MODELING BRAIN SIGNALS USING NON-LINEAR REGRESSION

Introduction To Statistical Methods For Data Science (Course Work)

Subject Code : 7089CEM

Student Name : Suleiman Adamu Yakubu

Student ID : 11796778

Summary **2**

1. Introduction **3**

The Dataset 3

2. Task 1: Preliminary Data Analysis **4**

2.1 Time Series Plots 5

2.2 Distribution of the Variables 6

2.3 Correlation and Scatter Plots 8

2.4 Box Plots 10

3. Task 2: Regression **11**

3.1 Estimating Model Parameters. 12

3.2 Computing the Sum of Squared Errors for Each Model 13

3.3 Computing the Log-likelihood Function of the Models 14

3.4 Evaluating the Akaike Information Criterion and the Bayesian Information Criterion 15

3.5 Plotting the Distributions of the Model Errors 16

3.6 Selecting the Best Model 22

3.7 Testing the Model 22

3.7.1 Estimating the Model Parameters 22

3.7.2 Computing the Model's Predictions 23

3.7.3 Computing the 95% Confidence Interval 23

4. Task 3: Rejection Approximate Bayesian Computation (ABC) **24**

5. Discussion **28**

6. Conclusion **29**

7. Appendices **29**

7.1 Setting up 29

7.2 Task 1 Preliminary Data Analysis 29

7.2.1 Time series plots 29

7.2.2 Density distributions 30

7.2.3 Scatter plot and correlation 30

7.2.4 Box plot 30

7.3 Regression 31

7.3.1 Preparing the data 31

7.3.2 Estimating the model parameters 31

7.3.3 Estimating the residual errors (RSS) of each candidate model 31

7.3.4 Computing log-likelihood for each candidate model 32
7.3.5 Computing AIC and BIC for each candidate model 32
7.3.6 Plotting the error distributions 33
7.3.7 Performing the train-test split 34
7.3.8 Estimating model parameters from the training data 34
7.3.9 Computing the model's predictions from the test data 35
7.3.10 Computing the confidence intervals and plotting them 35
7.4 Rejection ABC 36

Summary

This report is meant as an accompaniment to the Introduction to Statistical Learning (7089CEM) Module. The report is meant to cover all that was done during the coursework and explain the motivation behind some of the choices made. The goal of the coursework is to examine and pick the best model from 5 different regression models. The models are meant to represent a dataset which contains data about input signals from an advertisement and the resulting brain signals measured with magnetoencephalography (MEG).

The report is split into three (3) parts which correspond to the three tasks undertaken in the coursework. The first task, and correspondingly, the first part of this report covers any preliminary data analysis carried out. In this section, time series plots of the data are created to view how the data changes over time. The data is then visualized using a density plot. Scatter plots and a correlation plot are also plotted to visualize how the output MEG signal is affected by the input audio. Finally box plots are used to check and view outliers.

The second part of the report is mainly concerned with analyzing the 5 candidate regression models and choosing the best one. The metrics used for this comparison are mainly Sum of Squared Errors (SSE), Log-likelihood function of each model, the Akaike and Bayesian Information Criteria (AIC and BIC), and finally visualizing the error distribution. The data is then split into training and testing data using a 70:30 split and the model parameters are again evaluated. We use these to determine, and plot the errors and confidence intervals.

The third and final part of this report makes use of Approximate Bayesian Computation (ABC) on the best model selected from the previous task. ABC is used to compute the posterior distribution of the two parameters of the best model that have the largest absolute value. The marginal and joint distributions of the points generated are also plotted.

This project was carried out using R in RStudio. The visualizations were primarily carried out using the Corrplot, GGPlot2 and GGExtra libraries. All the code used to carry out the task is attached in the appendix.

1. Introduction

Companies have long sought the means to easily convert attention from potential customers into sales. One way to do that will be to be able to stimulate potential customers' interests in products using advertisements.

However, it is not definitely known which kinds of advertisements stimulate potential customers' interests the most. This report is meant to take a step in that direction. Using statistical learning techniques, a dataset of audio advertisements in different tones of voice and the resulting brain reactions (measured using magnetoencephalography) are measured. With the aid of these techniques, we should be able to discover which voice tones elicit the most responses from potential customers.

The Dataset

The dataset used for this report is meant to simulate a neuromarketing experiment whereby participants listen to audio advertisements narrated by 2 different types of voices (a neutral and an emotional voice) and then their brain response is measured using magnetoencephalography (MEG). The dataset consists of 4 variables contained in 3 csv files. These variables include the input audio signals (x1), the tone of the audio (x2), a time series variable (time), and the output MEG signal (y). The first file, called 'X.csv' contains the two 'X' variables. The time variable is stored in the 'time.csv' file, and the output MEG signal is stored in the 'y.csv' file. The table below shows a description of the dataset.

Variable name	Variable format	Description
x1	Real number	Represents the amplitude of the audio signals. It is the first column of the 'X.csv' file
x2	Categorical variable (0,1)	Represents whether the audio signal is in a neutral voice or an emotional one. It is the second column of the 'X.csv' file
time	Real number	Represents the time at which the input or output measurement is taken. This variable is the only column in the 'time.csv' file
y	Real number	Represents the value of the MEG signal of the brain when the participant listens to a particular audio signal. It is the only column in the 'y.csv' file.

Table 1: The dataset description/dictionary

2. Task 1: Preliminary Data Analysis

After the data is imported into the RStudio environment using the *read.csv()* function, the imported data is then used to populate appropriately named matrices. This is done to ease the mathematical calculations to be done on the data. The data is then subjected to a variety of analyses which will be enumerated in this section of the report. The analyses are carried out to investigate the hypothesis of the experimenters; "The emotional narrative should evoke an increased brain response." The code used for this task is in Appendix 7.2

2.1 Time Series Plots

In this section, the variables are plotted against the time data to better view how the input audio and the output MEG signals change with time. It is instructive to note that the entire audio sample is 20 seconds long, with the neutral voice occupying the first ten seconds and the emotional voice occupying the last ten. Refer to Appendix 7.2.1 for the code.

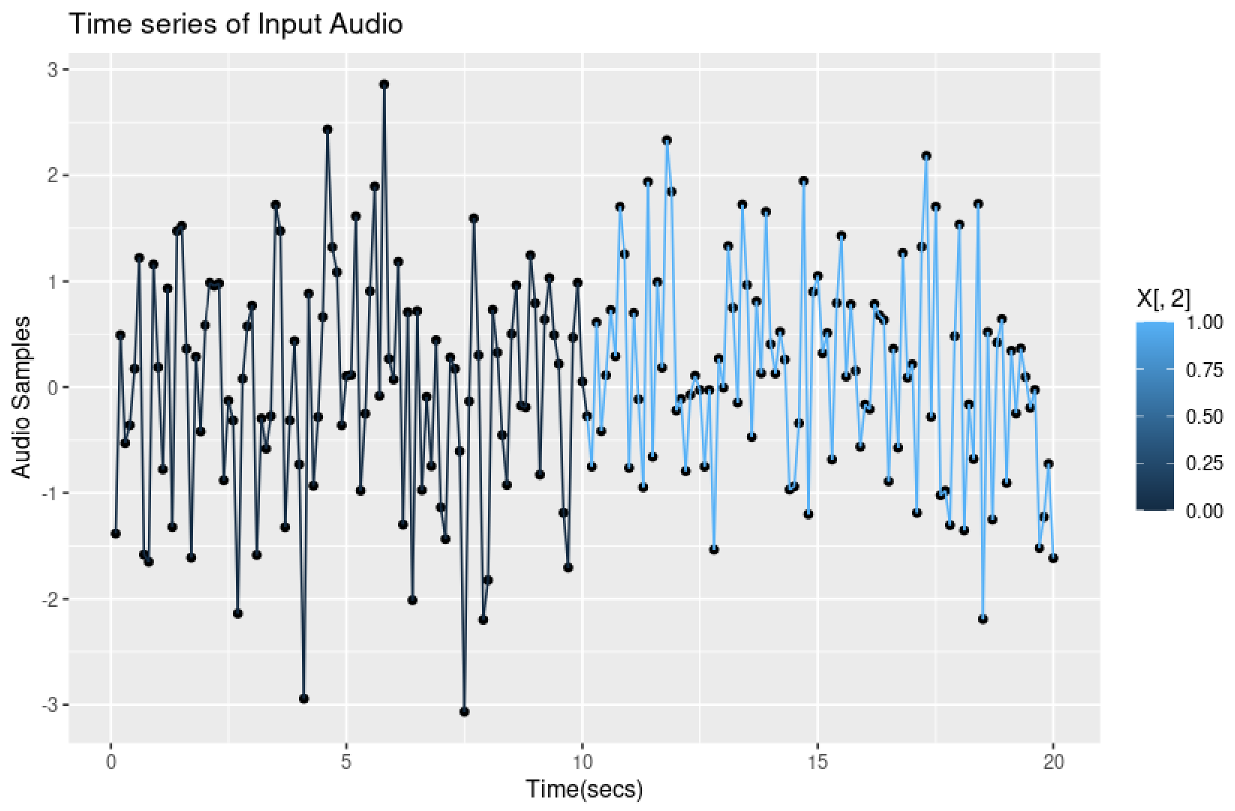

Fig 1: Time series plot of audio signal

From the above plot, it is possible to visually notice that the amplitude of the emotional audio does not oscillate as wildly as the neutral. For any conclusions however, more investigation is needed.

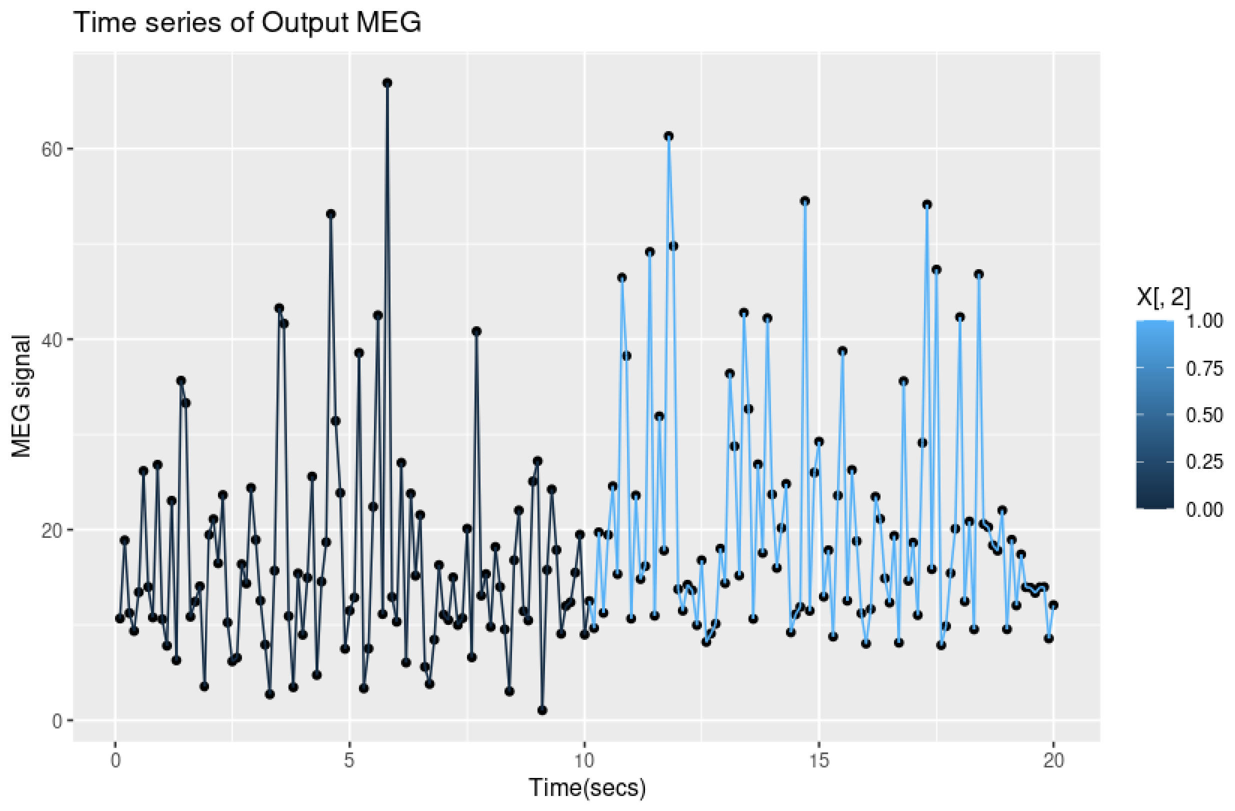

Fig 2: Time series plot of output MEG signals

As the legend shows in the images, the light blue color represents when the audio category is emotional, and the dark blue represents the neutral tone. Not much can be gleaned from this plot however, so the density distribution of the variables is examined.

2.2 Distribution of the Variables

This section is meant to show the distribution that the variables follow. The distribution gives a feel for the mean and median values, and shows which values a variable is more likely to assume.

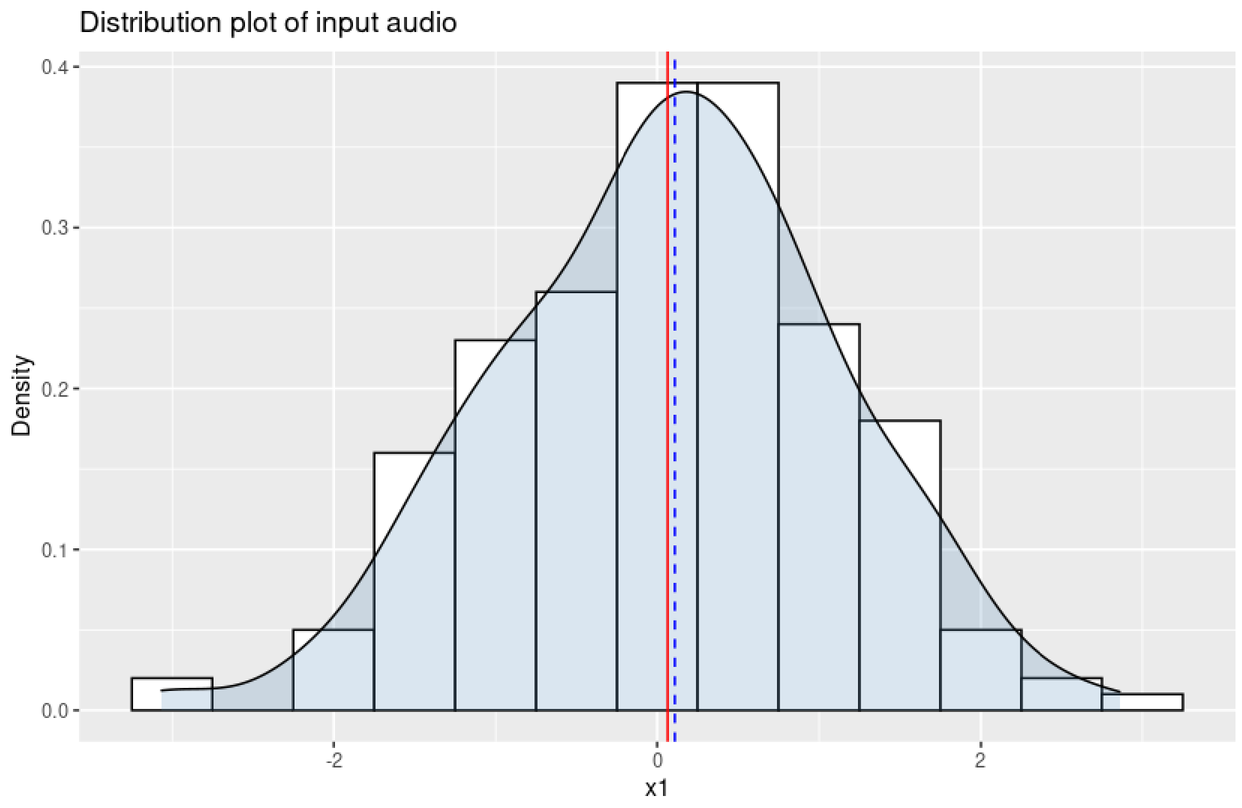

Fig 3: The density distribution of the input audio signal

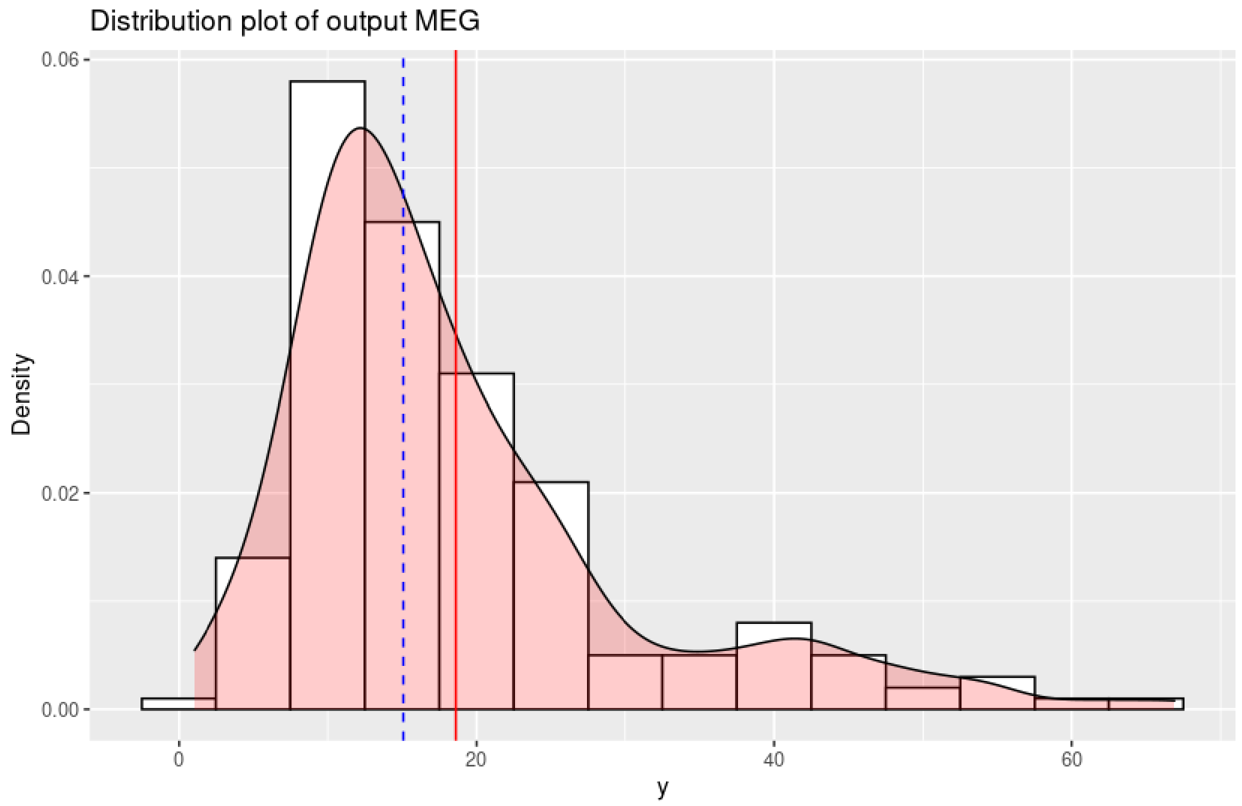

Fig 4: The density distribution of the output MEG signal

The distribution of the input audio visually shows that it follows an approximately Gaussian distribution. The output density distribution on the other hand is visibly skewed to the right. Refer to Appendix 7.2.2 for the code to generate the distributions.

2.3 Correlation and Scatter Plots

This section shows the correlation (i.e. how one variable is affected by changes in another) between the input variables (i.e. x1 and x2) and the output MEG signal (i.e. y). This is achieved using the *corr()* function of R. Using this function, the correlation between the x1 variable and the y variable was calculated as 0.7653645, and the correlation between the x2 variable and the label (y) as 0.1704505. This means that the audio category (x2 variable) does not have much of an influence on the MEG output (y variable) while the audio amplitude is strongly correlated with the MEG since the correlation coefficient is larger than 0.7. The correlation plot is shown below:

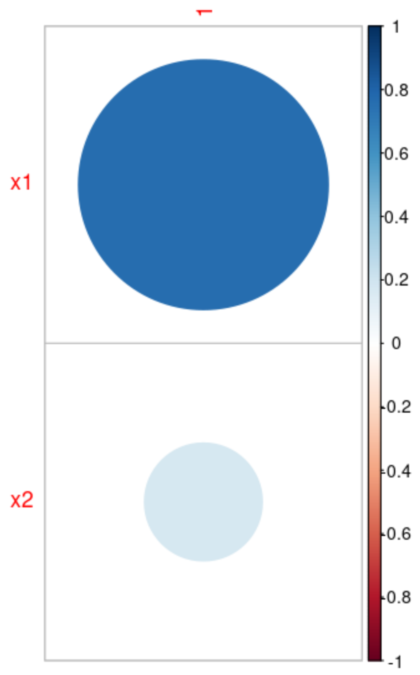

Fig 5: Correlation plot of x1 and x2 with y.

The scatter plot is also used to show the relationship between variables. In this case, the plot was created using the GGPlot2 library in R.

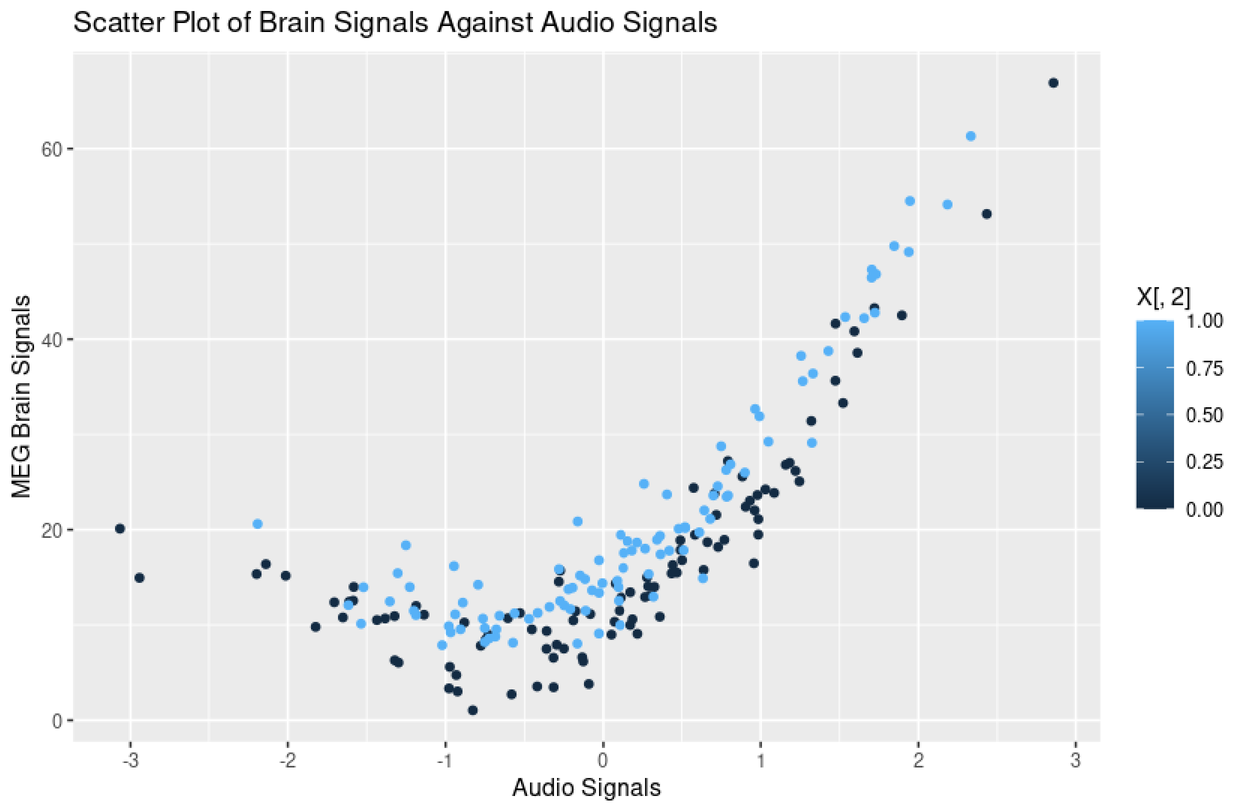

Figure 6: Scatter plot of audio against MEG signals

The plot implied that the emotional audio created a slightly higher MEG response than the neutral one. The plot also confirms what was seen from the correlation coefficient. The MEG signal mostly follows the audio signal. For the most part, it increases when the audio amplitude increases. Refer to appendix 7.2.3 for the code

2.4 Box Plots

The box plot is meant to show how the values of a variable are spread out within a range. In other words, the box plot visually shows the distribution of a variable as well as clearly showing any outliers. The box plot in this report was created using the GGPlot2 library.

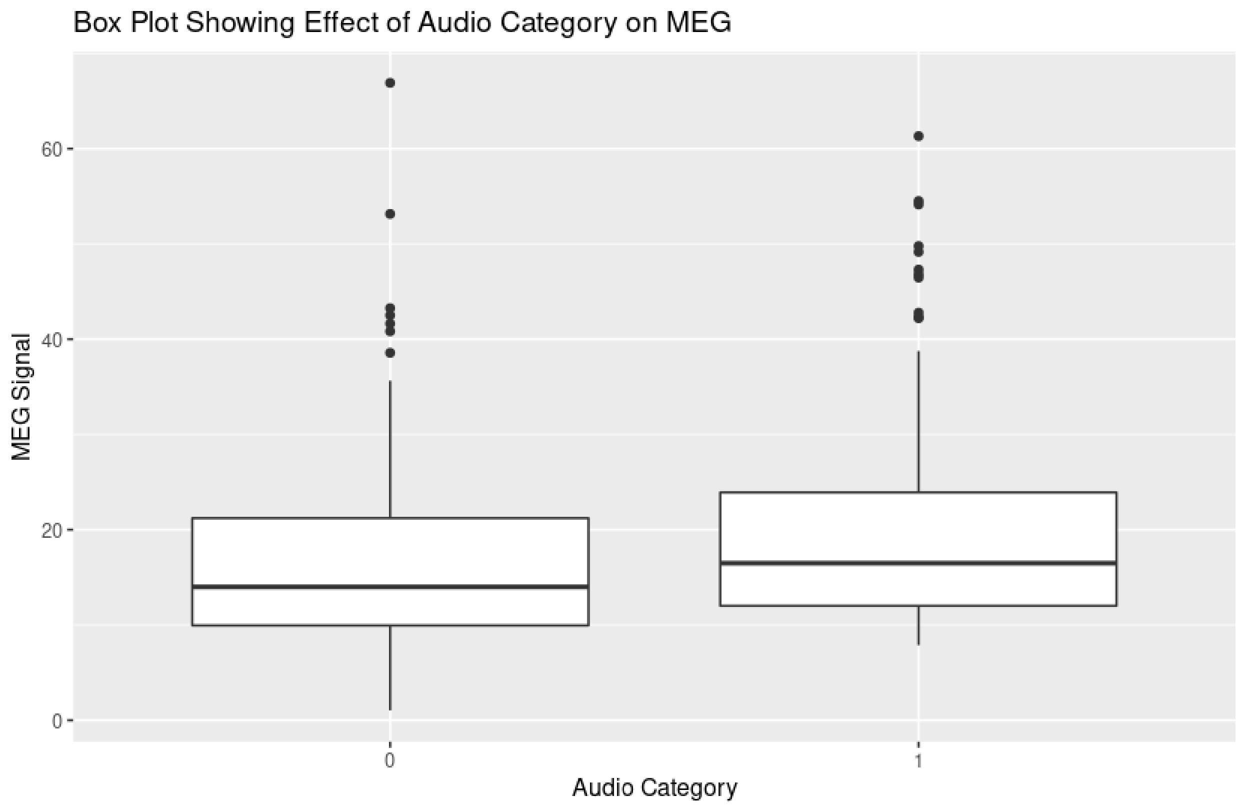

Figure 7: Box plot of output MEG (y)

The box plot further confirms what was shown by the correlation and scatter plots. It suggests that the emotional audio produces a slightly increased MEG signal, although this is not always the case, as is shown by the outliers in the box plot. Refer to Appendix 7.2.4 for the code.

3. Task 2: Regression

In this section of the report, the five candidate models are investigated and the best one is selected. The best model in this case is the one that truly represents the relationship between the variables of the dataset and scores the most on the various metrics used to evaluate them such as Sum of squared errors, AIC, BIC, etc. To do this, the models have to be represented accurately in R. This is done by building matrices that accurately represent the models. The models are as follows:

Model 1: $y = \theta_{bias} + \theta_1 x_1^3 + \theta_2 x_1^5 + \theta_3 x_2 + \varepsilon$

Model 2: $y = \theta_{bias} + \theta_1 x_1 + \theta_2 x_2 + \varepsilon$

Model 3: $y = \theta_{bias} + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^4 + \theta_4 x_2 + \varepsilon$

Model 4: $y = \theta_{bias} + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3 + \theta_4 x_1^5 + \theta_5 x_2 + \varepsilon$

Model 5: $y = \theta_{bias} + \theta_1 x_1 + \theta_2 x_1^3 + \theta_3 x_1^4 + \theta_4 x_2 + \varepsilon$

To represent the models accurately, they should be represented as a system of equations that can be solved using linear algebra. This means that the models can be represented as the following:

$$y = X * \theta$$

With y being the vector of MEG outputs, X being a matrix that corresponds to the x values of the model (x_1 being audio inputs and x_2 being the audio category), θ being a vector of the model parameters, and '*' representing matrix multiplication. The ε, which represents noise, is not added to the matrices because the output MEG already has Gaussian noise added to it. As such, we can safely ignore the ε parameter.

NB: Since the bias parameter does not have an accompanying x variable, it is represented in the X matrix as a column of ones.

3.1 Estimating Model Parameters.

To accurately estimate the model parameters, the method of least squares is used. In the method, the parameter vector 'θ', is calculated using the following formula:

$$\boldsymbol{\theta} = (X^T X)^{-1} X^T y$$

In R, the X matrix is built out by using the *cbind()* function. This function builds up a matrix by appending columns of specified values together. The generated matrix is built according to the specifications of the model being investigated. For instance, if the table given below represents hypothetical entries of a row in the dataset, then the corresponding row of the X matrix for the first model will be shown as below;

x1	**x2**	**y**
1.234	1	6.567

Table 2: Hypothetical values of a row in the dataset

$\mathbf{x_1^3}$	$\mathbf{x_1^5}$	$\mathbf{x_2}$	$\mathbf{x_0}$
1.879080904	2.861381721	1	1

Table 3: A row of the X matrix generated from hypothetical values using model 1.

The table below shows the model parameters by model. The code to generate them can be found in Appendix 7.3.2

	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**
θ_{bias}	16.9357507	16.635590	10.1736961	10.85328261	14.1317254
θ_1	5.1712728	9.629107	8.5521613	9.35397322	8.0794581
θ_2	-0.5373253	2.813987	6.2469171	4.64696205	0.3144501
θ_3	2.2020179	___________	-0.2829631	-0.57792162	0.5592050
θ_4	___________	___________	4.1598833	0.07479626	3.9681455
θ_5	___________	___________	___________	4.34321644	___________

Table 4: The model parameters of each of the candidate models

3.2 Computing the Sum of Squared Errors for Each Model

The sum of squared errors is a measure of how much the predicted value of the model differs from the actual label that is in the data. It is computed by obtaining the difference between these two values, squaring the difference to eliminate negative signs, and then summing over all the differences. This gives us a measure of how close to the label value given in the data the model prediction is. Hence a smaller value is generally more desirable. Formally, the sum of squared errors (residual error) of the model is given by:

$rss = \sum_{1}^{n} (y - y_{hat})^2$, where y is the target value given in the dataset and y_{hat} is the value predicted by the model. The code for this part is found in Appendix 7.3.3

	RSS
Model 1	11825.4213529512
Model 2	11238.9496385494
Model 3	1636.16760677224
Model 4	1902.06285599658
Model 5	4928.31206100396

Table 5: The residual error of all the models.

3.3 Computing the Log-likelihood Function of the Models

The likelihood of a variable simply means the odds of getting the data, given the model. It is an estimation of how well the model fits the data.

	Log-likelihood
Model 1	-691.757908679505
Model 2	-686.671291223832
Model 3	-493.968420272728
Model 4	-509.026653156095
Model 5	-604.232406896069

Table 6: The log-likelihood of the models

The log-likelihood is given by the following formula:

$$log\ L(\theta |\ D)\ =\ \sum_{i=1}^{n} log\ p(x_i\ |\theta)$$

The log-likelihood is used instead of the likelihood because the log makes the products become sums and exponents become products. This is done because when evaluating the likelihood of a large number of values, the likelihood becomes progressively smaller until it underflows (becomes too small to be accurately represented). Thus the log-likelihood neatly escapes that problem. Please refer to appendix 7.3.4 for the code.

3.4 Evaluating the Akaike Information Criterion and the Bayesian Information Criterion

The Akaike and Bayesian Information Criteria are measures of the relative quality of statistical models. They show how good a statistical model is in relation to other models. This means they are a good means to evaluate candidate models and select the best one. The below table shows the AIC and BIC of each model.

	AIC	**BIC**
Model 1	1391.51581735901	1404.7090868252
Model 2	1379.34258244766	1389.23753454731
Model 3	997.936840545456	1014.4284273782
Model 4	1030.05330631219	1049.84321051148
Model 5	1218.46481379214	1234.95640062488

Table 7: The AIC and BIC values of all the models.

The formula for the AIC and BIC are given as follows:

$$AIC\ =\ 2k\ -\ 2ln(L_{max})$$

$$BIC\ =\ kln(n)\ -2ln(L_{max})$$

where k is the number of parameters of the model and L_{max} is the maximum value of the likelihood function.

The first part of the formula for both the Akaike and Bayesian Information Criteria is a penalty for the complexity of the model. The more parameters the model has, the more complex it is. The second part of the formulae rewards the goodness of the fit of the model. For AIC and BIC, the lower the value, the better the model, this means that when evaluating a group of candidate models, the one with the lowest AIC or BIC score is most likely the best. Refer to Appendix 7.3.5 for the code.

3.5 Plotting the Distributions of the Model Errors

The distribution of the model errors were investigated to ascertain if they were close to Gaussian. This was done using various metrics such as QQ Plots, a histogram (distribution), and a Shapiro-Wilk normality test. The results are displayed below:

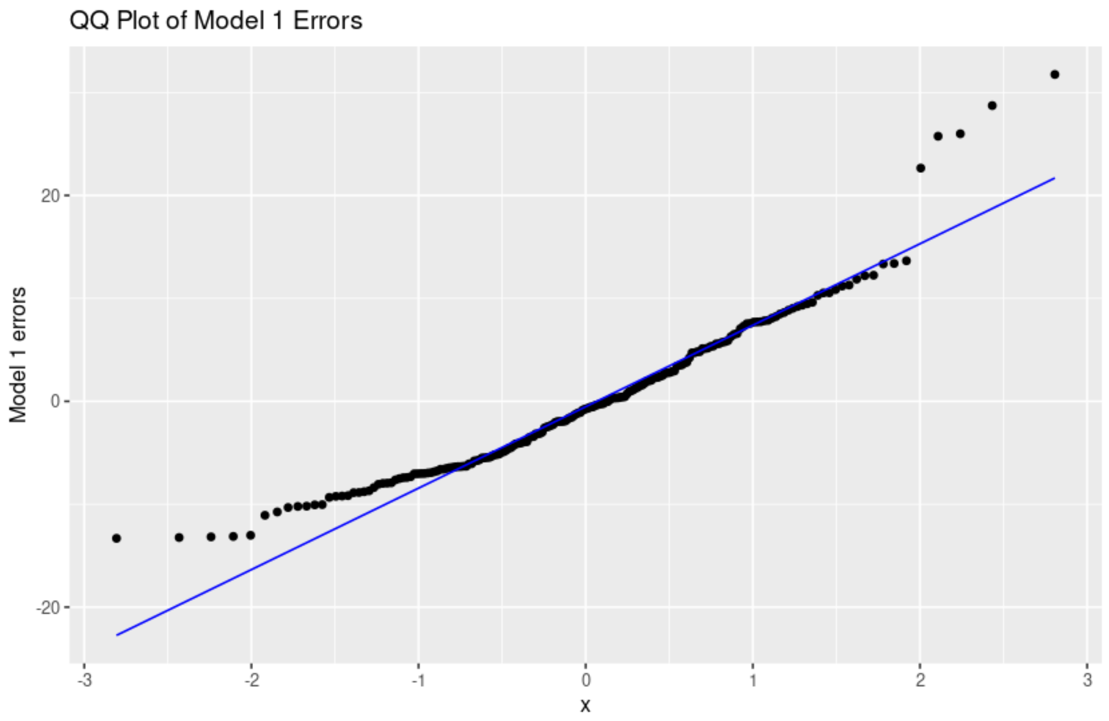

Figure 8: QQ Plot of Model 1 errors

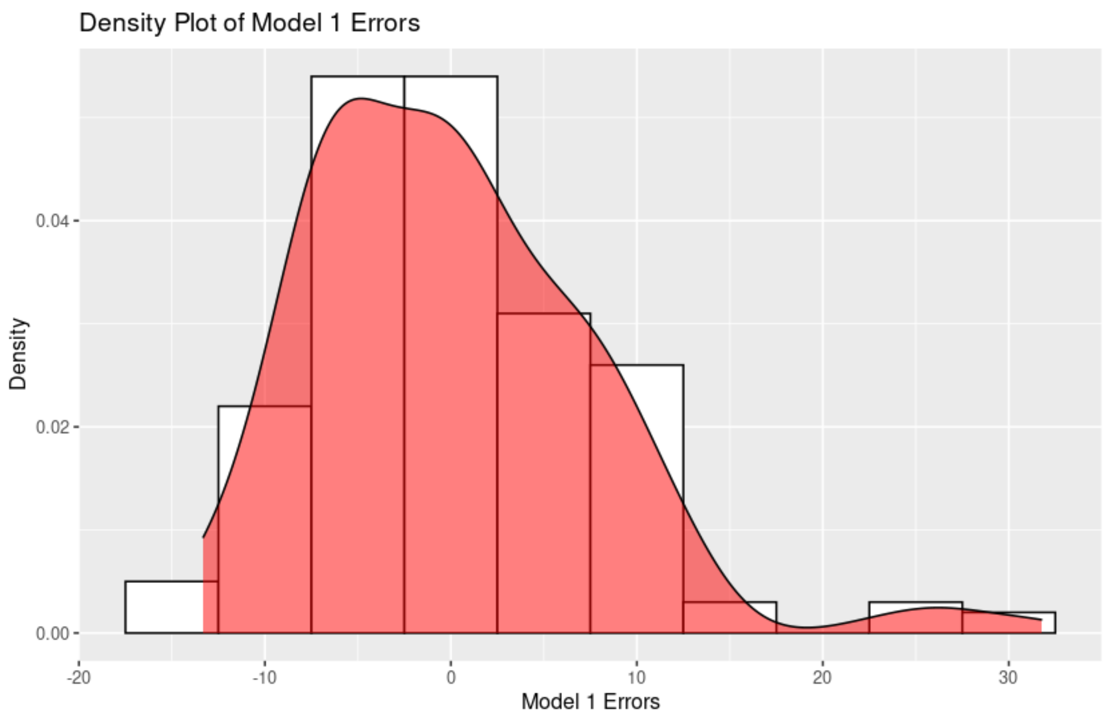

Figure 9: Density distribution of Model 1 errors.

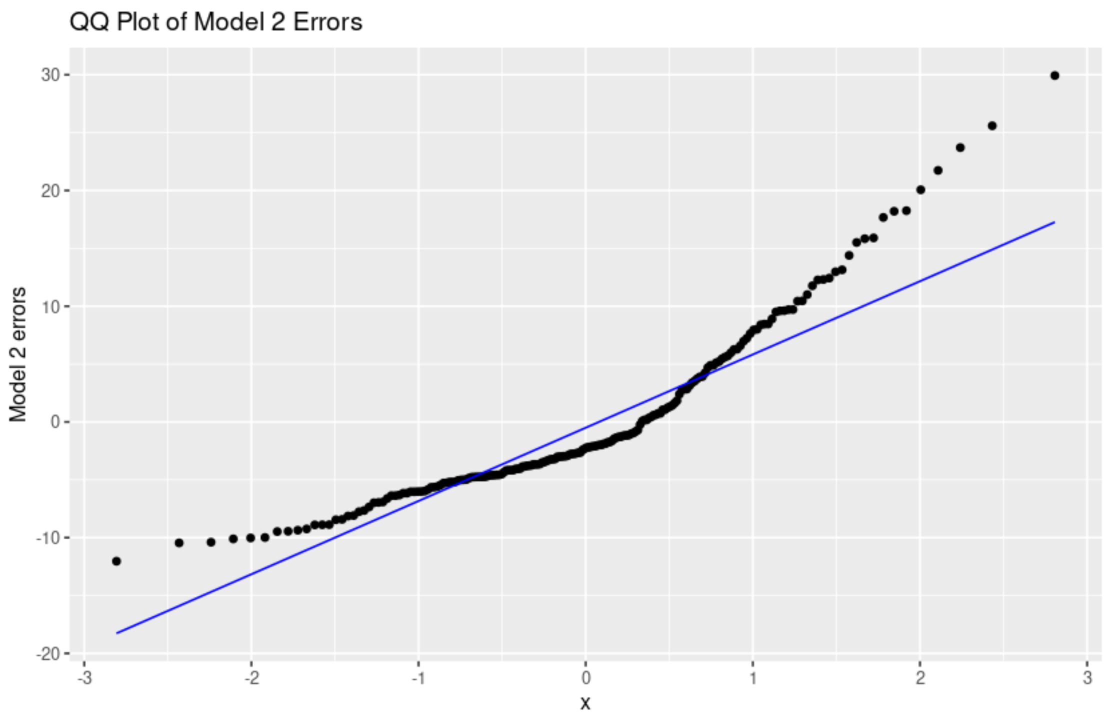

Figure 10: QQ Plot of Model 2 errors

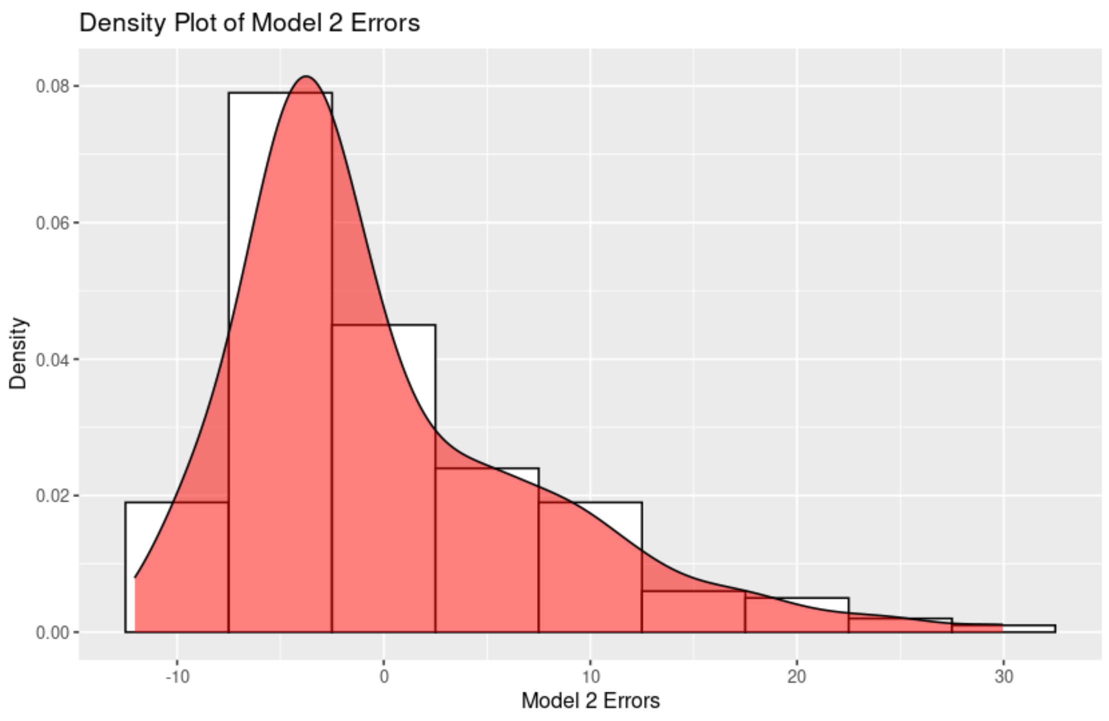

Figure 11: Density plot of model 2 errors

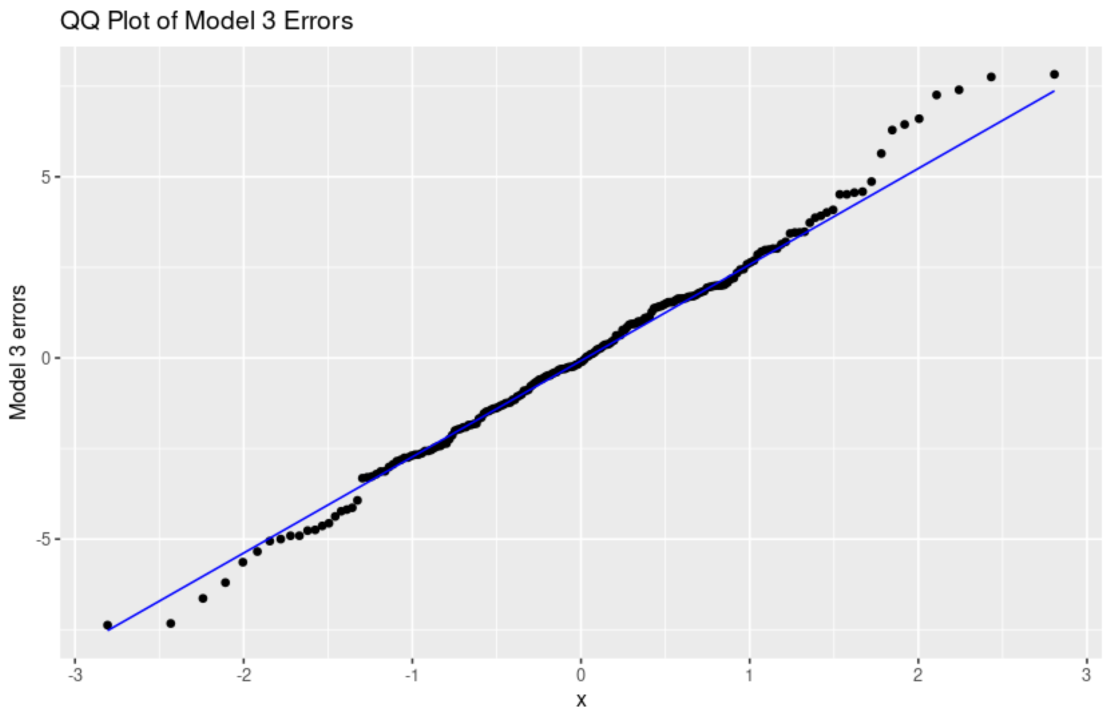

Figure 12: QQ Plot of Model 3 errors

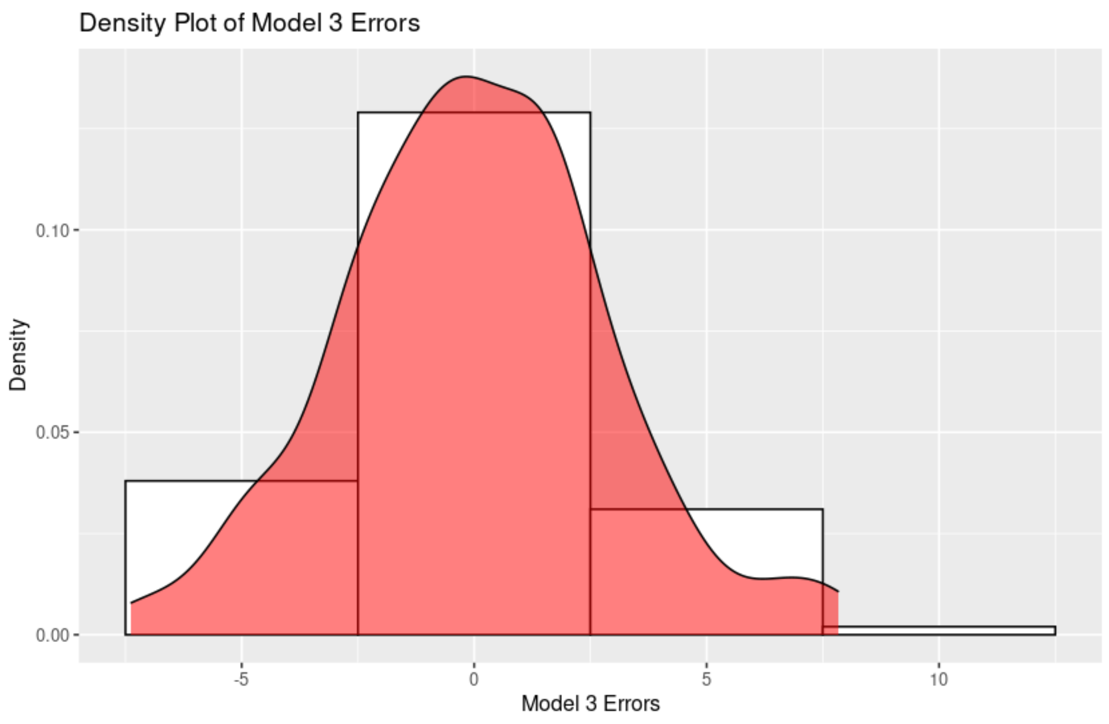

Figure 13: Density plot of model 3 errors

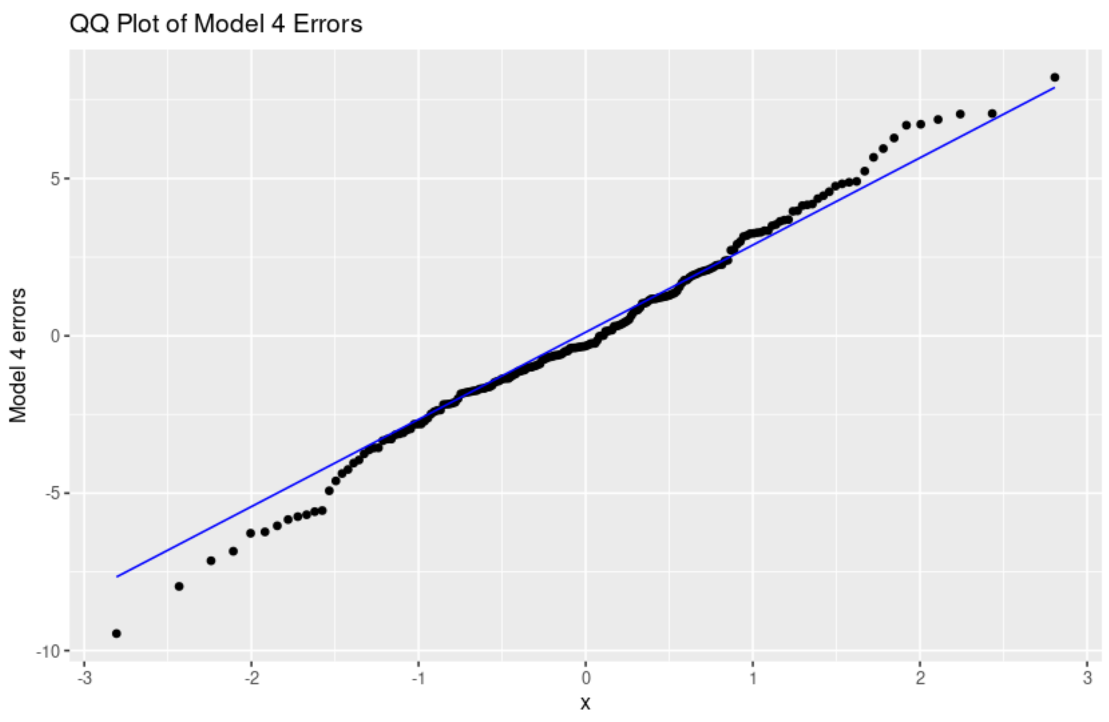

Figure 14: QQ plot of model 4 errors

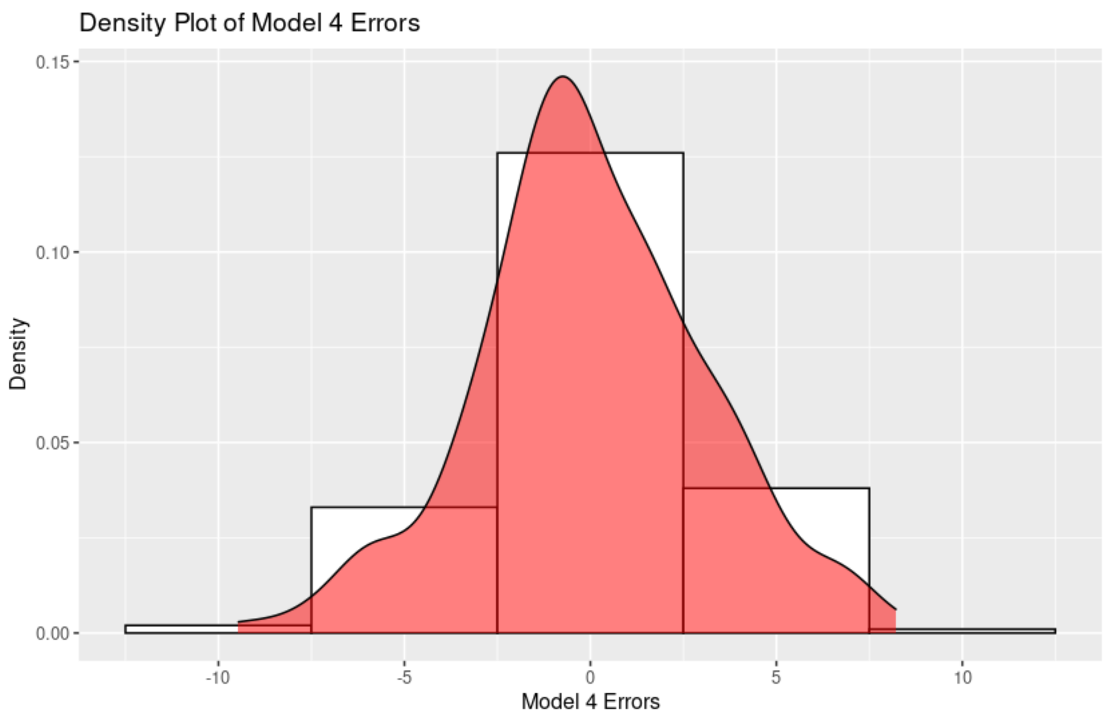

Figure 15: Density plot of Model 4 errors

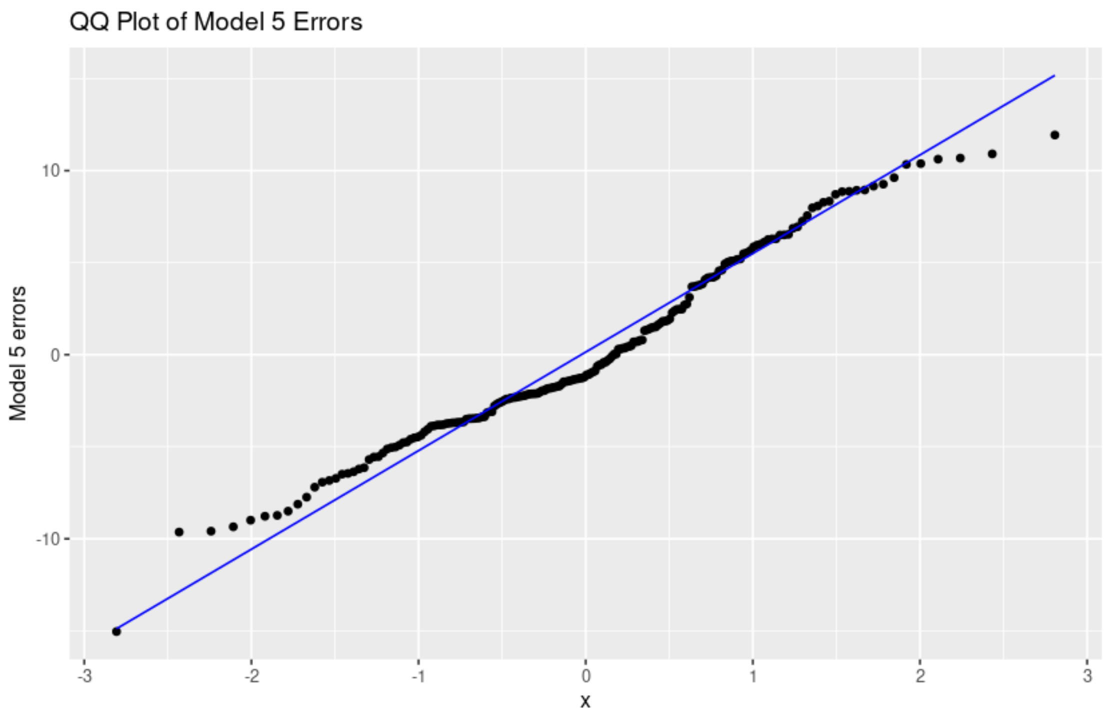

Figure 16: QQ plot of model 5 errors

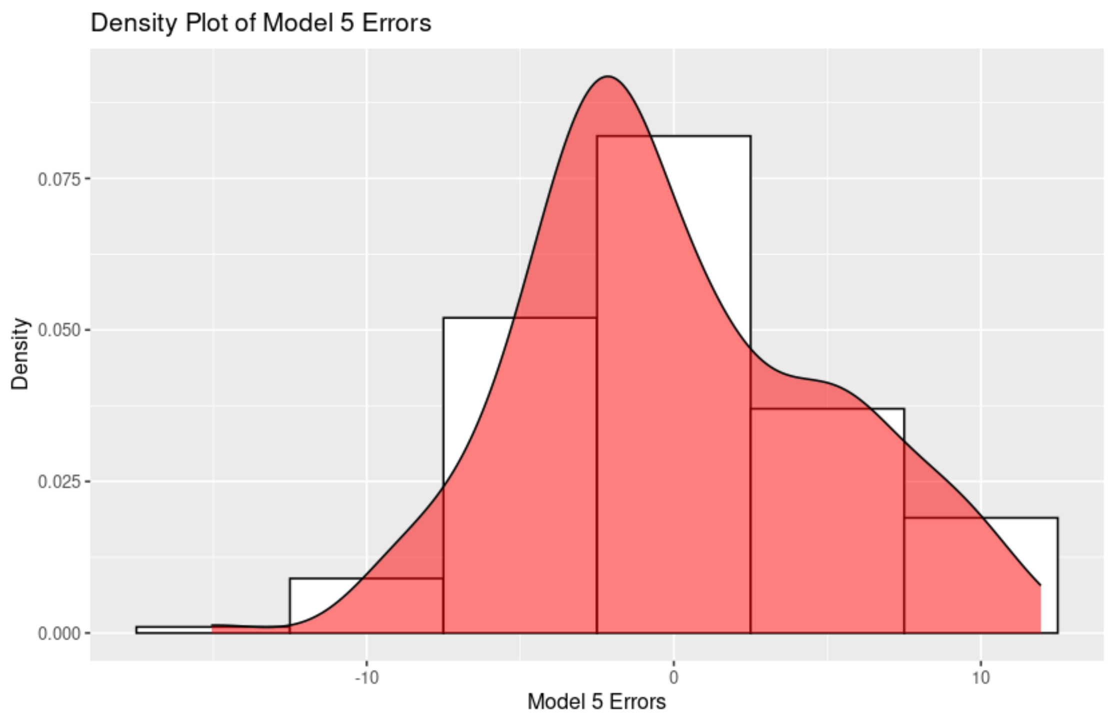

Figure 17: Density plot of model 5 errors

	W	**p-value**
Model 1	0.93663	1.179e-07
Model 2	0.89258	8.681e-11
Model 3	0.99227	0.3713
Model 4	0.99283	0.4384
Model 5	0.97763	0.00278

Table 8: The results of the Shapiro-Wilk test

The Shapiro-Wilk test rejects the hypothesis that a distribution is Gaussian if the p-value < 0.05. Thus, by visually inspecting the QQ Plots, density distributions, and based on the results obtained in this test, it is concluded that only the third and fourth models have error distributions that are approximately Gaussian. Refer to Appendix 7.3.6 for the code.

3.6 Selecting the Best Model

After thorough investigation and comparative analysis, Model 3 is selected as the best model out of all the other candidate models. The justification for that is as follows:

1. The model has the lowest sum of squared errors value. This means that the model's predictions were more accurate overall.
2. The model had the lowest AIC and BIC values. Since these values are used to compare models and select the best one, this shows that the Model 3 is better than the others.
3. The distribution of errors from Model 3 was approximately Gaussian. This is important because the output MEG variable had Gaussian noise added to it. This implies that the prediction errors of any appropriate model should also follow a Gaussian distribution.
4. The model had the highest Log-likelihood values. This means the model is the most likely one to produce with the data given.
5. The model visually produces the best fit to the data.

3.7 Testing the Model

After the best model is selected, the model is subjected to testing via the procedures outlined below.

The different matrices in the dataset are combined to form one matrix, the dataset is then subjected to a train-test split in a ratio of 70:30. This is done randomly. Please reference Appendix 7.3.7 for codes on this procedure.

3.7.1 Estimating the Model Parameters

This is similar to the procedure of task 2.1 using the formula outlined below:

$$\boldsymbol{\theta} = (X^T X)^{-1} X^T y$$

However, this formula is only going to be applied to the training part of the dataset. The supporting code is in Appendix 7.3.8.

3.7.2 Computing the Model's Predictions

Since the model parameters have been discovered in the previous step, the prediction is then evaluated using this formula:

$$y = X * \theta$$

Where y is the output/prediction, X is the relevant X matrix for the model, '*' refers to matrix multiplication, and θ refers to the vector of model parameters. This operation is carried out on the test portion of the dataset. Please refer to Appendix 7.3.9 for the relevant code.

3.7.3 Computing the 95% Confidence Interval

The 95% confidence interval of a variable is a range of values that is 95% likely to contain the mean. The 95% confidence interval can be computed using the formula below:

$$CI = \bar{x} \pm z \frac{s}{\sqrt{n}}$$

Where, $\bar{x}$ is the mean of the sample, z is the confidence interval value (1.96 for 95%), s is the standard deviation of the sample, and n is the sample size. The code to compute and plot the confidence interval can be found at Appendix 7.3.10.

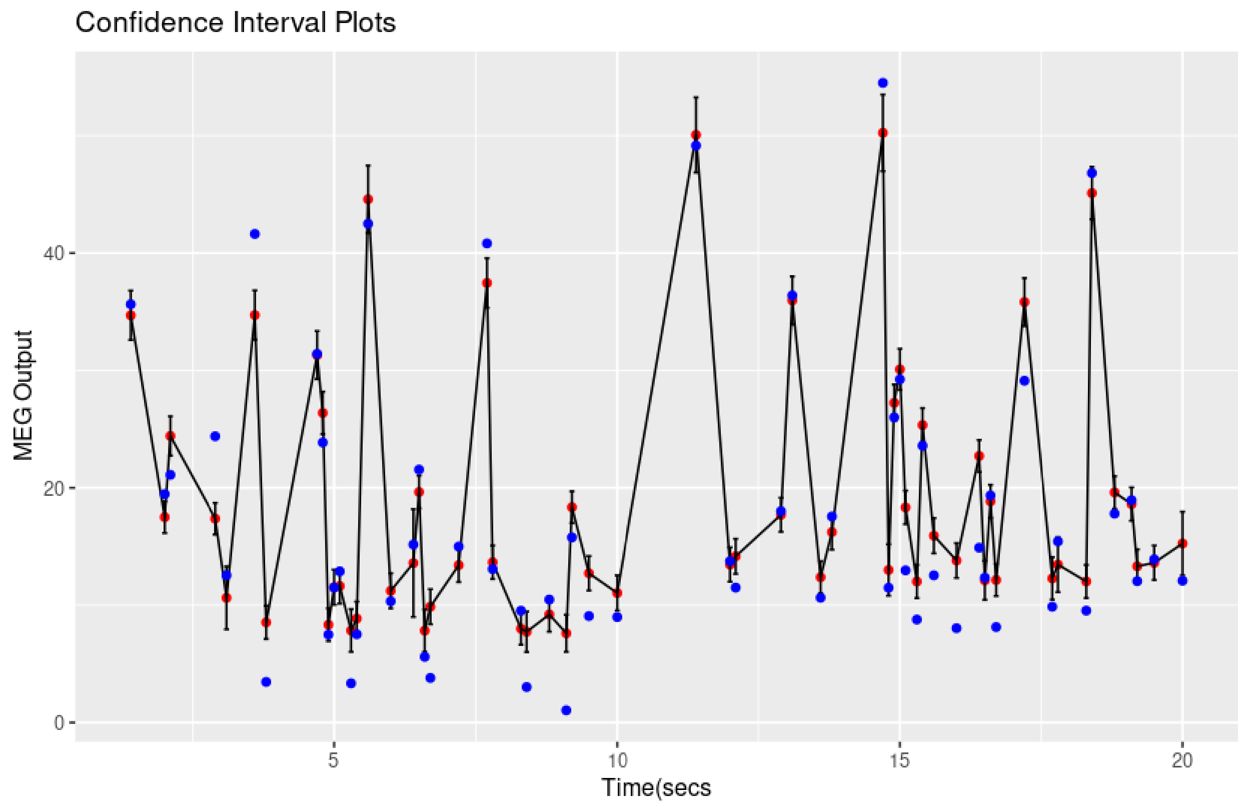

Figure 18: Plot showing the confidence interval of the prediction along with the data points

4. Task 3: Rejection Approximate Bayesian Computation (ABC)

ABC is used to evaluate the posterior distribution of model parameters without having to compute the likelihood. The procedure to perform the rejection ABC is outlined below:

1. Select two parameters from the best model, These parameters are to be the ones with the largest absolute values. For this project, those were θ_{bias} and θ_1.
2. Using a normal distribution as a prior distribution (the range of values will be given as $\theta_i - 0.5\,(\theta_i), \theta_i + 0.5\,(\theta_i)$), draw Monte Carlo samples around the two parameters. In this case, 3000 points were generated.

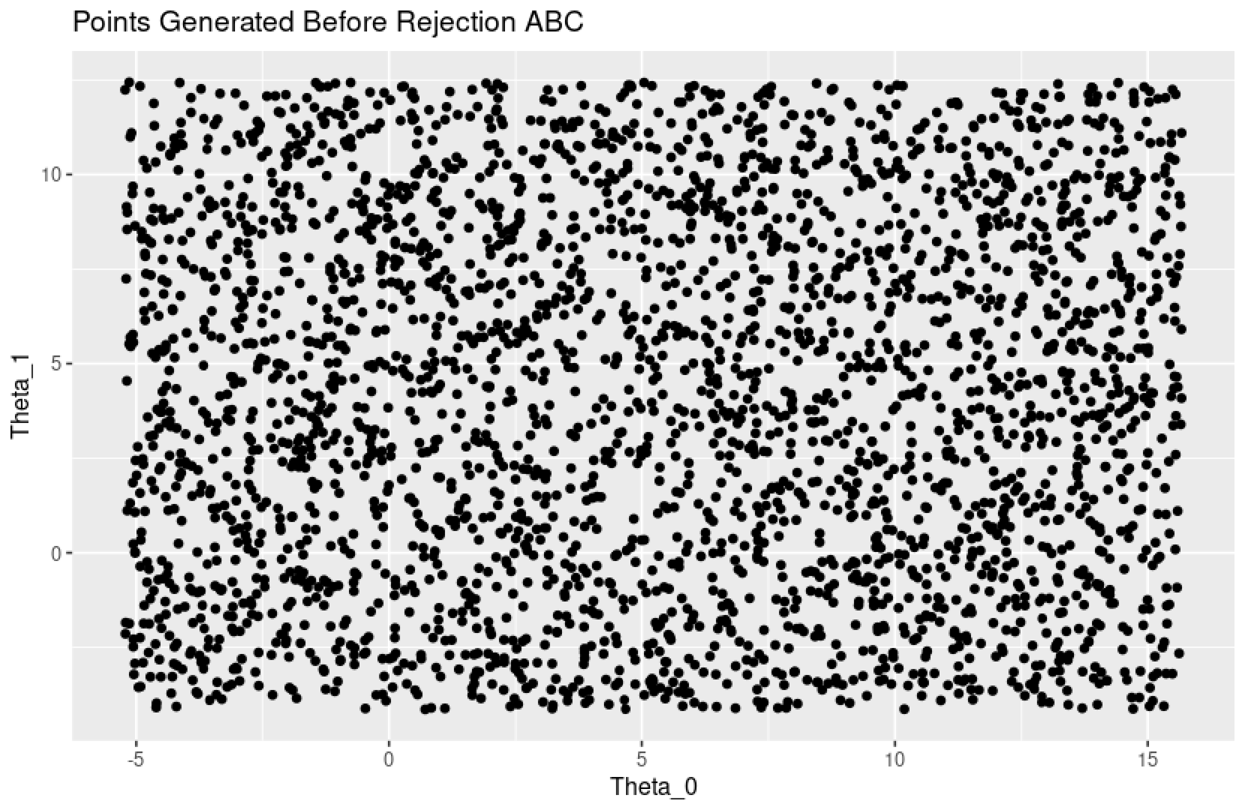

Figure 19: The points generated from the prior distribution

3. Each point in the Monte Carlo simulation will represent a value of the two parameters. Keeping the other parameters of the model constant, use the new values of the parameters to compute the model prediction and the residual errors.
4. If the residual error is greater than some threshold value (for this project, the threshold value was 2000), reject it. If not, accept it.

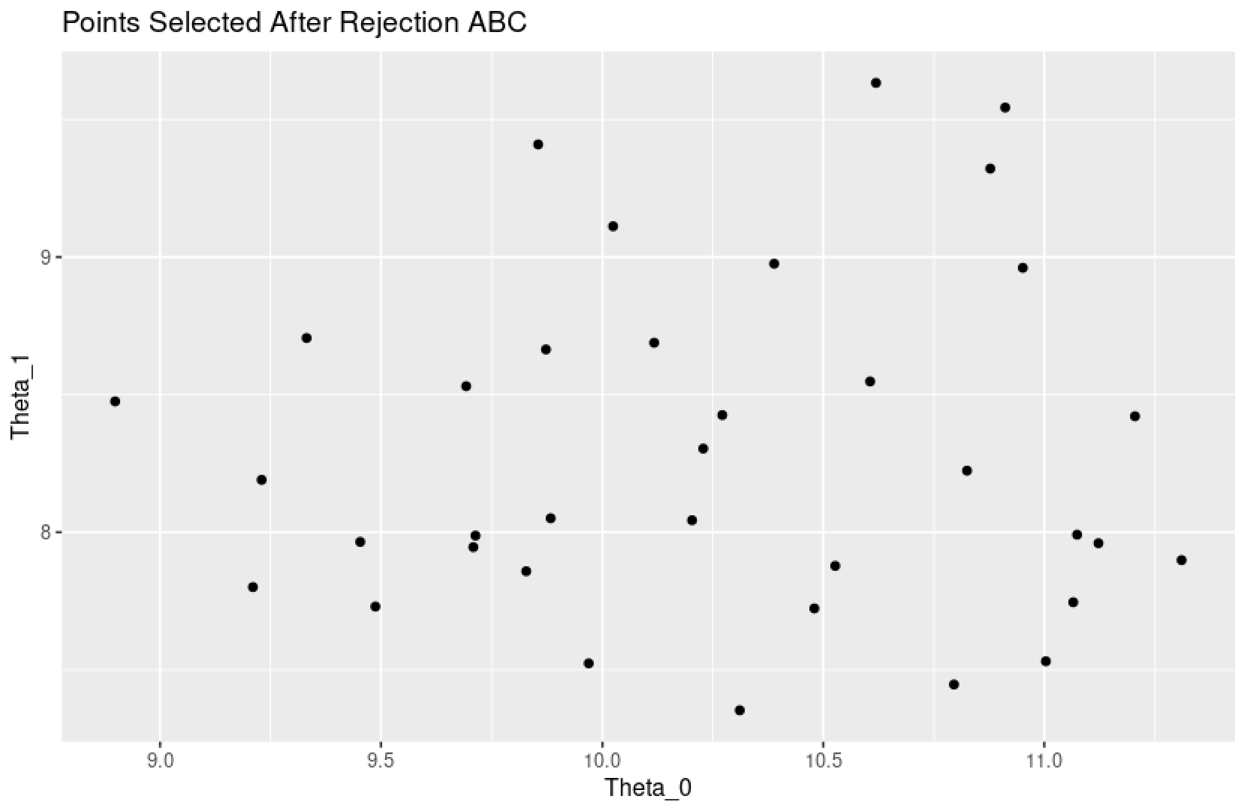

Figure 20: The points after Rejection ABC

For this task, 3000 points were generated and a threshold of 2000 was selected. Please refer to Appendix 7.4 for the code.

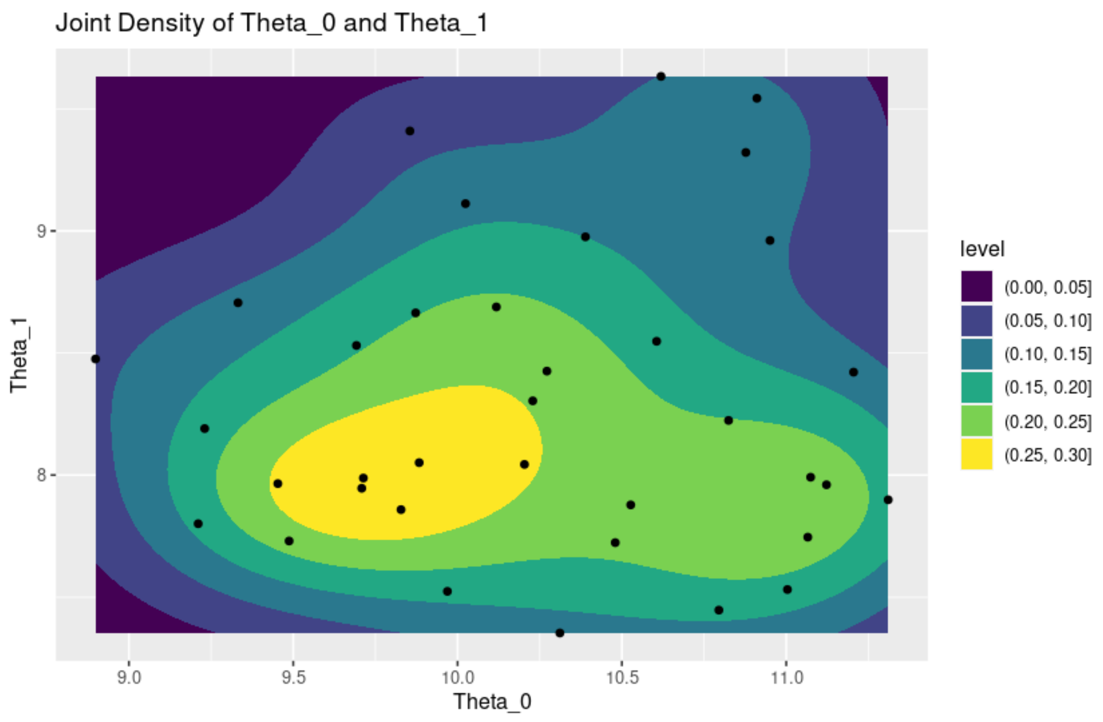

Figure 21: Joint probability distribution for Theta_1 and Theta_0

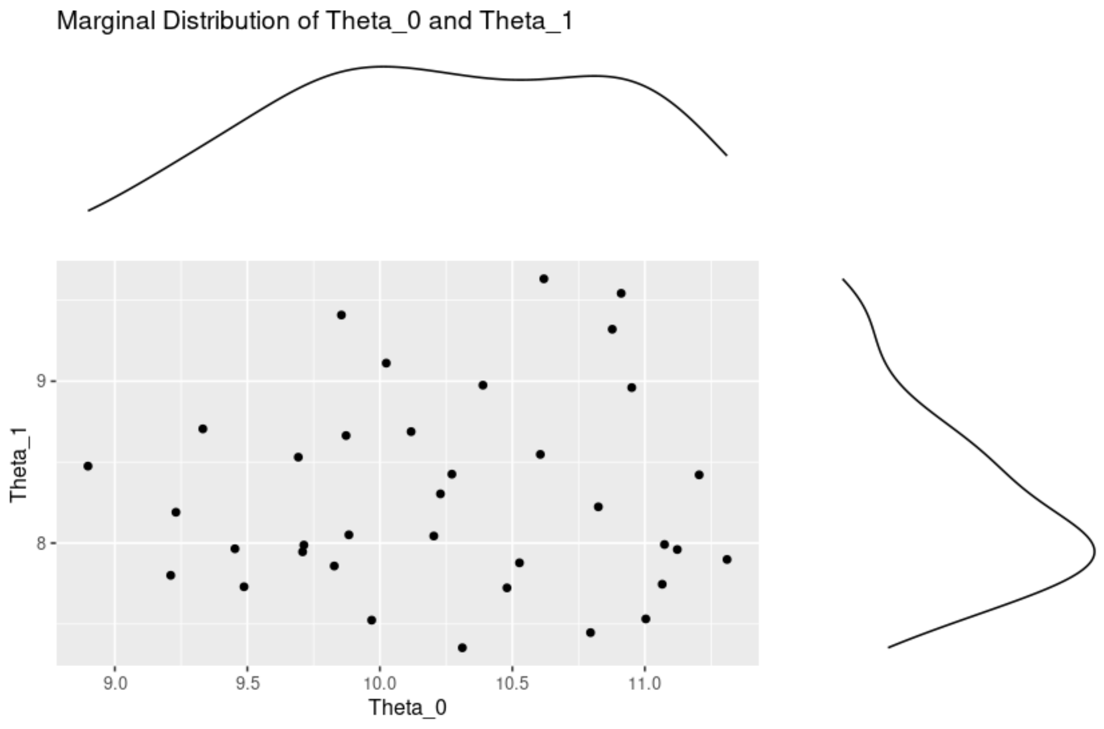

Figure 22: Marginal probability distribution for Theta_0 and Theta_1

The joint probability distribution shows us the most likely value for the parameters (theta_0 and theta_1) to occur jointly. Observing the distribution, that area is slightly above 8 for theta_1 and around 10 for theta_0. This is consistent with the values for the parameter which we computed analytically.

The marginal probability distribution on the other hand shows us their most likely values independently of each other. If the peaks of the distributions are traced down to their originating axes, we see that theta_0 is most likely to fall around 10 while theta_1 most likely falls around 8. Again, this is consistent with the values computed for the parameters in the previous tasks.

5. Discussion

In the first part of the report, the Preliminary Data Analysis (PDA), it was noticed that the time series plots were too chaotic to glean any meaningful information from even though the times series plot of the input audio variable (x1) suggested the amplitude oscillated less for the emotional audio. The probability distributions further suggested that the data contained outliers because of the disparity between the mean and median. This was confirmed by the box plot. The correlation and scatter plot both confirmed that the audio amplitude directly affected the output MEG signal. It also suggested that the MEG signal was affected to a lesser extent by the audio category. This suggests that while the emotional audio increased the MEG, the effect was minimal.

The second part of the report revealed the best model. This model had the smallest residual error meaning that its predictions came the closest to the actual data. The model further confirmed the minimal contribution of the audio category (x2) to the MEG signal; the parameter for x2 was the second smallest. It was further confirmed that the model was the best by looking at the distribution of its errors. The errors followed a Gaussian distribution. This makes sense considering the fact that Gaussian noise was added to the output variable from the outset.

Finally, in the third part of the report, Rejection ABC was performed and this was used to compute parameter values. This method could be used when the likelihood of the model is not known and cannot be evaluated.

6. Conclusion

After a careful review of the data, it is concluded that model 3 is the best model as it outperforms the other candidate models in all the metrics used. Furthermore, it is also concluded that the stated hypothesis for the experiment is correct. The emotional audio had a positive effect on the brain response. It must be said however, that while the effect was noticeable, it was minimal.

7. Appendices

7.1 Setting up

```r
# Importing the relevant libraries
library("corrplot")
library("ggplot2")
library("ggExtra")

# Reading the data in
X = read.csv("X.csv")
y = read.csv("y.csv")
time = read.csv("time.csv")

# Changing the data into matrix format
X = data.matrix(X)
y = data.matrix(y)
time = data.matrix(time)

#No of entries in the dataset
n = length(X[,2])
```

7.2 Task 1 Preliminary Data Analysis

7.2.1 Time series plots

```r
# Task 1.1 Plotting the time series of the input and output variables
```

```
time_series_plot_x = ggplot(data=NULL, aes(x=time[,1], y=X[,1])) +
geom_point() + labs(title="Time series of Input Audio", y="Audio Samples",
x="Time(secs)") + geom_line(aes(color=X[,2]))
time_series_plot_x

time_series_plot_y = ggplot(data=NULL, aes(x=time[,1], y=y[,1])) +
geom_point() + geom_line(aes(color=X[,2])) + ggtitle("Time series of Output
MEG")
time_series_plot_y
```

7.2.2 Density distributions

```
# Task 1.2 Plotting the density distributions of the input audio and output
MEG
x_density_plot = ggplot(data=NULL, aes(X[, 1], y=..density..)) +
geom_histogram(binwidth=0.5, color="black", fill="white") +
geom_density(fill="steelblue", alpha=0.2)
x_density_plot + labs(title="Distribution plot of input audio", x="x1",
y="Density") + geom_vline(xintercept=mean(X[,1]), col="red") +
  geom_vline(xintercept=median(X[,1]), col="blue", linetype="dashed")

y_density_plot = ggplot(data=NULL, aes(y[, 1], y=..density..)) +
geom_histogram(binwidth=5, color="black", fill="white") +
geom_density(fill="red", alpha=0.2)
y_density_plot + labs(title="Distribution plot of output MEG", x="y",
y="Density") + geom_vline(xintercept=mean(y[,1]), col="red") +
  geom_vline(xintercept=median(y[,1]), col="blue", linetype="dashed")
```

7.2.3 Scatter plot and correlation

```
# Task 1.3 Scatter plots and correlation for the input and output
variables.
scatter_plot = ggplot(data=NULL, mapping=aes(x=X[,1], y=y[,1],
color=X[,2])) + geom_point() + labs(title="Scatter Plot of Brain Signals
Against Audio Signals", y="MEG Brain Signals", x="Audio Signals")
scatter_plot

cor_x_y = cor(X, y[,1], method="pearson")
corrplot(cor_x_y)
cor_x_y
```

7.2.4 Box plot

```
# Task 1.4 Creating box plots for the output variables
ggplot(data=NULL, aes(x=factor(X[,2]), y = y[,1])) + geom_boxplot() +
labs(title="Box Plot Showing Effect of Audio Category on MEG", y="MEG
Signal", x="Audio Category")
```

7.3 Regression

7.3.1 Preparing the data

```
# Task 2
# Creating the X Matrix for the all the models
ones = matrix(1, length(X[, 1]), 1)

X_model_1 = cbind(ones, X[, 1] ^ 3, X[, 1] ^ 5, X[, 2])
X_model_2 = cbind(ones, X[, 1], X[, 2])
X_model_3 = cbind(ones, X[, 1], X[, 1] ^ 2, X[, 1] ^ 4, X[, 2])
X_model_4 = cbind(ones, X[, 1], X[, 1] ^ 2, X[, 1] ^ 3, X[, 1] ^ 5, X[, 2])
X_model_5 = cbind(ones, X[, 1], X[, 1] ^ 3, X[, 1] ^ 4, X[, 2])
```

7.3.2 Estimating the model parameters

```
# Task 2.1 Estimating Model Parameters Theta_Hat
theta_hat_1 = solve(t(X_model_1) %*% X_model_1) %*% t(X_model_1) %*% y
theta_hat_2 = solve(t(X_model_2) %*% X_model_2) %*% t(X_model_2) %*% y
theta_hat_3 = solve(t(X_model_3) %*% X_model_3) %*% t(X_model_3) %*% y
theta_hat_4 = solve(t(X_model_4) %*% X_model_4) %*% t(X_model_4) %*% y
theta_hat_5 = solve(t(X_model_5) %*% X_model_5) %*% t(X_model_5) %*% y
```

7.3.3 Estimating the residual errors (RSS) of each candidate model

```
# Task 2.2 Estimating Residual Sum of Square Errors (RSS) for each model
# First, we evaluate the model predictions y_hat for each model
y_hat_model1 = X_model_1 %*% theta_hat_1
y_hat_model2 = X_model_2 %*% theta_hat_2
y_hat_model3 = X_model_3 %*% theta_hat_3
y_hat_model4 = X_model_4 %*% theta_hat_4
y_hat_model5 = X_model_5 %*% theta_hat_5

# Then calculate the errors
error1 = y - y_hat_model1
error2 = y - y_hat_model2
error3 = y - y_hat_model3
error4 = y - y_hat_model4
error5 = y - y_hat_model5

# Calculating RSS for each model
```

```
rss_1 = sum((error1) ^ 2)
rss_2 = sum((error2) ^ 2)
rss_3 = sum((error3) ^ 2)
rss_4 = sum((error4) ^ 2)
rss_5 = sum((error5) ^ 2)

# Calculating the variance of the models' residuals rho_squared
rho_squared1 = rss_1 / (n - 1)
rho_squared2 = rss_2 / (n - 1)
rho_squared3 = rss_3 / (n - 1)
rho_squared4 = rss_4 / (n - 1)
rho_squared5 = rss_5 / (n - 1)
```

7.3.4 Computing log-likelihood for each candidate model

```
# Task 2.3 Computing the log-likelihood function of each model
log_likelihood1 = - ((n / 2) * log (2 * pi)) - ((n / 2) *
(log(rho_squared1))) - ((1 / (2 * rho_squared1)) * rss_1)
log_likelihood2 = - ((n / 2) * log (2 * pi)) - ((n / 2) *
(log(rho_squared2))) - ((1 / (2 * rho_squared2)) * rss_2)
log_likelihood3 = - ((n / 2) * log (2 * pi)) - ((n / 2) *
(log(rho_squared3))) - ((1 / (2 * rho_squared3)) * rss_3)
log_likelihood4 = - ((n / 2) * log (2 * pi)) - ((n / 2) *
(log(rho_squared4))) - ((1 / (2 * rho_squared4)) * rss_4)
log_likelihood5 = - ((n / 2) * log (2 * pi)) - ((n / 2) *
(log(rho_squared5))) - ((1 / (2 * rho_squared5)) * rss_5)
```

7.3.5 Computing AIC and BIC for each candidate model

```
# Task 2.4 Computing the Akaike Information Criterion (AIC) and Bayesian
Information Criterion (BIC) for each model
# For each model, we store the number of estimated parameters in a k
variable

k1 = length(theta_hat_1)
k2 = length(theta_hat_2)
k3 = length(theta_hat_3)
k4 = length(theta_hat_4)
k5 = length(theta_hat_5)

# Then calculate the AIC and BIC
AIC1 = 2 * k1 - 2 * log_likelihood1
AIC2 = 2 * k2 - 2 * log_likelihood2
AIC3 = 2 * k3 - 2 * log_likelihood3
AIC4 = 2 * k4 - 2 * log_likelihood4
AIC5 = 2 * k5 - 2 * log_likelihood5
```

```
BIC1 = k1 * log(n) - 2 * log_likelihood1
BIC2 = k2 * log(n) - 2 * log_likelihood2
BIC3 = k3 * log(n) - 2 * log_likelihood3
BIC4 = k4 * log(n) - 2 * log_likelihood4
BIC5 = k5 * log(n) - 2 * log_likelihood5
```

7.3.6 Plotting the error distributions

```
# Task 2.5. Next we plot the error distributions and investigate if the
errors assume a Gaussian Distribution.
# Error distributions
error_dist1 = ggplot(data=NULL, aes(x=error1, y=..density..)) +
geom_histogram(binwidth=5, color="black", fill="white")  +
geom_density(fill="red", alpha=0.5)
error_dist1 + labs(title="Density Plot of Model 1 Errors", x="Errors",
y="Density")

error_dist2 = ggplot(data=NULL, aes(x=error2, y=..density..)) +
geom_histogram(binwidth=5, color="black", fill="white")  +
geom_density(fill="red", alpha=0.5)
error_dist2 + labs(title="Density Plot of Model 2 Errors", x="Errors",
y="Density")

error_dist3 = ggplot(data=NULL, aes(x=error3, y=..density..)) +
geom_histogram(binwidth=5, color="black", fill="white")  +
geom_density(fill="red", alpha=0.5)
error_dist3 + labs(title="Density Plot of Model 3 Errors", x="Errors",
y="Density")

error_dist4 = ggplot(data=NULL, aes(x=error4, y=..density..)) +
geom_histogram(binwidth=5, color="black", fill="white")  +
geom_density(fill="red", alpha=0.5)
error_dist4 + labs(title="Density Plot of Model 4 Errors", x="Errors",
y="Density")

error_dist5 = ggplot(data=NULL, aes(x=error5, y=..density..)) +
geom_histogram(binwidth=5, color="black", fill="white")  +
geom_density(fill="red", alpha=0.5)
error_dist5 + labs(title="Density Plot of Model 5 Errors", x="Errors",
y="Density")

# QQ Plots
qq_plot1 = qplot(sample=error1, data=NULL) + stat_qq_line(color="blue")
qq_plot1 + labs(title="QQ Plot of Model 1 Errors", x="x", y="Model 1
Errors")

qq_plot2 = qplot(sample=error2, data=NULL) + stat_qq_line(color="blue")
```

```
qq_plot2 + labs(title="QQ Plot of Model 2 Errors", x="x", y="Model 2
Errors")

qq_plot3 = qplot(sample=error3, data=NULL) + stat_qq_line(color="blue")
qq_plot3 + labs(title="QQ Plot of Model 3 Errors", x="x", y="Model 3
Errors")

qq_plot4 = qplot(sample=error4, data=NULL) + stat_qq_line(color="blue")
qq_plot4 + labs(title="QQ Plot of Model 4 Errors", x="x", y="Model 4
Errors")

qq_plot5 = qplot(sample=error5, data=NULL) + stat_qq_line(color="blue")
qq_plot5 + labs(title="QQ Plot of Model 5 Errors", x="x", y="Model 5
Errors")

# Shapiro-Wilks tests
shapiro.test(error1)
shapiro.test(error2)
shapiro.test(error3)
shapiro.test(error4)
shapiro.test(error5)
```

7.3.7 Performing the train-test split

```
# Task 2.7 Splitting the data
# Task 2.7.1 70-30 Train test split

# First, we combine the input and output variables of the dataset
dataset = cbind(X,time, y)

# Then we split the dataset in a ration of 70:30
v<- as.vector(c(rep(TRUE,70),rep(FALSE,30))) #create 70 TRUE, 30 FALSE
ind <- sample(v) #Sample them randomly.
train <- dataset[ind, ]
test <- dataset[!ind, ]
#train <- dataset[sample1==0, ]
#test = dataset[sample1, ]
```

7.3.8 Estimating model parameters from the training data

```
# Task 2.7.0 Estimating the model parameters using the training data

# First, we build the X matrix for the training and test data. The model is
```

```
y = ThetaX0 + Theta1X1  + Theta2X1^2 + Theta3X1^4 + Theta4X2
training_ones = matrix(1, length(train[,1]), 1)
x1_train = train[,1]
x1_sq_train = x1_train ^ 2
x1_pow4_train = x1_train ^ 4
x2_train = train[,2]
y_train = train[,4]
time_train = train[,3]

testing_ones = matrix(1, length(test[,1]), 1)
x1_test = test[,1]
x1_sq_test = x1_test ^ 2
x1_pow4_test = x1_test ^ 4
x2_test = test[,2]
y_test = test[,4]
time_test = test[,3]
length(x1_sq_test)
# Create the X matrix
train_X = cbind(training_ones, x1_train, x1_sq_train, x1_pow4_train,
x2_train)
test_X = cbind(testing_ones, x1_test, x1_sq_test, x1_pow4_test, x2_test)

# Task 2.7.1 Estimating the model parameters
theta_hat_train = solve(t(train_X) %*% train_X) %*% t(train_X) %*% y_train
# theta_hat_test = solve(t(test_X) %*% test_X) %*% t(test_X) %*% y_test
```

7.3.9 Computing the model's predictions from the test data

```
# Task 2.7.2 Computing the model's prediction on the test data
y_hat_train = train_X %*% theta_hat_train
y_hat_test = test_X %*% theta_hat_train

# Computing the errors of the model (RSS)
len = length(testing_ones)
error_test = y_test - y_hat_test
rss_test = sum((error_test) ^ 2)
rss_test
```

7.3.10 Computing the confidence intervals and plotting them

```
# Task 2.7.3 Computing the confidence intervals of the errors
# Error variance sigma^2
sigma_2 = rss_test/( len - 1 )
number_of_parameters = 5

cov_thetaHat = sigma_2 * (solve(t(test_X) %*% test_X))
```

```
var_y_hat = matrix(0 , len , 1)

length(test_X[,1:2])

for( i in 1:len){
  X_i = matrix( test_X[i,] , 1 , number_of_parameters ) # X[i,] creates a
vector. Convert it to matrix
  var_y_hat[i,1] = X_i %*% cov_thetaHat %*% t(X_i) # same as sigma_2 * (
X_i %*% ( solve(t(X) %*% X)  ) %*% t(X_i) )
}

# Calculate the 95% confidence interval
CI = 2 * sqrt(var_y_hat)

# Plot the errors against time and show the confidence intervals as error
bars in the plot
ci_plot = ggplot(data=NULL, mapping=aes(x=time_test, y=y_hat_test)) +
geom_point(color="red") + geom_line()
ci_plot + geom_errorbar(aes(ymin=y_hat_test-CI, ymax=y_hat_test+CI)) +
geom_point(mapping=aes(x=time_test, y=y_test), color="blue") +
  labs(title="Confidence Interval Plot", y="MEG Output", x="Time(secs)")
```

7.4 Rejection ABC

```
# Task 3: Perform Approximate Bayesian Computation (ABC) on the dataset

# Evaluate the posterior distribution
# Step 1: Pick 2 parameters from the model with the largest absolute
values.
first_param = theta_hat_train[1]
second_param = theta_hat_train[2]

# Number of random points to generate
no_points = 3000
# Generate a collection of normally distributed numbers within +- 0.5 of
the parameters chosen
first_param_unif = runif(no_points, (first_param * 0.5) - first_param,
(first_param * 0.5) + first_param)
second_param_unif = runif(no_points, (second_param * 0.5) - second_param,
(second_param * 0.5) + second_param)

# Plot the points
ggplot(data=NULL, aes(first_param_unif, second_param_unif)) + geom_point()
+
  labs(title="Points Generated Before Rejection ABC", x="Theta_0",
y="Theta_1")
```

```r
# Create 2 empty matrices
params_X0_ABC = NULL
params_X1_ABC = NULL

# Loop through the points and test the errors (RSS) of each point. Discard
any points that generate errors that fall above a threshold value
for (i in 1: no_points) {
  theta_hat_ABC = matrix(c(first_param_unif[i], second_param_unif[i],
6.247, -0.283, 4.16))
  y_hat_ABC = X_model_3 %*% theta_hat_ABC

  errors_ABC = y - y_hat_ABC
  rss_ABC = sum(errors_ABC ^ 2)

  if (rss_ABC < 2000) {
      print(rss_ABC)
      params_X0_ABC[i] = first_param_unif[i]
      params_X1_ABC[i] = second_param_unif[i]
  }
}

# Clean the vector of selected points by removing all NA/NULL values
params_X0_complete = params_X0_ABC[complete.cases(params_X0_ABC)]
params_X1_complete = params_X1_ABC[complete.cases(params_X1_ABC)]
print(length(params_X0_complete))

# Plot the selected points
ggplot(data=NULL, aes(params_X0_complete, params_X1_complete)) +
geom_point() +
  labs(title="Points Selected After Rejection ABC", x="Theta_0",
y="Theta_1")

# Plot the joint probability distribution
joint_density_plot = ggplot(data=NULL, aes(x=params_X0_complete,
y=params_X1_complete)) + geom_density2d_filled()
joint_density_plot + labs(title="Joint Density of Theta_0 and Theta_1", x =
"Theta_0", y="Theta_1") + geom_point()

# Plot the marginal distribution
marginal_dist = ggplot(data=NULL, aes(x=params_X0_complete,
y=params_X1_complete)) + geom_point() +
  labs(title="Marginal Distribution of Theta_0 and Theta_1", x = "Theta_0",
y="Theta_1")

marginal_dist_plot = ggMarginal(marginal_dist, type="density", size=2)
marginal_dist_plot
```