Assignment4

Abhishek kumar Gupta

April 27, 2019

Contents

All questions have been answered

1 Soft drinks delivefy times:

a) What is the mathematical model used here in terms of the GLM? Write out the mathematics using Latex

Ans: The model in genreal form used here is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

or, in particular:

$$deliveryTime = \beta_0 + \beta_1 * cases + \beta_2 * distance$$

i. What is the Link in this case?

The link is th linear function given as:

$$link = \beta_1 * x_1 + \beta_2 * x_2$$

ii. What is the linear predictor in this case?

predictor variable: cases and distance predicted variable: delivery time

iii. What is the distribution of the error in this case?

Normal distribution: ~ dnorm(mu[i], tau)

b) Plot the data as boxplots using ggplot (you will need to reformat the data as a data frame, data.frame())

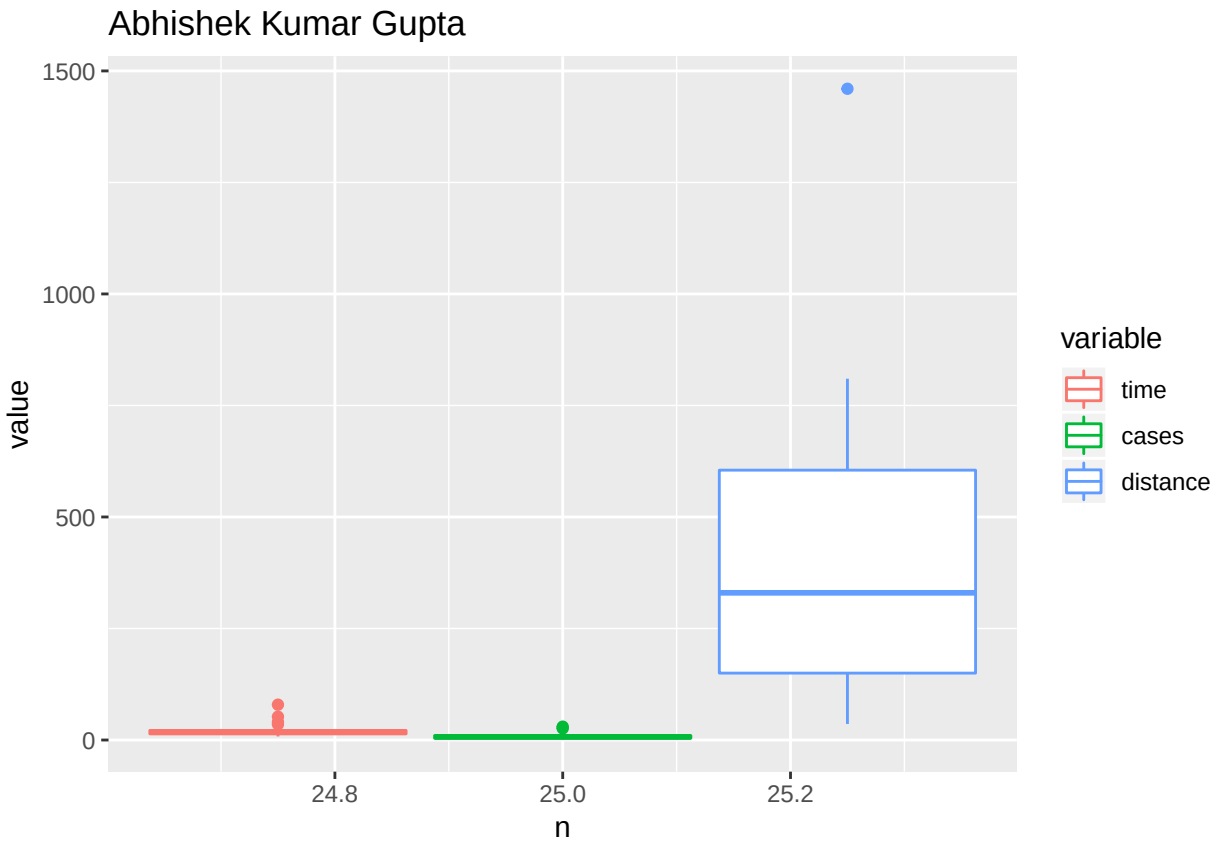
```
df = data.frame(list( n=25, time = c(16.68, 11.5, 12.03, 14.88, 13.75, 18.11, 8, 17.83, 79.24, 21.5,
                     40.33, 21, 13.5, 19.75, 24, 29, 15.35, 19, 9.5, 35.1, 17.9, 52.32,
                     18.75, 19.83, 10.75), distance = c(560, 220, 340, 80, 150, 330, 110,
                    210, 1460, 605, 688, 215, 255, 462, 448, 776, 200, 132, 36, 770, 140,
                    810, 450, 635, 150), cases = c( 7, 3, 3, 4, 6, 7, 2, 7, 30, 5, 16, 10,
                                                    4, 6, 9, 10, 6, 7, 3, 17, 10, 26, 9, 8, 4) ))
head(df)
```

```
##    n  time distance cases
## 1 25 16.68      560     7
## 2 25 11.50      220     3
## 3 25 12.03      340     3
## 4 25 14.88       80     4
## 5 25 13.75      150     6
## 6 25 18.11      330     7
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```
library(reshape2)
data.m = melt(df,id.vars = 'n', measure.vars = c('time', 'cases', 'distance'))
p <- ggplot(data.m) +
  geom_boxplot(aes(x=n, y=value, color=variable))+labs(title ="Abhishek Kumar Gupta")
p
```



c) Make density plots of all stochastic nodes using the ggmcmc package. You should run the model for 10000 iterations and throw away the first 1000 as a burn in. Use three chains.

```
require(rjags)# Must have previously installed package rjags.
```

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

```
## Warning: package 'rjags' was built under R version 3.5.3
```

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

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

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

```
fileNameRoot="Assignment4" # For output file names.
```

```
Ntotal = 25  # Compute the total number of x,y pairs.

dataList =  list( time = c(16.68, 11.5, 12.03, 14.88, 13.75, 18.11, 8, 17.83, 79.24, 21.5, 40.33, 21, 13



#Define the model:
modelString = "
model{
    for( i in 1 : Ntotal ) {
        time[i] ~ dnorm(mu[i], tau)
        mu[i] <- beta0 + beta1 * cases[i] + beta2*distance[i]
    }
    beta0 ~ dnorm(0.0, 1.0E-4)
    beta1 ~ dnorm(0.0, 1.0E-4)
    beta2 ~ dnorm(0.0, 1.0E-4)
    tau ~ dgamma(0.01, 0.01)
}

" # close quote for modelString
writeLines( modelString , con="TEMPmodel.txt" )
initsList = list(beta0 = 0, beta1 = 0, beta2 =0, tau=1)

# Run the chains:
jagsModel = jags.model( file="TEMPmodel.txt" , data=dataList , inits=initsList ,
                        n.chains=3 , n.adapt=1000 )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 25
##    Unobserved stochastic nodes: 4
##    Total graph size: 145
##
## Initializing model
```

```
update( jagsModel , n.iter=1000 )
codaSamples = coda.samples( jagsModel , variable.names=c("beta0", "beta1", "beta2") ,
                            n.iter=10000)
save( codaSamples , file=paste0(fileNameRoot,"Mcmc.Rdata") )



source("DBDA2E-utilities.R")
```

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

```
library(ggmcmc)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

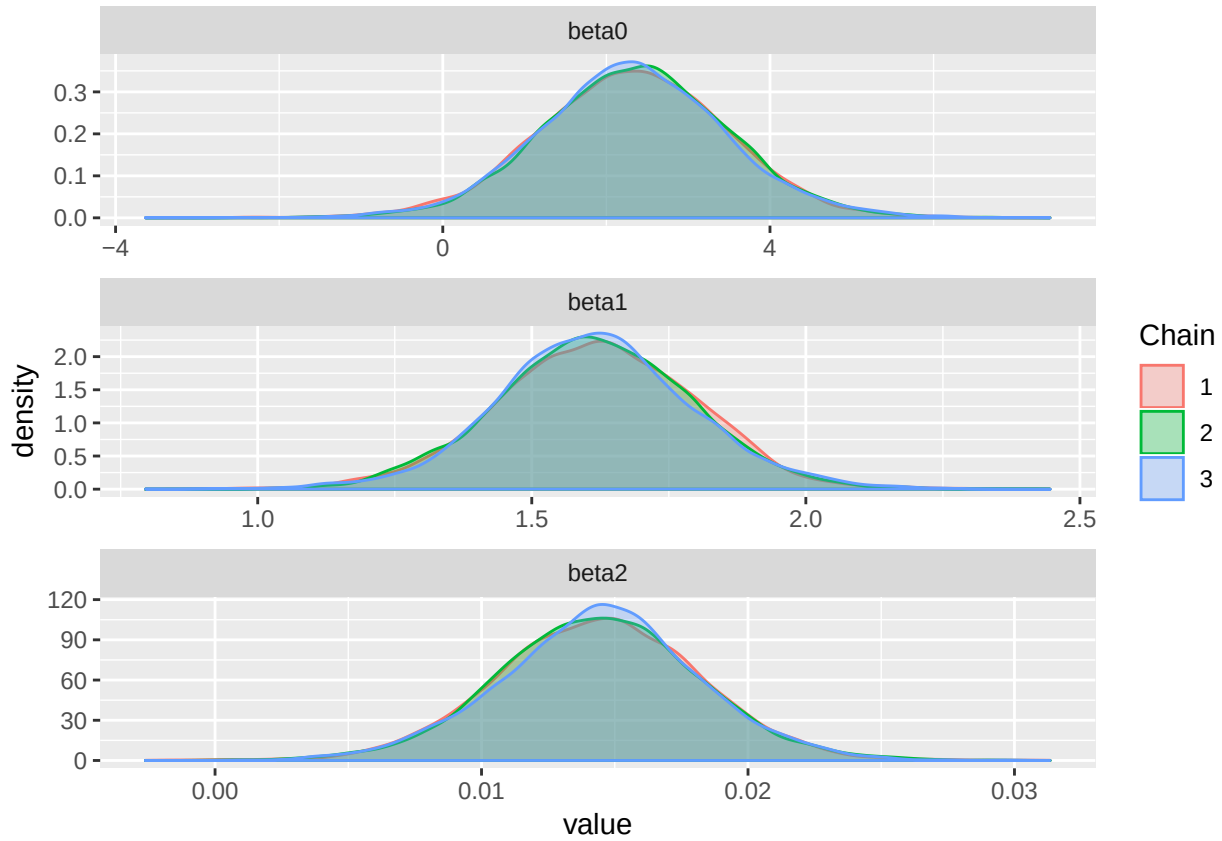
```
## Loading required package: tidyr
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:runjags':
##
##     extract
```

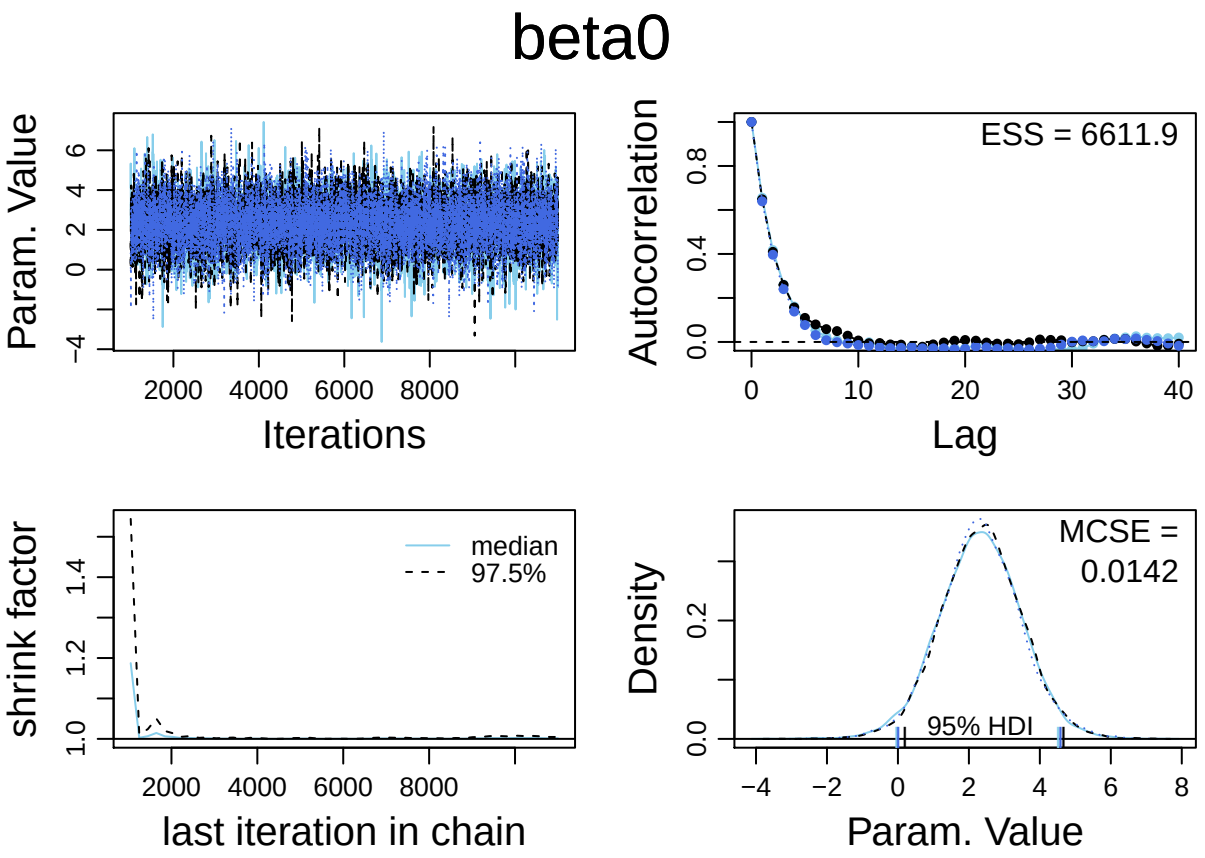
```
## The following object is masked from 'package:reshape2':
##
##     smiths
```

```
s = ggs(codaSamples)
d=ggs_density(s)
print(d)
```

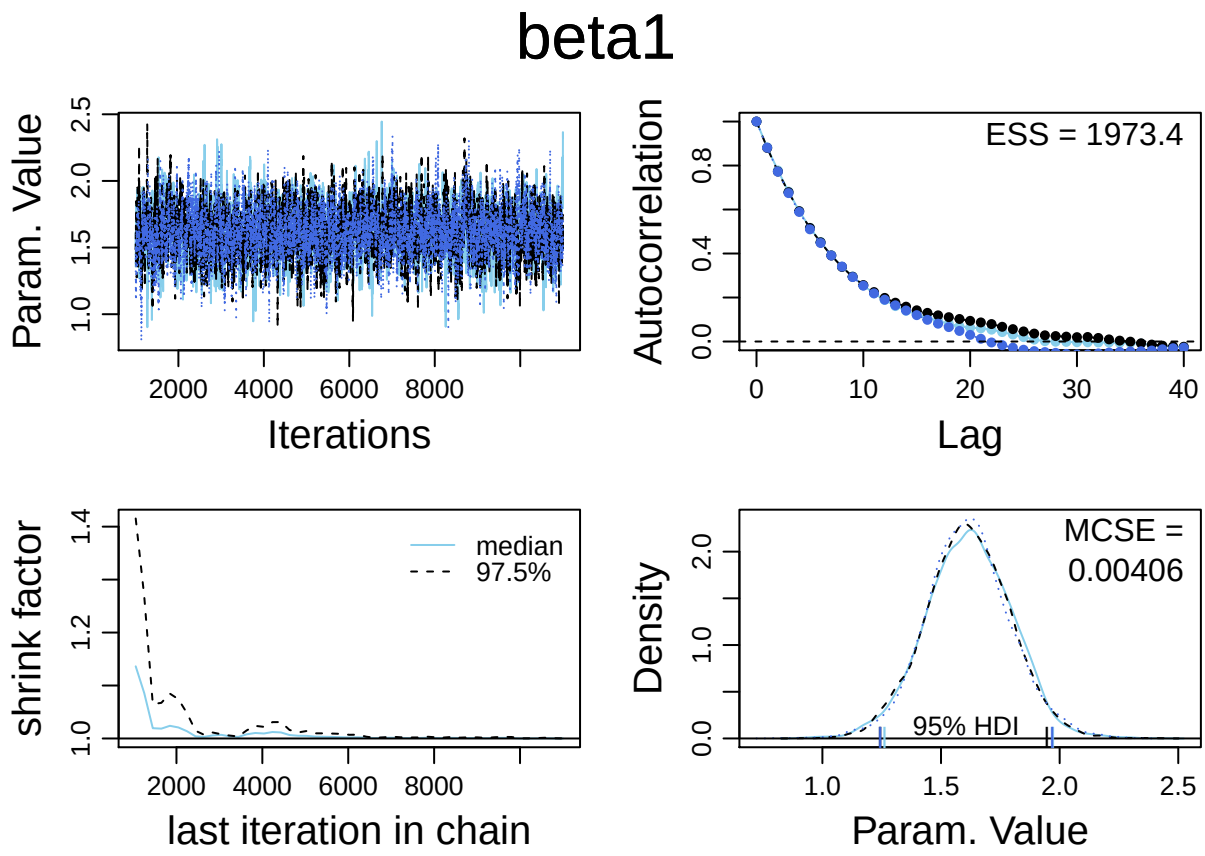


d) Give diagnostic plots of the MCMC for all primary stochastic nodes.

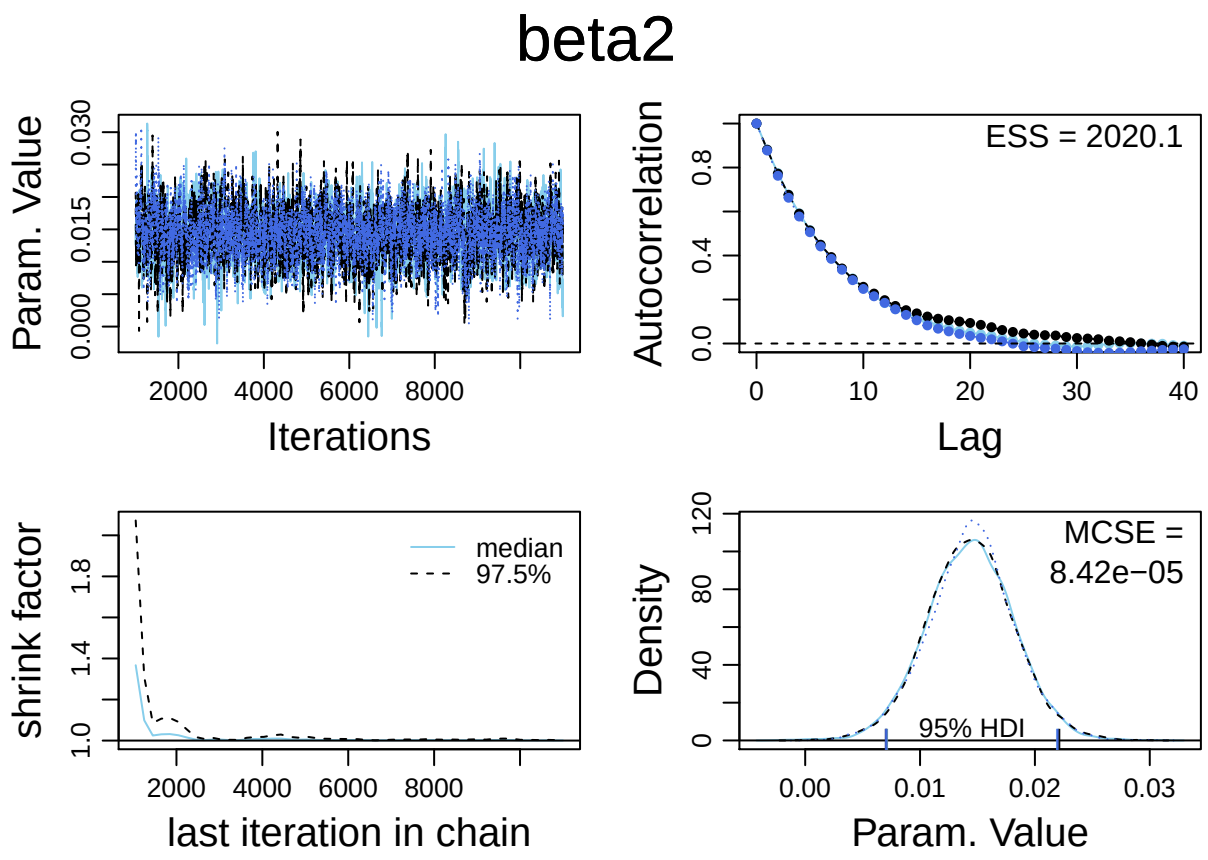
```
diagMCMC( codaObject=codaSamples , parName="beta0" )
saveGraph( file=paste0(fileNameRoot,"m") , type="jpeg" )
```




```
diagMCMC( codaObject=codaSamples , parName="beta1" )
saveGraph( file=paste0(fileNameRoot,"m") , type="jpeg" )
```

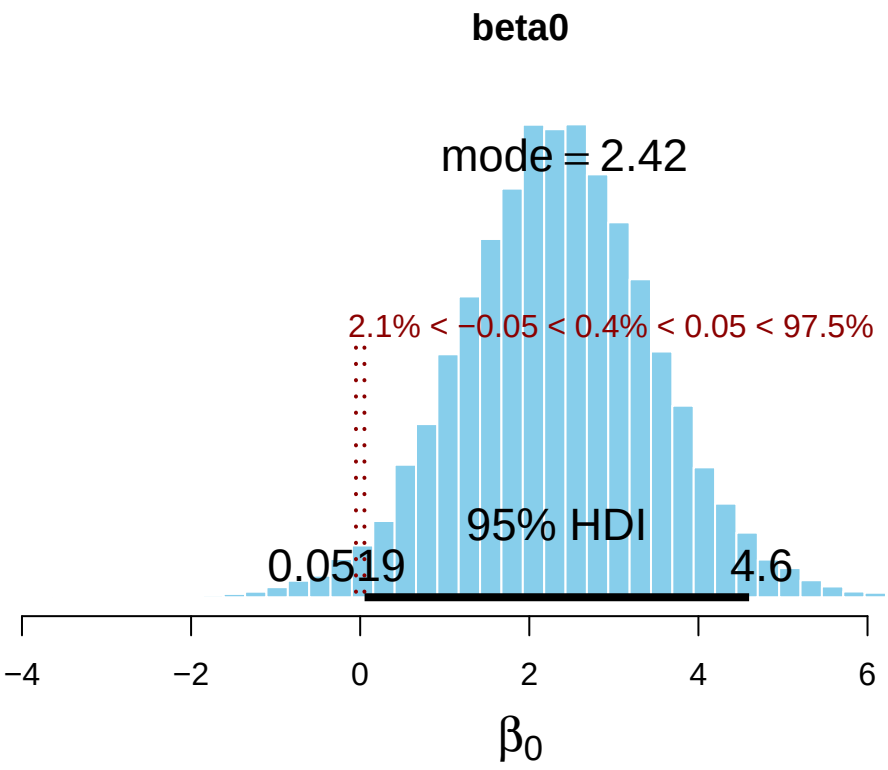



```
diagMCMC( codaObject=codaSamples , parName="beta2" )
saveGraph( file=paste0(fileNameRoot,"m") , type="jpeg" )
```



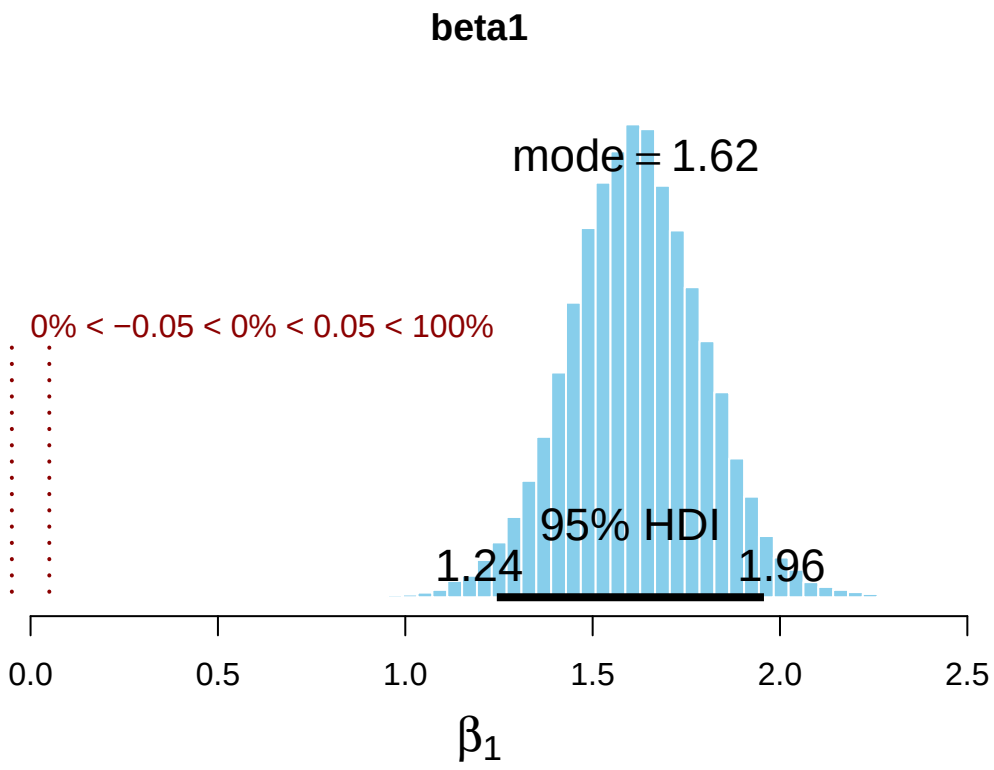
e) Make posterior histograms of beta0, beta1 and beta2 using JK's code with a suitable rope around 0.

```
plotPost(codaSamples[,"beta0"], main = "beta0", xlab = bquote(beta[0]), ROPE = c(-0.05, 0.05))
```



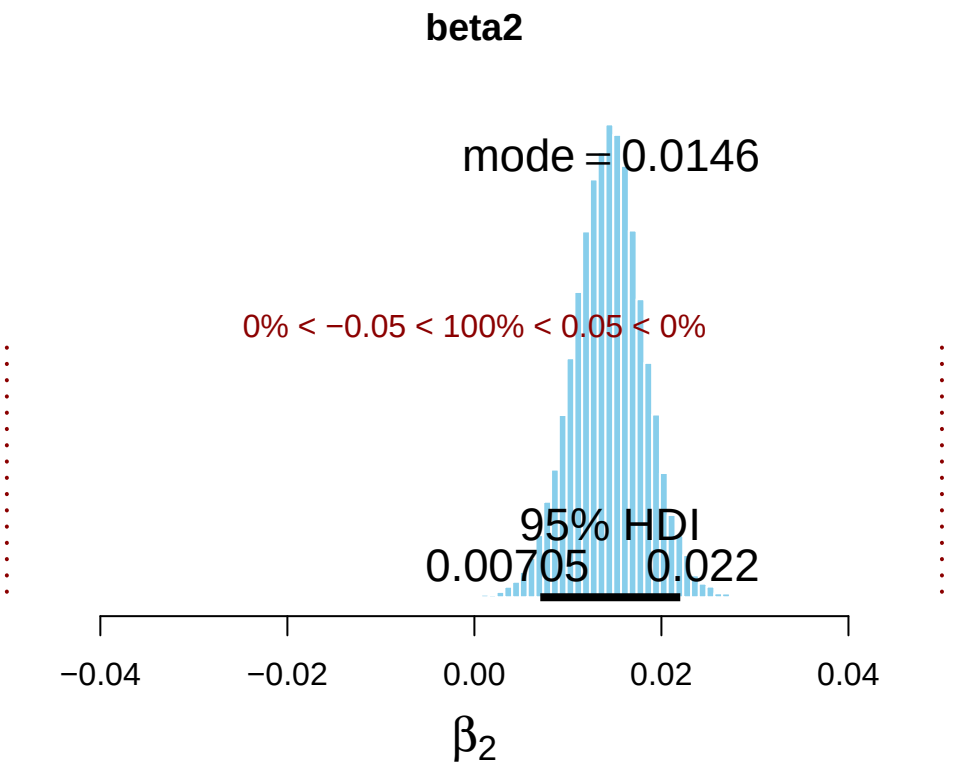

```
##              ESS     mean   median     mode hdiMass     hdiLow  hdiHigh
## beta[0] 6665.733 2.330634 2.334205 2.415462    0.95 0.05194044 4.599593
##         compVal pGtCompVal ROPElow ROPEhigh    pLtROPE pInROPE   pGtROPE
## beta[0]      NA         NA   -0.05     0.05 0.02106667  0.0037 0.9752333
```

```
plotPost(codaSamples[,"beta1"], main = "beta1", xlab = bquote(beta[1]), ROPE = c(-0.05, 0.05))
```




```
##              ESS     mean   median     mode hdiMass   hdiLow  hdiHigh
## beta[1] 2032.283 1.615542 1.614688 1.615165    0.95 1.244262 1.957202
##         compVal pGtCompVal ROPElow ROPEhigh pLtROPE pInROPE pGtROPE
## beta[1]      NA         NA   -0.05     0.05       0       0       1
```

```
plotPost(codaSamples[,"beta2"], main = "beta2", xlab = bquote(beta[2]), ROPE = c(-0.05, 0.05))
```




```
##              ESS       mean     median       mode hdiMass     hdiLow
## beta[2] 1995.169 0.01440592 0.01443907 0.01461173    0.95 0.00704729
##            hdiHigh compVal pGtCompVal ROPElow ROPEhigh pLtROPE pInROPE
## beta[2] 0.02201563      NA         NA   -0.05     0.05       0       1
##         pGtROPE
## beta[2]       0
```

f) Give a summary of all the posterior parameter estimates.

```
summary(codaSamples)
```

```
##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean       SD  Naive SE Time-series SE
## beta0 2.33063 1.151736 6.650e-03      1.417e-02
## beta1 1.61554 0.180578 1.043e-03      4.065e-03
## beta2 0.01441 0.003786 2.186e-05      8.427e-05
##
## 2. Quantiles for each variable:
```

```
##
##           2.5%     25%     50%     75%   97.5%
## beta0 0.053672 1.58403 2.33421 3.08339 4.60344
## beta1 1.255480 1.49751 1.61469 1.73339 1.97139
## beta2 0.006916 0.01193 0.01444 0.01687 0.02189
```

g) Give all Bayesian point estimates of parameters.

```
su = summary(codaSamples)
su$statistics
```

```
##             Mean         SD     Naive SE Time-series SE
## beta0 2.33063385 1.15173620 6.649552e-03   1.416909e-02
## beta1 1.61554195 0.18057849 1.042570e-03   4.065155e-03
## beta2 0.01440592 0.00378587 2.185773e-05   8.426682e-05
```

h) Write down the formula (using Latex) for the mean service (or delivery) time as predicted by the model .

The required equation is given as:

$$deliveryTime = 2.35 + 1.61 * cases + 0.01 * distance$$

i) Using the above expression for the mean service time and summary information from the posterior find:

i. For each additional case stocked by the employee

a. how much delivery time will be required on average (point estimate)?

Ans: 1.61 mins

b. how much delivery time will be required on average (interval estimate) and with what posterior probability?

Ans: 1.26-1.96 with 95% probability

ii. For every increase of walking distance by 100 feet

a. what delivery time will be required on average (point estimate)?

ANs: 1.43 mins

b. what delivery time will be required on average (interval estimate) and with what posterior probability?

Ans: 0.7-2.17 with 95% probability

j) The engineer wished to find a typical or representative delivery route. He suggested the following code chunk. Complete the code by supplying the missing function

typical.y <- beta0 + beta1*mean(cases[]) + beta2*mean(distance[])

k) Add the typical.y to the JAGS model and re-run this time including typical.y as a monitored node. Give a full interpretation of its posterior distribution

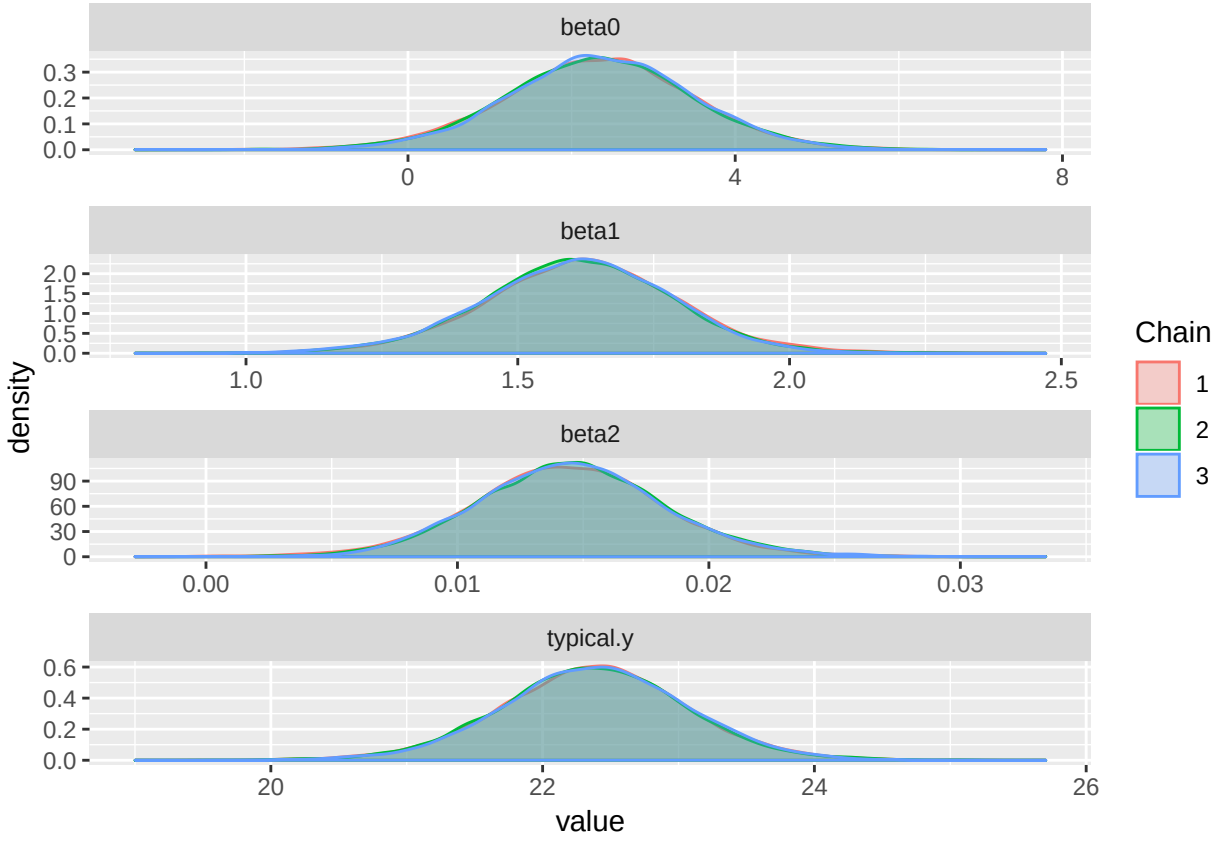
```
require(rjags)# Must have previously installed package rjags.

fileNameRoot="Assignment4" # For output file names.

Ntotal = 25  # Compute the total number of x,y pairs.

dataList =  list( time = c(16.68, 11.5, 12.03, 14.88, 13.75, 18.11, 8, 17.83, 79.24, 21.5, 40.33, 21, 13



#Define the model:
modelString = "
model{
    for( i in 1 : Ntotal ) {
        time[i] ~ dnorm(mu[i], tau)
        mu[i] <- beta0 + beta1 * cases[i] + beta2*distance[i]
    }
    beta0 ~ dnorm(0.0, 1.0E-4)
    beta1 ~ dnorm(0.0, 1.0E-4)
    beta2 ~ dnorm(0.0, 1.0E-4)
    tau ~ dgamma(0.01, 0.01)
typical.y <- beta0 + beta1*mean(cases[]) + beta2*mean(distance[])

}

" # close quote for modelString
writeLines( modelString , con="TEMPmodel.txt" )
initsList = list(beta0 = 0, beta1 = 0, beta2 =0, tau=1)

# Run the chains:
jagsModel = jags.model( file="TEMPmodel.txt" , data=dataList , inits=initsList ,
                        n.chains=3 , n.adapt=1000 )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 25
##    Unobserved stochastic nodes: 4
##    Total graph size: 152
##
## Initializing model
```

```
update( jagsModel , n.iter=1000 )
codaSamples = coda.samples( jagsModel , variable.names=c("beta0", "beta1", "beta2", "typical.y") ,
                            n.iter=10000)
save( codaSamples , file=paste0(fileNameRoot,"Mcmc.Rdata") )



source("DBDA2E-utilities.R")
```

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

```
library(ggmcmc)
s = ggs(codaSamples)
d=ggs_density(s)
print(d)
```




```
summary(codaSamples)
```

```
##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean       SD  Naive SE Time-series SE
## beta0      2.33761 1.156038 6.674e-03      1.366e-02
## beta1      1.61121 0.177351 1.024e-03      3.985e-03
## beta2      0.01448 0.003763 2.173e-05      8.349e-05
## typical.y 22.37772 0.686074 3.961e-03      4.000e-03
```

```
##
## 2. Quantiles for each variable:
##
##                2.5%    25%      50%      75%    97.5%
## beta0      0.021169  1.587  2.34319  3.09270  4.61038
## beta1      1.250970  1.498  1.61295  1.72832  1.95707
## beta2      0.007209  0.012  0.01445  0.01688  0.02212
## typical.y 21.017001 21.936 22.38189 22.82635 23.72774
```

Form the above summary, we see typical.y has the mean vlaue of 22.38 with a range of (21.04 - 23.72) with 95% probability.

l. i. The following code is incomplete - give the rest of it as would be needed to calculate R^2_B

definition of sigma s2<-1/tau s <-sqrt(s2) # calculation of the sample variance for (i in 1:n){ c.time[i]<-time[i]-mean(time[]) } sy2 <- inprod(c.time[], c.time[])/(n-1) # calculation of Bayesian version of adj R squared

R2B <- 1-s2/sy2

ii) Include the code in the model and re-run after monitoring R2B - give a full interpretation of the R2B output.

```
require(rjags)# Must have previously installed package rjags.

fileNameRoot="Assignment4" # For output file names.

Ntotal = 25  # Compute the total number of x,y pairs.
n = 25

dataList =  list( time = c(16.68, 11.5, 12.03, 14.88, 13.75, 18.11, 8, 17.83, 79.24, 21.5, 40.33, 21, 13

#Define the model:
modelString = "
model{
    for( i in 1 : Ntotal ) {
        time[i] ~ dnorm(mu[i], tau)
        mu[i] <- beta0 + beta1 * cases[i] + beta2*distance[i]
    }
    beta0 ~ dnorm(0.0, 1.0E-4)
    beta1 ~ dnorm(0.0, 1.0E-4)
    beta2 ~ dnorm(0.0, 1.0E-4)
    tau ~ dgamma(0.01, 0.01)
typical.y <- beta0 + beta1*mean(cases[]) + beta2*mean(distance[])

 s2<-1/tau
 s <-sqrt(s2)
 # calculation of the sample variance
 for (i in 1:Ntotal){ c.time[i]<-time[i]-mean(time[]) }
 sy2 <- inprod( c.time[], c.time[] )/(Ntotal-1)
 # calculation of Bayesian version of adj R squared

 R2B <- 1-s2/sy2
```

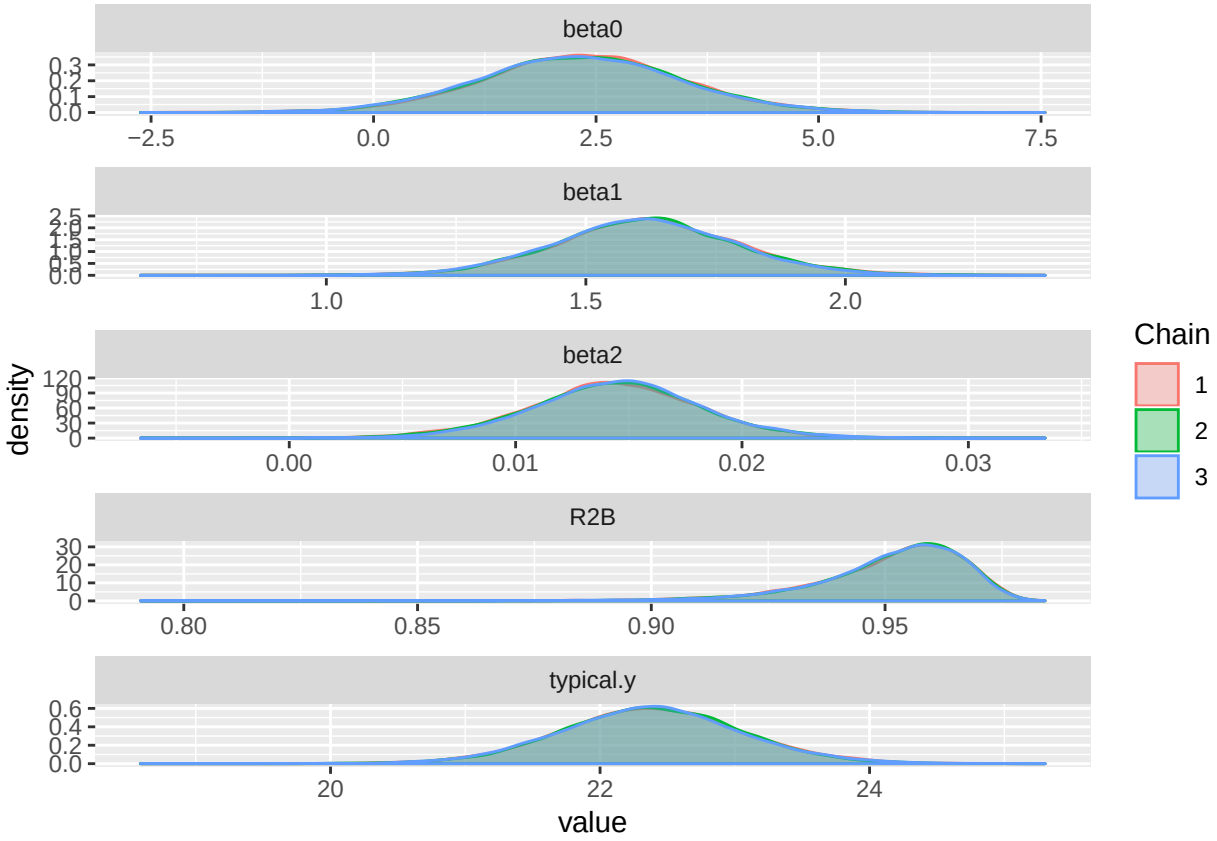
```
}

" # close quote for modelString
writeLines( modelString , con="TEMPmodel.txt" )
initsList = list(beta0 = 0, beta1 = 0, beta2 =0, tau=1)

# Run the chains:
jagsModel = jags.model( file="TEMPmodel.txt" , data=dataList , inits=initsList ,
                        n.chains=3 , n.adapt=1000 )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 25
##    Unobserved stochastic nodes: 4
##    Total graph size: 188
##
## Initializing model
```

```
update( jagsModel , n.iter=1000 )
codaSamples = coda.samples( jagsModel , variable.names=c("beta0", "beta1", "beta2", "R2B", "typical.y")
                            n.iter=10000)
save( codaSamples , file=paste0(fileNameRoot,"Mcmc.Rdata") )

source("DBDA2E-utilities.R")
```

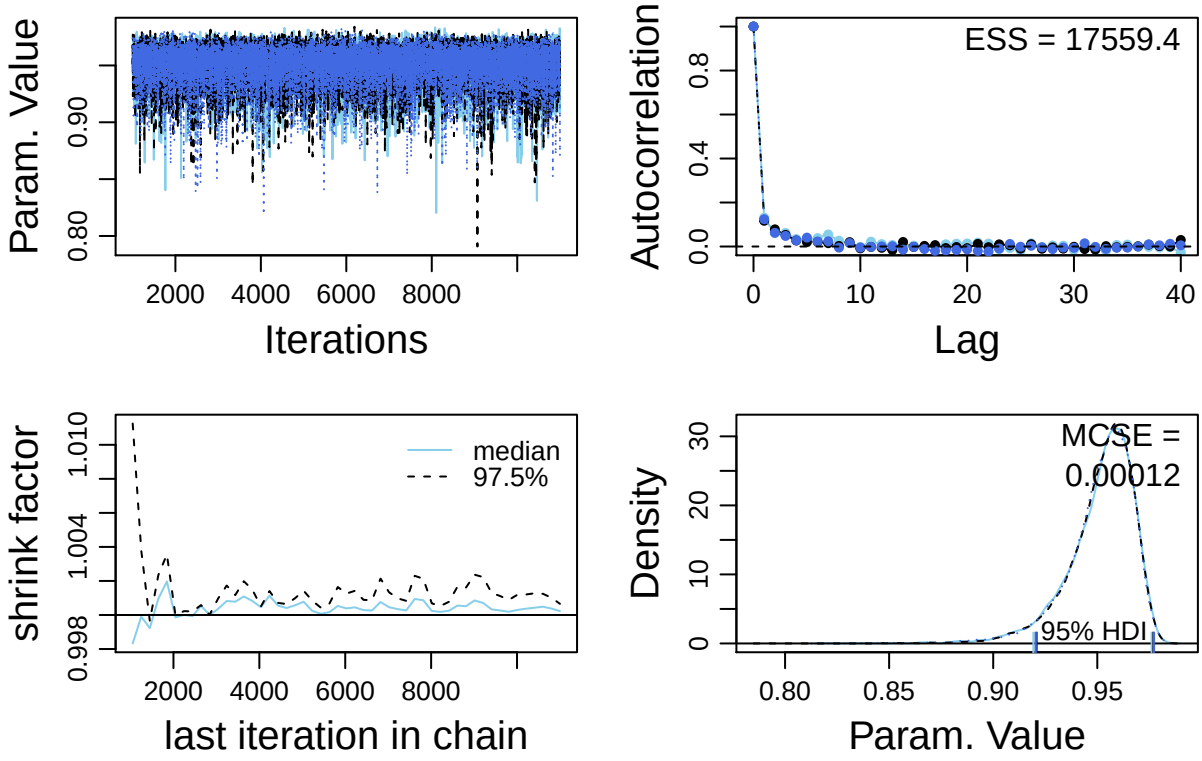
```
##
## ***********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## ***********************************************************************
```

```
library(ggmcmc)
s = ggs(codaSamples)
d=ggs_density(s)
print(d)
```




```
diagMCMC( codaObject=codaSamples , parName="R2B" )
saveGraph( file=paste0(fileNameRoot,"m") , type="jpeg" )
```

R2B




```
summary(codaSamples)
```

```
##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean       SD  Naive SE Time-series SE
## R2B        0.95170 0.015955 9.211e-05      1.210e-04
## beta0      2.32728 1.144905 6.610e-03      1.351e-02
## beta1      1.61591 0.177967 1.027e-03      3.787e-03
## beta2      0.01441 0.003782 2.184e-05      8.135e-05
## typical.y 22.38044 0.676574 3.906e-03      3.910e-03
##
## 2. Quantiles for each variable:
##
##                2.5%      25%      50%      75%    97.5%
## R2B        0.911967  0.94411  0.95471  0.96274  0.97373
## beta0      0.061537  1.58357  2.32867  3.07467  4.60528
## beta1      1.267287  1.50112  1.61526  1.73102  1.97330
## beta2      0.006763  0.01202  0.01444  0.01684  0.02189
## typical.y 21.045215 21.93761 22.37978 22.81912 23.72548
```

from the above summary, we get the mean R2B as 0.95 with a range (0.91, 0.97) with 95% probability.

iii. Include code that will calculate $p(\beta_2 > 0.01|D)$ where D is hte data.

```
lm1 = lm(time~distance+cases, data = df)
slm1 = step(lm1)
```

```
## Start:  AIC=61.88
## time ~ distance + cases
##
##            Df Sum of Sq     RSS     AIC
## <none>                  233.73  61.882
## - distance  1    168.40  402.13  73.448
## - cases     1    951.66 1185.39 100.474
```

iv. Find the point estimate for the above probability.

```
summary(slm1)
```

```
##
## Call:
## lm(formula = time ~ distance + cases, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7880 -0.6629  0.4364  1.1566  7.4197
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.341231   1.096730   2.135 0.044170 *
## distance    0.014385   0.003613   3.981 0.000631 ***
## cases       1.615907   0.170735   9.464 3.25e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.259 on 22 degrees of freedom
## Multiple R-squared:  0.9596, Adjusted R-squared:  0.9559
## F-statistic: 261.2 on 2 and 22 DF,  p-value: 4.687e-16
```

```
slm1$anova
```

```
##   Step Df Deviance Resid. Df Resid. Dev      AIC
## 1      NA       NA        22   233.7317 61.88245
```

2. Dobson (1983) analyses binary dose-response data published by Bliss (1935), in which the numbers of beetles killed after 5 hour exposure to carbon disulphide at N = 8 different concentrations are recorded.

a) Using ggplot make appropriate plots of the data.

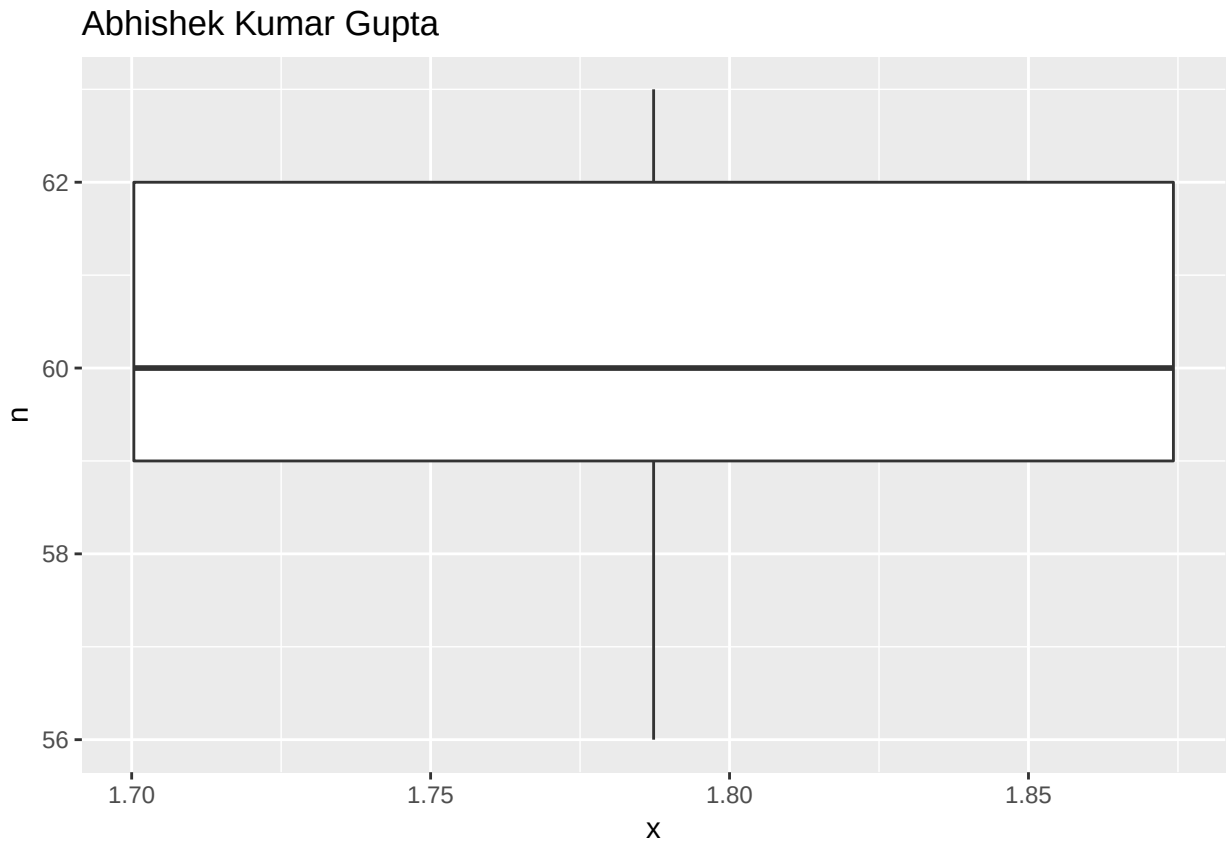
```
df = data.frame(list(x = c(1.6907, 1.7242, 1.7552, 1.7842, 1.8113, 1.8369, 1.8610, 1.8839),
                     n = c(59, 60, 62, 56, 63, 59, 62, 60),
```

```
                    y = c(6, 13, 18, 28, 52, 53, 61, 60)))
head(df)
```

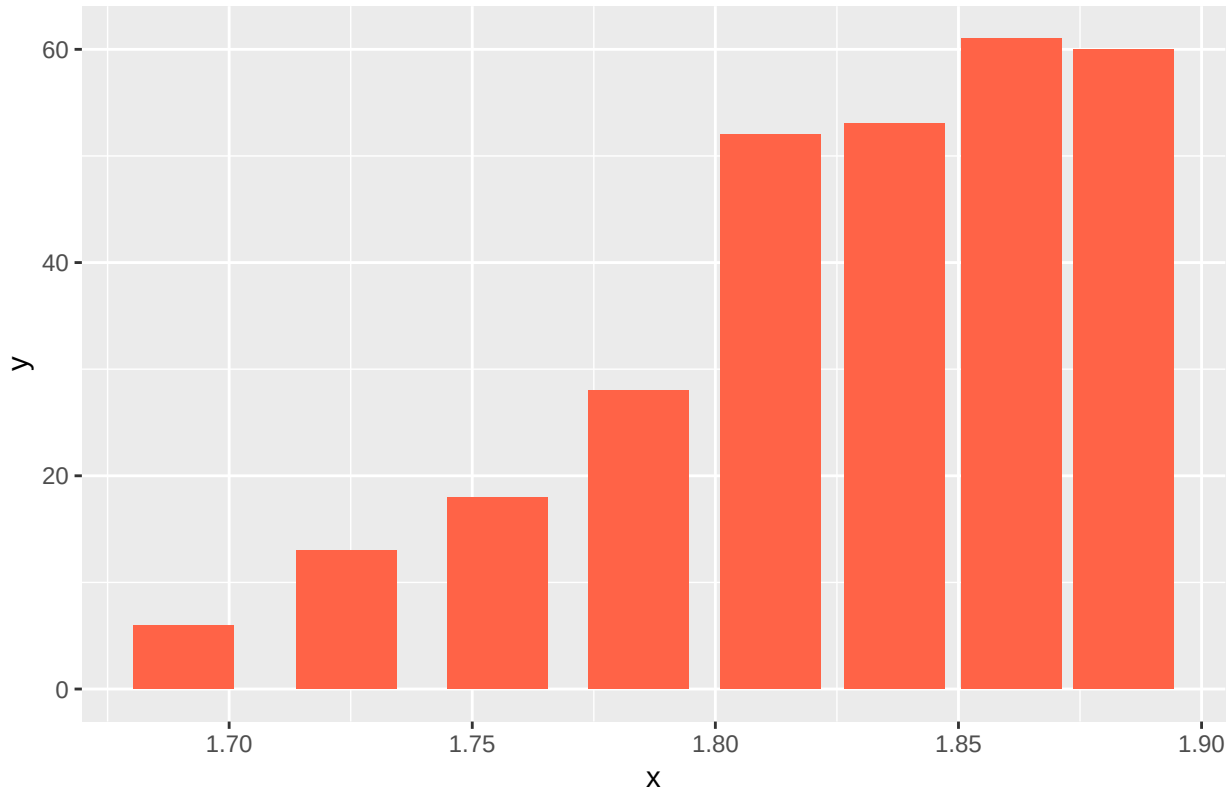
```
##        x  n  y
## 1 1.6907 59  6
## 2 1.7242 60 13
## 3 1.7552 62 18
## 4 1.7842 56 28
## 5 1.8113 63 52
## 6 1.8369 59 53
```

```
library(ggplot2)
ggplot(data=df, aes(x=x, y=n)) +geom_boxplot()+labs(title ="Abhishek Kumar Gupta")
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```




```
ggplot(data=df, aes(x=x, y=y)) +geom_bar(stat = "identity", fill = "tomato")+labs(title ="Abhishek Kuma
```



b) Complete/correct the code above and make a JAGS script to find posterior MCMC for betas and derived node p hat and y hat.

```
require(rjags)# Must have previously installed package rjags.

fileNameRoot="Assignment4" # For output file names.

Ntotal = 8  # Compute the total number of x,y pairs.

dataList = list(x = c(1.6907, 1.7242, 1.7552, 1.7842, 1.8113, 1.8369, 1.8610, 1.8839),
           n = c(59, 60, 62, 56, 63, 59, 62, 60),
           y = c(6, 13, 18, 28, 52, 53, 61, 60),Ntotal = Ntotal)

#Define the model:
modelString = "
model{
for( i in 1 : Ntotal ) {

y[i] ~ dbin(p[i], n[i])
logit(p[i]) = beta0 + beta1*(x[i] - mean(x[]) )

phat[i] <- y[i]/n[i]
yhat[i] <- n[i]*p[i]

}
```

```
beta0 ~ dnorm(0.0, 1.0E-4)
beta1 ~ dnorm(0.0, 1.0E-4)

}

" # close quote for modelString
writeLines( modelString , con="TEMPmodel.txt" )
initsList = list(beta0 = 0, beta1 = 0)

# Run the chains:
jagsModel = jags.model( file="TEMPmodel.txt" , data=dataList , inits=initsList ,
                        n.chains=3 , n.adapt=500 )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 8
##    Unobserved stochastic nodes: 2
##    Total graph size: 79
##
## Initializing model
```

```
update( jagsModel , n.iter=500 )
codaSamples = coda.samples( jagsModel , variable.names=c("beta0", "beta1", "yhat", "phat"), n.iter=5000)
save( codaSamples , file=paste0(fileNameRoot,"Mcmc.Rdata") )
summary(codaSamples)
```

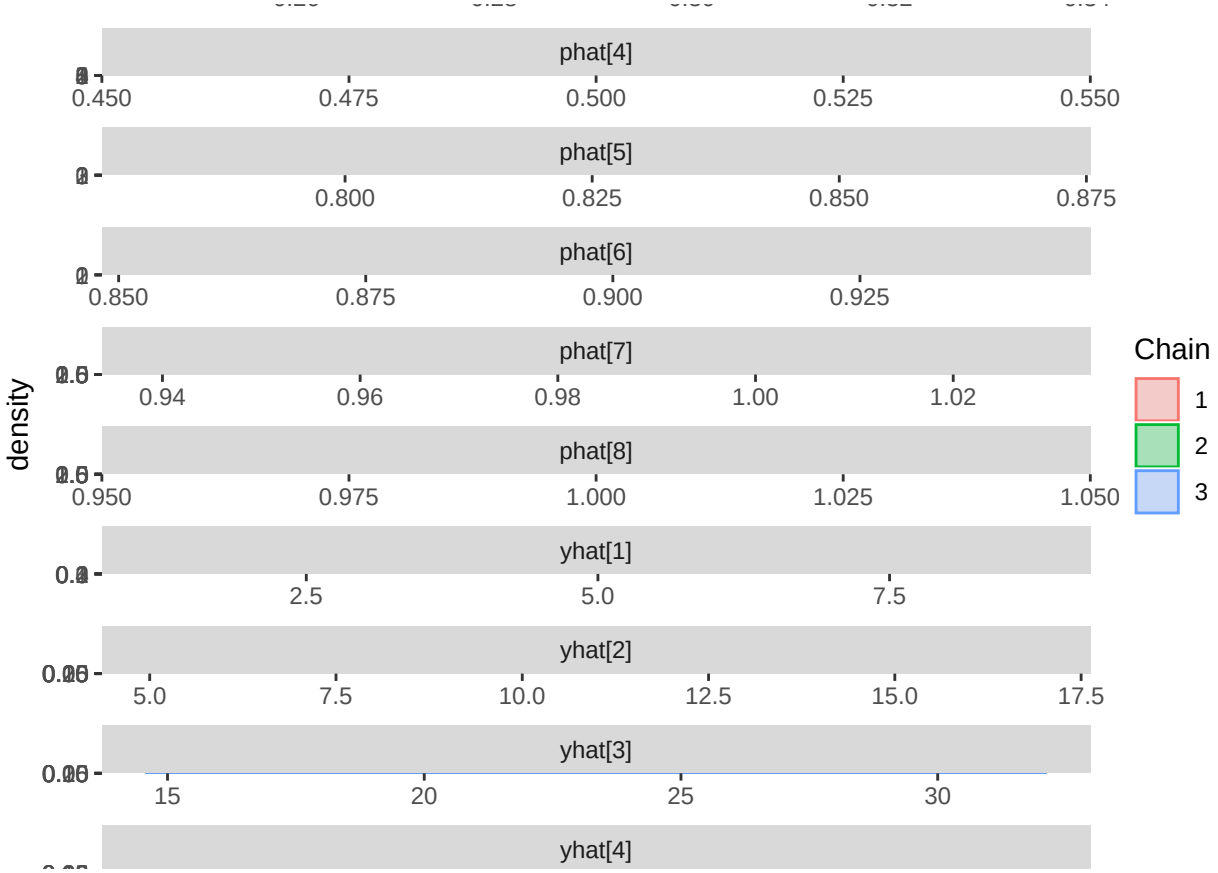
```
##
## Iterations = 1001:6000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##             Mean     SD Naive SE Time-series SE
## beta0     0.7475 0.1386 0.001132       0.001533
## beta1    34.5092 2.9214 0.023853       0.033470
## phat[1]   0.1017 0.0000 0.000000       0.000000
## phat[2]   0.2167 0.0000 0.000000       0.000000
## phat[3]   0.2903 0.0000 0.000000       0.000000
## phat[4]   0.5000 0.0000 0.000000       0.000000
## phat[5]   0.8254 0.0000 0.000000       0.000000
## phat[6]   0.8983 0.0000 0.000000       0.000000
## phat[7]   0.9839 0.0000 0.000000       0.000000
## phat[8]   1.0000 0.0000 0.000000       0.000000
## yhat[1]   3.5075 0.9443 0.007710       0.009175
## yhat[2]   9.8526 1.6823 0.013736       0.015544
## yhat[3]  22.4170 2.1098 0.017226       0.018855
## yhat[4]  33.8928 1.7676 0.014433       0.018140
## yhat[5]  50.0977 1.6588 0.013544       0.019866
## yhat[6]  53.2680 1.1063 0.009033       0.013761
```

```
## yhat[7] 59.1849 0.7336 0.005990       0.008973
## yhat[8] 58.7088 0.4224 0.003449       0.005131
##
## 2. Quantiles for each variable:
##
##            2.5%     25%     50%     75%   97.5%
## beta0    0.4802  0.6535  0.7455  0.8402  1.0228
## beta1   29.0097 32.4954 34.4383 36.4120 40.4341
## phat[1]  0.1017  0.1017  0.1017  0.1017  0.1017
## phat[2]  0.2167  0.2167  0.2167  0.2167  0.2167
## phat[3]  0.2903  0.2903  0.2903  0.2903  0.2903
## phat[4]  0.5000  0.5000  0.5000  0.5000  0.5000
## phat[5]  0.8254  0.8254  0.8254  0.8254  0.8254
## phat[6]  0.8983  0.8983  0.8983  0.8983  0.8983
## phat[7]  0.9839  0.9839  0.9839  0.9839  0.9839
## phat[8]  1.0000  1.0000  1.0000  1.0000  1.0000
## yhat[1]  1.9378  2.8307  3.4157  4.0756  5.6126
## yhat[2]  6.7982  8.6756  9.7646 10.9520 13.3547
## yhat[3] 18.3152 20.9844 22.4057 23.8434 26.5430
## yhat[4] 30.4023 32.7103 33.9061 35.0931 37.2641
## yhat[5] 46.6711 48.9930 50.1575 51.2565 53.1835
## yhat[6] 50.8936 52.5653 53.3472 54.0589 55.2019
## yhat[7] 57.5343 58.7406 59.2584 59.7156 60.4050
## yhat[8] 57.7221 58.4661 58.7630 59.0149 59.3704
```

```
source("DBDA2E-utilities.R")
```

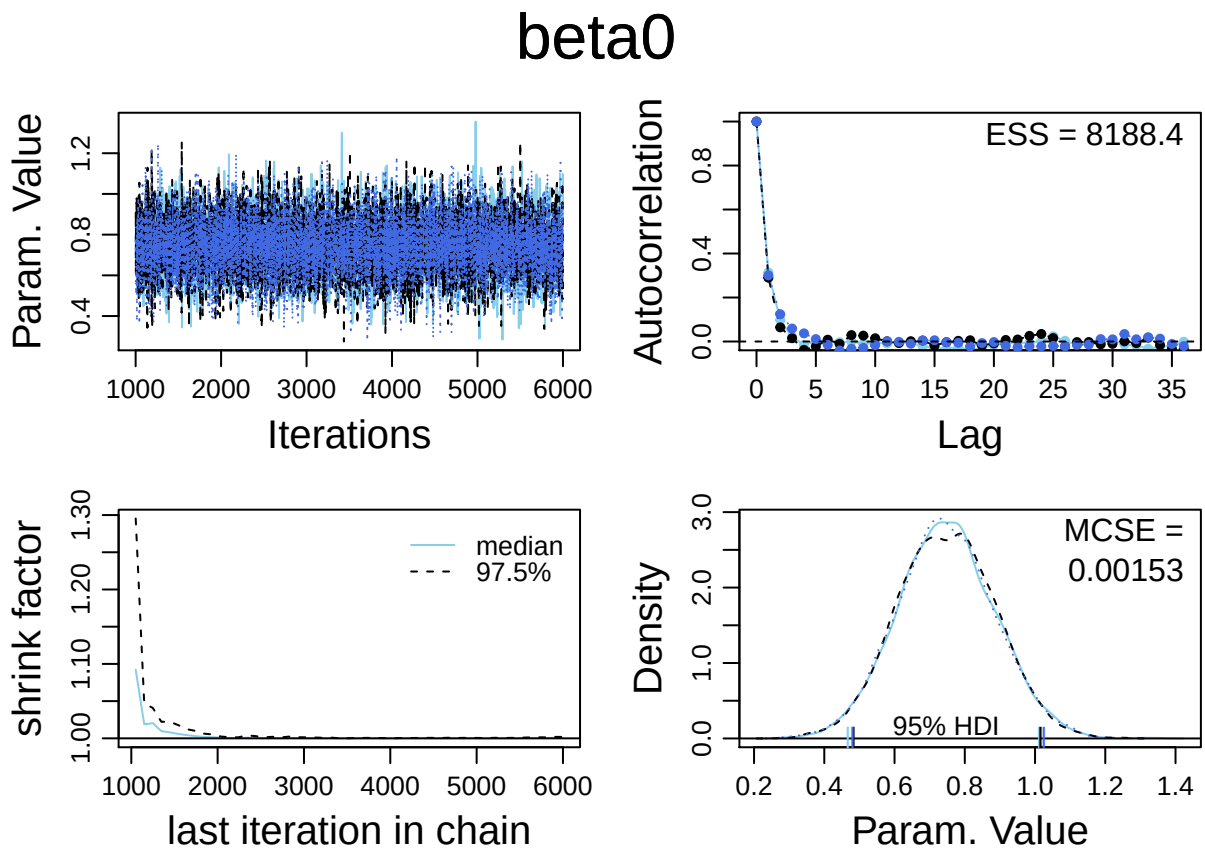
```
##
## ***********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## ***********************************************************************
```

```
library(ggmcmc)
s = ggs(codaSamples)
d=ggs_density(s)
print(d)
```

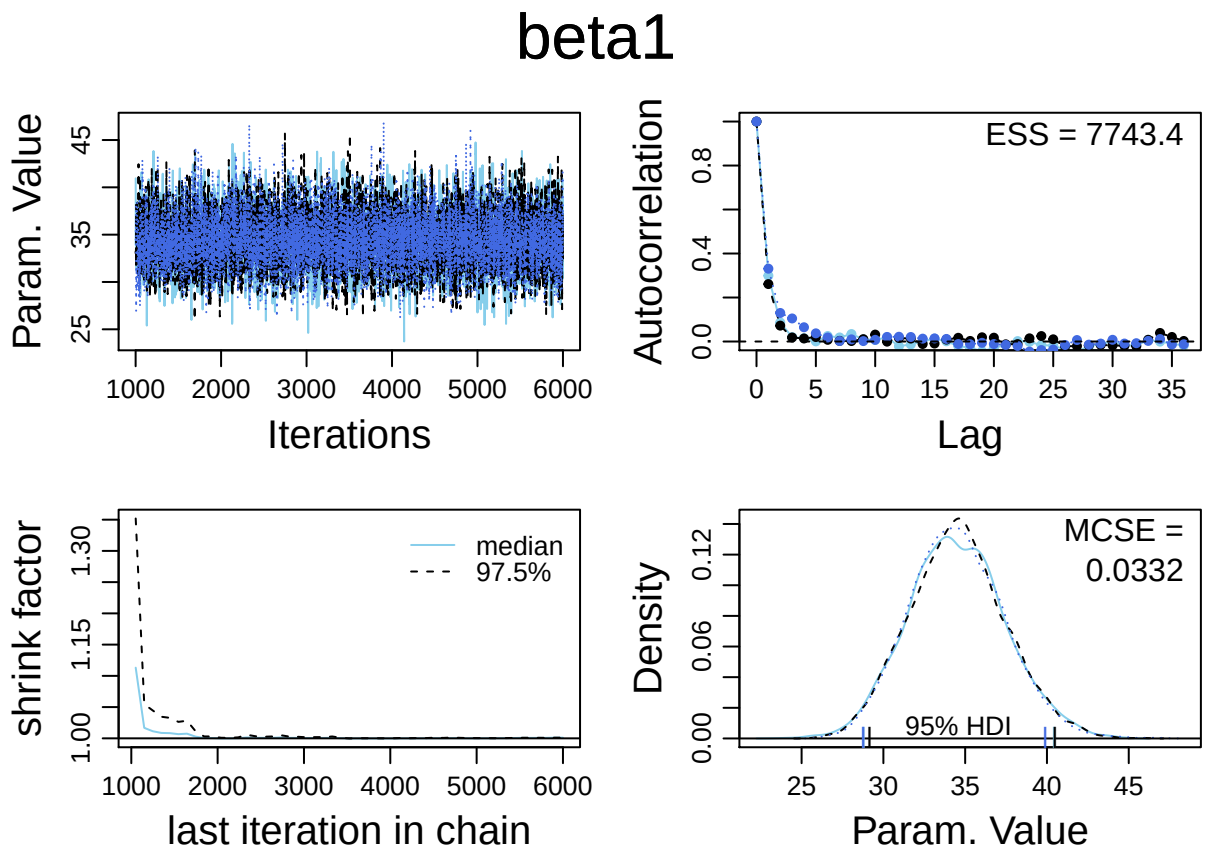


c. Give the MCMC diagnostics for 3 chains and 5000 iterations on each

```
diagMCMC( codaObject=codaSamples , parName="beta0" )
saveGraph( file=paste0(fileNameRoot,"m") , type="jpeg" )
```




```
diagMCMC( codaObject=codaSamples , parName="beta1" )
saveGraph( file=paste0(fileNameRoot,"m") , type="jpeg" )
```



d. Make a JAGS script that does not center the independent variable (Dose)

```
require(rjags)# Must have previously installed package rjags.

fileNameRoot="Assignment4" # For output file names.

Ntotal = 8  # Compute the total number of x,y pairs.

dataList = list(x = c(1.6907, 1.7242, 1.7552, 1.7842, 1.8113, 1.8369, 1.8610, 1.8839),
           n = c(59, 60, 62, 56, 63, 59, 62, 60),
           y = c(6, 13, 18, 28, 52, 53, 61, 60),Ntotal = Ntotal)

#Define the model:
modelString = "
model{
for( i in 1 : Ntotal ) {

y[i] ~ dbin(p[i], n[i])
logit(p[i]) = beta0 + beta1*(x[i])

phat[i] <- y[i]/n[i]
yhat[i] <- n[i]*p[i]

}
```

```
beta0 ~ dnorm(0.0, 1.0E-4)
beta1 ~ dnorm(0.0, 1.0E-4)

}

" # close quote for modelString
writeLines( modelString , con="TEMPmodel.txt" )
initsList = list(beta0 = 0, beta1 = 0)

# Run the chains:
jagsModel = jags.model( file="TEMPmodel.txt" , data=dataList , inits=initsList ,
                        n.chains=3 , n.adapt=500 )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 8
##    Unobserved stochastic nodes: 2
##    Total graph size: 69
##
## Initializing model
```

```
update( jagsModel , n.iter=500 )
codaSamples = coda.samples( jagsModel , variable.names=c("beta0", "beta1", "yhat", "phat"), n.iter=5000)
save( codaSamples , file=paste0(fileNameRoot,"Mcmc.Rdata") )
summary(codaSamples)
```

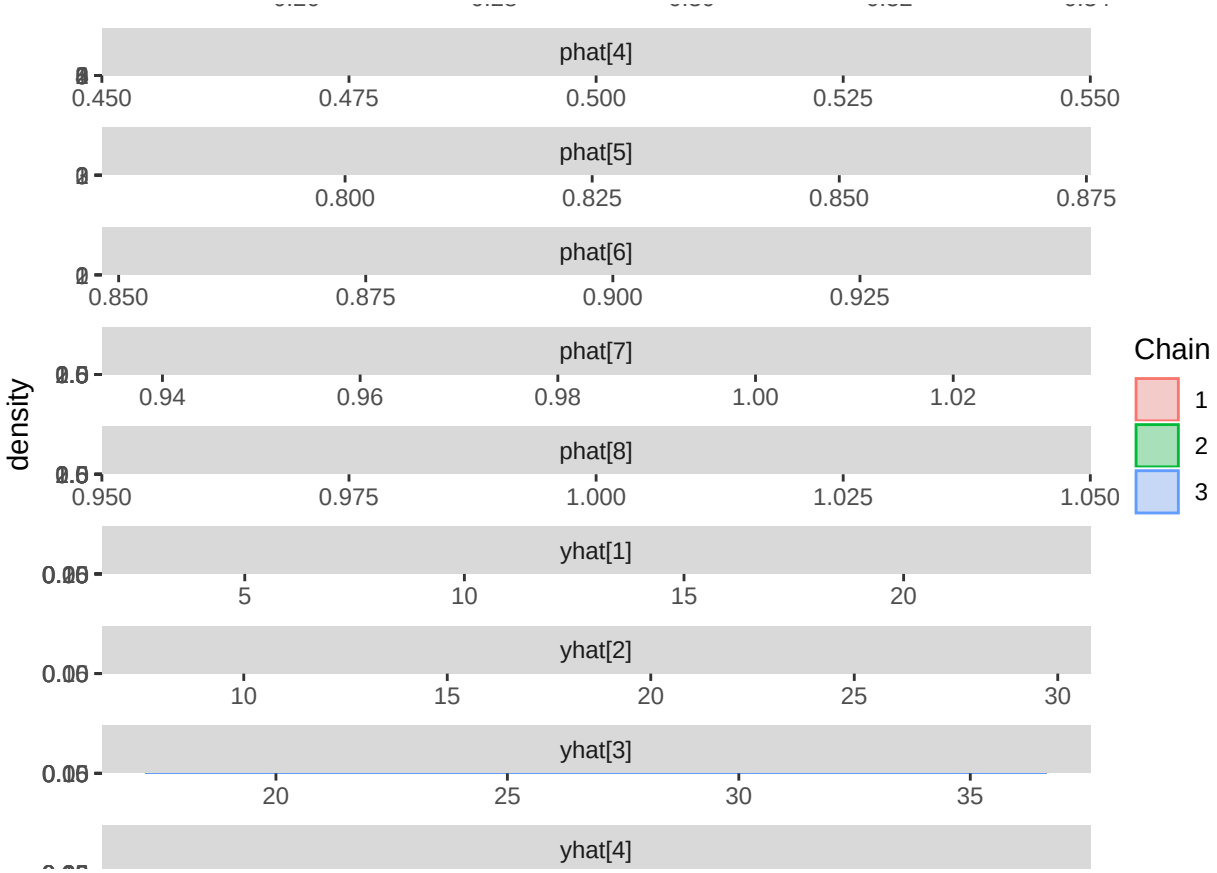
```
##
## Iterations = 1001:6000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##             Mean     SD Naive SE Time-series SE
## beta0   -43.7353 10.054  0.08209        4.27604
## beta1    24.7353  5.641  0.04606        2.40990
## phat[1]   0.1017  0.000  0.00000        0.00000
## phat[2]   0.2167  0.000  0.00000        0.00000
## phat[3]   0.2903  0.000  0.00000        0.00000
## phat[4]   0.5000  0.000  0.00000        0.00000
## phat[5]   0.8254  0.000  0.00000        0.00000
## phat[6]   0.8983  0.000  0.00000        0.00000
## phat[7]   0.9839  0.000  0.00000        0.00000
## phat[8]   1.0000  0.000  0.00000        0.00000
## yhat[1]   8.2662  3.930  0.03208        1.42358
## yhat[2]  15.4591  4.034  0.03294        1.38646
## yhat[3]  26.1260  2.893  0.02362        0.61188
## yhat[4]  33.4733  1.599  0.01305        0.01667
## yhat[5]  46.7635  2.443  0.01995        0.57970
## yhat[6]  49.6030  2.667  0.02177        0.92826
```

```
## yhat[7]  55.8761  2.652  0.02165        0.96791
## yhat[8]  56.2552  2.191  0.01789        0.78501
##
## 2. Quantiles for each variable:
##
##             2.5%      25%      50%      75%    97.5%
## beta0   -57.7642 -52.3867 -44.3495 -36.4813 -24.0709
## beta1    13.7005  20.6735  25.0687  29.5907  32.6116
## phat[1]   0.1017   0.1017   0.1017   0.1017   0.1017
## phat[2]   0.2167   0.2167   0.2167   0.2167   0.2167
## phat[3]   0.2903   0.2903   0.2903   0.2903   0.2903
## phat[4]   0.5000   0.5000   0.5000   0.5000   0.5000
## phat[5]   0.8254   0.8254   0.8254   0.8254   0.8254
## phat[6]   0.8983   0.8983   0.8983   0.8983   0.8983
## phat[7]   0.9839   0.9839   0.9839   0.9839   0.9839
## phat[8]   1.0000   1.0000   1.0000   1.0000   1.0000
## yhat[1]   3.6921   5.1096   7.1651  10.5331  17.4207
## yhat[2]   9.7250  12.2272  14.6213  18.2007  24.1547
## yhat[3]  20.9672  24.0149  25.9051  28.1209  31.9492
## yhat[4]  30.3510  32.4032  33.4593  34.5452  36.6334
## yhat[5]  41.7547  45.0751  46.9482  48.5948  50.9936
## yhat[6]  43.7809  47.8260  50.1359  51.7329  53.3859
## yhat[7]  49.7312  54.2869  56.5752  58.0024  59.1109
## yhat[8]  50.9048  55.1208  56.9319  57.9758  58.6226
```

```
source("DBDA2E-utilities.R")
```

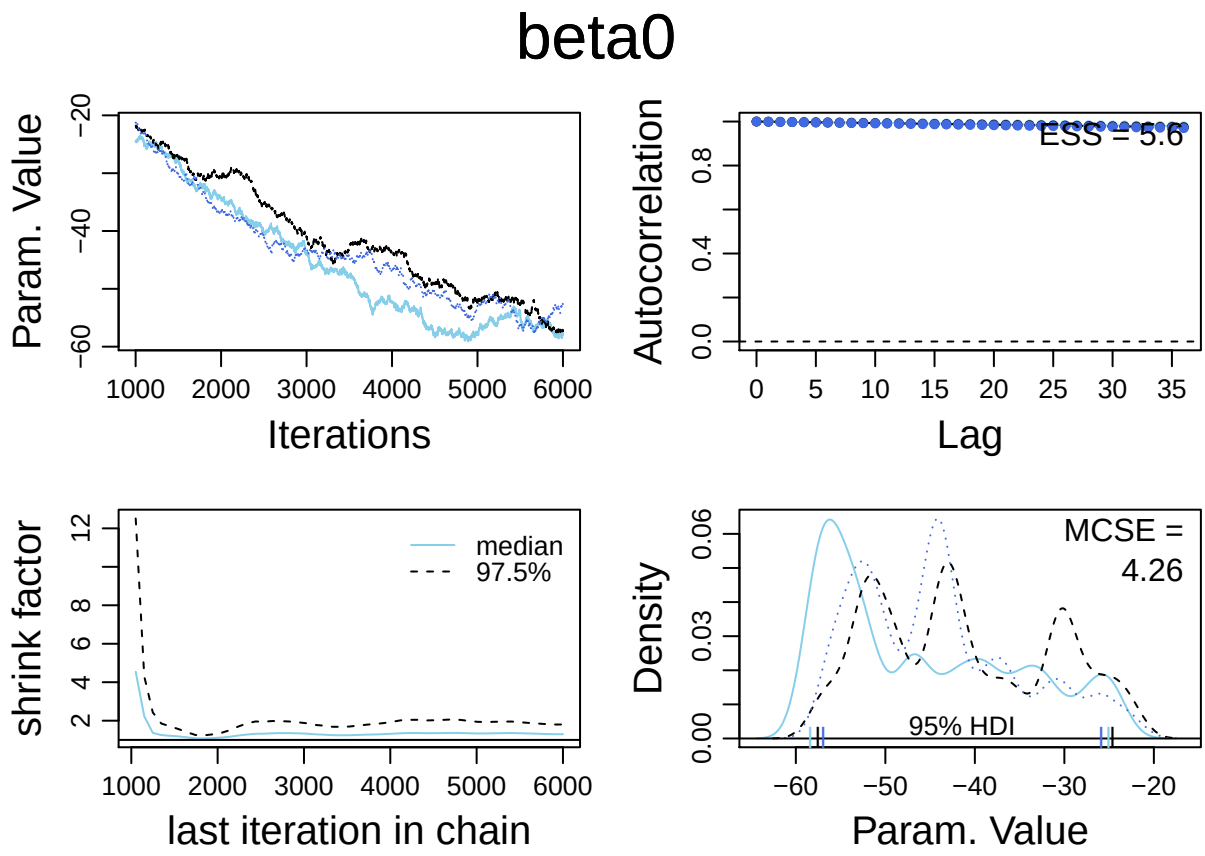
```
##
## ***********************************************************************
## Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition:
## A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.
## ***********************************************************************
```

```
library(ggmcmc)
s = ggs(codaSamples)
d=ggs_density(s)
print(d)
```

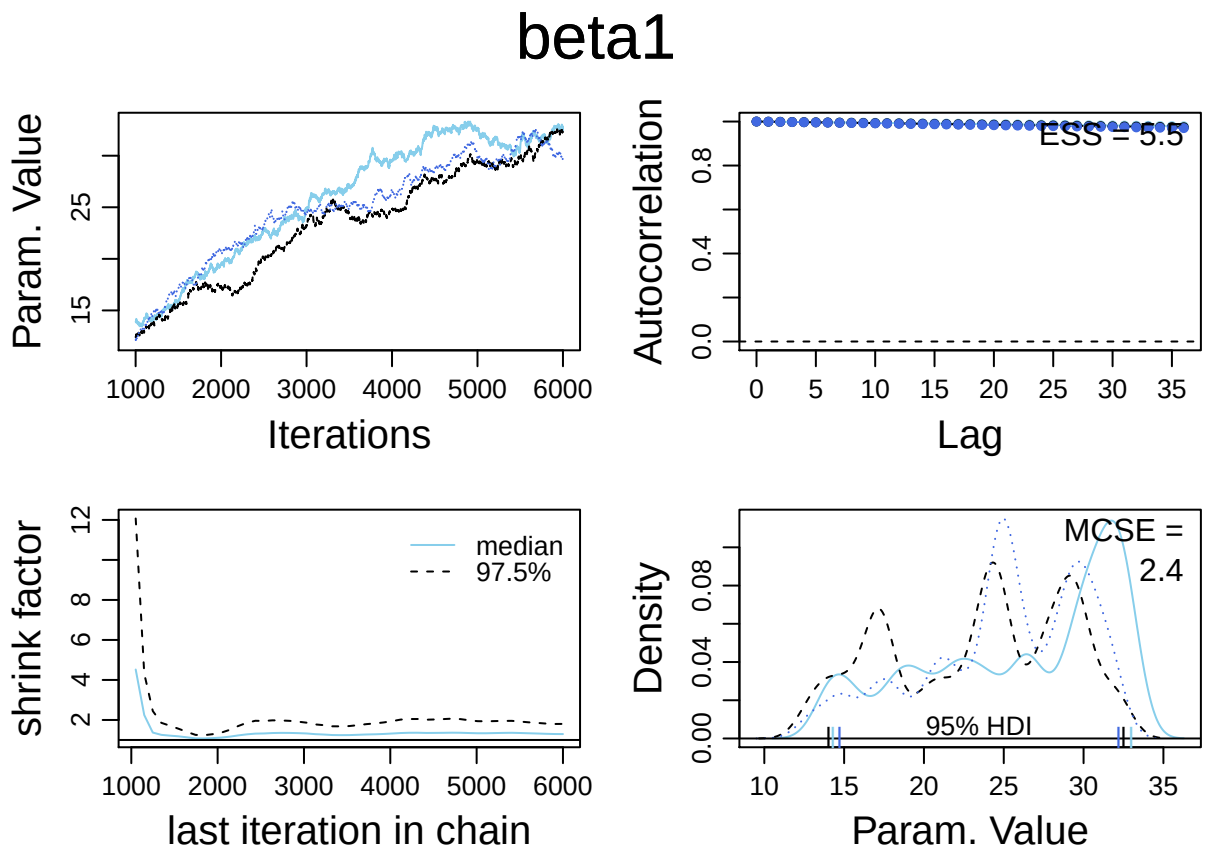


e) Run the code and show MCMC diagnostics - what do you conclude?

```
diagMCMC( codaObject=codaSamples , parName="beta0" )
saveGraph( file=paste0(fileNameRoot,"m") , type="jpeg" )
```




```
diagMCMC( codaObject=codaSamples , parName="beta1" )
saveGraph( file=paste0(fileNameRoot,"m") , type="jpeg" )
```



From the above diagnostics, we see there is a huge auto co-relation when we do not centre the independent variable dose. Also, the density don't lie over each other. However, centering the independent variable dose gives better density and low auto-correlation as can be seen from the mcmc diagnostics.

f) Using the ggmcmc package make a pairs plot

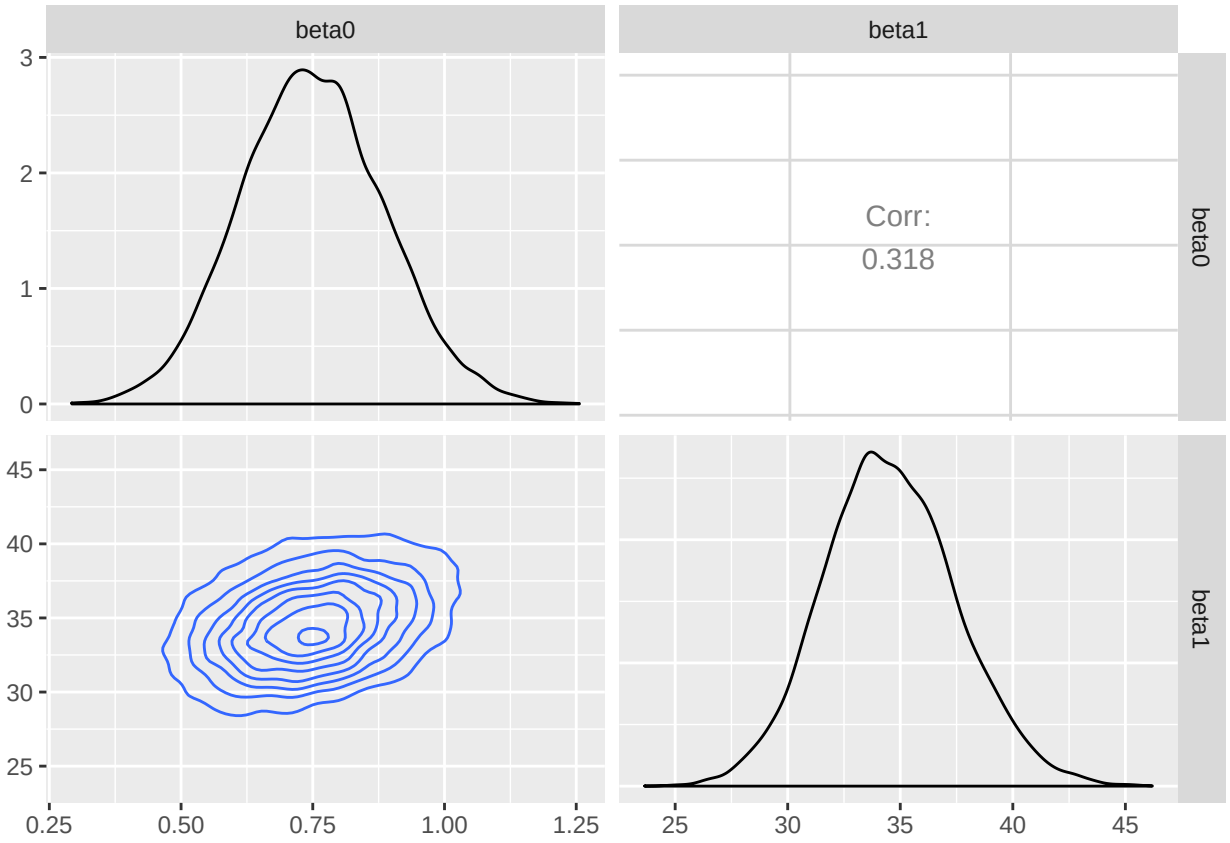
f) a.) of the posterior betas centred at x

```
#Define the model:
modelString = "
model{
for( i in 1 : Ntotal ) {

y[i] ~ dbin(p[i], n[i])
logit(p[i]) = beta0 + beta1*(x[i] - mean(x[]) )

phat[i] <- y[i]/n[i]
yhat[i] <- n[i]*p[i]

}

beta0 ~ dnorm(0.0, 1.0E-4)
beta1 ~ dnorm(0.0, 1.0E-4)

}
```

```
" # close quote for modelString
writeLines( modelString , con="TEMPmodel.txt" )
initsList = list(beta0 = 0, beta1 = 0)

# Run the chains:
jagsModel = jags.model( file="TEMPmodel.txt" , data=dataList , inits=initsList ,
                        n.chains=3 , n.adapt=500 )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 8
##    Unobserved stochastic nodes: 2
##    Total graph size: 79
##
## Initializing model
```

```
update( jagsModel , n.iter=500 )
codaSamples = coda.samples( jagsModel , variable.names=c("beta0", "beta1"), n.iter=5000)
save( codaSamples , file=paste0(fileNameRoot,"Mcmc.Rdata") )
s = ggs(codaSamples)
ggs_pairs(s, lower = list(continuous = "density"))
```

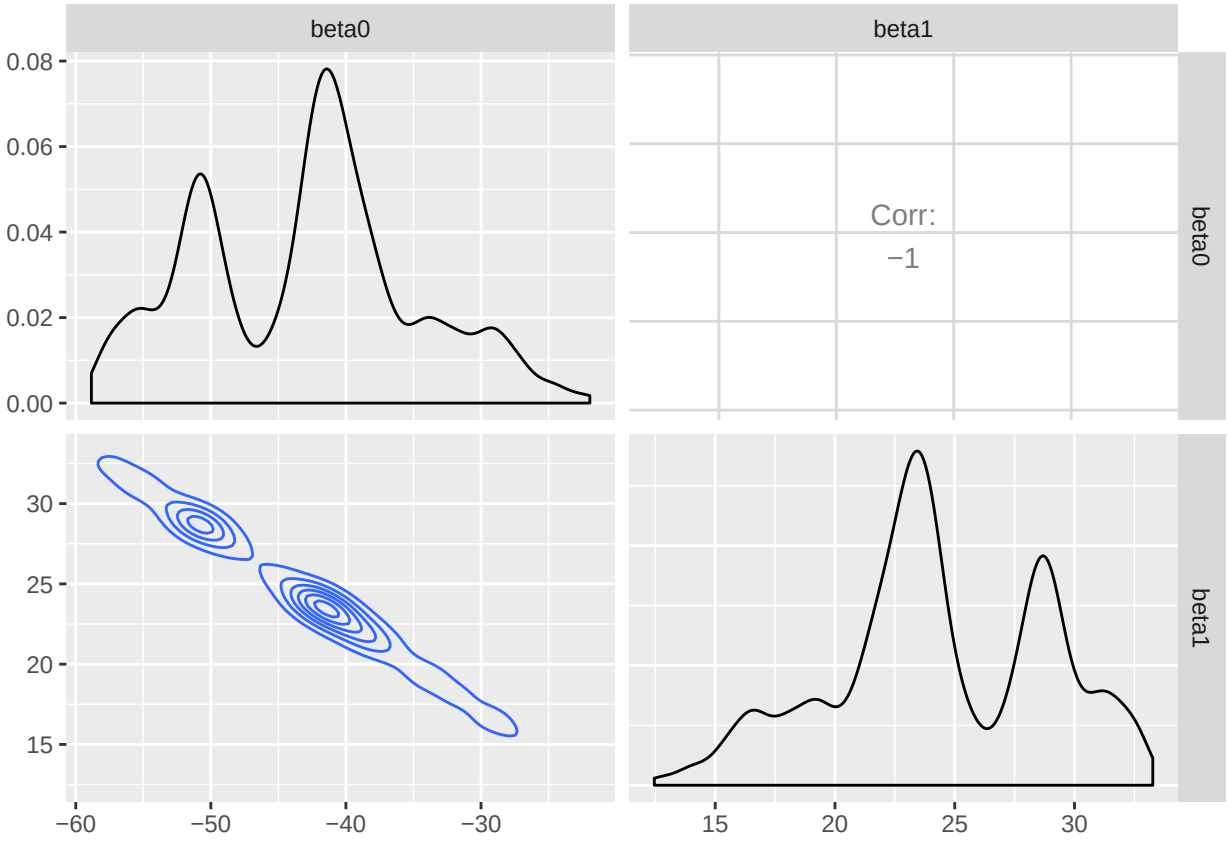


f) b.) of the posterior betas not centred at x

```
#Define the model:
modelString = "
model{
for( i in 1 : Ntotal ) {

y[i] ~ dbin(p[i], n[i])
logit(p[i]) = beta0 + beta1*x[i]

phat[i] <- y[i]/n[i]
yhat[i] <- n[i]*p[i]

}

beta0 ~ dnorm(0.0, 1.0E-4)
beta1 ~ dnorm(0.0, 1.0E-4)

}

" # close quote for modelString
writeLines( modelString , con="TEMPmodel.txt" )
initsList = list(beta0 = 0, beta1 = 0)

# Run the chains:
jagsModel = jags.model( file="TEMPmodel.txt" , data=dataList , inits=initsList ,
                        n.chains=3 , n.adapt=500 )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 8
##    Unobserved stochastic nodes: 2
##    Total graph size: 69
##
## Initializing model
```

```
update( jagsModel , n.iter=500 )
codaSamples = coda.samples( jagsModel , variable.names=c("beta0", "beta1"), n.iter=5000)
save( codaSamples , file=paste0(fileNameRoot,"Mcmc.Rdata") )
s = ggs(codaSamples)
ggs_pairs(s, lower = list(continuous = "density"))
```



f) c. Compare the pictures and make some conclusions - what does centering accomplish?

From the above pairs plot, we clearly see that centering gives well normally distributed betas whereas non centering results in betas with highly irregular distribution.

g) Say which are random:

yhat[i], beta0, beta1, p[i]

h) In the model above you used a logit link - what other links could you use?

we could use probit, cloglog models

a. With centered data use a different link within your model

Here, I have used cloglog model.

```
require(rjags)# Must have previously installed package rjags.

fileNameRoot="Assignment4" # For output file names.

Ntotal = 8  # Compute the total number of x,y pairs.

dataList = list(x = c(1.6907, 1.7242, 1.7552, 1.7842, 1.8113, 1.8369, 1.8610, 1.8839),
           n = c(59, 60, 62, 56, 63, 59, 62, 60),
           y = c(6, 13, 18, 28, 52, 53, 61, 60),Ntotal = Ntotal)
```

```
#Define the model:
modelString = "
model{
for( i in 1 : Ntotal ) {

y[i] ~ dbin(p[i], n[i])
cloglog(p[i]) = beta0 + beta1*(x[i] - mean(x[]) )

phat[i] <- y[i]/n[i]
yhat[i] <- n[i]*p[i]

}

beta0 ~ dnorm(0.0, 1.0E-4)
beta1 ~ dnorm(0.0, 1.0E-4)

}

" # close quote for modelString
writeLines( modelString , con="TEMPmodel.txt" )
initsList = list(beta0 = 0, beta1 = 0)

# Run the chains:
jagsModel = jags.model( file="TEMPmodel.txt" , data=dataList , inits=initsList ,
                        n.chains=3 , n.adapt=500 )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 8
##    Unobserved stochastic nodes: 2
##    Total graph size: 79
##
## Initializing model
```

```
update( jagsModel , n.iter=500 )
codaSamples = coda.samples( jagsModel , variable.names=c("beta0", "beta1"), n.iter=5000)
save( codaSamples , file=paste0(fileNameRoot,"Mcmc.Rdata") )
summary(codaSamples)
```

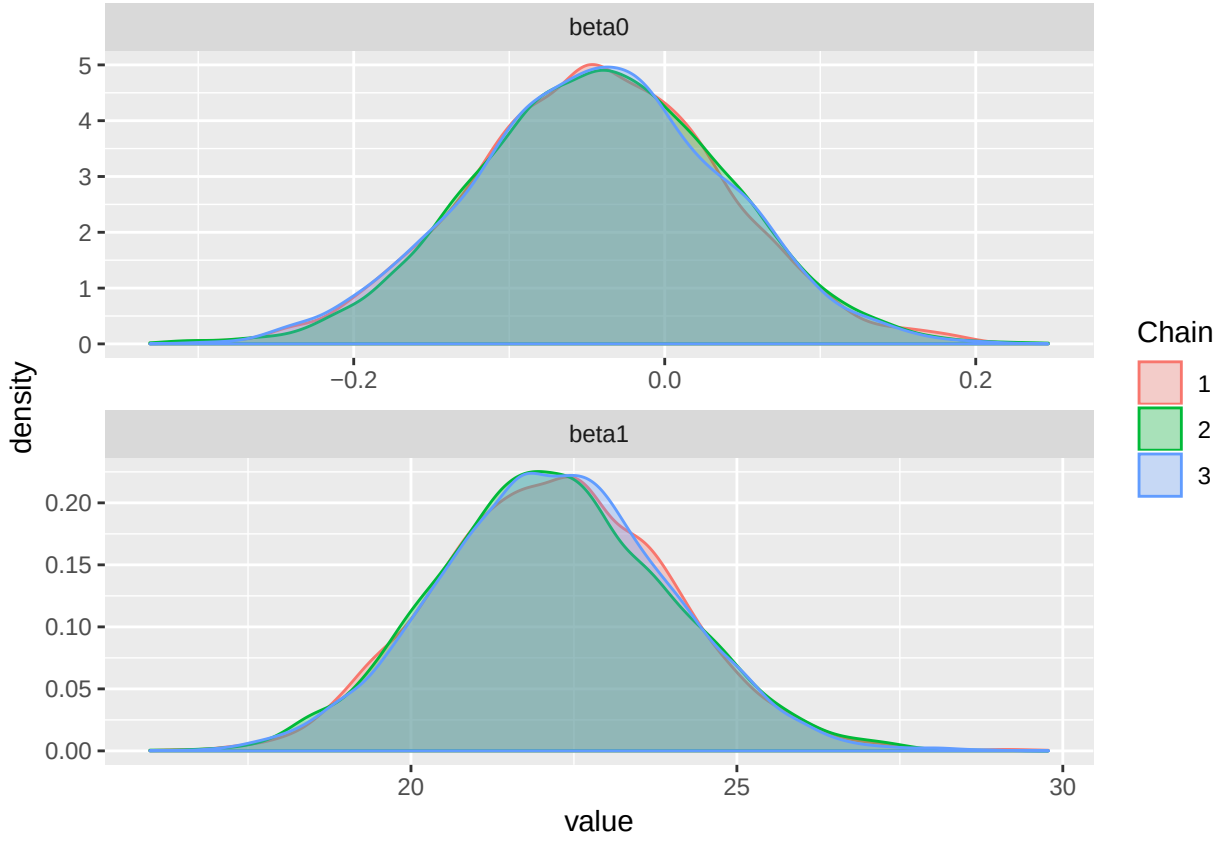
```
##
## Iterations = 1001:6000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean      SD  Naive SE Time-series SE
## beta0 -0.04331 0.08049 0.0006572      0.0008676
## beta1 22.19848 1.77155 0.0144646      0.0187109
##
```

```
## 2. Quantiles for each variable:
##
##           2.5%      25%      50%      75%   97.5%
## beta0 -0.2023 -0.09708 -0.04278  0.01082  0.1125
## beta1 18.7874 20.98836 22.17255 23.38898 25.7486
```

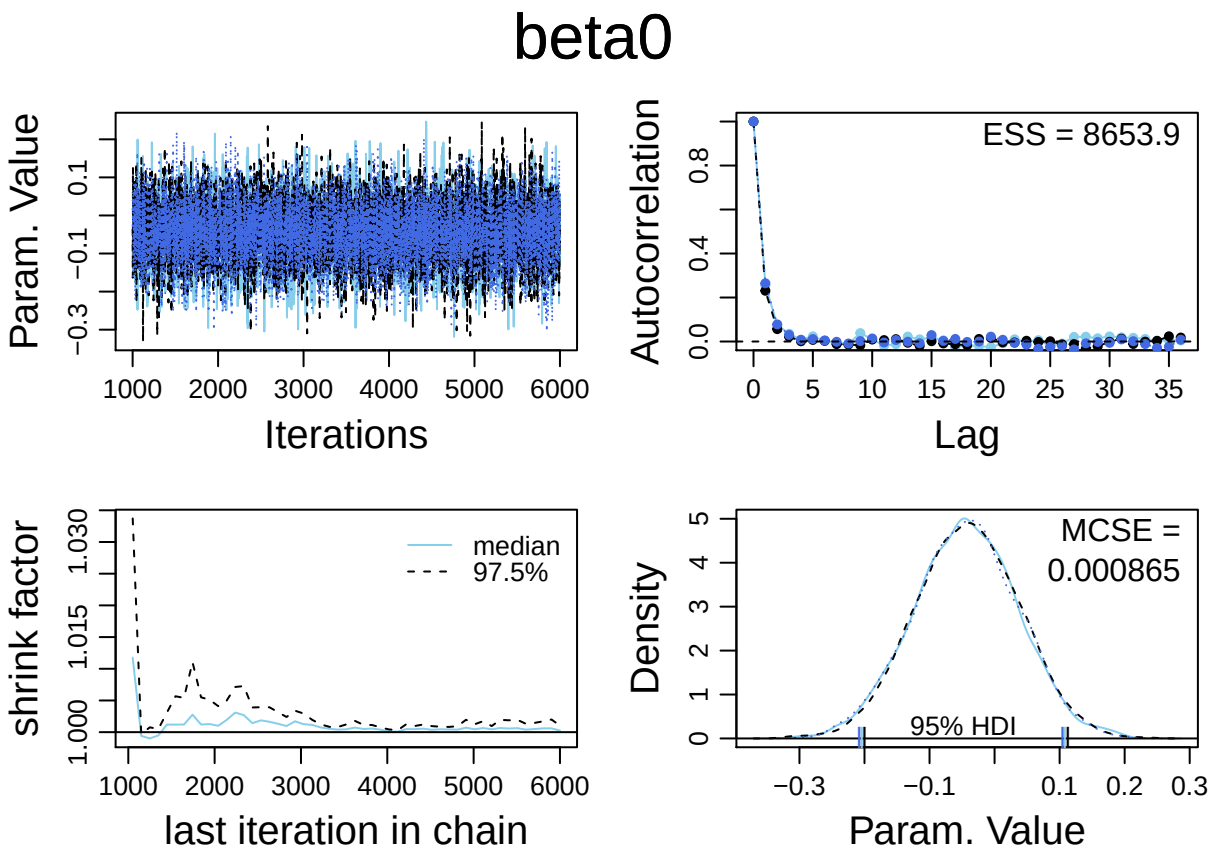
```
source("DBDA2E-utilities.R")
```

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

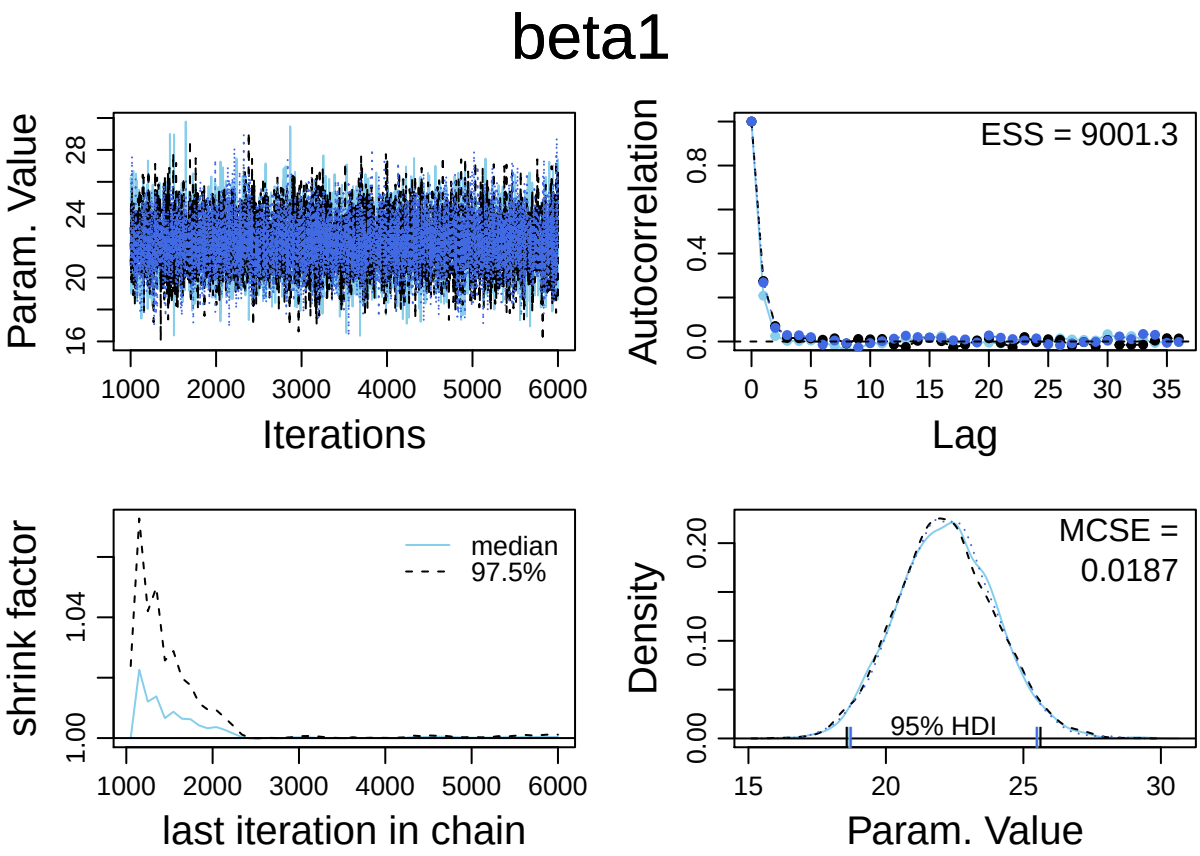
```
library(ggmcmc)
s = ggs(codaSamples)
d=ggs_density(s)
print(d)
```



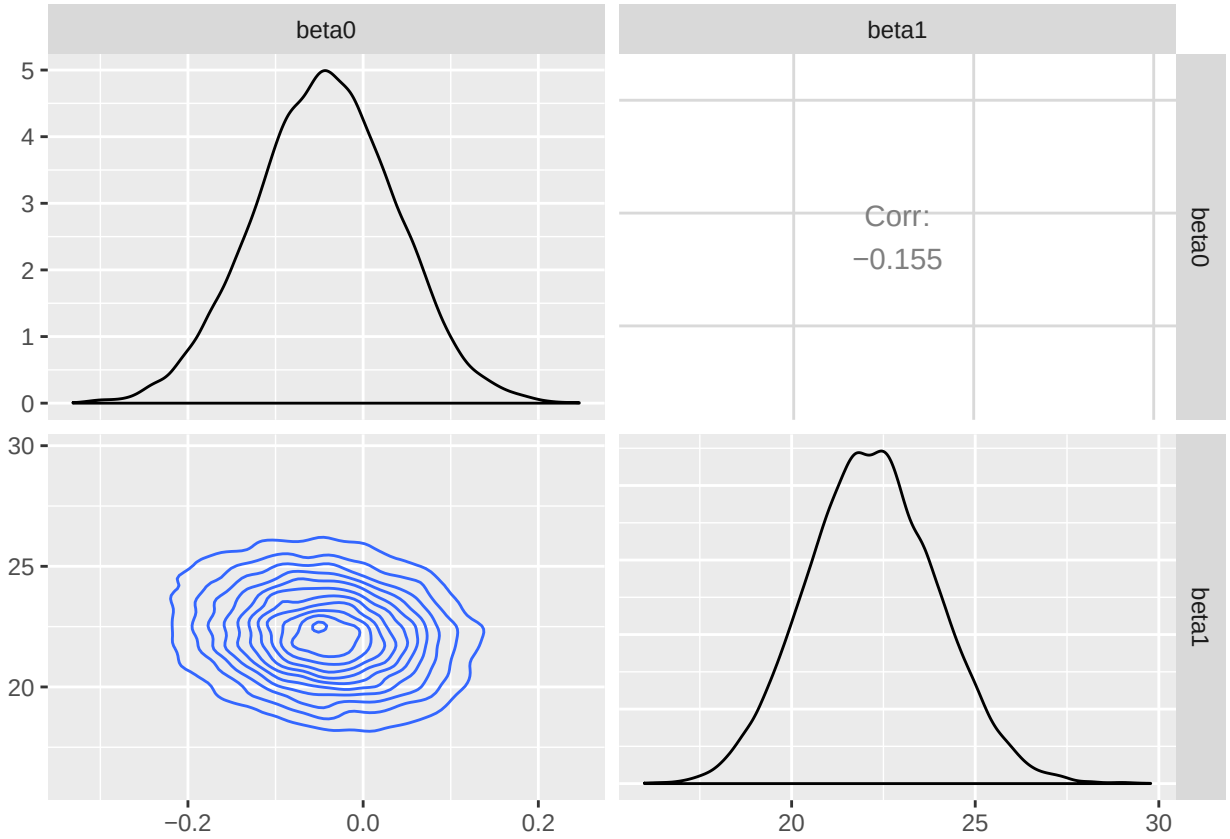

```
diagMCMC( codaObject=codaSamples , parName="beta0" )
saveGraph( file=paste0(fileNameRoot,"m") , type="jpeg" )
```




```
diagMCMC( codaObject=codaSamples , parName="beta1" )
saveGraph( file=paste0(fileNameRoot,"m") , type="jpeg" )
```




```
ggs_pairs(s, lower = list(continuous = "density"))
```



###b. Any difference in the conclusions? We see little variaitons in beta values. However, using centred independent variable, we see low auto correlation and density lies over each other.Pairs plot also show normally dstributed beta values.

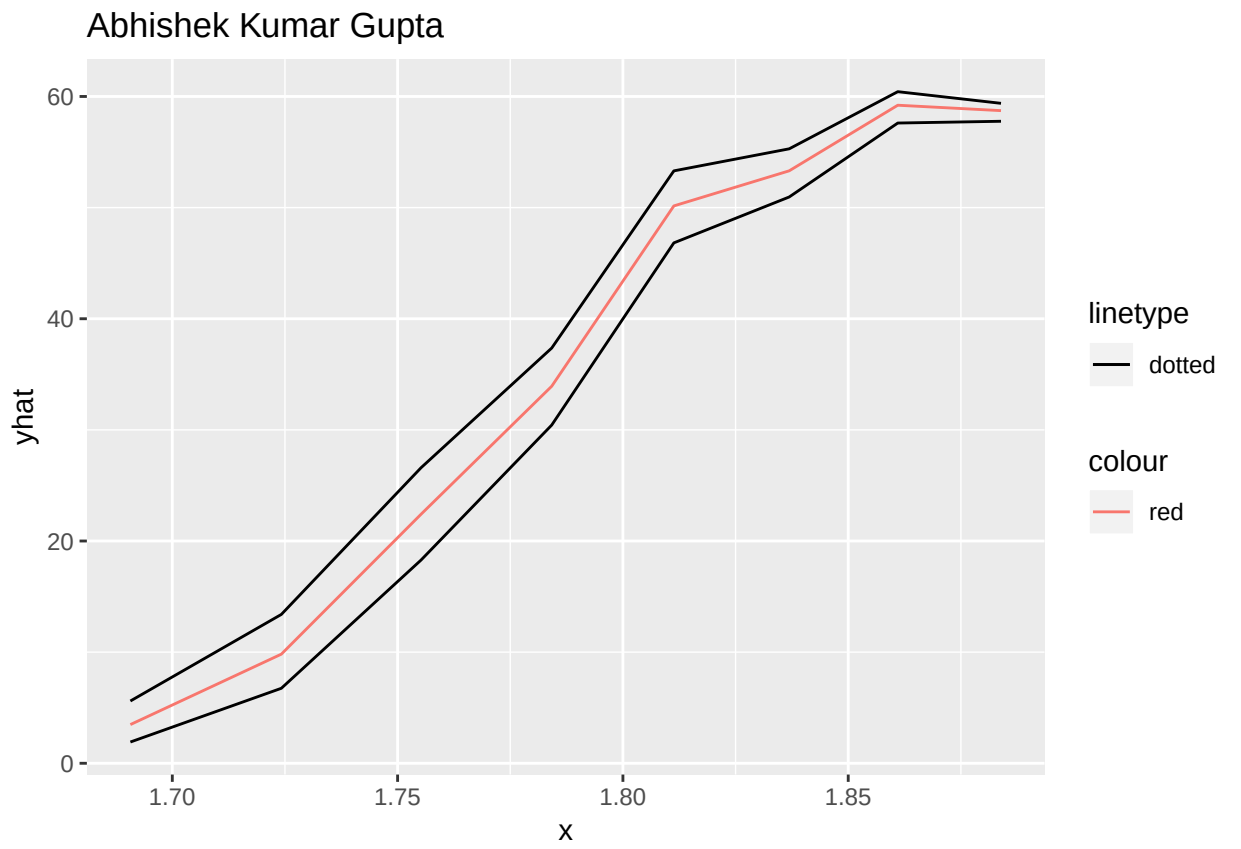
i) Duplicate the pictures below in R by making your own script that will take the data and MCMC output (from model with centered x, logit link) and make the plots (these should be far more sophisticated and clear). The plots are p Vs Dose and yhat Vs Dose.

```
require(rjags)# Must have previously installed package rjags.

fileNameRoot="Assignment4" # For output file names.

Ntotal = 8  # Compute the total number of x,y pairs.

dataList = list(x = c(1.6907, 1.7242, 1.7552, 1.7842, 1.8113, 1.8369, 1.8610, 1.8839),
           n = c(59, 60, 62, 56, 63, 59, 62, 60),
           y = c(6, 13, 18, 28, 52, 53, 61, 60),Ntotal = Ntotal)

#Define the model:
modelString = "
model{
for( i in 1 : Ntotal ) {

y[i] ~ dbin(p[i], n[i])
logit(p[i]) = beta0 + beta1*(x[i] - mean(x[]) )
```

```
phat[i] <- y[i]/n[i]
yhat[i] <- n[i]*p[i]

}

beta0 ~ dnorm(0.0, 1.0E-4)
beta1 ~ dnorm(0.0, 1.0E-4)

}

" # close quote for modelString
writeLines( modelString , con="TEMPmodel.txt" )
initsList = list(beta0 = 0, beta1 = 0)

# Run the chains:
jagsModel = jags.model( file="TEMPmodel.txt" , data=dataList , inits=initsList ,
                        n.chains=3 , n.adapt=500 )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 8
##    Unobserved stochastic nodes: 2
##    Total graph size: 79
##
## Initializing model
```

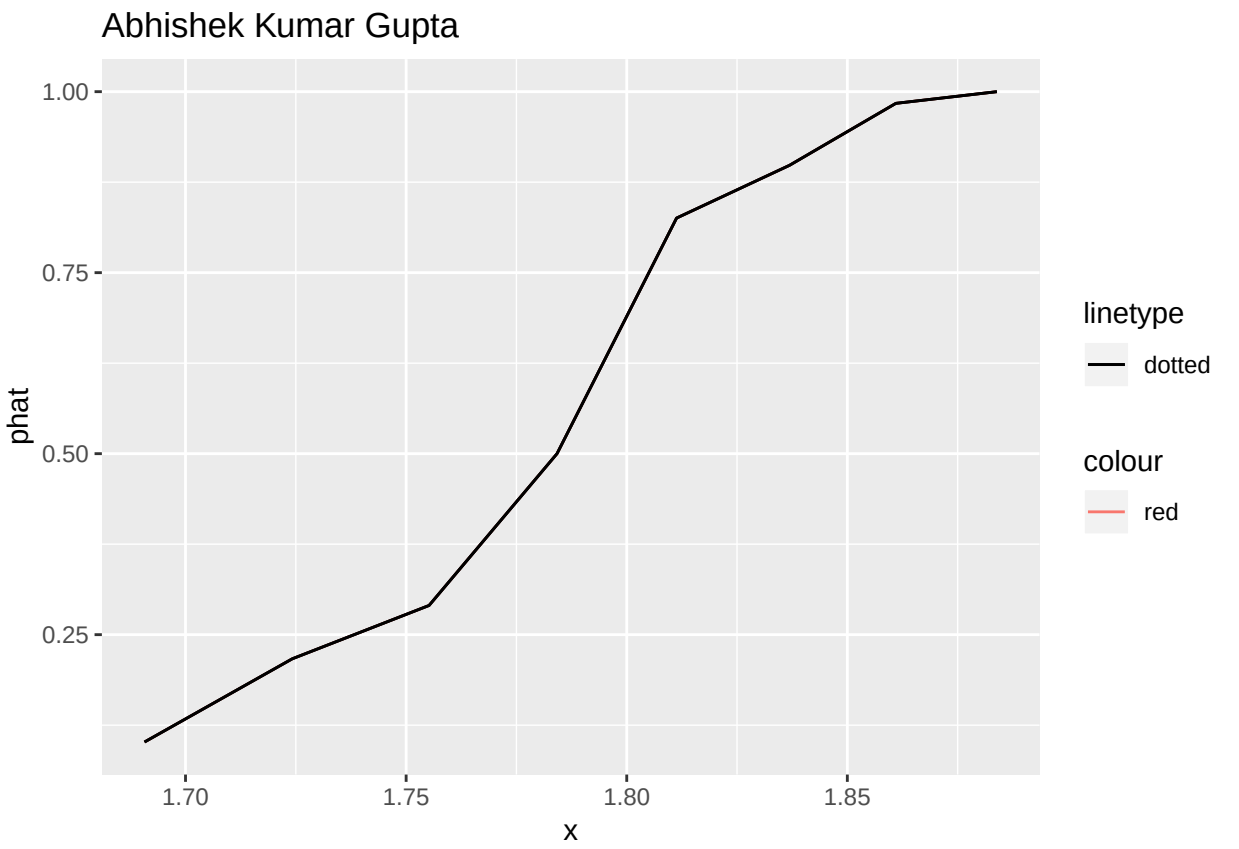
```
update( jagsModel , n.iter=500 )
codaSamples = coda.samples( jagsModel , variable.names=c("beta0", "beta1", "yhat", "phat"), n.iter=5000)
save( codaSamples , file=paste0(fileNameRoot,"Mcmc.Rdata") )

su = summary(codaSamples)
stat = su$statistic
yhat = stat[11:18, 1]
quant = su$quantiles
bci = quant[11:18, c(1,5)]

df = data.frame(list(x = c(1.6907, 1.7242, 1.7552, 1.7842, 1.8113, 1.8369, 1.8610, 1.8839),
                     n = c(59, 60, 62, 56, 63, 59, 62, 60),
                     y = c(6, 13, 18, 28, 52, 53, 61, 60)))

df2 = data.frame(cbind(yhat,bci,x = df$x))

ggplot(data = df2, aes(x)) +geom_line(aes(y = yhat, colour = "red"))+
  geom_line(aes(y = X2.5.,linetype = "dotted"))+
  geom_line(aes(y = X97.5.,linetype = "dotted"))+labs(title ="Abhishek Kumar Gupta")
```




```
phat = stat[3:10, 1]
bci = quant[3:10, c(1,5)]
df3 = data.frame(cbind(phat,bci,x = df$x))
df3
```

```
##              phat     X2.5.    X97.5.      x
## phat[1] 0.1016949 0.1016949 0.1016949 1.6907
## phat[2] 0.2166667 0.2166667 0.2166667 1.7242
## phat[3] 0.2903226 0.2903226 0.2903226 1.7552
## phat[4] 0.5000000 0.5000000 0.5000000 1.7842
## phat[5] 0.8253968 0.8253968 0.8253968 1.8113
## phat[6] 0.8983051 0.8983051 0.8983051 1.8369
## phat[7] 0.9838710 0.9838710 0.9838710 1.8610
## phat[8] 1.0000000 1.0000000 1.0000000 1.8839
```

```
ggplot(data = df3, aes(x)) +geom_line(aes(y = phat, colour = "red"))+
  geom_line(aes(y = X2.5.,linetype = "dotted"))+
  geom_line(aes(y = X97.5.,linetype = "dotted"))+labs(title ="Abhishek Kumar Gupta")
```



3. Now you will need to analyze the Titanic data set. I want you to perform a logistic regression where "Survived" is the response. Please note that this question is open for you to be creative and answer as best you can. Show me what you can do!!

1. Describe the data - that is give a full description of the variables. See https://www.youtube.com/watch?v=49fADBfcDD4&t=3401s for help.

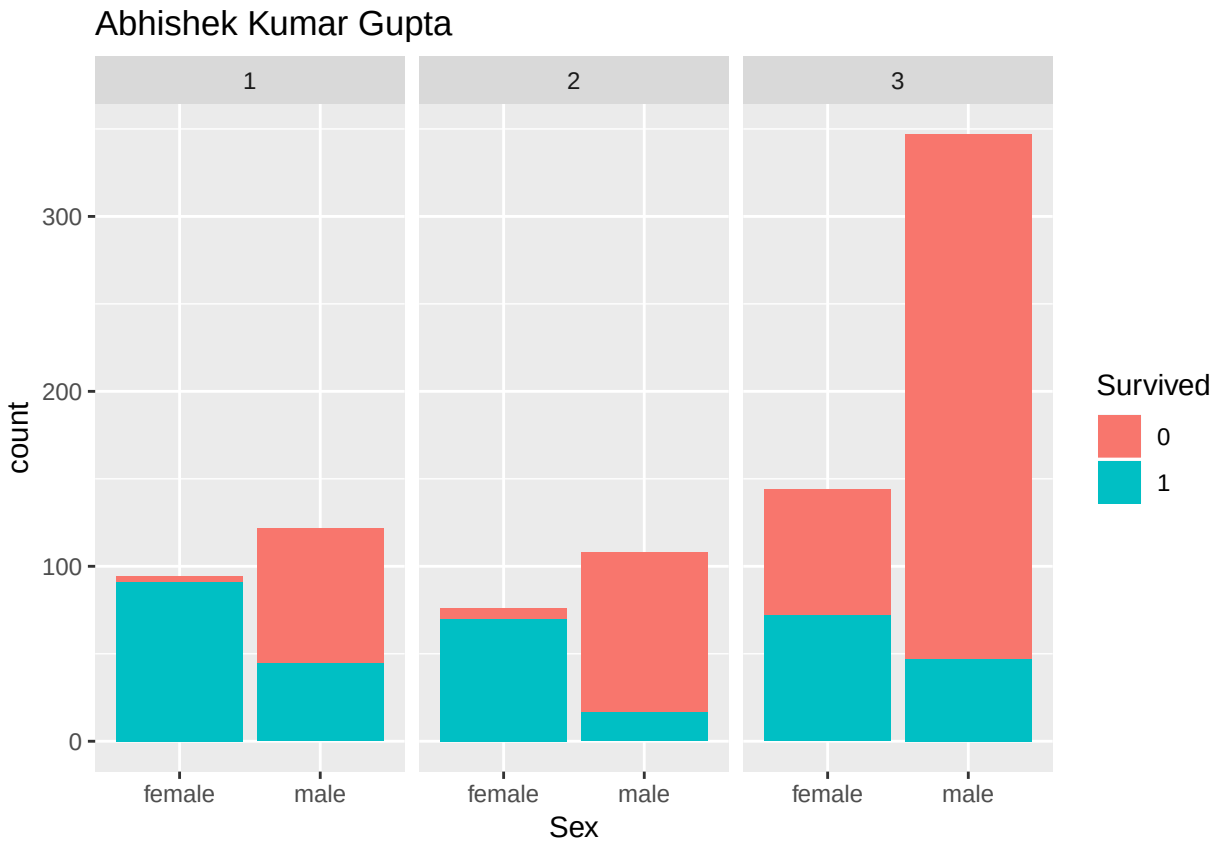
The data consists of 11 variables/predictors/features and 890 rows/dta points for each variables. The description of each variable are:

Survived: binary variably where 1 denotes that passenger survived, 0 indicates died Pclass: indiactes the class of the ticket with which the passenger was travelling which could be either 1 (for first class), 2 (for second class) and 3 (for 3rd class). Name: name of the passenger Sex: whether the passenger is male or female Age: age of the passenger sibsp: number of siblings and/or if travelling with spouse parch: indicates number of children and/or if travelling with paretns too Ticket: gives the ticket number of the passenger fare: fare of the passenger for the travel cabin: cabin assigned embarked: place where the journey started by the passenger

2.) plot the data in at least four useful ways using ggplot. Make sure you describe the plots.

```
titaniDf<- read.csv("titanic.csv")
library(ggplot2)
```

```
titaniDf$Survived  =as.factor(titaniDf$Survived)
titaniDf$Pclass = as.factor(titaniDf$Pclass)
titaniDf$Sex = as.factor(titaniDf$Sex)
titaniDf$Embarked = as.factor(titaniDf$Embarked)
ggplot(data = titaniDf, aes(Sex, fill = Survived))+geom_bar()+facet_wrap(~Pclass)+labs(title ="Abhishek
```

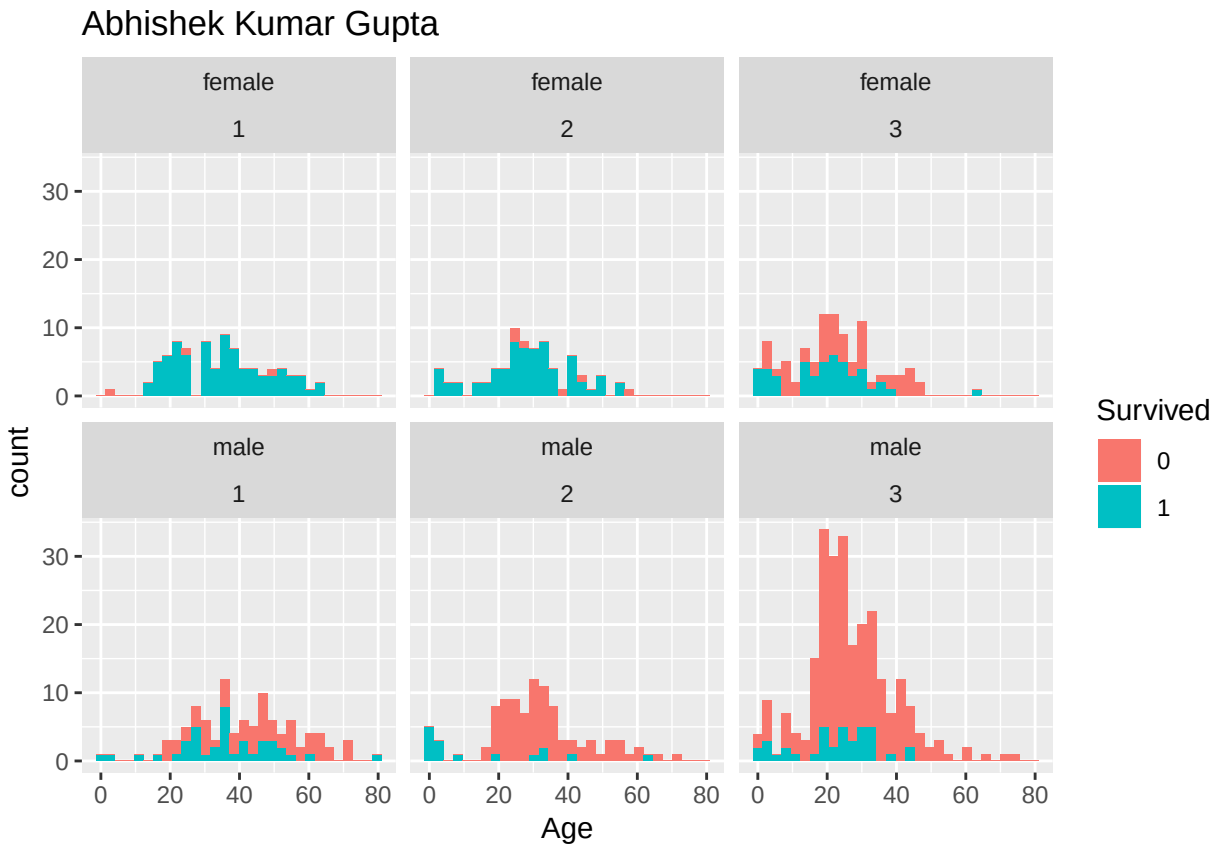


The above bar graph shows the number of people survived for each class. Also, the bars have been color coded with sex m/f. From the plots, we see for class 1 almost all the female survived and proprtion of ale survival is high compared to rest of the class. Similar observaions can be seen for second class. However, we see maximum casualty in thrid class people especially males.

```
ggplot(data = titaniDf, aes(x= Age, fill = Survived)) +
  geom_histogram()+facet_wrap(Sex~Pclass)+labs(title ="Abhishek Kumar Gupta")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

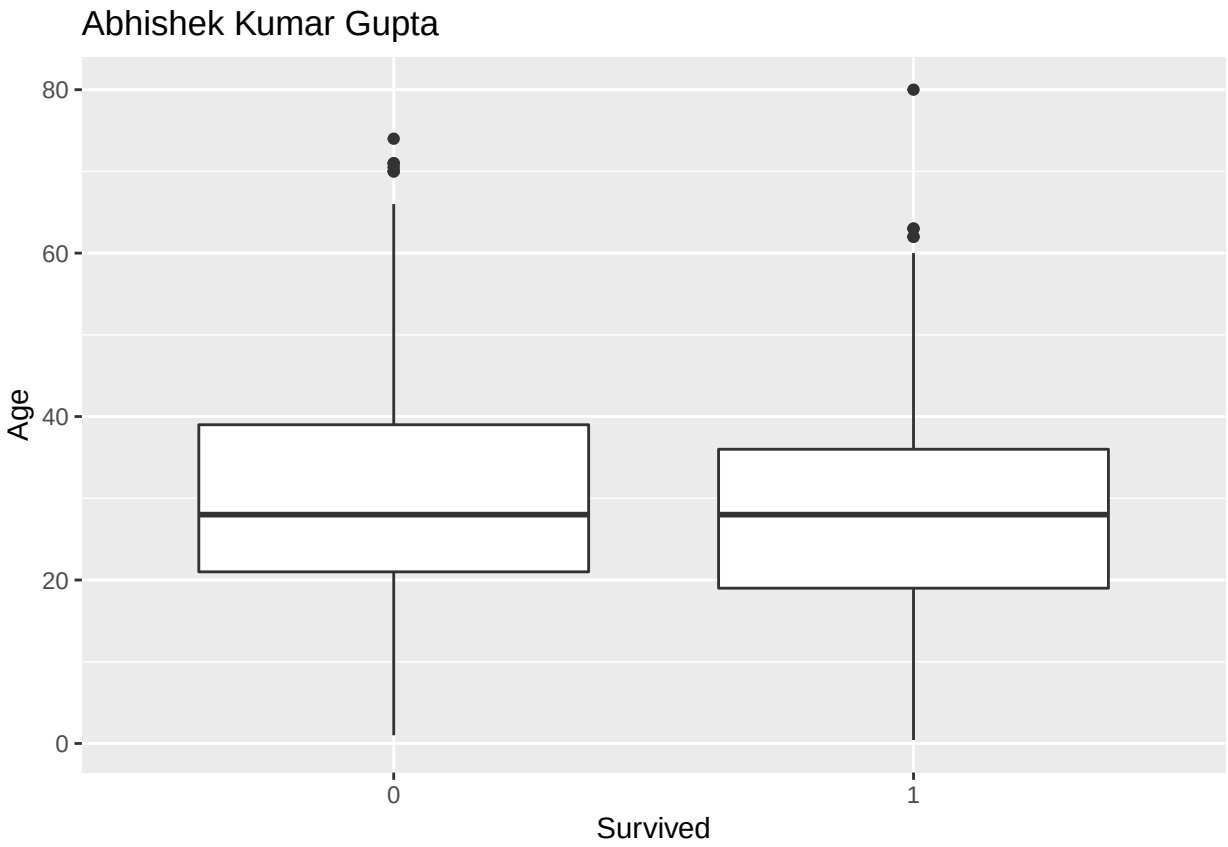
```
## Warning: Removed 177 rows containing non-finite values (stat_bin).
```



The ablove graph shows the histogram for survival count for each sex and exch class of people. Again, we see similar observations as above with high female survival in first class and maximum deaths in third class male.

```
ggplot(data = titaniDf, aes(y=Age,x=Survived)) +
  geom_boxplot()+labs(title ="Abhishek Kumar Gupta")
```

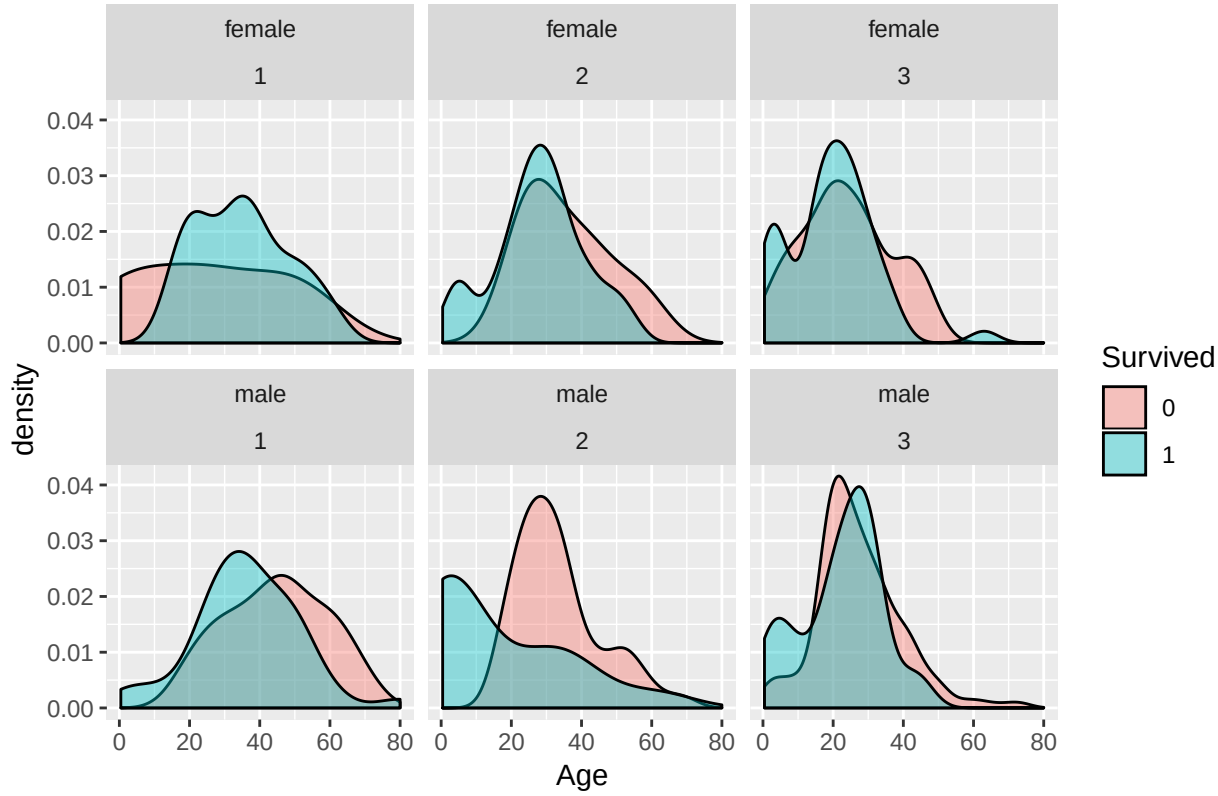
```
## Warning: Removed 177 rows containing non-finite values (stat_boxplot).
```



The above box plot shows the suvival and age variance among the passengers. young poeple were given prefernce as seen by low age in survival boxlot median values.

```
ggplot(data = titaniDf, aes(x=Age, fill = Survived)) +geom_density(alpha=0.4)+
  facet_wrap(Sex~Pclass)+labs(title ="Abhishek Kumar Gupta")
```

```
## Warning: Removed 177 rows containing non-finite values (stat_density).
```



Above is the density plot of age faceted with sex and class.

3. Make a JAGS script to analyze the data using whatever x variables you wish - make different linear predictors and use DIC to choose between the models.

Here, I have used the predictors age, sex and their interaction to model the survival.

```
clglm = glm(Survived ~ Sex + Age + Sex:Age,family = "binomial", data = titaniDf)
summary(clglm)
```

```
##
## Call:
## glm(formula = Survived ~ Sex + Age + Sex:Age, family = "binomial",
##     data = titaniDf)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9401  -0.7136  -0.5883   0.7626   2.2455
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.59380    0.31032   1.913  0.05569 .
## Sexmale     -1.31775    0.40842  -3.226  0.00125 **
## Age          0.01970    0.01057   1.863  0.06240 .
## Sexmale:Age -0.04112    0.01355  -3.034  0.00241 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 964.52  on 713  degrees of freedom
## Residual deviance: 740.40  on 710  degrees of freedom
##   (177 observations deleted due to missingness)
## AIC: 748.4
##
## Number of Fisher Scoring iterations: 4
```

```
mat1=model.matrix(clglm)
mat2=model.frame(clglm)
head(mat1)
```

```
##   (Intercept) Sexmale Age Sexmale:Age
## 1           1       1  22          22
## 2           1       0  38           0
## 3           1       0  26           0
## 4           1       0  35           0
## 5           1       1  35          35
## 7           1       1  54          54
```

```
head(mat2)
```

```
##   Survived    Sex Age
## 1        0   male  22
## 2        1 female  38
## 3        1 female  26
## 4        1 female  35
## 5        0   male  35
## 7        0   male  54
```

```
y = with(mat2, ifelse(Survived == "1", 1,0))
n = length(y)

dataList=list(y = y, x = mat1[, "Age"],Sexm = mat1[,"Sexmale"], Sexmx = mat1[,"Sexmale:Age"] , n = leng
df = data.frame(dataList)
head(df)
```

```
##   y  x Sexm Sexmx   n
## 1 0 22    1    22 714
## 2 1 38    0     0 714
## 3 1 26    0     0 714
## 4 1 35    0     0 714
## 5 0 35    1    35 714
## 7 0 54    1    54 714
```

```
length(mat1[,"Sexmale:Age"])
```

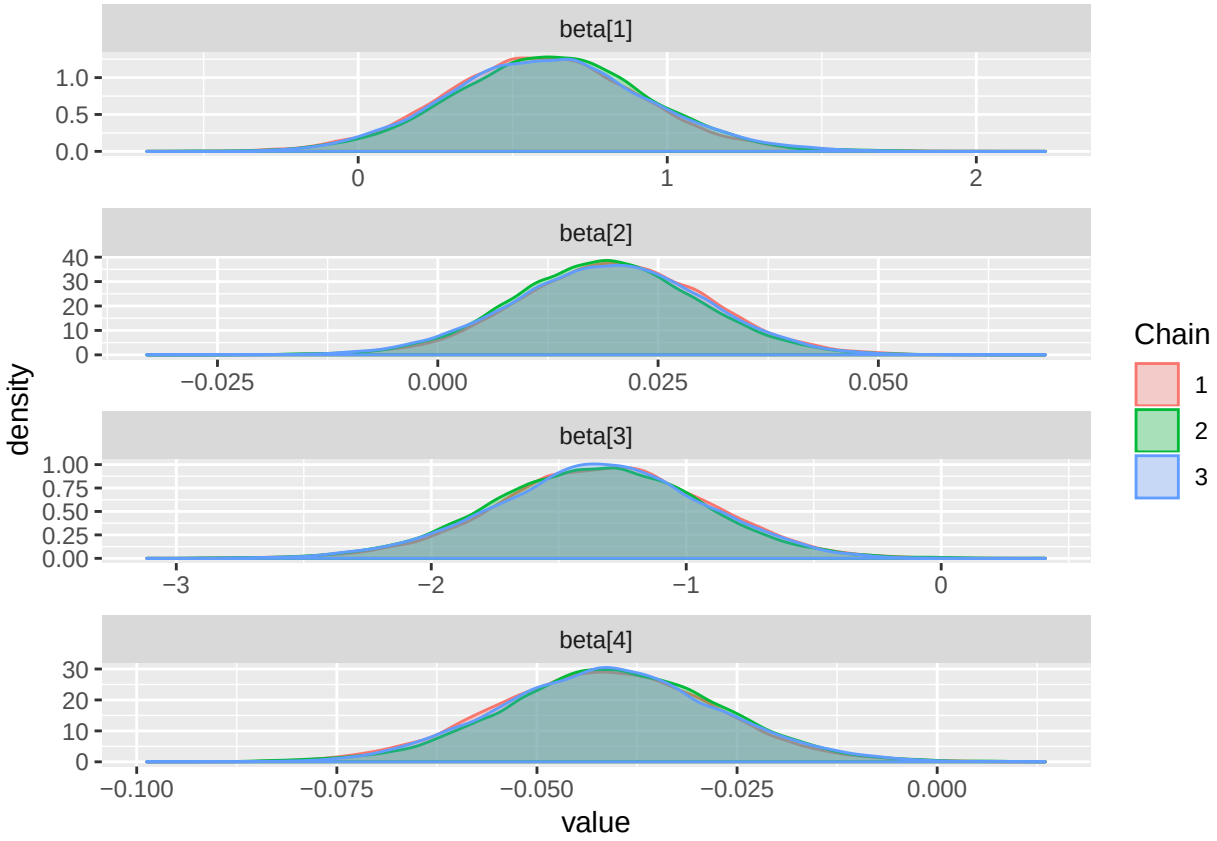
```
## [1] 714
```

```
library(rjags)

#Define the model:
modelString = "
model{
for(i in 1:n){
y[i] ~ dbin(theta[i], 1)
```

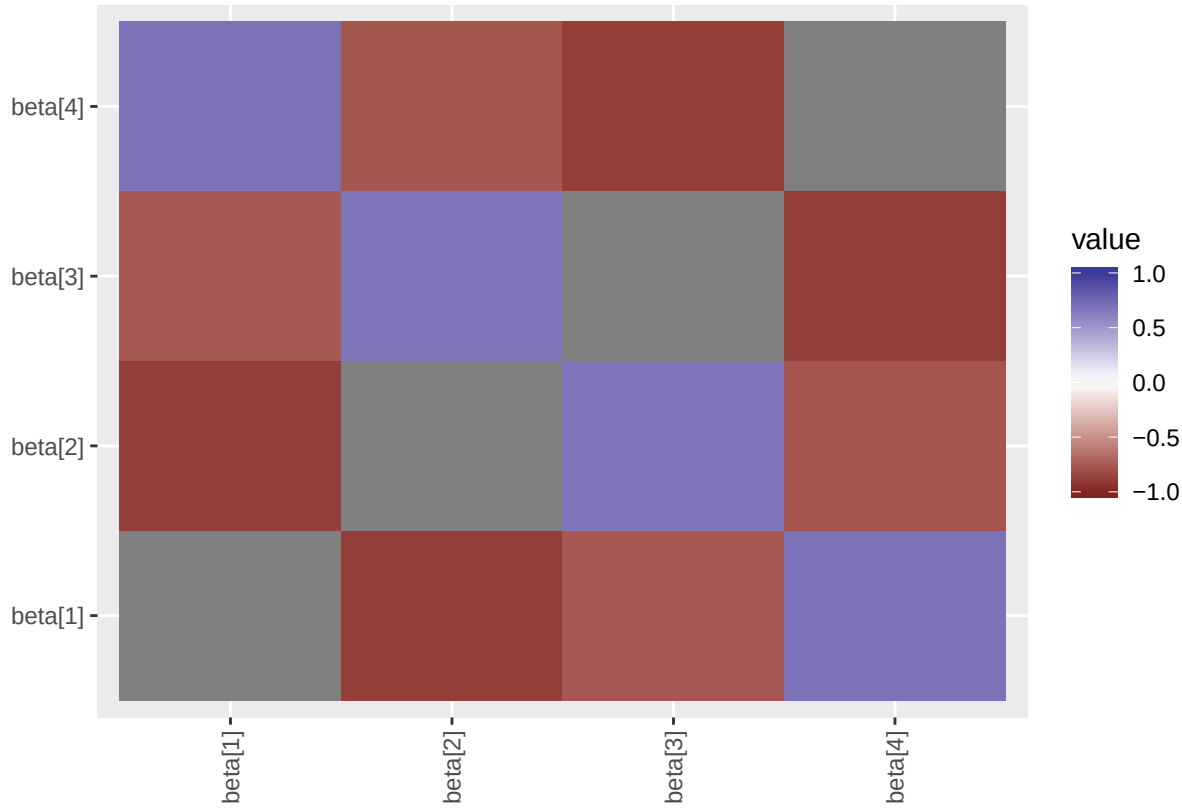
```
logit(theta[i]) <- beta[1] + beta[2]*x[i] + beta[3]*Sexm[i] + beta[4]*Sexmx[i]

}
for(j in 1:4){
beta[j] ~ dnorm(0,1.0E-3)
}
}
"
writeLines( modelString , con="TEMPmodel.txt" )

# close quote for modelStri
#  initsList = list( theta=thetaInit )

initsList = list(beta = c(0.6,0.02,-1.31,-0.04))
# Run the chains:

jagsModel = jags.model( file="TEMPmodel.txt" , data=dataList , inits=initsList ,
                        n.chains=3 , n.adapt=500 )
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 714
##    Unobserved stochastic nodes: 4
##    Total graph size: 3327
##
## Initializing model
```

```
update( jagsModel , n.iter=500 )
codaSamples = coda.samples( jagsModel , variable.names=c("beta"),
                            n.iter=33350 )
save( codaSamples , file=paste0("assignment4","Mcmc.Rdata") )



library(ggmcmc)
s = ggs(codaSamples)
d=ggs_density(s)

print(d)
```




```
cr =  ggs_crosscorrelation(s)
print(cr)
```




```
summary(codaSamples)
```

```
##
## Iterations = 1001:34350
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 33350
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##             Mean      SD  Naive SE Time-series SE
## beta[1]  0.61318 0.31555 9.976e-04      0.0072827
## beta[2]  0.01945 0.01072 3.389e-05      0.0002448
## beta[3] -1.34203 0.41019 1.297e-03      0.0095274
## beta[4] -0.04100 0.01361 4.301e-05      0.0003223
##
## 2. Quantiles for each variable:
##
##              2.5%      25%      50%      75%    97.5%
## beta[1]  0.003605  0.39934  0.60914  0.81889  1.24769
## beta[2] -0.001828  0.01227  0.01948  0.02663  0.04039
## beta[3] -2.167250 -1.61226 -1.33719 -1.06706 -0.55218
## beta[4] -0.067624 -0.05012 -0.04105 -0.03195 -0.01406
```

```
dic.samples(jagsModel, n.iter = 5000)
```

```
## Mean deviance:  744.3
## penalty 3.833
## Penalized deviance: 748.1
```

ANother model without interaction term:

```
clglm = glm(Survived ~ Sex + Age,family = "binomial", data = titaniDf)
summary(clglm)
```

```
##
## Call:
## glm(formula = Survived ~ Sex + Age, family = "binomial", data = titaniDf)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7405  -0.6885  -0.6558   0.7533   1.8989
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.277273   0.230169   5.549 2.87e-08 ***
## Sexmale     -2.465920   0.185384 -13.302  < 2e-16 ***
## Age         -0.005426   0.006310  -0.860     0.39
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 964.52  on 713  degrees of freedom
## Residual deviance: 749.96  on 711  degrees of freedom
##   (177 observations deleted due to missingness)
## AIC: 755.96
##
## Number of Fisher Scoring iterations: 4
```

```
mat1=model.matrix(clglm)
mat2=model.frame(clglm)
head(mat1)
```

```
##   (Intercept) Sexmale Age
## 1           1       1  22
## 2           1       0  38
## 3           1       0  26
## 4           1       0  35
## 5           1       1  35
## 7           1       1  54
```

```
head(mat2)
```

```
##   Survived    Sex Age
## 1        0   male  22
## 2        1 female  38
## 3        1 female  26
## 4        1 female  35
## 5        0   male  35
## 7        0   male  54
```

```
y = with(mat2, ifelse(Survived == "1", 1,0))
n = length(y)

dataList=list(y = y, x = mat1[, "Age"],Sexm = mat1[,"Sexmale"],n = length(y))
df = data.frame(dataList)
head(df)
```

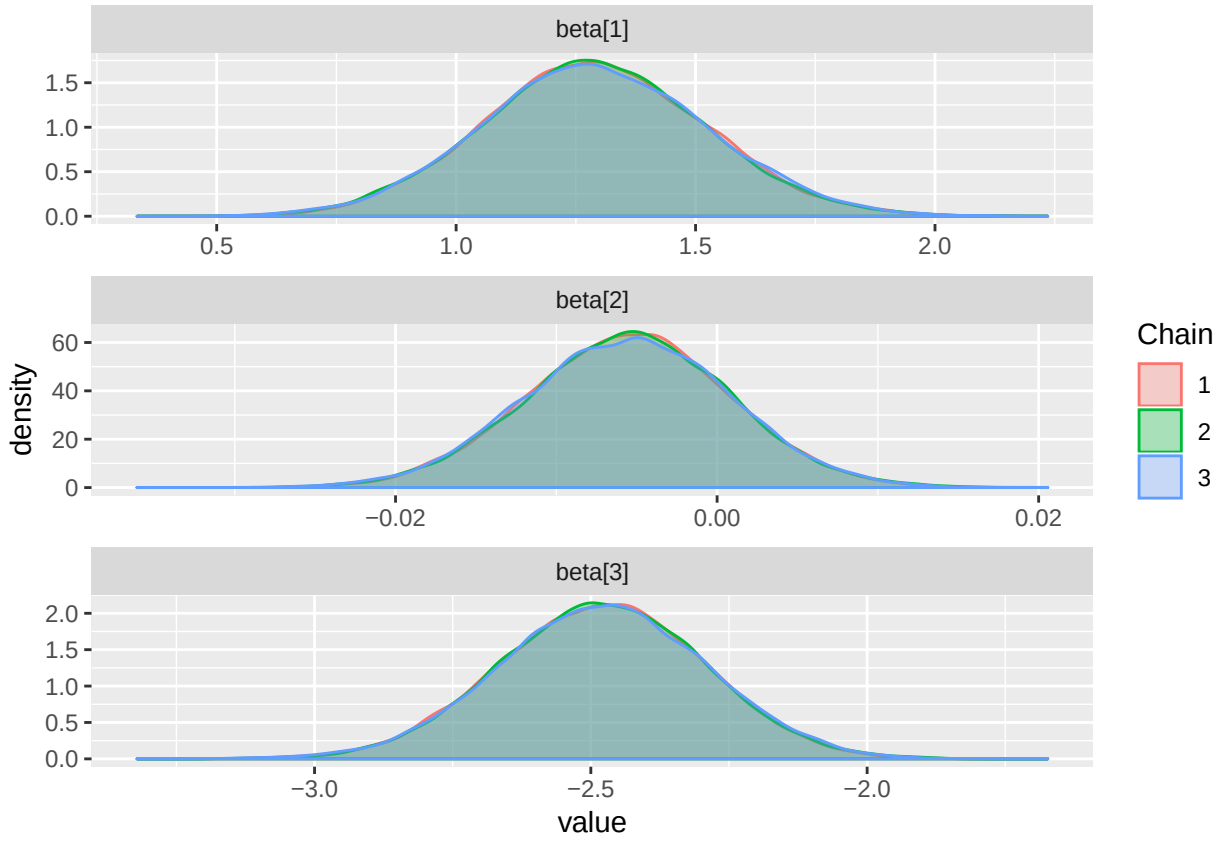
```
##   y  x Sexm   n
## 1 0 22    1 714
## 2 1 38    0 714
## 3 1 26    0 714
## 4 1 35    0 714
## 5 0 35    1 714
## 7 0 54    1 714
```

```
library(rjags)

#Define the model:
modelString = "
model{
for(i in 1:n){
y[i] ~ dbin(theta[i], 1)
logit(theta[i]) <- beta[1] + beta[2]*x[i] + beta[3]*Sexm[i]

}
for(j in 1:3){
beta[j] ~ dnorm(0,1.0E-3)
}
}
"
writeLines( modelString , con="TEMPmodel.txt" )
initsList = list(beta = c(1.27,-0.005,-2.46))
# Run the chains:
jagsModel = jags.model( file="TEMPmodel.txt" , data=dataList , inits=initsList ,
                        n.chains=3 , n.adapt=500 )
```

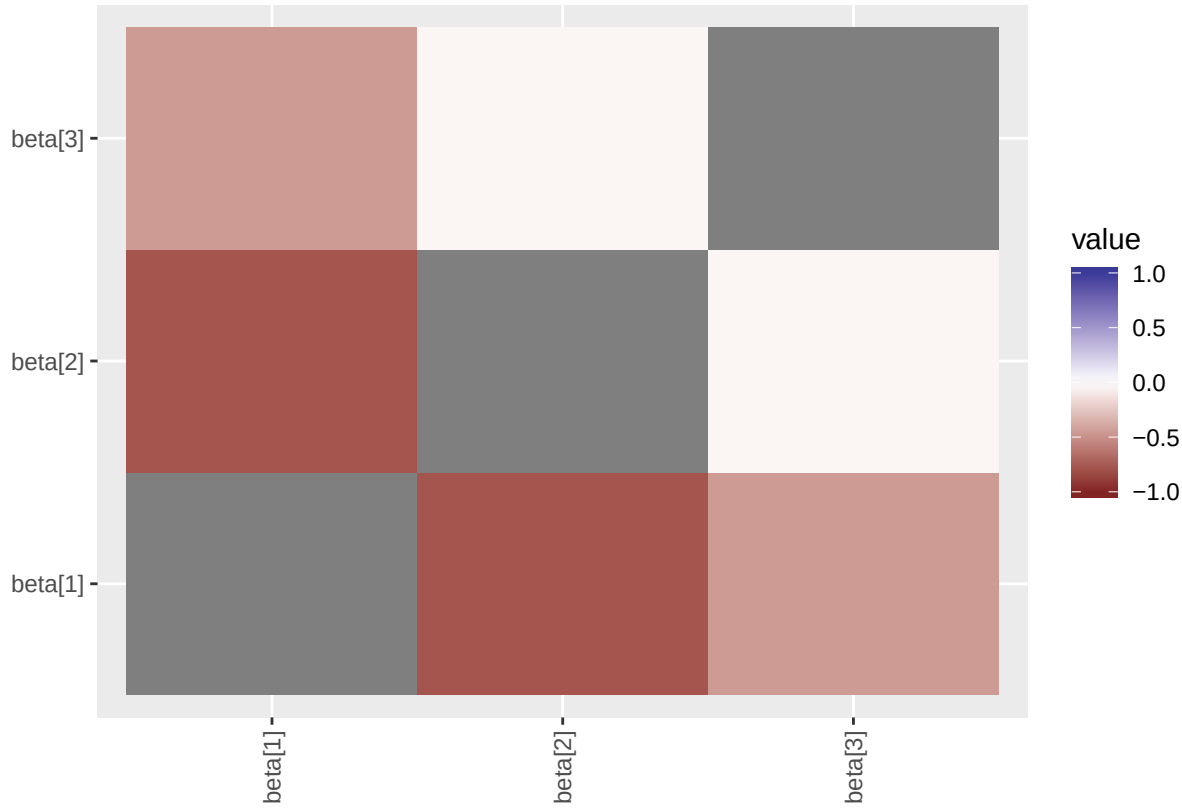
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 714
##    Unobserved stochastic nodes: 3
##    Total graph size: 2529
##
## Initializing model
```

```
update( jagsModel , n.iter=500 )
codaSamples = coda.samples( jagsModel , variable.names=c("beta"),
                            n.iter=33350 )
save( codaSamples , file=paste0("assignment4","Mcmc.Rdata") )



library(ggmcmc)
s = ggs(codaSamples)
```

```
d=ggs_density(s)

print(d)
```




```
cr =  ggs_crosscorrelation(s)
print(cr)
```




```
summary(codaSamples)
```

```
##
## Iterations = 1001:34350
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 33350
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean      SD  Naive SE Time-series SE
## beta[1]  1.288659 0.23279 7.360e-04      0.0029905
## beta[2] -0.005547 0.00636 2.011e-05      0.0000766
## beta[3] -2.480618 0.18692 5.909e-04      0.0014514
##
## 2. Quantiles for each variable:
##
##            2.5%       25%       50%       75%     97.5%
## beta[1]  0.8379  1.132136  1.285605  1.444215  1.752478
## beta[2] -0.0181 -0.009803 -0.005469 -0.001249  0.006905
## beta[3] -2.8478 -2.605960 -2.479158 -2.353475 -2.117511
```

```
dic.samples(jagsModel, n.iter = 5000)
```

```
## Mean deviance:  752.9
## penalty 2.957
```

```
## Penalized deviance: 755.9
```

From the above two models, we clearly see DIC values with interaction terms are lower compacre to without interaction. SInce lower DIC values corresponds to the better models. first model with interaction term is preferred.

4. Make conclusions about the probability of survival based on different combinations of independent variables.

Based on the above model the probability of survival is given as :

$$logit(survival) = 0.59 + 0.02 * age - 1.32 * Sexmale - 0.04 * Sexmale * Age$$

case1, if male=1

$$logit(survival) = 0.59 + 0.02 * age - 1.32 - 0.04 * Age$$

or,

$$logit(survival) = -0.73 - 0.02 * Age$$

case2, if male=0, i.e, for female

$$logit(survival) = 0.59 + 0.02 * age$$

Therefore, for females unit increase in age leads to 0.02 increase in logs of odds of survival. However for malesunit increase in age leads to decrease in logs of odds of survival by 0.02.