



**Module Title - Introduction to Statistical Methods for Data Science**

**Coursework Title - Modelling brain signals using nonlinear regression**

**Submitted By**

**Arjun Jayan**

**ID – 12577022**

**Lecture**

**Dr.Fei He**

## TABLE OF CONTENTS

<b>1. Introduction.....</b>	<b>4</b>
<b>2. Task 1: Preliminary Data Analysis.....</b>	<b>4</b>
Installing Common packages.....	4
Installing Common Packages.....	4
Installing Date and Time-Series Packages.....	4
Setting Working Environment Directory.....	4
The setwd().....	4
Importing Data set.....	5
Combining Output and Input Signal and converting Data to Time Series.....	5
Converting the Time-series Data into a Data Frame Structure.....	6
Distribution of Output and Input Signal by Use of a Histogram.....	7
Check for Correlation.....	10
The Scatter Plot.....	11
Boxplot Representation.....	12
<b>3. Task 2:Regression – Modelling The Relationship Between MEG</b>	
<b>Signals.....</b>	<b>13</b>
Estimating Model Parameters Using Least Squares.....	13
Residual Sum of Squared Error.....	13
The log-likelihood Function.....	15
The AIC & BIC.....	16
Distribution of Model Prediction Errors.....	18
The Best Model.....	23
Splitting the Data.....	23
Estimating Model Parameter.....	24

<b>4.Task3:Approximation Bayesian Computation.....</b>	<b>27</b>
Absolute Values.....	27
Range of Prior Distribution.....	28
Rejection of ABC.....	28
Plot Joint Distribution of parameters input 3 & input 5.....	29
Plot for Marginal Distribution for input 3 parameters.....	29
Plot for Marginal Distribution for input 5 Parameters.....	30
Summary of the Tasks.....	31
 <b>5.   References.....</b>	 <b>32</b>
<b>6.   Appendix.....</b>	<b>33</b>

## 1. Introduction

This report aims to get a clear vision of what is entailed in all of my codes. It is a description of regression models that will be used to identify and measure the auditory-brain interaction and effect of the emotional narration. A hypothesis will help to explain how emotional narration will evoke the brain response. There is a visual representation of the data followed by clearly explained codes.

## 2. Task 1: Preliminary Data Analysis

### Installing Common Packages

Installing the common packages that will help in the data interpretation and visualization. Package **ggplot2** for assistance to create complex plots from data in a data frame. The package **tidyverse** is suited to make the data operations more friendly. The **dplyr** function comes with a consistent set of verbs that helps in solving data manipulations.

### Installing Data and Time-series Packages

For a time-series data set, it was necessary to import the **time series** packages. The **lubridate** package makes it easier to work with dates and times in the time series set of data. The **tseries** package is a crucial package used in the creation of time series objects. Another useful times series package is the **car** package (Companion to Applied Regression). It complements the techniques by providing functions that perform the test, creates visualizations, and transform data.

### Setting Working Environment Directory

#### The **setwd()**

Setting work directory environment in R that will be used to read and save files. To set a working directory a built-in function is used in r. The **setwd ()**

## Importing Data set

Importing the output signal, input signal and time data in the R environment using the `read.csv()` inbuilt function. The input signal is read and stored in a variable `input` while the output signal is stored in the `output` variable proceeding to the time data stored in the `times` variable.

## Combining Output and Input signal and converting Data to Time Series

For a good analysis, it was useful to combine the two data set output signals and input signals using a combining function `cbind()` that will combine them into a data set in a column form and then store it in a variable `meg`. After combining the data, there is a need to convert the data to time series. For this purpose, the built-in function `ts()` is used for the data that is chosen for the time-series data to be of the interval of 10 seconds.

```
meg<-cbind(output, input)
```

```
meg1<-ts(meg,end = 20,frequency = 10,start = 0.1)
```

```
head(meg1)
```

---

combining output signal and input signal

```
```{r}
meg<-cbind(output,input)
```
```

Converting data to time series with interval of 10 sec

```
```{r}
meg1<-ts(meg,end = 20,frequency = 10,start = 0.1)
head(meg1)
```
```

```
      y      x1 x2
[1,] 10.677616 -1.3834228 0
[2,] 18.885513  0.4911784 0
[3,] 11.246081 -0.5288271 0
[4,]  9.364583 -0.3594548 0
[5,] 13.434298  0.1733400 0
[6,] 26.166061  1.2207370 0
```

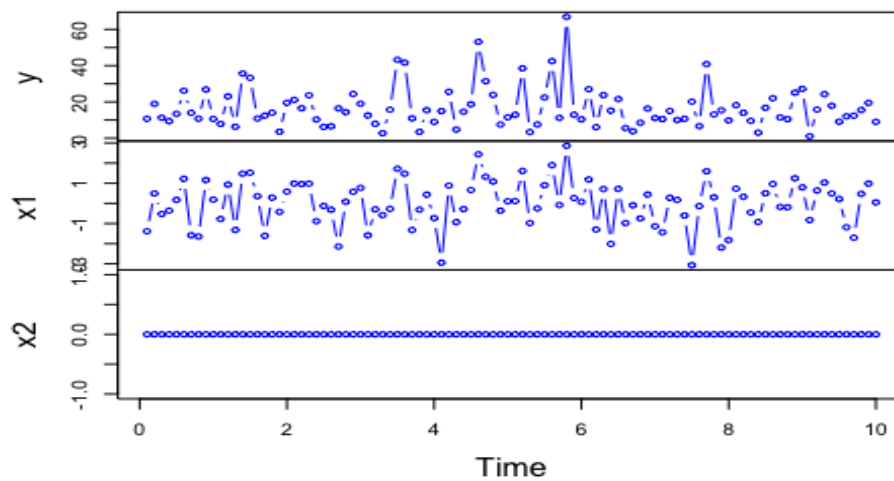
[Print X text](#)

## Converting the Time-Series Data into a Data Frame Structure

The visualization of the data needs to be improved. Here `as.data.frame()` is applied to get the data frame. the data frame quality is improved by making it visualizable into a plotted structure. There is a need to introduce r verbs like the filter verb to help the data manipulations.

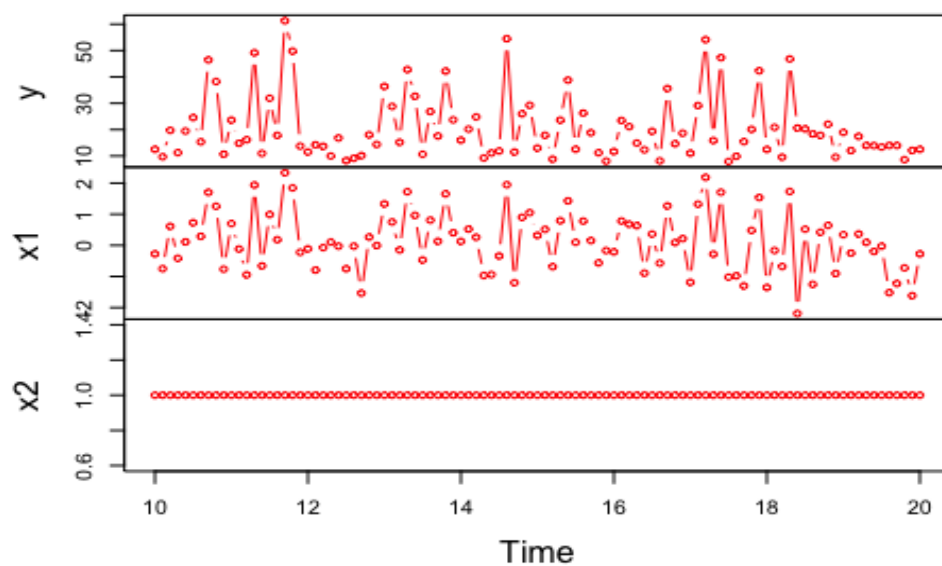
### Time Series Time series plotting for x2=0 (neutral audio)

**time series plot of input and output signal x2=0**



### Time series plotting for x2=1 (Emotional audio)

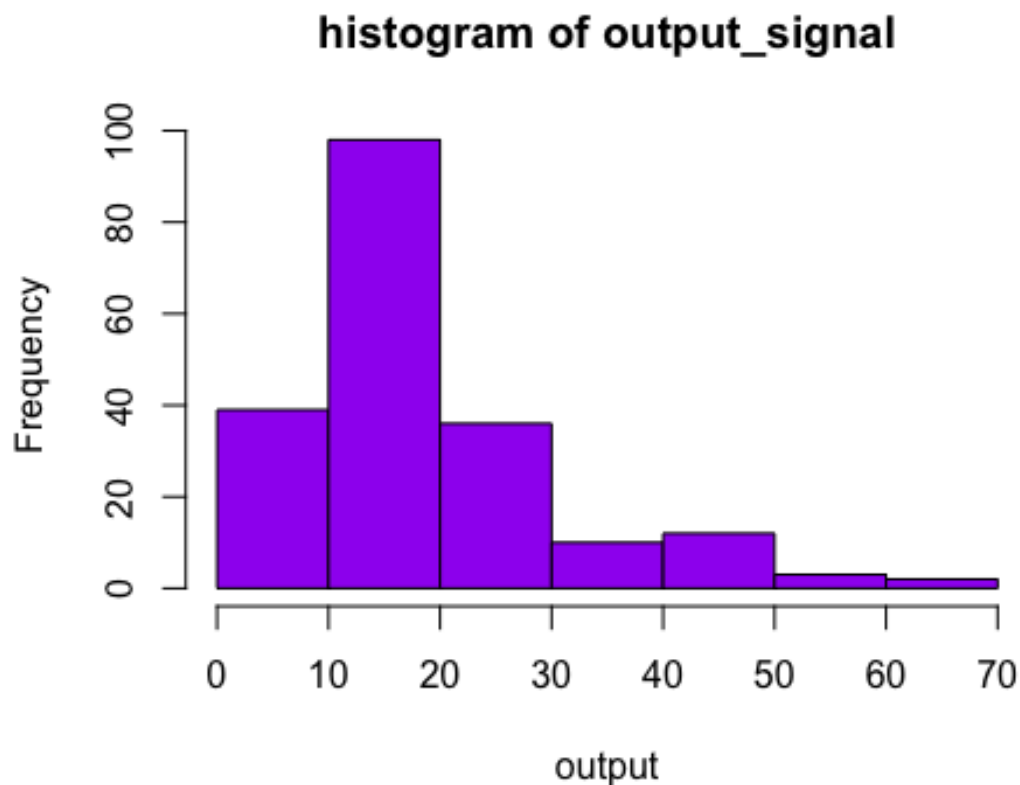
**time series plot of input and output signal x2=1**



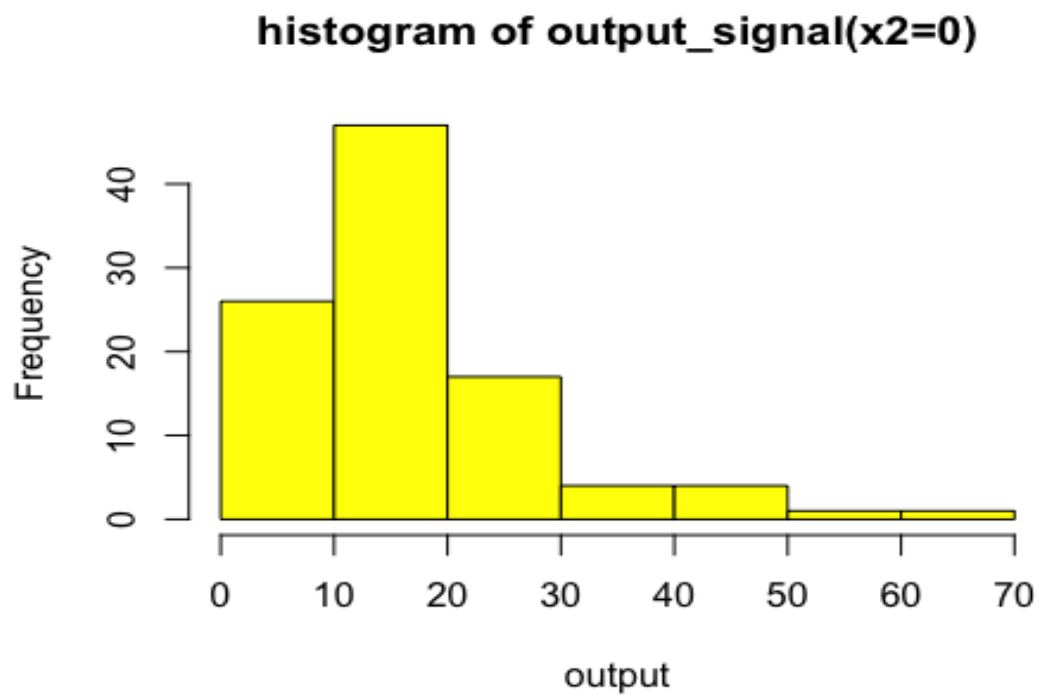
## Distribution of Output and Input Signal by Use of a Histogram

From the data frame created, it is needed to find out how the signals are distributed. The **select()** verb is used to only select the sections needed. the code needs to be stored in a variable for the case the **megdist** was used as the variable. After selecting the y,x1,x2, and filtering where x2 is 0. Here filtering is used to filter data where specific x2 is 0 and selecting only the y and x1 storing them in the variable megdist. After successfully creating the variable, the distributions of the data are seen now and for that, a data visual through histogram is created, the **hist()** is greatly useful. A similar method is used to filter where x2 is 1 and stored them in the **megdist1**.

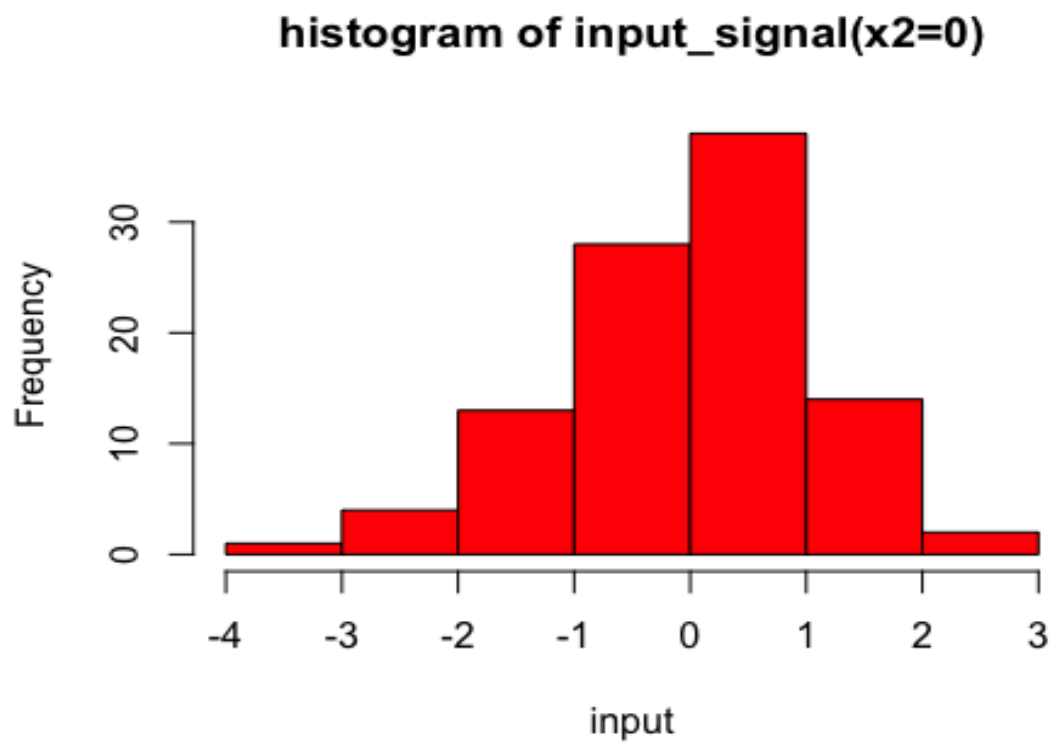
### Histogram of an output signal



**Histogram of the Output signal when  $x_2=0$**

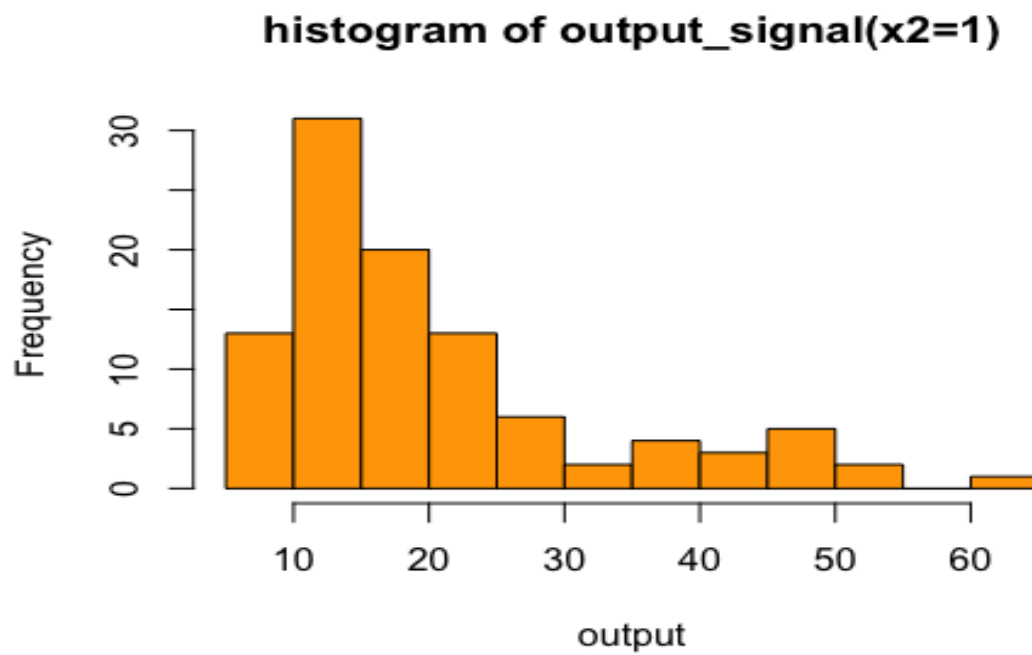


**Histogram of the Input signal when  $x_2=0$**

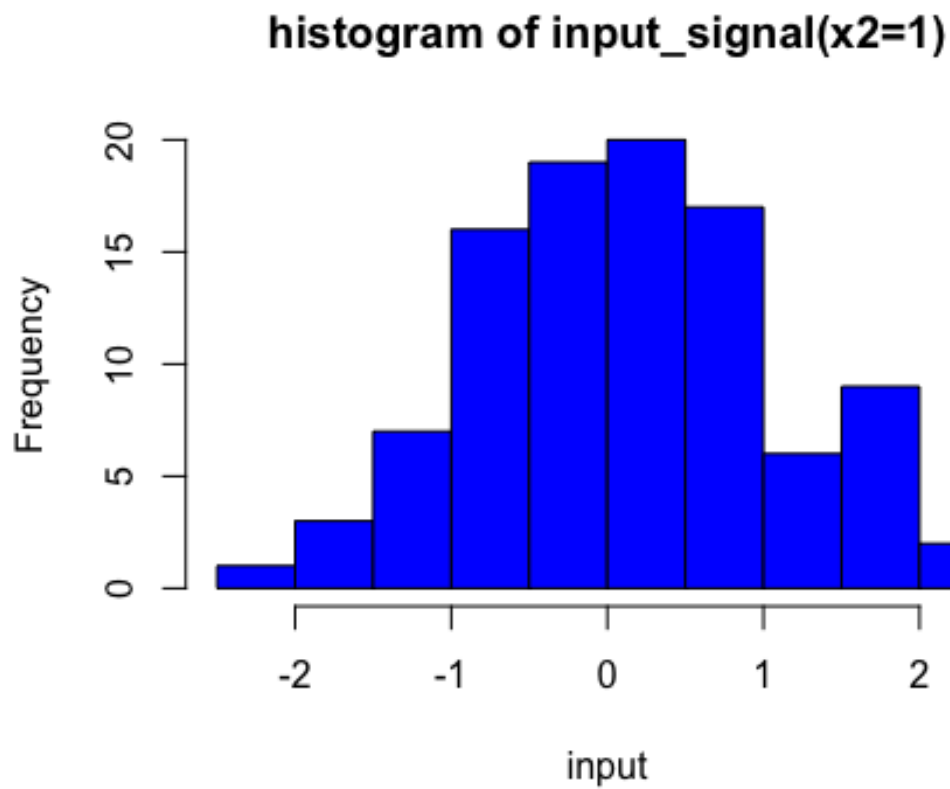




**Histogram of the Output signal when  $x_2=1$**



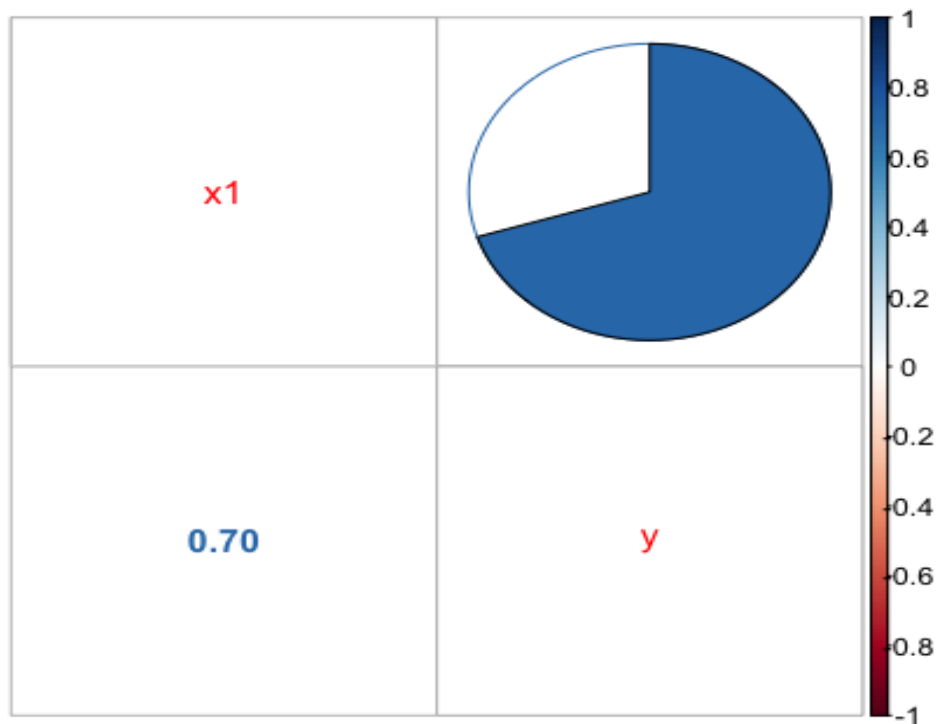
**Histogram of the Input signal when  $x_2=1$**



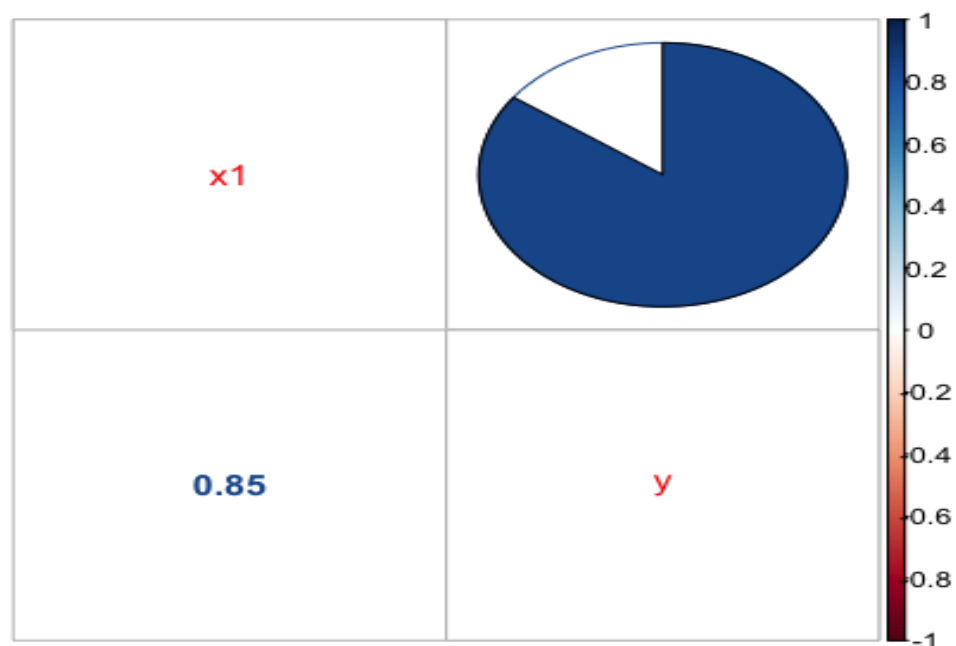
## Check for Correlation

Given two or more data sets it is in the best interest to check for the correlation between them. For checking there is a need to run a correlation test the **cor()** function is used to test. For running the test, the **cor ()** on the **megdist** variable is used. In this case, the correlation will be represented by a pie plot style.

### Correlation when $x_2=0$



### Correlation when $x_2=1$

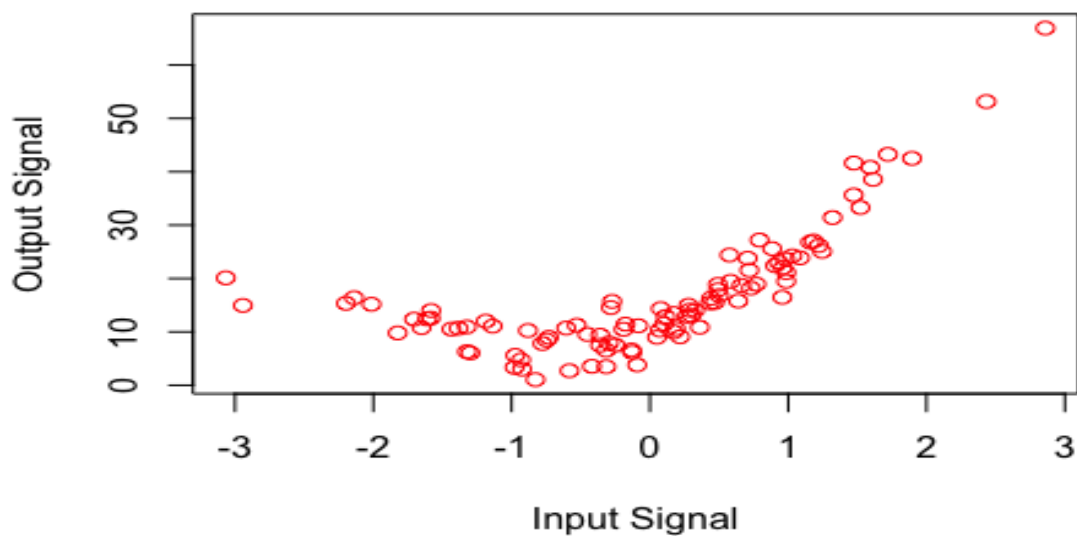


## The Scatter Plot

A scatter plot is used to show the relationship between two variables. In the data, the relationship between the input and output signal is taken and used. It will look into their dependencies. The **plot()** function is applied.

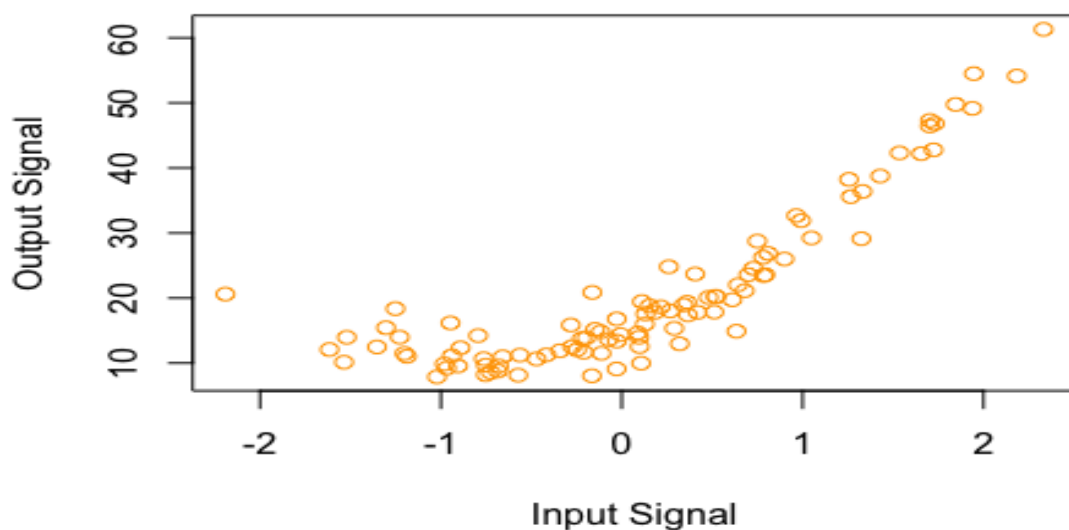
### Scatter Plot when $x_2=0$

**Scatter plot of input and output signal when  $x_2=0$**



### Scatter Plot when $x_2=1$

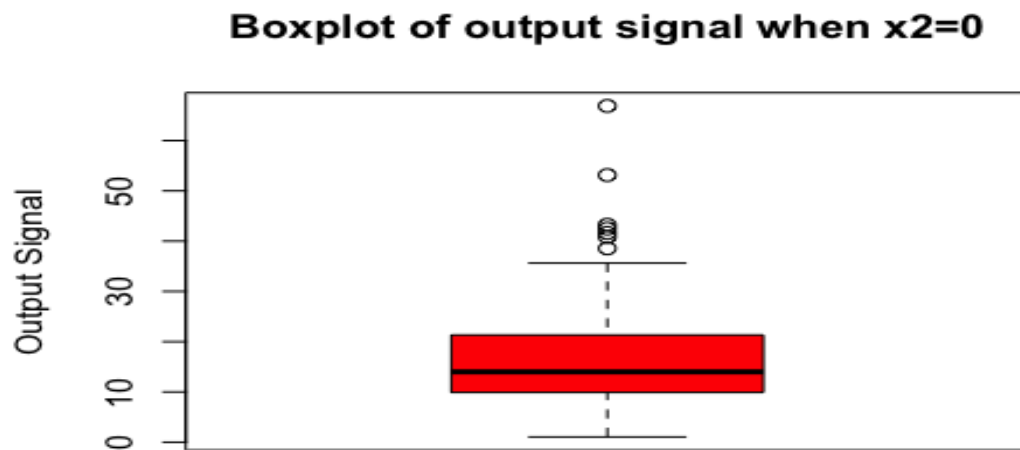
**Scatter plot of input and output signal when  $x_2=1$**



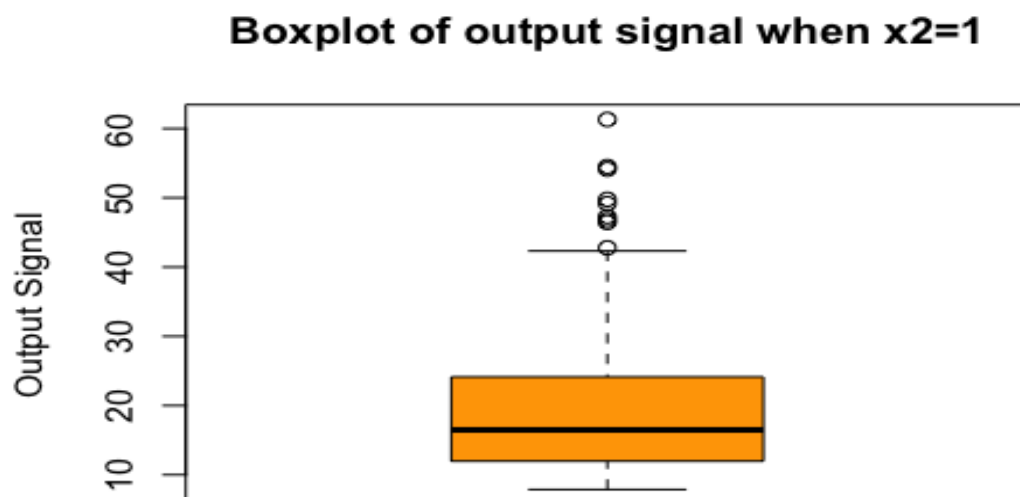
## Boxplot Representation

Boxplot is a data visualization tool that is used to check for the distribution of the data. In this case, the boxplot is used to examine the effect of sound categories using the output brain signals.

### Box Plot Box plot when $x_2=0$



### Box plot when $x_2=1$



### 3. Task 2:Regression – Modelling The Relationship Between MEG Signals

It is needed to come up with a mathematical model that can explain the relationship between the input signals and the output signals and how this relationship changes based on the content of the input audio signals. assuming such relation can be described using a regression model.

#### Task 2.1 Estimating Model Parameters Using Least Squares

Each model is analysed separately to aid in the relationship testing. Since there are different models, each value of each model is found separately. It is necessary to convert the data meg into a data frame, here the function **as.data.frame()** is used to make it a data frame and then stored to a variable **regrndata**. For the starting point, it is started with the evaluation of the models one by one. Then each model is made into a data frame. Since least squares are calculations concerning matrix, each data frame should be converted to a matrix form using the **as.matrix()**. After successfully transforming the data frame into a matrix form, R programming is used to do the calculations. **Solve()** is used to do inverse calculations concerning matrices.

**t()** function used for the purpose of transposing a matrix.

**%\*%** does the matrix multiplications.































**head()** is a function that returns the first part of the matrix (6 rows) that contains many rows.

#### Task 2.2 Residual Sum of Squared Error

Considering the given model the RSS(Residual Sum of Squared Error) of every model can be figured out using the given formula. For RSS the sum of the error is obtained by taking all the deviations of output stored in a variable y. The variable y is converted into the matrix format. The RSS for Model 1 is calculated using the R code.

```
meg_rss1<-norm((y-(meg_mod1%*%meg_md1))^2)
meg_rss1
```

A similar method is used to find the RSS for the remaining models.

|  |   |
|--|---|
| Model 1 meg_RSS  |   |
| <pre> ````{r} meg_rss1&lt;-norm((y-(meg_mod1%*%meg_md1))^2) meg_rss1 ```` </pre> |          |
| [1] 11825.42   |          |
| Model 2 meg_RSS  |   |
| <pre> ````{r} meg_rss2&lt;-norm((y-(meg_mod2%*%meg_md2))^2) meg_rss2 ```` </pre> |          |
| [1] 11238.95   |          |
| Model 3 meg_RSS  |   |
| <pre> ````{r} meg_rss3&lt;-norm((y-(meg_mod3%*%meg_md3))^2) meg_rss3 ```` </pre> |          |
| [1] 1636.168   |          |
| Model 4 meg_RSS  |   |
| <pre> ````{r} meg_rss4&lt;-norm((y-(meg_mod4%*%meg_md4))^2) meg_rss4 ```` </pre> |          |
| [1] 1902.063   |    |
| Model 5 meg_RSS  |   |
| <pre> ````{r} meg_rss5&lt;-norm((y-(meg_mod5%*%meg_md5))^2) meg_rss5 ```` </pre> |    |
| [1] 4928.312   |    |

### Task 2.3 The log-likelihood Function

The log-likelihood function is a function declared in R to take at least two or more arguments.

Here every log function is performed by the provided formula on all the models i.e.

The log-likelihood for Model 1 is calculated using the R code.

```
meg_logfxn1<-      -(nrow(y)/2*log(2*pi))-(nrow(y)/2*(log(meg_rss1/(nrow(y)-1))))-  
meg_rss1*(1/(2*(meg_rss1/(nrow(y)-1))))  
meg_logfxn1
```

A similar method is used to find the log-likelihood for the remaining models.



```
log likelihood 1
```{r}
meg_logfxn1<- -(nrow(y)/2*log(2*pi))-(nrow(y)/2*(log(meg_rss1/(nrow(y)-1))))-meg_rss1*(1/(2*(meg_
s1/(nrow(y)-1))))
meg_logfxn1
```

[1] -691.7579

log likelihood 2
```{r}
meg_logfxn2<- -(nrow(y)/2*log(2*pi))-(nrow(y)/2*(log(meg_rss2/(nrow(y)-1))))-meg_rss2*(1/(2*(meg_
s2/(nrow(y)-1))))
meg_logfxn2
```

[1] -686.6713

log likelihood 3
```{r}
meg_logfxn3<- -(nrow(y)/2*log(2*pi))-(nrow(y)/2*(log(meg_rss3/(nrow(y)-1))))-meg_rss3*(1/(2*(meg_
s3/(nrow(y)-1))))
meg_logfxn3
```

[1] -493.9684

log likelihood 4
```{r}
meg_logfxn4<- -(nrow(y)/2*log(2*pi))-(nrow(y)/2*(log(meg_rss4/(nrow(y)-1))))-meg_rss4*(1/(2*(meg_
s4/(nrow(y)-1))))
meg_logfxn4
```

[1] -509.0267

log likelihood 5
```{r}
meg_logfxn5<- -(nrow(y)/2*log(2*pi))-(nrow(y)/2*(log(meg_rss5/(nrow(y)-1))))-meg_rss5*(1/(2*(meg_
s5/(nrow(y)-1))))
meg_logfxn5
```
```

## Task 2.4 The AIC & BIC

In each model the number  $k$  (the number of parameters estimated) is assigned differently i.e., **model 1( $k=4$ )**, **model 2( $k=3$ )**, **model 3( $k=5$ )**, **model 4( $k=6$ )** and **model 5( $k=5$ )**. The Value of  $K$  can be calculated by using the code.

```
length(meg_md1)
```

This will provide the no: of parameters passed to the First model.

The Akaike information criterion(AIC) and the Bayesian information criterion(BIC) provide a measure of model performance that accounts for model complexity. They also combine a reflection of how well the model fits the data. The lower the AIC & BIC Value the better the model.

The AIC & BIC for Model 1 is calculated using the R code.

```
k1=4
```

```
meg_AIC1<-2*k1-(2*meg_logfxn1)
```

```
meg_AIC1
```

```
meg_BIC1<-k1*log(nrow(y))-(2*(meg_logfxn1))
```

```
meg_BIC1
```

A similar method is used to find the AIC & BIC for the remaining models.

```
K=no:of parameters estimated

Model 1 AIC & BIC
```{r}
length(meg_md1)
```

[1] 4

```{r}
k1=4
meg_AIC1<-2*k1-(2*meg_logfxn1)
meg_AIC1
meg_BIC1<-k1*log(nrow(y))-(2*(meg_logfxn1))
meg_BIC1
```

[1] 1391.516
[1] 1404.709

Model 2 AIC & BIC
```{r}
length(meg_md2)
```

[1] 3

```{r}
k2=3
meg_AIC2<-2*k2-(2*meg_logfxn2)
meg_AIC2
meg_BIC2<-k2*log(nrow(y))-(2*(meg_logfxn2))
meg_BIC2
```

[1] 1379.343
[1] 1389.238
```

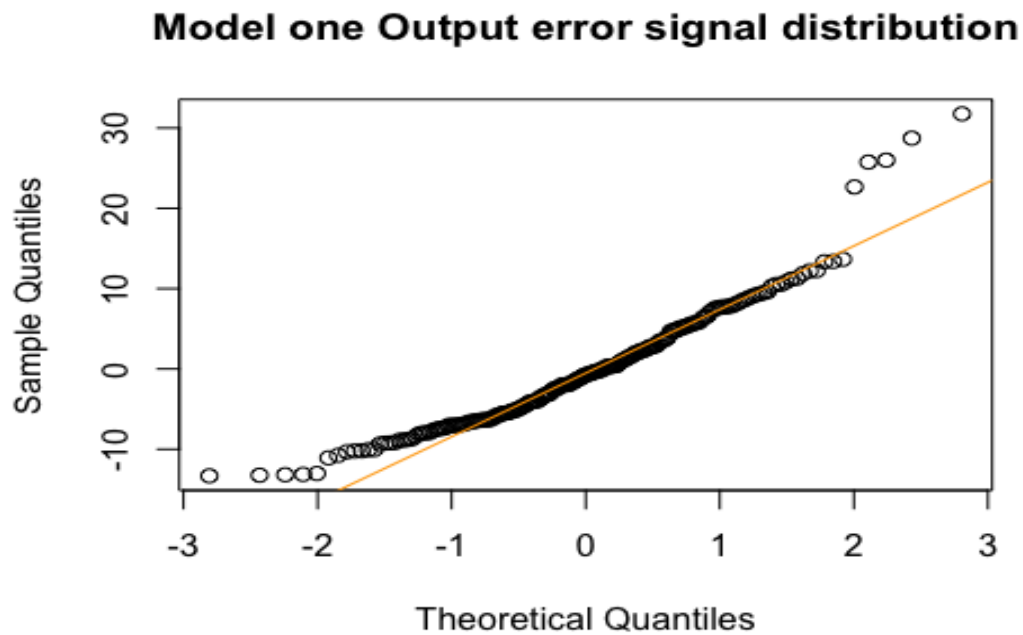


|   |  |
|---|--|
| Model 3 AIC & BIC   |  |
| <pre> ```{r} length(meg_md3) ``` </pre>   |  |
| [1] 5   |  |
| <pre> ```{r} k2=5 meg_AIC3&lt;-(2*k2-(2*meg_logfxn3)) meg_AIC3 meg_BIC3&lt;-k2*log(nrow(y))-(2*(meg_logfxn3)) meg_BIC3 ``` </pre> |  |
| <pre> [1] 997.9368 [1] 1014.428 </pre>  |  |
| Model 4 AIC & BIC   |  |
| <pre> ```{r} length(meg_md4) ``` </pre>   |  |
| [1] 6   |  |
| <pre> ```{r} k2=6 meg_AIC4&lt;-(2*k2-(2*meg_logfxn4)) meg_AIC4 meg_BIC4&lt;-k2*log(nrow(y))-(2*(meg_logfxn4)) meg_BIC4 ``` </pre> |  |
| <pre> [1] 1030.053 [1] 1049.843 </pre>  |  |
| Model 5 AIC & BIC   |  |
| <pre> ```{r} length(meg_md5) ``` </pre>   |  |
| [1] 5   |  |
| <pre> ```{r} k2=5 meg_AIC5&lt;-(2*k2-(2*meg_logfxn5)) meg_AIC5 meg_BIC5&lt;-k2*log(nrow(y))-(2*(meg_logfxn5)) meg_BIC5 ``` </pre> |  |
| <pre> [1] 1218.465 [1] 1234.956 </pre>  |  |

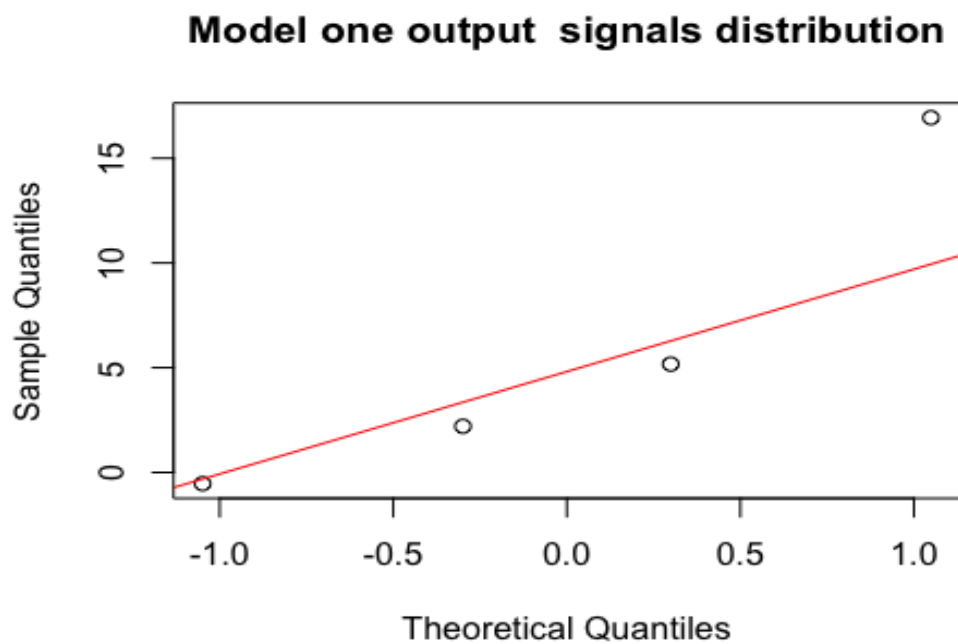
### Task 2.5 Distribution of Model Prediction Errors

To check for the model prediction errors it is needed to plot the error distribution by use of the `qqnorm()` & `qqline()` of each model.

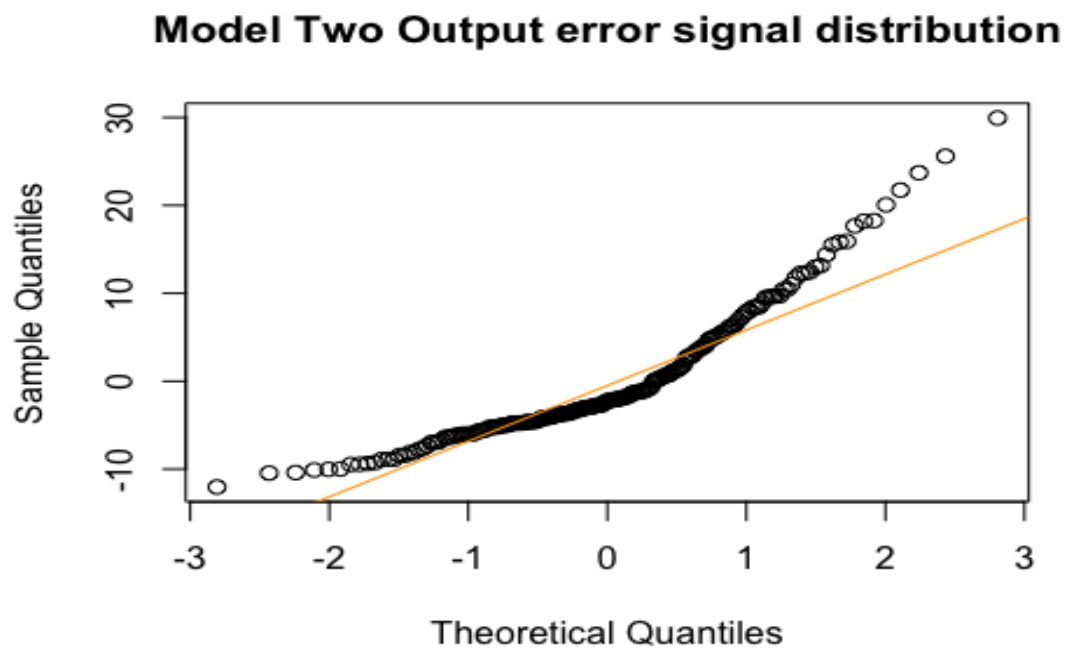
#### Model one Output error signal distribution



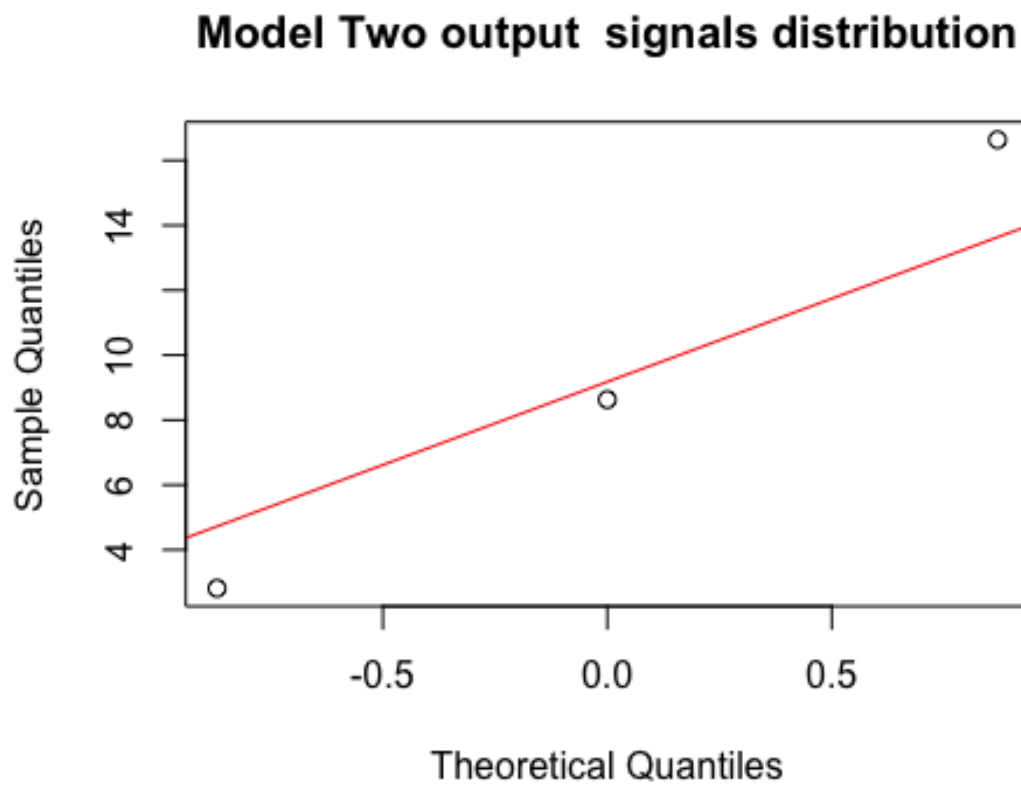
#### Model one output signals distribution



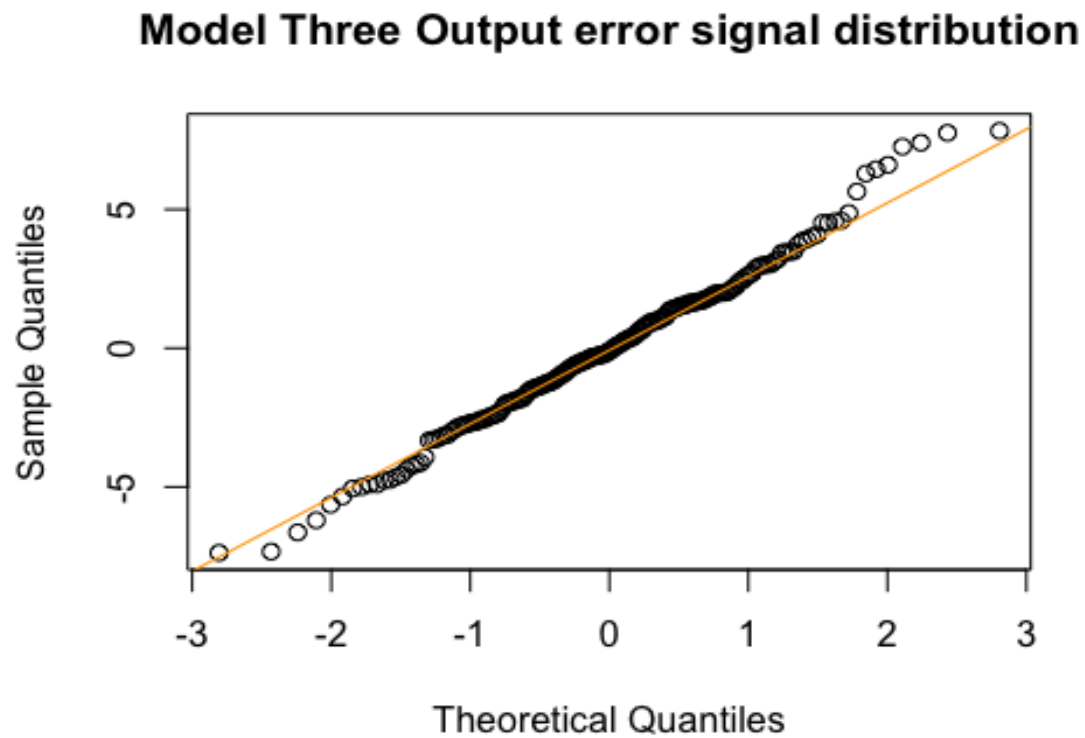
### Model Two Output error signal distribution



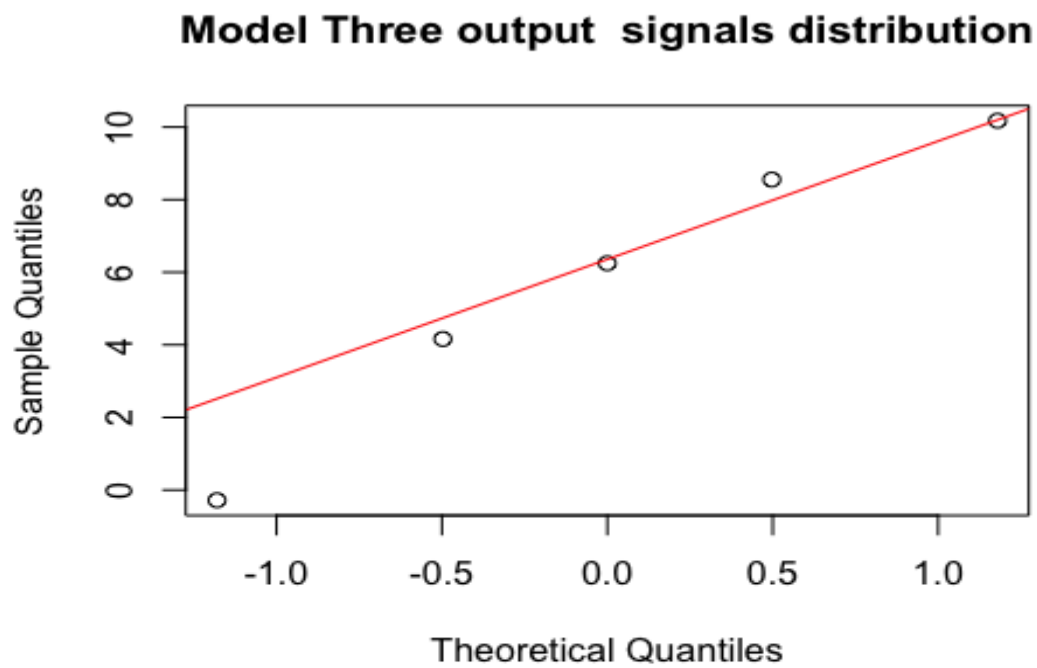
### Model Two output signals distribution



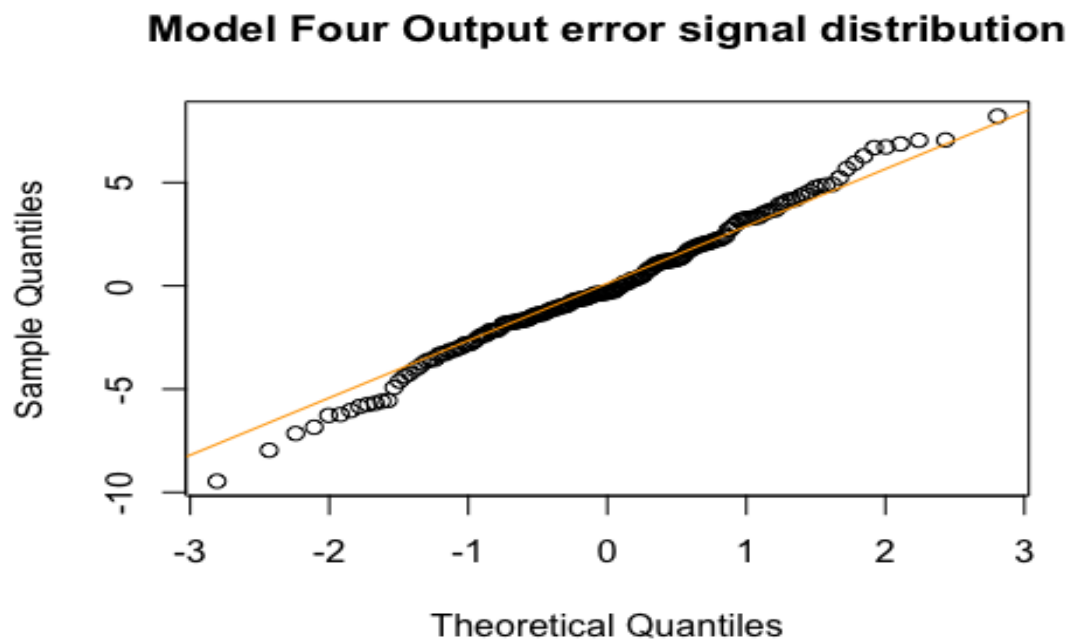
### Model Three Output error signal distribution



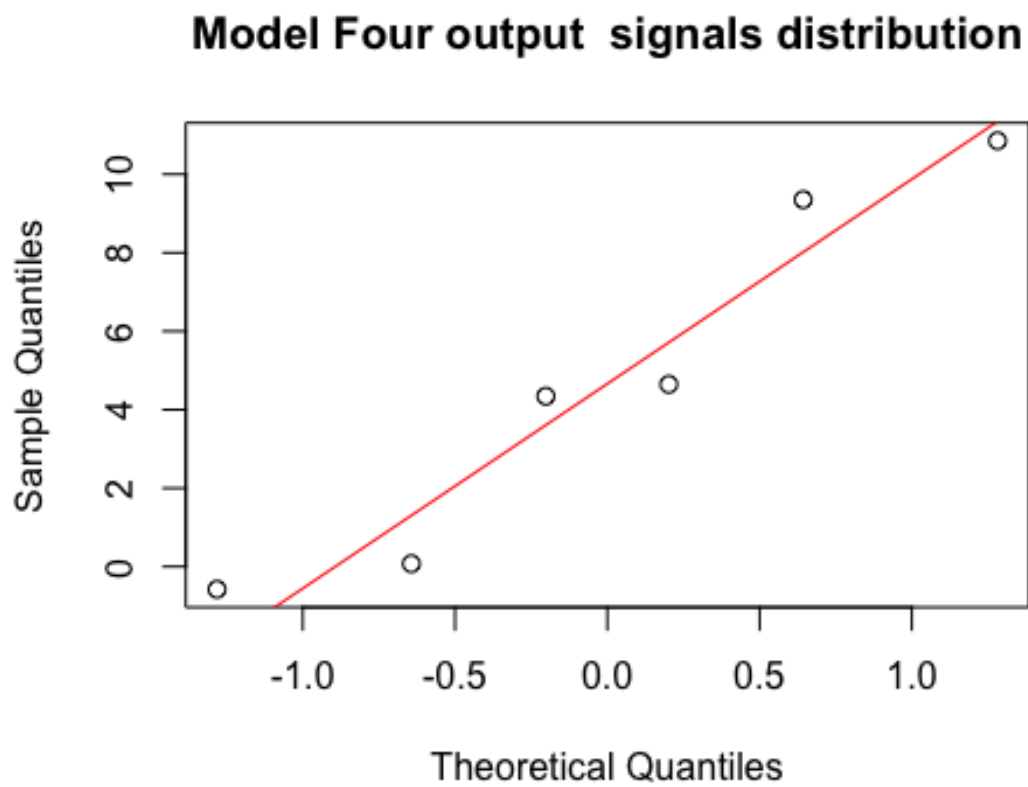
### Model Three output signals distribution



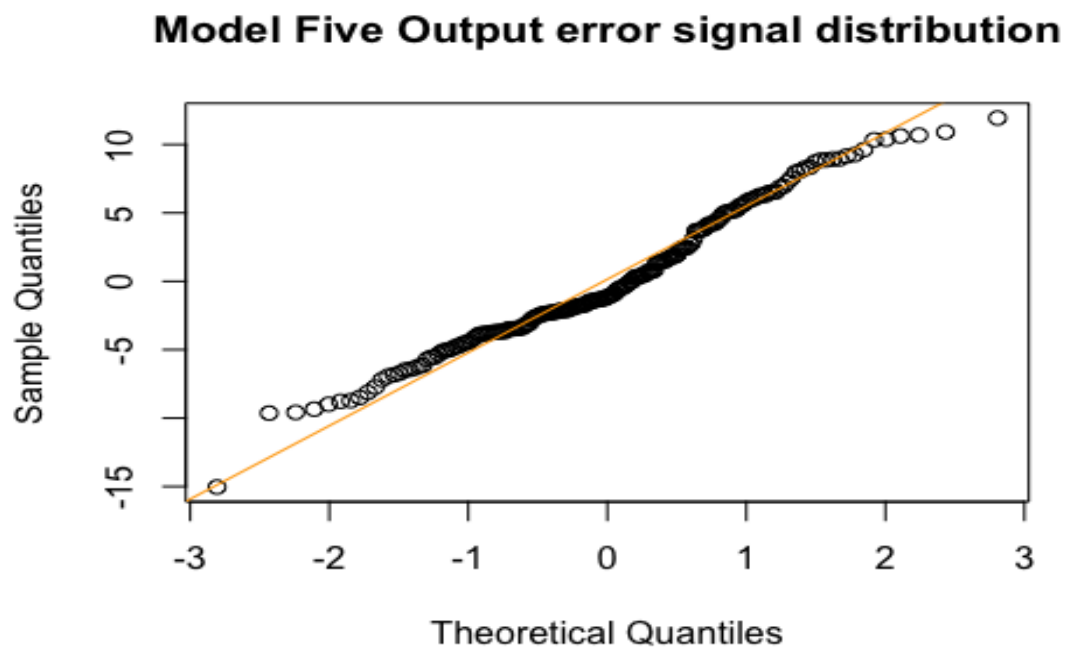
#### Model Four Output error signal distribution



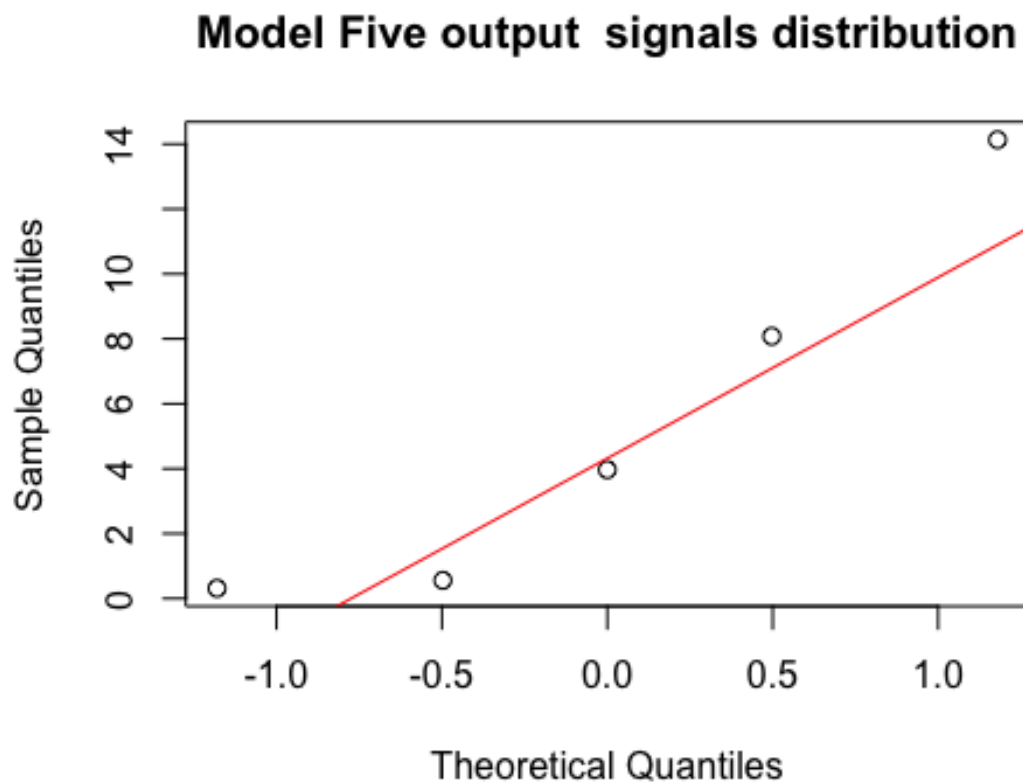
#### Model Four output signals distribution



### Model Five Output error signal distribution



### Model Five output signals distribution



## Task 2.6 The Best Model

For a model to be ranked the best it should have the least AIC and BIC values. Based on the calculations figured Model 3 is the best Regression model as it has the smallest values of AIC (997.368) & BIC (1014.428).

## Task 2.7 Splitting the Data

It is necessary to collect the random numbers, then it is needed to get random variables that will be the same for all the chosen random variables. For that to be achieved it is needed to use a function of generating random variables `set.seed()`. A sample of 140 rows is selected. The input and output are split into two parts one part is used for training and the other part is used for testing.

### Training Data

```
Training Data
```{r}
training_data<-regrndata[tradata,]
head(training_data)
```
```

Description: df [6 x 3]

|     | y<br><dbl> | x1<br><dbl> | x2<br><int> |
|-----|------------|-------------|-------------|
| 159 | 11.22192   | -0.5629525  | 1           |
| 179 | 20.09444   | 0.4802248   | 1           |
| 14  | 35.63663   | 1.4735968   | 0           |
| 195 | 13.90411   | -0.1955684  | 1           |
| 170 | 18.64306   | 0.2156213   | 1           |
| 50  | 11.49615   | 0.1037275   | 0           |

6 rows

### Testing Data

```
Testing Data
```{r}
testing_data<-regrndata[-tradata,]
head(testing_data)
```
```

Description: df [6 x 3]

|    | y<br><dbl> | x1<br><dbl> | x2<br><int> |
|----|------------|-------------|-------------|
| 2  | 18.885513  | 0.4911784   | 0           |
| 3  | 11.246081  | -0.5288271  | 0           |
| 10 | 10.595761  | 0.1871324   | 0           |
| 11 | 7.817391   | -0.7771865  | 0           |
| 15 | 33.297743  | 1.5221666   | 0           |
| 18 | 14.057977  | 0.2871348   | 0           |

6 rows

## Estimating Model Parameters

According to the previous evaluation of which is the best model, the result was that Model 3 is the best fit due to its low AIC & BIC values compared to the rest of the models. The parameters are estimated using the least squares. The training and testing methods are implemented for Model 3. It's required to create a data frame of the training input data. After getting the result the data frame is converted into a matrix format to enable an easy arithmetic calculation.

## Training Input and Output Signal

```
Estimating Model parameter
Model 3
Training Input Signal
```{r}
meg_model03<-data.frame(input0=c(1),training_data$x1,training_data$x1^2,training_data$x1^4,training_data$x2)
meg_md03<- as.matrix(meg_model03)
head(meg_md03)
```


	input0	training_data.x1	training_data.x1.2	training_data.x1.4	training_data.x2
[1,]	1	-0.5629525	0.31691550	0.1004354371	1
[2,]	1	0.4802248	0.23061589	0.0531836880	1
[3,]	1	1.4735968	2.17148746	4.7153578053	0
[4,]	1	-0.1955684	0.03824698	0.0014628315	1
[5,]	1	0.2156213	0.04649255	0.0021615574	1
[6,]	1	0.1037275	0.01075940	0.0001157646	0



```

Training Output Signal
```{r}
y1<-training_data$y
y1<-as.matrix(y1)
meg_md03<-solve(t(meg_md03)%*%meg_md03)%*%t(meg_md03)%*%y1
meg_md03
```


	[,1]
input0	9.8551858
training_data.x1	8.4179832
training_data.x1.2	6.5732037
training_data.x1.4	-0.3535717
training_data.x2	4.6077003


```


```

The testing data prediction is carried out by the formula sum of the training output signal divided by the length of the output, the obtained value is multiplied by the testing variable.

```
Prediction of Testing Data
```{r}
meg_est<-sum(meg_md03)/length(y1)
meg_est
```
[1] 0.1794182
```{r}
pred<-meg_est*testing_data
head(pred)
```
Description: df [6 x 3]


	y <dbl>	x1 <dbl>	x2 <dbl>
2	3.925556	0.10209669	0
3	2.337619	-0.10992239	0
10	2.202443	0.03889747	0
11	1.624929	-0.16154654	0
15	6.921293	0.31639866	0
18	2.922101	0.05968404	0


6 rows
```



## Confidence Interval

Confidence intervals measure the degree of uncertainties or certainty. To get the confidence intervals in r some of the functions are needed and are so useful.

- Sqrt() to get the square root.
- Var() to id in getting the variance.
- Mean() to get the mean.
- Paste() gives the same functions as the concatenate functions.
- \$y to extract specific data in r in the case data y is needed.

Testing will be done to find out about the two data sets the input and the output signal.

### 95% Confidence Interval of Input, the Output signal

```
95% Confidence Interval of Output signal
```{r}
sd<-sqrt(var(pred$y))
sd
means<-mean(pred$y)
means
paste('Confidence Interval',means-1.96*sd/sqrt(nrow(testing_data)),means+1.96*sd/sqrt(nrow(testing_data)))
```

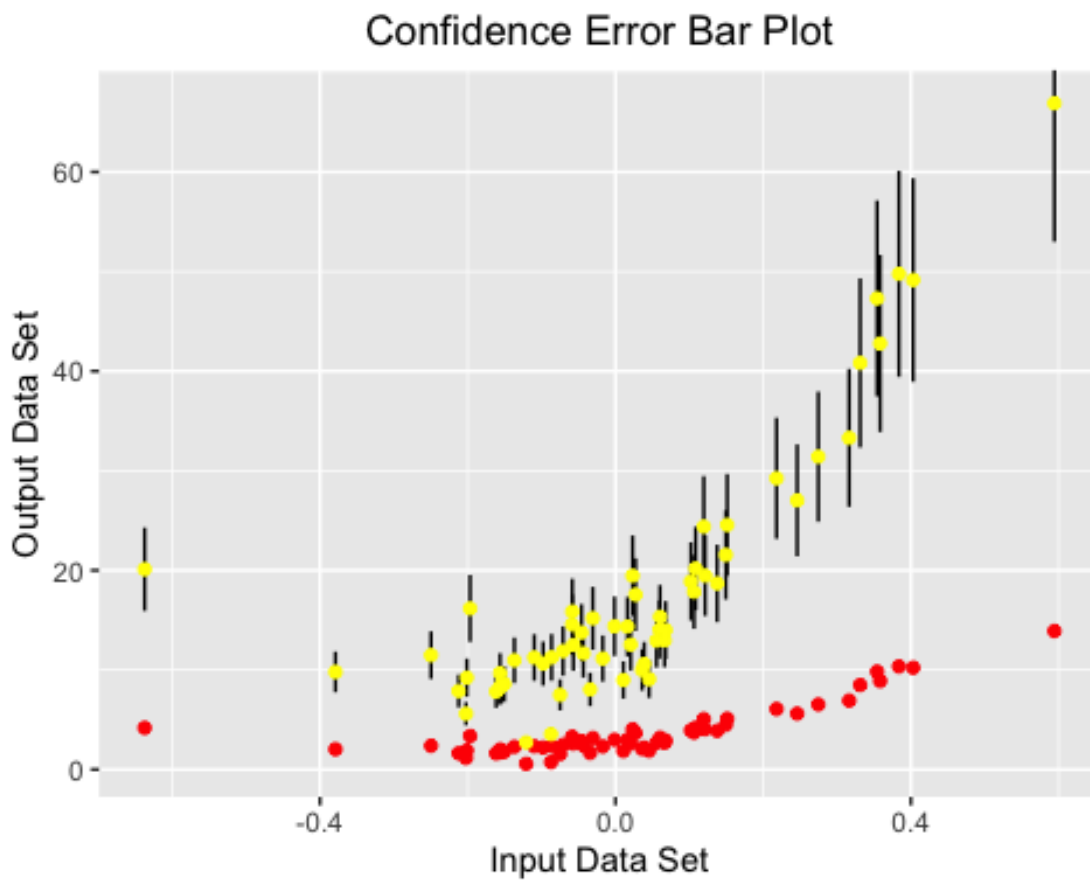
[1] 2.63193
[1] 3.689139
[1] "Confidence Interval 3.02316893445954 4.35510950962138"

95% Confidence Interval of Input signal
```{r}
sd1<-sqrt(var(pred$x1))
sd1
means1<-mean(pred$x1)
means1
paste('Confidence Interval',means1-1.96*sd1/sqrt(nrow(testing_data)),means1+1.96*sd1/sqrt(nrow(testing_data)))
```

[1] 0.2034628
[1] 0.01688682
[1] "Confidence Interval -0.0345963638226104 0.0683699988765016"
```

Error is calculated by subtracting the y values of testing data and prediction. Confidence limits are added to predict the data. With the help of ggplot Confidence Intervals and error bars are plotted.

## Plotting confidence intervals and error bars



## 4. Task 3: Approximation Bayesian Computation

To get the absolute values the sum of columns needs to be obtained and to achieve that a function `colSums()` is introduced and used.

### 2 parameters with the largest absolute least squares estimation

```
7 #Task 3 APPROXIMATE BAYESIAN COMPUTATION (ABC)
8 #1 2 parameters with largest absolute least squares estimation
9 ```{r}
0 meg_mean1<-colSums(abs(meg_mod1))/nrow(meg_mod1)
1 head(meg_mean1)
2 meg_mean2<-colSums(abs(meg_mod2))/nrow(meg_mod2)
3 head(meg_mean2)
4 meg_mean3<-colSums(abs(meg_mod3))/nrow(meg_mod3)
5 head(meg_mean3)
6 meg_mean4<-colSums(abs(meg_mod4))/nrow(meg_mod4)
7 head(meg_mean4)
8 meg_mean5<-colSums(abs(meg_mod5))/nrow(meg_mod5)
9 head(meg_mean5)
0 ^```

input0 input_3 input_5 x2
1.000000 1.799011 7.610171 0.500000
input0 input_1 x2
1.000000 0.8302997 0.5000000
input0 input_1 input_2 input_4 x2
1.000000 0.8302997 1.0856376 3.4913938 0.5000000
input0 input_1 input_2 input_3 input_5 x2
1.000000 0.8302997 1.0856376 1.7990107 7.6101708 0.5000000
input0 input_1 input_3 input_4 x2
1.000000 0.8302997 1.7990107 3.4913938 0.5000000
```

### Absolute Values

The largest absolute least square estimation is carried out for all the five models. Based on the results obtained Model 1 has the largest absolute value. It possesses the largest input values Input 3 and Input 5

```
1 #model 1 has the largest absolute least squares estimate Input 3 & Input 5
2 ```{r}
3 model1<-as.data.frame(meg_mod1)
4 head(model1)
5 ^```
```

Description: df [6 × 4]

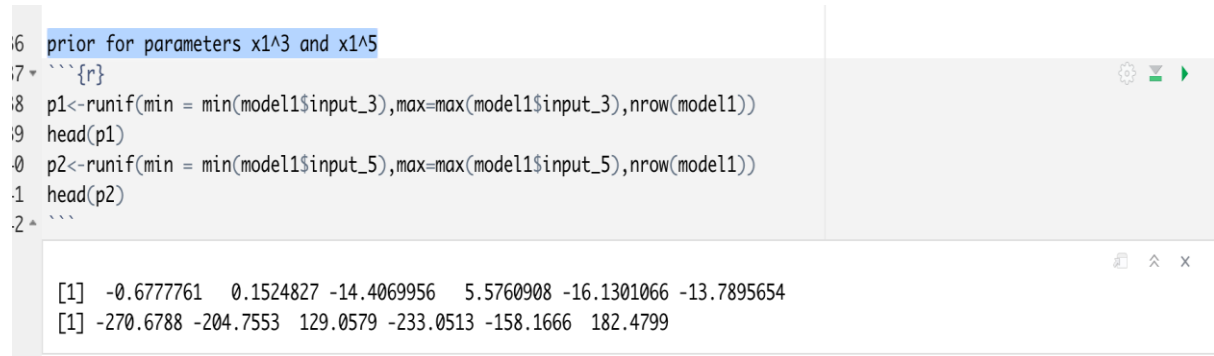
|   | input0<br><dbl> | input_3<br><dbl> | input_5<br><dbl> | x2<br><dbl> |
|---|-----------------|------------------|------------------|-------------|
| 1 | 1               | -2.647675868     | -5.0672775812    | 0           |
| 2 | 1               | 0.118499825      | 0.0285888168     | 0           |
| 3 | 1               | -0.147890824     | -0.0413588750    | 0           |
| 4 | 1               | -0.046444359     | -0.0060009722    | 0           |
| 5 | 1               | 0.005208302      | 0.0001564925     | 0           |
| 6 | 1               | 1.819140747      | 2.7108813071     | 0           |

6 rows

## Range of Prior Distribution

To get the range of the prior distribution, will have to call the model 1 data and get to a data frame. Using the **runif()** to use it as a uniform distribution prior. **runif()** will aid the generation of random deviates. It also helps in generating n numbers of randomness defined in the **runif()** argument.

## Prior for parameters input 3 and input 5



```

i6 prior for parameters x1^3 and x1^5
i7 ```{r}
i8 p1<-runif(min = min(model1$input_3),max=max(model1$input_3),nrow(model1))
i9 head(p1)
i0 p2<-runif(min = min(model1$input_5),max=max(model1$input_5),nrow(model1))
i1 head(p2)
i2 ```

[1] -0.6777761  0.1524827 -14.4069956  5.5760908 -16.1301066 -13.7895654
[1] -270.6788 -204.7553  129.0579 -233.0513 -158.1666  182.4799

```

A sample value of 50 is stored in the variable n

## Rejection of ABC

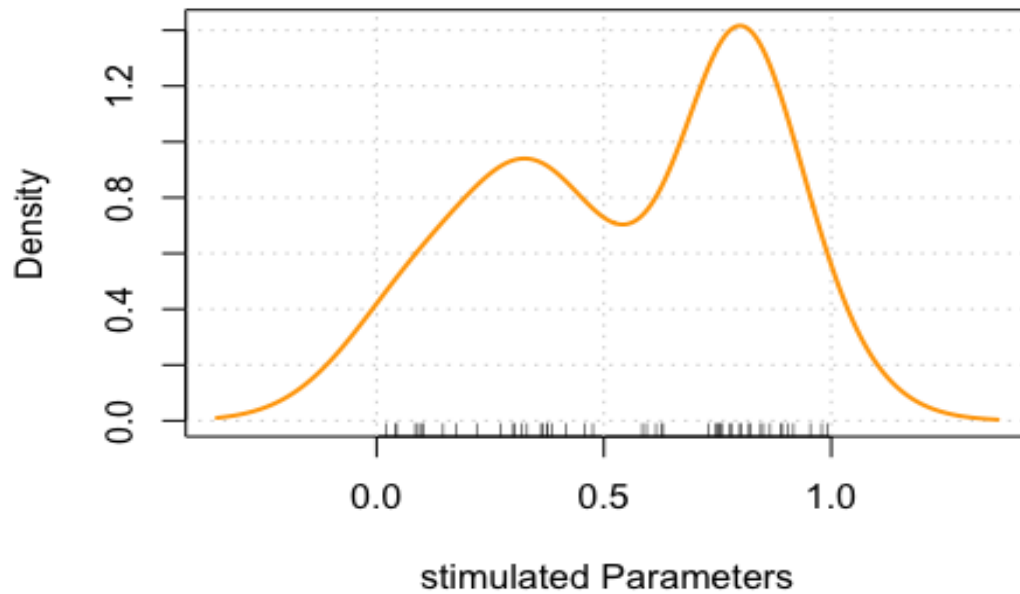
For performing the below tasks importing the EasyABC installed package in r by the library function.

## Plotting of Marginal Distribution

A plot of a marginal distribution will show the distribution of data along only one dimension. List() used to create a list of elements of different types. By using the Density plot we have plotted the joint distribution.

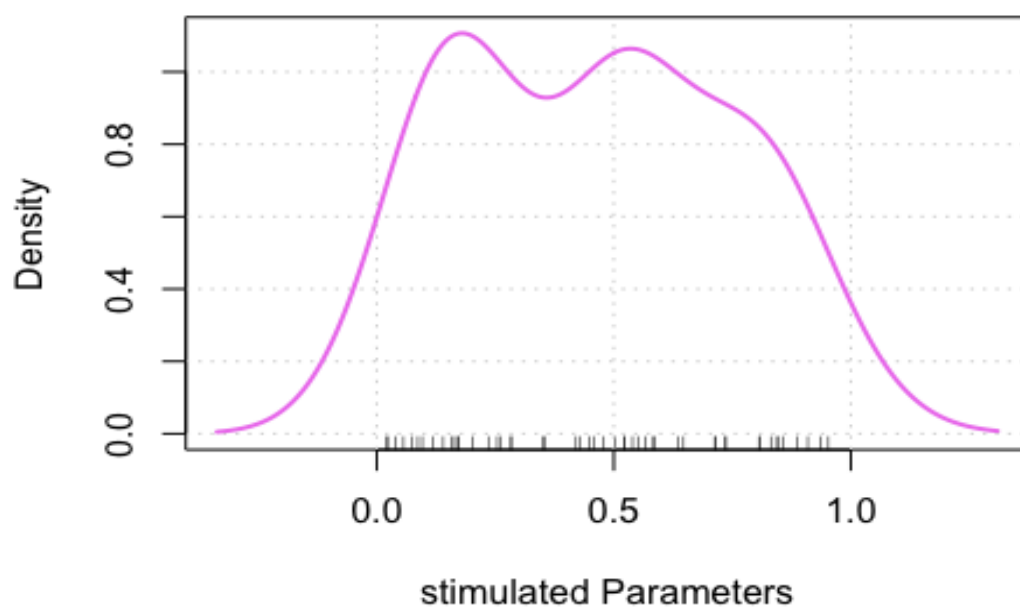
**Plot Joint Distribution of parameters input 3 & Input 5**

**joint distribution of 2 parameters input 3 & input 5**

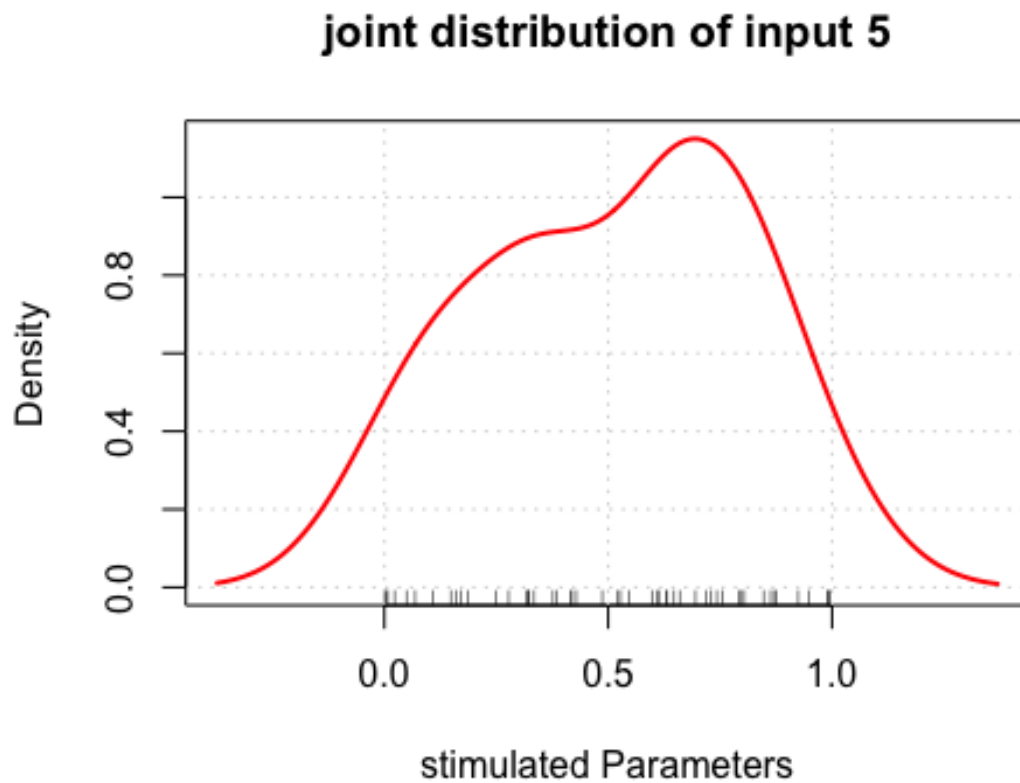


**plot the marginal distribution of Input 3 parameter**

**joint distribution of input 3**



**plot the marginal distribution of Input 5 parameter**



**From the plot, it is clearly understandable that the density curves seem to be normally distributed for both the marginal and joint distribution of the 2 posterior parameters.**

## Summary of the Task

- Time series analysis of the non-linear regression model is plotted using the `ts()` function. The graph is plotted for the input ( $x_2=0$  &  $x_2=1$ ) and also for the output.
- The distribution for each input and output signal is plotted using the Histogram method. The `hist()` function is used for the same.
- The correlation method is used to find the relation between dependent and independent variables. The input is the independent variable and the output is the dependent variable. The pie method is used for representing the correlation.
- The scatter plot is used to plot the dependencies between the input and output variables.
- The Boxplot is used to find the skewness of the output signal corresponding to the time interval. `Boxplot()` function is used to perform the task.
- A suitable model is figured out from the given 5 sets of models.
- Using the least-squares method theta parameter is calculated for all 5 models.
- Based on the model parameters estimated, the model residual sum of squared errors is calculated for every candidate's model using the provided formula.
- The log-likelihood function is calculated for every candidate model using the provided formula to find out how well a model fits the data.
- The AIC and BIC values are calculated for all 5 models to figure out the best regression model
- The errors are calculated for the given models and the estimated errors are plotted using `qqplot` to evaluate if the distribution is close to the normal one.
- According to the findings Model 3 is the best regression method due to its low AIC & BIC values.
- The input and output signals are split into two one for training and another one for testing. From the selected best model parameters, prediction and 95% confidence intervals are figured.
- `ggplot` is used for plotting the confidence intervals and error bars.
- Based on the calculation Model 1 has the largest Absolute value.
- ABC rejection method is used to compute the distribution of the selected regression model.
- The density curves seem to be normally distributed for both the marginal and joint distribution of the 2 posterior parameters.

## 5. References

- Data visualization with R and ggplot2 retrieved from geeks for geeks -  
<https://www.geeksforgeeks.org/data-visualization-with-r-and-ggplot2/>
- Linear regression from scratch in r retrieved from Data science plus  
<https://datascienceplus.com/linear-regression-from-scratch-in-r/>
- A grammar for Data manipulation retrieved from Cran.R project  
<https://cran.r-project.org/web/packages/dplyr/dplyr.pdf>
- For R Basics referred from R tutorial W3 Schools  
<https://www.w3schools.com/r/>



## 6. Appendix

R code Drive Link - <https://drive.google.com/drive/u/1/folders/1qSqGjZvmTseMYbrHhk1-TYwG8O0dNsGS>

```
---

importing common packages

```{r}
library(ggplot2)
library(tidyverse)
library(dplyr)
```

importing data and time series packages

```{r}
library(lubridate)
library(tseries)
library(car)
```

Setting working environment directory

```{r}
setwd("/Users/arjun/Downloads/Arjun_Course")
```

Task 1: Preliminary data analysis

Importing input signal, output signal, and time data

```{r}
input<-read.csv("X.csv")
output<-read.csv("y.csv")
times<-read.csv("time.csv")
```

combining output signal and input signal

```{r}
meg<-cbind(output,input)
```

Converting data to time series with interval of 10 sec

```{r}
meg1<-ts(meg,end = 20,frequency = 10,start = 0.1)
head(meg1)
```
```

Converting **time** series data to data frame structure Selecting column **y,x2,x1**

*##Time Series Time series plotting for x2=0 (neutral audio)*

```
` `{r}
tsmeg<-as.data.frame(meg1)%>%filter(x2=='0')%>%ts(start = 
0.1,end=10,frequency = 10)%>%
  plot(type='b',col='blue',main='time series plot of input and 
output signal x2=0')
` `
```

Time series plotting **for** x2=1 (Emotional audio)

```
` `{r}
tsmeg1<-as.data.frame(meg1)%>%filter(x2=='1')%>%ts(start = 10,end = 
20,frequency = 10)%>%
  plot(type='b',col='Red',main='time series plot of input and output 
signal x2=1')
` `
```

Distribution of **input signal and** output **signal**

Filtering the values **for** x2==0,neutral audio

```
` `{r}
megdist<-as.data.frame(meg1)%>%
filter(x2=='0')%>%
select(x1,y) #selecting column x1,y
` `
```

Filtering the values **for** x2==1,emotional audio

```
` `{r}
megdist1<-as.data.frame(meg1)%>%
  filter(x2=='1')%>%
  select(x1,y) #selecting column x1,y
` `
```

*##Histogram*

Histogram of output **signal**

```
` `{r}
hist(meg$y,xlab = 'output',main ='histogram of output_signal',col = 
'purple' )
` `
```

Histogram of Output **signal** when x2=0

```
` `{r}
hist(megdist$y,xlab = 'output',main ='histogram of 
output_signal(x2=0)',col = 'Yellow' )
` `
```

Histogram of Input **signal** when x2=0

```
```{r}
hist(megdist$x1,xlab = 'input',main='histogram of
input_signal(x2=0)',col = 'Red')
```
```

Histogram of Output **signal** when  $x_2=1$

```
```{r}
hist(megdist1$y,xlab = 'output',main = 'histogram of
output_signal(x2=1)',col = 'orange' )
```
```

Histogram of Input **signal** when  $x_2=1$

```
```{r}
hist(megdist1$x1,xlab = 'input',main='histogram of
input_signal(x2=1)',col = 'blue')
```
```

Correlation when  $x_2=0$

```
```{r}
megcor<-cor(megdist)
corrplot::corrplot.mixed(megcor,upper = "pie",lower = "number")
```
```

Correlation when  $x_2=1$

```
```{r}
megcor1<-cor(megdist1)
corrplot::corrplot.mixed(megcor1,upper = "pie",lower = "number")
```
```

*#Scatter Plot*

Scatter Plot when  $x_2=0$

```
```{r}
megdist%>%plot(type='p',main="Scatter plot of input and output
signal when  $x_2=0$ ",col='Red',xlab='Input Signal',ylab='Output
Signal')
```
```

Scatter Plot when  $x_2=1$

```
```{r}
megdist1%>%plot(type='p',main="Scatter plot of input and output
signal when  $x_2=1$ ",col='Orange',xlab='Input Signal',ylab='Output
Signal')
```
```

*##Box Plot Box plot when  $x_2=0$*

```
```{r}
boxplot(megdist$y,main='Boxplot of output signal when
 $x_2=0$ ',col='Red',ylab='Output Signal')
```

```

` ``
Box plot when x2=1

` ``{r}
boxplot(megdist1$y,main='Boxplot of output signal when
x2=1',col='Orange',ylab='Output Signal')
` ``

##Task 2 Regression
Task 2.1

` ``{r}
regrndata<-as.data.frame(meg)
` ``

Input values of x1

` ``{r}
input_1=regrndata$x1
input_2=regrndata$x1^2
input_3=regrndata$x1^3
input_4=regrndata$x1^4
input_5=regrndata$x1^5
` ``

Input values of x2

` ``{r}
x2=regrndata$x2
` ``

Output value

` ``{r}
y<-output
y<-as.matrix(y)
` ``

Model 1

` ``{r}
meg_model1<-data.frame(input0=c(1),input_3,input_5,x2) ## Creating
Data frame for Model 1
meg_mod1<-as.matrix(meg_model1)
head(meg_mod1)
meg_md1<-solve(t(meg_mod1)%*%meg_mod1)%*%t(meg_mod1)%*%y
##Estimating theta Parameter using least squares method
meg_md1
` ``

Model 2

` ``{r}
meg_model2<-data.frame(input0=c(1),input_1,x2) ##Creating Data frame
for Model 2

```

```

meg_mod2<-as.matrix(meg_model2)
head(meg_mod2)
meg_md2<-solve(t(meg_mod2)%*%meg_mod2)%*%t(meg_mod2)%*%y
##Estimating theta Parameter using least squares method
meg_md2
``,`

```

### Model 3

```

``,`{r}
meg_model3<-data.frame(input0=c(1),input_1,input_2,input_4,x2)
##Creating Data frame for Model 3
meg_mod3<-as.matrix(meg_model3)
head(meg_mod3)
meg_md3<-solve(t(meg_mod3)%*%meg_mod3)%*%t(meg_mod3)%*%y
##Estimating theta Parameter using least squares method
meg_md3
``,`

```

### Model 4

```

``,`{r}
meg_model4<-
data.frame(input0=c(1),input_1,input_2,input_3,input_5,x2)
##Creating Data frame for Model 4
meg_mod4<-as.matrix(meg_model4)
head(meg_mod4)
meg_md4<-solve(t(meg_mod4)%*%meg_mod4)%*%t(meg_mod4)%*%y
##Estimating theta Parameter using least squares method
meg_md4
``,`

```

### Model 5

```

``,`{r}
meg_model5<-data.frame(input0=c(1),input_1,input_3,input_4,x2)
##Creating Data frame for Model 5
meg_mod5<-as.matrix(meg_model5)
head(meg_mod5)
meg_md5<-solve(t(meg_mod5)%*%meg_mod5)%*%t(meg_mod5)%*%y
##Estimating theta Parameter using least squares method
meg_md5
``,`

```

### *##Task 2.2 RSS*

#### Model 1 meg\_RSS

```

``,`{r}
meg_rss1<-norm((y-(meg_mod1)%*%meg_md1))^2)
meg_rss1
``,`

```

#### Model 2 meg\_RSS

```

``,`{r}
meg_rss2<-norm((y-(meg_mod2)%*%meg_md2))^2)
meg_rss2
``,`

```

```

Model 3 meg_RSS
```{r}
meg_rss3<-norm((y-(meg_mod3%%meg_md3))^2)
meg_rss3
```

Model 4 meg_RSS
```{r}
meg_rss4<-norm((y-(meg_mod4%%meg_md4))^2)
meg_rss4
```

Model 5 meg_RSS
```{r}
meg_rss5<-norm((y-(meg_mod5%%meg_md5))^2)
meg_rss5
```

##Task2.3 Likelihood function

log likelihood 1
```{r}
meg_logfxn1<- -(nrow(y)/2*log(2*pi))-
(nrow(y)/2*(log(meg_rss1/(nrow(y)-1))))-
meg_rss1*(1/(2*(meg_rss1/(nrow(y)-1))))
meg_logfxn1
```

log likelihood 2
```{r}
meg_logfxn2<- -(nrow(y)/2*log(2*pi))-
(nrow(y)/2*(log(meg_rss2/(nrow(y)-1))))-
meg_rss2*(1/(2*(meg_rss2/(nrow(y)-1))))
meg_logfxn2
```

log likelihood 3
```{r}
meg_logfxn3<- -(nrow(y)/2*log(2*pi))-
(nrow(y)/2*(log(meg_rss3/(nrow(y)-1))))-
meg_rss3*(1/(2*(meg_rss3/(nrow(y)-1))))
meg_logfxn3
```

log likelihood 4
```{r}
meg_logfxn4<- -(nrow(y)/2*log(2*pi))-
(nrow(y)/2*(log(meg_rss4/(nrow(y)-1))))-
meg_rss4*(1/(2*(meg_rss4/(nrow(y)-1))))
meg_logfxn4
```

log likelihood 5
```{r}
meg_logfxn5<- -(nrow(y)/2*log(2*pi))-
(nrow(y)/2*(log(meg_rss5/(nrow(y)-1))))-
meg_rss5*(1/(2*(meg_rss5/(nrow(y)-1))))
meg_logfxn5
```

#TASK 2.4 AIC AND BIC
K=no:of parameters estimated

```

```

Model 1 AIC & BIC
```{r}
length(meg_md1)
```

```{r}
k1=4
meg_AIC1<-2*k1-(2*meg_logfxn1)
meg_AIC1
meg_BIC1<-k1*log(nrow(y))-(2*(meg_logfxn1))
meg_BIC1

```

Model 2 AIC & BIC
```{r}
length(meg_md2)
```

```{r}
k2=3
meg_AIC2<-2*k2-(2*meg_logfxn2)
meg_AIC2
meg_BIC2<-k2*log(nrow(y))-(2*(meg_logfxn2))
meg_BIC2
```

Model 3 AIC & BIC
```{r}
length(meg_md3)
```

```{r}
k2=5
meg_AIC3<-2*k2-(2*meg_logfxn3)
meg_AIC3
meg_BIC3<-k2*log(nrow(y))-(2*(meg_logfxn3))
meg_BIC3
```

Model 4 AIC & BIC
```{r}
length(meg_md4)
```

```{r}
k2=6
meg_AIC4<-2*k2-(2*meg_logfxn4)
meg_AIC4
meg_BIC4<-k2*log(nrow(y))-(2*(meg_logfxn4))
meg_BIC4
```

Model 5 AIC & BIC
```{r}
length(meg_md5)
```

```{r}
k2=5
meg_AIC5<-2*k2-(2*meg_logfxn5)
meg_AIC5
meg_BIC5<-k2*log(nrow(y))-(2*(meg_logfxn5))
meg_BIC5

```

```

` ``
##TASK 2.5 Check the distribution of each model prediction errors
Plot the errors and check if they are normally distributed

Model 1 residual distribution
````{r}
meg_err1=(y-(meg_mod1%*%meg_md1)) #error
head(meg_err1)
````

Model one Output error signal distribution
````{r}
qqnorm(meg_err1,main ="Model one Output error signal
distribution") #error plot
qqline(meg_err1,col="Orange")
````

Model one output  signals distribution
````{r}
qqnorm(meg_md1,main = "Model one output signals distribution")
qqline(meg_md1,col="Red")
````

Model 2 residual distribution
````{r}
meg_err2=(y-(meg_mod2%*%meg_md2)) #error
head(meg_err2)
````

Model Two Output error signal distribution
````{r}
qqnorm(meg_err2,main ="Model Two Output error signal
distribution") #error plot
qqline(meg_err2,col="Orange")
````

Model Two output  signals distribution
````{r}
qqnorm(meg_md2,main = "Model Two output signals distribution")
qqline(meg_md2,col="Red")
````

Model 3 residual distribution
````{r}
meg_err3=(y-(meg_mod3%*%meg_md3)) #error
head(meg_err3)
````

Model Three Output error signal distribution
````{r}
qqnorm(meg_err3,main ="Model Three Output error signal
distribution") #error plot
qqline(meg_err3,col="Orange")
````

Model Three output signals distribution
````{r}
qqnorm(meg_md3,main = "Model Three output signals distribution")
qqline(meg_md3,col="Red")
````

Model 4 residual distribution
````{r}
meg_err4=(y-(meg_mod4%*%meg_md4)) #error

```



```

head(meg_err4)
```
Model Four Output error signal distribution
```{r}
qqnorm(meg_err4,main ="Model Four Output error signal
distribution") #error plot
qqline(meg_err4,col="Orange")
```
Model Four output signals distribution
```{r}
qqnorm(meg_md4,main = "Model Four output signals distribution")
qqline(meg_md4,col="Red")
```
Model 5 residual distribution
```{r}
meg_err5=(y-(meg_mod5%*%meg_md5)) #error
head(meg_err5)
```
Model Five Output error signal distribution
```{r}
qqnorm(meg_err5,main ="Model Five Output error signal
distribution") #error plot
qqline(meg_err5,col="Orange")
```
Model Five output signals distribution
```{r}
qqnorm(meg_md5,main = "Model Five output signals distribution")
qqline(meg_md5,col="Red")
```

#TASK 2.6 Select the best regression model according to AIC and BIC

The best regression model is model 3, reason: it has the smallest
values of AIC(997.9368)
and BIC(1014.428) comparing of all the 5 regression models.

#Task 2.7
Splitting the data for training and testing

```{r}
set.seed(123)
tradata<-sample(1:200,140) #Taking sample of 140 rows
head(tradata)
```
Training Data
```{r}
training_data<-regrndata[tradata,]
head(training_data)
```
Testing Data
```{r}
testing_data<-regrndata[-tradata,]
head(testing_data)
```
Estimating Model parameter

```

```

Model 3
Training Input Signal
```{r}
meg_model03<-
data.frame(input0=c(1),training_data$x1,training_data$x1^2,training_
data$x1^4,training_data$x2)
meg_mod03<- as.matrix(meg_model03)
head(meg_mod03)
```

Training Output Signal
```{r}
y1<-training_data$y
y1<-as.matrix(y1)
meg_md03<-solve(t(meg_mod03)%*%meg_mod03)%*%t(meg_mod03)%*%y1
meg_md03
```

Prediction of Testing Data
```{r}
meg_est<-sum(meg_md03)/length(y1)
meg_est
```

```{r}
pred<-meg_est*testing_data
head(pred)
```

95% Confidence Interval of Output signal
```{r}
sd<-sqrt(var(pred$y))
sd
means<-mean(pred$y)
means
paste('Confidence Interval',means-
1.96*sd/sqrt(nrow(testing_data)),means+1.96*sd/sqrt(nrow(testing_dat
a)))
```

95% Confidence Interval of Input signal
```{r}
sd1<-sqrt(var(pred$x1))
sd1
means1<-mean(pred$x1)
means1
paste('Confidence Interval',means1-
1.96*sd1/sqrt(nrow(testing_data)),means1+1.96*sd1/sqrt(nrow(testing_
data)))
```

Error
```{r}
meg_error<-testing_data$y-pred$y
head(meg_error)
meg_error1<-testing_data$y+pred$y
head(meg_error1)
```

Adding Confidence limits to predict the data
```{r}
pred_Lower<-meg_error
pred_Upper<-meg_error1

```

```

pred_testy<-testing_data$y
```

Plotting Confidence intervals and error bars
```{r}
pred %>%
ggplot(aes(x1,y))+geom_errorbar(ymin=pred_Lower,ymax=pred_Upper)+
geom_point(aes(x1),col='Red')
+geom_point(aes(x1,pred_testy),col='Yellow')+ggtitle('Confidence
Error Bar Plot')+theme(plot.title = element_text(hjust =
0.5))+xlab('Input Data Set') + ylab('Output Data Set')
```

#Task 3 APPROXIMATE BAYESIAN COMPUTATION (ABC)
#1 2 parameters with largest absolute least squares estimation
```{r}
meg_mean1<-colSums(abs(meg_mod1))/nrow(meg_mod1)
head(meg_mean1)
meg_mean2<-colSums(abs(meg_mod2))/nrow(meg_mod2)
head(meg_mean2)
meg_mean3<-colSums(abs(meg_mod3))/nrow(meg_mod3)
head(meg_mean3)
meg_mean4<-colSums(abs(meg_mod4))/nrow(meg_mod4)
head(meg_mean4)
meg_mean5<-colSums(abs(meg_mod5))/nrow(meg_mod5)
head(meg_mean5)
```

#model 1 has the largest absolute least squares estimate Input 3 &
Input 5
```{r}
modell<-as.data.frame(meg_mod1)
head(modell)
```

prior for parameters Input 3 and input 5
```{r}
p1<-runif(min =
min(modell$input_3),max=max(modell$input_3),nrow(modell))
head(p1)
p2<-runif(min =
min(modell$input_5),max=max(modell$input_5),nrow(modell))
head(p2)
```

Sample Value
```{r}
n=50
```

Importing for ABC Rejection
```{r}
library(EasyABC)
```

Plot Joint Distribution of parameters
```{r}
sample_model <-function(x)
{
  c(x[3]^3+p1,x[5]^5+p2)
}
prior <- list(c('unif',0,1))
```

```

```

ABC rejection with tolerance of 0.2
```{r}
simulation <- ABC_rejection(model = sample_model,prior =
prior,nb_simul = n,tol = 0.2)
densityPlot(simulation$param,xlab = 'stimulated Parameters',main =
'joint distribution of 2 parameters input 3 & input 5',col =
'Orange')
```

plot marginal distribution of Input 3 parameter
```{r}
sample_model1<-function(x)
{
  x[3]^3+p1
}
prior1 <- list(c('unif',0,1))
```

ABC rejection with tolerance of 0.2
```{r}
simulation1<-ABC_rejection(model = sample_model1,prior =
prior1,nb_simul = n,tol = 0.2)
densityPlot(simulation1$param,xlab = 'stimulated Parameters',main =
'joint distribution of input 3',col = 'Violet')
```

plot marginal distribution of Input 5 parameter
```{r}
sample_model2<-function(x)
{
  x[5]^5+p2
}
prior2 <- list(c('unif',0,1))
```

ABC rejection with tolerance of 0.2
```{r}
simulation2<-ABC_rejection(model = sample_model2,prior =
prior2,nb_simul = n,tol = 0.2)
densityPlot(simulation2$param,xlab = 'stimulated Parameters',main =
'joint distribution of input 5',col = 'Red')
```

#Result
The density curves seems to normal distributed for both the marginal
and joint distribution of the 2 posterior parameters.

```