```
# small area Poisson 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=1
        s1=0
        s=s1+1

K=1
I=50
T=5
Nit=3
rho=0.4

Nsp=min(40,floor(I*T/4))


alpha <-0.1
beta  <-0.01               #  X coefficient

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

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

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

# generate fixed parts
mean.x=0
x <- rnorm(I*T, mean.x, 1)

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

degree=1
nbetas <- degree+1

x.quan<-as.integer((I*T)/(Nsp+1)*c(1:Nsp))
x.sort<-sort(x)
knot<-x.sort[x.quan]
```

```
pow <- function(a,b) a^b
step <- function(a) (a>0)*1

X <- matrix(NA, I*T, degree+1)
Z <- matrix(NA, I*T, Nsp)
u <- matrix(NA, I*T, Nsp)
MUZ <- matrix(NA, I*T, Nsp)
MUX<-matrix(NA, I*T, nbetas)

for (i in 1:(I*T))
{
        for (l in 1:(degree+1)){
                X[i,l]<-pow(x[i],l-1)
        }
}
for (i in 1:(I*T))
{
        for (k in 1:Nsp)
        {
                u[i,k]<-(x[i]-knot[k])*step(x[i]-knot[k])
                Z[i,k]<-pow(u[i,k],degree)
        }
}
for (i in 1:(I*T))
{
        #MUZ[i,]<-colMeans(Z[(1+(i-1)*m):(i*m),])
        MUX[i,]<-c(1,mean.x)
}
MUZ=Z

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


model.t.hb <- function() {
        for(k in 1:K){
                for(i in 1:I){
                  for(t in 1:T){
                        y[(i+I*(k-1)),t] ~ dpois(Nit*mu[(i+I*(k-
1)),t])
                        log[mu[(i+I*(k-1)),t]] <- mmu[(i+I*(k-1)),t]
                        mmu[(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.nu<-1/tau.nu
        sigma2.sp<-1/tau.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){tauE[i,i]<-1}
        for(i in 1:T-1){tauE[i,i+1]<-0
                        tauE[i+1,i]<-0}
        for(i in 1:T-2){tauE[i,i+2]<-0
                        tauE[i+2,i]<-0}
        for(i in 1:T-3){tauE[i,i+3]<-0
                        tauE[i+3,i]<-0}
        tauE[1,T]<-0
        tauE[T,1]<-0

        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.t.p <- function() {
  for(k in 1:K){
    for(i in 1:I){
      for(t in 1:T){
        y[(i+I*(k-1)),t] ~ dpois(Nit*mu[(i+I*(k-1)),t])
        log[mu[(i+I*(k-1)),t]] <- mmu[(i+I*(k-1)),t]
        mmu[(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){tauE[i,i]<-1}
        for(i in 1:T-1){tauE[i,i+1]<-0
                        tauE[i+1,i]<-0}
        for(i in 1:T-2){tauE[i,i+2]<-0
                        tauE[i+2,i]<-0}
        for(i in 1:T-3){tauE[i,i+3]<-0
                        tauE[i+3,i]<-0}
        tauE[1,T]<-0
        tauE[T,1]<-0

        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.t.hb <-list()
stats.t.dc<-list()
stats.t.p<-list()


modelout.sim<-modelout.mcmc<-list()

mean.t.hb<-MSE.t.hb<-mse.t.hb<-Bias.t.hb<-sd.t.hb<-rep(0,nbetas+I*T+4)
mean.t.dc<-MSE.t.dc<-mse.t.dc<-Bias.t.dc<-sd.t.dc<-rep(0,nbetas+4)
mean.t.p<-MSE.t.p<-mse.t.p<-Bias.t.p<-sd.t.p<-rep(0,I*T)
covp99.t.hb<-covp98.t.hb<-covp95.t.hb<-covp90.t.hb<-
rep(0,nbetas+I*T+4)
covp99.t.dc<-covp98.t.dc<-covp95.t.dc<-covp90.t.dc<-rep(0,nbetas+4)
covp99.t.p<-covp98.t.p<-covp95.t.p<-covp90.t.p<-rep(0,I*T)
domain99.t.hb<-domain98.t.hb<-domain95.t.hb<-domain90.t.hb<-
rep(0,nbetas+I*T+4)
domain99.t.dc<-domain98.t.dc<-domain95.t.dc<-domain90.t.dc<-
rep(0,nbetas+4)
domain99.t.p<-domain98.t.p<-domain95.t.p<-domain90.t.p<-rep(0,I*T)


paramvalue.t.hb<-rep(0,nbetas+I*T+4)
paramvalue.t.dc<-rep(0,nbetas+4)
paramvalue.t.p<-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))
        logtp<-matrix(alpha+beta*X[,-
1]+Z%*%sp+rep(area,T)+ar2,I,T,byrow=F)
        MU<-exp(logtp)

        y<-matrix(0,I,T)

         for(i in 1:I)
          for(j in 1:T){
            {y[i,j]<-rpois(1,Nit*exp(logtp[i,j])
            }}


        if(n>s1){
                # ------------------------------------------------------
HB for true model
                data.sim<-list(y=y, X=X, Z=Z, I=I, T=T,Nit=Nit,
Nsp=Nsp, nbetas=nbetas, K=1)
                print(paste("---------------- HB model",n,"-----------
-"))
                modelout.sim <- jags.fit(data=data.sim,
parameters.t.hb, model.t.hb, n.chains=2, thin = 5, n.iter=10000,
n.adapt=10000)
                update(updated.model(modelout.sim),10000)
                modelout.mcmc<-
coda.samples(updated.model(modelout.sim), parameters.t.hb,
n.iter=10000, thin=5)
                num<-0
                while( (max ((gelman.diag(modelout.mcmc))$psrf[,1])
>1.05) & num<20){
                        update(updated.model(modelout.sim),10000)
                        modelout.mcmc<-
coda.samples(updated.model(modelout.sim), parameters.t.hb,
n.iter=10000, thin=5)
                        num<-num+1
                }
                print(paste("model",n))
                print(paste("num",num))
                print(paste("Gelman"))
                print(max((gelman.diag(modelout.mcmc))$psrf[,1]))
                stats.t.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.t.hb)
```

```
                print(paste("Lambda for true model",n))
                lambda.t.hb<-as.numeric(lambdamax.diag(modelout.mcmc))
                print(paste("lambda.t.hb",lambda.t.hb))

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

                MSE.t.hb<-MSE.t.hb+((stats.t.hb[[1]][,1]-
paramvalue.t.hb)^2)
                Bias.t.hb<-Bias.t.hb+sqrt((stats.t.hb[[1]][,1]-
paramvalue.t.hb)^2)
                mse.t.hb<-mse.t.hb+((stats.t.hb[[1]][,2])^2)
                sd.t.hb<-sd.t.hb+((stats.t.hb[[1]][,2]))
                #-------------------------  DC for true model

                K1<-0 #initial number of clones
                r2<-1
                mse<-1
                lambda.ratio<-1
                r.hat=2
                num<-0
                while((((r2>0.01 & mse>0.01) | lambda.ratio>=0.05 |
r.hat>1.05) & num<5) ){
                        K1<-K1+80
                        #cl<-makeCluster(2,type="SOCK")
```

```
                        dcmod<-dc.fit(list(y=dclone(y,K1), X=X, Z=Z,
I=I, T=T, Nsp=Nsp,Nit=Nit,nbetas=nbetas, K=1), parameters.t.dc,
model.t.hb,
n.clones=c(K1),multiply=c("K"),unchanged=c("I","T","nbetas","X","Z",
"Nsp", "Nit"), n.chains=2, n.adapt=10000, n.update=10000,n.iter=10000,
thin=10)
                        dcd<-dcdiag(dcmod)
                        r2<-dcd[1,4]
                        r.hat<-dcd[1,5]
                        mse<-dcd[1,3]
                        lambda.ratio<-dcd[1,2]/lambda.t.hb
                        #stopCluster(cl)
                        num<-num+1
                        print(dcd)
                }
                paramvalue.t.dc<-c(alpha,beta, rho,
sigma2.e,sigma2.nu, sigma2.sp)
                print(paste("num",num))
                stats.t.dc<-
summary(dcmod,quantiles=c(0.005,0.01,0.025,0.05,0.95,0.975,0.99,0.995)
)
                print(stats.t.dc)

                if(max(abs((stats.t.dc[[1]][,1])-c(alpha,beta, rho,
sigma2.e,sigma2.nu, sigma2.sp)))>4){
                        outbound.dc[n]<-1
                }
                print(paste("outbound for true DC",n, "is",
outbound.dc[n]))

                mean.t.dc<-mean.t.dc+stats.t.dc[[1]][,1]
                MSE.t.dc<-MSE.t.dc+((stats.t.dc[[1]][,1]-
paramvalue.t.dc)^2)
                mse.t.dc<-mse.t.dc+((stats.t.dc[[1]][,3])^2)
                sd.t.dc<-sd.t.dc+(stats.t.dc[[1]][,3])
                Bias.t.dc<-Bias.t.dc+sqrt((stats.t.dc[[1]][,1]-
paramvalue.t.dc)^2)


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

                update(updated.model(mod1.1.pr), 10000)
```

```
                mod1.2.pr <- coda.samples(updated.model(mod1.1.pr),
parameters.t.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), 10000)
                        mod1.2.pr <-
coda.samples(updated.model(mod1.1.pr), parameters.t.p, n.iter=10000,
thin = 5)
                        num<-num+1
                }
                print(paste("num",num))
                stats.t.p<-
summary(mod1.2.pr,quantiles=c(0.005,0.01,0.025,0.05,0.95,0.975,0.99,0.
995))
                paramvalue.t.p<-c(MU)
                print(paste("---------------------------RB, MSE and
covarage for true model",n,"---------------"))
                mean.t.p<-mean.t.p+stats.t.p[[1]][,1]
                covp99.t.p<-
covp99.t.p+1*((stats.t.p[[2]][,1]<paramvalue.t.p &
paramvalue.t.p<stats.t.p[[2]][,8] ))
                covp98.t.p<-
covp98.t.p+1*((stats.t.p[[2]][,2]<paramvalue.t.p &
paramvalue.t.p<stats.t.p[[2]][,7] ))
                covp95.t.p<-
covp95.t.p+1*((stats.t.p[[2]][,3]<paramvalue.t.p &
paramvalue.t.p<stats.t.p[[2]][,6] ))
                covp90.t.p<-
covp90.t.p+1*((stats.t.p[[2]][,4]<paramvalue.t.p &
paramvalue.t.p<stats.t.p[[2]][,5] ))
                domain99.t.p<-domain99.t.p+((stats.t.p[[2]][,8]-
stats.t.p[[2]][,1] ))
                domain98.t.p<-domain98.t.p+((stats.t.p[[2]][,7]-
stats.t.p[[2]][,2] ))
                domain95.t.p<-domain95.t.p+((stats.t.p[[2]][,6]-
stats.t.p[[2]][,3] ))
                domain90.t.p<-domain90.t.p+((stats.t.p[[2]][,5]-
stats.t.p[[2]][,4] ))
                MSE.t.p<-MSE.t.p+((stats.t.p[[1]][,1]-
paramvalue.t.p)^2)
                Bias.t.p<-Bias.t.p+sqrt((stats.t.p[[1]][,1]-
paramvalue.t.p)^2)
                mse.t.p<-mse.t.p+((stats.t.p[[1]][,2])^2)
                sd.t.p<-sd.t.p+(stats.t.p[[1]][,2])

        }
}
date()
outbound.hb[R]
outbound.dc[R]
```