SYED NOOR RAZI ALI 2070326

DSC 425

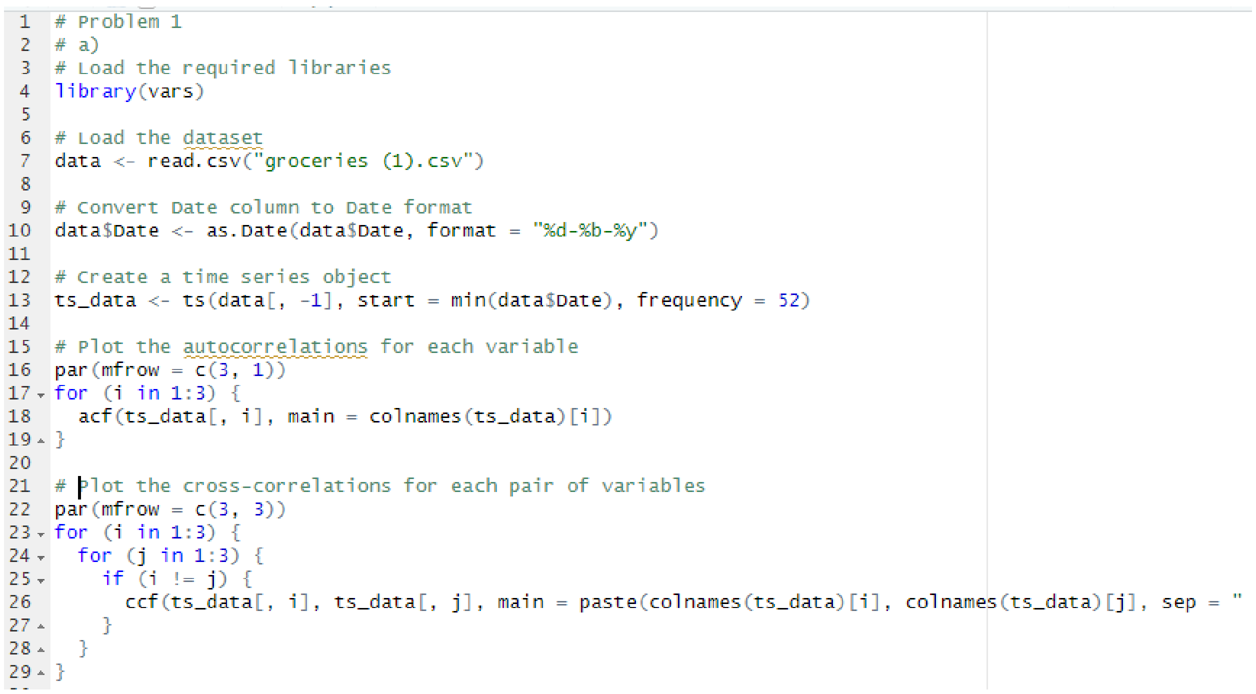
TIME SERIES ANALYSIS AND FORECASTING

HOMEWORK MODULE 8 & 9

`

Problem 1

a)

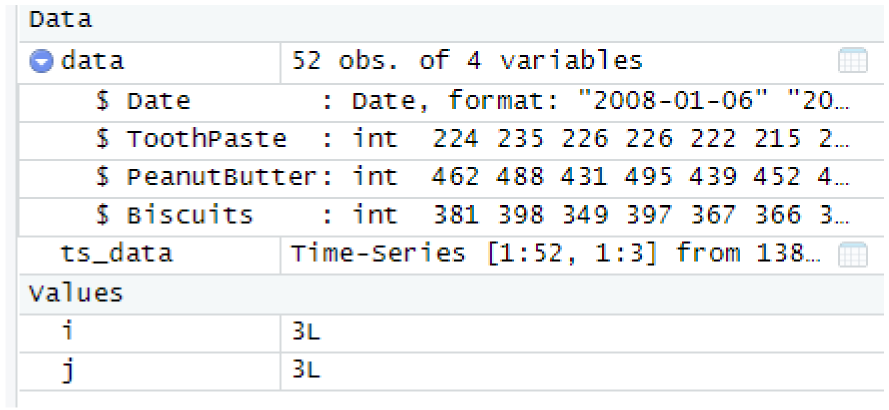


```r
# Problem 1
# a)
# Load the required libraries
library(vars)

# Load the dataset
data <- read.csv("groceries (1).csv")

# Convert Date column to Date format
data$Date <- as.Date(data$Date, format = "%d-%b-%y")

# Create a time series object
ts_data <- ts(data[, -1], start = min(data$Date), frequency = 52)

# Plot the autocorrelations for each variable
par(mfrow = c(3, 1))
for (i in 1:3) {
  acf(ts_data[, i], main = colnames(ts_data)[i])
}

# Plot the cross-correlations for each pair of variables
par(mfrow = c(3, 3))
for (i in 1:3) {
  for (j in 1:3) {
    if (i != j) {
      ccf(ts_data[, i], ts_data[, j], main = paste(colnames(ts_data)[i], colnames(ts_data)[j], sep = "
    }
  }
}
```

Figure 1: Loading required dataset for time series

(Source: Developed in R studio)

Implementing R studio, the above figure shows the time series for the loading required dataset.



```
Data
data            52 obs. of 4 variables
   $ Date        : Date, format: "2008-01-06" "20…
   $ ToothPaste  : int  224 235 226 226 222 215 2…
   $ PeanutButter: int  462 488 431 495 439 452 4…
   $ Biscuits    : int  381 398 349 397 367 366 3…
  ts_data       Time-Series [1:52, 1:3] from 138…
Values
  i             3L
  j             3L
```

Figure 2: Dataset of 4 variables

(Source: Developed in R studio)

This picture shows four variables of the dataset.

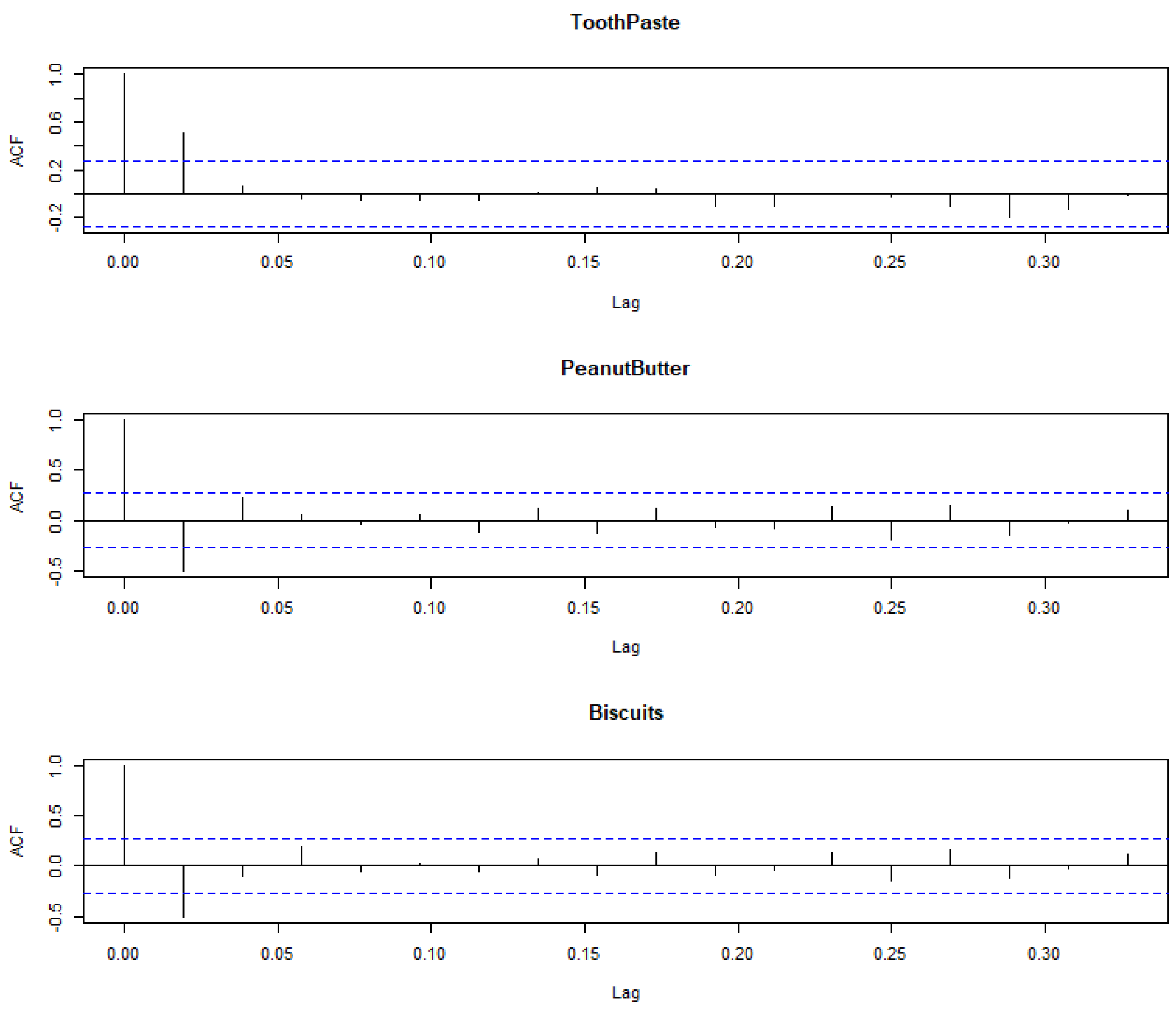


Figure 3: Plotting of three variables

(Source: Developed in R studio)

Implementing R studio, In this figure shows the plotting of ToothPaste, PeanutButter, and Biscuits.

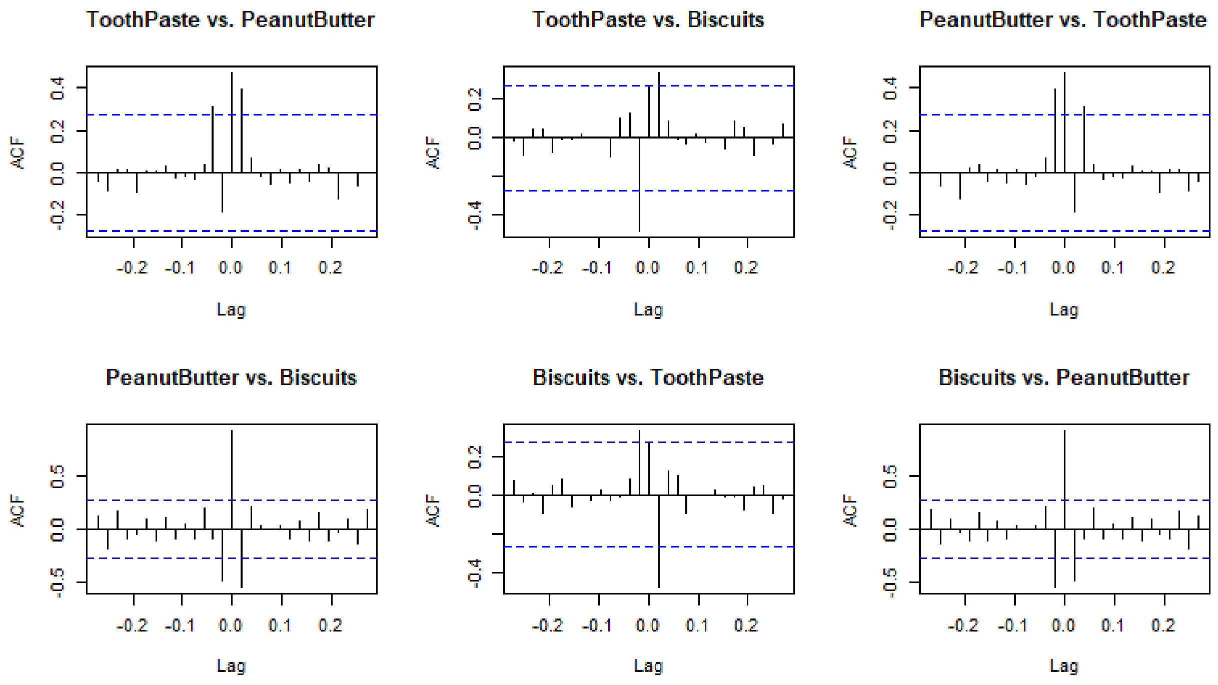
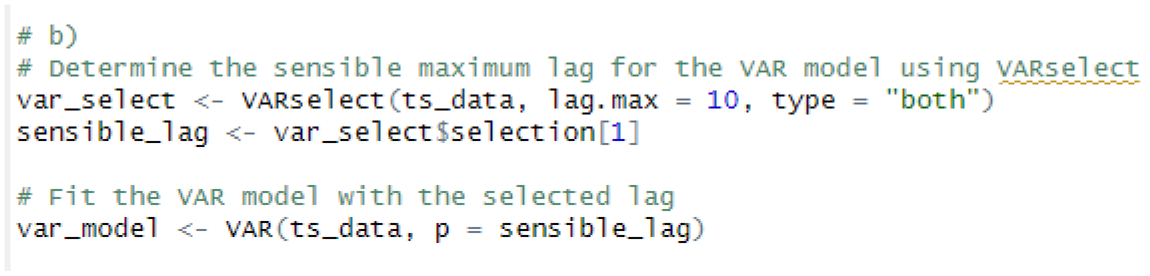


Figure 4: Different plots of three variables

(Source: Developed in R studio)

This figure was implemented by R studio and shows three variables of different plots where Toothpaste – Peanut Butter has the most autocorrelation.

b)

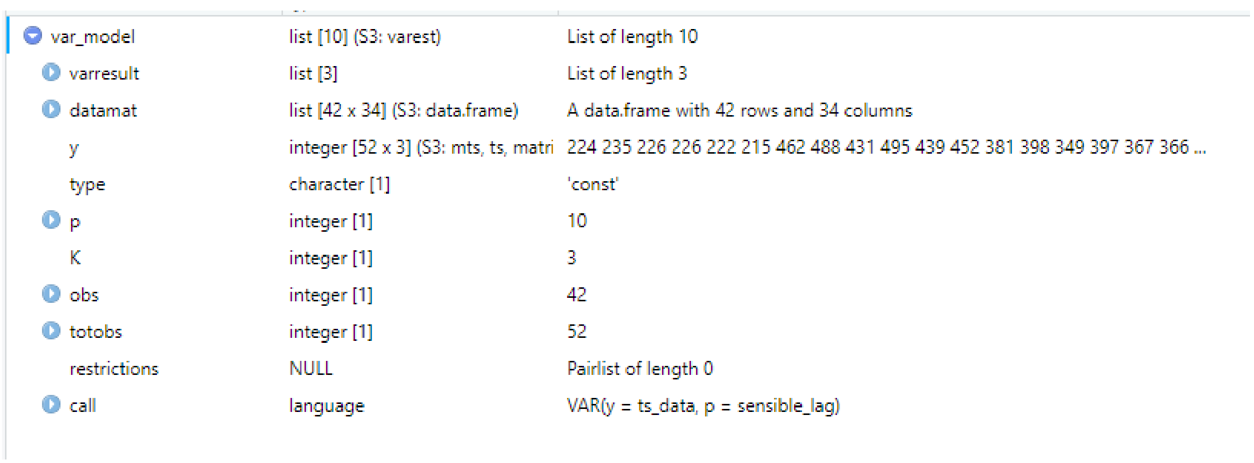


```
# b)
# Determine the sensible maximum lag for the VAR model using VARselect
var_select <- VARselect(ts_data, lag.max = 10, type = "both")
sensible_lag <- var_select$selection[1]

# Fit the VAR model with the selected lag
var_model <- VAR(ts_data, p = sensible_lag)
```

Figure 5: Determining sensible maximum lag for the VAR model using VAR select

(Source: Developed in R studio)

In this picture, easily determining the sensible maximum lag for the VAR model using VAR select.

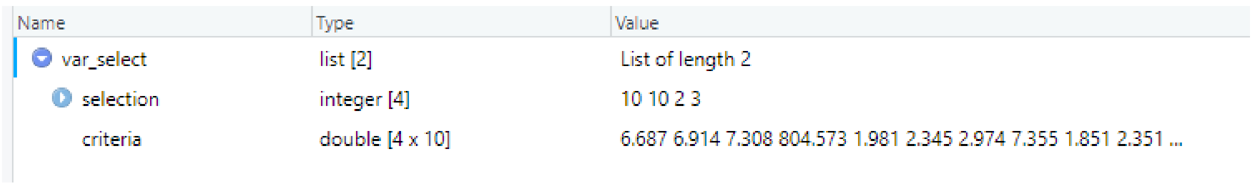


```
var_model        list [10] (S3: varest)                List of length 10
  varresult      list [3]                              List of length 3
  datamat        list [42 x 34] (S3: data.frame)       A data.frame with 42 rows and 34 columns
  y              integer [52 x 3] (S3: mts, ts, matri  224 235 226 226 222 215 462 488 431 495 439 452 381 398 349 397 367 366 ...
  type           character [1]                         'const'
  p              integer [1]                           10
  K              integer [1]                           3
  obs            integer [1]                           42
  totobs         integer [1]                           52
  restrictions   NULL                                  Pairlist of length 0
  call           language                              VAR(y = ts_data, p = sensible_lag)
```

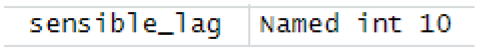
Figure 6: VAR selection function

(Source: Developed in R studio)

This picture shows the function of VAR selection.



```
Name           Type             Value
var_select     list [2]         List of length 2
  selection    integer [4]      10 10 2 3
  criteria     double [4 x 10]  6.687 6.914 7.308 804.573 1.981 2.345 2.974 7.355 1.851 2.351 ...
```



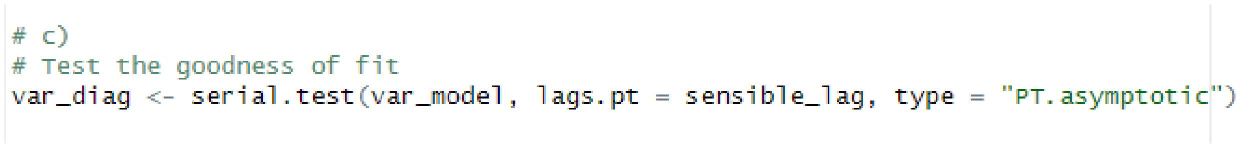
```
sensible_lag   Named int 10
```

Figure 7: Sensible lag of the VAR model

(Source: Developed in R studio)

This picture is used for the sensible lag of the VAR model.

c)

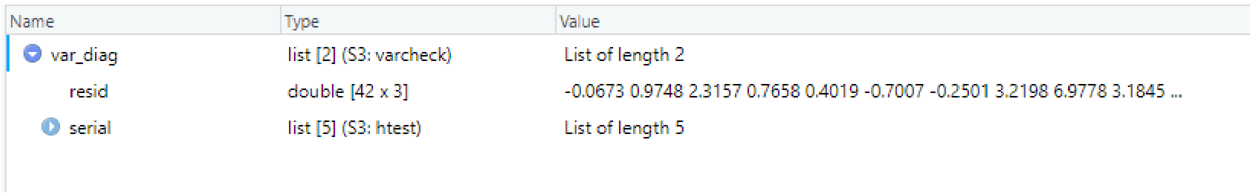


```
# c)
# Test the goodness of fit
var_diag <- serial.test(var_model, lags.pt = sensible_lag, type = "PT.asymptotic")
```

Figure 8: Testing the goodness of fit

(Source: Developed in R studio)

This figure shows testing the goodness of fit.



Name	Type	Value
var_diag	list [2] (S3: varcheck)	List of length 2
resid	double [42 x 3]	-0.0673 0.9748 2.3157 0.7658 0.4019 -0.7007 -0.2501 3.2198 6.9778 3.1845 ...
serial	list [5] (S3: htest)	List of length 5

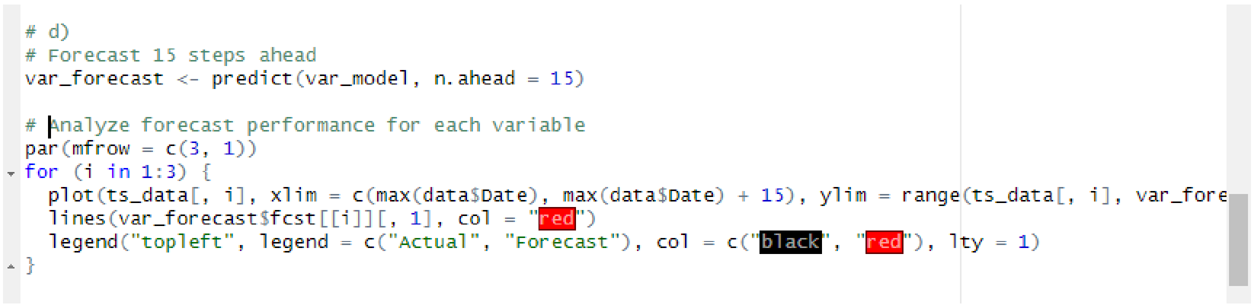
Figure 9: VAR selection results

(Source: Developed in R studio)

This figure shows the outcome of the VAR selection.

According to the VAR model, the lagged values of ToothPaste appear to have a substantial impact on the current value of ToothPaste, whereas the lagged values of PeanutButter appear to have a big impact on the current value of PeanutButter.

d)

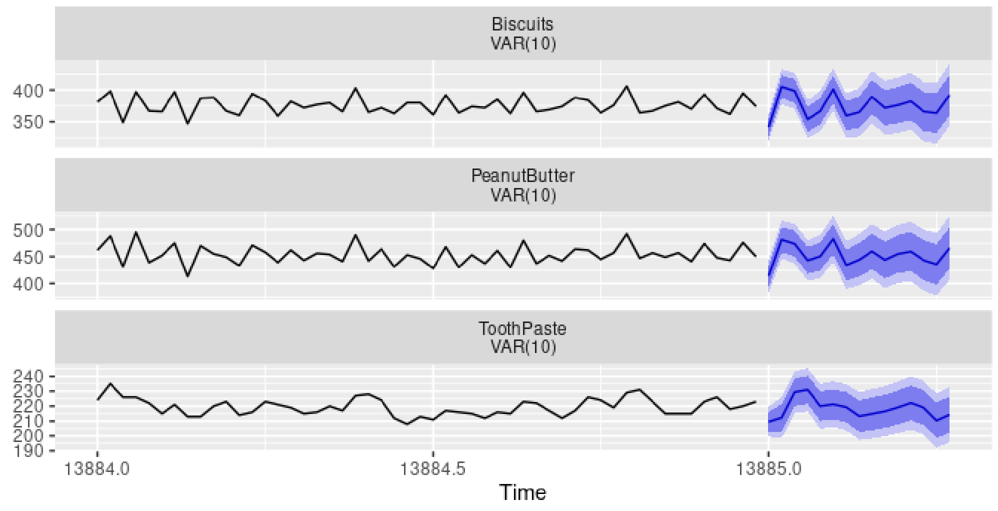


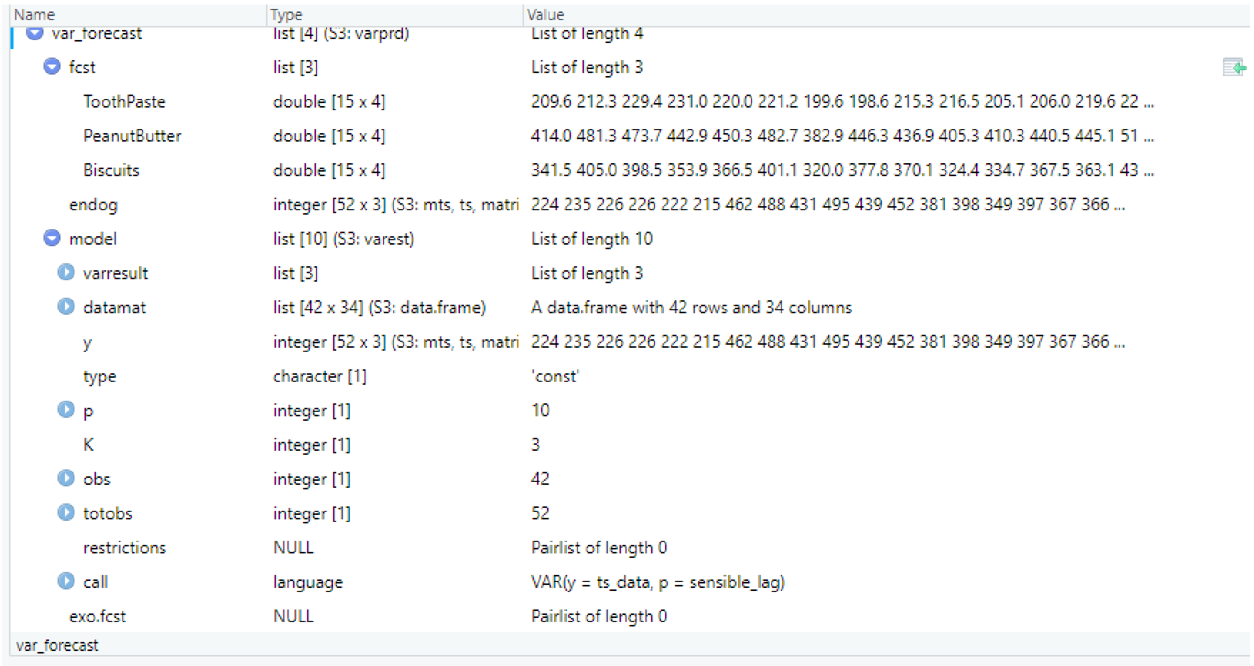
```
# d)
# Forecast 15 steps ahead
var_forecast <- predict(var_model, n.ahead = 15)

# Analyze forecast performance for each variable
par(mfrow = c(3, 1))
for (i in 1:3) {
  plot(ts_data[, i], xlim = c(max(data$Date), max(data$Date) + 15), ylim = range(ts_data[, i], var_fore
  lines(var_forecast$fcst[[i]][, 1], col = "red")
  legend("topleft", legend = c("Actual", "Forecast"), col = c("black", "red"), lty = 1)
}
```

Figure 10: Forecasting 15 steps ahead

(Source: Developed in R studio)

Implementing by R studio, this figure shows the 15 steps ahead for forecasting.





```
Name                  Type                                   Value
var_forecast          list [4] (S3: varprd)                  List of length 4
  fcst                list [3]                               List of length 3
    ToothPaste        double [15 x 4]                        209.6 212.3 229.4 231.0 220.0 221.2 199.6 198.6 215.3 216.5 205.1 206.0 219.6 22 ...
    PeanutButter      double [15 x 4]                        414.0 481.3 473.7 442.9 450.3 482.7 382.9 446.3 436.9 405.3 410.3 440.5 445.1 51 ...
    Biscuits          double [15 x 4]                        341.5 405.0 398.5 353.9 366.5 401.1 320.0 377.8 370.1 324.4 334.7 367.5 363.1 43 ...
  endog               integer [52 x 3] (S3: mts, ts, matri   224 235 226 226 222 215 462 488 431 495 439 452 381 398 349 397 367 366 ...
  model               list [10] (S3: varest)                 List of length 10
    varresult         list [3]                               List of length 3
    datamat           list [42 x 34] (S3: data.frame)        A data.frame with 42 rows and 34 columns
    y                 integer [52 x 3] (S3: mts, ts, matri   224 235 226 226 222 215 462 488 431 495 439 452 381 398 349 397 367 366 ...
    type              character [1]                          'const'
    p                 integer [1]                            10
    K                 integer [1]                            3
    obs               integer [1]                            42
    totobs            integer [1]                            52
    restrictions      NULL                                   Pairlist of length 0
    call              language                               VAR(y = ts_data, p = sensible_lag)
  exo.fcst            NULL                                   Pairlist of length 0
var_forecast
```

Figure 11: Loading required dataset for time series

(Source: Developed in R studio)

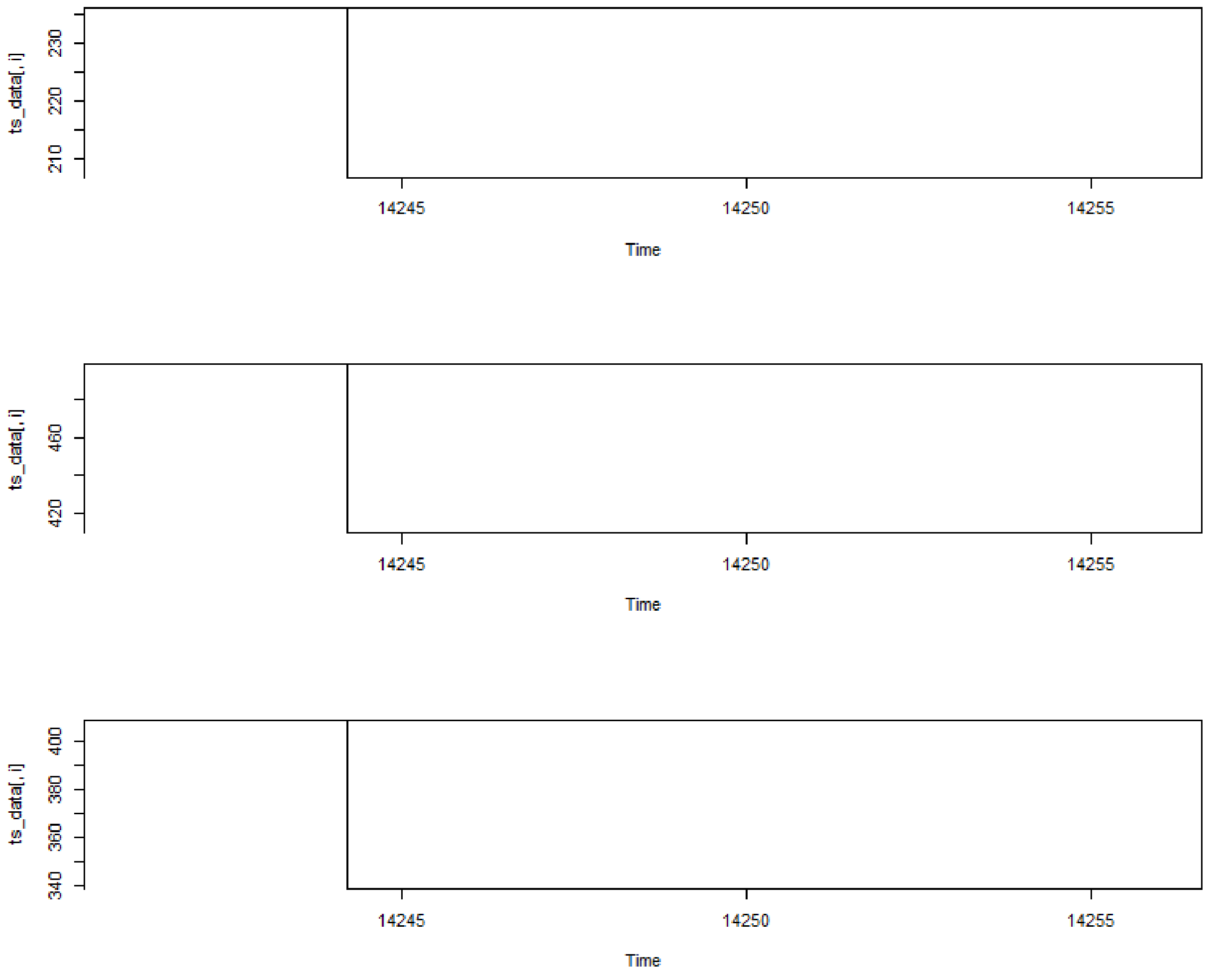
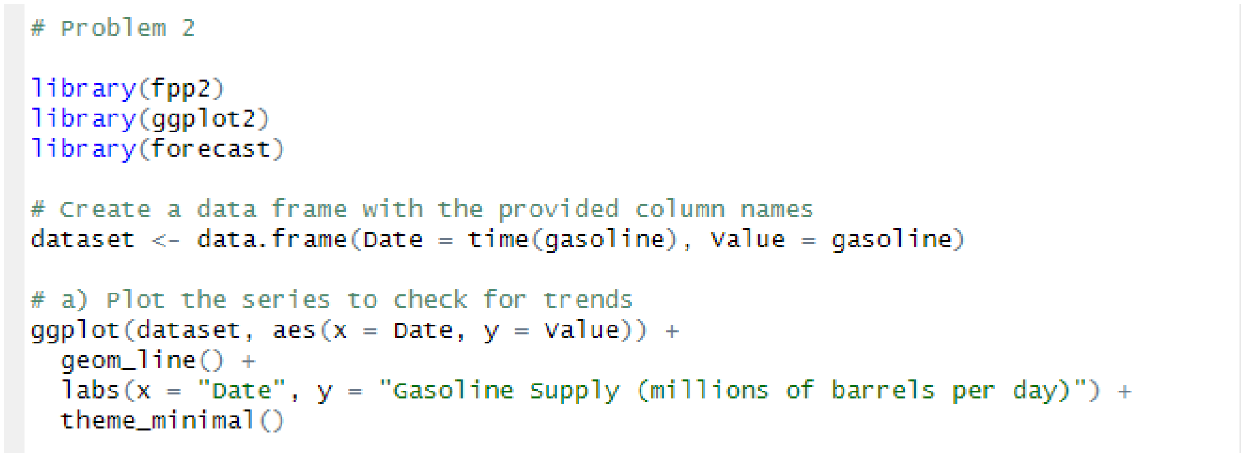


Figure 12: Loading required dataset for time series

(Source: Developed in R studio)

Problem 2

a)



```r
# Problem 2

library(fpp2)
library(ggplot2)
library(forecast)

# Create a data frame with the provided column names
dataset <- data.frame(Date = time(gasoline), Value = gasoline)

# a) Plot the series to check for trends
ggplot(dataset, aes(x = Date, y = Value)) +
  geom_line() +
  labs(x = "Date", y = "Gasoline Supply (millions of barrels per day)") +
  theme_minimal()
```

Figure 13: Creating a data frame with the provided column names

(Source: Developed in R studio)

The above figure represents creating a data frame using the column names supplied in R studio.

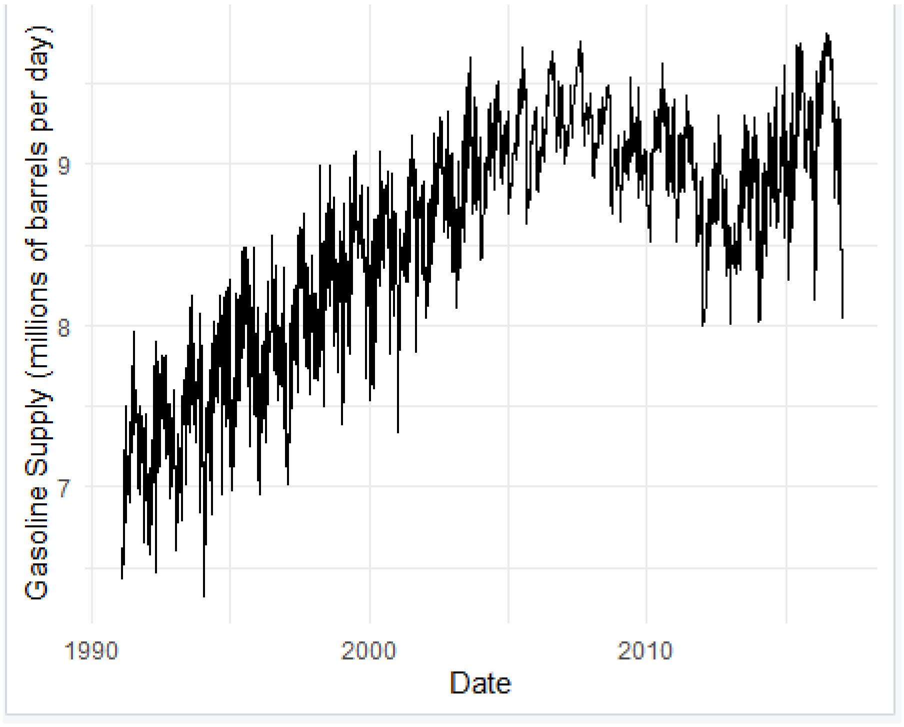
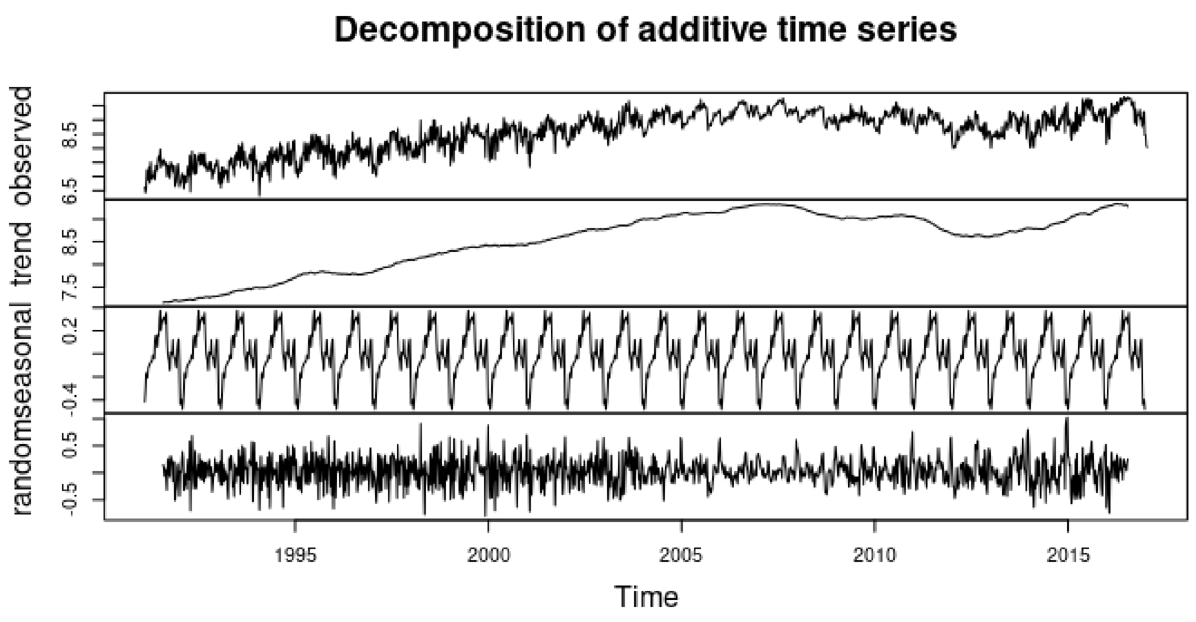


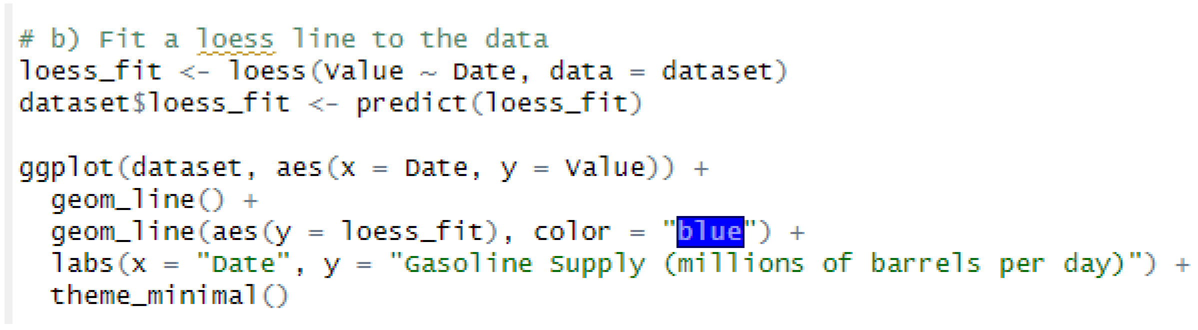
Figure 14: Plotting of Gasoline Supplies

(Source: Developed in R studio)

Plotting of the gasoline supplies is also shown based on the baseless per-day analysis also done for this section.



b)



```
# b) Fit a loess line to the data
loess_fit <- loess(Value ~ Date, data = dataset)
dataset$loess_fit <- predict(loess_fit)

ggplot(dataset, aes(x = Date, y = Value)) +
  geom_line() +
  geom_line(aes(y = loess_fit), color = "blue") +
  labs(x = "Date", y = "Gasoline Supply (millions of barrels per day)") +
  theme_minimal()
```

Figure 15: Fitting a loess line to the data

(Source: Developed in R studio)

Fitting of the model based on the x and y coordinates of the data and ggplot is also used for this section.

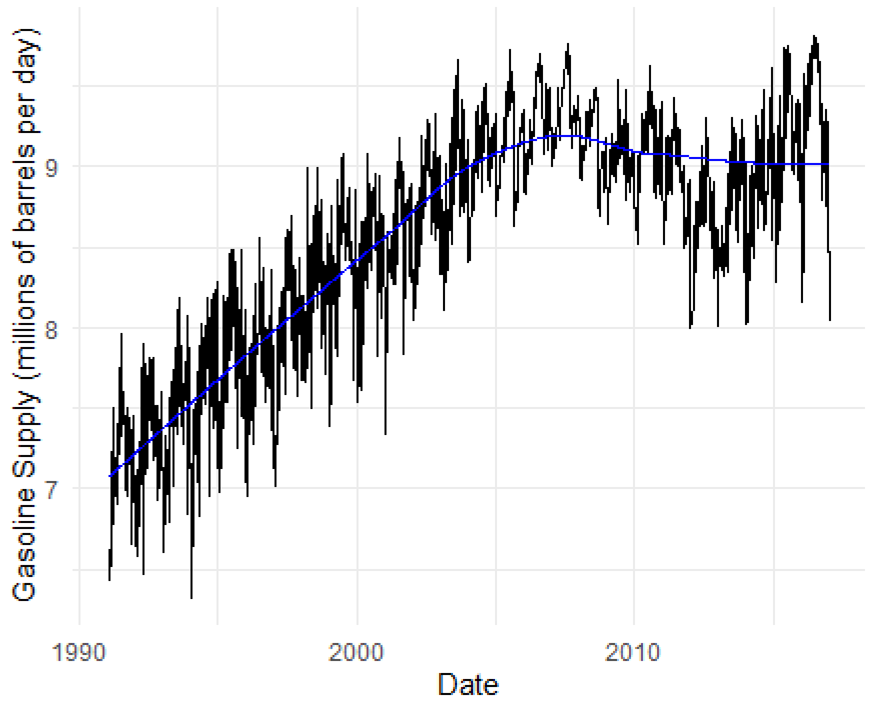
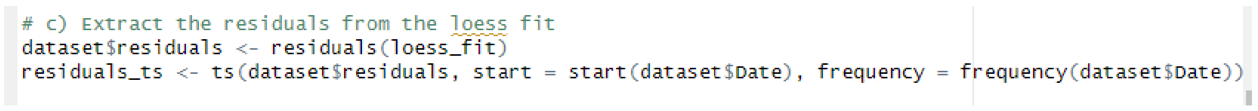


Figure 16: Loading required dataset for time series

(Source: Developed in R studio)

The dataset of time series and loading data is also analysed based on the section of the time series.

c)

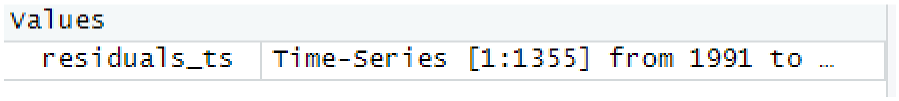


```
# c) Extract the residuals from the loess fit
dataset$residuals <- residuals(loess_fit)
residuals_ts <- ts(dataset$residuals, start = start(dataset$Date), frequency = frequency(dataset$Date))
```

Figure 17: Extracting the residuals from loess fit

(Source: Developed in R studio)

A residual of the loss fit is also done in the model for the extraction.



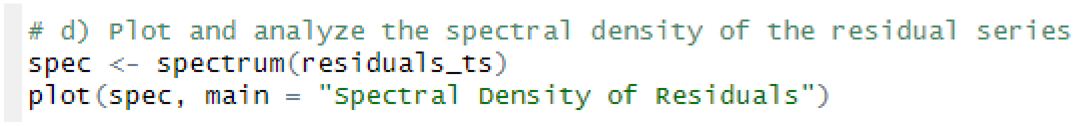
```
Values
  residuals_ts    Time-Series [1:1355] from 1991 to …
```

Figure 18: Value of residuals and time series

(Source: Developed in R studio)

The .above figure represents the value of residuals and time series that is 1:1355 from 1991.

d)



```
# d) Plot and analyze the spectral density of the residual series
spec <- spectrum(residuals_ts)
plot(spec, main = "Spectral Density of Residuals")
```

Figure 19: Plot and analyze the spectral density of the residual series

(Source: Developed in R studio)

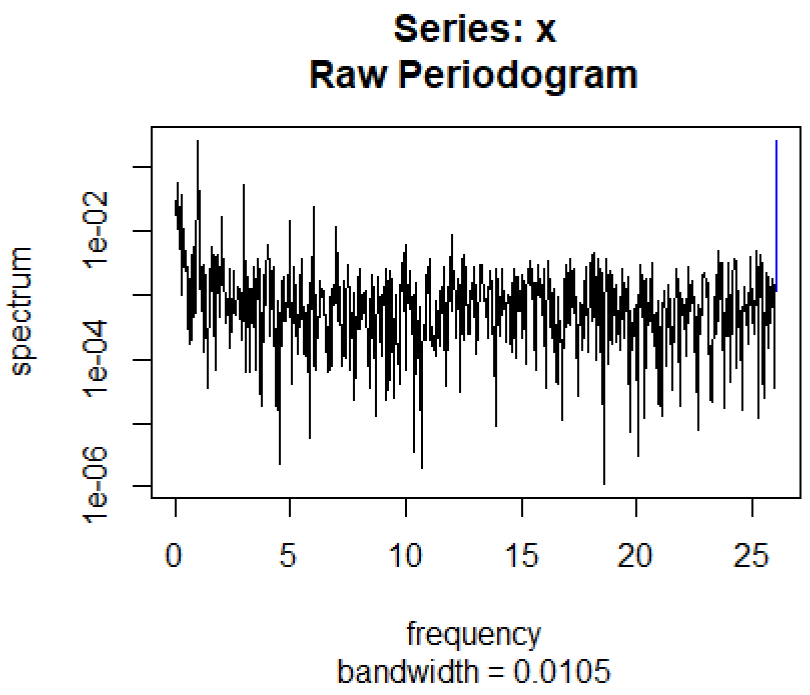


Figure 20: Plotting of raw periodogram

(Source: Developed in R studio)

Plotting of raw periodogram and analysis of result for this section. Based on the bandwidth analysis is focused on this analysis.

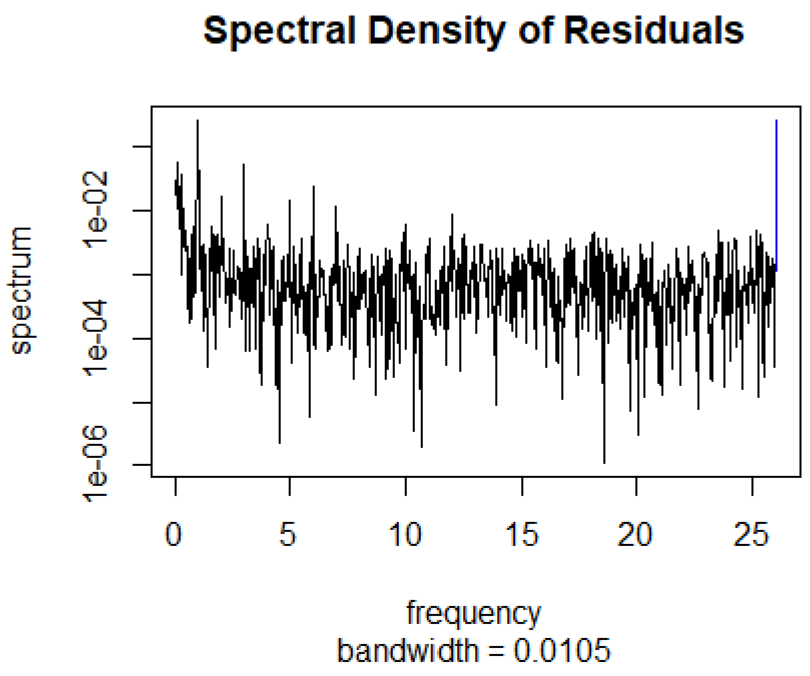
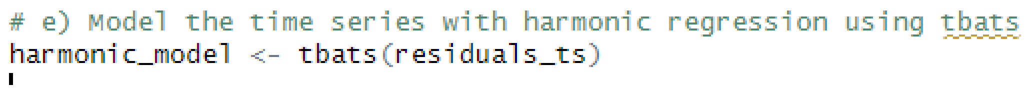


Figure 21: Plotting of spectral density of residuals

(Source: Developed in R studio)

The spectral density of the residual analysis and frequency section is shown in this section. Data based on bandwidth and frequency analysis.

e)

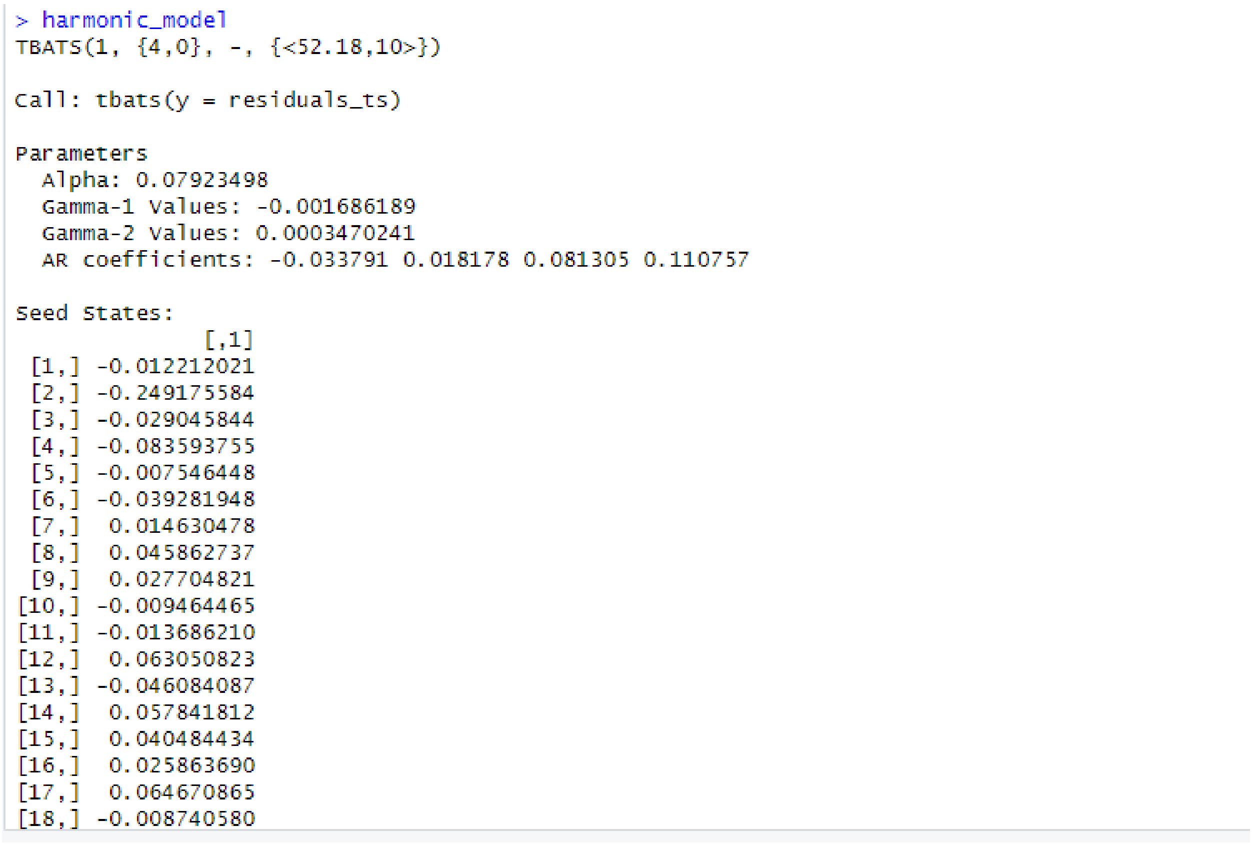


```
# e) Model the time series with harmonic regression using tbats
harmonic_model <- tbats(residuals_ts)
```

Figure 22: Modeling the time series with harmonic regression

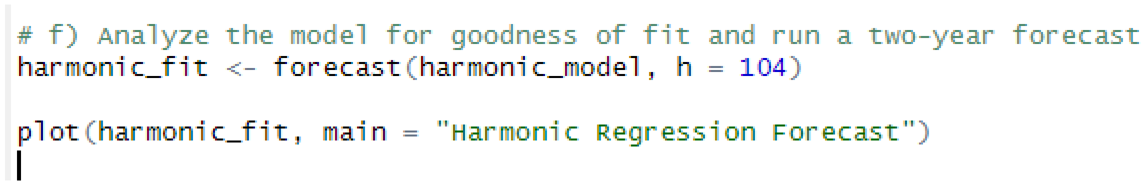
(Source: Developed in R studio)

Modeling for the time series based on the data and harmonic regression is analyzed in this section.



```
> harmonic_model
TBATS(1, {4,0}, -, {<52.18,10>})

Call: tbats(y = residuals_ts)

Parameters
  Alpha: 0.07923498
  Gamma-1 Values: -0.001686189
  Gamma-2 Values: 0.0003470241
  AR coefficients: -0.033791 0.018178 0.081305 0.110757

Seed States:
              [,1]
 [1,] -0.012212021
 [2,] -0.249175584
 [3,] -0.029045844
 [4,] -0.083593755
 [5,] -0.007546448
 [6,] -0.039281948
 [7,]  0.014630478
 [8,]  0.045862737
 [9,]  0.027704821
[10,] -0.009464465
[11,] -0.013686210
[12,]  0.063050823
[13,] -0.046084087
[14,]  0.057841812
[15,]  0.040484434
[16,]  0.025863690
[17,]  0.064670865
[18,] -0.008740580
```

f)

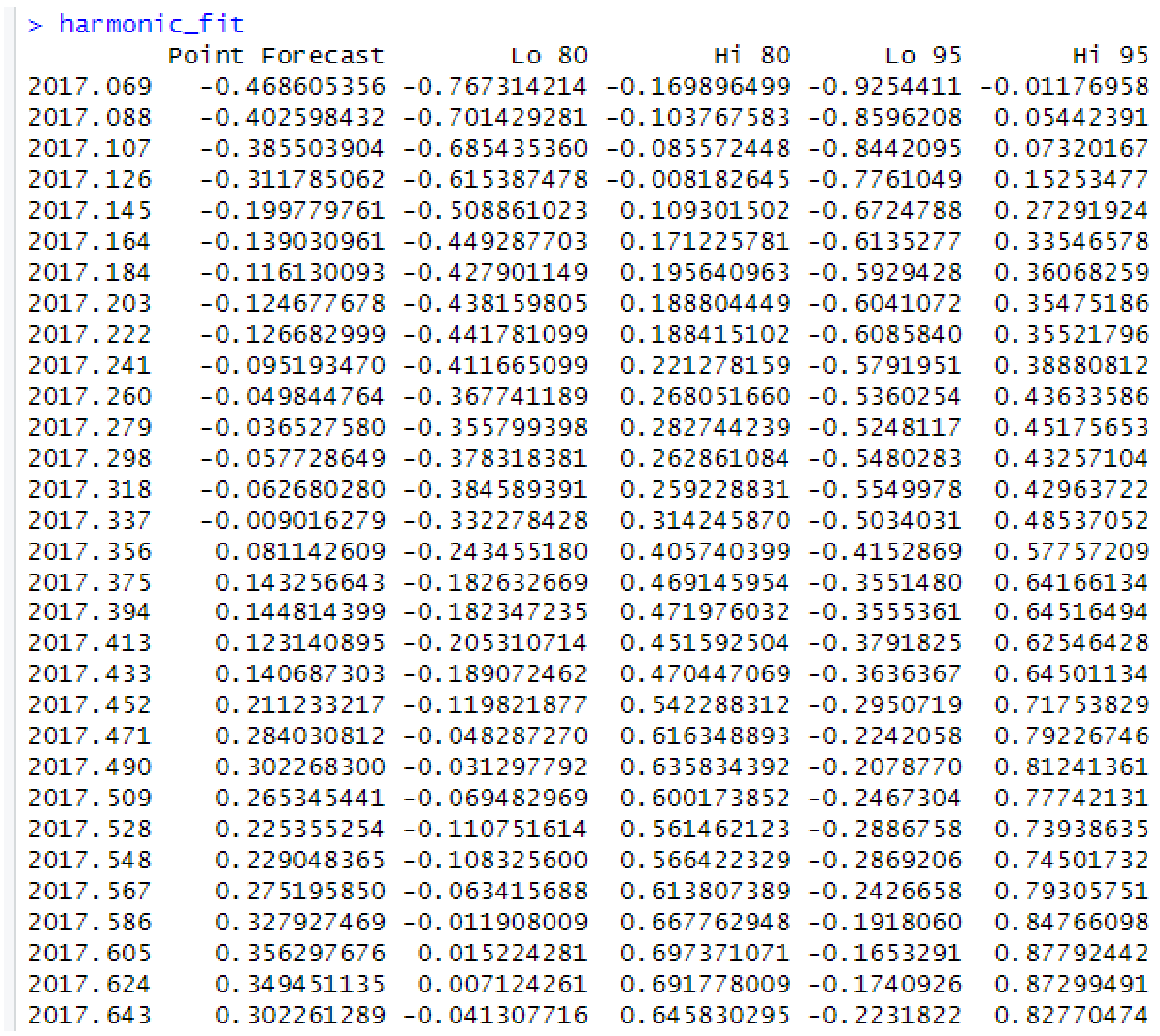


```
# f) Analyze the model for goodness of fit and run a two-year forecast
harmonic_fit <- forecast(harmonic_model, h = 104)

plot(harmonic_fit, main = "Harmonic Regression Forecast")
```

Figure 23: Analyzing the model for goodness of fit and running a two-year forecast

(Source: Developed in R studio)

A two-year forecast based on the data and fitted model for the analysis is also done in this analysis. Therefore, analysis based on the harmonic fitted model is also analysed for this section.



```
> harmonic_fit
         Point Forecast         Lo 80        Hi 80      Lo 95       Hi 95
2017.069   -0.468605356 -0.767314214 -0.169896499 -0.9254411 -0.01176958
2017.088   -0.402598432 -0.701429281 -0.103767583 -0.8596208  0.05442391
2017.107   -0.385503904 -0.685435360 -0.085572448 -0.8442095  0.07320167
2017.126   -0.311785062 -0.615387478 -0.008182645 -0.7761049  0.15253477
2017.145   -0.199779761 -0.508861023  0.109301502 -0.6724788  0.27291924
2017.164   -0.139030961 -0.449287703  0.171225781 -0.6135277  0.33546578
2017.184   -0.116130093 -0.427901149  0.195640963 -0.5929428  0.36068259
2017.203   -0.124677678 -0.438159805  0.188804449 -0.6041072  0.35475186
2017.222   -0.126682999 -0.441781099  0.188415102 -0.6085840  0.35521796
2017.241   -0.095193470 -0.411665099  0.221278159 -0.5791951  0.38880812
2017.260   -0.049844764 -0.367741189  0.268051660 -0.5360254  0.43633586
2017.279   -0.036527580 -0.355799398  0.282744239 -0.5248117  0.45175653
2017.298   -0.057728649 -0.378318381  0.262861084 -0.5480283  0.43257104
2017.318   -0.062680280 -0.384589391  0.259228831 -0.5549978  0.42963722
2017.337   -0.009016279 -0.332278428  0.314245870 -0.5034031  0.48537052
2017.356    0.081142609 -0.243455180  0.405740399 -0.4152869  0.57757209
2017.375    0.143256643 -0.182632669  0.469145954 -0.3551480  0.64166134
2017.394    0.144814399 -0.182347235  0.471976032 -0.3555361  0.64516494
2017.413    0.123140895 -0.205310714  0.451592504 -0.3791825  0.62546428
2017.433    0.140687303 -0.189072462  0.470447069 -0.3636367  0.64501134
2017.452    0.211233217 -0.119821877  0.542288312 -0.2950719  0.71753829
2017.471    0.284030812 -0.048287270  0.616348893 -0.2242058  0.79226746
2017.490    0.302268300 -0.031297792  0.635834392 -0.2078770  0.81241361
2017.509    0.265345441 -0.069482969  0.600173852 -0.2467304  0.77742131
2017.528    0.225355254 -0.110751614  0.561462123 -0.2886758  0.73938635
2017.548    0.229048365 -0.108325600  0.566422329 -0.2869206  0.74501732
2017.567    0.275195850 -0.063415688  0.613807389 -0.2426658  0.79305751
2017.586    0.327927469 -0.011908009  0.667762948 -0.1918060  0.84766098
2017.605    0.356297676  0.015224281  0.697371071 -0.1653291  0.87792442
2017.624    0.349451135  0.007124261  0.691778009 -0.1740926  0.87299491
2017.643    0.302261289 -0.041307716  0.645830295 -0.2231822  0.82770474
```

Figure 24: Output of harmonic fit

(Source: Developed in R studio)

The harmonic fitted result for the analysis and result section is analysed in this result. Different result based on the analysis is also done for this result.

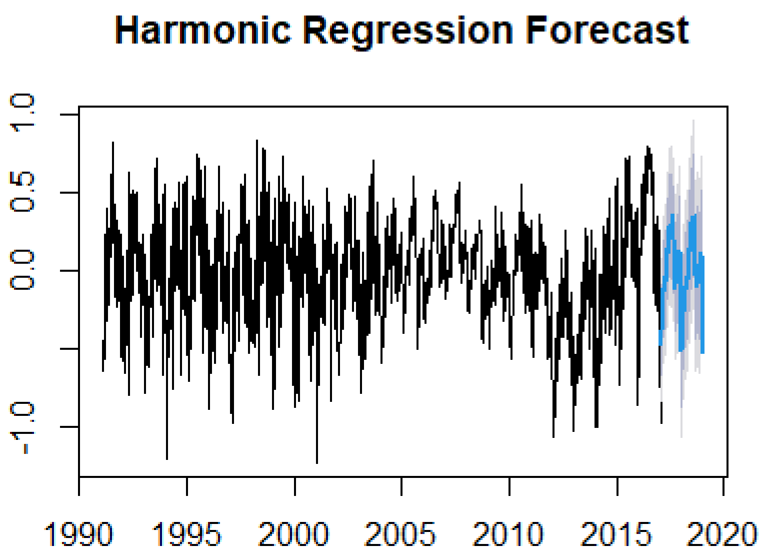


Figure 21: Harmonic Regression

(Source: Developed in R studio)

Harmonic regression for the analysis and yearly basis results are shown based on the result analysis.

The uncertainty in the predictions are shown by the forecasted values. Narrower bands denote more precise forecasts, while wider bands denote higher levels of uncertainty.

The series is expected to remain stable.