Homework - Chapter 9

[redacted]

[redacted]
```
getwd()
```

[redacted]
```
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 4.3.0
```

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

Ex 9.1

A) The shape parameter is 0.01 and rate is 0.01 for the gamma distribution when mean is 1 and sd is 10

```
gammaShRaFromMeanSD(mean = 1, sd = 10)
```

```
## $shape
## [1] 0.01
##
## $rate
## [1] 0.01
```

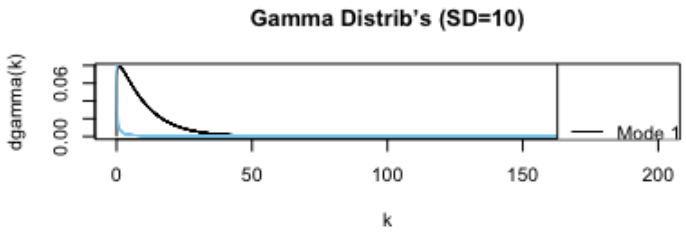
The shape parameter is 1.105125 and rate is 0.1051249 for the gamma distribution when mode is 1 and sd is 10

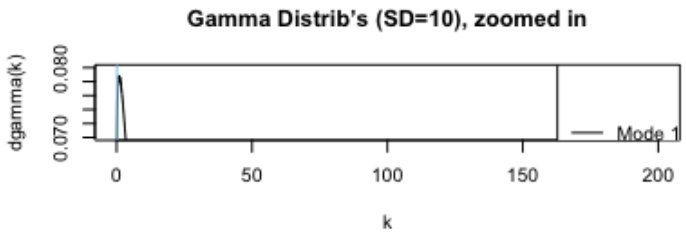
```
gammaShRaFromModeSD(mode = 1, sd = 10)
```

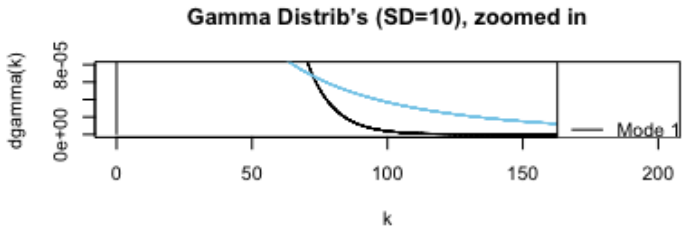
```
## $shape
## [1] 1.105125
##
## $rate
## [1] 0.1051249
```

B) Relative to each other, the gamma distribution with mode = 1 favors values of kappa between about 0.1 and 75 (it peaks near .1, in the range of .1 and 75). The gamma distribution with mean = 1 favors values of kappa that are tiny or greater than 75 (it has higher density less than .1 and greater than 75 as shown in the 2nd and 3rd graphs).

```
### R code provided in the textbook to plot the graphs
# openGraph(height=7,width=7)
layout(matrix(1:3,ncol=1))
k=seq(0,200,length=10001)
plot( k , dgamma(k,1.105125,0.105125) , ylab="dgamma(k)" ,type="l" , main="Gamma Distrib's (SD=10)" )
lines( k , dgamma(k,0.01,0.01) , col="skyblue" )
legend( "topright" , c("Mode 1","Mean 1") ,lty=c(1,1) , col=c("black","skyblue") , text.col=c("black", "skyblue
plot( k , dgamma(k,1.105125,0.105125) , ylab="dgamma(k)" , type="l" , ylim=c(.07,.08) , main="Gamma Distrib's (
lines( k , dgamma(k,0.01,0.01) , col="skyblue" )
legend( "topright" , c("Mode 1","Mean 1"), lty=c(1,1) , col=c("black","skyblue") , text.col=c("black", "skyblue
plot( k , dgamma(k,1.105125,0.105125) , ylab="dgamma(k)" ,type="l" , ylim=c(0,8.0e-5) , main="Gamma Distrib's (
lines( k , dgamma(k,0.01,0.01) , col="skyblue" )
legend( "topright" , c("Mode 1","Mean 1") ,lty=c(1,1) , col=c("black","skyblue") , text.col=c("black", "skyblue
```



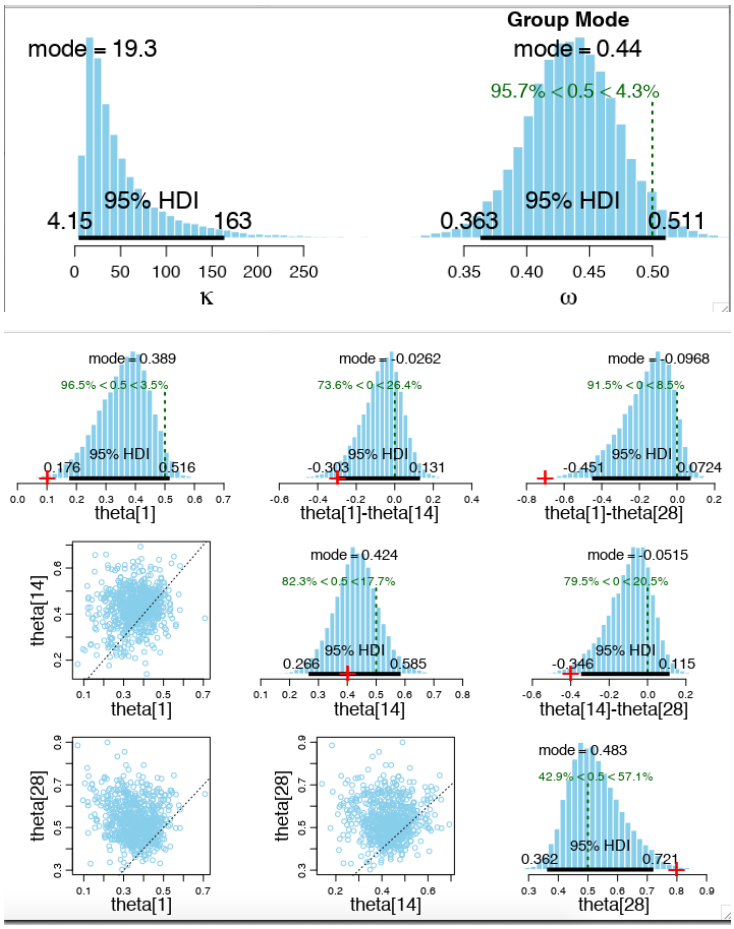




C) using a gamma with a mean of 1.0:

```
setwd("/Users/[redacted]Bayesian Data Analysis/HW")
graphics.off() # This closes all of R's graphics windows.
# Load the relevant model into R's working memory:
source("Jags-Ydich-XnomSsubj-MbinomBetaOmegaKappa-HW8-Mean1.R") # mean of 1
# Read the data
myData = read.csv("TherapeuticTouchData.csv")
# Optional: Specify filename root and graphical format for saving output.
# Otherwise specify as NULL or leave saveName and saveType arguments
# out of function calls.
fileNameRoot = "Jags-Ydich-XnomSsubj-MbinomBetaOmegaKappa-HW8-C-"
graphFileType = "eps"
#------------------------------------------------------------------------------
# Generate the MCMC chain:
startTime = proc.time()
mcmcCoda = genMCMC( data=myData , sName="s" , yName="y" ,
                    numSavedSteps=10000 , saveName=fileNameRoot ,
                    thinSteps=20 )
stopTime = proc.time()
show( stopTime-startTime )
#------------------------------------------------------------------------------
# Display diagnostics of chain, for specified parameters:
parameterNames = varnames(mcmcCoda) # get all parameter names for reference
for ( parName in parameterNames[c(1:3,length(parameterNames))] ) {
  diagMCMC( codaObject=mcmcCoda , parName=parName ,
            saveName=fileNameRoot , saveType=graphFileType )
}
#------------------------------------------------------------------------------
# Get summary statistics of chain:
summaryInfo = smryMCMC( mcmcCoda , compVal=0.5 ,
                        diffIdVec=c(1,14,28), compValDiff=0.0,
                        saveName=fileNameRoot )
```

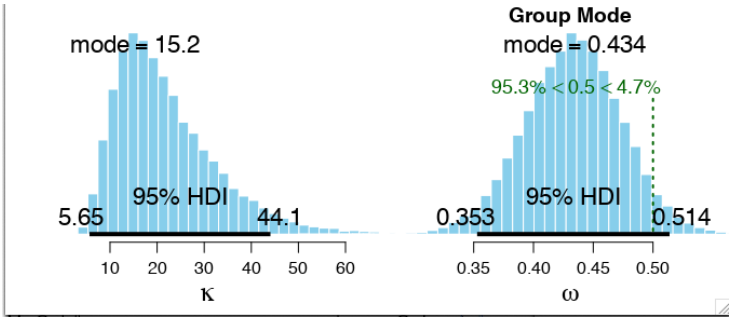
```
# Display posterior information:
plotMCMC( mcmcCoda , data=myData , sName="s" , yName="y" ,
          compVal=0.5 , #rope=c(0.45,0.55) ,
          diffIdVec=c(1,14,28), compValDiff=0.0, #ropeDiff = c(-0.05,0.05) ,
          saveName=fileNameRoot , saveType=graphFileType )
```

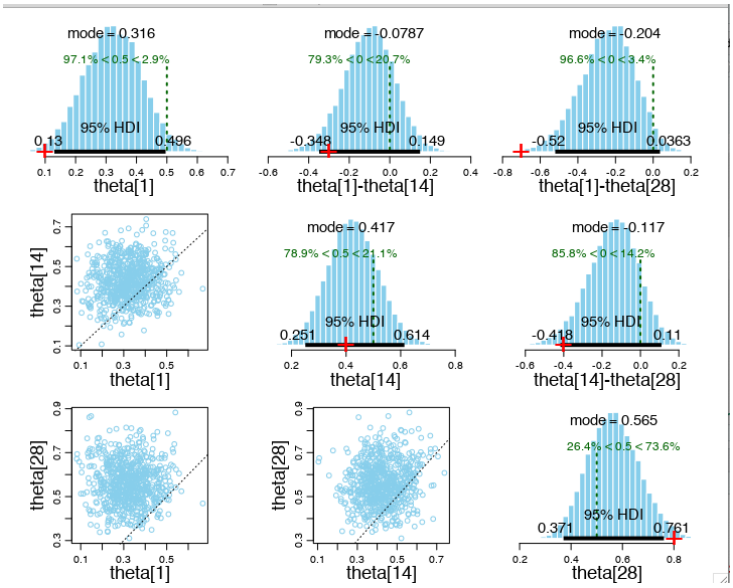


using a gamma with a mode of 1.0:

```
setwd("/Users/[redacted]Bayesian Data Analysis/HW")
source("Jags-Ydich-XnomSsubj-MbinomBetaOmegaKappa-HW8-Mode1.R") # mode of 1
# Read the data
myData = read.csv("TherapeuticTouchData.csv")
# Optional: Specify filename root and graphical format for saving output.
# Otherwise specify as NULL or leave saveName and saveType arguments
# out of function calls.
fileNameRoot = "Jags-Ydich-XnomSsubj-MbinomBetaOmegaKappa-HW8-C-Mode1-"
graphFileType = "eps"
#------------------------------------------------------------------------------
# Generate the MCMC chain:
startTime = proc.time()
mcmcCoda = genMCMC( data=myData , sName="s" , yName="y" ,
                    numSavedSteps=20000 , saveName=fileNameRoot ,
                    thinSteps=10 )
stopTime = proc.time()
show( stopTime-startTime )
#------------------------------------------------------------------------------
# Display diagnostics of chain, for specified parameters:
parameterNames = varnames(mcmcCoda) # get all parameter names for reference
for ( parName in parameterNames[c(1:3,length(parameterNames))] ) {
  diagMCMC( codaObject=mcmcCoda , parName=parName ,
            saveName=fileNameRoot , saveType=graphFileType )
```

```
}
#-------------------------------------------------------------------------------
# Get summary statistics of chain:
summaryInfo = smryMCMC( mcmcCoda , compVal=0.5 ,
                        diffIdVec=c(1,14,28), compValDiff=0.0,
                        saveName=fileNameRoot )
# Display posterior information:
plotMCMC( mcmcCoda , data=myData , sName="s" , yName="y" ,
          compVal=0.5 , #rope=c(0.45,0.55) ,
          diffIdVec=c(1,14,28), compValDiff=0.0, #ropeDiff = c(-0.05,0.05) ,
          saveName=fileNameRoot , saveType=graphFileType )
```





D) The posterior on kappa has a bigger large-value tail when the gamma distribution has a mean of 1 for kappa prior.

When kappa is larger, theta values have more shrinkage (more reduced variance in the estimators). Like when the gamma distribution has a mean of 1, the differences between theta1 and theta28, and between theta14 and theta28 are smaller, compared to when the gamma distribution prior has a mode of 1.

E) We'll probably want the prior has a mean of 1 (we get more variance and larger shrinkage).

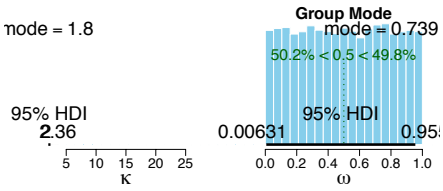
Ex 9.2

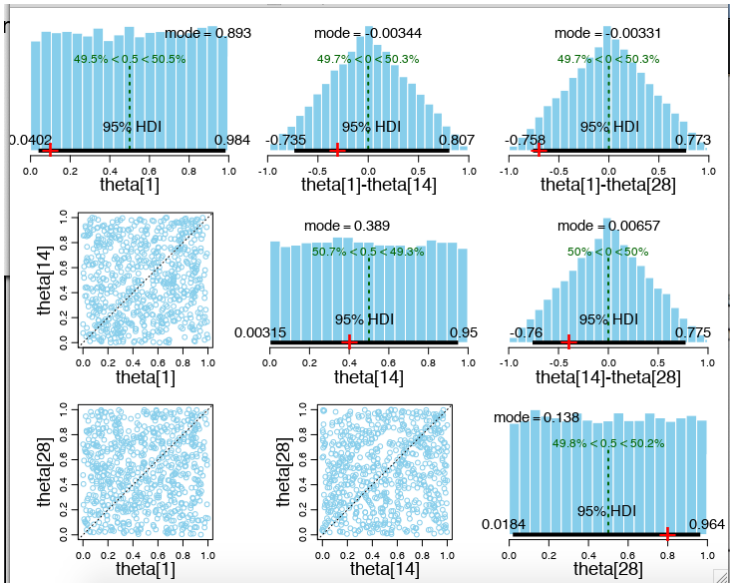
A) As shown in the graphs below, when the prior on kappa has a mean of 1, kappa gets extremely skewed to the right and has higher density of being extremely small and thus thetas can have uniform shapes. When the prior on kappa has a mode of 1, kappa doesn't get extremely small and thetas won't have uniform distributions.

When using a gamma with a mean of 1.0:

```
setwd("/Users/[redacted]Bayesian Data Analysis/HW")
graphics.off() # This closes all of R's graphics windows.
# Load the relevant model into R's working memory:
source("Jags-Ydich-XnomSsubj-MbinomBetaOmegaKappa-HW8-9.2-Mean1.R") # mean of 1
```

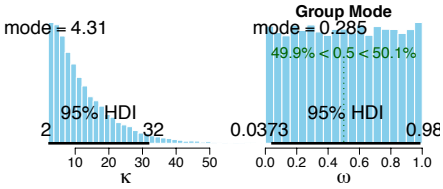
```r
# Read the data
myData = read.csv("TherapeuticTouchData.csv")
# Optional: Specify filename root and graphical format for saving output.
# Otherwise specify as NULL or leave saveName and saveType arguments
# out of function calls.
fileNameRoot = "Jags-Ydich-XnomSsubj-MbinomBetaOmegaKappa-HW8-C-"
graphFileType = "eps"
#------------------------------------------------------------------------------
# Generate the MCMC chain:
startTime = proc.time()
mcmcCoda = genMCMC( data=myData , sName="s" , yName="y" ,
                    numSavedSteps=10000 , saveName=fileNameRoot ,
                    thinSteps=20 )
stopTime = proc.time()
show( stopTime-startTime )
#------------------------------------------------------------------------------
# Display diagnostics of chain, for specified parameters:
parameterNames = varnames(mcmcCoda) # get all parameter names for reference
for ( parName in parameterNames[c(1:3,length(parameterNames))] ) {
  diagMCMC( codaObject=mcmcCoda , parName=parName ,
            saveName=fileNameRoot , saveType=graphFileType )
}
#------------------------------------------------------------------------------
# Get summary statistics of chain:
summaryInfo = smryMCMC( mcmcCoda , compVal=0.5 ,
                        diffIdVec=c(1,14,28), compValDiff=0.0,
                        saveName=fileNameRoot )
# Display posterior information:
plotMCMC( mcmcCoda , data=myData , sName="s" , yName="y" ,
          compVal=0.5 , #rope=c(0.45,0.55) ,
          diffIdVec=c(1,14,28), compValDiff=0.0, #ropeDiff = c(-0.05,0.05) ,
          saveName=fileNameRoot , saveType=graphFileType )
```

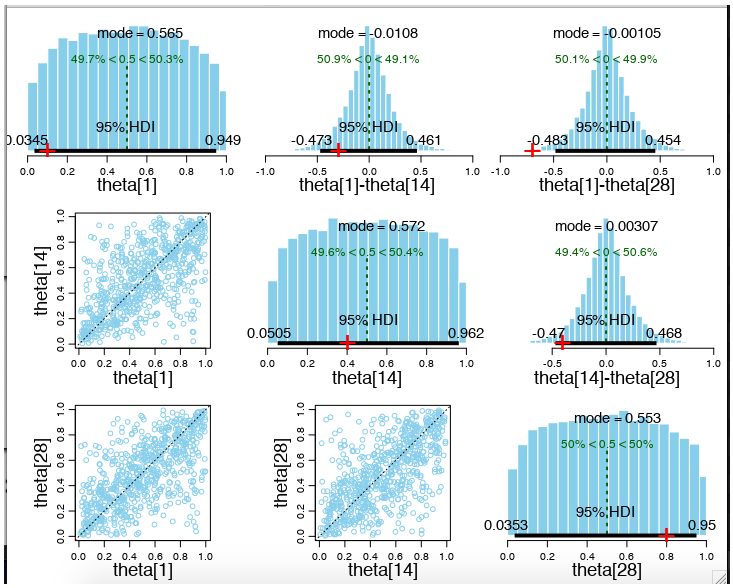




When using a gamma with a mode of 1.0:

```
setwd("/Users/[redacted]Bayesian Data Analysis/HW")
source("Jags-Ydich-XnomSsubj-MbinomBetaOmegaKappa-HW8-Mode1.R") # mean of 1
# Read the data
myData = read.csv("TherapeuticTouchData.csv")
# Optional: Specify filename root and graphical format for saving output.
# Otherwise specify as NULL or leave saveName and saveType arguments
# out of function calls.
fileNameRoot = "Jags-Ydich-XnomSsubj-MbinomBetaOmegaKappa-HW8-C-Mode1-"
graphFileType = "eps"
#------------------------------------------------------------------------------
# Generate the MCMC chain:
startTime = proc.time()
mcmcCoda = genMCMC( data=myData , sName="s" , yName="y" ,
                    numSavedSteps=20000 , saveName=fileNameRoot ,
                    thinSteps=10 )
stopTime = proc.time()
show( stopTime-startTime )
#------------------------------------------------------------------------------
# Display diagnostics of chain, for specified parameters:
parameterNames = varnames(mcmcCoda) # get all parameter names for reference
for ( parName in parameterNames[c(1:3,length(parameterNames))] ) {
  diagMCMC( codaObject=mcmcCoda , parName=parName ,
            saveName=fileNameRoot , saveType=graphFileType )
}
#------------------------------------------------------------------------------
# Get summary statistics of chain:
summaryInfo = smryMCMC( mcmcCoda , compVal=0.5 ,
                        diffIdVec=c(1,14,28), compValDiff=0.0,
                        saveName=fileNameRoot )
# Display posterior information:
plotMCMC( mcmcCoda , data=myData , sName="s" , yName="y" ,
          compVal=0.5 , #rope=c(0.45,0.55) ,
          diffIdVec=c(1,14,28), compValDiff=0.0, #ropeDiff = c(-0.05,0.05) ,
          saveName=fileNameRoot , saveType=graphFileType )
```





B) We probably want the kappa priors with a mean of 1 so we'll have uniform priors on thetas because we are uncertain and want to be noncommittal for theta priors.

Ex 9.4

A) To run in parallel, I modified the following section in Jags-Ydich-XnomSsubj-MbinomBetaOmegaKappa.R and saved it as Jags-Ydich-XnomSsubj-MbinomBetaOmegaKappa-parallel.R to use later.

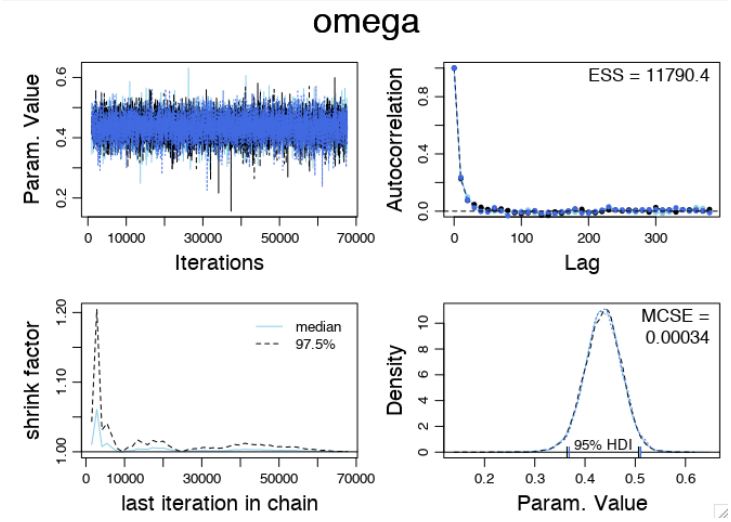
```
nChains = 3
parameters = c( "theta","omega","kappa") # The parameters to be monitored
adaptSteps = 500              # Number of steps to adapt the samplers
burnInSteps = 500             # Number of steps to burn-in the chains
useRunjags = TRUE
if ( useRunjags ) {
  runJagsOut <- run.jags( method=c("parallel") ,
                          model="TEMPmodel.txt" ,
                          monitor=parameters ,
                          data=dataList ,
                          inits=initsList ,
                          n.chains=nChains ,
                          adapt=adaptSteps ,
                          burnin=burnInSteps ,
                          sample=ceiling(numSavedSteps/nChains) ,
```

```
                            thin=thinSteps ,
                            summarise=FALSE ,
                            plots=FALSE )
    codaSamples = as.mcmc.list( runJagsOut )
  }
```

The running time was 38.45 seconds when the chains were run in parallel.The diagnostic plot shows ESS=11790.4

```
myData = read.csv("TherapeuticTouchData.csv")
source("Jags-Ydich-XnomSsubj-MbinomBetaOmegaKappa-parallel.R")
startTime = proc.time()
mcmcCoda = genMCMC( data=myData , sName="s" , yName="y" ,
                    numSavedSteps=20000,
                    thinSteps=10)
stopTime = proc.time()
show( stopTime-startTime )
diagMCMC( codaObject=mcmcCoda , parName="omega" )
```

```
> show( stopTime-startTime )
   user  system elapsed
  3.648   1.389  38.450
```



B) run the chains serially, the following code was modified.

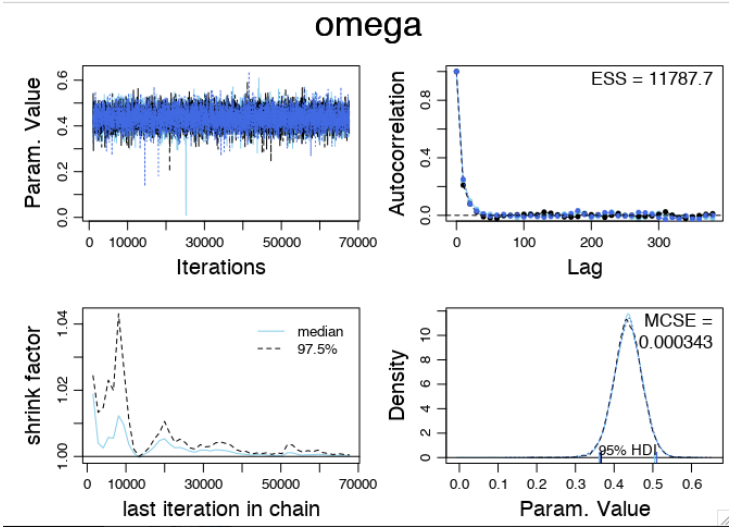
```
  if ( useRunjags ) {
    runJagsOut <- run.jags( method=c("rjags") ,
                            model="TEMPmodel.txt" ,
                            monitor=parameters ,
                            data=dataList ,
                            inits=initsList ,
                            n.chains=nChains ,
                            adapt=adaptSteps ,
                            burnin=burnInSteps ,
                            sample=ceiling(numSavedSteps/nChains) ,
                            thin=thinSteps ,
                            summarise=FALSE ,
                            plots=FALSE )
    codaSamples = as.mcmc.list( runJagsOut )
  }
```

The running time was 70.583 seconds when the chains were run in parallel.The diagnostic plot shows ESS=11787.7, slightly smaller than the ESS for parallel chains (basically the same).

```
myData = read.csv("TherapeuticTouchData.csv")
source("Jags-Ydich-XnomSsubj-MbinomBetaOmegaKappa-rjags.R")
```

```
startTime = proc.time()
mcmcCoda.serial = genMCMC( data=myData , sName="s" , yName="y" ,
                    numSavedSteps=20000,
                    thinSteps=10)
stopTime = proc.time()
show( stopTime-startTime )
diagMCMC( codaObject=mcmcCoda.serial , parName="omega" )
```

```
> show( stopTime-startTime )
   user  system elapsed
 41.741   2.111  70.583
```



C) The parallel chains took much less time to run, almost half of the time of serial chains. The ESSs indicate that the quality was the same. Definitely run the chains in parallel if you can.