(7089CEM)

Coursework
Demonstration of Statistical Method Program

MODULE LEADER: Dr. Fei He
Students Name: Manoj Sibi Mogan
SID: 10224061

STATISTICAL METHODS

I can confirm that all work submitted is my own: Yes

Table of Contents

1. Introduction

In this paper I developed a coherent mathematical framework for linking EEG-based regression. As a result, it is proposed that regulation from linear combinations of unrelated statistical sources can be considered as ict. Then I displayed numerical simulations that deal with different regression models with the model violations commonly encountered. Next, I made a detailed model comparisons for issues within and between subject EEG data. Finally, I examined practical issues related to the availability of source modelling and pre-processing options.

EEG analysis is largely based on visual inspection of relevant EEG signals. However, in most cases visual inspection of EEG signals is subjective and inadequate. This is because the statistical information contained in the EEG signals cannot be adequately utilized and used. To obtain more relatively objective and reliable diagnostic results, several methods for quantitative analysis of EEG signals have been proposed.

2. Implementation

2.1.Background Study

2.1.1. Regression

Regression Analysis is the initial analysis part of data science. It is a well-understood model in numerical simulation. It is easy for developers to work with all machine learning algorithms when they have good hands-on experience with working on regression models. These models are easily interpretable which are based on the mathematical bases like matrix and algebra etc. This process is implemented to study the relationship between the set of independent variable and dependent variable.

2.1.2. Linear Regression

Linear regression is the most commonly considered analysis method used for examining the relationship between a quantitative outcome and a single quantitative explanatory variable. Linear regression analysis is a significant concept in statistics or machine learning and frequently used as a simple model. It is also widely used in practice because the underlying models are interpretable and does not require much data to use. In linear regression contains many applications and associated pitfalls which requires careful study.

2.1.3. Non-linear Regression

Non-linear regression analysis is a common technique which is used in mathematics and engineering. Least-squares with Gauss-Newton method is the most widely used approach in Non-linear regression. This regression method is suited for analysing data for which there is a practically or theoretically functional relationship between response and predictor.

2.1.4. Polynomial Regression

Polynomial regression is a form of linear regression in which the relationship between the independent variable x and a dependent variable y is modelled as an nth degree polynomial. It also fits a non-linear relationship between the value of x and the mean y, which denotes E(y|x). Polynomial regression may lead to increase in loss function, high error and decrease in accuracy.

2.2. Installation Steps

Machine used to perform this study has the hard disk drive with the capacity of 1TB, RAM with 16GB and the used operating system is macOS Big Sur version 11.2.2.

Step 1: To install R and RStudio in a Mac system we need to download Xcode from the App store and install it in the system

Step 2: Need to download and install Command Line Tool to support Xcode (https://developer.apple.com/download/more/)

Step 3: After then type "sudo xcodebuild -license" in Terminal and then type the system password and "Agree" the license for installing Xcode

Step 4: Need to download and install MacPorts (https://www.macports.org/install.php)

Step 5: Need to download and install XQuartz from their website (https://www.xquartz.org)

Step 6: Download R from (http://lib.stat.cmu.edu/R/CRAN/)

Step 7: Click on the Download R for (Mac) OS X available in the webpage and Download R-4.0.4.pkg(or a newer version) (http://lib.stat.cmu.edu/R/CRAN/bin/macosx/R-4.0.4.pkg)

Step 8: Install R with all default settings

Step 9: Download RStudio Desktop for mac (https://download1.rstudio.org/RStudio-1.0.153.dmg)

2.3. Modelling EEG signals using polynomial regression

```
library(PerformanceAnalytics)
library(mcmc)
library(coda)
data(logit)

x<-read.table("/Users/manoj/desktop/7089CEM - Introduction to
Statistical methods/Course
Work/x.csv",header=FALSE,sep=",",dec=".")
y<-read.table("/Users/manoj/desktop/7089CEM - Introduction to
Statistical methods/Course
Work/y.csv",header=FALSE,sep=",",dec=".")
time<-read.table("/Users/manoj/desktop/7089CEM - Introduction
to Statistical methods/Course
Work/time.csv",header=FALSE,sep=",",dec=".")
data<-data.frame(time,y,x)
names(data)<-c("time","y","x1","x2","x3","x4")
head(data)
```

Output

	time	y	x1	x2	x3	x4
1	0.000	-1.11678535	-3.0615656	-1.8651967	-2.90981227	-0.3394280
2	0.002	1.33388908	0.2446936	2.0391609	0.99930151	0.6122276
3	0.004	0.21606254	-1.2009670	0.1774090	-0.79654755	-2.6770103
4	0.006	0.17645157	-0.7279816	-1.5566153	0.01039548	-1.6230083
5	0.008	0.01104331	2.2049284	0.7661962	1.86810725	1.4313685
6	0.01	1.03716149	3.1672849	1.8484216	1.73971053	2.4726877

2.4. Task 1: Preliminary data analysis

2.4.1. Time series plots (input and output EEG signals)

Output Signal

```
library(PerformanceAnalytics)
library(mcmc)
library(coda)
data(logit)

x<-read.table("/Users/manoj/desktop/7089CEM – Introduction to
Statistical methods/Course
Work/x.csv",header=FALSE,sep=",",dec=".")
y<-read.table("/Users/manoj/desktop/7089CEM – Introduction to
Statistical methods/Course
Work/y.csv",header=FALSE,sep=",",dec=".")
time<-read.table("/Users/manoj/desktop/7089CEM – Introduction
to Statistical methods/Course
Work/time.csv",header=FALSE,sep=",",dec=".")
data<-data.frame(time,y,x)
names(data)<-c("time","y","x1","x2","x3","x4")
head(data)

plot(data$time,data$y, type="l", col="red")
```

There is a natural segmentation of time in 15 second intervals for this case. In principal the discretization of time is arbitrary and I divided the period into overlapping, or even varying length (major peak-to-peak say) that can be classified.

To give finer resolution, I eventually settled on an interval of 2 seconds that seemed to give the optimum trade-off between sensitivity and computational intensity (Alkan et al., 2005 Wei, et al., 2010).

Another important question that needs to be address is the amount of down sampling that to be used. At 8000Hz, a time period of 5 minutes represents 2,400,000 (8000*5*60) data points for each channel. This is typically too much information, and in particular, risk overtraining and fitting to underlying noise.

Output

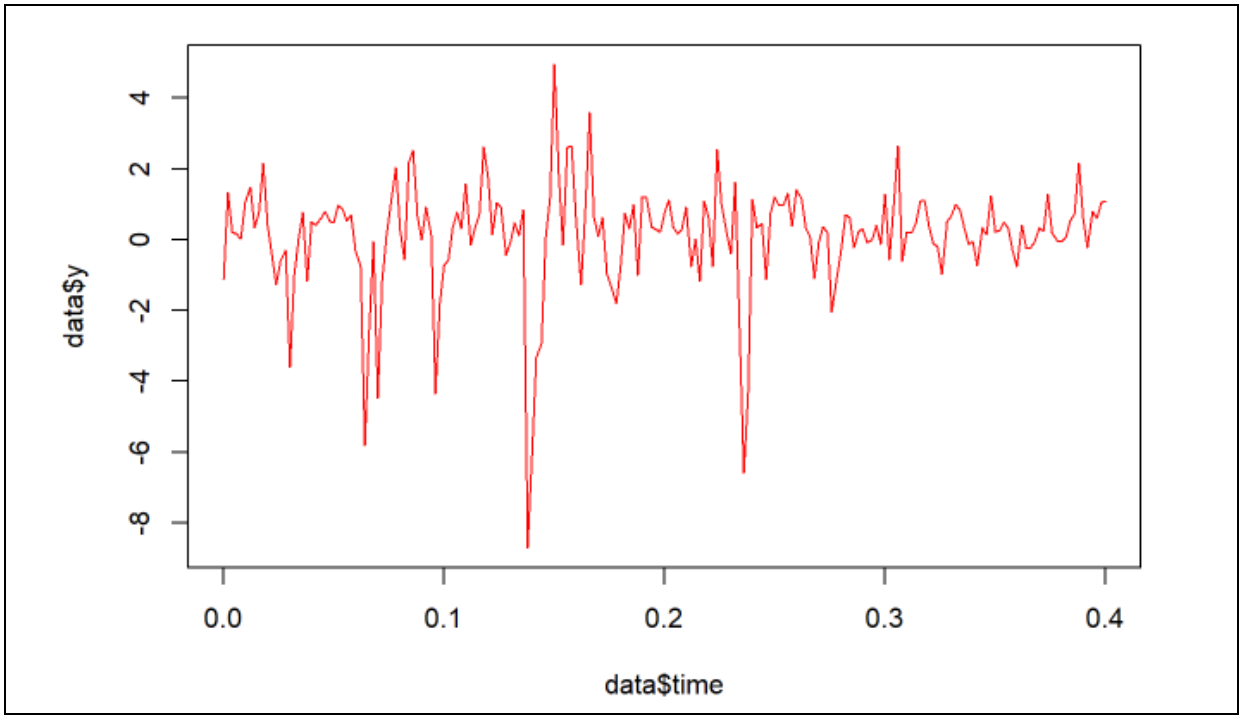

Image 1. Parameters distribution between X and Y

This is the function of x and y. While x is the input signal and y output signal. The parameters are estimated accordingly.

Input Signal

```
library(PerformanceAnalytics)
library(coda)

x<-read.table("/Users/manoj/desktop/7089CEM – Introduction to
Statistical methods/Course
Work/x.csv",header=FALSE,sep=",",dec=".")
y<-read.table("/Users/manoj/desktop/7089CEM – Introduction to
Statistical methods/Course
Work/y.csv",header=FALSE,sep=",",dec=".")
time<-read.table("/Users/manoj/desktop/7089CEM – Introduction
to Statistical methods/Course
Work/time.csv",header=FALSE,sep=",",dec=".")
data<-data.frame(time,y,x)
names(data)<-c("time","y","x1","x2","x3","x4")
head(data)

par(mfrow=c(2,2))
plot(data$time,data$x1, type="l", col="red",main="EEG signal
x1")
```

```
plot(data$time,data$x2, type="l", col="red",main="EEG signal
x2")
plot(data$time,data$x3, type="l", col="red",main="EEG signal
x3")
plot(data$time,data$x4, type="l", col="red",main="EEG signal
x4")
```

Output

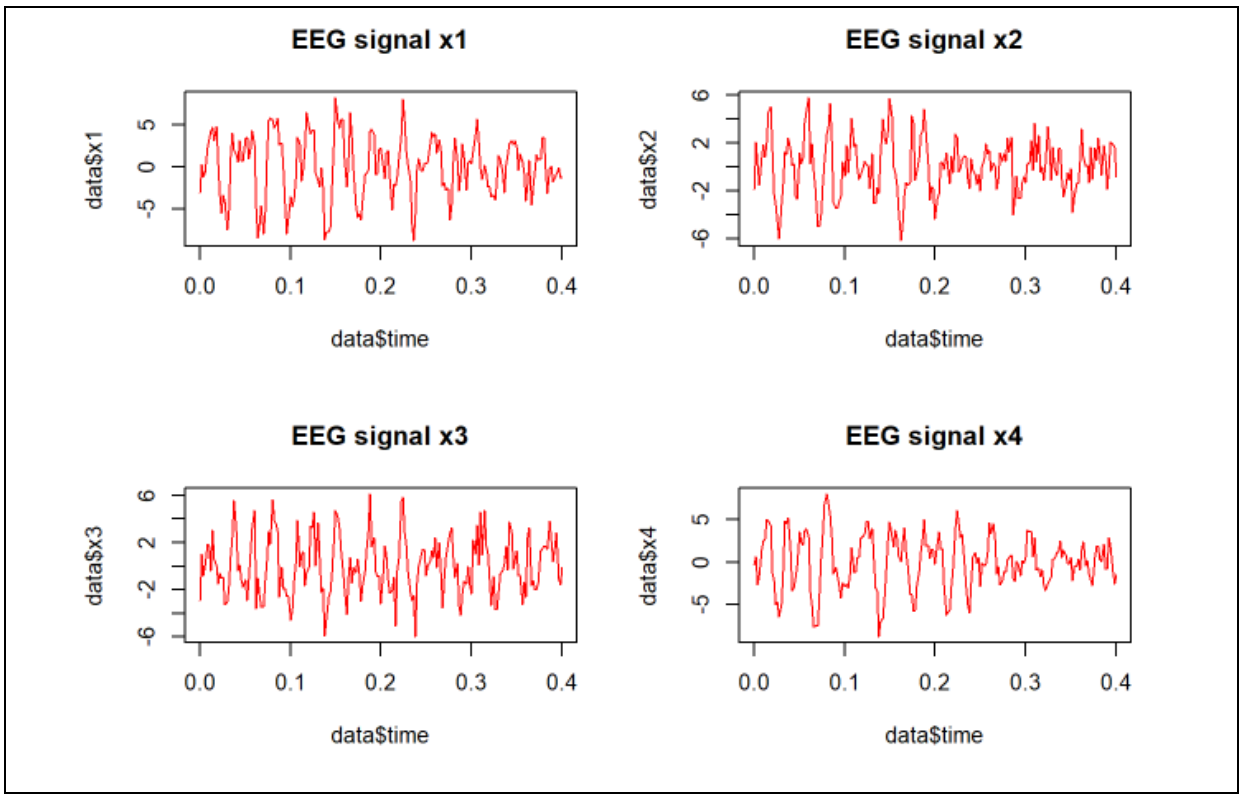

Image 2. Histogram of all signal distribution

Turning a time-series dataset into high-level features, frequency-domain or even time-frequency domain information, can yield very powerful and accurate predictions.

Window optimizations, feature crafting, and basic analysis flow would have to be tweaked but with these few modifications, depending on the exact problem, any time-series data can be analysed in this manner (Subasi and Ercelebi, 2005).

There are various machine vision techniques, such as CWT and spectrograms, that can also be added to increase our predictive power and help classify complex datasets.

2.4.2. Distribution for each EEG signal

```
library(PerformanceAnalytics)
library(mcmc)
library(coda)
data(logit)

x<-read.table("/Users/manoj/desktop/7089CEM - Introduction to
Statistical methods/Course
Work/x.csv",header=FALSE,sep=",",dec=".")
y<-read.table("/Users/manoj/desktop/7089CEM - Introduction to
Statistical methods/Course
Work/y.csv",header=FALSE,sep=",",dec=".")
time<-read.table("/Users/manoj/desktop/7089CEM - Introduction
to Statistical methods/Course
Work/time.csv",header=FALSE,sep=",",dec=".")
data<-data.frame(time,y,x)
names(data)<-c("time","y","x1","x2","x3","x4")
head(data)

hist(data$y,main="Distribution of output signal y")
```

The electroencephalogram (EEG) is a recording of the electrical activity of the brain from the scalp. The recorded waveforms reflect the cortical electrical activity.

Signal intensity: EEG activity is quite small, measured in microvolts (mV).

Signal frequency: the main frequencies of the human EEG waves are:

Delta: has a frequency of 3 Hz or below. It tends to be the highest in amplitude and the slowest waves. It is normal as the dominant rhythm in infants up to one year and in stages 3 and 4 of sleep. It may occur focally with subcortical lesions and in general distribution with diffuse lesions, metabolic encephalopathy hydrocephalus or deep midline lesions. It is usually most prominent frontally in adults (e.g. FIRDA - Frontal Intermittent Rhythmic Delta) and posteriorly in children e.g. OIRDA - Occipital Intermittent Rhythmic Delta) (Ghergulescu, and Muntean, 2016).

Theta: has a frequency of 3.5 to 7.5 Hz and is classified as "slow" activity. It is perfectly normal in children up to 13 years and in sleep but abnormal in awake adults. It can be seen as a manifestation of focal subcortical lesions; it can also be seen in generalized distribution in diffuse disorders such as metabolic encephalopathy or some instances of hydrocephalus.

Alpha: has a frequency between 7.5 and 13 Hz. Is usually best seen in the posterior regions of the head on each side, being higher in amplitude on the dominant side. It appears when closing the eyes and relaxing, and disappears when opening the eyes or alerting by any mechanism

(thinking, calculating). It is the major rhythm seen in normal relaxed adults. It is present during most of life especially after the thirteenth year.

Beta: beta activity is "fast" activity. It has a frequency of 14 and greater Hz. It is usually seen on both sides in symmetrical distribution and is most evident frontally. It is accentuated by sedative-hypnotic drugs especially the benzodiazepines and the barbiturates. It may be absent or reduced in areas of cortical damage. It is generally regarded as a normal rhythm. It is the dominant rhythm in patients who are alert or anxious or have their eyes open (Vigário., 1997).

Output

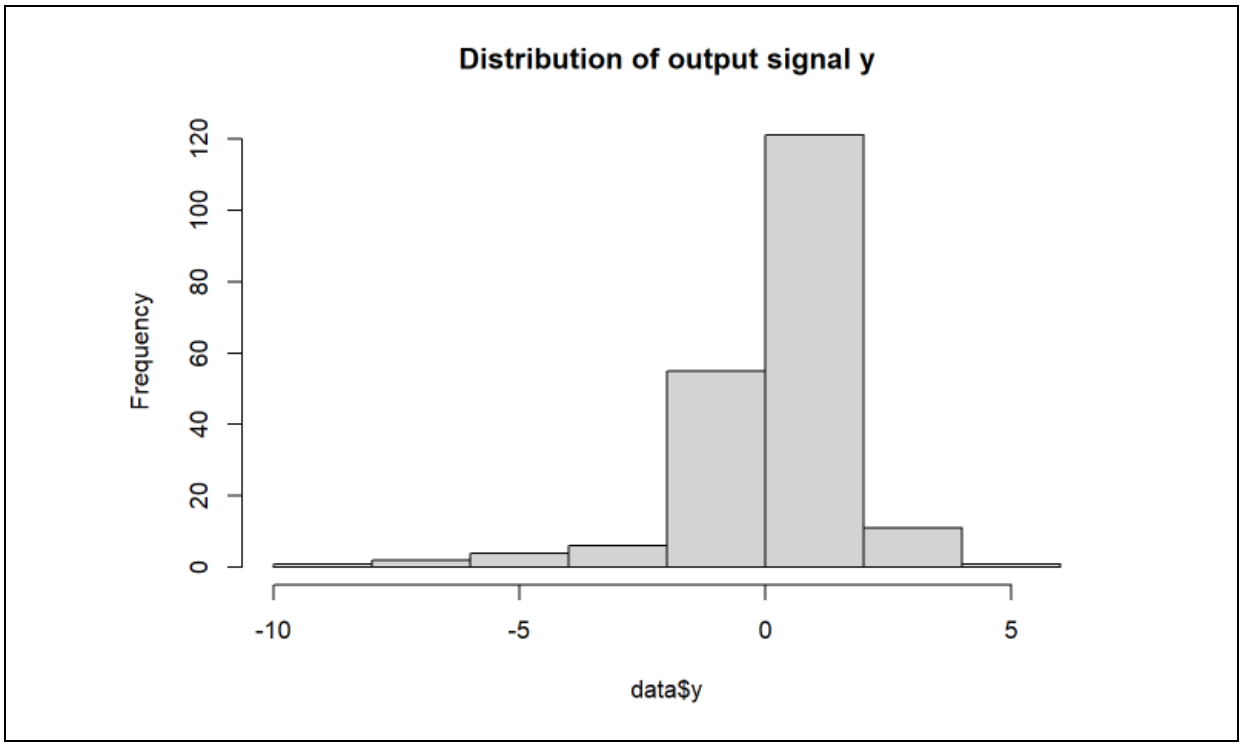

Image 3. ***Distribution of output signal y***

The above graph gives distribution of output signal y. the frequency was found to be highest with output value after o and the frequency was found to the around -10 and 5.

```
library(PerformanceAnalytics)
library(mcmc)
library(coda)
data(logit)

x<-read.table("/Users/manoj/desktop/7089CEM – Introduction to
Statistical methods/Course
Work/x.csv",header=FALSE,sep=",",dec=".")
```

```
y<-read.table("/Users/manoj/desktop/7089CEM - Introduction to
Statistical methods/Course
Work/y.csv",header=FALSE,sep=",",dec=".")
time<-read.table("/Users/manoj/desktop/7089CEM - Introduction
to Statistical methods/Course
Work/time.csv",header=FALSE,sep=",",dec=".")
data<-data.frame(time,y,x)
names(data)<-c("time","y","x1","x2","x3","x4")
head(data)

par(mfrow=c(2,2))
hist(data$x1,main="Distribution of signal x1")
hist(data$x2,main="Distribution of signal x2")
hist(data$x3,main="Distribution of signal x3")
hist(data$x4,main="Distribution of signal x4")
```

Output

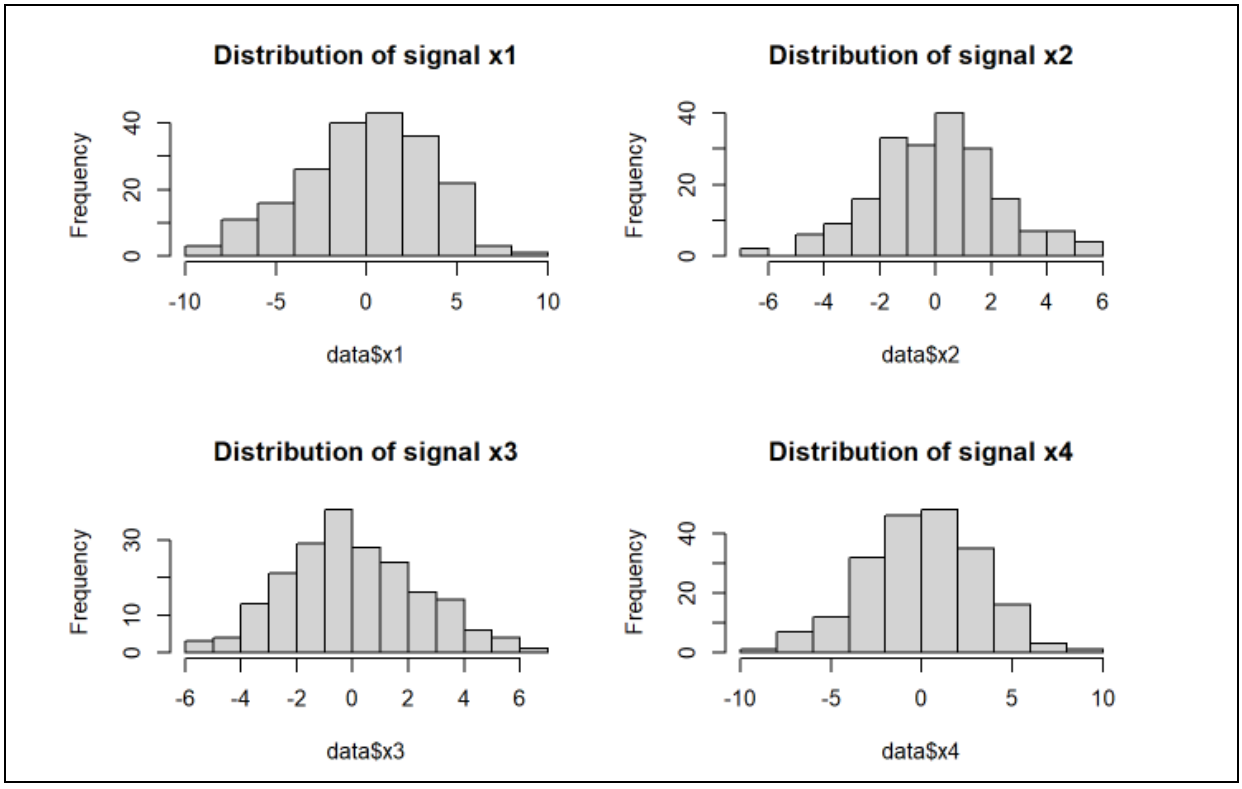

Image 4. Frequency distribution of x1, x2, x3 and x4

The above graphs provide the frequency distribution of x1, x2, x3, and x4. The highest frequency distribution of 40 was reached in point 1 in x, x2, x3 and x4 while the least frequency distribution was found at point 10 in x1, -5 in x2, 6 in x3 and 10 in x4.

2.4.3. Correlation and scatter plots (between different input EEG signals and the output EEG)

chart.Correlation(data[,-1], histogram=TRUE, pch=19)

Output

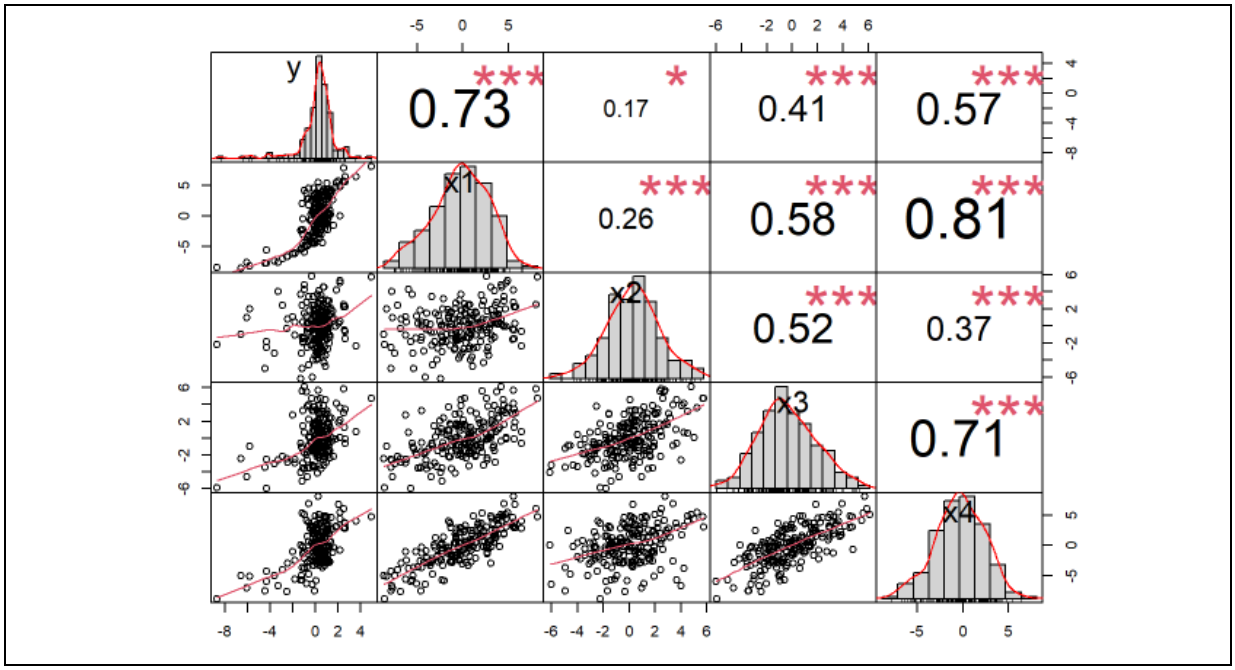

Image 5. Correlation between input and output signals

The above graph provides the correlation between input x1, x2, x3, and x4 and output Y. the correlation between input and output signal was 0.81 and the least correlation was achieved at 0.17 between input and output signals.

2.5. Task 2: Regression- modelling the relationship between EEG signals

```
library(PerformanceAnalytics)
library(mcmc)
library(coda)
data(logit)

x<-read.table("/Users/manoj/desktop/7089CEM - Introduction to
Statistical methods/Course
Work/x.csv",header=FALSE,sep=",",dec=".")
y<-read.table("/Users/manoj/desktop/7089CEM - Introduction to
Statistical methods/Course
Work/y.csv",header=FALSE,sep=",",dec=".")
```

```
time<-read.table("/Users/manoj/desktop/7089CEM - Introduction
to Statistical methods/Course
Work/time.csv",header=FALSE,sep=",",dec=".")
data<-data.frame(time,y,x)
names(data)<-c("time","y","x1","x2","x3","x4")

model1<-
data.frame(z1=data$x4,z2=data$x1^2,z3=data$x1^3,z4=data$x3^4,z
0=1)

head(model1)
```

Output (Model1)

	z1	z2	z3	z4	z0
1	-0.3394280	9.37318384	-28.69661707	7.169022e+01	1
2	0.6122276	0.05987496	0.01465102	9.972090e-01	1
3	-2.6770103	1.44232163	-1.73218061	4.025750e-01	1
4	-1.6230083	0.52995728	-0.38579917	1.167824e-08	1
5	1.4313685	4.86170924	10.71972075	1.217888e+01	1
6	2.4726877	10.03169378	31.77323244	9.160264e+00	1

```
model2<-data.frame(z1=data$x3^3,z2=data$x3^4,z0=1)

head(model2)
```

Output (Model2)

	z1	z2	z0
1	-2.463740e+01	7.169022e+01	1
2	9.979060e-01	9.972090e-01	1
3	-5.053999e-01	4.025750e-01	1
4	1.123397e-06	1.167824e-08	1
5	6.519367e+00	1.217888e+01	1
6	5.265395e+00	9.160264e+00	1

```
model3<-data.frame(z1=data$x2,z2=data$x1^3,z3=data$x3^4,z0=1)
head(model3)
```

Output (Model3)

	z1	z2	z3	z0
1	-1.8651967	-28.69661707	7.169022e+01	1
2	2.0391609	0.01465102	9.972090e-01	1
3	0.1774090	-1.73218061	4.025750e-01	1
4	-1.5566153	-0.38579917	1.167824e-08	1
5	0.7661962	10.71972075	1.217888e+01	1
6	1.8484216	31.77323244	9.160264e+00	1

```
model4<-data.frame(z1=data$x4,z2=data$x1^3,z3=data$x3^4,z0=1)
head(model4)
```

Output (Model4)

	z1	z2	z3	z0
1	-0.3394280	-28.69661707	7.169022e+01	1
2	0.6122276	0.01465102	9.972090e-01	1
3	-2.6770103	-1.73218061	4.025750e-01	1
4	-1.6230083	-0.38579917	1.167824e-08	1
5	1.4313685	10.71972075	1.217888e+01	1
6	2.4726877	31.77323244	9.160264e+00	1

```
model5<-
data.frame(z1=data$x4,z2=data$x1^2,z3=data$x1^3,z4=data$x3^4,z
5=data$x1^4,z0=1)

head(model5)
```

Output (Model5)

	z1	z2	z3	z4	z5	z0
1	-0.3394280	9.37318384	-28.69661707	7.169022e+01	8.785658e+01	1
2	0.6122276	0.05987496	0.01465102	9.972090e-01	3.585011e-03	1
3	-2.6770103	1.44232163	-1.73218061	4.025750e-01	2.080292e+00	1
4	-1.6230083	0.52995728	-0.38579917	1.167824e-08	2.808547e-01	1
5	1.4313685	4.86170924	10.71972075	1.217888e+01	2.363622e+01	1
6	2.4726877	10.03169378	31.77323244	9.160264e+00	1.006349e+02	1

There are totally five models with defined variables as z1 to z5 in design matrix. These variables depend and vary according the models. For example, some models can have 3 variables while some have four and its goes on and on. But, all these models have intercept in theta_bias.

Task 2.1

```
model1<-
data.frame(z1=data$x4,z2=data$x1^2,z3=data$x1^3,z4=data$x3^4,z0=1)

head(model1)

ols<-function(X,y)
     {
       X<-as.matrix(X)
       Y<-as.matrix(y)
       X.X<-solve(t(X) %*% X)
       b<-b<-X.X %*% (t(X) %*% Y)
       colnames(b)<- "OLS_parm"
       return(b)}
X1<-as.matrix(model1)
pram_m1<-solve(t(X1) %*% X1) %*% (t(X1) %*% data$y)
# Model-1 parameters
pram_m1
```

Output

```
           [,1]

z1  -0.034101975

z2  -0.001849575

z3   0.010381813

z4  -0.001949154

z0   0.468551685
```

```
model2<-data.frame(z1=data$x3^3,z2=data$x3^4,z0=1)
head(model2)

ols<-function(X,y)
     {
       X<-as.matrix(X)
       Y<-as.matrix(y)
       X.X<-solve(t(X) %*% X)
       b<-b<-X.X %*% (t(X) %*% Y)
       colnames(b)<- "OLS_parm"
       return(b)}
X2<-as.matrix(model2)
pram_m2<-solve(t(X2) %*% X2) %*% (t(X2) %*% data$y)
# Model-2 parameters
```

```
pram_m2
```

Output

```
           [,1]

z1   0.016334677

z2   -0.002713985

z0   0.303501144
```

```
model3<-data.frame(z1=data$x2,z2=data$x1^3,z3=data$x3^4,z0=1)
head(model3)

ols<-function(X,y)
     {
       X<-as.matrix(X)
       Y<-as.matrix(y)
       X.X<-solve(t(X) %*% X)
       b<-b<-X.X %*% (t(X) %*% Y)
       colnames(b)<- "OLS_parm"
       return(b)}
X3<-as.matrix(model3)
pram_m3<-solve(t(X3) %*% X3) %*% (t(X3) %*% data$y)
# Model-3 parameters
pram_m3
```

Output

```
          [,1]

z1   0.038109255

z2  0.009827804

z3  -0.002092558

z0  0.448299550
```

```
model4<-data.frame(z1=data$x4,z2=data$x1^3,z3=data$x3^4,z0=1)
head(model4)

ols<-function(X,y)
     {
       X<-as.matrix(X)
```

```
        Y<-as.matrix(y)
        X.X<-solve(t(X) %*% X)
        b<-b<-X.X %*% (t(X) %*% Y)
        colnames(b)<- "OLS_parm"
        return(b)}
X4<-as.matrix(model4)
pram_m4<-solve(t(X4) %*% X4) %*% (t(X4) %*% data$y)
# Model-4 parameters
pram_m4
```

Output

```
           [,1]

z1  -0.034881772

z2   0.010482178

z3  -0.001990496

z0   0.450329209
```

```
model5<-
data.frame(z1=data$x4,z2=data$x1^2,z3=data$x1^3,z4=data$x3^4,z
5=data$x1^4,z0=1)
head(model5)

ols<-function(X,y)
     {
       X<-as.matrix(X)
       Y<-as.matrix(y)
       X.X<-solve(t(X) %*% X)
       b<-b<-X.X %*% (t(X) %*% Y)
       colnames(b)<- "OLS_parm"
       return(b)}
X5<-as.matrix(model5)
pram_m5<-solve(t(X5) %*% X5) %*% (t(X5) %*% data$y)
# Model-5 parameters
pram_m5
```

Output

```
            [,1]

z1   -3.395313e-02

z2   -2.701644e-04

z3   1.034764e-02

z4   -1.943452e-03

z5   -3.083194e-05

z0   4.606560e-01
```

2.5.1. Task 2.2

```
rss_m1<-sum((data$y - X1 %*% pram_m1)^2)
rss_m2<-sum((data$y - X2 %*% pram_m2)^2)
rss_m3<-sum((data$y - X3 %*% pram_m3)^2)
rss_m4<-sum((data$y - X4 %*% pram_m4)^2)
rss_m5<-sum((data$y - X5 %*% pram_m5)^2)
rss<-data.frame(Model=paste("Model",1:5, sep="-"),
                RSS=c(rss_m1,rss_m2,rss_m3,rss_m4,rss_m5))
Rss
```

Output

```
     Model       RSS

1    Model-1   57.32927

2    Model-2   351.41466

3    Model-3   57.32536

4    Model-4   57.46405

5    Model-5   57.30923
```

2.5.2. Task 2.3

```
n<- dim(data)[1]
A<- -(n/2)*log(2*pi)
log_lik1<- A - (n/2)*log(rss_m1/(n-1)) -(1/2*rss_m1/(n-1))*rss_m1
log_lik2<- A - (n/2)*log(rss_m2/(n-1)) -(1/2*rss_m2/(n-1))*rss_m2
log_lik3<- A - (n/2)*log(rss_m3/(n-1)) -(1/2*rss_m3/(n-1))*rss_m3
log_lik4<- A - (n/2)*log(rss_m4/(n-1)) -(1/2*rss_m4/(n-1))*rss_m4
log_lik5<- A - (n/2)*log(rss_m5/(n-1)) -(1/2*rss_m5/(n-1))*rss_m5
likelihood<-data.frame(Model=paste("Model",1:5, sep="-"),

Log_Likelihood=c(log_lik1,log_lik2,log_lik3,log_lik4,log_lik5)
)

Likelihood
```

Output

	Model	Log_Likelihood
1	Model-1	-67.34790
2	Model-2	-550.08409
3	Model-3	-67.33992
4	Model-4	-67.62258
5	Model-5	-67.30703

2.5.3. Task 2.4

```
aic_m1<- 2*length(pram_m1) - 2*log_lik1
aic_m2<- 2*length(pram_m2) - 2*log_lik2
aic_m3<- 2*length(pram_m3) - 2*log_lik3
aic_m4<- 2*length(pram_m4) - 2*log_lik4
aic_m5<- 2*length(pram_m5) - 2*log_lik5

bic_m1<- length(pram_m1)*log(n) - 2*log_lik1
bic_m2<- length(pram_m2)*log(n) - 2*log_lik2
bic_m3<- length(pram_m3)*log(n) - 2*log_lik3
bic_m4<- length(pram_m4)*log(n) - 2*log_lik4
```

```
bic_m5<- length(pram_m5)*log(n) - 2*log_lik5

aic_bic<-data.frame(Model=paste("Model",1:5, sep="-"),
            AIC=c(aic_m1,aic_m2,aic_m3,aic_m4,aic_m5),
            BIC=c(bic_m1,bic_m2,bic_m3,bic_m4,bic_m5))

aic_bic
```

Output

	Model	AIC	BIC
1	Model-1	144.6958	161.2123
2	Model-2	1106.1682	1116.0781
3	Model-3	142.6798	155.8931
4	Model-4	143.2452	156.4584
5	Model-5	146.6141	166.4339

2.5.4. Task 2.5

```
e1<- data$y - X1 %*% pram_m1
e2<- data$y - X2 %*% pram_m2
e3<- data$y - X3 %*% pram_m3
e4<- data$y - X4 %*% pram_m4
e5<- data$y - X5 %*% pram_m5

# Error distribution
par(mfrow=c(3,2))
hist(e1,main="Distribution of residual in model-1")
hist(e2,main="Distribution of residual in model-2")
hist(e3,main="Distribution of residual in model-3")
hist(e4,main="Distribution of residual in model-4")
hist(e5,main="Distribution of residual in model-5")

# qq-plot for all model residual

par(mfrow=c(3,2))
```

Output

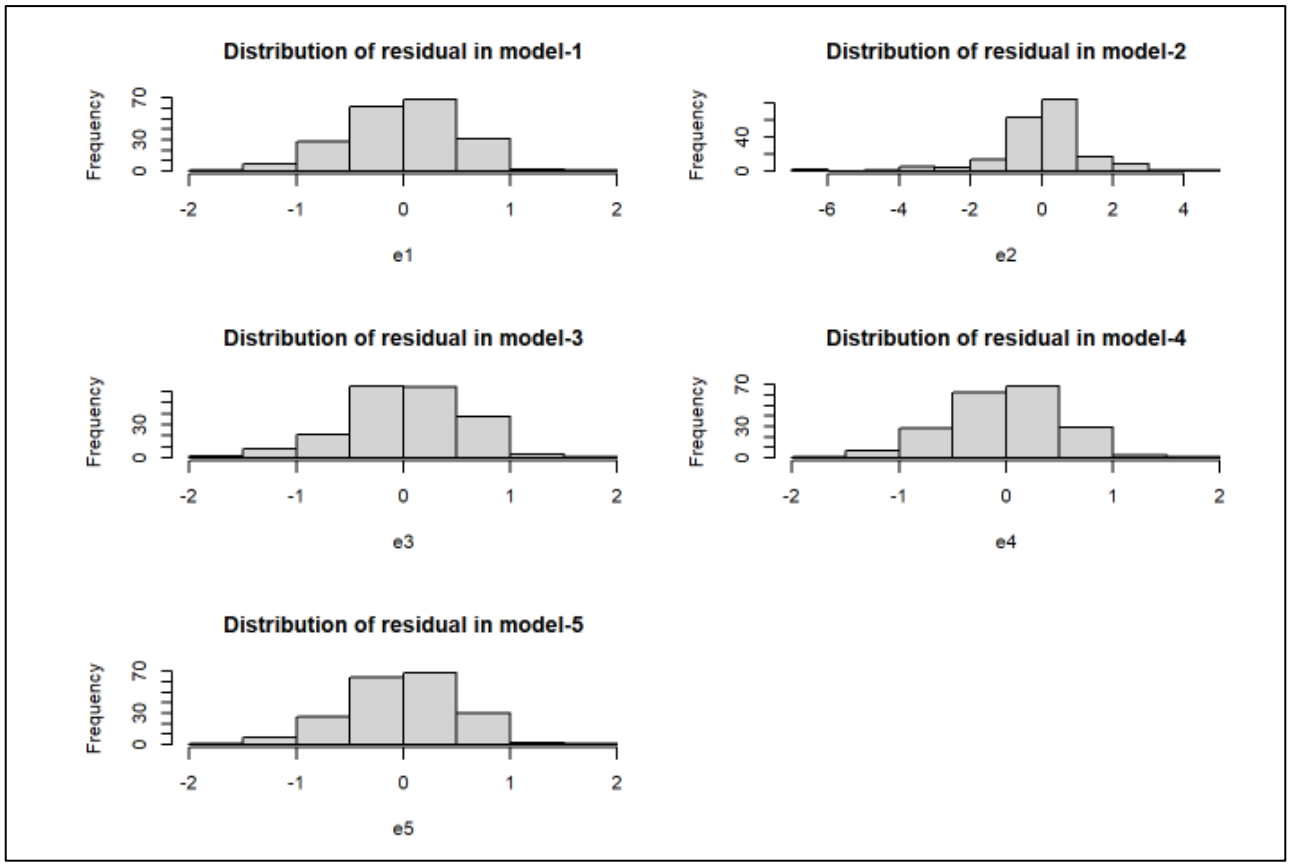

Image 6. Residual Plot

```
qqnorm(e1, pch = 1, frame = FALSE,main="QQ-plot for Model-1")
qqline(e1, col = "steelblue", lwd = 2)

qqnorm(e2, pch = 1, frame = FALSE)
qqline(e2, col = "steelblue", lwd = 2)

qqnorm(e3, pch = 1, frame = FALSE)
qqline(e3, col = "steelblue", lwd = 2)

qqnorm(e4, pch = 1, frame = FALSE)
qqline(e4, col = "steelblue", lwd = 2)

qqnorm(e5, pch = 1, frame = FALSE)
qqline(e5, col = "steelblue", lwd = 2)
```

Output

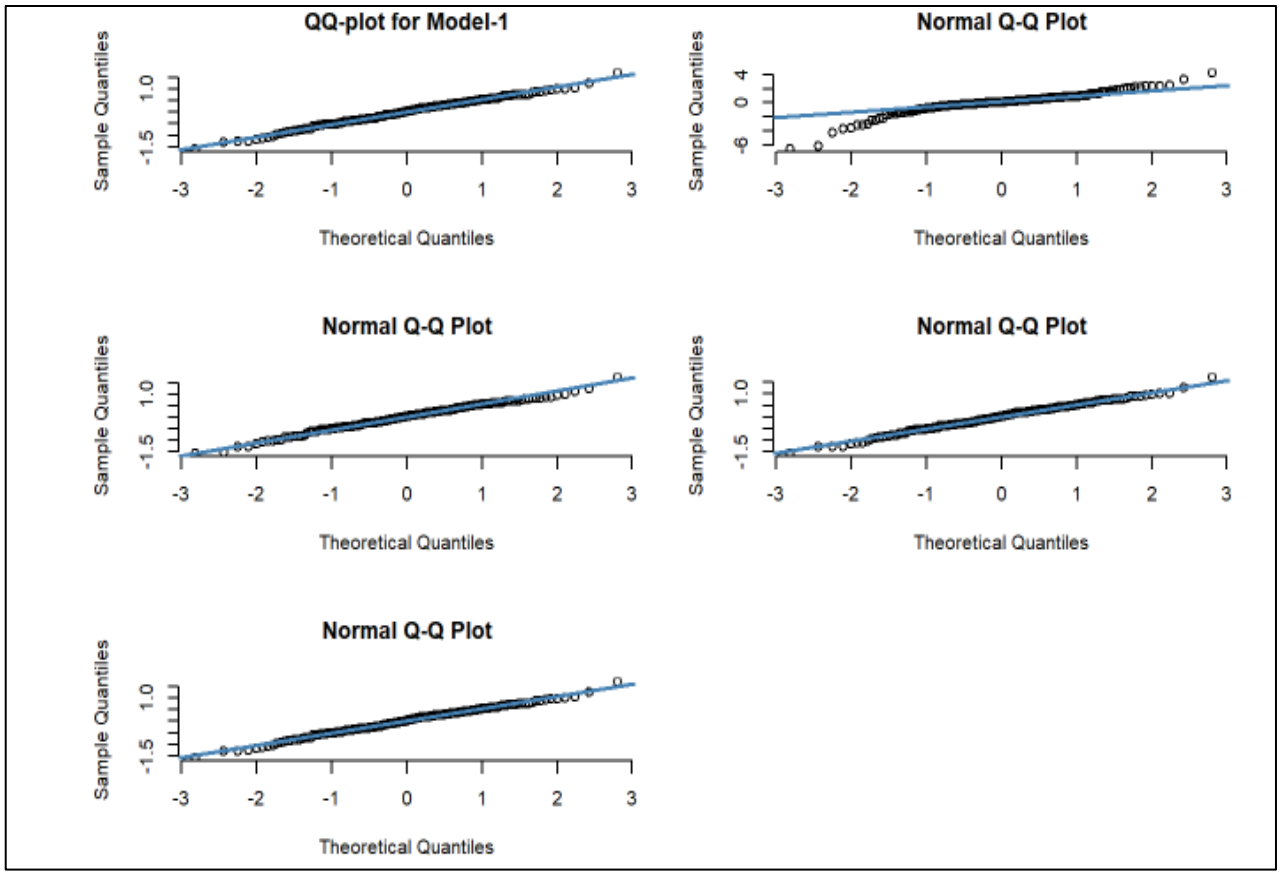

Image 7. Normal Q-Q Plot

2.5.5. Task 2.6

```
aic_m1<- 2*length(pram_m1) - 2*log_lik1
aic_m2<- 2*length(pram_m2) - 2*log_lik2
aic_m3<- 2*length(pram_m3) - 2*log_lik3
aic_m4<- 2*length(pram_m4) - 2*log_lik4
aic_m5<- 2*length(pram_m5) - 2*log_lik5

bic_m1<- length(pram_m1)*log(n) - 2*log_lik1
bic_m2<- length(pram_m2)*log(n) - 2*log_lik2
bic_m3<- length(pram_m3)*log(n) - 2*log_lik3
bic_m4<- length(pram_m4)*log(n) - 2*log_lik4
bic_m5<- length(pram_m5)*log(n) - 2*log_lik5

aic_bic<-data.frame(Model=paste("Model",1:5, sep="-"),
            AIC=c(aic_m1,aic_m2,aic_m3,aic_m4,aic_m5),
            BIC=c(bic_m1,bic_m2,bic_m3,bic_m4,bic_m5))

aic_bic
```

Output

	Model	AIC	BIC
1	Model-1	144.6958	161.2123
2	Model-2	1106.1682	1116.0781
3	Model-3	142.6798	155.8931
4	Model-4	143.2452	156.4584
5	Model-5	146.6141	166.4339

The smallest AIC and BIC values was found at "**Model-3**" and the residual of this model are more normal comparative to others model.

2.5.6. Task 2.7

```
aic_m1<- 2*length(pram_m1) - 2*log_lik1
aic_m2<- 2*length(pram_m2) - 2*log_lik2
aic_m3<- 2*length(pram_m3) - 2*log_lik3
aic_m4<- 2*length(pram_m4) - 2*log_lik4
aic_m5<- 2*length(pram_m5) - 2*log_lik5

bic_m1<- length(pram_m1)*log(n) - 2*log_lik1
bic_m2<- length(pram_m2)*log(n) - 2*log_lik2
bic_m3<- length(pram_m3)*log(n) - 2*log_lik3
bic_m4<- length(pram_m4)*log(n) - 2*log_lik4
bic_m5<- length(pram_m5)*log(n) - 2*log_lik5

aic_bic<-data.frame(Model=paste("Model",1:5, sep="-"),
            AIC=c(aic_m1,aic_m2,aic_m3,aic_m4,aic_m5),
            BIC=c(bic_m1,bic_m2,bic_m3,bic_m4,bic_m5))
aic_bic

data_m3<-data.frame(t=data$t,model3, y=data$y)
tr_sample <-sort(sample(n, 0.7*n))
tr_data<-data_m3[tr_sample ,]
ts_data<-data_m3[-tr_sample ,]

train_m3<- lm(y ~ z1+z2+z3 , data=tr_data)
test_fit <- predict(train_m3,
newdata=ts_data,interval="confidence",level = 0.95,
se.fit=TRUE)
res_df<- data.frame(ts_data,test_fit$fit,se=test_fit$se.fit)
```

```
head(res_df)
```

Output

	t	z1	z2	z3	z0	y	fit
4	0.006	-1.5566153	-0.38579917	1.167824e-08	1	0.17645157	0.38411325
5	0.008	0.7661962	10.71972075	1.217888e+01	1	0.01104331	0.57950516
10	0.018	5.0272662	107.98768988	3.082394e-05	1	2.16322436	1.77375208
11	0.02	1.2761762	0.03484228	5.295107e+00	1	0.40687177	0.51293715
12	0.022	-2.2902154	-38.32577955	1.232205e-01	1	-0.40473476	-0.02719052
15	0.028	-6.0209836	-68.98091730	1.063624e+02	1	-0.29157064	-0.73812263

	lwr	upr	se
4	0.2695988	0.4986277	0.05790692
5	0.4794482	0.6795621	0.05059617
10	1.5395622	2.0079419	0.11842359
11	0.4035619	0.6223124	0.05530816
12	-0.1586631	0.1042821	0.06648219
15	-1.0048752	-0.4713701	0.13488970

```
plot(res_df$t,res_df$fit, type="l",col="blue")
arrows(x0=res_df$t, y0=res_df$fit-res_df$se, x1=res_df$t,
y1=res_df$fit+res_df$se, code=3, angle=90, length=0.1)
points(res_df$t,res_df$y, pch=16)
lines(res_df$t,res_df$upr, type="l", col="red",lty=3)
lines(res_df$t,res_df$lwr, type="l", col="red",lty=3)
```

Output

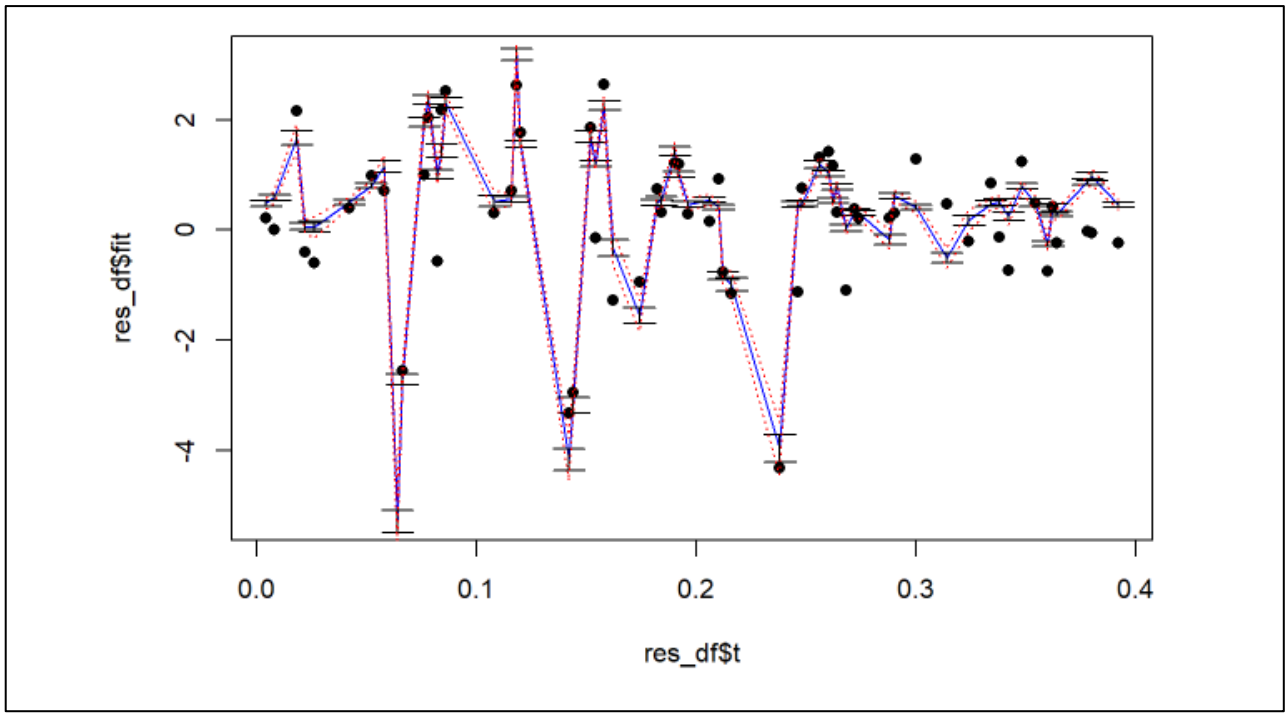

Image 8. Prediction Plot among the signals

2.6. Task 3

Parameters with largest absolute values in your least squares estimation.

```
two_pram<-sort(abs(pram_m3),decreasing =TRUE)[1:2]

two_pram
```

Output

```
    theta_bias             theta_1

[1] 0.44829955           0.03810926
```

```
prior<- function () {
  theta_bias<-runif(1, 0, 1)
  theta_1<-runif(1, -0.5, 0.5)
  c(theta_bias,theta_1)
 }
```

```
ABC_acceptance <- function(par,y,X, theta){

         parm_sel<-order(abs(theta),decreasing =TRUE)[1:2]
         theta[parm_sel]<-par
         y_sim<- as.matrix(X) %*% as.matrix(theta)
         diffmean <- abs(mean(y) - mean(y_sim))
      if(diffmean < 0.05) return(TRUE) else return(FALSE)
}


run_MCMC_ABC <- function(y,X, theta,iterations){
    chain = array(dim = c(iterations+1,2))
        chain[1,] = prior()
    for (i in 1:iterations){
        # proposalfunction
        proposal = prior()

        if(ABC_acceptance(proposal,y,X,theta)){
            chain[i+1,] = proposal
        }else{
            chain[i+1,] = chain[i,]
        }
    }
    return(mcmc(chain))
}


posterior <- run_MCMC_ABC(data$y,model3, pram_m3,10000)

head(posterior)
```

Output

```
Markov Chain Monte Carlo (MCMC) output:

Start = 1

End = 7

Thinning interval = 1

           [,1]           [,2]

[1,]      0.8456448      -0.3477730
[2,]      0.8456448      -0.3477730
[3,]      0.8456448      -0.3477730
[4,]      0.8456448      -0.3477730
[5,]      0.8456448      -0.3477730
[6,]      0.8456448      -0.3477730
[7,]      0.3955653      0.2273343
```

```
summary(posterior)
```

Output

```
Iterations = 1:10001
Thinning interval = 1
Number of chains = 1
Sample size per chain = 10001

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

            Mean    SD       Naive SE Time-series    SE
[1,]      0.44977   0.03181   0.0003181            0.00141
[2,]      0.02067   0.28013   0.0028012            0.01239

2. Quantiles for each variable:

           2.5%      25%        50%       75%      97.5%
var1      0.3971    0.4239    0.44957   0.4744    0.5041
var2      -0.4526   -0.2251   0.03463   0.2509    0.4737
```

```
plot(posterior)
```

Output

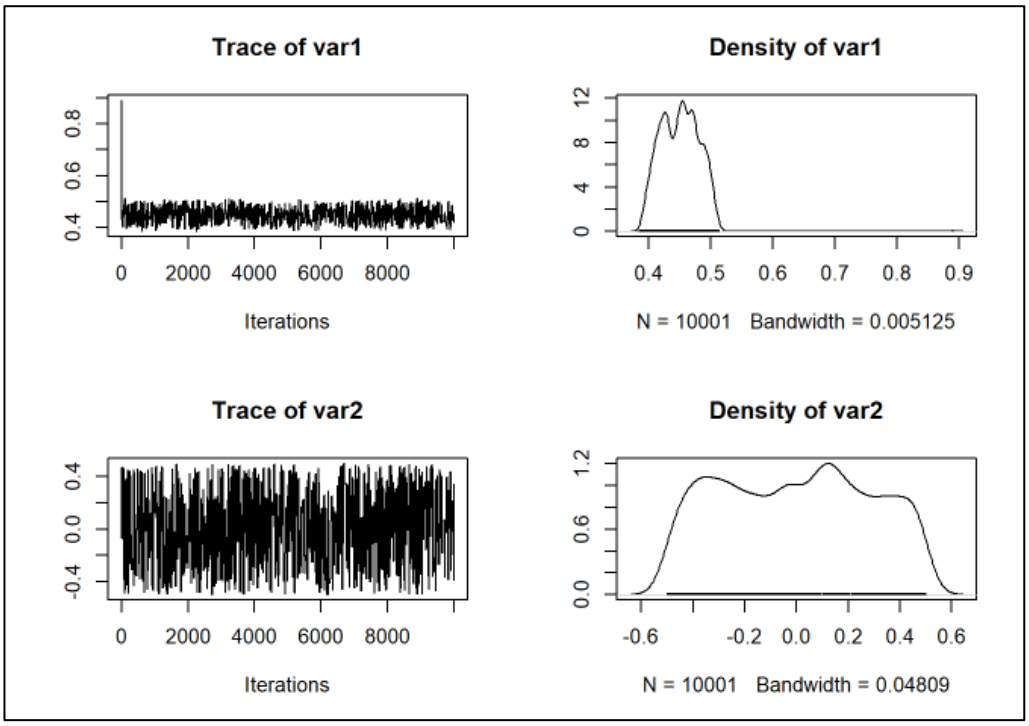

Image 9. Posterior Distributions between the signals

Here are four other important metrics - **AIC**, **AICc**, **BIC** and **Mallows Cp** - that are commonly used for model evaluation and selection. These are an unbiased estimate of the model prediction error MSE. The lower these metrics, the better the model.

1. **AIC** stands for (*Akaike's Information Criteria*), a metric developed by the Japanese Statistician, Hirotugu Akaike, 1970. The basic idea of AIC is to penalize the inclusion of additional variables to a model. It adds a penalty that increases the error when including additional terms. The lower the AIC, the better the model.

2. **AICc** is a version of AIC corrected for small sample sizes.

3. **BIC** (or *Bayesian information criteria*) is a variant of AIC with a stronger penalty for including additional variables to the model.

4. **Mallows Cp**: A variant of AIC developed by Colin Mallows.

Generally, the most commonly used metrics, for measuring regression model quality and for comparing models, are: Adjusted R2, AIC, BIC and Cp.

The most important metrics are the Adjusted R-square, RMSE, AIC and the BIC. These metrics are also used as the basis of model comparison and optimal model selection.

When comparing models fitted by maximum likelihood to the same data, the smaller the AIC or BIC, the better the fit.

The theory of AIC requires that the log-likelihood has been maximized: whereas AIC can be computed for models not fitted by maximum likelihood, their AIC values should not be compared.

Examples of models not 'fitted to the same data' are where the response is transformed (accelerated- life models are fitted to log-times) and where contingency tables have been used to summarize data.

These are generic functions (with S4 generics defined in package stats4): however methods should be defined for the log-likelihood function logLik rather than these functions: the action of their default methods is to call logLik on all the supplied objects and assemble the results. Note that in several common cases logLik does not return the value at the MLE.

The log-likelihood and hence the AIC/BIC is only defined up to an additive constant. Different constants have conventionally been used for different purposes and so extract AIC and AIC may give different values (and do for models of class "lm"). Particular care is needed when comparing fits of different classes (with, for example, a comparison of a Poisson and gamma GLM being meaningless since one has a discrete response, the other continuous).

BIC is defined as AIC(object, ..., k = log(nobs(object))). This needs the number of observations to be known: the default method looks first for a "nobs" attribute on the return value from the logLik method, then tries the nobs generic, and if neither succeed returns BIC as NA.

Value

If just one object is provided, a numeric value with the corresponding AIC (or BIC, or ..., depending on k).

If multiple objects are provided, a dataframe with rows corresponding to the objects and columns representing the number of parameters in the model (df) and the AIC or BIC.

Note that, these regression metrics are all internal measures, that is they have been computed on the same data that was used to build the regression model. They tell you how well the model fits to the data in hand, called training data set.

In general, we do not really care how well the method works on the training data. Rather, we are interested in the accuracy of the predictions that we obtain when we apply our method to previously unseen test data.

However, the test data is not always available making the test error very difficult to estimate. In this situation, methods such as **cross-validation** and **bootstrap** are applied for estimating the test error (or the prediction error rate) using training data.

2.7. References

1. Alkan, A., Koklukaya, E., & Subasi, A. 2005. Automatic seizure detection in EEG using logistic regression and artificial neural network. *Journal of Neuroscience Methods*, *148*(2), 167-176.

2. Wei, H.L., Billings, S.A. and Liu, J.J., 2010. Time-varying parametric modelling and time- dependent spectral characterisation with applications to EEG signals using multiwavelets. *International Journal of Modelling, Identification and Control*, *9*(3), pp.215-224.

3. Subasi, A. and Ercelebi, E., 2005. Classification of EEG signals using neural network and logistic regression. *Computer methods and programs in biomedicine*, *78*(2), pp.87-99.

4. Ghergulescu, I. and Muntean, C.H., 2016. ToTCompute: a novel EEG-based TimeOnTask threshold computation mechanism for engagement modelling and monitoring. *International Journal of Artificial Intelligence in Education*, *26*(3), pp.821-854.

5. Vigário, R.N., 1997. Extraction of ocular artefacts from EEG using independent component analysis. *Electroencephalography and clinical neurophysiology*, *103*(3), pp.395-404.

6. Bingham, N., Fry, J. and London, S., 2021. *Regression - Linear Models in Statistics* | N. H. Bingham | Springer. [online] Springer.com. Available at: https://www.springer.com/gp/book/9781848829688

7. *The Best Books on Linear Regression*. Datasciencetexts.com. (2021). Retrieved 11 March 2021, from https://datasciencetexts.com/subjects/linear_regression.html.

8. *Nonlinear Regression Analysis - an overview | ScienceDirect Topics*. Sciencedirect.com. (2021). Retrieved 11 March 2021, from https://www.sciencedirect.com/topics/medicine-and-dentistry/nonlinear-regression-analysis.

9. *Polynomial Regression in R Programming - GeeksforGeeks*. GeeksforGeeks. (2021). Retrieved 11 March 2021, from https://www.geeksforgeeks.org/polynomial-regression-in-r-programming/.

3. Appendix

Please see the link below to access the R codes used for this study.
<https://github.com/Manojsibi/Manoj_SM-Projects/blob/master/Statistics%20Course%20Work/Stastics%20R%20code.R>