```
# small area logit model
date()
set.seed(1980)
path<-getwd()
library("coda")
library("splines")
library("rbugs")
library("rjags")
library("R2WinBUGS")
library("dclone")
library("cluster")
library("nlme")
library("MASS")
R=~~R~~
s1=~~s1~~
            s=s1+1
            K=1
            m=5
            I=50
            T=5
            Nsp=40
            alpha=0.1
            beta<-0.01#  X coefficient
            rho<-0.4

            sigma2.e=1  #Error
            tau.e=1/(sigma2.e)

            sigma2.nu=1 #area
            tau.nu=1/(sigma2.nu)

            sigma2.sp=1 #spline
            tau.sp=1/(sigma2.sp)

            # generate fixed parts
           # mean.x=0
            x <- runif(I*T,-10,0)

            sp<-rnorm(Nsp,0,sqrt(sigma2.sp))

            degree=1

            #------------------------------- finally the mean is

            matXZ   <-
bs(x,df=Nsp+degree,degree=degree,intercept=FALSE)
            X <- matXZ[,1]
            Z <- matXZ[,-1]

            X    <- cbind(1,X)
            nbetas <- ncol(X)
```

```
            parameters.hb <- c("beta", "py","rho", "sigma2.e",
"sigma2.nu", "sigma2.sp")
            parameters.dc <- c("beta", "rho","sigma2.e", "sigma2.nu",
"sigma2.sp")
            parameters.p  <- c("py")

            model <- function() {
            for(k in 1:K){
            for(i in 1:I){
            for(t in 1:T){
            y[(i+I*(k-1)),t] ~ dbin(py[(i+I*(k-1)),t],m)
            logit(py[(i+I*(k-1)),t]) <- mupy[(i+I*(k-1)),t]
            }
            mupy[(i+I*(k-1)),1:T] ~ dmnorm(mmupy[(i+I*(k-
1)),1:T],tauG[,])
            for(t in 1:T){
            mmupy[(i+I*(k-1)),t]<-inprod(X[(i+(t-
1)*I),],beta[1,])+inprod(Z[(i+(t-1)*I),],sp[,k])+nu[i,k]
            }
            }
            }

            for(l in 1:nbetas){beta[1,l]~dnorm(0.0,1.0E-6)}
            rho ~ dunif(-1,1)
            tau.e ~ dgamma(1.0E-3,1.0E-3)
            tau.nu~dgamma(1.0E-3,1.0E-3)
            tau.sp~dgamma(1.0E-3,1.0E-3)

            sigma2.e<-1/tau.e
            sigma2.sp<-1/tau.sp
            sigma2.nu<-1/tau.nu

            for(k in 1:K){
            for(i in 1:Nsp){
            sp[i,k] ~ dnorm(0,tau.sp)
            }}

            for(k in 1:K){
            for(i in 1:I){
            nu[i,k]~dnorm(0,tau.nu)
            }}


            for(i in 1:T){gam[i,i]<-(1/(1-(rho*rho)))}
            for(i in 1:T-1){gam[i,i+1]<-rho/(1-rho*rho)
            gam[i+1,i]<-rho/(1-rho*rho)}
            for(i in 1:T-2){gam[i,i+2]<-pow(rho,2)/(1-rho*rho)
            gam[i+2,i]<-pow(rho,2)/(1-rho*rho)}
            for(i in 1:T-3){gam[i,i+3]<-pow(rho,3)/(1-rho*rho)
            gam[i+3,i]<-pow(rho,3)/(1-rho*rho)}
            gam[1,T]<-pow(rho,4)/(1-rho*rho)
```

```
            gam[T,1]<-pow(rho,4)/(1-rho*rho)

            sigmaG<-sigma2.e*gam
            invG<-inverse(sigmaG)

            for(i in 1:T){tauG[i,i]<-invG[i,i]}
            for(i in 1:T-1){tauG[i,i+1]<-(invG[i,i+1]+invG[i+1,i])/2
            tauG[i+1,i]<-(invG[i,i+1]+invG[i+1,i])/2}
            for(i in 1:T-2){tauG[i,i+2]<-(invG[i,i+2]+invG[i+2,i])/2
            tauG[i+2,i]<-(invG[i,i+2]+invG[i+2,i])/2}
            for(i in 1:T-3){tauG[i,i+3]<-(invG[i,i+3]+invG[i+3,i])/2
            tauG[i+3,i]<-(invG[i,i+3]+invG[i+3,i])/2}
            tauG[1,T]<-(invG[1,5]+invG[5,1])/2
            tauG[T,1]<-(invG[1,5]+invG[5,1])/2
}
            #----------------------------model for prediction of
mu[,]
            model.p <- function() {
            for(k in 1:K){
            for(i in 1:I){
            for(t in 1:T){
            y[(i+I*(k-1)),t] ~ dbin(py[(i+I*(k-1)),t],m)
            logit(py[(i+I*(k-1)),t]) <- mupy[(i+I*(k-1)),t]
            }
            mupy[(i+I*(k-1)),1:T] ~ dmnorm(mmupy[(i+I*(k-
1)),1:T],tauG[,])
            for(t in 1:T){
            mmupy[(i+I*(k-1)),t]<-inprod(X[(i+(t-
1)*I),],beta[1,])+inprod(Z[(i+(t-1)*I),],sp[,k])+nu[i,k]
            }
            }
            }

            tmp[1:(nbetas+4)] ~ dmnorm(param[], prec[,])
            for(l in 1:nbetas){beta[1,l] <- tmp[l]}
            rho <- (1-step(tmp[(nbetas+1)]+1))*(-
0.9999)+step(tmp[(nbetas+1)]-1)*(0.9999)+(1-step(abs(tmp[(nbetas+1)])-
1))*(tmp[(nbetas+1)])
            sigma2.e  <- abs(tmp[(nbetas+2)])
            sigma2.nu <- abs(tmp[(nbetas+3)])
            sigma2.sp <- abs(tmp[(nbetas+4)])

            tau.e  <- 1/sigma2.e
            tau.nu <- 1/sigma2.nu
            tau.sp <- 1/sigma2.sp

            for(k in 1:K){
            for(i in 1:I){
            nu[i,k] ~ dnorm(0,tau.nu)
            }}

            for(k in 1:K){
```

```
            for(i in 1:Nsp){
            sp[i,k] ~ dnorm(0,tau.sp)
            }}

            for(i in 1:T){gam[i,i]<-(1/(1-(rho*rho)))}
            for(i in 1:T-1){gam[i,i+1]<-rho/(1-rho*rho)
            gam[i+1,i]<-rho/(1-rho*rho)}
            for(i in 1:T-2){gam[i,i+2]<-pow(rho,2)/(1-rho*rho)
            gam[i+2,i]<-pow(rho,2)/(1-rho*rho)}
            for(i in 1:T-3){gam[i,i+3]<-pow(rho,3)/(1-rho*rho)
            gam[i+3,i]<-pow(rho,3)/(1-rho*rho)}
            gam[1,T]<-pow(rho,4)/(1-rho*rho)
            gam[T,1]<-pow(rho,4)/(1-rho*rho)

            sigmaG<-sigma2.e*gam
            invG<-inverse(sigmaG)

            for(i in 1:T){tauG[i,i]<-invG[i,i]}
            for(i in 1:T-1){tauG[i,i+1]<-(invG[i,i+1]+invG[i+1,i])/2
            tauG[i+1,i]<-(invG[i,i+1]+invG[i+1,i])/2}
            for(i in 1:T-2){tauG[i,i+2]<-(invG[i,i+2]+invG[i+2,i])/2
            tauG[i+2,i]<-(invG[i,i+2]+invG[i+2,i])/2}
            for(i in 1:T-3){tauG[i,i+3]<-(invG[i,i+3]+invG[i+3,i])/2
            tauG[i+3,i]<-(invG[i,i+3]+invG[i+3,i])/2}
            tauG[1,T]<-(invG[1,5]+invG[5,1])/2
            tauG[T,1]<-(invG[1,5]+invG[5,1])/2
            }


            stats.hb <-list()
            stats.dc<-list()
            stats.p<-list()
            modelout.sim<-modelout.mcmc<-list()
            mean.hb<-MSE.hb<-mse.hb<-Bias.hb<-
c(rep(0,(nbetas+(I*T)+4)))
            mean.dc<-MSE.dc<-mse.dc<-Bias.dc<-c(rep(0,(nbetas+4)))
            mean.p<-MSE.p<-mse.p<-Bias.p<-c(rep(0,(I*T)))
            covp99.hb<-covp98.hb<-covp95.hb<-covp90.hb<-
c(rep(0,(nbetas+(I*T)+4)))
            covp99.dc<-covp98.dc<-covp95.dc<-covp90.dc<-
c(rep(0,(nbetas+4)))
            covp99.p<-covp98.p<-covp95.p<-covp90.p<-c(rep(0,(I*T)))
            domain99.hb<-domain98.hb<-domain95.hb<-domain90.hb<-0
            domain99.dc<-domain98.dc<-domain95.dc<-domain90.dc<-0
            domain99.p<-domain98.p<-domain95.p<-domain90.p<-0

            paramvalue.hb<-c(rep(0,(nbetas+(I*T)+4)))
            paramvalue.dc<-c(rep(0,(nbetas+4)))
            paramvalue.p<-c(rep(0,(I*T)))
            outbound.hb<-rep(0,R)
            outbound.dc<-rep(0,R)
```

```
            for(n in 1:R){
            ar<-matrix(0,I,T)
            for(j in 1:I){
            ar[j,]<-
arima.sim(model=list(ar=c(rho)),n=T,sd=sqrt(sigma2.e))
            }
            ar2<-c(ar)
            area<-rnorm(I,0,sqrt(sigma2.nu))
            lgtp<-matrix(alpha+beta*X[,-
1]+Z%*%sp+rep(area,T)+ar2,I,T,byrow=F)
            PP.ij<-exp(lgtp)/(1+exp(lgtp))

            y<-matrix(0,I,T)

        for(i in 1:I)
        for(j in 1:T){
        {y[i,j]<-rbinom(1,m, PP.ij[i,j])
        }}



            # ------------------------------------------------------HB
model
            if(n>s1){
            data.sim<-list(y=y, X=X, Z=Z, I=I, T=T, m=m, Nsp=Nsp,
nbetas=nbetas, K=1)
            print(paste("---------------- HB model",n,"------------"))
            modelout.sim <- jags.fit(data=data.sim,parameters.hb,
model,n.chains=2,thin = 5, n.iter=10000, n.adapt=5000)
            update(updated.model(modelout.sim),5000)
            modelout.mcmc<-
coda.samples(updated.model(modelout.sim),parameters.hb,n.iter=10000,th
in=5)
            num<-0
            while( (max ((gelman.diag(modelout.mcmc))$psrf[,1]) >1.05)
& num<20){
            update(updated.model(modelout.sim),5000)
            modelout.mcmc<-
coda.samples(updated.model(modelout.sim),parameters.hb,n.iter=10000,th
in=5)
            num<-num+1
            }
            print(paste("model",n))
            print(paste("num",num))
            print(paste("Gelman"))
            print(max((gelman.diag(modelout.mcmc))$psrf[,1]))
            stats.hb<-
summary(modelout.mcmc,quantiles=c(0.005,0.01,0.025,0.05,0.95,0.975,0.9
9,0.995))
            print(stats.hb)
            outbound.hb[n]<-0
```

```
            if(max(abs((stats.hb[[1]][c(1,2,(I*T+3):(I*T+6)),1])-
c(alpha,beta,rho,sigma2.e,sigma2.nu,sigma2.sp)))>4){
            outbound.hb[n]<-1
            }
            print(paste("outbound for HB",n, "is", outbound.hb[n]))
            print(paste("lambda for model",n))
            lambda.hb<-as.numeric(lambdamax.diag(modelout.mcmc))
            print(paste("lambda.hb",lambda.hb))

            if(outbound.hb[n]==0){

            print(paste("-----------------------------------------
----------------------"))
            print(paste("---------------------------------- Data
cloning number",n,"--------------------"))
            r2<-1
            mse<-1
            lambda.ratio<-1
            r.hat=2
            num<-0
            indx<-0
            Kclone<-c(20,30,40,80,120)
            while((((r2>0.01 & mse>0.01) | lambda.ratio>=0.05 |
r.hat>1.05) & num<5) ){
            indx<-indx+1
            K1<-Kclone[indx]
            #cl<-makeCluster(2,type="SOCK")
            dcmod<-dc.fit(list(y=dclone(y,K1),  X=X, Z=Z, I=I, T=T,
m=m, Nsp=Nsp, nbetas=nbetas, K=1),
            parameters.dc, model,
n.clones=c(K1),multiply=c("K"),unchanged=c("I","T","Nsp","m","nbetas")
,
            n.chains=2, n.adapt=5000, n.update=5000,n.iter=10000,
thin=5)
            dcd<-dcdiag(dcmod)
            r2<-dcd[1,4]
            r.hat<-dcd[1,5]
            mse<-dcd[1,3]
            lambda.ratio<-dcd[1,2]/lambda.hb
            #stopCluster(cl)
            num<-num+1
            print(dcd)
            }
            print(paste("num",num))
            stats.dc<-
summary(dcmod,quantiles=c(0.005,0.01,0.025,0.05,0.95,0.975,0.99,0.995)
)
            print(stats.dc)
            outbound.dc[n]<-0
            if(max(abs((stats.dc[[1]][,1])-
c(alpha,beta,rho,sigma2.e,sigma2.nu,sigma2.sp)))>4){
            outbound.dc[n]<-1
```

```
            }
            print(paste("outbound for DC",n, "is", outbound.dc[n]))
            if(outbound.dc[n]==0){
            if(((r2<0.01) || (mse<0.01)) && (lambda.ratio<0.05) &&
(r.hat<1.05)){

            paramvalue.hb<-
c(alpha,beta,PP.ij,rho,sigma2.e,sigma2.nu,sigma2.sp)
            print(paste("RB, MSE and covarage for model",n))
            mean.hb<-mean.hb+stats.hb[[1]][,1]
            covp99.hb<-covp99.hb+1*((stats.hb[[2]][,1]<paramvalue.hb &
paramvalue.hb<stats.hb[[2]][,8] ))
            covp98.hb<-covp98.hb+1*((stats.hb[[2]][,2]<paramvalue.hb &
paramvalue.hb<stats.hb[[2]][,7] ))
            covp95.hb<-covp95.hb+1*((stats.hb[[2]][,3]<paramvalue.hb &
paramvalue.hb<stats.hb[[2]][,6] ))
            covp90.hb<-covp90.hb+1*((stats.hb[[2]][,4]<paramvalue.hb &
paramvalue.hb<stats.hb[[2]][,5] ))
            domain99.hb<-domain99.hb+((stats.hb[[2]][,8]-
stats.hb[[2]][,1] ))
            domain98.hb<-domain98.hb+((stats.hb[[2]][,7]-
stats.hb[[2]][,2] ))
            domain95.hb<-domain95.hb+((stats.hb[[2]][,6]-
stats.hb[[2]][,3] ))
            domain90.hb<-domain90.hb+((stats.hb[[2]][,5]-
stats.hb[[2]][,4] ))

            MSE.hb<-MSE.hb+((stats.hb[[1]][,1]-paramvalue.hb)^2)
            Bias.hb<-Bias.hb+sqrt((stats.hb[[1]][,1]-paramvalue.hb)^2)
            mse.hb<-mse.hb+((stats.hb[[1]][,2])^2)

            paramvalue.dc<-
c(alpha,beta,rho,sigma2.e,sigma2.nu,sigma2.sp)
            mean.dc<-mean.dc+stats.dc[[1]][,1]
            MSE.dc<-MSE.dc+((stats.dc[[1]][,1]-paramvalue.dc)^2)

            Bias.dc<-Bias.dc+sqrt((stats.dc[[1]][,1]-paramvalue.dc)^2)


            #------------------------Prediction of random effects:
            print(paste("Prediction for Data cloning model ",n))
            prec <- make.symmetric(K1*solve(vcov(dcmod)))
            prdat <-list(y=y, X=X, Z=Z, I=I, T=T, Nsp=Nsp, m=m,
nbetas=nbetas, param = coef(dcmod), prec = prec,K=1)
            mod1.1.pr <- jags.fit(prdat, parameters.p, model.p,
n.chains=2, n.adapt=5000, n.update=5000, n.iter=10000, thin=5)

            update(updated.model(mod1.1.pr), 5000)
            mod1.2.pr <- coda.samples(updated.model(mod1.1.pr),
parameters.p, n.iter=10000, thin = 5)

            num<-0
```

```
            while( (max ((gelman.diag(mod1.2.pr))$psrf[,1]) >1.05) &
num<20){
            update(updated.model(mod1.1.pr), 5000)
            mod1.2.pr <- coda.samples(updated.model(mod1.1.pr),
parameters.p, n.iter=10000, thin = 5)
            num<-num+1
            }
            print(paste("num",num))
            stats.p<-
summary(mod1.2.pr,quantiles=c(0.005,0.01,0.025,0.05,0.95,0.975,0.99,0.
995))
            paramvalue.p<-c(PP.ij)
            print(paste("---------------------------RB, MSE and
covarage for model",n,"---------------"))
            mean.p<-mean.p+stats.p[[1]][,1]
            covp99.p<-covp99.p+1*((stats.p[[2]][,1]<paramvalue.p &
paramvalue.p<stats.p[[2]][,8] ))
            covp98.p<-covp98.p+1*((stats.p[[2]][,2]<paramvalue.p &
paramvalue.p<stats.p[[2]][,7] ))
            covp95.p<-covp95.p+1*((stats.p[[2]][,3]<paramvalue.p &
paramvalue.p<stats.p[[2]][,6] ))
            covp90.p<-covp90.p+1*((stats.p[[2]][,4]<paramvalue.p &
paramvalue.p<stats.p[[2]][,5] ))
            domain99.p<-domain99.p+((stats.p[[2]][,8]-stats.p[[2]][,1]
))
            domain98.p<-domain98.p+((stats.p[[2]][,7]-stats.p[[2]][,2]
))
            domain95.p<-domain95.p+((stats.p[[2]][,6]-stats.p[[2]][,3]
))
            domain90.p<-domain90.p+((stats.p[[2]][,5]-stats.p[[2]][,4]
))
            MSE.p<-MSE.p+((stats.p[[1]][,1]-paramvalue.p)^2)
            Bias.p<-Bias.p+sqrt((stats.p[[1]][,1]-paramvalue.p)^2)
            mse.p<-mse.p+((stats.p[[1]][,2])^2)


            #----------------------------
            }
                     else { outbound.dc[n]<-1}
                }
            }
            }
            }
date()
outbound.hb[R]
outbound.dc[R]
```