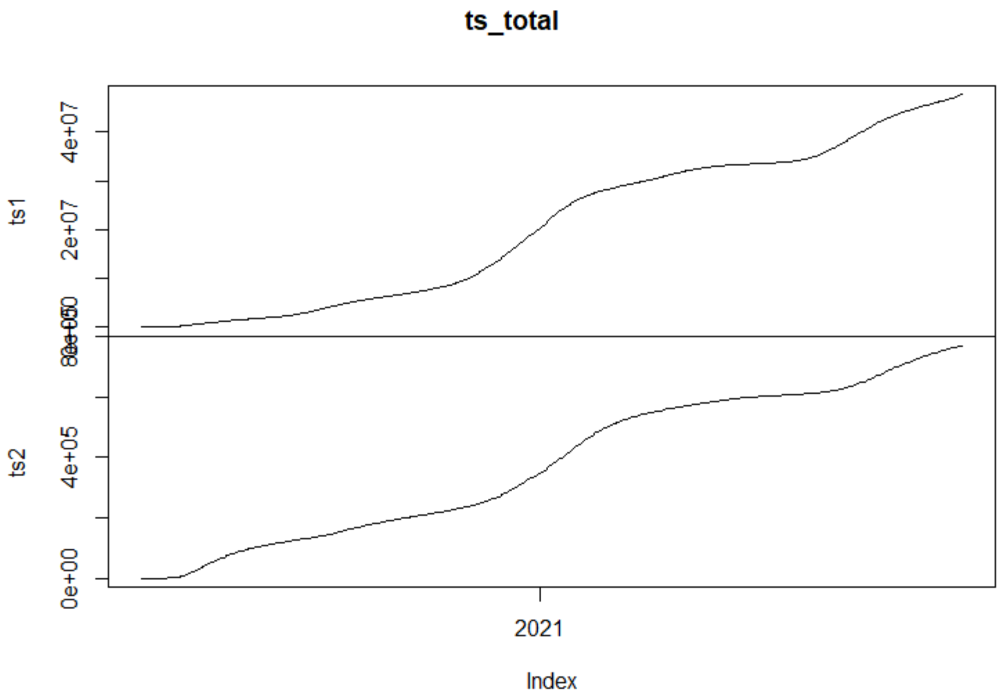
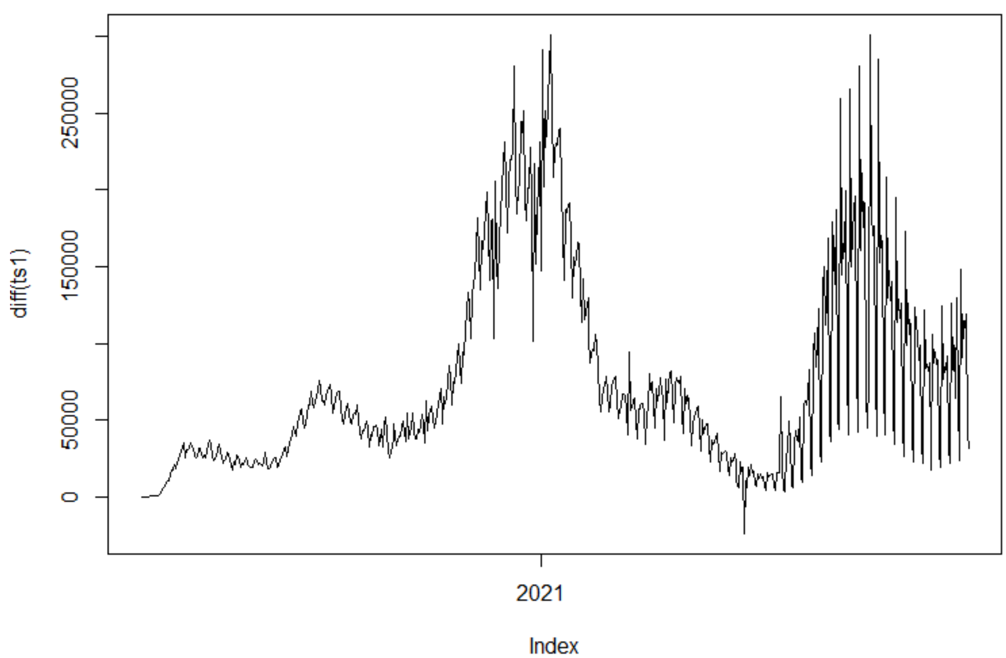
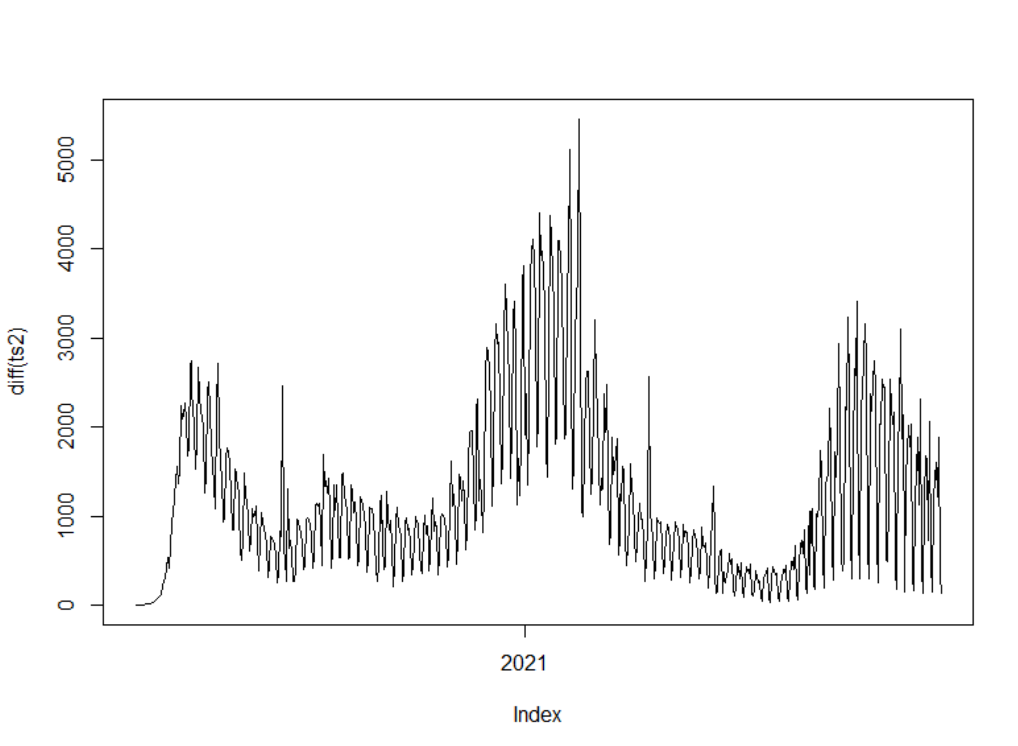
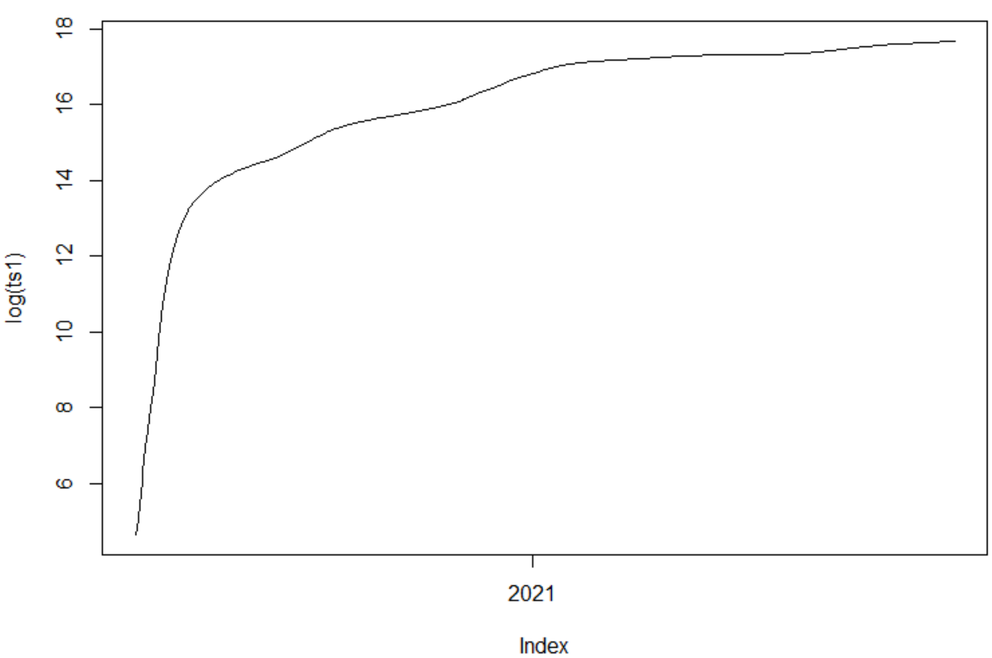
Q1

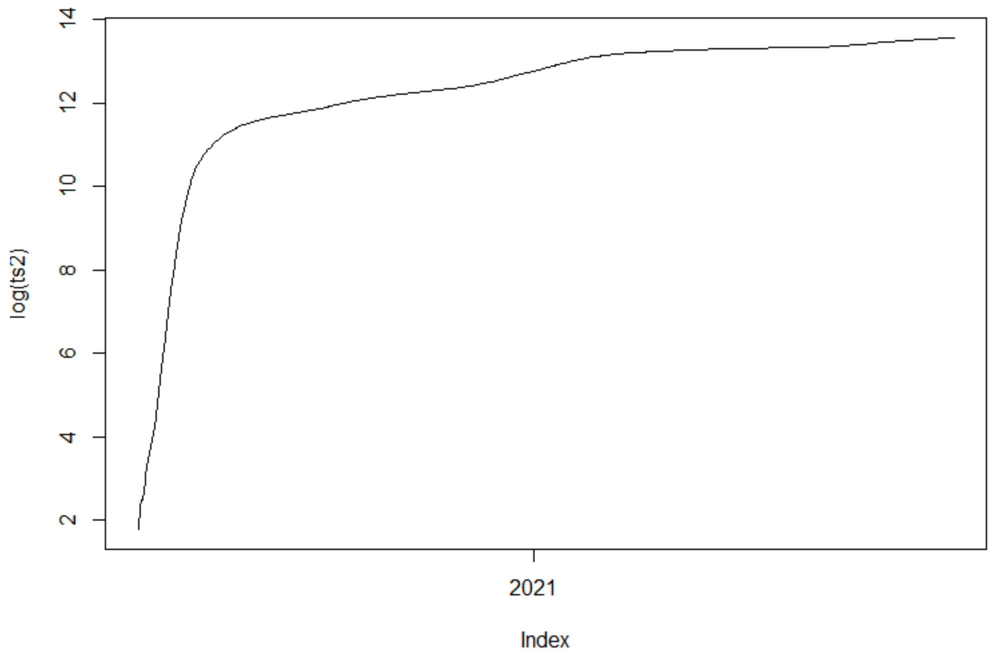
```
#IE 500 SMLE HW 4 Q1

cases_usa<- read.csv("https://raw.githubusercontent.com/nytimes/covid-19-data/master/us.csv")
library(ggplot2)
library(forecast)
library(fpp)

daily_cases <- cases_usa$cases
daily_deaths<- cases_usa$deaths
cases_usa$date <- as.Date(cases_usa$date)
dates_new<- cases_usa[cases_usa$date > "2020-03-01"&
                        cases_usa$date < "2021-11-22",]
dt <- seq(as.Date("2020-03-02"),as.Date("2021-11-21"),by = "days")
ts1 <- zoo(dates_new$cases,dt)
ts2 <- zoo(dates_new$deaths,dt)
print(ts1)
print(ts2)
ts_total <-merge(ts1,ts2)
plot(ts_total)
plot(diff(ts1))
plot(diff(ts2))
plot(log(ts1))
plot(log(ts2))
plot(diff(log(ts1)))
plot(diff(log(ts2)))
auto.arima(diff(log(ts1)))
auto.arima(diff(log(ts2)))


arima_cases1 <- predict(arima(dates_new$cases, order = c(4,1,2)), n.ahead = 1)
arima_cases1
arima_cases5 <- predict(arima(dates_new$cases, order = c(4,1,2)), n.ahead = 5)
arima_cases5
arima_cases10 <- predict(arima(dates_new$cases, order = c(4,1,2)), n.ahead = 10)
arima_cases10

arima_deaths1 <- predict(arima(dates_new$deaths, order = c(4,1,2)), n.ahead = 1)
arima_deaths1
arima_deaths5 <- predict(arima(dates_new$deaths, order = c(4,1,2)), n.ahead = 5)
arima_deaths5
arima_deaths10 <- predict(arima(dates_new$deaths, order = c(4,1,2)), n.ahead = 10)
arima_deaths10
library(Metrics)
rmse(dates_new[630,2],arima_cases1$pred)
rmse(dates_new[625:630,2],arima_cases5$pred[1:5])
rmse(dates_new[620:630,2],arima_cases10$pred[1:10])
rmse(dates_new[630,3],arima_deaths1$pred)
rmse(dates_new[625:630,3],arima_deaths5$pred[1:5])
rmse(dates_new[620:630,3],arima_deaths10$pred[1:10])
```

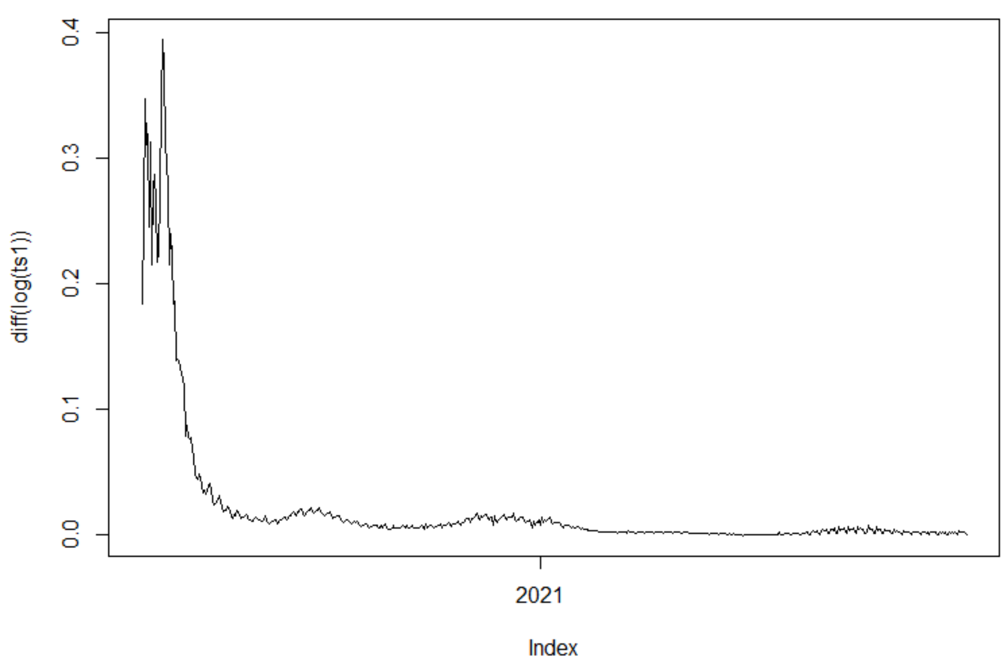

ts_total
ts1
ts2
0e+00
4e+05
8e+05
2e+07
4e+07
2021
Index

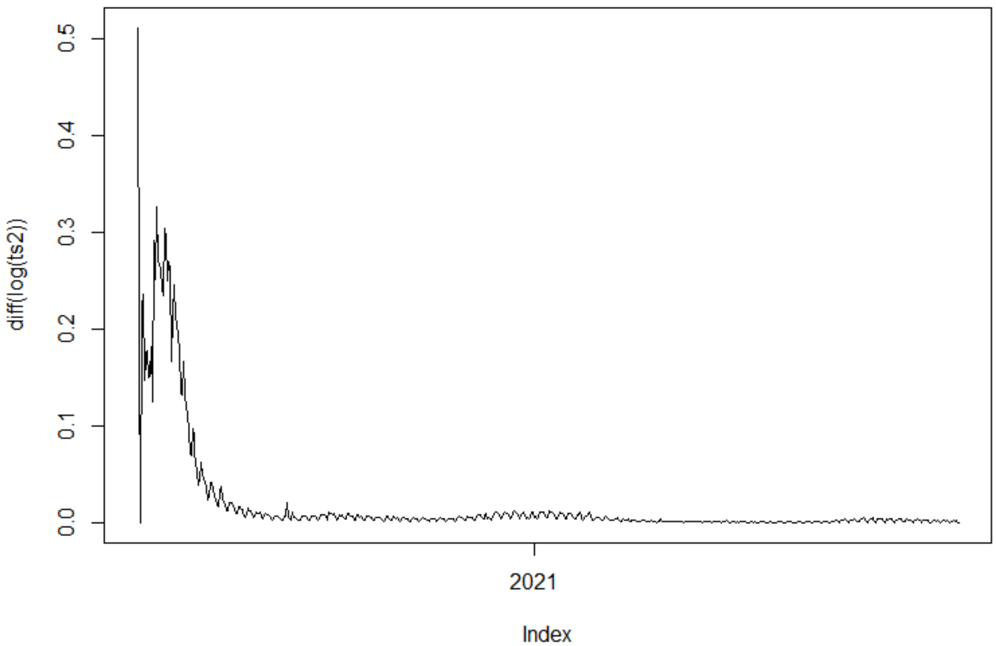

diff(ts1)
0
50000
150000
250000
2021
Index


diff(ts2)
0
1000
2000
3000
4000
5000
2021
Index


log(ts1)
6
8
10
12
14
16
18
2021
Index


log(ts2)
2
4
6
8
10
12
14
2021
Index


diff(log(ts1))
0.0
0.1
0.2
0.3
0.4
2021
Index




```
> auto.arima(diff(log(ts1)))
Series: diff(log(ts1))
ARIMA(4,1,2)

Coefficients:
         ar1      ar2      ar3      ar4     ma1     ma2
      -1.292  -0.6145  -0.2086  -0.2565  1.4158  0.4981
s.e.   0.365   0.3383   0.0774   0.0561  0.3728  0.3894

sigma^2 estimated as 9.192e-05:  log likelihood=2029.52
AIC=-4045.04   AICc=-4044.86   BIC=-4013.94
> auto.arima(diff(log(ts2)))
Series: diff(log(ts2))
ARIMA(4,1,2)

Coefficients:
          ar1      ar2     ar3     ar4      ma1      ma2
      -0.1596  -0.3101  0.2815  0.5929  -0.2862  -0.4259
s.e.   0.1839   0.1553  0.1596  0.1277   0.1874   0.1527

sigma^2 estimated as 0.0002972:  log likelihood=1659.56
AIC=-3305.13   AICc=-3304.95   BIC=-3274.03
```

```
> arima_cases1 <- predict(arima(dates_new$cases, order = c(4,1,2)), n.ahead = 1)
> arima_cases1
$pred
Time Series:
Start = 631
End = 631
Frequency = 1
[1] 47772508

$se
Time Series:
Start = 631
End = 631
Frequency = 1
[1] 28852.23

> arima_cases5 <- predict(arima(dates_new$cases, order = c(4,1,2)), n.ahead = 5)
> arima_cases5
$pred
Time Series:
Start = 631
End = 635
Frequency = 1
[1] 47772508 47831731 47852516 47906867 47982608

$se
Time Series:
Start = 631
End = 635
Frequency = 1
[1]  28852.23  52004.23  80197.63 112616.55 149235.55

> arima_cases10 <- predict(arima(dates_new$cases, order = c(4,1,2)), n.ahead = 10)
> arima_cases10
$pred
Time Series:
Start = 631
End = 640
Frequency = 1
 [1] 47772508 47831731 47852516 47906867 47982608 48013031 48040020 48111022 48163783 48178793

$se
Time Series:
Start = 631
End = 640
Frequency = 1
 [1]  28852.23  52004.23  80197.63 112616.55 149235.55 187220.37 227443.77 270946.87 316142.32 362202.71
```

```
> arima_deaths1 <- predict(arima(dates_new$deaths, order = c(4,1,2)), n.ahead = 1)
> arima_deaths1
$pred
Time Series:
Start = 631
End = 631
Frequency = 1
[1] 770563.6

$se
Time Series:
Start = 631
End = 631
Frequency = 1
[1] 501.589

> arima_deaths5 <- predict(arima(dates_new$deaths, order = c(4,1,2)), n.ahead = 5)
> arima_deaths5
$pred
Time Series:
Start = 631
End = 635
Frequency = 1
[1] 770563.6 771761.4 773023.6 774056.2 775010.7

$se
Time Series:
Start = 631
End = 635
Frequency = 1
[1]  501.589 1031.887 1446.685 1759.106 2032.977

> arima_deaths10 <- predict(arima(dates_new$deaths, order = c(4,1,2)), n.ahead = 10)
> arima_deaths10
$pred
Time Series:
Start = 631
End = 640
Frequency = 1
 [1] 770563.6 771761.4 773023.6 774056.2 775010.7 775917.0 776916.3 777900.1 778922.3 779883.8

$se
Time Series:
Start = 631
End = 640
Frequency = 1
 [1]  501.589 1031.887 1446.685 1759.106 2032.977 2321.787 2637.534 2973.901 3316.826 3665.457
```

```
> library(Metrics)
> rmse(dates_new[630,2],arima_cases1$pred)
[1] 79894.45
> rmse(dates_new[625:630,2],arima_cases5$pred[1:5])
[1] 355342.7
Warning message:
In actual - predicted :
  longer object length is not a multiple of shorter object length
> rmse(dates_new[620:630,2],arima_cases10$pred[1:10])
[1] 718751.3
Warning message:
In actual - predicted :
  longer object length is not a multiple of shorter object length
> rmse(dates_new[630,3],arima_deaths1$pred)
[1] 794.6357
> rmse(dates_new[625:630,3],arima_deaths5$pred[1:5])
[1] 4996.239
Warning message:
In actual - predicted :
  longer object length is not a multiple of shorter object length
> rmse(dates_new[620:630,3],arima_deaths10$pred[1:10])
[1] 10427.16
Warning message:
In actual - predicted :
  longer object length is not a multiple of shorter object length
> |
```

The arima model is a good fit as it provides a good measure of stationary data for this case and allows for forecasting

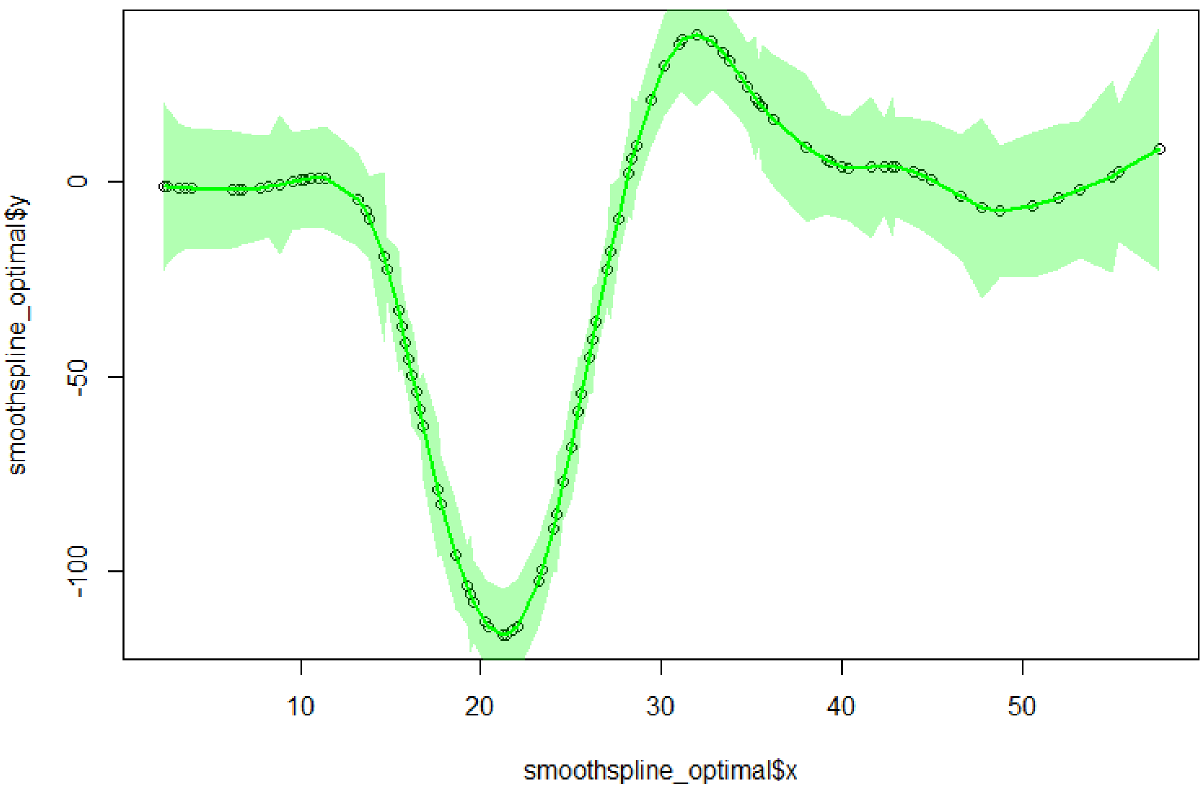
Q2

```
library(dplyr)
library(wavethresh)
library(MASS)
data(mcycle)


# Finding best degree of freedom
smoothspline_fit <- smooth.spline(mcycle$times, mcycle$accel, cv=T)
smoothspline_fit

# Plotting the best spline
smoothspline_optimal  <- smooth.spline(mcycle$times, mcycle$accel, df=12.74136)
plot(smoothspline_optimal)
lines(smoothspline_optimal, lwd = 2, col = "green")


#plotting smooth splines with 5,10,15 degrees of freedoms

smoothspline_5 <- smooth.spline(mcycle$times, mcycle$accel, df=5)
smoothspline_10 <- smooth.spline(mcycle$times, mcycle$accel, df=10)
smoothspline_15 <- smooth.spline(mcycle$times, mcycle$accel, df=15)
smoothspline_20 <- smooth.spline(mcycle$times, mcycle$accel, df=20)

plot(mcycle$times, mcycle$accel, xlab = "Time (ms)", ylab = "Acceleration (g)")
lines(smoothspline_5, lwd = 2, col = "blue")
lines(smoothspline_10, lwd = 2, col = "red")
lines(smoothspline_15, lwd = 2, col = "yellow")
lines(smoothspline_20, lwd = 2, col = "gray")
legend(1, 70, legend = c("Smoothing Spline (df =5)",
                         "Smoothing Spline (df = 10)",
                         "Smoothing Spline (df = 15)",
       "Smoothing Spline(df =20)"),
       col = c("blue","red", "yellow", "gray"),
       lty = 1:4, cex = 1)


# Cross Validation Errrors with respect to degrees of freedom ranging from 5 to 20 with step 0.5.

df <- as.numeric()
cve <- as.numeric()

df_cv <- data.frame(df, cve)


for (i in seq(from = 5, to = 20, by = 0.5)){
  spline <- smooth.spline(mcycle$times, mcycle$accel, df = i)
  newline <- data.frame(df = i, CVE = spline$cv.crit)
  df_cv <- rbind(dfcv, newline)
}

plot(df_cv$df, df_cv$cve , xlab = "Degree of Freedon", ylab = "Cross Validation Error")
lines(df_cv$df, df_cv$cve)
```

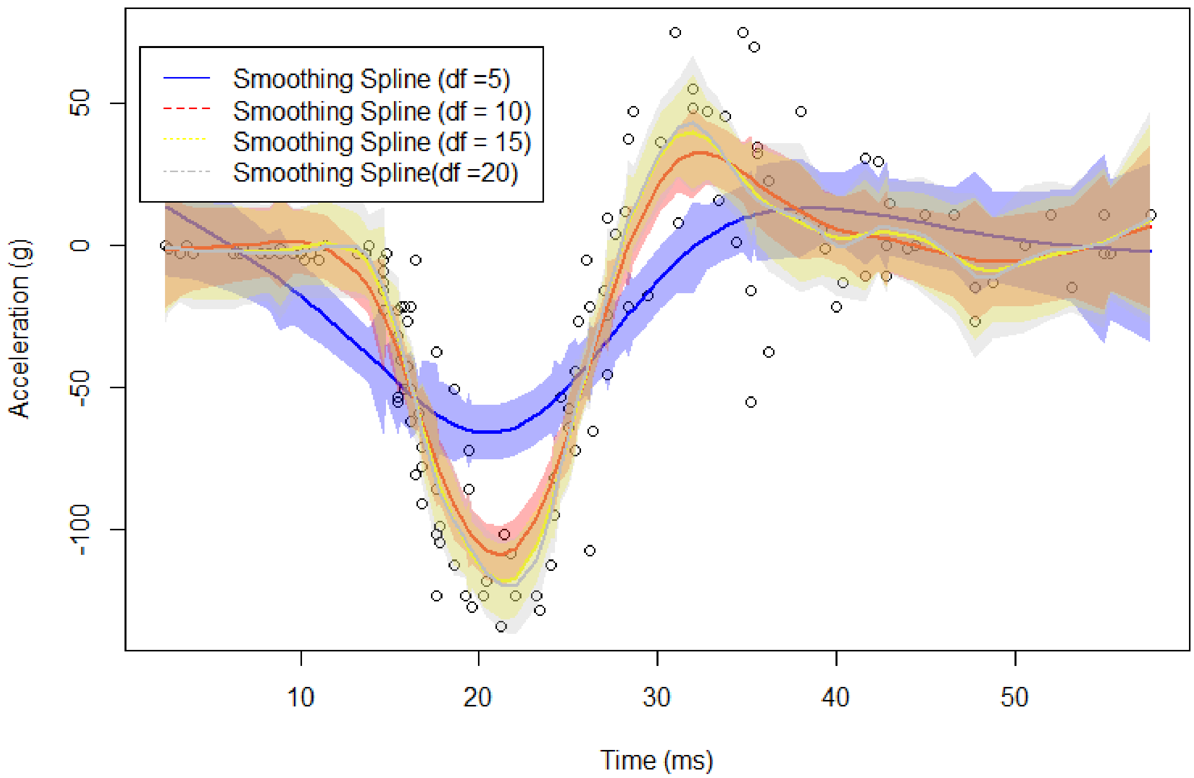
```
#Perform the wavelet analysis with any wavelet basis and resolution for the first 128 points of "accel".

wavelet_analysis = wd(c(mcycle$accel[1:128]))
summary(wavelet_analysis)
plot(wavelet_analysis)

#Soft threshold for the wavelet coefficients

soft_threshold = threshold(wavelet_analysis, type = 'soft')
soft_threshold

#Remaking of Profile and comparing against basis spline smoothing

par(mfcol = c(2,1))
plot(wr(soft_threshold), xlabels = mcycle$times)
plot(mcycle$times, mcycle$accel)
```

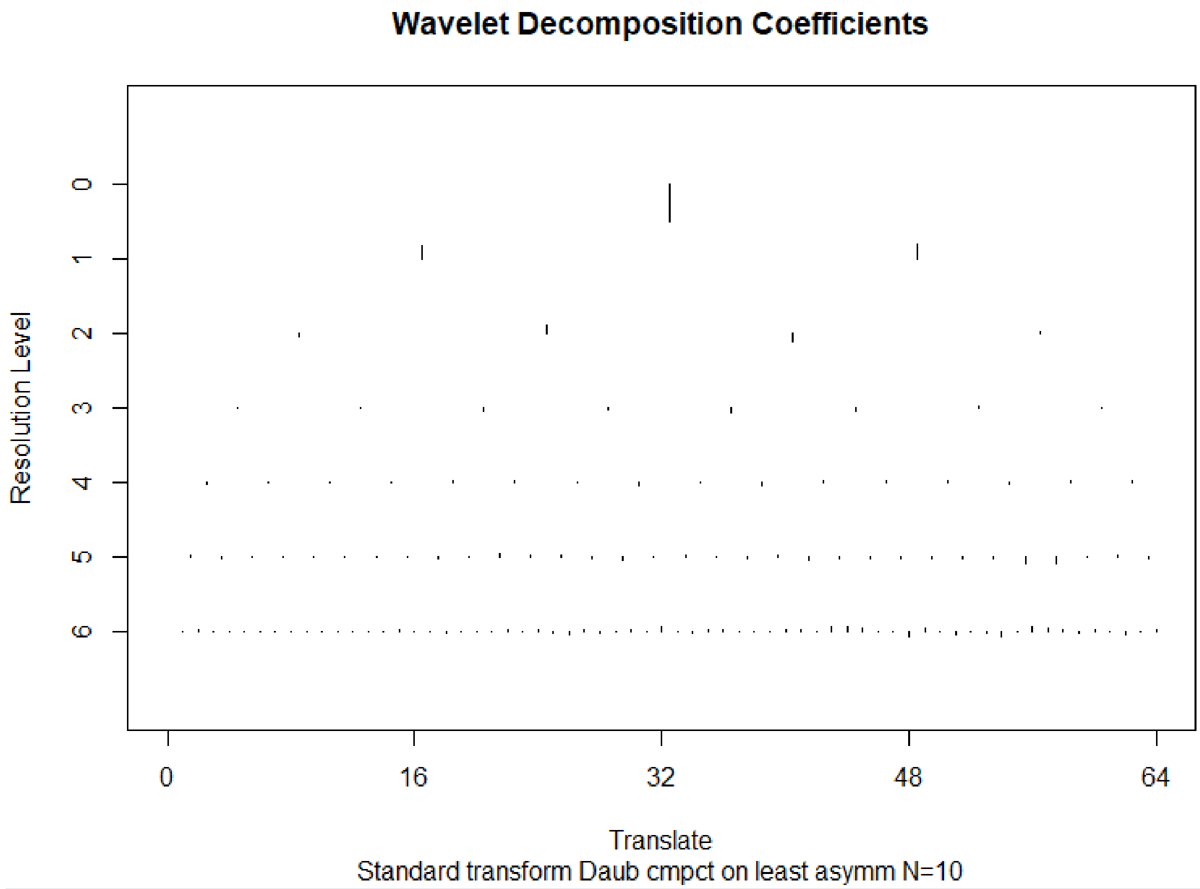
```
> smoothspline_fit <- smooth.spline(mcycle$times, mcycle$accel, cv=T)
Warning message:
In smooth.spline(mcycle$times, mcycle$accel, cv = T) :
  cross-validation with non-unique 'x' values seems doubtful
> smoothspline_fit
Call:
smooth.spline(x = mcycle$times, y = mcycle$accel, cv = T)

Smoothing Parameter  spar= 0.6483577  lambda= 9.146889e-05 (13 iterations)
Equivalent Degrees of Freedom (Df): 12.74136
Penalized Criterion (RSS): 38159.55
PRESS(l.o.o. CV): 543.1745
```

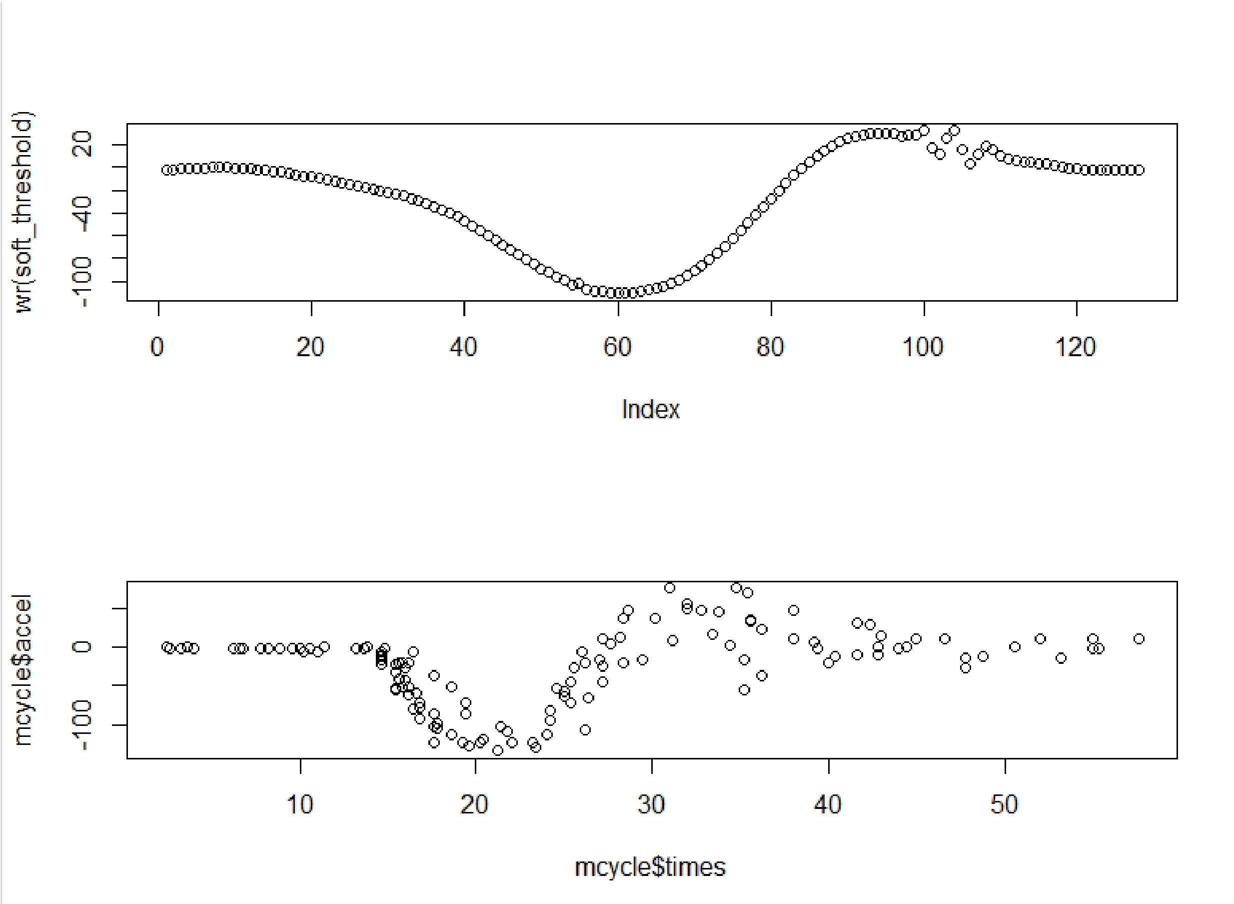
```
> wavelet_analysis = wd(c(mcycle$accel[1:128]))
> summary(wavelet_analysis)
Levels:  7
Length of original:  128
Filter was:  Daub cmpct on least asymm N=10
Boundary handling:  periodic
Transform type:  wavelet
Date:  Mon Nov 22 15:21:18 2021
> plot(wavelet_analysis)
[1] 419.7696 419.7696 419.7696 419.7696 419.7696 419.7696 419.7696
>
> #Soft threshold for the wavelet coefficients
>
> soft_threshold = threshold(wavelet_analysis, type = 'soft')
> soft_threshold
Class 'wd' : Discrete Wavelet Transform Object:
       ~~  : List with 8 components with names
              C D nlevels fl.dbase filter type bc date

$C and $D are LONG coefficient vectors

Created on : Mon Nov 22 15:21:18 2021
Type of decomposition:  wavelet

summary(.):
----------
Levels:  7
Length of original:  128
Filter was:  Daub cmpct on least asymm N=10
Boundary handling:  periodic
Transform type:  wavelet
Date:  Mon Nov 22 15:21:18 2021
>
```



smoothspline_optimal$y
0
-50
-100
10
20
30
40
50
smoothspline_optimal$x


Smoothing Spline (df =5)
Smoothing Spline (df = 10)
Smoothing Spline (df = 15)
Smoothing Spline(df =20)
Acceleration (g)
50
0
-50
-100
10
20
30
40
50
Time (ms)


Wavelet Decomposition Coefficients
Resolution Level
0
1
2
3
4
5
6
0
16
32
48
64
Translate
Standard transform Daub cmpct on least asymm N=10


wr(soft_threshold)
20
-40
-100
0
20
40
60
80
100
120
Index
mcycle$accel
0
-100
10
20
30
40
50
mcycle$times

Q3

```r
library(fda)
library(caret)
library(glmnet)

ecg_data_train = read.csv("C:/Users/ppill/Desktop/R files/ECG200TRAIN.csv",
                          header=F, sep=',', na.strings=c('.','NA','99999999'))

ecg_data_test = read.csv("C:/Users/ppill/Desktop/R files/ECG200TEST.csv",
                         header=F, sep=',', na.strings=c('.','NA','99999999'))

#Data Manipulation to get matrix form for analysis

head(ecg_data_train)
ecg_data_train_matrix <- matrix(data = unlist(ecg_data_train), nrow = 100)
ecg_data_test_matrix <- matrix(data = unlist(ecg_data_test), nrow = 100)

y = ecg_data_train_matrix[,1]
x = ecg_data_train_matrix[,2:97]
y[y ==-1] = 0

y_test = ecg_data_test_matrix[,1 ]
x_test = ecg_data_test_matrix[,2:97]
y_test[y_test ==-1] = 0

argvals = seq(0,1, length.out = dim(x)[2])

#Generating Spline Basis for training data

nbasis = 12
bbasis = create.bspline.basis(c(0,1), nbasis)

train_Coef = matrix(0,dim(x)[1],nbasis)


for (i in 1:dim(x)[1]) {
  train_Coef[i,] = smooth.basis(argvals = argvals,
                                y = as.numeric(x[i,]),
                                fdParobj = bbasis)$fd$coefs
}

#Generating Basis for Testing Data

argvals = seq(0,1, length.out = dim(x_test)[2])
nbasis = 24
bbasis = create.bspline.basis(c(0,1), nbasis)



test_Coef = matrix(0,dim(x_test)[1],nbasis)
for (i in 1:dim(x_test)[1]) {
  test_Coef[i,] = smooth.basis(argvals = argvals,
                               y = as.numeric(x_test[i,]),
                               fdParobj = bbasis)$fd$coefs
}

```

```r
#Decomposing the coefficient for test and training datasets

test_Coef <- as.data.frame(test_Coef)
train_Coef <- as.data.frame(train_Coef)

tdata_x <- test_Coef
tdata_y <-  y_test

#Fitting the decomposed model to a Logistic Regression

ecg_data_train[[1]] <- gsub(-1,0, ecg_data_train[[1]])
ecg_data_train[[1]] <- as.factor(as.numeric(ecg_data_train[[1]]))
train_Coef$y<- ecg_data_train[[1]]

m0_log <- glm(y~ ., family = 'binomial', data = train_Coef)
pred_m0 <- predict(m0_log, test_Coef, type = 'response')
pred_m0

confusionMatrix(data = as.factor(as.numeric(pred_m0>0.5)), reference = as.factor(y))
```

```
> m0_log <- glm(y~ ., family = 'binomial', data = train_Coef)
> pred_m0 <- predict(m0_log, test_Coef, type = 'response')
> pred_m0
           1            2            3            4            5            6            7            8            9           10 
1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 
          11           12           13           14           15           16           17           18           19           20 
1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 2.220446e-16 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 
          21           22           23           24           25           26           27           28           29           30 
1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 
          31           32           33           34           35           36           37           38           39           40 
1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 
          41           42           43           44           45           46           47           48           49           50 
1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 
          51           52           53           54           55           56           57           58           59           60 
1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 
          61           62           63           64           65           66           67           68           69           70 
1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 2.220446e-16 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 
          71           72           73           74           75           76           77           78           79           80 
1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 
          81           82           83           84           85           86           87           88           89           90 
1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 2.220446e-16 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 
          91           92           93           94           95           96           97           98           99          100 
1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 1.000000e+00 2.220446e-16 1.000000e+00 1.000000e+00 
> 
> confusionMatrix(data = as.factor(as.numeric(pred_m0>0.5)), reference = as.factor(y))
```