RAINFALL PREDICTION IN MELBOURNE

GROUP MEMBERS:

PARTH DESHMUKH (S3825055)

SUBBIAH SOUNDARAPANDIAN (S3825012)

TABLE OF CONTENTS

TABLE OF FIGURES

- ## INTRODUCTION

The weather conditions in Melbourne are famous for its four seasons in a day and this can be easily noticeable on daily basis, but that does not mean that it cannot be predicted. By considering the bigger picture of the meteorological events in this project the main aim will be to analyze the rainfall for the next day by considering the different meteorological features of the present day so that it can help the people to be prepared for the weather conditions tomorrow.

- ## METHODOLOGY

In this section we will go through a basic flow for the project which will help in understanding the analysis in a simpler way:

STEP 1: (DATASET)

The dataset is originally collected by Commonwealth of Australia, Bureau of Meteorology and we referred this dataset from the Kaggle[1] for the analysis. The dataset contains information about numerous Australian weather stations, but we are mainly focusing on Melbourne city for the rainfall prediction.

STEP 2: (DESCRIPTIVE STATISTICS)

As the dataset contains too many variables, we will remove the unwanted attributes in order to focus on the more relevant data and minimize the complications. Now, this dataset is used to generate a descriptive statistics where we will have a look over the numerical values such as mean, median, minimum and maximum for each attributes. Moreover, it will help us to impute mean and median values over the NAs.

STEP 3: (VISUAL ANALYSIS)

Visual Analysis gives you a clear picture about your data and the message it wants to convey. The visual information for all 22 attributes were performed and out of which almost 12 impactful variables were selected. To be precise with the analysis for the predictions the complexity was not negotiated, rather we used all 12 features to gain the meaningful extracts from the scatter plots. The key parameters for the variable where observed which will help in achieving the target variable and most significant variables are prioritized to make our predictions precise.

STEP 4: (MATHEMATICAL AND JAGS MODEL)

To start the analysis after the visual statistics it is important to select the appropriate approach and model for the analysis. In this project we are using JAGS model with Bernoulli distribution for dependent variable and robust logistics regression for all the independent variables. Robust logistics regression is used in this project because of the outliers present in the dataset.

STEP 5: (INTERPRETATIONS)

Interpretations are the inferences which can be gathered from the posterior distributions in order to make the predictions. In this project we will go through numerous interpretations and go through all interpretation thoroughly and see which fits the model best and finalize the results accordingly.

STEP 6: (RESULTS)

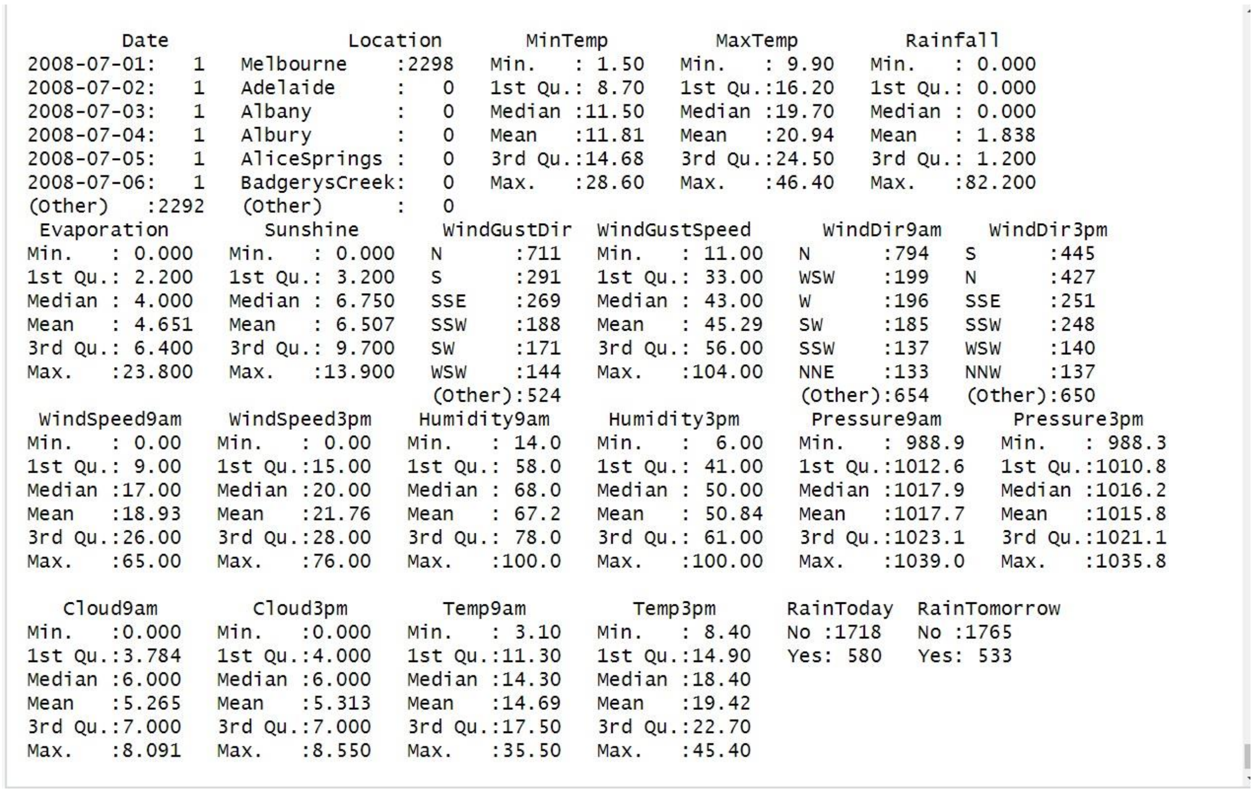
Results are basically outcomes for the analysis performed on the Bayesian model. In this part of the project we will discuss about the interpretations in detail about why this interpretation suits the best and reason behind selecting it. Moreover, we will compute the predictions with some conclusive evidence which can make more sense for the analysis.

• ANALYSIS

1. DESCRIPTIVE STATISTICS

The initial descriptive statistics are performed to describe the basic features of the dataset. The simple summaries about the rainfall samples are used to present quantitative descriptions for many attributes for Australia weather dataset which help us to simplify large amounts of data in a sensible way. The main motive for the descriptive statistic is that it reduces large data into a simpler summary with meaningful outcomes and easy way to find the NAs.

Figure-1 shows the statistics for 22 features which shows the mean, median, minimum, and maximum values for each attribute. Moreover, the dataset is filtered for Melbourne city and all the NAs are imputed with genuine values (mean and median values) to keep the data reliable and unbiased.



```
        Date                Location        MinTemp          MaxTemp          Rainfall
 2008-07-01:   1   Melbourne    :2298   Min.   : 1.50   Min.   : 9.90   Min.   : 0.000
 2008-07-02:   1   Adelaide     :   0   1st Qu.: 8.70   1st Qu.:16.20   1st Qu.: 0.000
 2008-07-03:   1   Albany       :   0   Median :11.50   Median :19.70   Median : 0.000
 2008-07-04:   1   Albury       :   0   Mean   :11.81   Mean   :20.94   Mean   : 1.838
 2008-07-05:   1   AliceSprings :   0   3rd Qu.:14.68   3rd Qu.:24.50   3rd Qu.: 1.200
 2008-07-06:   1   BadgerysCreek:   0   Max.   :28.60   Max.   :46.40   Max.   :82.200
 (Other)   :2292   (Other)      :   0
  Evaporation       Sunshine        WindGustDir  WindGustSpeed     WindDir9am    WindDir3pm
 Min.   : 0.000   Min.   : 0.000   N      :711   Min.   : 11.00   N      :794   S      :445
 1st Qu.: 2.200   1st Qu.: 3.200   S      :291   1st Qu.: 33.00   WSW    :199   N      :427
 Median : 4.000   Median : 6.750   SSE    :269   Median : 43.00   W      :196   SSE    :251
 Mean   : 4.651   Mean   : 6.507   SSW    :188   Mean   : 45.29   SW     :185   SSW    :248
 3rd Qu.: 6.400   3rd Qu.: 9.700   SW     :171   3rd Qu.: 56.00   SSW    :137   WSW    :140
 Max.   :23.800   Max.   :13.900   WSW    :144   Max.   :104.00   NNE    :133   NNW    :137
                                   (Other):524                    (Other):654   (Other):650
  WindSpeed9am    WindSpeed3pm     Humidity9am      Humidity3pm       Pressure9am      Pressure3pm
 Min.   : 0.00   Min.   : 0.00   Min.   : 14.0   Min.   :  6.00   Min.   : 988.9   Min.   : 988.3
 1st Qu.: 9.00   1st Qu.:15.00   1st Qu.: 58.0   1st Qu.: 41.00   1st Qu.:1012.6   1st Qu.:1010.8
 Median :17.00   Median :20.00   Median : 68.0   Median : 50.00   Median :1017.9   Median :1016.2
 Mean   :18.93   Mean   :21.76   Mean   : 67.2   Mean   : 50.84   Mean   :1017.7   Mean   :1015.8
 3rd Qu.:26.00   3rd Qu.:28.00   3rd Qu.: 78.0   3rd Qu.: 61.00   3rd Qu.:1023.1   3rd Qu.:1021.1
 Max.   :65.00   Max.   :76.00   Max.   :100.0   Max.   :100.00   Max.   :1039.0   Max.   :1035.8

    Cloud9am        Cloud3pm         Temp9am          Temp3pm       RainToday  RainTomorrow
 Min.   :0.000   Min.   :0.000   Min.   : 3.10   Min.   : 8.40   No :1718   No :1765
 1st Qu.:3.784   1st Qu.:4.000   1st Qu.:11.30   1st Qu.:14.90   Yes: 580   Yes: 533
 Median :6.000   Median :6.000   Median :14.30   Median :18.40
 Mean   :5.265   Mean   :5.313   Mean   :14.69   Mean   :19.42
 3rd Qu.:7.000   3rd Qu.:7.000   3rd Qu.:17.50   3rd Qu.:22.70
 Max.   :8.091   Max.   :8.550   Max.   :35.50   Max.   :45.40
```

Figure-1 Descriptive summary

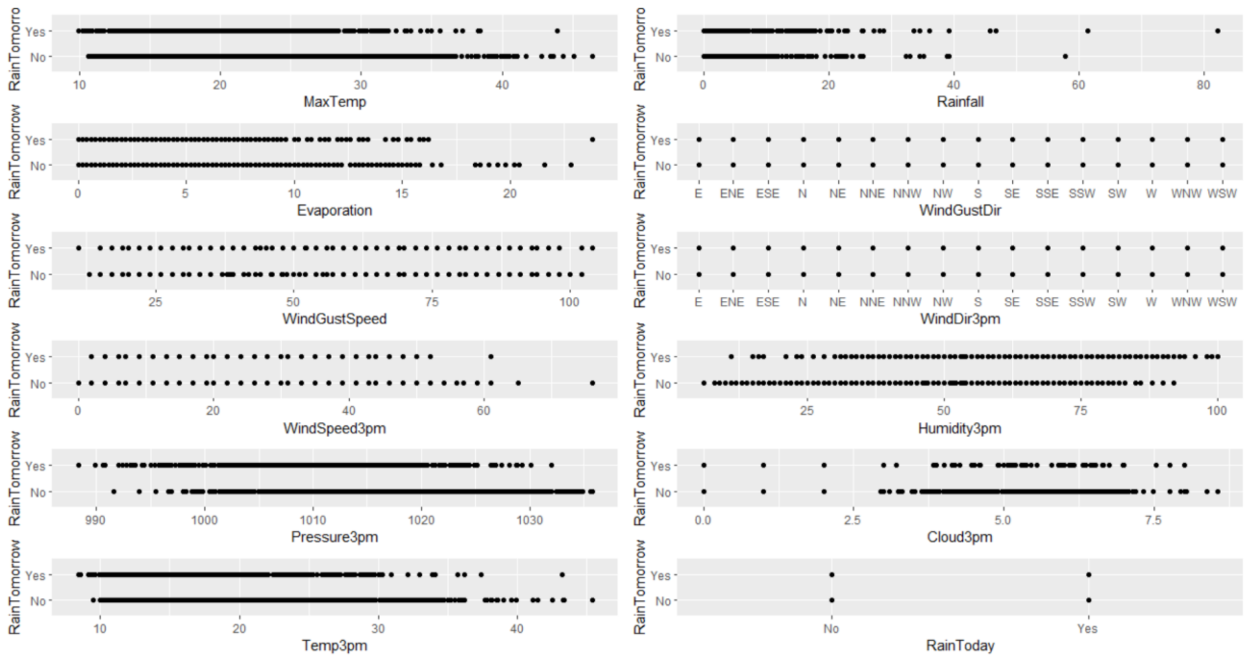


Figure-2 Visualization summary

The visualization graphs show a clear view about the most significant variables and least significant variables. As it can be examined right portion of the figure-2 shows more significant graphs as compared to the left.

For example, The MaxTemp variable says that it is less probable to rain tomorrow if the maximum temperature increases more than 40 whereas if the maximum temperature tends to be below 10 it is mostly likely to rain. Similarly, it can be diagnosed from other 5 graphs below it.

The MaxTemp variable gives an overview of maximum temperature for each day over a period of 10 years where we can see that it ranges from 1.50 to 46.40 with mean and median values almost similar 19.7 and 20.9, respectively. The first and third quartile for the max temp is observed to be 16.2 and 24.5 which means that 25% of values are below 16.2 while 25% of values are higher than 24.5.

The Evaporation variable gives an overview of water evaporated for each day over a period of 10 years, more the rate of evaporation more likely it is to rain tomorrow. The first and third quartile for the evaporation is observed to be 2.2 and 6.4 with mean value around 4.65.

The WindGust Speed variable provides an estimate of wind speed for each day over a period of 10 years, wind speed tends to be higher most of time with minimum and maximum of 11 and 104, respectively. The first and third quartile for the WindGust Speed is 33 and 46 respectively with mean value around 45.29.

Three different parameters are observed at 3pm time zone which are WindSpeed3pm, Pressure3pm and Temp3pm. These are the most significant variables which shows a higher impact towards rainfall according to the visualization graphs. The WindSpeed3pm generally tends to be in between 0 – 76 with its first and third quartile ranging 15 and 28 respectively, Temp3pm ranging from 8.4 to 45.4 with mean around 19.42 and Pressure3pm around 988.3 – 1035.8 with mean 1015.8.

2. MATHEMATICAL MODEL

Mathematical model inculcates the real-world data into the equations. These equation helps us predict the outcomes for the analysis. So, to predict the outcome for analysis a mathematical model is designed in a mathematical form to numerically calculate the predictions for next day by considering the present-day conditions.
The target variable is a binary response variable so we will estimate it using Bernoulli Likelihood. The Logistic regression function is used for the parameters which is expanded in the form of multiple independent variables and regression parameters.

$$Y \sim \text{Bernoulli}(\theta)$$

$$\begin{aligned}\theta = \text{logistic}\ (&\beta_0 + \beta_1\text{MinTemp} + \beta_2\text{MaxTemp} + \beta_3\text{Rainfall} + \\ &\beta_4\text{Evaporation} + \beta_5\text{Sunshine} + \beta_6\text{WindGustDir} + \beta_7\text{WindGustSpeed} + \\ &\beta_8\text{WindDir9am} + \beta_9\text{WindDir3pm} + \beta_{10}\text{WindSpeed9am} + \beta_{11}\text{WindSpeed3pm} + \\ &\beta_{12}\text{Humidity9am} + \beta_{13}\text{Humidity3pm} + \beta_{14}\text{Pressure9am} + \beta_{15}\text{Pressure3pm} + \\ &\beta_{16}\text{Cloud9am} + \beta_{17}\text{Cloud3pm} + \beta_{18}\text{Temp9am} + \beta_{19}\text{Temp3pm} + \beta_{20}\text{RainToday})\end{aligned}$$

3. JAGS MODEL DIAGRAM

Figure-2 shows the full structured model for the Analysis. The target variable for the prediction in this project is a binary so we will use Bernoulli Likelihood distribution for the dependent variable as it is a discrete variable. Further, Robust logistic regression is applied to the success probability of the likelihood with normal priors for all the regression parameters beta so that it can take any values between negative infinity to positive infinity. The use of Robust logistic regression and guess parameter is due to the outliers in the dataset.

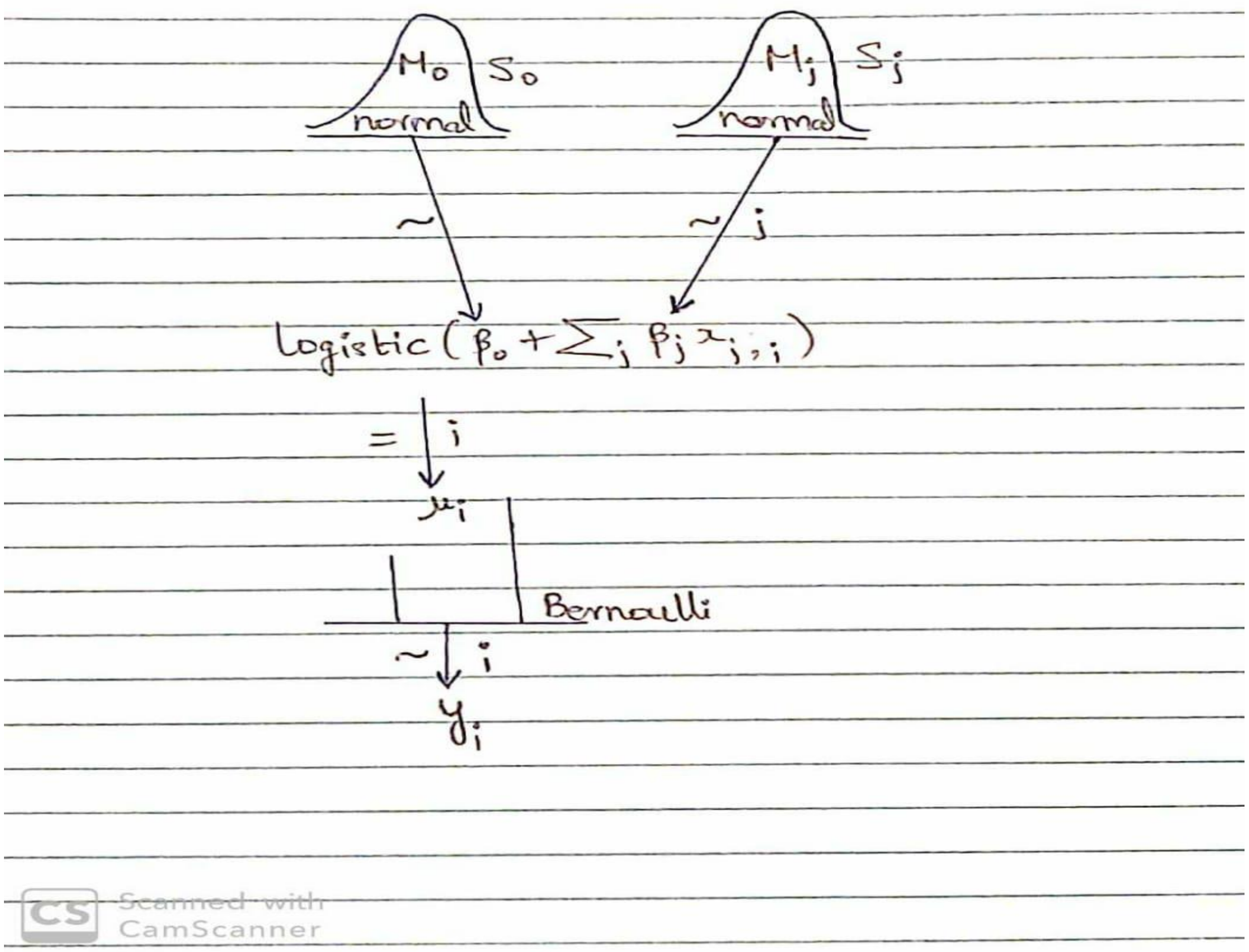


Figure-3 JAGS Model Diagram

4. PRIOR DISTRIBUTION

Prior Distribution can be mainly categorized into two parts:

1] Informative Prior

2] Non-Informative Prior

Informative prior is basically assigning the probabilities to the model by using some expert information. Expert knowledge is the information about the similar events in the past which can be used in our current model to make the predictions. However, Non-Informative prior tends to have no prior expert knowledge.

In this project as we do not have any prior expert knowledge we will use a non-informative prior distribution and Normal prior is given to all the regression parameters (beta) so that it can take any values from negative infinity to positive infinity.

Since we are assuming our prior to be a non-informative we are assuming a value as 2 in our non-informative normal prior distribution.

As the data consist of some outliers which tends to bend the S-shape logistic regression curve we will introduce a guessing parameter which will help us to control the stiffness of the curve at an acceptable level.

5. POSTERIOR DISTRIBUTION AND INTERPRETATION

Coming to the posterior distributions, we will find them using the rjags and runjags library in R statistical software. The R Code used to find the predictions using the Bayesian statistical analyses has been included in the appendix section of the report (Please refer to Appendix A). The PlotMCMC function and summaryMCMC function is taken from the textbook and implemented to find the required predictions of whether the rainfall would happen or not based on the previous day's meteorological observations.

The dataset is imported using read.csv function under the library 'base r'. The NA values present in the data are imputed with mean values. And in some cases, the missing values are replaced with normal distribution data with mean equal to the mean of the variable. This is done to avoid autocorrelations and thereby getting less auto-correlated outputs in MCMC sampler. The variables "WindGustDir" and "WindDir9am" are categorical variables with multiple factors. So using prop.table() function, we are finding out which factor has most contributed to rain and that variable is set to 1 and others are set to 0. Hence it turns into an indicator variable.

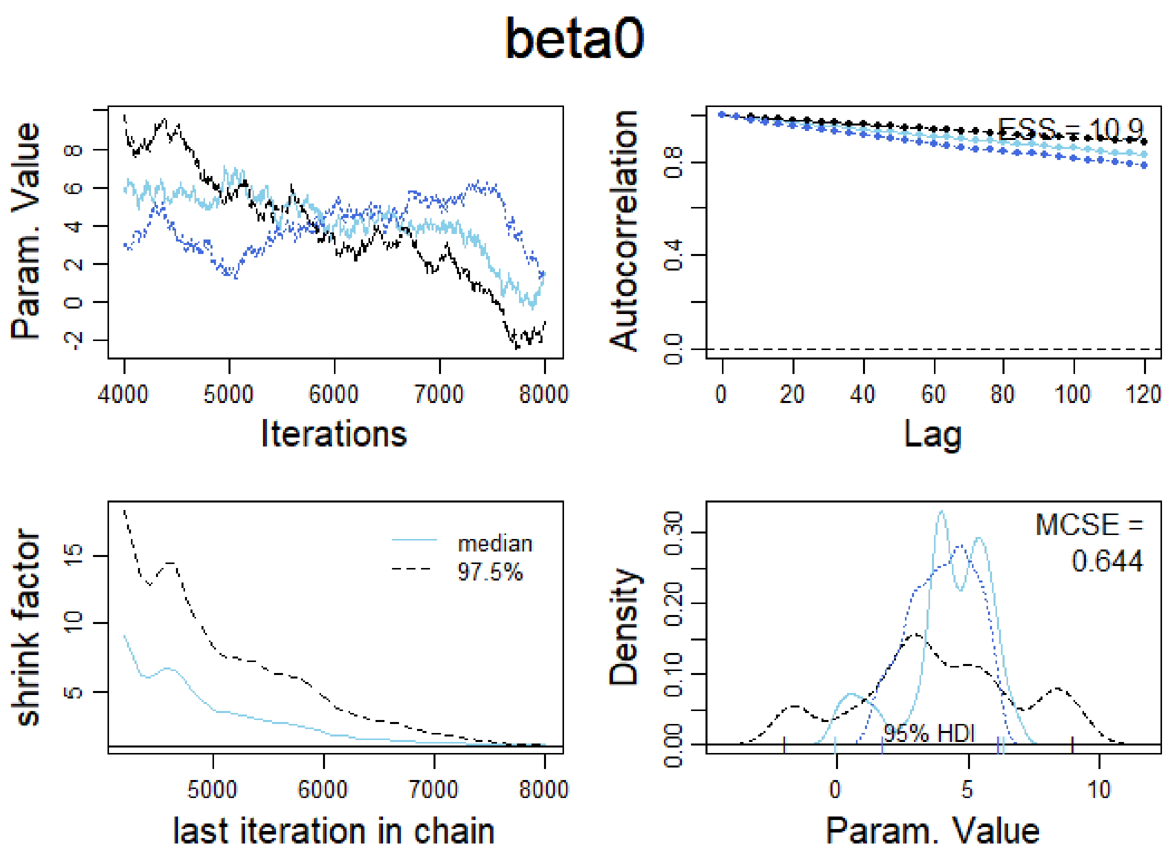
For example, "WindGustDir" variable is turned to Variable that has only 'North' as '1'and not having north values as '0'. After doing the required data pre-processing, the dataset is split into test data and training data. Model will learn using logistic regression through the training data and use the resulting Bayesian estimates of beta coefficients to predict the values in test data.The priors are defined and written to a model text. It will be used by run.jags() function under the runjags library in r. Guess parameter are introduced to group the outliers in all variables. It is given a non-informative beta prior in the model text. As mentioned earlier, for all other parameters, since we have no expert information, we are going with non-informative prior.

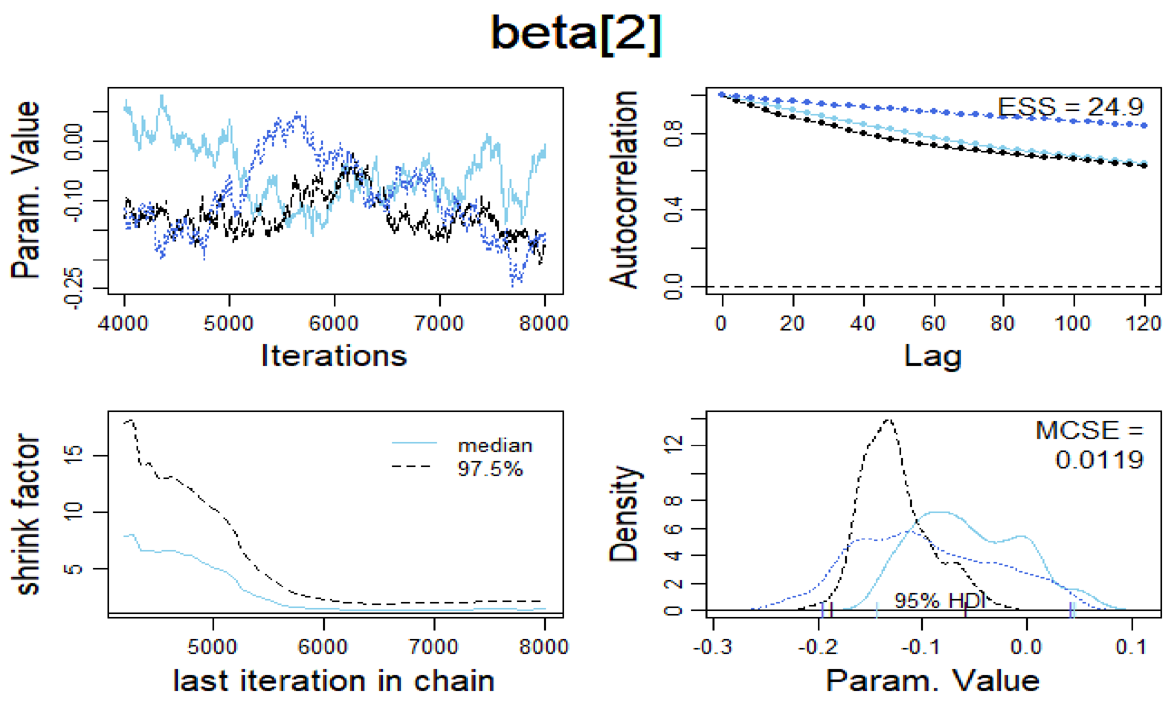
Then diagMCMC function will help us to do the diagnostic MCMC check on the MCMC algorithm done using Just Another Gibs Sampler. The Jags model is fed with values like adaptation steps, burn-in steps, number of chains, thinning steps, and number of saved steps. These values are adjusted for each run, until we get MCMC outputs with high accuracy and representativeness of the sample. Once we get the desired output, we get the posterior Bayesian estimates with which we can predict the response variable of the test data.

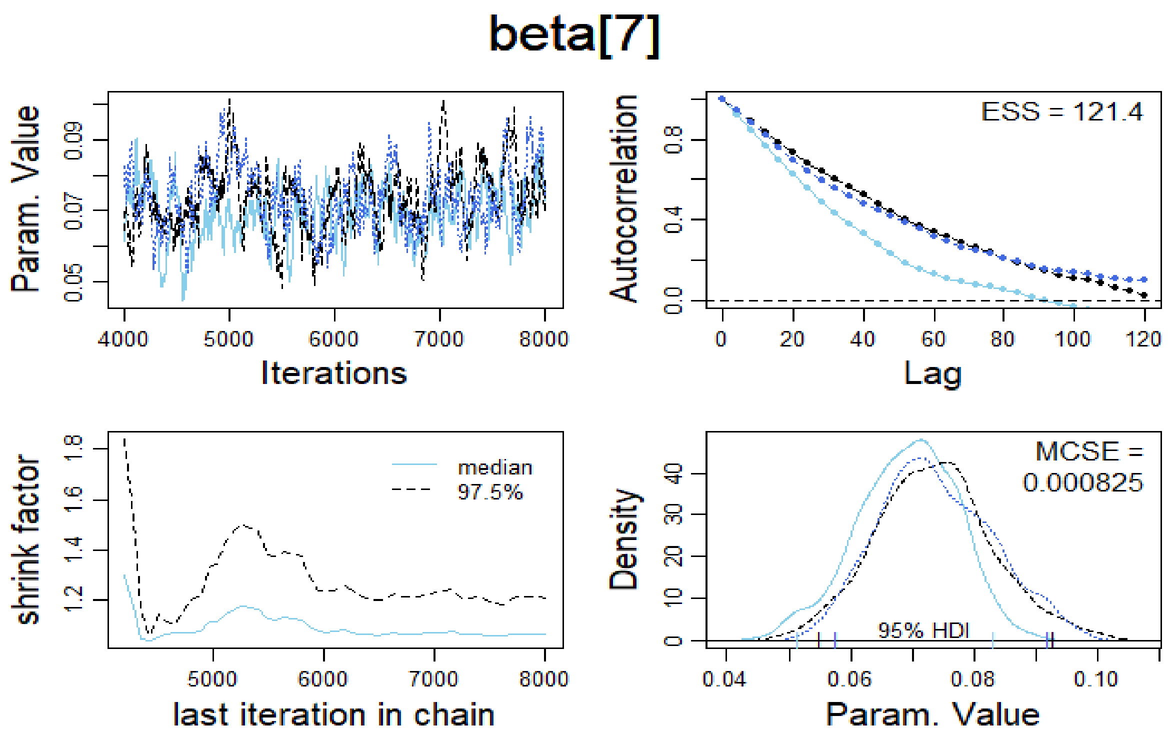
Interpretation of Results:

- As a first step, we feed the following set of parameters to the JAGS model and get the output.
- For the first run, the explanatory variables' values are not standardized.
- The first set of parameters is as follows,
  adaptSteps = 1000
  burnInSteps = 3000
  nChains = 3
  thinSteps = 4
  numSavedSteps = 1000

- We run the JAGS model parallelly for given number of chains to save the time. Hence we pass parameter method = "parallel" to the run.jags function which would set the jags model to work parallelly.
- Below are the few of the MCMC diagnostics that we got by feeding the first set of values to JAGS Model.


beta0
Param. Value
Iterations
Autocorrelation
Lag
ESS = 10.9
shrink factor
last iteration in chain
median
97.5%
Density
MCSE = 0.644
95% HDI
Param. Value


beta[2]
Param. Value
Iterations
Autocorrelation
Lag
ESS = 24.9
shrink factor
last iteration in chain
median
97.5%
Density
MCSE = 0.0119
95% HDI
Param. Value



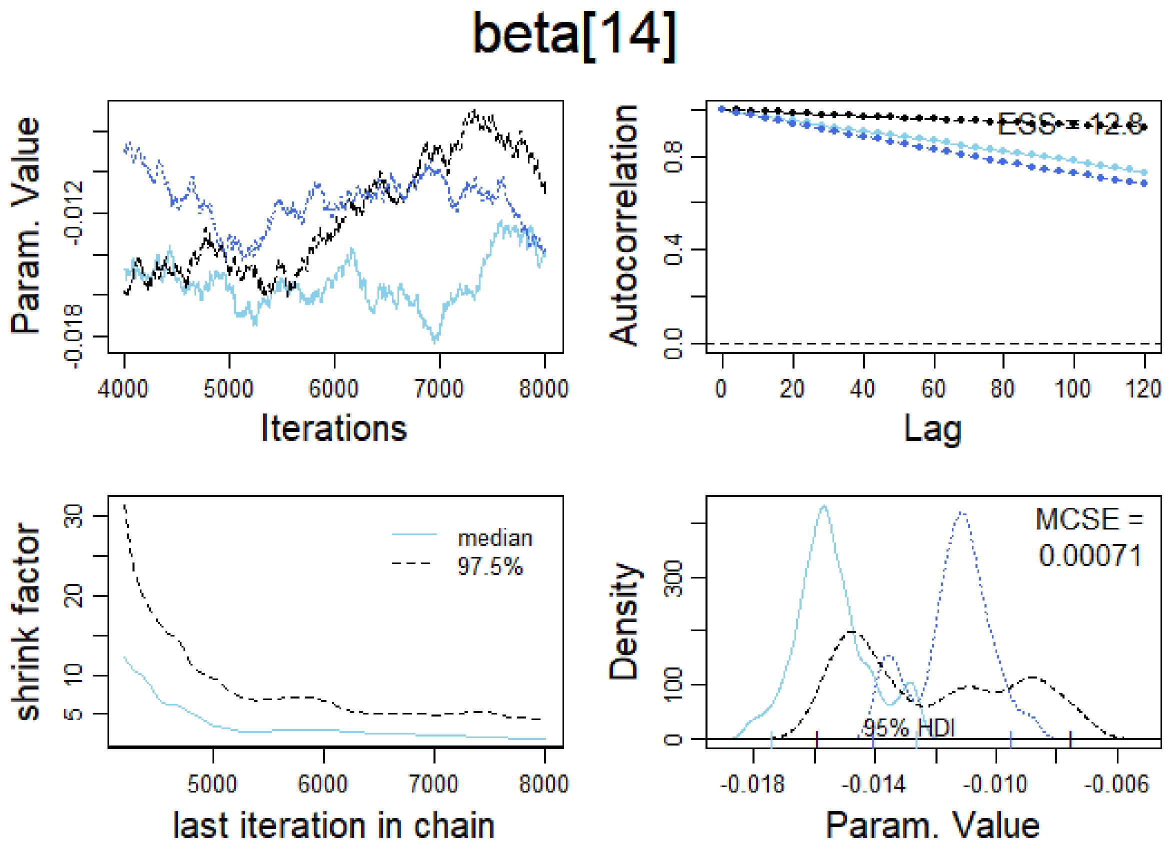
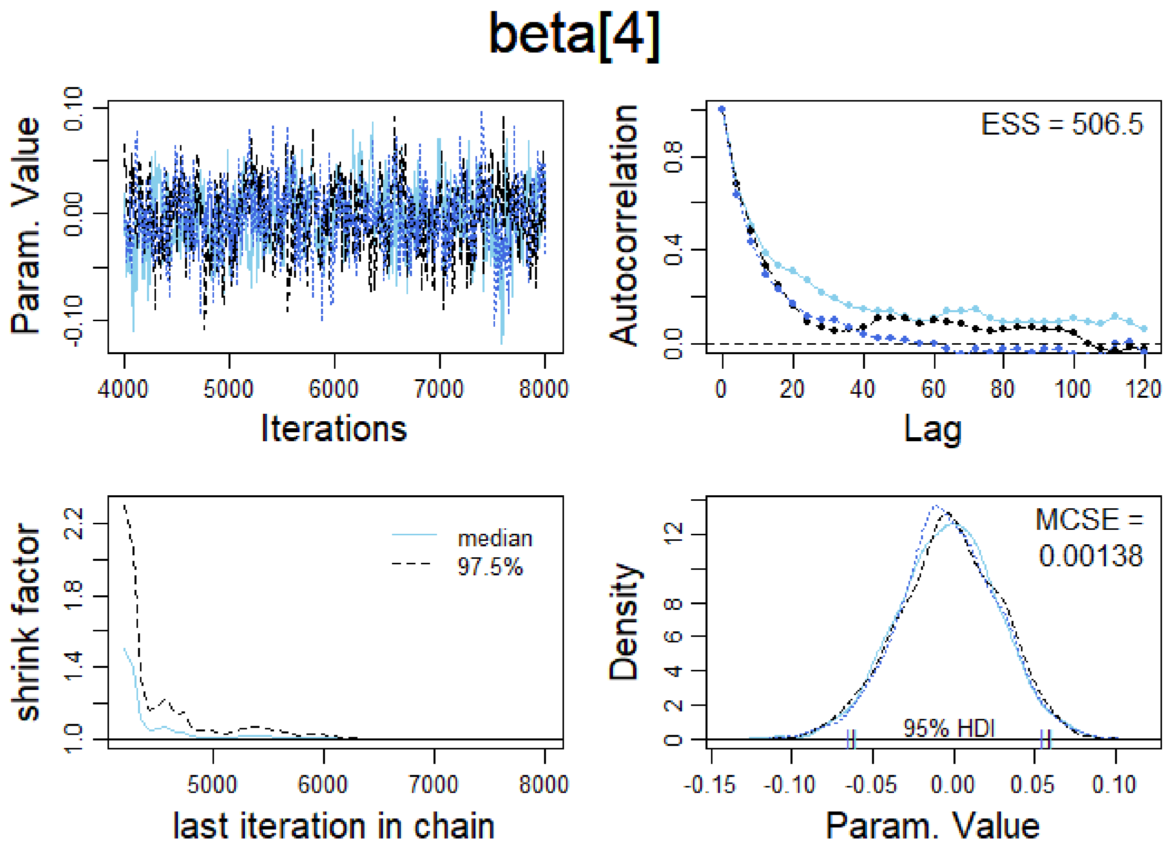
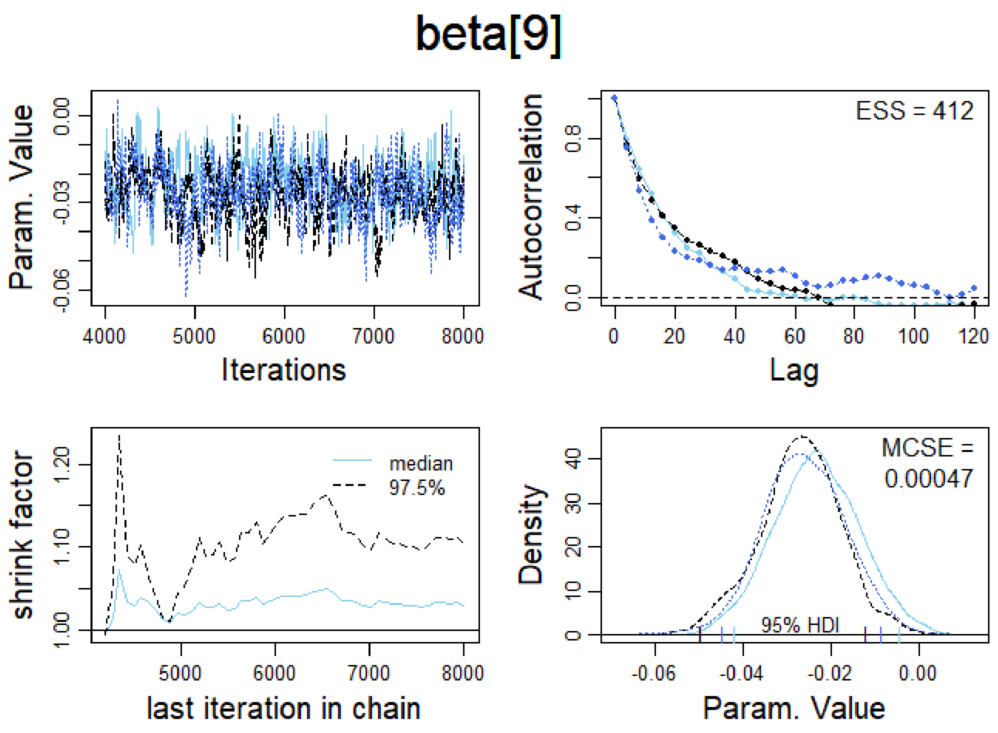


Figure-4 MCMC Diagnostics of all the coefficients - First Run

- Figure-4 shows the MCMC diagnostics for the first set of values that are given as input to JAGS.
- We could see that MCMC diagnostic of beta0 has very high shrink factor. Shrink factor greater than 1.2 are not recommended but in our case, it was around 5. Hence it shows that there is a problem with representativeness of the chains.
- Trace Plot also shows the failure of convergence of chains. Chains do not overlap and mix well and hence shows the problem of poor representativeness of the chains.
- The density Plot also can be used to assess the representativeness of the chains. For a chain of high representativeness, chains should overlap at the burn in period, in the density plot. But in our case, it is not so. Hence it needs to be taken care.
- The autocorrelation plot can be interpreted to assess the accuracy of the chains. Lower the autocorrelation, higher the accuracy of the chains.
- The autocorrelation in diagnostic of all the beta parameters seems to be very high in this case. Hence, we should go with different set of values for the next run.
- Since the first run with unstandardized values fails to meet our requirements, the next run is done with standardized values of independent variables.

The next run is ran with same set of values and below are few of the diagnostics obtained,





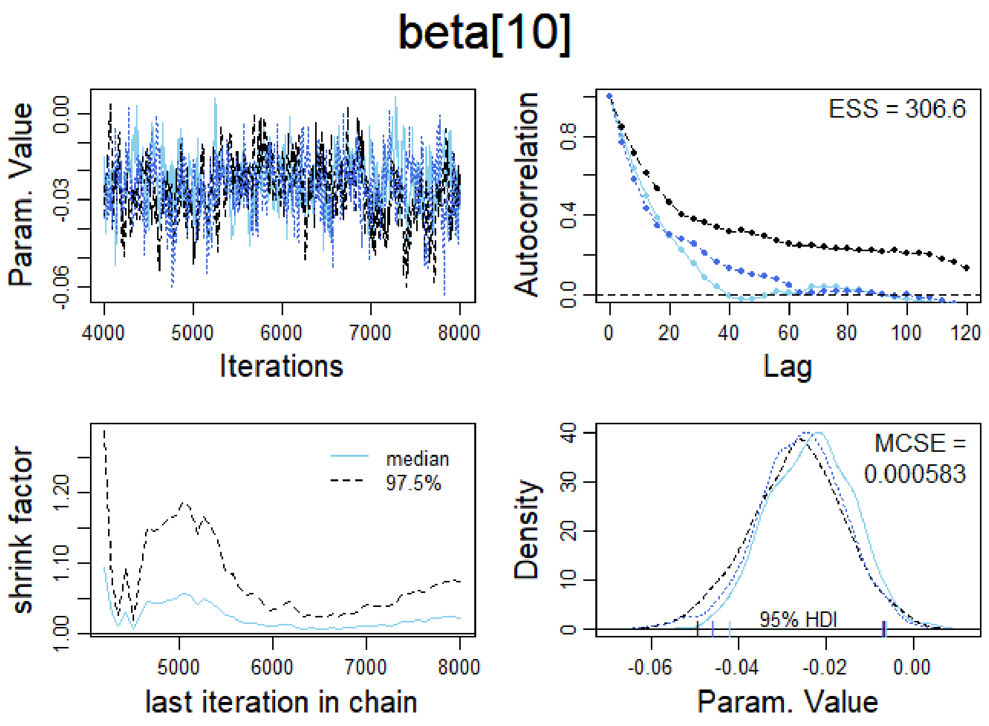


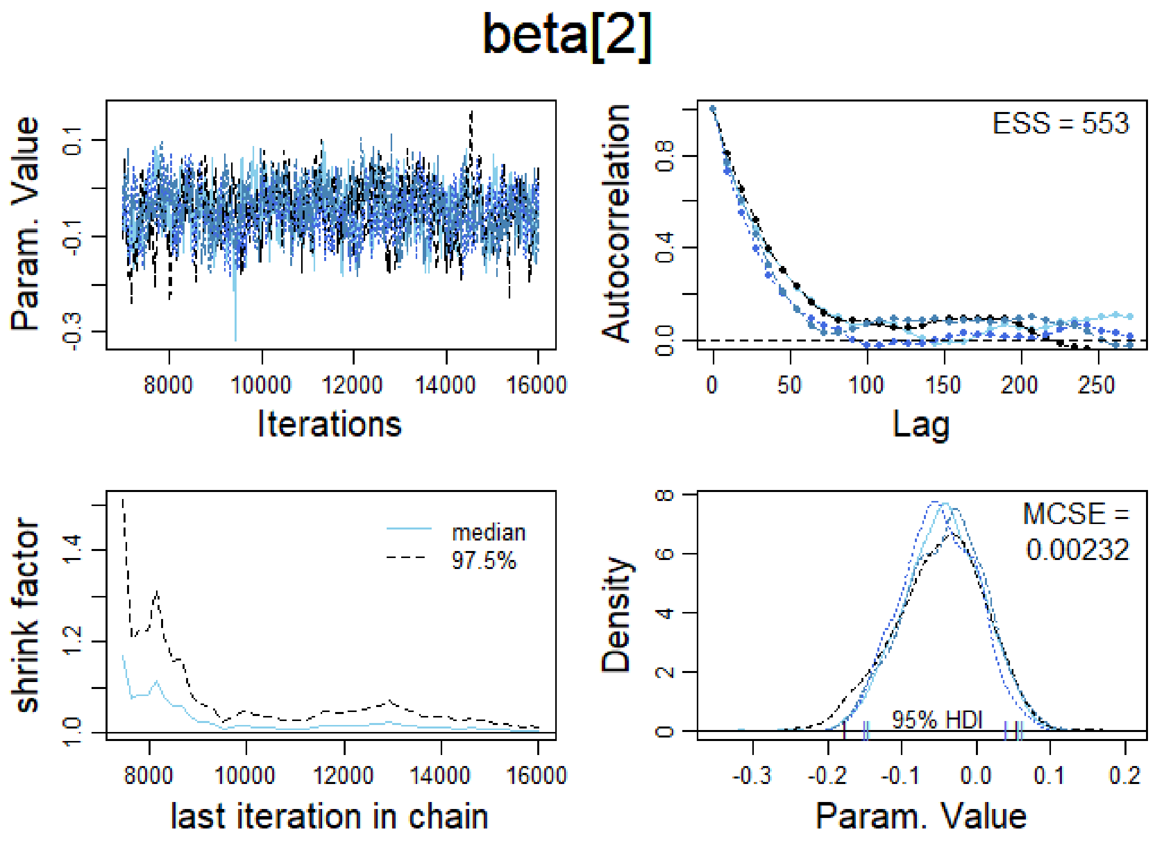
Figure-5 MCMC Diagnostics of all the beta coefficients - Second Run

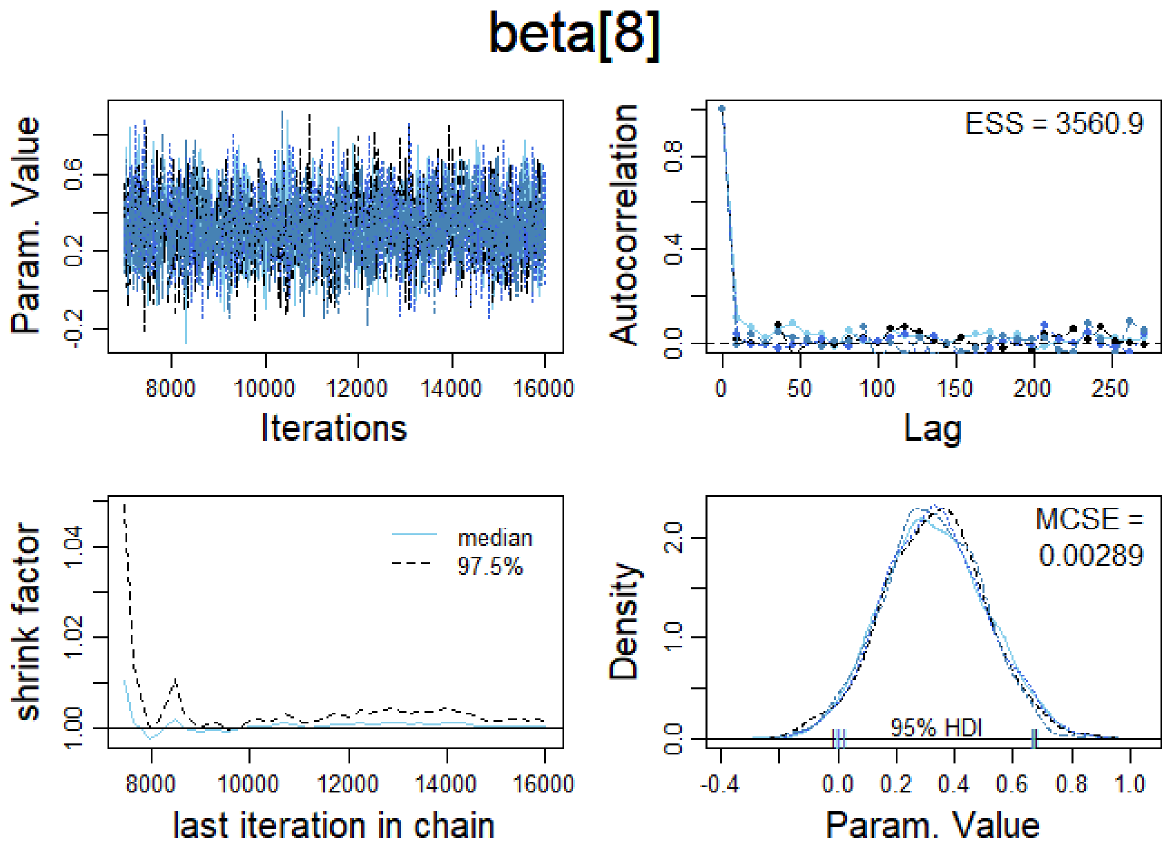
- Figure-5 shows the MCMC diagnostics for the first set of values with standardised dependent variables that are given as input to JAGS.
- There is a bit of improvement here compared to the figure 1 but standardization has not helped to a greater extent.
- The Shrink factor's upper quantile still seems to hover around 1.2 for all the coefficients, while the ideal value is below 1.2. Hence it is much needed to increase the number of chains to fix the shrink factor issue.
- Trace Plot shows us a bit of improvement compared to figure 1, but still in some parts, the chains have got stuck in unrepresentative subset of parameter space. This is not recommended to proceed with Bayesian estimates.
- Hence as a remedy, we can try to increase the burn-in steps to fix the convergence of chains towards the mean value.
- The autocorrelation is also high with less effective sample size. Autocorrelation would affect the accuracy of the estimates since it will not provide independent information at the successive steps of the chains.
- Thinning steps can be introduced to fix the autocorrelation issue and thereby proceeding with Bayesian estimates of our beta coefficients.

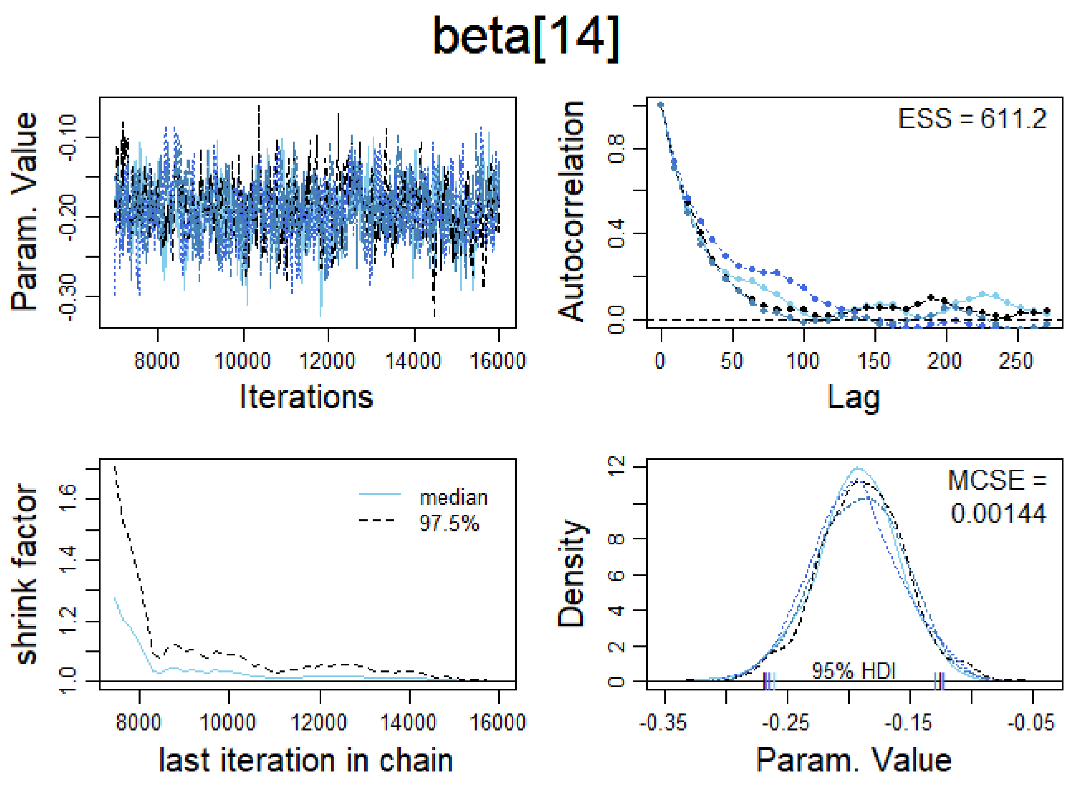
As discussed above, the next run is ran with improved values to get the best MCMC diagnostics of beta coefficients. The following set of values are fed to JAGS,

```
adaptSteps = 1000
burnInSteps = 6000
nChains = 4
thinSteps = 9
numSavedSteps = 1000
```

For the above values, we got the following diagnostics. Presenting the few of the outputs in the next page,


beta[2]
Param. Value
Iterations
Autocorrelation
Lag
ESS = 553
shrink factor
last iteration in chain
median
97.5%
Density
Param. Value
MCSE =
0.00232
95% HDI


beta[8]
Param. Value
Iterations
Autocorrelation
Lag
ESS = 3560.9
shrink factor
last iteration in chain
median
97.5%
Density
Param. Value
MCSE =
0.00289
95% HDI



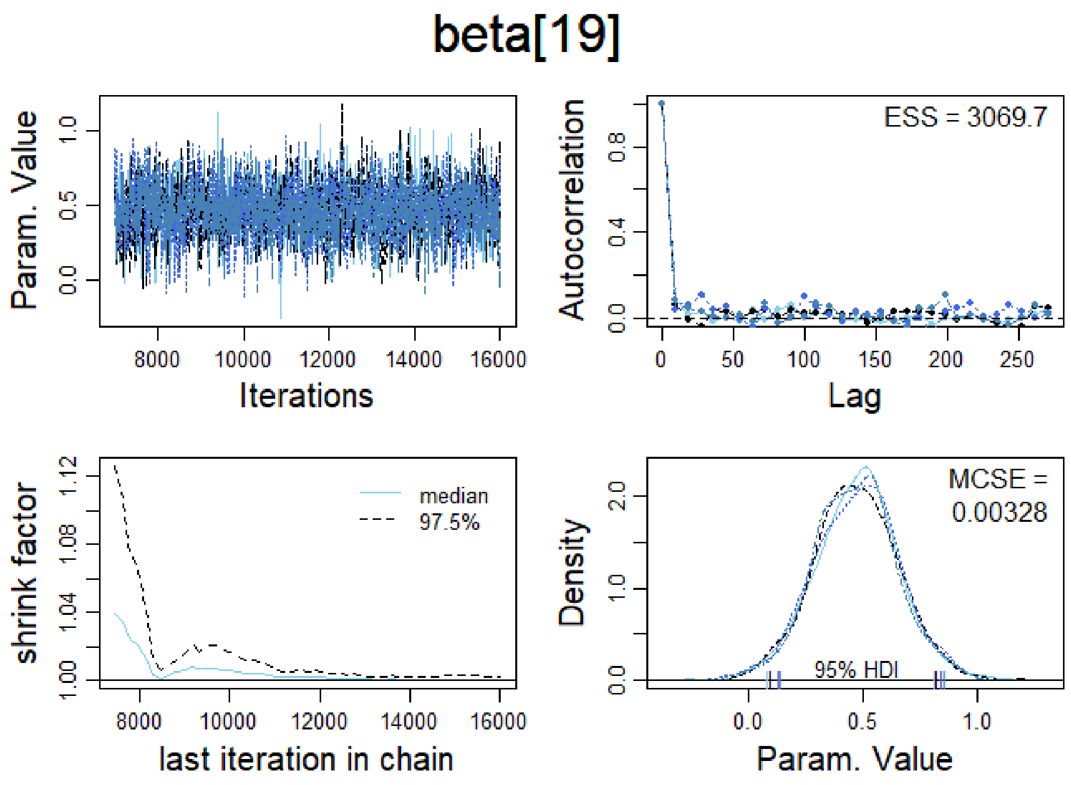


Figure-6 MCMC Diagnostics of all the beta coefficients - Third Run

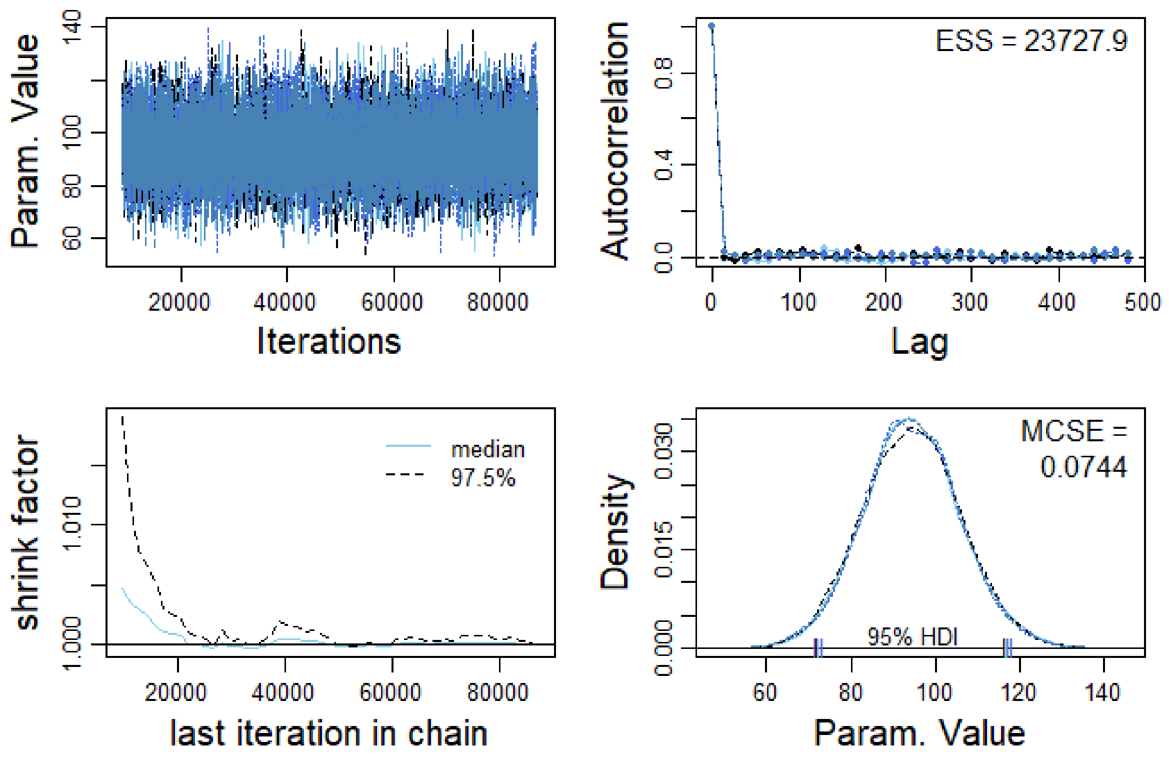
- After we change the values, Figure-6 shows that there is a mix of good and bad diagnostics.
- Few samples of diagnostics have been reported. Beta8 and beta19 show how good the diagnostics should be and the other two diagnostics show how it should not be.
- In beta8 and Beta19 diagnostics,
    - Trace Plot is good, and the chains are converging well towards the mean value.
    - The Shrink factor is below 1.2 and hence assures the representativeness of the sample.
    - In density plot, at burn-ins, all the three chains are overlapping which is recommended.
    - The autocorrelation is very low with high effective sample sizes. Hence the accuracy can be assumed high.
- Hence diagnostics of beta8 and beta19 diagnostics almost represent the ideal diagnostics.
- On the other hand, beta2 and beta14 diagnostics show,
    - Trace plot is poor with chains stuck at some places.
    - The shrink factor or Gelman-Rubin statistic is higher than 1.2 which shows that chains have not been converged properly resulting in poor representativeness.
    - The chains are not overlapping at the burn-in period, in the density plot.
    - Finally, the autocorrelation is high with less effective sample sizes. Hence the independent information at successive stops will not be there and it will finally affect the accuracy of Bayesian estimates.
- Hence a more robust model is required to get the desired MCMC diagnostics. So, we go again with increased thinning steps, number of saved steps and adaptation steps.
- Number of saved steps are increased to increase the length of the chains and thereby getting good convergence.

The final model is run with different set of improved values as follows,

```
adaptSteps = 2000
burnInSteps = 6000
nChains = 4
thinSteps = 9
numSavedSteps = 6000
```

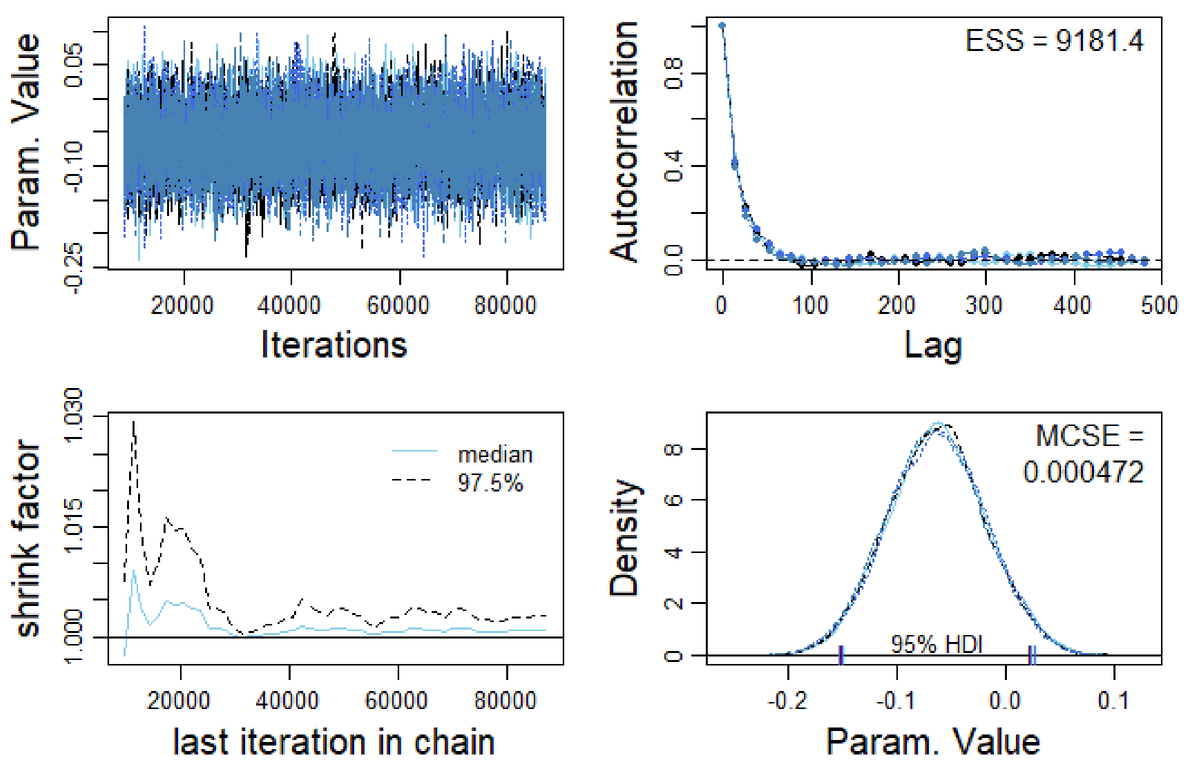
We get the following MCMC diagnostics. Since it is a successful run, all MCMC diagnostics are reported,

beta0

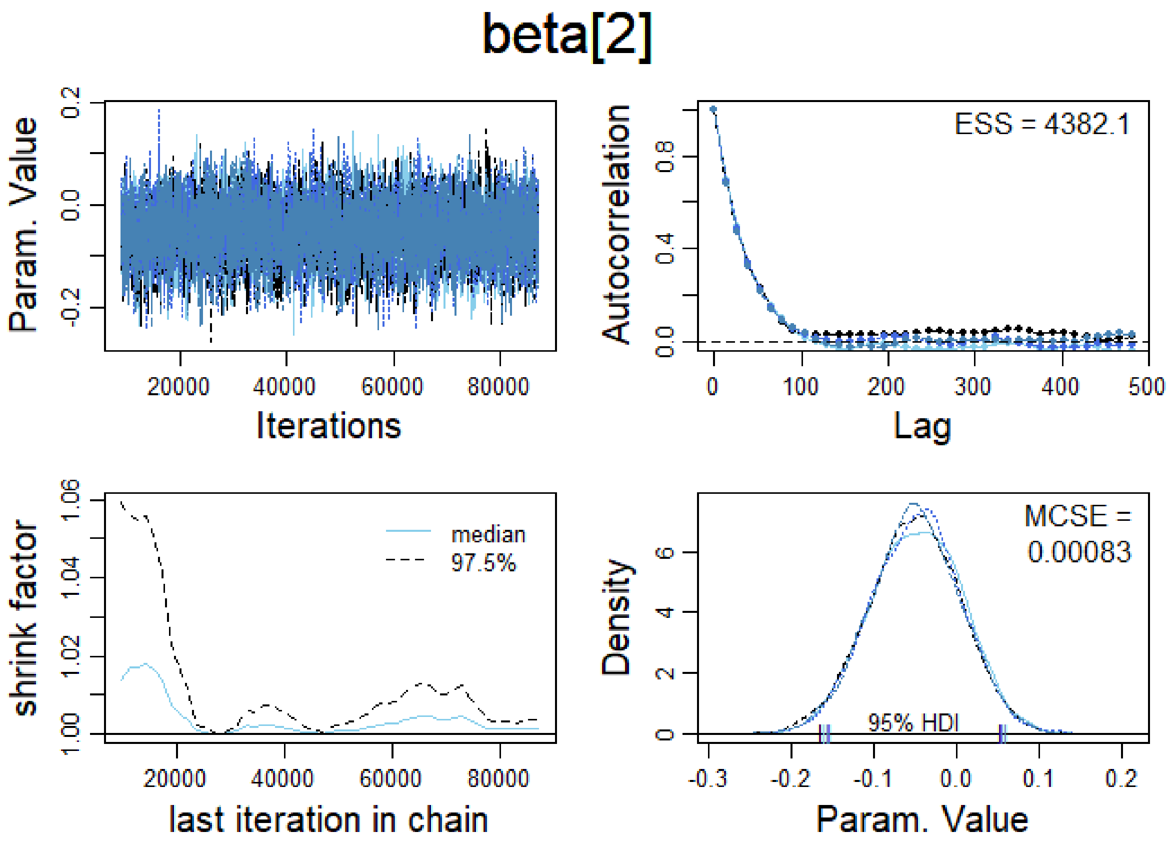


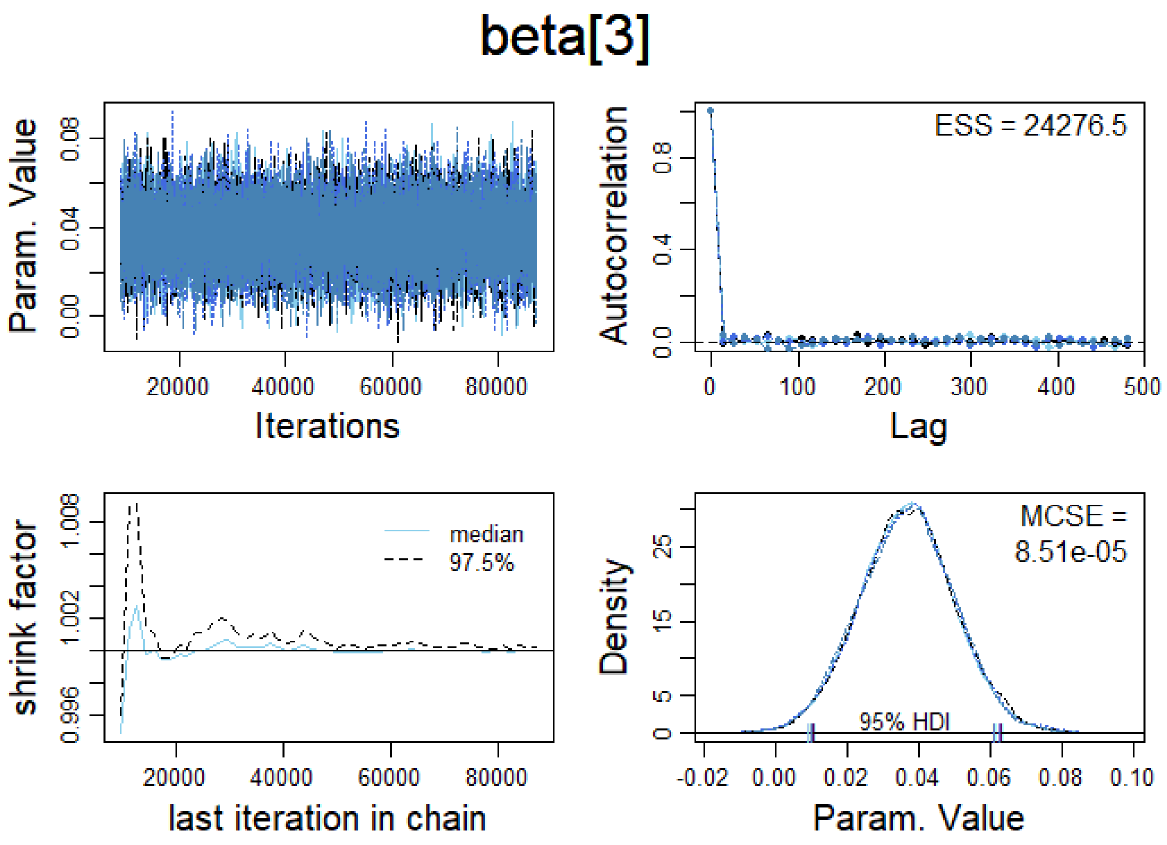
Param. Value
Iterations
Autocorrelation
ESS = 23727.9
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE =
0.0744
95% HDI
Param. Value

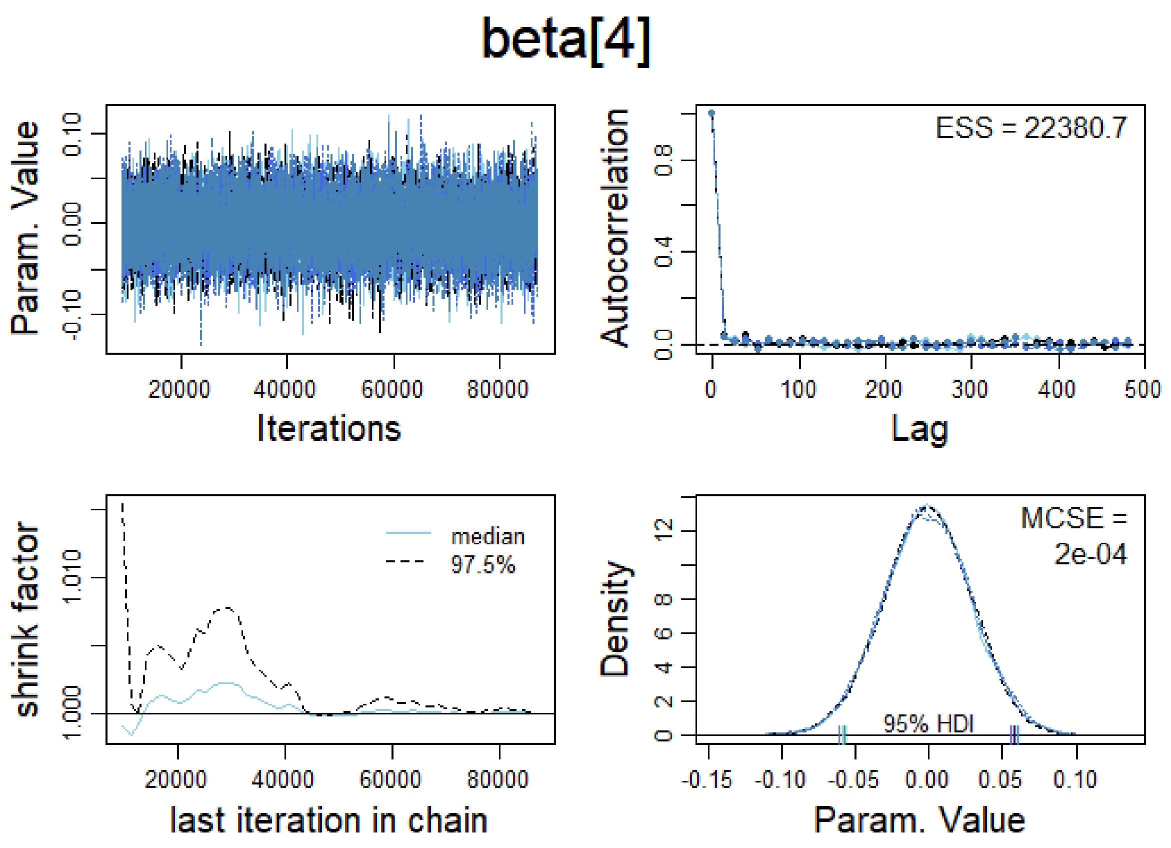
beta[1]

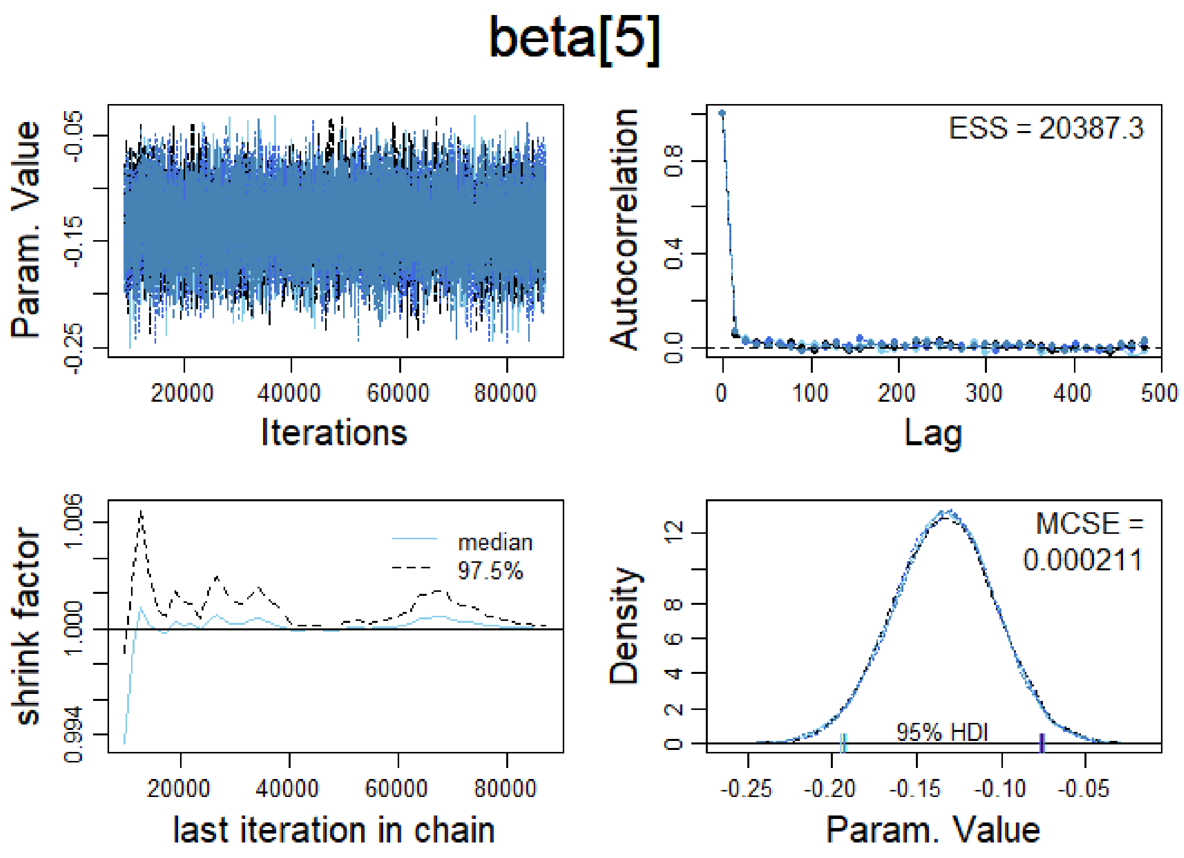


Param. Value
Iterations
Autocorrelation
ESS = 9181.4
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE =
0.000472
95% HDI
Param. Value

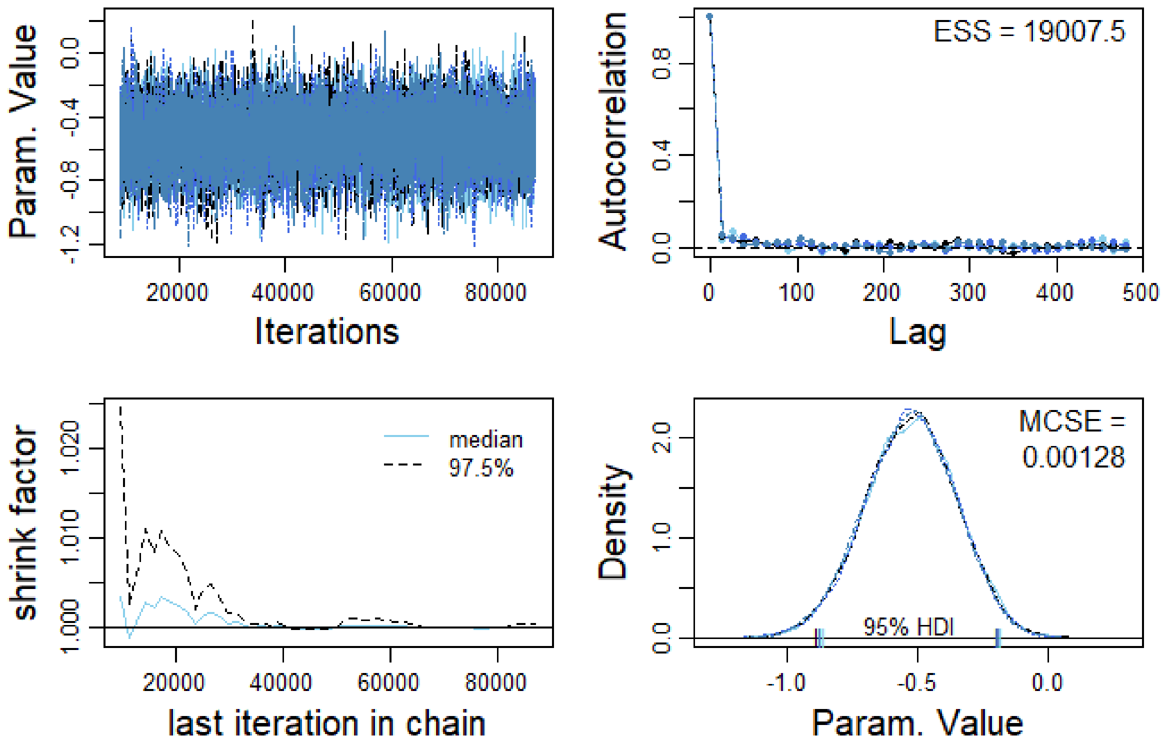

beta[2]
Param. Value
Iterations
Autocorrelation
ESS = 4382.1
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE =
0.00083
95% HDI
Param. Value


beta[3]
Param. Value
Iterations
Autocorrelation
ESS = 24276.5
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE =
8.51e-05
95% HDI
Param. Value


beta[4]
Param. Value
Iterations
Autocorrelation
ESS = 22380.7
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE =
2e-04
95% HDI
Param. Value

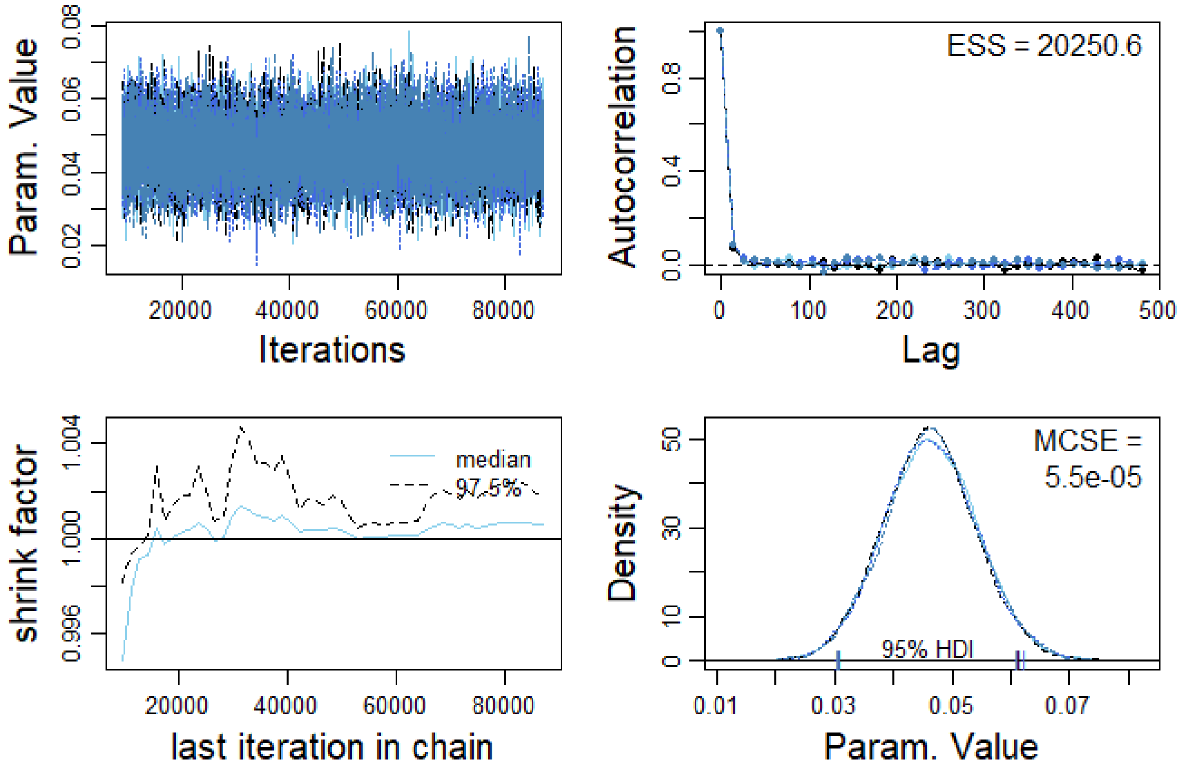

beta[5]
Param. Value
Iterations
Autocorrelation
ESS = 20387.3
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE =
0.000211
95% HDI
Param. Value

beta[6]

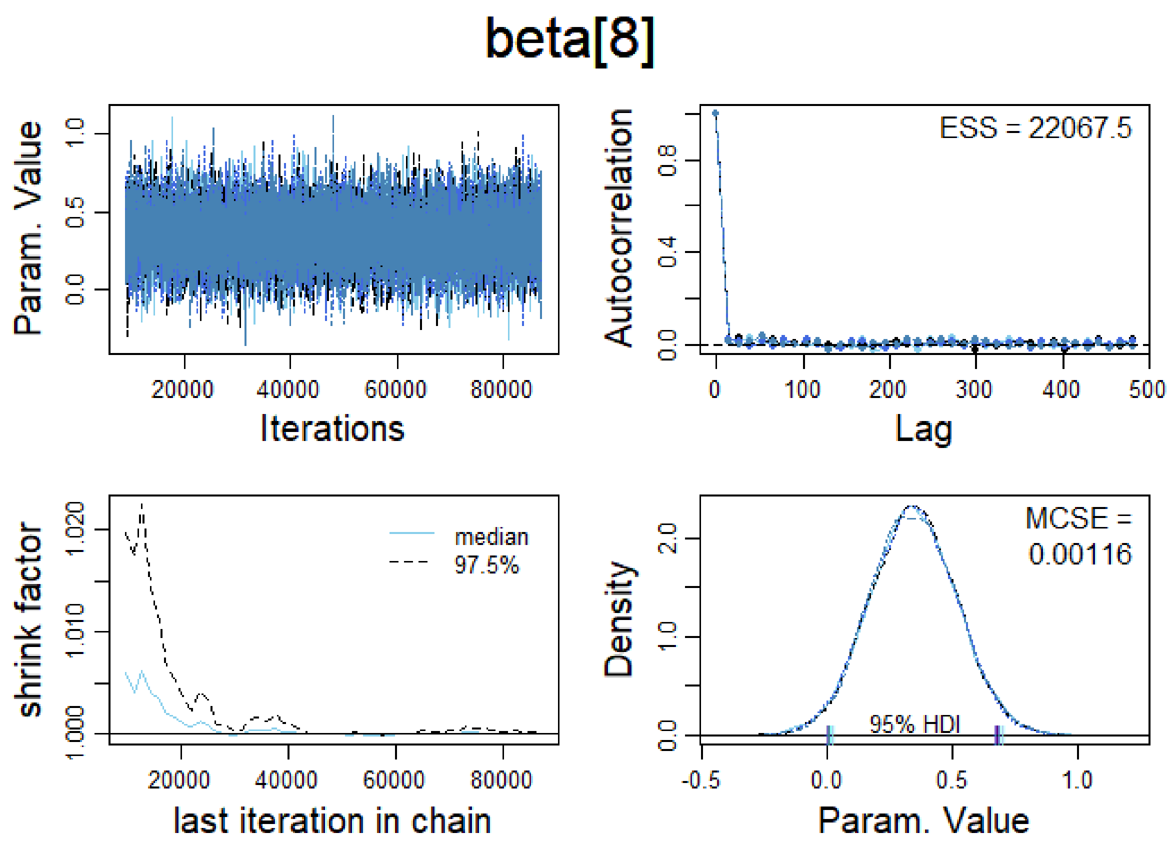


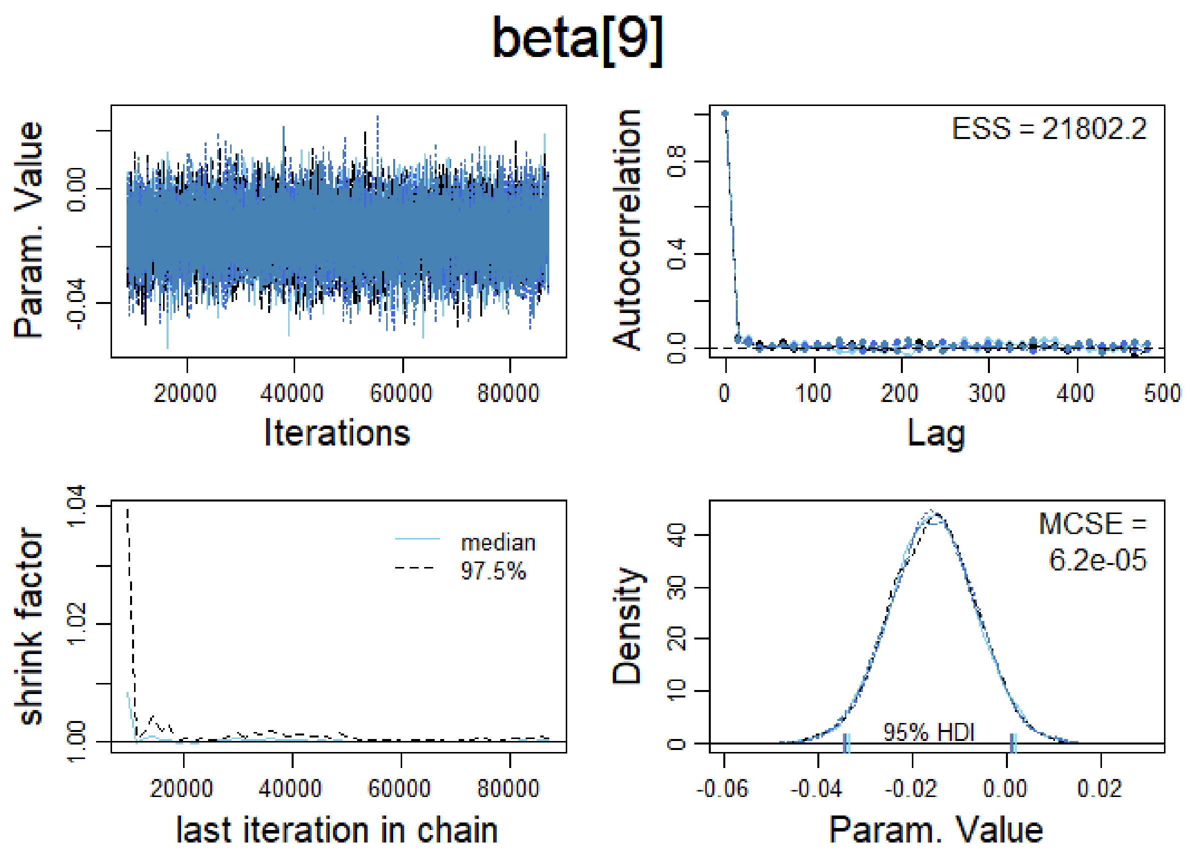
Param. Value
Iterations
Autocorrelation
ESS = 19007.5
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE =
0.00128
95% HDI
Param. Value

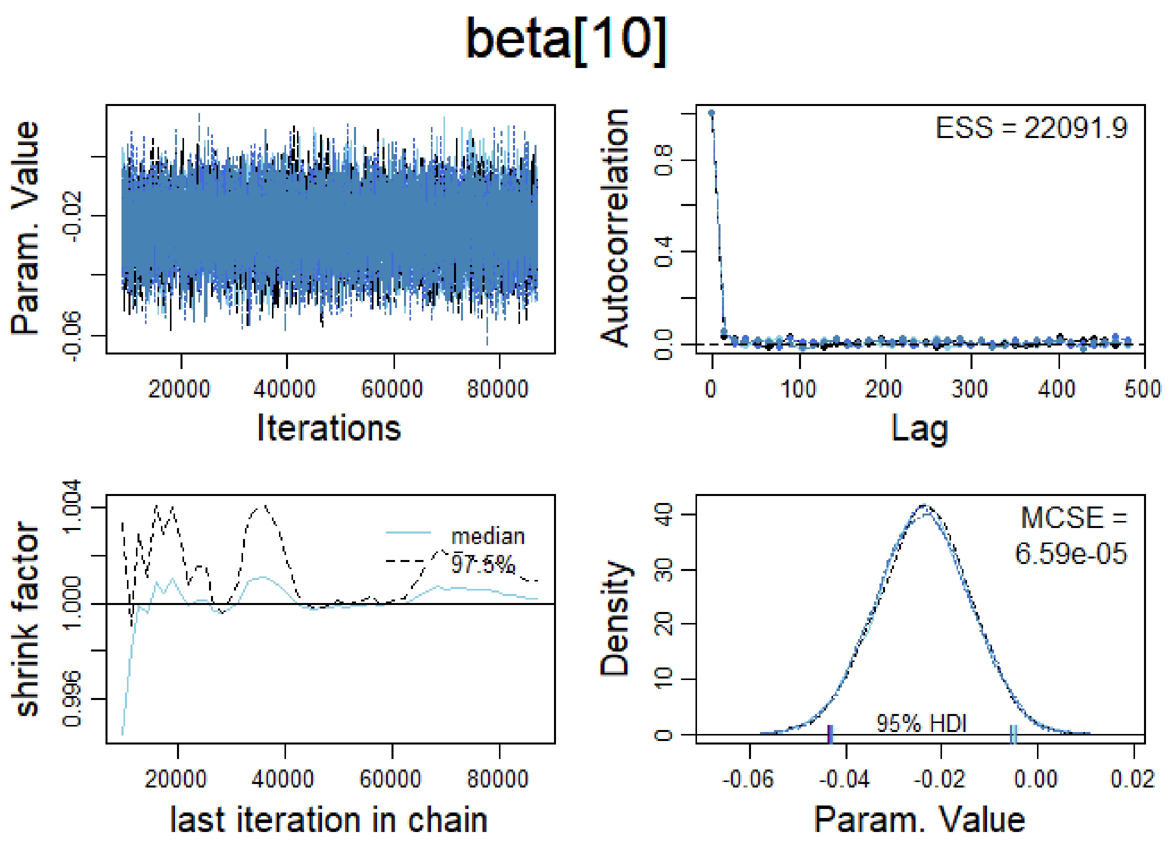
beta[7]

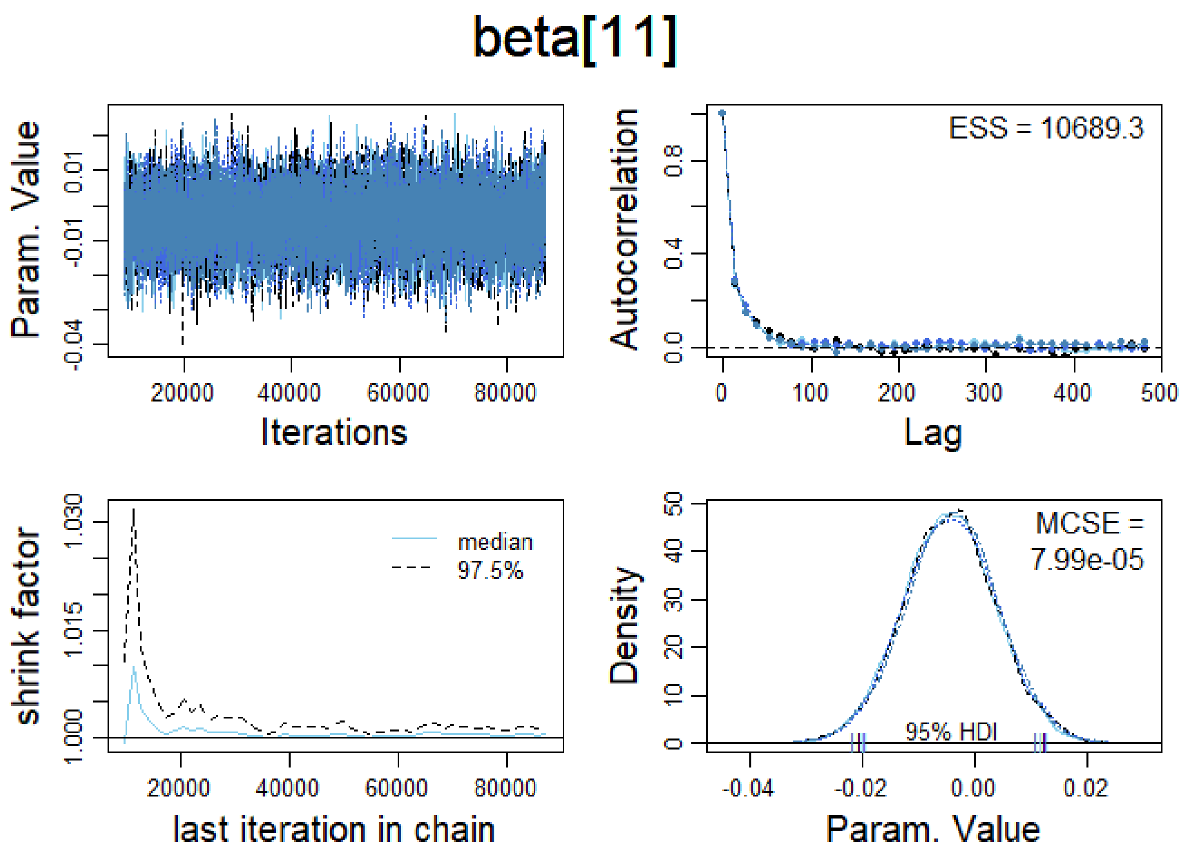


Param. Value
Iterations
Autocorrelation
ESS = 20250.6
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE =
5.5e-05
95% HDI
Param. Value

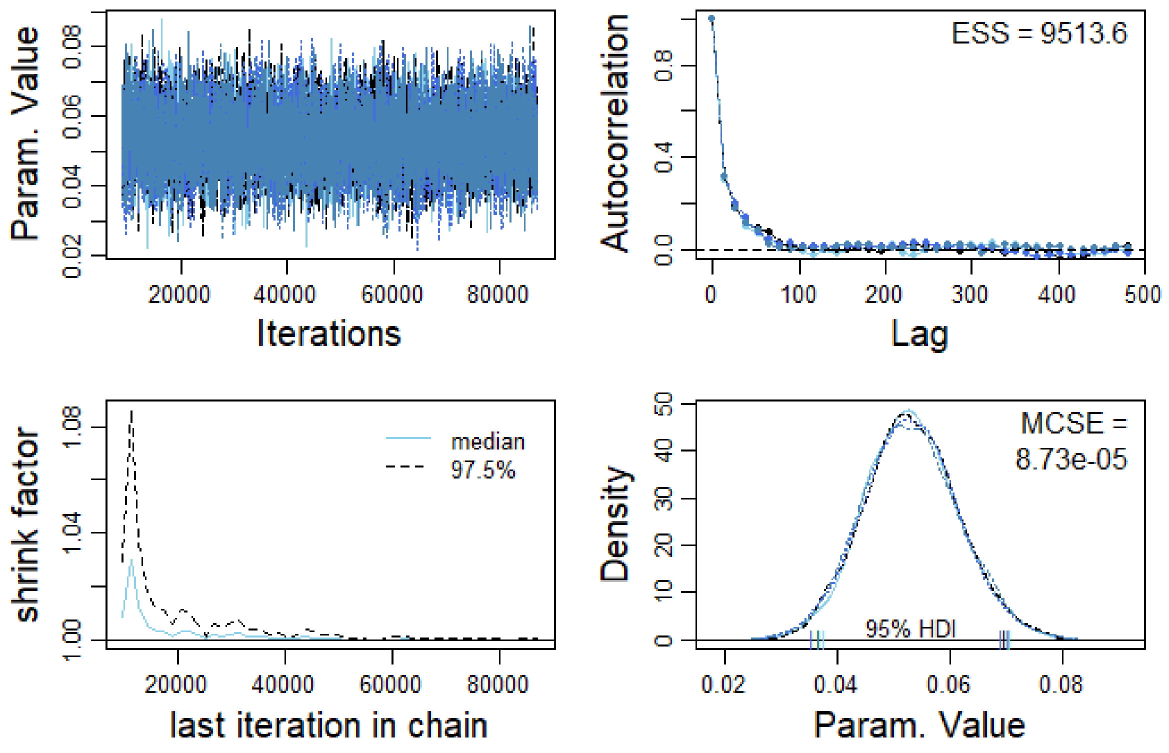

beta[8]
Param. Value
Iterations
Autocorrelation
ESS = 22067.5
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE = 0.00116
95% HDI
Param. Value


beta[9]
Param. Value
Iterations
Autocorrelation
ESS = 21802.2
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE = 6.2e-05
95% HDI
Param. Value

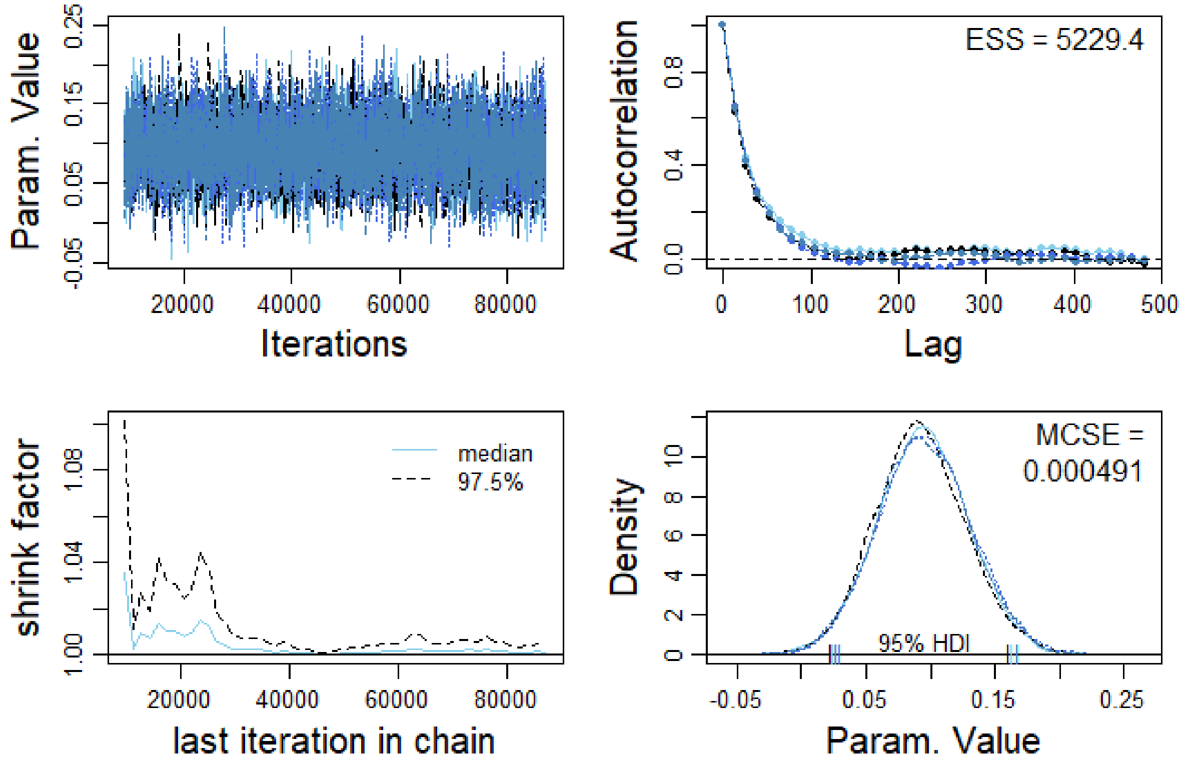

beta[10]
Param. Value
Iterations
Autocorrelation
ESS = 22091.9
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE = 6.59e-05
95% HDI
Param. Value


beta[11]
Param. Value
Iterations
Autocorrelation
ESS = 10689.3
Lag
shrink factor
median
97.5%
last iteration in chain
Density
MCSE = 7.99e-05
95% HDI
Param. Value

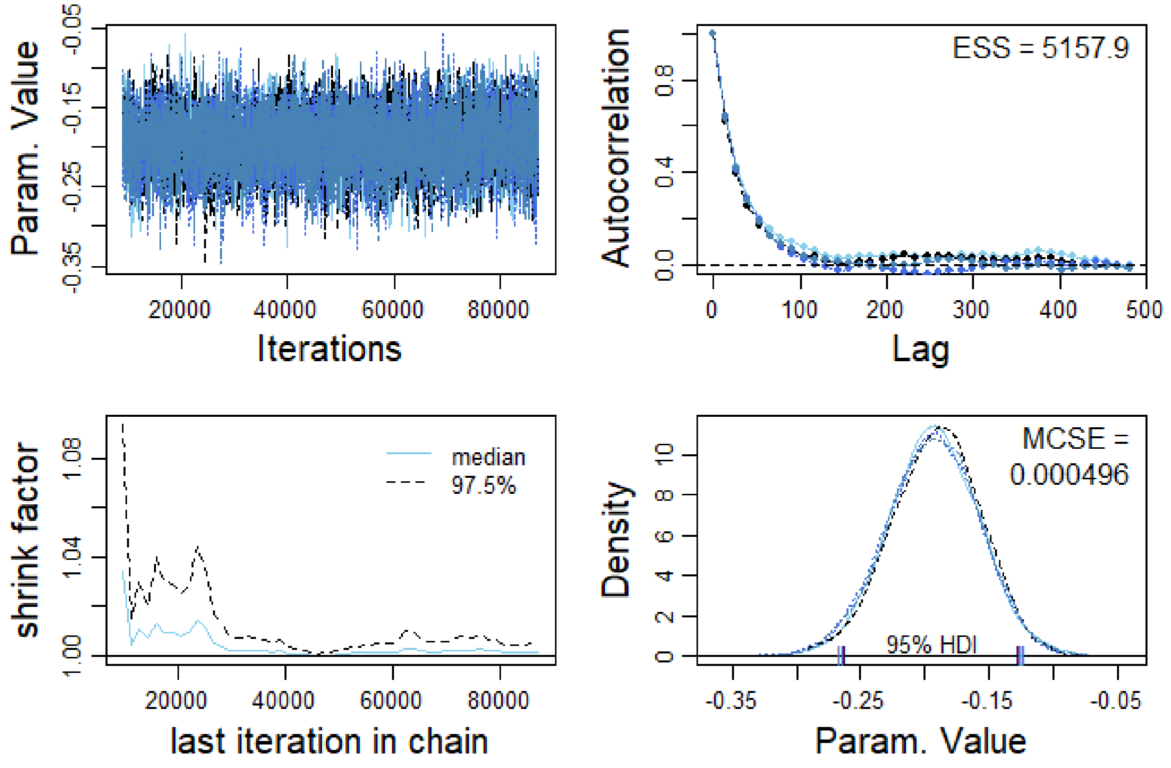
beta[12]



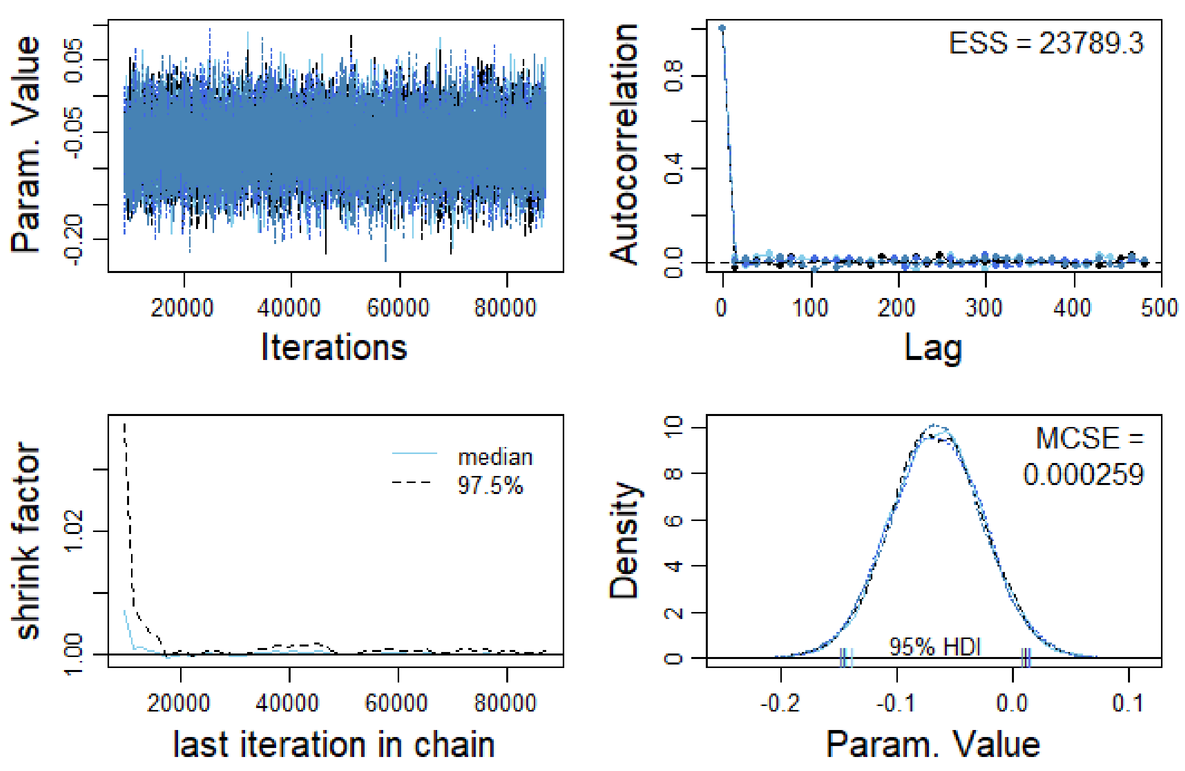
beta[13]



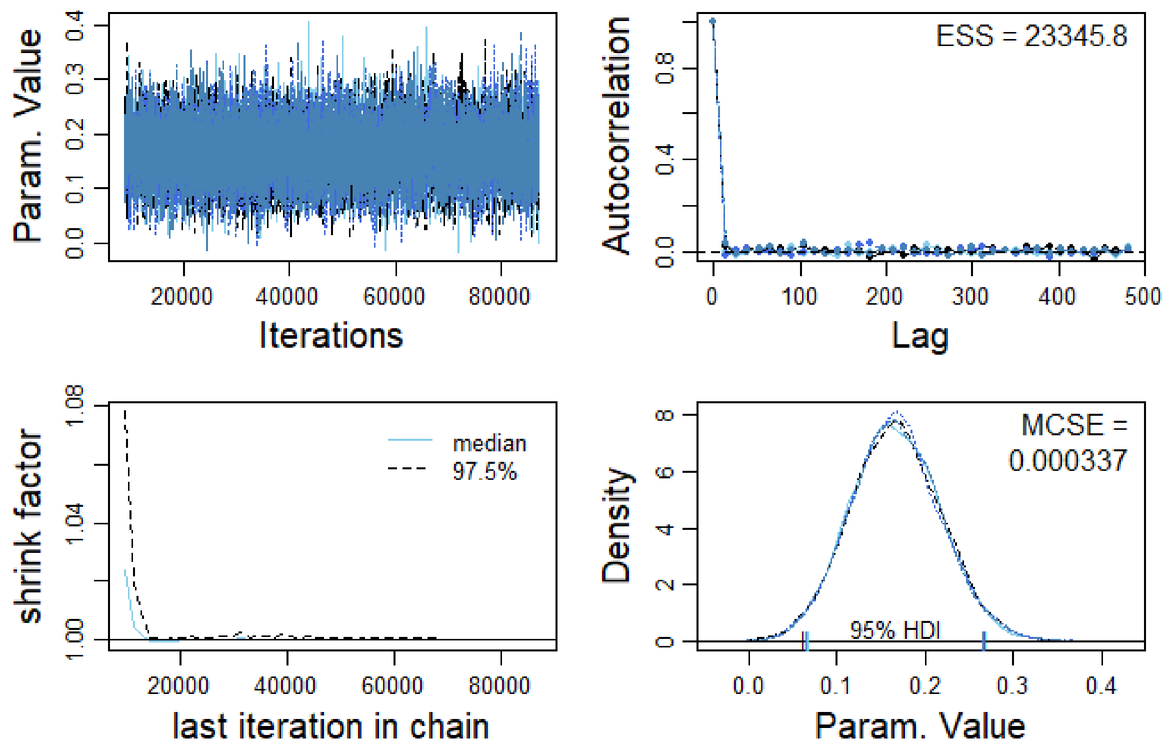
beta[14]



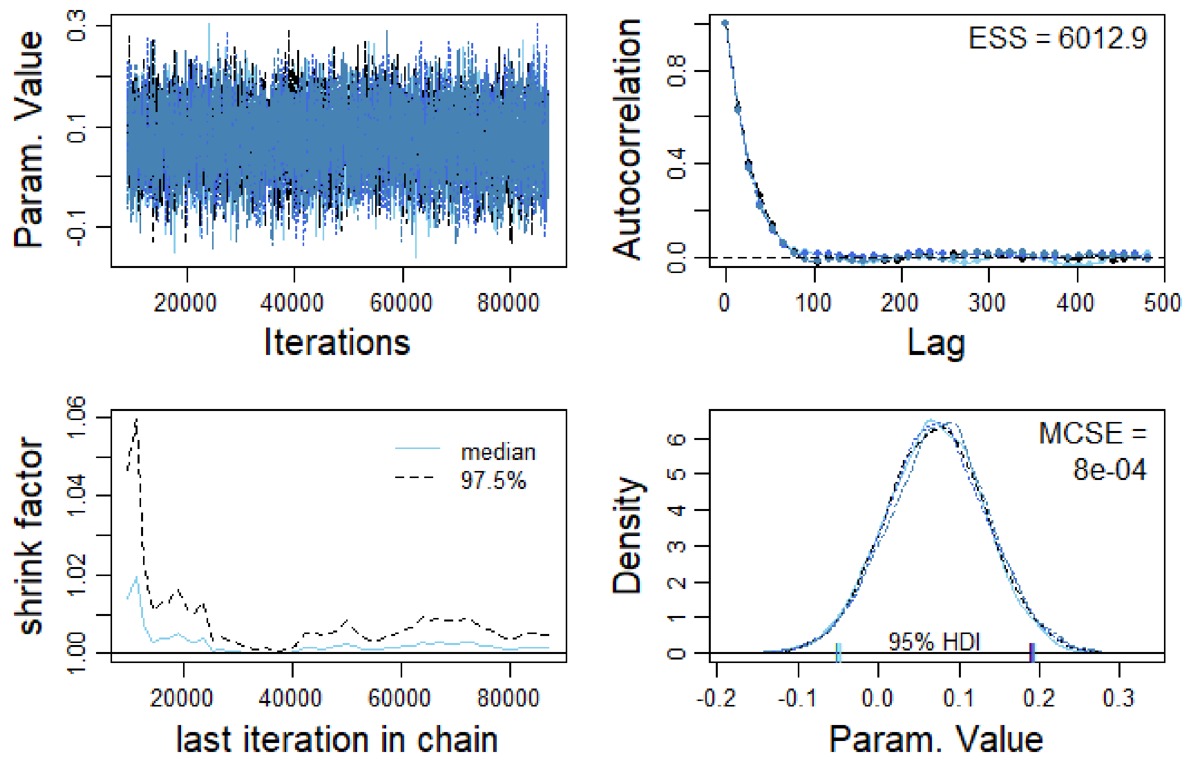
beta[15]

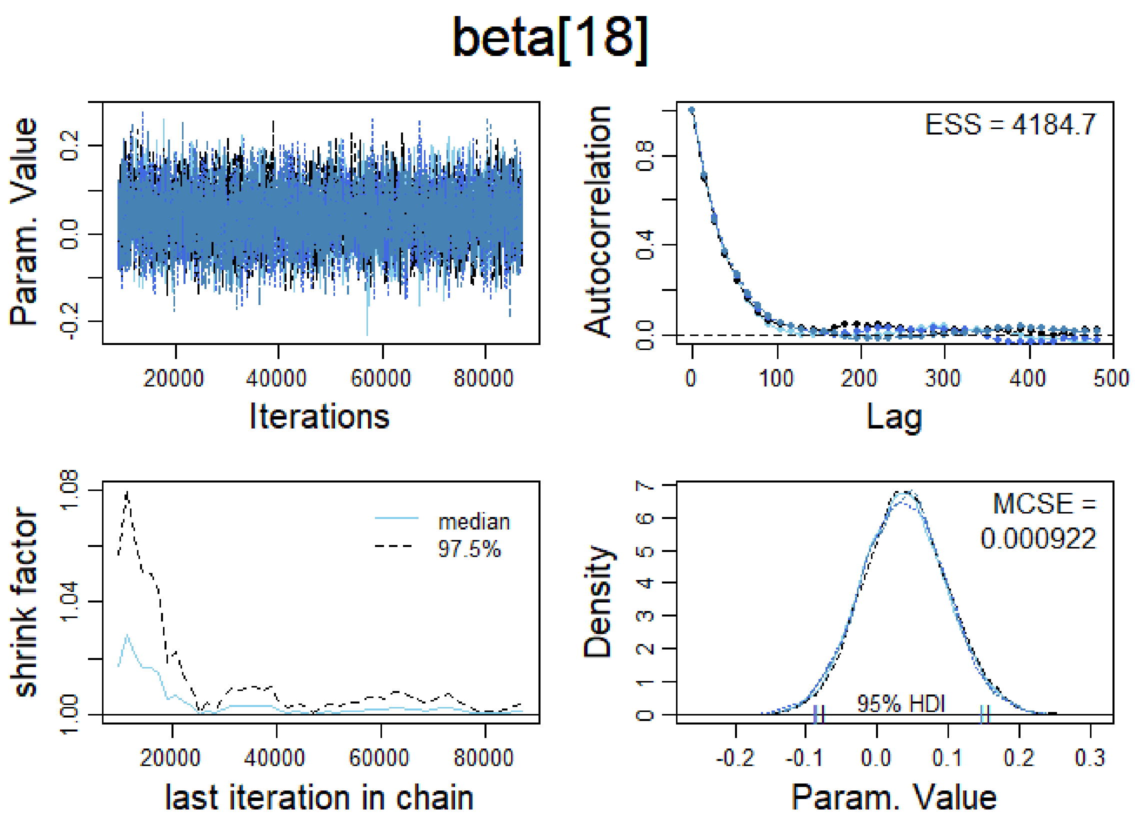


beta[16]



beta[17]





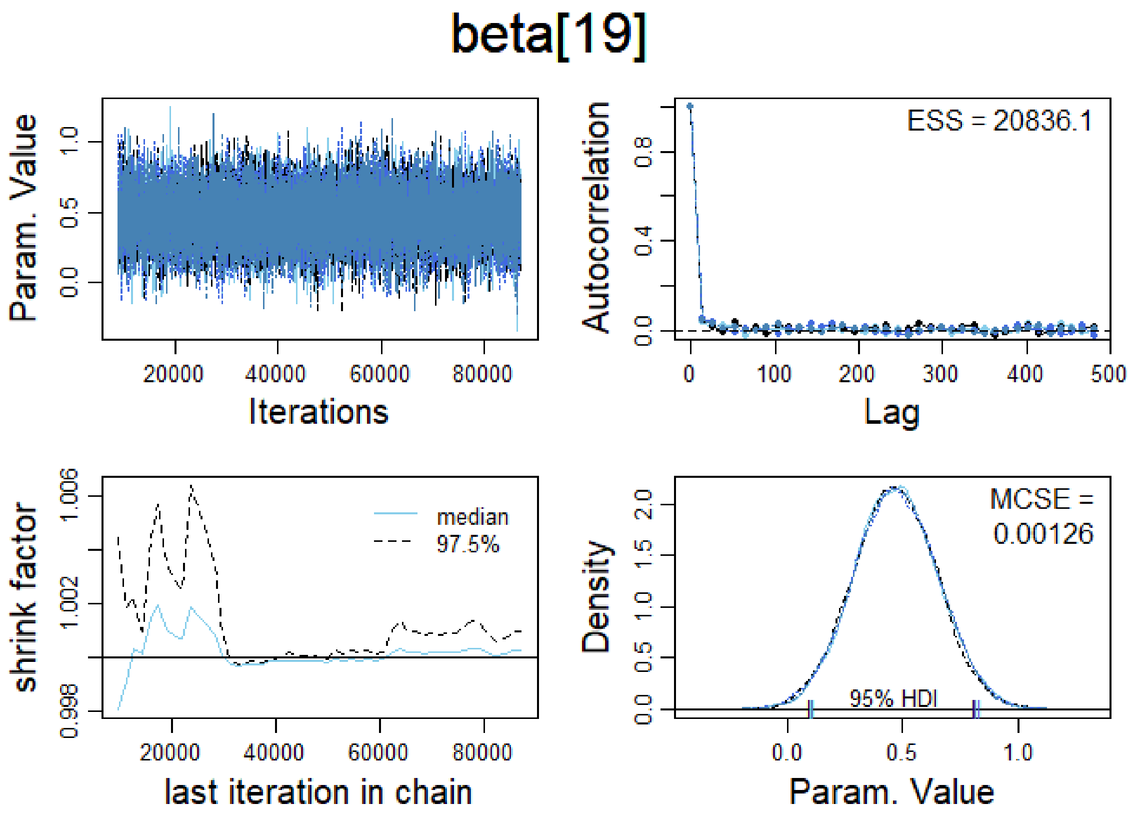
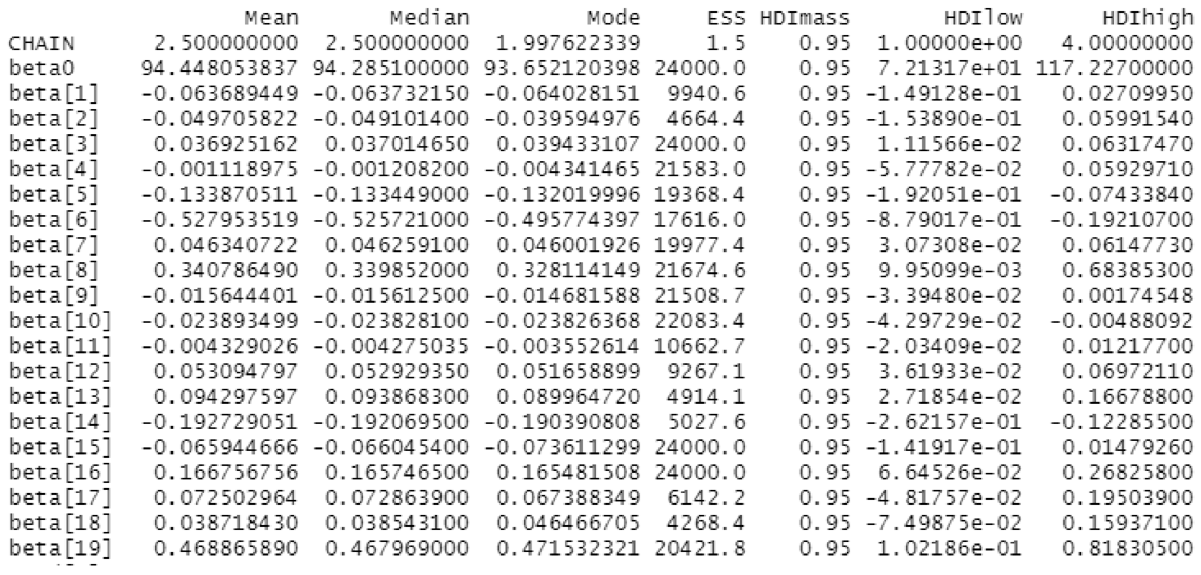


Figure-7 MCMC Diagnostics of all the beta coefficients – Fourth Run

- As we could see in above figure, the MCMC diagnostics of all beta coefficients seem to fulfil all the necessary criteria to proceed with Bayesian estimates.
- In all the diagnostics, all the 4 chains in the trace plot cycle towards the mean value. Hence, we can conclude it represent the sample and target towards the posterior distribution.
- Similarly, the density plots show that, at the burn in period, all the chains are overlapping perfectly showing a good representativeness. However, there are very minute changes in 95% HDI intervals of each chains.
- All the shrink factors are well below 1.2, and hence can be concluded that the chains have converged enough, to proceed with further analysis.
- The autocorrelation is low for all the beta coefficients and hence we can get an independent information perfectly thereby getting high accuracy Bayesian estimates. However, the autocorrelation is still a bit high for beta1, beta2, beta11, beta18 which might be due to imputation of NA values. It is negligible.
- Hence, we are good to go for further analysis with this Bayesian estimates.

Now let us look at the summary information of Bayesian estimates of all the beta coefficients,



```
                 Mean       Median         Mode     ESS HDImass       HDIlow      HDIhigh
CHAIN     2.500000000  2.500000000  1.997622339     1.5    0.95  1.00000e+00   4.00000000
beta0    94.448053837 94.285100000 93.652120398 24000.0    0.95  7.21317e+01 117.22700000
beta[1]  -0.063689449 -0.063732150 -0.064028151  9940.6    0.95 -1.49128e-01   0.02709950
beta[2]  -0.049705822 -0.049101400 -0.039594976  4664.4    0.95 -1.53890e-01   0.05991540
beta[3]   0.036925162  0.037014650  0.039433107 24000.0    0.95  1.11566e-02   0.06317470
beta[4]  -0.001118975 -0.001208200 -0.004341465 21583.0    0.95 -5.77782e-02   0.05929710
beta[5]  -0.133870511 -0.133449000 -0.132019996 19368.4    0.95 -1.92051e-01  -0.07433840
beta[6]  -0.527953519 -0.525721000 -0.495774397 17616.0    0.95 -8.79017e-01  -0.19210700
beta[7]   0.046340722  0.046259100  0.046001926 19977.4    0.95  3.07308e-02   0.06147730
beta[8]   0.340786490  0.339852000  0.328114149 21674.6    0.95  9.95099e-03   0.68385300
beta[9]  -0.015644401 -0.015612500 -0.014681588 21508.7    0.95 -3.39480e-02   0.00174548
beta[10] -0.023893499 -0.023828100 -0.023826368 22083.4    0.95 -4.29729e-02  -0.00488092
beta[11] -0.004329026 -0.004275035 -0.003552614 10662.7    0.95 -2.03409e-02   0.01217700
beta[12]  0.053094797  0.052929350  0.051658899  9267.1    0.95  3.61933e-02   0.06972110
beta[13]  0.094297597  0.093868300  0.089964720  4914.1    0.95  2.71854e-02   0.16678800
beta[14] -0.192729051 -0.192069500 -0.190390808  5027.6    0.95 -2.62157e-01  -0.12285500
beta[15] -0.065944666 -0.066045400 -0.073611299 24000.0    0.95 -1.41917e-01   0.01479260
beta[16]  0.166756756  0.165746500  0.165481508 24000.0    0.95  6.64526e-02   0.26825800
beta[17]  0.072502964  0.072863900  0.067388349  6142.2    0.95 -4.81757e-02   0.19503900
beta[18]  0.038718430  0.038543100  0.046466705  4268.4    0.95 -7.49875e-02   0.15937100
beta[19]  0.468865890  0.467969000  0.471532321 20421.8    0.95  1.02186e-01   0.81830500
```

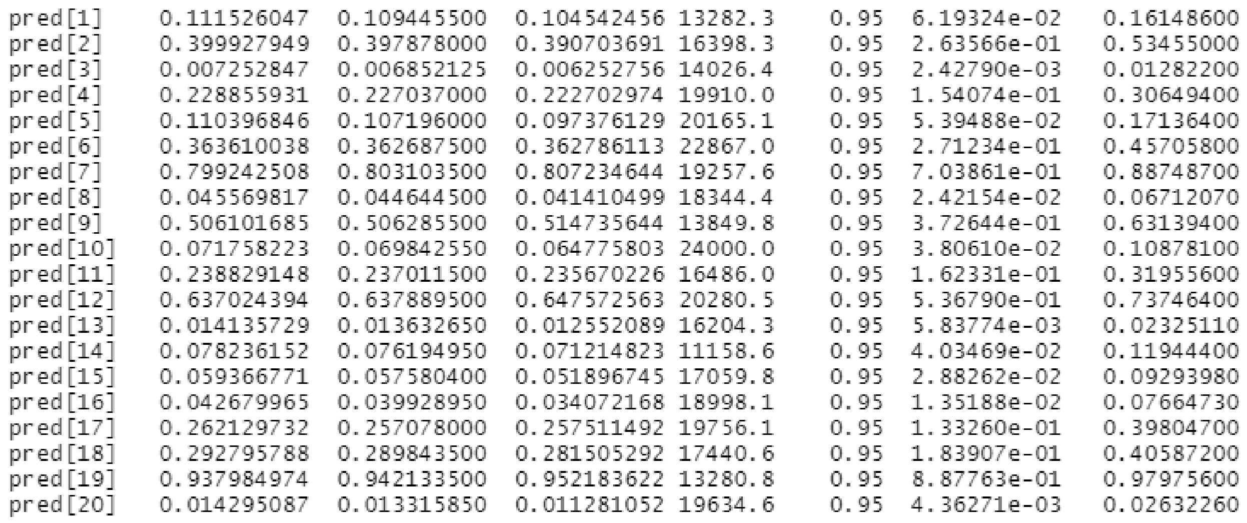
Figure-8.1 Summary Statistics

Let us interpret some of the beta coefficients to get some idea from the summaryMCMC function's output,

- The mean, median and mode of beta0 or the intercept is 94.45, 94.29 and 93.65, respectively. The median shows that 50% of values are below 94.29 and 50% of values are above 94.29. The Mode shows the highest number of values in posterior distribution of beta0. HDI interval shows that, the probability that the intercept term will be in the range [72.13,117.23] is 95%.
  These range contains the most credible values of beta0 term.

- Similarly, the mean, median and mode of beta7 or Coefficient of the variable “WindGustSpeed” is 0.05,0.05 and 0.05, respectively. Here the probability that the coefficient of “WindGustSpeed” variable will be between [0.031,0.061] is 95%.
- The coefficient values are known, and it means, for every unit increase in ‘WindGustSpeed’ variable, the probability of raining will increase by 5%.
- Similarly, for every unit increase in ‘Temperature3pm’, the probability of raining the next day will increase by 4.6%.

Based on the above mode of beta coefficients, summary statistics present the output of each observation of test data file. Few of the predictions are as below,



```
pred[1]    0.111526047  0.109445500  0.104542456 13282.3    0.95  6.19324e-02   0.16148600
pred[2]    0.399927949  0.397878000  0.390703691 16398.3    0.95  2.63566e-01   0.53455000
pred[3]    0.007252847  0.006852125  0.006252756 14026.4    0.95  2.42790e-03   0.01282200
pred[4]    0.228855931  0.227037000  0.222702974 19910.0    0.95  1.54074e-01   0.30649400
pred[5]    0.110396846  0.107196000  0.097376129 20165.1    0.95  5.39488e-02   0.17136400
pred[6]    0.363610038  0.362687500  0.362786113 22867.0    0.95  2.71234e-01   0.45705800
pred[7]    0.799242508  0.803103500  0.807234644 19257.6    0.95  7.03861e-01   0.88748700
pred[8]    0.045569817  0.044644500  0.041410499 18344.4    0.95  2.42154e-02   0.06712070
pred[9]    0.506101685  0.506285500  0.514735644 13849.8    0.95  3.72644e-01   0.63139400
pred[10]   0.071758223  0.069842550  0.064775803 24000.0    0.95  3.80610e-02   0.10878100
pred[11]   0.238829148  0.237011500  0.235670226 16486.0    0.95  1.62331e-01   0.31955600
pred[12]   0.637024394  0.637889500  0.647572563 20280.5    0.95  5.36790e-01   0.73746400
pred[13]   0.014135729  0.013632650  0.012552089 16204.3    0.95  5.83774e-03   0.02325110
pred[14]   0.078236152  0.076194950  0.071214823 11158.6    0.95  4.03469e-02   0.11944400
pred[15]   0.059366771  0.057580400  0.051896745 17059.8    0.95  2.88262e-02   0.09293980
pred[16]   0.042679965  0.039928950  0.034072168 18998.1    0.95  1.35188e-02   0.07664730
pred[17]   0.262129732  0.257078000  0.257511492 19756.1    0.95  1.33260e-01   0.39804700
pred[18]   0.292795788  0.289843500  0.281505292 17440.6    0.95  1.83907e-01   0.40587200
pred[19]   0.937984974  0.942133500  0.952183622 13280.8    0.95  8.87763e-01   0.97975600
pred[20]   0.014295087  0.013315850  0.011281052 19634.6    0.95  4.36271e-03   0.02632260
```

Figure 8.2 – Predictions

- Figure 8 shows the different mean values of the predictions based on the regression parameters.
- Pred[2] data suggests that, there are 39.07% of probability that it will rain tomorrow. Similarly, the 95% Highest density interval of probability that it will rain tomorrow ranges from [26%,53%]
- Similarly, Pred[15] data suggests that the probability than it will rain the next day is 5%. Its HDI ranges from [2%,9%]
- Hence after the predictions, probability less than 50 % are grouped as ‘0’ (will not rain) and that of greater than 50% as ‘1’ (will rain).

Now let us visualize the posterior distributions along with the High-Density intervals,

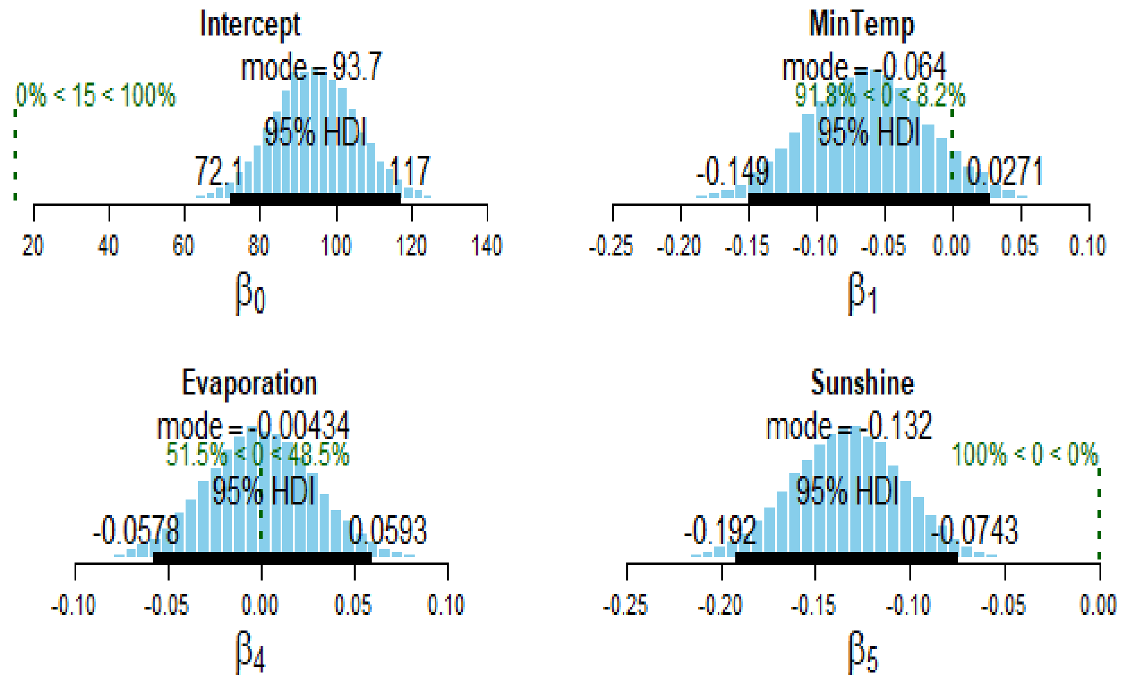
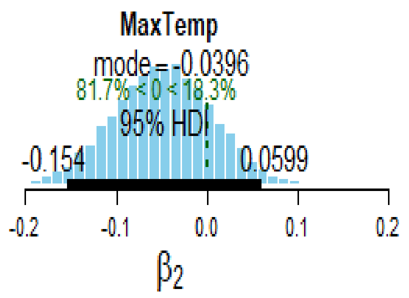
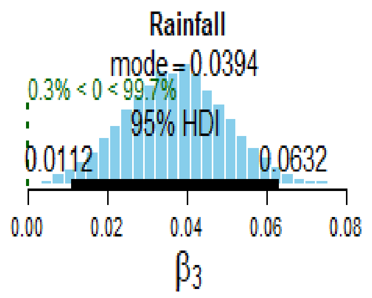
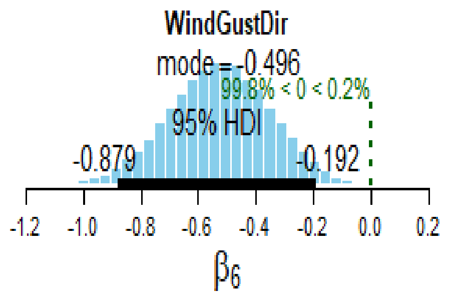


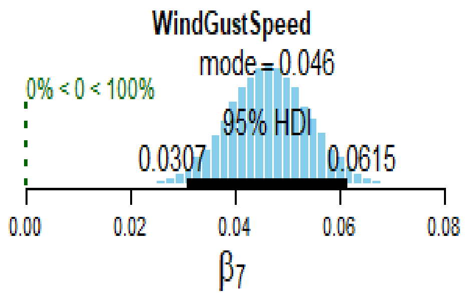
Figure-9 Posterior Distributions - Part 1

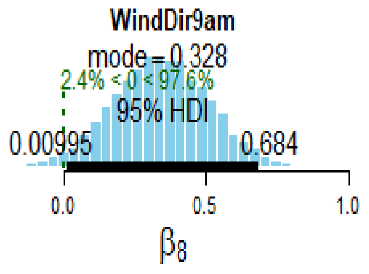
- The intercept term's posterior distribution shows that the probability of intercept being less than 15 is 0 % while it is greater than 15 is 100%. The HDI interval does not capture the Null Hypothesis value of 0 and hence this parameter is significant.
- The Min Temp coefficient's posterior distribution shows that the probability of Minimum temperature's coefficient less than 0 is 92% while the probability of it greater than 0 is 8.2%. Even though the interval captures the null hypothesis value of 0, the upper interval is very close to 0 and hence this parameter is assumed statistically significant.
- The Evaporation beta coefficient's posterior distribution plot shows that the probability that this coefficient will be less than 0 is 51.5% while it is greater than 0 is 48.5%. Here the HDI captures 0 and this is statistically insignificant. Hence this parameter can be left out if at all we do a Bayesian analysis using logistic regression of Rainfall Prediction.
- Similarly, Sunshine coefficient's posterior distribution depicts that the probability that the coefficient will be less than 0 is 100%. Here, the HDI does not capture the null hypothesis value of 0. Hence this will be statistically significant.

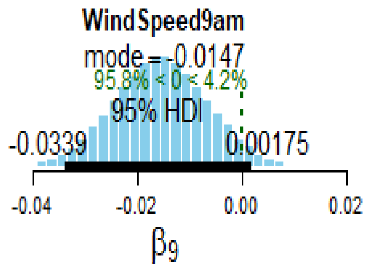

MaxTemp
mode = -0.0396
81.7% < 0 < 18.3%
95% HDI
-0.154
0.0599
-0.2
-0.1
0.0
0.1
0.2
β_2

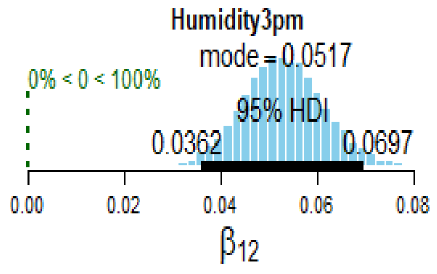

Rainfall
mode = 0.0394
0.3% < 0 < 99.7%
95% HDI
0.0112
0.0632
0.00
0.02
0.04
0.06
0.08
β_3

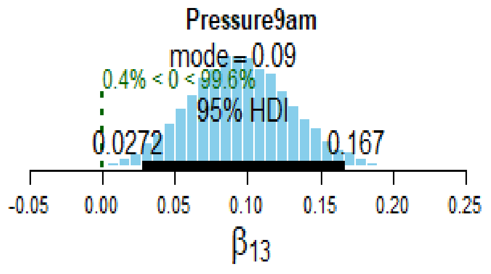

WindGustDir
mode = -0.496
99.8% < 0 < 0.2%
95% HDI
-0.879
-0.192
-1.2
-1.0
-0.8
-0.6
-0.4
-0.2
0.0
0.2
β_6

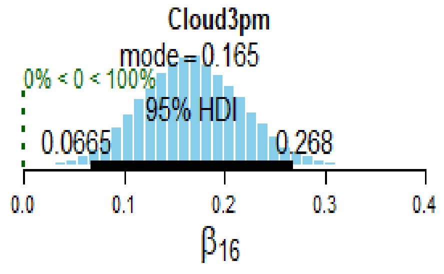

WindGustSpeed
mode = 0.046
0% < 0 < 100%
95% HDI
0.0307
0.0615
0.00
0.02
0.04
0.06
0.08
β_7

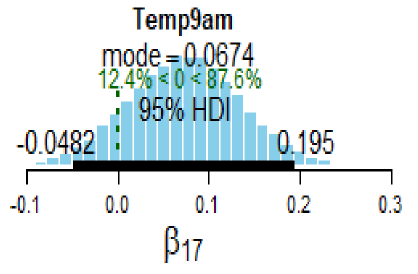

WindDir9am
mode = 0.328
2.4% < 0 < 97.6%
95% HDI
0.00995
0.684
0.0
0.5
1.0
β_8


WindSpeed9am
mode = -0.0147
95.8% < 0 < 4.2%
95% HDI
-0.0339
0.00175
-0.04
-0.02
0.00
0.02
β_9


Humidity3pm
mode = 0.0517
0% < 0 < 100%
95% HDI
0.0362
0.0697
0.00
0.02
0.04
0.06
0.08
β_{12}


Pressure9am
mode = 0.09
0.4% < 0 < 99.6%
95% HDI
0.0272
0.167
-0.05
0.00
0.05
0.10
0.15
0.20
0.25
β_{13}


Cloud3pm
mode = 0.165
0% < 0 < 100%
95% HDI
0.0665
0.268
0.0
0.1
0.2
0.3
0.4
β_{16}


Temp9am
mode = 0.0674
12.4% < 0 < 87.6%
95% HDI
-0.0482
0.195
-0.1
0.0
0.1
0.2
0.3
β_{17}

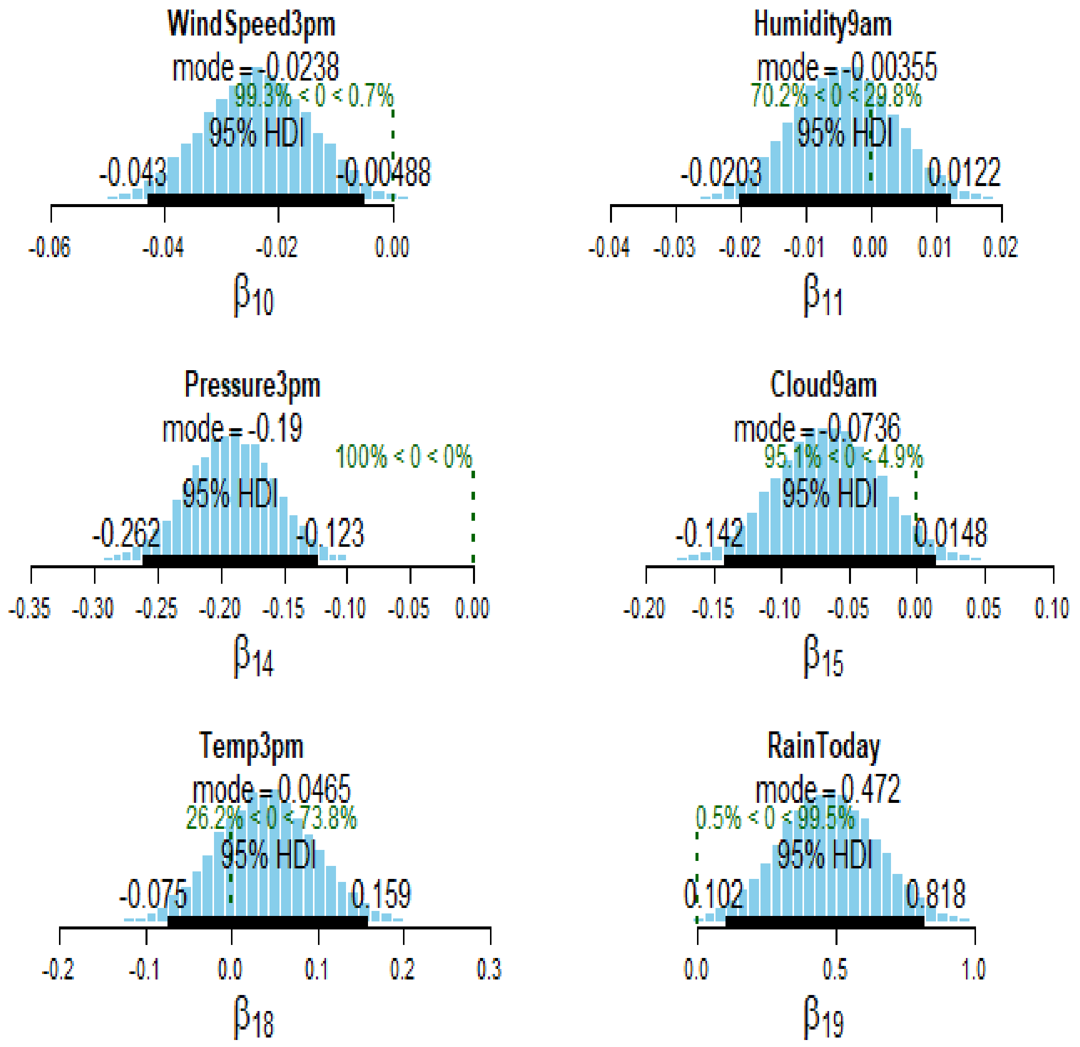


Figure-10 Posterior Distributions of Beta Coefficients - Part 2

- As we interpreted the first set of coefficients, we can do the same as well for rest of the coefficients.
- Based on the posterior distribution plots, we can conclude which parameters are statistically significant.
- MaxTemp, Rainfall, WindDir9am, WindGustSpeed, Humidity3pm, Pressure9am, Cloud3pm, WindSpeed3pm, Pressure3pm, Cloud9am and Rain Today are statistically significant since the HDI of all the parameters does not or partially captures the null hypothesis value.

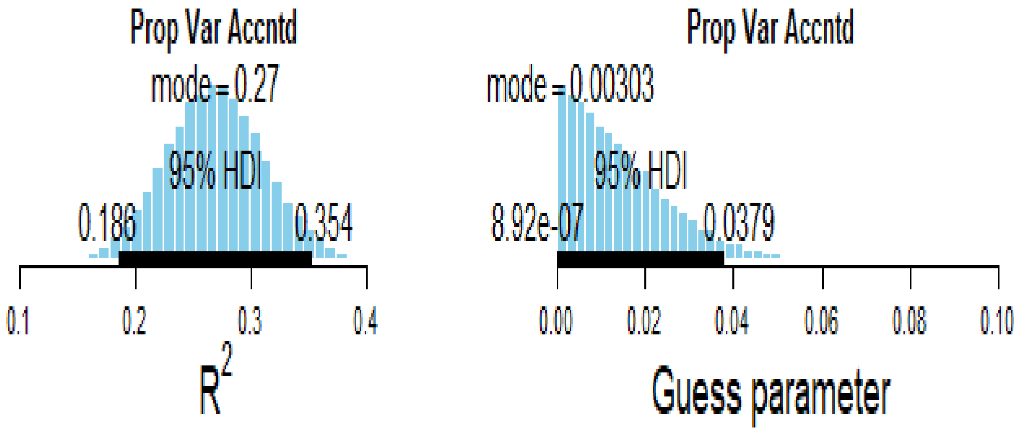


Figure-11 Posterior distributions of R^2 and Guess Parameter

- Figure 11 shows the posterior distribution of R^2 and the Guess parameter.
- R^2 does not do well in Bayesian statistics and is expected to be low for categorical response variables.
- The Guess parameter's posterior distribution shows that 0.3% of total values fall into outliers.

<u>Efficiency of different Model Runs</u>:

- Runtimes are reported in the below table which helps to define which model was accurate and efficient.

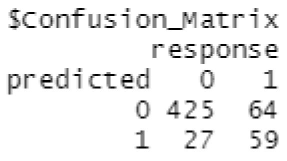
Runs	Number of Saved Steps	Adaptation Steps	Burn-In Steps	Thinning Steps	Number of Chains	Standardization	Run-Time (in hrs)	Reported
1	1000	1000	3000	4	3	No	2	Yes
2	1000	1000	3000	4	3	Yes	2.5	Yes
3	1000	1000	6000	9	4	Yes	13	Yes
4	5000	2000	7000	15	5	Yes	25.25	No
5	6000	2000	7000	13	4	Yes	22.75	Yes

Figure-12 Different Models' input and its efficiency based on RunTimes

- Figure-12 shows the different input values to JAGS model and its corresponding runtime.
- Run number 4 is not reported since it gave the similar output to the Run number 5 and was less efficient since it took more time.
- Hence Run number 5 was used as the final model to proceed with the Bayesian estimates.

Confusion Matrix:

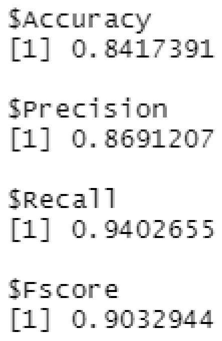
- Confusion Matrix is basically a 2x2 matrix of Predicted variables and Actual Variables for a binomial categorical response variable.
- It determines the efficiency of model showing how many values are correctly predicted by comparing the predicted and actual data.



```
$Confusion_Matrix
          response
predicted   0   1
        0 425  64
        1  27  59
```

Figure-13 Confusion Matrix

- Figure-11 shows the confusion matrix obtained from the results of our model.
- The Model has predicted that there will be no rain for 425 values out of 489 values and that it will rain, for 59 values out of 86 values.
- So, to check the efficiency, we go for success metrics and we get the following,



```
$Accuracy
[1] 0.8417391

$Precision
[1] 0.8691207

$Recall
[1] 0.9402655

$Fscore
[1] 0.9032944
```

Figure-14 Success Metrics

- Figure-12 shows that, there has been an accuracy of 84.17% in the Bayesian estimates provided by our logistic regression model.
- The Values are 86.91% precise and has a Recall score of 94.02% and F Score of 90.32%.
- The above values are pretty good for a good Bayesian model.

- **RESULTS**

The Robust Logistic Regression was performed for the rainfall prediction in the city of Melbourne with the help of non-informative prior JAGS Model. The guess parameter was included due to the outliers underlying in the variables and finally the mode of guess shows how much of total population went into outliers. So, we found out that there were few variables which had a bigger impact on rainfall prediction such as WindSpeed, Pressure, Temperature measured at 3pm and the evaporation rate. Further with the help of posterior distributions and interpretation it was more conclusive to see that the evaporation rate, Pressure and Temperature at 3 pm where higher it was more likely that it would not rain the next day. However, if the WindGustSpeed and Humidity measured at 3pm were higher it had higher probability of rainfall. So, finally to be efficient with the outcomes predicted multiple interpretations with different set of inputs where carried out and accuracy was estimated with help of the confusion matrix. Our model achieved an accuracy of 84% and precision of 86% making it more reliable to predict whether the rainfall would occur or not based on previous day's meteorological observations.

APPENDIX A

R-Code:

```
library(ggplot2)

library(ggpubr)

library(ks)

library(rjags)

library(runjags)

library(dplyr)

library(tidyr)

library(Hmisc)

source("F:\\Subbu\\RMIT\\sem 2\\Applied Bayesian Statistics\\Final Project\\DBDA2E-utilities.R")

setwd("F:\\Subbu\\RMIT\\sem 2\\Applied Bayesian Statistics\\Final Project\\")



#===============PRELIMINARY FUNCTIONS FOR POSTERIOR
INFERENCES====================


smryMCMC_HD = function(  codaSamples , compVal = NULL,  saveName=NULL) {

  summaryInfo = NULL

  mcmcMat = as.matrix(codaSamples,chains=TRUE)

  paramName = colnames(mcmcMat)

  for ( pName in paramName ) {

    if (pName %in% colnames(compVal)){

      if (!is.na(compVal[pName])) {

        summaryInfo = rbind( summaryInfo , summarizePost( paramSampleVec = mcmcMat[,pName] ,

                                                          compVal = as.numeric(compVal[pName]) ))

      }
      else {
        summaryInfo = rbind( summaryInfo , summarizePost( paramSampleVec = mcmcMat[,pName] ) )
      }
    } else {
      summaryInfo = rbind( summaryInfo , summarizePost( paramSampleVec = mcmcMat[,pName] ) )
    }
  }
  rownames(summaryInfo) = paramName

  # summaryInfo = rbind( summaryInfo ,
  #                      "tau" = summarizePost( mcmcMat[,"tau"] ) )
  if ( !is.null(saveName) ) {
    write.csv( summaryInfo , file=paste(saveName,"SummaryInfo.csv",sep="") )
  }
  return( summaryInfo )
}

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

plotMCMC_HD = function( codaSamples , data , xName="x" , yName="y", preds = FALSE ,
                        showCurve=FALSE ,  pairsPlot=FALSE , compVal = NULL,
                        saveName=NULL , saveType="jpg" ) {
   #-----------------------------------------------------------------------------
  y = data[,yName]
  x = as.matrix(data[,xName])
  mcmcMat = as.matrix(codaSamples,chains=TRUE)
  chainLength = NROW( mcmcMat )
  beta0 = mcmcMat[,"beta0"]
  beta  = mcmcMat[,grep("^beta$|^beta\\[",colnames(mcmcMat))]
```

```
if ( ncol(x)==1 ) { beta = matrix( beta , ncol=1 ) }
if (preds){
  pred = mcmcMat[,grep("^pred$|^pred\\[",colnames(mcmcMat))]
}
guess = mcmcMat[,"guess"]
#-----------------------------------------------------------------------------
# Compute R^2 for credible parameters:
YcorX = cor( y , x ) # correlation of y with each x predictor
Rsq = beta %*% matrix( YcorX , ncol=1 )
#-----------------------------------------------------------------------------
if ( pairsPlot ) {
  # Plot the parameters pairwise, to see correlations:
  openGraph()
  nPtToPlot = 1000
  plotIdx = floor(seq(1,chainLength,by=chainLength/nPtToPlot))
  panel.cor = function(x, y, digits=2, prefix="", cex.cor, ...) {
    usr = par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r = (cor(x, y))
    txt = format(c(r, 0.123456789), digits=digits)[1]
    txt = paste(prefix, txt, sep="")
    if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex=1.25 ) # was cex=cex.cor*r
  }
  pairs( cbind( beta0 , beta , tau )[plotIdx,] ,
         labels=c( "beta[0]" ,
                   paste0("beta[",1:ncol(beta),"]\n",xName) ,
                   expression(tau) ) ,
         lower.panel=panel.cor , col="skyblue" )
  if ( !is.null(saveName) ) {
    saveGraph( file=paste(saveName,"PostPairs",sep=""), type=saveType)
  }
}
#------------------------------------------------------------------------------
# Marginal histograms:

decideOpenGraph = function( panelCount , saveName , finished=FALSE ,
                            nRow=5 , nCol=4 ) {
  # If finishing a set:
  if ( finished==TRUE ) {
    if ( !is.null(saveName) ) {
      saveGraph( file=paste0(saveName,ceiling((panelCount-1)/(nRow*nCol))),
                 type=saveType)
    }
    panelCount = 1 # re-set panelCount
    return(panelCount)
  } else {
    # If this is first panel of a graph:
    if ( ( panelCount %% (nRow*nCol) ) == 1 ) {
      # If previous graph was open, save previous one:
      if ( panelCount>1 & !is.null(saveName) ) {
        saveGraph( file=paste0(saveName,(panelCount%/%(nRow*nCol))),
                   type=saveType)
      }
      # Open new graph
      openGraph(width=nCol*7.0/3,height=nRow*2.0)
      layout( matrix( 1:(nRow*nCol) , nrow=nRow, byrow=TRUE ) )
      par( mar=c(4,4,2.5,0.5) , mgp=c(2.5,0.7,0) )
    }
    # Increment and return panel count:
```

```
    panelCount = panelCount+1
    return(panelCount)
  }
}

# Original scale:
panelCount = 1
panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMarg") )
histInfo = plotPost( beta0 , cex.lab = 1.75 , showCurve=showCurve ,
                     xlab=bquote(beta[0]) , main="Intercept", compVal = as.numeric(compVal["beta0"] ))
for ( bIdx in 1:ncol(beta) ) {
  panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMarg") )
  if (!is.na(compVal[paste0("beta[",bIdx,"]")])){
    histInfo = plotPost( beta[,bIdx] , cex.lab = 1.75 , showCurve=showCurve ,
                         xlab=bquote(beta[.(bIdx)]) , main=xName[bIdx],
                         compVal = as.numeric(compVal[paste0("beta[",bIdx,"]")]))
  } else{
    histInfo = plotPost( beta[,bIdx] , cex.lab = 1.75 , showCurve=showCurve ,
                         xlab=bquote(beta[.(bIdx)]) , main=xName[bIdx])
  }
}
panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMarg") )
histInfo = plotPost( Rsq , cex.lab = 1.75 , showCurve=showCurve ,
                     xlab=bquote(R^2) , main=paste("Prop Var Accntd") , finished=FALSE )

panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMarg") )
histInfo = plotPost( guess , cex.lab = 1.75 , showCurve=showCurve ,
                     xlab="Guess parameter" , main=paste("Prop Var Accntd") , finished=TRUE )

panelCount = 1
```

```
  if ( pred){

    for ( pIdx in 1:ncol(pred) ) {
      panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMarg") )
      histInfo = plotPost( pred[,pIdx] , cex.lab = 1.75 , showCurve=showCurve ,
                           xlab=bquote(pred[.(pIdx)]) , main=paste0("Prediction ",pIdx) )
    }
  }
  panelCount = 1

  panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMargZ") )
  histInfo = plotPost( Rsq , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab=bquote(R^2) , main=paste("Prop Var Accntd") )
  panelCount = decideOpenGraph( panelCount , finished=TRUE ,
saveName=paste0(saveName,"PostMargZ") )

  #-----------------------------------------------------------------------------
}

#PRELIMINARY FUNCTIONS FOR POSTERIOR INFERENCES



assign_ds <- read.csv("weatherAUS.csv")
assign_ds <- filter(assign_ds, Location == "Melbourne")
assign_ds <- assign_ds[,-23] #Removing RISK_MM variable

colSums(is.na(assign_ds))
str(assign_ds)
summary(assign_ds$RainTomorrow)
summary(assign_ds)
summary(assign_ds)


# Relace NAs with mean+error

assign_ds$MinTemp[is.na(assign_ds$MinTemp)] = round(mean(assign_ds$MinTemp, na.rm = TRUE),2)

assign_ds$MaxTemp[is.na(assign_ds$MaxTemp)] = round(mean(assign_ds$MaxTemp, na.rm =
TRUE),2)


#rainfall - replacing NA values with normally distributed values with variance 1

mean_rainfall <- mean(assign_ds$Rainfall,na.rm = TRUE)


assign_ds[which(is.na(assign_ds[,'Rainfall'])),'Rainfall'] <-
rnorm(length(which(is.na(assign_ds[,'Rainfall']))),mean_rainfall,1)


#sunshine - replacing NA values with normally distributed values with variance 1

assign_ds$Sunshine[is.na(assign_ds$Sunshine)] = round(mean(assign_ds$Sunshine, na.rm = TRUE),1)


#WindGustDir

assign_ds$WindGustDir <- impute(assign_ds$WindGustDir,fun = mean)


#WindGustSpeed - replacing NA values with normally distributed values with variance 1

mean_wgspeed <- mean(assign_ds$WindGustSpeed,na.rm = TRUE)


assign_ds[which(is.na(assign_ds[,'WindGustSpeed'])),'WindGustSpeed'] <-
rnorm(length(which(is.na(assign_ds[,'WindGustSpeed']))),mean_wgspeed,5)


#WindDir9am

assign_ds$WindDir9am <- impute(assign_ds$WindDir9am, fun = median)


#WindDir3pm

assign_ds$WindDir3pm <- impute(assign_ds$WindDir3pm, fun = median)
```

```
#WindSpeed9am - replacing NA values with normally distributed values with variance 1

mean_wspd9 <- mean(assign_ds$WindSpeed9am,na.rm = TRUE)


assign_ds[which(is.na(assign_ds[,'WindSpeed9am'])),'WindSpeed9am'] <-
rnorm(length(which(is.na(assign_ds[,'WindSpeed9am']))),mean_wspd9,2)


#WindSpeed3pm - replacing NA values with normally distributed values with variance 1

mean_wspd3 <- mean(assign_ds$WindSpeed3pm,na.rm = TRUE)


assign_ds[which(is.na(assign_ds[,'WindSpeed3pm'])),'WindSpeed3pm'] <-
rnorm(length(which(is.na(assign_ds[,'WindSpeed3pm']))),mean_wspd3,2)


#Humidity9am - replacing NA values with normally distributed values with variance 1

mean_humid9 <- mean(assign_ds$Humidity9am,na.rm = TRUE)


assign_ds[which(is.na(assign_ds[,'Humidity9am'])),'Humidity9am'] <-
rnorm(length(which(is.na(assign_ds[,'Humidity9am']))),mean_humid9,5)


#Humidity3pm - replacing NA values with normally distributed values with variance 1

mean_humid3 <- mean(assign_ds$Humidity3pm,na.rm = TRUE)


assign_ds[which(is.na(assign_ds[,'Humidity3pm'])),'Humidity3pm'] <-
rnorm(length(which(is.na(assign_ds[,'Humidity3pm']))),mean_humid3,5)




#pressure9am - replacing NA values with normally distributed values with variance 1

mean_p9 <- mean(assign_ds$Pressure9am,na.rm = TRUE)

assign_ds[which(is.na(assign_ds[,'Pressure9am'])),'Pressure9am'] <-
rnorm(length(which(is.na(assign_ds[,'Pressure9am']))),mean_p9,1)
```

```
#pressure3pm - replacing NA values with normally distributed values with variance 1

mean_p3 <- mean(assign_ds$Pressure3pm,na.rm = TRUE)


assign_ds[which(is.na(assign_ds[,'Pressure3pm'])),'Pressure3pm'] <-
rnorm(length(which(is.na(assign_ds[,'Pressure3pm']))),mean_p3,1)



#cloud9am - replacing NA values with normally distributed values with variance 1

mean_c9 <- mean(assign_ds$Cloud9am,na.rm = TRUE)


assign_ds[which(is.na(assign_ds[,'Cloud9am'])),'Cloud9am'] <-
rnorm(length(which(is.na(assign_ds[,'Cloud9am']))),mean_c9,1)



#cloud3pm - replacing NA values with normally distributed values with variance 1

mean_c3 <- mean(assign_ds$Cloud3pm,na.rm = TRUE)


assign_ds[which(is.na(assign_ds[,'Cloud3pm'])),'Cloud3pm'] <-
rnorm(length(which(is.na(assign_ds[,'Cloud3pm']))),mean_c3,1)



#temp9am - replacing NA values with normally distributed values with variance 1

mean_t9 <- mean(assign_ds$Temp9am,na.rm = TRUE)


assign_ds[which(is.na(assign_ds[,'Temp9am'])),'Temp9am'] <-
rnorm(length(which(is.na(assign_ds[,'Temp9am']))),mean_t9,1)



#temp3pm - replacing NA values with normally distributed values with variance 1

mean_t3 <- mean(assign_ds$Temp3pm,na.rm = TRUE)
```

```
assign_ds[which(is.na(assign_ds[,'Temp3pm'])),'Temp3pm'] <-
rnorm(length(which(is.na(assign_ds[,'Temp3pm']))),mean_t3,1)


colSums(is.na(assign_ds))


assign_ds <- na.omit(assign_ds)



#=== 1. Descriptive look ===
# Scatter plots
p1 <- ggplot(assign_ds, aes(x=MinTemp, y = RainTomorrow)) +
  geom_point()
p1

p2 <- ggplot(assign_ds, aes(x=MaxTemp, y = RainTomorrow)) +
  geom_point()
p2
p3 <- ggplot(assign_ds, aes(x=Rainfall, y = RainTomorrow)) +
  geom_point()
p3
p4 <- ggplot(assign_ds, aes(x=Evaporation, y = RainTomorrow)) +
  geom_point()
p4
p5 <- ggplot(assign_ds, aes(x=Sunshine, y = RainTomorrow)) +
  geom_point()
p5
p6 <- ggplot(assign_ds, aes(x=WindGustDir, y = RainTomorrow)) +
  geom_point()
p6
```

```
p7 <- ggplot(assign_ds, aes(x=WindGustSpeed, y = RainTomorrow)) +
  geom_point()
p7
p8 <- ggplot(assign_ds, aes(x=WindDir9am, y = RainTomorrow)) +
  geom_point()
p8
p9 <- ggplot(assign_ds, aes(x=WindDir3pm, y = RainTomorrow)) +
  geom_point()
p9
p10 <- ggplot(assign_ds, aes(x=WindSpeed9am, y = RainTomorrow)) +
  geom_point()
p10

#Scatter plots
p11 <- ggplot(assign_ds, aes(x=WindSpeed3pm, y = RainTomorrow)) +
  geom_point()
p11

p12 <- ggplot(assign_ds, aes(x=Humidity9am, y = RainTomorrow)) +
  geom_point()
p12

p13 <- ggplot(assign_ds, aes(x=Humidity3pm, y = RainTomorrow)) +
  geom_point()
p13

p14 <- ggplot(assign_ds, aes(x=Pressure9am, y = RainTomorrow)) +
  geom_point()
p14
```

```
p15 <- ggplot(assign_ds, aes(x=Pressure3pm, y = RainTomorrow)) +
  geom_point()
p15

p16 <- ggplot(assign_ds, aes(x=Cloud9am, y = RainTomorrow)) +
  geom_point()
p16

p17 <- ggplot(assign_ds, aes(x=Cloud3pm, y = RainTomorrow)) +
  geom_point()
p17

p18 <- ggplot(assign_ds, aes(x=Temp9am, y = RainTomorrow)) +
  geom_point()
p18

p19 <- ggplot(assign_ds, aes(x=Temp3pm, y = RainTomorrow)) +
  geom_point()
p19

p20 <- ggplot(assign_ds, aes(x=RainToday, y = RainTomorrow)) +
  geom_point()
p20

figure <- ggarrange(p2, p3, p4,p6,p7,p9,p11,p13,p15,p17,p19,p20, nrow = 6, ncol = 2)
figure
# assign_bak <- assign_ds
assign_ds <- assign_bak
```

```
table1 <- table(assign_ds$WindGustDir,assign_ds$RainTomorrow)

#Check for which factor has higher influence

prop.table(table1,margin = 2)

assign_ds$WindGustDir <- as.numeric(as.factor(assign_ds$WindGustDir))

#converting as indicating variable

assign_ds$WindGustDir[assign_ds$WindGustDir != 4] <- 0

assign_ds$WindGustDir[assign_ds$WindGustDir == 4] <- 1



table2 <- table(assign_ds$WindDir9am,assign_ds$RainTomorrow)

#Check for which factor has higher influence

prop.table(table2,margin = 2)

assign_ds$WindDir9am <- as.numeric(as.factor(assign_ds$WindDir9am))

#converting as indicating variable

assign_ds$WindDir9am[assign_ds$WindDir9am != 4] <- 0

assign_ds$WindDir9am[assign_ds$WindDir9am == 4] <- 1


table3 <- table(assign_ds$WindDir3pm,assign_ds$RainTomorrow)

#Check for which factor has higher influence

prop.table(table3,margin = 2)


#All the factors contribute equally and hence assumed insignificant


str(assign_ds)


assign_ds$RainTomorrow <- as.numeric(as.factor(assign_ds$RainTomorrow)) - 1 #To obtain 0/1 instead
of 1/2;

assign_ds$RainToday <- as.numeric(as.factor(assign_ds$RainToday)) - 1 # To obtain 0/1 instead of 1/2;

assign_bak1 <- assign_ds
```

```
assign_ds <- assign_ds[,c(1,3:10,12:23)] #removing insignificant variables



# THE DATA.

y = assign_ds[,"RainTomorrow"]

x = as.matrix(assign_ds[,c(2:20)])

str(assign_ds)

x


show(round(cor(x[,1:19]),3))


sample_size = floor(0.75*nrow(assign_ds))

training_val <- sample(1:nrow(assign_ds),sample_size,replace = FALSE) # Find the indexes of
observations going into the training set

test_val <- setdiff(1:nrow(assign_ds),training_val)          # Find the indexes of observations going into
the test set

training_Data <- assign_ds[training_val,]

test_Data <- assign_ds[test_val,]

test_Data1 <- test_Data[,c(-1,-21)]


xPred = as.matrix(test_Data1)


Nx = ncol(x)


# Specify the data in a list, for later shipment to JAGS:

dataList <- list(

  x = x ,

  y = y ,

  xPred = xPred ,

  Ntotal = length(y),

  Nx = Nx,
  Npred = nrow(xPred)
)

# First run without initials!
initsList <- list(
  beta0 = 0,
  beta1 = 0,
  beta2 = 0,
  beta3 = 0,
  beta4 = 0
)

modelString = "
data {
  for ( j in 1:Nx ) {
    xm[j]  <- mean(x[,j])
    xsd[j] <-   sd(x[,j])
    for ( i in 1:Ntotal ) {
      zx[i,j] <- ( x[i,j] - xm[j] ) / xsd[j]
    }
  }
}

model {
  for ( i in 1:Ntotal ) {
    # In JAGS, ilogit is logistic:
    y[i] ~ dbern( mu[i] )
      mu[i] <- ( guess*(1/2) + (1.0-guess)*ilogit(beta0+sum(beta[1:Nx]*x[i,1:Nx])) )
  }
  # Priors vague on standardized scale:
```

```
  zbeta0 ~ dnorm( 0 , 1/2^2 )
  # non-informative run
  for ( j in 1:Nx ) {
    zbeta[j] ~ dnorm( 0 , 1/2^2 )
  }
  guess ~ dbeta(1,9)
  # Transform to original scale:
  beta[1:Nx] <- zbeta[1:Nx] / xsd[1:Nx]
  beta0 <- zbeta0 - sum( zbeta[1:Nx] * xm[1:Nx] / xsd[1:Nx] )

  # Compute predictions at every step of the MCMC
  for ( k in 1:Npred){
    pred[k] <- ilogit(beta0 + sum(beta[1:Nx] * xPred[k,1:Nx]))
  }
}
" # close quote for modelString
# Write out modelString to a text file
writeLines( modelString , con="TEMPmodel.txt" )

# parameters = c( "zbeta0" , "beta0")
parameters = c( "beta0")
for ( i in 1:Nx){
  # parameters = c(parameters, paste0("zbeta[",i,"]"), paste0("beta[",i,"]"))
  parameters = c(parameters, paste0("beta[",i,"]"))
}
for ( i in 1:nrow(xPred)){
  parameters = c(parameters, paste0("pred[",i,"]"))
}

parameters = c(parameters, "guess")
```

```
adaptSteps = 2000  # Number of steps to "tune" the samplers
burnInSteps = 7000
nChains = 4
thinSteps = 13
numSavedSteps = 6000
nIter = ceiling( ( numSavedSteps * thinSteps ) / nChains )

runJagsOut <- run.jags( method="parallel" ,
                        model="TEMPmodel.txt" ,
                        monitor=parameters  ,
                        data=dataList ,
                        n.chains=nChains ,
                        adapt=adaptSteps ,
                        burnin=burnInSteps ,
                        sample=numSavedSteps ,
                        thin=thinSteps , summarise=FALSE , plots=FALSE )
codaSamples = as.mcmc.list( runJagsOut )

diagMCMC( codaSamples , parName="beta0" )
for ( i in 1:Nx){
  diagMCMC( codaSamples , parName=paste0("beta[",i,"]") )
}
# diagMCMC( codaSamples , parName="zbeta0" )
# for ( i in 1:Nx){
#   diagMCMC( codaSamples , parName=paste0("zbeta[",i,"]") )
# }
# for ( i in 1:nrow(xPred)){
#   diagMCMC( codaSamples , parName=paste0("pred[",i,"]") )
# }
```

```
compVal <- data.frame("beta0" = 15, "beta[1]" = 0, "beta[2]" = 0, "beta[3]" = 0, "beta[4]" =  0,  "beta[5]"
=  0,
                      "beta[6]" =  0, "beta[7]" = 0,"beta[8]" = 0, "beta[9]" = 0,"beta[10]" = 0,
                      "beta[11]" = 0,"beta[12]" = 0, "beta[13]" = 0, "beta[14]" = 0, "beta[15]" = 0,
                      "beta[16]" = 0, "beta[17]" = 0, "beta[18]" = 0, "beta[19]" = 0, check.names=FALSE)

summaryInfo <- smryMCMC_HD( codaSamples = codaSamples , compVal = compVal )
print(summaryInfo)

plotMCMC_HD( codaSamples = codaSamples , data = assign_ds,
xName=c("MinTemp","MaxTemp","Rainfall","Evaporation","Sunshine","WindGustDir","WindGustSpee
d","WindDir9am","WindSpeed9am","WindSpeed3pm","Humidity9am","Humidity3pm","Pressure9am","
Pressure3pm","Cloud9am","Cloud3pm","Temp9am","Temp3pm","RainToday") ,
             yName="RainTomorrow", compVal = compVal, preds = FALSE)

# Predictions for full records in training set
preds <- data.frame(date = test_Data[,1], PredProb = summaryInfo[22:596,3], actual = test_Data[,21] )

threshold <- 0.5# summaryInfo[427,3]
preds[which(preds[,2]<threshold),3] <- 0
preds[which(preds[,2]>threshold),3] <- 1

predsSorted <- preds[order(preds$date),]

table(preds$actual)

actualrain <- test_Data[which(test_Data$Date %in% predsSorted$date),21]
# ============ Predictive check ============
confusionMatrix <- function(resp, pred){
  classRes <- data.frame(response = resp , predicted = pred)
  conf = xtabs(~ predicted + response, data = classRes)
  accuracy = sum(diag(conf))/sum(conf)
  accuracy
  precision = conf[1,1]/(conf[1,1]+conf[1,2])
  precision
  recall = conf[1,1]/(conf[1,1]+conf[2,1])
  recall
  Fscore = 2*((precision*recall)/(precision+recall))
  Fscore
  return(list(Accuracy = accuracy, Precision = precision, Recall = recall, Fscore = Fscore,
Confusion_Matrix = conf))
}

confusionMatrix(resp = actualrain, pred = predsSorted[,3])

# Saving Data
save.image(file = "firstrun.RData")
save.image(file = "secondrun.RData")
save.image(file = "thirdrun.RData")
save.image(file = "fourthrun.RData")
```

REFERENCES

[1] Rain in Australia. (2020). Retrieved 8 October 2020, from https://www.kaggle.com/jsphyg/weather-dataset-rattle-package?select=weatherAUS.csv