.rep, file = "y-rep-Spline-Bin.RData")

save(p.rep, file = "p-rep-Spline-Bin.RData")

###-------------------------------------------------------------------

# **R code-Intrinsic CAR-semiparametric BIN version NONE Spline**

rm(list=ls())

library(rgdal)

library(raster)

library(spdep)

library(igraph)

library(mclcar)

library(mvtnorm)

library(coda)

library(BRugs)

library(rjags)

library(R2OpenBUGS)

library(dclone)

library(stats)

library(nlme)

library(MASS)

library(splines)

library(lattice)

####### a spatial generalized semiparametric model with a ICAR spatial effect

###------------------------- Essential function -----------------------###

### define space domain based on Minnesota shape file

G.shape = readOGR(dsn=path.expand("C:\\Users\\Elahe\\Desktop\\shp\_bdry\_counties\_in\_minnesota"),layer="mn\_county\_boundaries")

minnesota <- G.shape[G.shape$CTYONLY\_ID<88,]

minnesota$CTY\_NAME

#plot(G.shape)

nb\_q <- poly2nb(minnesota)

ccMat=nb2mat(nb\_q,style='B') ### weight matrix

#is.symmetric.matrix(ccMat)

num = as.vector(rowSums(ccMat))

##---- define adjacency matrix to use in OpenBUGS

n.site <- dim(ccMat)[1] # n.site: number of areas

n.edge <- sum(ccMat)/2 # n.edge: number of unique pairs

SEind1 <- SEind2 <- 0

matmy <- ccMat

for(i in 1:(n.site-1)){

for(j in (i+1):n.site){

if (ccMat[i,j]>0) {SEind1<-c(SEind1,i)

SEind2<-c(SEind2,j)

matmy[i,j]<-matmy[j,i]<-length(SEind1)-1

}

}

}

SEind1 <- SEind1[-1] # edges sorted by row of upper triangle of the adj. matrix

SEind2 <- SEind2[-1] # SEind1[k]=i and SEind2[k]=j => kth edge is edge ij

dput(SEind1,"SEind1.txt")

dput(SEind2,"SEind2.txt")

dput(ccMat, "W.txt")

# create adjacency information needed for WinBUGS

mkAdj <- function(W){

n <- nrow(W)

adj <- 0

for(i in 1:n){

for(j in 1:n){

if(W[i,j]==1){adj<-append(adj,j)

}

}

}

adj <- adj[-1]

return(adj)

}

adj = mkAdj(ccMat)

###--------------------------- Simulation

set.seed(123456)

M = diag(num)

C = ccMat/num

n = nrow(C)

rho = 1

x = sort( runif(n, -3, 3))

#m <- 1 + 2\*x - 3\*sin(x)

m <- x + 2\*x^2 - 3\*(sin(x))^2

data.sim <- function(n=87, tau.phi){

Phi <- rnorm(n)

Phi.Mean <- c()

for (i in 1:n){

Phi.Mean[i] <- rho\*(sum(C[i,-i]\*Phi[-i])/num[i])

}

Phi.Sig <- 1/(tau.phi\*num)

Phi <- rnorm(n, Phi.Mean, Phi.Sig)

E = rep(30, 87)

eta = m + Phi

p = exp(eta)/(1+exp(eta))

y = rbinom(n, E, prob = p)

return(list(O=y, eta = eta ,E = E, p = p, u = Phi))

}

rep=5

y.rep = eta.rep = p.rep = u = l.rep = eta.hat.ebrep = matrix (NA, ncol = rep, nrow = n)

############################## REP for all################################

for (q in 1:rep) {

dt <- data.sim(n=87, tau.phi = 0.5)

y.rep[,q] = dt$O

eta.rep[,q] = dt$eta

p.rep[,q] = dt$p

u[,q] = dt$u

################ Fitting SGLMM with proper CAR using HB method

model <- function(){

# Likelihood

for(k in 1:K){

for (i in 1 : N) {

O[i,k] ~ dbin(mu[i,k], E[i])

mu[i,k] <- exp(inprod(X[i,],beta[1,]) + S[k,i])/(1+exp(inprod(X[i,],beta[1,]) + S[k,i]))

# Area-specific relative risk

# RR[k,i] <- exp(S[k,i])

}

}

# Proper CAR prior distribution for spatial random effects:

for(k in 1:K){

S[k,1:N] ~ car.normal(adj[], weights[], num[], prec)

}

for(l in 1:sumNumNeigh) {

weights[l] <- 1

}

# Other priors:

# alpha ~ dnorm(0, 0.0001)

for (j in 1:np) {

beta[1,j] ~ dnorm(0, 0.0001)

}

# prior on precision

prec ~ dgamma(0.05, 0.05)

sigma <- sqrt(1/prec) # standard deviation

#S.mean <- sum(S[])

}

#=============================================

#Data

N = length(y.rep[,q])

O = y.rep[,q]

E = dt$E

#X <- bs(m, knots=quantile(m, c(0.25,.5,.75)), degree=3) # By Spline

X <- x # Simple Linear Regression

X <- model.matrix(~X)

np = ncol(X)

num = num

sumNumNeigh = sum(num)

######---------------

#adjacency matrix

adj = c(3, 4, 7,

3, 5,

1, 2, 5, 7,

1, 6, 7, 14,

2, 3, 7, 8, 9,

4, 11, 14, 21, 24,

1, 3, 4, 5, 9, 12, 14, 17, 18,

5, 9, 12, 13, 15, 16,

5, 7, 8, 12, 13,

11,

6, 10,

7, 8, 9, 16, 18, 20,

8, 9,

4, 6, 7, 17, 21,

8, 16, 19, 20,

8, 12, 15, 20,

7, 14, 18, 21, 22, 23, 28, 29,

7, 12, 17, 20, 22,

15, 20, 25, 26,

12, 15, 16, 18, 19, 22, 25,

6, 14, 17, 23, 24, 27, 30, 31,

17, 18, 20, 25, 28,

17, 21, 29, 30,

6, 21, 27,

19, 20, 22, 26, 28, 32, 33,

19, 25, 32, 34,

21, 24, 31, 39, 40,

17, 22, 25, 29, 33, 37,

17, 23, 28, 30, 35, 37,

21, 23, 29, 31, 35, 39, 42,

21, 27, 30, 39, 40,

25, 26, 33, 34, 36, 38,

25, 28, 32, 36, 37, 38,

26, 32, 36, 41,

29, 30, 37, 42,

32, 33, 34, 38, 41, 43,

28, 29, 33, 35, 38, 42, 44, 45, 47,

32, 33, 36, 37, 43, 44,

27, 30, 31, 40, 42, 46,

27, 31, 39, 46, 49,

34, 36, 43, 48,

30, 35, 37, 39, 45, 46, 50,

36, 38, 41, 44, 48, 51,

37, 38, 43, 47, 51, 57,

37, 42, 47, 50, 53, 54,

39, 40, 42, 49, 50, 52,

37, 44, 45, 53, 57,

41, 43, 51, 55,

40, 46, 52, 56,

42, 45, 46, 52, 54, 56, 58,

43, 44, 48, 55, 57,

46, 49, 50, 56,

45, 47, 54, 57, 59,

45, 50, 53, 58, 59,

48, 51, 57, 60, 62, 63,

49, 50, 52, 58, 61, 65,

44, 47, 51, 53, 55, 59, 60, 66, 67,

50, 54, 56, 59, 64, 65,

53, 54, 57, 58, 64, 67,

55, 57, 63, 66, 71, 72,

56, 65, 68, 76, 77,

55, 63, 70,

55, 60, 62, 70, 71,

58, 59, 65, 67, 69, 74,

56, 58, 61, 64, 74, 75, 76,

57, 60, 67, 69, 72, 78,

57, 59, 64, 66, 69,

61, 73, 77,

64, 66, 67, 74, 78, 82, 84,

62, 63, 71, 79,

60, 63, 70, 72, 79, 80,

60, 66, 71, 78, 80, 81,

68, 77, 83, 85,

64, 65, 69, 75, 84, 86,

65, 74, 76, 86, 87,

61, 65, 75, 77, 87,

61, 68, 73, 76, 85, 87,

66, 69, 72, 81, 82,

70, 71, 80,

71, 72, 79, 81,

72, 78, 80, 82,

69, 78, 81, 84,

73, 85,

69, 74, 82, 86,

73, 77, 83, 87,

74, 75, 84, 87,

75, 76, 77, 85, 86

)

K <- 1

Y <- matrix(c(O), N, K, byrow=F)

#DC approach:

dat2 <- list(O=dcdim(O), X=X, E=E, N=N, adj=adj, num=num,

sumNumNeigh=sumNumNeigh, np=np, K=1)

parameters.dc <- c("beta", "prec", "sigma")

inits.dc <- list(list(beta=matrix(rep(0, np),1,np), prec=1),

list(beta=matrix(rep(0, np),1,np), prec=1))

K1 <- c(0) #number of clones

dcmod <-dc.fit(dat2, parameters.dc, model, inits.dc, n.iter=3000,

n.burnin=1000, n.thin=10, n.chains=2, n.clones=c(5), multiply=c("K"),

unchanged=c("N", "X", "np", "sumNumNeigh", "E", "num", "adj"),

flavour="bugs", program="openbugs", debug=F)

dcd <- dcdiag(dcmod)

r2 <- dcd[1,4]

mse <- dcd[1,3]

r.hat <- dcd[1,5]

#lambda.ratio <- dcd[1,2]/lambda.hb

stats.dc = summary(dcmod)

sigma.dc = stats.dc$statistics[np+2,1]

prec.dc = stats.dc$statistics[np+1,1]

Beta.dc = stats.dc$statistics[1:np,1]

#=====================prediction part is deleted==================================================

X.Beta = X%\*%Beta.dc

##########################################################

p0=c()

for(i in 1: nrow(p.rep)){

p0[i]<-p.rep[i,which.max(y.rep[i,])]

}

sigma = (sigma.dc \* num)\*(diag(rep(1,87))) # Sigma Matrix

P<- matrix(0, ncol=nrow(p.rep), nrow= nrow(p.rep))

dim(P)

for(j in 1:ncol(P)){

for(i in 1:nrow(P)){

if(i==j){ P[i,j]<-(1/n\*p0[i]%\*%(1-p0[i]))}

}

# P

}

Z<- diag(ncol(sigma))

R<- Z%\*%sigma%\*%t(Z)+P

#..........................................................

for(i in 1: nrow(l.rep)){

# for(q in 1:ncol(l.rep)){

l.rep[i,q]<- (y.rep[i,q] - n\*p0[i] + log(p0[i]/(1-p0[i]))\*(n\*p0[i]%\*%(1-p0[i])))/n\*p0[i]%\*%(1-p0[i])

# }

# l.rep

}

######### tolid eta hat EB marhale1

#==

#for(q in 1: ncol(eta.hat.ebrep)){

for(i in 1:nrow(eta.hat.ebrep)){

eta.hat.ebrep[i,q]<- X.Beta[i]+ t(Z[i,])%\*%sigma%\*%t(Z)%\*%solve(R)%\*%(l.rep[,q]-X.Beta)

# }

}

#eta.hat.ebrep

}

########################################################

mspe<- function(etahat, etahat.eb){

mspe<- c(rep(0, nrow(u)))

for(i in 1:nrow(etahat)){

for(j in 1: ncol(etahat))

mspe[i]<-sum(etahat[i,j]-etahat.eb[i,j])^2/ncol(etahat)

}

return(mspe)

}

MSE.Lin.Bin<- mspe(eta.rep,eta.hat.ebrep)

boxplot(MSE.Lin.Bin)

save(MSE.Lin.Bin, file = "MSE-Lin-Bin.RData")

save(eta.rep, file = "eta-Lin-Bin.RData")

save(eta.hat.ebrep, file = "eta-Lin-Bin.RData")

save(X.Beta, file = "XBeta-Lin-Bin.RData")

save(sigma, file = "sigma-Lin-Bin.RData")

save(y.rep, file = "y-rep-Lin-Bin.RData")

save(p.rep, file = "p-rep-Lin-Bin.RData")

###########################################

plot(density(MSE.Lin.Bin))

lines(density(MSE.Spline.Bin), col=2)

boxplot(MSE.Spline.Bin[which(MSE.Spline.Bin)],MSE.Lin.Bin[which(MSE.Lin.Bin)])

###########################################

plot(sort(X.Beta))

lines(sort(X.Beta2))

lines(sort(m),col=2)

plot(sort(X.Beta), type="l")

lines(sort(X.Beta2), col=2)

lines(sort(m),col=3)

legend("topleft", c("Linear","Spline","m"), col =1:3, lwd = 2, lty = 1 )

p1<- sort(exp(eta.rep+X.Beta)/(1+exp(eta.rep+X.Beta)))

p2<- sort(exp(eta.rep+X.Beta2)/(1+exp(eta.rep+X.Beta2)))

p.m <- sort(exp(eta.rep+m)/(1+exp(eta.rep+m)))

plot(p1, type="l")

lines(p2, col=2)

lines(p.m,col=3)

legend("topleft", c("Linear","Spline","m"), col =1:3, lwd = 2, lty = 1 )