Full bayesian analysis on Bank marketing dataset

Name: Vamsi Krishna Varma Gunturi

Metricola ID: 1794653

TABLE OF CONTENTS

Abstract

The primary goal of this project is to perform Bayesian analysis on bank marketing dataset (data set taken from UCLA) with Markov chain Monte Carlo (MCMC) simulation method.

Initially we did some data exploration to find out any visible patterns which we can use while developing Bayesian models using JAGS

The following models are utilized in this analysis: 1) Logit model with all the parameters 2) Logit model with reduced parameters 3) Probit model with reduced parameters

And then we tried to understand how above models fare on the given dataset using different methods such as HDP interval, DIC, Raftery and Lewis diagnostic, Heildelberger and Welchs convergence diagnostics etc..,

Then, we tried to compare the logit and probit models with Frequentist approach to further understand how our Bayesian models are performing.

To conclude our analysis we tried to verify how our models are able to recover features in the unseen data (test data) after training our model with train dataset.

We concluded our analysis in the conclusion section.

Introduction

Bayesian inference revolves around making use of Bayes theorem (Bayes, 1763) (Laplace, 1812) (Jeffrys, 1939) to update prior belief

$$\mathrm{P}(\theta|\mathrm{x}) = \frac{\mathrm{P}(\mathrm{x}|\theta)\mathrm{P}(\theta)}{\mathrm{P}(\mathrm{x})}$$

$$P(\theta \mid x) = \frac{P(x \mid \theta)P(\theta)}{P(x)}$$

and although denominator looks naive it is impossible to integrate for non-trivial problems hence through the years Monte Carlo Markov Chain methods were developed.

Markov Chain is a stochastic process that satisfies Markov property (Markov, 1954) aka memorylessness. In oversimplification, Markov properties are satisfied if the future process is predictable solely based on current state.

Monte Carlo method (Metropolis&Ulam, 1949) is class of algorithms that obtains results via random sampling and solves three broad problem classes: optimization, numerical integration and generating draws from posterior distribution.

In particular, ability to sample from posterior distribution is crucial for Bayesian analysis although direct sampling is not usually possible following algorithms have been developed: Metropolis-Hastings(MH), Gibbs, Hamiltonian, No-U-Turn Sampler.

Current state of the art tools for implementing probabilistic programming include: winBUGS, jags, pymc3, Stan and Edward. Differences come from underlying platform, syntax and implemented samplers.

Models presented below are implemented in R programming language via rjags library hence models are developed for jags which uses both Gibbs and MH sampler depending on the problem.

In reference to this particular project, the **bank marketing** dataset is used. This dataset contains the details of the targetted clients of a basic for a marketing campaign.

The variables of this dataset include age, job, martial_status, education, housing, loan, duration etc.., as input variables and variable y (which signifies whether a costomer has subscribed to a fixed term deposit) as output variable.

Therefore as an example that probability of y being true(1) or false(0) is conditioned on duration of the telephone call.

that is :

$$\mathrm{P(y|duration)}$$

$$P(y \mid duration)$$

This can be calculated with the formula

$$\mathrm{P(y|duration)} = \frac{\mathrm{P(y \cap duration)}}{\mathrm{P(duration)}} = \frac{\mathrm{P(duration|y)} * \mathrm{P(y)}}{\mathrm{P(duration)}}$$

$$P(y \mid duration) = \frac{P(y \cap duration)}{P(duration)} = \frac{P(duration \mid y) * P(y)}{P(duration)}$$

With Bayesian Inference, the posterior distribution is derived from the prior distribution and the likelihood function derived from a statistical model for the observed data.

That is;

$$\mathrm{posterior\ distribution} = \mathrm{prior\ distribution} * \mathrm{likelihood\ function}$$

Defining the Hypothesis for Bayesian analysis

We want to define hypothesis as “Marketing call duration is a major indicator whether the client subscribes to a term deposit. In other words if the client call duration is more then there is a high probability that he is interested in taking the term deposit”

Dataset description

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be (or not) subscribed.

There are two datasets:

1. bank-full.csv with all examples, ordered by date (from May 2008 to November 2010).
2. bank.csv with 10% of the examples (4521), randomly selected from bank-full.csv.

The classification goal is to predict if the client will subscribe a term deposit (variable y).

Number of Instances: 45211 for bank-full.csv (4521 for bank.csv)

Number of Attributes: 16 + output attribute.

Attribute information:

Note: For more information, read [Moro et al., 2011].

Input variables:

1. age (numeric)
2. job : type of job (categorical: “admin.”,“unknown”,“unemployed”,“management”,“housemaid”,“entrepreneur”,“student”, “blue-collar”,“self-employed”,“retired”,“technician”,“services”)
3. marital : marital status (categorical: “married”,“divorced”,“single”; note: “divorced” means divorced or widowed)
4. education (categorical: “unknown”,“secondary”,“primary”,“tertiary”)
5. default: has credit in default? (binary: “yes”,“no”)
6. balance: average yearly balance, in euros (numeric)
7. housing: has housing loan? (binary: “yes”,“no”)
8. loan: has personal loan? (binary: “yes”,“no”) # related with the last contact of the current campaign:
9. contact: contact communication type (categorical: “unknown”,“telephone”,“cellular”)
10. day: last contact day of the month (numeric)
11. month: last contact month of year (categorical: “jan”, “feb”, “mar”, …, “nov”, “dec”)
12. duration: last contact duration, in seconds (numeric) # other attributes:
13. campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
14. pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)
15. previous: number of contacts performed before this campaign and for this client (numeric)
16. poutcome: outcome of the previous marketing campaign (categorical: “unknown”,“other”,“failure”,“success”)

Output variable (desired target):

17. y - has the client subscribed a term deposit? (binary: “yes”,“no”)

```
# Loading the original dataset
# Train dataset
bank <- read.csv(file = "bank-full.csv", sep=";", header = TRUE, stringsAsFactors=FALSE)
bank_factor <- read.csv(file = "bank-full.csv", sep=";", header = TRUE)
# Test dataset
bank_test <- read.csv(file = "bank.csv", sep=";", header = TRUE, stringsAsFactors=FALSE)

bank_test_factor <- read.csv(file = "bank.csv", sep=";", header = TRUE)

# Check the dimension of datasets
dim(bank)[1]
```

```
## [1] 45211
```

```
dim(bank_test)[1]
```

```
## [1] 4521
```

```
# Structure of datasets
head(bank)
```

```
##   age          job marital education default balance housing loan contact
## 1  58   management married  tertiary      no    2143     yes   no unknown
## 2  44   technician  single secondary      no      29     yes   no unknown
## 3  33 entrepreneur married secondary      no       2     yes  yes unknown
## 4  47  blue-collar married   unknown      no    1506     yes   no unknown
## 5  33      unknown  single   unknown      no       1      no   no unknown
## 6  35   management married  tertiary      no     231     yes   no unknown
##   day month duration campaign pdays previous poutcome  y
## 1   5   may      261        1    -1        0  unknown no
## 2   5   may      151        1    -1        0  unknown no
## 3   5   may       76        1    -1        0  unknown no
## 4   5   may       92        1    -1        0  unknown no
## 5   5   may      198        1    -1        0  unknown no
## 6   5   may      139        1    -1        0  unknown no
```

```
head(bank_test)
```

```
##   age         job marital education default balance housing loan  contact
## 1  30  unemployed married   primary      no    1787      no   no cellular
## 2  33    services married secondary      no    4789     yes  yes cellular
## 3  35  management  single  tertiary      no    1350     yes   no cellular
## 4  30  management married  tertiary      no    1476     yes  yes  unknown
## 5  59 blue-collar married secondary      no       0     yes   no  unknown
## 6  35  management  single  tertiary      no     747      no   no cellular
##   day month duration campaign pdays previous poutcome  y
## 1  19   oct       79        1    -1        0  unknown no
## 2  11   may      220        1   339        4  failure no
## 3  16   apr      185        1   330        1  failure no
## 4   3   jun      199        4    -1        0  unknown no
## 5   5   may      226        1    -1        0  unknown no
## 6  23   feb      141        2   176        3  failure no
```

So following attributes needs factorization, job, martial, education, default, housing, loan, contact, month, poutcome, y

```
# Sub setting the data set for our analysis
# Train set and test set with 1500 observations
sample_bank_big = sample(1:dim(bank)[1], 4500, replace=TRUE)
sample_bank = sample(1:dim(bank)[1], 1500, replace=TRUE)
sample_bank_test = sample(1:dim(bank_test)[1], 1500, replace=TRUE)

bank_full = bank
bank_test_full = bank_test

bank_big = bank[sample_bank_big,]
bank = bank[sample_bank,]
bank_test = bank_test[sample_bank_test,]
dim(bank)
```

```
## [1] 1500   17
```

```
dim(bank_test)
```

```
## [1] 1500   17
```

```
dim(bank_big)
```

```
## [1] 4500   17
```

Exploring the dataset:

Summarise the dataset

```
# Summary
summary(bank_factor)
```

```
##       age                 job           marital          education
##  Min.   :18.00   blue-collar:9732   divorced: 5207   primary  : 6851
##  1st Qu.:33.00   management :9458   married :27214   secondary:23202
##  Median :39.00   technician :7597   single  :12790   tertiary :13301
##  Mean   :40.94   admin.     :5171                    unknown  : 1857
##  3rd Qu.:48.00   services   :4154
##  Max.   :95.00   retired    :2264
##                  (Other)    :6835
##  default        balance       housing      loan            contact
##  no :44396   Min.   : -8019   no :20081   no :37967   cellular :29285
##  yes:  815   1st Qu.:    72   yes:25130   yes: 7244   telephone: 2906
##              Median :   448                           unknown  :13020
##              Mean   :  1362
##              3rd Qu.:  1428
##              Max.   :102127
##
##       day            month          duration         campaign
##  Min.   : 1.00   may    :13766   Min.   :   0.0   Min.   : 1.000
##  1st Qu.: 8.00   jul    : 6895   1st Qu.: 103.0   1st Qu.: 1.000
##  Median :16.00   aug    : 6247   Median : 180.0   Median : 2.000
##  Mean   :15.81   jun    : 5341   Mean   : 258.2   Mean   : 2.764
##  3rd Qu.:21.00   nov    : 3970   3rd Qu.: 319.0   3rd Qu.: 3.000
##  Max.   :31.00   apr    : 2932   Max.   :4918.0   Max.   :63.000
##                  (Other): 6060
##      pdays          previous           poutcome       y
##  Min.   : -1.0   Min.   :  0.0000   failure: 4901   no :39922
##  1st Qu.: -1.0   1st Qu.:  0.0000   other  : 1840   yes: 5289
##  Median : -1.0   Median :  0.0000   success: 1511
##  Mean   : 40.2   Mean   :  0.5803   unknown:36959
##  3rd Qu.: -1.0   3rd Qu.:  0.0000
##  Max.   :871.0   Max.   :275.0000
##
```

```
summary(bank_test_factor)
```

```
##       age                 job          marital         education
##  Min.   :19.00   management :969   divorced: 528   primary  : 678
##  1st Qu.:33.00   blue-collar:946   married :2797   secondary:2306
##  Median :39.00   technician :768   single  :1196   tertiary :1350
##  Mean   :41.17   admin.     :478                   unknown  : 187
##  3rd Qu.:49.00   services   :417
##  Max.   :87.00   retired    :230
##                  (Other)    :713
##  default       balance      housing     loan           contact
##  no :4445   Min.   :-3313   no :1962   no :3830   cellular :2896
##  yes:  76   1st Qu.:   69   yes:2559   yes: 691   telephone: 301
##             Median :  444                         unknown  :1324
##             Mean   : 1423
##             3rd Qu.: 1480
##             Max.   :71188
##
##       day            month         duration       campaign
##  Min.   : 1.00   may    :1398   Min.   :   4   Min.   : 1.000
##  1st Qu.: 9.00   jul    : 706   1st Qu.: 104   1st Qu.: 1.000
##  Median :16.00   aug    : 633   Median : 185   Median : 2.000
##  Mean   :15.92   jun    : 531   Mean   : 264   Mean   : 2.794
##  3rd Qu.:21.00   nov    : 389   3rd Qu.: 329   3rd Qu.: 3.000
##  Max.   :31.00   apr    : 293   Max.   :3025   Max.   :50.000
##                  (Other): 571
##      pdays           previous          poutcome      y
##  Min.   : -1.00   Min.   : 0.0000   failure: 490   no :4000
##  1st Qu.: -1.00   1st Qu.: 0.0000   other  : 197   yes: 521
##  Median : -1.00   Median : 0.0000   success: 129
##  Mean   : 39.77   Mean   : 0.5426   unknown:3705
##  3rd Qu.: -1.00   3rd Qu.: 0.0000
##  Max.   :871.00   Max.   :25.0000
##
```

```
# Count the number of null values
sum(is.na(bank))
```

```
## [1] 0
```

```
sum(is.na(bank_test))
```

```
## [1] 0
```

so the dataset is free from null values

Label encoding for character features

```
# Perform label encoding for bank and bank_test datasets
names = colnames(bank)
for (f in names) {
    if (class(bank[[f]]) == 'character') {
      # We reach here if the current feature is character
      # print(f)
      levels <- sort(unique(c(bank[[f]], bank_test[[f]])))
      bank[[f]] <- as.integer(factor(bank[[f]], levels = levels))
      bank_test[[f]] <- as.integer(factor(bank_test[[f]], levels = levels))
      bank_big[[f]] <- as.integer(factor(bank_big[[f]], levels = levels))

      bank_full[[f]] <- as.integer(factor(bank_full[[f]], levels = levels))

      bank_test_full[[f]] <- as.integer(factor(bank_test_full[[f]], levels = levels))

    }
}

head(bank)
```

```
##       age job marital education default balance housing loan contact day
## 21564  45   1       2         2       1     740       1    2       1  19
## 40290  27   1       3         3       1     138       1    1       1  15
## 29544  53   2       2         1       1     751       2    1       1   3
## 19001  46   5       2         3       1    1720       1    1       1   5
## 9772   40   8       2         2       1      37       1    1       3   9
## 12417  49   1       1         2       1    2169       2    1       3  30
##       month duration campaign pdays previous poutcome y
## 21564     2      109        2    -1        0        4 1
## 40290     7       89        2    -1        0        4 2
## 29544     4     1528        2    -1        0        4 2
## 19001     2      250        2    -1        0        4 1
## 9772      7      356        5    -1        0        4 1
## 12417     7       54        3    -1        0        4 1
```

```
head(bank_test)
```

```
##      age job marital education default balance housing loan contact day
## 4048  75   6       2         2       1   26452       1    1       2  15
## 2470  57   6       2         1       1    1623       2    2       1  18
## 2452  57   1       2         2       1    3927       1    1       1  13
## 349   44   1       2         2       1       0       2    2       3  16
## 3105  33   1       3         2       1    2557       2    1       1  18
## 1712  32  10       3         3       1    2185       1    1       1  20
##      month duration campaign pdays previous poutcome y
## 4048     6      219        2    -1        0        4 1
## 2470     6      246        1    -1        0        4 1
## 2452    11       61        1    -1        0        4 1
## 349      9      159        1    -1        0        4 1
## 3105     9      186        1    -1        0        4 1
## 1712    10      354        2    -1        0        4 1
```

Preprocessing the output variable

Important: Run this only once

```
# Preprocessing the output variable so that is is suitable for JAGS
bank_test$y = bank_test$y - 1
bank$y = bank$y - 1
bank_big$y = bank_big$y - 1

bank_full$y = bank_full$y - 1
bank_test_full$y = bank_test_full$y - 1
```

Finding correlation of all the features with output feature (y)

```
#finding correlation with respect to y

# For train dataset
correlation_train = cor(bank, bank$y)
correlation_train
```

```
##                 [,1]
## age         0.07160137
## job        -0.02141999
## marital     0.01198600
## education   0.04280540
## default    -0.05086224
## balance     0.08548470
## housing    -0.14787955
## loan       -0.07753915
## contact    -0.15544806
## day        -0.06517424
## month      -0.01183364
## duration    0.48593708
## campaign   -0.06447669
## pdays       0.08250237
## previous    0.12557808
## poutcome   -0.05717770
## y           1.00000000
```

```
# For test dataset
correlation_test = cor(bank_test, bank_test$y)
correlation_test
```

```
##                  [,1]
## age       -0.042481045
## job        0.010222835
## marital    0.034974350
## education  0.033617561
## default   -0.024322331
## balance   -0.005944403
## housing   -0.071779733
## loan      -0.089269436
## contact   -0.115143527
## day        0.004107698
## month     -0.008655462
## duration   0.453991293
## campaign  -0.061652179
## pdays      0.083057237
## previous   0.132636791
## poutcome  -0.049356412
## y          1.000000000
```

As we can see from both train and test datasets, the output variable(y) is highly correlated with the duration feature which indicates the duration of telephone call

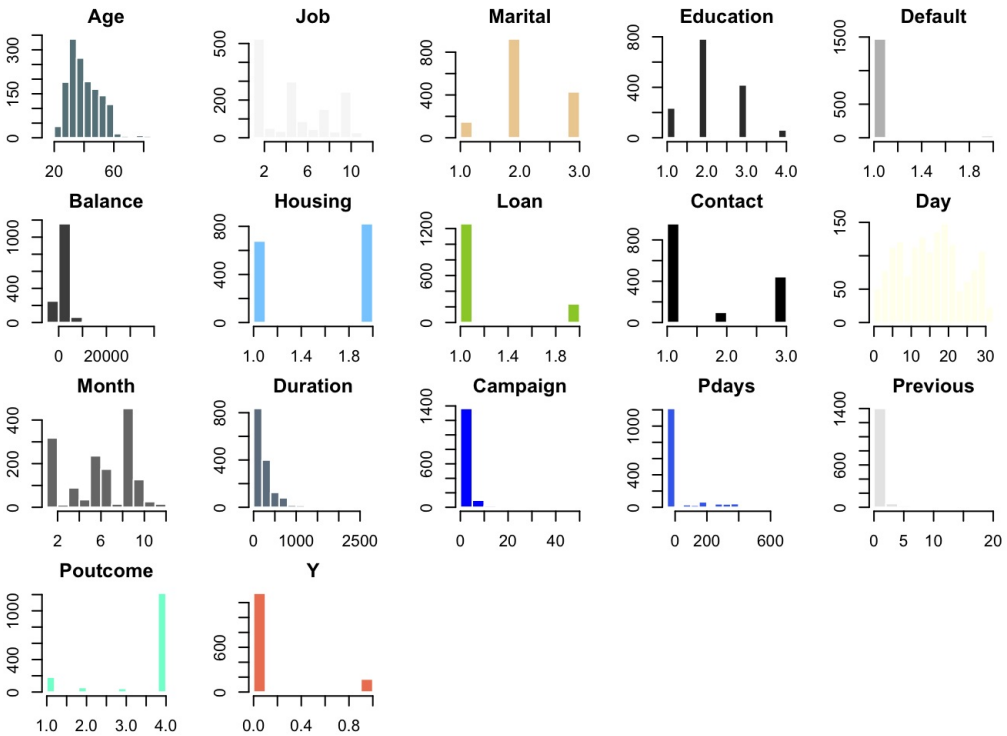
HISTOGRAMS SHOWING THE DISTRIBUTIONS OF EACH VARIABLE

```
# This is for randomly generating colors for histogram plots
sample(1:657, length(colnames(bank)), replace=TRUE)
```

```
##  [1] 310 295 295  81 552 591 611 481 148 132 400 116 196 335 618 306  56
```

```
# Function for captilizing a text
simpleCap <- function(x) {
  s <- strsplit(x, " ")[[1]]
  paste(toupper(substring(s, 1,1)), substring(s, 2),
      sep="", collapse=" ")
}
```

```
names = colnames(bank)
index = 1
colour_list = colors()
par(mar=c(2,2,2,2))
par(mfrow=c(4,5))
sample_colors = sample(1:657, length(colnames(bank)), replace=TRUE)
for (f in names) {
    hist(bank[[f]],xlab=simpleCap(f),ylab='Density', col = colour_list[sample_colors[index]], border = "white",ma
in=simpleCap(f))
  index = index + 1
}
```

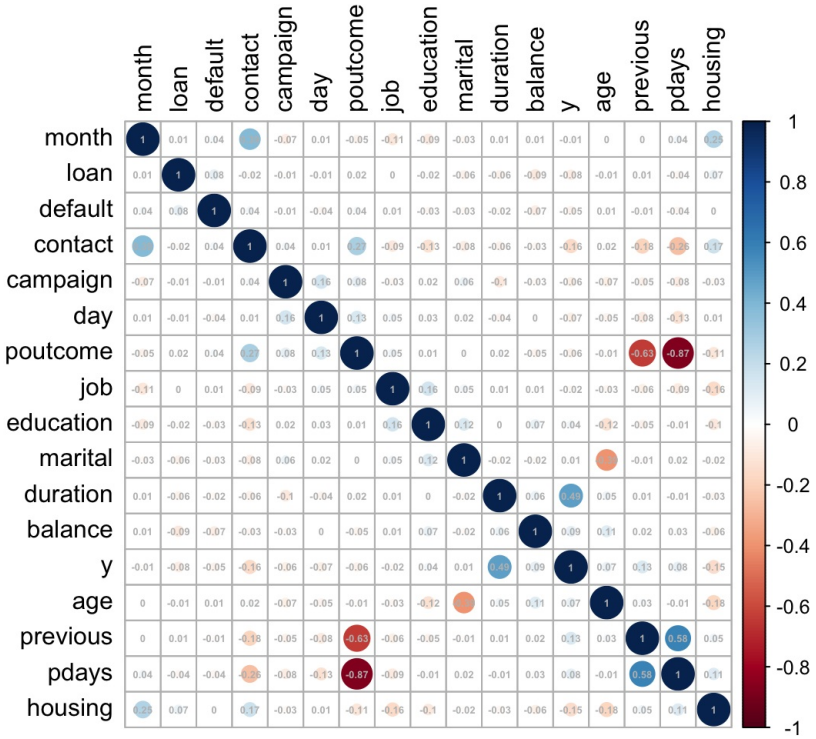


CORRELATION AMONG THE VARIABLES

```
#install.packages("corrplot")
library(corrplot,quietly = TRUE)
```

```
## corrplot 0.84 loaded
```

```
bankdata_Correlation = cor(bank)
corrplot(bankdata_Correlation, method = "circle", addCoef.col="grey", order = "AOE",number.cex= 7/ncol(bankdata_C
orrelation), tl.col = "black")
```



Further analysis with 2 features at a time:

housing vs y

```
table(bank_factor$housing,bank_factor$y)
```

```
##
##          no   yes
##   no  16727  3354
##   yes 23195  1935
```

By above table it is clear that people who actually doesnot own a house a more likely to take the term deposit plan because there is less montly overhead in terms of house mortgage if a person doesnot own a house making him more likely to take the term deposit

loan vs y

```
table(bank_factor$loan,bank_factor$y)
```

```
##
##          no   yes
##   no  33162  4805
##   yes  6760   484
```

So people who doesn't have a loan are more likely to take the term deposit plan as they will be having less montly overhead when compared to people who already have a car or house loan

contact vs y

```
table(bank_factor$contact,bank_factor$y)
```

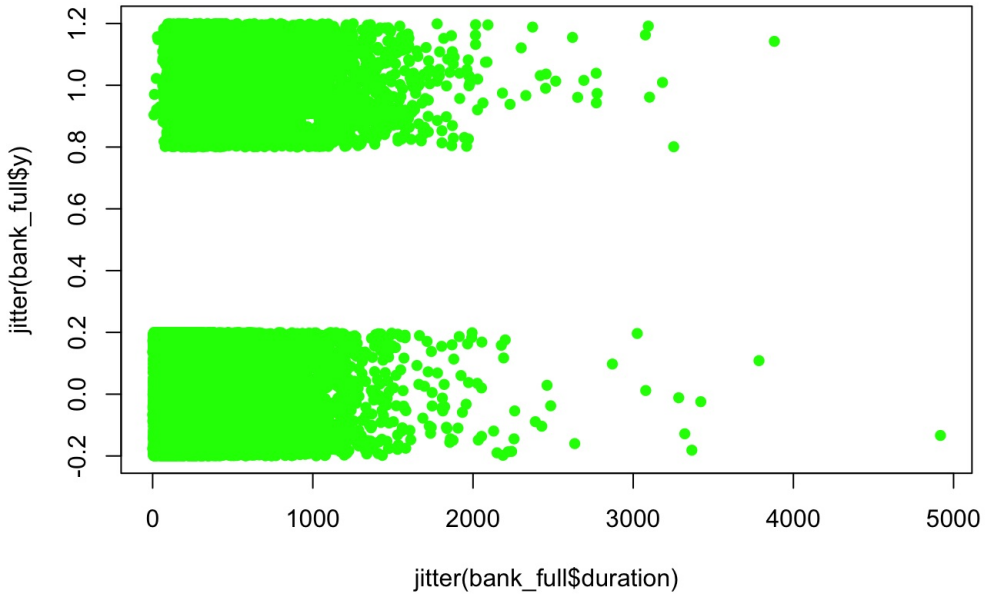
```
##
##               no   yes
##   cellular  24916  4369
##   telephone  2516   390
##   unknown   12490   530
```

so people who own a cellular phone are 5 times more likely to take the term deposit plan marketed through a phone as they will most probably be technologically educated hence more likely to trust the telephone representative to try to understand what he/she is talking about rather than prematurely disconnecting the telephone call

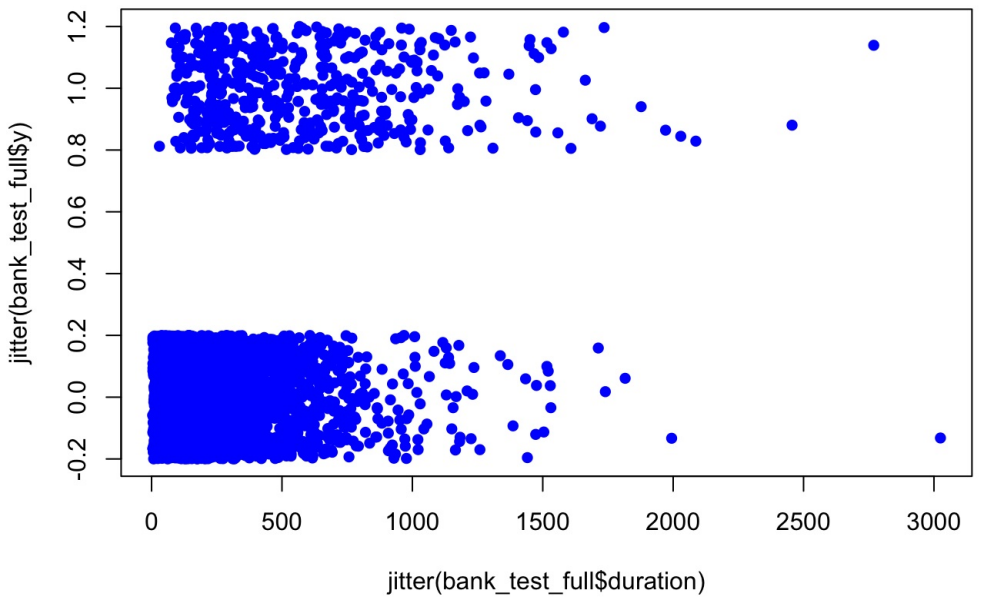
duration vs y

```
#dur_var = bank_factor$duration

plot(jitter(bank_full$duration),jitter(bank_full$y),col="green",pch=16 )
```

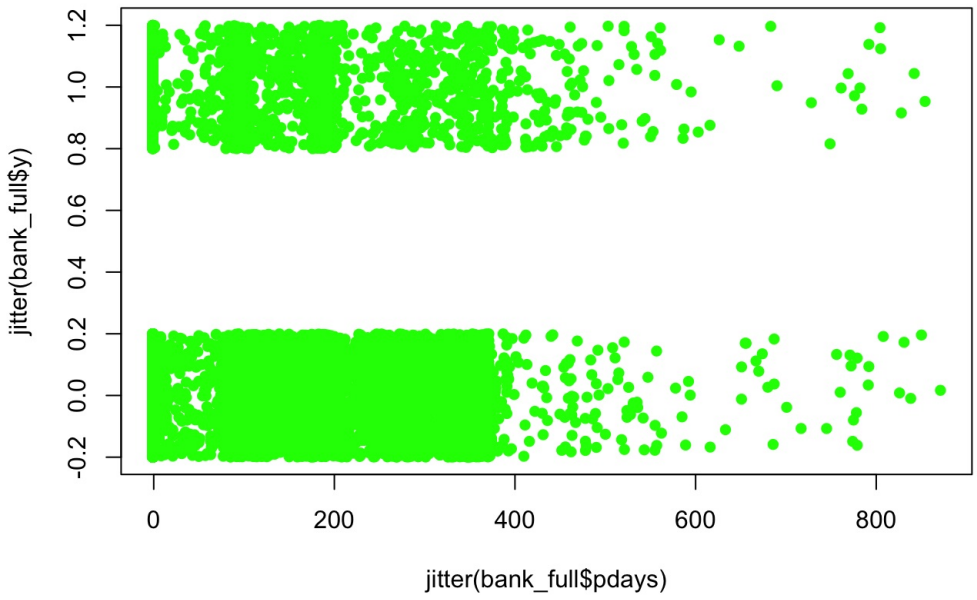



```
plot(jitter(bank_test_full$duration),jitter(bank_test_full$y),col="blue",pch=16 )
```

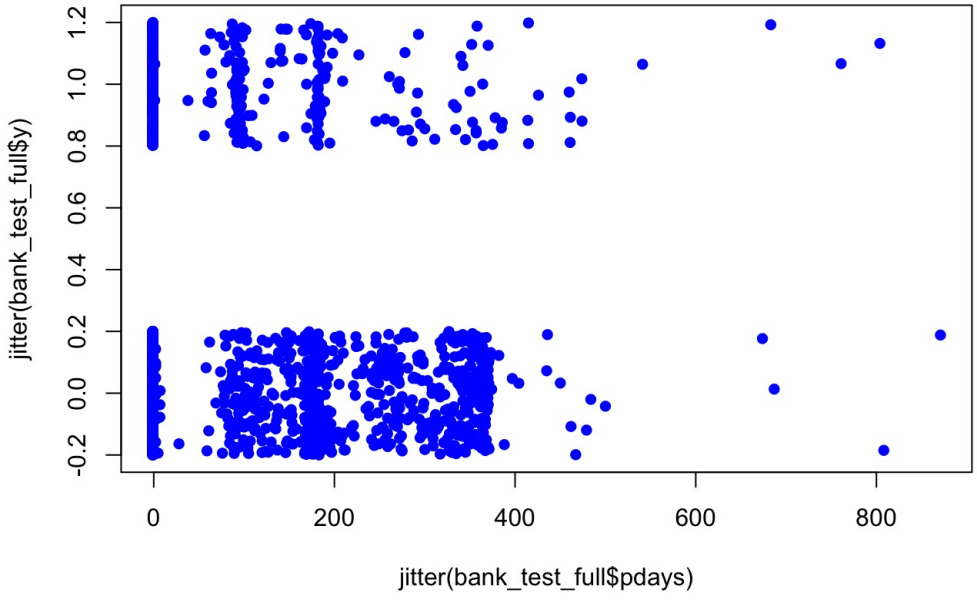


pdays vs y

```
plot(jitter(bank_full$pdays),jitter(bank_full$y),col="green",pch=16 )
```




```
plot(jitter(bank_test_full$pdays),jitter(bank_test_full$y),col="blue",pch=16 )
```



previous vs y

Note: previous represents number of contacts performed before this campaign and for this client

```
table_result = table(bank_factor$previous,bank_factor$y)
table_result[,"yes"]/table_result[,"no"]
```

```
##          0          1          2          3          4          5
## 0.10080429 0.26633166 0.27636364 0.34669811 0.31491713 0.35798817
##          6          7          8          9         10         11
## 0.42783505 0.35761589 0.43333333 0.35294118 0.63414634 0.30000000
##         12         13         14         15         16         17
## 0.29411765 0.31034483 0.35714286 0.05263158 0.00000000 0.25000000
##         18         19         20         21         22         23
## 0.00000000 0.22222222 0.14285714 0.33333333 0.20000000 0.14285714
##         24         25         26         27         28         29
## 0.00000000 0.00000000 1.00000000 0.00000000 0.00000000 0.33333333
##         30         32         35         37         38         40
## 0.50000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##         41         51         55         58        275
## 0.00000000 0.00000000        Inf        Inf 0.00000000
```

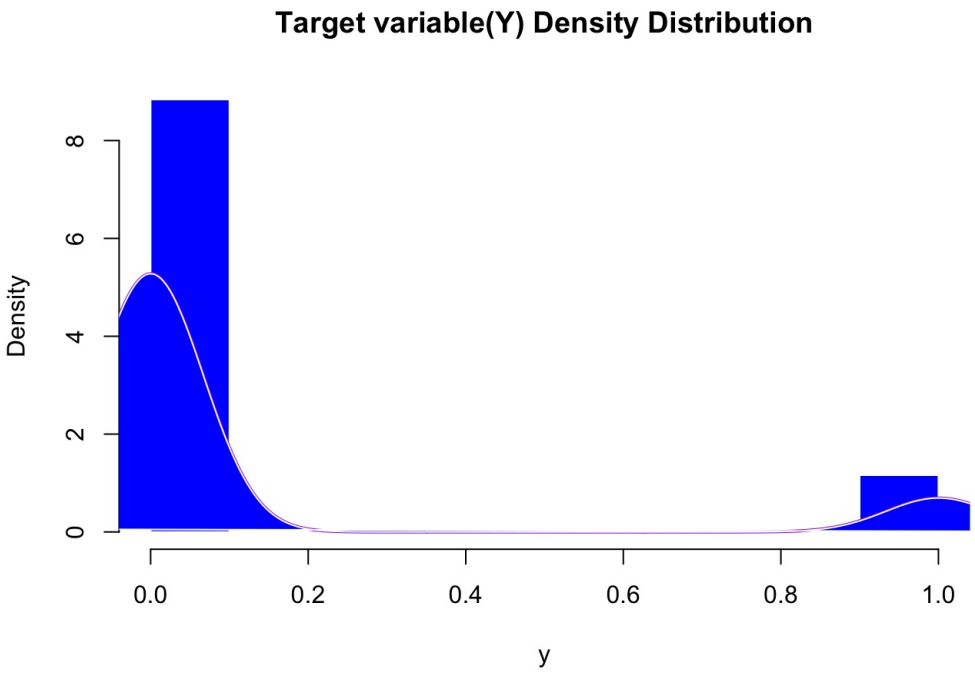
From the above aggregation, we can observe that customers who are followed up between 5 to 10 times have the high success rate of signing up for term deposit plan

TARGET VARIABLE DENSITY DISTRIBUTION

```
y = bank$y

hist(y,
 col="blue", # column color
 border="white",
 prob = TRUE, # show densities instead of frequencies
  main = "Target variable(Y) Density Distribution")


# Density plot
Targetdensity <- density(y)
lines(Targetdensity,
 lwd = 2, # thickness of line
 col = "purple")
polygon(Targetdensity, col = "blue", border = "white")
```



Our target variable (y) follows a Bernoulli Distribution.For our Data, a regression model is considerered

BUILDING THE MODELS

Loading required packages

```
# suppressing the errors/warnings with conflicts
#install.packages("R2jags")
#install.packages("mcmc")
#install.packages("ggmcmc")
#library(coda)
#library(ggmcmc)
library(R2jags,warn.conflicts = FALSE, quietly = TRUE)
```

```
## Warning: package 'coda' was built under R version 3.4.4
```

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

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

```
library(rjags,warn.conflicts = FALSE, quietly = TRUE)
library(coda,warn.conflicts = FALSE, quietly = TRUE)
library(loo,warn.conflicts = FALSE, quietly = TRUE)
```

```
## This is loo version 2.2.0
```

```
## - Online documentation and vignettes at mc-stan.org/loo
```

```
## - As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'cores' argument or
set options(mc.cores = NUM_CORES) for an entire session.
```

```
library(mcmc)
library(mcmcplots)
```

```
## Warning: package 'mcmcplots' was built under R version 3.4.4
```

MODEL 1 : USING BAYESIAN APPROACH

DEFINING THE STANDARD LOGIT MODEL WITH ALL PARAMETERS & VARIABLES

Contents:

```
# Data feeded into the model

#bank_test$y = bank_test$y - 1

#bank_test$y

mydata = list(age = bank_test$age,
              job = bank_test$job,
              martial = bank_test$marital,
              education = bank_test$education,
              default = bank_test$default,
              balance = bank_test$balance,
              housing = bank_test$housing,
              loan = bank_test$loan,
              contact = bank_test$contact,
              day = bank_test$day,
              month = bank_test$month,
              duration = bank_test$duration,
              campaign = bank_test$campaign,
              pdays = bank_test$pdays,
              previous = bank_test$previous,
              poutcome = bank_test$poutcome,
              y = bank_test$y,
              n = nrow(bank_test))

#mydata$outcome
# Total number of observations
n = nrow(bank_test)
n
```

```
## [1] 1500
```

Model 1 definition

```
#Parameters of interest

parameters = c("b0", "beta")

# First Model

bayes_model1 = "model {

#Likelihood Function

for(i in 1:n)
{ #LIKELIHOOD FUNCTION

y[i] ~ dbern(mu[i])

logit(mu[i]) <- b0 + beta[1]*age[i] + beta[2]*job[i] + beta[3]*martial[i] + beta[4]*education[i] + beta[5]*defaul
t[i] + beta[6]*balance[i] + beta[7]*housing[i] + beta[8]*loan[i] + beta[9]*contact[i] + beta[10]*day[i] + beta[11
] * month[i] + beta[12]* duration[i] + beta[13]* campaign[i] + beta[14]*pdays[i] + beta[15]* previous[i] + beta[1
6]* poutcome[i]
}

#Priors

b0 ~ dnorm(0,0.2)     # PARAMETERS WITH NORMAL PRIORS
for(j in 1:16){
beta[j] ~ dnorm(0,0.2)
}
} "
```

CREATING A JAGS MODEL OBJECT

```
jags_model1 = jags.model(textConnection(bayes_model1), data=mydata,n.chains=3, n.adapt = 500)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1500
##    Unobserved stochastic nodes: 17
##    Total graph size: 29881
##
## Initializing model
```

UPDATING THE MODEL & GENERATING POSTERIOR SAMPLES IN MCMC.LIST FORMAT

```
# UPDATING THE MODEL WITH BURN-IN = 1000
update(jags_model1, 1000)

# DRAWING POSTERIOR SAMPLES FROM 1000 ITERATIONS

jags_sample1 = coda.samples(model=jags_model1,
                        variable.names=parameters,
                        n.iter=1000)
```

The above update function performs burn-in iterations for our Markov chain using Jags Model. Here We are collecting samples from the posterior distribution with chains = 3 and iterations = 1000.

Note: If you fail to run those burn-in iterations, then your chain will not converged and any samples you generate will not actually be from the correct posterior distribution. Updating your chain is mandatory and we should perform convergence checks on our final samples to make sure we have burned our chain long enough.

CREATING MARKOV CHAIN MONTE CARLO (MCMC) OBJECTS

```
# MCMC object from Jags Model
bayesian_mcmc1_model1 = as.mcmc(do.call(rbind, jags_sample1))
summary(bayesian_mcmc1_model1)
```

```
##
## Iterations = 1:3000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 3000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean        SD  Naive SE Time-series SE
## b0       -7.970e-01 1.580e+00 2.884e-02      5.969e-01
## beta[1]  -1.593e-02 1.064e-02 1.943e-04      1.323e-03
## beta[2]  -1.744e-02 3.123e-02 5.701e-04      1.774e-03
## beta[3]   8.040e-02 1.670e-01 3.048e-03      1.876e-02
## beta[4]   7.026e-02 1.322e-01 2.414e-03      1.240e-02
## beta[5]  -5.394e-01 1.403e+00 2.561e-02      4.891e-01
## beta[6]  -3.715e-05 4.337e-05 7.919e-07      1.277e-06
## beta[7]  -9.179e-01 2.220e-01 4.052e-03      2.162e-02
## beta[8]  -1.283e+00 3.721e-01 6.794e-03      4.429e-02
## beta[9]  -7.438e-01 1.539e-01 2.809e-03      1.045e-02
## beta[10] -8.264e-03 1.322e-02 2.414e-04      9.870e-04
## beta[11]  5.360e-02 3.373e-02 6.158e-04      2.432e-03
## beta[12]  5.544e-03 4.142e-04 7.563e-06      2.310e-05
## beta[13] -1.037e-01 5.703e-02 1.041e-03      2.864e-03
## beta[14]  4.310e-03 1.850e-03 3.377e-05      2.732e-04
## beta[15]  2.465e-01 7.031e-02 1.284e-03      6.322e-03
## beta[16]  3.940e-01 2.089e-01 3.815e-03      5.006e-02
##
## 2. Quantiles for each variable:
##
##                2.5%        25%        50%        75%      97.5%
## b0       -4.4613641 -1.7723224 -6.789e-01  3.718e-01  1.769e+00
## beta[1]  -0.0371597 -0.0235069 -1.565e-02 -8.662e-03  4.065e-03
## beta[2]  -0.0805712 -0.0378475 -1.797e-02  3.333e-03  4.490e-02
## beta[3]  -0.2273808 -0.0384784  7.525e-02  1.912e-01  4.252e-01
## beta[4]  -0.1885305 -0.0273095  7.444e-02  1.667e-01  3.085e-01
## beta[5]  -3.6148615 -1.4663147 -4.622e-01  6.097e-01  1.691e+00
## beta[6]  -0.0001253 -0.0000646 -3.716e-05 -7.125e-06  4.477e-05
## beta[7]  -1.3829190 -1.0556378 -9.149e-01 -7.651e-01 -5.001e-01
## beta[8]  -2.0012610 -1.5316681 -1.277e+00 -1.040e+00 -5.373e-01
## beta[9]  -1.0424529 -0.8505047 -7.390e-01 -6.357e-01 -4.520e-01
## beta[10] -0.0333976 -0.0169326 -8.167e-03  3.618e-04  1.807e-02
## beta[11] -0.0148139  0.0308849  5.445e-02  7.601e-02  1.179e-01
## beta[12]  0.0047635  0.0052749  5.540e-03  5.798e-03  6.404e-03
## beta[13] -0.2166030 -0.1406604 -1.011e-01 -6.408e-02  1.962e-03
## beta[14]  0.0007868  0.0029556  4.302e-03  5.622e-03  7.858e-03
## beta[15]  0.1136804  0.1983160  2.436e-01  2.933e-01  3.908e-01
## beta[16]  0.0058818  0.2493813  3.852e-01  5.509e-01  7.713e-01
```

SUMMARY STATISTICS

```
#summary
summary(jags_sample1)
```

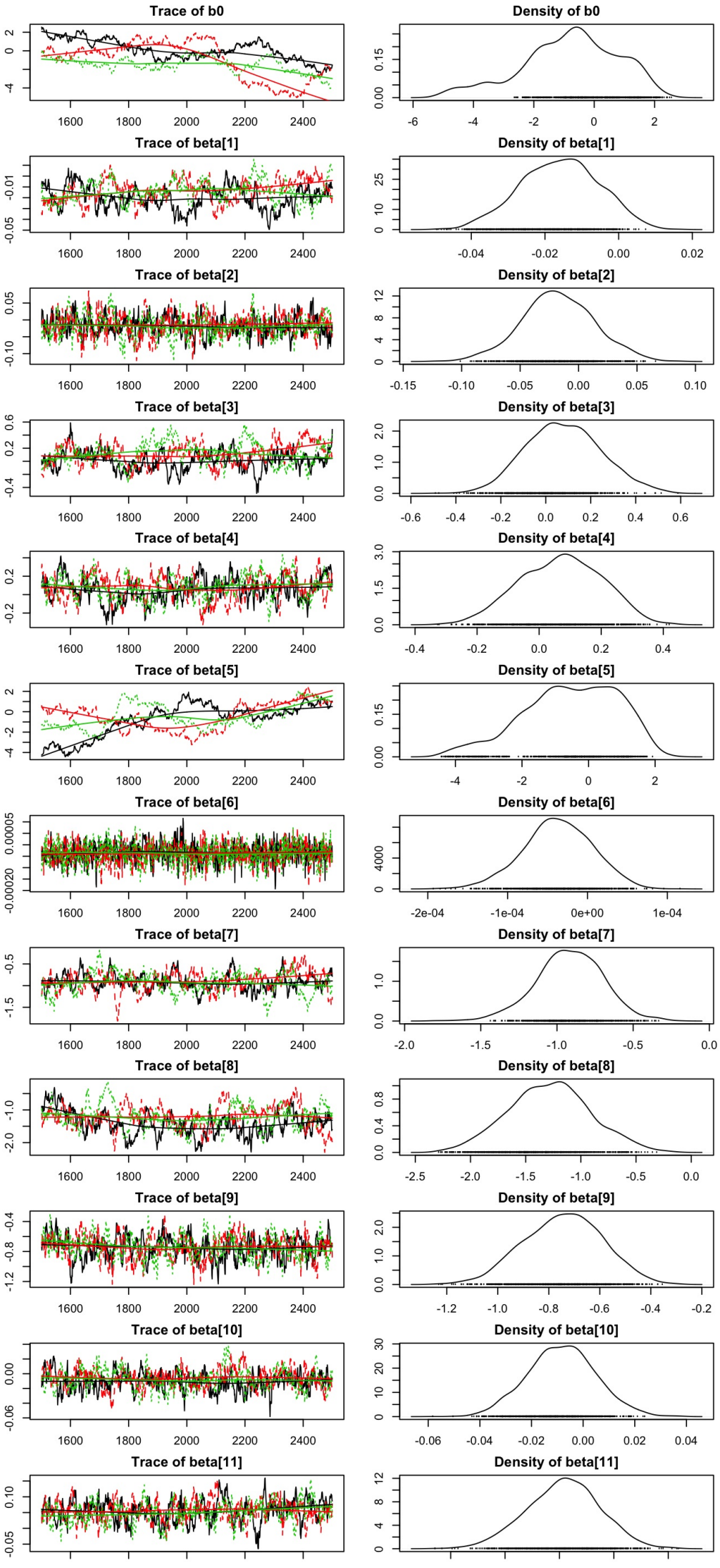
```
##
## Iterations = 1501:2500
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean        SD  Naive SE Time-series SE
## b0       -7.970e-01 1.580e+00 2.884e-02      7.138e-01
## beta[1]  -1.593e-02 1.064e-02 1.943e-04      1.302e-03
## beta[2]  -1.744e-02 3.123e-02 5.701e-04      1.770e-03
## beta[3]   8.040e-02 1.670e-01 3.048e-03      1.679e-02
## beta[4]   7.026e-02 1.322e-01 2.414e-03      1.169e-02
## beta[5]  -5.394e-01 1.403e+00 2.561e-02      4.872e-01
## beta[6]  -3.715e-05 4.337e-05 7.919e-07      1.341e-06
## beta[7]  -9.179e-01 2.220e-01 4.052e-03      2.258e-02
## beta[8]  -1.283e+00 3.721e-01 6.794e-03      4.182e-02
## beta[9]  -7.438e-01 1.539e-01 2.809e-03      1.025e-02
## beta[10] -8.264e-03 1.322e-02 2.414e-04      9.291e-04
## beta[11]  5.360e-02 3.373e-02 6.158e-04      2.462e-03
## beta[12]  5.544e-03 4.142e-04 7.563e-06      2.371e-05
## beta[13] -1.037e-01 5.703e-02 1.041e-03      2.842e-03
## beta[14]  4.310e-03 1.850e-03 3.377e-05      2.469e-04
## beta[15]  2.465e-01 7.031e-02 1.284e-03      3.369e-03
## beta[16]  3.940e-01 2.089e-01 3.815e-03      4.260e-02
##
## 2. Quantiles for each variable:
##
##                2.5%        25%        50%        75%      97.5%
## b0       -4.4613641 -1.7723224 -6.789e-01  3.718e-01  1.769e+00
## beta[1]  -0.0371597 -0.0235069 -1.565e-02 -8.662e-03  4.065e-03
## beta[2]  -0.0805712 -0.0378475 -1.797e-02  3.333e-03  4.490e-02
## beta[3]  -0.2273808 -0.0384784  7.525e-02  1.912e-01  4.252e-01
## beta[4]  -0.1885305 -0.0273095  7.444e-02  1.667e-01  3.085e-01
## beta[5]  -3.6148615 -1.4663147 -4.622e-01  6.097e-01  1.691e+00
## beta[6]  -0.0001253 -0.0000646 -3.716e-05 -7.125e-06  4.477e-05
## beta[7]  -1.3829190 -1.0556378 -9.149e-01 -7.651e-01 -5.001e-01
## beta[8]  -2.0012610 -1.5316681 -1.277e+00 -1.040e+00 -5.373e-01
## beta[9]  -1.0424529 -0.8505047 -7.390e-01 -6.357e-01 -4.520e-01
## beta[10] -0.0333976 -0.0169326 -8.167e-03  3.618e-04  1.807e-02
## beta[11] -0.0148139  0.0308849  5.445e-02  7.601e-02  1.179e-01
## beta[12]  0.0047635  0.0052749  5.540e-03  5.798e-03  6.404e-03
## beta[13] -0.2166030 -0.1406604 -1.011e-01 -6.408e-02  1.962e-03
## beta[14]  0.0007868  0.0029556  4.302e-03  5.622e-03  7.858e-03
## beta[15]  0.1136804  0.1983160  2.436e-01  2.933e-01  3.908e-01
## beta[16]  0.0058818  0.2493813  3.852e-01  5.509e-01  7.713e-01
```

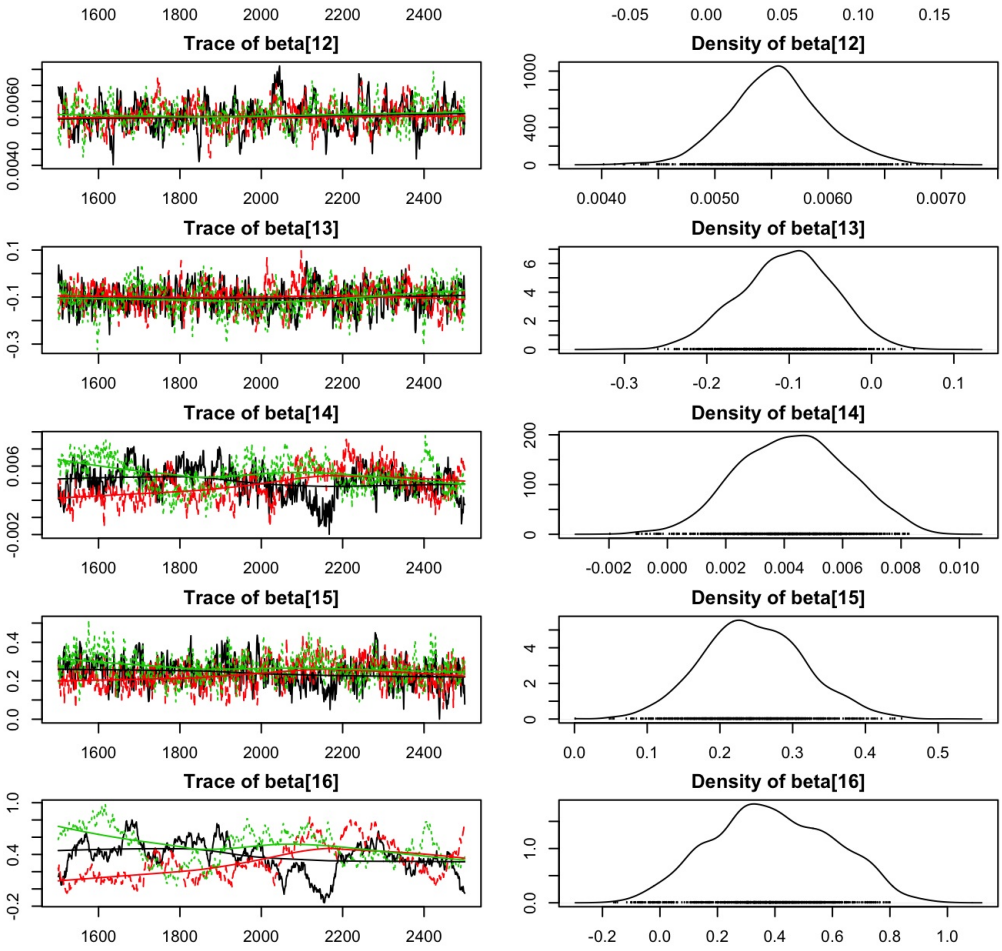
Convergence diagonostics for Model 1

An crucial step in MCMC is verifying the convergence for all the parameters. Bayesian Inference based on MCMC sampling is valid if and only if the Markov chain has convergence. MCMC creates a sample from the posterior distribution, and we usually want to know whether this sample is sufficiently close to the posterior to be used for analysis. Here, I have used traces - plots, marginal densities, autocorrelation diagnosis and Gelman-Rubin diagnostics.

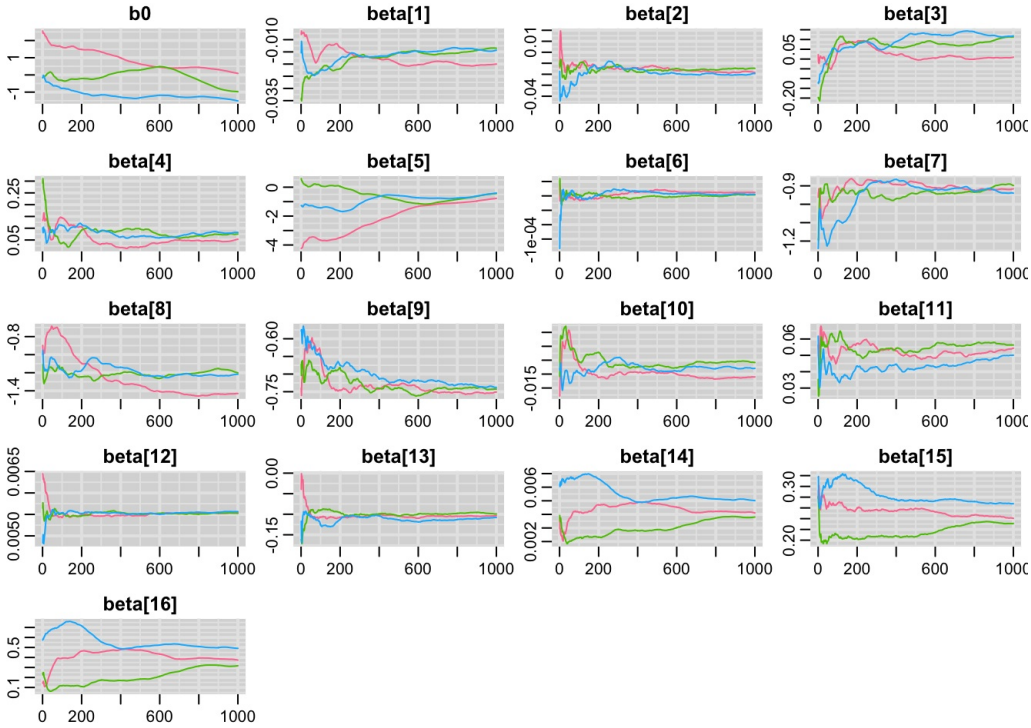
Plots

```
par(mar=c(2,2,2,2))
par(mfrow=c(5,4))
plot(jags_sample1, density=TRUE,trace=TRUE)
```


Trace of b0
Density of b0
Trace of beta[1]
Density of beta[1]
Trace of beta[2]
Density of beta[2]
Trace of beta[3]
Density of beta[3]
Trace of beta[4]
Density of beta[4]
Trace of beta[5]
Density of beta[5]
Trace of beta[6]
Density of beta[6]
Trace of beta[7]
Density of beta[7]
Trace of beta[8]
Density of beta[8]
Trace of beta[9]
Density of beta[9]
Trace of beta[10]
Density of beta[10]
Trace of beta[11]
Density of beta[11]



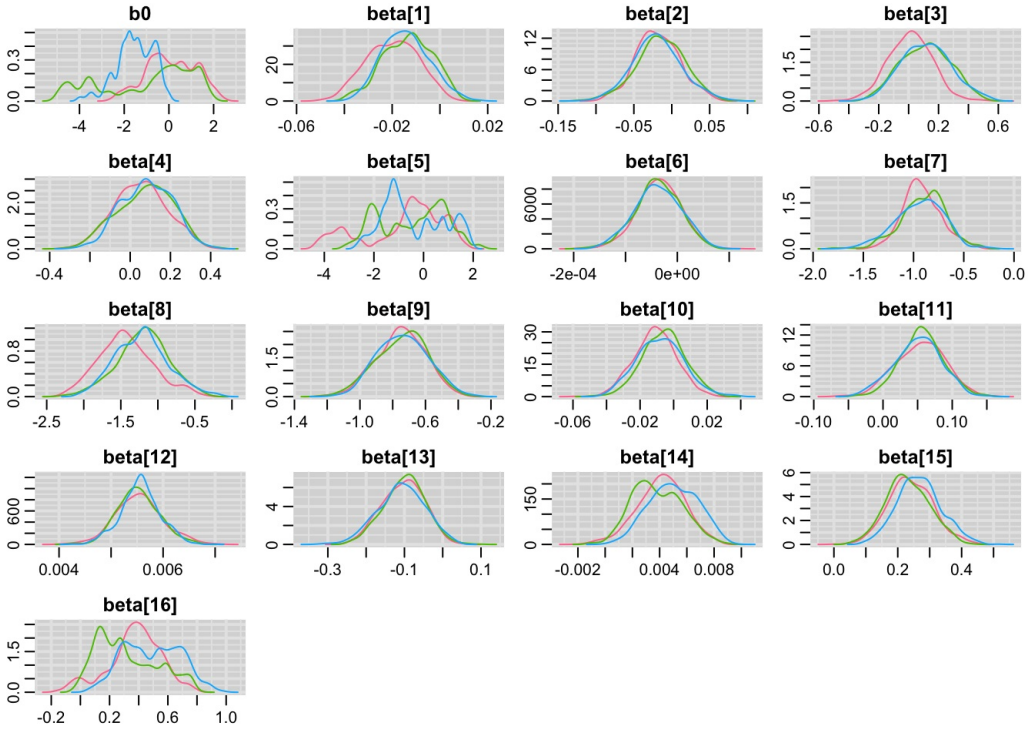

```
rmeanplot(jags_sample1)
```




```
#autplot1(jags_sample1)
```

Other plots:

```
denplot(jags_sample1)
```



Histogram plots for MCMC coeficients

```
#library("bayesplot")
#mcmc_hist(jags_sample1)
```

From the above Density plots you can see the parameters follows the normal distribution, it is a good performance measure for our analysis.

UPDATING THE MODEL WITH BURN-IN=1000 AND DRAWING SAMPLES FROM 4000 ITERATIONS

```
#Second sample with Burn-In = 2500
update(jags_model1, 1000)

# DRAWING POSTERIOR SAMPLES
jags_sample1_2 = coda.samples(model=jags_model1,
                        variable.names=parameters,
                        n.iter=4000)
```

SUMMARY STATISTICS FOR MCMC OBJECT

```
# MCMC object from Jags Model
bayesian_mcmc1_model1_2 = as.mcmc(do.call(rbind, jags_sample1_2))
summary(bayesian_mcmc1_model1_2)
```

```
##
## Iterations = 1:12000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 12000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean        SD  Naive SE Time-series SE
## b0       -9.107e-01 1.3230643 1.208e-02      2.601e-01
## beta[1]  -1.513e-02 0.0102374 9.345e-05      6.884e-04
## beta[2]  -1.550e-02 0.0306637 2.799e-04      8.725e-04
## beta[3]   6.359e-02 0.1829287 1.670e-03      1.133e-02
## beta[4]   7.119e-02 0.1365833 1.247e-03      6.405e-03
## beta[5]  -4.989e-01 1.0537028 9.619e-03      1.695e-01
## beta[6]  -3.772e-05 0.0000457 4.172e-07      8.562e-07
## beta[7]  -9.176e-01 0.2162036 1.974e-03      1.075e-02
## beta[8]  -1.306e+00 0.3621695 3.306e-03      2.152e-02
## beta[9]  -7.371e-01 0.1477679 1.349e-03      4.888e-03
## beta[10] -7.133e-03 0.0121676 1.111e-04      4.081e-04
## beta[11]  5.171e-02 0.0313124 2.858e-04      1.000e-03
## beta[12]  5.524e-03 0.0004039 3.687e-06      1.109e-05
## beta[13] -1.025e-01 0.0560910 5.120e-04      1.242e-03
## beta[14]  4.505e-03 0.0017165 1.567e-05      1.306e-04
## beta[15]  2.476e-01 0.0700335 6.393e-04      2.773e-03
## beta[16]  4.147e-01 0.1969114 1.798e-03      2.125e-02
##
## 2. Quantiles for each variable:
##
##                2.5%        25%        50%        75%      97.5%
## b0       -3.3736416 -1.804e+00 -9.613e-01 -1.169e-01  2.044e+00
## beta[1]  -0.0362201 -2.180e-02 -1.480e-02 -8.178e-03  4.750e-03
## beta[2]  -0.0747129 -3.629e-02 -1.550e-02  5.187e-03  4.476e-02
## beta[3]  -0.2898947 -6.157e-02  6.239e-02  1.844e-01  4.300e-01
## beta[4]  -0.1891387 -2.222e-02  7.128e-02  1.648e-01  3.397e-01
## beta[5]  -2.7425615 -1.185e+00 -4.506e-01  2.131e-01  1.465e+00
## beta[6]  -0.0001329 -6.643e-05 -3.536e-05 -6.562e-06  4.723e-05
## beta[7]  -1.3362832 -1.066e+00 -9.199e-01 -7.718e-01 -4.871e-01
## beta[8]  -2.0703540 -1.538e+00 -1.293e+00 -1.061e+00 -6.360e-01
## beta[9]  -1.0345511 -8.355e-01 -7.367e-01 -6.346e-01 -4.604e-01
## beta[10] -0.0309734 -1.541e-02 -7.036e-03  1.291e-03  1.660e-02
## beta[11] -0.0098743  3.029e-02  5.205e-02  7.263e-02  1.125e-01
## beta[12]  0.0047414  5.247e-03  5.518e-03  5.792e-03  6.330e-03
## beta[13] -0.2152862 -1.395e-01 -1.006e-01 -6.391e-02  4.661e-03
## beta[14]  0.0011129  3.374e-03  4.539e-03  5.710e-03  7.756e-03
## beta[15]  0.1094199  1.997e-01  2.486e-01  2.945e-01  3.848e-01
## beta[16]  0.0261964  2.851e-01  4.167e-01  5.499e-01  8.024e-01
```

SUMMARY FOR Updated model 1

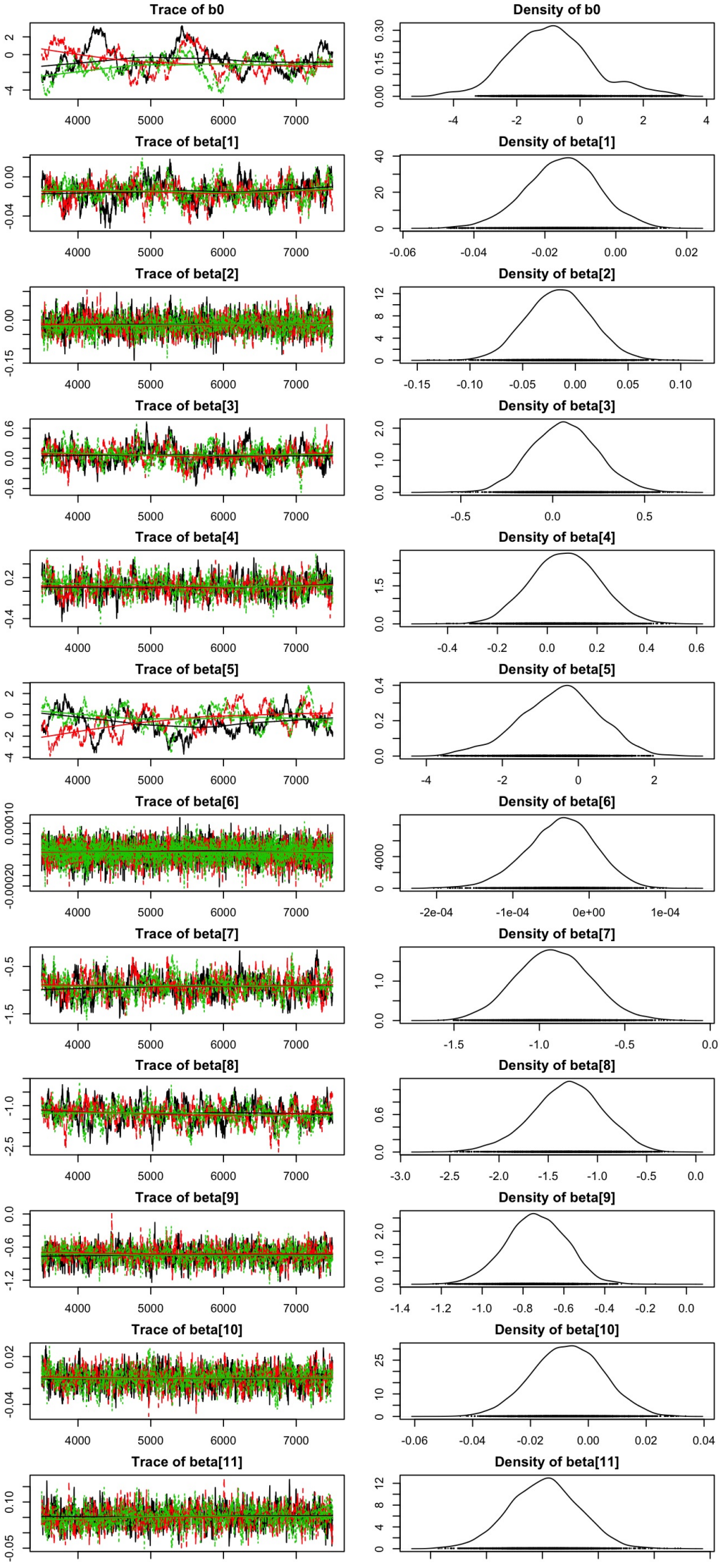
```
#summary
summary(jags_sample1_2)
```

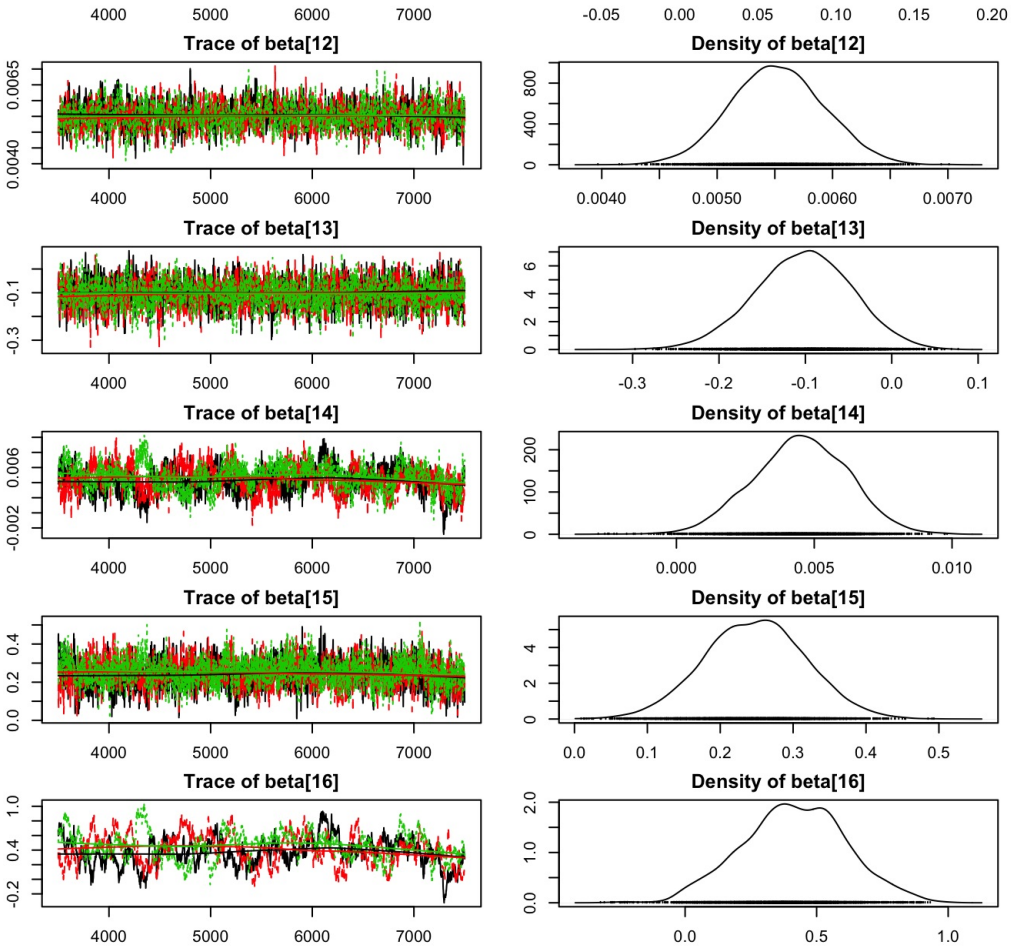
```
##
## Iterations = 3501:7500
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 4000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean        SD  Naive SE Time-series SE
## b0       -9.107e-01 1.3230643 1.208e-02      2.402e-01
## beta[1]  -1.513e-02 0.0102374 9.345e-05      6.591e-04
## beta[2]  -1.550e-02 0.0306637 2.799e-04      8.727e-04
## beta[3]   6.359e-02 0.1829287 1.670e-03      1.162e-02
## beta[4]   7.119e-02 0.1365833 1.247e-03      6.479e-03
## beta[5]  -4.989e-01 1.0537028 9.619e-03      1.594e-01
## beta[6]  -3.772e-05 0.0000457 4.172e-07      8.808e-07
## beta[7]  -9.176e-01 0.2162036 1.974e-03      1.080e-02
## beta[8]  -1.306e+00 0.3621695 3.306e-03      2.169e-02
## beta[9]  -7.371e-01 0.1477679 1.349e-03      4.838e-03
## beta[10] -7.133e-03 0.0121676 1.111e-04      4.045e-04
## beta[11]  5.171e-02 0.0313124 2.858e-04      9.978e-04
## beta[12]  5.524e-03 0.0004039 3.687e-06      1.113e-05
## beta[13] -1.025e-01 0.0560910 5.120e-04      1.240e-03
## beta[14]  4.505e-03 0.0017165 1.567e-05      1.331e-04
## beta[15]  2.476e-01 0.0700335 6.393e-04      2.173e-03
## beta[16]  4.147e-01 0.1969114 1.798e-03      2.044e-02
##
## 2. Quantiles for each variable:
##
##                2.5%        25%        50%        75%      97.5%
## b0       -3.3736416 -1.804e+00 -9.613e-01 -1.169e-01  2.044e+00
## beta[1]  -0.0362201 -2.180e-02 -1.480e-02 -8.178e-03  4.750e-03
## beta[2]  -0.0747129 -3.629e-02 -1.550e-02  5.187e-03  4.476e-02
## beta[3]  -0.2898947 -6.157e-02  6.239e-02  1.844e-01  4.300e-01
## beta[4]  -0.1891387 -2.222e-02  7.128e-02  1.648e-01  3.397e-01
## beta[5]  -2.7425615 -1.185e+00 -4.506e-01  2.131e-01  1.465e+00
## beta[6]  -0.0001329 -6.643e-05 -3.536e-05 -6.562e-06  4.723e-05
## beta[7]  -1.3362832 -1.066e+00 -9.199e-01 -7.718e-01 -4.871e-01
## beta[8]  -2.0703540 -1.538e+00 -1.293e+00 -1.061e+00 -6.360e-01
## beta[9]  -1.0345511 -8.355e-01 -7.367e-01 -6.346e-01 -4.604e-01
## beta[10] -0.0309734 -1.541e-02 -7.036e-03  1.291e-03  1.660e-02
## beta[11] -0.0098743  3.029e-02  5.205e-02  7.263e-02  1.125e-01
## beta[12]  0.0047414  5.247e-03  5.518e-03  5.792e-03  6.330e-03
## beta[13] -0.2152862 -1.395e-01 -1.006e-01 -6.391e-02  4.661e-03
## beta[14]  0.0011129  3.374e-03  4.539e-03  5.710e-03  7.756e-03
## beta[15]  0.1094199  1.997e-01  2.486e-01  2.945e-01  3.848e-01
## beta[16]  0.0261964  2.851e-01  4.167e-01  5.499e-01  8.024e-01
```

Convergence diagonostics

Plots

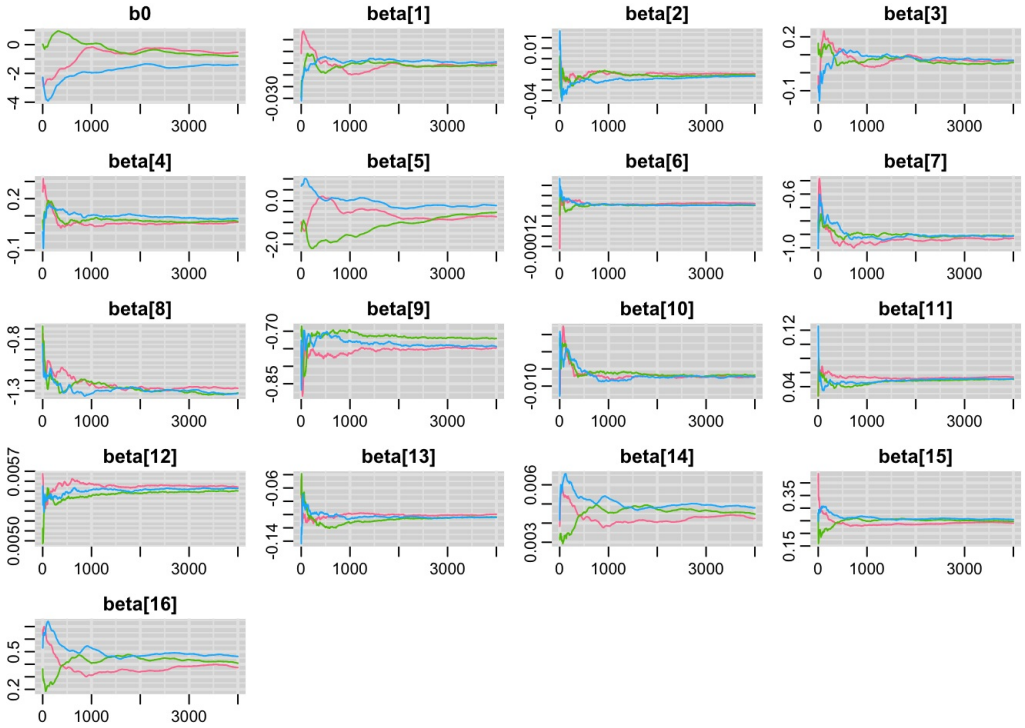
```
par(mar=c(2,2,2,2))
par(mfrow=c(5,4))
plot(jags_sample1_2)   # Plotting Density and Trace plots
```


Trace of b0
Density of b0
Trace of beta[1]
Density of beta[1]
Trace of beta[2]
Density of beta[2]
Trace of beta[3]
Density of beta[3]
Trace of beta[4]
Density of beta[4]
Trace of beta[5]
Density of beta[5]
Trace of beta[6]
Density of beta[6]
Trace of beta[7]
Density of beta[7]
Trace of beta[8]
Density of beta[8]
Trace of beta[9]
Density of beta[9]
Trace of beta[10]
Density of beta[10]
Trace of beta[11]
Density of beta[11]



As we can clearly observe from the above trace plots incresing burn in really made a difference on how our MCMC chains are converging. so the model update yeilded good performance on how our Model is converging.

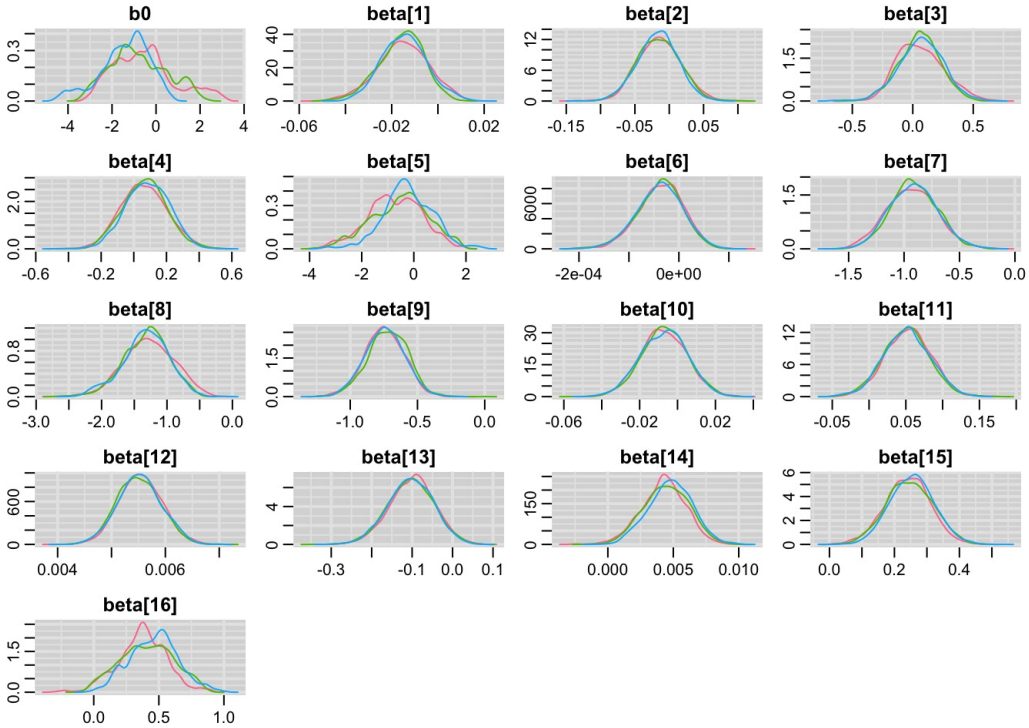
```
rmeanplot(jags_sample1_2)
```



So mean plot also indicates that our chains are converging well

```
#autplot1(jags_sample1_2)
```

```
denplot(jags_sample1_2)
```



So there is a lot of overlap in thhe density plots of individual chains so our model has improved

Calculation of Deviantion Information Criterion(DIC).

Model 1

DIC for standard Logit model is as follows

```
#rjags::load.module('dic')

## calculate DIC
DIC_model1 = dic.samples(jags_model1, n.iter=1000)
DIC_model1
```

```
## Mean deviance:  762.3
## penalty 16.15
## Penalized deviance: 778.5
```

DIC is calculated by adding the ‘effective number of parameters’ (pD) to the expected deviance with two or more parallel chains in our model.

It is used to approximate the penalized plug-in deviance,when only a point estimate of the parameters is of interest when the effective number of parameters is much smaller than the sample size asymptotically, and the model parameters have a normal posterior distribution.

MODEL 2 USING LOGIT WITH REDUCED FEATURES

DATA MODEL WITH REDUCED PARAMETERS

Contents:

Calculation of DIC

```
#DATA

# Data feeded into the model
RL_mydata = list(housing = bank_test$housing,
              loan = bank_test$loan,
              contact = bank_test$contact,
              duration = bank_test$duration,
              pdays = bank_test$pdays,
              previous = bank_test$previous,
              y = bank_test$y,
              n = nrow(bank_test))

# Total number of observations
n = nrow(bank_test)
```

LOGIT MODEL

```
# Parameters of interest

RL_parameters = c("beta")

# Logit Model with Reduced Parameters

RL_bayes_model1 = "model {

#Likelihood Function

for(i in 1:n)
{
#likelihood function
y[i] ~ dbern(mu[i])

logit(mu[i]) <- beta[1]*housing[i] + beta[2]*loan[i] + beta[3]*contact[i] + beta[4]*duration[i] + beta[5]*pdays[i
] + beta[6]*previous[i]
}

#Priors
for(j in 1:6){
beta[j] ~ dnorm(0,0.2)
}
} "
```

CREATING A JAGS OBJECT

```
RL_jags_model1 = jags.model(textConnection(RL_bayes_model1), data=RL_mydata,n.chains=3, n.adapt = 500)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1500
##    Unobserved stochastic nodes: 6
##    Total graph size: 13449
##
## Initializing model
```

MODEL UPDATION WITH BURIN-IN = 1000 AND 4000 ITERATIONS

```
#UPDATING MODEL WITH Burn-In = 1000

update(RL_jags_model1, 1000)

# DRAWING POSTERIOR SAMPLES
RL_jags_sample1 = coda.samples(model=RL_jags_model1,
                        variable.names=RL_parameters,
                        n.iter=4000)
```

```
# MCMC object from Jags Model
RL_bayesian_mcmc1_model1 = as.mcmc(do.call(rbind, RL_jags_sample1))
summary(RL_bayesian_mcmc1_model1)
```

```
##
## Iterations = 1:12000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 12000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean        SD  Naive SE Time-series SE
## beta[1] -0.878234 0.1797281 1.641e-03      7.310e-03
## beta[2] -1.571398 0.2426193 2.215e-03      9.601e-03
## beta[3] -0.707076 0.1391870 1.271e-03      4.232e-03
## beta[4]  0.005391 0.0004032 3.681e-06      1.100e-05
## beta[5]  0.001812 0.0010731 9.796e-06      2.093e-05
## beta[6]  0.190996 0.0599758 5.475e-04      1.085e-03
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%     97.5%
## beta[1] -1.2406277 -0.996219 -0.877347 -0.759630 -0.523327
## beta[2] -2.0585804 -1.732045 -1.568450 -1.408594 -1.100044
## beta[3] -0.9878853 -0.798777 -0.704501 -0.610226 -0.444589
## beta[4]  0.0046367  0.005117  0.005386  0.005651  0.006217
## beta[5] -0.0002943  0.001101  0.001819  0.002527  0.003904
## beta[6]  0.0753876  0.150487  0.191688  0.230851  0.307862
```

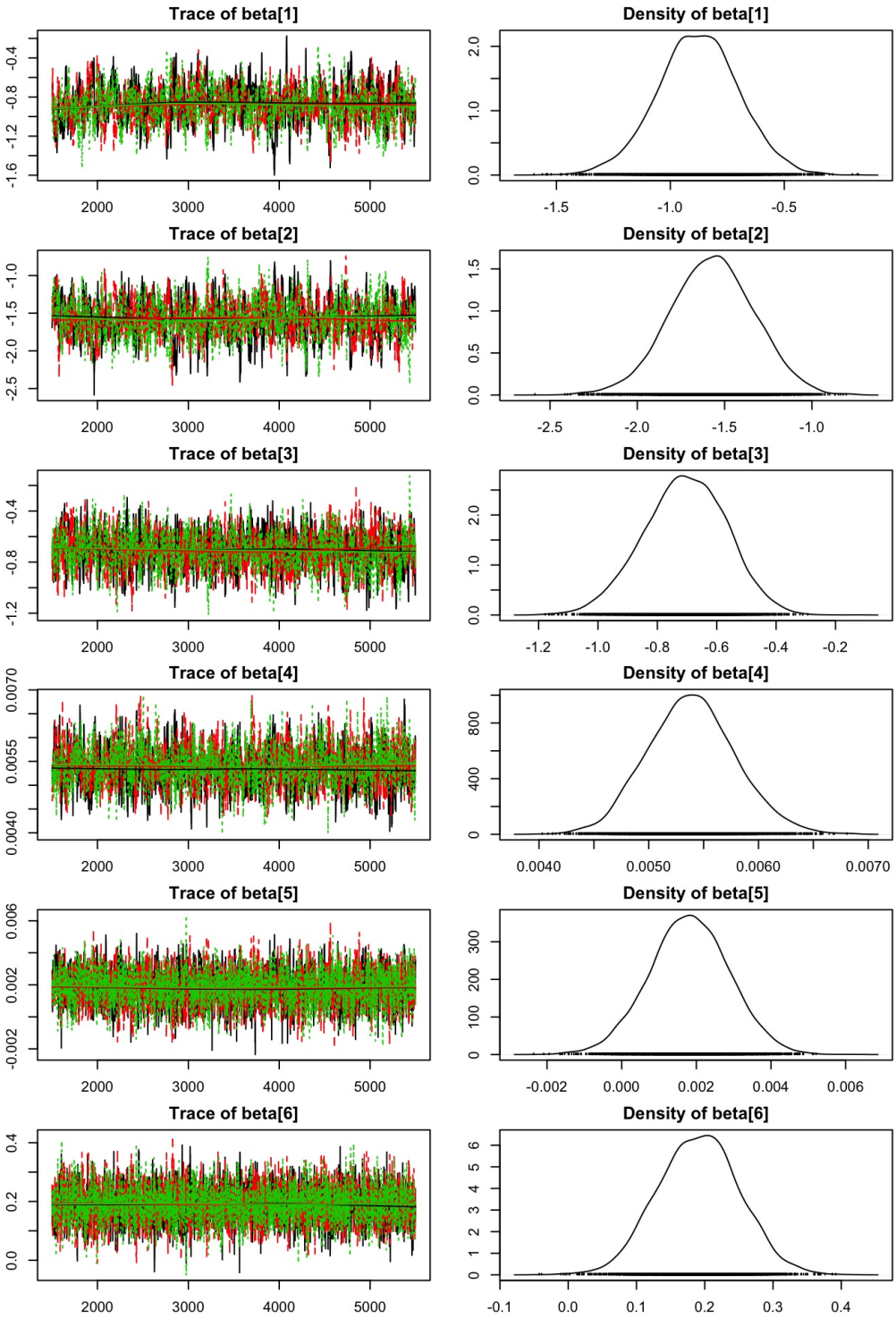
As we can clearly observe that beta[4] which is the duration parameter coefficient is converged rather quickly which indicates the importance of it for obtaning better model performance

Convergence diagonostics

Plots

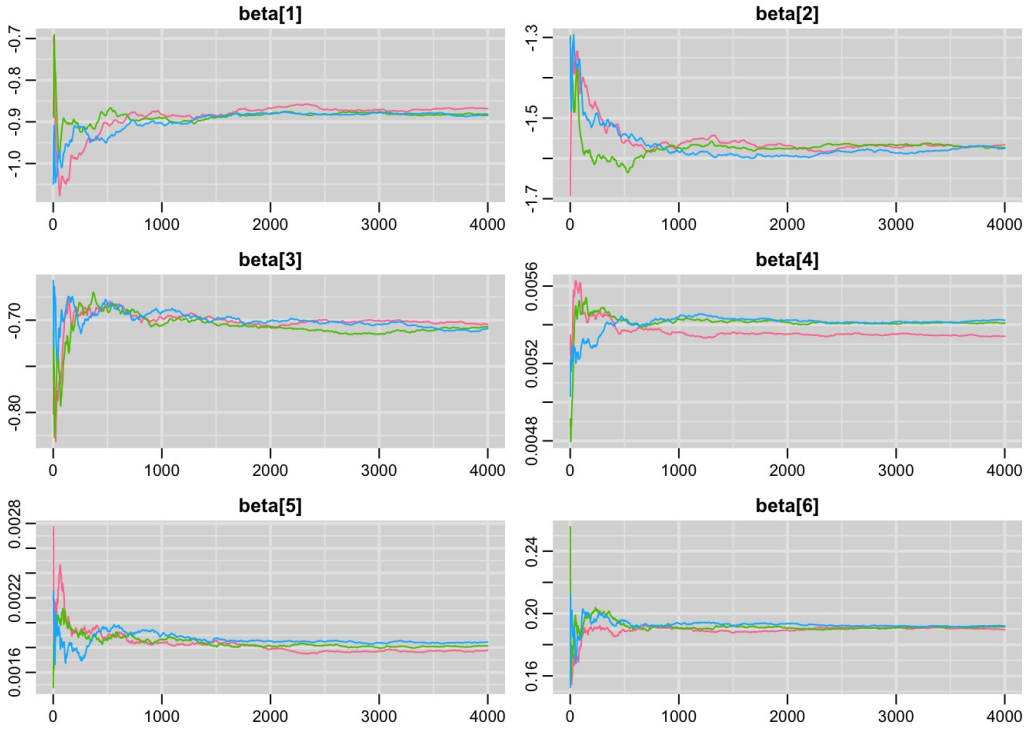
Model 2 trace plot

```
par(mar=c(2,2,2,2))
par(mfrow=c(3,2))
plot(RL_jags_sample1)   # Plotting Density and Trace plots
```



Model 2 mean plot

```
rmeanplot(RL_jags_sample1)
```

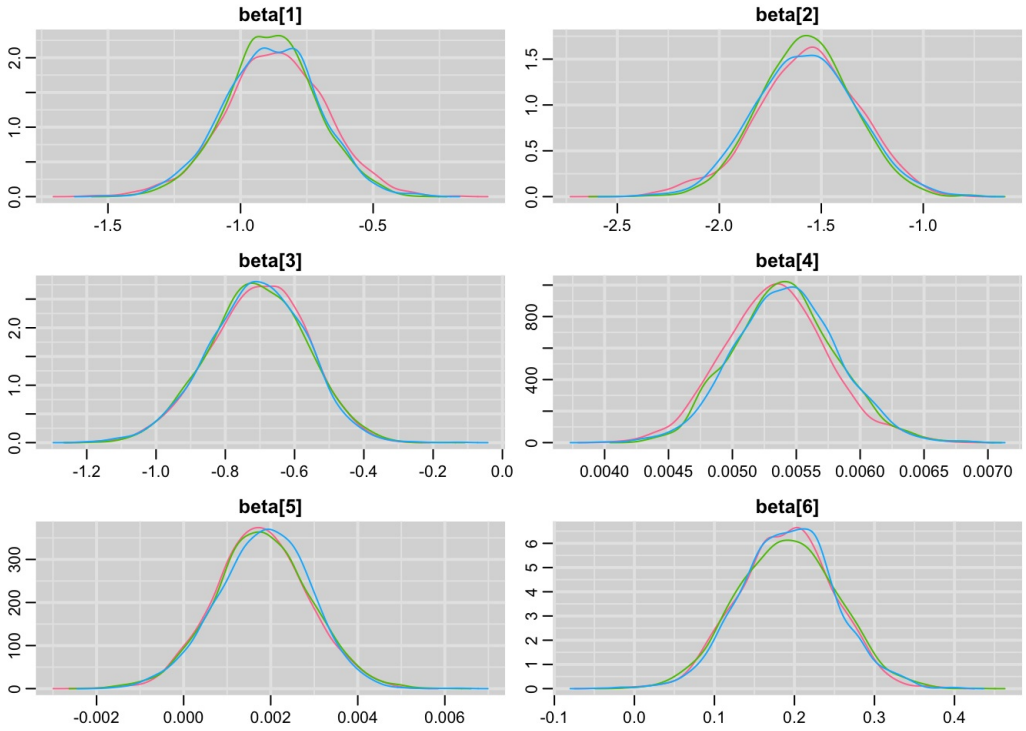


Model 2 autocorelation

```
#autplot1(RL_jags_sample1)
```

Model 2 density plot

```
denplot(RL_jags_sample1)
```



DIC CALCULATION

```
## calculate DIC
RL_DIC_model1 = dic.samples(RL_jags_model1, n.iter=1000)
RL_DIC_model1
```

```
## Mean deviance:  769.8
## penalty 6.04
## Penalized deviance: 775.8
```

MODEL 3 : PROBIT MODEL USING BAYESIAN APPROACH

This is a model with reduced parameters (only beta) and most significant risk factors that contribute in identifying y

DATA MODEL WITH REDUCED PARAMETERS

Contents:

Calculation of DIC

```
# Data feeded into the model
P_mydata = list(housing = bank_test$housing,
              loan = bank_test$loan,
              contact = bank_test$contact,
              duration = bank_test$duration,
              pdays = bank_test$pdays,
              previous = bank_test$previous,
              y = bank_test$y,
              n = nrow(bank_test))

# Total number of observations
n = nrow(bank_test)
```

USING PROBIT MODEL OF LOGISTIC REGRESSION

```
# Parameters of interest

reduced_parameters = c("beta")

# First Model

bayes_model2 = "model {

#Likelihood Function

for(i in 1:n)
{

#likelihood function
y[i] ~ dbern(mu[i])
mu[i] <- phi(theta[i])

theta[i] <- beta[1]*housing[i] + beta[2]*loan[i] + beta[3]*contact[i] + beta[4]*duration[i] + beta[5]*pdays[i] +
beta[6]*previous[i]
}

#Priors

for(j in 1:6){
beta[j] ~ dnorm(0, 0.2)
}
} "
```

Defining JAGS object for Model 3

```
set.seed(100)
jags_model2 = jags.model(textConnection(bayes_model2), data=P_mydata, n.chains=3,n.adapt = 500)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1500
##    Unobserved stochastic nodes: 6
##    Total graph size: 13449
##
## Initializing model
```

Generating MCMC list for Model 3

```
#Extract the samples with Burn-In = 1000

update(jags_model2, 1000)
jags_sample2 = coda.samples(model=jags_model2,
                        variable.names=reduced_parameters,
                        n.iter=4000)
```

```
# MCMC object from Jags Model
bayesian_mcmc2_model2 = as.mcmc(do.call(rbind, jags_sample2))
summary(bayesian_mcmc2_model2)
```

```
##
## Iterations = 1:12000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 12000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean        SD  Naive SE Time-series SE
## beta[1] -0.4805312 0.0934616 8.532e-04      3.729e-03
## beta[2] -0.8880736 0.1243281 1.135e-03      4.628e-03
## beta[3] -0.3819409 0.0713680 6.515e-04      2.185e-03
## beta[4]  0.0029204 0.0001982 1.809e-06      4.923e-06
## beta[5]  0.0009509 0.0005817 5.310e-06      1.154e-05
## beta[6]  0.0983528 0.0337912 3.085e-04      6.010e-04
##
## 2. Quantiles for each variable:
##
##              2.5%       25%        50%       75%     97.5%
## beta[1] -0.669267 -0.541948 -0.4790407 -0.418410 -0.296292
## beta[2] -1.137977 -0.969614 -0.8871271 -0.803230 -0.652099
## beta[3] -0.521857 -0.430431 -0.3814650 -0.333899 -0.244806
## beta[4]  0.002546  0.002785  0.0029174  0.003052  0.003314
## beta[5] -0.000217  0.000557  0.0009633  0.001343  0.002041
## beta[6]  0.031428  0.075564  0.0987549  0.121396  0.163852
```

Here also beta[4] which corresponds to our duration parameter converges rather quickly.

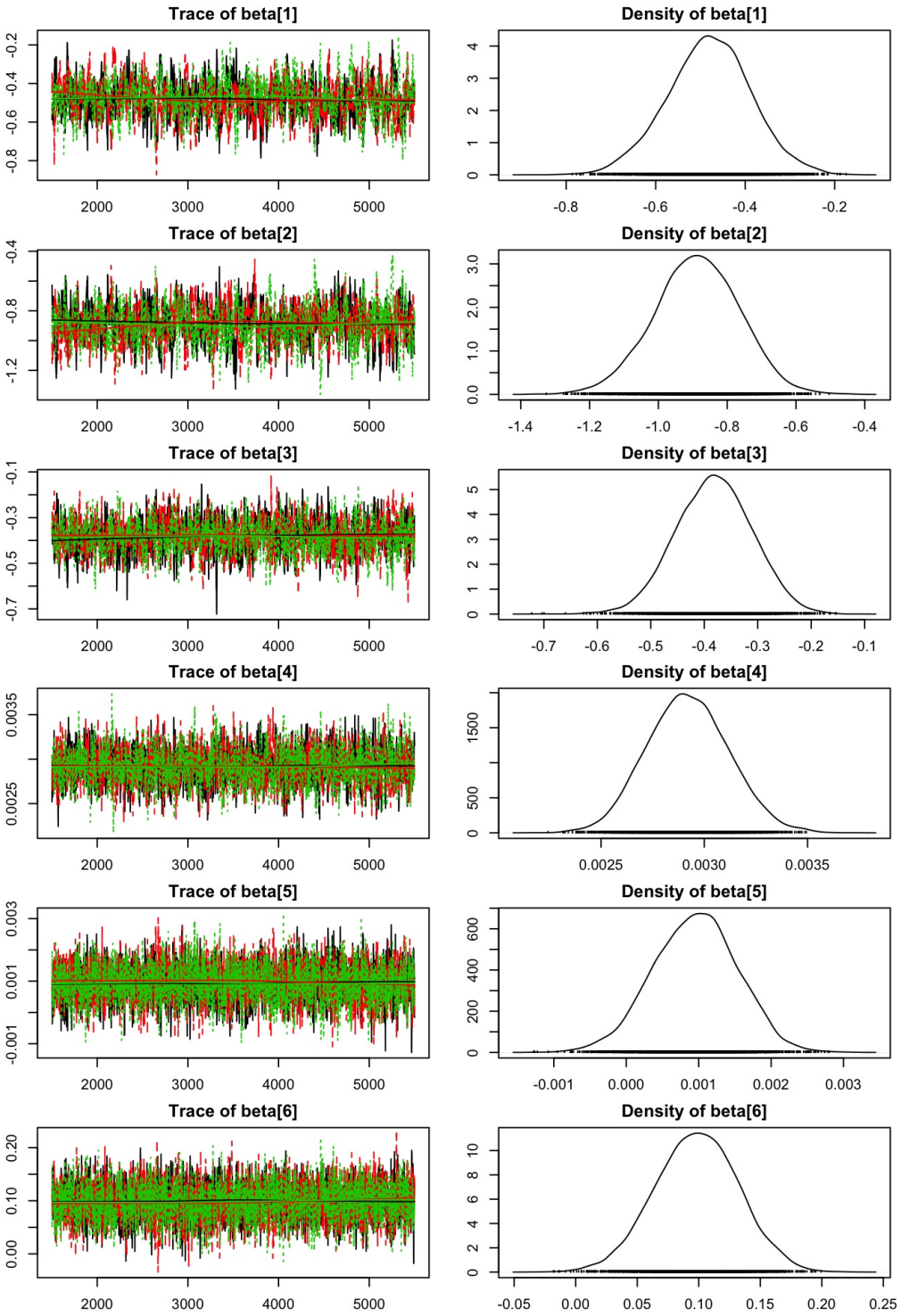
SUMMARY

```
summary(jags_sample2)
```

```
##
## Iterations = 1501:5500
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 4000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean        SD  Naive SE Time-series SE
## beta[1] -0.4805312 0.0934616 8.532e-04      3.752e-03
## beta[2] -0.8880736 0.1243281 1.135e-03      4.764e-03
## beta[3] -0.3819409 0.0713680 6.515e-04      2.153e-03
## beta[4]  0.0029204 0.0001982 1.809e-06      4.890e-06
## beta[5]  0.0009509 0.0005817 5.310e-06      1.113e-05
## beta[6]  0.0983528 0.0337912 3.085e-04      5.974e-04
##
## 2. Quantiles for each variable:
##
##              2.5%       25%        50%       75%     97.5%
## beta[1] -0.669267 -0.541948 -0.4790407 -0.418410 -0.296292
## beta[2] -1.137977 -0.969614 -0.8871271 -0.803230 -0.652099
## beta[3] -0.521857 -0.430431 -0.3814650 -0.333899 -0.244806
## beta[4]  0.002546  0.002785  0.0029174  0.003052  0.003314
## beta[5] -0.000217  0.000557  0.0009633  0.001343  0.002041
## beta[6]  0.031428  0.075564  0.0987549  0.121396  0.163852
```

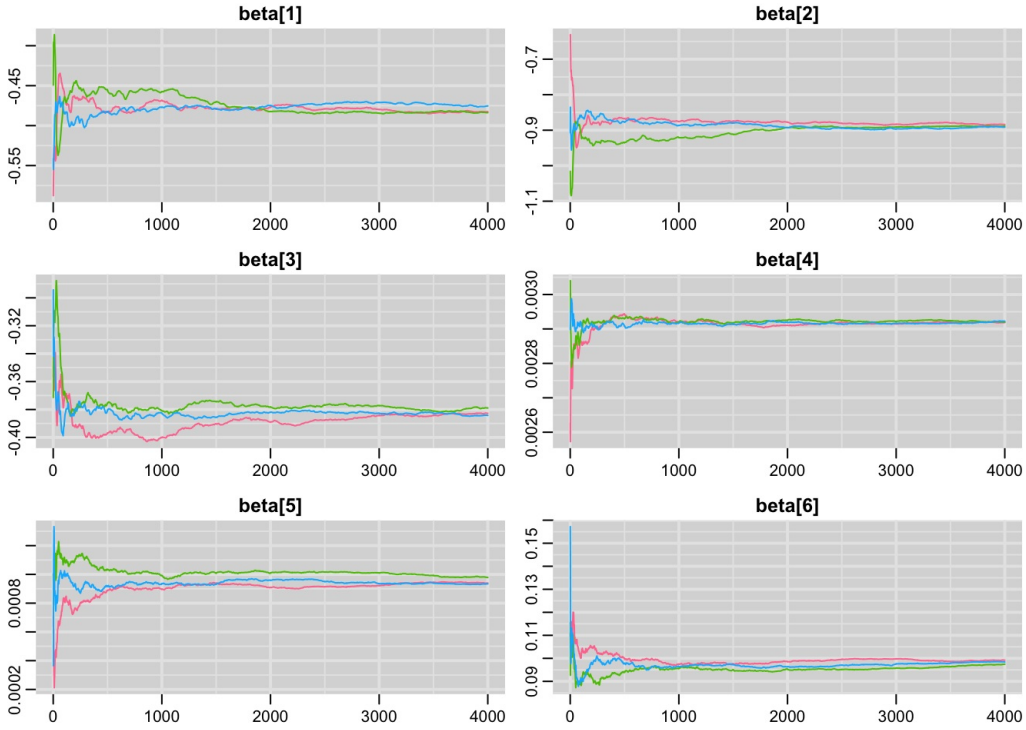
Model 3 trace plot

```
par(mar=c(2,2,2,2))
par(mfrow=c(3,2))
plot(jags_sample2)   # Plotting Density and Trace plots together
```



Model 3 mean plot

```
rmeanplot(jags_sample2)
```

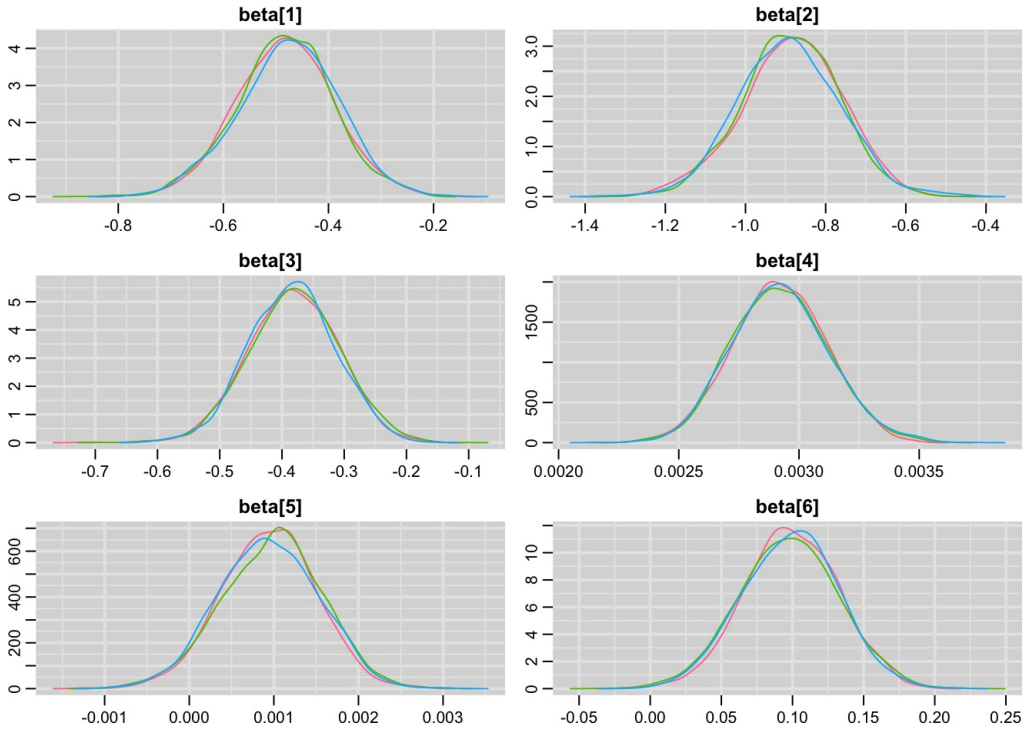


Model 3 autocorelation

```
#autplot1(jags_sample2)
```

Model 3 density plot

```
denplot(jags_sample2)
```



Model 3 DIC

```
## calculate DIC
DIC_model2 = dic.samples(jags_model2, n.iter=1000)
DIC_model2
```

```
## Mean deviance:  766.2
## penalty 5.639
## Penalized deviance: 771.8
```

Higher Posterior Density (HDP) interval

We also computed the Highest Posterior Density(HDP) interval. This is an interval where most of the distribution lies and it satisfies the following two conditions:

1. The posterior probability of that region is

$$((1 - \text{alpha}))$$

.
2. The minimum density of any point within that region is greater or equal than the density of any point of the region. In particular, it is computed for the 95% interval.

This is displayed for each one of the chains:

HDP for Model 1

```
hdpM1_95<-data.frame(HPDinterval(jags_sample1, prob=.95))
hdpM1_95
```

```
##                  lower         upper       lower.1       upper.1
## b0       -1.9446736625  2.171088e+00 -4.8393597314  1.624452e+00
## beta[1]  -0.0399089838 -3.680018e-04 -0.0325009569  5.651082e-03
## beta[2]  -0.0729762891  4.409771e-02 -0.0812009335  4.550823e-02
## beta[3]  -0.3033273101  2.667571e-01 -0.1684112112  4.564829e-01
## beta[4]  -0.1937965315  3.084914e-01 -0.1882221082  3.150704e-01
## beta[5]  -4.1026160034  1.335519e+00 -2.6045360831  1.806470e+00
## beta[6]  -0.0001225391  4.514389e-05 -0.0001204772  4.475681e-05
## beta[7]  -1.2668000810 -4.996294e-01 -1.3904828493 -5.040449e-01
## beta[8]  -2.0557012956 -5.859553e-01 -1.8839082311 -4.909031e-01
## beta[9]  -1.0326163809 -4.686631e-01 -1.0563148321 -4.473570e-01
## beta[10] -0.0352742807  1.235946e-02 -0.0315303415  1.816542e-02
## beta[11] -0.0162301014  1.199970e-01 -0.0013354989  1.210929e-01
## beta[12]  0.0046950976  6.517003e-03  0.0048143681  6.363299e-03
## beta[13] -0.2103393591 -7.074617e-03 -0.2076815075  1.956409e-03
## beta[14]  0.0008792378  7.864963e-03  0.0006371966  7.364063e-03
## beta[15]  0.1038503332  3.826074e-01  0.1150526222  3.776683e-01
## beta[16] -0.0684469479  6.756859e-01  0.0014528407  7.371375e-01
##                lower.2       upper.2
## b0       -2.9328463741 -1.311949e-02
## beta[1]  -0.0338292885  3.690695e-03
## beta[2]  -0.0849130199  4.520075e-02
## beta[3]  -0.1943382440  4.434490e-01
## beta[4]  -0.1396547071  3.278059e-01
## beta[5]  -2.1994713160  1.877918e+00
## beta[6]  -0.0001251426  4.692994e-05
## beta[7]  -1.4339871644 -5.237604e-01
## beta[8]  -1.8938366322 -4.984573e-01
## beta[9]  -1.0046628846 -4.372080e-01
## beta[10] -0.0349588846  1.811380e-02
## beta[11] -0.0182578704  1.119556e-01
## beta[12]  0.0047926769  6.287906e-03
## beta[13] -0.2207212750  1.601898e-02
## beta[14]  0.0018662572  8.325707e-03
## beta[15]  0.1357604192  4.064715e-01
## beta[16]  0.1248506859  8.348389e-01
```

Here,

beta[1] => housing beta[2] => loan beta[3] => contact beta[4] => duration
beta[5] => pdays beta[6] => previous

HDP for Model 2

```
hdpM2_95<-data.frame(HPDinterval(RL_jags_sample1, prob=.95))
hdpM2_95
```

```
##                 lower        upper       lower.1      upper.1
## beta[1] -1.2211052859 -0.466287994 -1.2160986633 -0.544735487
## beta[2] -2.0654755546 -1.049815125 -1.9976273045 -1.123479046
## beta[3] -0.9753645812 -0.437764802 -0.9761009123 -0.433548164
## beta[4]  0.0045435639  0.006169938  0.0046797192  0.006193863
## beta[5] -0.0002706567  0.003881669 -0.0003020513  0.003979460
## beta[6]  0.0747314375  0.297167416  0.0682515056  0.306004026
##               lower.2      upper.2
## beta[1] -1.2279692029 -0.532517846
## beta[2] -2.0434272403 -1.090958215
## beta[3] -0.9837948093 -0.443236474
## beta[4]  0.0046508598  0.006205275
## beta[5] -0.0002873968  0.003812528
## beta[6]  0.0821219189  0.314168728
```

HDP for Model 3

```
hdpM3_95<-data.frame(HPDinterval(jags_sample2, prob=.95))
hdpM3_95
```

```
##                lower        upper       lower.1      upper.1
## beta[1] -0.6579209229 -0.291351608 -0.6618871969 -0.286424866
## beta[2] -1.1468729895 -0.651076445 -1.1321083032 -0.665991039
## beta[3] -0.5258868086 -0.251463540 -0.5246772064 -0.242867384
## beta[4]  0.0025419679  0.003291988  0.0025418290  0.003305141
## beta[5] -0.0001785711  0.001993107 -0.0002295082  0.002078868
## beta[6]  0.0383880809  0.165768238  0.0287327503  0.165222199
##               lower.2      upper.2
## beta[1] -0.6775404935 -0.303457044
## beta[2] -1.1315786161 -0.635550411
## beta[3] -0.5127578517 -0.245473616
## beta[4]  0.0025239967  0.003304173
## beta[5] -0.0001679157  0.002078163
## beta[6]  0.0302545265  0.160480949
```

We can be observe that the 2 intervals are quite similar for beta[4] which is duration parameter and for other parameters a lot of variance is observed.

Convergence diagnostic

Raftery and Lewis diagnostic

We also computed as an example the Ratery and Lewis diagnostic. This is used to measure some posterior quantile (q) of interest with a predifined tolerance (r) for (q) and a probability (s) of being within that tolerance.

Then, the Raftery and Lewis diagnostic computes the number of necessary iterations N and burn-ins to satisfy the specified conditions.

This is calculated for each parameter in the chain.

```
#raftery.diag(jags_sample1)
```

Heildelberger and Welchs convergence diagnostics

To test the convergence we also computed the Heildelberger and Welch???s convergence diagnostics. This diagnostic computes a test statistic in order to accept or reject the null hypothesis that the Markov chain is from a stationary distritribution.

In order to this, we used the following procedure, which consists on two parts.
1) In the first part, the test is applied on the whole chain, and if it fails, it is discarded 10% of the data until the null hypothesis is accepted or until 50 % is discarded.

2. In the second part, it is computed half of the width of the $(1 - \alpha)$credible interval around the mean. If the ratio of the halfwidth and the mean is lower than the threshold , then more iterations should be consider.

Below, the results of the test are shown:

Model 1

```
heidel.diag(jags_sample1)
```

```
## [[1]]
##
##          Stationarity start     p-value
##          test         iteration
## b0       passed         1       0.0909
## beta[1]  passed         1       0.5334
## beta[2]  passed         1       0.3042
## beta[3]  passed         1       0.3609
## beta[4]  passed         1       0.1446
## beta[5]  passed       301       0.4094
## beta[6]  passed         1       0.2597
## beta[7]  passed         1       0.6768
## beta[8]  passed       101       0.4648
## beta[9]  passed         1       0.4198
## beta[10] passed         1       0.2603
## beta[11] passed         1       0.4881
## beta[12] passed         1       0.5590
## beta[13] passed         1       0.8462
## beta[14] passed         1       0.1229
## beta[15] passed       401       0.0575
## beta[16] passed         1       0.1696
##
##          Halfwidth Mean      Halfwidth
##          test
## b0       failed     9.17e-02 1.01e+00
## beta[1]  failed    -2.00e-02 4.66e-03
```

```
## beta[2]  failed    -1.83e-02 5.42e-03
## beta[3]  failed     1.00e-02 4.86e-02
## beta[4]  failed     5.30e-02 4.14e-02
## beta[5]  failed     1.01e-01 6.15e-01
## beta[6]  failed    -3.51e-05 4.26e-06
## beta[7]  passed    -9.16e-01 6.00e-02
## beta[8]  passed    -1.50e+00 1.05e-01
## beta[9]  passed    -7.51e-01 3.34e-02
## beta[10] failed    -1.10e-02 2.44e-03
## beta[11] failed     5.44e-02 9.54e-03
## beta[12] passed     5.54e-03 9.57e-05
## beta[13] passed    -1.04e-01 7.79e-03
## beta[14] failed     4.11e-03 6.47e-04
## beta[15] passed     2.31e-01 1.48e-02
## beta[16] failed     3.74e-01 1.17e-01
##
## [[2]]
##
##          Stationarity start     p-value
##          test         iteration
## b0       passed         1       0.5195
## beta[1]  passed         1       0.1459
## beta[2]  passed         1       0.6746
## beta[3]  passed         1       0.1487
## beta[4]  passed         1       0.6470
## beta[5]  passed         1       0.7291
## beta[6]  passed         1       0.3037
## beta[7]  passed         1       0.1610
## beta[8]  passed         1       0.9590
## beta[9]  passed         1       0.6961
## beta[10] passed         1       0.4671
## beta[11] passed         1       0.8030
## beta[12] passed         1       0.8533
## beta[13] passed         1       0.8547
## beta[14] passed       101       0.0853
## beta[15] passed       301       0.0525
## beta[16] passed         1       0.1052
##
##          Halfwidth Mean      Halfwidth
##          test
## b0       failed    -9.73e-01 4.03e+00
## beta[1]  failed    -1.35e-02 4.86e-03
## beta[2]  failed    -1.46e-02 6.07e-03
## beta[3]  failed     1.13e-01 6.04e-02
## beta[4]  failed     7.56e-02 3.99e-02
## beta[5]  failed    -4.35e-01 1.73e+00
## beta[6]  failed    -3.86e-05 4.34e-06
## beta[7]  passed    -8.98e-01 6.58e-02
## beta[8]  failed    -1.20e+00 1.30e-01
## beta[9]  passed    -7.43e-01 3.86e-02
## beta[10] failed    -5.83e-03 3.38e-03
## beta[11] failed     5.63e-02 7.41e-03
## beta[12] passed     5.53e-03 7.59e-05
## beta[13] passed    -9.94e-02 8.52e-03
## beta[14] failed     3.98e-03 8.58e-04
## beta[15] passed     2.41e-01 1.69e-02
## beta[16] failed     3.18e-01 1.65e-01
##
## [[3]]
##
##          Stationarity start     p-value
##          test         iteration
## b0       passed         1       0.5023
## beta[1]  passed         1       0.6282
## beta[2]  passed         1       0.2400
## beta[3]  passed         1       0.4923
## beta[4]  passed         1       0.5451
## beta[5]  passed         1       0.5784
## beta[6]  passed         1       0.3987
## beta[7]  passed         1       0.5469
## beta[8]  passed         1       0.5756
## beta[9]  passed         1       0.1231
## beta[10] passed         1       0.8088
## beta[11] failed        NA       0.0124
## beta[12] passed         1       0.1817
## beta[13] passed         1       0.0992
## beta[14] passed         1       0.2550
## beta[15] passed       101       0.2796
## beta[16] passed         1       0.1942
```

```
##
##          Halfwidth Mean      Halfwidth
##          test
## b0       failed    -1.51e+00 5.71e-01
## beta[1]  failed    -1.43e-02 3.66e-03
## beta[2]  failed    -1.94e-02 6.48e-03
## beta[3]  failed     1.18e-01 6.11e-02
## beta[4]  failed     8.21e-02 3.77e-02
## beta[5]  failed    -4.09e-01 1.36e+00
## beta[6]  failed    -3.77e-05 5.02e-06
## beta[7]  failed    -9.39e-01 9.85e-02
## beta[8]  failed    -1.22e+00 1.51e-01
## beta[9]  passed    -7.38e-01 3.21e-02
## beta[10] failed    -7.95e-03 3.53e-03
## beta[11] <NA>             NA       NA
## beta[12] passed     5.57e-03 6.73e-05
## beta[13] failed    -1.08e-01 1.21e-02
## beta[14] failed     5.01e-03 9.80e-04
## beta[15] passed     2.63e-01 1.00e-02
## beta[16] failed     4.91e-01 1.48e-01
```

Model 2

```
heidel.diag(RL_jags_sample1)
```

```
## [[1]]
##
##         Stationarity start     p-value
##         test         iteration
## beta[1] passed       1         0.760
## beta[2] passed       1         0.945
## beta[3] passed       1         0.646
## beta[4] passed       1         0.697
## beta[5] passed       1         0.498
## beta[6] passed       1         0.578
##
##         Halfwidth Mean     Halfwidth
##         test
## beta[1] passed    -0.86865 2.78e-02
## beta[2] passed    -1.56585 3.57e-02
## beta[3] passed    -0.70466 1.27e-02
## beta[4] passed     0.00534 3.71e-05
## beta[5] passed     0.00178 7.12e-05
## beta[6] passed     0.18954 3.49e-03
##
## [[2]]
##
##         Stationarity start     p-value
##         test         iteration
## beta[1] passed       1         0.919
## beta[2] passed       1         0.805
## beta[3] passed       1         0.413
## beta[4] passed       1         0.994
## beta[5] passed       1         0.908
## beta[6] passed       1         0.762
##
##         Halfwidth Mean     Halfwidth
##         test
## beta[1] passed    -0.88183 2.26e-02
## beta[2] passed    -1.57276 2.80e-02
## beta[3] passed    -0.70725 1.41e-02
## beta[4] passed     0.00541 3.80e-05
## beta[5] passed     0.00181 7.23e-05
## beta[6] passed     0.19147 3.76e-03
##
## [[3]]
##
##         Stationarity start     p-value
##         test         iteration
## beta[1] passed       1         0.707
## beta[2] passed       1         0.549
## beta[3] passed       1         0.279
## beta[4] passed       1         0.688
## beta[5] passed       1         0.802
## beta[6] passed       1         0.934
##
##         Halfwidth Mean     Halfwidth
##         test
## beta[1] passed    -0.88421 2.46e-02
## beta[2] passed    -1.57558 3.41e-02
## beta[3] passed    -0.70931 1.55e-02
## beta[4] passed     0.00542 3.67e-05
## beta[5] passed     0.00184 6.72e-05
## beta[6] passed     0.19197 3.69e-03
```

Model 3

```
heidel.diag(jags_sample2)
```

```
## [[1]]
##
##         Stationarity start     p-value
##         test         iteration
## beta[1] passed         1       0.590
## beta[2] passed         1       0.609
## beta[3] passed       401       0.172
## beta[4] passed         1       0.760
## beta[5] passed         1       0.134
## beta[6] passed         1       0.437
##
##         Halfwidth Mean     Halfwidth
##         test
## beta[1] passed    -0.48251 1.26e-02
## beta[2] passed    -0.88410 1.63e-02
## beta[3] passed    -0.38076 7.63e-03
## beta[4] passed     0.00292 1.55e-05
## beta[5] passed     0.00094 3.59e-05
## beta[6] passed     0.09919 1.93e-03
##
## [[2]]
##
##         Stationarity start     p-value
##         test         iteration
## beta[1] passed         1       0.1063
## beta[2] passed       401       0.3115
## beta[3] passed         1       0.8382
## beta[4] passed         1       0.8493
## beta[5] failed        NA       0.0211
## beta[6] failed        NA       0.0201
##
##         Halfwidth Mean     Halfwidth
##         test
## beta[1] passed    -0.48372 1.24e-02
## beta[2] passed    -0.88358 1.55e-02
## beta[3] passed    -0.37890 7.64e-03
## beta[4] passed     0.00292 1.68e-05
## beta[5] <NA>            NA       NA
## beta[6] <NA>            NA       NA
##
## [[3]]
##
##         Stationarity start     p-value
##         test         iteration
## beta[1] passed       1         0.910
## beta[2] passed       1         0.831
## beta[3] passed       1         0.791
## beta[4] passed       1         0.440
## beta[5] passed       1         0.446
## beta[6] passed       1         0.068
##
##         Halfwidth Mean      Halfwidth
##         test
## beta[1] passed    -0.475362 1.32e-02
## beta[2] passed    -0.891540 1.68e-02
## beta[3] passed    -0.384234 6.99e-03
## beta[4] passed     0.002922 1.75e-05
## beta[5] passed     0.000934 3.92e-05
## beta[6] passed     0.098414 2.07e-03
```

We can see that in all three chains most of the parameters passed the test, so we do not need more iterations for convergence.

FREQUENTIST APPROACH

Model 1 with Frequentist approach

In this model we only consider the important risk factors involved to estimate the output variable y

```
Frequentist_model1 = glm( y ~ age + job + marital + education + default + balance + housing + loan + contact + da
y + month + duration + campaign + pdays + previous +
poutcome , data = bank_test , family = binomial(link = "logit"))

summary(Frequentist_model1)
```

```
##
## Call:
## glm(formula = y ~ age + job + marital + education + default +
##     balance + housing + loan + contact + day + month + duration +
##     campaign + pdays + previous + poutcome, family = binomial(link = "logit"),
##     data = bank_test)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.9454  -0.4078  -0.2540  -0.1438   2.8438
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.552e+00  1.725e+00  -1.479 0.139050
## age         -1.237e-02  1.072e-02  -1.154 0.248657
## job         -1.504e-02  3.065e-02  -0.491 0.623706
## marital      1.118e-01  1.847e-01   0.605 0.544997
## education    8.070e-02  1.369e-01   0.590 0.555424
## default      6.032e-01  1.064e+00   0.567 0.570921
## balance     -3.249e-05  4.421e-05  -0.735 0.462378
## housing     -8.792e-01  2.128e-01  -4.131 3.61e-05 ***
## loan        -1.230e+00  3.600e-01  -3.416 0.000636 ***
## contact     -7.189e-01  1.485e-01  -4.841 1.29e-06 ***
## day         -6.534e-03  1.216e-02  -0.537 0.591183
## month        5.207e-02  3.183e-02   1.636 0.101862
## duration     5.412e-03  4.034e-04  13.416  < 2e-16 ***
## campaign    -9.558e-02  5.489e-02  -1.741 0.081596 .
## pdays        4.855e-03  1.780e-03   2.728 0.006366 **
## previous     2.524e-01  6.912e-02   3.651 0.000261 ***
## poutcome     4.589e-01  2.072e-01   2.214 0.026804 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1084.73  on 1499  degrees of freedom
## Residual deviance:  744.47  on 1483  degrees of freedom
## AIC: 778.47
##
## Number of Fisher Scoring iterations: 6
```

Model 2 with Frequentist approach

```
RL_Frequentist_model2 = glm( y ~ 0 + housing + loan + contact + duration + pdays + previous , data = bank_test ,
family = binomial(link = "logit"))

summary(RL_Frequentist_model2)
```

```
##
## Call:
## glm(formula = y ~ 0 + housing + loan + contact + duration + pdays +
##     previous, family = binomial(link = "logit"), data = bank_test)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7439  -0.4282  -0.2678  -0.1480   2.8960
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## housing  -0.8768137  0.1794324  -4.887 1.03e-06 ***
## loan     -1.5515493  0.2461007  -6.305 2.89e-10 ***
## contact  -0.7018219  0.1392346  -5.041 4.64e-07 ***
## duration  0.0053435  0.0003986  13.407  < 2e-16 ***
## pdays     0.0018315  0.0010453   1.752  0.07975 .
## previous  0.1899788  0.0584989   3.248  0.00116 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2079.44  on 1500  degrees of freedom
## Residual deviance:  763.66  on 1494  degrees of freedom
## AIC: 775.66
##
## Number of Fisher Scoring iterations: 6
```

Note: Logit and probit differ in how they define function. The logit model uses the cumulative distribution function of the logistic distribution and The probit model uses the cumulative distribution function of the standard normal distribution to define function.

Both functions will take any number and rescale it to fall between 0 and 1.

MODEL 3 USING FREQUENTIST APPROACH

This Probit model is built considering the most significant risk factors that contribute in identifying y

```
RL_Frequentist_model3 = glm( y ~ 0 + housing + loan + contact + duration + pdays + previous , data = bank_test ,
family = binomial(link = "probit"))

summary(RL_Frequentist_model3)
```

```
##
## Call:
## glm(formula = y ~ 0 + housing + loan + contact + duration + pdays +
##     previous, family = binomial(link = "probit"), data = bank_test)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7430  -0.4418  -0.2567  -0.1115   3.0212
##
## Coefficients:
##            Estimate Std. Error z value Pr(>|z|)
## housing  -0.4823638  0.0937739  -5.144 2.69e-07 ***
## loan     -0.8808425  0.1278244  -6.891 5.54e-12 ***
## contact  -0.3782497  0.0713318  -5.303 1.14e-07 ***
## duration  0.0029027  0.0002097  13.844  < 2e-16 ***
## pdays     0.0009737  0.0005785   1.683  0.09232 .
## previous  0.0973730  0.0334151   2.914  0.00357 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2079.44  on 1500  degrees of freedom
## Residual deviance:  760.29  on 1494  degrees of freedom
## AIC: 772.29
##
## Number of Fisher Scoring iterations: 6
```

COMPARISION OF MODELS WITH RESULTS : BAYESIAN Vs. FREQUENTIST

Model 1 comparison (Logit model with all the parameters)

```
Frequentist_Results1 = Frequentist_model1$coefficients
Bayesian_Results1 = colMeans(bayesian_mcmc1_model1)

cat("Frequentist vs Bayesian for Logit Model with all the features: \n\n ")
```

```
## Frequentist vs Bayesian for Logit Model with all the features:
##
##
```

```
Frequentist_Results1
```

```
##   (Intercept)           age           job       marital     education
## -2.551778e+00 -1.237252e-02 -1.503741e-02  1.117956e-01  8.070442e-02
##       default       balance       housing          loan       contact
##  6.031888e-01 -3.249477e-05 -8.792398e-01 -1.229581e+00 -7.188855e-01
##           day         month      duration      campaign         pdays
## -6.533537e-03  5.207030e-02  5.411874e-03 -9.558246e-02  4.855149e-03
##      previous      poutcome
##  2.523791e-01  4.588738e-01
```

```
Bayesian_Results1
```

```
##            b0       beta[1]       beta[2]       beta[3]       beta[4]
## -7.970222e-01 -1.592519e-02 -1.744037e-02  8.039565e-02  7.025679e-02
##       beta[5]       beta[6]       beta[7]       beta[8]       beta[9]
## -5.394243e-01 -3.714886e-05 -9.178953e-01 -1.283007e+00 -7.437767e-01
##      beta[10]      beta[11]      beta[12]      beta[13]      beta[14]
## -8.264464e-03  5.359986e-02  5.543654e-03 -1.036681e-01  4.310253e-03
##      beta[15]      beta[16]
##  2.465097e-01  3.940369e-01
```

Model 2 comparison(Logit model with reduced parameters)

```
RL_Frequentist_Results2 = RL_Frequentist_model2$coefficients
RL_Bayesian_Results2 = colMeans(RL_bayesian_mcmc1_model1)

cat("Frequentist vs Bayesian for  Reduced Logit Model: \n\n ")
```

```
## Frequentist vs Bayesian for  Reduced Logit Model:
##
##
```

```
RL_Frequentist_Results2
```

```
##      housing         loan      contact     duration        pdays
## -0.876813714 -1.551549317 -0.701821909  0.005343510  0.001831542
##     previous
##  0.189978752
```

```
RL_Bayesian_Results2
```

```
##      beta[1]      beta[2]      beta[3]      beta[4]      beta[5]
## -0.878234019 -1.571398439 -0.707076333  0.005390987  0.001811956
##      beta[6]
##  0.190996269
```

PROBIT MODEL COMPARISION

```
Frequentist_Results3 = RL_Frequentist_model3$coefficients
Bayesian_Results3 = colMeans(bayesian_mcmc2_model2)

cat("Frequentist vs Bayesian for Probit model: \n\n ")
```

```
## Frequentist vs Bayesian for Probit model:
##
##
```

```
Frequentist_Results3
```

```
##      housing         loan      contact     duration        pdays
## -0.482363828 -0.880842514 -0.378249746  0.002902720  0.000973738
##     previous
##  0.097373042
```

```
Bayesian_Results3
```

```
##       beta[1]       beta[2]       beta[3]       beta[4]       beta[5]
## -0.4805311965 -0.8880736290 -0.3819409211  0.0029204359  0.0009508598
##       beta[6]
##  0.0983527836
```

As we can observe from above comparison tables that Bayesian models have better capability to interpret the coefficients (beta) than their frequentist counterparts. This has to do with the practice of prior with Bayesian models so they were able to clearly make accurate correlations between input and output variables.

COMPARISION OF MODELS WITH DIC CALCULATION

The comparisons between two models is calculated by the means of Deviance Information Criteria. #### DIC for standard logit model:

```
DIC_model1
```

```
## Mean deviance:  762.3
## penalty 16.15
## Penalized deviance: 778.5
```

DIC for logit model with reduced parameters:

```
RL_DIC_model1
```

```
## Mean deviance:  769.8
## penalty 6.04
## Penalized deviance: 775.8
```

DIC for probit model:

```
DIC_model2
```

```
## Mean deviance:  766.2
## penalty 5.639
## Penalized deviance: 771.8
```

From the above results we can observe that the probit model fares better than logit model interms of mean deviance and penalty, which means probit model is less complex when compared to Logit model.

Comparing model 2 to model 3 has the smaller DIC, so it can be concluded that model 3 best fits the data as compared to model 2.

Smaller DIC values indicate a better-fitting model. That is, models with smaller DIC values are supported by the data. From the above density plots(of all the models), since the posterior mean is not highly skewd or bimodal, it is safe to use the DIC measure of model comparison and adequacy.

RECOVERING FEATURES FROM THE ESTIMATED MODELS

Bayesian logistic regression model

Regression models are predictive models widely used across many disciplines and applications. These models aim to fit some functional relationship between a number of categorical dependent variables (regressors) to an independent variable. The relationship between the dependent and independent variable determines the type of regression model. In situations where the independent variable only takes on two states (binary classes) we use logistic regression.

In Bayesian theory, predictions of future or missing observations are based on predictive distributions. This is giving in the Bayesian Modeling Using WinBugs book by Ntzoufras.

Data simulation from the model to predict the target variable using train and test datasets

```
library(rjags, warn.conflicts = FALSE, quietly = TRUE)
library(R2jags, warn.conflicts = FALSE, quietly = TRUE)
library(coda, warn.conflicts = FALSE, quietly = TRUE)
library(loo, warn.conflicts = FALSE, quietly = TRUE)
```

Data model for probit model

```
data_pred_probit = list(housing = bank$housing,
                        loan = bank$loan,
                        contact = bank$contact,
                        duration = bank$duration,
                        pdays = bank$pdays,
                        previous = bank$previous,
                        y = bank$y,
                        n_tr = nrow(bank),
                        phousing = bank_test$housing,
                        ploan = bank_test$loan,
                        pcontact = bank_test$contact,
                        pduration = bank_test$duration,
                        ppdays = bank_test$pdays,
                        pprevious = bank_test$previous,
                        py = bank_test$y,
                        n_ts = nrow(bank_test))

nrow(bank)
```

```
## [1] 1500
```

```
nrow(bank_test)
```

```
## [1] 1500
```

Data model for logit model

```
data_pred_logit = list(housing = bank$housing,
                       loan = bank$loan,
                       contact = bank$contact,
                       duration = bank$duration,
                       pdays = bank$pdays,
                       previous = bank$previous,
                       y = bank$y,
                       n_tr = nrow(bank),
                       phousing = bank_test$housing,
                       ploan = bank_test$loan,
                       pcontact = bank_test$contact,
                       pduration = bank_test$duration,
                       ppdays = bank_test$pdays,
                       pprevious = bank_test$previous,
                       py = bank_test$y,
                       n_ts = nrow(bank_test))
```

PRED PROBIT MODEL

```
# PREDICTION WITH PROBIT MODEL
pred_probit_model =  "model {

#Likelihood Function
for(i in 1:n_tr)
{ #likelihood function

theta[i] <- beta[1]*housing[i] + beta[2]*loan[i] + beta[3]*contact[i] + beta[4]*duration[i] + beta[5]*pdays[i] +
beta[6]*previous[i]

mu[i] <- phi(theta[i])

y[i] ~ dbern(mu[i])

}

#Priors
for(j in 1:6){
beta[j] ~ dnorm(0, 0.2)
}

# Prediction
for (k in 1:n_ts)
{
  logit(p[k]) <- beta[1]*phousing[k] + beta[2]*ploan[k] + beta[3]*pcontact[k] + beta[4]*pduration[k] + beta[5]*pp
days[k] + beta[6]*pprevious[k]

  y_pred[k] ~ dbern(p[k])
}
}
"
```

Pred Probit Model JAGS model

```
pred_probit_jags_model = jags.model(textConnection(pred_probit_model), data = data_pred_probit,n.chains = 3,n.ada
pt = 500)
```

```
## Warning in jags.model(textConnection(pred_probit_model), data =
## data_pred_probit, : Unused variable "py" in data
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1500
##    Unobserved stochastic nodes: 1506
##    Total graph size: 26465
##
## Initializing model
```

```
probit_parameters = c("y_pred")

update(pred_probit_jags_model, 1000)
pred_probit_jags_sample = coda.samples(model=pred_probit_jags_model,
                        variable.names=probit_parameters,
                        n.iter=4000)
```

MCMC Object

```
pred_probit_mcmc_model = as.mcmc(do.call(rbind, pred_probit_jags_sample))
```

Probit model calculation of RMSE : (PREDICTED - ACTUAL)

```
library('Metrics')
```

```
## Warning: package 'Metrics' was built under R version 3.4.4
```

```
P_MSE_BETA1 = rmse(pred_probit_mcmc_model[,1],bayesian_mcmc2_model2[,1])
P_MSE_BETA2 = rmse(pred_probit_mcmc_model[,2],bayesian_mcmc2_model2[,2])
P_MSE_BETA3 = rmse(pred_probit_mcmc_model[,3],bayesian_mcmc2_model2[,3])
P_MSE_BETA4 = rmse(pred_probit_mcmc_model[,4],bayesian_mcmc2_model2[,4])

P_MSE_BETA5 = rmse(pred_probit_mcmc_model[,5],bayesian_mcmc2_model2[,5])
P_MSE_BETA6 = rmse(pred_probit_mcmc_model[,6],bayesian_mcmc2_model2[,6])
```

```
cat("Mean Squared Error for beta 1 : ", P_MSE_BETA1, "\n")
```

```
## Mean Squared Error for beta 1 :  0.7817642
```

```
cat("Mean Squared Error for beta 2 : ", P_MSE_BETA2, "\n")
```

```
## Mean Squared Error for beta 2 :  1.031084
```

```
cat("Mean Squared Error for beta 3 : ", P_MSE_BETA3, "\n")
```

```
## Mean Squared Error for beta 3 :  0.6941541
```

```
cat("Mean Squared Error for beta 4 : ", P_MSE_BETA4, "\n")
```

```
## Mean Squared Error for beta 4 :  0.1996444
```

```
cat("Mean Squared Error for beta 5 : ", P_MSE_BETA5, "\n")
```

```
## Mean Squared Error for beta 5 :  0.3754669
```

```
cat("Mean Squared Error for beta 6 : ", P_MSE_BETA6, "\n")
```

```
## Mean Squared Error for beta 6 :  0.527059
```

Probit pred model DIC calculation

```
P_DIC_PRED_MODEL = dic.samples(pred_probit_jags_model,n.iter=1000)
P_DIC_PRED_MODEL
```

```
## Mean deviance:  734.6
## penalty 5.491
## Penalized deviance: 740.1
```

Just to make a valid comparison to understand which model is performing better (in other ways in terms of recovering the features of the data), let us implement the above procedure to logit model as well

Predictions for target variable (y) with Logit model

```
pred_logit_model <- "model{

   # Likelihood

   for(i in 1:n_tr){

    logit(q[i]) <- beta[1]*housing[i] + beta[2]*loan[i] + beta[3]*contact[i] + beta[4]*duration[i] + beta[5]*pday
s[i] + beta[6]*previous[i]

    y[i] ~ dbern(q[i])

   }

   # Priors

   for(j in 1:6){
    beta[j] ~ dnorm(0, 0.2)
   }

   # Prediction

   for(k in 1:n_ts){

    logit(p[k]) <- beta[1]*phousing[k] + beta[2]*ploan[k] + beta[3]*pcontact[k] + beta[4]*pduration[k] + beta[5]*
ppdays[k] + beta[6]*pprevious[k]

    y_pred[k] ~ dbern(p[k])

   }

  }"
```

Pred Logit Model JAGS model

JAGS object in MCMC.LIST format

```
pred_logit_jags_model = jags.model(textConnection(pred_logit_model), data = data_pred_logit,n.chains=3,n.adapt =
500)
```

```
## Warning in jags.model(textConnection(pred_logit_model), data =
## data_pred_logit, : Unused variable "py" in data
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1500
##    Unobserved stochastic nodes: 1506
##    Total graph size: 26163
##
## Initializing model
```

Updating the model with burnin and drawing samples from posterior distribution using 4000 iterations

```
logit_parameters = c("y_pred")

update(pred_logit_jags_model, 1000)
pred_logit_jags_sample = coda.samples(model = pred_logit_jags_model,
                        variable.names = logit_parameters,
                        n.iter = 4000)
```

MCMC object

```
pred_logit_mcmc_model = as.mcmc(do.call(rbind, pred_logit_jags_sample))
```

Logit model, calculation of RMSE : (predicted - actual)

```
library(Metrics, warn.conflicts = FALSE, quietly = TRUE)

L_MSE_BETA1 = rmse(pred_logit_mcmc_model[,1],RL_bayesian_mcmc1_model1[,1])
L_MSE_BETA2 = rmse(pred_logit_mcmc_model[,2],RL_bayesian_mcmc1_model1[,2])
L_MSE_BETA3 = rmse(pred_logit_mcmc_model[,3],RL_bayesian_mcmc1_model1[,3])
L_MSE_BETA4 = rmse(pred_logit_mcmc_model[,4],RL_bayesian_mcmc1_model1[,4])

L_MSE_BETA5 = rmse(pred_logit_mcmc_model[,5],RL_bayesian_mcmc1_model1[,5])
L_MSE_BETA6 = rmse(pred_logit_mcmc_model[,6],RL_bayesian_mcmc1_model1[,6])
```

```
cat("Mean Squared Error for beta 1 : ", L_MSE_BETA1, "\n")
```

```
## Mean Squared Error for beta 1 :  0.9991349
```

```
cat("Mean Squared Error for beta 2 : ", L_MSE_BETA2, "\n")
```

```
## Mean Squared Error for beta 2 :  1.614344
```

```
cat("Mean Squared Error for beta 3 : ", L_MSE_BETA3, "\n")
```

```
## Mean Squared Error for beta 3 :  0.8253375
```

```
cat("Mean Squared Error for beta 4 : ", L_MSE_BETA4, "\n")
```

```
## Mean Squared Error for beta 4 :  0.05398755
```

```
cat("Mean Squared Error for beta 5 : ", L_MSE_BETA5, "\n")
```

```
## Mean Squared Error for beta 5 :  0.1860917
```

```
cat("Mean Squared Error for beta 6 : ", L_MSE_BETA6, "\n")
```

```
## Mean Squared Error for beta 6 :  0.4217925
```

Logit DIC calculation

```
L_DIC_PRED_MODEL = dic.samples(pred_logit_jags_model,n.iter=1000)
L_DIC_PRED_MODEL
```

```
## Mean deviance:  739
## penalty 5.717
## Penalized deviance: 744.7
```

Probit model comparison with real data

```
cat("DIC calculation for the probit model with real data is : ", sum(DIC_model2$deviance))
```

```
## DIC calculation for the probit model with real data is :  766.2093
```

```
cat("DIC calculation for the probit model with predicted data is : ", sum(P_DIC_PRED_MODEL$deviance))
```

```
## DIC calculation for the probit model with predicted data is :  734.6025
```

Logit model comparison with real data

```
cat("DIC calculation for the logit model with real data is : ", sum(RL_DIC_model1$deviance))
```

```
## DIC calculation for the logit model with real data is :  769.7546
```

```
cat("DIC calculation for the logit model with predicted data is :", sum(L_DIC_PRED_MODEL$deviance))
```

```
## DIC calculation for the logit model with predicted data is : 738.9823
```

As we can observe from DIC variance, the probit model performs better in terms of recovering features

Logit model

```
colMeans(RL_bayesian_mcmc1_model1)
```

```
##      beta[1]      beta[2]      beta[3]      beta[4]      beta[5]
## -0.878234019 -1.571398439 -0.707076333  0.005390987  0.001811956
##      beta[6]
##  0.190996269
```

Probit Model

```
colMeans(bayesian_mcmc2_model2)
```

```
##       beta[1]       beta[2]       beta[3]       beta[4]       beta[5]
## -0.4805311965 -0.8880736290 -0.3819409211  0.0029204359  0.0009508598
##       beta[6]
##  0.0983527836
```

Summary of analysis:

1. We observed that parameters housing, loan, contact, duration, pdays and previous are primary contributers for the output variable y
2. We also observed that our data follows Bernouli's distribution and normal is good enough proior for our Bayesian models than other priors such as beta and Gamma because for Bernouli distribution, normal is the best conjugate prior
3. We observed that Probit model performs better compared to logit model for our dataset
4. We also got a good confidence on our hypothesis which is "Marketing call duration is a major indicator whether the client subscribes to a term deposit. In other words if the client call duration is more then there is a high probability that he is interested in taking the term deposit" from all the Bayesian analysis that we carried out in this project
5. We also observed that there isn't a significant difference between results from Frequentist and Bayesian models

REFERENCES:

1. (Ntzoufras, 2009) I. Ntzoufras, Bayesian Modeling Using WinBugs
2. Introduction to Bayesian Monte Carlo methods in WINBUGS link (https://www.stat.ubc.ca/~gavin/STEPIBookNewStyle/computing/winbugs/bayes-intro-2007-slides.pdf)
3. Bayesian Inference for Linear and Logistic Regression Parameters link (http://www.medicine.mcgill.ca/epidemiology/Joseph/courses/EPIB-668/bayesreg.pdf)
4. [Moro et al., 2011] S. Moro, R. Laureano and P. Cortez Available at: [pdf] http://hdl.handle.net/1822/14838 (http://hdl.handle.net/1822/14838) [bib] http://www3.dsi.uminho.pt/pcortez/bib/2011-esm-1.txt (http://www3.dsi.uminho.pt/pcortez/bib/2011-esm-1.txt)
5. Peter Hoff, A First Course in Bayesian Statistical Methods. Springer-Verlag Inc, 2009.
6. https://www4.stat.ncsu.edu/~reich/ABA/notes/JAGS.pdf (https://www4.stat.ncsu.edu/~reich/ABA/notes/JAGS.pdf)
7. https://r2012-bordeaux.sciencesconf.org/file/14430 (https://r2012-bordeaux.sciencesconf.org/file/14430)
8. https://www.coursera.org/lecture/mcmc-bayesian-statistics/jags-model-logistic-regression-xOkT9 (https://www.coursera.org/lecture/mcmc-bayesian-statistics/jags-model-logistic-regression-xOkT9)
9. https://rpubs.com/corey_sparks/30893 (https://rpubs.com/corey_sparks/30893)
10. https://www4.stat.ncsu.edu/~reich/ABA/code/GLM (https://www4.stat.ncsu.edu/~reich/ABA/code/GLM)
11. http://www.medicine.mcgill.ca/epidemiology/joseph/courses/EPIB-621/bayeslogit.pdf (http://www.medicine.mcgill.ca/epidemiology/joseph/courses/EPIB-621/bayeslogit.pdf)
12. https://docs.pymc.io/notebooks/GLM-logistic.html (https://docs.pymc.io/notebooks/GLM-logistic.html)
13. https://pymc-devs.github.io/pymc/theory.html#monte-carlo-methods-in-bayesian-analysis (https://pymc-devs.github.io/pymc/theory.html#monte-carlo-methods-in-bayesian-analysis)
14. https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/Bayesian%20Logistic%20Regression_bank%20marketing.ipynb (https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/Bayesian%20Logistic%20Regression_bank%20marketing.ipynb)
15. https://towardsdatascience.com/building-a-bayesian-logistic-regression-with-python-and-pymc3-4dd463bbb16 (https://towardsdatascience.com/building-a-bayesian-logistic-regression-with-python-and-pymc3-4dd463bbb16)
16. https://github.com/susanli2016/Machine-Learning-with-Python (https://github.com/susanli2016/Machine-Learning-with-Python)
17. https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/Bayesian%20Modeling%20Customer%20Support%20Response%20time.ipynb (https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/Bayesian%20Modeling%20Customer%20Support%20Response%20time.ipynb)
18. https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/Bayesian%20Statistics%20Python_PyMC3_ArviZ.ipynb (https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/Bayesian%20Statistics%20Python_PyMC3_ArviZ.ipynb)
19. https://theoreticalecology.wordpress.com/2011/12/09/mcmc-chain-analysis-and-convergence-diagnostics-with-coda-in-r/ (https://theoreticalecology.wordpress.com/2011/12/09/mcmc-chain-analysis-and-convergence-diagnostics-with-coda-in-r/)
20. https://blog.stata.com/2016/05/26/gelman-rubin-convergence-diagnostic-using-multiple-chains/ (https://blog.stata.com/2016/05/26/gelman-rubin-convergence-diagnostic-using-multiple-chains/)
21. https://stats.stackexchange.com/questions/296059/effective-sample-size-greater-than-actual-sample-size (https://stats.stackexchange.com/questions/296059/effective-sample-size-greater-than-actual-sample-size)
22. https://www.udemy.com/course/r-for-data-science-learn-r-programming-in-2-hours/learn/lecture/16028830#overview (https://www.udemy.com/course/r-for-data-science-learn-r-programming-in-2-hours/learn/lecture/16028830#overview)

23. https://www.udemy.com/course/bayesian-computational-analyses-with-r/learn/lecture/4032658#overview (https://www.udemy.com/course/bayesian-computational-analyses-with-r/learn/lecture/4032658#overview)
24. https://towardsdatascience.com/hands-on-bayesian-statistics-with-python-pymc3-arviz-499db9a59501 (https://towardsdatascience.com/hands-on-bayesian-statistics-with-python-pymc3-arviz-499db9a59501)
25. https://github.com/fonnesbeck/scipy2014_tutorial (https://github.com/fonnesbeck/scipy2014_tutorial)
26. https://github.com/AllenDowney/BayesMadeSimple/blob/master/world_cup01.ipynb (https://github.com/AllenDowney/BayesMadeSimple/blob/master/world_cup01.ipynb)
27. http://allendowney.github.io/BayesMadeSimple/ (http://allendowney.github.io/BayesMadeSimple/)
28. https://github.com/canyon289/bayesian-model-evaluation (https://github.com/canyon289/bayesian-model-evaluation)
29. https://github.com/canyon289 (https://github.com/canyon289)
30. https://github.com/canyon289/bayesian-stats-modelling-tutorial (https://github.com/canyon289/bayesian-stats-modelling-tutorial)