Modelling a Time Series Dataset

Buse Felamur  
*Faculty of Engineering, Environment and Computing*  
*Coventry University*Coventry, England  
[felamurb@uni.coventry.ac.uk](mailto:felamurb@uni.coventry.ac.uk)

1. Introduction

In this project, aim is to model the given dataset. Given dataset is made from two synthetic electroencephalogram (EEG) time-series data. Both of these EEG channels are measured from two different parts of the brain. Purpose is to model and find out the degree of potential nonlinear synergy between the EEG channels.

First signal is referred as X and second signal is Y. To make it easier for our modelling processes we say X is input and Y is output. Since Y is output, that makes it dependent and makes X independent. This implies channel Y can be found by channel X. Both X and Y consists of 250 rows of data. Additionally, signal Y already contains additive noise of an unknown amount.

First step of our process is doing an initial data analysis. After that correct model structure found by doing AIC. Next step is estimating parameters of found model. After finalizing the model and parameters, we can move on to model’s evaluation. This includes parameter covariance matrix, uncertainty p.d.f and 95% confidence intervals. Our last step is validation our model by using train-test split validation to see if identified model is good or not. Code is written with R. Version used for R is version 3.6.1. Integrated development environment (IDE) that was chosen for R is rstudio.

1. Exploratory Data Analysis

Purpose of initial data analysis is finding more about the data before we attempt to find the true model. This makes everything easier. Graphs are color coded to make it simple to understand. Very first thing we do, after loading the data, is plotting time series plots of both input and output. It can be seen at Figure 1.

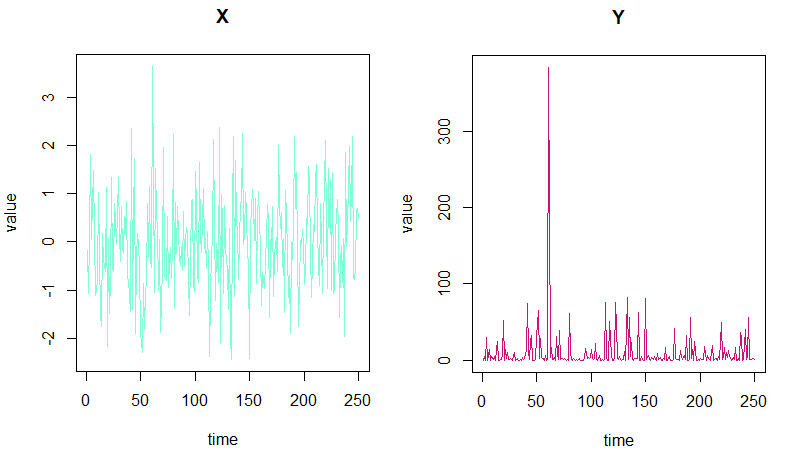


Figure 1 - Time series plots of X and Y

Then we plot both signals against each other. It can be seen that this graph resembles a graph of x square. If we limit Y signal to between zero to hundred, it becomes more visible to us that shape is very similar to x square. This indicates our model has a high possibility of having x square as one of the terms. It can been at Figure 2.

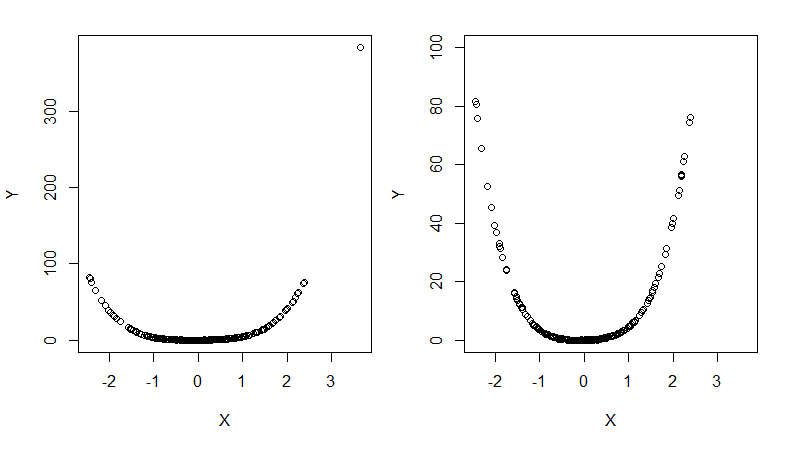


Figure 2 - X and Y against each other and zoomed version

We then plot distribution for both signals. Histogram of X shows that it is left tailed and slightly skewed. This might say it can be a normal distribution. On the other hand, histogram of Y is right tailed and looks like a non-normal distribution, probably an exponential distribution. Histograms of X and Y can be seen at Figure 3.

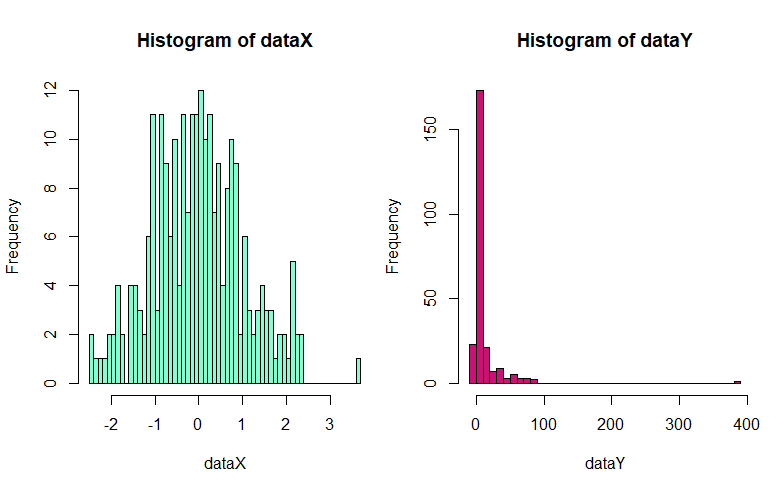


Figure 3 - Histograms of X and Y

QQ plots were used to get a better idea of finding out the right distribution. Points in the QQ plot of X lie on the straight diagonal line. There are some deviations from that line but it still indicates that data is normally distributed. If we compare this to QQ plot of Y, it can be seen that points do not follow the straight line but another shape. This indicates the non-normality. Look at Figure 4 for QQ graphs.

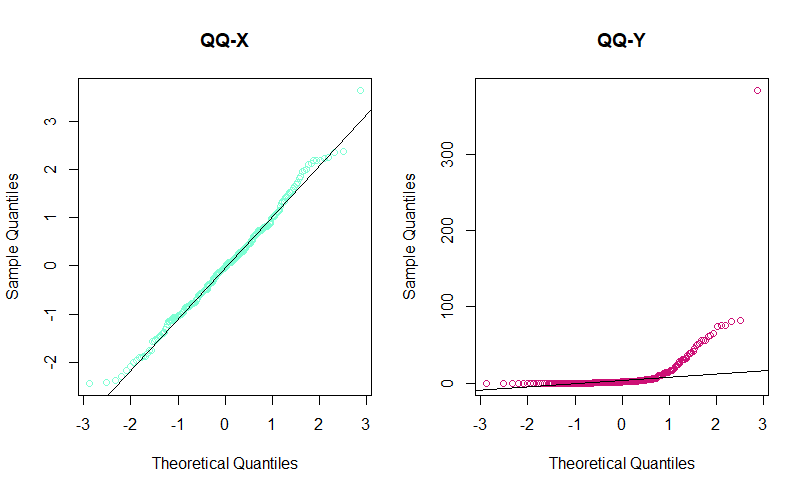


Figure 4 - QQ plots of X and Y

Another way to confirm this is Shapiro-Wilk Test of Normality. If the p-value is lower than or equal to .05, it rejects the null hypothesis that sample comes from a normal distribution. P-value for X and Y are 0.2854 and < 0. 00000000000000022. Value of X is bigger than .05 so this confirms X comes from a normal distribution. On the other hand, Y is smaller so this mean it can not be normally distributed.

Now we look at correlation analysis. To do this, Pearson’s product moment correlation is used. Null hypothesis is true correlation is equal to 0. In this method, if p-value is lower than 5% then correlation between signal X and signal Y is significant. Our p-value is 0.0004763. Null hypothesis is rejected. It is lower so there is a correlation between X and Y.

Additionally, boxplots for both signal X and signal Y were given. Boxplots are good at showing distribution of the data according to the five number summary. These five number summaries are minimum, first quartile, median, third quartile and of course maximum. These graphs can tell the outliers and what exactly their values are. It can also help to see if data is symmetrical and if and how exactly the data is skewed. Boxplots for both signals, X any Y can be seen at Figure 5.

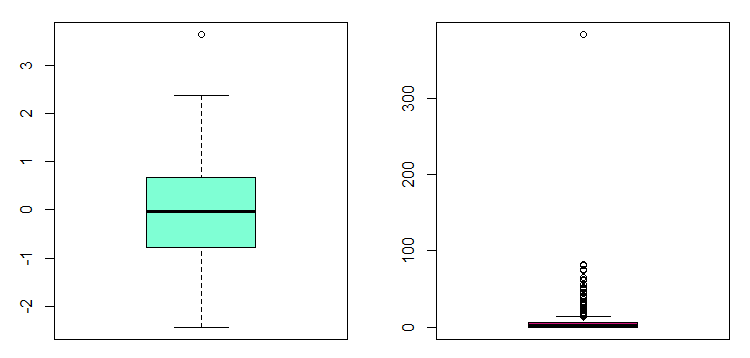


Figure 5 - Boxplots of X and Y

Final step of data analysis is fitting a linear model to our data and find out its performance. The linear model we picked is y=w\_0 + w\_1\*x. This model was picked because it is linear and it is better to use a simpler model since we have no idea what is the actual model for these signal dataset. To calculate the performance, we need to find mean squared error (MSE) and residual analysis. Mean squared error measures the average of the squares of the errors. That error is the difference between the estimator and what is estimated. In the formula, y is true value of y, y hat is the prediction and n is the amount of data points.

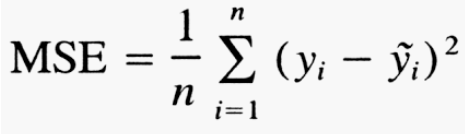


Figure 6 - MSE Formula

To come to that step, first we created a matrix full of ones with the size of our X. These matrix were bound to our X. This happened because of the linear model we have decided to use for this part. We used least squares method to find the parameters. We later calculated predictions and then the residuals. Fitted model graph can be seen at Figure 7 and histogram of residuals is at Figure 8.

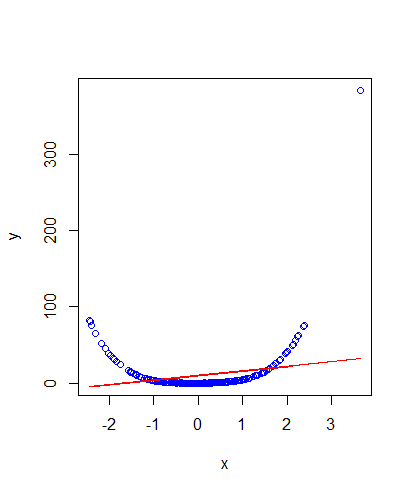


Figure 7 - Fitted model graph

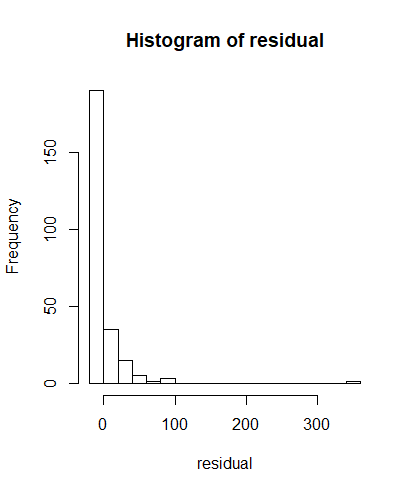


Figure 8 - Histogram of residuals

MSE of the fitted model is 787.5948, SSE is 196898 and AIC is 2408.219. This shows that the linear model we used are not the best fit for our data. We must try many more different models than this. Code used in data analysis part can be seen in the following figure.

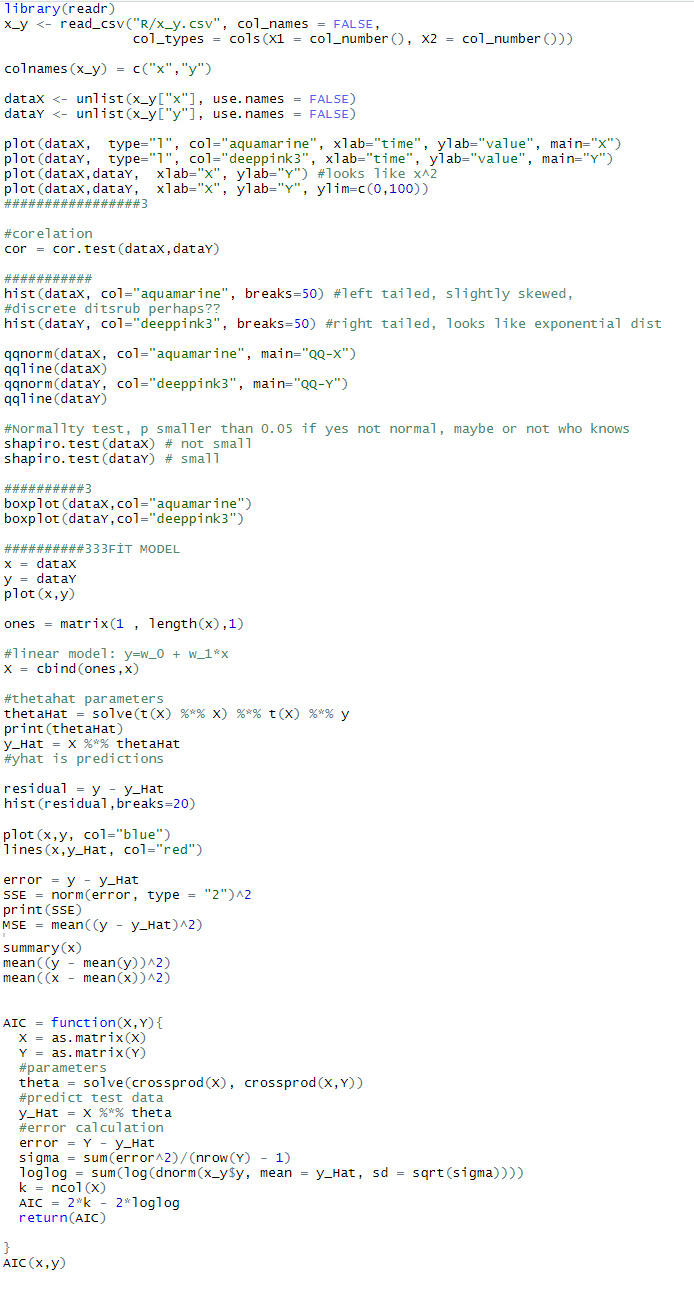


Figure 9 - Code of data analysis part

1. Model Selection

In this part of the report, we concentrated on identifying the model. There are multiple ways to do it but only one way was chosen. For example, dataset can be split into two (training and test) and apply the forward subset selection method and picking the model based on MSE. AIC or BIC are another possible options. Both of the methods are very similar to each other. For example, they do not require dataset to be split into two. Their respective were used to find out AIC or BIC. Another similarity is that with AIC or BIC all of the potential combinations must be considered.

For this project, AIC (Akaike Information Criterion) was used to identify the nonlinear regression model. AIC finds the quality of all of the available models. Based on that quality we can use it for model selection. Lowest AIC value means it is the best.

* 1. AIC and Model

With AIC method, we do not have to split the data into train and test dataset. Full data can be used and was used to calculate this. We have applied AIC’s own formula to calculate the AIC values. AIC formula can be seen at Figure 10.

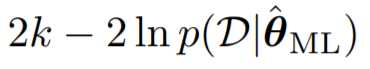


Figure 10 - AIC Formula

In the formula, k is the number of estimated parameters in the model.

According to the coursework brief our model can not have more than 3 terms. AIC requires all possible combinations to be explored but since true model can not be longer than 3 terms, there is no point in calculating any model that includes more than 3 term.

Initial action is putting ones (x0) and x (x1) to x^5 (x5) in a dataframe. At first, we applied AIC formula to y equals to only one of the x terms including x0. For example, y = x^4 or y = x^3. We calculated the lowest AIC value and we found out that it is 1104.983. Next step is same but this time with two terms instead. As an example, y = x^4 + y^5 can be given. Lowest AIC value end up being 392.4286. Last one is same again but with three possible terms such as y = x + x^3 + x^5. Lowest AIC value is for this one is -432.3259.

Lowest AIC belongs to the model with three terms. Now we need to find out which three terms are these. Later we had discovered that our terms are x1, x2 and x4. This makes our model equal to y = x + x^2 + x^4. Code can be seen at following figure.

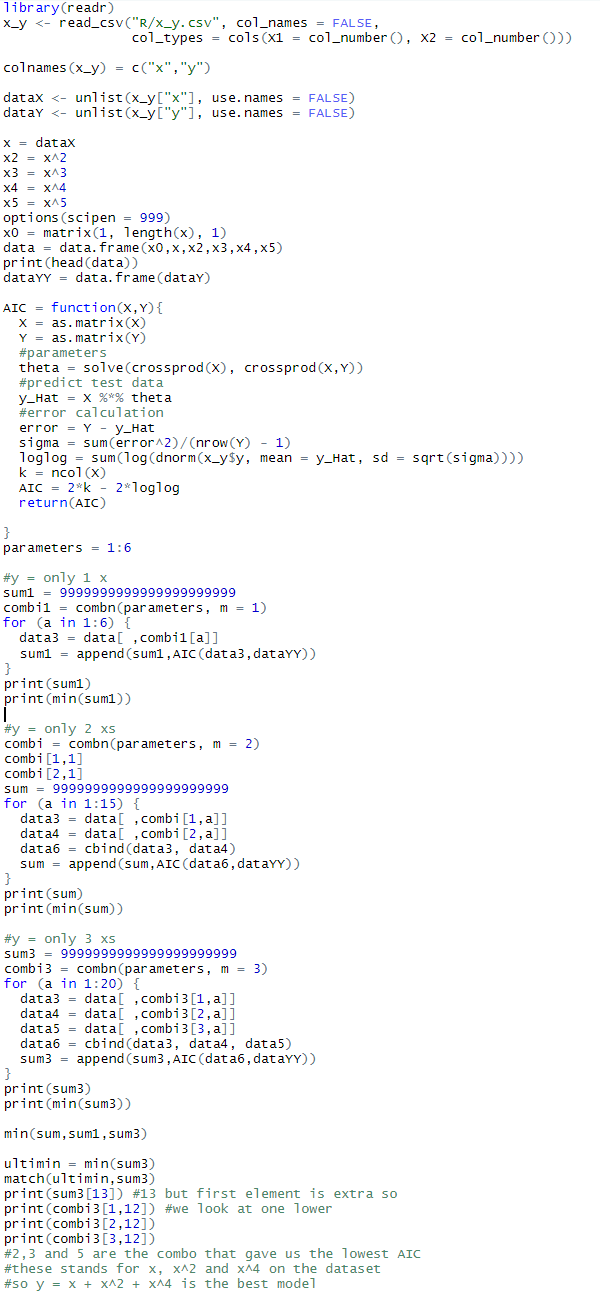


Figure 11 - AIC Code

1. Model Evaluation

In this part of the report, first we will talk about how parameters for the picked model were estimated. After that, we will move on to parameter covariance matrix, parameter uncertainty p.d.f. Next step is computing our model’s output, prediction and finding confidence intervals. Last step is validating the data with splitting the dataset into two (training dataset and testing dataset) and check the predictions.

* 1. Parameters

This part we will talk about the parameter. After finding the true model, we need to find the parameters. Best way to find parameters for our model is using the least square method. Least squares are used in regression analysis. Purpose is approximating the solution in a set of equations which number of unknowns exceeds number of equations. Way to do it is minimizing the sum of the squares of the residuals which are found as a result of every one of the equations.

After calculating the result, we found out that our parameters are 0.499, 2.003 and lastly 2.000. This makes our model equal to y = 0.499x + 2.003x^2 + 2.000x^4 + error.

* 1. Covariance and P.D.F

Now that we have the true model structure and the parameters, parameter covariance matrix and parameter uncertainty p.d.f can be estimated.

Covariance matrix also known as variance matrix, dispersion matrix and variance-covariance matrix are a square matrix which gives us the covariance between every couple of the elements of a given random vector. The reason we calculate parameter covariance matrix is because it is needed to evaluate parameter correlation. Covariance we found on first column is 4.055091e-05, 4.233711e-06 and -1.124719e-06. Second column is made from 4.233711e-06, 3.642431e-05 and -4.427568e-06. Last column is made from -1.124719e-06, -4.427568e-06 and 7.836886e-07. Now we can calculate the parameter uncertainty p.d.f.. There are three parameters total so they need to be grouped into two for them to be calculated and then be plotted. Three different combinations were made. They were plotted using contour function in r. There are total of three contour graphs and they can be seen in the following figures.

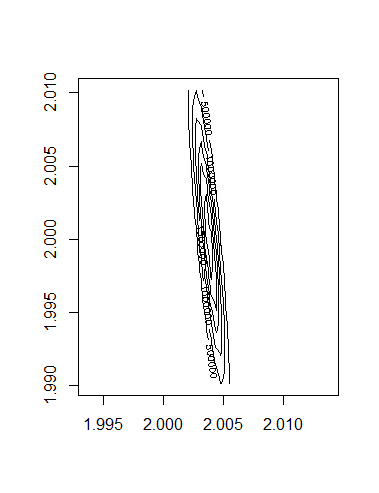


Figure 12 - Contour for x2 and x4

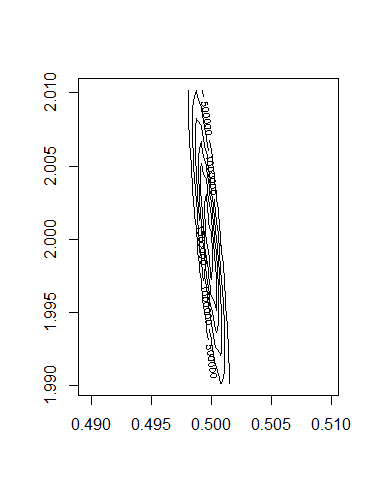


Figure 13 - Contour for x1 and x4

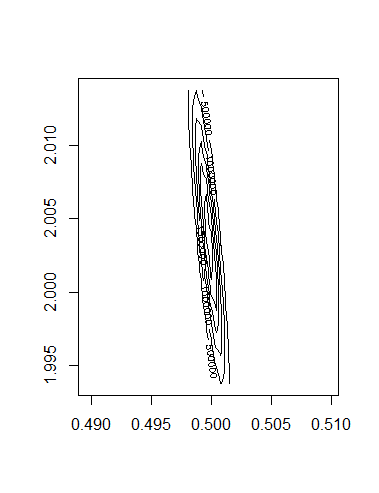


Figure 14 - Contour for x1 and x2

* 1. Output and Confidence

In this part of the report as the title indicates, our main focus is about model’s output and 95% confidence intervals. There needs to be a training set so first dataset divided into two (training set and testing set). Training set with 80% of data and testing set with 20% of data. We calculated the parameters, prediction and error. The way we done is very similar to the part in initial data analysis where we fitted a linear model to our data. As results, MSE of training set end up being 0.01069. Furthermore, MSE of testing set is 0.0076. Now let’s move onto confidence intervals. Formula for confidence interval can be seen at Figure 15. As the picture shows, x hat is sample mean, lower case sigma symbol is the population standard deviation and z is the value for desired confidence level. In our case our desired confidence level is 95%.

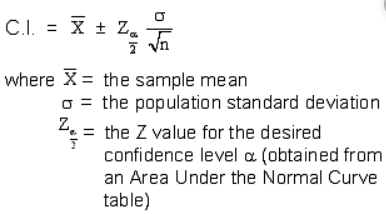


Figure 15 - CI Formula

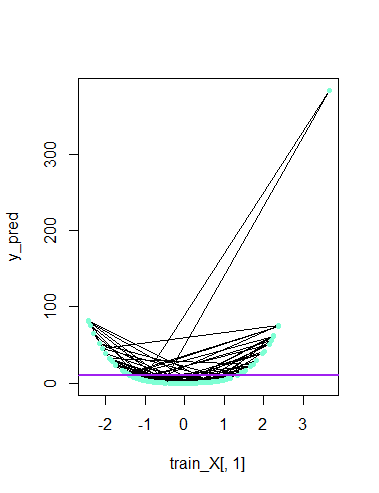


Figure 16 - CI Plot

After calculating the confidence interval with 95%, we plotted them to see it on a graph. Graph can be seen on the Figure 16. Purple line is mean values. Black lines are plot between x values and prediction values. Aquamarine dots are error bars. This graph is satisfying. To see the r code for everything in part 4 so far, you can look at the next figure.

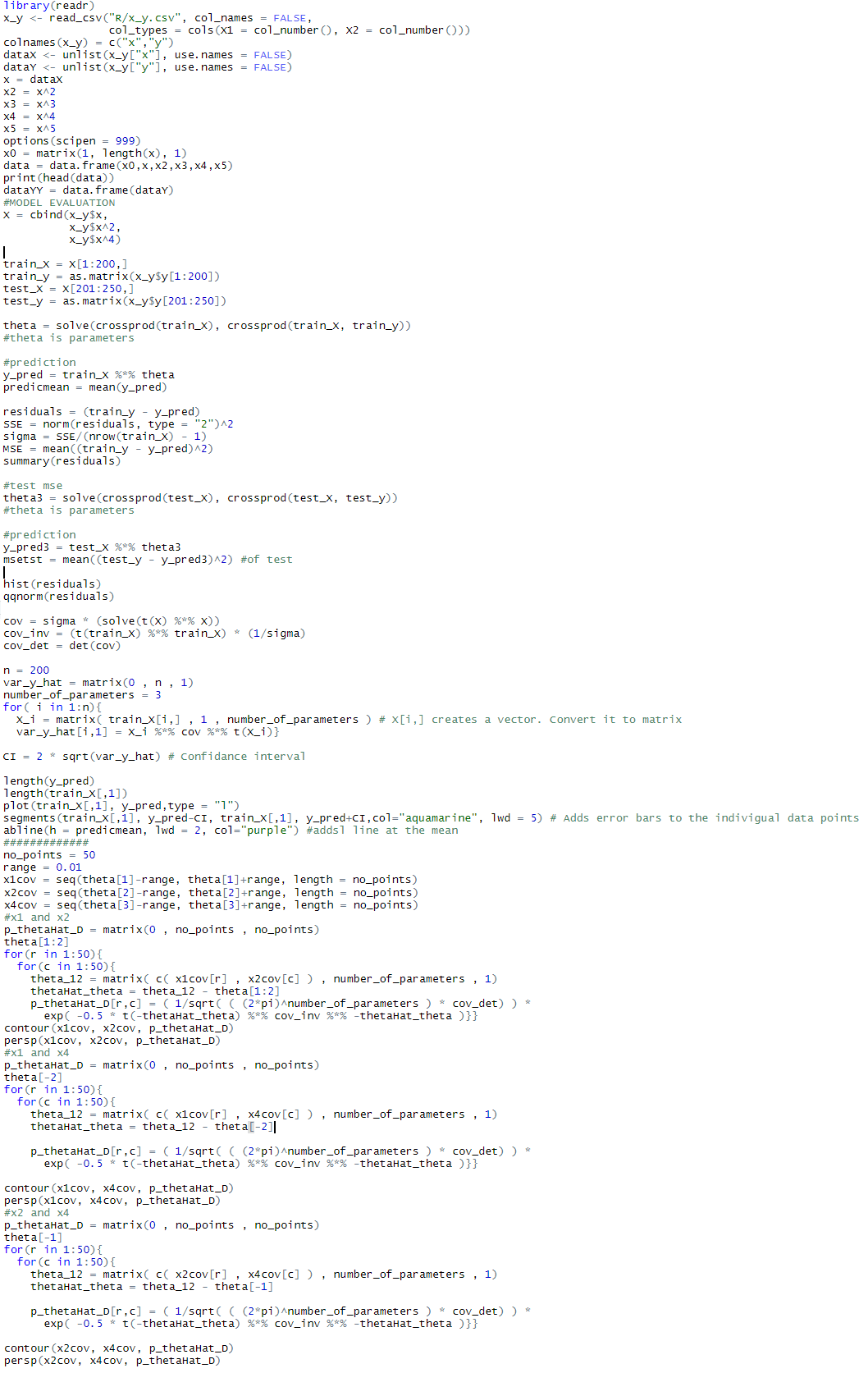


Figure 17 - Code for 4. section except validation part

* 1. Model Validation

Our chosen model needs to be validated to see if it provides a good prediction and output on the testing dataset. Dataset must be divided into two different sets to do this part. Training dataset and of course testing dataset. Dataset spitted as %70 for training data and %30 for testing data unlike the last split which is %80 for training data and %20 for testing data. Data is fitted into our true models for both datasets. Testing set is sufficient but it is done so it can be compared if it is needed. We have used the same parameters from older parts of the code for our new calculations. Predictions and errors were calculated for both test and training set. MSE and AIC values were also calculated using the same formulas as before so it can also be compared.

For AIC we do not have to divide dataset into two but project requires us to use a testing dataset. Easy way to do it is just take first 175 rows from the csv file and remove rest of the rows. We just deleted extra rows and saved rest of them to another csv file. This csv file was used to calculate AIC value.

It is calculated that MSE score for testing data is 0.0827. Additionally, MSE value for training data is 0.0109. However, AIC value for testing data is -287.1717. Boxplots were created to these MSE values can be compared to the older ones. MSE boxplots can be seen in the following figures. Code for validation part can be seen at Figure 20.

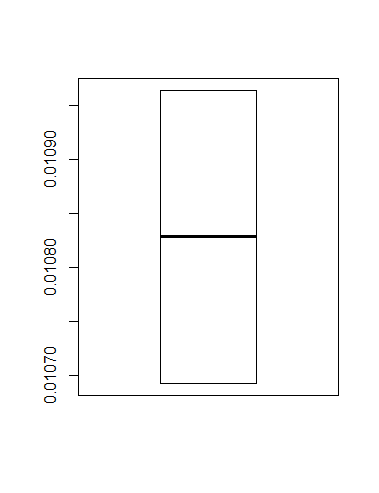


Figure 18 - Trainset Boxplot

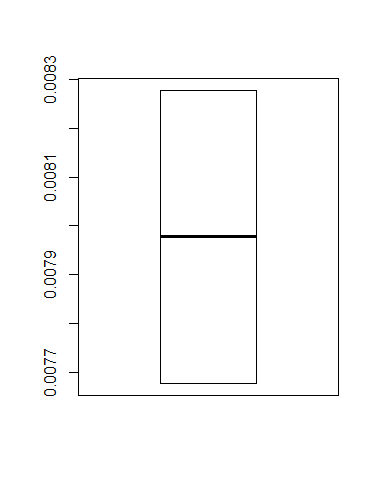


Figure 19 - Testset Boxplot

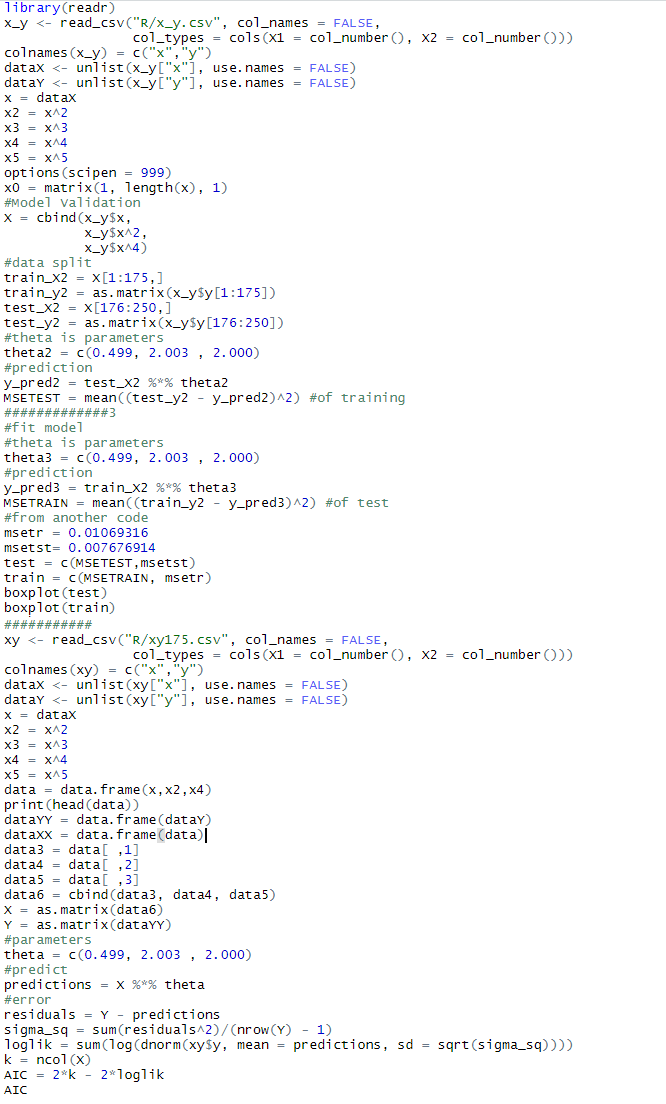


Figure 20 - Code for validation

1. Conclusion

Goal of this project is to find out which polynomial model structure is the true model, calculate parameters of the model with using the training data set, apply these to the found model and in the end use our findings to estimate the response signal.

We were given a csv file for two synthetic electroencephalogram (EEG) signals. X being the first column and Y being the second column. We took signal X as input and signal Y as output to make our job easier. Only other thing apart from these we knew about the model is that it can not have more than three terms, signal Y comes with its noise already added in and basic model of this model is just a generic nonlinear polynomial regression model.

First of all, before actually doing all of these mentioned steps, we had to do a initial exploratory data analysis. If we have an idea what kind of a dataset we have, we can find out the true model easier. This initial exploratory data analysis shows us that signal X might be normally distributed. On the other hand, signal Y is not normally distributed. Signal Y might be exponentially distributed. We found out this from examining various graphs and doing multiple different tests. Another thing we discovered from our initial data analysis is that possible model has a high chance of including a x square as one of it’s terms.

Second step is actually finding the model. There might be multiple ways to do it but, in this project, AIC method were picked. AIC value for all possible combinations were explored. Best and lowest AIC value were given for the model with three terms but which three terms? Our next move was calculating these three terms. Our terms end up being x, x^2 and x^4. Best model is y = x + x^2 + x^4.

Still our project was not done yet. This model needed parameters too. Parameters were found using the least square method. Results were 0.499, 2.003 and 2.000. Final model is y = 0.499x + 2.003x^2 + 2.000x^4 + error.

After deciding on right model structure and parameters of this model structure, we calculated parameter covariance matrix. This was used to plot parameter uncertainty p.d.f. In the end there were three plots because we have three parameters. All combinations of these three parameters were plotted.

Moving onto next step we had to calculate training data’s prediction as known as training data’s output. We had to split data into two since it was only asked of us to do training set. We had found training set’s MSE as 0.01069. It is a good MSE since it is very low. After that 95% confidence intervals were calculated with error bars. It shows us an acceptable and good result.

Finally, last step of this project is validating the model. In this step train-test split validation model was used. Splitting proportions were %70 to %30 this time. When we were calculating the training dataset we had used %80 to %20 split instead. MSE of training data in both of the splits were very similar. However, testing sets MSE seem to got worse somehow. Both of them are still under zero but in the 70/30 split MSE is slightly higher than the 80/20 split for testing datasets

Overall conclusion is it is believed that model structure and the parameters of the said model structure which were found for the electroencephalogram (EEG) time-series data are correct. The reason testing data’s MSE is spoiled the validation is probably because of how little data we had when we spitted the dataset. Training data has more data and MSE of both training sets are very similar to each other.

1. Appendix

There are 4 different r scripts. There are comments included in code but it is advised to check the report for more detailed information and compare following code with the screenshots of the code just in case there was a mistake with copy/pasting.

Data analysis;

library(readr)

x\_y <- read\_csv("R/x\_y.csv", col\_names = FALSE,

col\_types = cols(X1 = col\_number(), X2 = col\_number()))

colnames(x\_y) = c("x","y")

dataX <- unlist(x\_y["x"], use.names = FALSE)

dataY <- unlist(x\_y["y"], use.names = FALSE)

#Plots

plot(dataX, type="l", col="aquamarine", xlab="time", ylab="value", main="X")

plot(dataY, type="l", col="deeppink3", xlab="time", ylab="value", main="Y")

plot(dataX,dataY, xlab="X", ylab="Y") #looks like x^2

plot(dataX,dataY, xlab="X", ylab="Y", ylim=c(0,100))

#################3

#corelation

cor = cor.test(dataX,dataY)

###########

hist(dataX, col="aquamarine", breaks=50) #left tailed, slightly skewed,

#discrete ditsrub perhaps??

hist(dataY, col="deeppink3", breaks=50) #right tailed, looks like exponential dist

qqnorm(dataX, col="aquamarine", main="QQ-X")

qqline(dataX)

qqnorm(dataY, col="deeppink3", main="QQ-Y")

qqline(dataY)

#Normallty test, p smaller than 0.05 if yes not normal, maybe or not

shapiro.test(dataX) # not small

shapiro.test(dataY) # small

##########3Boxplots

boxplot(dataX,col="aquamarine")

boxplot(dataY,col="deeppink3")

##########333FİT MODEL

x = dataX

y = dataY

plot(x,y)

ones = matrix(1 , length(x),1)

#linear model: y=w\_0 + w\_1\*x

X = cbind(ones,x)

#thetahat parameters

thetaHat = solve(t(X) %\*% X) %\*% t(X) %\*% y

print(thetaHat)

y\_Hat = X %\*% thetaHat

#yhat is predictions

residual = y - y\_Hat

hist(residual,breaks=20)

plot(x,y, col="blue")

lines(x,y\_Hat, col="red")

error = y - y\_Hat

SSE = norm(error, type = "2")^2

print(SSE)

MSE = mean((y - y\_Hat)^2)

summary(x)

mean((y - mean(y))^2)

mean((x - mean(x))^2)

AIC = function(X,Y){

X = as.matrix(X)

Y = as.matrix(Y)

#parameters

theta = solve(crossprod(X), crossprod(X,Y))

#predict test data

y\_Hat = X %\*% theta

#error calculation

error = Y - y\_Hat

sigma = sum(error^2)/(nrow(Y) - 1)

loglog = sum(log(dnorm(x\_y$y, mean = y\_Hat, sd = sqrt(sigma))))

k = ncol(X)

AIC = 2\*k - 2\*loglog

return(AIC)

}

AIC(x,y)

Model selection;

library(readr)

x\_y <- read\_csv("R/x\_y.csv", col\_names = FALSE,

col\_types = cols(X1 = col\_number(), X2 = col\_number()))

colnames(x\_y) = c("x","y")

dataX <- unlist(x\_y["x"], use.names = FALSE)

dataY <- unlist(x\_y["y"], use.names = FALSE)

x = dataX

x2 = x^2

x3 = x^3

x4 = x^4

x5 = x^5

options(scipen = 999)

x0 = matrix(1, length(x), 1)

data = data.frame(x0,x,x2,x3,x4,x5)

print(head(data))

dataYY = data.frame(dataY)

AIC = function(X,Y){

X = as.matrix(X)

Y = as.matrix(Y)

#parameters

theta = solve(crossprod(X), crossprod(X,Y))

#predict test data

y\_Hat = X %\*% theta

#error calculation

error = Y - y\_Hat

sigma = sum(error^2)/(nrow(Y) - 1)

loglog = sum(log(dnorm(x\_y$y, mean = y\_Hat, sd = sqrt(sigma))))

k = ncol(X)

AIC = 2\*k - 2\*loglog

return(AIC)

}

parameters = 1:6

#y = only 1 x

sum1 = 9999999999999999999999

combi1 = combn(parameters, m = 1)

for (a in 1:6) {

data3 = data[ ,combi1[a]]

sum1 = append(sum1,AIC(data3,dataYY))

}

print(sum1)

print(min(sum1))

#y = only 2 xs

combi = combn(parameters, m = 2)

combi[1,1]

combi[2,1]

sum = 9999999999999999999999

for (a in 1:15) {

data3 = data[ ,combi[1,a]]

data4 = data[ ,combi[2,a]]

data6 = cbind(data3, data4)

sum = append(sum,AIC(data6,dataYY))

}

print(sum)

print(min(sum))

#y = only 3 xs

sum3 = 9999999999999999999999

combi3 = combn(parameters, m = 3)

for (a in 1:20) {

data3 = data[ ,combi3[1,a]]

data4 = data[ ,combi3[2,a]]

data5 = data[ ,combi3[3,a]]

data6 = cbind(data3, data4, data5)

sum3 = append(sum3,AIC(data6,dataYY))

}

print(sum3)

print(min(sum3))

min(sum,sum1,sum3)

ultimin = min(sum3)

match(ultimin,sum3)

print(sum3[13]) #13 but first element is extra so

print(combi3[1,12]) #we look at one lower

print(combi3[2,12])

print(combi3[3,12])

#2,3 and 5 are the combo that gave us the lowest AIC

#these stands for x, x^2 and x^4 on the dataset

#so y = x + x^2 + x^4 is the best model

Model evaluation;

library(readr)

x\_y <- read\_csv("R/x\_y.csv", col\_names = FALSE,

col\_types = cols(X1 = col\_number(), X2 = col\_number()))

colnames(x\_y) = c("x","y")

dataX <- unlist(x\_y["x"], use.names = FALSE)

dataY <- unlist(x\_y["y"], use.names = FALSE)

x = dataX

x2 = x^2

x3 = x^3

x4 = x^4

x5 = x^5

options(scipen = 999)

x0 = matrix(1, length(x), 1)

data = data.frame(x0,x,x2,x3,x4,x5)

print(head(data))

dataYY = data.frame(dataY)

#MODEL EVALUATION

X = cbind(x\_y$x,

x\_y$x^2,

x\_y$x^4)

train\_X = X[1:200,]

train\_y = as.matrix(x\_y$y[1:200])

test\_X = X[201:250,]

test\_y = as.matrix(x\_y$y[201:250])

theta = solve(crossprod(train\_X), crossprod(train\_X, train\_y))

#theta is parameters

#prediction

y\_pred = train\_X %\*% theta

predicmean = mean(y\_pred)

residuals = (train\_y - y\_pred)

SSE = norm(residuals, type = "2")^2

sigma = SSE/(nrow(train\_X) - 1)

MSE = mean((train\_y - y\_pred)^2)

summary(residuals)

#test mse

theta3 = solve(crossprod(test\_X), crossprod(test\_X, test\_y))

#theta is parameters

#prediction

y\_pred3 = test\_X %\*% theta3

msetst = mean((test\_y - y\_pred3)^2) #of test

hist(residuals)

qqnorm(residuals)

cov = sigma \* (solve(t(X) %\*% X))

cov\_inv = (t(train\_X) %\*% train\_X) \* (1/sigma)

cov\_det = det(cov)

n = 200

var\_y\_hat = matrix(0 , n , 1)

number\_of\_parameters = 3

for( i in 1:n){

X\_i = matrix( train\_X[i,] , 1 , number\_of\_parameters ) # X[i,] creates a vector. Convert it to matrix

var\_y\_hat[i,1] = X\_i %\*% cov %\*% t(X\_i)}

CI = 2 \* sqrt(var\_y\_hat) # Confidance interval

length(y\_pred)

length(train\_X[,1])

plot(train\_X[,1], y\_pred,type = "l")

segments(train\_X[,1], y\_pred-CI, train\_X[,1], y\_pred+CI,col="aquamarine", lwd = 5) # Adds error bars to the indivigual data points

abline(h = predicmean, lwd = 2, col="purple") #addsl line at the mean

#############

no\_points = 50

range = 0.01

x1cov = seq(theta[1]-range, theta[1]+range, length = no\_points)

x2cov = seq(theta[2]-range, theta[2]+range, length = no\_points)

x4cov = seq(theta[3]-range, theta[3]+range, length = no\_points)

#x1 and x2

p\_thetaHat\_D = matrix(0 , no\_points , no\_points)

theta[1:2]

for(r in 1:50){

for(c in 1:50){

theta\_12 = matrix( c( x1cov[r] , x2cov[c] ) , number\_of\_parameters , 1)

thetaHat\_theta = theta\_12 - theta[1:2]

p\_thetaHat\_D[r,c] = ( 1/sqrt( ( (2\*pi)^number\_of\_parameters ) \* cov\_det) ) \*

exp( -0.5 \* t(-thetaHat\_theta) %\*% cov\_inv %\*% -thetaHat\_theta )}}

contour(x1cov, x2cov, p\_thetaHat\_D)

persp(x1cov, x2cov, p\_thetaHat\_D)

#x1 and x4

p\_thetaHat\_D = matrix(0 , no\_points , no\_points)

theta[-2]

for(r in 1:50){

for(c in 1:50){

theta\_12 = matrix( c( x1cov[r] , x4cov[c] ) , number\_of\_parameters , 1)

thetaHat\_theta = theta\_12 - theta[-2]

p\_thetaHat\_D[r,c] = ( 1/sqrt( ( (2\*pi)^number\_of\_parameters ) \* cov\_det) ) \*

exp( -0.5 \* t(-thetaHat\_theta) %\*% cov\_inv %\*% -thetaHat\_theta )}}

contour(x1cov, x4cov, p\_thetaHat\_D)

persp(x1cov, x4cov, p\_thetaHat\_D)

#x2 and x4

p\_thetaHat\_D = matrix(0 , no\_points , no\_points)

theta[-1]

for(r in 1:50){

for(c in 1:50){

theta\_12 = matrix( c( x2cov[r] , x4cov[c] ) , number\_of\_parameters , 1)

thetaHat\_theta = theta\_12 - theta[-1]

p\_thetaHat\_D[r,c] = ( 1/sqrt( ( (2\*pi)^number\_of\_parameters ) \* cov\_det) ) \*

exp( -0.5 \* t(-thetaHat\_theta) %\*% cov\_inv %\*% -thetaHat\_theta )}}

contour(x2cov, x4cov, p\_thetaHat\_D)

persp(x2cov, x4cov, p\_thetaHat\_D)

Model Validation;

library(readr)

x\_y <- read\_csv("R/x\_y.csv", col\_names = FALSE,

col\_types = cols(X1 = col\_number(), X2 = col\_number()))

colnames(x\_y) = c("x","y")

dataX <- unlist(x\_y["x"], use.names = FALSE)

dataY <- unlist(x\_y["y"], use.names = FALSE)

x = dataX

x2 = x^2

x3 = x^3

x4 = x^4

x5 = x^5

options(scipen = 999)

x0 = matrix(1, length(x), 1)

#Model Validation

X = cbind(x\_y$x,

x\_y$x^2,

x\_y$x^4)

#data split

train\_X2 = X[1:175,]

train\_y2 = as.matrix(x\_y$y[1:175])

test\_X2 = X[176:250,]

test\_y2 = as.matrix(x\_y$y[176:250])

#theta is parameters

theta2 = c(0.499, 2.003 , 2.000)

#prediction

y\_pred2 = test\_X2 %\*% theta2

MSETEST = mean((test\_y2 - y\_pred2)^2) #of training

#############3

#fit model

#theta is parameters

theta3 = c(0.499, 2.003 , 2.000)

#prediction

y\_pred3 = train\_X2 %\*% theta3

MSETRAIN = mean((train\_y2 - y\_pred3)^2) #of test

#from another code

msetr = 0.01069316

msetst= 0.007676914

test = c(MSETEST,msetst)

train = c(MSETRAIN, msetr)

boxplot(test)

boxplot(train)

###########175.csv only has first 175 rows, explained in report

xy <- read\_csv("R/xy175.csv", col\_names = FALSE,

col\_types = cols(X1 = col\_number(), X2 = col\_number()))

colnames(xy) = c("x","y")

dataX <- unlist(xy["x"], use.names = FALSE)

dataY <- unlist(xy["y"], use.names = FALSE)

x = dataX

x2 = x^2

x3 = x^3

x4 = x^4

x5 = x^5

data = data.frame(x,x2,x4)

print(head(data))

dataYY = data.frame(dataY)

dataXX = data.frame(data)

data3 = data[ ,1]

data4 = data[ ,2]

data5 = data[ ,3]

data6 = cbind(data3, data4, data5)

X = as.matrix(data6)

Y = as.matrix(dataYY)

#parameters

theta = c(0.499, 2.003 , 2.000)

#predict

predictions = X %\*% theta

#error

residuals = Y - predictions

sigma\_sq = sum(residuals^2)/(nrow(Y) - 1)

loglik = sum(log(dnorm(xy$y, mean = predictions, sd = sqrt(sigma\_sq))))

k = ncol(X)

AIC = 2\*k - 2\*loglik

AIC