MATH2269/2305 APPLIED BAYESIAN STATISTICS

FINAL PROJECT REPORT

Name: Ratnak Saha

RMIT UNIVERSITY,

SCHOOL OF SCIENCE

1 INTRODUCTION

Access to clean, potable water is crucial for public health, making accurate water quality assessment essential in addressing environmental concerns and ensuring safe water consumption. Traditional water quality testing methods rely on measuring various chemical and physical parameters, but predictive modeling provides an efficient alternative to classify water quality. The ability to classify water samples as potable (safe for drinking) or non-potable based on these measurements can significantly aid water resource management and safeguard human health. This report focuses on using Bayesian Logistic Regression to predict water potability based on nine key physicochemical attributes: pH, hardness, solids (TDS), chloramines, sulfate, conductivity, organic carbon, trihalomethanes, and turbidity.

The primary research question guiding this analysis is: *Can Bayesian Logistic Regression effectively classify water samples as potable or non-potable based on their chemical and physical properties?*

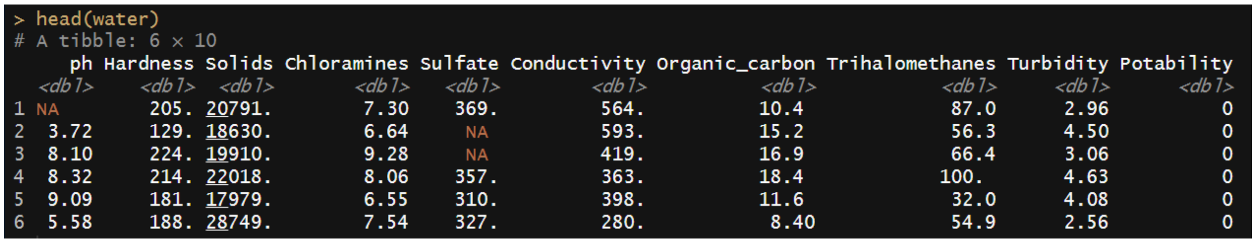
In this analysis, multiple Bayesian models will be compared, including models using informative and non-informative priors. Informative priors incorporate domain-specific knowledge about the relationships between water quality variables and potability, while non-informative priors allow the data to speak for itself by assuming minimal prior knowledge. By comparing these different models, the analysis will explore how incorporating prior beliefs affects the classification accuracy and uncertainty of predictions. The goal is to evaluate the effectiveness of Bayesian Logistic Regression in predicting water potability, assess the impact of prior knowledge on model performance, and interpret the roles of key water quality parameters in determining whether water is safe for drinking.

2 DATASET DESCRIPTION

The dataset used for this analysis comes from the Kaggle Water Quality and Potability dataset, which includes data on various physicochemical properties of water samples and a binary indicator of potability. The dataset contains 3,276 rows (samples) and 10 columns (variables), where the columns represent both predictor variables (water quality measurements) and the target variable, *Potability*. The goal of the dataset is to classify whether a water sample is potable (safe for drinking) or non-potable based on its chemical and physical properties.

Each variable is of numeric datatype, and the dataset contains missing values in some of the predictor columns, which need to be handled as part of the preprocessing steps. The aim of the analysis is to predict the *Potability* variable using the nine water quality features.

Figure 1: Head of the Dataset



```
> head(water)
# A tibble: 6 × 10
     ph Hardness Solids Chloramines Sulfate Conductivity Organic_carbon Trihalomethanes Turbidity Potability
  <dbl>    <dbl>  <dbl>       <dbl>   <dbl>        <dbl>          <dbl>           <dbl>     <dbl>      <dbl>
1 NA        205. 20791.        7.30    369.         564.          10.4             87.0      2.96          0
2  3.72     129. 18630.        6.64     NA          593.          15.2             56.3      4.50          0
3  8.10     224. 19910.        9.28     NA          419.          16.9             66.4      3.06          0
4  8.32     214. 22018.        8.06    357.         363.          18.4            100.       4.63          0
5  9.09     181. 17979.        6.55    310.         398.          11.6             32.0      4.08          0
6  5.58     188. 28749.        7.54    327.         280.           8.40            54.9      2.56          0
```

Name	Data Type	Description
Predictor		
pH	Numeric	The pH level of the water, which indicates its acidity or alkalinity.
Hardness	Numeric	A measure of the mineral content in the water.
Solids	Numeric	Total dissolved solids (TDS) in the water, measured in parts per million.
Chloramines	Numeric	Concentration of chloramines in the water, a disinfectant used in water treatment.
Sulfate	Numeric	Concentration of sulfate ions in the water.
Conductivity	Numeric	Electrical conductivity of the water, which reflects its ion concentration.
Organic Carbon	Numeric	Amount of organic carbon in the water, indicating organic material content.
Trihalomethanes	Numeric	Concentration of trihalomethanes, a byproduct of water disinfection.
Turbidity	Numeric	The turbidity level of the water, which measures its clarity or cloudiness.
Target		
Potability	Binary	The target variable indicating the potability of the water, with 1 representing potable (safe for drinking) and 0 representing non-potable.

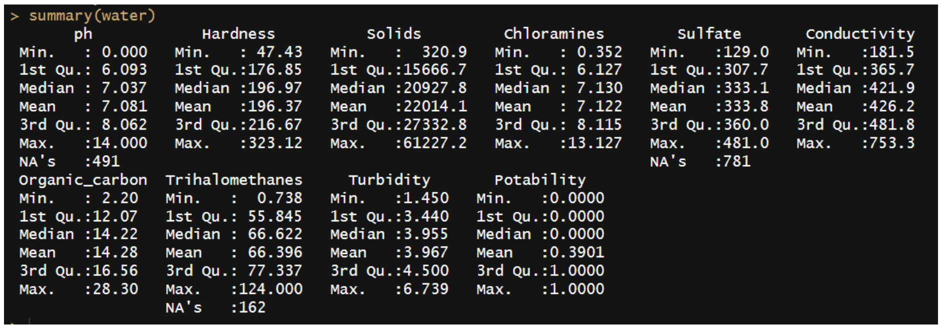
Table 1: Variables Description

3 METHODOLOGY

3.1 EXPLANATORY DATA ANALYSIS

To gain a better understanding of the dataset and its characteristics, an Exploratory Data Analysis (EDA) was performed. This involved summarizing key statistics, visualizing the distributions of both the predictor and dependent variables, examining potential outliers, and analyzing the relationships between the variables through correlation heatmaps.

The dataset consists of 3,276 observations and 10 variables, including 9 predictor variables and 1 target variable, *Potability*. The summary statistics (as seen in the descriptive statistics table) provide insights into the range, mean, median, and distribution of values for each variable.



```
> summary(water)
       ph            Hardness           Solids         Chloramines         Sulfate        Conductivity
 Min.   : 0.000   Min.   : 47.43   Min.   :  320.9   Min.   : 0.352   Min.   :129.0   Min.   :181.5
 1st Qu.: 6.093   1st Qu.:176.85   1st Qu.:15666.7   1st Qu.: 6.127   1st Qu.:307.7   1st Qu.:365.7
 Median : 7.037   Median :196.97   Median :20927.8   Median : 7.130   Median :333.1   Median :421.9
 Mean   : 7.081   Mean   :196.37   Mean   :22014.1   Mean   : 7.122   Mean   :333.8   Mean   :426.2
 3rd Qu.: 8.062   3rd Qu.:216.67   3rd Qu.:27332.8   3rd Qu.: 8.115   3rd Qu.:360.0   3rd Qu.:481.8
 Max.   :14.000   Max.   :323.12   Max.   :61227.2   Max.   :13.127   Max.   :481.0   Max.   :753.3
 NA's   :491                                                          NA's   :781
 Organic_carbon  Trihalomethanes     Turbidity       Potability
 Min.   : 2.20   Min.   :  0.738   Min.   :1.450   Min.   :0.0000
 1st Qu.:12.07   1st Qu.: 55.845   1st Qu.:3.440   1st Qu.:0.0000
 Median :14.22   Median : 66.622   Median :3.955   Median :0.0000
 Mean   :14.28   Mean   : 66.396   Mean   :3.967   Mean   :0.3901
 3rd Qu.:16.56   3rd Qu.: 77.337   3rd Qu.:4.500   3rd Qu.:1.0000
 Max.   :28.30   Max.   :124.000   Max.   :6.739   Max.   :1.0000
                 NA's   :162
```

Figure 2: Summary Statistics of the dataset

- The mean pH level is approximately 7.08, with values ranging from 0 to 14, indicating that some extreme values are present.
- Water hardness, total dissolved solids (Solids), and other key water quality measures also exhibit significant variability across the dataset.
- The Potability variable is imbalanced, with more non-potable (0) than potable (1) water samples.
- Additionally, the dataset contains missing values in the following columns:
  - **pH**: 491 missing values.
  - **Sulfate**: 781 missing values.
  - **Trihalomethanes**: 162 missing values.

These missing values were taken into consideration for imputation or handling during the model-building process.

Figure 3: Histograms of Predictor Variables

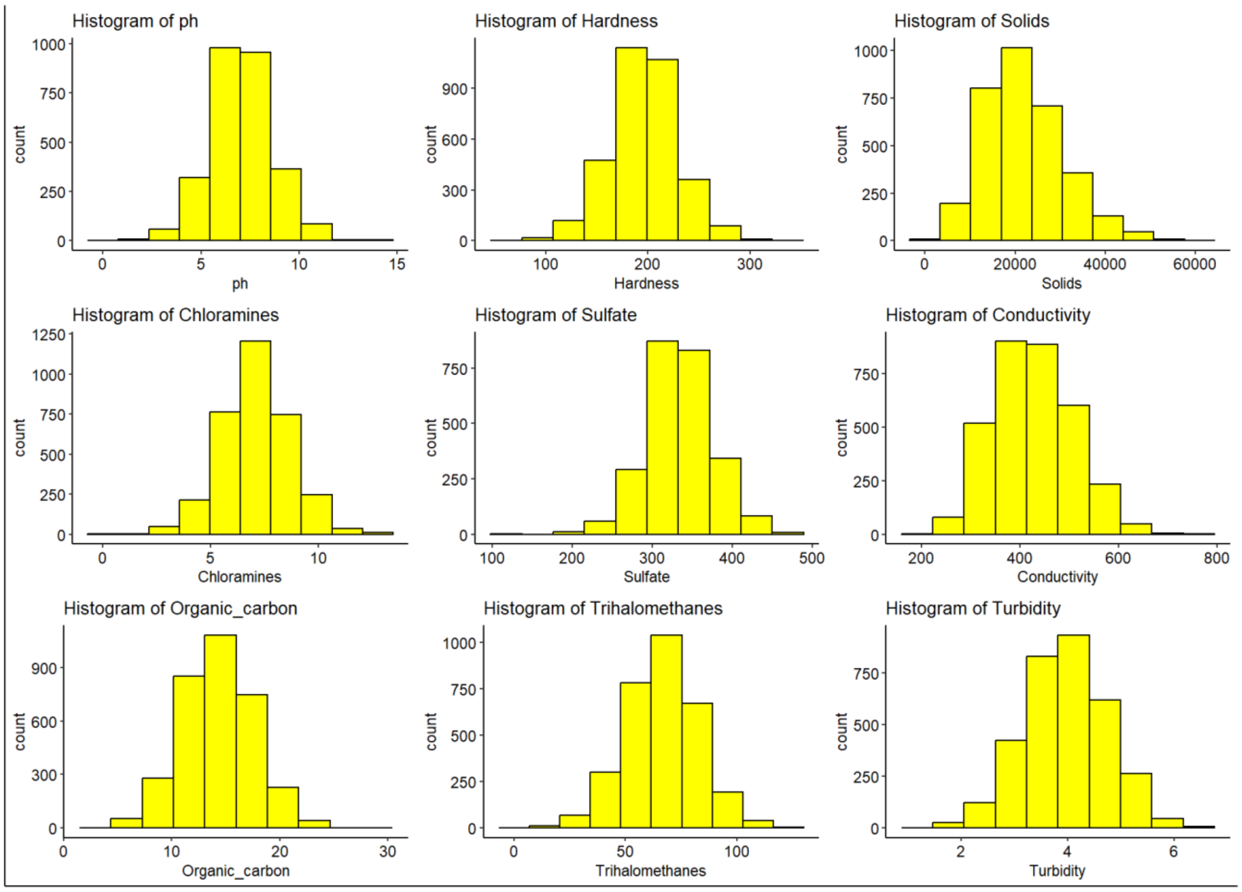


Figure 4: Histogram of Dependent Variable

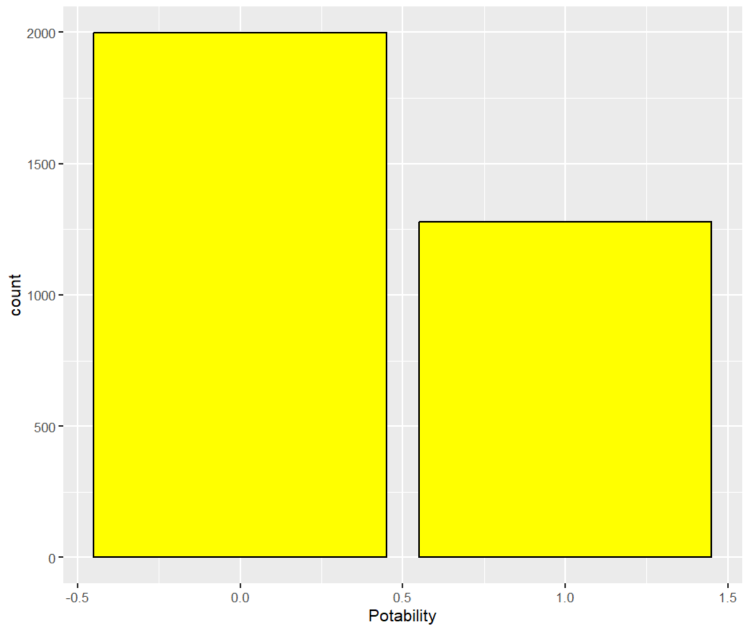


Figure 3 and 4 shows the histograms plotted for all predictor variables and the target variable (*Potability*). These histograms provide a visual understanding of the distribution of each variable. Key takeaways from the histograms:

- Most of the predictor variables, such as *pH*, *Hardness*, *Solids*, and *Chloramines*, exhibit a relatively normal distribution, though some skewness is present, particularly in variables like *Solids* and *Trihalomethanes*.
- The target variable (*Potability*) shows class imbalance, with a higher count of non-potable samples (0) compared to potable samples (1).

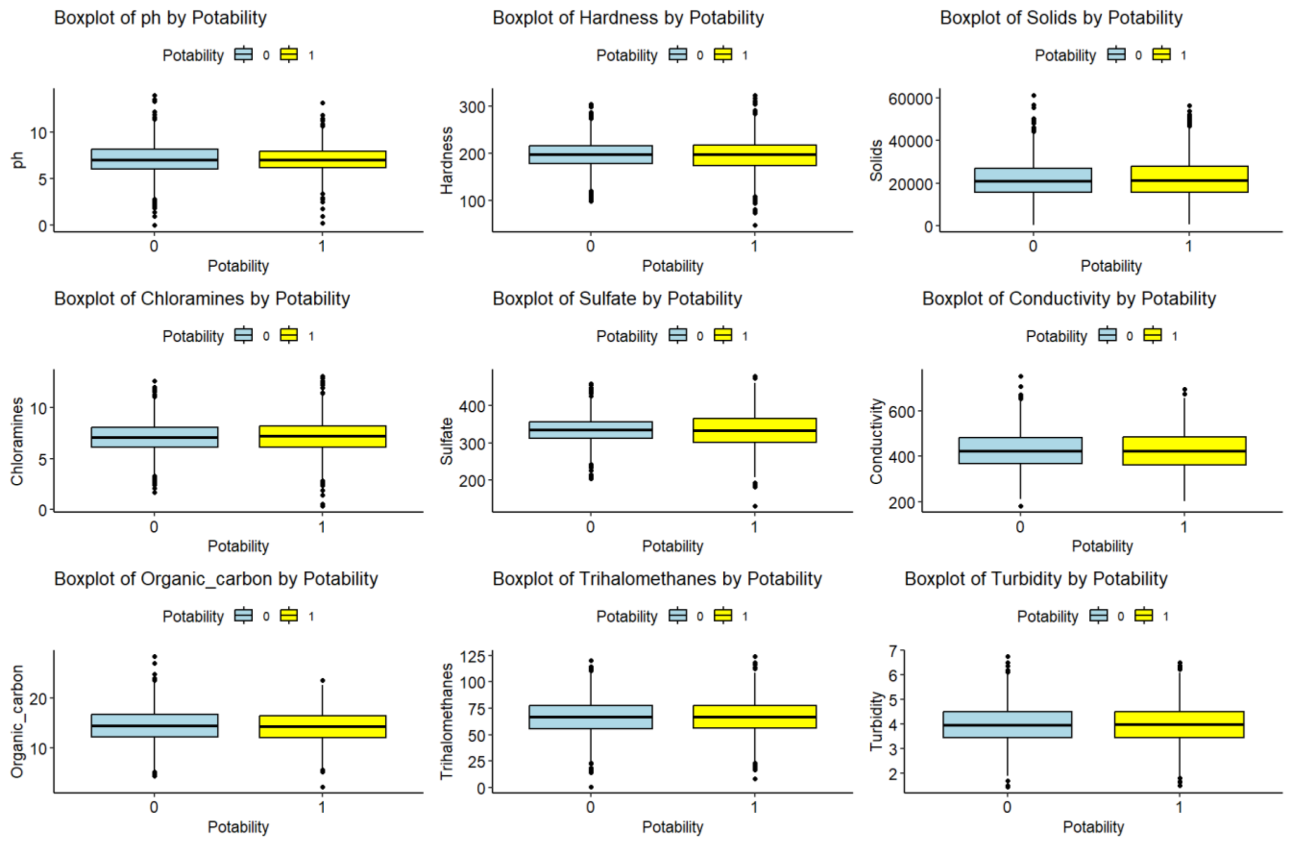


Figure 5: Boxplot of Independent Variables grouped by Potability

Figure 5 displays the boxplots to show the distribution of each predictor variable across the two classes of the dependent variable (Potability = 0 and Potability = 1). These boxplots indicate that there is no notable difference in the distributions between potable and non-potable water samples across most characteristics. For example:

- The median values and interquartile ranges (IQRs) for variables such as *pH*, *Hardness*, *Sulfate*, and *Chloramines* are similar across both potable and non-potable water samples.
- While some variables, like *Sulfate* and *pH*, show slight shifts between the two classes, these differences are not substantial enough to suggest strong separability between potable and non-potable water samples based on these individual characteristics alone.

Additionally, the boxplots reveal the presence of several outliers across all predictor variables, indicating that extreme values exist in the dataset. These outliers may impact model performance and will need to be addressed during the data preprocessing and modeling stages.

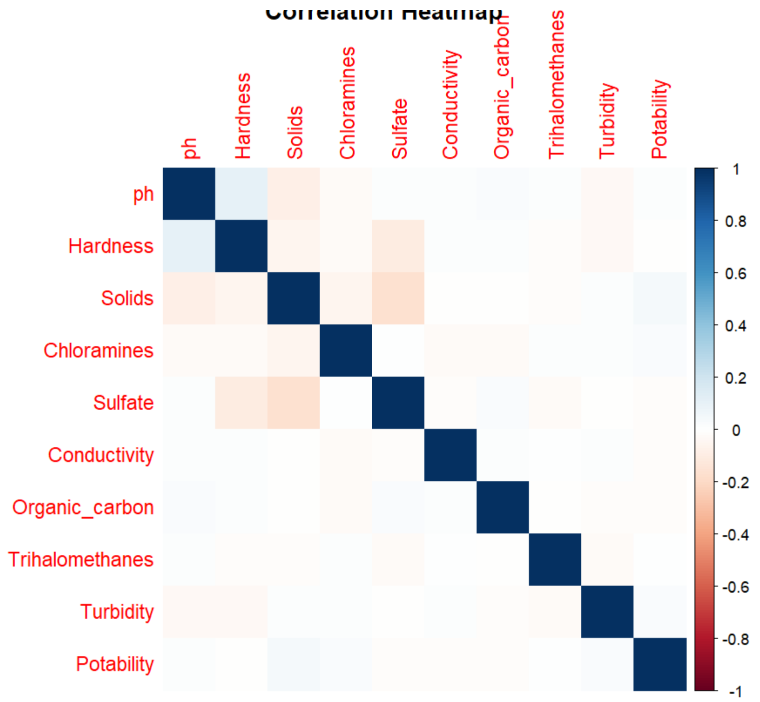


Figure 6: Correlation Heatmap

The correlation heatmap in Figure 6 visualizes the relationships between all predictor variables and the target variable. The heatmap shows that there is no notable correlation between the predictor variables, with all correlations being weak or close to zero. This suggests that the dataset does not suffer from multicollinearity issues, meaning the predictor variables are independent of each other, which simplifies the modeling process. Additionally, the weak correlations between the predictor variables and the target variable (*Potability*) indicate that no single feature has a particularly strong influence on water potability classification.

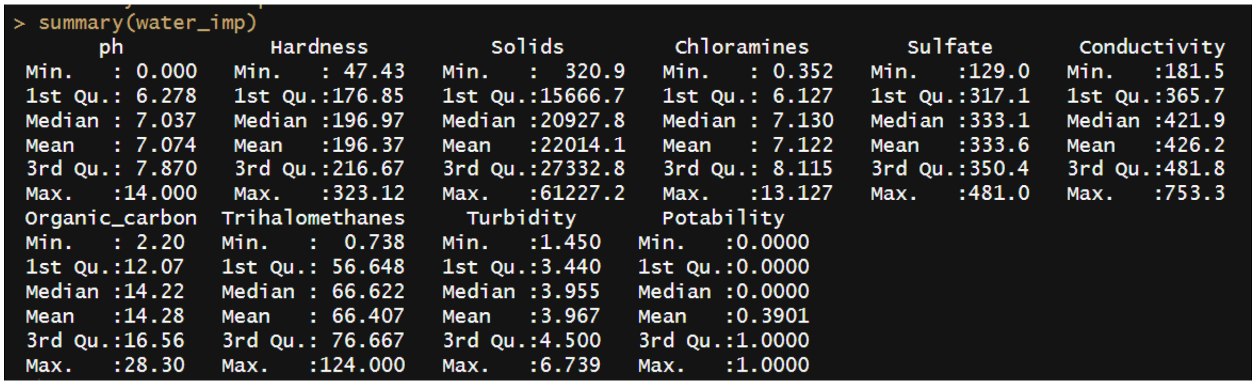
3.2 DATA PREPROCESSING

In preparation for modeling, preprocessing steps were applied to the dataset to ensure data quality and suitability for analysis. One key preprocessing step involved handling missing values, which were present in several predictor columns, including *pH*, *Sulfate*, and *Trihalomethanes*. Given the significant proportion of missing data, removing rows with missing values would have resulted in a substantial loss of information, potentially affecting the model's performance and reducing its ability to make accurate predictions.

Variable	Potability – 0			Potability – 1		
	Mean	**Median**	**% of NA**	**Mean**	**Median**	**% of NA**
pH	7.09	7.04	15.7	7.07	7.03	13.84
Sulfate	334.5	333.3	24.42	332.5	331.8	22.92
Trihalomethanes	66.3	66.54	5.35	66.53	66.67	4.3

Table 2: Summary of the missing values in the dataset

To address this, the missing values were imputed using the median values of their respective columns. The mean and median values for each variable in both potability classes (Potability = 0 and Potability = 1) are similar and close to each other (Table 2), indicating consistency in central tendency across classes. This similarity suggests that the distributions of these variables do not differ significantly between potable and non-potable water samples, which justifies the use of the median for imputation. The decision to impute with the median was also based on its robustness to outliers, which are present in the dataset, ensuring that extreme values do not skew the imputed data. This method allowed us to preserve the overall structure of the data while addressing the issue of missing values .



```
> summary(water_imp)
       ph            Hardness           Solids           Chloramines         Sulfate        Conductivity
 Min.   : 0.000   Min.   : 47.43   Min.   :  320.9   Min.   : 0.352   Min.   :129.0   Min.   :181.5
 1st Qu.: 6.278   1st Qu.:176.85   1st Qu.:15666.7   1st Qu.: 6.127   1st Qu.:317.1   1st Qu.:365.7
 Median : 7.037   Median :196.97   Median :20927.8   Median : 7.130   Median :333.1   Median :421.9
 Mean   : 7.074   Mean   :196.37   Mean   :22014.1   Mean   : 7.122   Mean   :333.6   Mean   :426.2
 3rd Qu.: 7.870   3rd Qu.:216.67   3rd Qu.:27332.8   3rd Qu.: 8.115   3rd Qu.:350.4   3rd Qu.:481.8
 Max.   :14.000   Max.   :323.12   Max.   :61227.2   Max.   :13.127   Max.   :481.0   Max.   :753.3
 Organic_carbon  Trihalomethanes     Turbidity       Potability
 Min.   : 2.20   Min.   :  0.738   Min.   :1.450   Min.   :0.0000
 1st Qu.:12.07   1st Qu.: 56.648   1st Qu.:3.440   1st Qu.:0.0000
 Median :14.22   Median : 66.622   Median :3.955   Median :0.0000
 Mean   :14.28   Mean   : 66.407   Mean   :3.967   Mean   :0.3901
 3rd Qu.:16.56   3rd Qu.: 76.667   3rd Qu.:4.500   3rd Qu.:1.0000
 Max.   :28.30   Max.   :124.000   Max.   :6.739   Max.   :1.0000
```

Figure 7: Summary Statistics after Imputing Missing Values

The summary statistics before and after (shown in Figure 7) the imputation process show that the distribution of the variables has remained largely unchanged. These similarities in summary statistics suggest that the imputation process has been effective in filling missing values without significantly distorting the original data characteristics. (Detailed descriptive plots are given in the appendix)

After imputing the missing data, the dataset was split into training and test sets in an 80:20 ratio. The training set, which consists of 80% of the data, was used to train the Bayesian Logistic Regression models. The test set, comprising the remaining 20%, was reserved for evaluating the model's predictive performance. This split allows for an unbiased assessment of the model's ability to generalize to unseen data and provides a reliable measure of its accuracy.

3.3 MODEL SPECIFICATION

In this analysis, a robust Bayesian Logistic Regression model is used to classify the potability of water based on various physicochemical properties. Given the presence of outliers in the dataset, a *guess parameter,* **α**, is incorporated into the model to account for the potential influence of these outliers. This approach helps enhance the model's robustness by mitigating the impact of extreme values on the regression coefficients. The likelihood function used in this model is based on the **Bernoulli distribution**. Since the outcome variable, *Potability*, is binary (1 for potable, 0 for non-potable), the likelihood of each observation is modeled as a Bernoulli distribution with probability Y_i, where Y_i is the probability that the ith sample is potable.

The logistic function is used to model the probability Y_i as a function of the predictor variables:

$$Y_i \sim Bernoulli(\mu)$$

where,

$$\mu = \alpha * 1/2 + (1 - \alpha) * \frac{1}{\exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 + \beta_8 X_8 + \beta_9 X_9)}$$

$$= \alpha * 1/2 + (1 - \alpha) * logistic(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 + \beta_8 X_8 + \beta_9 X_9)$$

Therefore,

$$Y \sim Bern(logistic(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + \beta_6 X_6 + \beta_7 X_7 + \beta_8 X_8 + \beta_9 X_9)$$

where,

pH – X1; **Hardness** – X2; **Solids** – X3; **Chloramines** – X4; **Sulfate** – X5; **Conductivity** – X6; **Organic_carbon** – X7; **Trihalomethanes** – X8; **Turbidity** – X9

The prior for the guess parameter is specified as: **α ~ Beta(2,8)**

This Beta prior reflects the belief that the model is generally reliable, with only a small proportion of predictions expected to be affected by outliers. The shape parameters (2, 8) indicate a low prior mean for α, suggesting a minimal probability of random guessing, but it allows for some flexibility.

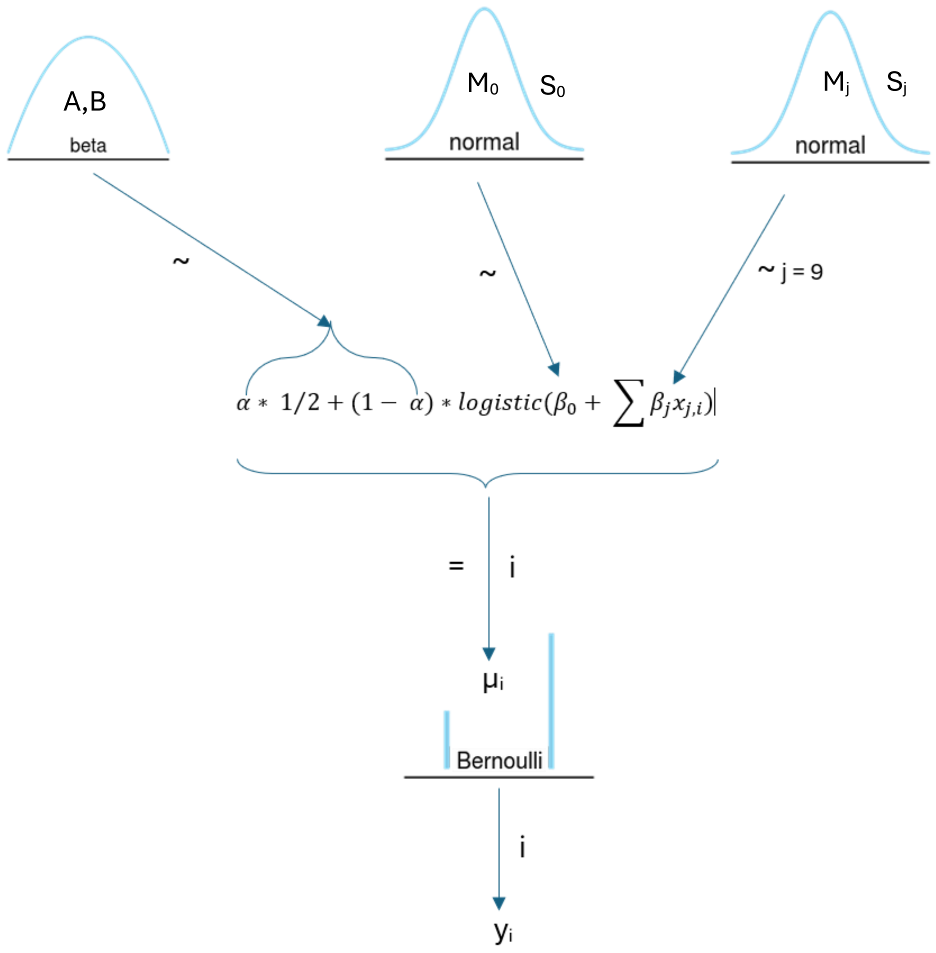


Figure 8: JAGS Model Diagram for the logistic regression model

3.3.1 Prior Information

Both informative and non-informative priors are used in this analysis to explore the effects of prior knowledge on model performance and to assess the sensitivity of the Bayesian logistic regression model to different assumptions.

3.3.1.1 Informative Priors

For this analysis, informative priors were specified for each predictor based on external domain knowledge about water quality and its effect on potability. These priors are centered at 0 to reflect the belief that the relationship between each predictor and potability could be either positive or negative, with varying degrees of uncertainty expressed by the variances of the priors.

Here are the informative priors used for each predictor:

- **Intercept**: $\beta_0 \sim N(0,2^2)$
    - **Moderate belief**: The intercept term reflects the baseline log-odds of potability when all predictors are held at zero.
- **pH**: $\beta_1 \sim N(0,1^2)$
    - **Moderate belief**: pH is crucial in water quality, with acceptable ranges for drinking water (6.5–8.5). A moderate belief reflects that pH plays a significant role.
- **Hardness**: $\beta_2 \sim N(0,4^2)$
    - **Weak belief**: Hardness affects taste but has a lesser impact on health. A weak belief is applied here.
- **Total Dissolved Solids (TDS)**: $\beta_3 \sim N(0,0.5^2)$
    - **Strong belief**: High TDS can indicate contaminants, making it an important predictor for water quality. A strong belief is applied with lower variance.
- **Chloramines**: $\beta_4 \sim N(0,1.5^2)$
    - **Moderate belief**: Chloramines are commonly used in water disinfection, and their safe levels are well established, justifying a moderate belief.
- **Sulfate**: $\beta_5 \sim N(0,2^2)$
    - **Weak to moderate belief**: Sulfate can cause bitterness and digestive issues at high concentrations, leading to a weak-to-moderate prior.
- **Conductivity**: $\beta_6 \sim N(0,0.5^2)$
    - **Strong belief**: Conductivity is related to the ion concentration in water, making it a strong indicator of potability.
- **Organic Carbon**: $\beta_7 \sim N(0,1.5^2)$
    - **Moderate belief**: Organic carbon can lead to harmful byproducts during water treatment, leading to a moderate belief.

- **Trihalomethanes**: $\beta_8 \sim N(0,1^2)$
    - **Moderate to strong belief**: Trihalomethanes are known byproducts of water chlorination, with well-known risks. Hence, a moderate-to-strong belief is applied.
- **Turbidity**: $\beta_9 \sim N(0,0.5^2)$
    - **Strong belief**: Turbidity indicates the presence of contaminants, making it an important factor in water safety. A strong belief is appropriate here.

3.3.1.2 Non-Informative Priors

For comparison, a non-informative prior is also used to assess the sensitivity of the model to prior assumptions. Non-informative priors represent a lack of strong prior beliefs about the parameters and allow the data to have a greater influence on the posterior estimates.

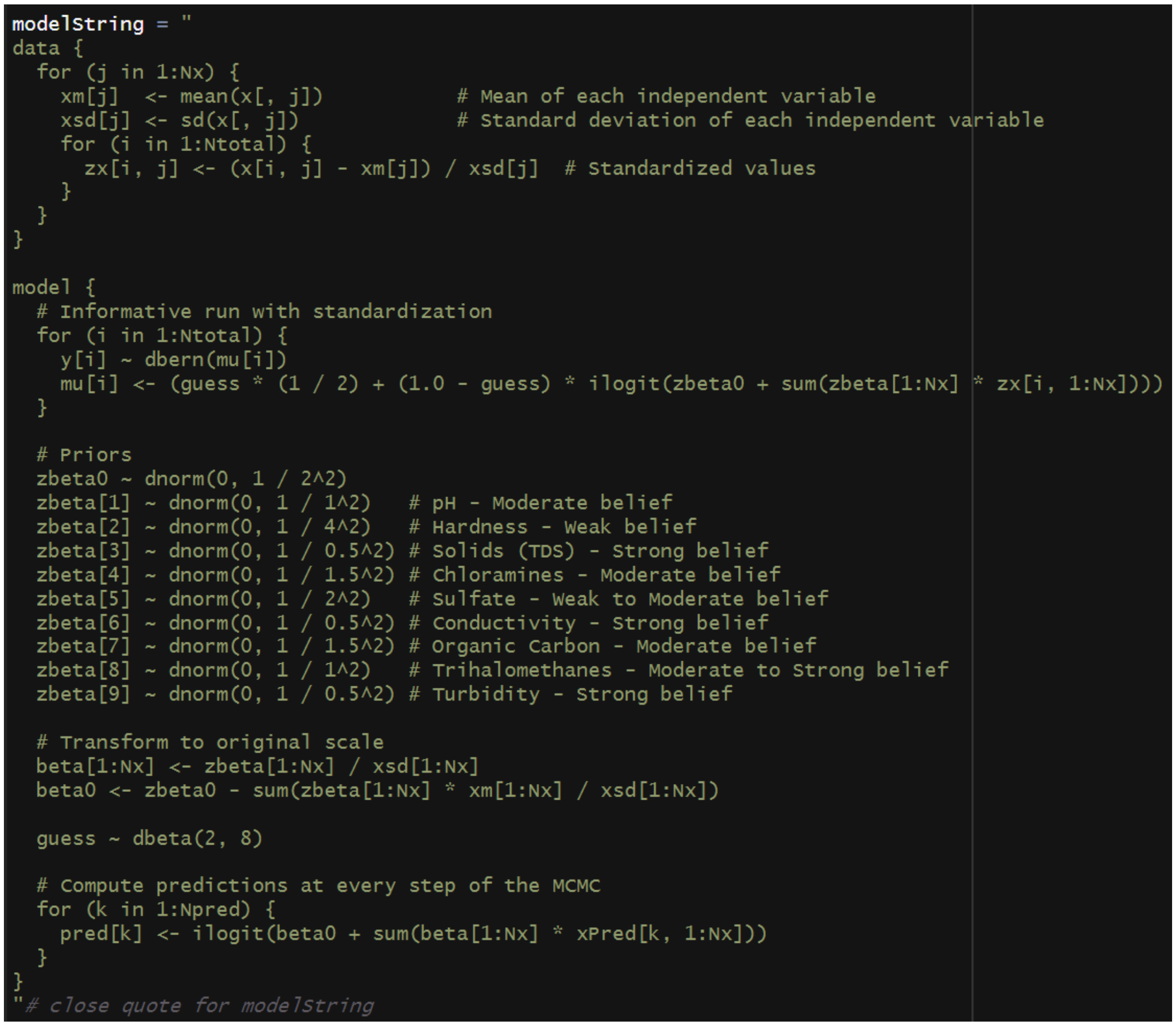
In this model, the non-informative priors are specified as: $\boldsymbol{\beta_j \sim N(0, 10^2)}$

This prior distribution has a large variance, which reflects vague prior knowledge about the parameters. This allows the model to be more flexible and lets the data primarily dictate the estimates of the regression coefficients.

The combination of these priors, along with the guess parameter, helps improve the robustness of the Bayesian logistic regression model in the presence of outliers while allowing an exploration of the effects of prior knowledge on model performance. This setup also provides a way to assess the sensitivity of the model to different prior assumptions.

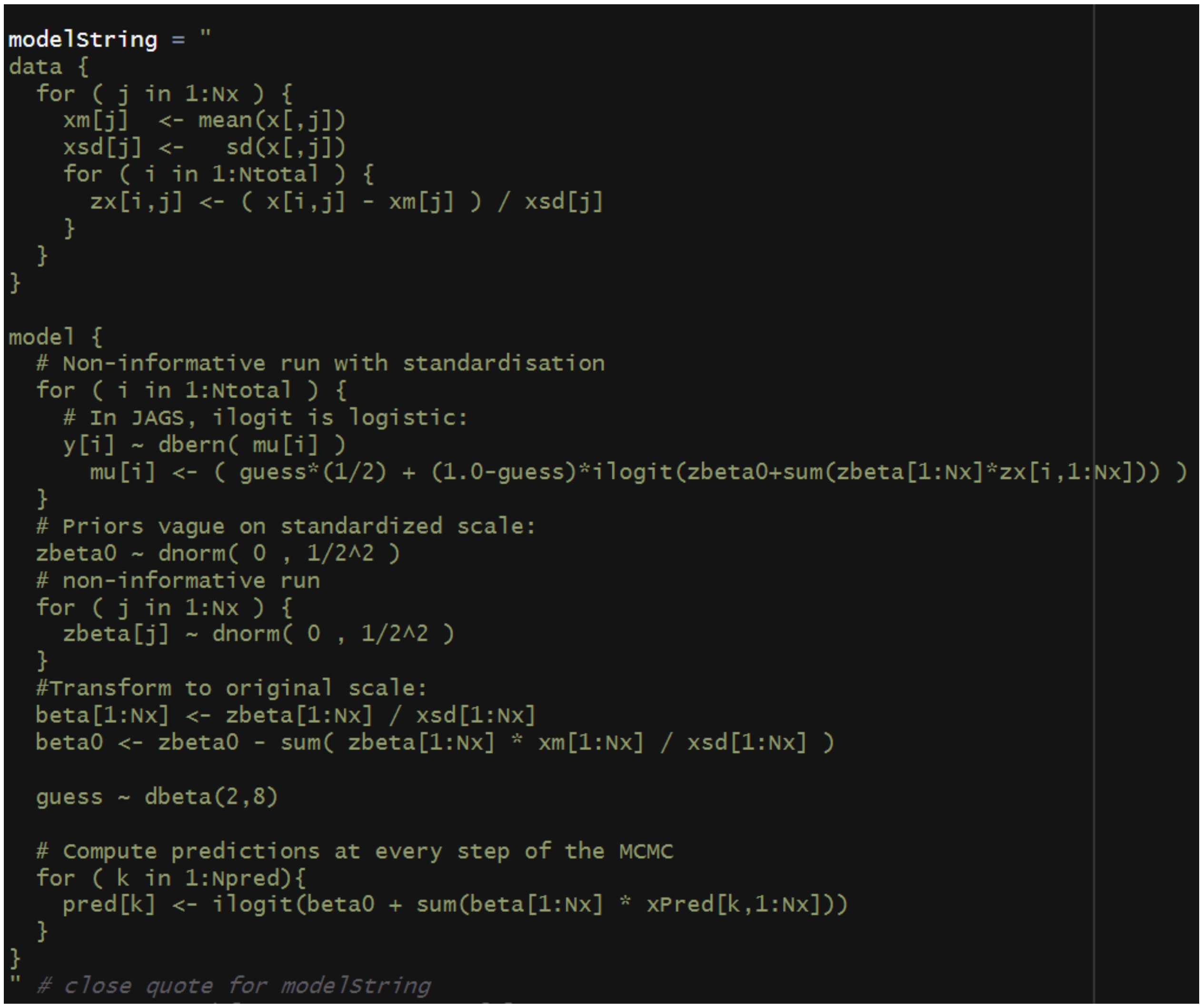
3.4 MCMC Implementation and Diagnostics

In this analysis, the Bayesian Logistic Regression model was implemented in JAGS (Just Another Gibbs Sampler) to perform Markov Chain Monte Carlo (MCMC) sampling. MCMC methods are essential in Bayesian inference as they allow for the generation of posterior samples for model parameters. These samples provide estimates for parameter distributions, enabling probabilistic predictions. The JAGS model was structured into a *model block* and a *data block*, and the MCMC chains were run in parallel to enhance computational efficiency. The JAGS model was run using parallel processing to improve computation time. By running multiple chains in parallel, the overall runtime is reduced significantly compared to running chains sequentially. This approach also provides better convergence diagnostics, as multiple independent chains allow for more accurate assessment of convergence and sampling quality.



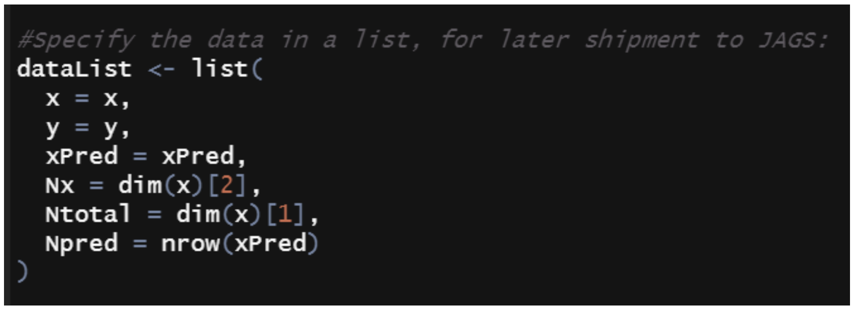
```
modelString = "
data {
  for (j in 1:Nx) {
    xm[j]  <- mean(x[, j])          # Mean of each independent variable
    xsd[j] <- sd(x[, j])            # Standard deviation of each independent variable
    for (i in 1:Ntotal) {
      zx[i, j] <- (x[i, j] - xm[j]) / xsd[j]  # Standardized values
    }
  }
}

model {
  # Informative run with standardization
  for (i in 1:Ntotal) {
    y[i] ~ dbern(mu[i])
    mu[i] <- (guess * (1 / 2) + (1.0 - guess) * ilogit(zbeta0 + sum(zbeta[1:Nx] * zx[i, 1:Nx])))
  }

  # Priors
  zbeta0 ~ dnorm(0, 1 / 2^2)
  zbeta[1] ~ dnorm(0, 1 / 1^2)   # pH - Moderate belief
  zbeta[2] ~ dnorm(0, 1 / 4^2)   # Hardness - Weak belief
  zbeta[3] ~ dnorm(0, 1 / 0.5^2) # Solids (TDS) - Strong belief
  zbeta[4] ~ dnorm(0, 1 / 1.5^2) # Chloramines - Moderate belief
  zbeta[5] ~ dnorm(0, 1 / 2^2)   # Sulfate - Weak to Moderate belief
  zbeta[6] ~ dnorm(0, 1 / 0.5^2) # Conductivity - Strong belief
  zbeta[7] ~ dnorm(0, 1 / 1.5^2) # Organic Carbon - Moderate belief
  zbeta[8] ~ dnorm(0, 1 / 1^2)   # Trihalomethanes - Moderate to Strong belief
  zbeta[9] ~ dnorm(0, 1 / 0.5^2) # Turbidity - Strong belief

  # 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])

  guess ~ dbeta(2, 8)

  # 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
```

Figure 9: Model String for Informative Model



```
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 {
  # Non-informative run with standardisation
  for ( i in 1:Ntotal ) {
    # In JAGS, ilogit is logistic:
    y[i] ~ dbern( mu[i] )
      mu[i] <- ( guess*(1/2) + (1.0-guess)*ilogit(zbeta0+sum(zbeta[1:Nx]*zx[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 )
  }
  #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] )

  guess ~ dbeta(2,8)

  # 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
```

Figure 10: Model String for Non-Informative Model



```
#Specify the data in a list, for later shipment to JAGS:
dataList <- list(
  x = x,
  y = y,
  xPred = xPred,
  Nx = dim(x)[2],
  Ntotal = dim(x)[1],
  Npred = nrow(xPred)
)
```

Figure 11: Data Block for the Model

In Figure 9 and 10, the model block in JAGS specifies the likelihood function, the logistic regression model, and the priors for each parameter. In this case, the likelihood function follows a **Bernoulli distribution** for the binary outcome variable (Potability), where each sample's probability of being potable is estimated through the logistic function. The guess parameter, modeled with a Beta(2, 8) prior, was included to account for potential misclassification due to outliers, enhancing the model's robustness. Each predictor variable was assigned a prior based on informative prior (Figure 9) or a non-informative prior (Figure 10) for comparison purposes.

The data block in JAGS defines the observed data, including the predictor variables and the target variable (Potability). It includes information on sample size, data vectors for each predictor, and the response vector for potability. This block ensures that the data are correctly structured for analysis in JAGS, facilitating a smooth model-fitting process.

To identify the optimal MCMC settings for efficiency, accuracy, and representativeness of the posterior samples, different configurations of the MCMC parameters were tested. These settings include the number of adaptation steps, burn-in steps, chains, thinning steps, saved steps, and total iterations. The table below summarizes the different MCMC settings tested, along with their respective runtimes:

Adapt steps	**Burn-in**	**No of chains**	**Thinning steps**	**No of saved steps**	**No of iterations**	**Run time**
100	500	3	2	1000	**2000**	333.6
500	1000	4	3	1000	**3000**	658.29
500	1000	4	5	1000	**5000**	1036.46
500	1000	4	10	1000	**10000**	1773.17

Table 3: MCMC Settings used for this analysis

These configurations were assessed to identify the optimal balance between computational efficiency and the representativeness of the posterior samples. Convergence diagnostics, such as trace plots, shrink factor, autocorrelation plots and density plots were used to evaluate the performance of each setting, and the most appropriate configuration was selected based on a combination of runtime, effective sample size, and diagnostic results shown in Figure 12-15.

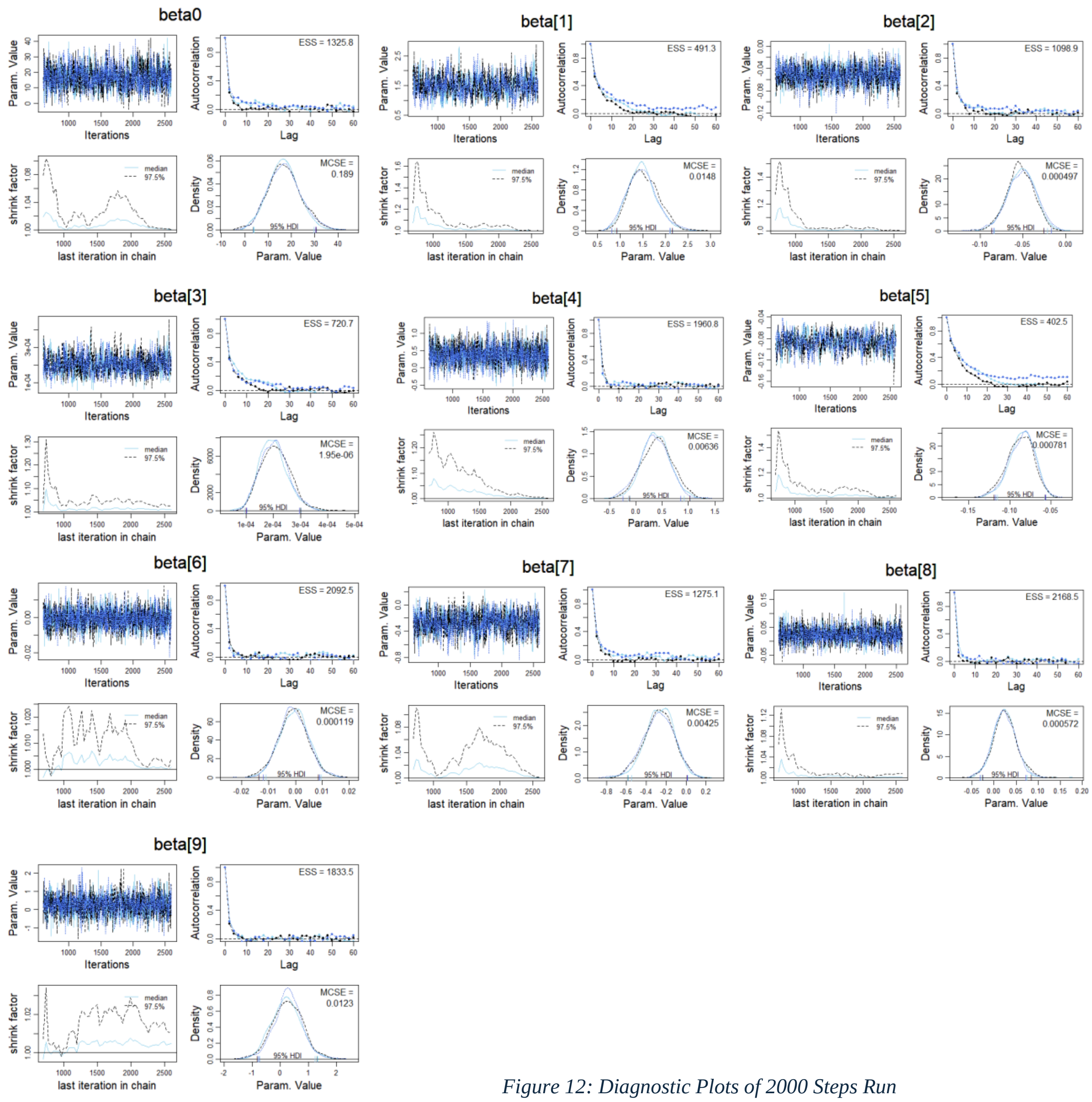


Figure 12: Diagnostic Plots of 2000 Steps Run

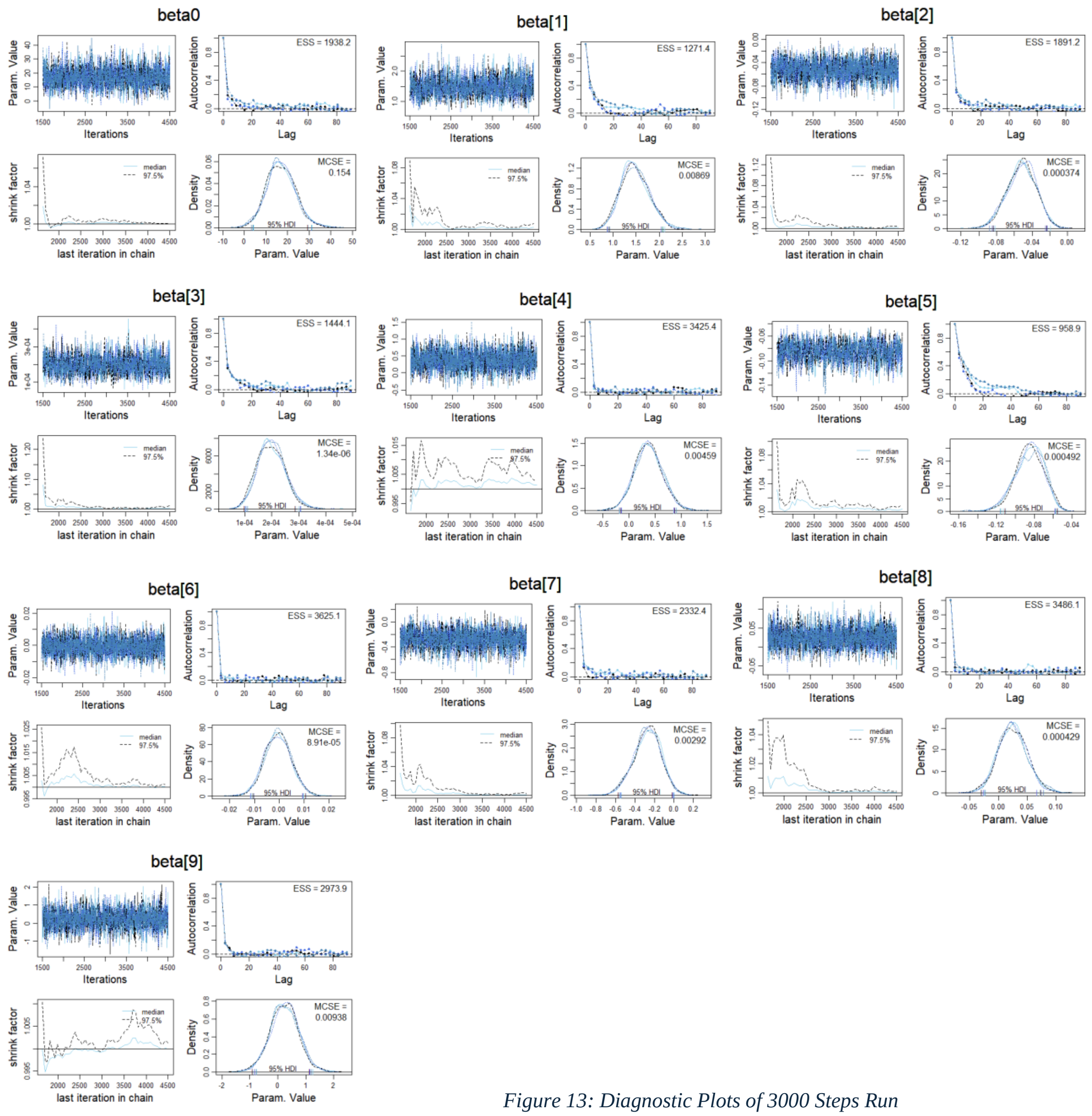


Figure 13: Diagnostic Plots of 3000 Steps Run

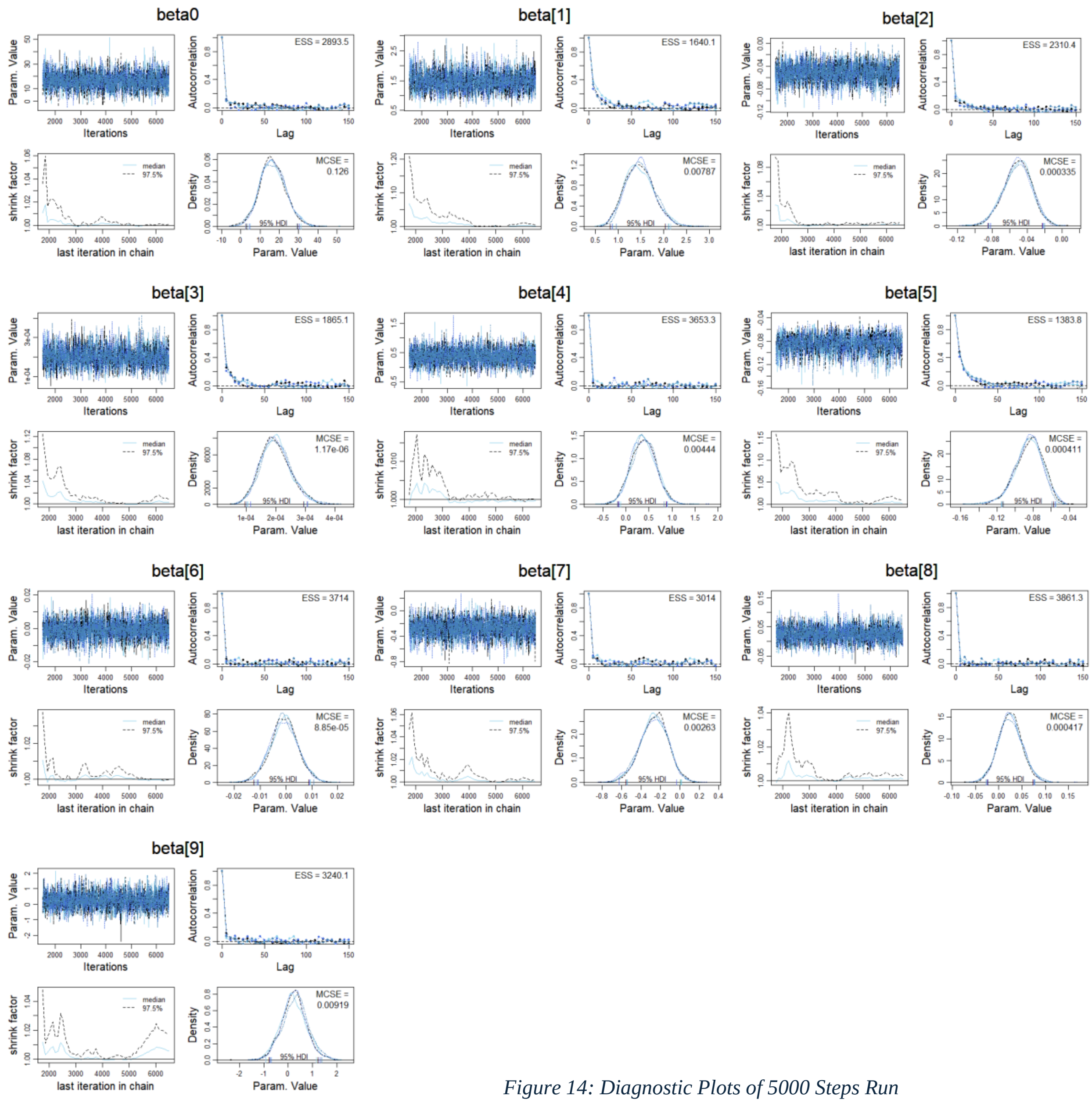


Figure 14: Diagnostic Plots of 5000 Steps Run

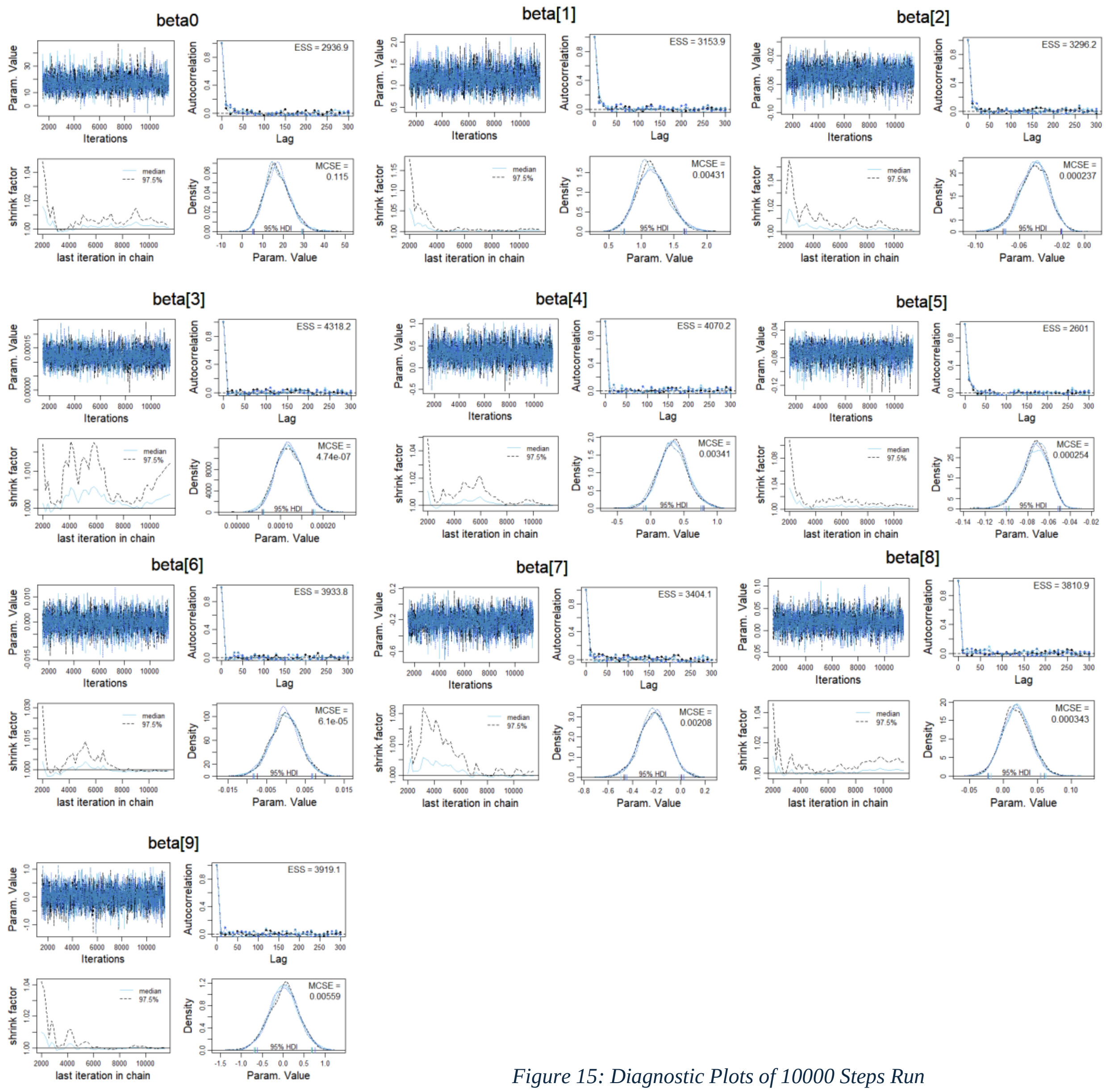


Figure 15: Diagnostic Plots of 10000 Steps Run

From Figure 12, the diagnostics for 2000 iterations show the following observations:

- For most coefficients, the shrink factor exceeded 1.2, indicating potential convergence issues. A shrink factor above 1.2 suggests that the chains may not have fully converged to the posterior distribution.
- The trace plots for coefficients beta5 and beta8 showed discernible patterns instead of the desired randomness, indicating non-convergence or insufficient exploration of the parameter space.
- Significant autocorrelation was observed across most coefficients, suggesting that the samples were highly correlated and not sufficiently independent.
- Despite these issues, the density plots appeared to have converged, showing smooth unimodal distributions for each coefficient. However, due to issues with trace plots and autocorrelation, this setting was not sufficient for accurate parameter estimation.

For the next implementation, increasing adaptation steps can improve chain mixing, but it also increases runtime. Starting with an initial 500 steps, we increased the burn-in period to 1000 steps to eliminate the effect of initial parameter settings and ensure that chains have reached a stationary distribution. Running more chains also helps detect any potential issues with convergence, as all chains should ideally converge to the same posterior distribution. Thinning intervals of 3 were tested for the second implementation. Increasing the thinning interval reduces autocorrelation but increases runtime since more iterations are needed to reach the desired number of saved samples.

Figure 13 displays the diagnostics for 3000 iterations showing the following observations:

- The shrink factor improved, falling below 1.2 for most coefficients. However, for beta3, the shrink factor initially started above 1.2, indicating partial convergence.
- Autocorrelation remained an issue, indicating that the chains still contained dependent samples, which affects the quality of the posterior estimates.
- The trace plots showed some improvement in randomness, although not to the extent seen in higher iteration settings.

For the third implementation, the thinning steps were further increased to 5 to reduce the autocorrelation. The diagnostics from Figure 14 shows the following improvements:

- The shrink factor was no longer an issue, with values below 1.2 across all coefficients, indicating better convergence compared to the previous settings.
- The trace plots showed a desirable level of randomness, suggesting that the chains were beginning to explore the parameter space more effectively.
- Although the autocorrelation issue persisted, it had improved compared to the 2000 and 3000 iteration settings. This suggests that the chains were closer to producing independent samples but had not fully achieved it.

On the final implementation, the thinning steps were further increased to 10 to completely get rid of the autocorrelation issue. From Figure 15, the observations from the final implementation are as follows:

- The shrink factor was consistently below 1.2 for all coefficients, indicating strong convergence across all chains.
- The trace plots showed excellent randomness and mixing for all coefficients, with no discernible patterns, indicating effective exploration of the posterior distribution.
- Autocorrelation was no longer a concern at this setting, as the thinning steps and higher number of iterations effectively reduced autocorrelation, producing near-independent samples.

- The density plots showed smooth, unimodal distributions for all coefficients, with all chains aligning well, confirming that the samples accurately represent the posterior distribution.
- The 10,000-iteration setting produced a high ESS for all coefficients, indicating that the samples were effectively independent and sufficiently numerous to capture the posterior distribution accurately. In contrast, lower iteration settings resulted in lower ESS values, especially in the 2000-iteration and 3000-iteration runs, where significant autocorrelation reduced the effective sample size.
- The MCSE values were lowest in the 10,000-iteration setting, indicating precise parameter estimates with minimal variability due to sampling error. Higher MCSE values were observed in the lower iteration settings, further highlighting the benefit of longer chains for stable and reliable estimates.

Given these results, the 10,000-iteration setting not only showed the best diagnostics visually but also provided superior effective sample sizes and lower Monte Carlo errors, ensuring the robustness of the posterior estimates. Thus, despite the increased runtime, the 10,000-iteration setting was selected as the final configuration for this analysis.

3.5 MODEL POSTERIOR DISTRIBUTIONS

The relationship between the predictor variables and water potability (the outcome) in this Bayesian Logistic Regression model is expressed through **log odds**. Log odds represent the natural logarithm of the odds of the outcome being 1 (potable water) versus 0 (non-potable water). Each regression coefficient quantifies the impact of a one-unit increase in the corresponding predictor variable on the log odds of water being potable. For each predictor variable, the **odds ratio** can be calculated as e^{β}. This value represents the multiplicative change in the odds of the outcome (water being potable) for a one-unit increase in the predictor variable. An odds ratio greater than 1 indicates that the predictor increases the odds of potability, while an odds ratio less than 1 implies a decrease in these odds.

The following are the posterior distributions for each regression coefficient, separately for the models with non-informative and informative priors, as shown in Figures 16 and 17.

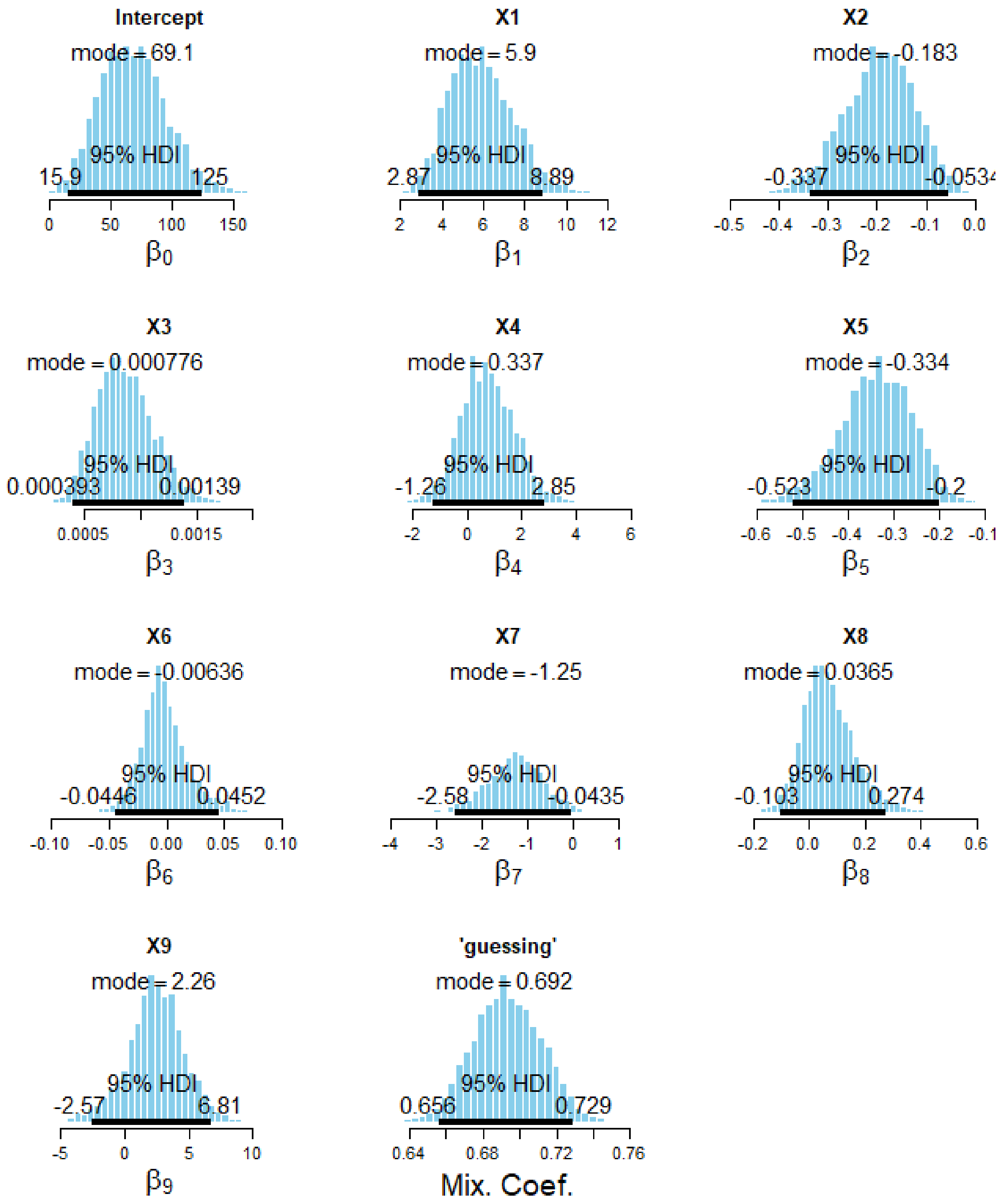


Figure 16: Posterior Distributions for Non-Informative Model

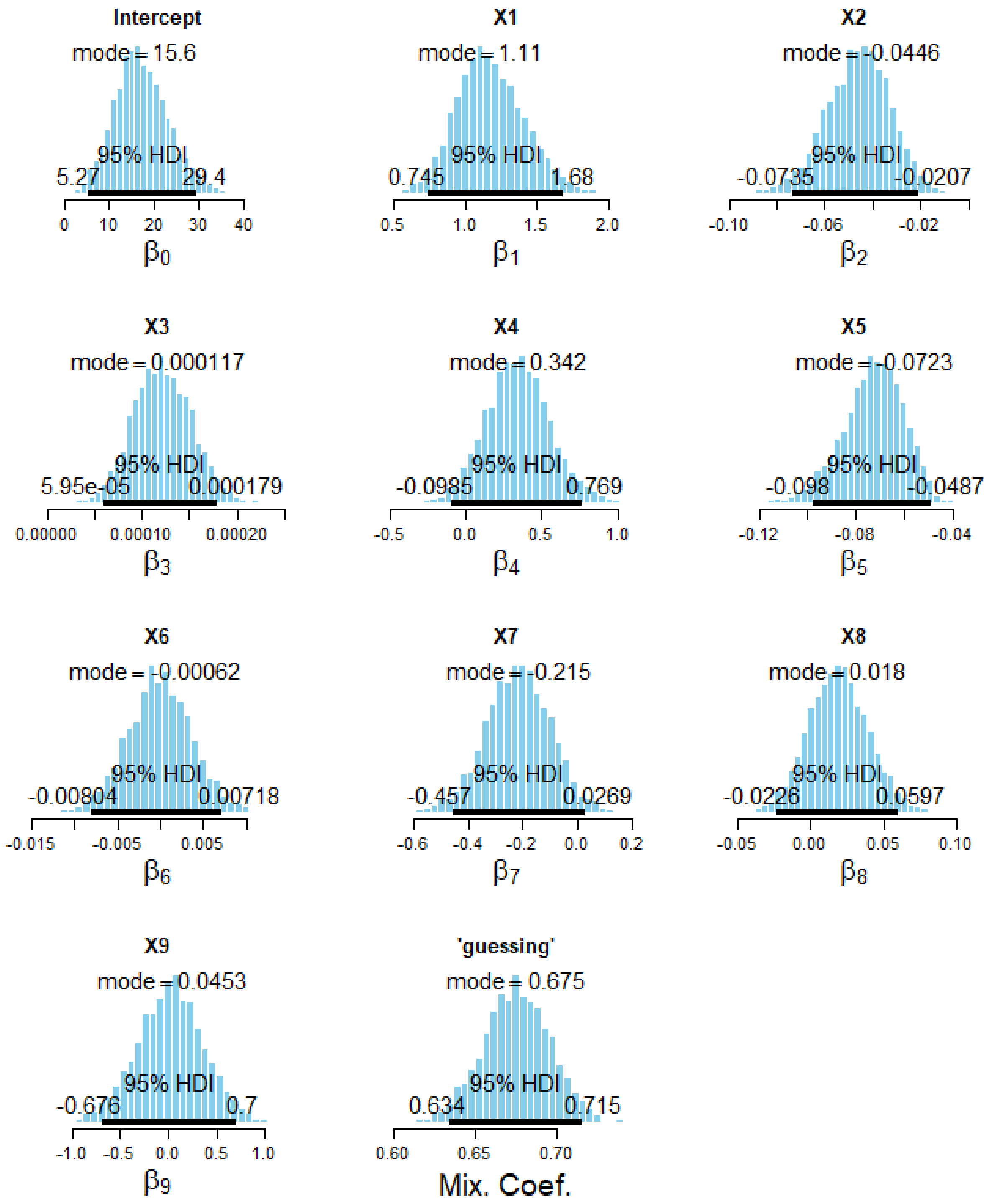
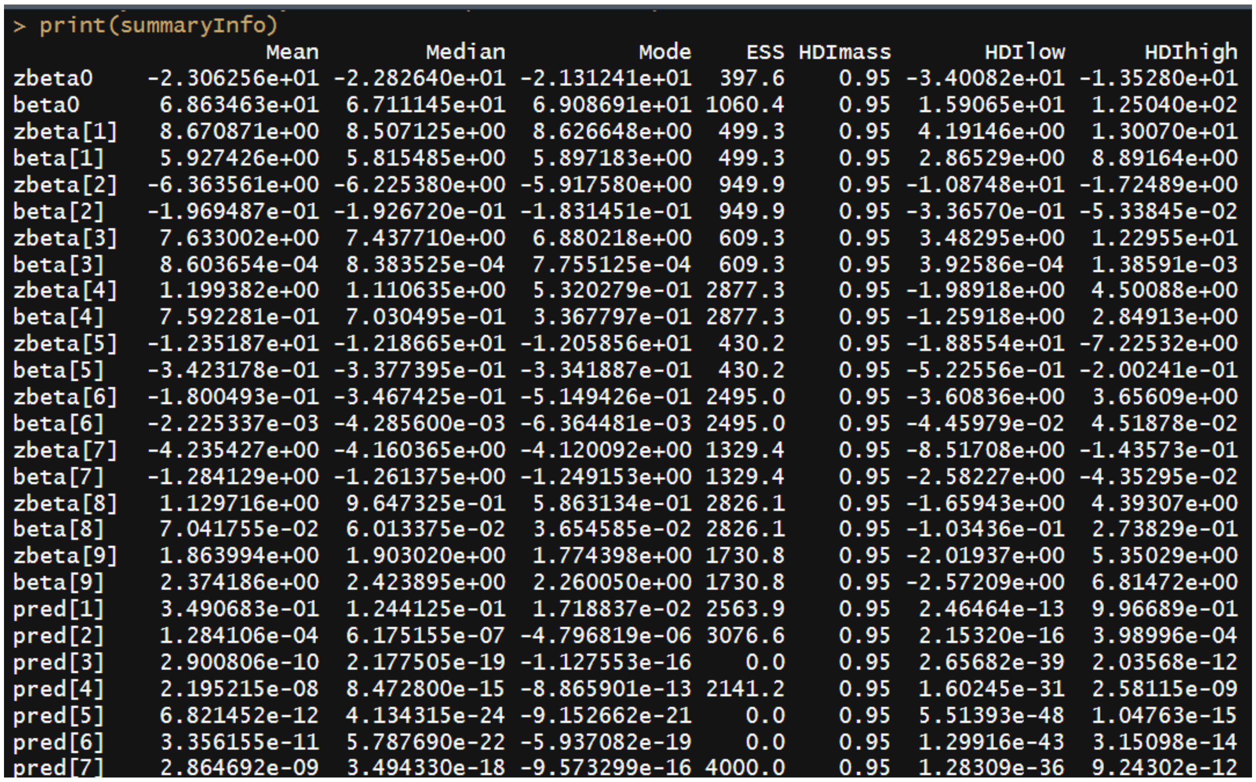
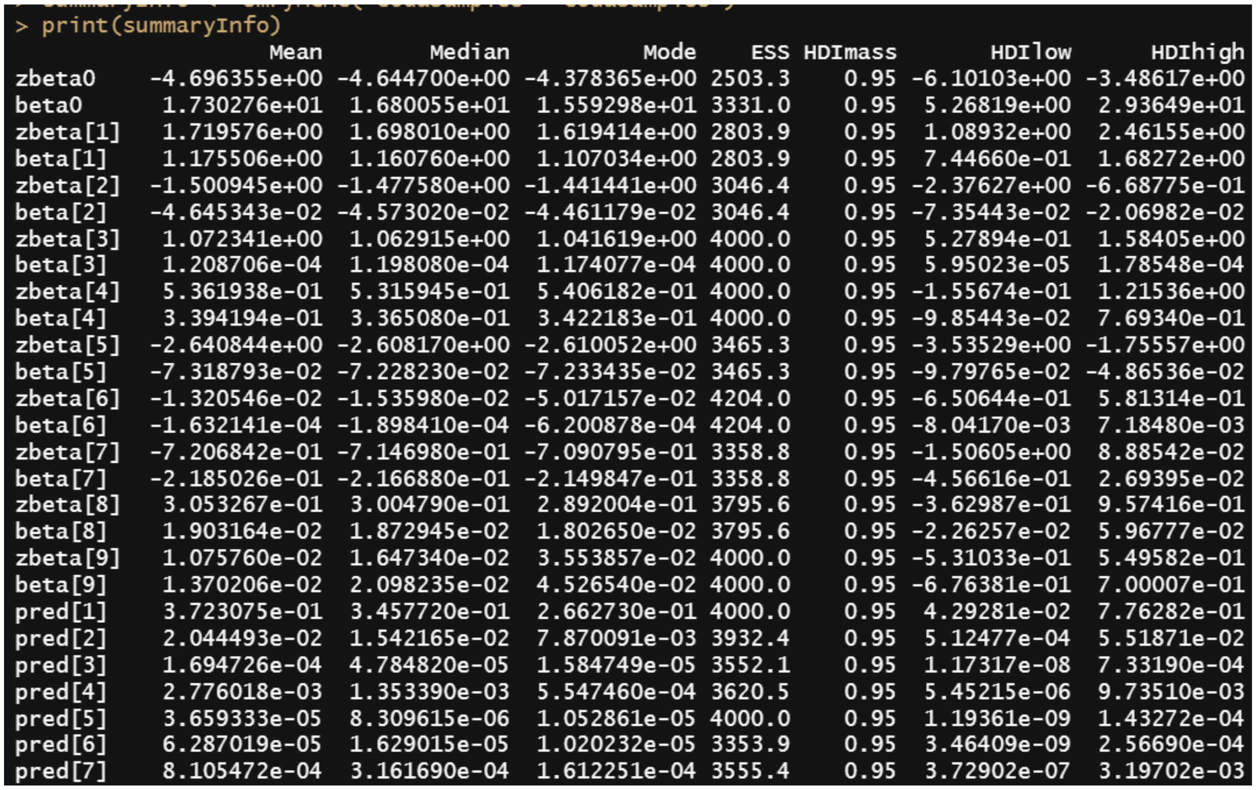


Figure 17: Posterior Distributions for Informative Prior Model



```
> print(summaryInfo)
                  Mean        Median          Mode    ESS HDImass      HDIlow     HDIhigh
zbeta0   -2.306256e+01 -2.282640e+01 -2.131241e+01  397.6    0.95 -3.40082e+01 -1.35280e+01
beta0     6.863463e+01  6.711145e+01  6.908691e+01 1060.4    0.95  1.59065e+01  1.25040e+02
zbeta[1]  8.670871e+00  8.507125e+00  8.626648e+00  499.3    0.95  4.19146e+00  1.30070e+01
beta[1]   5.927426e+00  5.815485e+00  5.897183e+00  499.3    0.95  2.86529e+00  8.89164e+00
zbeta[2] -6.363561e+00 -6.225380e+00 -5.917580e+00  949.9    0.95 -1.08748e+01 -1.72489e+00
beta[2]  -1.969487e-01 -1.926720e-01 -1.831451e-01  949.9    0.95 -3.36570e-01 -5.33845e-02
zbeta[3]  7.633002e+00  7.437710e+00  6.880218e+00  609.3    0.95  3.48295e+00  1.22955e+01
beta[3]   8.603654e-04  8.383525e-04  7.755125e-04  609.3    0.95  3.92586e-04  1.38591e-03
zbeta[4]  1.199382e+00  1.110635e+00  5.320279e-01 2877.3    0.95 -1.98918e+00  4.50088e+00
beta[4]   7.592281e-01  7.030495e-01  3.367797e-01 2877.3    0.95 -1.25918e+00  2.84913e+00
zbeta[5] -1.235187e+01 -1.218665e+01 -1.205856e+01  430.2    0.95 -1.88554e+01 -7.22532e+00
beta[5]  -3.423178e-01 -3.377395e-01 -3.341887e-01  430.2    0.95 -5.22556e-01 -2.00241e-01
zbeta[6] -1.800493e-01 -3.467425e-01 -5.149426e-01 2495.0    0.95 -3.60836e+00  3.65609e+00
beta[6]  -2.225337e-03 -4.285600e-03 -6.364481e-03 2495.0    0.95 -4.45979e-02  4.51878e-02
zbeta[7] -4.235427e+00 -4.160365e+00 -4.120092e+00 1329.4    0.95 -8.51708e+00 -1.43573e-01
beta[7]  -1.284129e+00 -1.261375e+00 -1.249153e+00 1329.4    0.95 -2.58227e+00 -4.35295e-02
zbeta[8]  1.129716e+00  9.647325e-01  5.863134e-01 2826.1    0.95 -1.65943e+00  4.39307e+00
beta[8]   7.041755e-02  6.013375e-02  3.654585e-02 2826.1    0.95 -1.03436e-01  2.73829e-01
zbeta[9]  1.863994e+00  1.903020e+00  1.774398e+00 1730.8    0.95 -2.01937e+00  5.35029e+00
beta[9]   2.374186e+00  2.423895e+00  2.260050e+00 1730.8    0.95 -2.57209e+00  6.81472e+00
pred[1]   3.490683e-01  1.244125e-01  1.718837e-02 2563.9    0.95  2.46464e-13  9.96689e-01
pred[2]   1.284106e-04  6.175155e-07 -4.796819e-06 3076.6    0.95  2.15320e-16  3.98996e-04
pred[3]   2.900806e-10  2.177505e-19 -1.127553e-16    0.0    0.95  2.65682e-39  2.03568e-12
pred[4]   2.195215e-08  8.472800e-15 -8.865901e-13 2141.2    0.95  1.60245e-31  2.58115e-09
pred[5]   6.821452e-12  4.134315e-24 -9.152662e-21    0.0    0.95  5.51393e-48  1.04763e-15
pred[6]   3.356155e-11  5.787690e-22 -5.937082e-19    0.0    0.95  1.29916e-43  3.15098e-14
pred[7]   2.864692e-09  3.494330e-18 -9.573299e-16 4000.0    0.95  1.28309e-36  9.24302e-12
```

Figure 128: Summary Statistics of the Posterior Estimates for Non-Informative Model



```
> print(summaryInfo)
                  Mean        Median          Mode    ESS HDImass      HDIlow     HDIhigh
zbeta0   -4.696355e+00 -4.644700e+00 -4.378365e+00 2503.3    0.95 -6.10103e+00 -3.48617e+00
beta0     1.730276e+01  1.680055e+01  1.559298e+01 3331.0    0.95  5.26819e+00  2.93649e+01
zbeta[1]  1.719576e+00  1.698010e+00  1.619414e+00 2803.9    0.95  1.08932e+00  2.46155e+00
beta[1]   1.175506e+00  1.160760e+00  1.107034e+00 2803.9    0.95  7.44660e-01  1.68272e+00
zbeta[2] -1.500945e+00 -1.477580e+00 -1.441441e+00 3046.4    0.95 -2.37627e+00 -6.68775e-01
beta[2]  -4.645343e-02 -4.573020e-02 -4.461179e-02 3046.4    0.95 -7.35443e-02 -2.06982e-02
zbeta[3]  1.072341e+00  1.062915e+00  1.041619e+00 4000.0    0.95  5.27894e-01  1.58405e+00
beta[3]   1.208706e-04  1.198080e-04  1.174077e-04 4000.0    0.95  5.95023e-05  1.78548e-04
zbeta[4]  5.361938e-01  5.315945e-01  5.406182e-01 4000.0    0.95 -1.55674e-01  1.21536e+00
beta[4]   3.394194e-01  3.365080e-01  3.422183e-01 4000.0    0.95 -9.85443e-02  7.69340e-01
zbeta[5] -2.640844e+00 -2.608170e+00 -2.610052e+00 3465.3    0.95 -3.53529e+00 -1.75557e+00
beta[5]  -7.318793e-02 -7.228230e-02 -7.233435e-02 3465.3    0.95 -9.79765e-02 -4.86536e-02
zbeta[6] -1.320546e-02 -1.535980e-02 -5.017157e-02 4204.0    0.95 -6.50644e-01  5.81314e-01
beta[6]  -1.632141e-04 -1.898410e-04 -6.200878e-04 4204.0    0.95 -8.04170e-03  7.18480e-03
zbeta[7] -7.206842e-01 -7.146980e-01 -7.090795e-01 3358.8    0.95 -1.50605e+00  8.88542e-02
beta[7]  -2.185026e-01 -2.166880e-01 -2.149847e-01 3358.8    0.95 -4.56616e-01  2.69395e-02
zbeta[8]  3.053267e-01  3.004790e-01  2.892004e-01 3795.6    0.95 -3.62987e-01  9.57416e-01
beta[8]   1.903164e-02  1.872945e-02  1.802650e-02 3795.6    0.95 -2.26257e-02  5.96777e-02
zbeta[9]  1.075760e-02  1.647340e-02  3.553857e-02 4000.0    0.95 -5.31033e-01  5.49582e-01
beta[9]   1.370206e-02  2.098235e-02  4.526540e-02 4000.0    0.95 -6.76381e-01  7.00007e-01
pred[1]   3.723075e-01  3.457720e-01  2.662730e-01 4000.0    0.95  4.29281e-02  7.76282e-01
pred[2]   2.044493e-02  1.542165e-02  7.870091e-03 3932.4    0.95  5.12477e-04  5.51871e-02
pred[3]   1.694726e-04  4.784820e-05  1.584749e-05 3552.1    0.95  1.17317e-08  7.33190e-04
pred[4]   2.776018e-03  1.353390e-03  5.547460e-04 3620.5    0.95  5.45215e-06  9.73510e-03
pred[5]   3.659333e-05  8.309615e-06  1.052861e-05 4000.0    0.95  1.19361e-09  1.43272e-04
pred[6]   6.287019e-05  1.629015e-05  1.020232e-05 3353.9    0.95  3.46409e-09  2.56690e-04
pred[7]   8.105472e-04  3.161690e-04  1.612251e-04 3555.4    0.95  3.72902e-07  3.19702e-03
```

Figure 139: Summary Statistics of the Posterior Estimates for Informative Model

3.5.1 Non-Informative Prior Model (Figure 16 and 18)

With the non-informative priors, the model relies primarily on the data to estimate parameter values, resulting in broader posterior distributions.

Predictor Variable	Posterior Mode	Log-Odds
X1 – pH	5.9	365.03
X2 – Hardness	-0.183	0.83
X3 – Solids	0.000776	1.0007
X4 – Chloramines	0.337	1.4
X5 – Sulfate	-0.334	0.716
X6 – Conductivity	-0.00636	0.993
X7 – Organic Carbon	-1.25	0.286
X8 – Trihalomethanes	0.0365	1.037
X9 - Turbidity	2.26	9.583

Table 4: Log-Odds table for Non-Informative Model

- Predictors like *pH* and *Chloramines* exhibit modes of 5.9 and 0.337, respectively, with 95% Highest Density Intervals (HDI) that do not include zero. This suggests a strong positive effect on potability. The odds ratio for *X1*, for instance, is approximately $e^{5.9} \approx 365$ approx., which implies a substantial increase in the odds of potability for a one-unit increase in *X1*. Similarly, *X4* has an odds ratio of $e^{0.337} \approx 1.4$, indicating a 40% increase in potability odds with a one-unit increase in *X4*.
- The modes of *Solids* and *Conductivity* are very close to zero, and the 95% HDI of *Turbidity* include zero even though posterior estimate is higher, suggesting minimal or no impact on potability.
- The mode of the "guessing" coefficient is around 0.692, indicating a small probability of random guessing, which accounts for potential outliers. This adjustment helps improve the model's robustness against unusual data points.

3.5.2 Informative Prior Model (Figure 17 and 19)

The informative prior model incorporates prior beliefs about each predictor's influence, resulting in narrower posterior distributions and slightly more concentrated estimates.

Predictor Variable	Posterior Mode	Log-Odds
X1 – pH	1.11	3.03
X2 – Hardness	-0.0446	0.956
X3 – Solids	0.000117	1.0001
X4 – Chloramines	0.342	1.407
X5 – Sulfate	-0.0723	0.9302
X6 – Conductivity	-0.00062	0.99
X7 – Organic Carbon	-0.0215	0.978
X8 – Trihalomethanes	0.018	1.018
X9 - Turbidity	0.0453	1.04

Table 5: Log-Odds Table for Informative Model

- *Chloramines* maintain similar modes to the non-informative model (0.342, respectively), with 95% HDIs that exclude zero. *pH* showed substantial decrease in the posterior estimate with the 95% HDI being narrower than in the non-informative model. The odds ratios of them in this model are approximately $e^{1.11} \approx 3.03$ approx., and $e^{0.342} \approx 1.41$, indicating that *pH* triples the odds of potability while *Chloramines* increases the odds by about 41%. This consistency across models suggests that these predictors have a stable, positive influence on potability.

- Like the non-informative model, predictors *Solids* and *Conductivity* show minimal or no impact on potability as its estimates are very close to zero. The HDI of *Turbidity* now evidently shows that it includes zero at the peak compared to non-informative model. This suggests these predictors do not contribute meaningfully to the model and could be excluded.
- The mode of the guessing coefficient is 0.675 in the informative model, slightly lower than in the non-informative model. This indicates that, even with informative priors, the model maintains a minor allowance for random guesses due to outliers, thus enhancing robustness.

The posterior distributions for the non-informative and informative models show notable differences in their estimates, especially for significant predictors like *X1*. The larger uncertainty in the non-informative model (due to a broader prior) leads to more extreme estimates, while the informative model, guided by prior knowledge, produces more moderate estimates. This comparison demonstrates that in the informative model, prior beliefs have a stabilizing effect on the estimates, suggesting that the non-informative model's broad prior assumptions may introduce excessive variability in the absence of domain knowledge.

Given these differences, the informative model may be preferable in cases where domain knowledge is available, as it provides more stable estimates that align with expected relationships, while the non-informative model may overstate effects due to the lack of constraints.

3.6 Model Accuracy Check

To evaluate the performance of each Bayesian Logistic Regression model, we used key classification metrics, including accuracy, precision, recall, and F-score, along with the confusion matrix. These metrics provide a comprehensive view of each model's ability to classify water samples as potable or non-potable. Below is a comparison of the performance across three different model configurations: the non-informative model, the informative model, and the informative model with insignificant variables removed from the informative model.

MODEL	ACCURACY	PRECISION	RECALL	F-SCORE	RUNTIME
Non-Informative	0.622	0.624	0.926	0.746	1773.17
Informative	0.627	0.627	0.928	0.748	1520.99
Reduced	0.613	0.619	0.920	0.740	1163.49

Table 6: Prediction Test Results on the models under comparison

Comparing these results, the slight performance differences between the non-informative and informative models show that prior knowledge has substantial impact on the model’s predictions, as there was a significant impact in the coefficients posterior estimates. Additionally, the reduced informative model performs nearly as well as the full model, confirming that *X3*, *X6*, and *X9* contribute minimally to the model's predictive ability. Therefore, the reduced model can be considered an efficient choice, offering a simpler structure without sacrificing accuracy.

3.7 Prior Sensitivity Check

To examine the sensitivity of the posterior distributions to the priors, I conducted a prior sensitivity analysis by comparing two models: the original informative prior model and an alternative model with moderately vague priors. The posterior distributions for the alternative model with moderately vague priors are shown in Figure 20.

The prior sensitivity analysis demonstrates that while changing the priors affects the posterior distributions, the impact varies across predictors. Strong predictors such as *X1* and *X4* show slight shifts toward higher modes under moderately vague priors, indicating that their posterior estimates are somewhat sensitive to prior specifications. However, for predictors with minimal impact (e.g., *X3*, *X6*, and *X9*), the posterior distributions remain largely unchanged, suggesting that the data likelihood dominates the estimates for these coefficients. Overall, this analysis highlights that the choice of priors can influence the strength of association for certain predictors but has limited impact on those with weak or negligible contributions. This insight underscores the importance of carefully selecting priors, especially when the goal is to capture subtle but meaningful effects in Bayesian models.

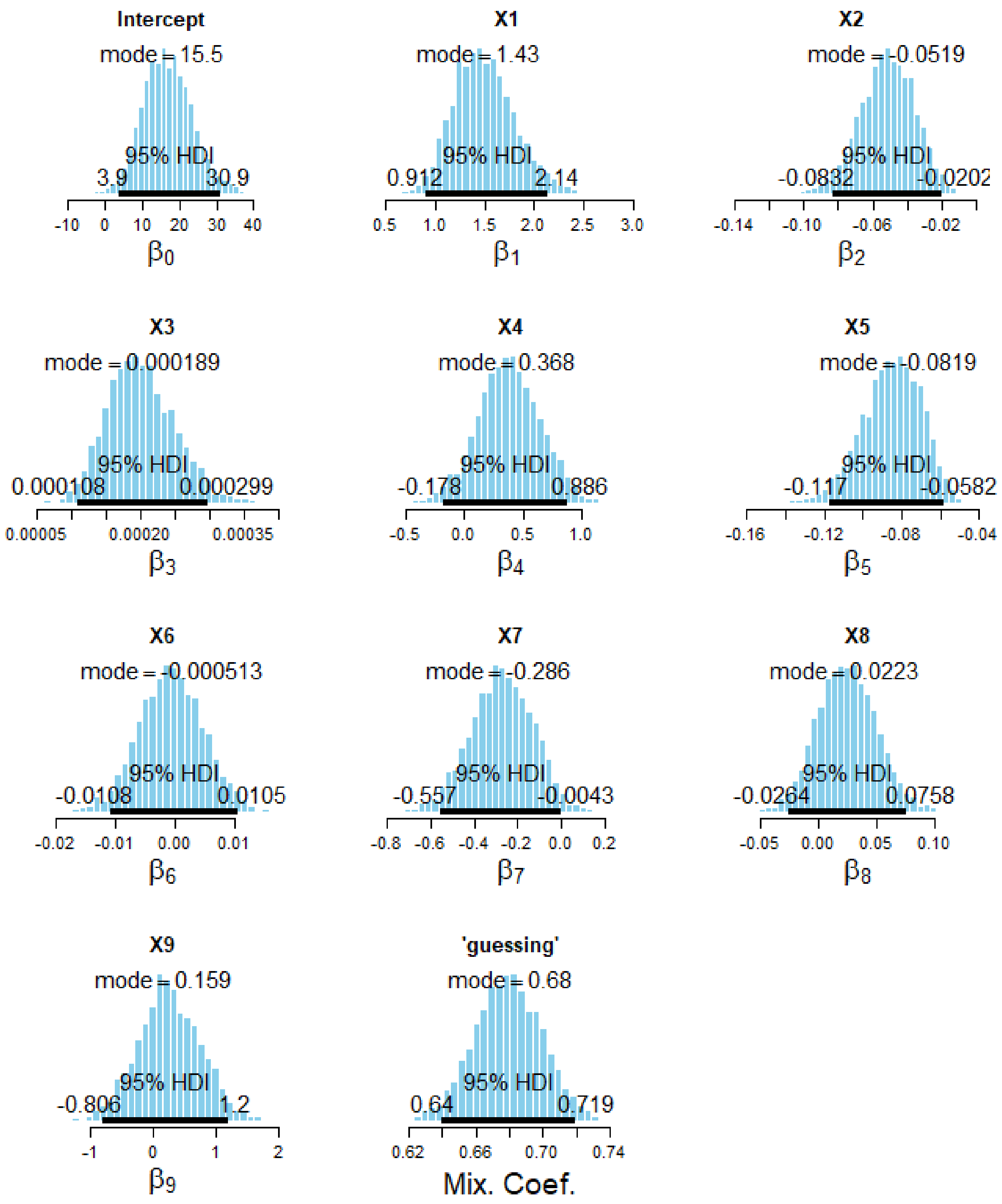


Figure 20: Posterior Distributions for Informative Prior Model with uniform moderately vague priors

4 CONCLUSION

In conclusion, this Bayesian Logistic Regression analysis successfully addressed the research question of whether water potability can be effectively classified using physicochemical properties. By choosing Bayesian methods, this study offered a flexible and robust framework that can incorporate domain knowledge and handle uncertainty in water quality classification.

Throughout the analysis, careful attention was given to selecting the optimal MCMC settings. After experimenting with different adaptation, burn-in, and thinning steps, the final configuration achieved a balance between computational efficiency and diagnostic quality, as demonstrated by low autocorrelation, well-mixed trace plots, and high effective sample sizes. These settings ensured that the posterior estimates were robust and representative of the true distribution.

The inclusion of informative priors, based on water quality knowledge, played a significant role in refining the model. Informative priors narrowed the Highest Density Intervals (HDI) of the posterior estimates, resulting in more concentrated and stable parameter estimates. This was particularly beneficial for important predictors such as pH and Chloramines, where prior knowledge allowed the model to make more precise predictions. The informative prior model also offered an advantage in terms of interpretability and consistency in predictions, as it aligned well with expected relationships in water quality.

The analysis also identified several insignificant predictors—Solids, Conductivity, and Turbidity—which contributed minimally to model performance. By removing these predictors, the reduced informative model achieved nearly identical accuracy metrics compared to the full model. This reduction in complexity made the model more efficient without compromising predictive accuracy, as shown in the similar accuracy, precision, recall, and F-score metrics across models. These results suggest that the reduced model is an effective and practical choice for water potability classification.

Additionally, a prior sensitivity analysis was conducted, testing moderately vague priors within the informative model. The analysis revealed that while the choice of priors can slightly influence posterior distributions for strong predictors, it has limited effect on predictors with negligible contributions. This finding reinforces the importance of selecting priors that reflect meaningful domain knowledge to enhance model robustness without introducing unnecessary variability.

In summary, this Bayesian analysis confirms that Bayesian Logistic Regression, with carefully chosen MCMC settings and informative priors, is a powerful tool for water quality classification. The reduced model efficiently addressed the problem statement, providing a reliable classification framework that leverages domain knowledge while maintaining simplicity. This approach, integrating statistical rigor with expert knowledge, holds potential for broader applications in environmental monitoring and public health. Future work could extend this framework to real-time monitoring systems and other datasets, further enhancing its utility in water resource management.

5 REFERENCES

1. World Health Organization, *Guidelines for Drinking-water Quality*, 4th ed. Geneva: WHO Press, 2011. Available: https://www.who.int/publications/i/item/9789241548151

2. World Health Organization, *Hardness in Drinking Water*, 2003.

3. United States Environmental Protection Agency, *Secondary Drinking Water Standards: Guidance for Nuisance Chemicals*, EPA-816-F-18-002, 2018. Available: https://www.epa.gov/sdwa/secondary-drinking-water-standards-guidance-nuisance-chemicals

4. United States Environmental Protection Agency, *National Primary Drinking Water Regulations*, EPA-816-F-09-004, 2009. Available: https://www.epa.gov/dwstandardsregulations

5. United States Environmental Protection Agency, *Basic Information about Chloramines in Drinking Water*, 2020. Available: https://www.epa.gov/dwreginfo/chloramines-drinking-water

6. Health Canada, *Guidelines for Canadian Drinking Water Quality: Sulfate*, 1994.

7. National Research Council, *Safe Drinking Water: A Committee Report to the United States Environmental Protection Agency*. Washington, DC: National Academy of Sciences, 1977.

8. United States Environmental Protection Agency, *Total Organic Carbon (TOC) in Drinking Water*, Available: https://www.epa.gov/sites/default/files/2015-09/documents/toctech.pdf

9. United States Environmental Protection Agency, *National Primary Drinking Water Regulations: Stage 1 and Stage 2 Disinfectants and Disinfection Byproducts Rules*. Available: https://www.epa.gov/dwreginfo/stage-1-and-stage-2-disinfectants-and-disinfection-byproducts-rules

10. United States Environmental Protection Agency, *National Primary Drinking Water Regulations: Surface Water Treatment Rule*. Available: https://www.epa.gov/dwstandardsregulations/surface-water-treatment-rules

6 APPENDIX

```
graphics.off() # This closes all of R's graphics windows.
rm(list=ls())  # Careful! This clears all of R's memory!
setwd("C:/Users/Asus/Desktop/Bayesian Stats/Assignment 3/Assignment - 3")
source("DBDA2E-utilities.R")
library(dplyr)
library(readr)
library(ggplot2)
library(ggpubr)
library(GGally)
library(corrplot)
library(ks)
library(rjags)
library(runjags)
library(benchmarkme)
library(caret)

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

smryMCMC = function(  codaSamples ,
                      saveName=NULL ) {
  summaryInfo = NULL
  mcmcMat = as.matrix(codaSamples)
  paramName = colnames(mcmcMat)
  for ( pName in paramName ) {
    summaryInfo = rbind( summaryInfo , summarizePost( mcmcMat[,pName] ) )
  }
  rownames(summaryInfo) = paramName
  if ( !is.null(saveName) ) {
    write.csv( summaryInfo , file=paste(saveName,"SummaryInfo.csv",sep="") )
  }
  return( summaryInfo )
}

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

plotMCMC = function( codaSamples , data , xName="x" , yName="y" ,
                     showCurve=FALSE ,  pairsPlot=FALSE ,
                     saveName=NULL , saveType="jpg" ) {
  # showCurve is TRUE or FALSE and indicates whether the posterior should
  #   be displayed as a histogram (by default) or by an approximate curve.
  # pairsPlot is TRUE or FALSE and indicates whether scatterplots of pairs
  #   of parameters should be displayed.
  #-----------------------------------------------------------------------------
-
```

```
  y = data[,yName]
  x = as.matrix(data[,xName])
  mcmcMat = as.matrix(codaSamples,chains=TRUE)
  chainLength = NROW( mcmcMat )
  guess = mcmcMat[,"guess"]
  zbeta0 = mcmcMat[,"zbeta0"]
  zbeta  = mcmcMat[,grep("^zbeta$|^zbeta\\[",colnames(mcmcMat))]
  if ( ncol(x)==1 ) { zbeta = matrix( zbeta , ncol=1 ) }
  beta0 = mcmcMat[,"beta0"]
  beta  = mcmcMat[,grep("^beta$|^beta\\[",colnames(mcmcMat))]
  if ( ncol(x)==1 ) { beta = matrix( beta , 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=1.5 ) # was cex=cex.cor*r
    }
    pairs( cbind( beta0 , beta , guess )[plotIdx,] ,
           labels=c( "beta[0]" ,
                     paste0("beta[",1:ncol(beta),"]\n",xName) , "guessing" ) ,
           lower.panel=panel.cor , col="skyblue" )
    if ( !is.null(saveName) ) {
      saveGraph( file=paste(saveName,"PostPairs",sep=""), type=saveType)
    }
  }
  #-----------------------------------------------------------------------------
-
  # Data with posterior predictive:
  # If only 1 predictor:
  if ( ncol(x)==1 ) {
    openGraph(width=7,height=6)
    par( mar=c(3.5,3.5,2,1) , mgp=c(2.0,0.7,0) )
    plot( x[,1] , y , xlab=xName[1] , ylab=yName ,
          cex=2.0 , cex.lab=1.5 , col="black" , main="Data with Post. Pred." )
    abline(h=0.5,lty="dotted")
    cVec = floor(seq(1,chainLength,length=30))
    xWid=max(x)-min(x)
    xComb = seq(min(x)-0.1*xWid,max(x)+0.1*xWid,length=201)
    for ( cIdx in cVec ) {
      lines( xComb ,
             guess[cIdx]*0.5
             + (1.0-guess[cIdx])*1/(1+exp(-(beta0[cIdx]+beta[cIdx,1]*xComb )))
,
             lwd=1.5 ,
             col="skyblue" )
      xInt = -beta0[cIdx]/beta[cIdx,1]
      arrows( xInt,0.5, xInt,-0.04, length=0.1 , col="skyblue" , lty="dashed" )
    }
    if ( !is.null(saveName) ) {
      saveGraph( file=paste(saveName,"DataThresh",sep=""), type=saveType)
    }
  }
  # If only 2 predictors:
  if ( ncol(x)==2 ) {
    openGraph(width=7,height=7)
    par( mar=c(3.5,3.5,2,1) , mgp=c(2.0,0.7,0) )
    plot( x[,1] , x[,2] , pch=as.character(y) , xlab=xName[1] , ylab=xName[2] ,
          col="black" , main="Data with Post. Pred.")
    cVec = floor(seq(1,chainLength,length=30))
    for ( cIdx in cVec ) {
      abline( -beta0[cIdx]/beta[cIdx,2] , -beta[cIdx,1]/beta[cIdx,2] ,
col="skyblue" )
    }
    if ( !is.null(saveName) ) {
      saveGraph( file=paste(saveName,"DataThresh",sep=""), type=saveType)
    }
  }
  #-----------------------------------------------------------------------------
-
  # Marginal histograms:

  decideOpenGraph = function( panelCount , saveName , finished=FALSE ,
                              nRow=1 , nCol=3 ) {
    # 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" )
  for ( bIdx in 1:ncol(beta) ) {
    panelCount = decideOpenGraph( panelCount ,
saveName=paste0(saveName,"PostMarg") )
    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( guess , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab=bquote("Mix. Coef.") , main="'guessing'" )
  panelCount = decideOpenGraph( panelCount , finished=TRUE ,
saveName=paste0(saveName,"PostMarg") )

  # # Standardized scale:
  # panelCount = 1
  # panelCount = decideOpenGraph( panelCount ,
saveName=paste0(saveName,"PostMargZ") )
  # histInfo = plotPost( zbeta0 , cex.lab = 1.75 , showCurve=showCurve ,
  #                      xlab=bquote(z*beta[0]) , main="Intercept" )
  # for ( bIdx in 1:ncol(beta) ) {
  #   panelCount = decideOpenGraph( panelCount ,
saveName=paste0(saveName,"PostMargZ") )
  #   histInfo = plotPost( zbeta[,bIdx] , cex.lab = 1.75 , showCurve=showCurve
,
  #                        xlab=bquote(z*beta[.(bIdx)]) , main=xName[bIdx] )
  # }
  # panelCount = decideOpenGraph( panelCount , finished=TRUE ,
saveName=paste0(saveName,"PostMargZ") )

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

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

#Import Dataset into R
```

```r
water <- read_csv("water_potability.csv")

#Head of the dataset
head(water)

#Structure of the dataset
str(water)

#Datatypes of the variables
sapply(water, class)

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

#Descriptive Analysis

#Summary statistics of the variables
summary(water)

#Histograms of Independent Variables
histgm <- lapply(1:9, function(i) {
  ggplot(water, aes_string(x = colnames(water)[i])) +
    geom_histogram(bins = 10, fill = "yellow", color = "black", na.rm = TRUE) +
    labs(title = paste("Histogram of", colnames(water)[i]), x =
colnames(water)[i]) + theme_pubr()
})
p1 <- ggarrange(plotlist = histgm) + theme_pubr()
p1 #Display the Histogram

#Histograms of the response variable
p2 <- ggplot(water, aes(x = Potability)) + geom_bar(fill = "yellow", color =
"black", stat = "count")
p2 #Display the Histogram of Dependent Variables

#Box plot of Independent Variables by Potability
boxplt <- lapply(1:9, function(i) {
  ggplot(water, aes_string(x = "factor(Potability)", y = colnames(water)[i],
fill = "factor(Potability)")) +
    geom_boxplot(color = "black", na.rm = TRUE) +
    labs(title = paste("Boxplot of", colnames(water)[i], "by Potability"),
         x = "Potability",
         y = colnames(water)[i]) +
    theme_pubr() +
    scale_fill_manual(values = c("lightblue", "yellow"), name = "Potability")
})
p3 <- ggarrange(plotlist = boxplt)
p3

#Correlation heatmap
p4 <- corrplot(cor(na.omit(water)), method = "color", title = "Correlation
Heatmap")
```

```
p4

#Pairs Plot by Potability
p5 <- ggpairs(na.omit(water), columns = 1:9, aes(color = as.factor(Potability),
alpha = 0.5))
p5
#===============================================================================
=

#Handle Missing Values

#Summary of the NA values in the dataset
NA_info <- water %>% group_by(Potability) %>% summarise(
  ph_mean = mean(ph, na.rm = TRUE),
  ph_median = median(ph, na.rm = TRUE),
  ph_NA_perc = (sum(is.na(ph))/length(ph))*100,

  Sulfate_mean = mean(Sulfate, na.rm = TRUE),
  Sulfate_median = median(Sulfate, na.rm = TRUE),
  Sulfate_NA_perc = (sum(is.na(Sulfate))/length(Sulfate))*100,

  Trihalomethanes_mean = mean(Trihalomethanes, na.rm = TRUE),
  Trihalomethanes_median = median(Trihalomethanes, na.rm = TRUE),
  Trihalomethanes_NA_perc =
(sum(is.na(Trihalomethanes))/length(Trihalomethanes))*100
)
NA_info

#Impute NA with the group-specifc medians of the particular column
water_imp <- water %>%
  mutate(
    ph = ifelse(is.na(ph), median(ph, na.rm = TRUE), ph),
    Sulfate = ifelse(is.na(Sulfate), median(Sulfate, na.rm = TRUE), Sulfate),
    Trihalomethanes = ifelse(is.na(Trihalomethanes), median(Trihalomethanes,
na.rm = TRUE), Trihalomethanes)
  ) %>%
  ungroup()

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

#Plots after Imputation

#Summary of the Imputed dataset
summary(water_imp)

#Histograms of Independent Variables
histgm1 <- lapply(1:9, function(i) {
  ggplot(water_imp, aes_string(x = colnames(water_imp)[i])) +
    geom_histogram(bins = 10, fill = "yellow", color = "black", na.rm = TRUE) +
    labs(title = paste("Histogram of", colnames(water_imp)[i]), x =
colnames(water_imp)[i]) + theme_pubr()
})
p6 <- ggarrange(plotlist = histgm1) + theme_pubr()
p6 #Dispx lay the Histogram

#Histograms of the response variable
p7 <- ggplot(water_imp, aes(x = Potability)) + geom_bar(fill = "yellow", color
= "black", stat = "count") +
  labs(title = "Histogram of Potability")
p7 #Display the Histogram of Dependent Variables

#Box plot of Independent Variables by Potability
boxplt1 <- lapply(1:9, function(i) {
  ggplot(water_imp, aes_string(x = "factor(Potability)", y =
colnames(water_imp)[i], fill = "factor(Potability)")) +
    geom_boxplot(color = "black", na.rm = TRUE) +
    labs(title = paste("Boxplot of", colnames(water_imp)[i], "by Potability"),
         x = "Potability",
         y = colnames(water_imp)[i]) +
    theme_pubr() +
    scale_fill_manual(values = c("lightblue", "yellow"), name = "Potability")
})
p8 <- ggarrange(plotlist = boxplt1)
p8

#Correlation heatmap
p9 <- corrplot(cor(water_imp), method = "color", title = "Correlation Heatmap")
p9

#Pairs Plot by Potability
p10 <- ggpairs(water_imp, columns = 1:9, aes(color = as.factor(Potability),
alpha = 0.5))
p10

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

#Splitting the Dataset

#Create a stratified sample split
set.seed(0707)  # Set seed for reproducibility
train_index <- createDataPartition(water_imp$Potability, p = 0.8, list = FALSE)

#Create training and test sets
train <- water_imp[train_index, ]
test <- water_imp[-train_index, ]

# #Check the dimensions
# dim(train)
# dim(test)
```

```
#
# #Check the distribution of the response variable
# table(train$Potability)
# table(test$Potability)

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

#Prepare the data for JAGS input

colnames(train) = c("X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8", "X9", "Y")
colnames(test) = c("X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8", "X9", "Y")
y = train$Y
x = as.matrix(train[, c("X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8", "X9")])
xPred = as.matrix(test[, c("X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8",
"X9")])
Nx = ncol(x)

#Specify the data in a list, for later shipment to JAGS:
dataList <- list(
  x = x,
  y = y,
  xPred = xPred,
  Nx = dim(x)[2],
  Ntotal = dim(x)[1],
  Npred = nrow(xPred)
)

#First run without initials!
initsList <- list(
  zbeta0 = 0,
  zbeta1 = rep(0, ncol(x))  # Initialize zbeta for all predictor variables
)

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

#Informative Run with Robust Logistic Regression Model

modelString = "
data {
  for (j in 1:Nx) {
    xm[j]  <- mean(x[, j])            # Mean of each independent variable
    xsd[j] <- sd(x[, j])              # Standard deviation of each independent
variable
    for (i in 1:Ntotal) {
      zx[i, j] <- (x[i, j] - xm[j]) / xsd[j]  # Standardized values
    }
  }
}
```

```
model {
  # Informative run with standardization
  for (i in 1:Ntotal) {
    y[i] ~ dbern(mu[i])
    mu[i] <- (guess * (1 / 2) + (1.0 - guess) * ilogit(zbeta0 + sum(zbeta[1:Nx]
* zx[i, 1:Nx])))
  }

  # Priors
  zbeta0 ~ dnorm(0, 1 / 2^2)
  zbeta[1] ~ dnorm(0, 1 / 1^2)   # pH - Moderate belief
  zbeta[2] ~ dnorm(0, 1 / 4^2)   # Hardness - Weak belief
  zbeta[3] ~ dnorm(0, 1 / 0.5^2) # Solids (TDS) - Strong belief
  zbeta[4] ~ dnorm(0, 1 / 1.5^2) # Chloramines - Moderate belief
  zbeta[5] ~ dnorm(0, 1 / 2^2)   # Sulfate - Weak to Moderate belief
  zbeta[6] ~ dnorm(0, 1 / 0.5^2) # Conductivity - Strong belief
  zbeta[7] ~ dnorm(0, 1 / 1.5^2) # Organic Carbon - Moderate belief
  zbeta[8] ~ dnorm(0, 1 / 1^2)   # Trihalomethanes - Moderate to Strong belief
  zbeta[9] ~ dnorm(0, 1 / 0.5^2) # Turbidity - Strong belief

  # 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])

  guess ~ dbeta(2, 8)

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

parameters = c(parameters, "guess")

adaptSteps = 500  # Number of steps to "tune" the samplers
burnInSteps = 1000
nChains = 4
thinSteps = 10
numSavedSteps = 1000
```

```
nIter = ceiling( ( numSavedSteps * thinSteps ) / nChains )

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

#Save Environment after running JAGS
save.image(file='WaterPotability_10000stepsInformativeRobust.Rdata')
load('WaterPotability_10000stepsInformativeRobust.Rdata') #Informative Run with
Robust Model
#===============================================================================
=

#Removing X3, X6, X9 as they do not significantly contribute to the model

y = train$Y
x = as.matrix(train[, c("X1", "X2", "X4", "X5", "X7", "X8")])
xPred = as.matrix(test[, c("X1", "X2", "X4", "X5", "X7", "X8")])
Nx = ncol(x)

#Specify the data in a list, for later shipment to JAGS:
dataList <- list(
  x = x,
  y = y,
  xPred = xPred,
  Nx = ncol(x),
  Ntotal = dim(x)[1],
  Npred = nrow(xPred)
)

#Informative Run with Robust Logistic Regression Model

modelString = "
data {
  for (j in 1:Nx) {
    xm[j]  <- mean(x[, j])           # Mean of each independent variable
    xsd[j] <- sd(x[, j])             # Standard deviation of each independent
variable
    for (i in 1:Ntotal) {
      zx[i, j] <- (x[i, j] - xm[j]) / xsd[j]  # Standardized values
    }
  }
}

model {
  # Informative run with standardization
  for (i in 1:Ntotal) {
    y[i] ~ dbern(mu[i])
    mu[i] <- (guess * (1 / 2) + (1.0 - guess) * ilogit(zbeta0 + sum(zbeta[1:Nx]
* zx[i, 1:Nx])))
  }

  # Priors
  zbeta0 ~ dnorm(0, 1 / 2^2)
  zbeta[1] ~ dnorm(0, 1 / 1^2)   # pH - Moderate belief
  zbeta[2] ~ dnorm(0, 1 / 4^2)   # Hardness - Weak belief
  zbeta[3] ~ dnorm(0, 1 / 1.5^2) # Chloramines - Moderate belief
  zbeta[4] ~ dnorm(0, 1 / 2^2)   # Sulfate - Weak to Moderate belief
  zbeta[5] ~ dnorm(0, 1 / 1.5^2) # Organic Carbon - Moderate belief
  zbeta[6] ~ dnorm(0, 1 / 1^2)   # Trihalomethanes - Moderate to Strong belief

  # 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])

  guess ~ dbeta(2, 8)

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

adaptSteps = 500  # Number of steps to "tune" the samplers
burnInSteps = 1000
nChains = 4
thinSteps = 10
```

```
numSavedSteps = 1000
nIter = ceiling( ( numSavedSteps * thinSteps ) / nChains )

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

#Save Environment after running JAGS
save.image(file='WaterPotability_10000stepsInformativeRobustFiltered.Rdata')
load('WaterPotability_10000stepsInformativeRobustFiltered.Rdata') #Informative
Run with Robust Model


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

#Non-Informative Robust

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 {
  # Non-informative run with standardisation
  for ( i in 1:Ntotal ) {
    # In JAGS, ilogit is logistic:
    y[i] ~ dbern( mu[i] )
      mu[i] <- ( guess*(1/2) + (1.0-
guess)*ilogit(zbeta0+sum(zbeta[1:Nx]*zx[i,1:Nx])) )
  }
  # Priors vague on standardized scale:
  zbeta0 ~ dnorm( 0 , 1/10^2 )
  # non-informative run
  for ( j in 1:Nx ) {
    zbeta[j] ~ dnorm( 0 , 1/10^2 )
  }
  #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] )

  guess ~ dbeta(2,8)

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

parameters = c(parameters, "guess")

adaptSteps = 500  # Number of steps to "tune" the samplers
burnInSteps = 1000
nChains = 4
thinSteps = 10
numSavedSteps = 1000
nIter = ceiling( ( numSavedSteps * thinSteps ) / nChains )

startTime = proc.time()d
runJagsOut <- run.jags( method="parallel" ,
                        model="TEMPmodel.txt" ,
                        monitor=parameters  ,
                        data=dataList ,
                        # inits=initsList ,
                        n.chains=nChains ,
                        adapt=adaptSteps ,
                        burnin=burnInSteps ,
                        sample=numSavedSteps ,
                        thin=thinSteps , summarise=FALSE , plots=FALSE )
```

```
stopTime = proc.time()
duration = stopTime - startTime
show(duration)
codaSamples = as.mcmc.list( runJagsOut )

#Save Environment after running JAGS

save.image(file='WaterPotability_10000stepsNonInformativeRobustPriorSensitivity
.Rdata')
load('WaterPotability_10000stepsNonInformativeRobustPriorSensitivity.Rdata')


# adaptSteps = 100 ;burnInSteps = 500; nChains = 3; thinSteps = 2;
numSavedSteps = 1000
# save.image(file='WaterPotability_2000stepsNonInformativeRobust.Rdata')
load('WaterPotability_2000stepsNonInformativeRobust.Rdata')

# adaptSteps = 500 ;burnInSteps = 1000; nChains = 4; thinSteps = 3;
numSavedSteps = 1000
# save.image(file='WaterPotability_3000stepsNonInformativeRobust.Rdata')
load('WaterPotability_3000stepsNonInformativeRobust.Rdata')

# adaptSteps = 500 ;burnInSteps = 1000; nChains = 4; thinSteps = 5;
numSavedSteps = 1000
# save.image(file='WaterPotability_5000stepsNonInformativeRobust.Rdata')
load('WaterPotability_5000stepsNonInformativeRobust.Rdata')

# adaptSteps = 500 ;burnInSteps = 1000; nChains = 4; thinSteps = 10;
numSavedSteps = 1000
save.image(file='WaterPotability_10000stepsNonInformativeRobust.Rdata')
load('WaterPotability_10000stepsNonInformativeRobust.Rdata')

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

#Diagnostic Check

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

graphics.off()
```

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

#Posteriors Check

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, check.names=FALSE)

summaryInfo <- smryMCMC( codaSamples = codaSamples )
print(summaryInfo)

plotMCMC( codaSamples = codaSamples , data = train, yName = "Y",
          xName = c("X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8", "X9") )


plotMCMC( codaSamples = codaSamples , data = train, yName = "Y",
          xName = c("X1", "X2", "X4", "X5", "X7", "X8") )

graphics.off()

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


BayesTheta = summaryInfo[22:nrow(summaryInfo),3] # Take the Bayesian estimates
for the test set
# summaryInfo[16:nrow(summaryInfo),3] use this after filtering insignificant
variables
# Classify based on the threshold of 0.5
preds = BayesTheta
preds[which(BayesTheta < 0.5)] = 0
preds[which(BayesTheta > 0.5)] = 1
# Actual Potabilities types
ActualPotability = as.numeric(test$Y)

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, conf = conf))
}

confusionMatrix(resp = ActualPotability, pred = preds)
```