Time Series Forecasting with R

A Beginner's Guide

Dr.Arunachalam Rajagopal

Former Professor, Sharda University,

Greater Noida, UP, India

Table of Contents:

About the author:

The author Dr.Arunachlam Rajagopal graduated in mechanical engineering from Anna University, Chennai. He obtained his post graduate degree M.Tech (Industrial Management) from IIT Madras and also holds a MBA degree from REC, Trichy (Presently, NIT, Trichy). He obtained his doctoral degree from Anna University. His area of research interest lies in Operations Management, Supply Chain Management, and Business Analytics.

He worked as Engineer / Senior Engineer at BHEL, Trichy for nine years during which period he was involved in the project management function. He subsequently operated a small transportation firm for five years and then took up teaching assignments at various leading business schools in India. He was involved in teaching Operations Management, Supply Chain Management, Operations Research, Business Statistics, and Business Analytics. He has been very innovative in all his teaching assignments at Anna University, Chennai; BITS-Pilani; Sona College of Technology, Salem; Adhiyamaan College of Engineering, Hosur; Sharda University, Greater Noida, UP, India.

Preface to the first edition

This book offers the reader with basic concepts in R programming for time series forecasting. The tools covered are Simple Moving Average (SMA), Exponential Moving Average (EMA), HoltWinter's model, Auto Regressive Integrated Moving Average (ARIMA), SARIMA (Seasonal ARIMA), and Dynamic Regression or ARIMAX.

The residuals analysis is an important aspect of time series forecasting and tools like qqplot, Cumulative periodogram (cpgram), and Box test have been used for this purpose throughout the book. Proper residual analysis will ensure model validity and accuracy of prediction.

Knowledge of Business Statistics and R programming is prerequisite for this book.

"This book is dedicated to my daughter Indumathi and son Elanchezhiyan"

The author can be contacted at: arg1962@gmail.com

Chapter 01
Simple Moving Average (SMA)

Simple Moving Average (SMA): BSE500825 ClosePrice

The R code given below makes use of TTR package in R for finding the simple moving average of BSE (Bombay Stock Exchange) equity stock of a leading biscuits manufacturing company in India. First of all download TTR package from R-CRAN website and install it into the R platform using the command line: 'install.packages("TTR", dependencies=TRUE)'.

The dataset holds the stock price and other trade related details. The dataset used for demonstrating holds stock price from 01-Jan-19 to 20-JJan-20 for 259 trading days for Britannia with BSE code 500825. The dataset holds details on the daily price movement (OpenPrice, HighPrice, LowPrice, and ClosePrice) apart from other details like trade volume, price spread, etc. The 4-day as well as 20-day simple moving average has been estimated for this company's stock price. A line graph depicting the ClosePrice of the stock has been shown in Figure 1.1.

```
> library(TTR)
> input<-read.csv("500825.csv", sep=",", header=T)
> datadf<-data.frame(TrDate=as.Date(input$Date,format="%d-%b-%Y"),
OpenPrice=input$Open.Price, HighPrice=input$High.Price, LowPrice=input$Low.Price,
ClosePrice=input$Close.Price,volume=input$No.of.Shares, spCloseOpen=input$Spread.Close.Open)

> abc<-datadf[order(datadf$TrDate, decreasing=F),]
> rownames(abc)<-c(1:nrow(datadf))
> str(abc)
'data.frame':   259 obs. of  7 variables:
 $ TrDate     : Date, format: "2019-01-01" "2019-01-02" ...
 $ OpenPrice  : num  3130 3087 3116 3141 3125 ...
 $ HighPrice  : num  3132 3130 3177 3160 3155 ...
 $ LowPrice   : num  3084 3075 3094 3107 3125 ...
 $ ClosePrice : num  3087 3116 3140 3122 3139 ...
 $ volume     : int  60219 8982 13194 5821 7025 20449 51425 9251 72716 8180 ...
 $ spCloseOpen: num  -42.8 28.7 24.4 -19.5 13.6 ...

> abc$TrDate<-as.character(abc$TrDate, format="%d-%b-%y")
```

```
> head(abc)
```

	TrDate	OpenPrice	HighPrice	LowPrice	ClosePrice	volume	spCloseOpen
1	1-Jan-19	3129.90	3132.35	3084.35	3087.15	60219	-42.75
2	2-Jan-19	3087.15	3130.00	3075.20	3115.85	8982	28.70
3	3-Jan-19	3115.80	3177.45	3094.00	3140.20	13194	24.40
4	4-Jan-19	3141.10	3159.80	3107.00	3121.60	5821	-19.50
5	7-Jan-19	3125.25	3154.80	3125.10	3138.85	7025	13.60
6	8-Jan-19	3145.00	3148.80	3085.05	3093.00	20449	-52.00

```
> tail(abc)
```

	TrDate	OpenPrice	HighPrice	LowPrice	ClosePrice	volume	spCloseOpen
254	13-Jan-20	2990.00	3053.45	2989.45	3043.60	8918	53.60
255	14-Jan-20	3050.50	3118.15	3040.60	3104.80	15339	54.30
256	15-Jan-20	3120.00	3133.75	3094.85	3112.15	5567	-7.85
257	16-Jan-20	3115.00	3159.80	3115.00	3149.50	114587	34.50
258	17-Jan-20	3157.95	3167.85	3117.10	3127.15	5798	-30.80
259	20-Jan-20	3127.00	3140.60	3105.90	3110.30	29121	-16.70

```
 > tsClosePrice<-ts(abc$ClosePrice, start=1)

plot.ts(tsClosePrice, xaxt="n", type="n", col=2, xlab="Date", ylab= "Close Price", main="bse500825
ClosePrice plot")
#command line below will use TrDate as labels for x-axis
axis(1, at=1:nrow(abc), labels=abc$TrDate)
#command line below will add gridlines for y-axis
axis(2, tck=1, col.ticks="lightgray")
#for loop used for manually creating x-axis gridlines
#abline function could be used for drawing straight line on the existing plot
for (i in 0:10){
abline(v=i*nrow(abc)/10, lty=2, col="lightgray")
}
> lines(tsClosePrice, col="red")
```

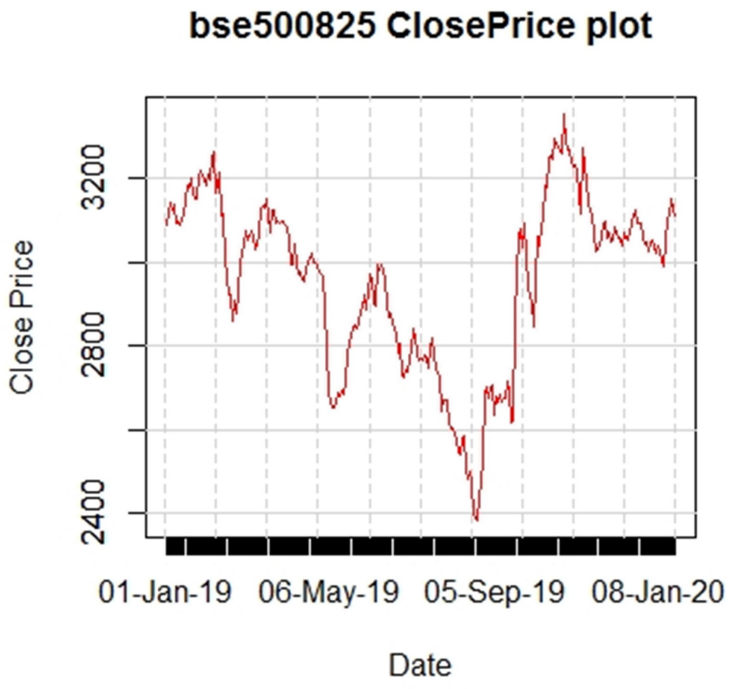


Figure 1.1 ClosePrice ts plot

> sma4<-SMA(tsClosePrice, n=4) # simple moving average with period 4
> sma20<-SMA(tsClosePrice, n=20) # simple moving average with period 20
> xyz<-data.frame(TrDate=abc$TrDate, ClosePrice=abc$ClosePrice, volume=abc$volume, sma4, sma20)

> head(xyz, 24)

	TrDate	ClosePrice	volume	sma4	sma20
1	1-Jan-19	3087.15	60219	NA	NA
2	2-Jan-19	3115.85	8982	NA	NA
3	3-Jan-19	3140.20	13194	NA	NA
4	4-Jan-19	3121.60	5821	3116.20	NA
5	7-Jan-19	3138.85	7025	3129.13	NA
6	8-Jan-19	3093.00	20449	3123.41	NA
7	9-Jan-19	3094.90	51425	3112.09	NA
8	10-Jan-19	3088.40	9251	3103.79	NA
9	11-Jan-19	3099.15	72716	3093.86	NA
10	14-Jan-19	3111.90	8180	3098.59	NA
11	15-Jan-19	3182.50	12129	3120.49	NA
12	16-Jan-19	3177.20	9115	3142.69	NA

13	17-Jan-19	3201.35	7440	3168.24	NA
14	18-Jan-19	3163.20	9444	3181.06	NA
15	21-Jan-19	3148.75	7655	3172.63	NA
16	22-Jan-19	3150.95	13640	3166.06	NA
17	23-Jan-19	3203.15	11787	3166.51	NA
18	24-Jan-19	3216.00	6631	3179.71	NA
19	25-Jan-19	3207.75	14366	3194.46	NA
20	28-Jan-19	3196.10	6617	3205.75	3146.90
21	29-Jan-19	3180.75	3791	3200.15	3151.58
22	30-Jan-19	3206.95	4589	3197.89	3156.13
23	31-Jan-19	3195.95	18070	3194.94	3158.92
24	1-Feb-19	3248.65	15556	3208.08	3165.27

```
> plot.ts(tsClosePrice, xaxt="n", type="n", col=2, xlab="Date", ylim=c(min(xyz$ClosePrice)*0.95,
max(xyz$ClosePrice)*1.05), ylab= "Close Price", main="ClosePrice with sma4 and sma20")
> axis(1, at=1:nrow(abc), labels=abc$TrDate)
> axis(2, tck=1, col.ticks="lightgray")
> for (i in 0:10){
+ abline(v=i*nrow(abc)/10, lty=2, col="lightgray")
+ }

> lines(tsClosePrice, col="red")
> lines(xyz$sma4, type="l", col="blue")
> lines(xyz$sma20, type="l", col="green")
> legend("topleft", lty=c(1,1,1), col=c('red', 'blue', 'green'), legend=c("Close Price", "SMA4",
+ "SMA20"), bty="n", cex=0.6)

> Resid<-tsClosePrice - sma4
```

The Figure 1.2 shows the simple moving average (SMA) with period equal to 4 & 20. The simple moving average with period n=20 is much smoother in representing the data with less influence of peaks and valleys in stock price. Such moving average will be useful in understanding and predicting the trend in the stock price. The moving average with shorter period like n=4 is more responsive to the variations in the stock price. Such responsive moving average will be useful for responding to fast changes in stock price in trading. Moving average with lesser period will provide trade signals much faster.

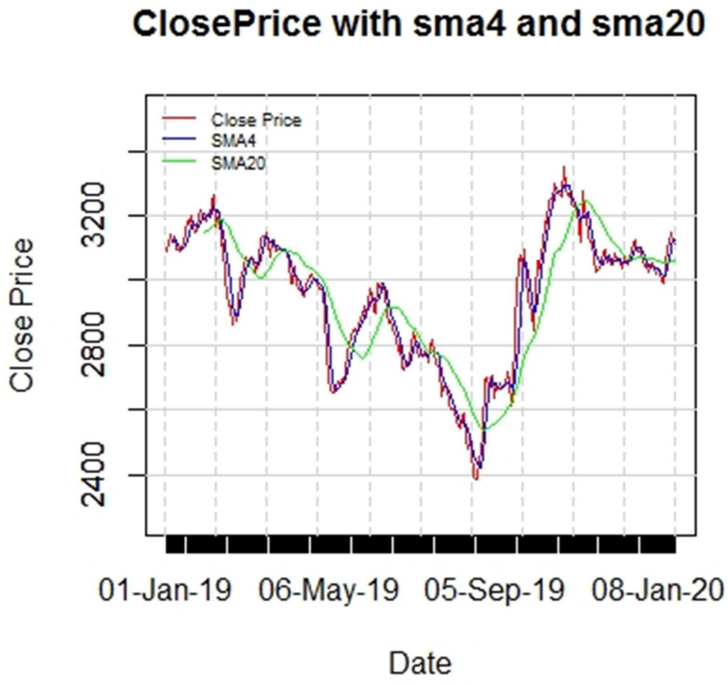


Figure 1.2 ClosePrice with sma4 and sma20 combo plot

> plot(Resid, main="bse500825 sma4 residuals")

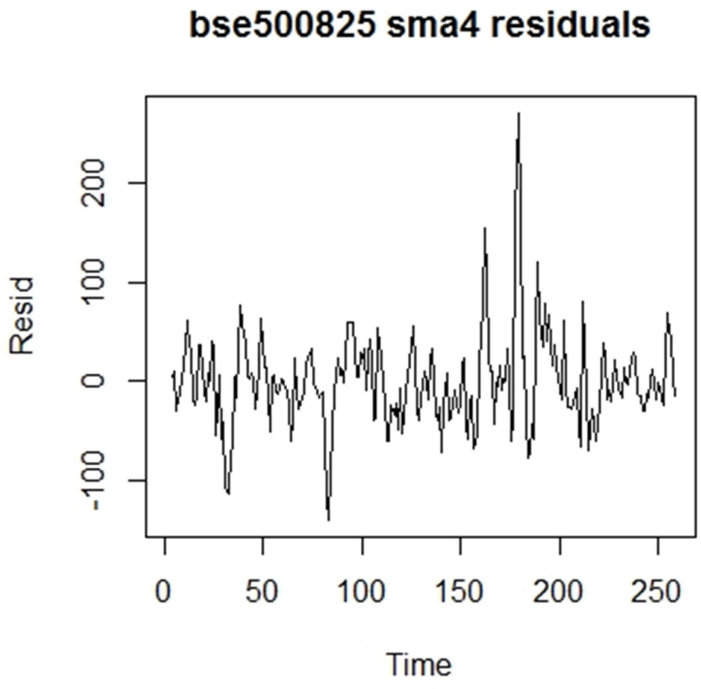


Figure 1.3 bse500825 sma4 residuals plot

> hist(Resid, main="bse500825 sma4 residuals histogram")

The Figure 1.3 depicts the model residuals as line graph and the same has been plotted as histogram in Figure 1.4. Both indicate that the residuals are not white noise.

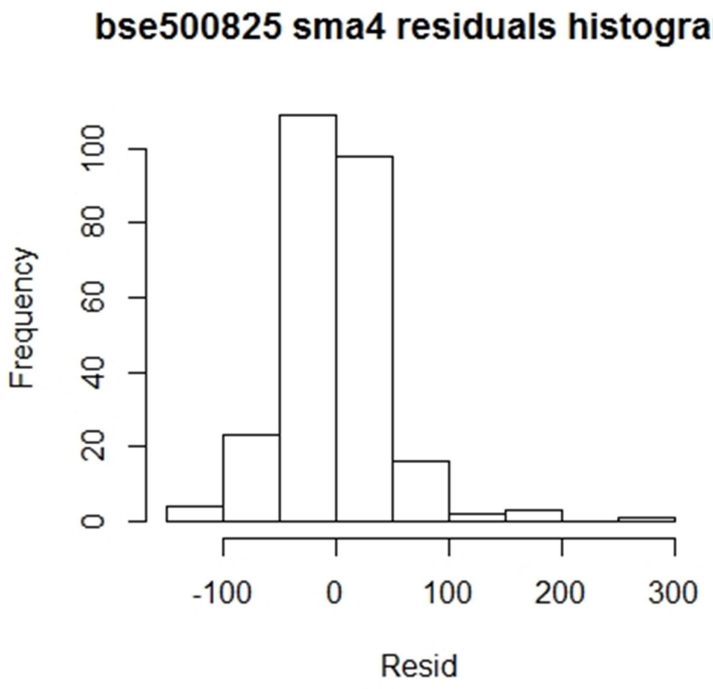


Figure 1.4 histogram for sma4 residuals

> Box.test(Resid, lag=20, type="Ljung-Box")

Box-Ljung test

data: Resid

X-squared = 154.58, df = 20, p-value < 2.2e-16

The Box test p value is much less than 0.05. The Box test has clearly indicated that the residuals of the model fitted does not follow normal distribution and the residuals are not white noise. The dataset requires some other model to represent it in proper perspective. The qqplot shown in Figure 1.5 and cpgram shown in Figure 1.6 also indicate that the model is not adequate.

> qqnorm(Resid, main="bse500825 sma4 residuals qqplot")

> qqline(Resid)

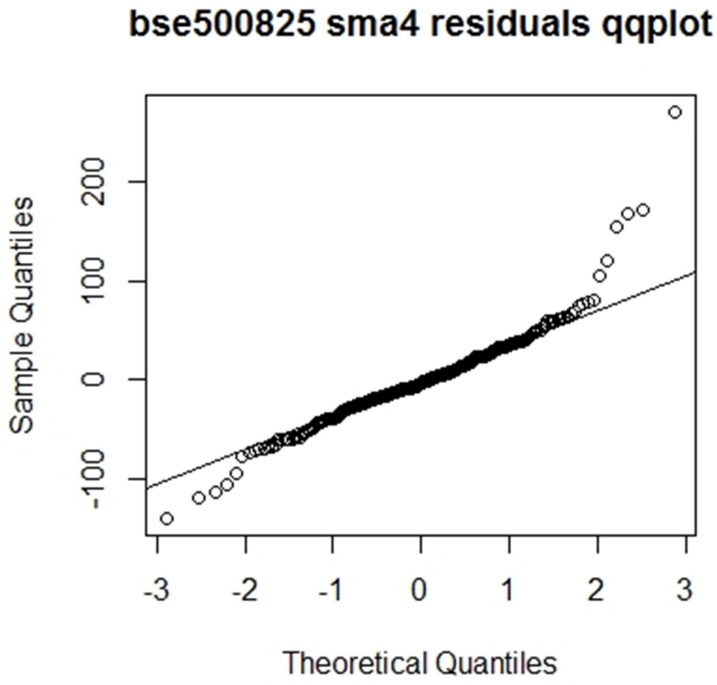


Figure 1.5 sma4 residuals qqplot

> cpgram(Resid, main="bse500825 sma4 residuals cpgram")

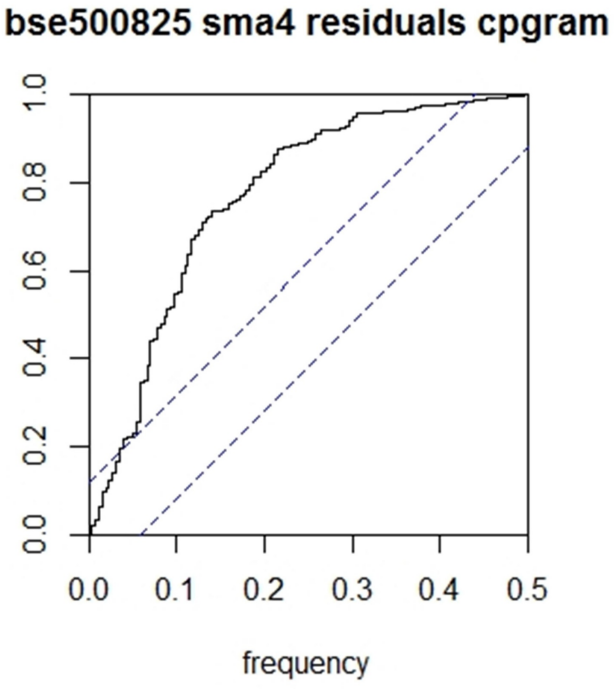


Figure 1.6 sma4 residuals cpgram

Chapter 02
Exponential Moving Average

Exponential Moving Average: BSE500790 ClosePrice

The following R code produces the exponential moving average with period equal to 4 & 20 for the dataset '500790.csv'. The analysis makes use of TTR package and this package should be installed in the R platform. The Figure 2.1 shows the daily stock 'ClosePrice' from 1-Jan-19 to 20-Jan-20 for 259 trading days for Nestle with BSE code 500790.

```
> library(TTR)

 > input<-read.csv("500790.csv", sep=",", header=T)

> datadf<-data.frame(TrDate=as.Date(input$Date,format="%d-%b-%Y"),
OpenPrice=input$Open.Price, HighPrice=input$High.Price, LowPrice=input$Low.Price,
ClosePrice=input$Close.Price,volume=input$No.of.Shares, spCloseOpen=input$Spread.Close.Open)

> abc<-datadf[order(datadf$TrDate, decreasing=F),]
> rownames(abc)<-c(1:nrow(datadf))

> str(abc)
'data.frame':   259 obs. of  7 variables:
 $ TrDate     : Date, format: "2019-01-01" "2019-01-02" ...
 $ OpenPrice  : num  11112 11002 11133 11200 10870 ...
 $ HighPrice  : num  11227 11145 11499 11200 10940 ...
 $ LowPrice   : num  10991 10941 11050 10667 10719 ...
 $ ClosePrice : num  11062 11041 11191 10878 10779 ...
 $ volume     : int  6029 1702 28106 14436 5916 13964 4675 8937 1654 7030 ...
 $ spCloseOpen: num  -50.1 39.5 57.6 -321.7 -90.5 ...

> abc$TrDate<-as.character(abc$TrDate, format="%d-%b-%y")
```

The head and tail function provide a portion of the total dataset. The head function provides the first six (default) rows in the numeric vector or data frame. The number of records displayed could be altered by giving suitable change in the function argument. Similarly, the tail function provides the last six rows of the data frame or last six records in numeric vector.

```
> head(abc)
```

	TrDate	OpenPrice	HighPrice	LowPrice	ClosePrice	volume	spCloseOpen
1	1-Jan-19	11112.00	11226.85	10991.05	11061.90	6029	-50.10
2	2-Jan-19	11001.60	11145.00	10941.15	11041.05	1702	39.45
3	3-Jan-19	11133.45	11499.00	11050.00	11191.10	28106	57.65
4	4-Jan-19	11200.00	11200.00	10666.90	10878.30	14436	-321.70
5	7-Jan-19	10870.00	10940.25	10719.25	10779.45	5916	-90.55
6	8-Jan-19	10853.65	11232.90	10701.05	11016.80	13964	163.15

```
> tail(abc)
```

	TrDate	OpenPrice	HighPrice	LowPrice	ClosePrice	volume	spCloseOpen
254	13-Jan-20	14728.30	14735.45	14604.30	14661.90	747	-66.40
255	14-Jan-20	14662.00	14900.00	14615.00	14869.05	1388	207.05
256	15-Jan-20	14868.00	14950.00	14738.40	14867.60	874	-0.40
257	16-Jan-20	14869.00	15399.00	14860.00	15347.25	5450	478.25
258	17-Jan-20	15399.00	15599.00	15365.00	15441.35	2271	42.35
259	20-Jan-20	15436.00	15474.70	15311.00	15419.80	1805	-16.20

```
> tsClosePrice<-ts(abc$ClosePrice, start=1)

> plot.ts(tsClosePrice, xaxt="n", type="n", col=2, xlab="Date", ylab= "Close Price", main="bse500790
+ ClosePrice plot")
> axis(1, at=1:nrow(abc), labels=abc$TrDate)
> axis(2, tck=1, col.ticks="lightgray")

> for (i in 0:10){
+ abline(v=i*nrow(abc)/10, lty=2, col="lightgray")
+ }

> lines(tsClosePrice, col="red")
```

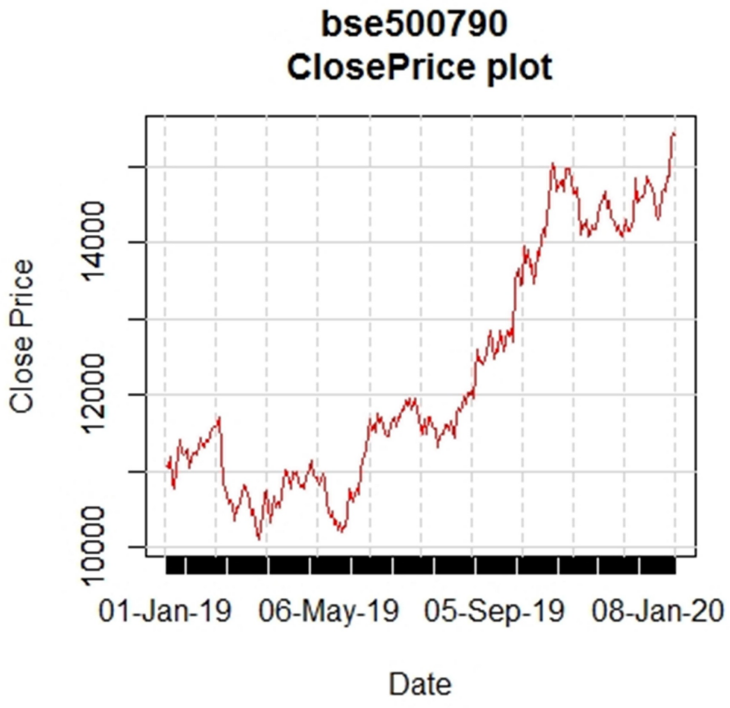


Figure 2.1 bse500790 ClosePrice ts plot

> ema4<-EMA(tsClosePrice, n=4) # exponential moving average with period 4
> ema20<-EMA(tsClosePrice, n=20) # exponential moving average with period 20
> xyz<-data.frame(TrDate=abc$TrDate, ClosePrice=abc$ClosePrice, volume=abc$volume, ema4, ema20)

> head(xyz, 24)

	TrDate	ClosePrice	volume	ema4	ema20
1	1-Jan-19	11061.90	6029	NA	NA
2	2-Jan-19	11041.05	1702	NA	NA
3	3-Jan-19	11191.10	28106	NA	NA
4	4-Jan-19	10878.30	14436	11043.09	NA
5	7-Jan-19	10779.45	5916	10937.63	NA
6	8-Jan-19	11016.80	13964	10969.30	NA
7	9-Jan-19	11252.20	4675	11082.46	NA
8	10-Jan-19	11416.85	8937	11216.22	NA
9	11-Jan-19	11265.05	1654	11235.75	NA
10	14-Jan-19	11224.65	7030	11231.31	NA
11	15-Jan-19	11286.85	1254	11253.53	NA
12	16-Jan-19	11048.85	4351	11171.66	NA

13	17-Jan-19	11169.90	1465	11170.95	NA
14	18-Jan-19	11241.05	2929	11198.99	NA
15	21-Jan-19	11231.95	1762	11212.18	NA
16	22-Jan-19	11215.60	5153	11213.55	NA
17	23-Jan-19	11310.55	1461	11252.35	NA
18	24-Jan-19	11430.55	2263	11323.63	NA
19	25-Jan-19	11343.80	1314	11331.70	NA
20	28-Jan-19	11325.45	2528	11329.20	11186.59
21	29-Jan-19	11407.10	1749	11360.36	11207.60
22	30-Jan-19	11394.15	1064	11373.88	11225.36
23	31-Jan-19	11469.80	5711	11412.25	11248.64
24	1-Feb-19	11568.25	2546	11474.65	11279.08

```
> plot.ts(tsClosePrice, xaxt="n", type="n", col=2, xlab="Date", ylim=c(min(xyz$ClosePrice)*0.95,
max(xyz$ClosePrice)*1.05), ylab= "Close Price", main="ClosePrice with ema4 and ema20")
> axis(1, at=1:nrow(abc), labels=abc$TrDate)
> axis(2, tck=1, col.ticks="lightgray")
> for (i in 0:10){
+ abline(v=i*nrow(abc)/10, lty=2, col="lightgray")
+ }

> lines(tsClosePrice, col="red")
> lines(xyz$ema4, type="l", col="blue")
> lines(xyz$ema20, type="l", col="green")
> legend("topleft", lty=c(1,1,1), col=c("red", "blue", "green"), legend=c("Close Price", "EMA4",
+ "EMA20"), bty="n", cex=0.6)


> Resid<-tsClosePrice - ema4
```

The Figure 2.2 shows the ClosePrice, ema4, ema20 combo plot. This graph would clearly demonstrate the effect of moving average period selection on the level of smoothing achieved. The larger the smoothing period the more smoother will be the smoothed line. The larger moving average period will smooth out peaks and lows resulting in a smoother graph.

The Figure 2.3 shows the residuals in graphical form and the residuals graph exhibits wide fluctuations in value. The residuals has also been shown as histogram in Figure 2.4

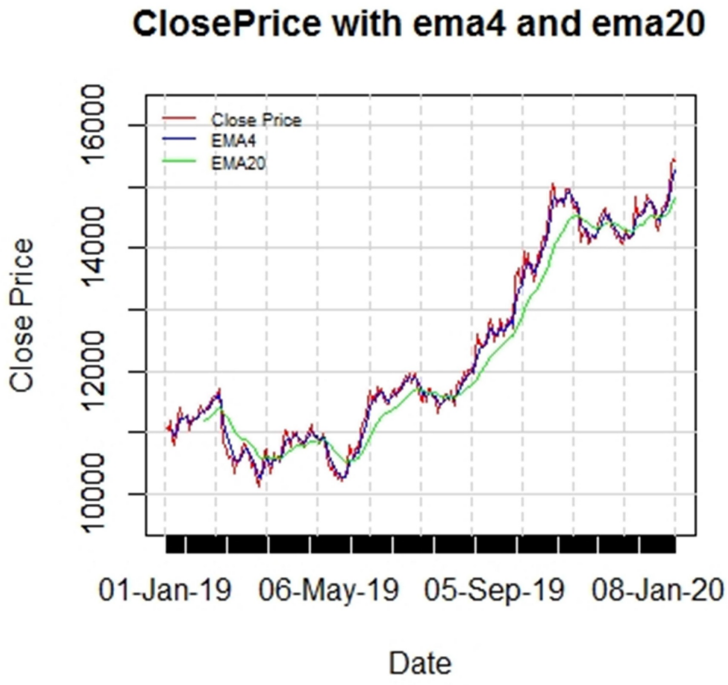


Figure 2.2 ClosePrice with ema4 and ema20 combo plot

> plot(Resid, main="bse500790 ema4 residuals")

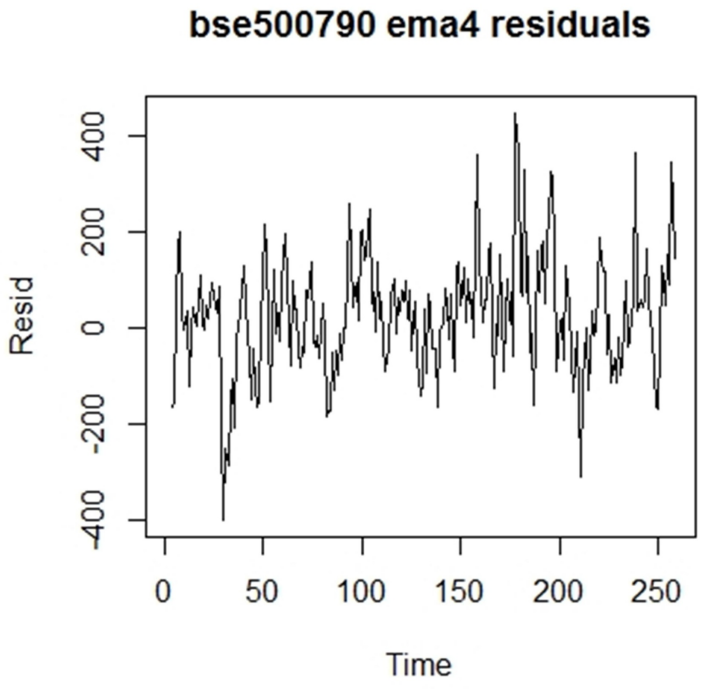


Figure 2.3 ema4 residuals plot

> hist(Resid, main="bse500790 ema4 residuals histogram")

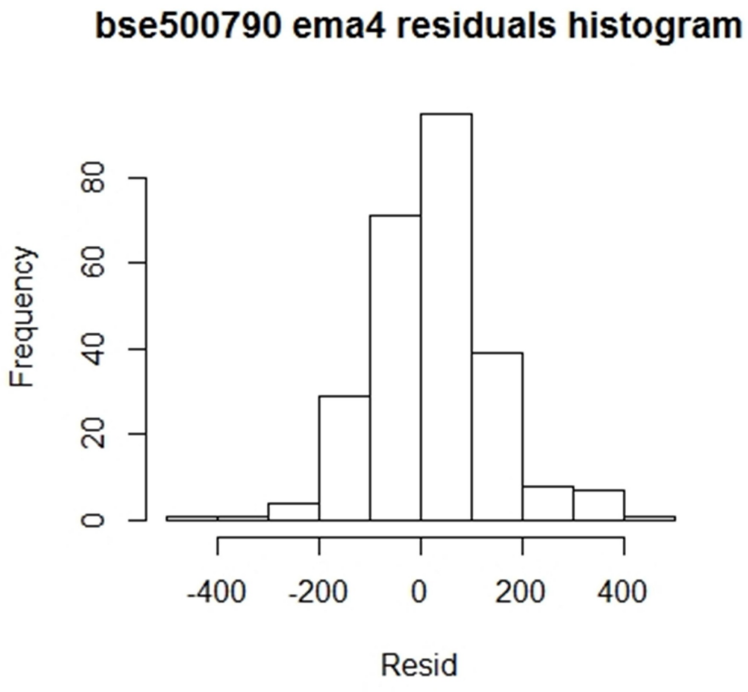


Figure 2.4 ema4 residuals histogram

> Box.test(Resid, lag=20, type="Ljung-Box")

Box-Ljung test

data: Resid

X-squared = 137.53, df = 20, p-value < 2.2e-16

The Box test p value is much less than 0.05. The Box test has clearly indicated that the residuals of the model fitted does not follow normal distribution and the residuals are not white noise. The dataset requires some other better model to represent in proper perspective.

The qqplot shown in Figure 2.5 and cpgram shown in Figure 2.6 also indicate that the residuals is not white noise and the EMA model is not an adequate representation of the 500790.csv dataset.

> qqnorm(Resid, main="bse500790 ema4 residuals qqplot")

> qqline(Resid)

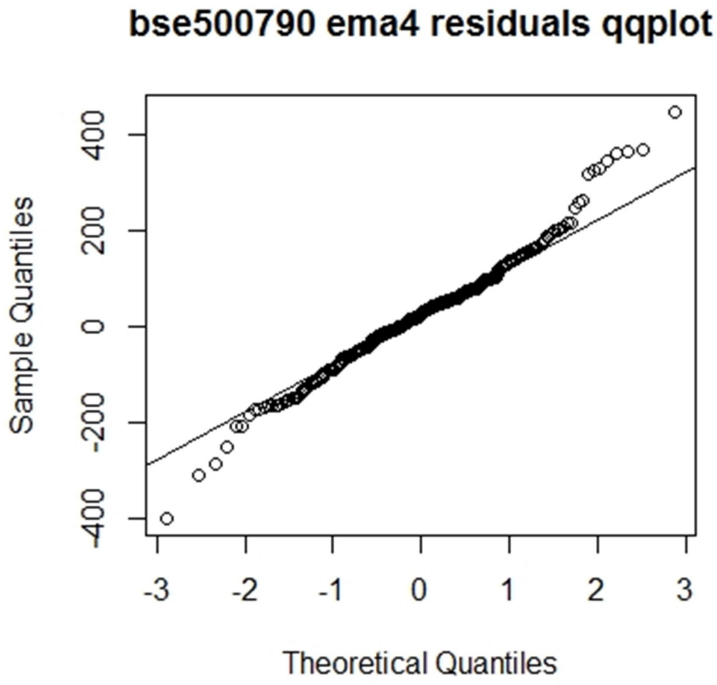


Figure 2.5 ema4 residuals qqplot

> cpgram(Resid, main="bse500790 ema4 residuals cpgram")

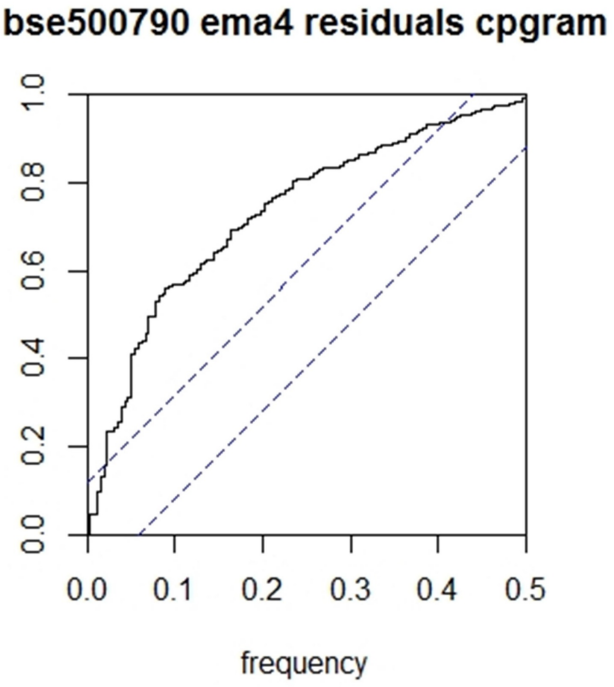


Figure 2.6 ema4 residuals cpgram

Chapter 03
HoltWinter's Model without Trend

The Brent crude oil price per barrel in dollars from 2-Jan-19 to 21-Jan-20 has been used for developing HoltWinter's model without trend. Essentially, this model is same as the simple exponential smoothing technique for a defined alpha, the smoothing coefficient. The Figure 3.1 shows the Brent crude price in dollars as line graph.

```
> #HW without trend: Brent Crude Price

> brent<-read.csv("brentcrude.csv", sep=",", header=T)

> head(brent)
    TrDate     dollarsPB
1    2-Jan-19    54.06
2    3-Jan-19    53.23
3    4-Jan-19    55.64
4    7-Jan-19    57.10
5    8-Jan-19    56.91
6    9-Jan-19    59.46

> tail(brent)
       TrDate     dollarsPB
265   14-Jan-20    64.45
266   15-Jan-20    63.29
267   16-Jan-20    64.63
268   17-Jan-20    64.05
269   20-Jan-20    64.63
270   21-Jan-20    63.66

> dollarsPB<-ts(brent$dollarsPB, start=1)
> ts.plot(dollarsPB, main="Brent crude price in dollars")
```

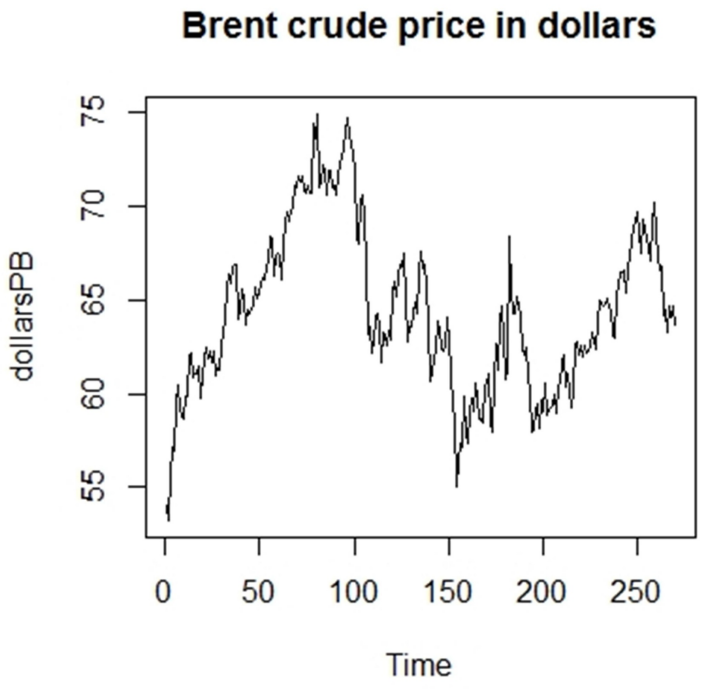


Figure 3.1 Brent crude price in dollars

```
> hwexp<-HoltWinters(dollarsPB, beta=F, gamma=F)

> head(hwexp$fitted)
         xhat        level
[1,] 54.06000 54.06000
[2,] 53.25956 53.25956
[3,] 55.55522 55.55522
[4,] 57.04498 57.04498
[5,] 56.91481 56.91481
[6,] 59.36935 59.36935
```

The Figure 3.2 shows the HoltWinter's model observed versus fitted price for the Brent crude and the Figure 3.3 shows the fitted model residuals as line graph. The residual analysis tools qqplot and cpgram have been shown in Figure 3.4 and 3.5 respectively.

```
> plot(hwexp, main="Brent crude price observed vs fitted")
```

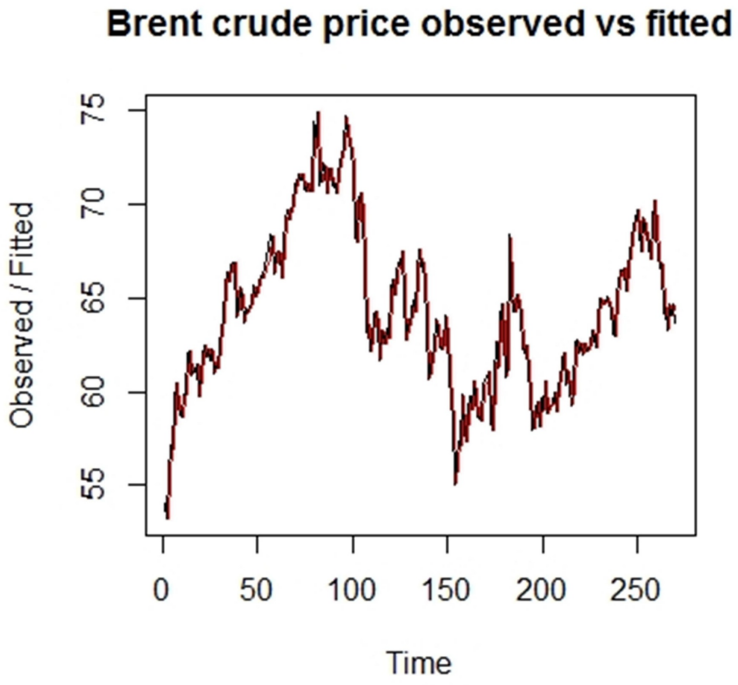


Figure 3.2 Brent crude price observed vs fitted

> ts.plot(residuals(hwexp), main="Brent crude HW with out trend")

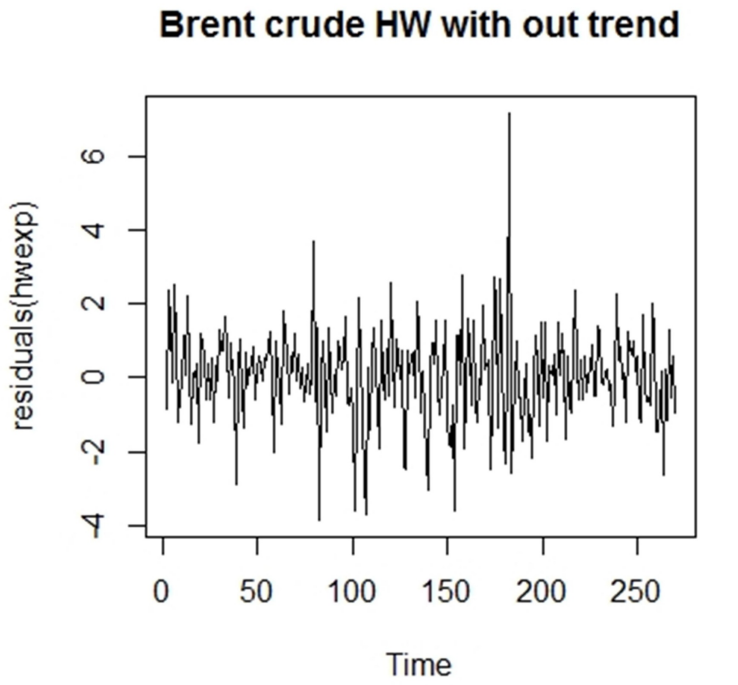


Figure 3.3 HW model residuals plot

```
> qqnorm(residuals(hwexp), main="qqplot of residuals")
> qqline(residuals(hwexp))
```

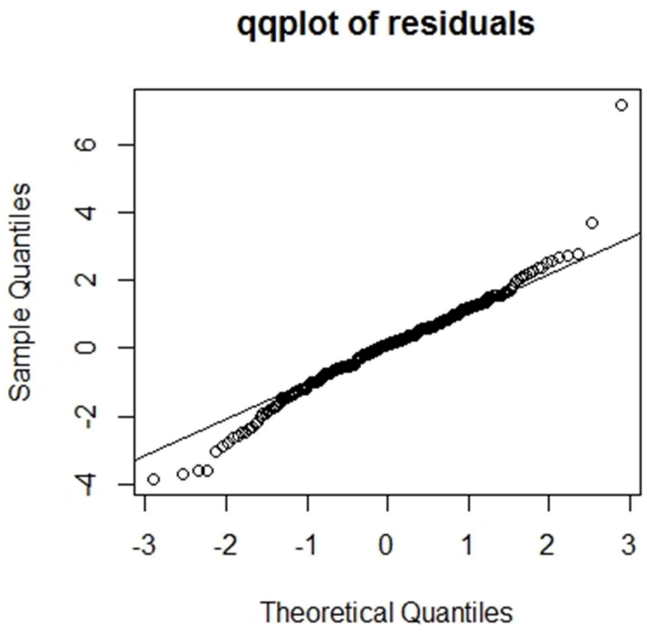


Figure 3.4 qqplot of HW model residuals

```
> cpgram(residuals(hwexp))
```

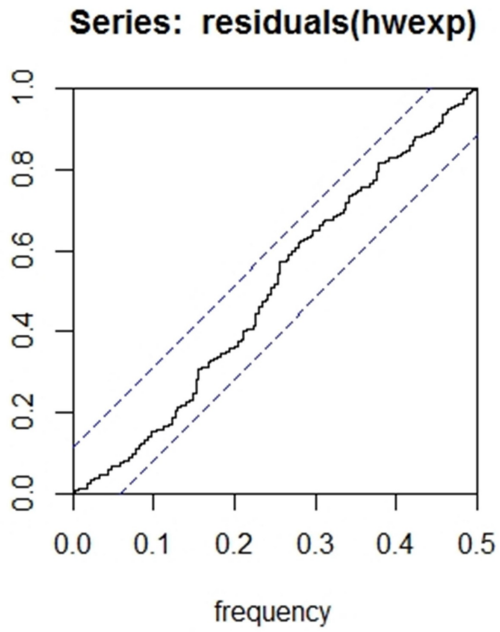


Figure 3.5 cpgram of HW model residuals

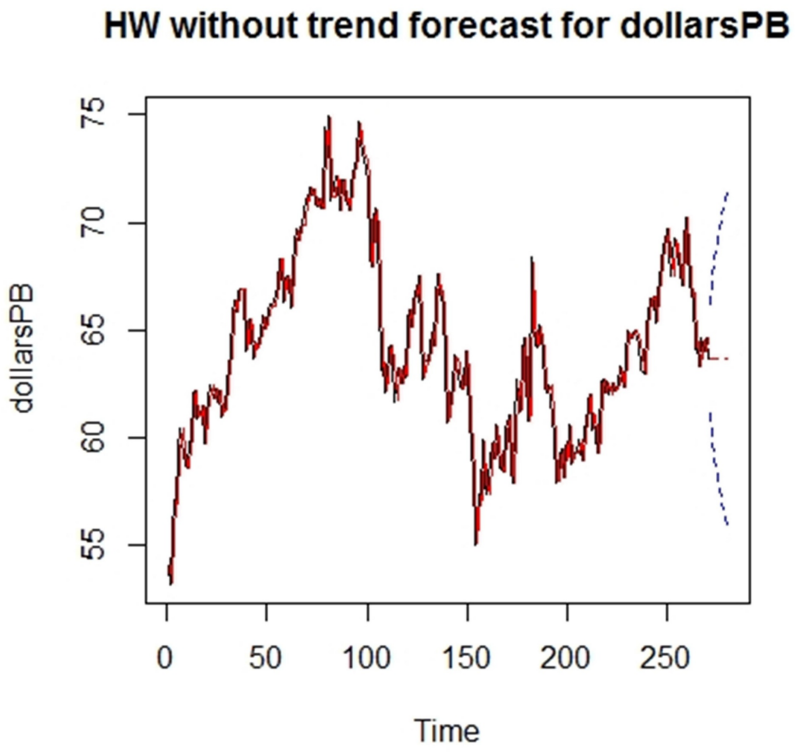
```
> Box.test(residuals(hwexp), lag=20, type="Ljung-Box")

        Box-Ljung test

data:  residuals(hwexp)
X-squared = 18.15, df = 20, p-value = 0.5775
```

The analysis of the residuals clearly indicates that the residuals are white nois and they are under control. This has been confirmed by the Box test results also which indicates that the p-value is 0.5775. This value is higher than the level of significance 0.05 and hence the null hypothesis that the residuals are white noise could not be rejected.

```
> fpre<-predict(hwexp, n.ahead=10, prediction.interval=TRUE, level=0.95)
> fpre
Time Series:
Start = 271
End = 280
Frequency = 1
          fit            upr          lwr
271 63.69384 66.24944 61.13824
272 63.69384 67.24422 60.14345
273 63.69384 68.01580 59.37187
274 63.69384 68.66913 58.71854
275 63.69384 69.24611 58.14156
276 63.69384 69.76853 57.61914
277 63.69384 70.24945 57.13822
278 63.69384 70.69742 56.69025
279 63.69384 71.11841 56.26926
280 63.69384 71.51678 55.87089
```

```
> ts.plot(dollarsPB, hwexp$fitted[,1], fpre, lty=c(1,1,2,2,2), col=c(1,2,2,4,4), ylab="dollarsPB",
 main = "HW without trend forecast for dollarsPB")
```

The Figure 3.6 shows the forecast for next ten days along with control limits.



FFigure 3.6 Brent Crude Price forecast

Chapter 04
Holtwinter's Models with Trend

4.1 HoltWinter's model with Trend: BJsales

The BJsales dataset is available in the R platform data library. The Figure 4.1 shows the BJsales and the fitted value by HW with trend model.

```
> #HW model with Trend: BJsales
> BJsales
Time Series:
Start = 1
End = 150
Frequency = 1
  [1]   200.1 199.5 199.4 198.9 199.0 200.2 198.6 200.0 200.3 201.2 201.6 201.5
 [13]  201.5 203.5 204.9 207.1 210.5 210.5 209.8 208.8 209.5 213.2 213.7 215.1
 [25]  218.7 219.8 220.5 223.8 222.8 223.8 221.7 222.3 220.8 219.4 220.1 220.6
 [37]  218.9 217.8 217.7 215.0 215.3 215.9 216.7 216.7 217.7 218.7 222.9 224.9
 [49]  222.2 220.7 220.0 218.7 217.0 215.9 215.8 214.1 212.3 213.9 214.6 213.6
 [61]  212.1 211.4 213.1 212.9 213.3 211.5 212.3 213.0 211.0 210.7 210.1 211.4
 [73   210.0 209.7 208.8 208.8 208.8 210.6 211.9 212.8 212.5 214.8 215.3 217.5
 [85]  218.8 220.7 222.2 226.7 228.4 233.2 235.7 237.1 240.6 243.8 245.3 246.0
 [97]  246.3 247.7 247.6 247.8 249.4 249.0 249.9 250.5 251.5 249.0 247.6 248.8
[109] 250.4 250.7 253.0 253.7 255.0 256.2 256.0 257.4 260.4 260.0 261.3 260.4
[121] 261.6 260.8 259.8 259.0 258.9 257.4 257.7 257.9 257.4 257.3 257.6 258.9
[133] 257.8 257.7 257.2 257.5 256.8 257.5 257.0 257.6 257.3 257.5 259.6 261.1
[145] 262.9 263.3 262.8 261.8 262.2 262.7

> fitsales<-HoltWinters(BJsales, gamma=FALSE)

> fitsales
Holt-Winters exponential smoothing with trend and without seasonal component.

Call:
HoltWinters(x = BJsales, gamma = FALSE)

Smoothing parameters:
 alpha: 1
 beta : 0.2520611
 gamma: FALSE
```

```
Coefficients:
      [,1]
a 262.7000000
b   0.2836854

> head(fitsales$fitted)
        Xhat     level      trend
[1,] 198.9000 199.5 -0.60000000
[2,] 198.9260 199.4 -0.47396944
[3,] 198.4195 198.9 -0.48053073
[4,] 198.6658 199.0 -0.33420150
[5,] 200.2525 200.2  0.05251105
[6,] 198.2360 198.6 -0.36402274

> plot(BJsales, type="l", col=2, xlab="Date", ylab= "BJsales", main="HW with Trend: BJsales")
> axis(2, tck=1, col.ticks="lightgray")
> for (i in 0:10){
+ abline(v=length(BJsales)*i/10, lty=2, col="lightgray")
+ }

> lines(fitsales$fitted[,1], type="l", col='blue')
> legend("topleft", lty=c(1,1), col=c('red', 'blue'), legend=c("BJsales", "fitted value"), bty="n",
+ cex=0.6)
```

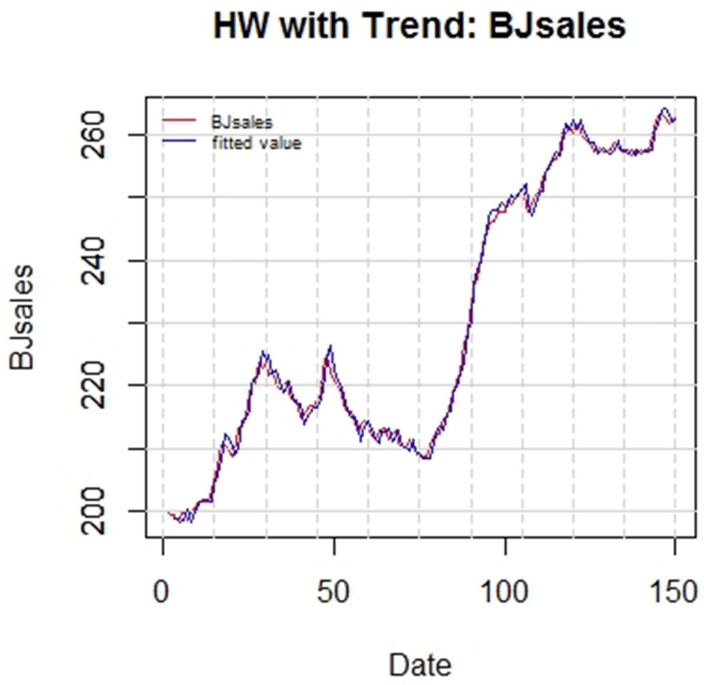


Figure 4.1 ts plot BJsales and fitted value

```
> fpre<-predict(fitsales, n.ahead=10, prediction.interval=TRUE, level=0.95)
> fpre
Time Series:
Start = 151
End = 160
Frequency = 1
         fit         upr         lwr
151 262.9837 265.6726 260.2948
152 263.2674 267.5760 258.9587
153 263.5511 269.4605 257.6416
154 263.8347 271.3992 256.2703
155 264.1184 273.4125 254.8243
156 264.4021 275.5070 253.2972
157 264.6858 277.6841 251.6875
158 264.9695 279.9431 249.9959
159 265.2532 282.2825 248.2238
160 265.5369 284.7004 246.3733
```

```
> ts.plot(BJsales, fitsales$fitted[,1], fpre, lty=c(1,1,2,2,2), col=c(1,2,2,4,4), ylab="BJsales",  main =
"HW with trend forecast for BJsales")
```

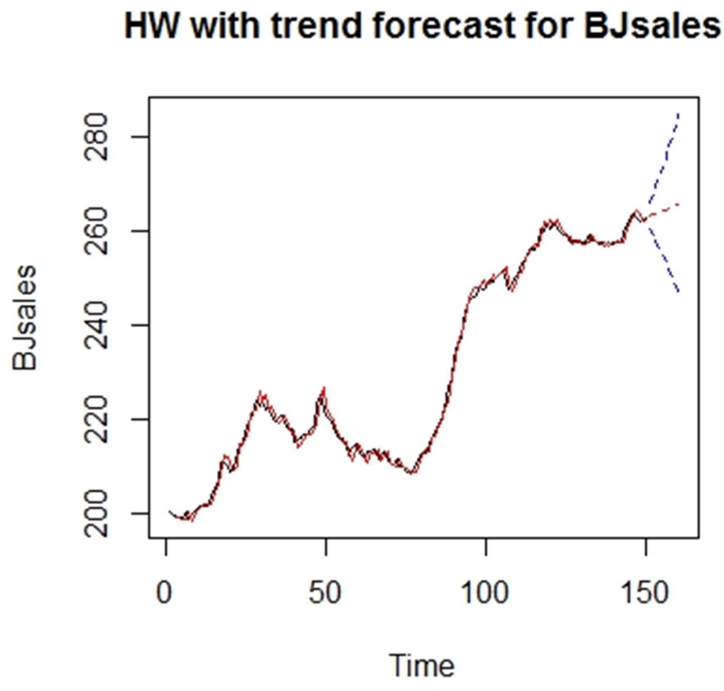


Figure 4.2 ts plot BJsales forecast

```
> Resid<-BJsales - fitsales$fitted[,1]
```

```
> head(Resid)
[1]  0.50000000 -0.02603056  0.58053073  1.53420150 -1.65251105  1.76402274
> plot(Resid, main="Residuals plot: BJsales")
```

The Figure 4.1 shows the Bsales data in graphical form and Figure 4.2 shows the forecast for the next ten periods.

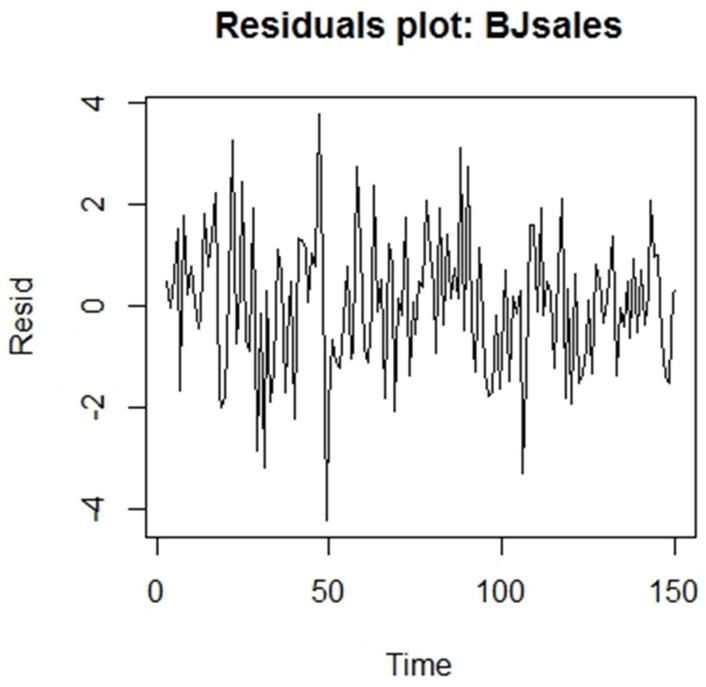


Figure 4.3 Residuals ts plot: BJsales

```
> Box.test(Resid, lag=20, type="Ljung-Box")

        Box-Ljung test

data:  Resid
X-squared = 17.263, df = 20, p-value = 0.6358
```

```
> hist(Resid, prob=T, main="BJsales HW with trend residuals")
> d<-density(Resid)
> lines(d, col=3)
```

The residuals has been plotted as line graph in Figure 4.3 and further the residuals has been shown as histogram in Figure 4.4. The residuals appear to be in control. The qqplot in Figure 4.5 and cpgram in Figure 4.6 also exhibit that the model residuals are well behaved. The Box test for the model residuals confirm that the residuals are white noise since the test p-value is 0.6358.

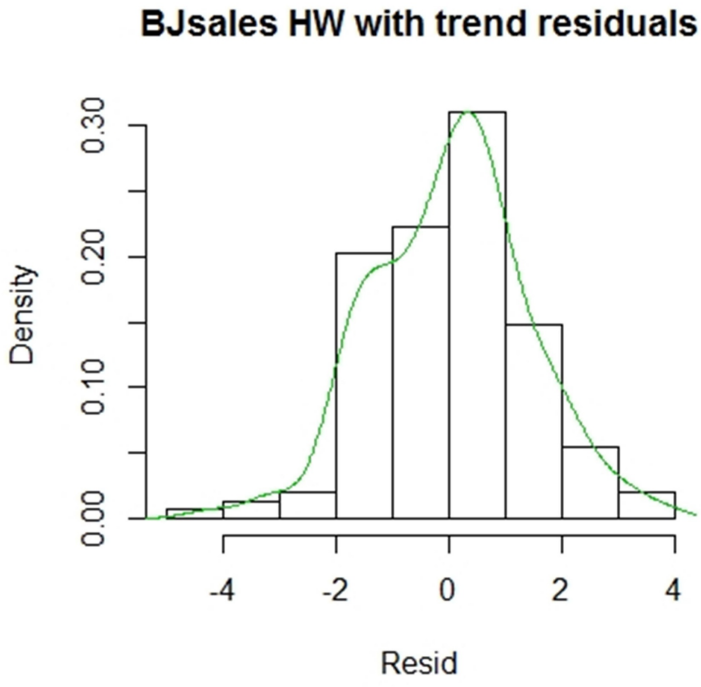


Figure 4.4 histogram of residuals with density

```
> qqnorm(Resid, main="qqplot of residuals: BJsales")
> qqline(Resid)
```

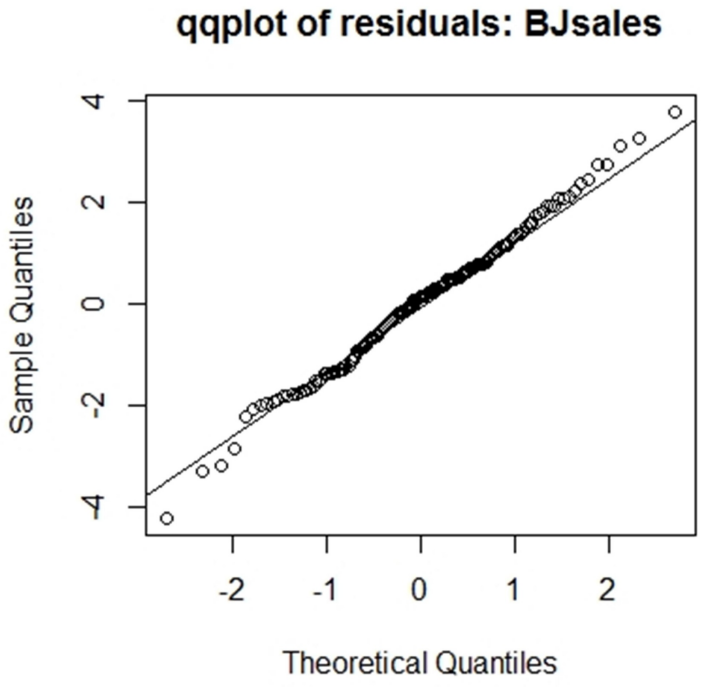


Figure 4.5 qqplot of residuals: BJsales

```
> cpgram(Resid, main="cpgram of residuals: BJsales")
```

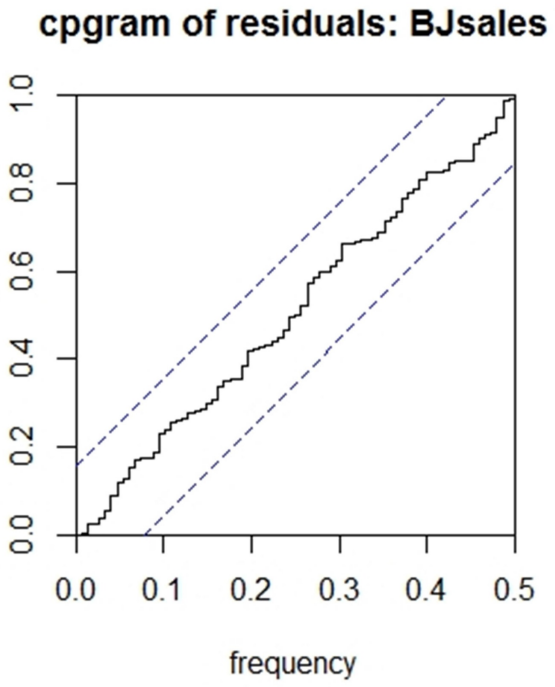


Figure 4.6 cpgram of residuals: BJsales

Chapter 05
HoltWinter's Seasonal Model

5.1 HoltWinter's Additive Seasonal Model: JJ Quarterly Returns Log scale

The dataset JohnsonJohnson is available in base R and can be used straightaway. The dataset holds the Johnson & Johnson Quarterly returns in dollars (JJReturns).

```
> #JohnsonJohnson Quarterly Returns: HW Log scale seasonal model
> tsJohnson<-JohnsonJohnson
> tsJohnson
```

	Qtr1	Qtr2	Qtr3	Qtr4
1960	0.71	0.63	0.85	0.44
1961	0.61	0.69	0.92	0.55
1962	0.72	0.77	0.92	0.60
1963	0.83	0.80	1.00	0.77
1964	0.92	1.00	1.24	1.00
1965	1.16	1.30	1.45	1.25
1966	1.26	1.38	1.86	1.56
1967	1.53	1.59	1.83	1.86
1968	1.53	2.07	2.34	2.25
1969	2.16	2.43	2.70	2.25
1970	2.79	3.42	3.69	3.60
1971	3.60	4.32	4.32	4.05
1972	4.86	5.04	5.04	4.41
1973	5.58	5.85	6.57	5.31
1974	6.03	6.39	6.93	5.85
1975	6.93	7.74	7.83	6.12
1976	7.74	8.91	8.28	6.84
1977	9.54	10.26	9.54	8.73
1978	11.88	12.06	12.15	8.91
1979	14.04	12.96	14.85	9.99
1980	16.20	14.67	16.02	11.61

```
> plot(tsJohnson, ylab="Quarterly Return in Dollars", main="HW Seasonal: JJ Quarterly Returns")
```

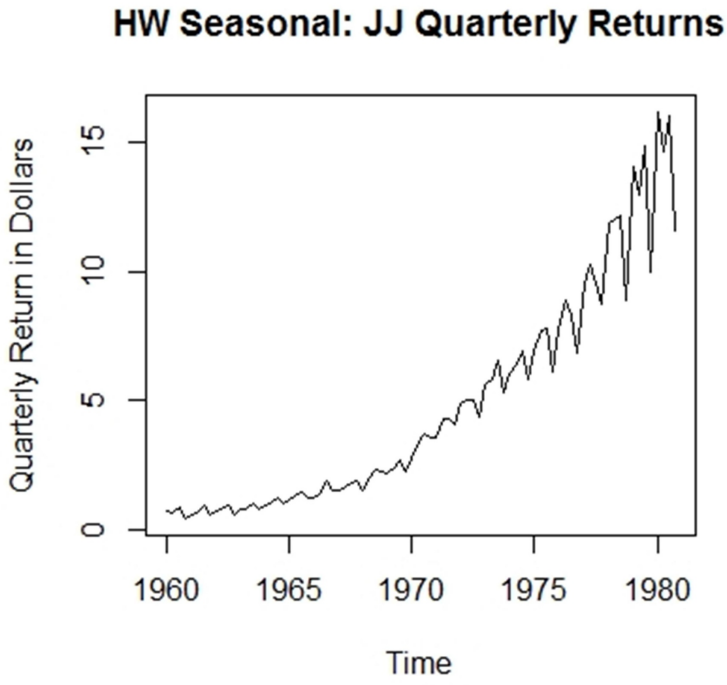


Figure 5.1 JJ quarterly returns ts plot

The dataset has been plotted as line graph and is shown in Figure 5.1 for better understanding. It is quite clear from the line graph that there is uptrend and seasonality present in the Johnson & Johnson quarterly returns. The dataset represents the JJReturns for a period of 21 years from 1960 to 1980. As is evident from the graph, there is an increasing effect of seasonal amplitude. In such cases, we are left with the following two options:

i. Fit HoltWinter's (HW) additive seasonal model by applying Log scale / square root to the dataset
ii. Fit HolWinter's (HW) multiplicative seasonal model at level scale

Now, let us fit an additive seasonality model by applying log scale to the JJReturns data. Later we shall develop multiplicative seasonal model for this dataset.

The Figure 5.2 shows the components of the JJ dataset. It is evident from the components that the there is an uptrend as well as seasonality present in the dataset. The increasing variance as indicated by the pattern in the random component requires logarithmic transformation. The dataset could be dealt with HW additive seasonality model when it has been logarithmic transformed.

```
> tsJohnComponents<-decompose(tsJohnson)
> plot(tsJohnComponents)
```

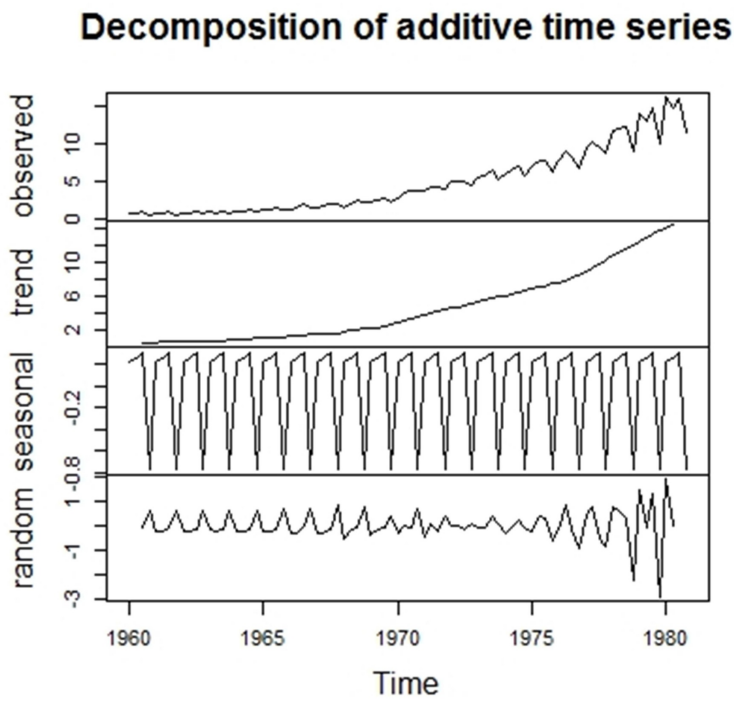


Figure 5.2 JJ Quarterly Returns Components

> fitJJ<-HoltWinters(log(tsJohnson))

> fitJJ
Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:
HoltWinters(x = log(tsJohnson))

Smoothing parameters:
 alpha: 0.242867
 beta : 0.1432447
 gamma: 0.7759241

Coefficients:
        [,1]
a   2.65291318
b   0.03438131
s1 0.21972966
s2 0.10507844
s3 0.16182122
s4 -0.20256826

```
> head(fitJJ$fitted)
          xhat          level         trend         season
[1,] -0.5093127 -0.4955064 0.01755970 -0.03136592
[2,] -0.4021360 -0.4742998 0.01808211  0.05408161
[3,] -0.1154373 -0.4486712 0.01916310  0.31407083
[4,] -0.7382310 -0.4217228 0.02027830 -0.33678652
[5,] -0.3647291 -0.3673474 0.02516253 -0.02254416
[6,] -0.2346284 -0.3333871 0.02642277  0.07233593

> tail(fitJJ$fitted)
        xhat    level     trend     season
[75,] 2.596457 2.442996 0.03519683  0.1182644
[76,] 2.379546 2.502854 0.03872944 -0.1620373
[77,] 2.768976 2.522649 0.03601719  0.2103092
[78,] 2.730432 2.562561 0.03657505  0.1312963
[79,] 2.801238 2.588297 0.03502248  0.1779183
[80,] 2.442896 2.616665 0.03406924 -0.2078382
```

The additive model is: xhat = level + trend + season
(The Generic HW Additive Model is: Ycap = L + T + S)

Where,

'xhat'	is the predicted or fitted value (Ycap)
'level'	is the level component of the prediction (L)
'trend'	is the trend component of the prediction (T)
'season'	is the seasonal component of the prediction (S)

The fitted model has been plotted in Figure 5.3 and the model appears to be appropriate. We shall confirm the appropriateness of the model fit with residuals analysis. The model residuals has been plotted in Figure 5.4 for visual clarity. The qqplot of the residuals has been shown in Figure 5.5 and the Figure 5.6 depicts the cpgram of the model residuals. The residual analysis indicates that the model residuals is white noise and the model is very appropriate.

```
> plot(fitJJ, main="fitJJ model Log scale")
```

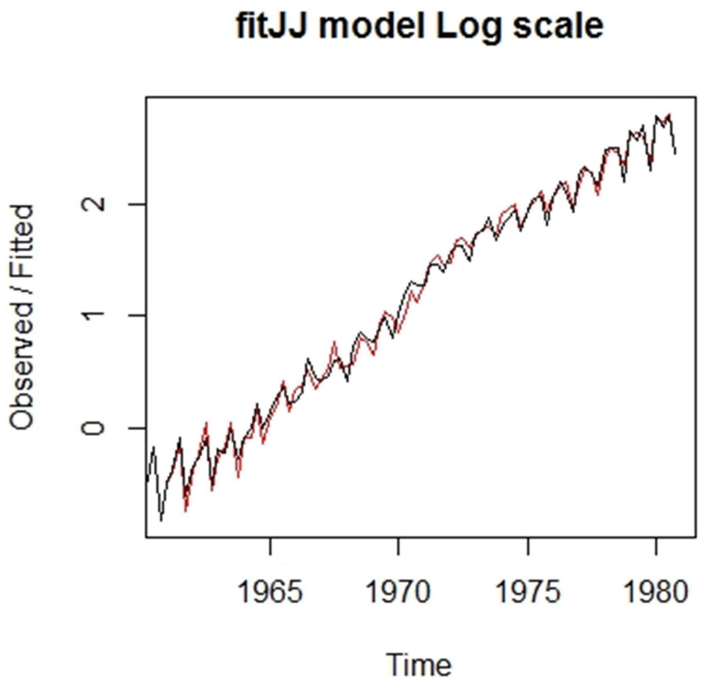


Figure 5.3 fitJJ model Log scale

```
> Resid<-log(tsJohnson)-fitJJ$fitted[,1]
> plot(Resid, main="Residuals plot JJ quarterly returns")      #refer Figure 5.4


> Box.test(Resid, lag=20, type="Ljung-Box")

        Box-Ljung test

data:  Resid
X-squared = 14.33, df = 20, p-value = 0.8134
```

The Box test indicates that the residuals could be considered as white noise since the p-value is 0.8134 which is more than 0.05, the significance level. The critical value of Chi-square, for 20 degrees of freedom and significance level of 0.05, is 31.41 and this value is pretty higher than calculated chi square value which is 14.33. This also indicates that the null hypothesis that the model residuals is white noise could not be rejected.

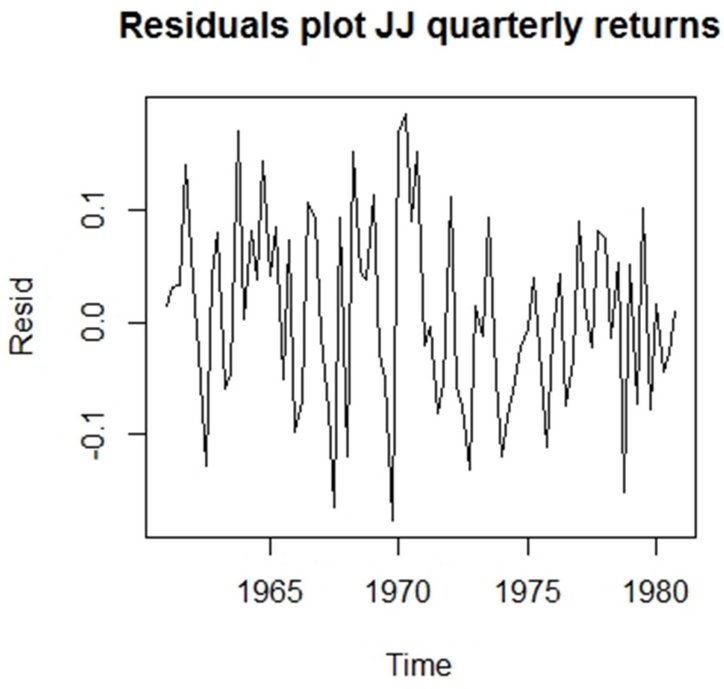


Figure 5.4 JJ Returns Log scale Residuals

```
> qqnorm(Resid, main="JJ Returns Log scale qqplot")
> qqline(Resid)
```

The combo qqnorm and qqplot both together is referred to as qqplot and is shown in Figure 5.5. The qqnorm is a scatter plot between the quantile values of the residuals and the theoretical quantiles for the residuals to be considered as white noise. The qqline is simply the regression line between the actual and theoretical quantiles. The regression line should pass through the origin and should be at 45 degrees angle from either axis for the model residuals being considered as white noise.

The cpgram of the model residuals has been shown in Figure 5.6 and the residuals are in control. The HW seasonal model fitted for the Johnson and Johnson quarterly returns is appropriate.

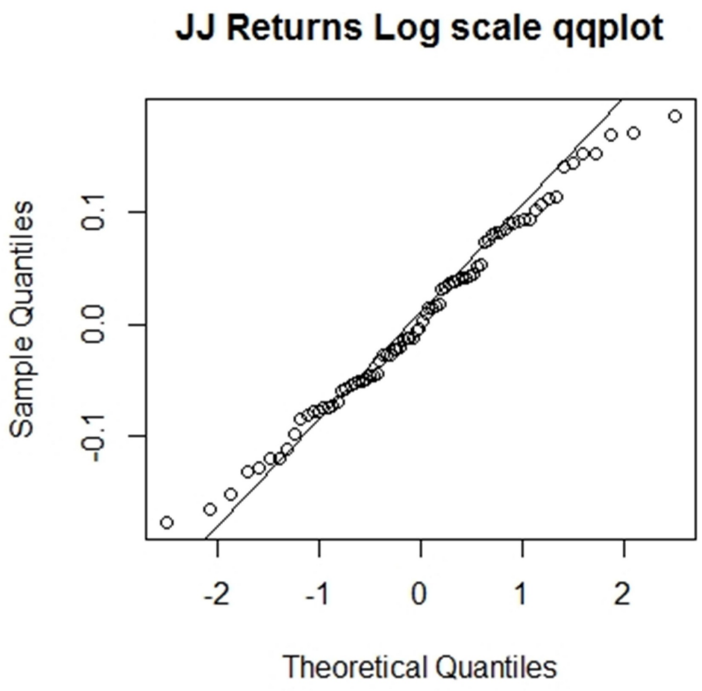


Figure 5.5 JJ Returns Log scale Residuals qqplot

> cpgram(Resid, main="JJ Returns Log scale cpgram") #refer Figure 5.6

> fpre<-predict(fitJJ, n.ahead=12, prediction.interval=TRUE, level=0.95)
> fpre

	fit	upr	lwr
1981 Q1	2.907024	3.073005	2.741044
1981 Q2	2.826754	2.999014	2.654495
1981 Q3	2.917878	3.097775	2.737982
1981 Q4	2.587870	2.776774	2.398967
1982 Q1	3.044549	3.292701	2.796398
1982 Q2	2.964279	3.221895	2.706664
1982 Q3	3.055404	3.323702	2.787105
1982 Q4	2.725395	3.005577	2.445214
1983 Q1	3.182075	3.517282	2.846868
1983 Q2	3.101805	3.449482	2.754128
1983 Q3	3.192929	3.554169	2.831689
1983 Q4	2.862921	3.238786	2.487055

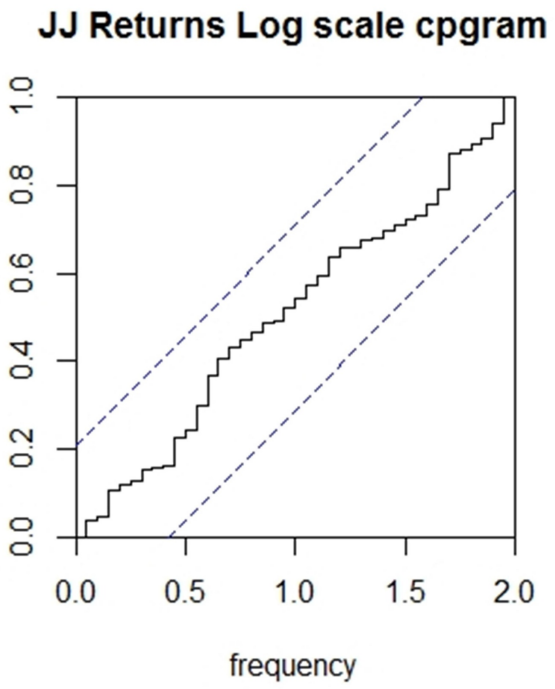


Figure 5.6 JJ Returns Log scale Residuals cpgram

> ts.plot(log(tsJohnson), fitJJ$fitted[,1], fpre, lty=c(1,1,2,2,2), col=c(1,2,2,4,4), ylab = "JJ Returns Log", main = "JJ Quarterly Return Log scale")

The Figure 5.7 shows the forecast of Johnson & Johnson Quarterly Returns for the next three years in log scale. The forecast is for the years 1981, 1982, and 1983. The Figure 5.8 shows the forecast of JJReturns in dollars for the same period in level scale.

> ts.plot(tsJohnson, exp(fitJJ$fitted[,1]), exp(fpre), lty=c(1,1,2,2,2), col=c(1,2,2,4,4), ylab="JJ Returns", main="JJ Quarterly Return Level scale")

> #JJReturns actual versus predicted
> resultdf<-cbind(tsJohnson,exp(fitJJ$fitted[,1]))
> colnames(resultdf)<-c("JJReturns", "predicted")

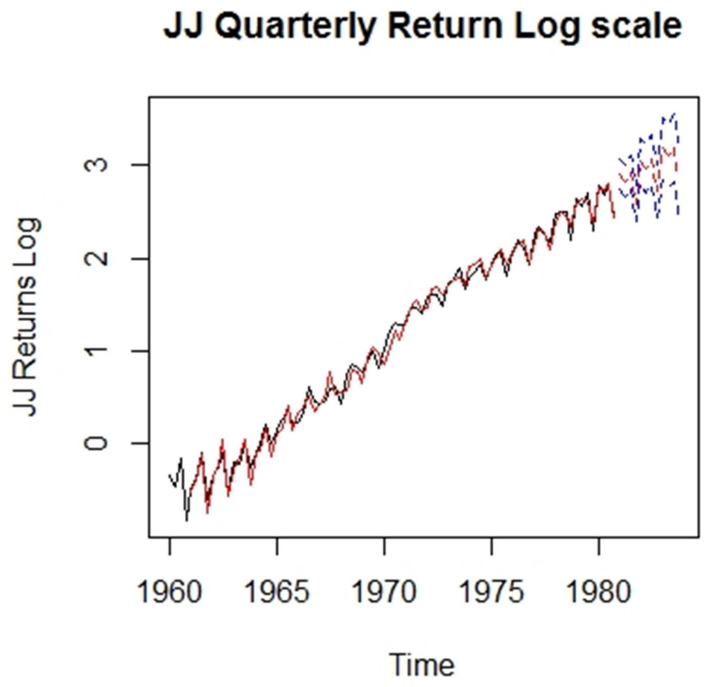


Figure 5.7 JJ Quarterly Returns Log scale forecast

```
> head(resultdf)
    JJReturns predicted
[1,]     0.71        NA
[2,]     0.63        NA
[3,]     0.85        NA
[4,]     0.44        NA
[5,]     0.61 0.6009085
[6,]     0.69 0.6688897

> tail(resultdf)
     JJReturns predicted
[79,]    14.85  13.41613
[80,]     9.99  10.80000
[81,]    16.20  15.94230
[82,]    14.67  15.33952
[83,]    16.02  16.46502
[84,]    11.61  11.50632
```

As mentioned earlier, the Figure 5.8 shows the forecast for the next three years in level scale.

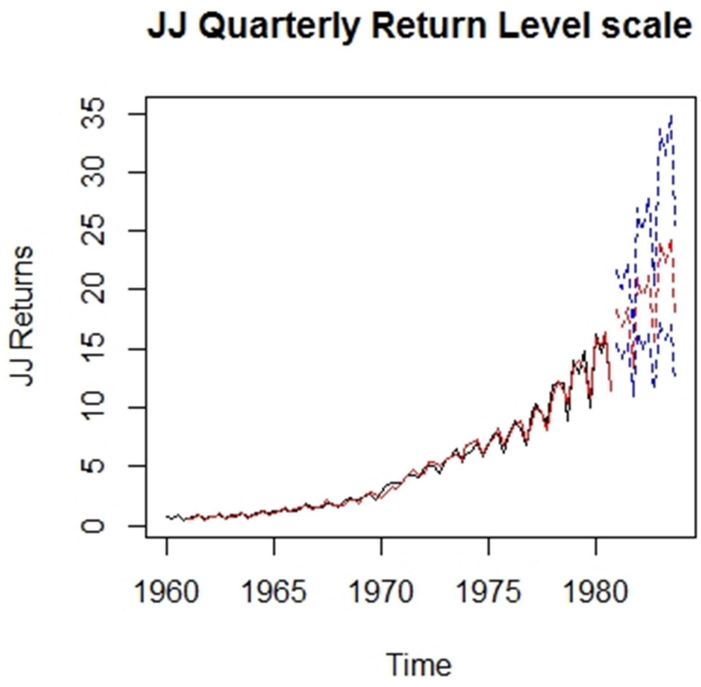


Figure 5.8 JJ Quarterly Returns Level scale forecast

5.2 HW Multiplicative Seasonal Model: John & Johnson Quarterly Returns

The Johnson & Johnson quarterly returns dataset has been analysed with HoltWinter's multiplicative seasonal model as given below. The Figure 5.9 shows the JJ Quarterly returns as line graph. The dataset exhibits an uptrend and seasonality.

```
> #HW Multiplicative Seasonal Model: JJReturns
> tsJohnson<-JohnsonJohnson
> plot(tsJohnson, ylab = "Quarterly Return in Dollars", main = "JJ Quarterly Returns")
```

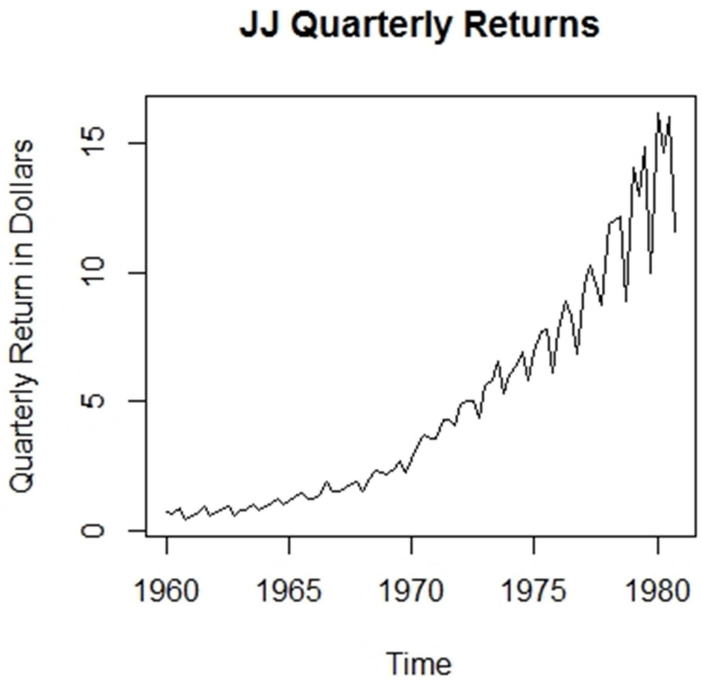


Figure 5.9 JJ quarterly returns ts plot

```
> fitJJmult<-HoltWinters(tsJohnson, seasonal="multiplicative")
> fitJJmult
Holt-Winters exponential smoothing with trend and multiplicative seasonal component.
Call:
HoltWinters(x = tsJohnson, seasonal = "multiplicative")

Smoothing parameters:
 alpha: 0.07910107
 beta : 0.876492
 gamma: 0.791695
```

Coefficients:

```
       [,1]
a  11.7924278
b   0.3214527
s1  1.4854198
s2  1.3178704
s3  1.4041410
s4  0.9791347
```

```
> head(fitJJmult$fitted)
         xhat          level          trend         season
[1,] 0.5997379  0.6256250  0.01175000  0.9409498
[2,] 0.6696614  0.6382377  0.01250614  1.0290706
[3,] 0.8887080  0.6523072  0.01387641 1.3340286
[4,] 0.4757115  0.6680391  0.01550270  0.6959510
[5,] 0.6810548  0.6919853  0.02290341  0.9526724
[6,] 0.7823940  0.7181224  0.02573768  1.0518027
```

```
> tail(fitJJmult$fitted)
          xhat          level       trend         season
 [75,] 13.63112   9.85265  0.2870445  1.3443326
[76,] 10.49113  10.21141  0.3499059  0.9933545
[77,] 15.81999  10.52141  0.3149291  1.4599013
[78,] 14.95481  10.85693  0.3329758  1.3364552
[79,] 16.44809  11.17305  0.3182006  1.4313576
[80,] 11.27833  11.46759  0.2974649  0.9586289
```

The multiplicative model is: xhat = (level + trend) * season
(The Generic HW Additive Model is: Ycap = (L + T) * S)

Where,

‘xhat’	is the predicted or fitted value (Ycap)
‘level’	is the level component of the prediction (L)
‘trend’	is the trend component of the prediction (T)
‘season’	is the seasonal component of the prediction (S)

A sample calculation is shown below, for the first row of fitted values:

0.5997379 = (0.6256250 + 0.01175000) * 0.9409498

> plot(fitJJmult, main="fitJJmult model observed/fitted")

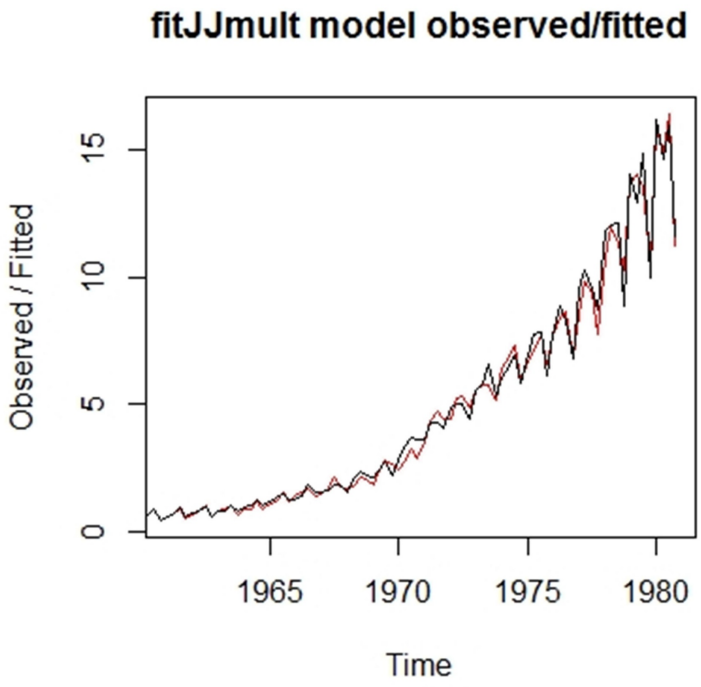


Figure 5.10 fitJJmult model plot

The Figure 5.10 shows the fitted model in graphical form. The actual quarterly returns and the fitted values have been plotted in Figure 5.10.

The Figure 5.11 shows the model residuals as line graph. The model residuals have been further analyzed by plotting as histogram in Figure 5.12. The qqplot and cpgram of the model residuals have benn shown in Figure 5.13 and 5.14 respectively. All these residuals analysis tools have given green signals for the model fitted.

> Resid<-tsJohnson-fitJJmult$fitted[,1]

> plot(Resid, main="Residuals plot JJ quarterly returns")

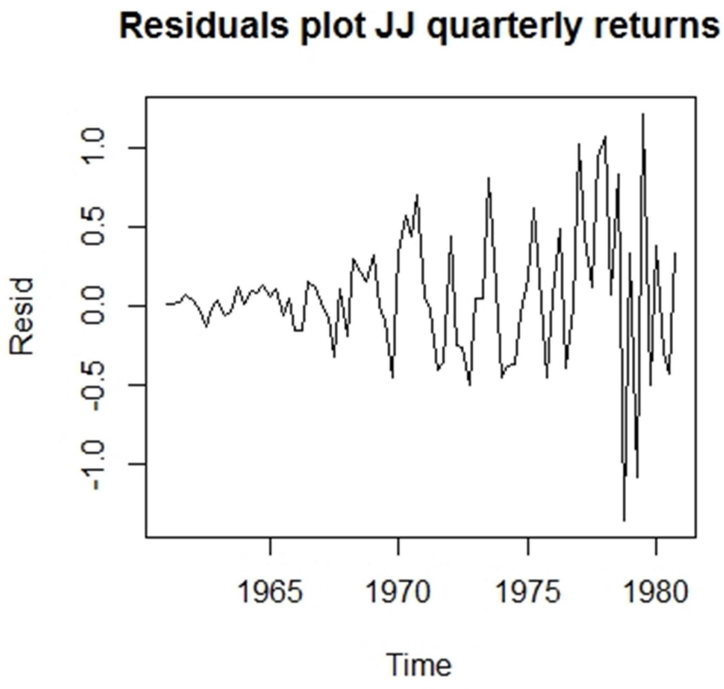


Figure 5.11 fitJJmult model residuals

> Box.test(Resid, lag=20, type="Ljung-Box")

Box-Ljung test

data: Resid
X-squared = 23.55, df = 20, p-value = 0.2626

The Box test also clears the model fit since the p-value is 0.2626 which is higher than 0.05, the level of significance. Further, the critical value of chi squared distribution for 20 degrees of freedom and 0.05 level of significance is 31.41. The calculated chi squared value is 23.55 which is less than 31.41 and hence the model residuals could be safely considered as white noise.

> hist(Resid, main="JJ multiplicative hw model residuals")

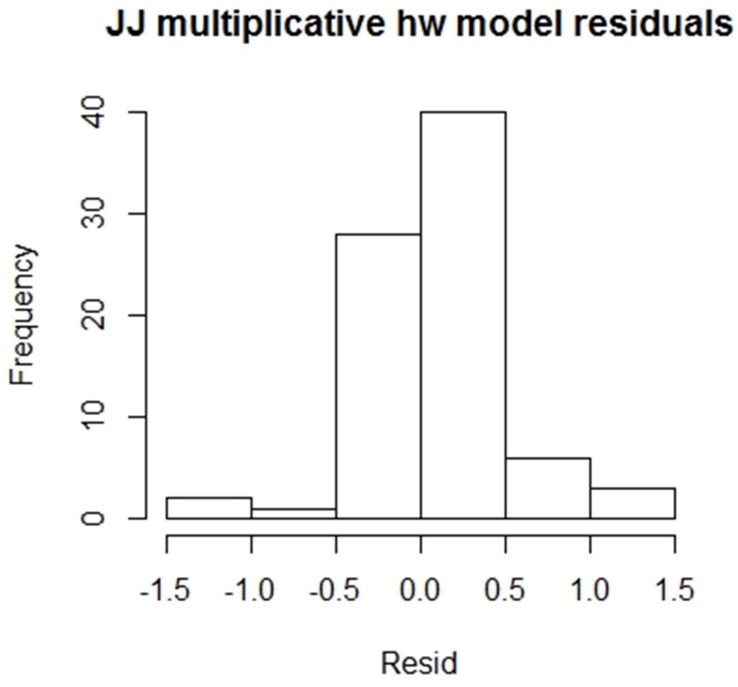


Figure 5.12 histogram of JJmult model residuals

> qqnorm(Resid, main="JJmult hw residuals qqplot")

> qqline(Resid)

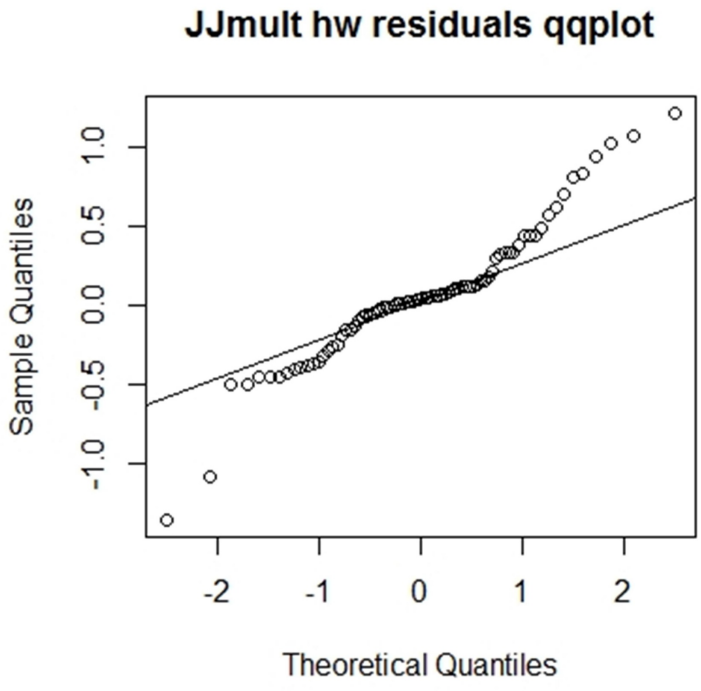


Figure 5.13 qqplot for JJmult model residuals

```
> cpgram(Resid, main="JJmult hw residuals cpgram")
```

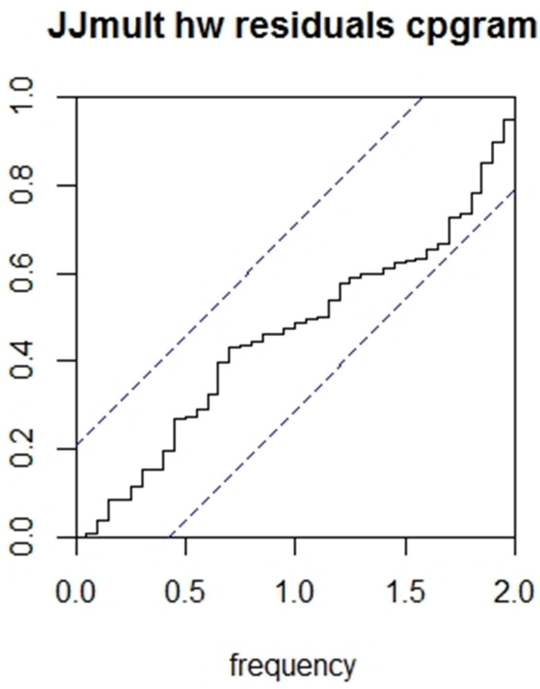


Figure 5.14 cpgram of hw model residuals

```
> fpre<-predict(fitJJmult, n.ahead=12, prediction.interval=TRUE, level=0.95)
> fpre
         fit      upr      lwr
1981 Q1 17.99420 18.66221 17.32618
1981 Q2 16.38816 17.06498 15.71133
1981 Q3 17.91233 18.61395 17.21070
1981 Q4 12.80536 14.85228 10.75844
1982 Q1 19.90417 21.20052 18.60782
1982 Q2 18.08269 19.48083 16.68455
1982 Q3 19.71779 21.34937 18.08620
1982 Q4 14.06434 17.51489 10.61379
1983 Q1 21.81414 24.27543 19.35284
1983 Q2 19.77722 22.32596 17.22848
1983 Q3 21.52325 24.49505 18.55144
1983 Q4 15.32332 20.25070 10.39595
```

The Figure 5.15 shows the forecast for Johnson & Johnson quarterly returns in dollars for the next three years as line graph. The forecast is for the years 1981, 1982, and 1983 based on the HW multiplicative seasonality model.

```
> ts.plot(tsJohnson, fitJJmult$fitted[,1], fpre, lty=c(1,1,2,1,1), col=c(1,2,2,4,4), ylab="quarterly
+ return", main="JJ Quarterly Return forecast")
```

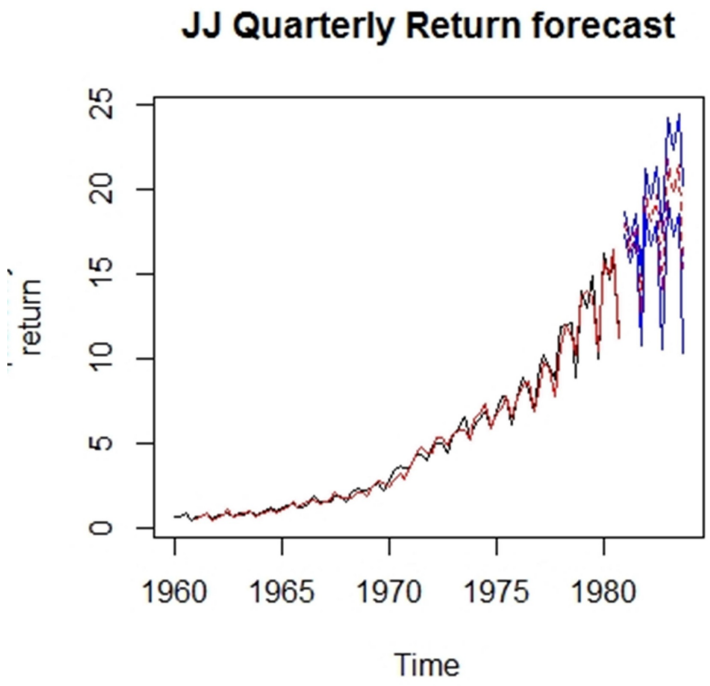


Figure 5.15 fitJJmult model forecast

Chapter 06

Auto Regressive Integrated Moving Average ARIMA

The time series data is record of observations with respect to time. It could be daily minimum, maximum temperature data in a particular location, annual average rainfall, gross domestic product (GDP) which would generally be annual data, etc. The time frame for time series could be different with respect to the type of data collected. For example, the gross domestic product of country could generally be annual, whereas the data traffic flow through a router with a service provider may have to be recorded on a minute basis. The time series data could also be seasonal in nature like umbrella sales or ice cream sales. The umbrella sales will be in maximum numbers during winter. However, the ice cream sales will be minimum in winter and maximum during summer. The time series data could either be non-seasonal or seasonal in nature. The seasonal data would in general be with respect to the four seasons in a year. However, the period of season could be different with respect to the type of data for example the electricity demand fluctuates on an hourly basis each day. The time series data could be recorded annual, quarterly, monthly, daily, hourly, or in minutes. The time period will depend on the type of the data and analysis needs. For example, the Internet traffic through the router with an Internet service provider could be recorded every minute.

Examples for time series data:
The following are a few of examples for time series data.
Non-seasonal:
1. Gross Domestic Product, GDP (year wise) of India
2. Average rainfall at a particular location
3. BSE Sensex: Financial Indices
4. Exports (year wise) by India to other countries
5. Minimum, Maximum temperature (daily) at any particular city or town
6. Data flow through a router (every minute).

Seasonal:
1. Umbrella sales (quarterly)
2. Ice cream sales (monthly)
3. Electricity demand (hourly requirement) on a day basis
4. Demand for agriculture seeds (monthly)
5. Seasonal greeting cards sales in a major store.

Non-seasonal time series data could be like gross domestic product of a country which would generally be annual or quarterly. The other end of time frame in a time series could be traffic flow in a major road junction or data flow through a router with a service provider which may be monitored

and recoded every minute. Seasonal data would generally be fluctuating with respect to the four seasons of the year. The example could be quarterly umbrella sales for an umbrella manufacturing company like STAG, India. The other end of time frame for a seasonal data could be the case of daily electricity demand which may have to be monitored and recorded on an hourly basis.

In this chapter, the non-seasonal arima model development has been demonstrated with 'Electricity Generation Data' from the year 1950-51 to 2018-19. The seasonal arima model development has been demonstrated with the AirPassengers dataset.

6.1 Electricity generation in billion kwh:

The dataset bnkwh_gdp.csv provides data with respect to population (in million), gross domestic product (in billion rupees), and electricity generated in billion kwh. This dataset holds data on a yearly basis from 1950-51 to 2018-19.

```
> elec<- read.csv("bnkwh_gdp.csv", sep=",", header=TRUE)
> elec
   pop_mn   gdp_bn  bn_kwh
1     361     103.60   5.10
2     365     110.19   6.01
3     372     108.25   6.93
4     379     117.91   7.84
5     386     111.41   8.75
6     393     113.61   9.66
.....................................
64   1239 111328.77 1014.80
65   1254 123407.72 1048.40
66   1324 135226.56 1090.85
67   1350 149941.09 1135.33
68   1354 166275.85 1206.31
69   1369 190100.00 1249.34

> bnkwh<- ts(elec$bn_kwh, start=1951)

> bnkwh
Time Series:
Start = 1951
End = 2019
Frequency = 1
 [1]   5.10   6.01   6.93   7.84   8.75   9.66  11.12  12.57  14.03
[10]  15.48  16.94  20.15  23.36  26.57  29.78  32.99  37.81  42.62
[19]  47.43  51.62  55.80  59.43  63.06  66.69  72.94  79.20  85.30
```

```
[28]  91.40  102.52  104.70  120.80  122.10  130.30  140.20  156.86  170.40
[37]  187.70  202.10  221.40  245.44  264.30  287.03  301.40  324.00  350.40
[46]  380.00  395.89  421.70  448.50  480.70  499.50  517.44  532.70  565.10
[55]  594.40  623.80  670.65  723.00  741.20  799.80  844.80  923.20  964.50
[64] 1014.80 1048.40 1090.85 1135.33 1206.31 1249.34
```

```
> ts.plot(bnkwh, main="Electricity generation in billion kwh")
```

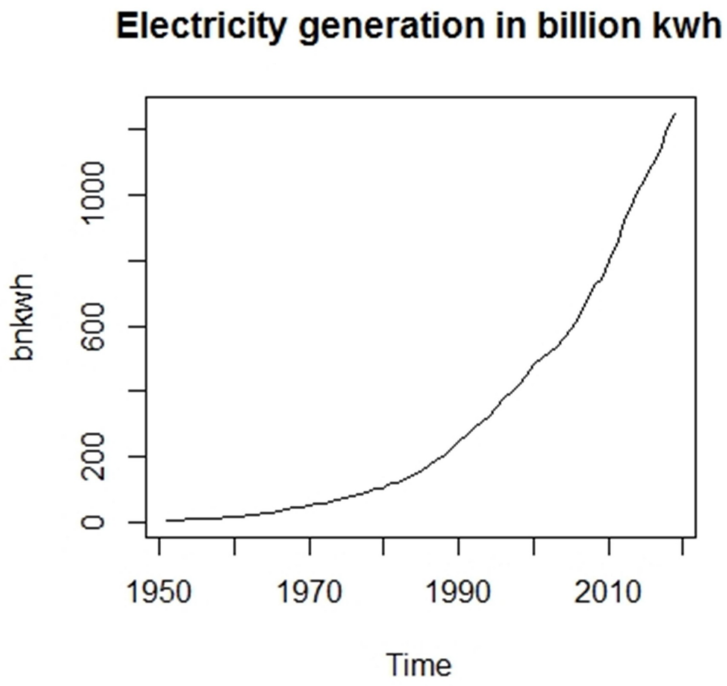


Figure 6.1 Electricity generation in billion kwh plot

The electricity generated shows an upward non-linear trend as depicted in Figure 6.1, and the time series is certainly non-stationary. We need to difference the data in order to make it stationary. The Figure 6.4 shows the first difference of electricity generated in billion kwh. It should be noted that the first differenced data is also non-stationary. Hence, we need to do one more differencing of the time series. The second differenced time series appears to be stationary. The variance is increasing with the increasing values of the time series. This is an indication of data transformation requirement in order to contain the long-term increase in the variance of the data. However, let us put under hold the data transformation for time being and we will carry it out subsequently. Let us proceed with arima model fitting without bothering about the treatment to be method out to the increasing variance of the time series.

```
> acf(bnkwh, main="ACF of bnkwh")
```

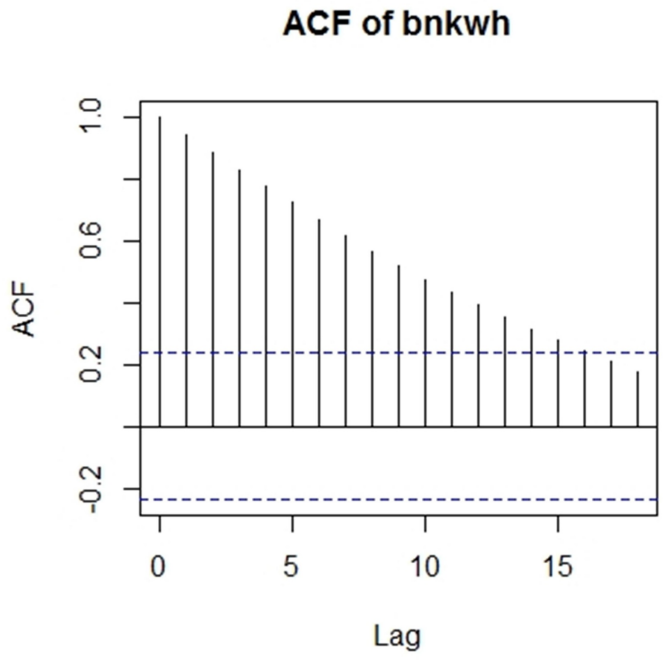


Figure 6.2 ACF of bnkwh

>pacf(bnkwh, main="PACF of bnkwh")

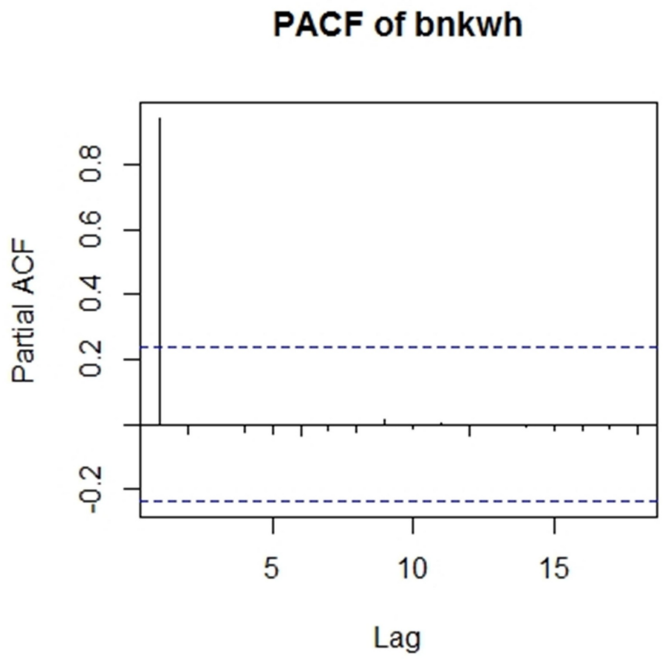


Figure 6.3 PACF of bnkwh

>ts.plot(diff(bnkwh), main="First difference of bnkwh")

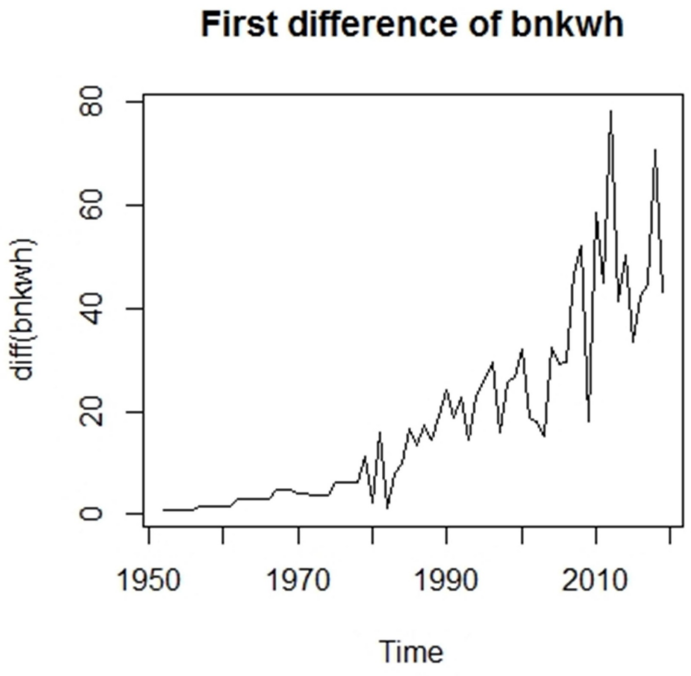


Figure 6.4 First difference of bnkwh plot

>ts.plot(diff(diff(bnkwh)), main="Second difference of bnkwh")

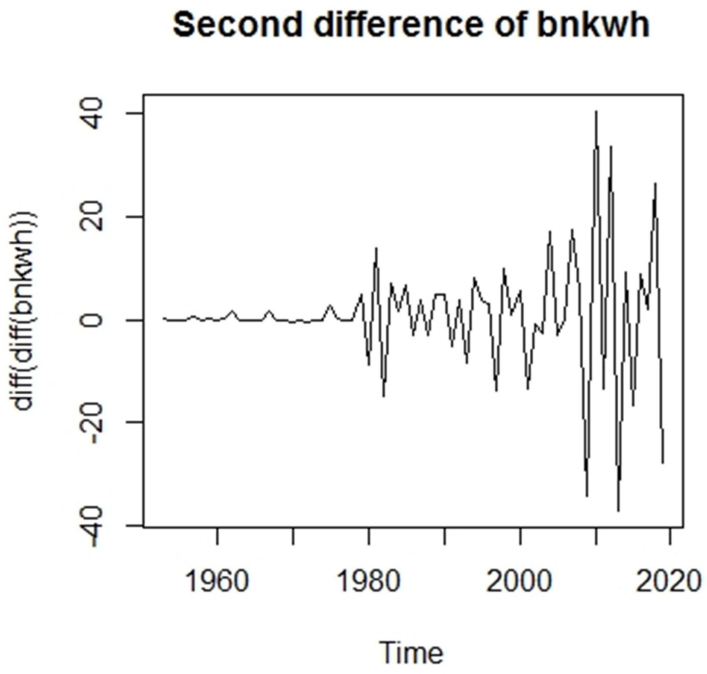


Figure 6.5 Second difference of bnkwh

>acf(diff(diff(bnkwh)), main="ACF of second difference of bnkwh")

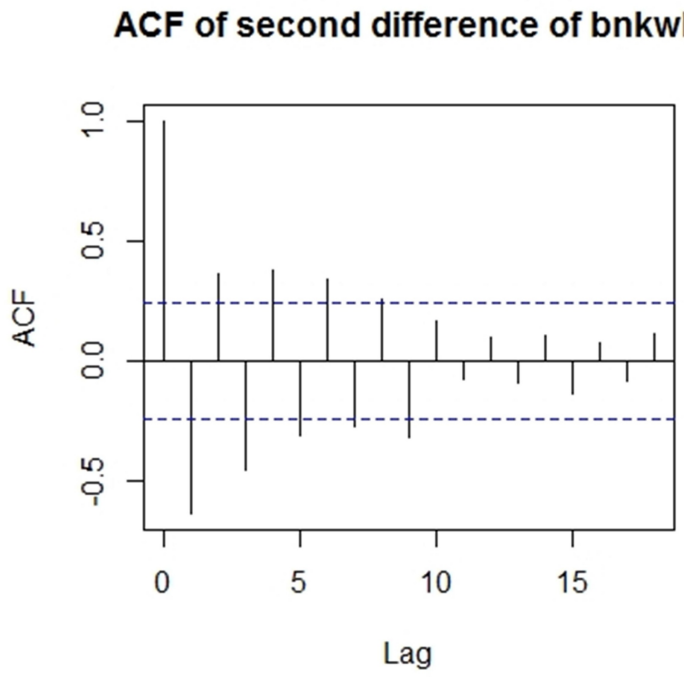


Figure 6.6 ACF of second difference of bnkwh

>pacf(diff(diff(bnkwh)), main="PACF of second difference of bnkwh")

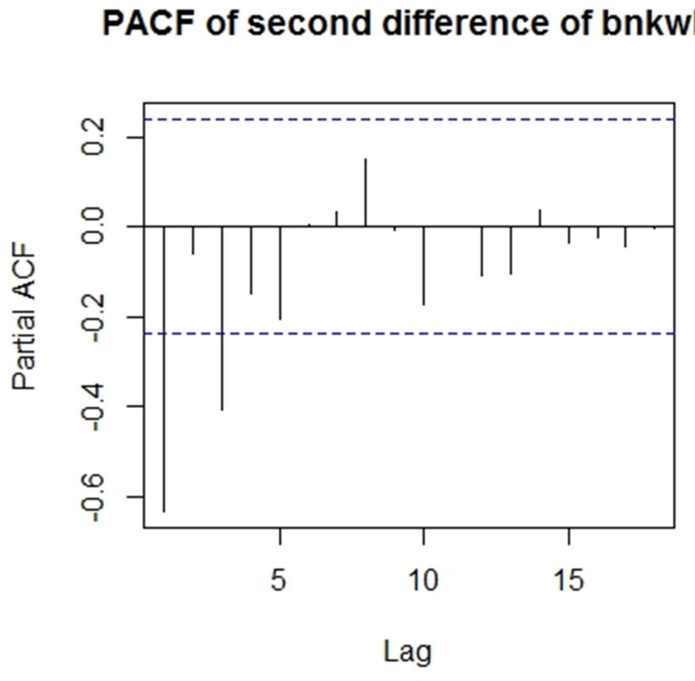


Figure 6.7 PACFof second difference of bnkwh

The ACF and PACF of the second differenced time series is shown in Figure 6.6 and Figure 6.7 respectively. The ACF of the second differenced time series is sinusoidal and decays quickly to zero, indicating a possible AR model. The PACF shows one significant spike at lag 1 and another at lagf 3. This is an indication of either AR(1) model or AR(3) model. Let us first consider the case of AR(1) model and hence the model being fitted for the bnkwh time series is arima(1,2,0).

```
> fm<- arima(bnkwh, order=c(1,2,0))
> fm

Call:
arima(x = bnkwh, order = c(1, 2, 0))

Coefficients:
         ar1
      -0.6721
s.e.   0.0944

sigma^2 estimated as 80.65:  log likelihood = -242.44,  aic = 488.88
```

The AR(1) coefficient is significant as per the arima(1,2,0) model results. The index plot of residuals shows an increasing funnel shape, an indication of possible log transformation requirement. Let us not go in for any data transformation now, and we will do it later. Checking up of the cumulative periodogram (cpgram) indicates that the model is OK and there is no left over autocorrelation.

```
> Box.test(residuals(fm), lag=20, type="Ljung-Box")

        Box-Ljung test

data:  residuals(fm)
X-squared = 20.811, df = 20, p-value = 0.4083

>ts.plot(residuals(fm), main="Residuals plot for arima(1,2,0)")
```

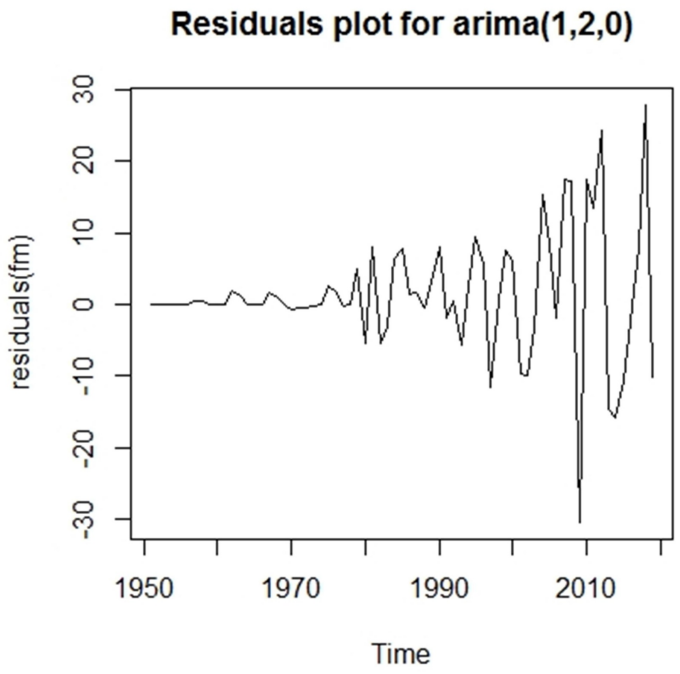


Figure 6.8 Index plot of residuals arima(120)

The index plot of the residuals shows an increasing variance of the time series with the increase in the values of the time series. This problem needs to be taken care of in the arima model development which requires a possible log transformation of the time series. The log transformation of the time series will be taken up in the arima model given in the next section.

```
> qqnorm(residuals(fm), main="qqplot of model residuals")
> qqline(residuals(fm))
```

The Figure 6.9 shows the qqplot of the model residuals. The combo ‘qqnorm’ and ‘qqline’ put together is called ‘qqplot’ and this is an important tool for verifying the normality of residuals and model validity. The Figure 6.10 shows the cpgram of the model residuals.

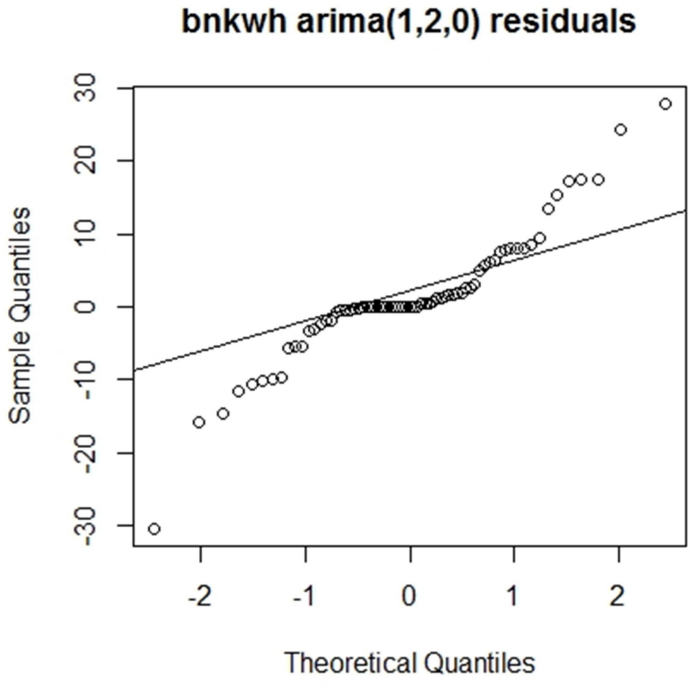


Figure 6.9 qqplot of model residuals

>cpgram(residuals(fm), main="bnkwh arima(1,2,0) residuals")

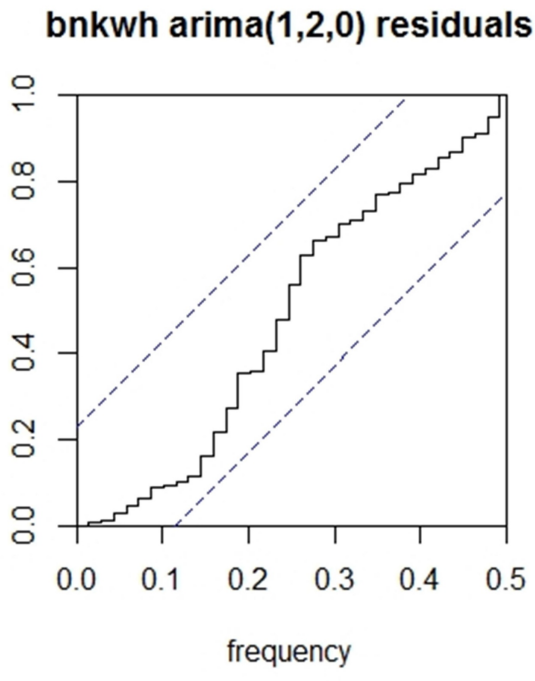


Figure 6.10 cpgram of residuals arima(120)

```
> fpre<- predict(fm, n.ahead=10, main="forecast for 10 years up to 2029")

> fpre
$pred
Time Series:
Start = 2020
End = 2029
Frequency = 1
 [1] 1311.154 1360.344 1418.018 1469.991 1525.795 1579.024 1633.984 1687.781
 [9] 1742.359 1796.412

$se
Time Series:
Start = 2020
End = 2029
Frequency = 1
 [1]   8.980743  14.929155  24.106474  33.458685  44.481049  56.100454
 [7]  68.836495  82.260743  96.555731 111.534120


>ts.plot(bnkwh, fpre$pred, fpre$pred+1.96*fpre$se, fpre$pred-1.96*fpre$se, main="Electricity
generation forecast up to 2029", col=c(1,2,4,4))
```

The model residuals are almost under control inspite of the fact that the model residuals exhibited a behaviour of increasing amplitude and funneling effect was observed. The Figure 6.11 shows the forecast up to the year 2029 based on the arima(1,2,0) model. The blue colour lines show the 95% confidence limits for the predicted forecast values up to year 2029.

The arima (1,2,0) model forecast for the electricity generation in billion kwh with confidence limits has been shown in Figure 6.11 for decision makers use.

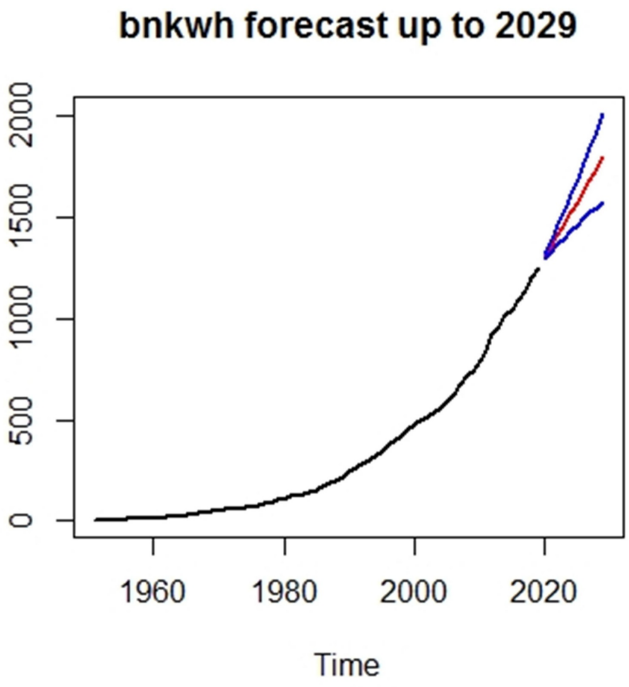


Figure 6.11 Electricity generation forecast up to 2029

6.2 Electricity generation in billion kwh with Log scale arima(1,2,0):

The increasing variance of the time series was indicated by the funnel shaped residuals plot of the arima(1,2,0) model in section 6.1 is a problem which needs to be addressed. In such a situation, transformation of the data with square root / logarithmic transformation could solve the problem.

```
> elec<- read.csv("bnkwh_gdp.csv", sep=",", header=TRUE)
> elec
   pop_mn   gdp_bn   bn_kwh
1    361      103.60     5.10
2    365      110.19     6.01
3    372      108.25     6.93
4    379      117.91     7.84
5    386      111.41     8.75
6    393      113.61     9.66
.....................................
64   1239 111328.77 1014.80
65   1254 123407.72 1048.40
66   1324 135226.56 1090.85
67   1350 149941.09 1135.33
68   1354 166275.85 1206.31
69   1369 190100.00 1249.34

> bnkwh<- ts(elec$bn_kwh, start=1951)

> bnkwh
Time Series:
Start = 1951
End = 2019
Frequency = 1
 [1]    5.10    6.01    6.93    7.84    8.75    9.66   11.12   12.57   14.03
[10]   15.48   16.94   20.15   23.36   26.57   29.78   32.99   37.81   42.62
[19]   47.43   51.62   55.80   59.43   63.06   66.69   72.94   79.20   85.30
[28]   91.40  102.52  104.70  120.80  122.10  130.30  140.20  156.86  170.40
[37]  187.70  202.10  221.40  245.44  264.30  287.03  301.40  324.00  350.40
[46]  380.00  395.89  421.70  448.50  480.70  499.50  517.44  532.70  565.10
[55]  594.40  623.80  670.65  723.00  741.20  799.80  844.80  923.20  964.50
[64] 1014.80 1048.40 1090.85 1135.33 1206.31 1249.34
```

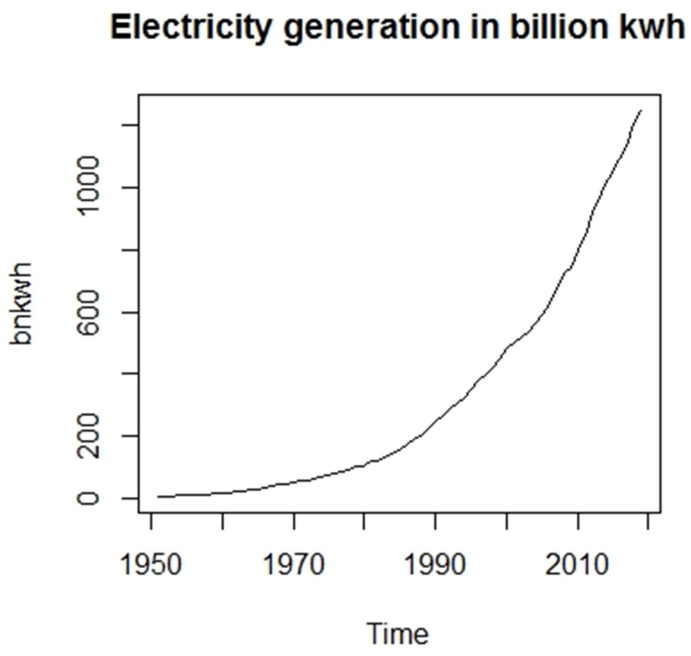


Figure 6.13 Electricity generation in bnkwh

> ts.plot(log(bnkwh), main="bnkwh with log scale")

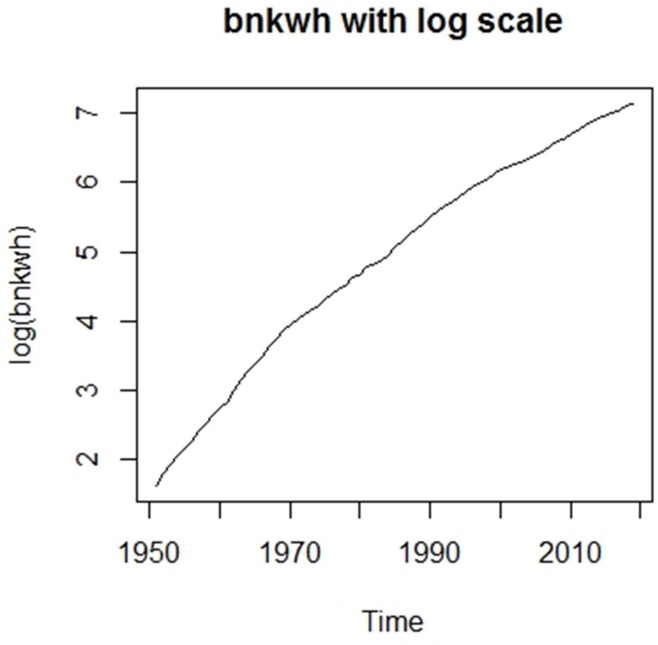


Figure 6.14 bnkwh with log scale plot

The Figure 6.13 shows the electricity generation in bnkwh from 1951 to 2019 and the electricity generation data shows an uptrend very clearly. Let us apply natural logarithm transformation in order to bring under control the funnelling effect observed in residuals of the arima model developed under section 6.1 earlier. The Figure 6.14 shows the time series which has been transformed with natural logarithm. The log transformed series is not stationary and needs to be differenced in order to make it stationary for fitting arima model. The Figure 6.15 and Figure 6.16 shows the bnkwh time series with log scale in first and second differenced form. The log transformed bnkwh time series requires differencing twice in order to make it stationary.

>ts.plot(diff(log(bnkwh)), main="bnkwh with log scale: first difference")

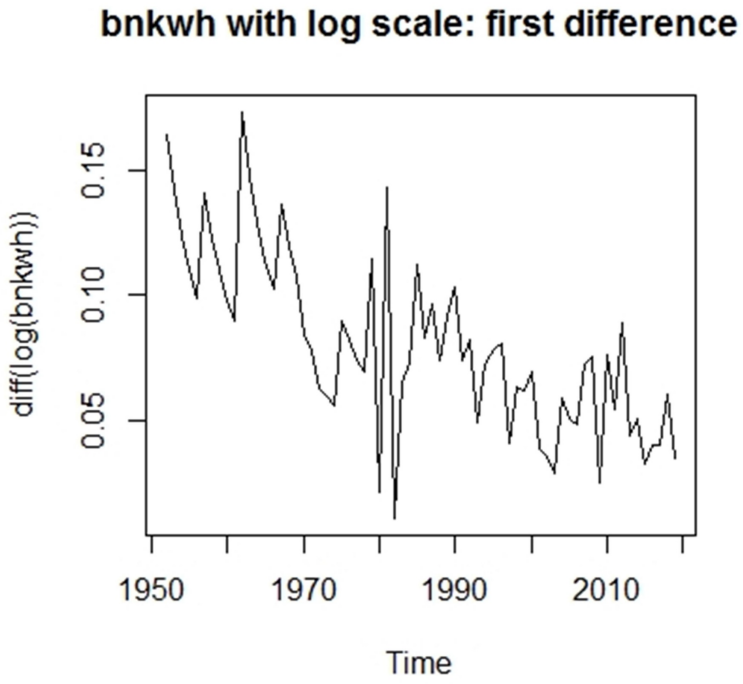


Figure 6.15 first difference of bnkwh with log scale plot

>ts.plot(diff(diff(log(bnkwh))), main="bnkwh with log scale: second difference")

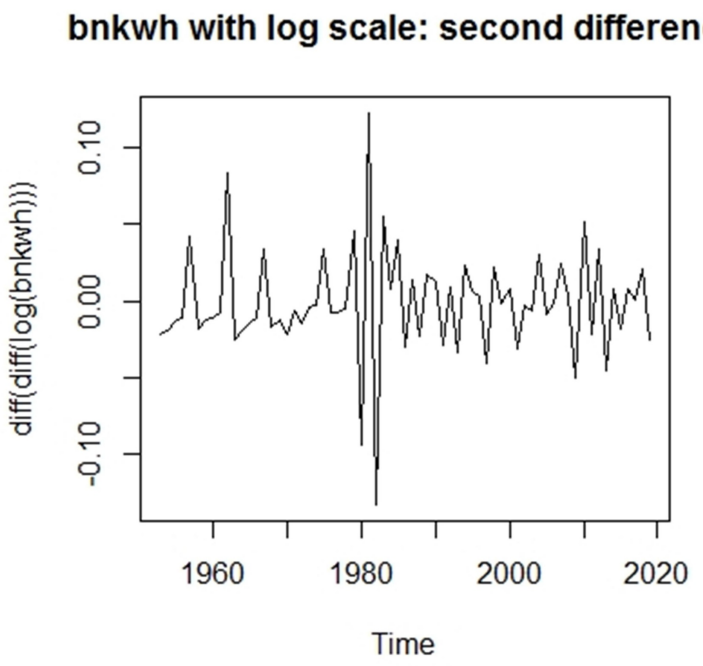


Figure 6.16 second difference of bnkwh with log scale plot

>acf(diff(diff(log(bnkwh))), main="ACF of bnkwh with log scale: second difference")

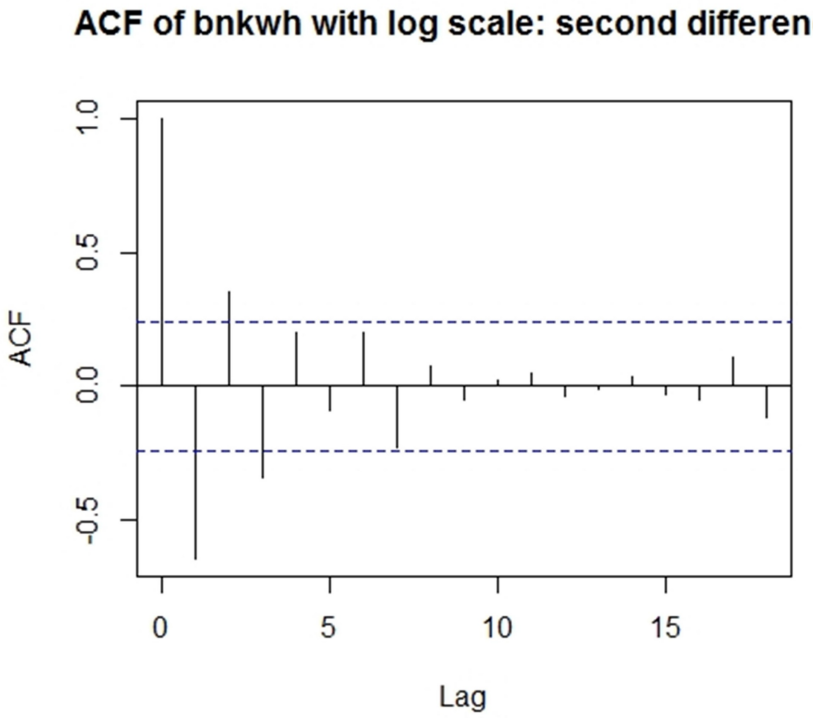


Figure 6.17 ACF: second difference of bnkwh with log scale plot

>pacf(diff(diff(log(bnkwh))), main="PACF of bnkwh with log scale: second difference")

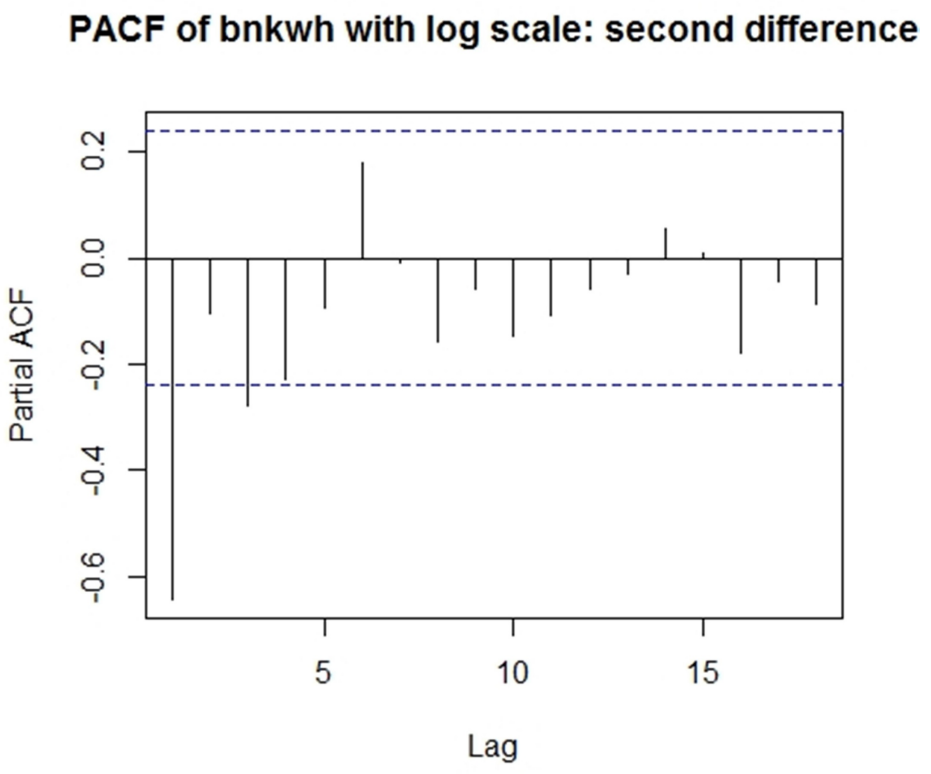


Figure 6.18 PACF: second difference of bnkwh with log scale plot

The ACF and PACF of the second differenced log scale bnkwh has been shown in Figure 6.17 and Figure 6.18 respectively. The ACF of the second differenced log scale bnkwh time series is sinusoidal in nature and quickly decays to zero. This is an indication of possible AR model for the time series. The PACF shows a significant spike at lag 1 and a just significant spike at lag 3. This is an indication of either AR(1) or AR(3) model. The ARIMA(1,2,0) model has been selected for bnkwh with log transformation. In compliance with the principle of parsimony in model fitting, let us try to fit the arima(1,2,0) model for log(bnkwh).

```
> fm1 <- arima(log(bnkwh), order=c(1,2,0))
> fm1
Call:
arima(x = log(bnkwh), order = c(1, 2, 0))
Coefficients:
         ar1
      -0.6377
s.e.   0.0927
```

sigma^2 estimated as 0.0007279: log likelihood = 146.72, aic = -289.43

>ts.plot(residuals(fm1), main="Residuals plot for bnkwh with log scale arima(1,2,0)")

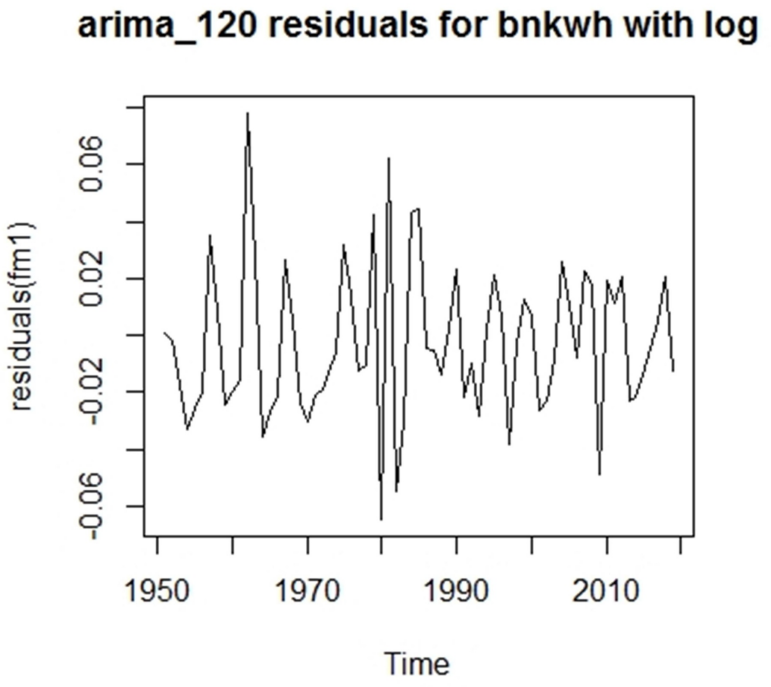


Figure 6.19 Residuals plot for bnkwh with log scale arima(120)

The Figure 6.19 shows the model residuals as line graph. The log transformation of bnkwh has removed the funnelling effect or increasing amplitude of the residuals. The Figure 6.20 and Figure 6.21 shows the qqplot and cpgram of the model residuals.

> Box.test(residuals(fm1), lag=20, type="Ljung-Box")

Box-Ljung test

data: residuals(fm1)
X-squared = 25.486, df = 20, p-value = 0.1835

> qqnorm(residuals(fm1), main="qqplot of model residuals")
> qqline(residuals(fm1))

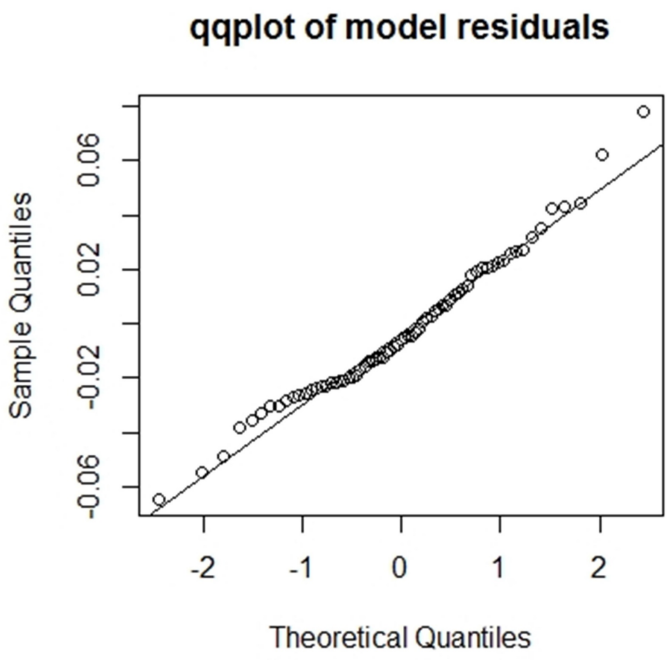


Figure 6.20 qqplot of model residuals

>cpgram(residuals(fm1), main="Residuals for bnkwh with log scale arima(1,2,0)")

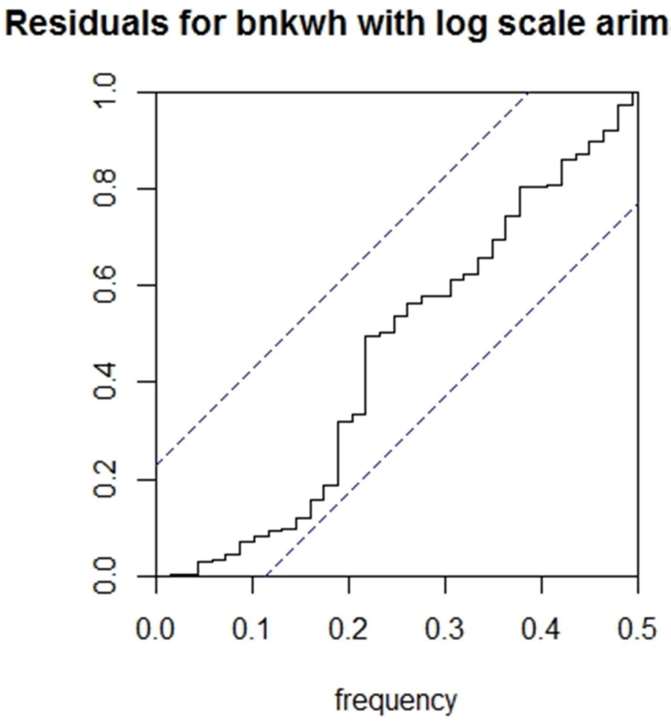


Figure 6.21 cpgram for residuals for bnkwh with log scale arima(120)

Forecast up to 2029 for arima(1,2,0) with log transformation of bnkwh:

```
> fpre1 <- predict(fm1, n.ahead=10)
> fpre1
$pred
Time Series:
Start = 2020
End = 2029
Frequency = 1
 [1] 7.181740 7.222703 7.270301 7.313668 7.359733 7.404078 7.449520 7.494262
 [9] 7.539451 7.584355

$se
Time Series:
Start = 2020
End = 2029
Frequency = 1
 [1] 0.02698049 0.04559600 0.07338678 0.10228791 0.13589351 0.17163094
 [7] 0.21058221 0.25179327 0.29557319 0.34152106


> forecast2029 <- exp(fpre1$pred)
> forecast2029
Time Series:
Start = 2020
End = 2029
Frequency = 1
 [1] 1315.195 1370.187 1436.983 1500.672 1571.418 1642.670 1719.038 1797.698
 [9] 1880.797 1967.177

> ts.plot(bnkwh, exp(fpre1$pred), exp(fpre1$pred + 1.96*fpre1$se), exp(fpre1$pred -
1.96*fpre1$se), main="Forecast upto 2029 with log scale arima(1,2,0)", col=c(1,2,4,4), lty=c(1,3,3,3))
```

The Figure 6.22 shows the forecast for next ten years up to the year 2029. HHowever, it may be noted that the width of confidence interval increases with the forecast time horizon.

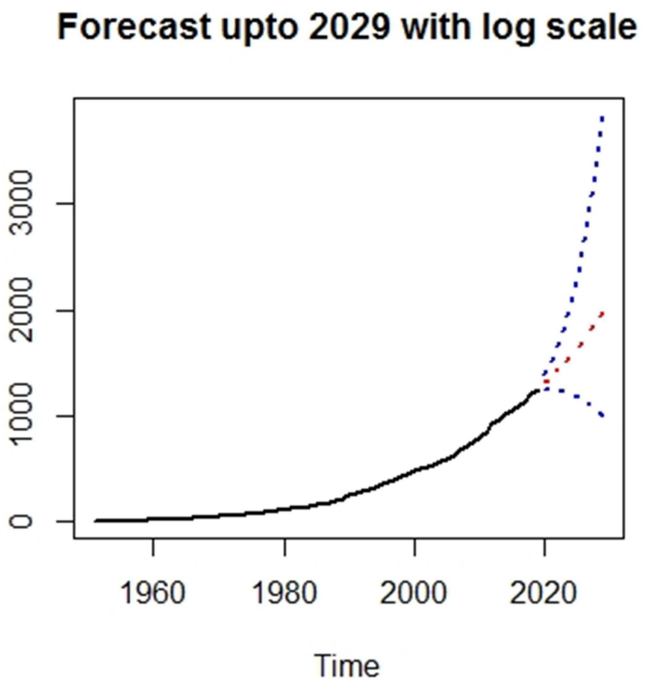


Figure 6.22 bnkwh forecast up to year 2029 with log scale

Chapter 07
Seasonal ARIMA

Seasonal ARIMA: AirPassengers

The dataset 'AirPassengers' is available in R data library. This dataset has been used for demonstration of seasonal ARIMA model building. An uptrend as well as seasonality has been observed in the total monthly number of international passengers (in thousands) who travelled between 1949 and 1960. This traditional Box and Jenkins dataset has been used for concept demonstration of seasonal arima.

```
> #Seasonal arima model: AirPassengers
> library(forecast)
> str(AirPassengers)
 Time-Series [1:144] from 1949 to 1961: 112 118 132 129 121 135 148 148 136 119 ...
> AP<-AirPassengers
> ts.plot(AP, ylab="Air Passengers", main="AirPassengers plot")
```

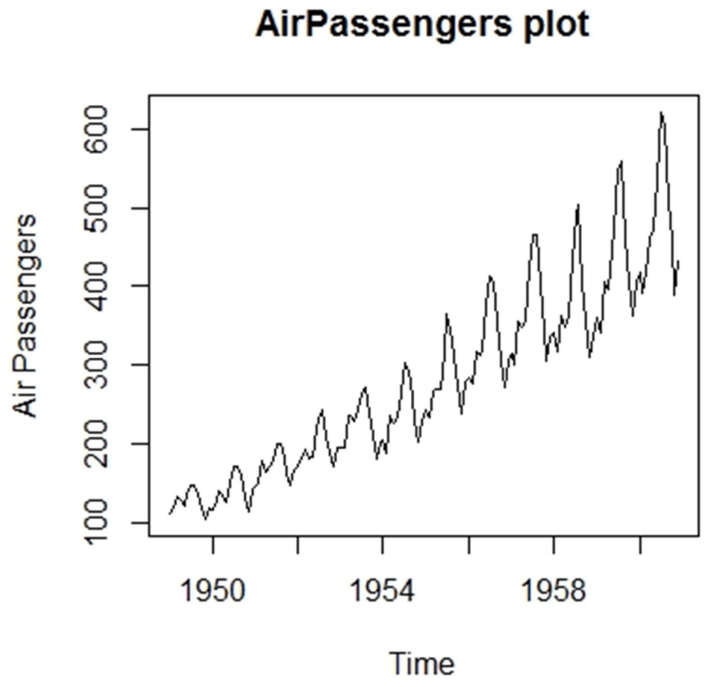


Figure 7.1 AirPassengers ts plot

The Figure 7.1 clearly shows the uptrend and seasonal variation in the AirPassengers dataset. The ACF and PACF of the 'AirPassenger' data with lag=20 has been shown in Figure 7.2 and Figure 7.3 respectively.

> acf(AP, main="ACF of AirPassengers")

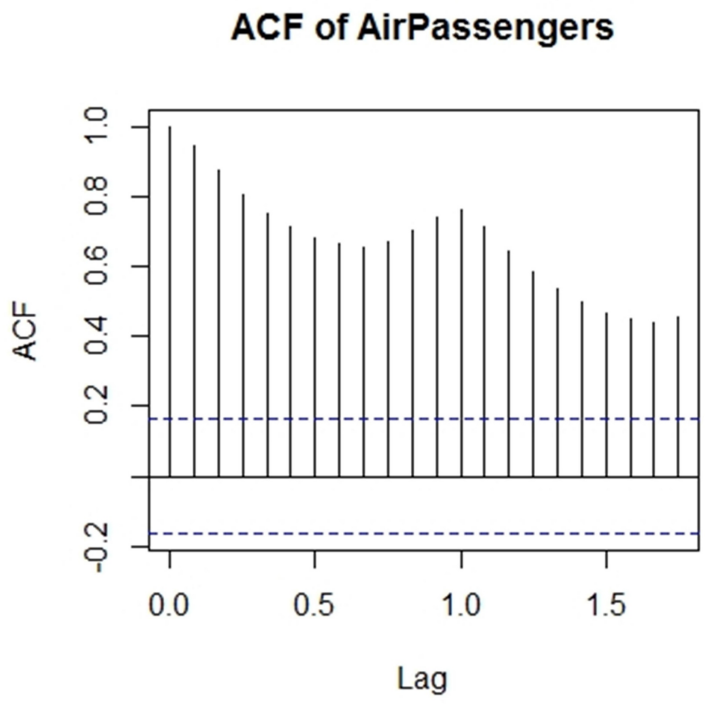


Figure 7.2 ACF of AirPassengers

> pacf(AP, main="PACF of AirPassengers")

First of all the AirPassengers data needs to be seasonal differenced using lag=12 as the seasonal data is monthly. ACF and PACF should be analyzed to find suitable seasonal model.

Then, subsequent differencing needs to be done if necessary to make the data stationary for fitting the arima model.

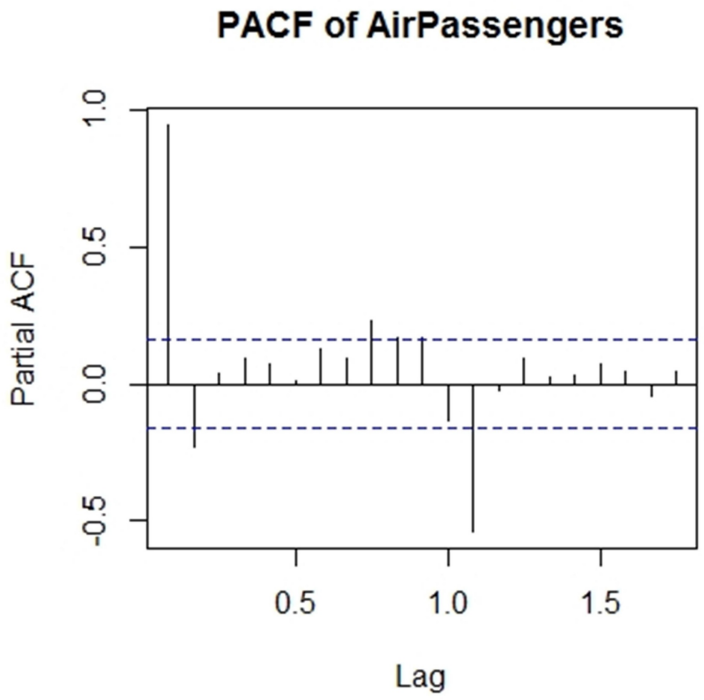


Figure 7.3 PACF of AirPassengers

> ts.plot(diff(AP, lag=12), ylab="Air Passengers", main="seasonal differenced AirPassengers")

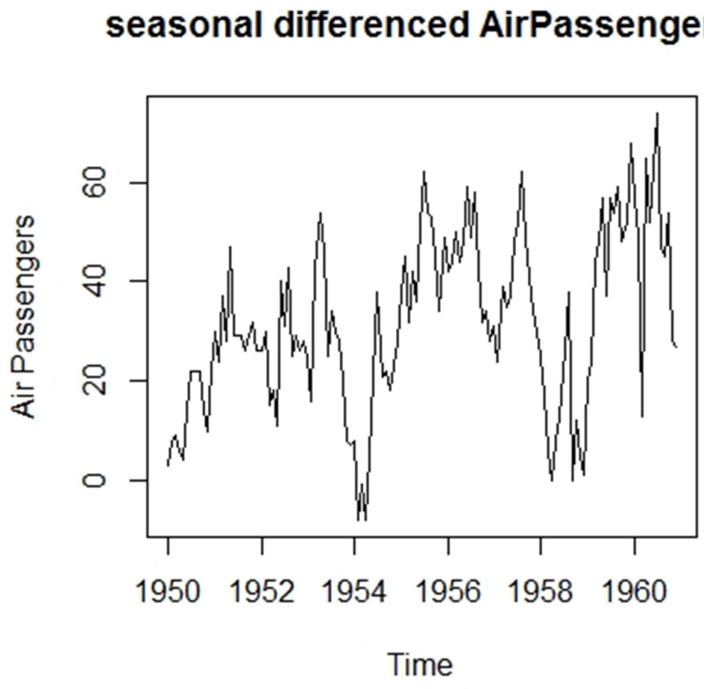


Figure 7.4 Seasonal differenced AirPassengers

The Figure 7.4 shows the seasonal differenced 'AirPassenger' data as a graphical plot. The data for ARIMA model should have constant mean and variance. However, the seasonal differenced data does not appear to have a constant mean and hence further differencing of the data need to be done. The first difference of the seasonal differenced 'AiPassengers' data appears to have constant mean and almost constant variance, as shown in Figure 5.23.

```
> ts.plot(diff(diff(AP, lag=12)), ylab="Air Passengers", main="first diff of seasonal diff AirPassengers")
```

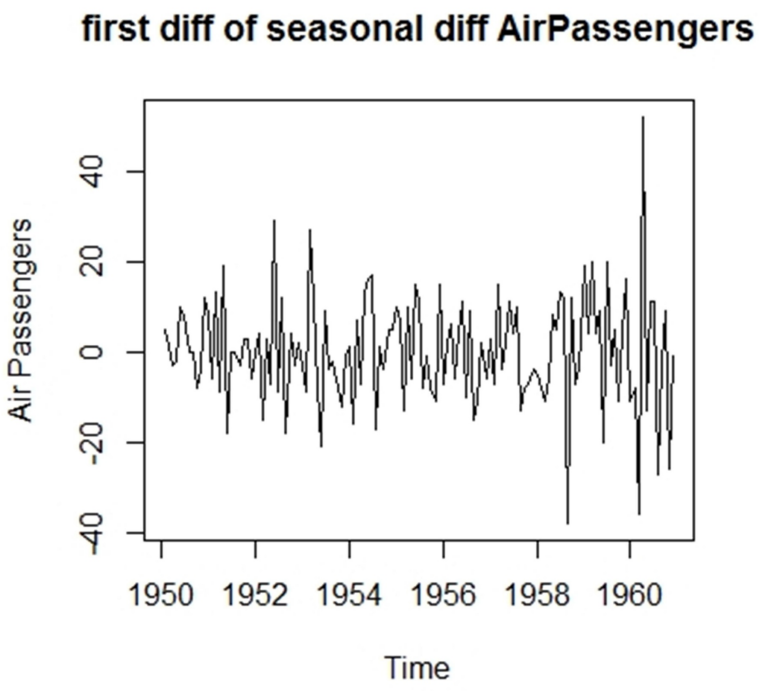


Figure 7.5 first diff of seasonal diff AirPassengers plot

The Figure 7.5 shows the first difference of seasonal differenced AirPassengers data. The ACF and PACF of the first difference of the seasonal differenced 'AiPassengers' data have been shown in Figure 7.6 and Figure 7.7 respectively. The model ARIMA(2,1,0)(0,1,0){12] has been selected after close inspection of the ACF and PACF. The model selection has been reconfirmed with auto.arima model developed by Hyndman.R.J.

The Figure 7.8 shows the sarima model residuals as line graph. The Figure 7.9 and Figure 7.10 show the qqplot and cpgram of model residuals respectively. The cpgram shown in Figure 7.10 shows that the model residuals are under control

> acf(diff(diff(AP, lag=12)), main="ACF first diff of seasonal diff AirPassengers")

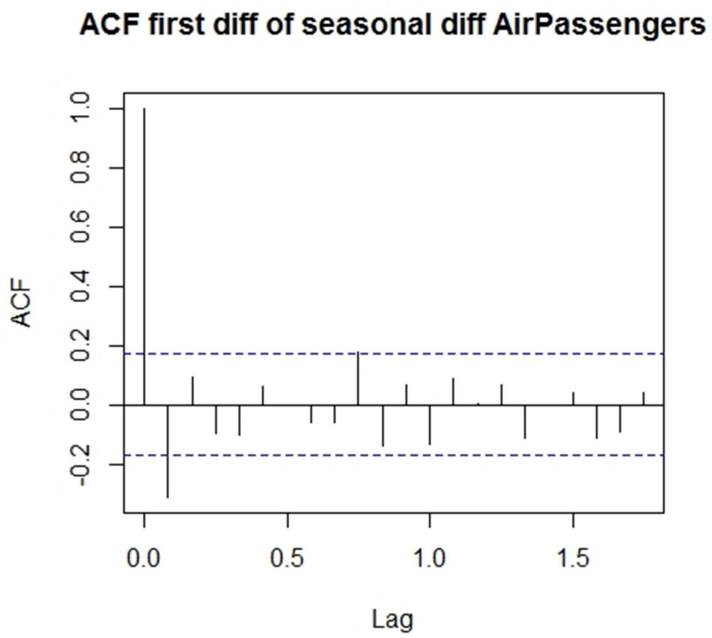


Figure 7.6 ACF of first diff of seasonal diff sales

> pacf(diff(diff(AP, lag=12)), main="PACF: first diff of seasonal diff AirPassengers")

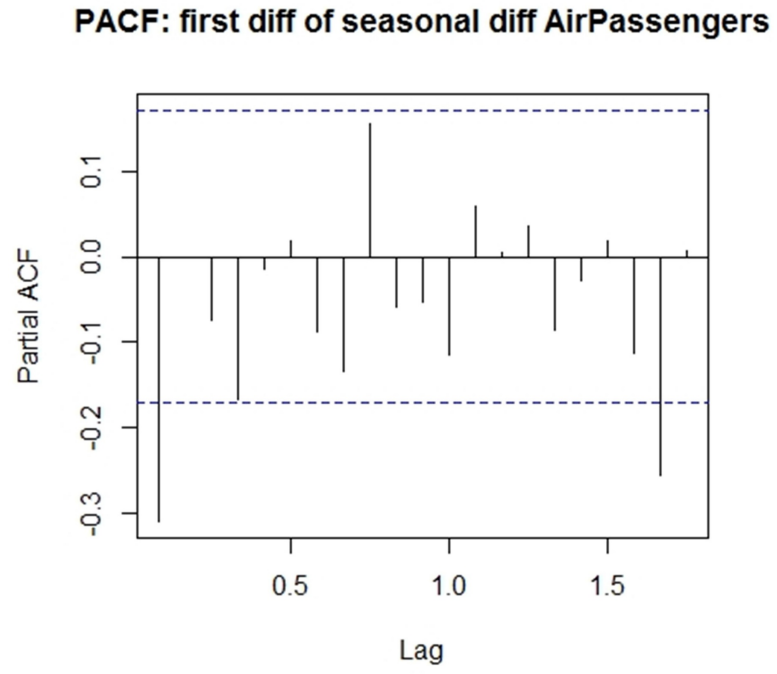


Figure 7.7 PACF of first diff of seasonal diff sales

```
> fm <- arima(AP, order=c(2,1,1), seasonal=list(order=c(0,1,0), frequency=12))
> fm

Call:
arima(x = AP, order = c(2, 1, 1), seasonal = list(order = c(0, 1, 0), frequency = 12))

Coefficients:
         ar1     ar2      ma1
      0.5960  0.2143  -0.9819
s.e.  0.0888  0.0880   0.0292

sigma^2 estimated as 129.3:  log likelihood = -504.92,  aic = 1017.85
```

```
> ts.plot(residuals(fm), main="AirPassengers arima model residuals")
```

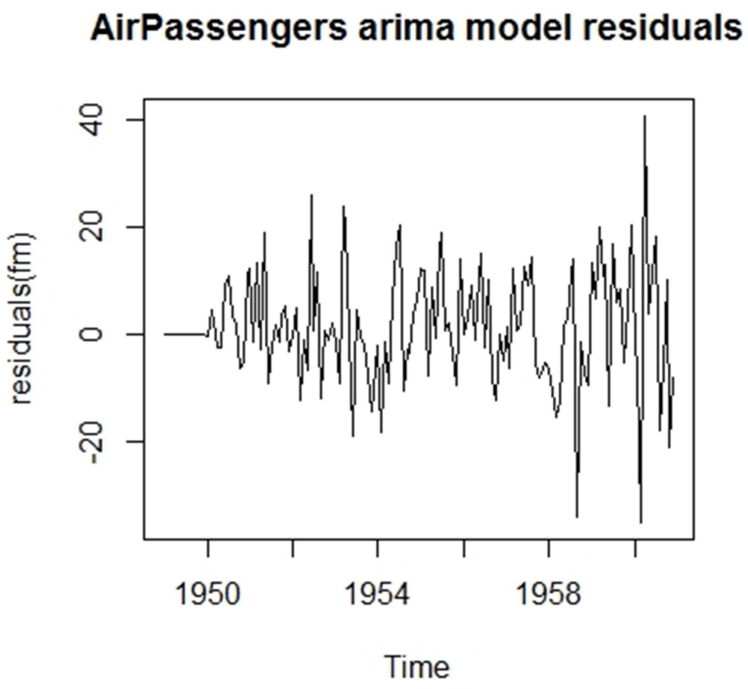


Figure 7.8 AirPasseners seasonal arima model residuals

```
> qqnorm(residuals(fm), main="qqplot of model residuals")
> qqline(residuals(fm))
```

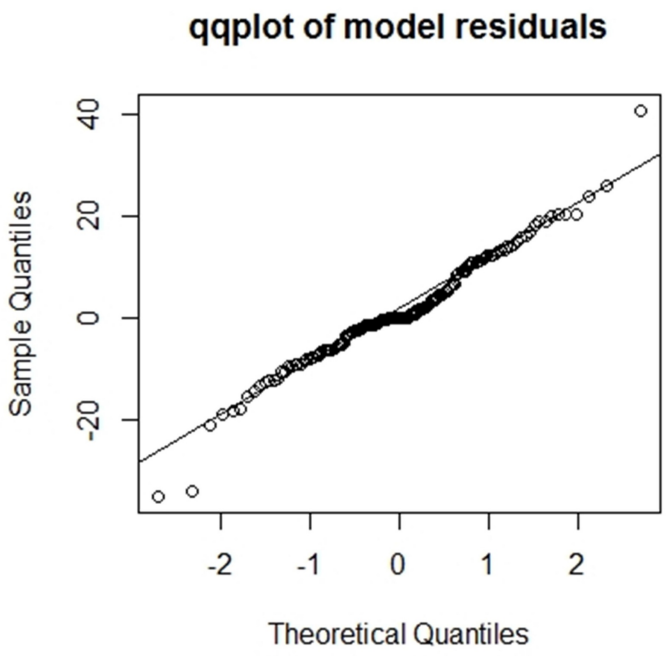


Figure 7.9 qqplot of seasonal arima model residuals

> cpgram(residuals(fm), main="Residuals arima_211_010 seasonal model")

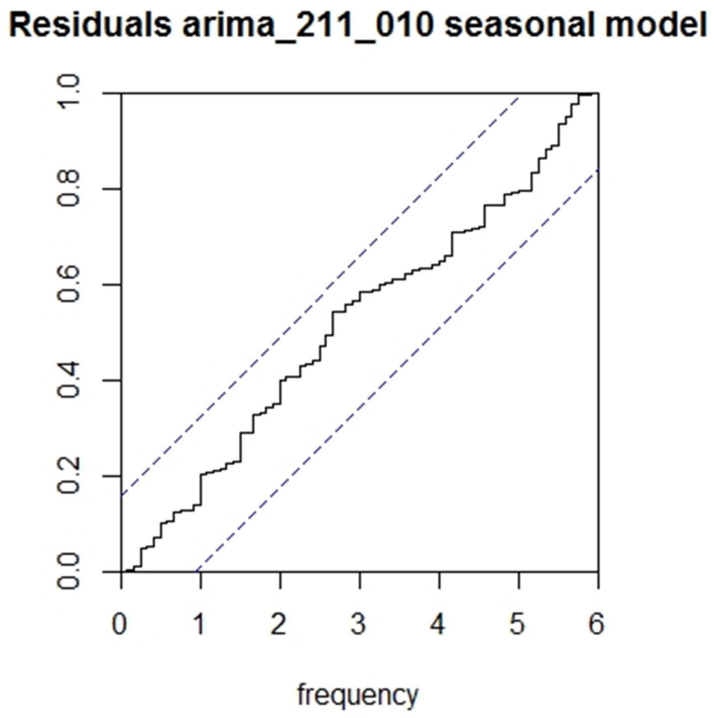


Figure 7.10 cpgram for arima model residuals

```
> fpre <- predict(fm, n.ahead=24)
> fpre
$pred
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
1961 445.6349 420.3950 449.1983 491.8399 503.3944 566.8624 654.2601 638.5974
1962 479.2908 454.1768 483.0869 525.8192 537.4506 600.9839 688.4370 672.8213
          Sep      Oct      Nov      Dec
1961 540.8837 494.1266 423.3327 465.5075
1962 575.1474 528.4241 457.6589 499.8581
$se
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
1961 11.37267 13.34705 14.98337 16.05340 16.84445 17.43079 17.87774 18.22401
1962 23.36497 25.03584 26.49687 27.54154 28.35715 28.99510 29.50585 29.92117
          Sep      Oct      Nov      Dec
1961 18.49694 18.71547 18.89315 19.03981
1962 30.26454 30.55286 30.79864 31.01121
```

The Figure 7.11 shows the forecast for the next two years (24 months).

```
> ts.plot(AP, fpre$pred, fpre$pred+1.96*fpre$se, fpre$pred-1.96*fpre$se, main="AirPassengers
1961 and  1962", lty=c(1,3,3,3), col=c(1,2,3,3))
```

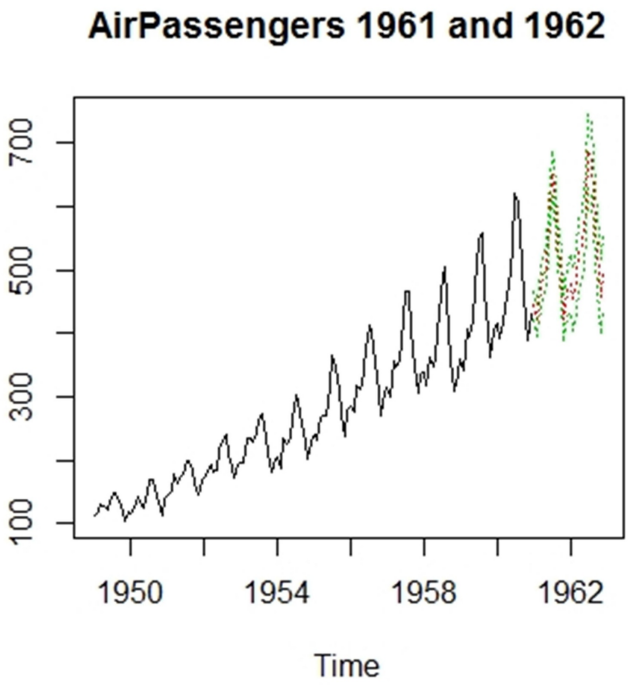


Figure 7.11 Air Passengers forecast for 1961 and 1962

Chapter 08
ARIMAX / Dynamic Regression

The electricity generation in billion kwh has been forecast for the next ten years with ARIMAX / Dynamic Regression model. The ARIMA model with independent regressors is dynamic regression or ARIMAX model. The ARIMAX model given below has two regressor variables which are gross domestic product (in trillion rupees) and population in million.

```
> #arimax model for electricity generation forecast
> elec <- read.csv("bnkwh_gdp.csv", sep=",", header=TRUE)
> head(elec)
  pop_mn  gdp_bn bn_kwh
1    361    103.60    5.10
2    365    110.19    6.01
3    372    108.25    6.93
4    379    117.91    7.84
5    386    111.41    8.75
6    393    113.61    9.66

> bnkwh<-ts(elec$bn_kwh)
> bnkwh
Time Series:
Start = 1
End = 69
Frequency = 1
 [1]    5.10    6.01    6.93    7.84    8.75    9.66   11.12   12.57   14.03
[10]   15.48   16.94   20.15   23.36   26.57   29.78   32.99   37.81   42.62
[19]   47.43   51.62   55.80   59.43   63.06   66.69   72.94   79.20   85.30
[28]   91.40  102.52  104.70  120.80  122.10  130.30  140.20  156.86  170.40
[37]  187.70  202.10  221.40  245.44  264.30  287.03  301.40  324.00  350.40
[46]  380.00  395.89  421.70  448.50  480.70  499.50  517.44  532.70  565.10
[55]  594.40  623.80  670.65  723.00  741.20  799.80  844.80  923.20  964.50
[64] 1014.80 1048.40 1090.85 1135.33 1206.31 1249.34

> plot(bnkwh, main="bnkwh ts plot")
```

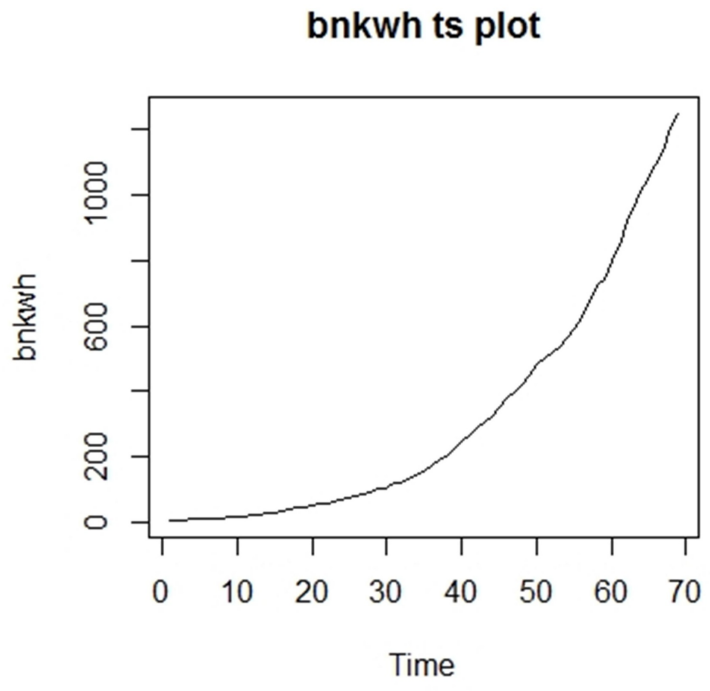


Figure 8.1 bnkwh ts plot

```
> popgdp<-data.frame(popmn=elec$pop_mn, gdpbn=elec$gdp_bn)
> head(popgdp)
  popmn  gdpbn
1   361 103.60
2   365 110.19
3   372 108.25
4   379 117.91
5   386 111.41
6   393 113.61

> tail(popgdp)
   popmn    gdpbn
64  1239 111328.8
65  1254 123407.7
66  1324 135226.6
67  1350 149941.1
68  1354 166275.9
69  1369 190100.0
```

```
> fm<-lm(bnkwh~popgdp$popmn+popgdp$gdpbn)
> summary(fm)

Call:
lm(formula = bnkwh ~ popgdp$popmn + popgdp$gdpbn)
Residuals:
    Min      1Q  Median    3Q   Max
-106.421 -33.574   1.211  32.550  66.704
Coefficients:
                    Estimate    Std. Error   t value  Pr(>|t|)
(Intercept)        -2.960e+02  1.809e+01  -16.36  <2e-16 ***
popgdp$popmn  6.736e-01   2.662e-02   25.30  <2e-16 ***
popgdp$gdpbn  3.838e-03   1.834e-04   20.93  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 40.37 on 66 degrees of freedom
Multiple R-squared:  0.9876,    Adjusted R-squared:  0.9873
F-statistic:  2636 on 2 and 66 DF,  p-value: < 2.2e-16

> et<-residuals(fm)              #residuals of regression model
> plot(et, type="l", main="Residuals of regression")
```

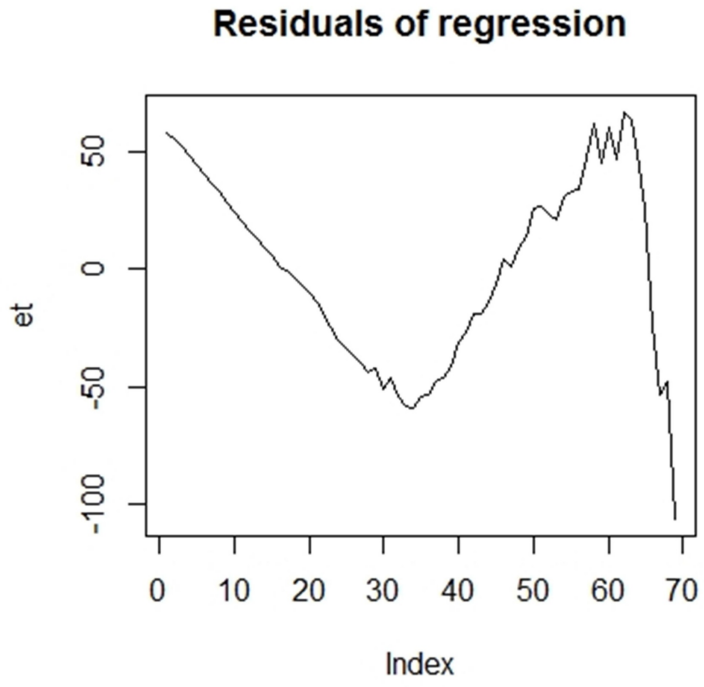


Figure 8.2 Residuals et ts plot

> plot(diff(et), type="l", main="First diff Residuals of regression")

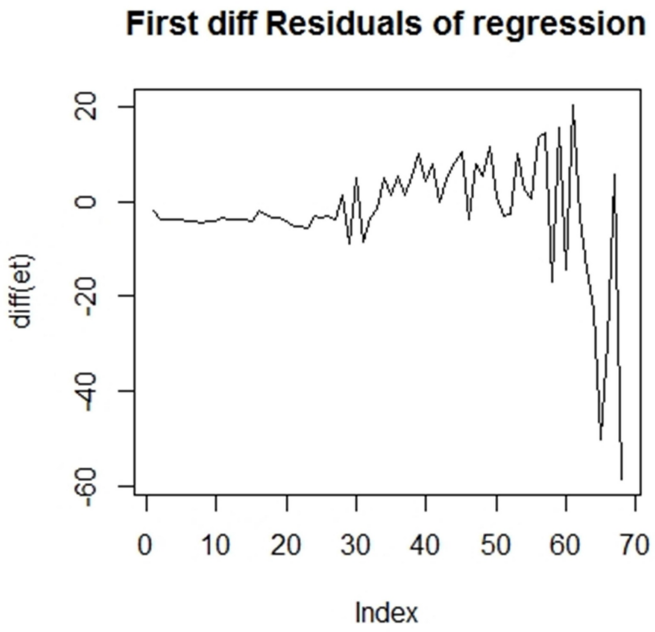


Figure 8.3 first diff et ts plot

> plot(diff(diff(et)), type="l", main="Second diff Residuals of regression")

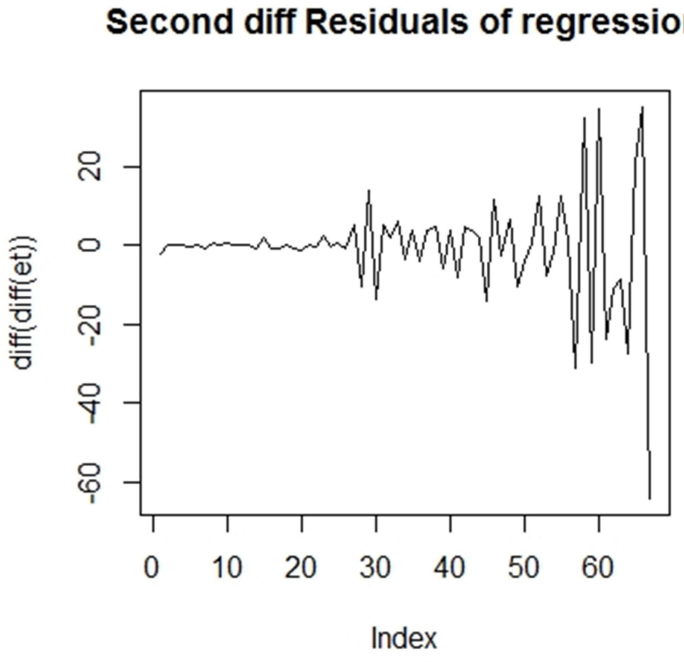


Figure 8.4 second diff et ts plot

> acf(diff(diff(et)), lag=20, main="acf of residuals")

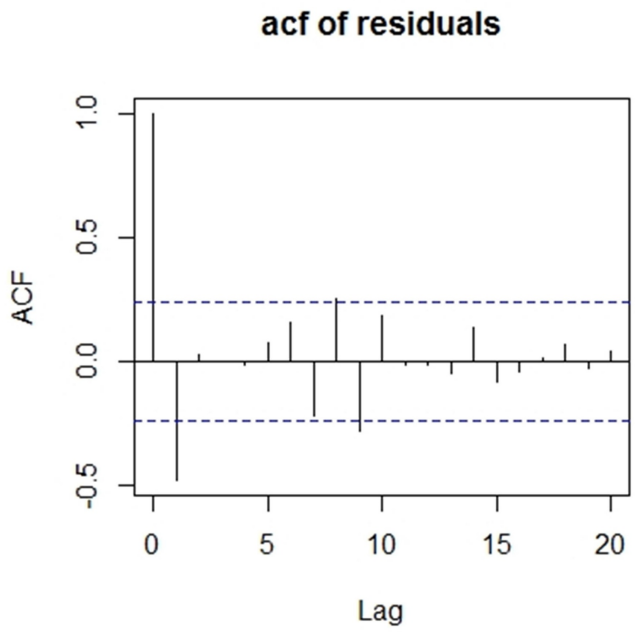


Figure 8.5 ACF of second diff et

> pacf(diff(diff(et)), lag=20, main="pacf of residuals")

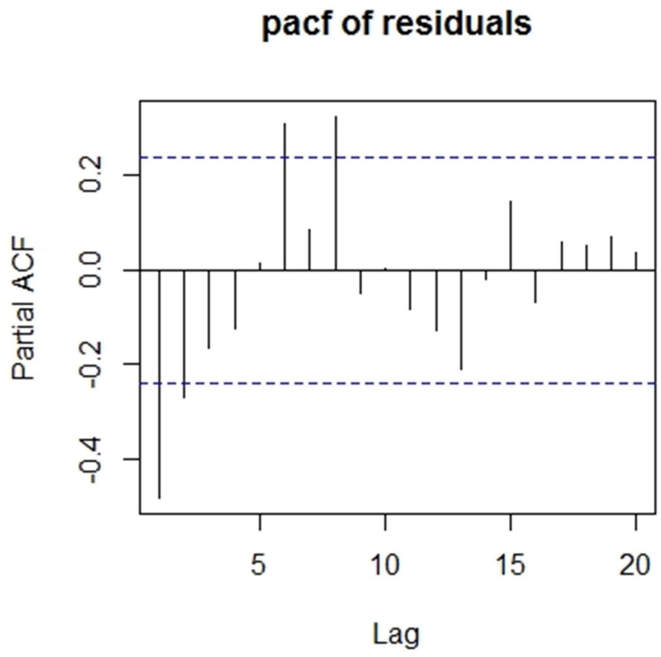


Figure 8.6 PACF of second diff et

```
> #after analyzing ACF and PACF residuals et, arima(0,2,2) has been selected
> fit1<-arima(bnkwh, order=c(0,2,2), xreg=popgdp)
> fit1

Call:
arima(x = bnkwh, order = c(0, 2, 2), xreg = popgdp)

Coefficients:
         ma1     ma2    popmn  gdpbn
      -1.1399  0.4556  -0.2359  2e-03
s.e.   0.1576  0.1830   0.1336  6e-04

sigma^2 estimated as 62.89:  log likelihood = -234.52,  aic = 479.03
> plot(residuals(fit1),main="residuals of arima(0,2,2))")
```

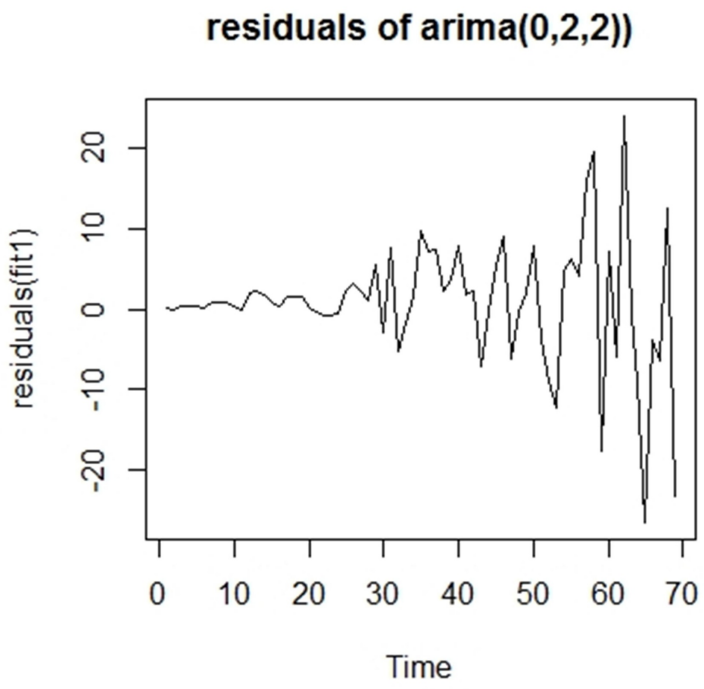


Figure 8.7 arima_022 residuals index plot

```
 > qqnorm(residuals(fit1), main="qqplot")
> qqline(residuals(fit1), col=2)
```

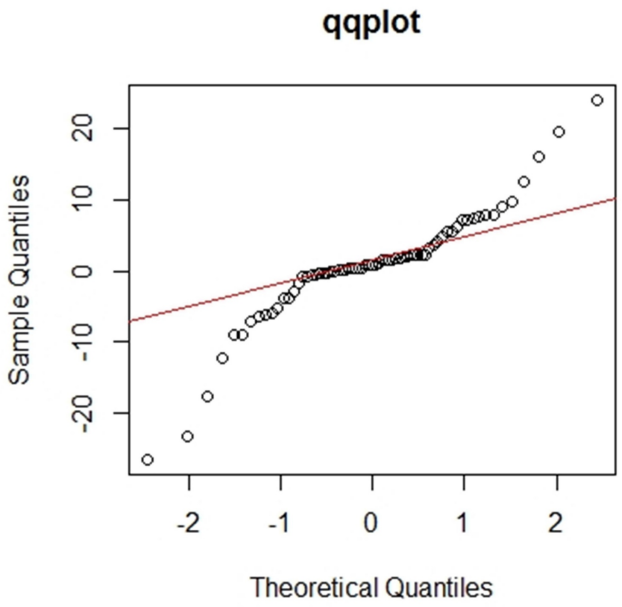


Figure 8.8 arima_022 residuals qqplot

> cpgram(residuals(fit1), main="arima_022 residuals")

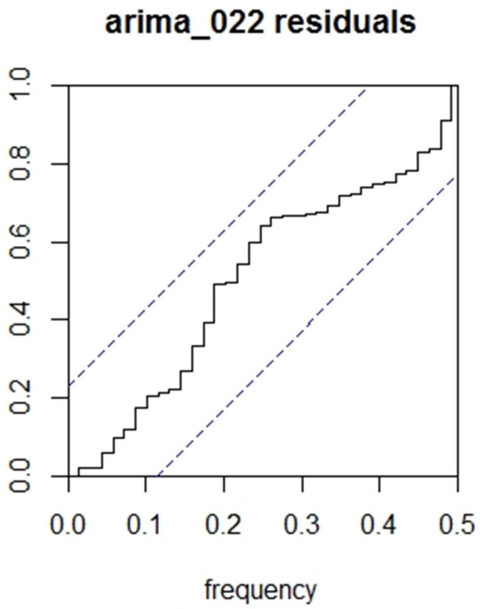


Figure 8.9 arima_022 residuals cpgram

> #arima prediction of 'popmn' for next ten years
> library(forecast)
> auto.arima(popgdp$popmn)

```
Series: popgdp$popmn
ARIMA(0,2,1)
Coefficients:
          ma1
      -0.8662
s.e.   0.0524

sigma^2 estimated as 53.43:  log likelihood=-228.53
AIC=461.06   AICc=461.25   BIC=465.47
> fit2<-arima(popgdp$popmn, order=c(1,2,0))
> fit2

Call:
arima(x = popgdp$popmn, order = c(1, 2, 0))
Coefficients:
          ar1
      -0.3803
s.e.   0.1131

sigma^2 estimated as 73.59:  log likelihood = -239.15,  aic = 482.29
> fpre2<-predict(fit2, n.ahead=10)
> fprepopmn<-fpre2$pred
> fprepopmn
Time Series:
Start = 70
End = 79
Frequency = 1
 [1] 1379.817 1392.224 1404.027 1416.060 1428.005 1439.983 1451.949 1463.919
 [9] 1475.888 1487.858

> #arima prediction of 'gdpbn' for next ten years
> #library(forecast)
> auto.arima(popgdp$gdpbn)
Series: popgdp$gdpbn
ARIMA(0,2,0)

sigma^2 estimated as 1867027:  log likelihood=-578.8
AIC=1159.61   AICc=1159.67   BIC=1161.81
> fit3<-arima(popgdp$gdpbn, order=c(0,2,0))
> fit3
```

```
Call:
arima(x = popgdp$gdpbn, order = c(0, 2, 0))
sigma^2 estimated as 1867027:  log likelihood = -578.8,  aic = 1159.61

> fpre3<-predict(fit3, n.ahead=10)
> fpregdpbn<-fpre3$pred
> fpregdpbn
Time Series:
Start = 70
End = 79
Frequency = 1
 [1] 213924.1 237748.3 261572.4 285396.6 309220.8 333044.9 356869.1 380693.2
 [9] 404517.4 428341.5

 > #Electricity generation forecast for next ten years
> futuredf<-data.frame(popmn=fprepopmn, gdpbn=fpregdpbn)

> futuredf
      popmn      gdpbn
1   1379.817   213924.1
2   1392.224   237748.3
3   1404.027   261572.4
4   1416.060   285396.6
5   1428.005   309220.8
6   1439.983   333044.9
7   1451.949   356869.1
8   1463.919   380693.2
9   1475.888   404517.4
10  1487.858  428341.5

> fpre1<-predict(fit1, n.ahead=10, futuredf)
> fpre1
$pred
Time Series:
Start = 70
End = 79
Frequency = 1
 [1] 1325.656 1390.954 1456.394 1521.781 1587.188 1652.587 1717.989 1783.390
 [9] 1848.791 1914.192
```

```
$se
Time Series:
Start = 70
End = 79
Frequency = 1
 [1]  7.930592 10.460535 14.013537 18.338545 23.275105 28.726341 34.630296
 [8] 40.944272 47.636847 54.683669
```

```
> ts.plot(bnkwh, fpre1$pred, fpre1$pred+1.96*fpre1$se,fpre1$pred-1.96*fpre1$se, col=c(1,2,4,4),
lty=c(1,1,2,2), main="bnkwh forecast for next ten years")
```

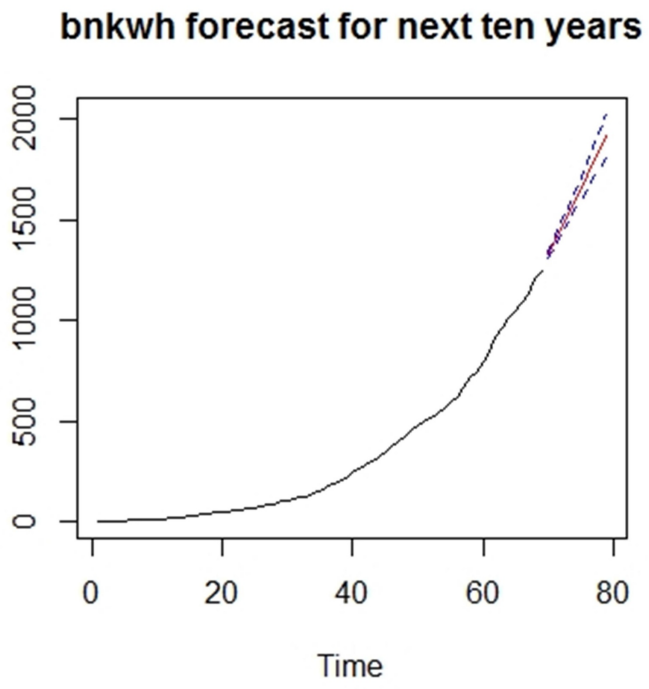


Figure 8.10 bnkwh forecast for next ten years plot

ANNEXURE-I: DATASET

A. bnkwh_gdp.csv

year	pop_mn	gdp_bn	bn_kwh
1950-51	361	103.6	5.1
1951-52	365	110.19	6.01
1952-53	372	108.25	6.93
1953-54	379	117.91	7.84
1954-55	386	111.41	8.75
1955-56	393	113.61	9.66
1956-57	401	135.3	11.12
1957-58	409	139.31	12.57
1958-59	418	155.16	14.03
1959-60	426	163.27	15.48
1960-61	434	178.7	16.94
1961-62	444	189.12	20.15
1962-63	454	203.21	23.36
1963-64	464	233.5	26.57
1964-65	474	272.22	29.78
1965-66	485	286.93	32.99
1966-67	495	324.39	37.81
1967-68	506	380.03	42.62
1968-69	518	402.57	47.43
1969-70	529	443.34	51.62
1970-71	541	473.54	55.8
1971-72	554	507.08	59.43
1972-73	567	559.12	63.06
1973-74	580	680.95	66.69
1974-75	593	804.79	72.94
1975-76	607	864.52	79.2
1976-77	620	931.89	85.3
1977-78	634	1056.15	91.4
1978-79	648	1144.91	102.52
1979-80	664	1258.82	104.7
1980-81	679	1499.87	120.8
1981-82	692	1758.45	122.1
1982-83	708	1960.1	130.3
1983-84	723	2280.77	140.2
1984-85	739	2551.87	156.86
1985-86	755	2880.95	170.4
1986-87	771	3221.44	187.7
1987-88	788	3655.92	202.1

1988-89	805	4323.97	221.4
1989-90	822	4961.97	245.44
1990-91	839	5786.67	264.3
1991-92	856	6637.98	287.03
1992-93	872	7629	301.4
1993-94	892	8792.75	324
1994-95	910	10325.07	350.4
1995-96	928	12132.41	380
1996-97	946	14061.95	395.89
1997-98	964	15591.89	421.7
1998-99	983	17884.1	448.5
1999-00	1001	20076.99	480.7
2000-01	1019	21546.8	499.5
2001-02	1040	23357.77	517.44
2002-03	1056	25196.37	532.7
2003-04	1072	28207.95	565.1
2004-05	1089	32198.35	594.4
2005-06	1106	36672.53	623.8
2006-07	1122	42614.72	670.65
2007-08	1138	49665.78	723
2008-09	1154	55971.4	741.2
2009-10	1170	64398.27	799.8
2010-11	1186	77023.08	844.8
2011-12	1202	89328.92	923.2
2012-13	1217	98345.81	964.5
2013-14	1239	111328.8	1014.8
2014-15	1254	123407.7	1048.4
2015-16	1324	135226.6	1090.85
2016-17	1350	149941.1	1135.33
2017-18	1354	166275.9	1206.31
2018-19	1369	190100	1249.34

B. brentcrude.csv

	TrDate	dollarsPB
1	2-Jan-19	54.06
2	3-Jan-19	53.23
3	4-Jan-19	55.64
4	7-Jan-19	57.10
5	8-Jan-19	56.91
6	9-Jan-19	59.46
7	10-Jan-19	60.47

8	11-Jan-19	59.24
9	14-Jan-19	58.80
10	15-Jan-19	58.65
11	16-Jan-19	59.81
12	17-Jan-19	59.85
13	18-Jan-19	62.04
14	21-Jan-19	62.18
15	22-Jan-19	60.90
16	23-Jan-19	61.05
17	24-Jan-19	61.09
18	25-Jan-19	61.49
19	28-Jan-19	59.71
20	29-Jan-19	60.98
21	30-Jan-19	61.89
22	31-Jan-19	62.46
23	1-Feb-19	61.86
24	4-Feb-19	62.26
25	5-Feb-19	61.67
26	6-Feb-19	62.22
27	7-Feb-19	61.01
28	8-Feb-19	61.37
29	11-Feb-19	61.30
30	12-Feb-19	62.58
31	13-Feb-19	63.27
32	14-Feb-19	64.00
33	15-Feb-19	65.65
34	18-Feb-19	66.41
35	19-Feb-19	65.86
36	20-Feb-19	66.82
37	21-Feb-19	66.91
38	22-Feb-19	66.91
39	25-Feb-19	64.02
40	26-Feb-19	64.51
41	27-Feb-19	65.55
42	28-Feb-19	65.03
43	1-Mar-19	63.71
44	4-Mar-19	64.44
45	5-Mar-19	64.24
46	6-Mar-19	64.51
47	7-Mar-19	64.82
48	8-Mar-19	65.66
49	11-Mar-19	65.06
50	12-Mar-19	65.33
51	13-Mar-19	65.89
52	14-Mar-19	66.18
53	15-Mar-19	66.11

54	18-Mar-19	66.65
55	19-Mar-19	67.13
56	20-Mar-19	68.35
57	21-Mar-19	68.30
58	22-Mar-19	66.29
59	25-Mar-19	67.37
60	26-Mar-19	67.51
61	27-Mar-19	67.35
62	28-Mar-19	66.08
63	29-Mar-19	67.93
64	1-Apr-19	69.08
65	2-Apr-19	69.68
66	3-Apr-19	69.21
67	4-Apr-19	69.80
68	5-Apr-19	69.93
69	8-Apr-19	71.12
70	9-Apr-19	71.02
71	10-Apr-19	71.63
72	11-Apr-19	71.30
73	12-Apr-19	71.57
74	15-Apr-19	70.90
75	16-Apr-19	70.74
76	17-Apr-19	71.14
77	18-Apr-19	70.71
78	22-Apr-19	70.71
79	23-Apr-19	74.39
80	24-Apr-19	73.59
81	25-Apr-19	74.94
82	26-Apr-19	71.03
83	29-Apr-19	71.22
84	30-Apr-19	72.19
85	1-May-19	72.01
86	2-May-19	70.56
87	3-May-19	71.95
88	6-May-19	71.95
89	7-May-19	70.98
90	8-May-19	71.09
91	9-May-19	70.61
92	10-May-19	71.63
93	13-May-19	72.35
94	14-May-19	72.53
95	15-May-19	73.09
96	16-May-19	74.70
97	17-May-19	73.94
98	20-May-19	73.21
99	21-May-19	72.94

100	22-May-19	71.94
101	23-May-19	68.37
102	24-May-19	67.98
103	28-May-19	70.19
104	29-May-19	70.64
105	30-May-19	69.55
106	31-May-19	66.78
107	3-Jun-19	63.16
108	4-Jun-19	63.56
109	5-Jun-19	62.14
110	6-Jun-19	62.77
111	7-Jun-19	64.10
112	10-Jun-19	64.31
113	11-Jun-19	63.56
114	12-Jun-19	61.66
115	13-Jun-19	63.28
116	14-Jun-19	63.13
117	17-Jun-19	62.56
118	18-Jun-19	63.35
119	19-Jun-19	62.85
120	20-Jun-19	65.44
121	21-Jun-19	65.99
122	24-Jun-19	65.16
123	25-Jun-19	66.24
124	26-Jun-19	66.85
125	27-Jun-19	66.78
126	28-Jun-19	67.52
127	1-Jul-19	65.10
128	2-Jul-19	62.72
129	3-Jul-19	63.53
130	4-Jul-19	63.62
131	5-Jul-19	64.23
132	8-Jul-19	64.89
133	9-Jul-19	64.30
134	10-Jul-19	66.41
135	11-Jul-19	67.64
136	12-Jul-19	66.65
137	15-Jul-19	66.86
138	16-Jul-19	65.87
139	17-Jul-19	63.67
140	18-Jul-19	60.70
141	19-Jul-19	61.04
142	22-Jul-19	61.96
143	23-Jul-19	62.28
144	24-Jul-19	63.83
145	25-Jul-19	63.47

146	26-Jul-19	62.46
147	29-Jul-19	62.29
148	30-Jul-19	62.55
149	31-Jul-19	64.07
150	1-Aug-19	62.90
151	2-Aug-19	61.12
152	5-Aug-19	59.32
153	6-Aug-19	58.63
154	7-Aug-19	55.03
155	8-Aug-19	56.29
156	9-Aug-19	57.37
157	12-Aug-19	57.13
158	13-Aug-19	59.90
159	14-Aug-19	57.86
160	15-Aug-19	57.37
161	16-Aug-19	59.00
162	19-Aug-19	59.79
163	20-Aug-19	59.03
164	21-Aug-19	60.60
165	22-Aug-19	59.81
166	23-Aug-19	58.64
167	26-Aug-19	58.64
168	27-Aug-19	58.44
169	28-Aug-19	60.42
170	29-Aug-19	60.59
171	30-Aug-19	61.04
172	2-Sep-19	58.55
173	3-Sep-19	57.93
174	4-Sep-19	60.68
175	5-Sep-19	62.70
176	6-Sep-19	61.28
177	9-Sep-19	63.99
178	10-Sep-19	64.67
179	11-Sep-19	63.02
180	12-Sep-19	60.76
181	13-Sep-19	61.25
182	16-Sep-19	68.42
183	17-Sep-19	65.59
184	18-Sep-19	64.29
185	19-Sep-19	64.25
186	20-Sep-19	65.23
187	23-Sep-19	64.66
188	24-Sep-19	64.13
189	25-Sep-19	62.41
190	26-Sep-19	62.08
191	27-Sep-19	62.48

192	30-Sep-19	60.99
193	1-Oct-19	60.06
194	2-Oct-19	57.92
195	3-Oct-19	58.01
196	4-Oct-19	59.13
197	7-Oct-19	59.46
198	8-Oct-19	58.14
199	9-Oct-19	59.70
200	10-Oct-19	59.08
201	11-Oct-19	60.59
202	14-Oct-19	58.81
203	15-Oct-19	59.19
204	16-Oct-19	59.30
205	17-Oct-19	59.35
206	18-Oct-19	59.96
207	21-Oct-19	58.95
208	22-Oct-19	60.50
209	23-Oct-19	60.52
210	24-Oct-19	61.71
211	25-Oct-19	62.06
212	28-Oct-19	60.39
213	29-Oct-19	61.05
214	30-Oct-19	60.22
215	31-Oct-19	59.30
216	1-Nov-19	60.17
217	4-Nov-19	62.52
218	5-Nov-19	62.72
219	6-Nov-19	62.11
220	7-Nov-19	62.60
221	8-Nov-19	62.00
222	11-Nov-19	62.58
223	12-Nov-19	62.19
224	13-Nov-19	62.27
225	14-Nov-19	62.46
226	15-Nov-19	63.32
227	18-Nov-19	62.82
228	19-Nov-19	62.37
229	20-Nov-19	63.80
230	21-Nov-19	64.99
231	22-Nov-19	64.83
232	25-Nov-19	64.67
233	26-Nov-19	64.82
234	27-Nov-19	65.03
235	28-Nov-19	64.68
236	29-Nov-19	64.50
237	2-Dec-19	63.20

238	3-Dec-19	62.95
239	4-Dec-19	65.25
240	5-Dec-19	65.67
241	6-Dec-19	66.50
242	9-Dec-19	66.44
243	10-Dec-19	66.57
244	11-Dec-19	65.37
245	12-Dec-19	66.67
246	13-Dec-19	67.44
247	16-Dec-19	68.04
248	17-Dec-19	68.99
249	18-Dec-19	69.12
250	19-Dec-19	69.70
251	20-Dec-19	68.66
252	23-Dec-19	67.49
253	24-Dec-19	69.26
254	27-Dec-19	68.91
255	30-Dec-19	68.30
256	31-Dec-19	67.77
257	2-Jan-20	67.05
258	3-Jan-20	69.08
259	6-Jan-20	70.25
260	7-Jan-20	68.74
261	8-Jan-20	67.31
262	9-Jan-20	66.58
263	10-Jan-20	66.77
264	13-Jan-20	64.14
265	14-Jan-20	64.45
266	15-Jan-20	63.29
267	16-Jan-20	64.63
268	17-Jan-20	64.05
269	20-Jan-20	64.63
270	21-Jan-20	63.66

ANNEXURE-II: REFERENCE AND BIBLIOGRAPHY

1. David Anderson et al (2011), Statistics for Business and Economics 9e, Cengage Learning, New Delhi – 110092, ISBN-10: 81-315-0288-0
2. Montgomery D.C. and Runger G.C. (2014), Applied Statistics and Probability for Engineers 6e, Wiley, New Jeresy, ISBN-13 9781118539712
3. Peter Dalgaard (2008), Introductory Statistics with R 2e, Springer, New York, ISBN: 978-0-387-79053-4
4. Kabacoff R.I. (2011), R in Action: Data Analysis and Graphics with R, Manning, Shelter Island, New York, ISBN: 9781935182399
5. R-CRAN website: https://www.r-project.org/

Author's other ebooks:

1. **Basics of Data Analysis Using R**
2. **Estimation and Hypothesis Testing with R**
3. **Regression Analysis with R**
4. **Technical Analysis using R**