//Time series analysis

//DATA

my

1406 observations of 6 variables

	Date	Open	High	Low	Close	Volume
1	2015-03-20	47.74	48.59	47.54	48.06	3652022
2	2015-03-19	46.93	46.98	46.51	46.60	2745271
3	2015-03-18	45.74	47.30	45.23	47.24	3525538
4	2015-03-17	45.91	46.34	45.70	46.14	2473132
5	2015-03-16	44.91	45.25	44.56	45.22	2123696
6	2015-03-13	45.16	45.22	44.07	44.55	3288800
7	2015-03-12	46.59	46.74	45.77	45.95	2115058
8	2015-03-11	45.91	45.93	45.32	45.58	2505461
9	2015-03-10	48.17	48.27	47.32	47.45	2897724
10	2015-03-09	49.64	49.74	49.26	49.44	2264531
11	2015-03-06	50.14	50.37	49.20	49.35	3126614
12	2015-03-05	51.61	51.68	50.84	50.93	2081923
13	2015-03-04	51.59	51.72	51.03	51.62	2232139
14	2015-03-03	52.18	52.58	51.89	51.99	1459722
15	2015-03-02	52.24	52.34	51.72	52.30	2904957
16	2015-02-27	52.31	52.61	52.23	52.52	2143380
17	2015-02-26	52.47	52.53	51.72	51.86	1788248
18	2015-02-25	52.63	52.76	52.24	52.55	1959566
19	2015-02-24	51.99	52.98	51.52	52.31	3736534
20	2015-02-23	49.82	49.92	49.26	49.55	2507006
21	2015-02-20	50.47	50.72	50.29	50.48	2225967
22	2015-02-19	50.56	50.77	50.22	50.47	1935967
23	2015-02-18	50.47	51.61	50.45	51.06	2490657
24	2015-02-17	51.17	51.24	50.79	50.99	3141354
25	2015-02-13	50.39	51.59	50.31	51.19	4193047
26	2015-02-12	48.54	48.98	48.25	48.81	2587431

Step 1://Reading the data into R

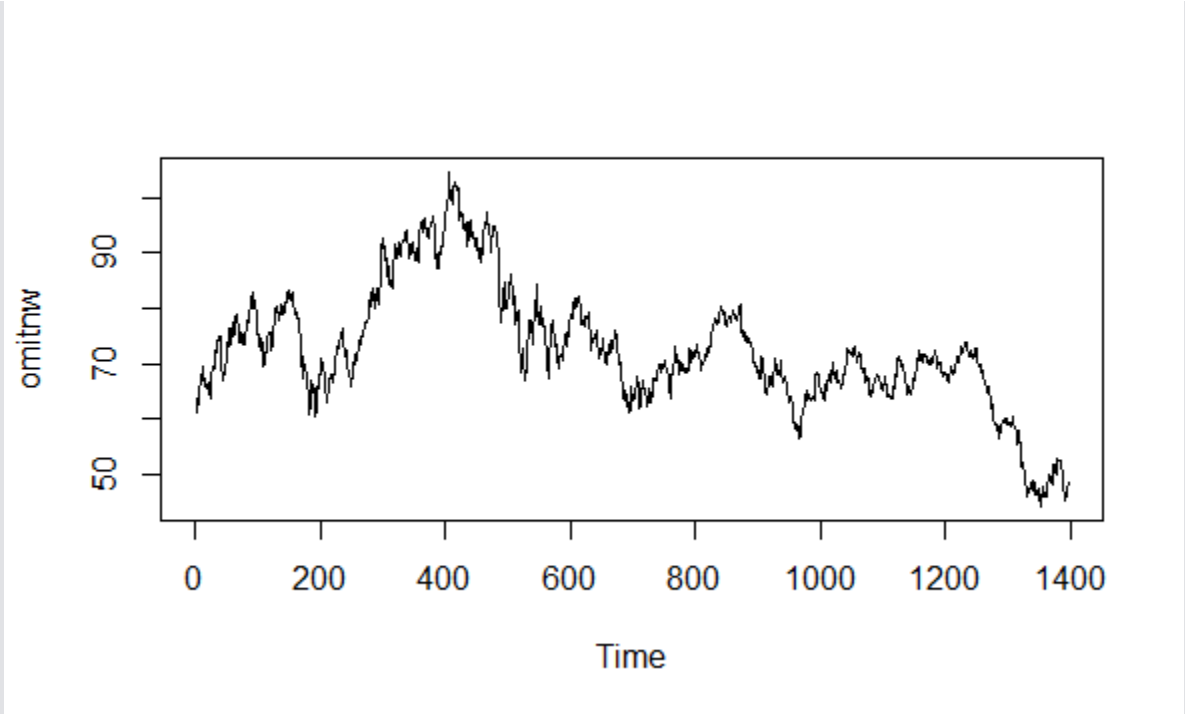
```
> my <- read.csv("C:/Users/user/Desktop/my.txt")
>   View(my)
```

Step 2://Arrange the data into decreasing order

```
> mytable<-data.table(my)
> newdata <- myt[order(Date)]
> newdata <- mytable[order(Date)]
> nw<-newdata$High
```

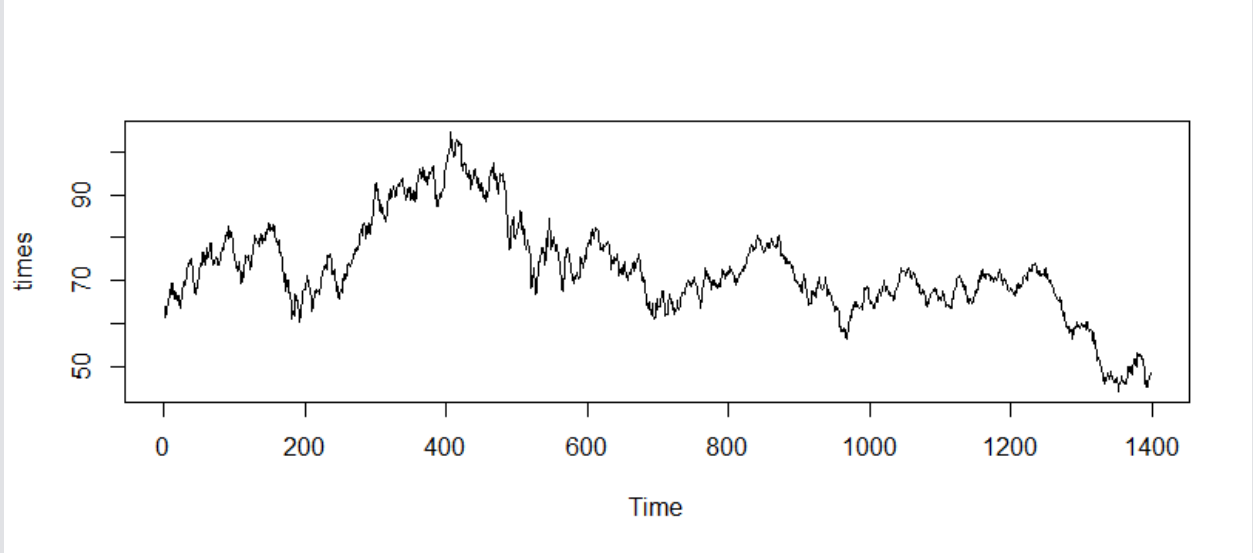
Step 3://Preprocessing the data

```
> omitnw<-na.omit(nw)
> plot.ts(omitnw)
```



Step 4 :Creating the time series object

```
> times<-ts(omitnw)
> plot.ts(times)
```



Step 5:// Decomposing the time series into 3 components:

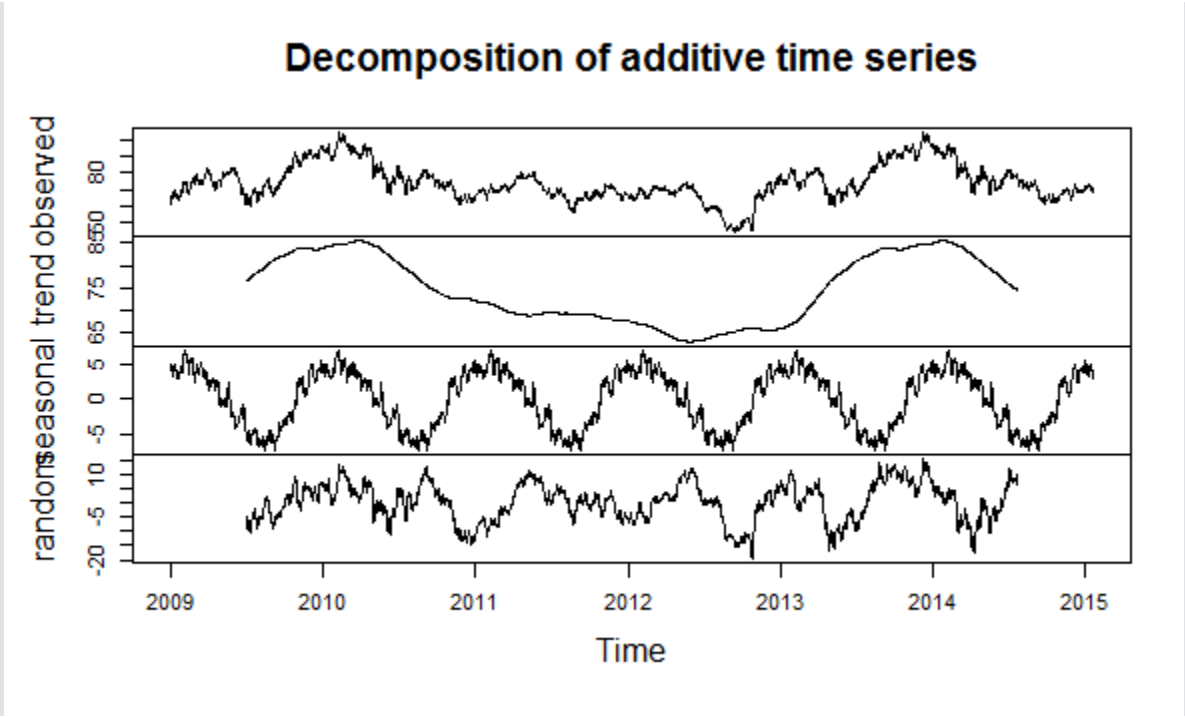
- Seasonal
- trend
- random
-

```
> times1<-ts(omitnw,frequency=365, start=c(2009,1),end=c(2015,20))
 > dtimes1<-decompose(times1)
 > dtimes1$seasonal
```

```
Time Series:
Start = c(2009, 1)
End = c(2015, 20)
Frequency = 365
   [1]  4.27441621  5.00317238  5.09191758  4.90973128  4.49938060  3.50311210
   [7]  3.36835046  3.82899156  3.83450388  4.57522717  4.95684909  4.87142991
  [13]  4.33045457  4.22153950  4.10978060  3.90214772  4.18641621  3.55622443
  [19]  3.07208197  3.32441621  3.14263539  3.42173128  3.27583539  3.56519977
  [25]  4.17179156  3.82671210  4.01507375  4.79027375  3.90018060  3.80907923
  [31]  4.58404361  5.57111210  6.33219156  6.37319977  6.53140799  6.60902443
  [37]  6.86038334  7.12738060  6.30870114  6.52949019  6.31794498  5.61832854
  [43]  5.68147923  4.78231210  3.65325457  3.49073950  4.01482169  5.50208745
  [49]  5.27408197  5.34396142  5.39001621  5.78782991  5.98140251  5.80265183
  [55]  5.83079430  4.91642169  4.19162717  2.88686005  2.97819156  2.45218060
  [61]  3.77932580  3.38732580  4.15512854  3.83984087  4.78653128  4.71530388
  [67]  4.97967649  4.13033950  3.73265183  3.54124909  3.64446553  4.33999430
  [73]  4.79287649  5.04784635  5.34159977  4.76750662  5.00477786  4.45020251
  [79]  3.55014224  3.31232854  3.77334772  4.22348471  2.99818060  3.14912854
  [85]  3.77911758  3.72258608  3.39751758  3.92926005  3.33294772  3.59212032
  [91]  2.80967649  1.26522169  1.41666827  1.46604361  2.24389567  2.83747375
  [97]  2.64723813  2.53582991  1.96496964  2.34234498  2.84199430  2.84761073
 [103]  2.89677786  2.61866279  2.39942443  2.60596142  1.58798882  1.25397238
 [109]  1.21171484  2.77438882  2.00094224  1.73630662  2.60369840  2.83189019
 [115]  2.68235046  2.51750114  2.19626279  1.74070662  0.59288197 -0.47457009
 [121] -1.05642762  0.08438334 -0.30781392  0.03679977 -0.38487146 -0.60371529
 [127] -0.47686050 -0.19282214 -0.63221940  0.03862443 -0.47718379 -0.66745228
 [133] -1.03997831 -0.99330433 -0.54706872 -0.78989612 -0.96517283  0.18493676
```

```
 [361]  4.82375320  5.05232854  5.56352306  4.38283265  3.86474498  4.27441621
 [367]  5.00317238  5.09191758  4.90973128  4.49938060  3.50311210  3.36835046
 [373]  3.82899156  3.83450388  4.57522717  4.95684909  4.87142991  4.33045457
 [379]  4.22153950  4.10978060  3.90214772  4.18641621  3.55622443  3.07208197
 [385]  3.32441621  3.14263539  3.42173128  3.27583539  3.56519977  4.17179156
 [391]  3.82671210  4.01507375  4.79027375  3.90018060  3.80907923  4.58404361
 [397]  5.57111210  6.33219156  6.37319977  6.53140799  6.60902443  6.86038334
 [403]  7.12738060  6.30870114  6.52949019  6.31794498  5.61832854  5.68147923
 [409]  4.78231210  3.65325457  3.49073950  4.01482169  5.50208745  5.27408197
 [415]  5.34396142  5.39001621  5.78782991  5.98140251  5.80265183  5.83079430
 [421]  4.91642169  4.19162717  2.88686005  2.97819156  2.45218060  3.77932580
 [427]  3.38732580  4.15512854  3.83984087  4.78653128  4.71530388  4.97967649
 [433]  4.13033950  3.73265183  3.54124909  3.64446553  4.33999430  4.79287649
```

```
> plot(dtimes1)
```

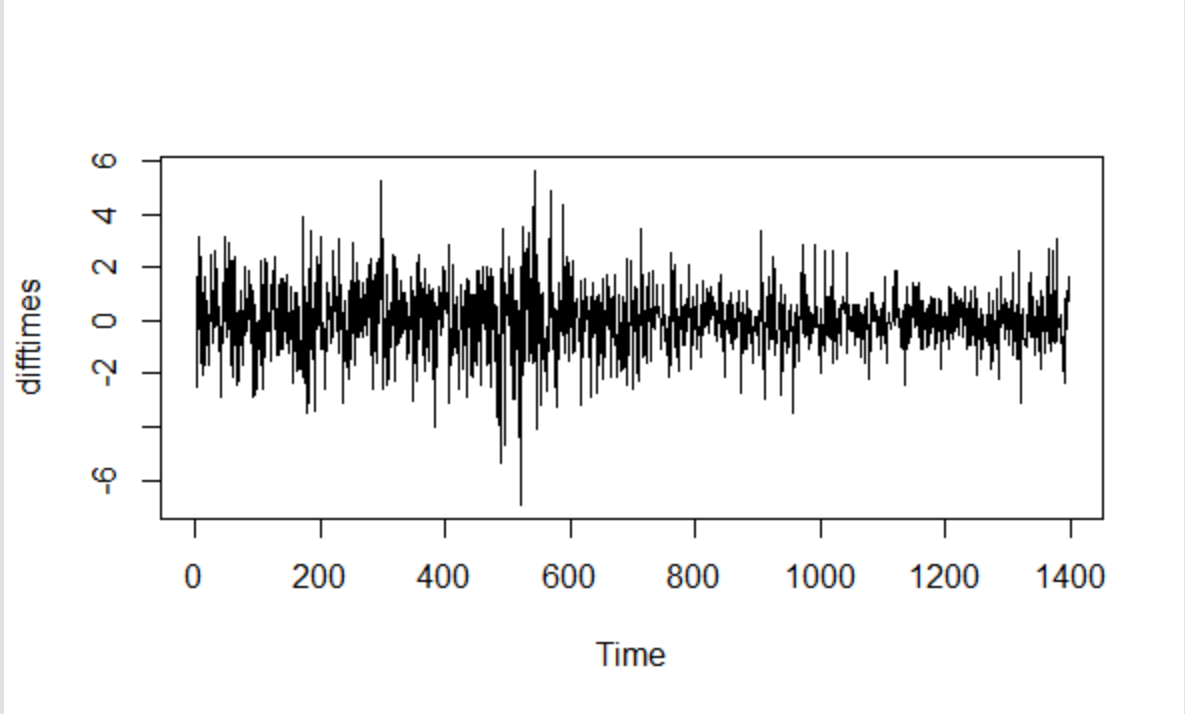


Step 6://Arima Model:

Step 6.1: Differencing the time series

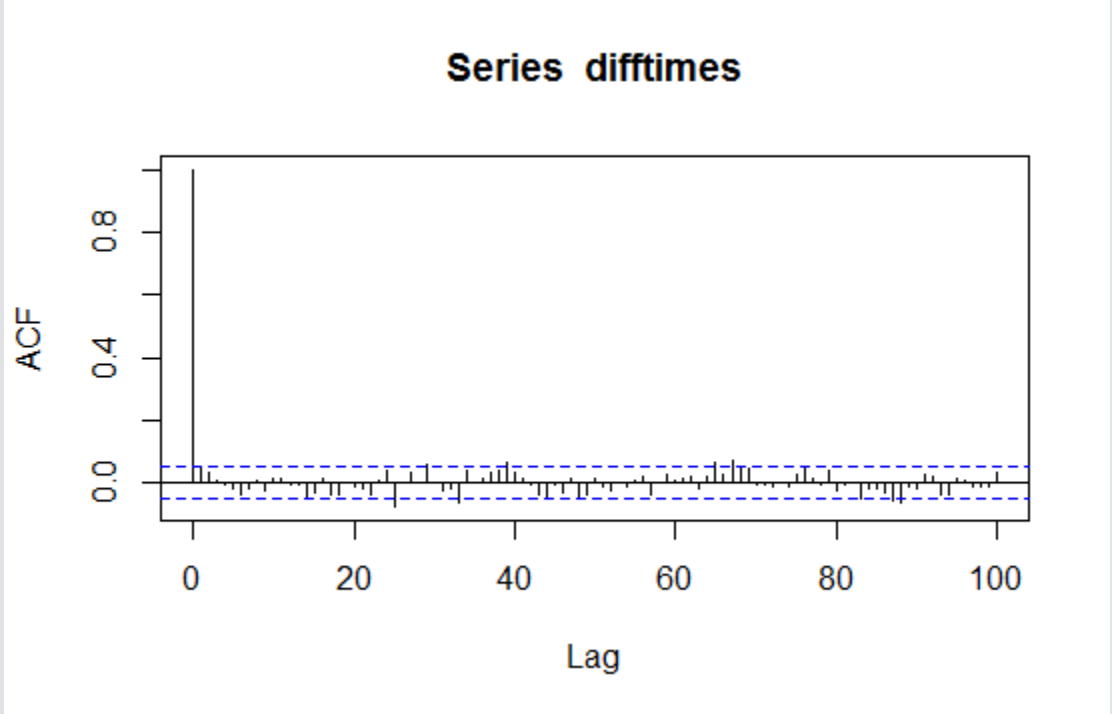
//Applying DIFFERENCING

```
> difftimes <- diff(times, differences=1)
> plot.ts(difftimes)
```

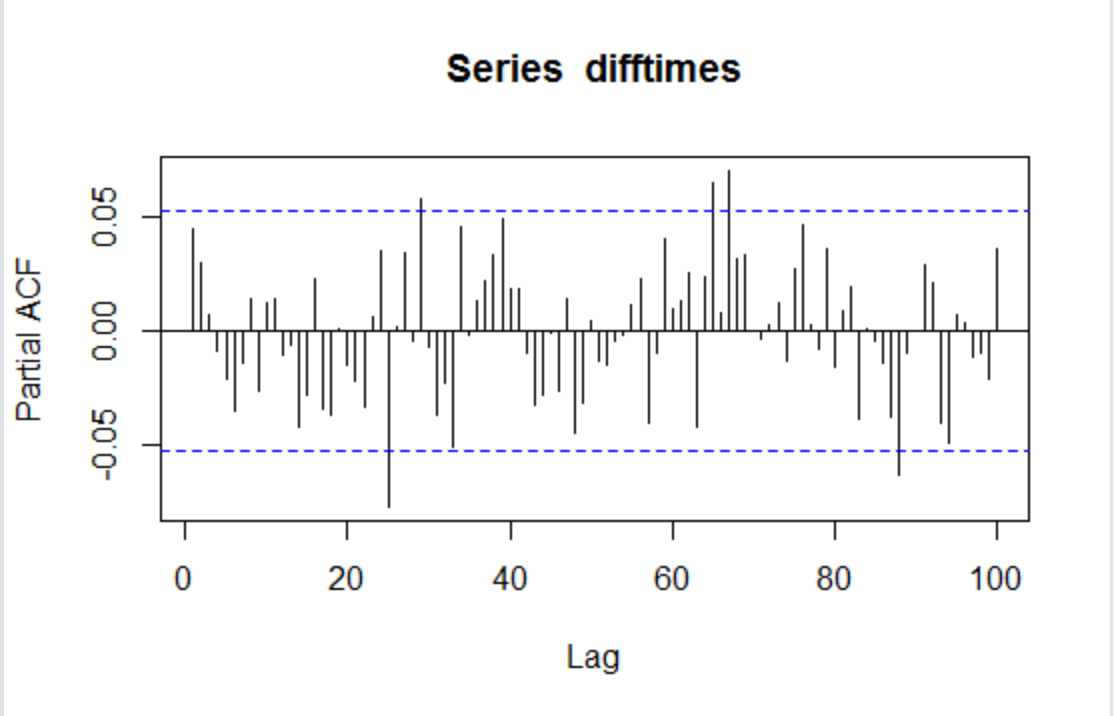


//Applying ACF

```
> acf(difftimes, lag.max=100)
```




```
> pacf(difftimes,lag.max=100)
```

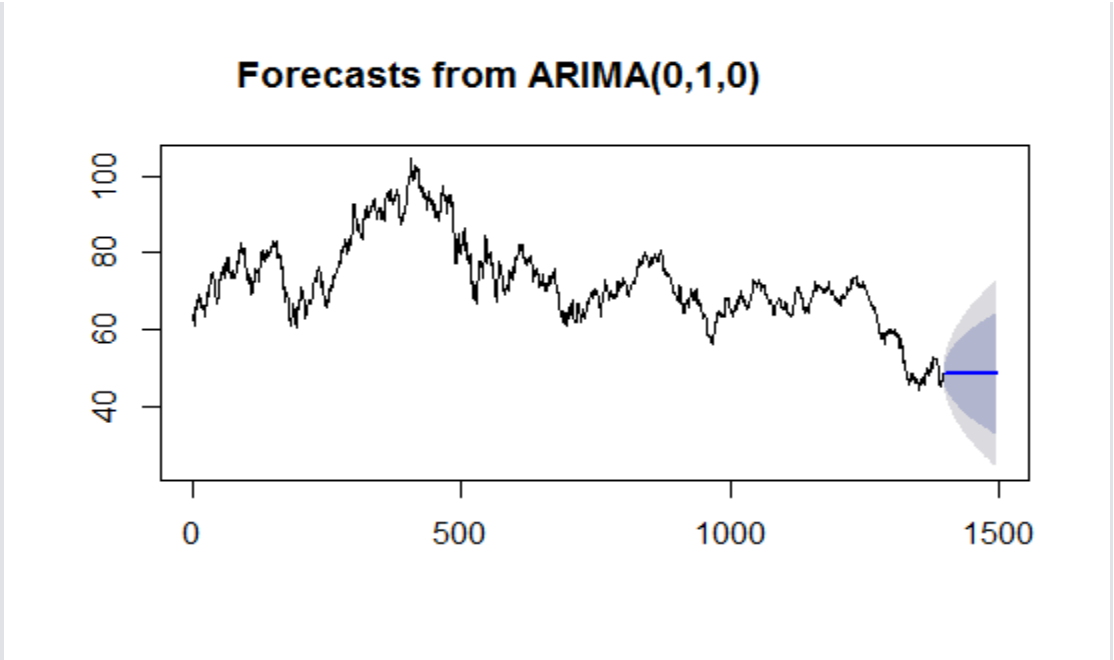


//model fitting

```
> finalfit<- arima(times, order=c(0,1,0))
```

//forcasting

```
> finalforcast <- forecast.Arima(finalfit, h=100)
> plot.forecast(finalforcast)
```

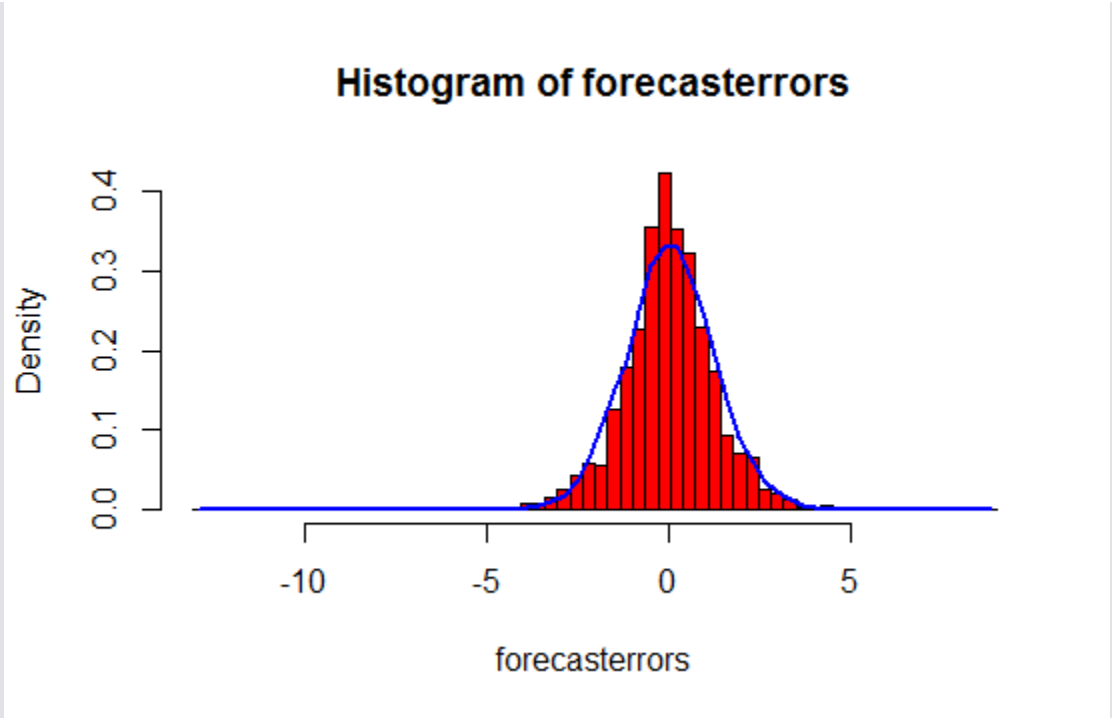


Step 7://Model Evaluation:

7.1 //Plotting forecast errors

```
plotForecastErrors <- function(forecasterrors)
+ {
+     # make a histogram of the forecast errors:
+     mybinsize <- IQR(forecasterrors)/4
+     mysd   <- sd(forecasterrors)
+     mymin  <- min(forecasterrors) - mysd*5
+     mymax  <- max(forecasterrors) + mysd*3
+     # generate normally distributed data with mean 0 and standard deviation
mysd
+     mynorm <- rnorm(10000, mean=0, sd=mysd)
+     mymin2 <- min(mynorm)
+     mymax2 <- max(mynorm)
+     if (mymin2 < mymin) { mymin <- mymin2 }
+     if (mymax2 > mymax) { mymax <- mymax2 }
+     # make a red histogram of the forecast errors, with the normally
distributed data overlaid:
+     mybins <- seq(mymin, mymax, mybinsize)
+     hist(forecasterrors, col="red", freq=FALSE, breaks=mybins)
+     # freq=FALSE ensures the area under the histogram = 1
+     # generate normally distributed data with mean 0 and standard deviation
mysd
+     myhist <- hist(mynorm, plot=FALSE, breaks=mybins)
+     # plot the normal curve as a blue line on top of the histogram of
forecast errors:
+     points(myhist$mids, myhist$density, type="l", col="blue", lwd=2)
+ }

> plotForecastErrors(finalforcast$residuals)
```



7.2//Superimposing the graphs of training and testing data

```
> `t1` <- read.csv("C:/Users/user/Desktop/t1.txt", header=FALSE)
>   View(`t1`)
> tt1<-data.table(t1)
> ntt1 <- tt1[order(V1)]
> high<-ntt1$V3
> high<-na.omit(high)
> timestrain1<-ts(high,frequency=365, start=c(2009,1),end=c(2014,31))
> finalfit2<- arima(timestrain1, order=c(0,1,0))
> finalforcast2 <- forecast.Arima(finalfit2, h=300)
> plot(finalforcast2)
> par(new=TRUE)
> plot(times1)
```