Class Test: GIBBS Sampler (Two Coin Toss)

```
rm(list = ls())
graphics.off()
source("openGraphSaveGraph.R")
source("plotPost.R")
require(rjags)
```

```
## Loading required package: rjags
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.0
```

```
## Loaded modules: basemod,bugs
```

```
#------------------------------------------------------------------------------
# THE MODEL.

modelString = "
# JAGS model specification begins here...
model {
# Likelihood. Each flip is Bernoulli.
for ( i in 1 : N1 ) { y1[i] ~ dbern( theta1 ) }
for ( i in 1 : N2 ) { y2[i] ~ dbern( theta2 ) }
# Prior. Independent beta distributions.
theta1 ~ dbeta( 3 , 3 )
theta2 ~ dbeta( 3 , 3 )
}
# ... end JAGS model specification
" # close quote for modelstring

# Write the modelString to a file, using R commands:
writeLines(modelString,con="model.txt")

#------------------------------------------------------------------------------
# THE DATA.

# Specify the data in a form that is compatible with JAGS model, as a list:
dataList = list(
  N1 = 100 , # no of tosses
  y1 = c( rep(1,32),rep(0,68) ) , # 32 heads
  N2 = 100 ,
  y2 = c( rep(1,70),rep(0,30))   #70 heads
)

#------------------------------------------------------------------------------
# INTIALIZE THE CHAIN.

# Can be done automatically in jags.model() by commenting out inits argument.
# Otherwise could be established as:
# initsList = list( theta1 = sum(dataList$y1)/length(dataList$y1) ,
#                   theta2 = sum(dataList$y2)/length(dataList$y2) )
```

```
#------------------------------------------------------------------------------
# RUN THE CHAINS.

parameters = c( "theta1" , "theta2" )     # The parameter(s) to be monitored.
adaptSteps = 500              # Number of steps to "tune" the samplers.
burnInSteps = 1000            # Number of steps to "burn-in" the samplers.
nChains = 3                   # Number of chains to run.
numSavedSteps=50000           # Total number of steps in chains to save.
thinSteps=1                   # Number of steps to "thin" (1=keep every step).
nIter = ceiling( ( numSavedSteps * thinSteps ) / nChains ) # Steps per chain.
# Create, initialize, and adapt the model:
jagsModel = jags.model( "model.txt" , data=dataList , # inits=initsList ,
                        n.chains=nChains , n.adapt=adaptSteps )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 200
##    Unobserved stochastic nodes: 2
##    Total graph size: 205
##
## Initializing model
```

```
# Burn-in:
cat( "Burning in the MCMC chain...\n" )
```

```
## Burning in the MCMC chain...
```

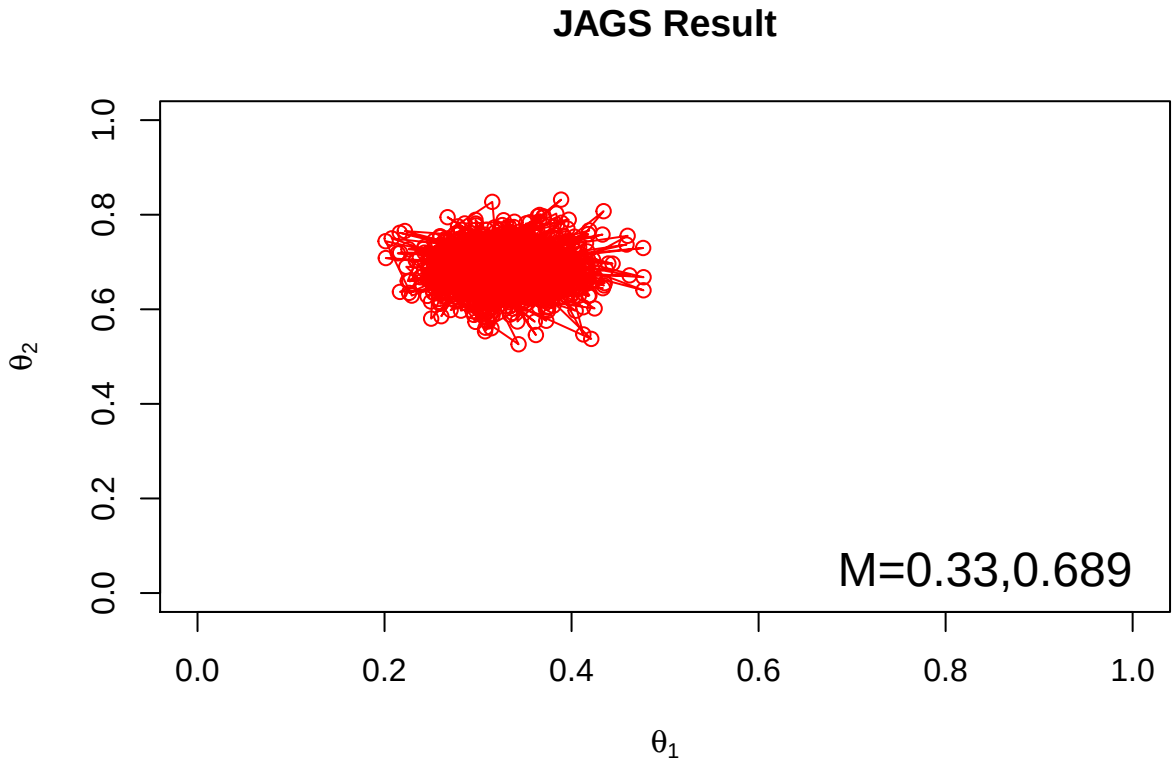
```
update( jagsModel , n.iter=burnInSteps )
# The saved MCMC chain:
cat( "Sampling final MCMC chain...\n" )
```

```
## Sampling final MCMC chain...
```

```
codaSamples = coda.samples( jagsModel , variable.names=parameters ,
                            n.iter=nIter , thin=thinSteps )
# resulting codaSamples object has these indices:
#   codaSamples[[ chainIdx ]][ stepIdx , paramIdx ]

#------------------------------------------------------------------------------
# EXAMINE THE RESULTS.

# Convert coda-object codaSamples to matrix object for easier handling.
# But note that this concatenates the different chains into one long chain.
# Result is mcmcChain[ stepIdx , paramIdx ]
mcmcChain = as.matrix( codaSamples )

theta1Sample = mcmcChain[,"theta1"] # Put sampled values in a vector.
theta2Sample = mcmcChain[,"theta2"] # Put sampled values in a vector.
```

```
chainlength=NROW(mcmcChain)
plot( theta1Sample[(chainlength-1000):chainlength] ,
      theta2Sample[(chainlength-1000):chainlength] , type = "o" ,
      xlim = c(0,1) , xlab = bquote(theta[1]) , ylim = c(0,1) ,
      ylab = bquote(theta[2]) , main="JAGS Result" , col="red" )
```

```
# Display means in plot.
theta1mean = mean(theta1Sample)
theta2mean = mean(theta2Sample)
if (theta1mean > .5) { xpos = 0.0 ; xadj = 0.0
} else { xpos = 1.0 ; xadj = 1.0 }
if (theta2mean > .5) { ypos = 0.0 ; yadj = 0.0
} else { ypos = 1.0 ; yadj = 1.0 }
text( xpos , ypos ,
      bquote(
        "M=" * .(signif(theta1mean,3)) * "," * .(signif(theta2mean,3))
      ) ,adj=c(xadj,yadj) ,cex=1.5  )
```




```
#saveGraph(file="BernTwoJags",type="eps")
```