Solution By: Shrawan

Introduction

Small metal discs (electrodes) attached to your scalp are used during an electroencephalogram (EEG) technique to find electrical activity in your brain. Our cells are active and exchange electrical impulses with one another when doing any action. These impulses are recorded as wavy lines or signals during EEG. In this course, a series of EEG data were examined, and the optimal regression model was determined.

Task 1

Time series plot

A time series plot helps you to comprehend the process that produced the relevant data by plotting it against time. In a time-series plot, time is plotted on the X-axis while the observed data is plotted on the Y-axis. R is used to create the time series plot of the input and output EEG data, and the results are shown below.

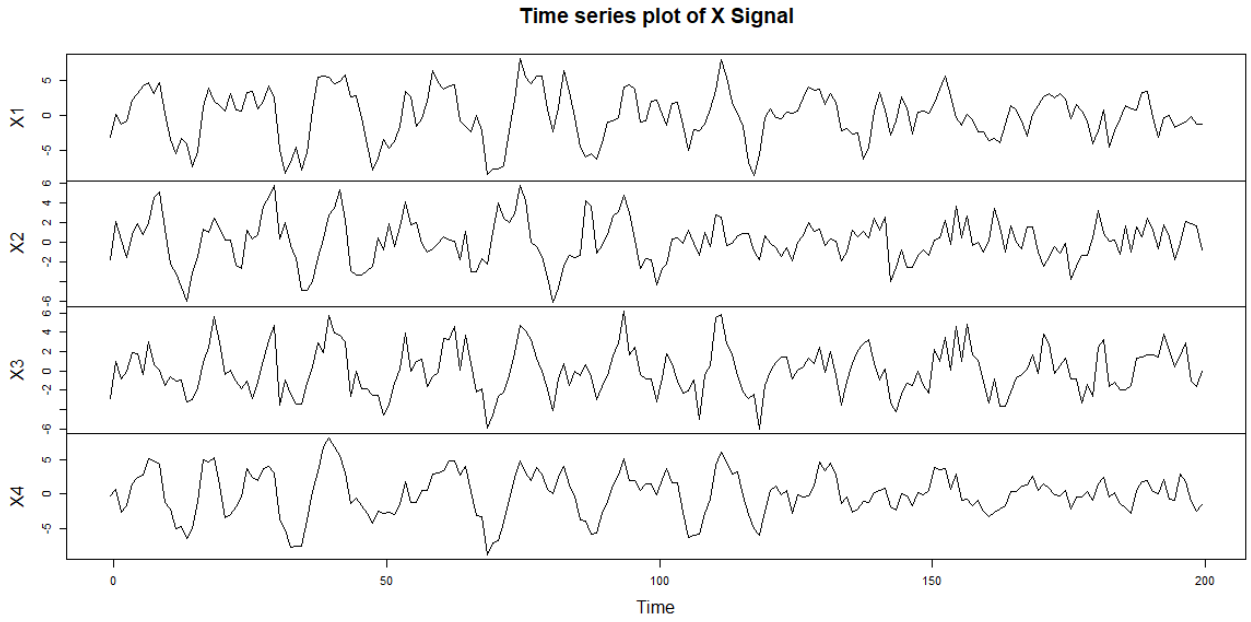


Diagram 1: Time series plot of input signal

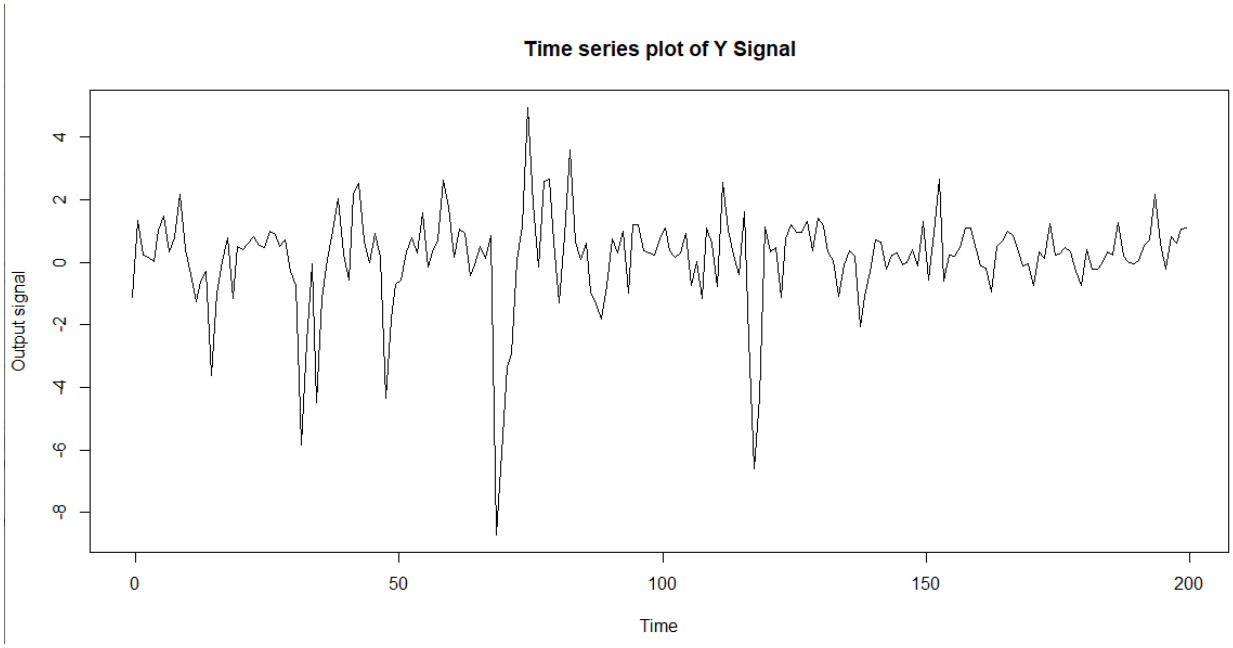


Diagram 2: Output signal Tine series plot

Making sure there are no outliers or abrupt changes in the plot that are caused by data inaccuracies is crucial when reading a time series display. There are no such abrupt spikes in the plots made using the provided data, indicating that neither external causes nor entry errors could have affected the data.

The four input values are plotted against a timestamp in the first picture, and the output of the input values against the same timestamp is shown in the second diagram. Diverse trends in the data can be seen by carefully examining the graphs. The input signal's time series plot reveals considerably less noise and fewer spikes.

After 120ms, we can observe that the plot is calming down and displaying some recurrent patterns. X4 appears to be the most stable of the four signals, with fewer spikes and a more recognizable pattern. The spikes are scarce after the 120 ms limit. The output signal begins gradually up until the 60 ms threshold. After the 120ms barrier, we can see the spikes are leveling down and following a steady pattern. Around 60ms, we can detect a significant surge. The introduction of noise has made the spikes more noticeable.

Distribution Plot

An explanation of the relationship between data samples and observed values that is sorted from smallest to largest is known as a distribution. We are comparing the input signal sent to the brain using the diagram below.

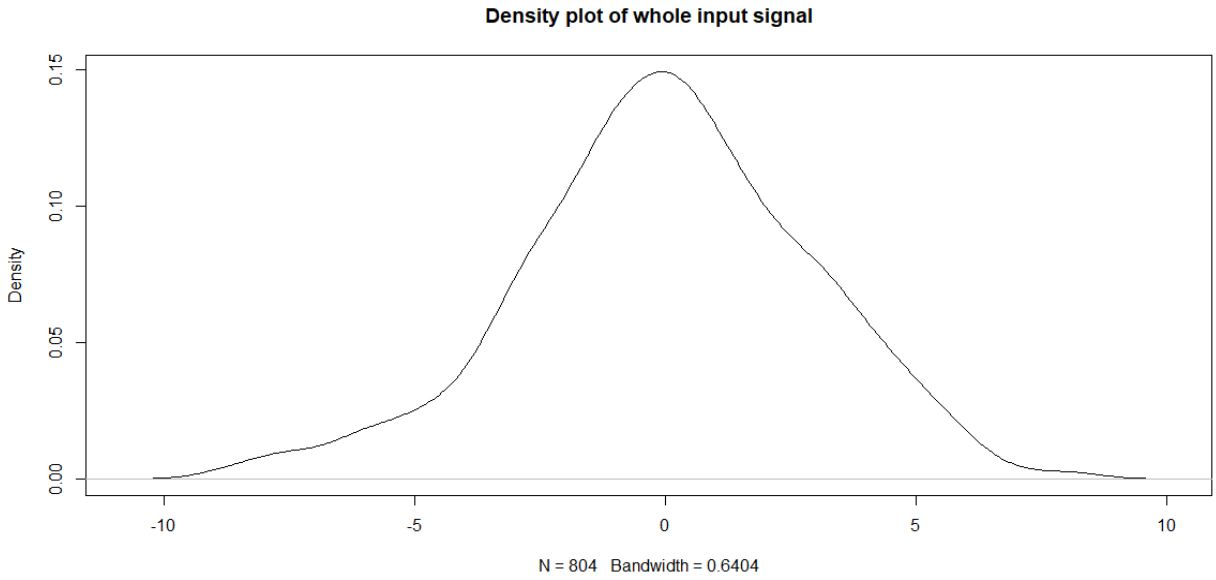


Diagram 3: Density Plot of Input signal

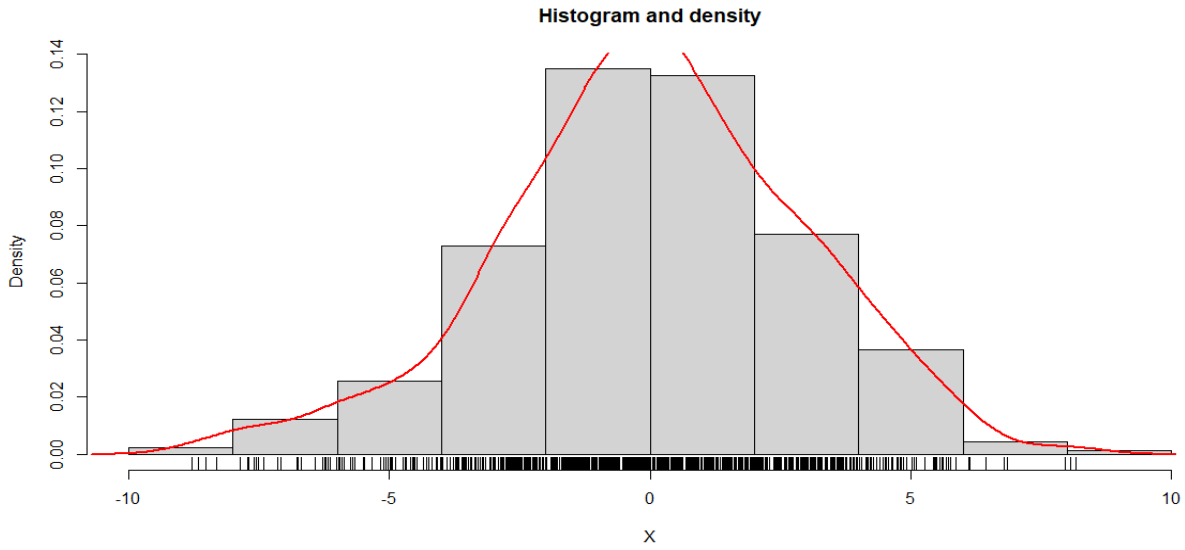


Diagram 4: Input signal Histogram and density plot

The curve has a bell-shaped shape and the mean of the input signal appears to be around "0" when observing the distribution of the input signal from the histogram and density. Since the peak of the input (X) signal diagram appears to be near "0," The highest input signal that can be sent to the brain may be between 9 and 10, while the lowest signal is -10. As the shape of the distribution starts to expand from -4 and shrink from 5, the majority of the input signal appears to lie between -4 and 5. Given that the length of both tails appears to be nearly equal, this distribution appears to be symmetrical, making it easy to locate the distribution's center.

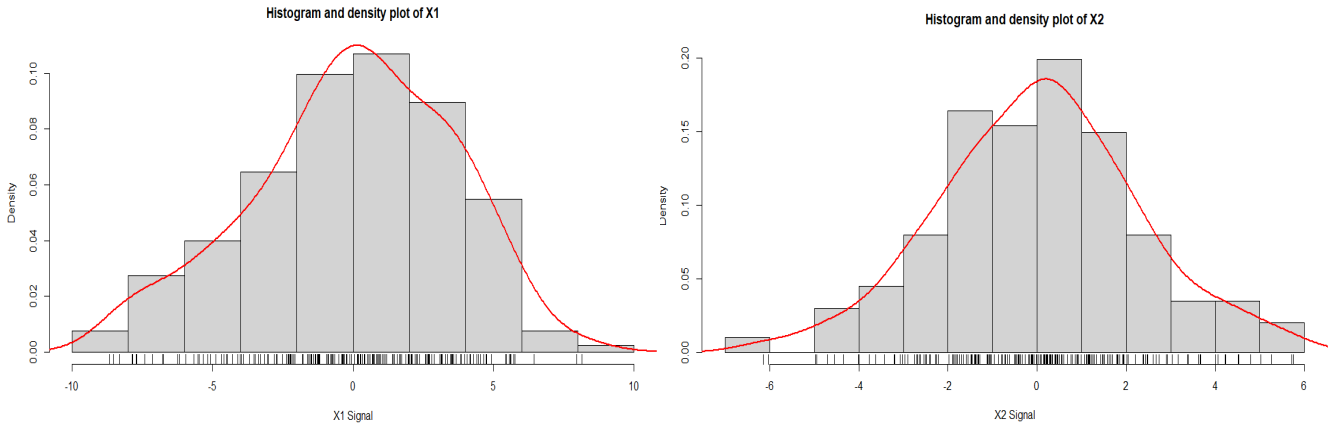


Diagram 6: X1 signal Histogram and density plot

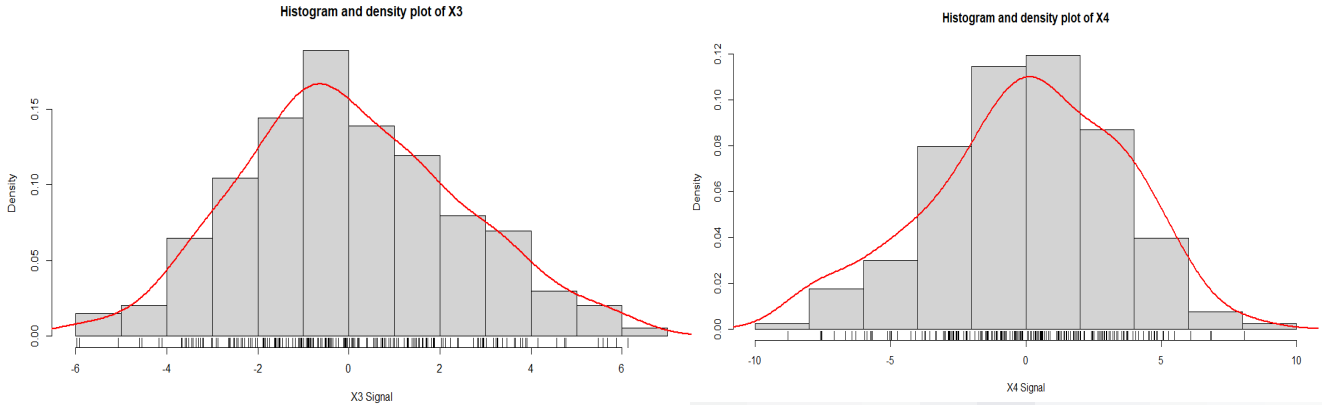


Diagram 8: X3 signal Histogram and density plot

Let's now examine each individual signal that the brain receives from the four different input signals. According to the graphic below, all input signals from X1 through X4 appear to follow the same pattern. Some of the input signals in X2 appear to deviate from the norm, or outliers, as they are sometimes referred to.

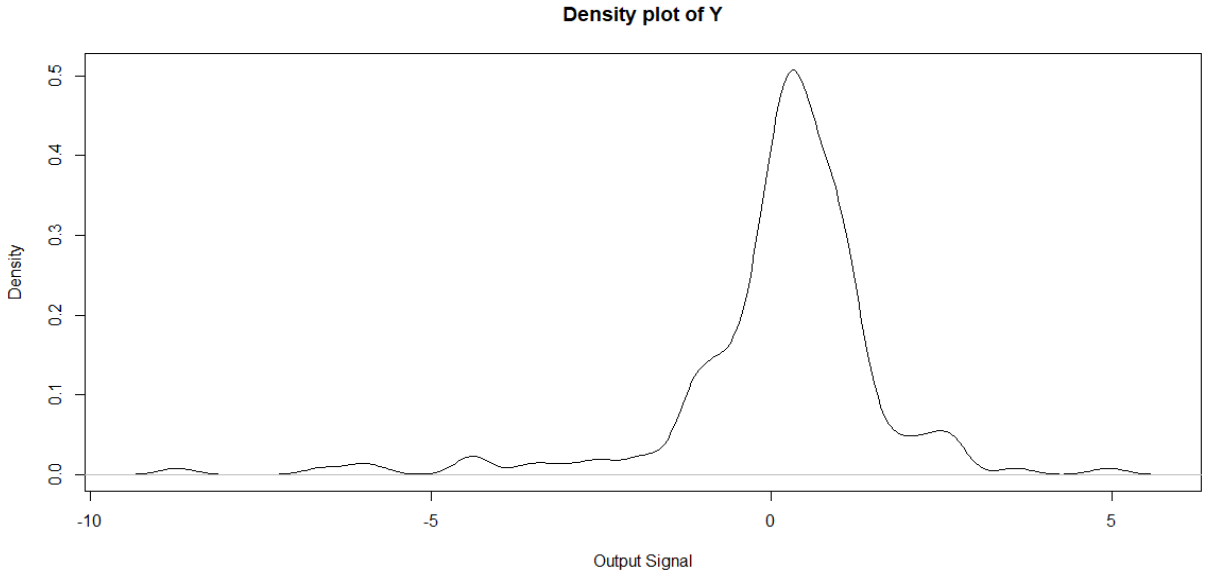


Diagram 9: Density plot of Output signal

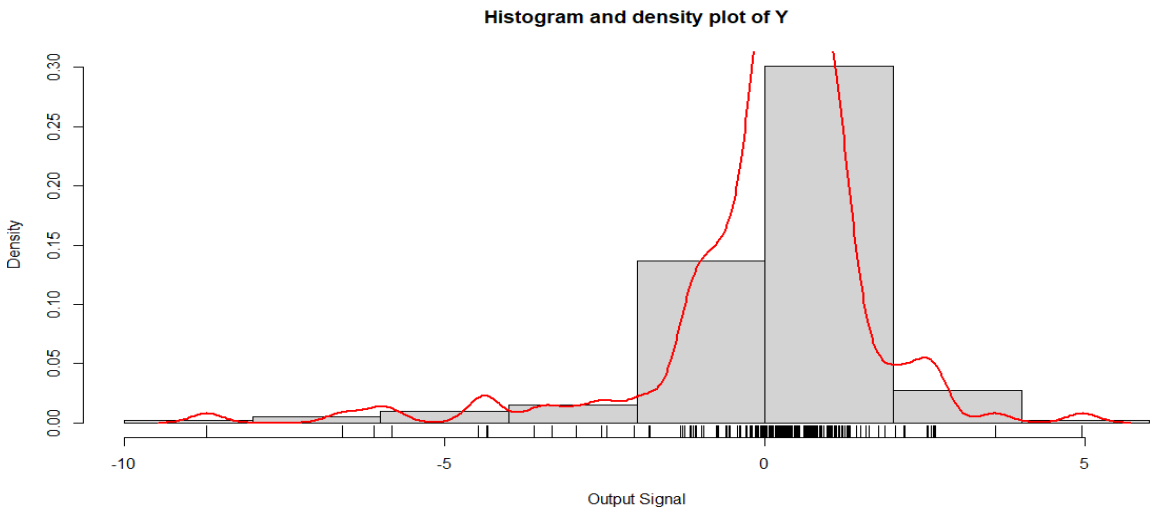


Diagram 10: Histogram and density plot of Output signal

Y signal, also known as the output signal, can also be referred to as a left-skewed distribution because, when looking at the diagram, the left side's tail appears to be considerably longer than the right. Most output appears to be between -2 and 2, with highest output being 5 and minimum being 9. The Y signal's mean could be less than 0 or 0.5.

Correlation and Scatter plots

In our scenario, the relationship between the input and output datasets is shown using a scatterplots diagram. A single dot in a scatterplot diagram represents a single piece of data, and different dots throughout the diagram can each reflect a different pattern, depending on how closely the data are grouped. The relationship is perfect positive correlation when a data set tends to form a straight line beginning at the origin of the y axis, and perfect negative correlation when a straight line begins at the origin of the x axis; otherwise, the relationship is referred to as no correlation. There may not always be a perfect positive or negative correlation, in which case the relationship is referred to as having a low positive or low negative correlation.

Let's now examine the input and output signals of the given data. In the picture below that shows the relationship between the X1, X2, and X3 signals and the Y signal, we can see a pattern where the data appears to follow a low positive correlation since the dot plot has a greater Y-value on the X-axis and the data are not dispersed.

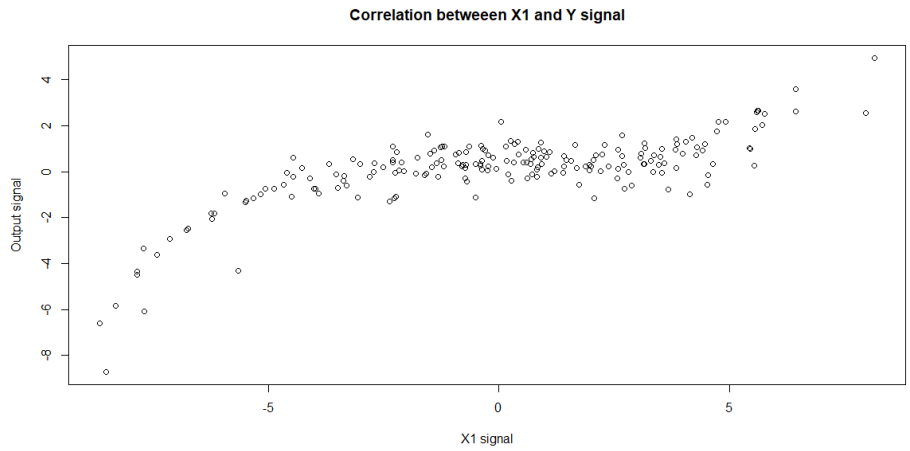


Diagram 11: Correlation and scatter plot of X1 signal

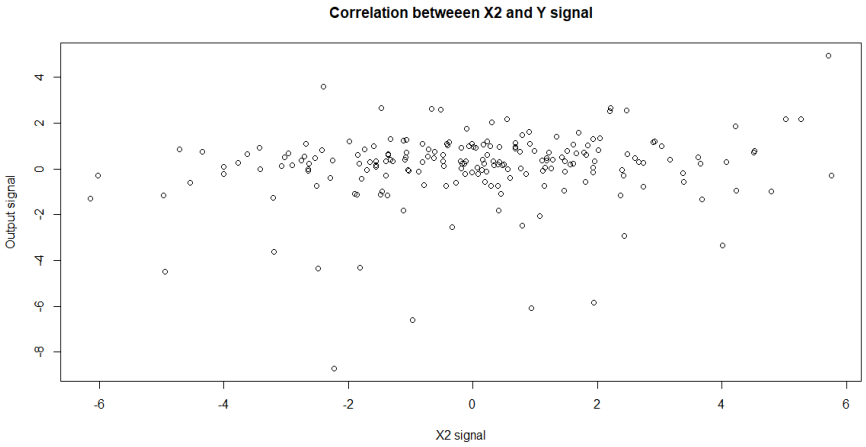


Diagram 12: Correlation and scatter plot of X2 signal

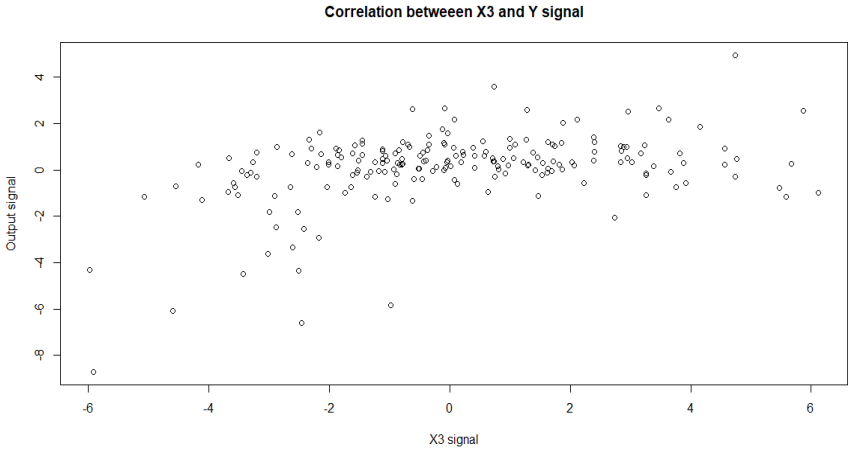


Diagram 13: Correlation and scatter plot of X3 signal

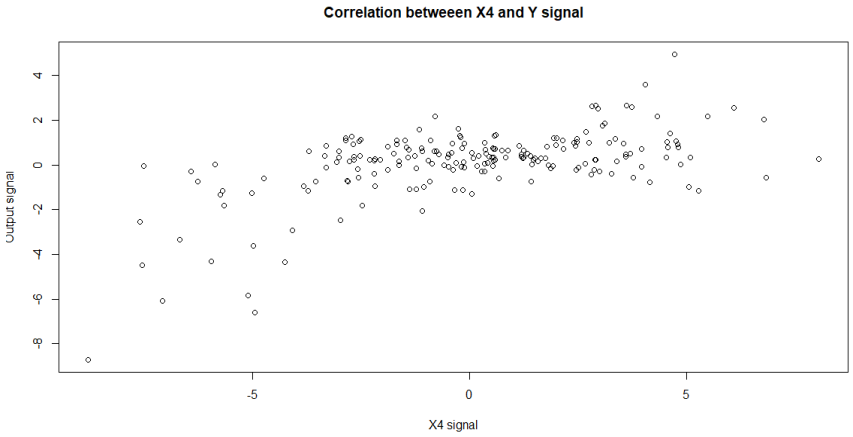


Diagram 14: Correlation and scatter plot of X4 signal

As for X2 and Y signals the data seems to have no correlation as the data do not follow any pattern and data are scattered unevenly.

Task 2.1

Generally, a random variable, also known as an estimator, is useful for observing the different characteristics of a distribution since the true value of that distribution is unknown. Estimator can be represented by the variable **"θ"**. Where **"θ = {θ, θ_1, θ_2, …, θ_{bisa}} ^{T}"**. So, to calculate the estimator model parameters for every given candidate module of EEG data we will be using least squares. Least squares are usually used to find the best possible fit line of the regression equation or the date point which shows the relationship between an unknown dependent value with the independent value. Least square is denoted as "$\hat{\theta}$" and it can be calculated as **"** $\hat{\theta} = (X^TX)^{-1}X^Ty$**"** where X and Y are the input and output data of EEG signals.

In **R** we can convert this formula as **"**$\hat{\theta}$ **=solve(t(X) %*% X) %*% t(X) %*% Y".** To calculate the least square, first we must bind the value/column of the X (input data) form the provided dataset. For example, "**X<-cbind (ones, (X1), (X2), (X3) ^3, (X4) ^4)" is** one of the ways to bind the dataset. Now after binding the data we can perform the Least squares form formula mentioned above. The result of all the candidate models is provided below.

Theta Hat for model 1

0.468551685	-0.034101975	-0.001849575	0.010381813	-0.001949154

Table 1:Theta Hat of model 1:

Theta Hat for model 2

0.303501144	0.016334677	-0.002713985

Table 2: Theta Hat of model 2

Theta Hat for model 3

0.448299550	0.038109255	0.009827804	-0.002092558

Table 3: Theta Hat of model 3

Theta Hat for model 4

0.450329209	-0.034881772	0.010482178	-0.001990496

Table 4: Theta Hat of model 4

Theta Hat for model 5

4.606560e-01	-3.395313e-02	-2.701644e-04	1.034764e-02	-1.943452e-03	-3.083194e-05

Table 5: Theta Hat of model 5

Task 2.2

Model residual Error (RSS)

Model residual error (RSS) also known as Sum Squared Error of an estimator is used to identity the error of squared average estimate value i.e. differences between average square actual and estimated value. RSS is usually used to determine the quality of an estimator and the value of RSS closer to zero is better as well as RSS is never negative. For calculating the RSS first, we calculate the error of every candidate model with the help of $\hat{\theta}$ from task 2.1.

$$RSS = \sum_{i=1}^{n} \left(y_i - \mathbf{x}_i \hat{\boldsymbol{\theta}}\right)^2$$

Diagram 15: RSS equation

While implementing the above formula in R programming we can calculate RSS as (**RSS=sum((Y-Yhat)^2)** where Yhat is a product of $\mathbf{X}_{i\hat{\theta}}$ and n is the total length of Y and ∑ is the sum of all the output Y. The table below represents all the RSS values of the candidate model. $\mathbf{X}_i$ is all the possible value of X. And $\hat{\theta}$ is the value that we calculated from task 2.1 where every model has different $\hat{\theta}$ value. Finally, the formula for calculating RSS is **RSS=sum((Y-Y_hat) ^2).** Where Y is output signal and sum is sum of every possible value.

Model	**RSS Value**
Model1	57.32927
Model2	351.4147
Model3	57.32536
Model4	57.46405
Model5	57.30923

Table 6: RSS of each candidate model

Task 2.3

Log-likelihood Function

The likelihood function is generally used to identify how well the measured value fits the sample data of a provided model as parameters are unknown. The goal of likelihood is to find an optimal way to fit a distribution to the data with a fixed observed value. Log likelihood is a logarithmic transformation of the likelihood function which maximizes the likelihood which is equivalent to maximum log-likelihood and can be convenient to work for practical purposes by maximizing the estimation.

$$\ln p\left(D|\hat{\boldsymbol{\theta}}\right) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\hat{\sigma}^2) - \frac{1}{2\hat{\sigma}^2}\text{RSS}$$

Diagram 16: Log-likelihood equation

From the above equation

“ln p(D|$\hat{\theta}$)” is represented as log-likelihood.

“n” is the total number of **“Y”** signals.

“ln” natural logarithm function represented as log in R language

"σ^2" is the variance of the RSS from task 2.2. **"σ^2"** can be calculated with the help of RSS as **"σ^2 =RSS/(n-1)".**
"π" value is 2.27 represented as "pi" in R language
"RSS" is the value obtained from task 2.2
In R above formula can be represented as:

likehood= -(N/2) *(log(2*pi)) - (N/2) * (log (Variance_model1)) - (1/(2*Variance)) * RSS

The results of likelihood obtained are listed below.

Model	Likelihood
Model1	-159.1313
Model2	-341.3534
Model3	-159.1244
Model4	-159.3673
Model5	-159.0962

Table 7: Log-likelihood of each candidate model

Task 2.4

Akaike Information criterion (AIC)

Akaike information criterion (ACI) is an estimator of prediction error which helps us to identify how well the model fits with the provided dataset without overfitting it. AIC helps to evaluate the quality of the model by comparing different candidate models.
k represents the length of parameters to be estimated that was calculated on task 2.1,
$\hat{L}$ represent the likelihood function that was calculate on 2.3

$$\mathrm{AIC} = 2k - 2\ln(\hat{L})$$

Diagram 17: AIC equation

Model	AIC
Model1	328.2626
Model2	688.7068
Model3	326.2489
Model4	328.7346
Model5	330.1923

Table 8: AIC of each candidate model

Bayesian Information criterion (BIC)

Bayesian information Criterion (BIC) is one of the standards for selecting the best and scoring a finite set of modules from various candidate models. While selecting the BIC, the candidate model with the lowest scoring value is preferred. BIC is related to AIC and it is also based on some part of the likelihood function. The calculation expression for BIC is provided below.
k represents the length of parameters to be estimated that was calculated on task 2.1,
$\hat{L}$ represent the likelihood function that was calculated on 2.3.
The result and discussion of BIC calculation in R is provided below.

$$\mathrm{BIC} = k\ln(n) - 2\ln(\widehat{L})$$

Diagram 18: BIC equation

Model	BIC
Model1	344.7791
Model2	698.6168
Model3	339.4621
Model4	345.2511
Model5	350.0121

Table 9: BIC of each candidate model

Task 2.5

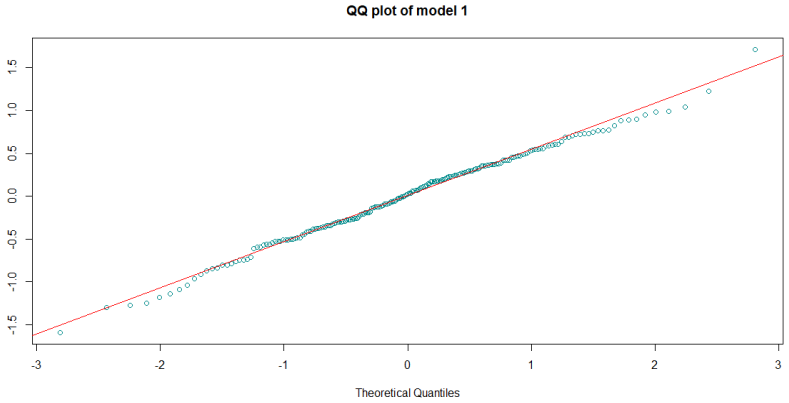


Diagram 19: Model 1 Q-Q plot

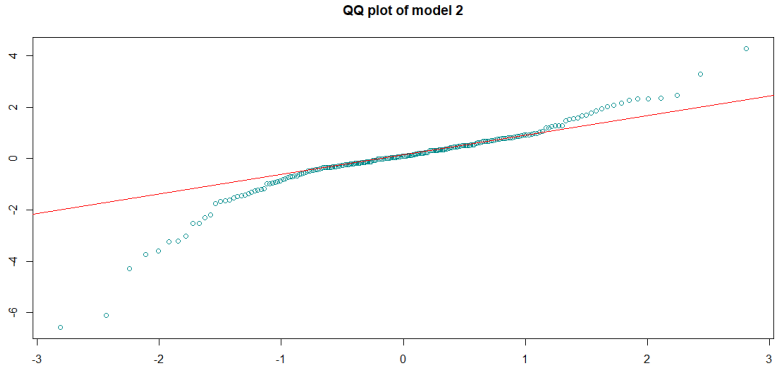


Diagram 20: Model 2 Q-Q plot

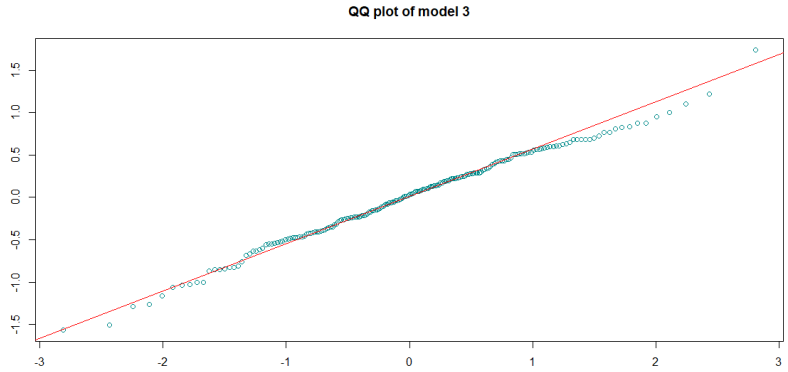


Diagram 21: Model 3 Q-Q plot

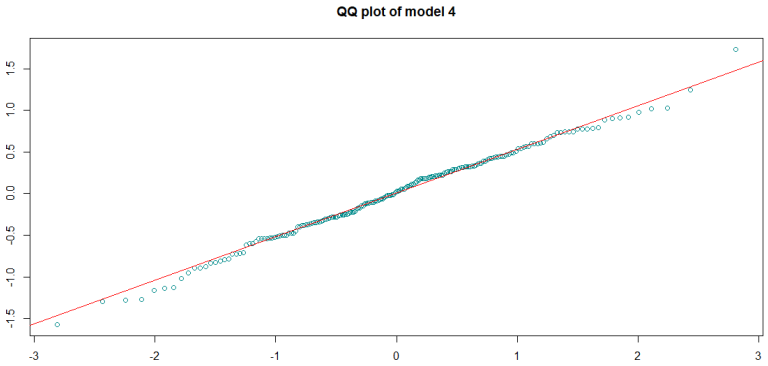


Diagram 22: Model 4 Q-Q plot

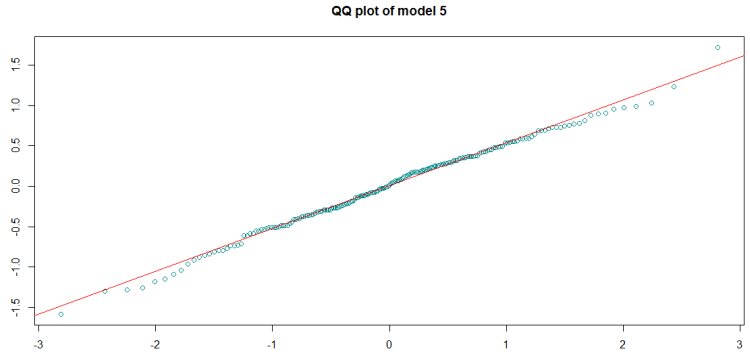


Diagram 23: Model 5 Q-Q plot

For plotting the normal/Gaussian distribution of output EEG signal with additive Gaussian noise, RSS also known as Model residual Error of every candidate model was calculated on task 2.2. For calculating RSS, expected Y(Output) i.e. Y_hat value was deducted by output value(Y) which will be used in qqnorm, qqline a plotting function of ggplot2 i.e. pre-defined in R programming. For every model a normal/gaussian distribution was plotted for identifying the best regression model and trend of the data.

Task 2.6

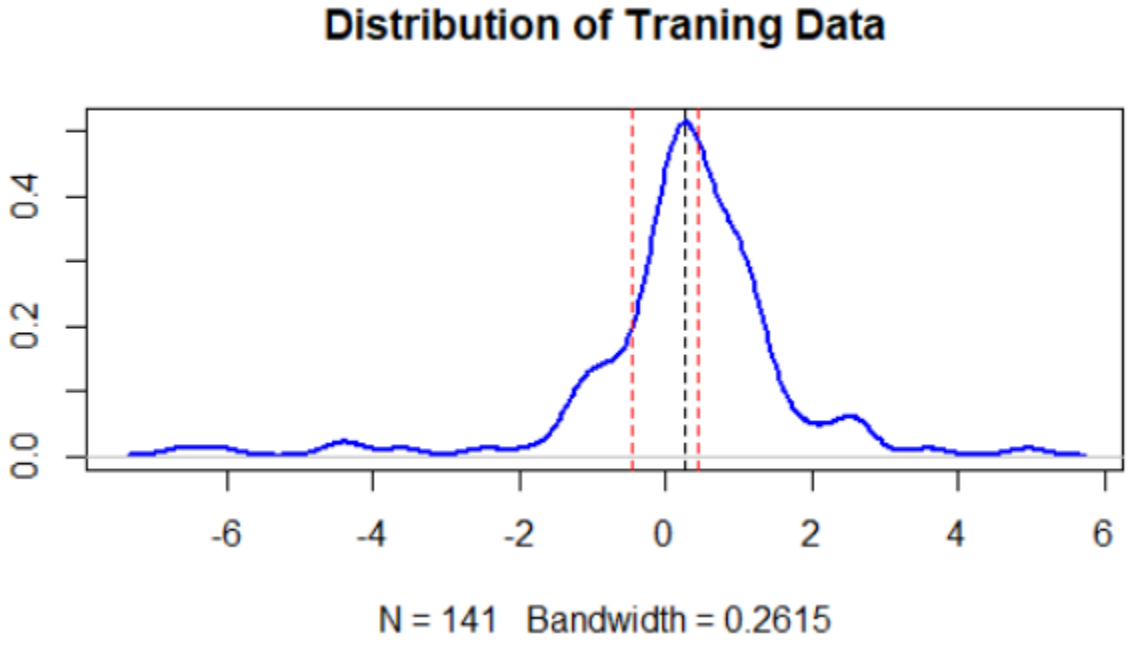
For selecting the best candidate model all the required tasks i.e. from 2.1 to 2.5 were completed. For example, calculating RSS, likelihood, plotting normal distribution graphs. Each step was carefully completed. From the plot of 2.5 and calculation of AIC and BIC I will be selecting the best candidate model for this regression. As we know AIC and BIC are widely used for the model selection process. As both models help to minimize the score of error while selecting a model. AIC tends to select a model based on relative distance between unknown likelihood of a data and fit the likelihood of the model. So, a

model can be selected if the AIC is lower because it is near the truth value. Whereas BIC compares the posterior probability function of a model which is being called a true model which follows a certain Bayesian system. So, a model can be selected if the BIC is lower because it is most likely to be near the truth. So, for this task I will be using both AIC and BIC for model selection.

By analyzing the generated output of every AIC and BIC of task 2.4 and normal distribution plot of task 2.5, I find candidate model 3 as a best model from other listed models. The generated value of model 3 seems to be consistent and lower for both AIC and BIC also the plot of model 3 seems to be closed to normal distribution.

Task 2.7

At the beginning input(X) and output(Y) data provided has been divided into two parts. One of which is testing dataset, and training dataset. Each dataset was divided according to information provided i.e., 70% for training and rest for testing purposes. Estimation of model parameters was carried out on training data with the help of selected best model i.e., model 3 where the value of q was calculated. Model output/prediction was performed on testing data after estimation of model parameter by calculating the RSS. After calculation of RSS and model prediction, confidence interval was performed with 95%.

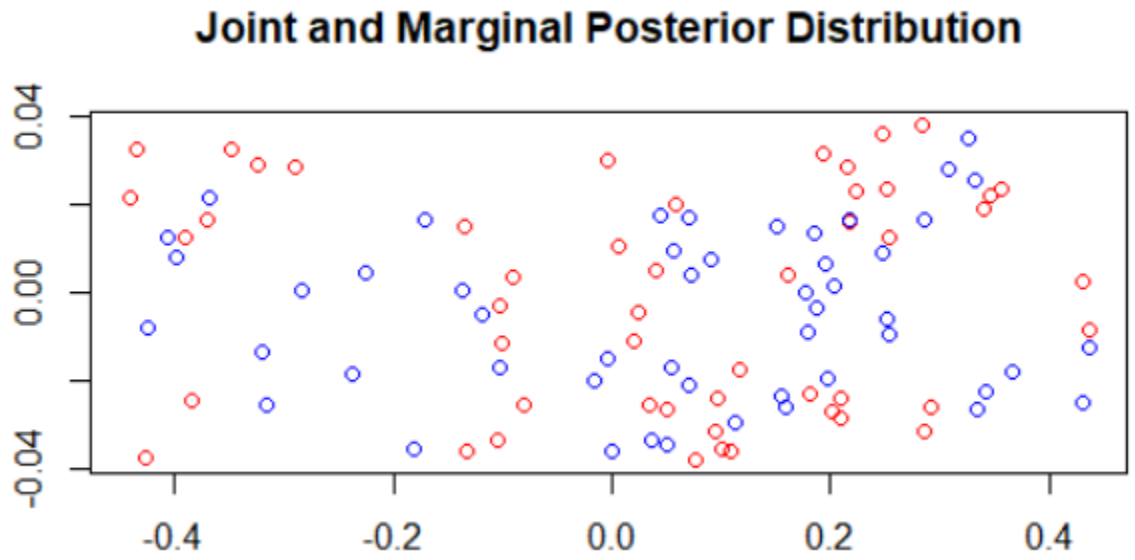


Task 3

Approximate Bayesian computation is a type of method that can help to calculate and evaluate a posterior distribution of a model parameter without calculating likelihood parameter. We are using the ABC method to compute the posterior distribution and perform a rejection ABC with the help of the best model selected from task 2.

While completing task 2 I have selected candidate model 3 as a regression module which I will be using to complete this task. From model 3 two values were selected for estimating model parameters which was carried out on task 2.1 for computing posterior distribution. The selected parameter must be the highest possible value. With the help of least squares of model 3 theta bias and theta one was selected for calculating posterior value. Two values of estimated parameters are kept constant.

After selecting different values, the range of those values i.e. chosen parameter should be calculated with the help of runif function that is predefined in R programming which provides 2 new values. Those 2 new values can be used for calculating Y_hat and error of output signal by combining new value and constant value kept in the estimated parameter. Since we generated the Y_hat and error, RSS can also be calculated just like task 2.2.
After calculating the RSS, we can perform rejection ABC. While performing rejection ABC if the value of RSS is greater than threshold value then those values are rejected. Else if RSS is smaller than the threshold then those values are accepted and stored for plotting. Both parameters are plotted which is shown below.



Overall, in this task, 2 parameters with highest value from mode 3 were selected for calculating the range in order to perform rejection sampling and identifying the range of estimated parameters. After calculating the range, Y_hat and error were generated to find RSS just like in task 2.2. After calculating the RSS rejection ABC was performed where values having higher RSS then estimated threshold were rejected and RSS with less threshold value were accepted and plotted. Those plots are provided above.

Conclusion

At the beginning, I was asked to select the best regression model from all those candidate models. So, while carrying out each task model 3 was selected as the best regression model.
For selecting the best regression model R programming was used and all processes to obtain the result are provided in the Appendix section. Where all code is provided.

Appendix

```
## All the linrary used for this task.
library(matlib)
library(ggplot2)
library(rsample)
# Importing X data
X=as.matrix(read.csv(file="X-data.csv",header = F))
colnames(X)<-c("X1","X2","X3","X4")

# Importing Y data
Y=as.matrix(read.csv(file="Y-data.csv",header = F))
colnames(Y)<-c("Y")

# Importing Time data
Time = read.csv("Time-data.csv", header = F, skip = 1)
Time = as.matrix(rbind(0, Time))
```

Task 1: Creating a time series plot

```
#Defining the value of X and Y against time for timeseries plot.
X.ts<-ts(X,start = c(min(Time),max(Time)),frequency =1)
Y.ts<-ts(Y,start = c(min(Time),max(Time)),frequency =1)

#Plotting those graphs
plot(X.ts,main = "Time series plot of X Signal", xlab = "Time", ylab = "Input signal")
plot(Y.ts,main = "Time series plot of Y Signal", xlab = "Time", ylab = "Output signal")
```

##Task 1: Creating a density plot with the help of histogram

```
#Creating a density if X signal
dis=density(X)
plot(dis,main = "Density plot of whole input signal")
# Creating a Histrogram of X signal
hist(X,freq = FALSE,main = "DEn")
#Adding density in the histogram
lines(dis,lwd=2,col="red")
rug(jitter(X))

#Creating a density if X1 signal
dis_X1=density(X["X1"])
hist(X["X1"],freq = FALSE,main = "Histogram and density plot of X1",xlab = "X1 Signal")
lines(dis_X1,lwd=2,col="red")
# Add the data-poins with noise in the X-axis
rug(jitter(X["X1"]))

#Creating a density if X2 signal
dis_X2=density(X["X2"])
hist(X["X2"],freq = FALSE,main = "Histogram and density plot of X2",xlab = "X2 Signal")
```

```
lines(dis_X2,lwd=2,col="red")
rug(jitter(X["X2"]))
#Creating a density if X3 signal
dis_X3=density(X["X3"])
hist(X["X3"],freq = FALSE,main = "Histogram and density plot of X3",xlab = "X3 Signal")
lines(dis_X3,lwd=2,col="red")
rug(jitter(X["X3"]))

#Creating a density if X3 signal
dis_X4=density(X["X4"])
hist(X["X4"],freq = FALSE,main = "Histogram and density plot of X4",xlab = "X4 Signal")
lines(dis_X1,lwd=2,col="red")
rug(jitter(X["X4"]))

#Creating a density if Y signal
dis_y=density(Y)
plot(dis_y,main = "Density plot of Y",,xlab = "Output Signal")
hist(Y,freq = FALSE,main = "Histogram and density plot of Y",xlab = "Output Signal")
lines(dis_y,lwd=2,col="red")
rug(jitter(Y))
```

Task 1: Creating Correlation and Scatter plot.

```
# Plotting X1 against Y
plot(X["X1"],Y,main = "Correlation betweeen X1 and Y signal", xlab = "X1 signal", ylab = "Output
signal" )
# Plotting X2 against Y
plot(X["X2"],Y,main = "Correlation betweeen X2 and Y signal", xlab = "X2 signal", ylab = "Output
signal")
# Plotting X3 against Y
plot(X["X3"],Y,main = "Correlation betweeen X3 and Y signal", xlab = "X3 signal", ylab = "Output
signal")
# Plotting X4 against Y
plot(X["X4"],Y,main = "Correlation betweeen X4 and Y signal", xlab = "X4 signal", ylab = "Output
signal")
```

#Task 2

```
# Calculating ones for binding the data
ones = matrix(1 , length(X)/4,1)
ones
```

#Task 2.1

```
#Calculating thetahat of Model 1
#Binding data from equation of model 1.
X_model1<-cbind(ones,(X["X4"]),(X["X1"])^2,(X["X1"])^3,(X["X3"])^4)
X_model1
#Calculating thetahat of model 1
```

```
Model1_thetahat=solve(t(X_model1) %*% X_model1) %*% t(X_model1) %*% Y
Model1_thetahat

#Model 2
#Binding data from equation of model 2.
X_model2<-cbind(ones,(X["X3"])^3,(X["X3"])^4)
X_model2
#Calculating thetahat of Model 2
Model2_thetahat=solve(t(X_model2) %*% X_model2) %*% t(X_model2) %*% Y
Model2_thetahat

#For model 3
#Binding data from equation of model 3.
X_model3<-cbind(ones,X["X2"],(X["X1"])^3,(X["X3"])^4)
X_model3
#Calculating thetahat of Model 3
Model3_thetahat=solve(t(X_model3) %*% X_model3) %*% t(X_model3) %*% Y
Model3_thetahat

#For model 4
#Binding data from equation of model 4.
X_model4<-cbind(ones,X["X4"],X["X1"]^3,X["X3"]^4)
X_model4
#Calculating thetahat of Model 4
Model4_thetahat=solve(t(X_model4) %*% X_model4) %*% t(X_model4) %*% Y
Model4_thetahat

#Binding data from equation of model 5.
X_model5<-cbind(ones,X["X4"],X["X1"]^2,X["X1"]^3,X["X3"]^4,X["X1"]^4)
X_model5
#Calculating thetahat of Model 5
Model5_thetahat=solve(t(X_model5) %*% X_model5) %*% t(X_model5) %*% Y
Model5_thetahat
```

##Task 2.2

```
#Calculating Y-hat and RSS Model 1
Y_hat_m1 = X_model1 %*% Model1_thetahat
Y_hat_m1
#Calculating RSS
RSS_Model_1=sum((Y-Y_hat_m1)^2)
RSS_Model_1

#Calculating Y-hat and RSS of model 2
Y_hat_m2 = X_model2 %*% Model2_thetahat
Y_hat_m2
RSS_Model_2=sum((Y-Y_hat_m2)^2)
RSS_Model_2
```

```
#Calculating Y-hat and RSS of model 3
Y_hat_m3 = X_model3 %*% Model3_thetahat
Y_hat_m3
RSS_Model_3=sum((Y-Y_hat_m3)^2)
RSS_Model_3

#Calculating Y-hat and RSS of model 4
Y_hat_m4 = X_model4 %*% Model4_thetahat
Y_hat_m4
RSS_Model_4=sum((Y-Y_hat_m4)^2)
RSS_Model_4

#Calculating Y-hat and RSS of model 5
Y_hat_m5 = X_model5 %*% Model5_thetahat
Y_hat_m5
RSS_Model_5=sum((Y-Y_hat_m5)^2)
RSS_Model_5
```

Task 2.3 Calculating likelihood and Variance of each model

```
N=length(Y)
#Calculating the Variance of Model 1
Variance_model1=RSS_Model_1/(N-1)
Variance_model1
#Calculating the log-likelihood of Model 1
likehood_Model_1=
-(N/2)*(log(2*pi))-(N/2)*(log(Variance_model1))-(1/(2*Variance_model1))*RSS_Model_1
likehood_Model_1

#Calculating Variance and log-likelihood of Model 2
Variance_model2=RSS_Model_2/(N-1)
Variance_model2

likehood_Model_2=
-(N/2)*(log(2*pi))-(N/2)*(log(Variance_model2))-(1/(2*Variance_model2))*RSS_Model_2
likehood_Model_2

#Calculating Variance and log-likelihood of Model 3
Variance_model3=RSS_Model_3/(N-1)
Variance_model3
likehood_Model_3=
-(N/2)*(log(2*pi))-(N/2)*(log(Variance_model3))-(1/(2*Variance_model3))*RSS_Model_3
likehood_Model_3

#Calculating Variance and log-likelihood of Model 4
Variance_model4=RSS_Model_4/(N-1)
Variance_model4
```

```
likehood_Model_4=
-(N/2)*(log(2*pi))-(N/2)*(log(Variance_model4))-(1/(2*Variance_model4))*RSS_Model_4
likehood_Model_4

#Calculating Variance and log-likelihood of Model 5
Variance_model5=RSS_Model_5/(N-1)
Variance_model5
likehood_Model_5=
-(N/2)*(log(2*pi))-(N/2)*(log(Variance_model5))-(1/(2*Variance_model5))*RSS_Model_5
likehood_Model_5
```

Task 2.4 Calculating AIC And BIC of Each model

```
##Calculating AIC and BIC of model 1
K_model1<-length(Model1_thetahat)
K_model1
AIC_model1=2*K_model1-2*likehood_Model_1
AIC_model1
BIC_model1=K_model1*log(N)-2*likehood_Model_1
BIC_model1

## thetahat of model 2
K_model2<-length(Model2_thetahat)
K_model2
##Calculating AIC and BIC of model 2
AIC_model2=2*K_model2-2*likehood_Model_2
AIC_model2
BIC_model2=K_model2*log(N)-2*likehood_Model_2
BIC_model2

## thetahat of model 3
K_model3<-length(Model3_thetahat)
K_model3

##Calculating AIC and BIC of model 3
AIC_model3=2*K_model3-2*likehood_Model_3
AIC_model3
BIC_model3=K_model3*log(N)-2*likehood_Model_3
BIC_model3

## thetahat of model 4
K_model4<-length(Model1_thetahat)
K_model4

##Calculating AIC and BIC of model 4
AIC_model4=2*K_model4-2*likehood_Model_4
AIC_model4
BIC_model4=K_model4*log(N)-2*likehood_Model_4
```

```
BIC_model4

## thetahat of model 5
K_model5<-length(Model5_thetahat)
K_model5

##Calculating AIC and BIC of model 5
AIC_model5=2*K_model5-2*likehood_Model_5
AIC_model5
BIC_model5=K_model5*log(N)-2*likehood_Model_5
BIC_model5
```

Task 2.5

```
## Error of model1
model1_error <- Y-Y_hat_m1
## Plotting the graph QQplot and QQ line of model 1
qqnorm(model1_error, col = "darkcyan",main = "QQ plot of model 1")
qqline(model1_error, col = "red",lwd=1)

## Error of model2
model2_error <- Y-Y_hat_m2 # error of model 2
## Plotting QQplot and QQ line of model 2
qqnorm(model2_error, col = "darkcyan",main = "QQ plot of model 2")
qqline(model2_error, col = "red")

## Error of model3
model3_error <- Y- Y_hat_m3
## Plotting QQplot and QQ line of model 3
qqnorm(model3_error, col = "darkcyan",main = "QQ plot of model 3")
qqline(model3_error, col = "red")

## Error of model4
model4_error <- Y-Y_hat_m4
## Plotting QQplot and QQ line of model 4
qqnorm(model4_error, col = "darkcyan",main = "QQ plot of model 4")
qqline(model4_error, col = "red")

## Error of model5
model5_error <- Y- Y_hat_m5
## Plotting QQplot and QQ line of model 5
qqnorm(model5_error, col = "darkcyan",main = "QQ plot of model 5")
qqline(model5_error, col = "red")
```

Task 2.7

```
## Spliting the data of y into 2 form i.e. Traning and testing data set.
split_Y<-initial_split(data = as.data.frame(Y),prop=.7)
```

```
## Traning Y data split
Y_training_set<-training(split_Y)
Y_testing_set<-as.matrix(testing(split_Y))
## Testing Y data split
Y_training_data<-as.matrix(Y_training_set)

## Spliting the data of y into 2 form i.e. Traning and testing data set.
split_X<-initial_split(data = as.data.frame(X),prop=.7)
## Traning X data split
X_training_set<-training(split_X)
## Testing X data split
X_testing_set<-as.matrix(testing(split_X))
X_testing_data<-as.matrix(X_testing_set)
X_training_data<-as.matrix(X_training_set)

### Estimating model parameters using Traning set
traning_ones=matrix(1 , length(X_training_set$X1),1)
X_traning_model<-cbind(traning_ones,X_training_set[,"X2"],(X_training_set[,"X1"])^3,(X_training_set[
,"X3"])^4)
traning_thetahat=solve(t(X_traning_model) %*% X_traning_model) %*% t(X_traning_model) %*%

### Model out/Prediction
Y_testing_hat = X_testing_data %*% traning_thetahat
Y_testing_hat
RSS_testing=sum((Y_testing_set-Y_testing_hat)^2)
RSS_testing

t.test(Y_training_data, mu=500, alternative="two.sided", conf.level=0.95)

C_I1=-0.2783950
C_I2=0.2762101

p2 <- plot(density(Y_training_data), col="blue", lwd=2,
         main="Distribution of Traning Data")
abline(v=C_I1,col="red", lty=2)
abline(v=C_I2,col="red", lty=2)

thetaHat_training =solve(t(X_training_data) %*% X_training_data) %*% t(X_training_data) %*%
Y_training_data
thetaHat_training
length(thetaHat_training)


dis_test=density(Y_training_data)
plot((dis_test))
plot(dis_test,main = "Density plot of Y Signal")
```

```
### Calculating Confidential interval
z=1.96 ##(95%) Confidential interval

error=((Y_testing_set-Y_testing_hat))
n_len=length(Y_testing_hat)
C_I_1= z * sqrt( (error * (1-error) ) / n_len)
C_I_1

error
C_I_2= z*sqrt((error*(1+error)/n_len)
```

##Task 3

```
## Model 3 will be used, parameter are selected and kept constant.
rr_1=0
arr_2=0
f_value=0
s_value=0
Model3_thetahat
#values from thetahat
thetebias <- 0.448299550 #choosen parameter
thetaone <- 0.038109255 # chosen prarameter
thetatwo <- 0.009827804 # constant value
thetafour <- 0.002092558 # constant value
Epison <- RSS_Model_3 * 2 ## fixing value of eplision
num <- 100 #number of iteration

##Calculating Y-hat for performing rejection ABC
counter <- 0
for (i in 1:num) {
 range1 <- runif(1,-0.448299550,0.448299550) # calculating the range
 range1
 range2 <- runif(1,-0.038109255,0.038109255)
 New_thetahat <- matrix(c(range1,range2,thetatwo,thetafour))
 New_Y_Hat <- X_model3 %*% New_thetahat ##  New Y hat
  new_RSS <- sum((Y-New_Y_Hat)^2)
 new_RSS
 if (new_RSS > Epison){
  arr_1[i] <- range1
  arr_2[i] <- range2
  counter = counter+1
   f_value <- matrix(arr_1)
  s_value <- matrix(arr_2)
 }
}

hist(f_value)
```

```
hist(s_value)
###ploting the graph
plot(f_value,s_value, col = c("red", "blue"), main = "Joint and Marginal Posterior Distribution")
```

Bibliography

Brownlee, J. (2021) *Confidence Intervals For Machine Learning* [online] available from <https://machinelearningmastery.com/confidence-intervals-for-machine-learning/> [7 March 2021]

Approximate Bayesian Computation (2021) available from <https://en.wikipedia.org/wiki/Approximate_Bayesian_computation> [4 March 2021]

AIC Vs. BIC – The Methodology Center (2021) available from <https://www.methodology.psu.edu/resources/AIC-vs-BIC/> [5 March 2021]

Understanding Q-Q Plots | University Of Virginia Library Research Data Services + Sciences (2021) available from <https://data.library.virginia.edu/understanding-q-q-plots/> [6 March 2021]