Bayesian insights into predicting math scores

Angelo Mandara

2023-06-26

1. Dataset Overview and Illustration

Data for this study came from Kaggle, a well-known website for data science and machine learning competitions. You can access the specified dataset at the given link (https://www.kaggle.com/datasets/spscientist/students-performance-in-exams).

Researchers, data scientists, and practitioners may access and analyse a wide range of real-world data thanks to the vast range of datasets provided by the community on Kaggle. Examining the connections between academic achievement and demographic variables is made possible by this dataset, which focuses on students' grades in various areas. For consistent and trustworthy analysis, the Kaggle platform makes sure that the dataset has been vetted, packaged, and made available to users.

The goal of this study is to evaluate students' performance across a range of courses, with a focus on forecasting mathematics outcomes based on a variety of demographic and academic variables. The dataset includes data on reading, writing, math, gender, race/ethnicity, parental education level, lunch choice, exam preparation course completion, and gender.

The major objective of this project is to construct a **regression model that makes use of Bayesian inference to forecast results in mathematics depending on available factors**. Understanding the variables that affect arithmetic performance can give educators and decision-makers new perspectives on how to raise student achievement

Why this study can be helpful:

- Math score predictions can offer insightful information about a student's academic achievement, according to the **Academic achievement Assessment**. It enables teachers to recognise kids who might require more assistance or resources in mathematics and to design interventions accordingly.
- **Identifying Influential elements**: Examining the correlation between math test results and different demographic and academic variables can aid in determining which elements have the greatest influence on a student's success. This knowledge can direct educational initiatives and policies aimed at enhancing math instruction.

The dataset consists of the following variables:

- **Gender**: Categorical variable indicating the gender of the student.
- **Race/Ethnicity**: Categorical variable representing the race or ethnicity of the student.
- **Parental Level of Education**: Categorical variable indicating the highest level of education attained by the student's parents.
- **Lunch**: Categorical variable indicating whether the student receives free/reduced lunch or standard lunch.
- **Test Preparation Course**: Categorical variable indicating whether the student completed a test preparation course.
- **Reading Score**: Numerical variable representing the score obtained by the student in the reading subject.
- **Writing Score**: Numerical variable representing the score obtained by the student in the writing subject.
- **Math Score**: Numerical variable representing the **target** variable, the score obtained by the student in the math subject.

Let's have a look of the first 10 rows of the dataset to have an idea:

```
head(data,10)
```

1
2
3
4
5
6
7
8
9
10

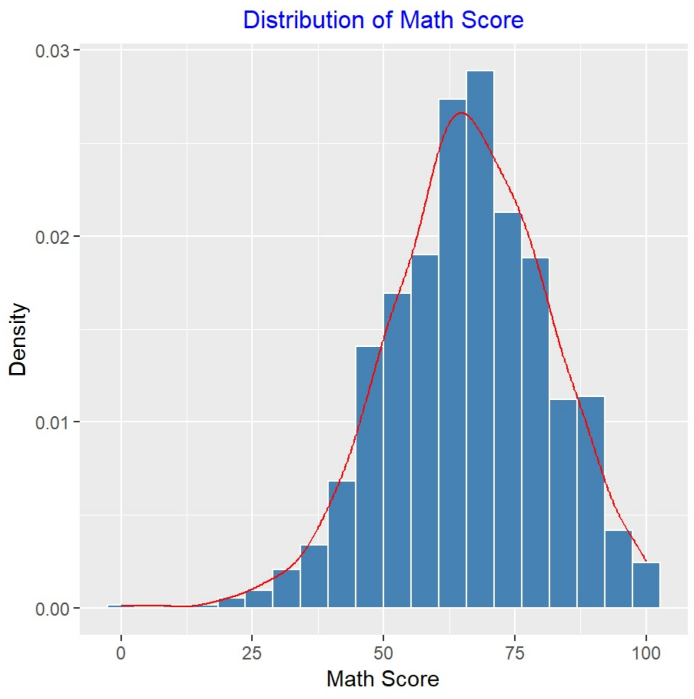
1-10 of 10 rows | 1-1 of 9 columns

Let's see also some summaries of the target variable:

```
print(summary(data$math.score))
```

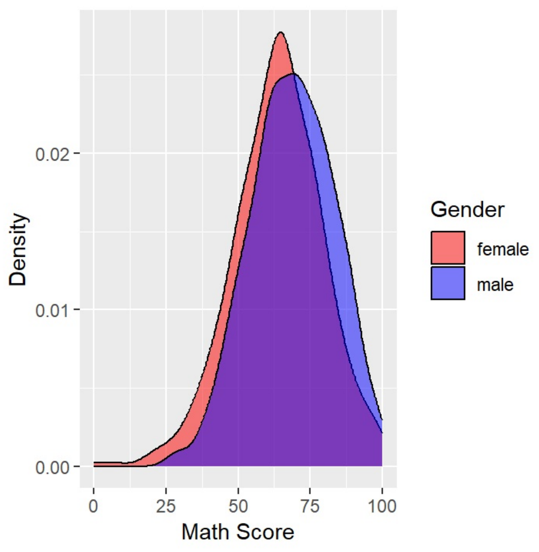
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    0.00   57.00   66.00   66.09   77.00  100.00
```

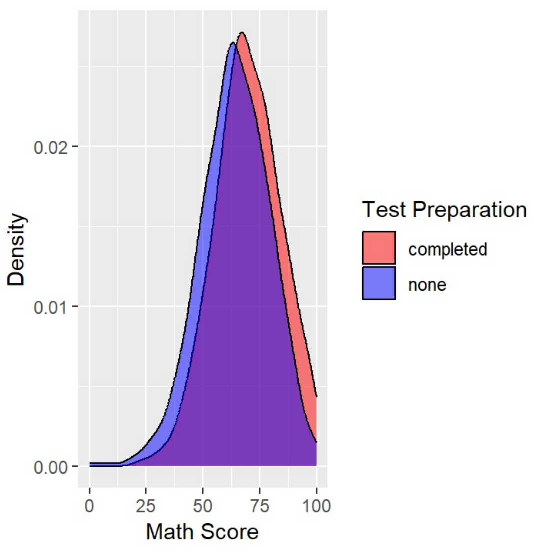
There is some simmetry, let’s plot it:

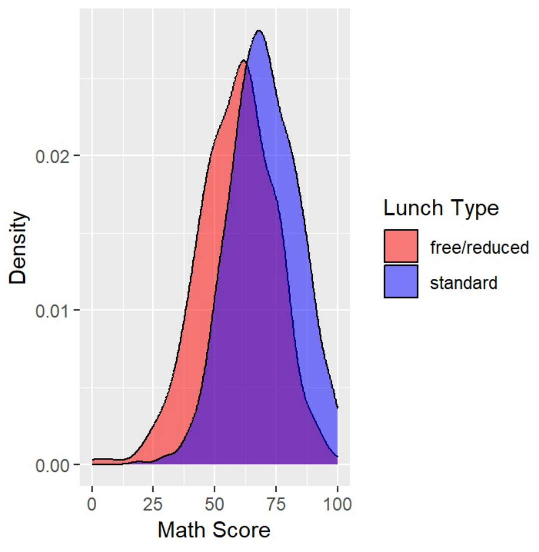


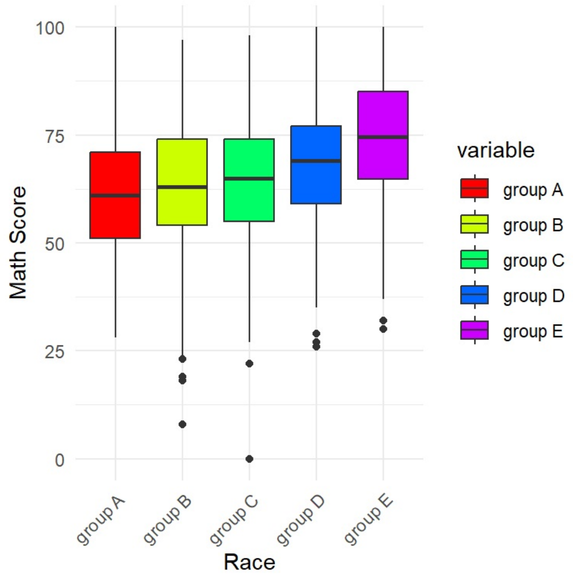
After analyzing the plot of the target variable, it can be observed that the distribution of the math scores is approximately normal, centered around 66.

Let’s do some studies about the datas.







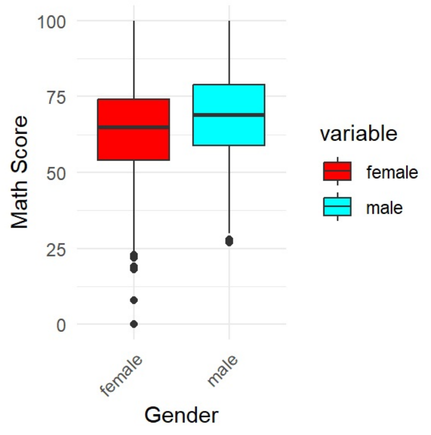


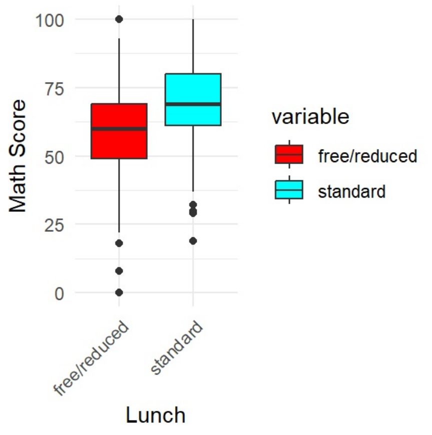
The density plot of the Math Score based on Gender shows that, in general, male students tend to have higher grades compared to female students for the higher scores. This observation suggests a potential gender difference in math performance.

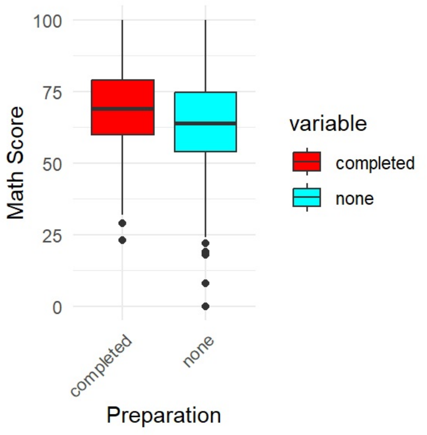
Similarly, the density plot based on Test Preparation indicates that students who completed the test preparation course tend to have higher math scores compared to those who didn't. This finding suggests that the test preparation course may positively impact math performance.

Regarding the lunch type, the density plot shows that students with a standard lunch tend to have slightly higher math scores compared to students with a free/reduced lunch. This finding may suggest that the type of lunch could be associated with math performance, although the difference appears to be relatively small.

For the analysis of race/ethnicity I have used box plots, that show the median, quartiles, and potential outliers for each group. From the box plots, is possible to observe some differences in math scores across racial/ethnic groups.





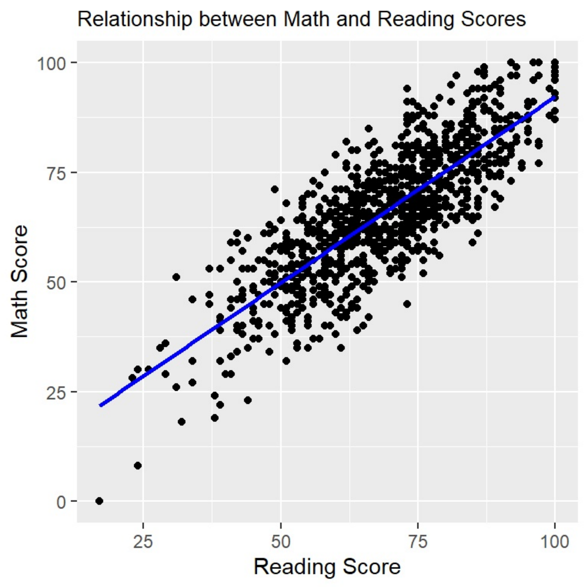


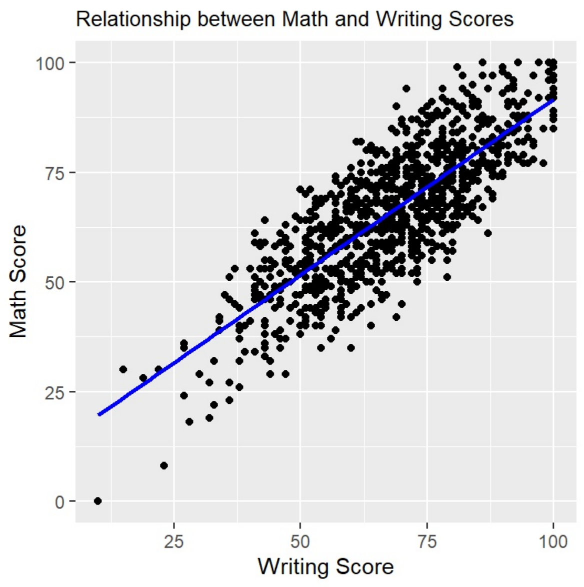
The box plots for gender, lunch, and test preparation provide insights into the distribution of math scores across different categories within each variable.

From the plot it appears that male students tend to have slightly higher median math scores compared to female students.

Also students with a standard lunch have slightly higher median math scores compared to those with a free/reduced lunch.

Last plot suggests that students who completed the test preparation course tend to have higher median math scores compared to those who did not.





The scatter plots between the Math Score (target variable) and the Writing Score, as well as the Reading Score, clearly demonstrate a strong positive correlation. This correlation can be observed from the overall upward trend in the scatter plots, as well as the fitted regression lines that show a positive slope.

The high positive correlation between the Math Score and the Writing Score suggests that students who perform well in writing also tend to perform well in math, and vice versa. Similarly, the correlation between the Math Score and the Reading Score indicates that students with higher reading scores also tend to have higher math scores.

Let's check if there are any NA values:

```
##                      gender              race.ethnicity
##                           0                           0
## parental.level.of.education                       lunch
##                           0                           0
##     test.preparation.course                  math.score
##                           0                           0
##               reading.score               writing.score
##                           0                           0
```

Now we will encode the categorical data into integers in this way:

- **Sex** $\rightarrow$ binary : 0 female, 1 - male;
- **Race Ethincity** $\rightarrow$ numeric:
  - 1 - group A
  - 2 - group B
  - 3 - group C
  - 4 - group D
- **Parental Level of Education** $\rightarrow$ numeric:
  - 1 - associate's degree
  - 2 - bachelor's degree
  - 3 - high school
  - 4 - master's degree
  - 5 - some college
- **Lunch** $\rightarrow$ binary : 0 free/reduced, 1 - standard;
- **Test Preparation Course** $\rightarrow$ binary : 0 completed, 1 - none;

The dataset is transformed and look this way now:

```
head(data,10)
```

▸
1
2
3
4
5

6
7
8
9
10

1-10 of 10 rows | 1-1 of 9 columns

To ensure accurate predictions and evaluate model performance, we split the dataset into two parts: a training set (85% of the observations) and a test set. The training set is used to build the model, while the test set is reserved for predicting math score and assessing model effectiveness.

```
set.seed(123)
indexes <- sample(dim(data)[1], size = ceiling(dim(data)[1] * 0.85))
train <- data[indexes, ]
test <- data[-indexes, ]
target <- test[, c("math.score")]
test[, c("math.score")] <- NULL
```

2. Linear Regression Model for Predicting Math Score

In the linear regression model, the Math Score is considered as the target variable. This model assumes that the data are random samples from a normal distribution, where the mean is a linear function of the predictors.

The aim of this analysis is to build a linear regression model to predict the Math Score based on various predictor variables. By examining the relationship between the Math Score and the predictors, we can estimate the model coefficients and make predictions on the Math Score for new observations.

The multivariate linear regression model can be expressed as:

$y_i \sim N(\mu_i, \sigma^2)$

$\mu_i = \alpha + \beta_1 \cdot x_1 + \ldots + \beta_p \cdot x_p$

where:

- $\boldsymbol{y_i}$ is our target variable.
- The predictors are denoted as $\boldsymbol{x_i}$.
- $\boldsymbol{\beta = (\alpha, \beta_1, \beta_2, \ldots, \beta_p)}$ is the vector of model coefficients.

We have 7 parameters(+α), so $p = 7$:

- $\boldsymbol{\alpha \in \mathbb{R}, \alpha \sim N(0, 10000)}$
- $\boldsymbol{\beta_1, .., \beta_7 \in \mathbb{R}, \beta_i \sim N(0, 10000)}$
- $\boldsymbol{\sigma \in [0, 100], \sigma \sim Unif(0, 100)}$

For better interpretability we will denote:

$\beta_1 = \beta_{gender}, \beta_2 = \beta_{race}, \beta_3 = \beta_{parenteduclevel}, \beta_4 = \beta_{lunch}, \beta_5 = \beta_{preplevel}, \beta_6 = \beta_{writing}, \beta_7 = \beta_{reading}$

In this model, the target variable (Math Score) is represented by the linear combination of the predictor variables: Gender, Race, Parent Education Level, Lunch, Test Preparation Level, Writing Score, and Reading Score. The coefficients β_1 to β_7 represent the influence of each predictor on the Math Score. The intercept α captures the baseline Math Score when all predictors are zero.

To estimate the model parameters, we used non informative priors, so all possible values approximately equally likely, assuming a normal distribution for the coefficients with a mean of zero and a large standard deviation(low precision). Also, we consider a uniform prior distribution for the standard deviation σ, reflecting the possible range of values for the Math Score, infact I choose a uniform distribution in the range $[0, 100]$ as the prior.

To approximate the posterior distribution of the target variable and the model coefficients, we can utilize Monte Carlo methods, such as Markov Chain Monte Carlo (MCMC). These methods allow us to draw samples from the posterior distribution, enabling inference and estimation of the model parameters.

We aim to learn more about the variables that affect math scores by building this linear regression model. We also aim to provide a solid framework for predicting math scores based on the provided predictor variables.

Let's start with our model(score for student i):

$score_i = \alpha + \beta_{gender} \cdot gender_i + \beta_{race} \cdot race_i + \beta_{parenteduclevel} \cdot parenteduclevel_i + \beta_{lunch} \cdot lunch_i + \beta_{prep\_level} \cdot preparlevel_i + \beta_{writing} \cdot writing_i + \beta_{reading} \cdot reading_i$

```
jags_data <- list(
  gender = train$gender,
  race = train$race.ethnicity,
  parent_educ_level = train$parental.level.of.education,
  lunch = train$lunch,
  prep_level = train$test.preparation.course,
  math_score = train$math.score,
  writing_score = train$writing.score,
  reading_score = train$reading.score,
  N = dim(train)[1]
)

# Define the parameters to be estimated
parameters <- c("alpha", "beta_gender", "beta_race","beta_parent_educ_level",
                "beta_lunch","beta_prep_level","beta_writing","beta_reading", "sigma")

# Initial parameter values for the MCMC sampler
inits <- list(
  list(
    alpha=rnorm(1), beta_gender=rnorm(1), beta_race=rnorm(1), beta_parent_educ_level=rnorm(1),
    beta_lunch=rnorm(1), beta_prep_level=rnorm(1), beta_writing=rnorm(1),
    beta_reading=rnorm(1), sigma=runif(0,1000)
  ),
  list(
    alpha=rnorm(1), beta_gender=rnorm(1), beta_race=rnorm(1), beta_parent_educ_level=rnorm(1),
    beta_lunch=rnorm(1), beta_prep_level=rnorm(1), beta_writing=rnorm(1),
    beta_reading=rnorm(1), sigma=runif(0,1000)
  )
)


# Run the MCMC chain with one chain
set.seed(123)
scoresjags <- jags(
  data = jags_data,
  inits = inits,
  parameters.to.save = parameters,
  model.file = "jags_mandara_one.txt",
  n.chains = 2,
  n.iter = 50000,
  n.burnin = 5000,
  n.thin = 10
)
```

```
## module glm loaded
```

```
scoresjags
```

```
## Inference for Bugs model at "jags_mandara_one.txt", fit using jags,
##  2 chains, each with 50000 iterations (first 5000 discarded), n.thin = 10
##  n.sims = 9000 iterations saved
##                          mu.vect sd.vect     2.5%      25%      50%      75%
## alpha                    -12.474   1.286  -15.014  -13.353  -12.471  -11.616
## beta_gender               13.016   0.405   12.224   12.742   13.020   13.295
## beta_lunch                 3.492   0.412    2.680    3.214    3.493    3.768
## beta_parent_educ_level     0.003   0.104   -0.201   -0.067    0.002    0.074
## beta_prep_level            3.093   0.425    2.263    2.812    3.092    3.377
## beta_race                  0.913   0.164    0.585    0.803    0.913    1.024
## beta_reading               0.321   0.045    0.231    0.290    0.321    0.352
## beta_writing               0.631   0.046    0.543    0.600    0.631    0.662
## sigma                      5.473   0.134    5.220    5.381    5.469    5.560
## deviance                5300.242   4.268 5293.929 5297.111 5299.572 5302.691
##                            97.5%  Rhat n.eff
## alpha                     -9.948 1.001  9000
## beta_gender               13.795 1.001  9000
## beta_lunch                 4.318 1.001  9000
## beta_parent_educ_level     0.205 1.001  9000
## beta_prep_level            3.911 1.001  9000
## beta_race                  1.230 1.001  3100
## beta_reading               0.409 1.001  9000
## beta_writing               0.721 1.001  9000
## sigma                      5.746 1.002  1900
## deviance                5310.112 1.001  9000
## 
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## 
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 9.1 and DIC = 5309.4
## DIC is an estimate of expected predictive error (lower deviance is better).
```

We can gain a lot regards our model parameters' characteristics by examining their posterior distributions.

- **Mean**: The mean reflects the central tendency of the posterior distribution and serves as a point estimate for each parameter.
- **Standard Deviation**: Each parameter's standard deviation reveals the degree of uncertainty surrounding its estimation. Wider credible intervals are implied by higher standard deviations for the parameter's point estimate. The parameter in our model with the highest posterior uncertainty is α.
- **Quantiles**: Additional details about the parameter estimations are provided by the quantiles of the marginal posterior distribution. The 95% credible interval can be thought of as having boundaries at the 2.5% and 97.5% quantiles, giving a range of possible values for the parameter.
- **Gelman-Rubin diagnostic**, often referred to as Rhat. Rhat compares the variation within each chain to the variation between chains. If the chains have converged, Rhat should be close to 1. Values greater than 1 suggest lack of convergence, indicating that the chains have not reached equilibrium.

In this, since all Rhat values are approximately 1.01, it suggests that the chains have converged and reached equilibrium.

The $beta_{gender}$ coefficient has a mean of 13.040. This suggests that, on average, there is a positive association between the gender predictor and the Math Score. For each unit increase in the gender predictor, the Math Score is expected to increase by approximately 13.040.

Similarly, the other coefficients $beta_{lunch}$, $beta_{parent_{e}duc_{l}evel}$, $beta_{prep_{l}evel}$, $beta_{race}$, $beta_{reading}$, $beta_{writing}$ indicate the expected change in the Math Score associated with a one-unit increase in the respective predictor.

The deviance measure contains information about the fit of our model to the data. In general, a lower deviance value indicates a better fit, as it reflects higher likelihood for observed data points. It helps us assess how well our model captures the patterns and relationships present in the data.

In this case, the effective sample size (n.eff) is reported as 9000(2*(50000-5000)/10) for all parameters except for β_{race}, which has an effective sample size of 3100, and σ, which has an effective sample size of 1900. The effective sample size represents the number of independent samples obtained from the MCMC chains.

The effective sample size is a measure that takes into account the autocorrelation in the saved samples. It is calculated as a fraction of the total number of samples saved from the chains, adjusted for thinning (in our case, a thinning interval of 10). In our scenario, the maximum possible effective sample size is achieved, indicating that the samples are nearly independent and the autocorrelation is low.

It's important to note that the effective sample size is a measure of the number of independent samples, and a higher effective sample size generally leads to more reliable and precise estimates of the posterior distribution.

3. MCMC diagnostics

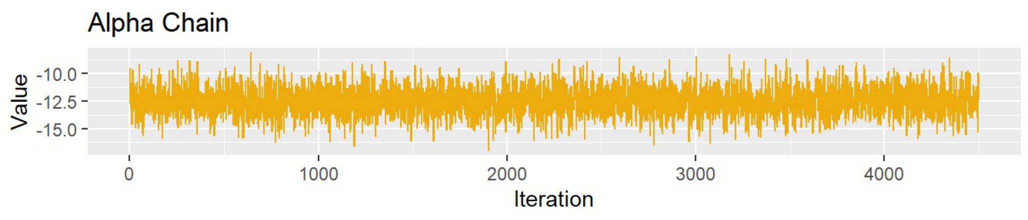
By creating simulations from this distribution, MCMC sampling aims to approximate the posterior distribution. It is necessary to evaluate the MCMC samples' reliability in reflecting the posterior, though.

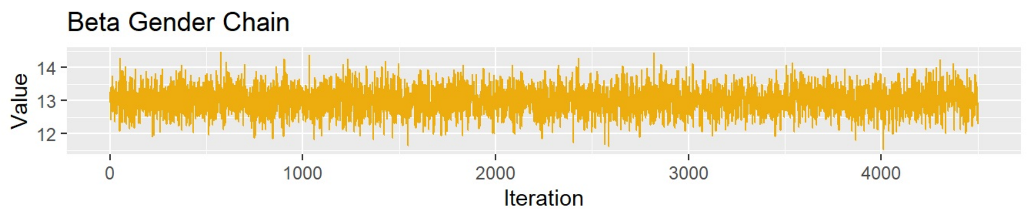
3.1 Traceplots and posterior distributions

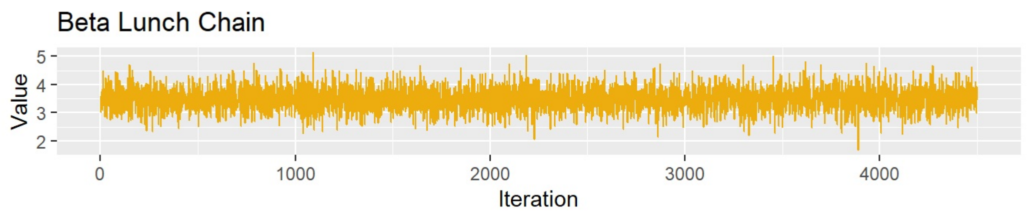
Traceplots and posterior distributions can be examined to determine the reliability of MCMC samples. These graphs offer useful details about the convergence and behaviour of the MCMC chain.

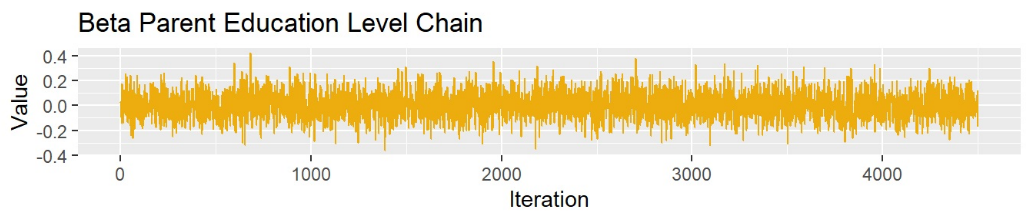
The progression of MCMC sample values along the iterations is shown using traceplots. The chain has converged when the traceplot is stable and smooth, appropriately capturing the posterior distribution. On the other side, it indicates that the chain may need additional iterations to adequately represent the distribution if the traceplot shows patterns or chaotic behaviour.

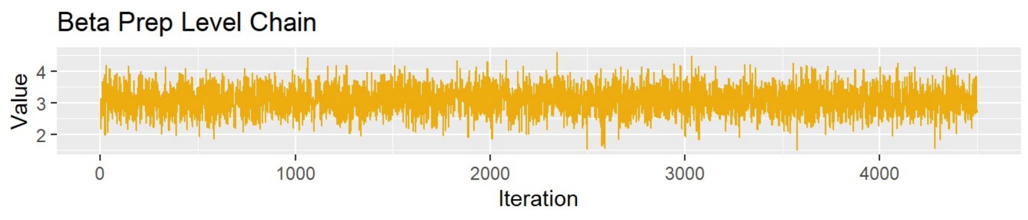
The probability density functions of the parameter estimates are shown by posterior distributions. They help us to see how the posterior distribution is distributed and how its central tendency is shaped.

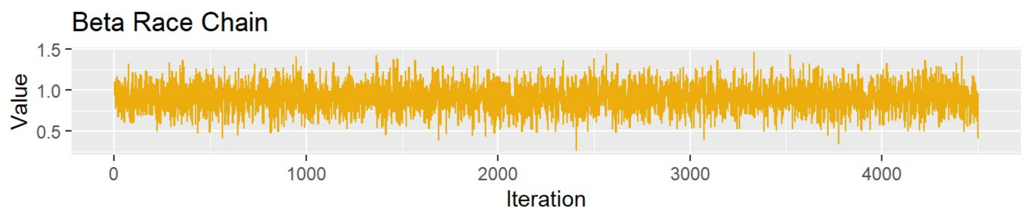


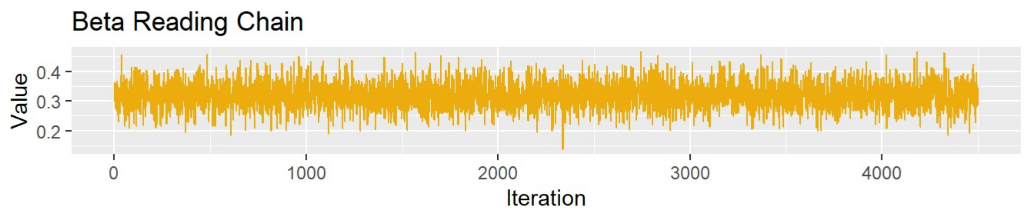


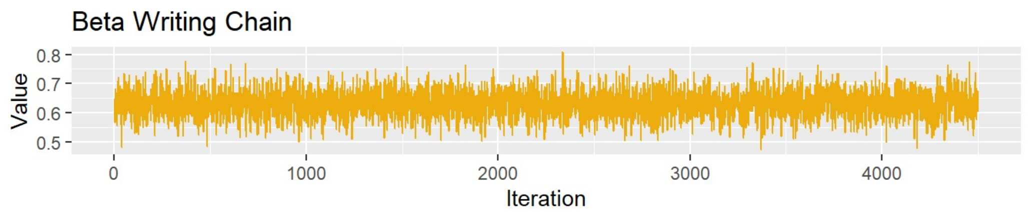


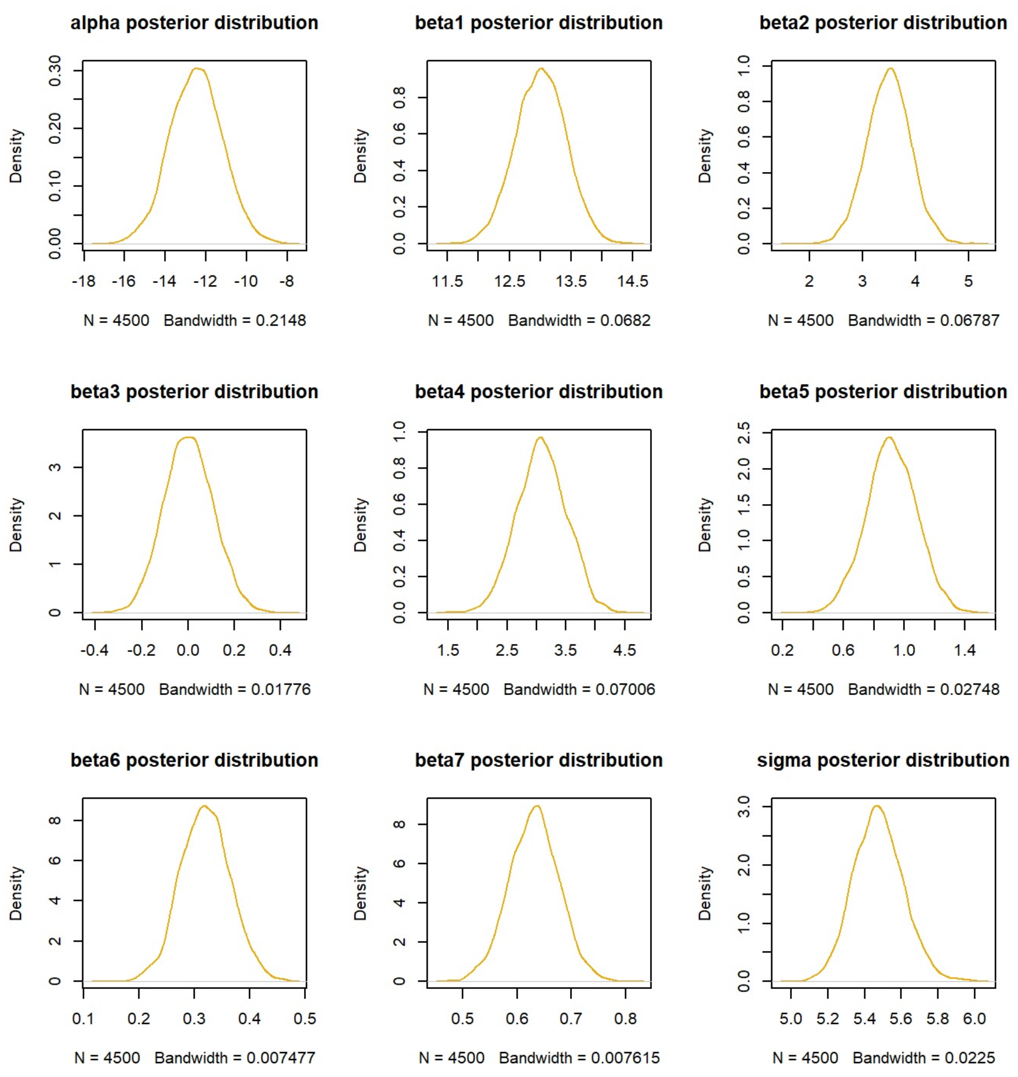












3.2 Convergence

To ensure the accuracy of the posterior estimates, it is essential to evaluate the convergence of MCMC samples. The Geweke diagnostic, which compares the means of the early and late regions of the Markov chain, is one often used diagnostic technique. These means ought to be close to being equal if the samples have converged.

We can also analyze the empirical average of the parameters over time to observe their convergence behavior. Calculating the average of parameter values at each iteration allows us to monitor how the estimates stabilize and approach their true values.

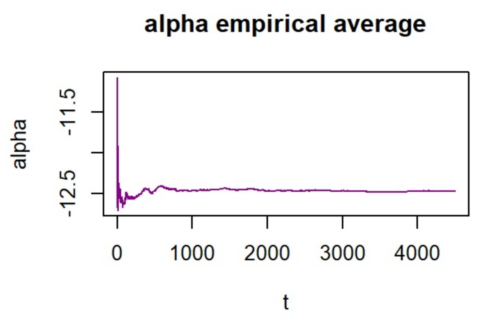
By examining these convergence diagnostics, we can determine if the MCMC samples accurately reflect the posterior distribution and if the estimates for the model parameters are reliable.

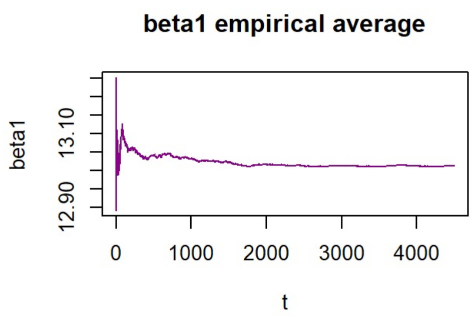
```
jags.mcmc<- as.mcmc(scoresjags)

geweke.diag(jags.mcmc)
```

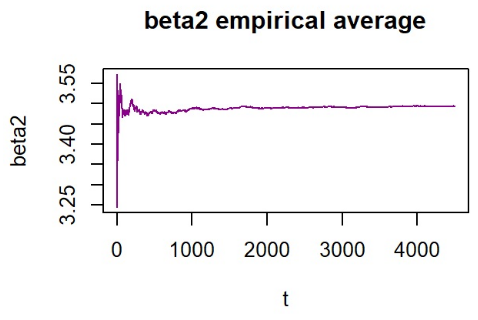
```
## [[1]]
## 
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5 
## 
##                alpha           beta_gender            beta_lunch 
##              -0.1644                1.4718               -0.9157 
## beta_parent_educ_level       beta_prep_level             beta_race 
##              -1.2307                0.5155               -0.9546 
##         beta_reading          beta_writing              deviance 
##              -0.1093                0.3350               -0.5408 
##                sigma 
##              -0.5627 
## 
## 
## [[2]]
## 
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5 
## 
##                alpha           beta_gender            beta_lunch 
##             -1.13340              -0.12902               0.63280 
## beta_parent_educ_level       beta_prep_level             beta_race 
##             -0.07974               1.30853               1.05695 
##         beta_reading          beta_writing              deviance 
##             -0.79282               0.90039              -0.63562 
##                sigma 
##             -0.56441
```

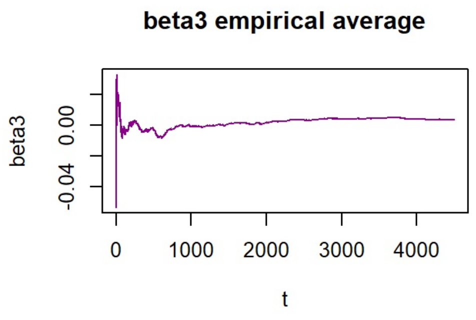
This output indicates the difference between the two sample means divided by their estimated standard error. Lower values of this difference suggest good convergence of the MCMC samples.

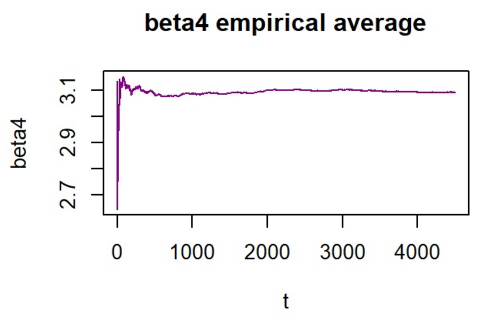
Next, we can examine the behavior of the empirical average of the parameters as the number of iterations increases. By calculating the average of the parameter values at each iteration, we can observe how the estimates evolve over time. This analysis provides insights into the stability and convergence of the parameter estimates as more iterations are performed.

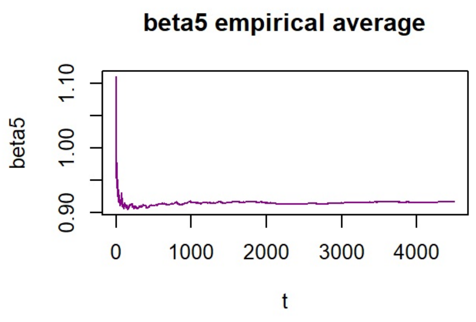

alpha empirical average
alpha
-11.5
-12.5
0
1000
2000
3000
4000
t

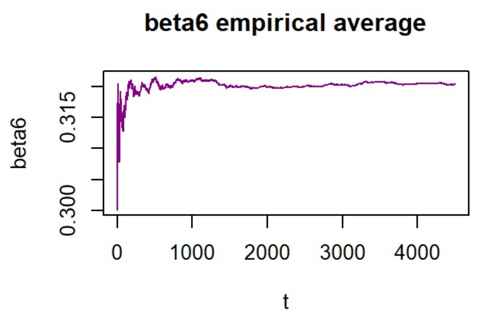

beta1 empirical average
beta1
13.10
12.90
0
1000
2000
3000
4000
t

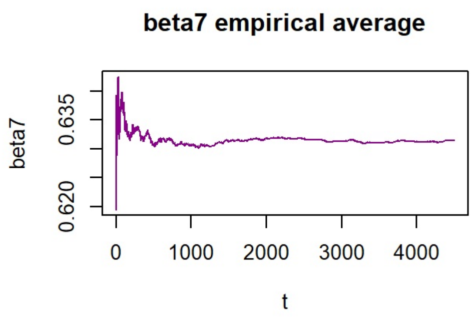

beta2 empirical average
beta2
3.55
3.40
3.25
0
1000
2000
3000
4000
t

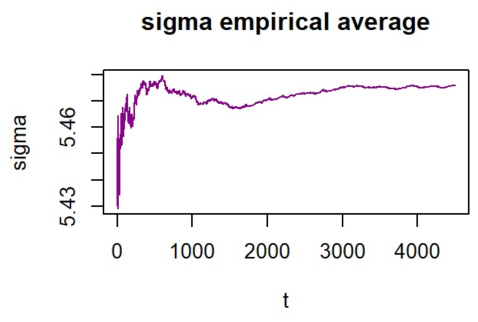

beta3 empirical average
beta3
0.00
-0.04
0
1000
2000
3000
4000
t


beta4 empirical average
beta4
3.1
2.9
2.7
0
1000
2000
3000
4000
t


beta5 empirical average
beta5
1.10
1.00
0.90
0
1000
2000
3000
4000
t


beta6 empirical average
beta6
0.315
0.300
0
1000
2000
3000
4000
t


beta7 empirical average
beta7
0.635
0.620
0
1000
2000
3000
4000
t


sigma empirical average
sigma
5.46
5.43
0
1000
2000
3000
4000
t

As the number of iterations increases, we see that the empirical averages of the parameters initially display large amounts of fluctuation. The averages have however a tendency to stabilise and converge when more iterations are run, showing less variability and getting closer to their true values. The MCMC sampling procedure successfully captured the characteristics of the posterior distribution based on the convergence behaviour.

3.3 Auto-correlation

Auto-correlation measures the degree of dependence between consecutive samples in the MCMC chain at different lags. It helps assess the mixing and convergence of the chain by examining the correlation patterns.

By calculating the auto-correlation at various lags, we can determine the level of independence between MCMC samples. Ideally, as the lag increases, the auto-correlation should decrease, indicating that the samples become more independent and reliable for parameter estimation.

The small auto-correlation values obtained suggest that the MCMC samples are nearly independent, which is favorable for accurate parameter estimation.

```
autocorr.diag(jags.mcmc)
```

```
##                alpha  beta_gender   beta_lunch beta_parent_educ_level
## Lag 0    1.000000000  1.000000000  1.000000000            1.000000000
## Lag 10   0.011075608  0.001340583  0.005997857            0.015318203
## Lag 50   0.005947135 -0.026045315 -0.005490612           -0.012655893
## Lag 100 -0.012707530 -0.001359487 -0.001315103           -0.008484789
## Lag 500 -0.012826252  0.017728836 -0.012400487           -0.019894144
##         beta_prep_level    beta_race beta_reading beta_writing     deviance
## Lag 0       1.000000000  1.000000000  1.000000000  1.000000000  1.000000000
## Lag 10      0.021439537 -0.005022253 -0.001333565  0.006235236 -0.002622315
## Lag 50     -0.004994190  0.009853342  0.008192098  0.003979654 -0.003666362
## Lag 100     0.011423579 -0.013018876 -0.001156812 -0.010790406 -0.004346112
## Lag 500    -0.006251438  0.005825162  0.011373442  0.007348292  0.016196070
##                sigma
## Lag 0    1.000000000
## Lag 10  -0.008497886
## Lag 50   0.011004346
## Lag 100 -0.002722549
## Lag 500 -0.004451240
```

This output show us that not always the autocorrelation decrease when the lag increase, but anyway the autocorrelation values are very small, so the MCMC samples seems to be almost indipendent.

3.4 Monte Carlo Standard Error

The Monte Carlo Standard Error (MCSE) quantifies the uncertainty in parameter estimates due to sampling error. When dealing with autocorrelated samples, it is important to use the effective sample size (Neff) instead of the total sample size (N) in the MCSE calculation.

The MCSE values for each parameter indicate the level of uncertainty associated with the estimates. A smaller MCSE suggests more precise estimates with lower sampling error, while a larger MCSE indicates greater uncertainty and potential bias in the estimates. Incorporating the effective sample size improves the accuracy of the MCSE calculation, accounting for the correlation among the MCMC samples.

```
mcse(alpha_chain)
```

```
## $est
## [1] -12.47657
##
## $se
## [1] 0.01913925
```

```
mcse(beta_gender_chain)
```

```
## $est
## [1] 13.01195
##
## $se
## [1] 0.006075319
```

```
mcse(beta_lunch_chain)
```

```
## $est
## [1] 3.493449
##
## $se
## [1] 0.00606118
```

```
mcse(beta_parent_educ_level_chain)
```

```
## $est
## [1] 0.003249446
##
## $se
## [1] 0.001581925
```

```
mcse(beta_prep_level_chain)
```

```
## $est
## [1] 3.093185
##
## $se
## [1] 0.006270968
```

```
mcse(beta_race_chain)
```

```
## $est
## [1] 0.9156869
##
## $se
## [1] 0.002448107
```

```
mcse(beta_reading_chain)
```

```
## $est
## [1] 0.3203575
##
## $se
## [1] 0.0006660407
```

```
mcse(beta_writing_chain)
```

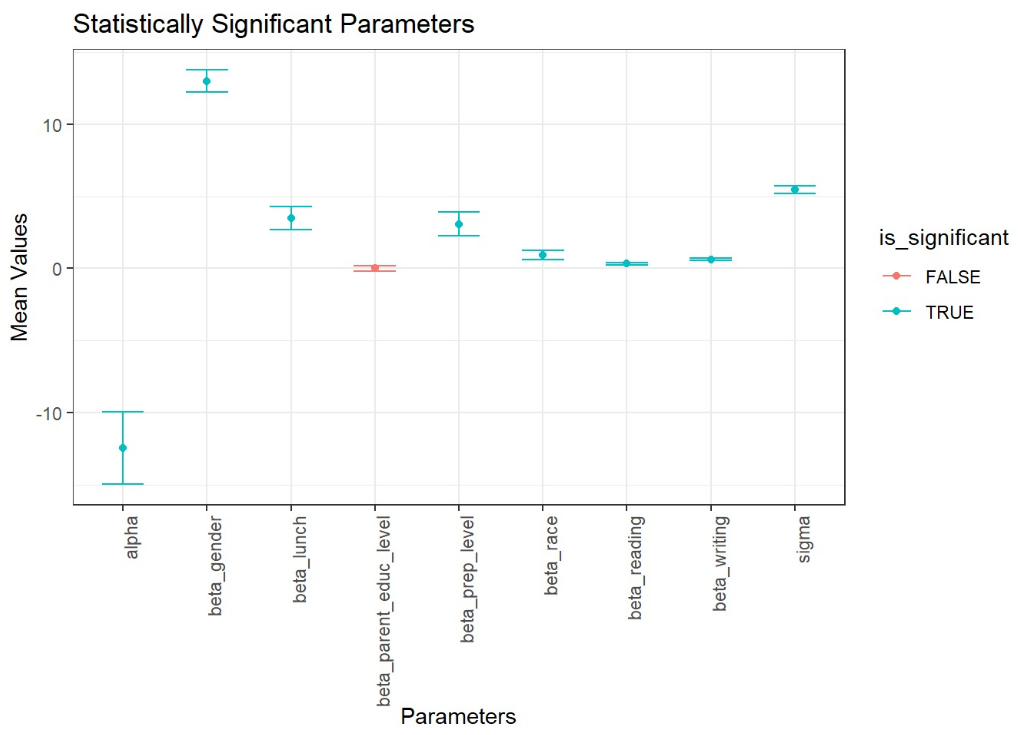
```
## $est
## [1] 0.631453
##
## $se
## [1] 0.0006783609
```

```
mcse(sigma_chain)
```

```
## $est
## [1] 5.475961
##
## $se
## [1] 0.002004615
```

3.5 Significance of the Parameters

To check if a result is statistically significant, it is common to examine whether the 95% credible interval (CI) includes the null hypothesis value or not. In the case of parameter estimates, if the CI does not include zero, it suggests that the parameter is statistically significant at the 0.05 level.



We can see that only one parameter is not significant, this information will be useful for the study of the complexity of our model.

4. Prediction of Math Scores

In this section, we will focus on predicting the Math Score based on our trained model. We will explore two different approaches for prediction.

4.1 Method 1: Using Mean Posterior Values

The first method involves using the mean posterior values of the model parameters to estimate the real parameters. We can then use these estimated parameters to compute the regression function and predict the Math Scores for each observation in the test dataset.

To implement this method, we calculate the predicted Math Scores as follows:

- We take the mean value of the alpha parameter from the posterior distribution.
- We multiply the mean value of each beta coefficient by the corresponding feature value of each test observation.
- We sum up these values to obtain the predicted Math Score for each test observation.

This process is repeated for each test observation, and the predicted Math Scores are stored in the “pred” variable.

```
pred <- numeric(dim(test)[1])

for (i in 1:dim(test)[1]) {
  pred[i] <- as.numeric(mean(scoresjags$BUGSoutput$sims.list$alpha)) +
             as.numeric(mean(scoresjags$BUGSoutput$sims.list$beta_gender) * test$gender[i]) +
             as.numeric(mean(scoresjags$BUGSoutput$sims.list$beta_race) * test$race.ethnicity[i]) +
             as.numeric(mean(scoresjags$BUGSoutput$sims.list$beta_parent_educ_level) * test$parental.level.of.edu
cation[i]) +
             as.numeric(mean(scoresjags$BUGSoutput$sims.list$beta_lunch) * test$lunch[i]) +
             as.numeric(mean(scoresjags$BUGSoutput$sims.list$beta_prep_level) * test$test.preparation.course[i])
+
             as.numeric(mean(scoresjags$BUGSoutput$sims.list$beta_writing) * test$writing.score[i])+
             as.numeric(mean(scoresjags$BUGSoutput$sims.list$beta_reading) * test$reading.score[i])

}
```

Let’s evaluate the RMSE and DIC.

We will employ the Deviance Information Criterion (DIC) to evaluate the performance of various models. DIC is a measurement that considers the number of parameters and is generated from the model’s deviation. It strikes a balance between model complexity and fit.

Different models’ DIC values can be computed and compared. Lower DIC values signify more effective models that offer a better fit to the data while taking model complexity into account.

```
rmse_first_method <- sqrt(sum((pred-target)^2)/length(pred))
rmse_first_method
```

```
## [1] 6.035321
```

```
dic_first_method <- scoresjags$BUGSoutput$DIC
dic_first_method
```

```
## [1] 5309.35
```

4.2 JAGS File-Based Approach

The second technique estimates the Math Scores for the test observations using the JAGS file. We make an approximation of the posterior predictive distribution of the Math Scores using the Markov chain samples from JAGS.

In each cycle, we extract the parameter values from the Markov chain samples. We determine the anticipated Math Scores for each test observation using these data.

We may approximate the posterior predictive distribution by averaging the projected Math Scores over all iterations, which enables us to make a variety of predictions for the students' Math Scores in the test dataset while taking into account the uncertainty represented by the Markov chain samples.

```
#let's re-write the dataset in more useful form
#let's insert also the test observation

jags_data_pred <- list(gender = train$gender,
                race = train$race.ethnicity,
                parent_educ_level = train$parental.level.of.education,
                lunch = train$lunch,
                prep_level = train$test.preparation.course,
                math_score = train$math.score,
                writing_score = train$writing.score,
                reading_score = train$reading.score,
                N = dim(train)[1],
                N_test = dim(test)[1],
                gender_test = test$gender,
                race_test = test$race.ethnicity,
                parent_educ_level_test = test$parental.level.of.education,
                lunch_test = test$lunch,
                prep_level_test = test$test.preparation.course,
                writing_score_test = test$writing.score,
                reading_score_test = test$reading.score)

parameters <- c("alpha", "beta_gender", "beta_race","beta_parent_educ_level",
                "beta_lunch","beta_prep_level","beta_writing","beta_reading", "sigma","pred")


#runs the MCMC chain
set.seed(123)
scoresjags_pred <- jags(data=jags_data_pred,
                        inits=inits,
                        parameters.to.save=parameters,
                        model.file="jags_mandara_pred.txt",
                        n.chains=2,
                        n.iter=50000,
                        n.burnin = 5000,
                        n.thin = 10)
```

```
pred_second_method <- scoresjags_pred$BUGSoutput$summary[11:dim(scoresjags_pred$BUGSoutput$summary)[1]-1,1]
pred_second_method <- sqrt(sum((target-as.numeric(pred_second_method))^2)/length(target))
pred_second_method
```

```
## [1] 6.040003
```

```
dic_second_method <- scoresjags_pred$BUGSoutput$DIC
dic_second_method
```

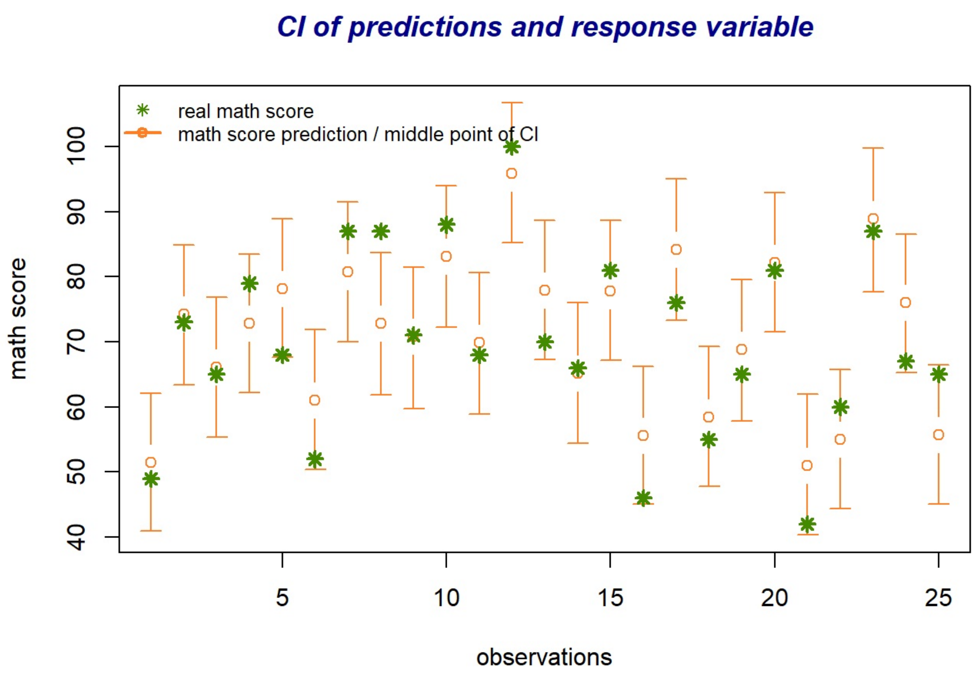
```
## [1] 5309.462
```

The MSE in this case is slightly worst, but as guessed basically the same.

Now that we have obtained point estimates for the Math Scores, we can also calculate and display credible intervals for these estimates using the JAGS output. This provides a measure of uncertainty around our point estimates.

```
##           lower_bound upper_bound
## pred[1]      55.16625    76.57115
## pred[2]      74.36431    96.03638
## pred[3]      70.73727    92.31244
## pred[4]      43.92863    65.55644
## pred[5]      46.75343    68.22891
## pred[6]      50.13839    71.78488
## pred[7]      55.71073    77.13138
## pred[8]      44.89073    66.14718
## pred[9]      80.68034   102.22461
## pred[10]     44.79594    66.29034
```

Now focusing on the predictions of the Math Scores, we will display the predicted scores along with their corresponding 95% credible intervals and the actual scores for comparison. This allows us to assess the accuracy of our predictions.



5. Additional Statistical Models

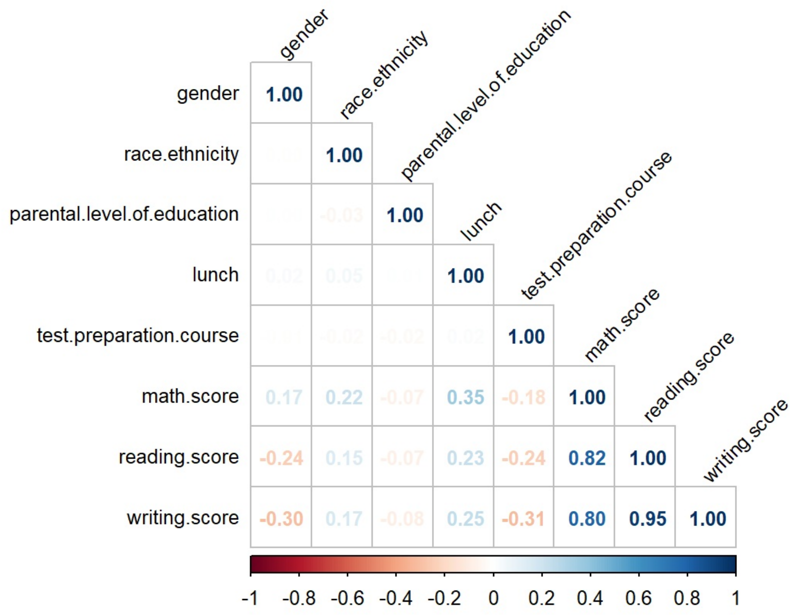
The performance of the model for predicting the Math Scores is examined using a variety of strategies in the sections that follow. These methods consist of:

- Examining the relationship between the target variable and additional characteristics.
- Examining the target variable's non-linear interactions with other features.
- Including multiplicative predictors (interaction variables) in the model.

The association between each variable and the Math Scores is examined at in order to choose which variables to include in our model. It makes sense to include elements with a high degree of correlation to the target variable.

5.1 Analyzing Correlation

To improve our model, we need to assess the correlation between each variable and the target variable, Math Score. By examining the correlation matrix and considering variables with a high correlation, we can identify potential predictors for our model.



Given this correlation plot and also considering the result obtained in the section **3.5**, let's the the RMSE value for the same model without considering $\beta_{parenteduclevel}$.

```
jags_data_mod <- list(
  gender = train$gender,
  race = train$race.ethnicity,
  lunch = train$lunch,
  prep_level = train$test.preparation.course,
  math_score = train$math.score,
  writing_score = train$writing.score,
  reading_score = train$reading.score,
  N = dim(train)[1]
)

# Define the parameters to be estimated
parameters_mod <- c("alpha", "beta_gender", "beta_race",
                "beta_lunch","beta_prep_level","beta_writing","beta_reading", "sigma")

# Initial parameter values for the MCMC sampler
inits_mod <- list(
  list(
    alpha=rnorm(1), beta_gender=rnorm(1), beta_race=rnorm(1),
    beta_lunch=rnorm(1), beta_prep_level=rnorm(1), beta_writing=rnorm(1),
    beta_reading=rnorm(1), sigma=runif(0,1000)
  ),
  list(
    alpha=rnorm(1), beta_gender=rnorm(1), beta_race=rnorm(1),
    beta_lunch=rnorm(1), beta_prep_level=rnorm(1), beta_writing=rnorm(1),
    beta_reading=rnorm(1), sigma=runif(0,1000)
  )

)

# Run the MCMC chain with one chain
set.seed(123)
scoresjags_mod <- jags(
  data = jags_data_mod,
  inits = inits_mod,
  parameters.to.save = parameters_mod,
  model.file = "jags_mandara_one_mod.txt",
  n.chains = 2,
  n.iter = 50000,
  n.burnin = 5000,
  n.thin = 10
)
```

Evaluating as before…

```
rmse_mod <- sqrt(sum((pred_mod-target)^2)/length(pred))
rmse_mod
```

```
## [1] 6.036421
```

```
dic_mod <- scoresjags_mod$BUGSoutput$DIC
dic_mod
```

```
## [1] 5307.442
```

So we can notice that the last model has a higher RMSE that the first model, but a smaller value of DIC.

5.2 Nonlinear Relationships

We can investigate the possibility of include nonlinear relationships even if it appears that the variables in our model largely have a linear relationship with the Math Score. Variables may need to be transformed, or nonlinear elements like squared or interaction terms may need to be included. We may discover if these nonlinear models offer more accurate predictions for the Math Score by evaluating their performance and contrasting it with the original linear model.

```
# Define the parameters to be estimated
parameters_im <- c("alpha", "beta_gender", "beta_race","beta_gender_lunch",
                  "beta_lunch","beta_prep_level","beta_writing","beta_reading", "sigma")

# Initial parameter values for the MCMC sampler
inits_1 <- list(
  list(
    alpha=rnorm(1), beta_gender=rnorm(1), beta_race=rnorm(1),
    beta_lunch=rnorm(1), beta_prep_level=rnorm(1), beta_writing=rnorm(1),
    beta_reading=rnorm(1), beta_gender_lunch=rnorm(1), sigma=runif(0,1000)
  ),
  list(
    alpha=rnorm(1), beta_gender=rnorm(1), beta_race=rnorm(1),
    beta_lunch=rnorm(1), beta_prep_level=rnorm(1), beta_writing=rnorm(1),
    beta_reading=rnorm(1), beta_gender_lunch=rnorm(1), sigma=runif(0,1000)
  )
)


# Run the MCMC chain with one chains
set.seed(123)
scoresjags_improve <- jags(
  data = jags_data,
  inits = inits_1,
  parameters.to.save = parameters_im,
  model.file = "jags_mandara_improve.txt",
  n.chains = 2,
  n.iter = 50000,
  n.burnin = 5000,
  n.thin = 10
)
```

```
pred_improve <- numeric(dim(test)[1])

for (i in 1:dim(test)[1]) {
  pred_improve[i] <- as.numeric(mean(scoresjags_improve$BUGSoutput$sims.list$alpha)) +
             as.numeric(mean(scoresjags_improve$BUGSoutput$sims.list$beta_gender) * test$gender[i]) +
             as.numeric(mean(scoresjags_improve$BUGSoutput$sims.list$beta_race) * test$race.ethnicity[i]) +
             as.numeric(mean(scoresjags_improve$BUGSoutput$sims.list$beta_lunch) * test$lunch[i]) +
             as.numeric(mean(scoresjags_improve$BUGSoutput$sims.list$beta_prep_level) * test$test.preparation.cou
rse[i]) +
             as.numeric(mean(scoresjags_improve$BUGSoutput$sims.list$beta_writing) * test$writing.score[i])+
             as.numeric(mean(scoresjags_improve$BUGSoutput$sims.list$beta_reading) * test$reading.score[i])

}

rmse_improve <- sqrt(sum((pred_improve-target)^2)/length(target))
rmse_improve
```

```
## [1] 6.033852
```

```
dic_improve <- scoresjags_improve$BUGSoutput$DIC
dic_improve
```

```
## [1] 5308.945
```

We can notice in this case that we have a smaller value of RMSE but a higher value of DIC. So we stick with the **scoresjags_mod** model(the one using the first model and removing the $\beta_{prepeduclevel}$ parameter)

6. Comparison with frequentist inference

To perform a comparative analysis with frequentist inference, it's possible to compare the estimates of the parameters and predictions obtained from the Bayesian model(let's evaluate the first one) with the results obtained from a frequentist model.

Remember that the first model was:

$$score_i = \alpha + \beta_{race} \cdot gender_i + \beta_{race} \cdot race_i + \beta_{parenteduclevel} \cdot parenteduclevel_i + \beta_{lunch} \cdot lunch_i + \beta_{prep\_level} \cdot preparlevel_i + \beta_{writing} \cdot writing_i + \beta_{reading} \cdot reading_i$$

Here's an example of this comparative analysis:

```
first_model<- lm( math.score ~  gender+ race.ethnicity + parental.level.of.education +
                  lunch + test.preparation.course + reading.score + writing.score , data=train)

first_model
```

```
##
## Call:
## lm(formula = math.score ~ gender + race.ethnicity + parental.level.of.education +
##     lunch + test.preparation.course + reading.score + writing.score,
##     data = train)
##
## Coefficients:
##                (Intercept)                       gender
##                 -12.741919                    13.067558
##             race.ethnicity  parental.level.of.education
##                   0.920191                     0.009434
##                      lunch      test.preparation.course
##                   3.494187                     3.137696
##              reading.score                writing.score
##                   0.320767                     0.633530
```

```
##                        frequentist bayesian
## alpha                      -12.742  -12.474
## beta_gender                 13.068   13.016
## beta_race                    0.920    0.913
## beta_parent_educ_level       0.009    0.003
## beta_lunch                   3.494    3.492
## beta_prep_level              3.138    3.093
## beta_reading                 0.321    0.321
## beta_writing                 0.634    0.631
```

Now let's predict on the test data and after let's show the difference between the previous prediction and the new one.

```
pred_lm <- predict(first_model, test)

sqrt(sum((pred_lm-target)^2)/length(pred_lm))
```

```
## [1] 6.036826
```

Let's print the first 15 predictions.

```
##         frequentis pred.   bayesian pred.   difference
## pred_1             65.73            65.74         0.02
## pred_2             85.16            85.12        -0.04
## pred_3             81.40            81.40         0.00
## pred_4             54.57            54.60         0.03
## pred_5             57.55            57.60         0.05
## pred_6             61.01            61.00        -0.01
## pred_7             66.32            66.32         0.01
## pred_8             55.27            55.30         0.03
## pred_9             91.46            91.37        -0.10
## pred_10            55.52            55.57         0.05
## pred_11            51.81            51.92         0.10
## pred_12             5.00             5.14         0.14
## pred_13            68.03            68.00        -0.03
## pred_14            53.30            53.36         0.06
## pred_15            51.47            51.48         0.01
```

7. Final observations

The comparison between the frequentist and Bayesian estimates for the parameters shows that there is generally good agreement between the two approaches. The parameter estimates are similar, indicating that both methods provide consistent results.

Examining the magnitude of the parameters can provide insights into the relative importance of different factors in influencing student performance. Larger magnitude parameters suggest stronger associations between the corresponding variables and the outcome (in this case, student math performance).

Here are some potential actions or strategies that can be explored to improve student performance:

- **Gender**: Given that gender significantly affects math test scores, you may want to consider gender-specific educational interventions or teaching practises that can help reduce the achievement gap between male and female students.
- **Race/Ethnicity**: Promoting inclusion and cultural understanding in the classroom is essential because race and ethnicity have a significant impact on students' academic performance. Programmes that help students from different racial and ethnic backgrounds, promote diversity, and address any potential biases or barriers should be implemented.
- **Lunch**: Access to healthy food can improve a student's academic performance and overall health. Explore programmes like school meal programmes or collaborations with community organisations to guarantee that kids, especially those from poor families, have access to nutritious food.
- **Preparation Level**: Helping students prepare for tests and exams can have a favourable effect on their performance. To assist students in improving their preparation methods, take into consideration providing study skills workshops, tutoring services, or online learning environments.