**Modelling EEG signals using polynomial regression – (7089CEM)**

**Submitted To** :- Dr. Fie Hei

**Name** :- ASHISH SOLANKI

**Student** **ID** :- 10889538

Coventry University , Coventry, U.K

**Table of Contents**

1. Introduction
2. Preliminary Data Analysis
3. Regression
4. AIC and BIC
5. ABC
6. Appendix
7. References

**Introduction**

The main focus of this assignment is to fit and determine the best possible nonlinear regression model that can well relate the relationship among the input predictors (EEG signals). Through modelling and analysing the four input EEG time-series data 𝐗 = {𝐱1, 𝐱2, 𝐱3, 𝐱4}, and an output EEG signal 𝐲. All the input predictors and output response data is subject to additive noise with unknown variance along with a time.csv file containing sampling time in seconds for all EEG data.

**Preliminary Data Analysis**

The objective of data analysis is the determine whether the data is of categorical variables or continuous variables, Descriptive data statistics for continuous variables provide details regarding mean, skewness and standard deviation.

Time Series plot are the visualisation between the time and the input EEG signals.

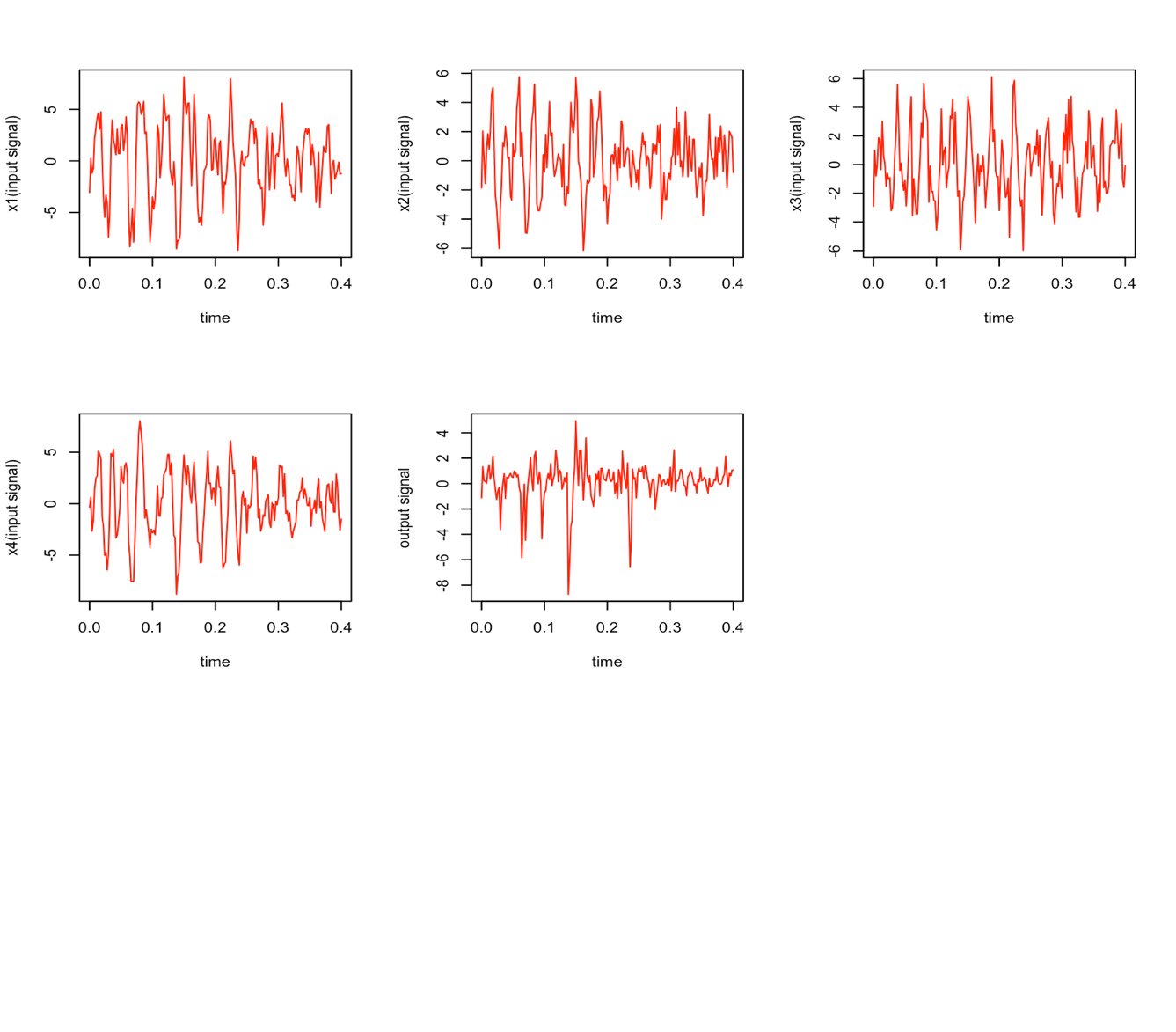


Figure 1: Four time series plots of input EEG signals and one of output EEG signal

Here we can infer that the time series plot are stationary.

**Distribution**

Using the histogram and density plot, we can analyse the input signals distribution and determine the skewness of the input signals or if is it normally distributed or not. It is very important to know the normality, skewness or kurtosis of the data before performing any modelling or regression.

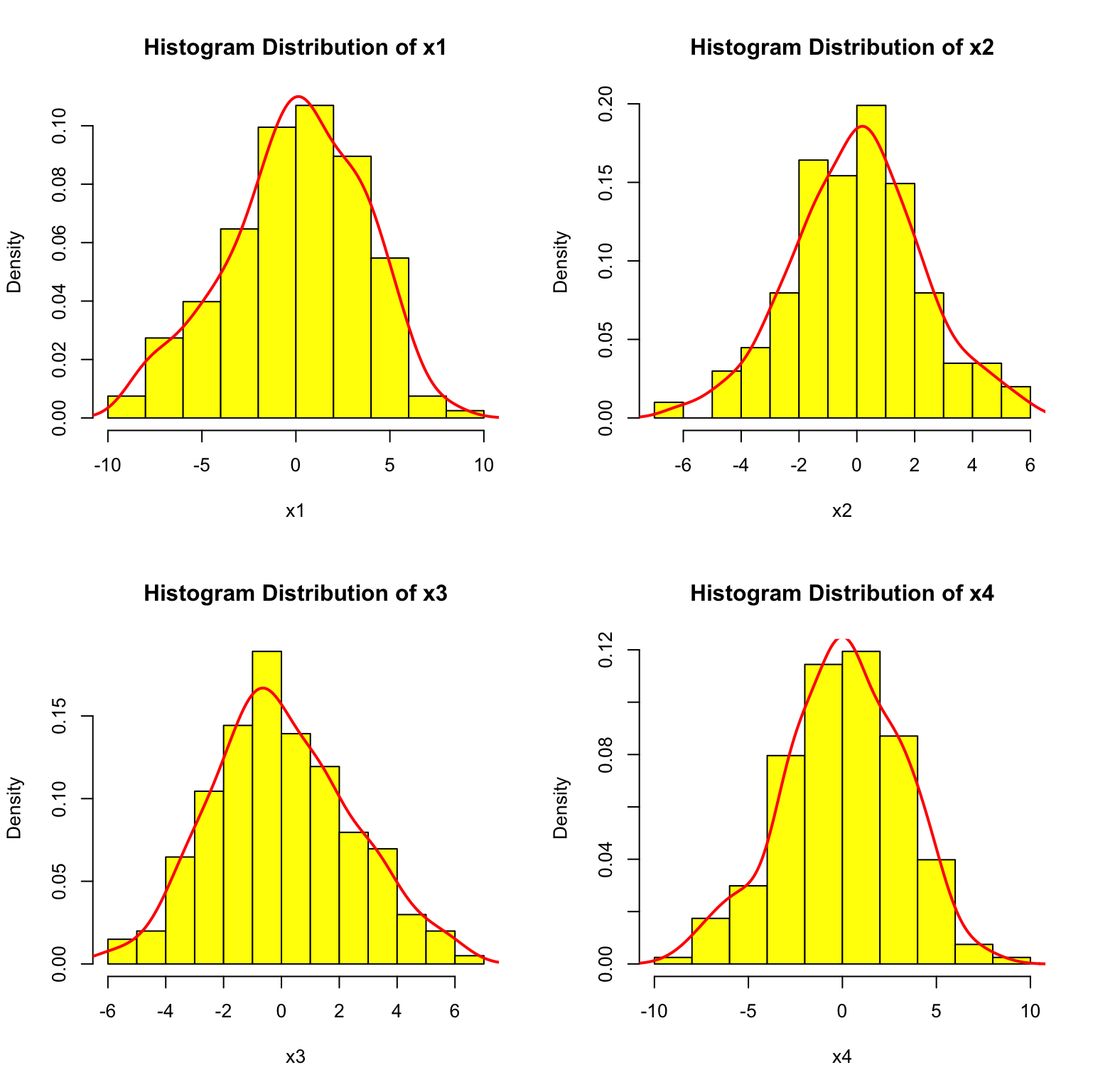
****

Figure 2:Four Histograms of input EEG signals with density plot

**Conclusion**, In Fig 2 it is visible by the smoothening line that input signals x1, x2 and x4 are normally distributed whereas, input signal x3 has a clustering of data sets at the left hand side therefore Positive Skewness.

**Correlation and Scatter Plots**

Another method of visualisation is scatter plot from which we can determine the relationship between the particular input signal with the output signal furthermore the trend showing the best fit to the data.

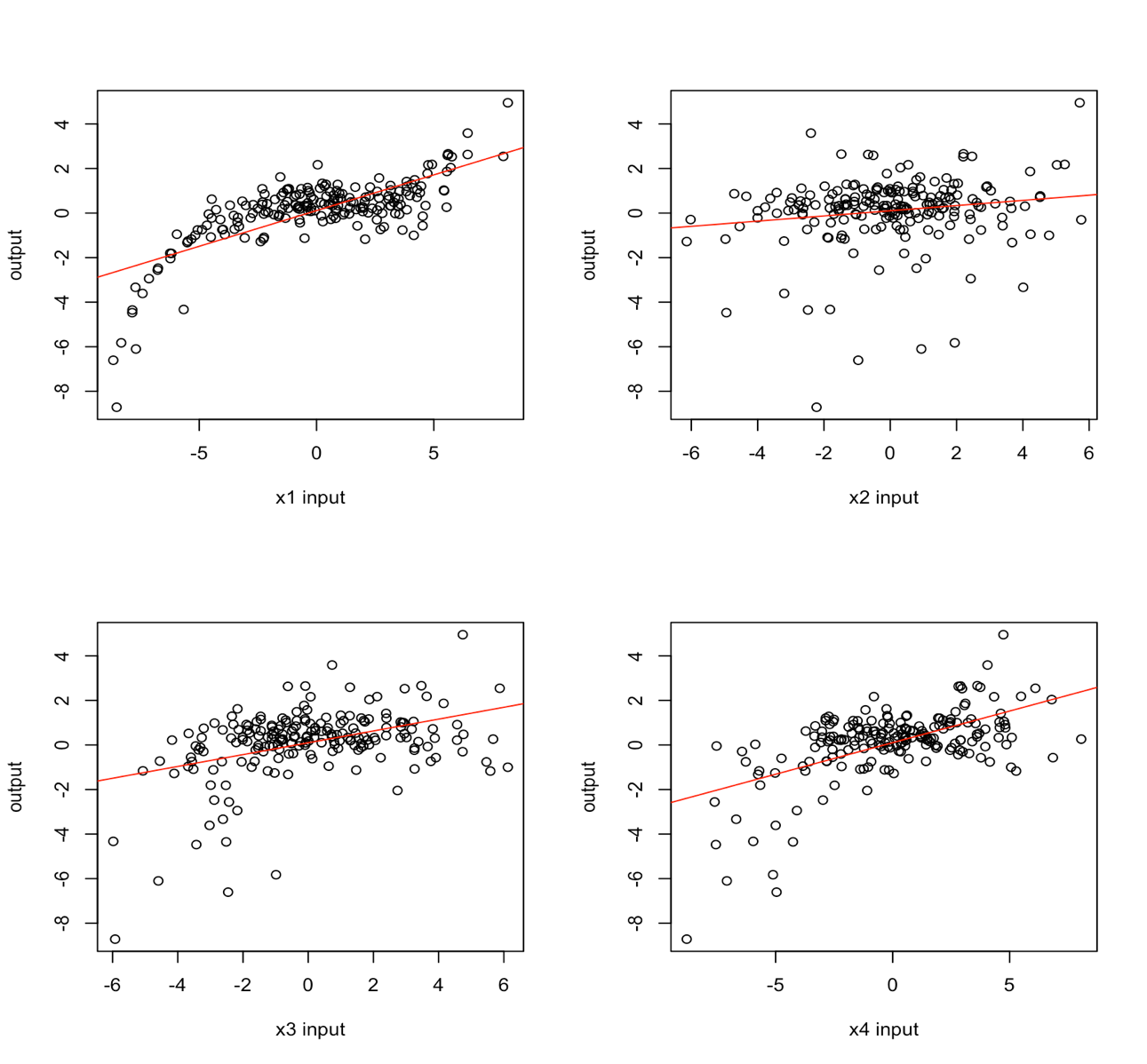


Figure 3: Scatter Plot between input EEG signals and output EEG signal

Correlation is the strength measure of a linear relationship between two variables (input EEG signal , output EEG signal) there are two types of correlation positive and negative. If the value of the correlation is close to 1 then it is the most significant variable affecting the output.

**Conclusion**, From Fig. 3 we can learn that x1(r = 0.7288) input signal has the closest value to 1 therefore it has a strong positive correlation.

**Regression**

Linear Regression is the most straight forward statistical techniques. Regression is used for two analysis forecasting and prediction. It can also determine the casual relationship between dependent variable and one or more independent variables. Like we have in the above example the input EEG signals x1, x2, x3, x4 are all independent & identically distributed and the output signal y. Here we have to find the best regression model to fit in and predict or forecast the best possible output for new observations. We are provided with 5 candidate non-linear polynomial regression models to find which is the best fit model for the data sets.

**Least square** estimate of the unknown model parameters calculates regression parameters by minimizing SSE (Sum of Squared Error), The resulting line is known as regression line. Least Square Estimates the slope and intercept of the model, but for multi-linear polynomial equation there are multiple slopes because of more than one variables.

When computing least squares we will get same number of estimates that of variables and plus one for the bias parameter.

Y hat is the predicted value of the output y in a regression equation it is also known as the response variable. It is the matrix multiplication between model matrix and parameters estimated.

**Residual Sum of Errors** (RSS) can be easily be determined based on the parameters computed above for each model to find which model has the least residual sum of errors. It is the measure of the discrepancy between the estimated model output and the data. If RSS is small for a given model then it indicates a good fit of the model to the data. It is used in model selection and parameter selection too.

RSS is the difference of the square of true value and the predicted value.

Model 1: **57.32927** Model 2: **351.4147** Model 3: **57.32536** Model 4: **57.46405** Model 5: **57.35923**

**Conclusion**, Model 3 and 5 have the highest RSS value so we can reject the model as it would not be the good fit for regression whereas, Model 3 has the lowest RSS.

**Log-Likelihood estimation** also known as MLE ( Maximum Likelihood Error ) is a method used for parameter estimating for a given distribution, using some observed data sets. MLE method is also used to learn about the best model parameters of a linear regression model. MSE choses as estimates the values of the parameters which are most consistent with the data sets.

It is most of the times easier to minimise the negative of the log-likelihood rather than minimise it. So we put a minus sign in front of log-likelihood to give us the negative log-likelihood. We use log-likelihood function instead of likelihood function because it is quite often easier to work on the log of the density mathematically.

**Akaike information criterion (AIC) and Bayesian information criterion (BIC)**

AIC and BIC both are the estimation of sample prediction error. They provide a measure of estimation for the model if it may be in its prediction if applied for the whole data set (population) that our training data aims to represent. The best model is that one which neither overfits nor underfits the population. So the lowest measurement of AIC/BIC for a model is to be selected

Both AIC/BIC get similar results mostly. They are an approach to find the balance between complexity and a good fit for a model.

Model 1: **328.2626, 344.7791** | Model 2: **688.7068, 698.6168** | Model 3**: 326.2489, 339.4621**

Model 4: **326.7346, 339.9478** | Model 5: **-13308.77, -13295.56**

**Distribution of model prediction errors**

For distribution we can plot the histogram of residual values for the models given. This will help us understand the distribution if it is gaussian or not.

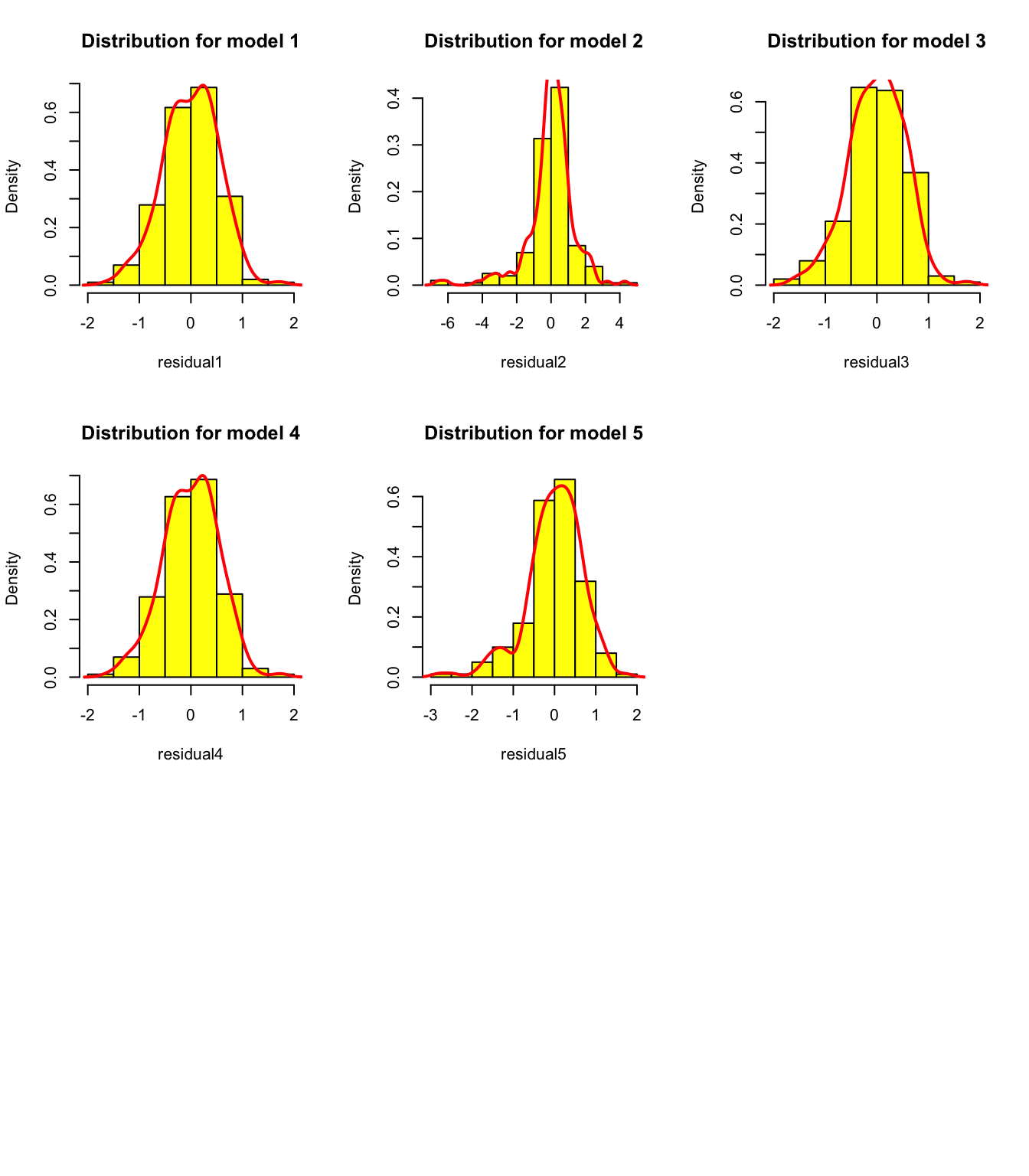


Figure 4: Distribution for Residual of all five models

Q- Q Plot is a type of diagnostic of the residuals of the model, it plots quantiles of the data versus quantiles of a distribution. When we assess the plot we want the data points to be ideally clustered to the zero line which means smaller residuals. And what we don’t want is any pattern where the residual is increasing or decreasing from the line with our predicted values Or any kind of pattern where the residual looks non-linear.

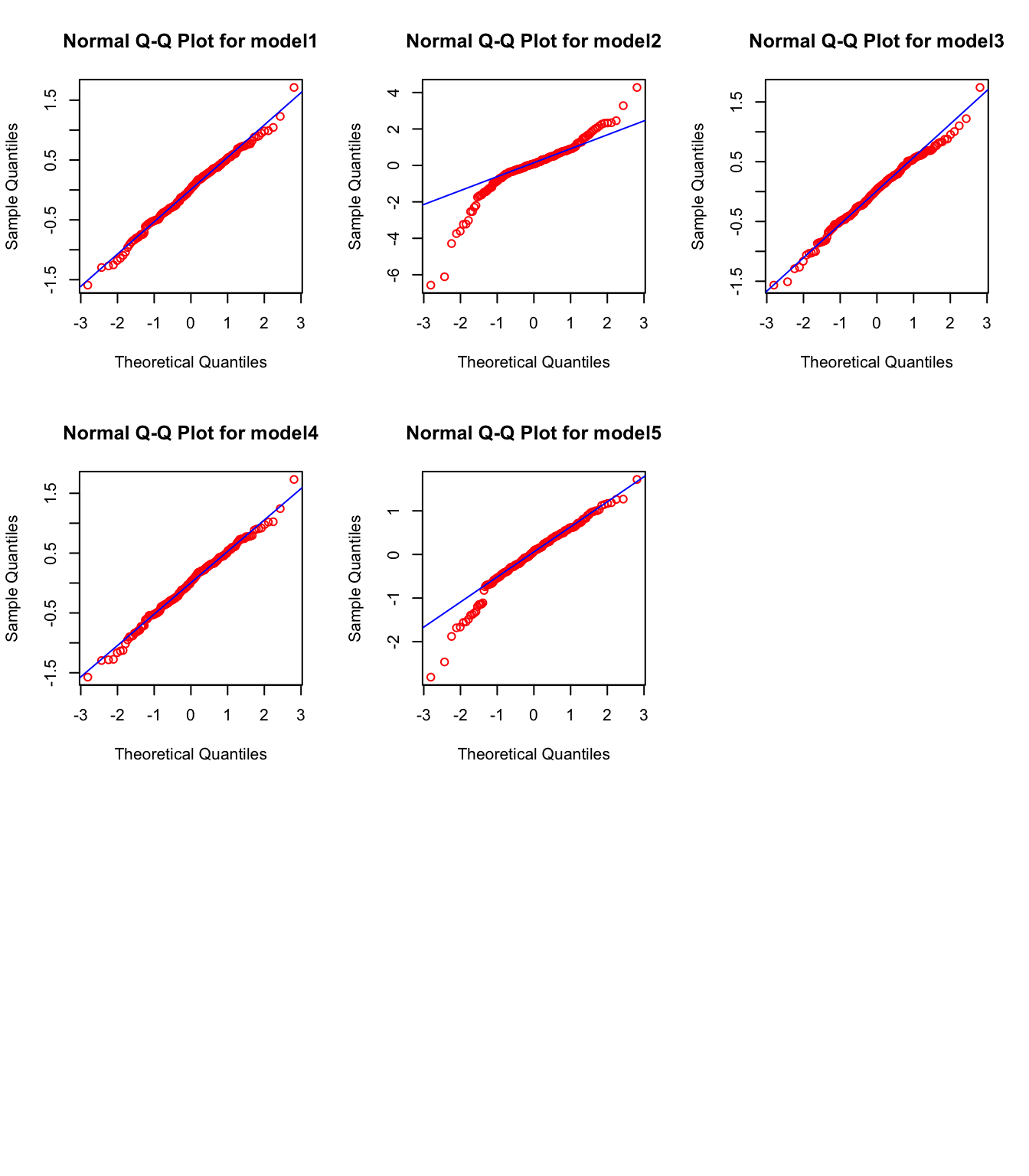


Figure 5: Q-Q Plot of the residuals for all five models

Normal Q-Q plot is to determine if our residual is normally distributed or not. If our data points are closely following the straight line in an upward trend and the data points are not deviating that looks like a S-shaped or bending at either end.

In Fig 5 we can clearly see that model 1, model 3 and model 4 are showing a normal distribution whereas, model 2 and model 5 we can see a S-Shaped pattern and a decreasing residual at the left side of the plot respectively.

Now, we have to find which has the least residual among the three models (1,3,4) which we can find by using boxplot, help us find the number of outliers and the model residual having less number of outliers. The Outliers affects the assumptions and results.

This type of plot helps us to easily detect outliers and tells us if our data is symmetrical or how tightly our data is grouped.

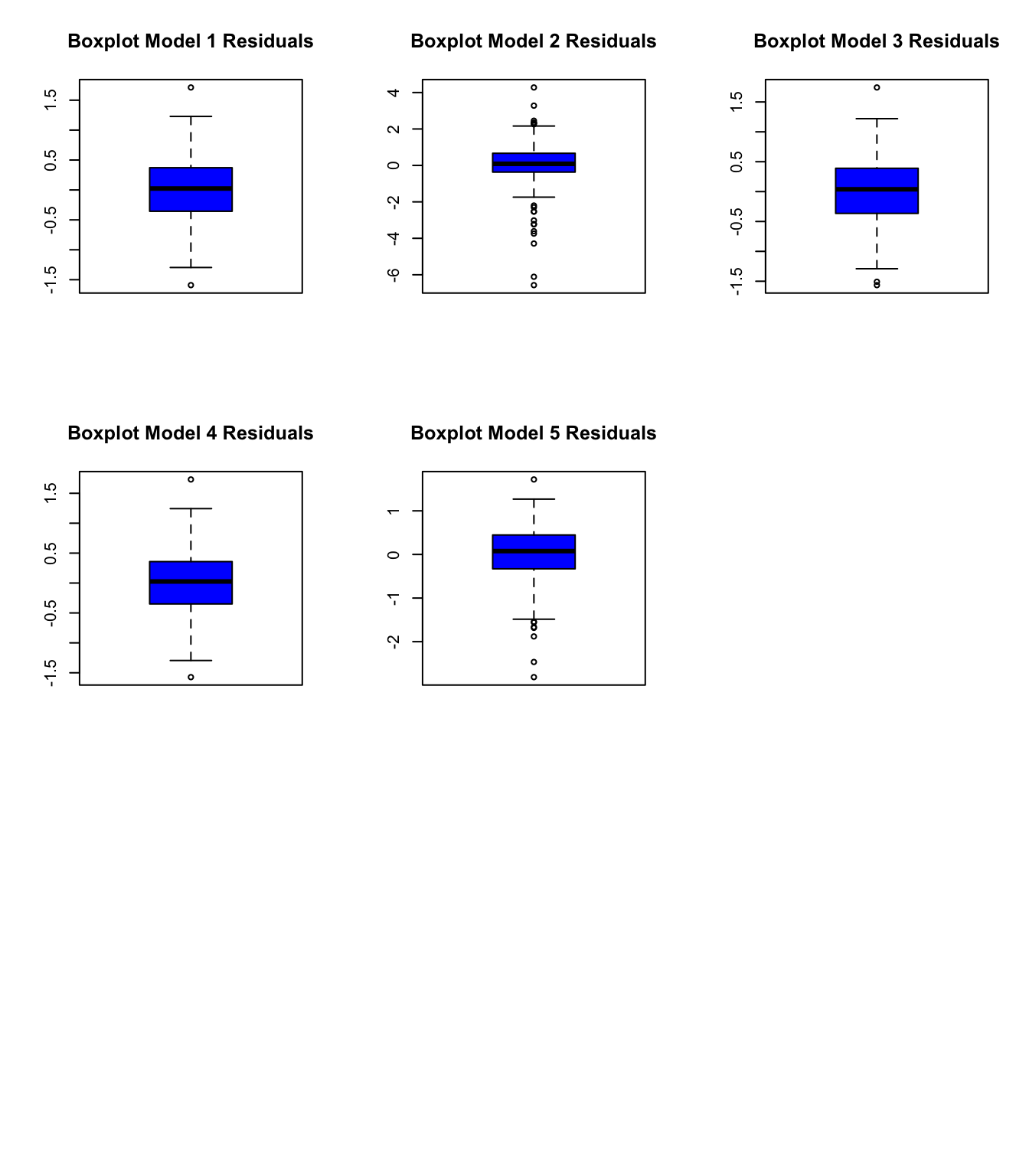


Figure 6: Box-Plot for the residuals

**Conclusion**, From Fig 6 we can eliminate model 2 and 5 because they do not have a normal distribution and also there are too many outliers which affects our prediction if we use these models. Furthermore, they will result in underfitting model for the testing data.

Now, we have to select a model best fit for our population data using the AIC , BIC and distribution of residuals from the five models we were provided with.

We will use Statistical Approaches to estimate how complex a model is and how well it fits with a dataset.

* + Akaike Information Criterion (AIC)
  + Bayesian Information Criterion (BIC)
  + Model Residual Distribution

In AIC model selection approach, we simply select a model with the smallest AIC. Model 3 has the smallest value of all that is **326.2489** among all the other models.

In BIC the quantity calculated is different from that of AIC rather proportional to AIC, but for BIC the more complex model will have the worse (larger) value and hence it will not be selected. Model 3 has the smallest value of all that is **339.4621.**

Model Residual Distribution shows us the normality and helps us find the outliers, from Fig 5 - model3 and the RSS value of model 3 (**57.32536** ). we deduce that it is the most normal distribution among the other models.

**Conclusion**, Model 3 is the best model for our population data set and is a good fit for prediction because it has the lowest residual value and AIC and BIC values too. Because BIC value we can also compute that the model is not too complex and from AIC that it is not too simple to be used on testing data.

Now, Let’s create sample training data and testing data from our population data set used previously. Here we will be using 70% random data from population for training and 30% for testing.

From Model selection it is clear that we will be using model 3 for prediction and estimation. After we have split all the input EEG signals and output EEG signal into 70:30 proportion. Estimate the Model Parameters for the observations in training data using Least Squares.

Now to compute the model’s prediction on the testing data we just need to follow substitute the estimated parameter from the training data into the linear regression model generated for testing data sets.

So, here we have used the slopes and intercept of training data and associated them with the testing data variables which gives us a predicted function.

We can also compute the Confidence interval for the testing data to check the residual error for each data point.

Here we will calculate the 95% CI for the testing data.

**CI = 5.422729**

ABC is used to estimate the posterior distribution of the model parameters. For complex models calculating likelihood is mere impossible.

Now, for the selected best model we will perform the Approximation Bayesian Computation (ABC) Firstly lets, compute two largest absolute parameters from the least square estimation and fix all the other values as zero’s (constant).

Theta1 : **0.038109255** and Theta bias: **0.448299550**

We will be using runif() function to calculate the 100 uniform distribution values in the range of the parameters that is for theta1 (0 to 0.5) and for theta bias (0,0.1).

Now, we have to find the Y\_Hat for the model 3 and parameters we have computed for the ABC. This will give us the residual error which is the difference between output values and Y\_Hat of ABC.

After this we need to draw the samples for accepting the parameter or rejecting the parameter for this we run a for loop in residual error if the value of error is less than equal to epsilon value(0.1) then it will be accepted and if it is greater than epsilon it is rejected.

Now we will be using the 2D Contour plot to between the values of 2 parameters we have selected using contourPlot library here x axis is first parameter samples y axis is second parameter samples and z axis is the accepted values of residual error using ABC.

**APPENDIX**

**R Code**

# Coursework

#libraries Used

library(ggplot2)

library(tidyverse)

library("contourPlot")

#TASK 1

# reading the input signals

X\_file = read.table("X.csv", header = F, sep = ",")

# reading output signals

y\_file = read.table("y.csv", header = F, sep = ",")

# reading time data

time\_file =read.table("time.csv", header = F, sep = ",")

#length of data

n = length(X\_file[ ,1])

#slicing each column of x

x1 = matrix(X\_file[ ,1], nrow = n, ncol = 1)

x2 = matrix(X\_file[ ,2], nrow = n, ncol = 1)

x3 = matrix(X\_file[ ,3], nrow = n, ncol = 1)

x4 = matrix(X\_file[ ,4], nrow = n, ncol = 1)

#time matrix

time\_matrix = matrix(time\_file[, 1], n , 1)

#X- input matrix

X\_matrix = cbind(x1,x2,x3,x4)

#y- output matrix

y\_matrix = matrix(y\_file[ ,1], n, 1)

#X and Y matrix

X\_y\_matrix = cbind(x1,x2,x3,x4,y\_matrix)

X\_y\_matrix

# plotting time-series between x1,x2,x3,x4 and time

par(mfrow = c(3,3))

plot(time\_matrix, x1, type = "l", col = "red", xlab = "time", ylab = "x1(input signal)")

plot(time\_matrix, x2, type = "l", col = "red", xlab = "time", ylab = "x2(input signal)")

plot(time\_matrix, x3, type = "l", col = "red", xlab = "time", ylab = "x3(input signal)")

plot(time\_matrix, x4, type = "l", col = "red", xlab = "time", ylab = "x4(input signal)")

plot(time\_matrix, y, type = "l", col = "red", xlab = "time", ylab = "output signal")

#distribution of EEG signals

par(mfrow = c(2,2))

hist(x1, main = "Histogram Distribution of x1", freq = FALSE, col = 'yellow')

lines(density(x1), col = 'red', lwd = 2)

hist(x2, main = "Histogram Distribution of x2", freq = FALSE, col = 'yellow')

lines(density(x2), col = 'red', lwd = 2)

hist(x3, main = "Histogram Distribution of x3", freq = FALSE, col = 'yellow')

lines(density(x3), col = 'red', lwd = 2)

hist(x4, main = "Histogram Distribution of x4", freq = FALSE, col = 'yellow')

lines(density(x4), col = 'red', lwd = 2)

#correlation scatter plot

pairs(X)

par(mfrow = c(2,2))

plot(x1,y, ylab = 'output', xlab = 'x1 input')

abline(lm(y~x1), col = "red")

plot(x2,y, ylab = 'output', xlab = 'x2 input')

abline(lm(y~x2), col = "red")

plot(x3,y, ylab = 'output', xlab = 'x3 input')

abline(lm(y~x3), col = "red")

plot(x4,y, ylab = 'output', xlab = 'x4 input')

abline(lm(y~x4), col = "red")

# Correlation between inputs and outputs

cor.test(x1,y)

cor.test(x2,y)

cor.test(x3,y)

cor.test(x4,y)

#----------------------------------------------

#TASK 2

# Regression

# TASK 2.1 - Estimate model parameter

#------------------------------------------------------

# Model 1 - y = w1x4 + w2x1^2 + w3x1^3 + w4x3^4 + w\_bias + E

# Model Matrix 1

first\_col\_X1 = matrix(x4, n, 1)

second\_col\_X1 = matrix(x1^2, n, 1)

third\_col\_X1 = matrix(x1^3, n, 1)

fourth\_col\_X1 = matrix(x3^4, n, 1)

fifth\_col\_X1 = matrix(1, n, 1)

model1\_X = cbind(first\_col\_X1, second\_col\_X1, third\_col\_X1, fourth\_col\_X1, fifth\_col\_X1)

#Least Square for model 1

thetaHat1 = solve(t(model1\_X) %\*% model1\_X) %\*% t(model1\_X) %\*% y\_matrix

thetaHat1

yHat1 = model1\_X %\*% thetaHat1

#Task 2.2 - RSS for model 1

RSS1 = sum((y\_matrix - yHat1)^2)

RSS1

#Task 2.3 - log-likelihood function for model 1

sigma\_square1 = RSS1/(n-1)

log\_like1 = -n/2\*(log(2\*pi)) - n/2 \* log(sigma\_square1) - (n-1)/2

log\_like1

#Task 2.4 - AIC BIC for model 1

k1 = 5

AIC1 = 2\*k1 - 2\*log\_like1

BIC1 = log(n)\*k1 - 2\*log\_like1

#Task 2.5

# model 1

residual1 = y\_matrix-yHat1

par(mfrow = c(3,3))

hist(residual1, col = 'yellow', main = "Distribution for model 1", freq = FALSE)

lines(density(residual1), col = 'red', lwd = 2)

hist(residual2, col = 'yellow', main = "Distribution for model 2", freq = FALSE)

lines(density(residual2), col = 'red', lwd = 2)

hist(residual3, col = 'yellow', main = "Distribution for model 3", freq = FALSE)

lines(density(residual3), col = 'red', lwd = 2)

hist(residual4, col = 'yellow', main = "Distribution for model 4", freq = FALSE)

lines(density(residual4), col = 'red', lwd = 2)

hist(residual5, col = 'yellow', main = "Distribution for model 5", freq = FALSE)

lines(density(residual5), col = 'red', lwd = 2)

par(mfrow = c(3,3))

qqnorm(residual1, col = 'red', main = 'Normal Q-Q Plot for model1')

qqline(residual1, col = 'blue', lwd = 1)

qqnorm(residual2, col = 'red', main = 'Normal Q-Q Plot for model2')

qqline(residual2, col = 'blue', lwd = 1)

qqnorm(residual3, col = 'red', main = 'Normal Q-Q Plot for model3')

qqline(residual3, col = 'blue', lwd = 1)

qqnorm(residual4, col = 'red', main = 'Normal Q-Q Plot for model4')

qqline(residual4, col = 'blue', lwd = 1)

qqnorm(residual5, col = 'red', main = 'Normal Q-Q Plot for model5')

qqline(residual5, col = 'blue', lwd = 1)

par(mfrow = c(3,3))

boxplot(residual1, main = "Boxplot Model 1 Residuals", col = 'blue')

boxplot(residual2, main = "Boxplot Model 2 Residuals", col = 'blue')

boxplot(residual3, main = "Boxplot Model 3 Residuals", col = 'blue')

boxplot(residual4, main = "Boxplot Model 4 Residuals", col = 'blue')

boxplot(residual5, main = "Boxplot Model 5 Residuals", col = 'blue')

boxplot.stats(residual5)$out

#---------------------------------------------------

# Model 2 - y = w1\*x3^3 + w2\*x3^4 + w\_bias + E

# Model Matrix 2

first\_col\_X2 = matrix(x3^3, n, 1)

second\_col\_X2 = matrix(x3^4, n, 1)

third\_col\_X2 = matrix(1, n, 1)

model2\_X = cbind(first\_col\_X2, second\_col\_X2, third\_col\_X2)

#Least Square for model 2

thetaHat2 = solve(t(model2\_X) %\*% model2\_X) %\*% t(model2\_X) %\*% y\_matrix

thetaHat2

yHat2 = model2\_X %\*% thetaHat2

#Task 2.2 - RSS for model 2

RSS2 = sum((y\_matrix - yHat2)^2)

RSS2

#Task 2.3 - log-likelihood function for model 2

sigma\_square2 = RSS2/(n-1)

log\_like2 = -n/2\*(log(2\*pi)) - n/2 \* log(sigma\_square2) - (n-1)/2

log\_like2

#Task 2.4 - AIC BIC for model 2

k2 = 3

AIC2 = 2\*k2 - 2\*log\_like2

AIC2

BIC2 = log(n)\*k2 - 2\*log\_like2

BIC2

#---------------------------------------------------

# Model 3 - y = w1\*x2 + w2\*x1^3 + w3\*x3^4 + w\_bias + E

# Model Matrix 3

first\_col\_X3 = matrix(x2, n, 1)

second\_col\_X3 = matrix(x1^3, n, 1)

third\_col\_X3 = matrix(x3^4, n, 1)

forth\_col\_X3 = matrix(1, n, 1)

model3\_X = cbind(first\_col\_X3, second\_col\_X3, third\_col\_X3, forth\_col\_X3)

#Least Square for model 3

thetaHat3 = solve(t(model3\_X) %\*% model3\_X) %\*% t(model3\_X) %\*% y\_matrix

thetaHat3

yHat3 = model3\_X %\*% thetaHat3

#Task 2.2 - RSS for model 3

RSS3 = sum((y\_matrix - yHat3)^2)

RSS3

#Task 2.3 - log-likelihood function for model 3

sigma\_square3 = RSS3/(n-1)

log\_like3 = -n/2\*(log(2\*pi)) - n/2 \* log(sigma\_square3) - (n-1)/2

log\_like3

#Task 2.4 - AIC BIC for model 3

k3 = length(thetaHat3)

AIC3 = 2\*k3 - 2\*log\_like3

AIC3

BIC3 = log(n)\*k3 - 2\*log\_like3

BIC3

#-------------------------------------------------------------------

# Model 4 - y = w1\*x4 + w2\*x1^3 + w3\*x3^4 + w\_bias + E

# Model Matrix 4

first\_col\_X4 = matrix(x4, n, 1)

second\_col\_X4 = matrix(x1^3, n, 1)

third\_col\_X4 = matrix(x3^4, n, 1)

forth\_col\_X4 = matrix(1, n, 1)

model4\_X = cbind(first\_col\_X4, second\_col\_X4, third\_col\_X4, forth\_col\_X4)

#Least Square for model 4

thetaHat4 = solve(t(model4\_X) %\*% model4\_X) %\*% t(model4\_X) %\*% y\_matrix

thetaHat4

yHat4 = model4\_X %\*% thetaHat4

#Task 2.2 - RSS for model 4

RSS4 = sum((y\_matrix - yHat4)^2)

RSS4

#Task 2.3 - log-likelihood function for model 4

sigma\_square4 = RSS4/(n-1)

log\_like4 = -n/2\*(log(2\*pi)) - n/2 \* log(sigma\_square4) - (n-1)/2

log\_like4

#Task 2.4 - AIC BIC for model 4

k4 =length(thetaHat4)

k4

AIC4 = 2\*k4 - 2\*log\_like4

AIC4

BIC4 = log(n)\*k4 - 2\*log\_like4

BIC4

#---------------------------------------------------

# Model 5 - y = w1\*x4 + w2\*x1^2 + w3\*x1^3 + w4\*x3^4 + w5\*x1^4 + w\_bias + E

# Model Matrix 5

first\_col\_X5 = matrix(x4, n, 1)

second\_col\_X5 = matrix(x1^2, n, 1)

third\_col\_X5 = matrix(x1^3, n, 1)

fourth\_col\_X5 = matrix(x3^4, n, 1)

fifth\_col\_X5 = matrix(x1^4, n, 1)

sixth\_col\_X5 = matrix(1, n, 1)

model5\_X = cbind(first\_col\_X5, second\_col\_X5, third\_col\_X5, sixth\_col\_X5)

#Least Square for model 5

thetaHat5 = solve(t(model5\_X) %\*% model5\_X) %\*% t(model5\_X) %\*% y\_matrix

thetaHat5

yHat5 = model5\_X %\*% thetaHat5

#Task 2.2 - RSS for model 5

RSS5 = sum((y\_matrix - yHat5)^2)

RSS5

#Task 2.3 - log-likelihood function for model 5

sigma\_square5 = RSS5/(n-1)

log\_like5 = -n/2\*(log(2\*pi)) - n/2 \* log(sigma\_square5) - (n-1)/2

log\_like5

#Task 2.4 - AIC BIC for model 5

k5 = length(thetaHat5)

AIC5 = 2\*k5 - 2\*log\_like5

AIC5

BIC5 = log(n)\*k5 - 2\*log\_like5

BIC5

#---------------------------------------------------------------

# ----------------------------------------------------------------

#Task 2.6 - inference on report

#Task 2.7

# Splitting the data randomly into (70:30) proportion for x1

require(caTools)

sample = sample.split(x1, SplitRatio = 0.70)

training\_data\_x1 = subset(x1, sample == TRUE)

testing\_data\_x1 = subset(x1, sample == FALSE)

# Splitting the data into (70:30) proportion for x2

sample = sample.split(x2, SplitRatio = 0.70)

training\_data\_x2 = subset(x2, sample == TRUE)

testing\_data\_x2 = subset(x2, sample == FALSE)

# Splitting the data into (70:30) proportion for x3

sample = sample.split(x3, SplitRatio = 0.70)

training\_data\_x3 = subset(x3, sample == TRUE)

testing\_data\_x3 = subset(x3, sample == FALSE)

# Splitting the data into (70:30) proportion for x4

sample = sample.split(x4, SplitRatio = 0.70)

training\_data\_x4 = subset(x4, sample == TRUE)

testing\_data\_x4 = subset(x4, sample == FALSE)

# Splitting the data into (70:30) proportion for y (output)

sample = sample.split(y, SplitRatio = 0.70)

training\_data\_y = subset(y, sample == TRUE)

testing\_data\_y = subset(y, sample == FALSE)

# matrix for training data

training\_data\_matrix = cbind(training\_data\_x1, training\_data\_x2, training\_data\_x3, training\_data\_x4)

testing\_data\_matrix = cbind(testing\_data\_x1, testing\_data\_x2, testing\_data\_x3, testing\_data\_x4)

# creating the third model for training data

# Model 3 - y = w1\*x2 + w2\*x1^3 + w3\*x3^4 + w\_bias + E

n\_training\_data = length(training\_data\_matrix[,1])

first\_col\_training\_data\_model3 = matrix(training\_data\_x2, n\_training\_data, 1)

second\_col\_training\_data\_model3 = matrix((training\_data\_x1)^3, n\_training\_data, 1)

third\_col\_training\_data\_model3 = matrix((training\_data\_x3)^4, n\_training\_data, 1)

fourth\_col\_training\_data\_model3 = matrix(1, n\_training\_data, 1)

model3\_matrix\_training\_data = cbind(first\_col\_training\_data\_model3, second\_col\_training\_data\_model3, third\_col\_training\_data\_model3, fourth\_col\_training\_data\_model3)

# performing least square to estimate model parameters for training data

thetaHat\_training\_data = solve(t(model3\_matrix\_training\_data) %\*% model3\_matrix\_training\_data) %\*% t(model3\_matrix\_training\_data) %\*% training\_data\_y

yHat\_training\_data = model3\_matrix\_training\_data %\*% thetaHat\_training\_data

# compute the models prediction on the testing data

#length for testing data

n\_testing\_data = length(testing\_data\_x1[, 1])

#model for testing data

first\_col\_testing\_data\_model3 = matrix(testing\_data\_x2, n\_testing\_data, 1)

second\_col\_testing\_data\_model3 = matrix((testing\_data\_x1)^3, n\_testing\_data, 1)

third\_col\_testing\_data\_model3 = matrix((testing\_data\_x3)^4, n\_testing\_data, 1)

fourth\_col\_testing\_data\_model3 = matrix(1, n\_testing\_data, 1)

model3\_matrix\_testing\_data = cbind(first\_col\_testing\_data\_model3, second\_col\_testing\_data\_model3, third\_col\_testing\_data\_model3, fourth\_col\_testing\_data\_model3)

# output for testing data using estimated parameters for training data

yHat\_testing\_data = model3\_matrix\_testing\_data %\*% thetaHat\_training\_data

#residual for testing data

residual\_testing\_data = testing\_data\_y - yHat\_testing\_data

# plotting the residual

qqnorm(residual\_testing\_data)

qqline(residual\_testing\_data)

boxplot(residual\_testing\_data)

plot(testing\_data\_y, residual\_testing\_data)

abline(lm(residual\_testing\_data~testing\_data\_y), col = "red")

# Model 3 - y = w1\*x2 + w2\*x1^3 + w3\*x3^4 + w\_bias + E

model = lm((training\_data\_y)~(training\_data\_x2)+I(training\_data\_x1^3)+I(training\_data\_x3^4),data = data.frame(training\_data\_matrix))

error = testing\_data\_y - testing\_predict\_yhat

error

# compute 95% confidence interval and plot them (with error bars) with the mean value of the model prediction and testing data

CI = 1.96 \* (sd(testing\_data\_y)/sqrt(length(testing\_data\_y)))

CI

mean\_testing\_data\_y = mean(testing\_data\_y)

ggplot(data.frame(testing\_data\_y), aes(x = testing\_data\_y, y = residual\_testing\_data)) +

geom\_point() +

geom\_smooth(method = lm, col = "red" , se = TRUE)

par(mfrow = c(2,2))

plot((testing\_data\_y)~(testing\_data\_x2), data = data.frame(testing\_data\_y),

xlab = "Predictor", ylab = "Response",

main = "Linear Regression",

pch = 20, cex = 3, col = "darkgrey")

abline(model, lwd = 3, col = "dodgerblue")

plot((testing\_data\_y)~I((testing\_data\_x1)^3), data = data.frame(testing\_data\_y),

xlab = "Predictor", ylab = "Response",

main = "Linear Regression",

pch = 20, cex = 3, col = "darkgrey")

abline(model, lwd = 3, col = "dodgerblue")

plot((testing\_data\_y)~I((testing\_data\_x3)^4), data = data.frame(testing\_data\_y),

xlab = "Predictor", ylab = "Response",

main = "Linear Regression",

pch = 20, cex = 3, col = "darkgrey")

abline(model, lwd = 3, col = "dodgerblue")

plot((testing\_data\_y)~(testing\_data\_x2)+I(testing\_data\_x1^3)+I(testing\_data\_x3^4), data = data.frame(testing\_data\_y),

xlab = "Predictor", ylab = "Response",

main = "Smoothing",

pch = 20, cex = 3, col = "darkgrey")

lines(xplot, predict(sim\_fit\_2, newdata = data.frame(predictor = xplot)),

lwd = 3, lty = 2, col = "darkorange")

# Task 3 - ABC rejection

# selecting two parameters the best model

row.names(thetaHat3) = c("w1","w2","w3","w\_bias")

colnames(thetaHat3) = c("parameters")

thetaHat3\_ABC = thetaHat3

# fix remaining parameters as 0 and keeping the two largest parameters w1 and w\_bias

thetaHat3\_ABC[c("w2","w3"),"parameters"] = 0

thetaHat3\_ABC

# using uniform distribution as prior y = w1\*x2 + w2\*x1^3 + w3\*x3^4 + w\_bias + E

theta\_bias\_values = runif(201, min = 0, max = 0.5)

theta\_1\_values = runif(201, min = 0, max = 0.1)

# Draw samples from the above Uniform prior

yHat\_ABC = model3\_X %\*% thetaHat3\_ABC

yHat\_ABC

error\_ABC = y\_matrix - yHat\_ABC

error\_ABC

epsilon = 0.02

#Reject > e ,Accept <= e

for (i in error\_ABC) {

if(i <= epsilon ) {

print(i)

}

}

contourPlot(theta\_bias\_values, theta\_1\_values, error\_ABC, col = NULL)

ggplot() +

geom\_point(data = data.frame(model3\_X), aes(x = theta\_bias\_values, y = theta\_1\_values), shape = 1, size = 2.5, color = "red") + stat\_contour()

**REFFRENCES**

1. James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. [***An introduction to statistical learning***](https://coventry.rl.talis.com/users/2DB18655-B5C6-0D2A-003D-87C351B6143D/bookmarks/5586595E-AF18-E393-E9D9-D8BE19DB8F5D.html). Vol. 112. New York: springer, 2013.
2. Bishop, Christopher M. ***Pattern recognition and machine learning***. springer, 2006.
3. <https://medium.com/swlh/what-do-we-see-with-aic-bic-57b3e12685d>
4. <https://www.tutorialspoint.com/managerial_economics/regression_technique.htm>
5. <https://en.wikipedia.org/wiki/Approximate_Bayesian_computation>