Homework2

Seungjun Lee

2022 10 17

Before solving the problems, I implemented some useful functions. (Appendix 1-(a)) (ploteqc, hpd, expCov, sqeCov, …)

1-(a) Simulate spatially correlated binary data with N= 200.

I made a function to simulate the binary samples with sample size n.

```
Simulate_binary_data = function(n=200,cv=0.2){
  set.seed(55555)
  ncv=n*cv
  beta=c(1,1) ; phi=0.2 ; sigma2=1


  gridLocation<-cbind(runif(n,min = 0,max = 1),runif(n,min = 0,max = 1))
  CVgridLocation<-cbind(runif(ncv,min = 0,max = 1),runif(ncv,min = 0,max = 1))

  comboLocation<-rbind(gridLocation,CVgridLocation)
  distMatFull<-as.matrix(rdist(comboLocation))
  modInd<-1:n
  CVInd<-(n+1):nrow(distMatFull)
  distMatMod<-distMatFull[modInd,modInd]
  # Covariates
  XMat<-cbind(runif(n,-1,1),runif(n,-1,1))
  XMatCV<-cbind(runif(ncv,-1,1),runif(ncv,-1,1))
  XB<-XMat%*%beta
  cvXB<-XMatCV%*%beta
  XBFull<-rbind(XB,cvXB)

  # Covariance Matrix
  CovMat<-sigma2*matCov(distMatFull,phi)

  # Latent Gaussian Random Field
  gpWFull <- as.numeric(rmvnorm(n=1,mean=rep(0,nrow(CovMat)),sigma = CovMat,m
ethod = "chol"))
  pWFullLinear<-gpWFull+XBFull
  pWFullBin<-exp(gpWFull+XBFull)/(1+exp(gpWFull+XBFull))

  # Observations
  obsFullLinear<-pWFullLinear
  obsFullBin<-sapply(pWFullBin,rbinom,n=1,size=1)
```

```
  ##################
  # Model Sample
  # Latent Process
  gpWMod<-gpWFull[modInd]
  # Expected Value
  pWModLinear<-pWFullLinear[modInd] #400*1
  pWModBin<-pWFullBin[modInd]

  # Observations
  obsModLinear<-obsFullLinear[modInd]
  obsModBin<-obsFullBin[modInd]

  # CV Sample
  # Latent Process
  gpWCV<-gpWFull[CVInd]
  # Expected Value
  pWCVLinear<-pWFullLinear[CVInd]
  pWCVBin<-pWFullBin[CVInd]
  # Observations
  obsCVLinear<-obsFullLinear[CVInd] # 100*1
  obsCVBin<-obsFullBin[CVInd]

  return(list("obsModBin"=obsModBin,"obsCVBin"=obsCVBin,"XMat"=XMat,"XmatCV"=
XMatCV,
              "distMatMod"=distMatMod,"gridLocation"=gridLocation,"CVgridLoca
tion"=CVgridLocation))
}

n =200
sim.data = Simulate_binary_data(n=n,cv=0.2) # sumulated data
```

1-(b) Report some quantities.(Computing time, trace plots, acc rate, prediction acc)

The iterations(about MCMC samples) are 50,000.

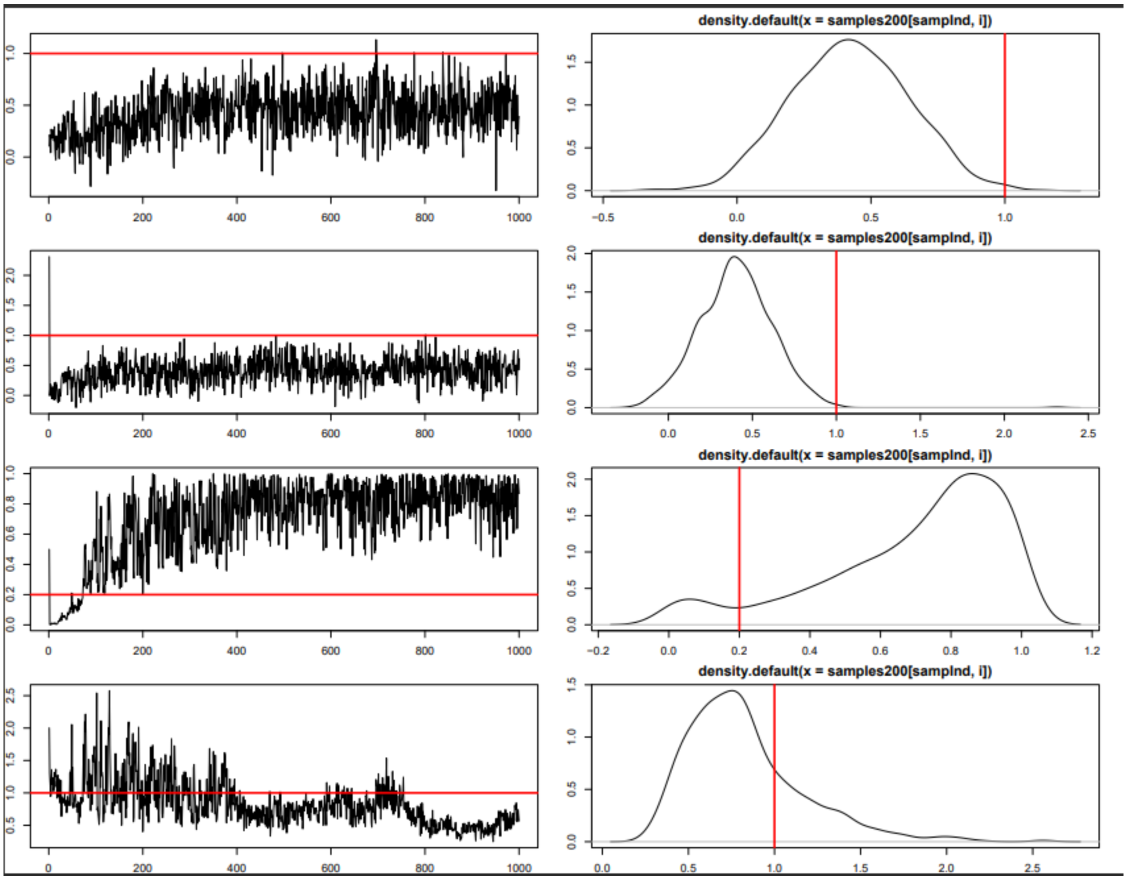
```
niter=50000
obsModBinom = sim.data$obsModBin
XMat = sim.data$XMat
distMatMod = sim.data$distMatMod

consts   <- list(n=n,X=XMat,dists=distMatMod,mn=rep(0,n))
data     <- list(Z=obsModBinom)
inits    <- list(beta1=rnorm(1),beta2=rnorm(1),phi=0.5,sigma2=2,
                 W=rnorm(n))
```

```
# Run MCMC


nimble_model = nimbleModel(model_string, data = data, inits = inits,constants
=consts)
MCMC_conf = configureMCMC(nimble_model,monitors = c("beta1", "beta2","phi","s
igma2"),
                          control = list(adaptFactorExponent = 1))
RMCMC = buildMCMC(MCMC_conf,samplesAsCodaMCMC=TRUE,WAIC=FALSE,summary=FALSE,
thin=20,
                  niter = niter, nburnin = 0, nchains = 1)
Cmodel = compileNimble(nimble_model)
Cmcmc = compileNimble(RMCMC,project = nimble_model)

pt<-proc.time()
Cmcmc$run(niter = niter)
samples200 = as.matrix(Cmcmc$mvSamples)

# samples200  <- nimbleMCMC(model_string, data = data, inits = inits,
#                           constants=consts,
#                           monitors = c("beta1", "beta2","phi","sigma2"),
#                           samplesAsCodaMCMC=TRUE,WAIC=FALSE,summary=FALSE,
thin=20,
#                           niter = niter, nburnin = 0, nchains = 1)
ptFinal200<-proc.time()-pt

# computing time
ptFinal200

##  사용자  시스템 elapsed
##  129.52    0.29  141.03
```

Computing time is around 141 seconds(n=200).

```
# trace plot
pdf(file = "BinomResults200.pdf",width=11,height=8.5)
par(mfrow=c(4,2),mar=c(2,2,2,2))
sampInd<-floor(seq(1,nrow(samples200),length.out = 1000))
beta=c(1,1) ; phi=0.2 ; sigma2=1

for(i in 1:4){
  plot.ts(samples200[sampInd,i]); abline(h=c(beta,phi,sigma2)[i],col="red",lw
d=2)
  plot(density(samples200[sampInd,i])); abline(v=c(beta,phi,sigma2)[i],col="r
ed",lwd=2)
}

summaryMat200<-list()
```

```r
summaryMat200[[1]]<-round(summaryFunction(samples200[,c("beta1", "beta2","phi","sigma2")],
                                          time=ptFinal200[3]),3)
```



Above trace plots are about β_1, β_2 , ρ and σ^2 when n=200 and iteration=50,000.

The posterior mean, highest posterior density, and acc rate for some parameters are as follow.

```r
# posterior mean, hpd, acc rate
summaryMat200[[1]]
save(samples200,file="BinomMCMCsamples200.RData")
save(summaryMat200,samples200,ptFinal200,file="BinomMCMCResults200.RData")

##                 beta1    beta2     phi  sigma2
## Mean            0.409    0.319   0.751   0.608
## 95%CI-Low      -0.009   -0.054   0.107   0.167
## 95%CI-High      0.818    0.682   1.000   1.331
## Accept          0.442    0.441   0.442   0.411
## BMSE            0.007    0.003   0.016   0.023
## 0.01 x mean     0.004    0.003   0.008   0.006
## ESS          4153.883 7819.504 631.567 610.617
## ESS/sec        29.454   55.446   4.478   4.330
```

After taking a burn-in and thinning, reported the prediction accuracy. I used conditional distribution of $\eta_{pred}|\eta_{obs}$.

$$\eta_{pred}|\eta_{obs}, \beta, \tau, Y \sim N(m_{pred}, V_{pred})$$

$$m_{pred} = X_{pred}\beta + \gamma(\rho)'\Gamma(\rho)^{-1}(Y - X\beta)$$

$$V_{pred} = \sigma^2[\Gamma_{pred}(\rho) - \gamma(\rho)'\Gamma(\rho)^{-1}\gamma(\rho)]$$

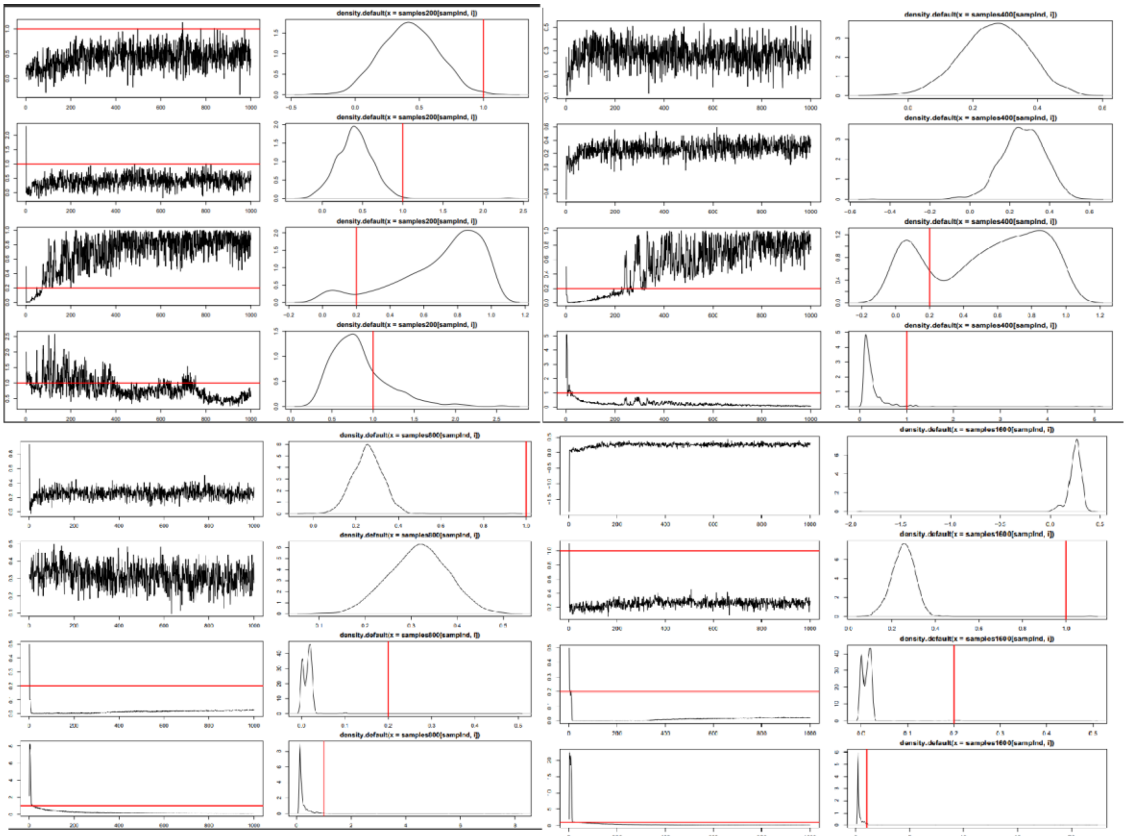
$$p_{pred} = \frac{exp(\eta_{pred})}{1 + exp(\eta_{pred})}$$

```
summaryMat200<-list()
summaryMat200[[1]]<-round(summaryFunction(samples200[,c("beta1", "beta2","phi
","sigma2")],
                                          time=ptFinal200[3]),3)
# posterior mean, hpd, acc rate
save(samples200,file="BinomMCMCsamples200.RData")
save(summaryMat200,samples200,ptFinal200,file="BinomMCMCResults200.RData")

# prediction acc
burnin <- 100
s2.final <- samples200[,c("sigma2")][-(1:burnin)]
beta.final <- cbind(samples200[,c("beta1")][-(1:burnin)],samples200[,c("beta2
")][-(1:burnin)])
rho.final <- samples200[,c("phi")][-(1:burnin)]


obs.grid = sim.data$gridLocation
pred.grid = sim.data$CVgridLocation
X = sim.data$XMat
Xpred = sim.data$XmatCV

full.grid = rbind(obs.grid,pred.grid)
full.X = rbind(X,Xpred)
# fixed part
full.XB = full.X%*%t(beta.final)
d <- rdist.earth(coordinates(obs.grid))
dcross <- rdist.earth(coordinates(obs.grid), coordinates(pred.grid)) # 200*40
dpred <- rdist.earth(coordinates(pred.grid)) # 40*40

eta.mat <- matrix(NA, nrow = nrow(pred.grid), ncol = niter-burnin) # 40*
```

```r
# random part
distMatFull = rdist.earth(coordinates(full.grid))

for(j in 1:ncol(eta.mat)){
  if(j%%200 == 0){print(j)}

  # Construct the covariance matrices
  Gamma <- exp(-d/rho.final[j])
  Ginv <- solve(Gamma)
  g <- exp(-dcross/rho.final[j])
  Gpred <- exp(-dpred/rho.final[j])

  m <- Xpred %*% beta.final[j,] + t(g) %*% Ginv %*%
    (y - X %*% beta.final[j,])
  V <- s2.final[j] * (Gpred - t(g)%*%Ginv%*%g)
  eta.mat[,j] <- rmvnorm(1, m, V, method = "svd")
}
save(eta.mat,file="Binom_eta200.RData")
dim(eta.mat)

acc_lst = NULL
actual.y = sim.data$obsCVBin

for(i in 1:ncol(eta.mat)){
  bin.p = exp(eta.mat[,i])/(1+exp(eta.mat[,i]))
  pred.y = sapply(bin.p,rbinom,n=1,size=1)
  ct = table(pred.y,actual.y)
  acc_lst[i] = sum(diag(ct))/sum(ct)

}

length(acc_lst)
mean(acc_lst)

## [1] 49900

## [1] 0.5032445
```

1-(c)

The Detailed codes are in appendix 1-(c).



Above trace plots are about β_1, β_2 , ρ and σ^2 when n=200,400,800 and 1600 with iteration=50,000. when the sample size goes large, we need to sample more MCMC samples to get well converged samples. In this situation, we fixed the iteration. Thus, The trace plots are getting worse and worse.

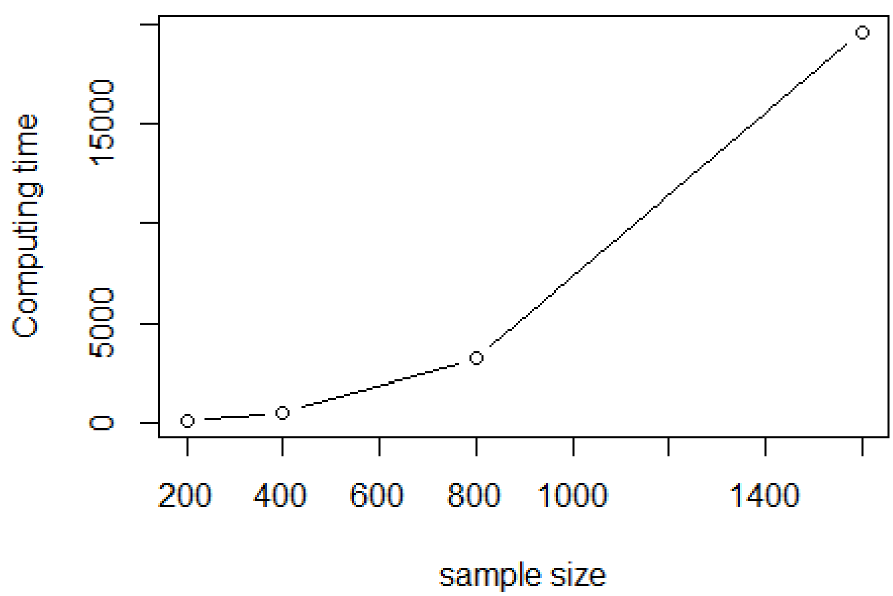
```
summaryMat400[[1]] # summary table when n=400

##                  beta1    beta2     phi  sigma2
## Mean             0.265    0.268   0.531   0.257
## 95%CI-Low        0.064    0.052   0.000   0.057
## 95%CI-High       0.471    0.472   0.962   0.635
## Accept           0.440    0.439   0.465   0.408
## BMSE             0.002    0.003   0.020   0.023
## 0.01 x mean      0.003    0.003   0.005   0.003
## ESS           8199.358 6484.087 585.015 584.500
## ESS/sec         14.852   11.745   1.060   1.059

summaryMat800[[1]] # summary table when n=800

##                  beta1    beta2     phi  sigma2
## Mean             0.257    0.315   0.014   0.270
## 95%CI-Low        0.117    0.188   0.000   0.089
## 95%CI-High       0.386    0.445   0.025   0.663
## Accept           0.438    0.438   0.442   0.417
## BMSE             0.002    0.001   0.001   0.041
```

```
## 0.01 x mean     0.003    0.003   0.000    0.003
## ESS           6249.841 7625.949 750.319 567.803
## ESS/sec          1.926    2.351   0.231    0.175
```

```
summaryMat1600[[1]] # summary table when n=1600
```

```
##                  beta1    beta2     phi   sigma2
## Mean             0.247    0.253   0.014    0.536
## 95%CI-Low        0.089    0.153   0.000    0.167
## 95%CI-High       0.366    0.352   0.023    0.840
## Accept           0.438    0.438   0.447    0.415
## BMSE             0.003    0.002   0.001    0.131
## 0.01 x mean      0.002    0.003   0.000    0.005
## ESS           3132.240 4938.700 590.097 560.434
## ESS/sec          0.160    0.253   0.030    0.029
```

```
plot(x=c(200,400,800,1600),y=c(ptFinal200[3],ptFinal400[3],ptFinal800[3],ptFi
nal1600[3])
     ,xlab="sample size",ylab="Computing time",type="b",pch=1)
```



computing time
looks increase more quickly.
Since common MCMC for SGLMM requires $O(n^3)$, it takes a lot of time at n=1600 rather than n=800.

2-(a)

I used same datasets in problem 1(using same seed number).

```
# install.packages("fastLaplace")
library(fastLaplace)
length(sim.data$obsModBin)
data = data.frame(cbind(sim.data$obsModBin,sim.data$XMat))
colnames(data) = c("Y","X1","X2")
coords = sim.data$gridLocation

# for initial values
mod.glm <- glm(Y~-1+X1+X2,family="binomial",data=data)
mod.glm.esp <- predict(mod.glm,data, type="response")
mod.glm.s2 <- var(data$Y - mod.glm.esp)
mod.glm.phi <- 0.1*max(dist(coords))
startinit <- c(mod.glm$coef,log(mod.glm.s2),log(mod.glm.phi))
names(startinit) <- c("X1","X2","logsigma2","logphi")

pt<-proc.time()
result.bin200 <- fsglmm(Y~-1+X1+X2, kappa=2.5, inits = startinit, data = data,
coords = coords,  family = "binomial", ntrial = 1, offset = NA,method.optim =
 "CG", method.integrate = "NR", control = list(maxit=1000,ndeps=rep(1e-2,4),r
eltol=0.01),rank = 50)
FL200<-proc.time()-pt


result.bin$summary

X.pred <- sim.data$XmatCV
coords.pred <- sim.data$CVgridLocation

bin.p = pred.sglmm(result.bin,data=X.pred,coords=coords.pred)
fl.pred.y = sapply(bin.p,rbinom,n=1,size=1)
actual.y = sim.data$obsCVBin

ct = table(fl.pred.y,actual.y)
fl200_acc = sum(diag(ct))/sum(ct)
fl200_acc

save(FL200,result.bin200,fl200_acc,file="FL200.RData")
```

I repeated the same code with different sample size.

```
load("FL200.RData")
load("FL400.RData")
load("FL800.RData")
load("FL1600.RData")

FL200[3] # Computing time when n =200

## elapsed
##     4.8
```

```
FL400[3] # Computing time when n =400
```

```
## elapsed
##   22.07
```

```
FL800[3] # Computing time when n =800
```

```
## elapsed
##   54.88
```

```
FL1600[3] # Computing time when n =1600
```

```
## elapsed
##  228.05
```

```
result.bin200$summary # Estimates, standard errors for beta,rho,sigma with n=
200
```

```
## Maximum likelihood estimation
##
## Call:
## bbmle::mle2(minuslogl = nlikSGLMM, start = inits, method = method.optim,
##     data = list(Y = Y, X = X, mat.dist = mat.dist, ntrial = ntrial,
##         family = family, method = method.integrate, kappa = kappa,
##         offset = offset, U1 = U1, rank = rank), vecpar = TRUE,
##     skip.hessian = TRUE, control = control)
##
## Coefficients:
##            Estimate Std. Error z value Pr(z)
## X1          1.04385    0.33621      NA    NA
## X2          0.96609    0.34740      NA    NA
## logsigma2   0.63473    0.61381      NA    NA
## logphi     -1.22672    0.50351      NA    NA
##
## -2 log L: 227.3415
```

```
result.bin400$summary # Estimates, standard errors for beta,rho,sigma with n=
400
```

```
## Maximum likelihood estimation
##
## Call:
## bbmle::mle2(minuslogl = nlikSGLMM, start = inits, method = method.optim,
##     data = list(Y = Y, X = X, mat.dist = mat.dist, ntrial = ntrial,
##         family = family, method = method.integrate, kappa = kappa,
##         offset = offset, U1 = U1, rank = rank), vecpar = TRUE,
##     skip.hessian = TRUE, control = control)
##
## Coefficients:
##            Estimate Std. Error z value Pr(z)
## X1          0.91140    0.23040      NA    NA
```

```
## X2         1.38831    0.24193      NA    NA
## logsigma2  1.18184    0.57276      NA    NA
## logphi    -1.38482    0.27843      NA    NA
##
## -2 log L: 436.5978
```

```
result.bin800$summary # Estimates, standard errors for beta,rho,sigma with n=
800
```

```
## Maximum likelihood estimation
##
## Call:
## bbmle::mle2(minuslogl = nlikSGLMM, start = inits, method = method.optim,
##     data = list(Y = Y, X = X, mat.dist = mat.dist, ntrial = ntrial,
##         family = family, method = method.integrate, kappa = kappa,
##         offset = offset, U1 = U1, rank = rank), vecpar = TRUE,
##     skip.hessian = TRUE, control = control)
##
## Coefficients:
##           Estimate Std. Error z value Pr(z)
## X1         1.13504    0.14499      NA    NA
## X2         1.19619    0.14566      NA    NA
## logsigma2 -1.44326    0.62223      NA    NA
## logphi    -2.25148    0.38854      NA    NA
##
## -2 log L: 952.9866
```

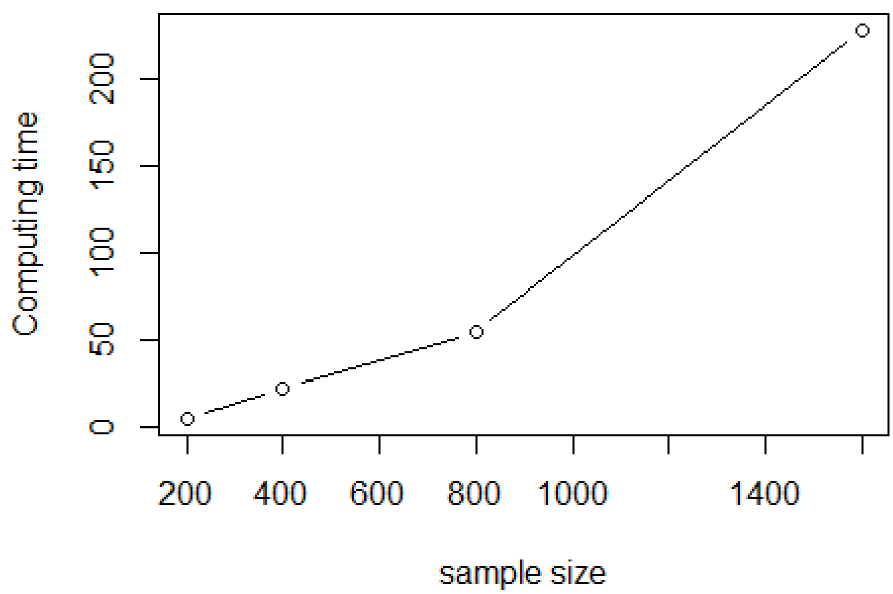
```
result.bin1600$summary # Estimates, standard errors for beta,rho,sigma with n
=1600
```

```
## Maximum likelihood estimation
##
## Call:
## bbmle::mle2(minuslogl = nlikSGLMM, start = inits, method = method.optim,
##     data = list(Y = Y, X = X, mat.dist = mat.dist, ntrial = ntrial,
##         family = family, method = method.integrate, kappa = kappa,
##         offset = offset, U1 = U1, rank = rank), vecpar = TRUE,
##     skip.hessian = TRUE, control = control)
##
## Coefficients:
##            Estimate Std. Error z value Pr(z)
## X1         1.153023   0.108278      NA    NA
## X2         1.194692   0.109868      NA    NA
## logsigma2  0.074416   0.393735      NA    NA
## logphi    -1.638462   0.211126      NA    NA
##
## -2 log L: 1797.541
```

```
fl200_acc # Prediction accuracy with n =200
```

```
## [1] 0.675
fl400_acc # Prediction accuracy with n =400
## [1] 0.65
fl800_acc # Prediction accuracy with n =800
## [1] 0.5875
fl1600_acc # Prediction accuracy with n =1600
## [1] 0.61875
```

2-(b)



The overall computing time is greatly reduced.
Compared to the computing time in problem 1, It seems to have more linear relationship.

2-(c)

True beta =c(1,1)
True rho(phi) = 0.2
True sigma2 = 1

Overall results of FastLaplace cases seems to be more accurate. Beta estimates are more close to the populations.

rho and sigma, which is the variance estimates, are little bit less accurate on both methods. FastLaplace method is much faster than nimble-MCMC algorithm especially sample size is large.

Prediction results of FastLaplace method are also more accurate than nimble-MCMC algorithm.

Appendix

Appendix 1-(a)

```
ploteqc <- function(spobj, z, breaks, ...){
  pal <- tim.colors(length(breaks)-1)
  fb <- classIntervals(z, n = length(pal),
                       style = "fixed", fixedBreaks = breaks)
  col <- findColours(fb, pal)
  plot(spobj, col = col, ...)
  image.plot(legend.only = TRUE, zlim = range(breaks), col = pal)
}

## Using Ming-Hui Chen's paper in Journal of Computational and Graphical Stat
s.
hpd <- function(samp,p=0.05){
  ## to find an approximate (1-p)*100% HPD interval from a
  ## given posterior sample vector samp

  r <- length(samp)
  samp <- sort(samp)
  rang <- matrix(0,nrow=trunc(p*r),ncol=3)
  dimnames(rang) <- list(NULL,c("low","high","range"))
  for (i in 1:trunc(p*r)) {
    rang[i,1] <- samp[i]
    rang[i,2] <- samp[i+(1-p)*r]
    rang[i,3] <- rang[i,2]-rang[i,1]
  }
  hpd <- rang[order(rang[,3])[1],1:2]
  return(hpd)
}

# Exponential Covariance Function
expCov<-function(distMat,phi){
  exp(-distMat/phi)
}

sqeCov<-function(distMat,phi){
  exp(-0.5*(distMat/phi)^2)
}
```

```
matCov<-function(distMat,phi){
  (1+(sqrt(5)*(distMat/phi))+((5*distMat^2)/(3*(phi^2))))*exp(-(sqrt(5)*(dist
Mat/phi)))
}


# Matern Cov Function + Acceptance Rate function
Matern <- function(d, param = c(scale = 1, range = 1, smoothness = 2)) {
  scale <- param[1]
  range <- param[2]
  smoothness <- param[3]
  if (any(d < 0))
    stop("distance argument must be nonnegative")
  d <- d / range
  d[d == 0] <- 1e-10
  rootcon<-sqrt(2*smoothness)
  con <- (2^(smoothness - 1)) * gamma(smoothness)
  con <- 1 / con
  return(scale * con * ((rootcon*d)^smoothness) * besselK(rootcon*d, smoothne
ss))
}
accRateFunc<-function(x){
  accRate<-(length(unique(x))-1)/(length(x)-1)
  return(accRate)
}


# Summary
summaryFunction<-function(mcmcDat,bmseThresh=0.01,time){

  # Parameters
  summaryMat<-rbind(apply(mcmcDat,2,mean),
                    apply(mcmcDat,2,hpd),
                    apply(mcmcDat,2,accRateFunc),
                    bmmat(mcmcDat)[,2],
                    abs(apply(mcmcDat,2,mean))*bmseThresh,
                    apply(mcmcDat,2,ess),
                    apply(mcmcDat,2,ess)/time)

  rownames(summaryMat)<-c("Mean","95%CI-Low","95%CI-High",
                          "Accept","BMSE",paste(bmseThresh,"x mean"),
                          "ESS","ESS/sec")
  return(summaryMat)
}

# NIMBLE FUNCTIONS

expcov <- nimbleFunction(
```

```
  run = function(dists = double(2), phi = double(0)) {
    returnType(double(2))
    n <- dim(dists)[1]
    result <- matrix(nrow = n, ncol = n, init = FALSE)
    for(i in 1:n){
      for(j in 1:n){
        result[i, j] <- exp(-dists[i,j]/phi)
      }
    }

    return(result)
  })


matcov <- nimbleFunction(
  run = function(dists = double(2), phi = double(0)) {
    returnType(double(2))
    n <- dim(dists)[1]
    result <- matrix(nrow = n, ncol = n, init = FALSE)
    for(i in 1:n){
      for(j in 1:n){
        result[i, j] <- (1+(sqrt(5)*(dists[i,j]/phi))+((5*dists[i,j]^2)/(3*(p
hi^2))))*exp(-(sqrt(5)*(dists[i,j]/phi)))
      }
    }

    return(result)
  })
```

Appendix 1-(c)

```
#----------------------------------------------------------------------------
--#
#----------------------------------------------------------------------------
--#
# for n = 400

niter=50000
n =400
sim.data = Simulate_binary_data(n=n,cv=0.2)
obsModBinom = sim.data$obsModBin
XMat = sim.data$XMat
distMatMod = sim.data$distMatMod

consts   <- list(n=n,X=XMat,dists=distMatMod,mn=rep(0,n))
data     <- list(Z=obsModBinom)
inits    <- list(beta1=rnorm(1),beta2=rnorm(1),phi=0.5,sigma2=2,
                 W=rnorm(n))
```

```
# Run MCMC


nimble_model = nimbleModel(model_string, data = data, inits = inits,constants
=consts)
MCMC_conf = configureMCMC(nimble_model,monitors = c("beta1", "beta2","phi","s
igma2"),
                          control = list(adaptFactorExponent = 0.75))
RMCMC = buildMCMC(MCMC_conf,samplesAsCodaMCMC=TRUE,WAIC=FALSE,summary=FALSE,
thin=20,
                  niter = niter, nburnin = 0, nchains = 1)
Cmodel = compileNimble(nimble_model)
Cmcmc = compileNimble(RMCMC,project = nimble_model)

pt<-proc.time()
Cmcmc$run(niter = niter)
samples400 = as.matrix(Cmcmc$mvSamples)

# samples400  <- nimbleMCMC(model_string, data = data, inits = inits,
#                           constants=consts,
#                           monitors = c("beta1", "beta2","phi","sigma2"),
#                           samplesAsCodaMCMC=TRUE,WAIC=FALSE,summary=FALSE,
thin=20,
#                           niter = niter, nburnin = 0, nchains = 1)
ptFinal400<-proc.time()-pt

# computing time
ptFinal400

# trace plot
pdf(file = "BinomResults400.pdf",width=11,height=8.5)
par(mfrow=c(4,2),mar=c(2,2,2,2))
sampInd<-floor(seq(1,nrow(samples400),length.out = 1000))
beta=c(1,1) ; phi=0.2 ; sigma2=1

for(i in 1:4){
  plot.ts(samples400[sampInd,i]); abline(h=c(beta,phi,sigma2)[i],col="red",lw
d=2)
  plot(density(samples400[sampInd,i])); abline(v=c(beta,phi,sigma2)[i],col="r
ed",lwd=2)
}

dev.off()

summaryMat400<-list()
summaryMat400[[1]]<-round(summaryFunction(samples400[,c("beta1", "beta2","phi
","sigma2")],
                                          time=ptFinal400[3]),3)
```

```
# posterior mean, hpd, acc rate
summaryMat400[[1]]
save(samples400,file="BinomMCMCsamples400.RData")
save(summaryMat400,samples400,ptFinal400,file="BinomMCMCResults400.RData")

# prediction acc
burnin <- 100
s2.final <- samples400[,c("sigma2")][-(1:burnin)]
beta.final <- cbind(samples400[,c("beta1")][-(1:burnin)],samples400[,c("beta2
")][-(1:burnin)])
rho.final <- samples400[,c("phi")][-(1:burnin)]


obs.grid = sim.data$gridLocation
pred.grid = sim.data$CVgridLocation
X = sim.data$XMat
Xpred = sim.data$XmatCV

full.grid = rbind(obs.grid,pred.grid)
full.X = rbind(X,Xpred)
# fixed part
full.XB = full.X%*%t(beta.final)
d <- rdist.earth(coordinates(obs.grid))
dcross <- rdist.earth(coordinates(obs.grid), coordinates(pred.grid)) # 200*40
dpred <- rdist.earth(coordinates(pred.grid)) # 40*40

eta.mat <- matrix(NA, nrow = nrow(pred.grid), ncol = niter-burnin) # 40*

# random part
distMatFull = rdist.earth(coordinates(full.grid))

for(j in 1:ncol(eta.mat)){
  if(j%%200 == 0){print(j)}

  # Construct the covariance matrices
  Gamma <- exp(-d/rho.final[j])
  Ginv <- solve(Gamma)
  g <- exp(-dcross/rho.final[j])
  Gpred <- exp(-dpred/rho.final[j])

  m <- Xpred %*% beta.final[j,] + t(g) %*% Ginv %*%
    (y - X %*% beta.final[j,])
  V <- s2.final[j] * (Gpred - t(g)%*%Ginv%*%g)
  eta.mat[,j] <- rmvnorm(1, m, V, method = "svd")
}
save(eta.mat,file="Binom_eta400.RData")
dim(eta.mat)
```

```r
acc_lst = NULL
actual.y = sim.data$obsCVBin

for(i in 1:ncol(eta.mat)){
  bin.p = exp(eta.mat[,i])/(1+exp(eta.mat[,i]))
  pred.y = sapply(bin.p,rbinom,n=1,size=1)
  ct = table(pred.y,actual.y)
  acc_lst[i] = sum(diag(ct))/sum(ct)

}
save(acc_lst,file="acc_lst400.RData")

length(acc_lst)
mean(acc_lst)

#-----------------------------------------------------------------------------
--#
#-----------------------------------------------------------------------------
--#
# for n = 800

niter=50000
n =800
sim.data = Simulate_binary_data(n=n,cv=0.2)
obsModBinom = sim.data$obsModBin
XMat = sim.data$XMat
distMatMod = sim.data$distMatMod

consts   <- list(n=n,X=XMat,dists=distMatMod,mn=rep(0,n))
data     <- list(Z=obsModBinom)
inits    <- list(beta1=rnorm(1),beta2=rnorm(1),phi=0.5,sigma2=2,
                 W=rnorm(n))

# Run MCMC


nimble_model = nimbleModel(model_string, data = data, inits = inits,constants
=consts)
MCMC_conf = configureMCMC(nimble_model,monitors = c("beta1", "beta2","phi","s
igma2"),
                          control = list(adaptFactorExponent = 0.75))
RMCMC = buildMCMC(MCMC_conf,samplesAsCodaMCMC=TRUE,WAIC=FALSE,summary=FALSE,
thin=20,
                  niter = niter, nburnin = 0, nchains = 1)
Cmodel = compileNimble(nimble_model)
Cmcmc = compileNimble(RMCMC,project = nimble_model)

pt<-proc.time()
```

```
Cmcmc$run(niter = niter)
samples800 = as.matrix(Cmcmc$mvSamples)


ptFinal800<-proc.time()-pt

# computing time
ptFinal800

# trace plot
pdf(file = "BinomResults800.pdf",width=11,height=8.5)
par(mfrow=c(4,2),mar=c(2,2,2,2))
sampInd<-floor(seq(1,nrow(samples800),length.out = 1000))
beta=c(1,1) ; phi=0.2 ; sigma2=1

for(i in 1:4){
  plot.ts(samples800[sampInd,i]); abline(h=c(beta,phi,sigma2)[i],col="red",lw
d=2)
  plot(density(samples800[sampInd,i])); abline(v=c(beta,phi,sigma2)[i],col="r
ed",lwd=2)
}

dev.off()

summaryMat800<-list()
summaryMat800[[1]]<-round(summaryFunction(samples800[,c("beta1", "beta2","phi
","sigma2")],
                                          time=ptFinal800[3]),3)
# posterior mean, hpd, acc rate
summaryMat800[[1]]
save(samples800,file="BinomMCMCsamples800.RData")
save(summaryMat800,samples800,ptFinal800,file="BinomMCMCResults800.RData")

# prediction acc
burnin <- 100
s2.final <- samples800[,c("sigma2")][-(1:burnin)]
beta.final <- cbind(samples800[,c("beta1")][-(1:burnin)],samples800[,c("beta2
")][-(1:burnin)])
rho.final <- samples800[,c("phi")][-(1:burnin)]


obs.grid = sim.data$gridLocation
pred.grid = sim.data$CVgridLocation
X = sim.data$XMat
Xpred = sim.data$XmatCV

full.grid = rbind(obs.grid,pred.grid)
full.X = rbind(X,Xpred)
```

```
# fixed part
full.XB = full.X%*%t(beta.final)
d <- rdist.earth(coordinates(obs.grid))
dcross <- rdist.earth(coordinates(obs.grid), coordinates(pred.grid)) # 200*40
dpred <- rdist.earth(coordinates(pred.grid)) # 40*40

eta.mat <- matrix(NA, nrow = nrow(pred.grid), ncol = niter-burnin) # 40*

# random part
distMatFull = rdist.earth(coordinates(full.grid))

for(j in 1:ncol(eta.mat)){
  if(j%%200 == 0){print(j)}

  # Construct the covariance matrices
  Gamma <- exp(-d/rho.final[j])
  Ginv <- solve(Gamma)
  g <- exp(-dcross/rho.final[j])
  Gpred <- exp(-dpred/rho.final[j])

  m <- Xpred %*% beta.final[j,] + t(g) %*% Ginv %*%
    (y - X %*% beta.final[j,])
  V <- s2.final[j] * (Gpred - t(g)%*%Ginv%*%g)
  eta.mat[,j] <- rmvnorm(1, m, V, method = "svd")
}
save(eta.mat,file="Binom_eta800.RData")
load(file="Binom_eta800.RData")
dim(eta.mat)

acc_lst = NULL
actual.y = sim.data$obsCVBin

for(i in 1:ncol(eta.mat)){
  bin.p = exp(eta.mat[,i])/(1+exp(eta.mat[,i]))
  pred.y = sapply(bin.p,rbinom,n=1,size=1)
  ct = table(pred.y,actual.y)
  acc_lst[i] = sum(diag(ct))/sum(ct)

}
save(acc_lst,file="acc_lst800.RData")


length(acc_lst)
mean(acc_lst)

#-----------------------------------------------------------------------------
--#
#-----------------------------------------------------------------------------
```

```
--#
# for n = 1600

niter=50000
n =1600
sim.data = Simulate_binary_data(n=n,cv=0.2)
obsModBinom = sim.data$obsModBin
XMat = sim.data$XMat
distMatMod = sim.data$distMatMod

consts   <- list(n=n,X=XMat,dists=distMatMod,mn=rep(0,n))
data     <- list(Z=obsModBinom)
inits    <- list(beta1=rnorm(1),beta2=rnorm(1),phi=0.5,sigma2=2,
                 W=rnorm(n))

# Run MCMC


nimble_model = nimbleModel(model_string, data = data, inits = inits,constants
=consts)
MCMC_conf = configureMCMC(nimble_model,monitors = c("beta1", "beta2","phi","s
igma2"),
                          control = list(adaptFactorExponent = 0.3))
RMCMC = buildMCMC(MCMC_conf,samplesAsCodaMCMC=TRUE,WAIC=FALSE,summary=FALSE,
thin=20,
                  niter = niter, nburnin = 0, nchains = 1)
Cmodel = compileNimble(nimble_model)
Cmcmc = compileNimble(RMCMC,project = nimble_model)

pt<-proc.time()
Cmcmc$run(niter = niter)
samples1600 = as.matrix(Cmcmc$mvSamples)

# samples1600  <- nimbleMCMC(model_string, data = data, inits = inits,
#                           constants=consts,
#                           monitors = c("beta1", "beta2","phi","sigma2"),
#                           samplesAsCodaMCMC=TRUE,WAIC=FALSE,summary=FALSE,
thin=20,
#                           niter = niter, nburnin = 0, nchains = 1)
ptFinal1600<-proc.time()-pt

# computing time
ptFinal1600

# trace plot
pdf(file = "BinomResults1600.pdf",width=11,height=8.5)
par(mfrow=c(4,2),mar=c(2,2,2,2))
sampInd<-floor(seq(1,nrow(samples1600),length.out = 1000))
```

```
beta=c(1,1) ; phi=0.2 ; sigma2=1

for(i in 1:4){
  plot.ts(samples1600[sampInd,i]); abline(h=c(beta,phi,sigma2)[i],col="red",l
wd=2)
  plot(density(samples1600[sampInd,i])); abline(v=c(beta,phi,sigma2)[i],col="
red",lwd=2)
}

dev.off()

summaryMat1600<-list()
summaryMat1600[[1]]<-round(summaryFunction(samples1600[,c("beta1", "beta2","p
hi","sigma2")],
                                          time=ptFinal1600[3]),3)
# posterior mean, hpd, acc rate
summaryMat1600[[1]]
save(samples1600,file="BinomMCMCsamples1600.RData")
save(summaryMat1600,samples1600,ptFinal1600,file="BinomMCMCResults1600.RData")
load(file="BinomMCMCsamples1600.RData")
load(file="BinomMCMCResults1600.RData")

# prediction acc
burnin <- 100
s2.final <- samples1600[,c("sigma2")][-(1:burnin)]
beta.final <- cbind(samples1600[,c("beta1")][-(1:burnin)],samples1600[,c("bet
a2")][-(1:burnin)])
rho.final <- samples1600[,c("phi")][-(1:burnin)]


obs.grid = sim.data$gridLocation
pred.grid = sim.data$CVgridLocation
X = sim.data$XMat
Xpred = sim.data$XmatCV

full.grid = rbind(obs.grid,pred.grid)
full.X = rbind(X,Xpred)
# fixed part
full.XB = full.X%*%t(beta.final)
d <- rdist.earth(coordinates(obs.grid))
dcross <- rdist.earth(coordinates(obs.grid), coordinates(pred.grid)) # 200*40
dpred <- rdist.earth(coordinates(pred.grid)) # 40*40

eta.mat <- matrix(NA, nrow = nrow(pred.grid), ncol = niter-burnin) # 40*

# random part
distMatFull = rdist.earth(coordinates(full.grid))
```

```r
for(j in 1:ncol(eta.mat)){
  if(j%%200 == 0){print(j)}

  # Construct the covariance matrices
  Gamma <- exp(-d/rho.final[j])
  Ginv <- solve(Gamma)
  g <- exp(-dcross/rho.final[j])
  Gpred <- exp(-dpred/rho.final[j])

  m <- Xpred %*% beta.final[j,] + t(g) %*% Ginv %*%
    (y - X %*% beta.final[j,])
  V <- s2.final[j] * (Gpred - t(g)%*%Ginv%*%g)
  eta.mat[,j] <- rmvnorm(1, m, V, method = "svd")
}
save(eta.mat,file="Binom_eta1600.RData")
load(file="Binom_eta1600.RData")
dim(eta.mat)

acc_lst = NULL
actual.y = sim.data$obsCVBin

for(i in 1:ncol(eta.mat)){
  bin.p = exp(eta.mat[,i])/(1+exp(eta.mat[,i]))
  pred.y = sapply(bin.p,rbinom,n=1,size=1)
  ct = table(pred.y,actual.y)
  acc_lst[i] = sum(diag(ct))/sum(ct)

}
save(acc_lst,file="acc_lst1600.RData")




length(acc_lst)
mean(acc_lst)
```