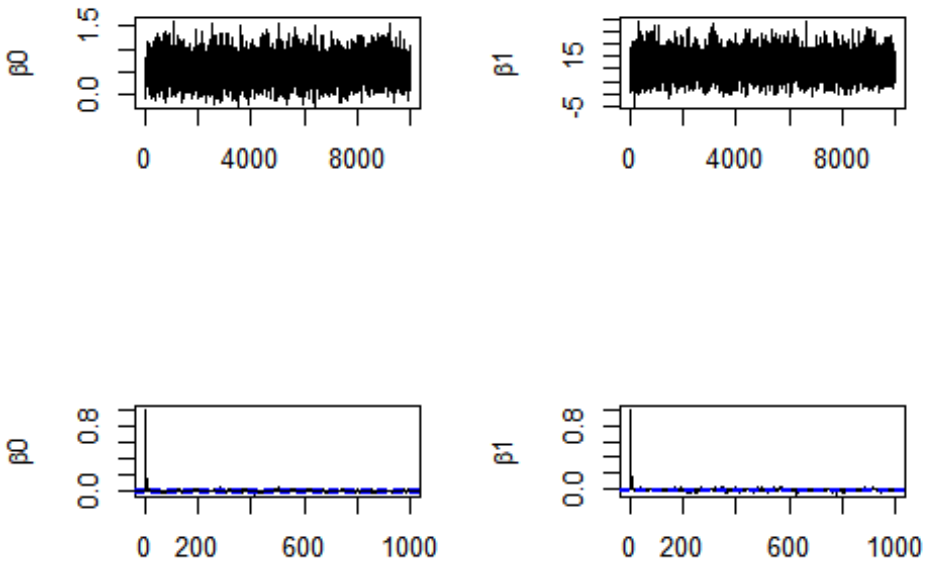
hw3

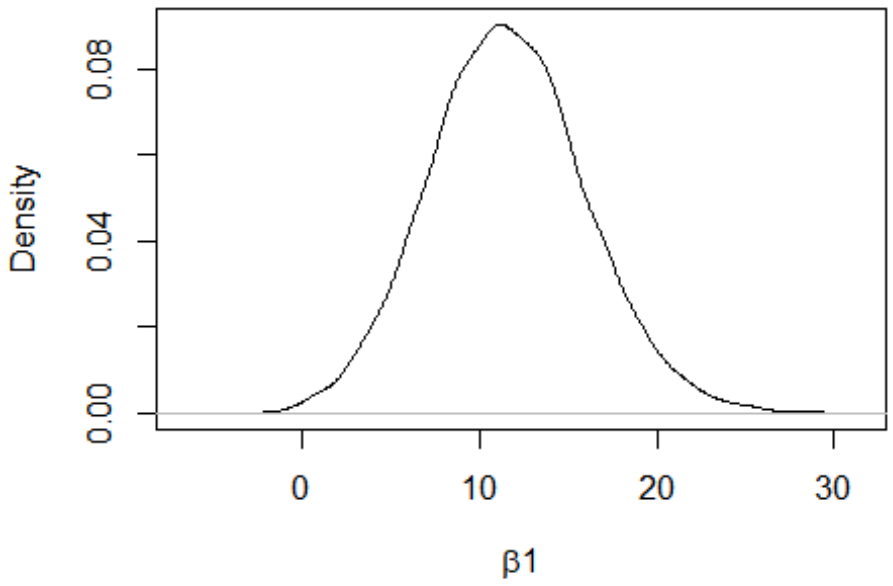
192STG11 우나영

```
pacman::p_load(coda, rjags, runjags)
rm(list=ls())
setwd('C:/Users/dnskd/Desktop/20Spring/Bayesian/week6')

#Multivariate normal variable generator
rmvnorm = function(mu, Sigma){
  R = t(chol(Sigma))
  R %*% (rnorm(length(mu), 0, 1)) + mu
}

# truncated normal random generator
trunnor = function(mu, sigma, l, u){
  unif = runif(1)
  trunnor = qnorm(unif*pnorm((u-mu)/sigma)+(1-unif)*pnorm((l-mu)/sigma)) * si
gma + mu
}

# data
nn = c(4, 4, 5, 3, 5, 4, 4, 4)
yy = c(2, 1, 2, 3, 3, 3, 4, 4)
xx = c(1.69, 1.72, 1.75, 1.78, 1.81, 1.83, 1.86, 1.88)
K = length(yy)
x = rep(xx, nn)
SF = as.vector(rbind(yy, nn-yy))
y = rep(rep(c(1,0), times = K), SF)

x.mean = mean(x)
x = x - x.mean
n = length(x)
X = cbind(rep(1, n), x)
p = 2

beta0 = rep(0, p)
Sigma0.inv = diag(0, p)

# initialize
beta = beta0
nsim = 10000; nwarm = 1000
beta.save = matrix(0, nsim, p)
Sigma.beta = solve(t(X) %*% X + Sigma0.inv)
ystar = rep(0, n)
```

```r
# MCMC
for(isim in 1:(nsim + nwarm)){
  # generate ystar
  for(i in 1:n){
    if(y[i]==1)
      ystar[i] = trunnor(t(X[i,]) %*% beta, 1, 0, Inf)
    else
      ystar[i] = trunnor(t(X[i,]) %*% beta, 1, -Inf, 0)
  }
  # generate beta
  Mu.beta = Sigma.beta %*% (t(X) %*% ystar + Sigma0.inv %*% beta0)
  beta = rmvnorm(Mu.beta, Sigma.beta)
  if(isim>nwarm)beta.save[isim-nwarm,] = beta
}

par(mfrow=c(2,2))
plot(beta.save[,1], type = 'l', xlab = "", ylab = expression(paste(beta,"0
")), main = "")
plot(beta.save[,2], type = 'l', xlab = "", ylab = expression(paste(beta,"1
")), main = "")
acf(beta.save[,1], xlab = "", ylab = expression(paste(beta, "0")), lag.max =
1000, main = "")
acf(beta.save[,2], xlab = "", ylab = expression(paste(beta, "1")), lag.max =
1000, main = "")
```

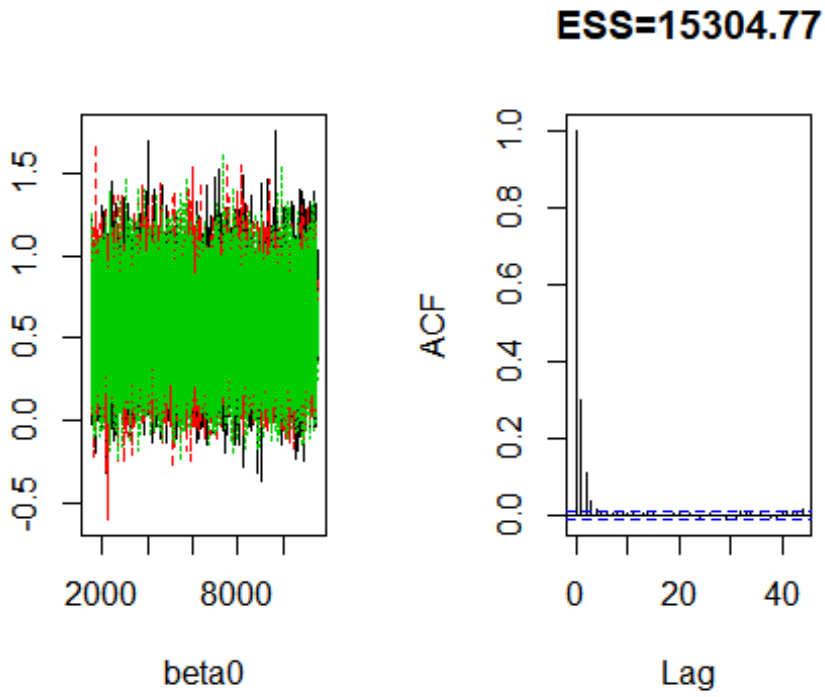


```r
par(mfrow=c(1,1))
plot(density(beta.save[,2]), type = "l", xlab = expression(paste(beta, "1")),
 main = "")
```

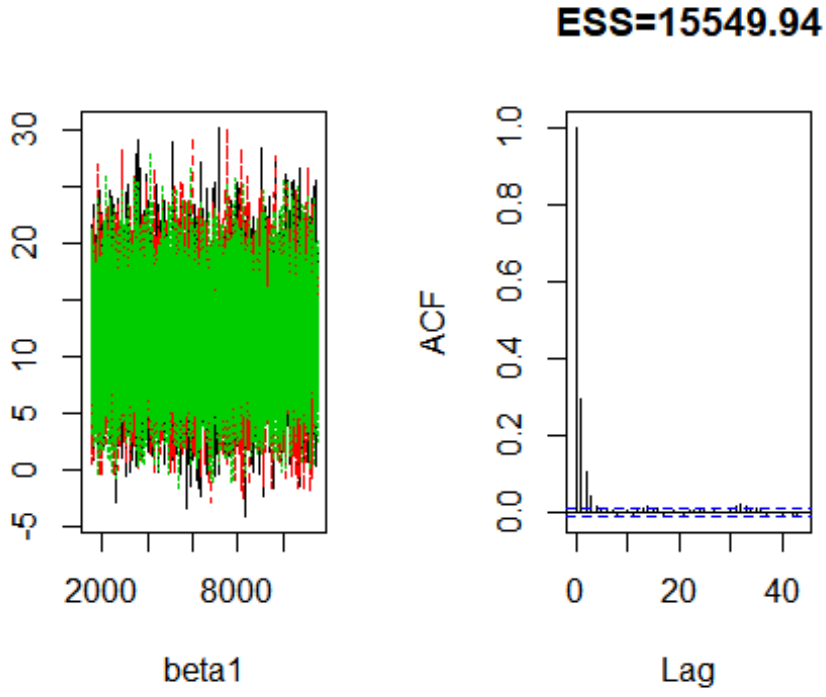



```r
#### model 설정 ####
modelString = "
model
{
for(i in 1:K){
    y[i] ~ dbin(pi[i], n[i])
    probit(pi[i]) <- beta0 + beta1*(x[i]-mean(x[]))
}
beta0 ~ dnorm(mu0, invsig0)
beta1 ~ dnorm(mu1, invsig1)
}
"

writeLines(modelString, "model_binary_probit.txt")

n = nn; y = yy; x = xx; K = length(yy)
dataList = list(n = n, y = y, x = x, K=K, mu0 = 0, invsig0=0.00001, mu1 = 0,
invsig1 = 0.00001)
initsList = list(beta0 = 0, beta1 = 0)
nChains = 3; nThin = 1; nAdapt = 500; nBurn = 1000; nIter = 10000
```

```r
jagsModel = jags.model(file = "model_binary_probit.txt", data = dataList, ini
ts = initsList, n.chains = nChains, n.adapt = nAdapt)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 8
##    Unobserved stochastic nodes: 2
##    Total graph size: 65
##
## Initializing model

update(jagsModel, n.iter = nBurn)
codaSamples = coda.samples(jagsModel, variable.names = c("beta0", "beta1"),
n.iter = nIter)
1 - rejectionRate(codaSamples)

## beta0 beta1
##     1     1

# multiple chains
convDiag.plot = function(codaSamples, var){
  ESS = effectiveSize(codaSamples[, var])
  par(mfrow = c(1,2))
  traceplot(codaSamples[, var], xlab = var)
  acf(as.matrix(codaSamples[,var]), main = paste0("ESS=", round(ESS,2)))
}

convDiag = function(codaSamples){
  ESS = effectiveSize(codaSamples); cat("ESS = ", ESS, "\n")
  acceptRate = 1-rejectionRate(codaSamples); cat("Accept rate=", acceptRate,
"\n")
  gelman = gelman.diag(codaSamples); print(gelman)
  gelman.1 = as.matrix(gelman$psrf)
  if(max(gelman.1)>1.1) cat("Warning : Gelman Shrink Factor > 1.1", "\n")
  gelman.2 = gelman$mpsrf
  if(gelman.2>1.1) cat("Warning : Gelman Multivariate Shrink Factor > 1.1", "
\n")
}


convDiag.plot(codaSamples, "beta0")
```




```
convDiag.plot(codaSamples, "beta1")
```

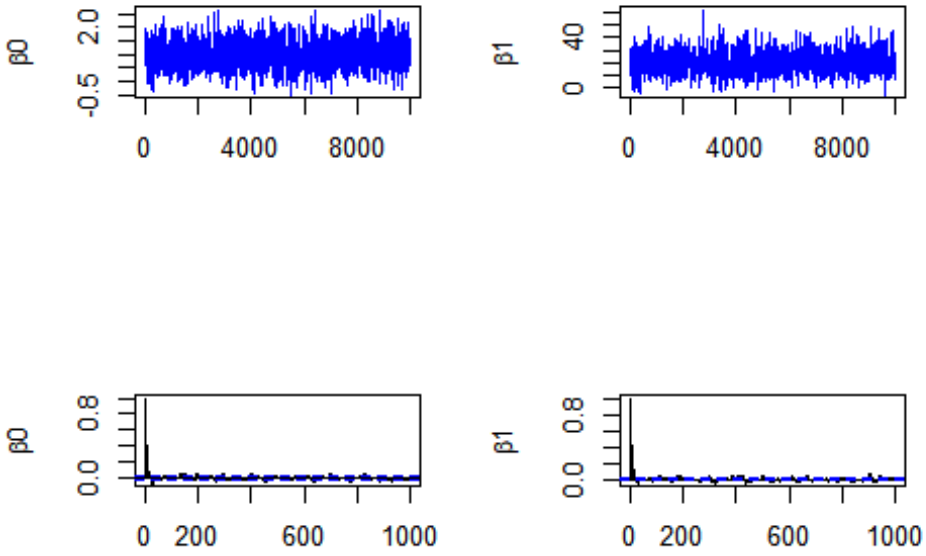


```r
convDiag(codaSamples)
```

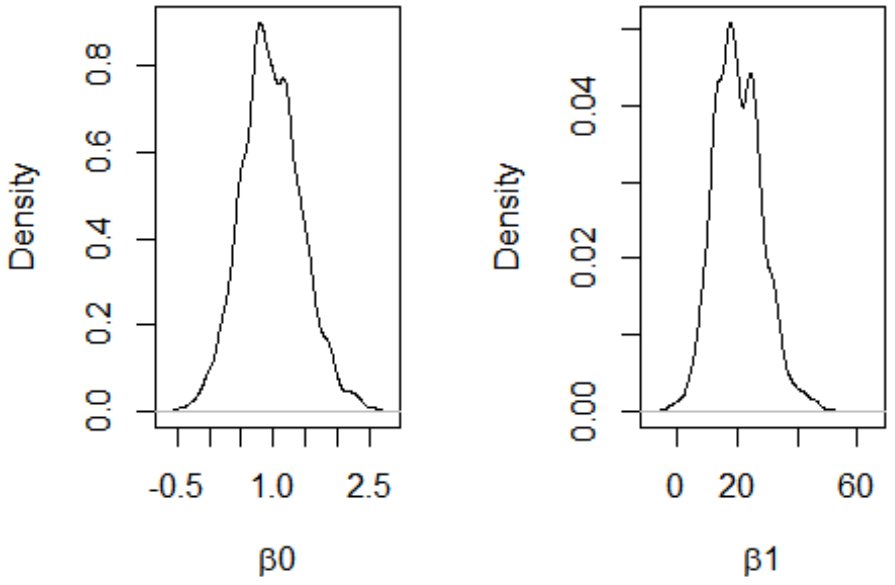
```
## ESS =  15304.77 15549.94
## Accept rate= 1 1
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## beta0          1          1
## beta1          1          1
##
## Multivariate psrf
##
## 1
```

```r
#### Logistic Model: M-H ####
x = xx; y = yy; n = nn; K = length(y)
x = x - mean(x)
X = cbind(rep(1, K), x)
p = ncol(X)

# log posterior function
logpost <- function(beta, K, n, x, y, beta0, Sig0.inv){
  Xbeta = X %*% beta
  p = 1/(1+exp(-Xbeta))
  logpost = sum(y*log(p)+(n-y)*log(1-p))-0.5*t(beta-beta0)%*%Sig0.inv%*%(beta
-beta0)
}

#---- Random Walk Metropolis
glm.out <- glm(cbind(y, n-y) ~ x, family = binomial(link = 'logit'))
beta.MLE <- as.vector(glm.out$coefficient)
beta = beta.MLE
beta0 = beta.MLE
Sig0.inv = diag(0, p) # non-informative prior
XtX.inv = solve(t(X)%*%X)
nsim <- 10000; nthin = 1; nwarm <- 1000
beta.save <- matrix(0, nsim, p)


# random walk Metropolis
deltasq = 9; nAccept = 0
for(isim in 1:(nsim*nthin + nwarm)){
  beta.star <- as.vector(rmvnorm(beta, deltasq*XtX.inv))
  log.alpha <- logpost(beta.star, K, n, X, y, beta0, Sig0.inv) - logpost(bet
a, K, n, X, y, beta0, Sig0.inv)
  u <- runif(1)
  if(log(u)<log.alpha){beta <- beta.star; nAccept = nAccept + 1}
  if(isim > nwarm & isim %% nthin == 0) beta.save[(isim-nwarm)/nthin, ] <- be
ta
```

```r
}

par(mfrow = c(2,2))
plot(beta.save[,1], type = "l", xlab = "", ylab = quote(paste(beta, 0)), main
 = "", col = "blue")
plot(beta.save[,2], type = "l", xlab = "", ylab = quote(paste(beta, 1)), main
 = "", col = "blue")
acf(beta.save[,1], xlab = "", ylab = quote(paste(beta, 0)), lag.max = 1000, m
ain = "")
acf(beta.save[,2], xlab = "", ylab = quote(paste(beta, 1)), lag.max = 1000, m
ain = "")
```

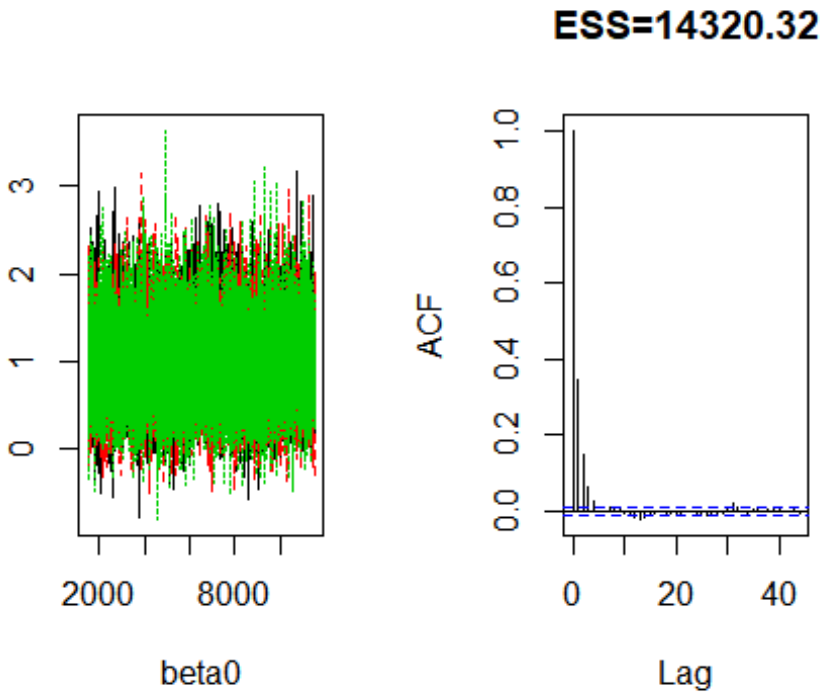



```r
par(mfrow = c(1,2))
plot(density(beta.save[,1]), xlab = expression(paste(beta,"0")), main = "")
plot(density(beta.save[,2]), xlab = expression(paste(beta,"1")), main = "")
```

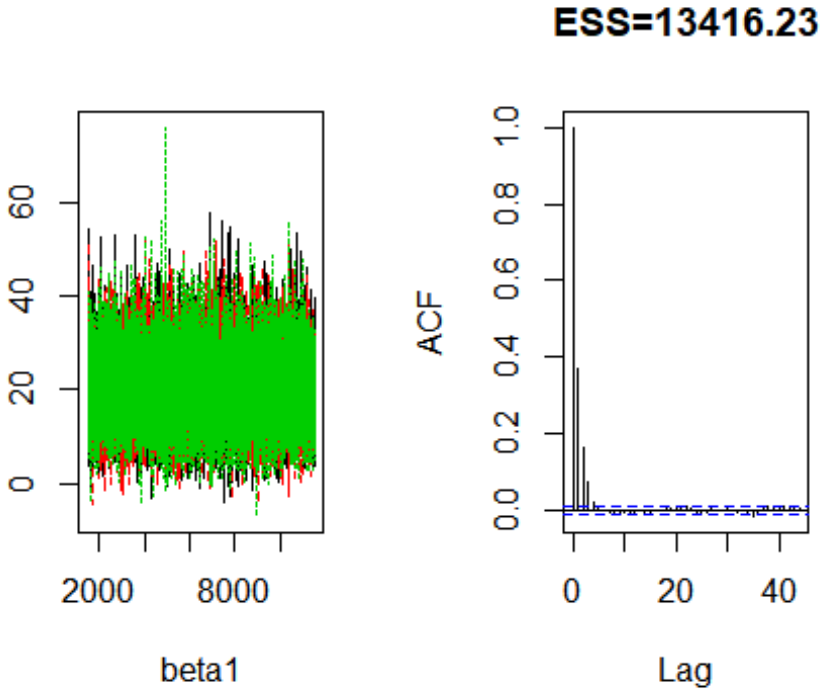



```
# independent Metropolis-Hastings
XtX = t(X) %*% X
XtX.inv = solve(XtX)
deltasq = 2.5; nAccept = 0
for(isim in 1:(nsim*nthin + nwarm)){
  beta.star <- as.vector(rmvnorm(beta.MLE, deltasq*XtX.inv))
  log.alpha <- logpost(beta.star, K, n, X, y, beta0, Sig0.inv)-logpost(beta,
K, n, X, y, beta0, Sig0.inv) + (-0.5/deltasq*t(beta-beta.MLE)%*%XtX%*%(beta-b
eta.MLE) + 0.5/deltasq*t(beta.star - beta.MLE) %*% XtX %*% (beta.star - beta.
MLE))
  u <- runif(1)
  if(log(u) < log.alpha){beta <- beta.star; nAccept = nAccept + 1}
  if(isim > nwarm & isim %% nthin == 0) beta.save[(isim - nwarm)/nthin, ] <-
beta
}

modelString="
model
{
for(i in 1:K){
    y[i] ~ dbin(pi[i], n[i])
    logit(pi[i]) <- beta0 + beta1 * (x[i] - mean(x[]))
}
beta0 ~ dnorm(mu0, invsig0)
beta1 ~ dnorm(mu1, invsig1)
```

```
}
"
writeLines(modelString, "model_binary_logit.txt")

jagsModel = jags.model(file = "model_binary_logit.txt", data = dataList, init
s = initsList, n.chains = nChains, n.adapt = nAdapt)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 8
##    Unobserved stochastic nodes: 2
##    Total graph size: 65
##
## Initializing model

update(jagsModel, n.iter = nBurn)
codaSamples = coda.samples(jagsModel, variable.names = c("beta0", "beta1"),
n.iter = nIter)
1 - rejectionRate(codaSamples)

## beta0 beta1
##     1     1

# multiple chains
convDiag.plot = function(codaSamples, var){
  ESS = effectiveSize(codaSamples[, var])
  par(mfrow = c(1,2))
  traceplot(codaSamples[, var], xlab = var)
  acf(as.matrix(codaSamples[,var]), main = paste0("ESS=", round(ESS,2)))
}

convDiag = function(codaSamples){
  ESS = effectiveSize(codaSamples); cat("ESS = ", ESS, "\n")
  acceptRate = 1-rejectionRate(codaSamples); cat("Accept rate=", acceptRate,
"\n")
  gelman = gelman.diag(codaSamples); print(gelman)
  gelman.1 = as.matrix(gelman$psrf)
  if(max(gelman.1)>1.1) cat("Warning : Gelman Shrink Factor > 1.1", "\n")
  gelman.2 = gelman$mpsrf
  if(gelman.2>1.1) cat("Warning : Gelman Multivariate Shrink Factor > 1.1", "
\n")
}


convDiag.plot(codaSamples, "beta0")
```




```
convDiag.plot(codaSamples, "beta1")
```



```r
convDiag(codaSamples)
```

```
## ESS =  14320.32 13416.23
## Accept rate= 1 1
## Potential scale reduction factors:
##
##       Point est. Upper C.I.
## beta0          1          1
## beta1          1          1
##
## Multivariate psrf
##
## 1
```

```r
kk = NULL
for(i in 1:length(xx)) kk = c(kk, rep(c(1,0), c(yy[i], nn[i]-yy[i])))
xxx = rep(xx, nn); xbar = mean(xxx); xxx = xxx - xbar
glm.out = glm(kk ~ xxx, family = binomial(link = "probit"))
beta.mle = glm.out$coefficients
SigBeta.mle = as.vector(diag(vcov(glm.out)))
dataList = list(K=8, n = nn, y = yy, x = xx, mu0 = beta.mle[1], invsig0 = 0.0
1/SigBeta.mle[1], mu1 = beta.mle[2], invsig1 = 0.01/SigBeta.mle[2])
```