HW04

Shashi

February 4, 2018

I have executed these exercises on my own and written the answers in my own words. Signed: Shashi Shankar

1A.

```
# Creating Data set
y = c( rep(1,48),rep(0,8) )
s = c( rep("A", 48) , rep("B", 8) )
write.csv( data.frame(y=y,s=s) , file="1A.csv" , row.names=FALSE )

# Below is just the essential lines of Jags-Ydich-XnomSsubj-MbernBeta-Example.R
# with the data file changed:
graphics.off()
rm(list=ls(all=TRUE))
fileNameRoot="1A" # for output filenames
source("DBDA2E-utilities.R")
```

```
##
## *********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *********************************************************************

## Loading required package: coda

## Linked to JAGS 4.2.0

## Loaded modules: basemod,bugs
```

```
# Load The data from the file:
myData = read.csv("1A.csv")
# Load the relevant model into R's working memory:
source("Jags-Ydich-XnomSsubj-MbernBeta.R")
```

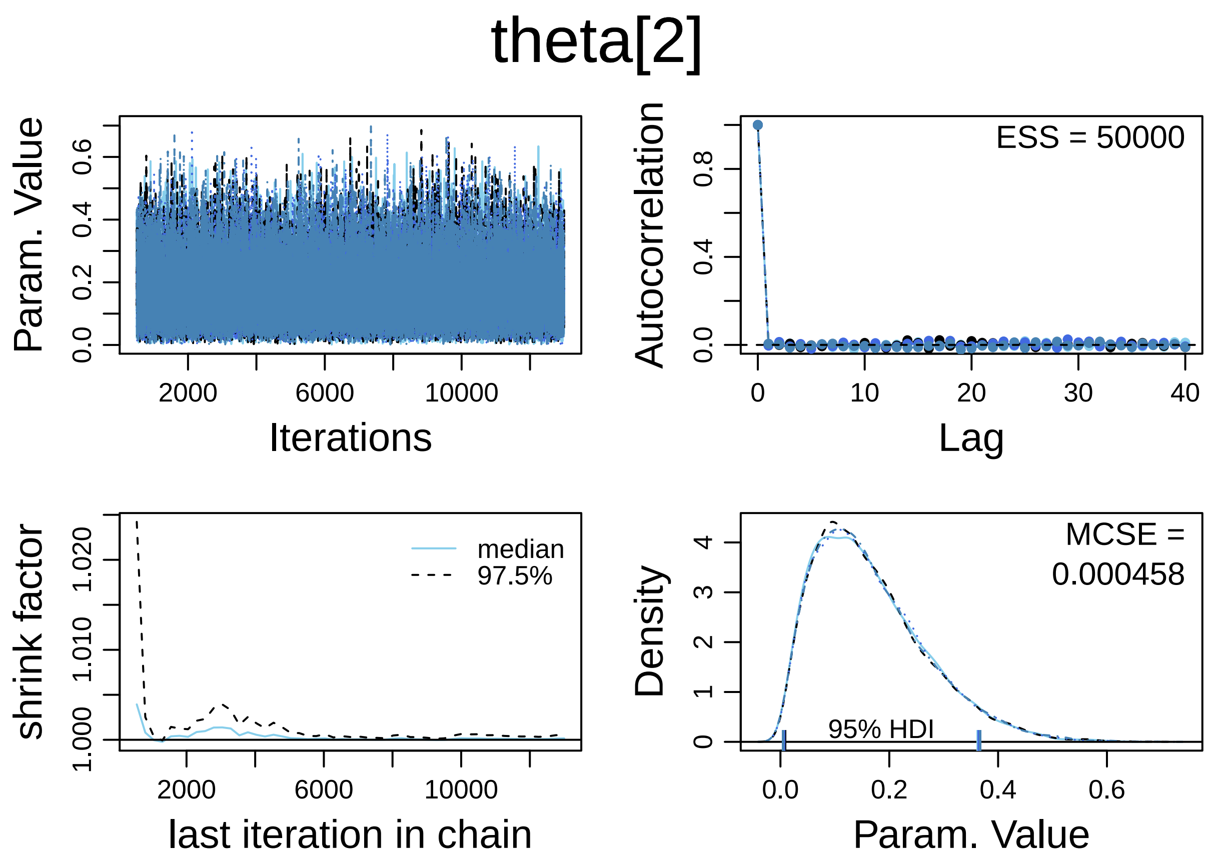
```
##
## *********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *********************************************************************
```

```
# Generate the MCMC chain:
mcmcCoda = genMCMC( data=myData , numSavedSteps=50000 , saveName=fileNameRoot )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 56
##    Unobserved stochastic nodes: 2
##    Total graph size: 120
```

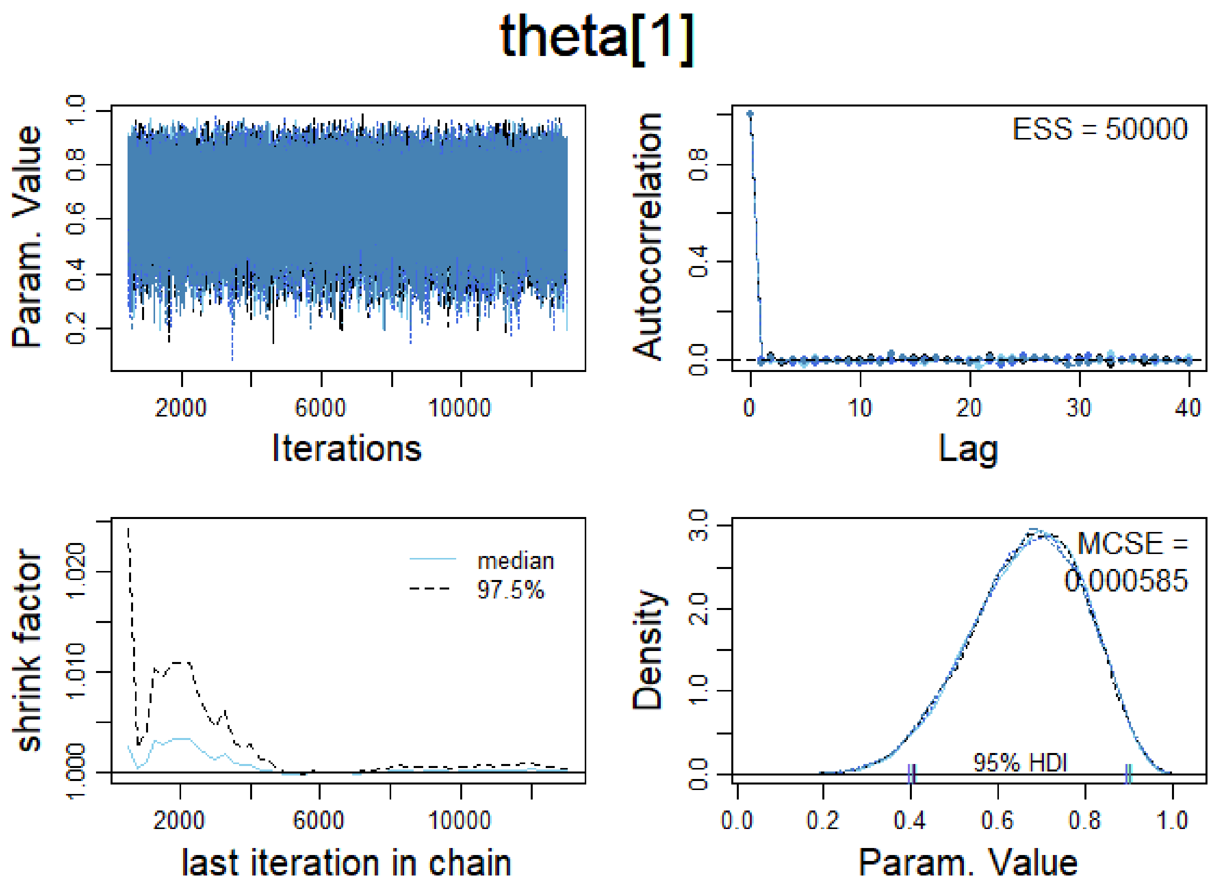
```
##
## Initializing model
##
## Burning in the MCMC chain...
## Sampling final MCMC chain...
```

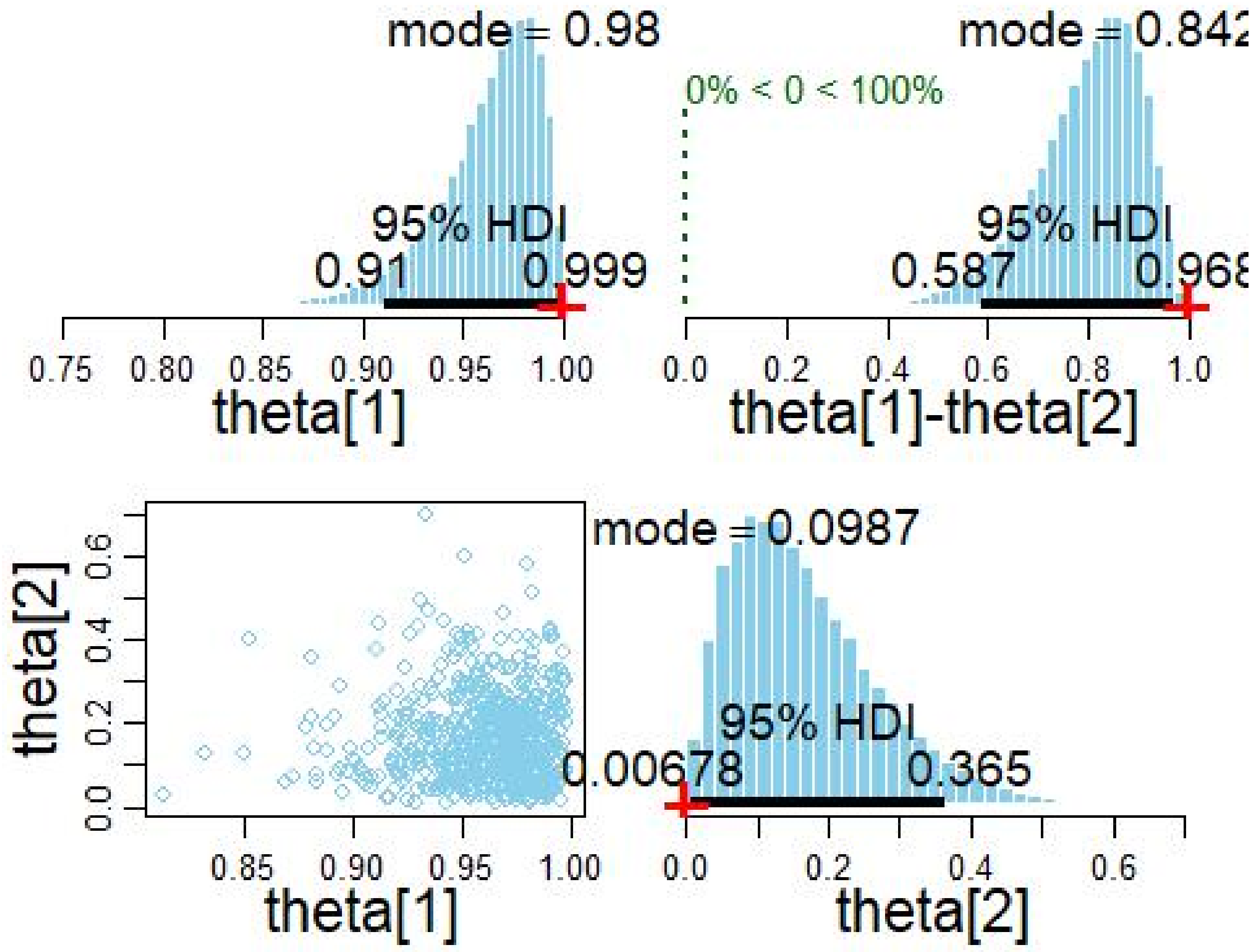
```
parameterNames = varnames(mcmcCoda) # get all parameter names
for ( parName in parameterNames ) {
diagMCMC( codaObject=mcmcCoda , parName=parName )
}
# Get summary statistics of chain:
summaryInfo = smryMCMC( mcmcCoda , compVal=NULL , #rope=c(0.45,0.55) ,
compValDiff=0.0 , #ropeDiff = c(-0.05,0.05) ,
saveName=fileNameRoot )
```



```
##                          Mean    Median      Mode   ESS HDImass      HDIlow
## theta[1]            0.9616187 0.9673847 0.9804835 50000    0.95 0.910296523
## theta[2]            0.1658624 0.1471507 0.0987357 50000    0.95 0.006783952
## theta[1]-theta[2]   0.7957563 0.8133614 0.8420911 50000    0.95 0.586909449
##                       HDIhigh CompVal PcntGtCompVal ROPElow ROPEhigh
## theta[1]            0.9988139      NA            NA      NA       NA
## theta[2]            0.3646234      NA            NA      NA       NA
## theta[1]-theta[2]   0.9678060       0           100      NA       NA
##                     PcntLtROPE PcntInROPE PcntGtROPE
## theta[1]                    NA         NA         NA
## theta[2]                    NA         NA         NA
## theta[1]-theta[2]           NA         NA         NA
```

```
# Display posterior information:
plotMCMC( mcmcCoda , data=myData , compVal=NULL , #rope=c(0.45,0.55) ,
compValDiff=0.0 , #ropeDiff = c(-0.05,0.05) ,
saveName=fileNameRoot )
```





It is evident from the posterior distribution that 95% HDI of theta1 is between 0.9 and 0.999 whereas for theta2 it's between 0 t0 0.366. The scatterplot between them also indicates the region where theta1 values are very high and and theta2 values are low.

1B.

```
# Creating Data set
y = c( rep(1,48),rep(0,8) )
s = c( rep("H", 48) , rep("T", 8) )
write.csv( data.frame(y=y,s=s) , file="1B.csv" , row.names=FALSE )

# Below is just the essential lines of Jags-Ydich-XnomSsubj-MbernBeta-Example.R
# with the data file changed:
graphics.off()
rm(list=ls(all=TRUE))
fileNameRoot="1B" # for output filenames
source("DBDA2E-utilities.R")
```

```
##
## *********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
```

```
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *******************************************************************
```

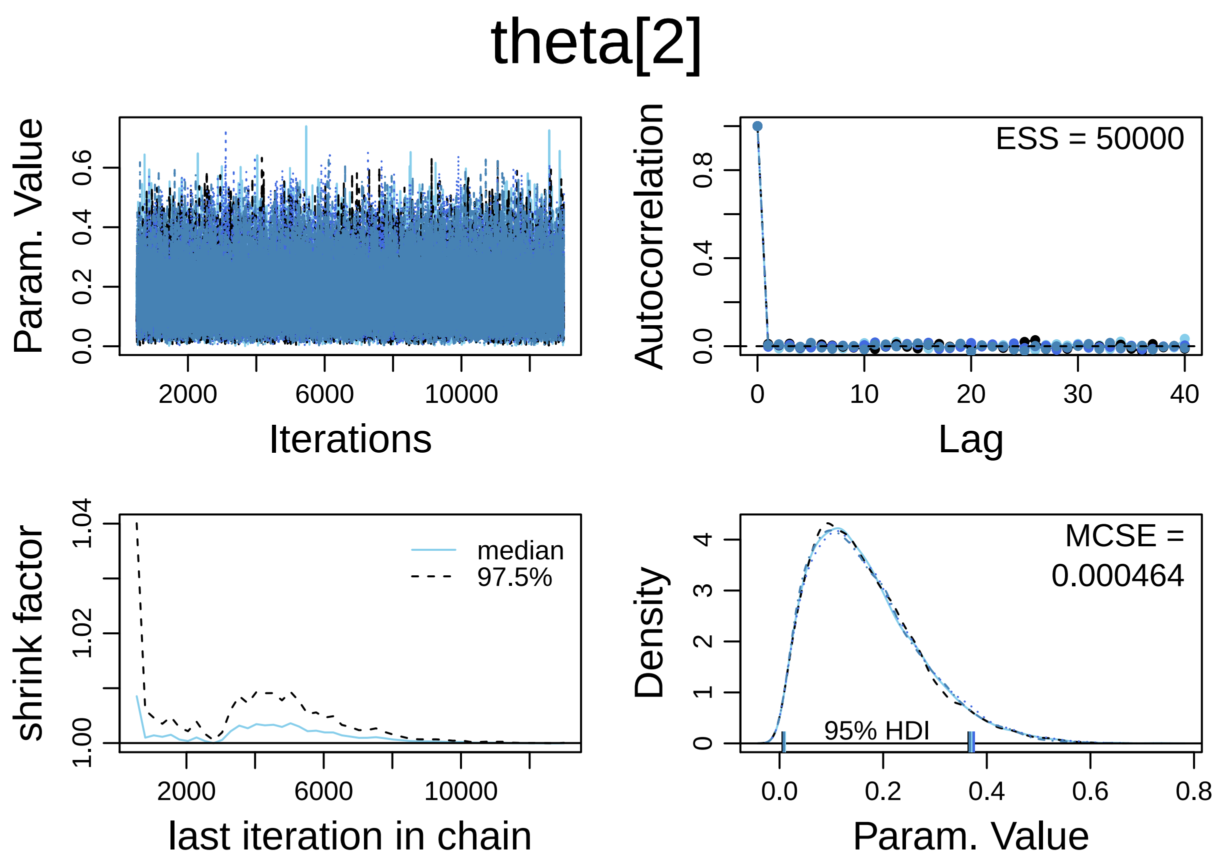
```r
# Load The data from the file:
myData = read.csv("1B.csv")
# Load the relevant model into R's working memory:
source("Jags-Ydich-XnomSsubj-MbernBeta.R")
```

```
##
## *******************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *******************************************************************
```

```r
# Generate the MCMC chain:
mcmcCoda = genMCMC( data=myData , numSavedSteps=50000 , saveName=fileNameRoot )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 56
##    Unobserved stochastic nodes: 2
##    Total graph size: 120
##
## Initializing model
##
## Burning in the MCMC chain...
## Sampling final MCMC chain...
```

```r
parameterNames = varnames(mcmcCoda) # get all parameter names
for ( parName in parameterNames ) {
diagMCMC( codaObject=mcmcCoda , parName=parName )
}
# Get summary statistics of chain:
summaryInfo = smryMCMC( mcmcCoda , compVal=NULL , #rope=c(0.45,0.55) ,
compValDiff=0.0 , #ropeDiff = c(-0.05,0.05) ,
saveName=fileNameRoot )
```



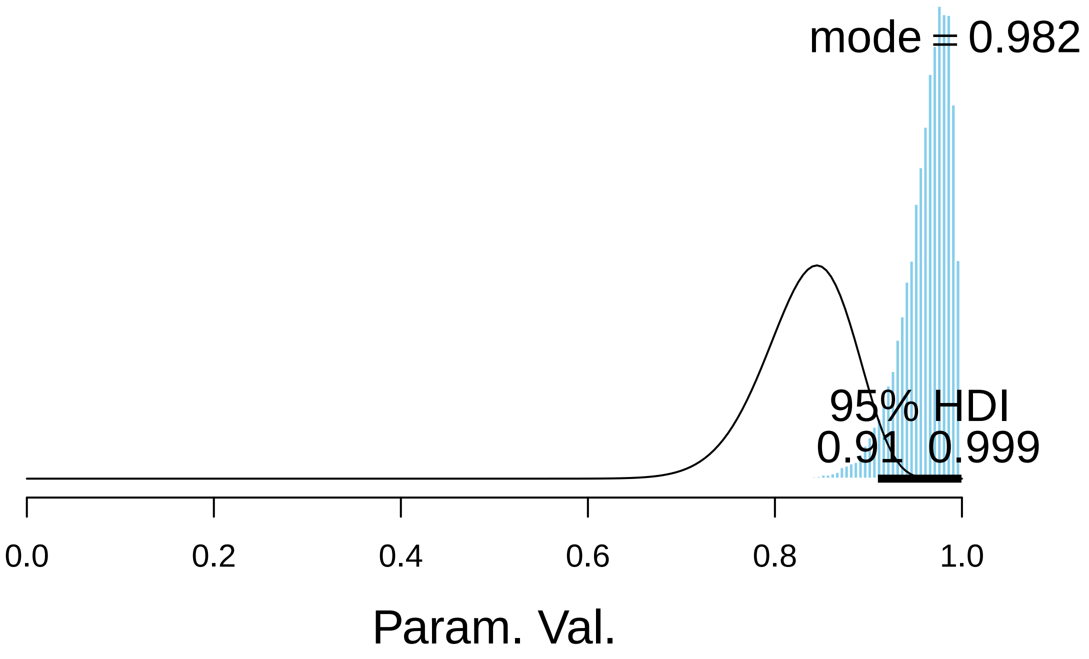
```
##                         Mean    Median      Mode     ESS HDImass
## theta[1]           0.9616096 0.9673771 0.9820770 50691.9    0.95
## theta[2]           0.1673916 0.1487069 0.0985926 50000.0    0.95
## theta[1]-theta[2] 0.7942181 0.8116921 0.8517733 50000.0    0.95
##                        HDIlow   HDIhigh CompVal PcntGtCompVal ROPElow
## theta[1]           0.910133565 0.9994874      NA            NA      NA
## theta[2]           0.006852833 0.3690453      NA            NA      NA
## theta[1]-theta[2] 0.582096630 0.9675411       0           100      NA
##                   ROPEhigh PcntLtROPE PcntInROPE PcntGtROPE
## theta[1]                NA         NA         NA         NA
## theta[2]                NA         NA         NA         NA
## theta[1]-theta[2]       NA         NA         NA         NA
```

```
# Display posterior information:
plotMCMC( mcmcCoda , data=myData , compVal=NULL , #rope=c(0.45,0.55) ,
compValDiff=0.0 , #ropeDiff = c(-0.05,0.05) ,
saveName=fileNameRoot )

openGraph()
plotPost( mcmcCoda[,"theta[1]"] , xlim=c(0,1) )
```

```
##                    ESS      mean    median     mode hdiMass    hdiLow
## Param. Val. 50691.94 0.9616096 0.9673771 0.982077    0.95 0.9101336
##                hdiHigh compVal pGtCompVal ROPElow ROPEhigh pLtROPE pInROPE
## Param. Val. 0.9994874      NA         NA      NA       NA      NA      NA
##             pGtROPE
## Param. Val.      NA
```

```
a = 2 ; b = 2 # constants from prior in Jags-Ydich-XnomSsubj-MbernBeta.R
H = 48 ; T = 8 # heads and tails from your data
thetaGrid=seq(0,1,length=201)
lines( thetaGrid , dbeta( thetaGrid , a+H , b+T ) )
```



H and T are the number of heads and number of tails in the data. Using z and N notation, H is z and T is N-z. We are plotting a beta distribution with shape constants of a+H and b+T, Because when we start with a beta(a,b) prior we end up with a beta(a+z,b+N-z) posterior.

a and b (both 2) were determined from the model specification in low level script file dbeta(2,2). Heads and tails are 48 and 8 respectively.

The lines( ) function adds information to a graph. It can not produce a graph on its own. Usually it follows a plot(x, y) command that produces a graph.

The superimposed curve is closely matching the histogram.

2.

```
source("Jags-Ydich-XnomSsubj-Mbernbeta-Example.R")
```

```
##
## *********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *********************************************************************
```

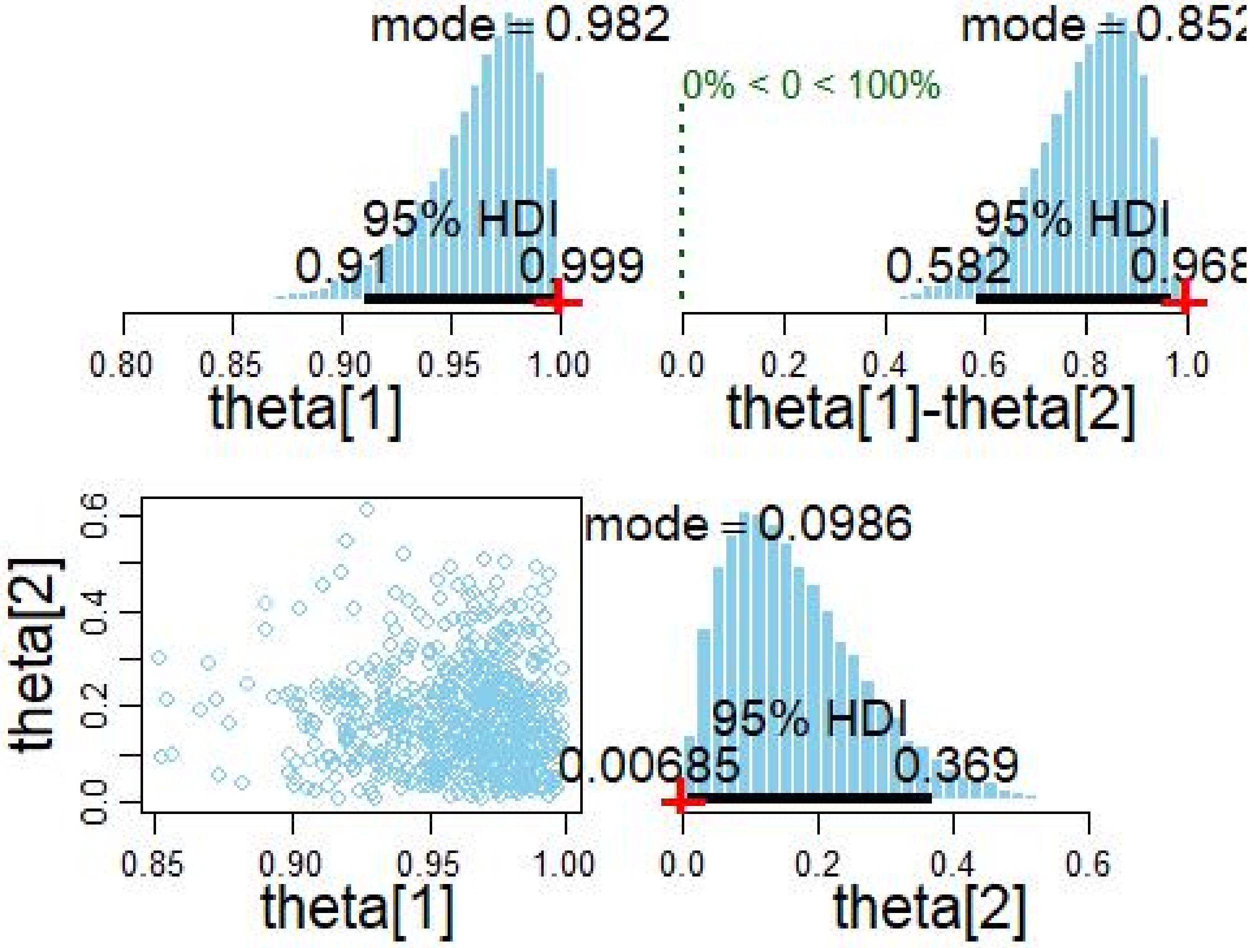

mode = 0.982
95% HDI
0.91
0.999
0.80 0.85 0.90 0.95 1.00
theta[1]
mode = 0.852
0% < 0 < 100%
95% HDI
0.582
0.968
0.0 0.2 0.4 0.6 0.8 1.0
theta[1]-theta[2]
theta[2]
0.0 0.2 0.4 0.6
0.85 0.90 0.95 1.00
theta[1]
mode = 0.0986
95% HDI
0.00685
0.369
0.0 0.2 0.4 0.6
theta[2]

Figure 1:

```
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 2
##    Total graph size: 38
##
## Initializing model
##
## Burning in the MCMC chain...
## Sampling final MCMC chain...
```

```
##                          Mean    Median      Mode     ESS HDImass
## theta[1]           0.6666179 0.6765121 0.7045672 51183.4    0.95
## theta[2]           0.3634230 0.3550149 0.3070597 50000.0    0.95
## theta[1]-theta[2] 0.3031949 0.3115623 0.3333145 50000.0    0.95
##                          HDIlow   HDIhigh CompVal PcntGtCompVal ROPElow
## theta[1]            0.41292323 0.9078335      NA            NA    0.45
## theta[2]            0.10608969 0.6336525      NA            NA    0.45
## theta[1]-theta[2] -0.06854586 0.6685918       0        93.586   -0.05
##                   ROPEhigh PcntLtROPE PcntInROPE PcntGtROPE
## theta[1]              0.55      6.198     12.936     80.866
## theta[2]              0.55     73.466     16.370     10.164
## theta[1]-theta[2]     0.05      3.944      6.010     90.046
```

The summary results display the Mean, Median, Mode of the MCMC chain for either the parameter or parameter difference. Each row corresponds to the parameter or parameter difference indicated in the left-most column. ESS is the effective sample size, which is the chain length divided by the autocorrelation. HDImass indicates the probability mass of highest density interval (default 95 %). HDIlow is the lower limit of the HDI, and HDIhigh is the upper limit. Comparison value (CompVal) for single-parameter decisions. It's value is shown as NA because it was commented in the code. The next column indicates the percentage of the posterior that is greater than the comparison value (PcntGtCompVal). Next are the columns for the ROPE (region of practical equivalence), which show the specifications in the arguments. The last three columns indicate the percentage of the posterior distribution that is less than the ROPE lower limit, within the ROPE limits, and greater than the ROPE upper limit.

3.

fileNameRoot = "Jags-Ydich-XnomSsubj-MbernBeta-" graphFileType = "eps" fileNameRoot specifies the beginning of the filenames for saved information, and graphFileType specifies the graphics format for the saved graphs.

Generate the MCMC chain: mcmcCoda = genMCMC(data=myData , numSavedSteps=50000 , saveName=fileNameRoot)

genMCMC returns mcmccoda object containing information about every step. The MCMC chain is saved in a file named <fileNameRoot-Mcmc.Rdata file.

Display diagnostics of chain, for specified parameters:

parameterNames = varnames(mcmcCoda) # get all parameter names for (parName in parameterNames) { diagMCMC(codaObject=mcmcCoda , parName=parName , saveName=fileNameRoot , saveType=graphFileType) }

The above line gets all the parameters from the mcmccoda object and saves the diagnostic graphs for each parameter. file names are the fileNameRoot with Diag appended.

Get summary statistics of chain:

summaryInfo = smryMCMC(mcmcCoda , compVal=0.5 , rope=c(0.45,0.55) , compValDiff=0.0 , ropeDiff = c(-0.05,0.05) , saveName=fileNameRoot)

The summary information is saved in a file named Jags-Ydich-XnomSsubj-MbernBeta-SummaryInfo.csv. The file name is the fileNameRoot with SummaryInfo appended.

Display posterior information:

plotMCMC(mcmcCoda , data=myData , compVal=NULL , #rope=c(0.45,0.55) , compValDiff=0.0 , #ropeDiff = c(-0.05,0.05) , saveName=fileNameRoot , saveType=graphFileType)

The graph of the posterior distribution is saved in a file named Jags-Ydich-XnomSsubj-MbernBeta-Post.eps file name is the fileNameRoot with Post appended.

Note: .eps is a file format optimized for creating large LaTeX files.

4A.

To reproduce figure 8.7 we need to run JAGS without the data included. So, the y values must be omitted, but all the other constants must be retained in order to define the structure of the model, such as the number of (absent) data values and the number of (absent) subjects etc.

dataList = list(# y = y , s = s , Ntotal = Ntotal , Nsubj = Nsubj)

```r
source("Jags-Ydich-XnomSsubj-Mbernbeta-Example.R")
```

```
##
## *********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *********************************************************************
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 2
##    Total graph size: 38
##
## Initializing model
##
```

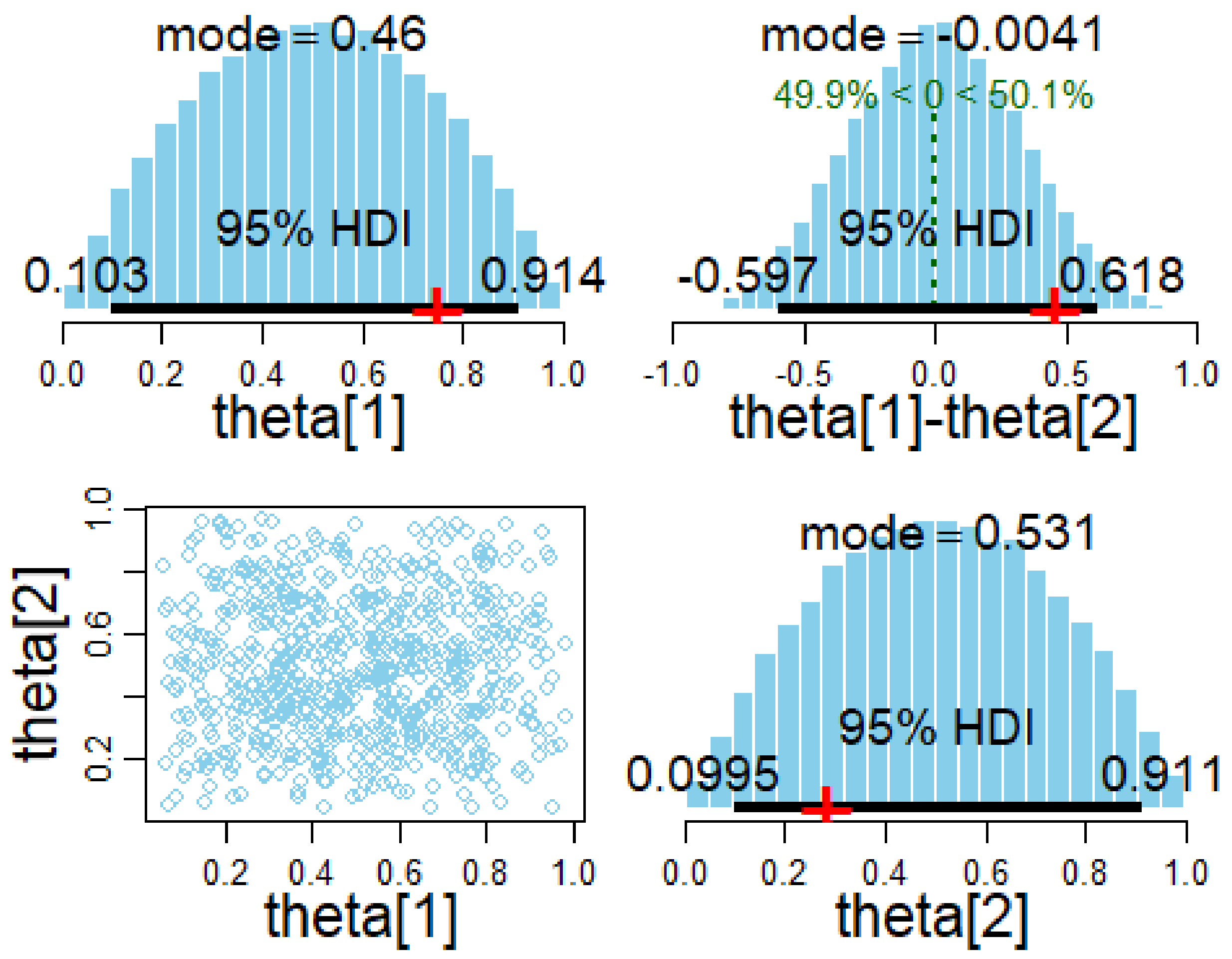


Figure 2: Posterior

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

```
##                        Mean    Median      Mode   ESS HDImass      HDIlow
## theta[1]          0.6657118 0.6742455 0.6767565 50000    0.95  0.41083556
## theta[2]          0.3639285 0.3553744 0.3305371 50000    0.95  0.10818487
## theta[1]-theta[2] 0.3017833 0.3095400 0.3505052 50000    0.95 -0.08042172
##                     HDIhigh CompVal PcntGtCompVal ROPElow ROPEhigh
## theta[1]          0.9045921      NA            NA    0.45     0.55
## theta[2]          0.6391385      NA            NA    0.45     0.55
## theta[1]-theta[2] 0.6559541       0        93.612   -0.05     0.05
##                   PcntLtROPE PcntInROPE PcntGtROPE
## theta[1]               6.176     13.148     80.676
## theta[2]              73.440     16.138     10.422
## theta[1]-theta[2]      3.936      6.142     89.922
```

The graph of the "posterior" looks like that in Figure 8.7 from the book. ## 4B.

Changed the prior specification to dbeta(1, 1) # THE MODEL. modelString = " model { for (i in 1:Ntotal) { y[i] ~ dbern(theta[s[i]]) } for (sIdx in 1:Nsubj) { theta[sIdx] ~ dbeta(1 , 1) # N.B.: 2,2 prior; change as appropriate. } } " # close quote for modelString writeLines(modelString , con="TEMPmodel.txt")

```
source("Jags-Ydich-XnomSsubj-Mbernbeta-Example.R")
```

```
##
## *********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *********************************************************************
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 2
##    Total graph size: 38
##
## Initializing model
##
## Burning in the MCMC chain...
## Sampling final MCMC chain...
```

```
##                          Mean    Median      Mode     ESS HDImass
## theta[1]            0.6670184 0.6761939 0.7116743 50000.0    0.95
## theta[2]            0.3641594 0.3554050 0.3398825 50000.0    0.95
## theta[1]-theta[2]   0.3028589 0.3116637 0.3404633 50675.7    0.95
##                          HDIlow   HDIhigh CompVal PcntGtCompVal ROPElow
## theta[1]             0.41137733 0.9042805      NA            NA    0.45
## theta[2]             0.10759847 0.6332177      NA            NA    0.45
## theta[1]-theta[2]   -0.07505761 0.6606989       0        93.528   -0.05
##                     ROPEhigh PcntLtROPE PcntInROPE PcntGtROPE
## theta[1]                0.55      5.978     12.898     81.124
## theta[2]                0.55     73.206     16.426     10.368
## theta[1]-theta[2]       0.05      3.846      6.374     89.780
```

The distributions on theta[1] and theta[2] look uniform, because that is a dbeta(1,1) distribution and we have commented the data y in the low level script. However, the prior distribution on theta[1]-theta[2] is triangular. There are a lot of points along the diagonal, but the number of points drops off linearly toward the corners. It shows that uniform priors on theta[1] and theta[2] do not imply a uniform prior on the difference of parameters.

4C.

Changed the prior specification to dbeta(0.5, 0.5)

```
source("Jags-Ydich-XnomSsubj-Mbernbeta-Example.R")
```

```
##
## *********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *********************************************************************
##
## Compiling model graph
##    Resolving undeclared variables
```

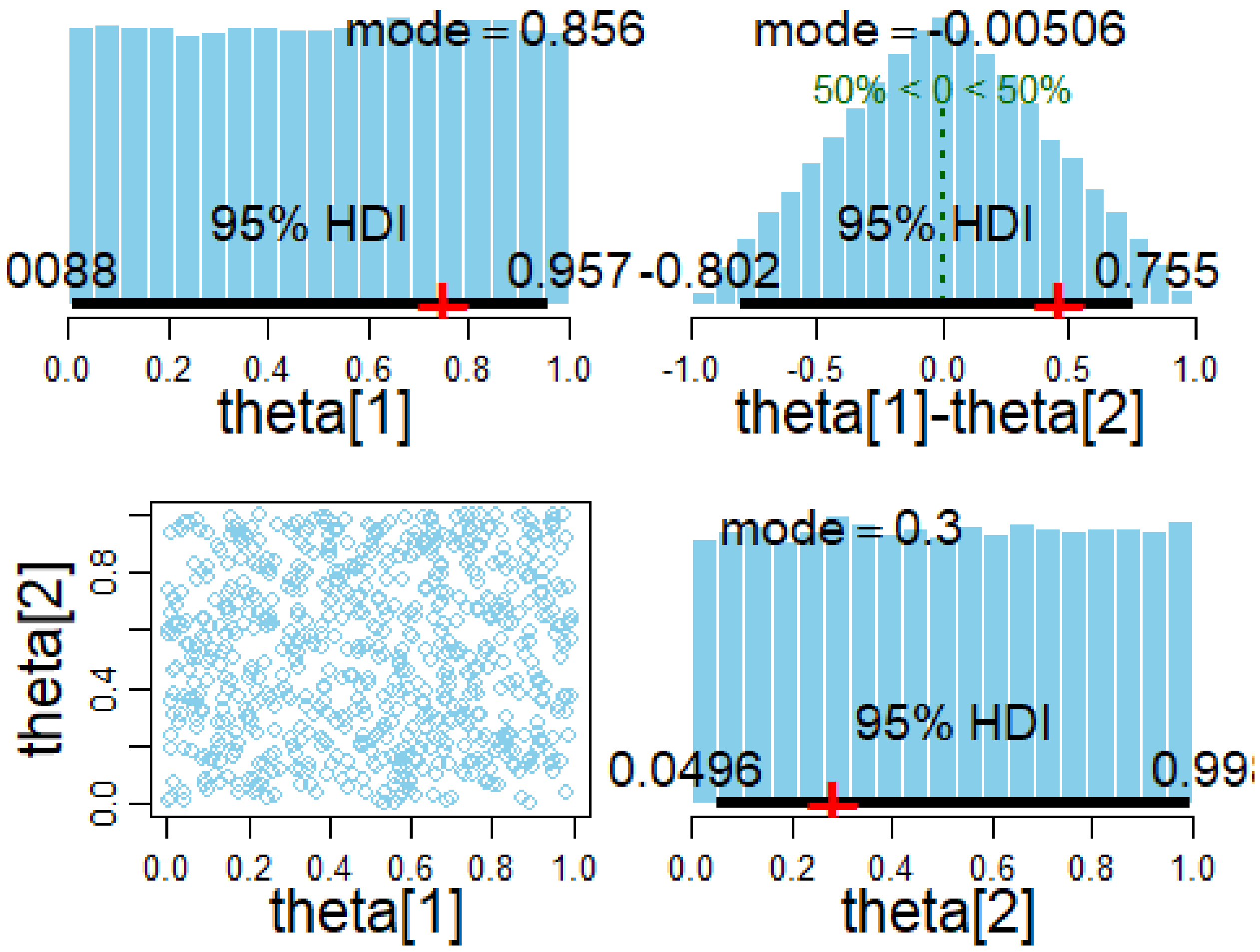


Figure 3: Posterior

```
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 2
##    Total graph size: 38
##
## Initializing model
##
## Burning in the MCMC chain...
## Sampling final MCMC chain...
```

```
##                         Mean    Median      Mode   ESS HDImass      HDIlow
## theta[1]           0.6658859 0.6762045 0.6937260 50000    0.95  0.41141731
## theta[2]           0.3635732 0.3553315 0.3462667 50000    0.95  0.11013089
## theta[1]-theta[2] 0.3023127 0.3099067 0.3142692 50000    0.95 -0.05841552
##                      HDIhigh CompVal PcntGtCompVal ROPElow ROPEhigh
## theta[1]           0.9054943      NA            NA    0.45     0.55
## theta[2]           0.6349476      NA            NA    0.45     0.55
## theta[1]-theta[2] 0.6769620       0        93.676   -0.05     0.05
##                    PcntLtROPE PcntInROPE PcntGtROPE
## theta[1]                6.252     12.914     80.834
## theta[2]               73.258     16.580     10.162
## theta[1]-theta[2]       3.736      6.100     90.164
```

The individual parameters have dbeta(0.5,0.5) marginal distributions. The scatter plot has higher density toward the corners. It shows that a prior on individual parameters may have unforeseen implications for the prior on the difference of parameters.

5A.

```
# Creating Data set
y = c( rep(1,48),rep(0,8) )
s = c( rep("A", 48) , rep("B", 8) )
write.csv( data.frame(y=y,s=s) , file="1A.csv" , row.names=FALSE )

# Below is just the essential lines of Jags-Ydich-XnomSsubj-MbernBeta-Example.R
# with the data file changed:
graphics.off()
rm(list=ls(all=TRUE))
fileNameRoot="5A_2000" # for output filenames
source("DBDA2E-utilities.R")
```

```
##
## *********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *********************************************************************
```

```
# Load The data from the file:
myData = read.csv("1A.csv")
# Load the relevant model into R's working memory:
source("Jags-Ydich-XnomSsubj-MbernBeta.R")
```

```
##
## *********************************************************************
```

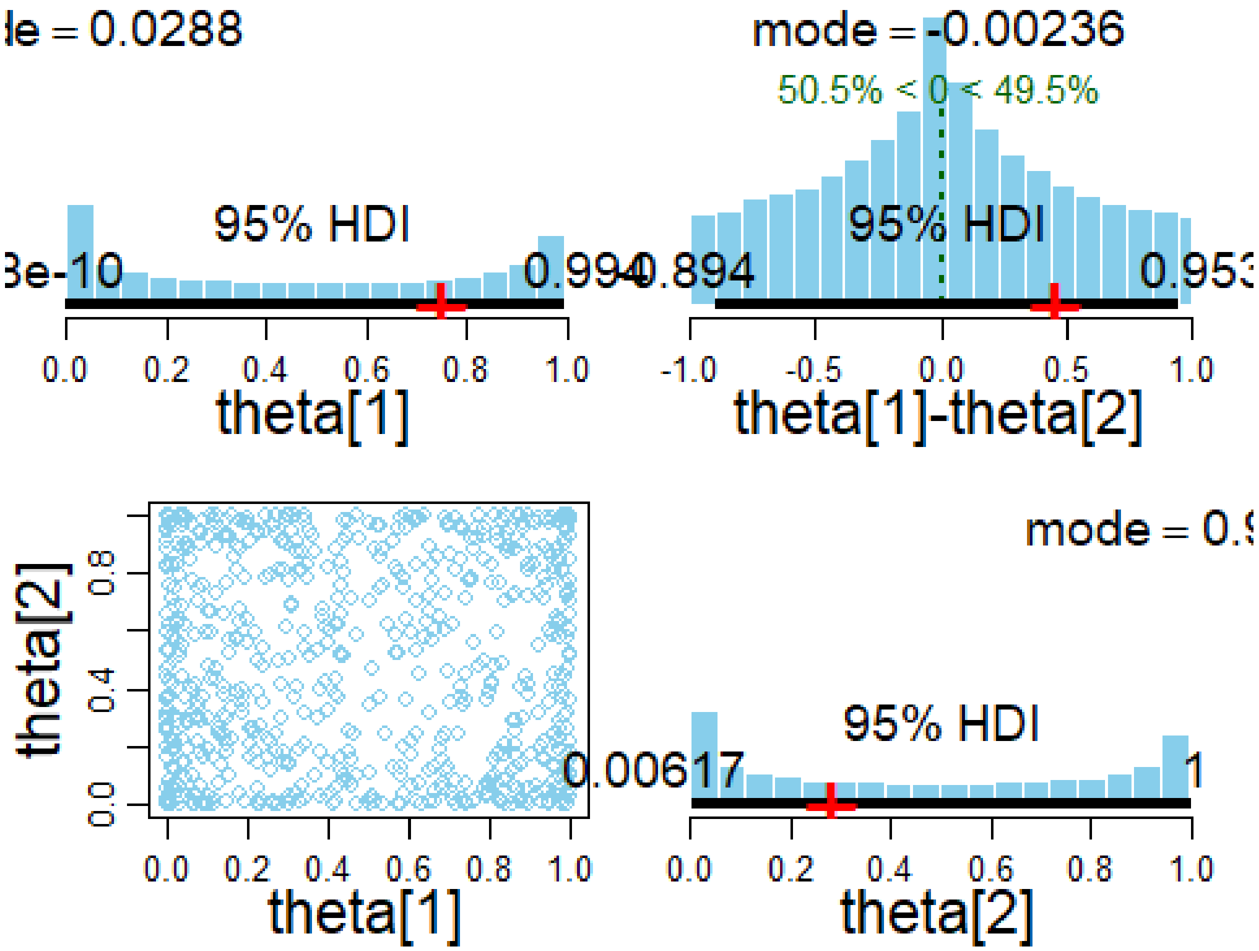


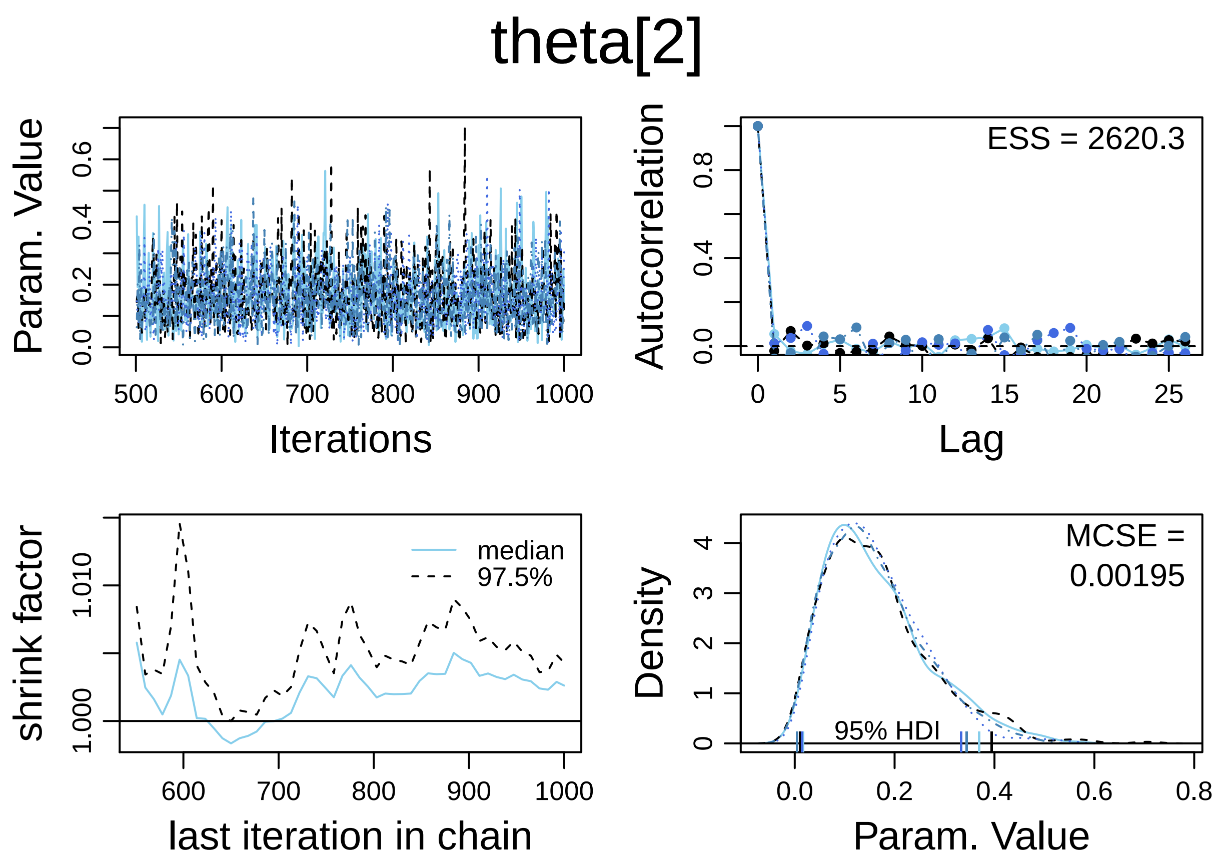
Figure 4: Posterior

```
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *******************************************************************
```

```
# Generate the MCMC chain:
mcmcCoda = genMCMC( data=myData , numSavedSteps=2000 , saveName=fileNameRoot )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 56
##    Unobserved stochastic nodes: 2
##    Total graph size: 120
##
## Initializing model
##
## Burning in the MCMC chain...
## Sampling final MCMC chain...
```

```
parameterNames = varnames(mcmcCoda) # get all parameter names
for ( parName in parameterNames ) {
diagMCMC( codaObject=mcmcCoda , parName=parName )
}
# Get summary statistics of chain:
summaryInfo = smryMCMC( mcmcCoda , compVal=NULL , #rope=c(0.45,0.55) ,
compValDiff=0.0 , #ropeDiff = c(-0.05,0.05) ,
saveName=fileNameRoot )
```



```
##                       Mean    Median      Mode  ESS HDImass    HDIlow
## theta[1]         0.9620389 0.9683388 0.9801023 2000    0.95 0.91005739
## theta[2]         0.1635556 0.1455136 0.1063253 2000    0.95 0.01554036
## theta[1]-theta[2] 0.7984832 0.8156241 0.8286634 2000    0.95 0.59395543
##                    HDIhigh CompVal PcntGtCompVal ROPElow ROPEhigh
## theta[1]         0.9985612      NA            NA      NA       NA
## theta[2]         0.3666454      NA            NA      NA       NA
## theta[1]-theta[2] 0.9675582       0           100      NA       NA
##                  PcntLtROPE PcntInROPE PcntGtROPE
## theta[1]                 NA         NA         NA
## theta[2]                 NA         NA         NA
## theta[1]-theta[2]        NA         NA         NA
```

```
# Display posterior information:
plotMCMC( mcmcCoda , data=myData , compVal=NULL , #rope=c(0.45,0.55) ,
compValDiff=0.0 , #ropeDiff = c(-0.05,0.05) ,
saveName=fileNameRoot )
```

With 2000 steps the diagnostic plot for autocoreelation is not very steady. The chains are not very well superimposed on each other.

```
# Creating Data set
y = c( rep(1,48),rep(0,8) )
s = c( rep("A", 48) , rep("B", 8) )
write.csv( data.frame(y=y,s=s) , file="1A.csv" , row.names=FALSE )

# Below is just the essential lines of Jags-Ydich-XnomSsubj-MbernBeta-Example.R
# with the data file changed:
```

```r
graphics.off()
rm(list=ls(all=TRUE))
fileNameRoot="5A_50000" # for output filenames
source("DBDA2E-utilities.R")
```

```
##
## *********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *********************************************************************
```

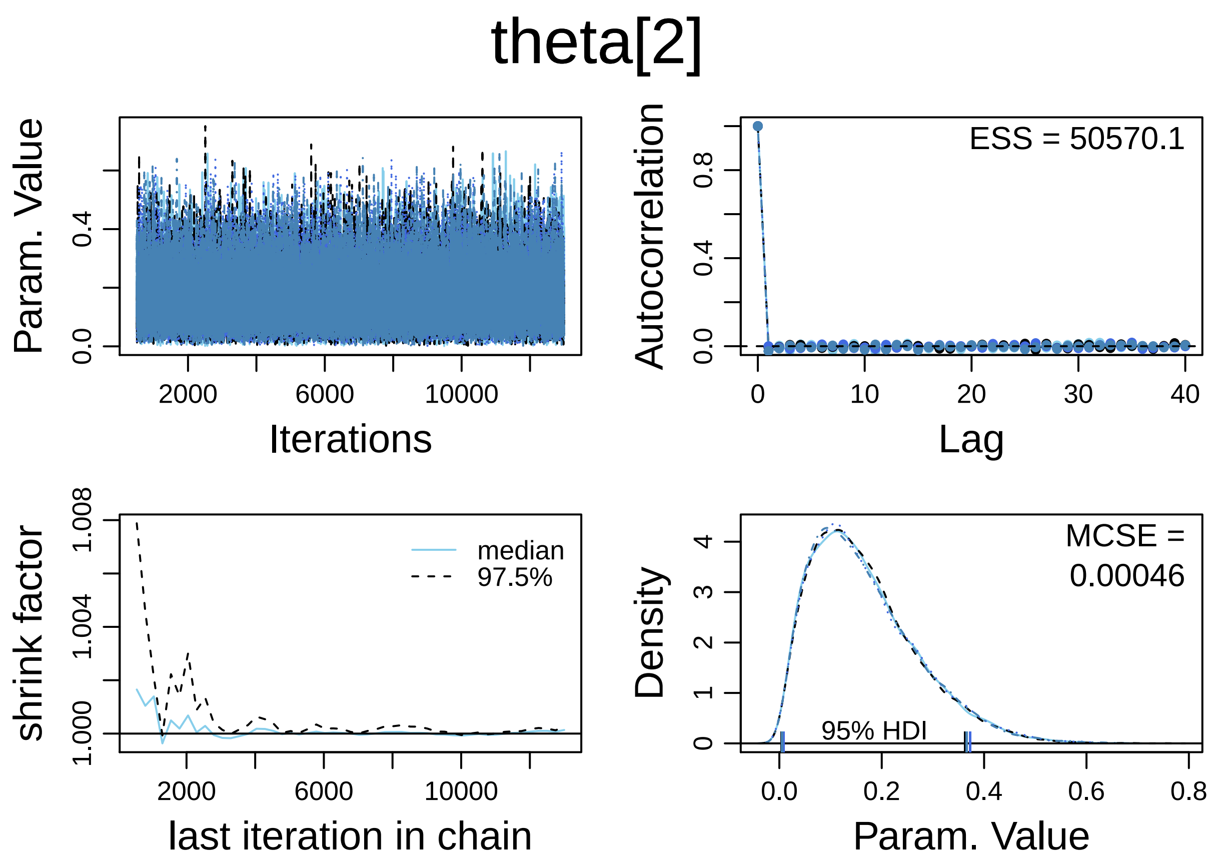
```r
# Load The data from the file:
myData = read.csv("1A.csv")
# Load the relevant model into R's working memory:
source("Jags-Ydich-XnomSsubj-MbernBeta.R")
```

```
##
## *********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *********************************************************************
```

```r
# Generate the MCMC chain:
mcmcCoda = genMCMC( data=myData , numSavedSteps=50000 , saveName=fileNameRoot )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 56
##    Unobserved stochastic nodes: 2
##    Total graph size: 120
##
## Initializing model
##
## Burning in the MCMC chain...
## Sampling final MCMC chain...
```

```r
parameterNames = varnames(mcmcCoda) # get all parameter names
for ( parName in parameterNames ) {
diagMCMC( codaObject=mcmcCoda , parName=parName )
}
# Get summary statistics of chain:
summaryInfo = smryMCMC( mcmcCoda , compVal=NULL , #rope=c(0.45,0.55) ,
compValDiff=0.0 , #ropeDiff = c(-0.05,0.05) ,
saveName=fileNameRoot )
```



```
##                         Mean    Median      Mode     ESS HDImass
## theta[1]           0.9615905 0.9673809 0.9793615 50000.0    0.95
## theta[2]           0.1668195 0.1480921 0.1119860 50682.1    0.95
## theta[1]-theta[2] 0.7947710 0.8128063 0.8524979 50000.0    0.95
##                         HDIlow   HDIhigh CompVal PcntGtCompVal ROPElow
## theta[1]           0.910641573 0.9987797      NA            NA      NA
## theta[2]           0.005069766 0.3662000      NA            NA      NA
## theta[1]-theta[2] 0.583425287 0.9674661       0           100      NA
##                   ROPEhigh PcntLtROPE PcntInROPE PcntGtROPE
## theta[1]                NA         NA         NA         NA
## theta[2]                NA         NA         NA         NA
## theta[1]-theta[2]       NA         NA         NA         NA
```

```
# Display posterior information:
plotMCMC( mcmcCoda , data=myData , compVal=NULL , #rope=c(0.45,0.55) ,
compValDiff=0.0 , #ropeDiff = c(-0.05,0.05) ,
saveName=fileNameRoot )
```

With 50000 steps the diagnostic plot for autocoreelation for all the chains are close to 0. The chains are very well superimposed on each other.

5B.

```
# Jags-Ydich-XnomSsubj-Mbernbeta.R
# Accompanies the book:
#   Kruschke, J. K. (2014). Doing Bayesian Data Analysis:
```

```r
#   A Tutorial with R, JAGS, and Stan. 2nd Edition. Academic Press / Elsevier.
source("DBDA2E-utilities.R")
```

```
##
## *********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *********************************************************************
```

```r
#===============================================================================

genMCMC = function( data , numSavedSteps=50000 , saveName=NULL ) {
  require(rjags)
  #-----------------------------------------------------------------------------
  # THE DATA.
  # N.B.: This function expects the data to be a data frame,
  # with one component named y being a vector of integer 0,1 values,
  # and one component named s being a factor of subject identifiers.
  y = data$y
  s = as.numeric(data$s) # converts character to consecutive integer levels
  # Do some checking that data make sense:
  if ( any( y!=0 & y!=1 ) ) { stop("All y values must be 0 or 1.") }
  Ntotal = length(y)
  Nsubj = length(unique(s))
  # Specify the data in a list, for later shipment to JAGS:
  dataList = list(
    y = y ,
    s = s ,
    Ntotal = Ntotal ,
    Nsubj = Nsubj
  )
  #-----------------------------------------------------------------------------
  # THE MODEL.
  modelString = "
  model {
    for ( i in 1:Ntotal ) {
      y[i] ~ dbern( theta[s[i]] )
    }
    for ( sIdx in 1:Nsubj ) {
      theta[sIdx] ~ dbeta( 0.5 , 0.5 ) # N.B.: 2,2 prior; change as appropriate.
    }
  }
  " # close quote for modelString
  writeLines( modelString , con="TEMPmodel.txt" )
  #-----------------------------------------------------------------------------
  # INTIALIZE THE CHAINS.
  # Initial values of MCMC chains based on data:
  # Option 1: Use single initial value for all chains:
  #  thetaInit = rep(0,Nsubj)
  #  for ( sIdx in 1:Nsubj ) { # for each subject
  #    includeRows = ( s == sIdx ) # identify rows of this subject
  #    yThisSubj = y[includeRows]  # extract data of this subject
  #    thetaInit[sIdx] = sum(yThisSubj)/length(yThisSubj) # proportion
  #  }
  #  initsList = list( theta=thetaInit )
  # Option 2: Use function that generates random values near MLE:
  initsList = function() {
    thetaInit = rep(0,Nsubj)
    for ( sIdx in 1:Nsubj ) { # for each subject
      includeRows = ( s == sIdx ) # identify rows of this subject
      yThisSubj = y[includeRows]  # extract data of this subject
      resampledY = sample( yThisSubj , replace=TRUE ) # resample
      thetaInit[sIdx] = sum(resampledY)/length(resampledY)
    }
    thetaInit = 0.001+0.998*thetaInit # keep away from 0,1
    return( list( theta=thetaInit ) )
  }
  #-----------------------------------------------------------------------------
  # RUN THE CHAINS
  parameters = c( "theta")     # The parameters to be monitored
  adaptSteps = 500             # Number of steps to adapt the samplers
  burnInSteps = 500            # Number of steps to burn-in the chains
  nChains = 4                  # nChains should be 2 or more for diagnostics
  thinSteps = 1
  nIter = ceiling( ( numSavedSteps * thinSteps ) / nChains )
  # Create, initialize, and adapt the model:
  jagsModel = jags.model( "TEMPmodel.txt" , data=dataList , #inits=initsList ,
                          n.chains=nChains , n.adapt=adaptSteps )
  # Burn-in:
  cat( "Burning in the MCMC chain...\n" )
  update( jagsModel , n.iter=burnInSteps )
  # The saved MCMC chain:
  cat( "Sampling final MCMC chain...\n" )
  codaSamples = coda.samples( jagsModel , variable.names=parameters ,
                              n.iter=nIter , thin=thinSteps )
  # resulting codaSamples object has these indices:
  #   codaSamples[[ chainIdx ]][ stepIdx , paramIdx ]
  if ( !is.null(saveName) ) {
    save( codaSamples , file=paste(saveName,"Mcmc.Rdata",sep="") )
  }
  return( codaSamples )
} # end function

#===============================================================================

smryMCMC = function(  codaSamples , compVal=0.5 , rope=NULL ,
                      compValDiff=0.0 , ropeDiff=NULL , saveName=NULL ) {
  mcmcMat = as.matrix(codaSamples,chains=TRUE)
  Ntheta = length(grep("theta",colnames(mcmcMat)))
  summaryInfo = NULL
  rowIdx = 0
  for ( tIdx in 1:Ntheta ) {
    parName = paste0("theta[",tIdx,"]")
    summaryInfo = rbind( summaryInfo ,
      summarizePost( mcmcMat[,parName] , compVal=compVal , ROPE=rope ) )
    rowIdx = rowIdx+1
    rownames(summaryInfo)[rowIdx] = parName
  }
  for ( t1Idx in 1:(Ntheta-1) ) {
    for ( t2Idx in (t1Idx+1):Ntheta ) {
      parName1 = paste0("theta[",t1Idx,"]")
      parName2 = paste0("theta[",t2Idx,"]")
      summaryInfo = rbind( summaryInfo ,
        summarizePost( mcmcMat[,parName1]-mcmcMat[,parName2] ,
                       compVal=compValDiff , ROPE=ropeDiff ) )
      rowIdx = rowIdx+1
      rownames(summaryInfo)[rowIdx] = paste0(parName1,"-",parName2)
    }
  }
  if ( !is.null(saveName) ) {
    write.csv( summaryInfo , file=paste(saveName,"SummaryInfo.csv",sep="") )
  }
  show( summaryInfo )
  return( summaryInfo )
}

#===============================================================================

plotMCMC = function( codaSamples , data , compVal=0.5 , rope=NULL ,
                     compValDiff=0.0 , ropeDiff=NULL ,
                     saveName=NULL , saveType="jpg" ) {
  #-----------------------------------------------------------------------------
  # N.B.: This function expects the data to be a data frame,
  # with one component named y being a vector of integer 0,1 values,
  # and one component named s being a factor of subject identifiers.
  y = data$y
  s = as.numeric(data$s) # converts character to consecutive integer levels
  # Now plot the posterior:
  mcmcMat = as.matrix(codaSamples,chains=TRUE)
  chainLength = NROW( mcmcMat )
  Ntheta = length(grep("theta",colnames(mcmcMat)))
  openGraph(width=2.5*Ntheta,height=2.0*Ntheta)
  par( mfrow=c(Ntheta,Ntheta) )
  for ( t1Idx in 1:(Ntheta) ) {
    for ( t2Idx in (1):Ntheta ) {
      parName1 = paste0("theta[",t1Idx,"]")
      parName2 = paste0("theta[",t2Idx,"]")
      if ( t1Idx > t2Idx) {
        # plot.new() # empty plot, advance to next
        par( mar=c(3.5,3.5,1,1) , mgp=c(2.0,0.7,0) )
        nToPlot = 700
        ptIdx = round(seq(1,chainLength,length=nToPlot))
        plot ( mcmcMat[ptIdx,parName2] , mcmcMat[ptIdx,parName1] , cex.lab=1.75 ,
               xlab=parName2 , ylab=parName1 , col="skyblue" )
      } else if ( t1Idx == t2Idx ) {
        par( mar=c(3.5,1,1,1) , mgp=c(2.0,0.7,0) )
        postInfo = plotPost( mcmcMat[,parName1] , cex.lab = 1.75 ,
                             compVal=compVal , ROPE=rope , cex.main=1.5 ,
                             xlab=parName1 , main="" )
        includeRows = ( s == t1Idx ) # identify rows of this subject in data
        dataPropor = sum(y[includeRows])/sum(includeRows)
        points( dataPropor , 0 , pch="+" , col="red" , cex=3 )
      } else if ( t1Idx < t2Idx ) {
        par( mar=c(3.5,1,1,1) , mgp=c(2.0,0.7,0) )
        postInfo = plotPost(mcmcMat[,parName1]-mcmcMat[,parName2] , cex.lab = 1.75 ,
                            compVal=compValDiff , ROPE=ropeDiff , cex.main=1.5 ,
                            xlab=paste0(parName1,"-",parName2) , main="" )
        includeRows1 = ( s == t1Idx ) # identify rows of this subject in data
        dataPropor1 = sum(y[includeRows1])/sum(includeRows1)
        includeRows2 = ( s == t2Idx ) # identify rows of this subject in data
        dataPropor2 = sum(y[includeRows2])/sum(includeRows2)
        points( dataPropor1-dataPropor2 , 0 , pch="+" , col="red" , cex=3 )
      }
    }
  }
  #-----------------------------------------------------------------------------
  if ( !is.null(saveName) ) {
    saveGraph( file=paste(saveName,"Post",sep=""), type=saveType)
  }
}

#===============================================================================
```

Although JAGS can automatically start the MCMC chains at default values, the efficiency of the MCMC process can sometimes be improved if we intelligently provide reasonable starting values to JAGS. Generally, a useful choice for initial values of the parameters is their maximum likelihood estimate (MLE).

In this question we are supposed to omit the inits argument entirely so that JAGS would create its own initial values for the chains.

```
source("Jags-Ydich-XnomSsubj-MbernBeta-Example.R")
```

```
##
## *********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## *********************************************************************
##
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 15
##    Unobserved stochastic nodes: 2
##    Total graph size: 38
##
## Initializing model
##
## Burning in the MCMC chain...
## Sampling final MCMC chain...
```

```
##                        Mean    Median      Mode   ESS HDImass      HDIlow
## theta[1]          0.6662753 0.6765425 0.7086431 50000    0.95  0.41326188
## theta[2]          0.3634623 0.3544594 0.3315970 50000    0.95  0.10890806
## theta[1]-theta[2] 0.3028130 0.3117332 0.3311916 50000    0.95 -0.07604725
##                     HDIhigh CompVal PcntGtCompVal ROPElow ROPEhigh
```

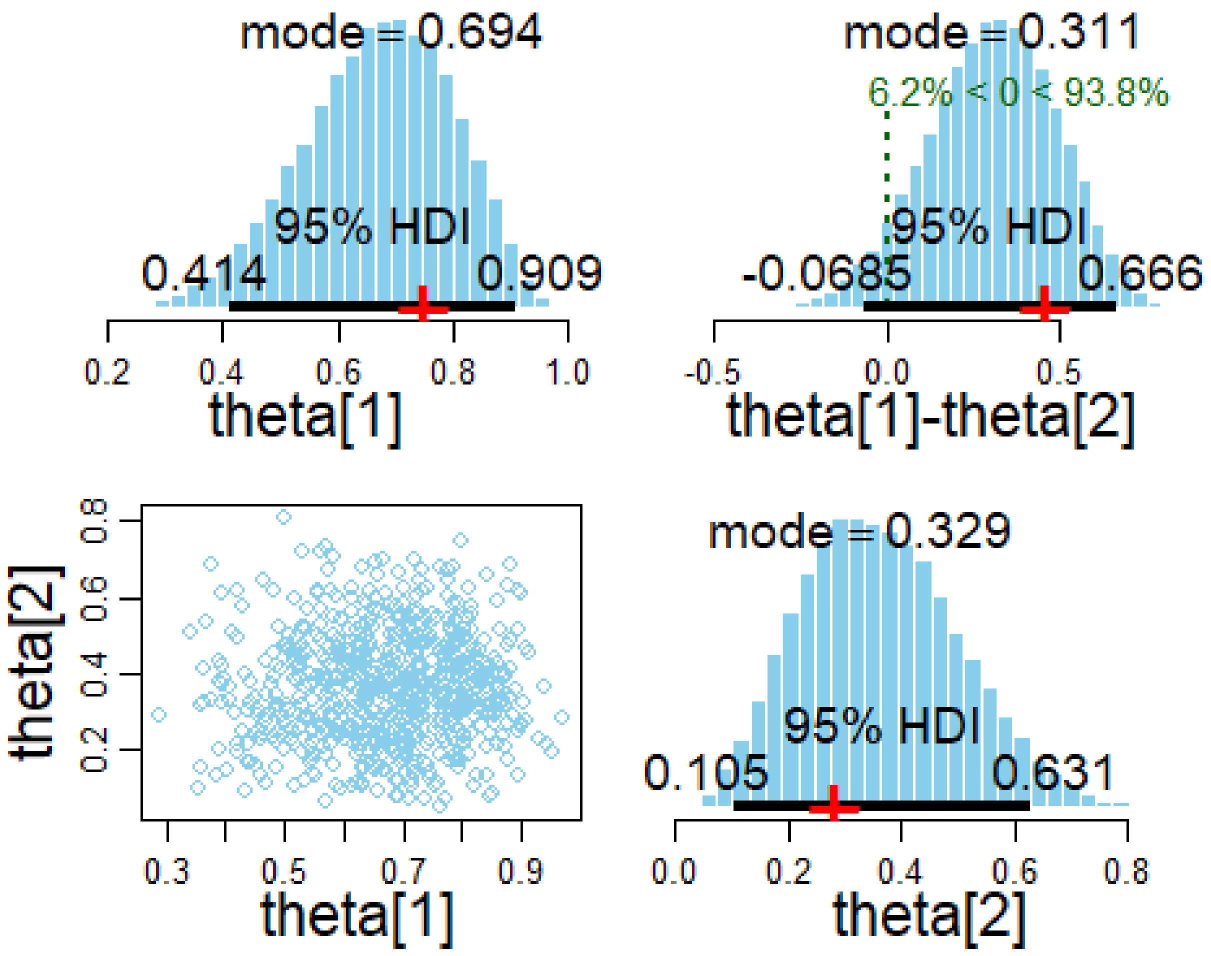
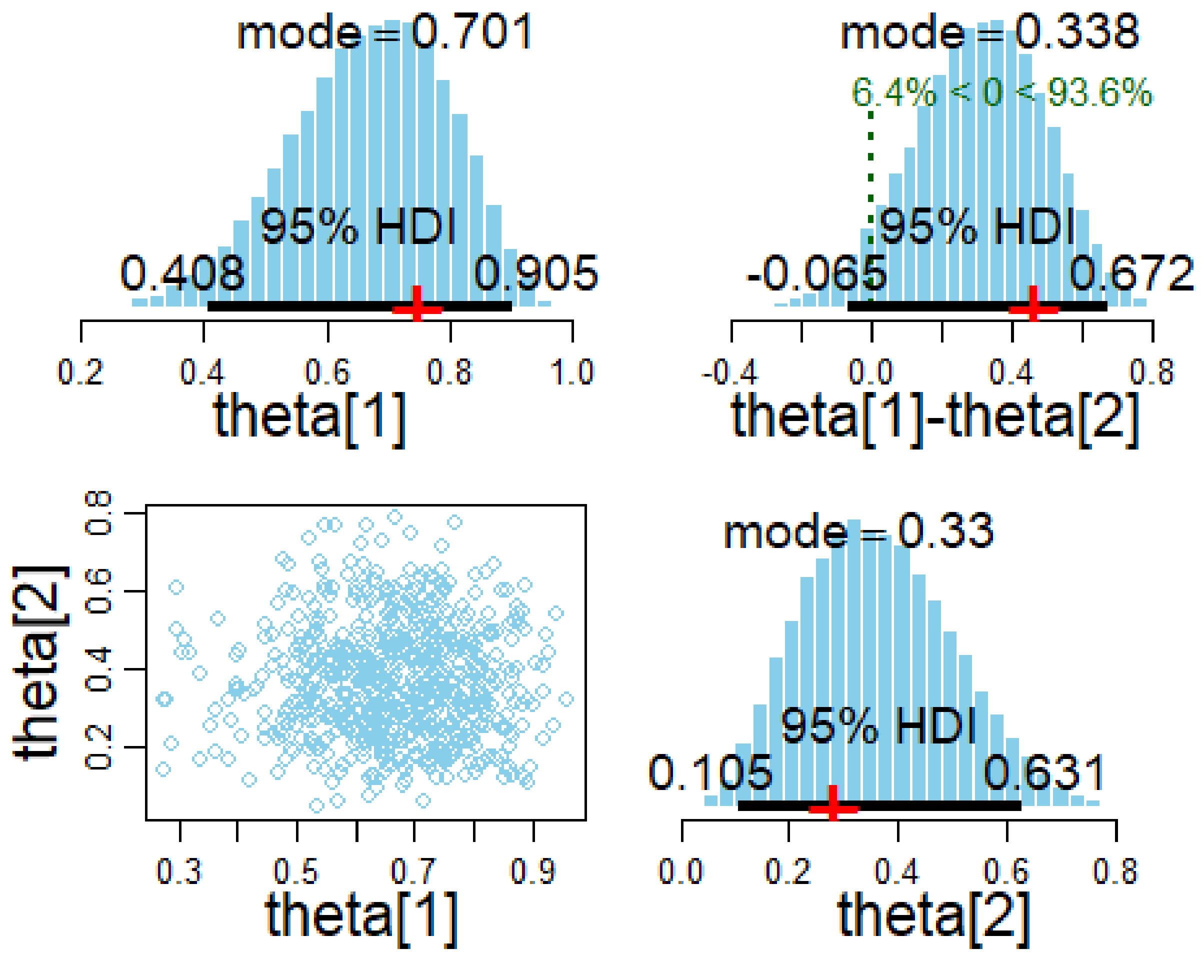


Figure 5: Without Inits

```
## theta[1]          0.9109175       NA            NA    0.45      0.55
## theta[2]          0.6353115       NA            NA    0.45      0.55
## theta[1]-theta[2] 0.6618275        0         93.56   -0.05      0.05
##                   PcntLtROPE PcntInROPE PcntGtROPE
## theta[1]               6.246     13.054     80.700
## theta[2]              73.236     16.628     10.136
## theta[1]-theta[2]      3.888      6.186     89.926
```



No, MCMC output did not change in a systematic way, other than MCMC sampling noise. In fact the HDI (95 percent) for both theta1 and theta2 are almost the same.