FORECASTING FOREIGN EXCHANGE RATE

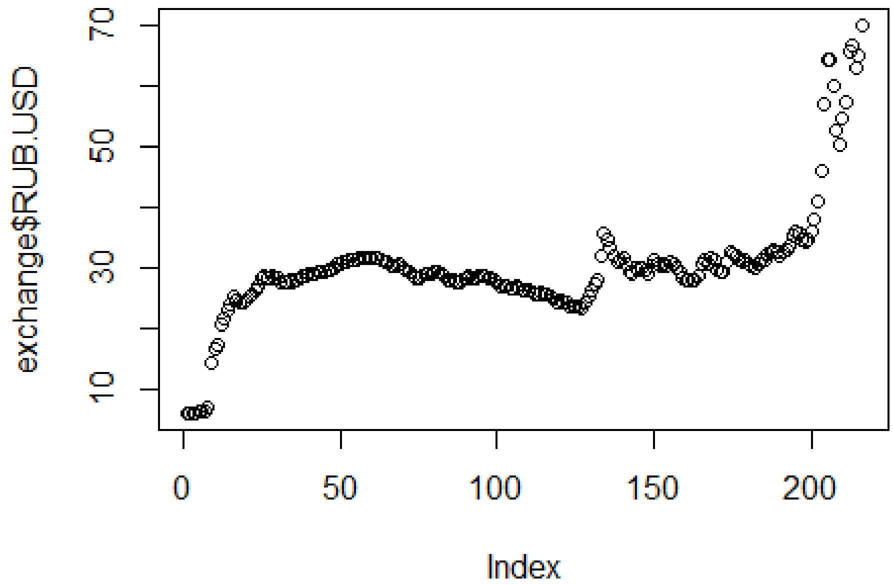
Soukhyada Vaidya
Roshni Suhanda
Daksh Gupta
Shravankumar Hiregoudar

```
library(readr)
library(tseries)
library(forecast)

exchange <- read.csv("E:/MITA/Bf/project/exchange.csv", header = TRUE)
summary(exchange)

##        ï..DATE        BRL.USD          RUB.USD          INR.USD
##  01-01-1998:  1   Min.   :1.122   Min.   : 5.998   Min.   :38.79
##  01-01-1999:  1   1st Qu.:1.789   1st Qu.:27.260   1st Qu.:43.98
##  01-01-2000:  1   Median :2.114   Median :29.203   Median :46.06
##  01-01-2001:  1   Mean   :2.206   Mean   :30.174   Mean   :48.41
##  01-01-2002:  1   3rd Qu.:2.468   3rd Qu.:31.352   3rd Qu.:50.08
##  01-01-2003:  1   Max.   :3.892   Max.   :70.188   Max.   :66.55
##  (Other)   :210
##     YUAN.USD
##  Min.   :6.052
##  1st Qu.:6.471
##  Median :7.805
##  Mean   :7.425
##  3rd Qu.:8.279
##  Max.   :8.307
##
```

The currencies are: Brazilian Real Russian Ruble Indian rupees Chinese Yuan

```
head (exchange)

##      ï..DATE BRL.USD RUB.USD INR.USD YUAN.USD
## 1 01-01-1998  1.1218  6.0100 39.5402   8.2940
## 2 01-02-1998  1.1264  5.9983 38.7942   8.2780
## 3 01-03-1998  1.1322  6.0100 39.4955   8.2771
## 4 01-04-1998  1.1359  6.0449 39.4695   8.2514
## 5 01-05-1998  1.1441  6.1356 40.1854   8.2621
## 6 01-06-1998  1.1532  6.2005 42.0108   8.2816
```

```
plot(exchange$RUB.USD)
```



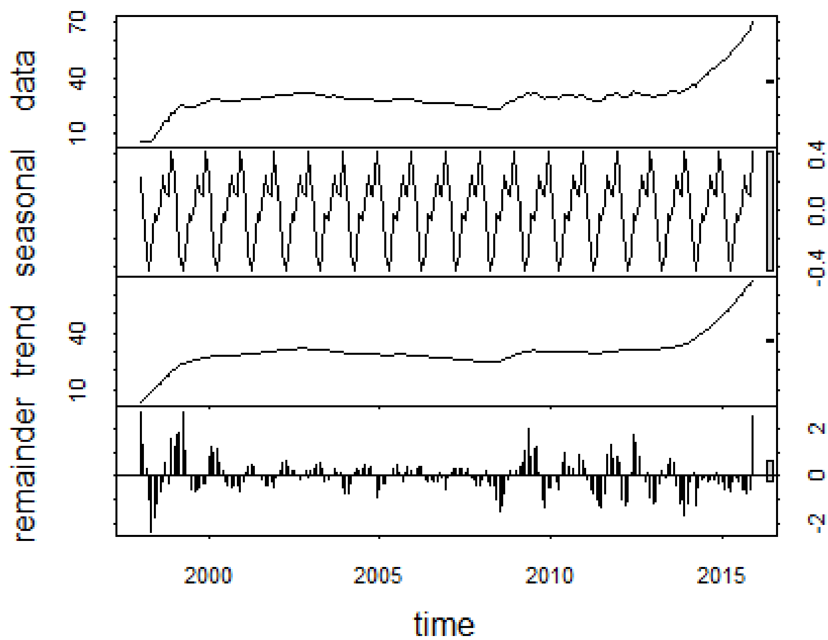
Create a time series object

```
exchange <- read.csv("E:/MITA/Bf/project/exchange.csv", header = TRUE)

timeseries=ts(exchange[,"RUB.USD"],frequency=12, start=1998)
exchange$RUB.USD=tsclean(timeseries)
```

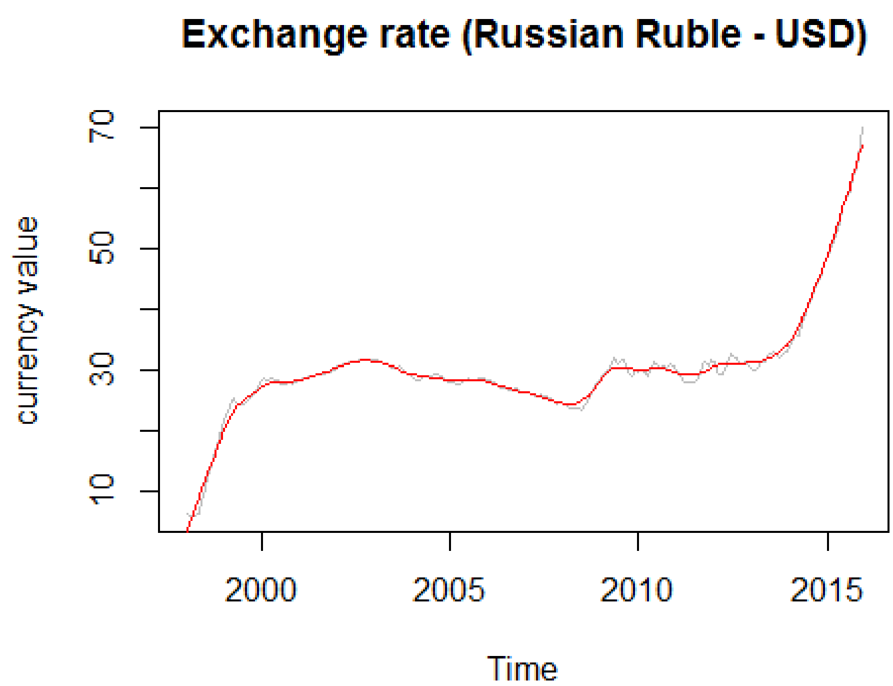
Decomposition of time series

It calculate the seasonal component using smoothing and then adjust the original series for seasonality. The result is a seasonality adjusted time series

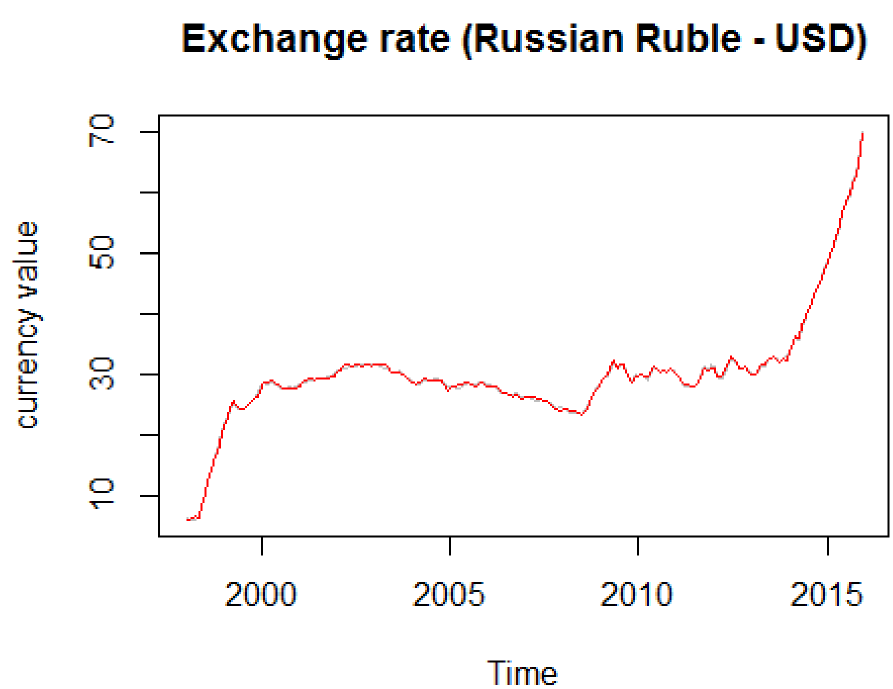
```
count_ts=ts(na.omit(exchange$RUB.USD), frequency=12, start = 1998)

# Making the data into trend,seasonal and reminder
decomp <- stl(count_ts, s.window = "periodic")

# Seasonally adjusting the data
deseasonal_ts <- seasadj(decomp)
plot(decomp)
```




```
# Ploting the data with trend
plot(count_ts, col="gray", main="Exchange rate (Russian Ruble - USD)",
ylab="currency value", xlab="Time")

# 1 is seasonal, 2 is trend and 3 is reminder
lines(decomp$time.series[,2],col="red")
```




```
plot(count_ts, col="grey", main="Exchange rate (Russian Ruble - USD)",
ylab="currency value", xlab="Time")
lines(seasadj(decomp),col="red",ylab="Seasonally adjusted")
```



Stationarity test

Augmented Dickey-Fuller test for stationarity/ non-stationarity of data. A large p-value(>0.05) suggests that the time serie is non-stationary (Reject H0 hypotesis)

Kwiatkowski-Phillips-Schmidt-Shin test of stationarity. In this case, if p-value are smaller then 5% differencing is required => for KPSS test H0:stationarity; H1:non-stationarity

```
adf<-adf.test(count_ts, alternative = "stationary")

## Warning in adf.test(count_ts, alternative = "stationary"): p-value greater
## than printed p-value

kpss<-kpss.test(count_ts)

## Warning in kpss.test(count_ts): p-value smaller than printed p-value

adf

##
##  Augmented Dickey-Fuller Test
##
## data:  count_ts
## Dickey-Fuller = 1.3739, Lag order = 5, p-value = 0.99
## alternative hypothesis: stationary

kpss

##
##  KPSS Test for Level Stationarity
##
## data:  count_ts
## KPSS Level = 1.8503, Truncation lag parameter = 4, p-value = 0.01
```

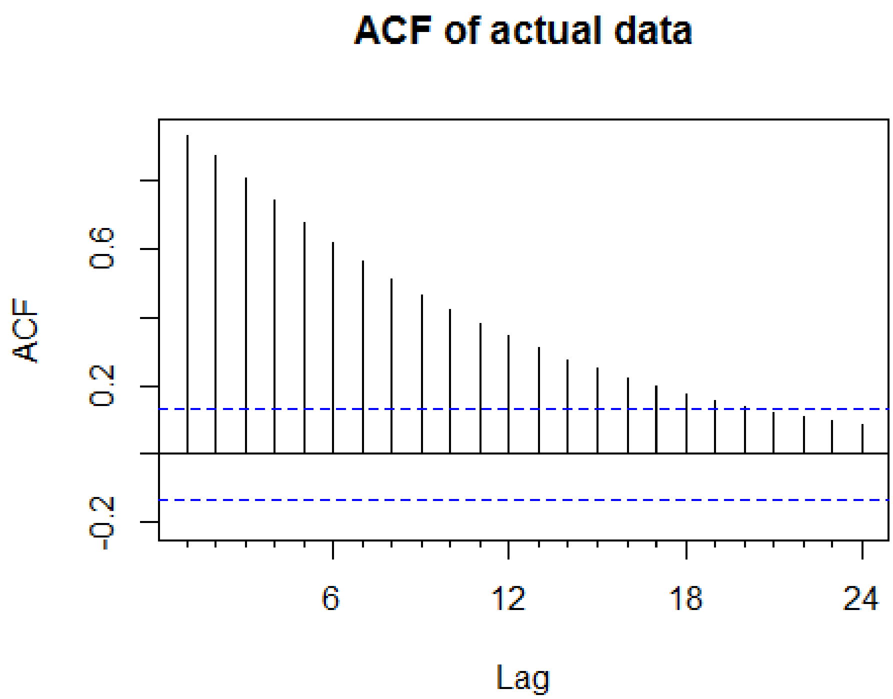
Autocorrelation is the linear correlation of a signal with itself at two different points in time, ACF (autocorrelation function) is just such correlation as a function of the lag h between two points of time. It correlates with itself through time.

PACF (partial autocorrelation function) is essentially the autocorrelation of a signal with itself at different points in time, with linear dependency with that signal at shorter lags removed, as a function of lag between points of time.

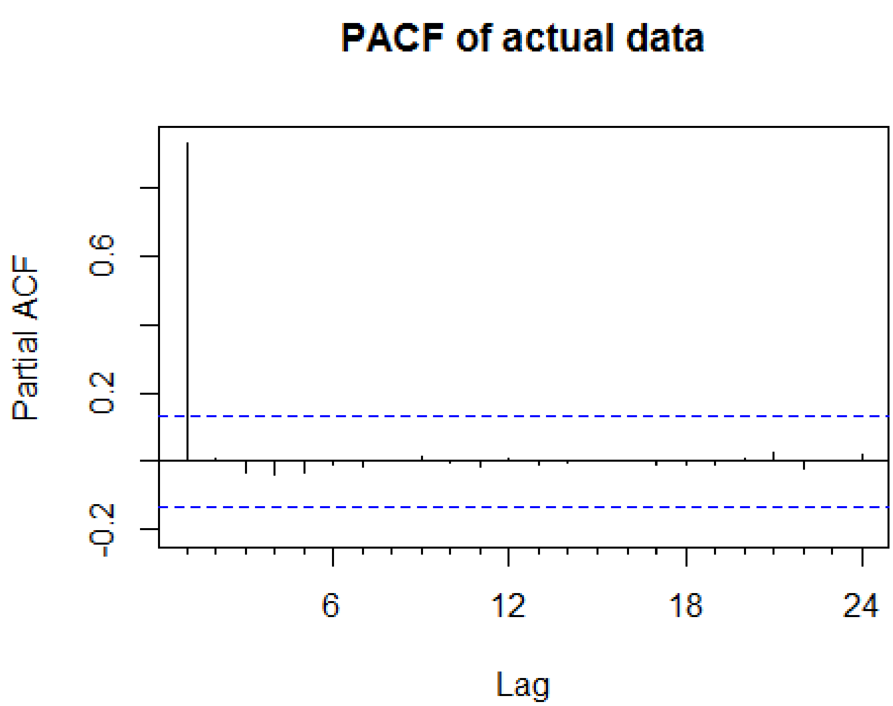
augmented Dickey–Fuller test (ADF) Null Hypothesis (H0): If failed to be rejected, it suggests the time series has a unit root, meaning it is non-stationary. It has some time dependent structure. Alternate Hypothesis (H1): The null hypothesis is rejected; it suggests the time series does not have a unit root, meaning it is stationary. p-value > 0.05: Fail to reject the null hypothesis (H0), the data has a unit root and is non-stationary. p-value <= 0.05: Reject the null hypothesis (H0), the data does not have a unit root and is stationary.

Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test The null hypothesis (H0) for the test is that the data is stationary. The alternate hypothesis (H1) for the test is that the data is not stationary. p-value > 0.05: Reject H0 p-value <= 0.05: Accept H0

```r
Acf(count_ts, main='ACF of actual data')
```




```r
Pacf(count_ts, main='PACF of actual data')
```



Results of Adf and Kpss test showed that the time series is not stationarity.

```
ndiffs(count_ts)

## [1] 2
```

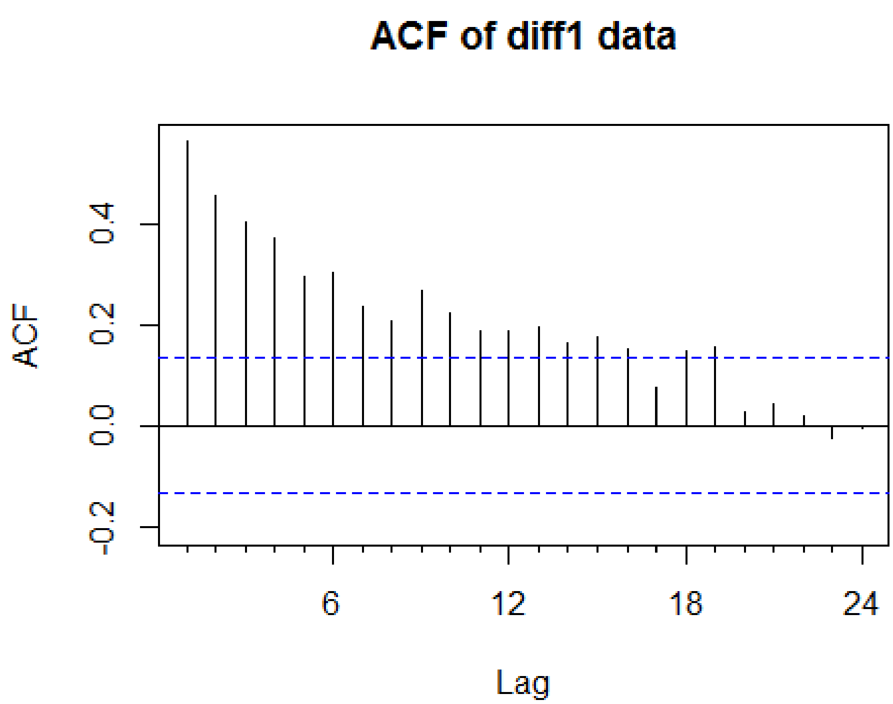
So, The data gets stationary after 2 differences.

Differencing the data d = 1;

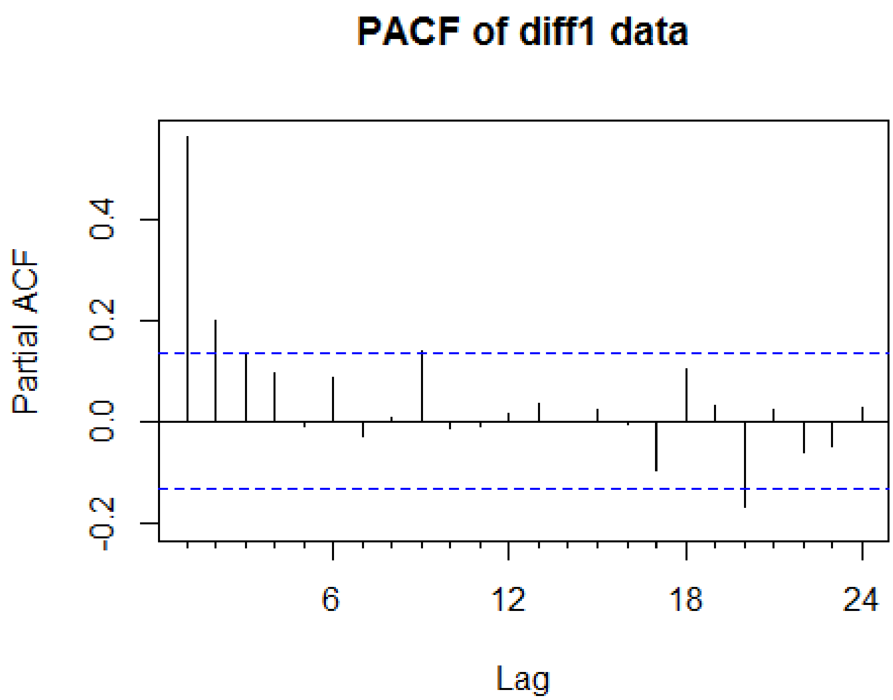
```
# deseasonal_ts <- seasadj(decomp) is seasonaly adjusted data
ts_d1=diff(deseasonal_ts, differences=1)

#stationary test for d=1
adfd1<-adf.test(ts_d1, alternative = "stationary")
adfd1

##
##  Augmented Dickey-Fuller Test
##
## data:  ts_d1
## Dickey-Fuller = -1.9613, Lag order = 5, p-value = 0.5922
## alternative hypothesis: stationary
```

Even now p > 0.05 suggests that the time serie is non-stationary (Reject H0 hypotesis)

```
Acf(ts_d1, main='ACF of diff1 data')
```




```
Pacf(ts_d1, main='PACF of diff1 data')
```

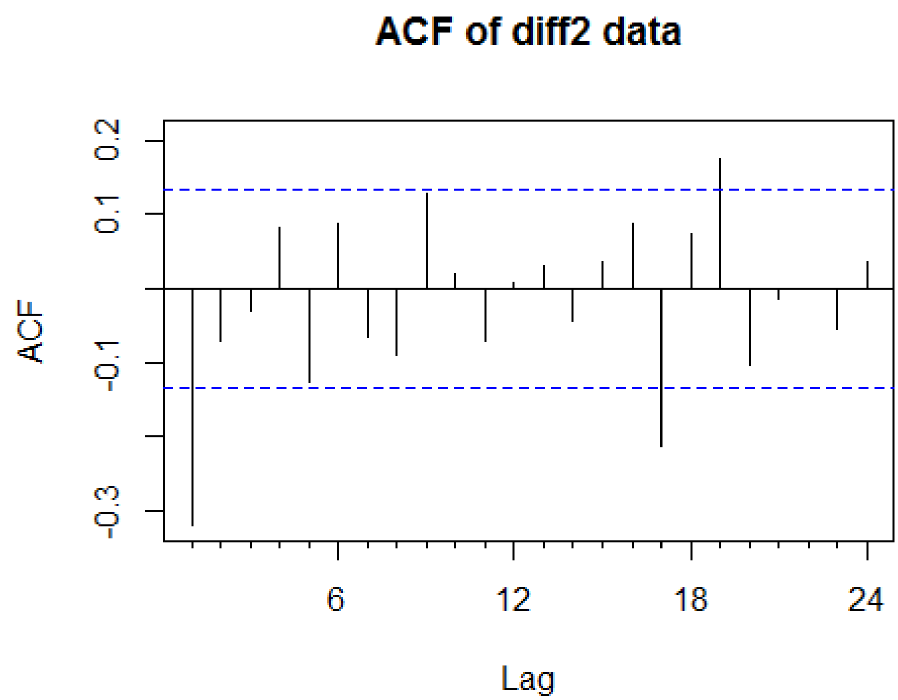


Differencing the data d = 2;

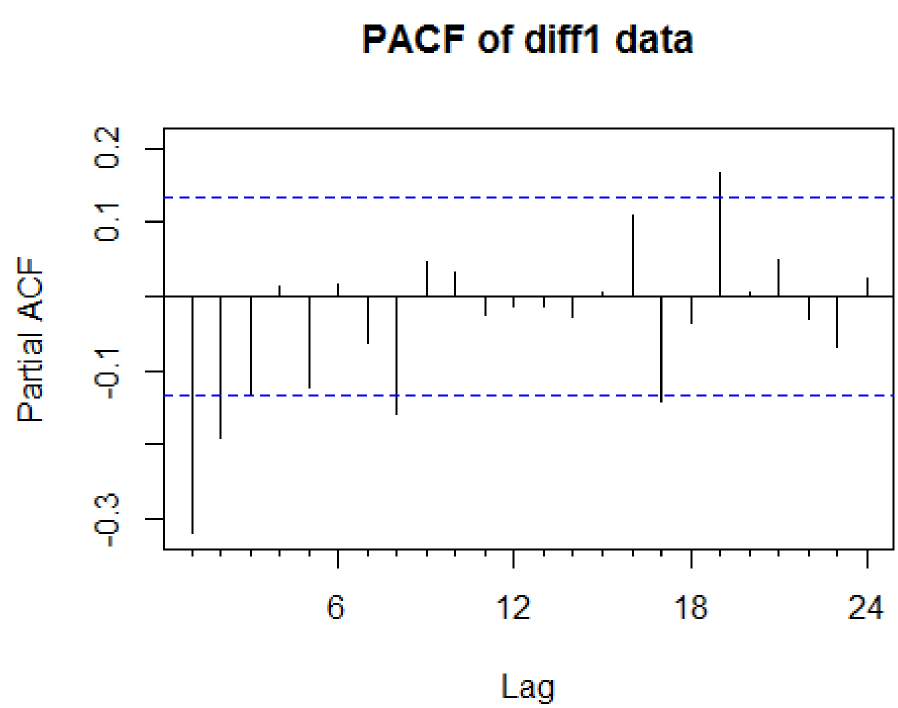
```
# deseasonal_ts <- seasadj(decomp) is seasonaly adjusted data
ts_d2=diff(ts_d1, differences=1)

#stationary test for d=2
adfd2<-adf.test(ts_d2, alternative = "stationary")

## Warning in adf.test(ts_d2, alternative = "stationary"): p-value smaller
## than printed p-value

adfd2

##
##  Augmented Dickey-Fuller Test
##
## data:  ts_d2
## Dickey-Fuller = -7.932, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

Now, $p < 0.05$ suggests that the time serie is stationary (Accept H0 hypotesis)

```
Acf(ts_d2, main='ACF of diff2 data')
```



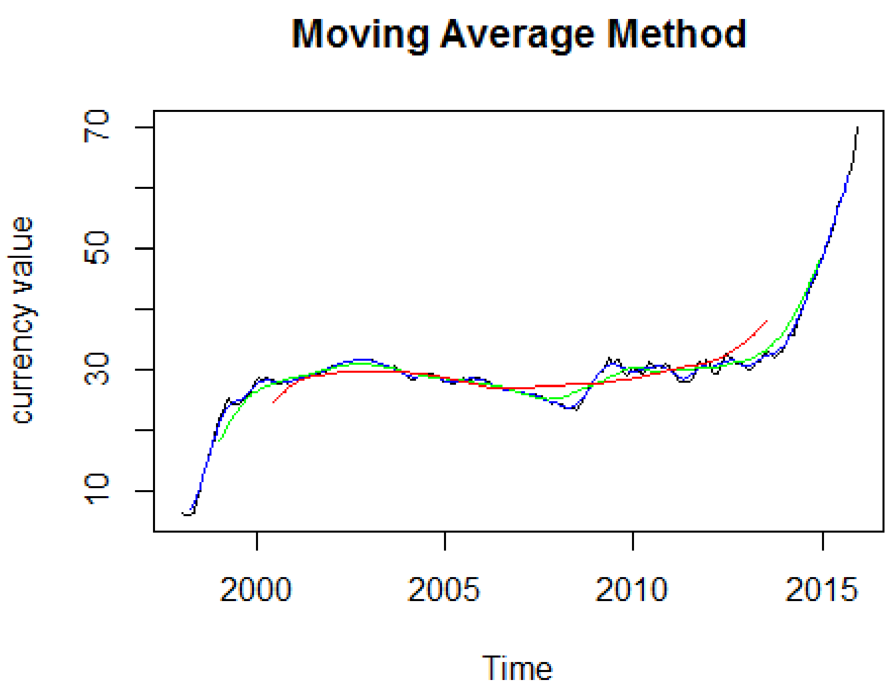

```
Pacf(ts_d2, main='PACF of diff1 data')
```



After a time series has been stationarized by differencing, the next step in fitting an ARIMA model is to determine whether AR or MA terms are needed to correct any autocorrelation that remains in the differenced series. By looking at the autocorrelation function (ACF) and partial autocorrelation (PACF) plots of the differenced series, you can tentatively identify the numbers of AR and/or MA terms that are needed. ACF plot: it is merely a bar chart of the coefficients of correlation between a time series and lags of itself. The PACF plot is a plot of the partial correlation coefficients between the series and lags of itself.

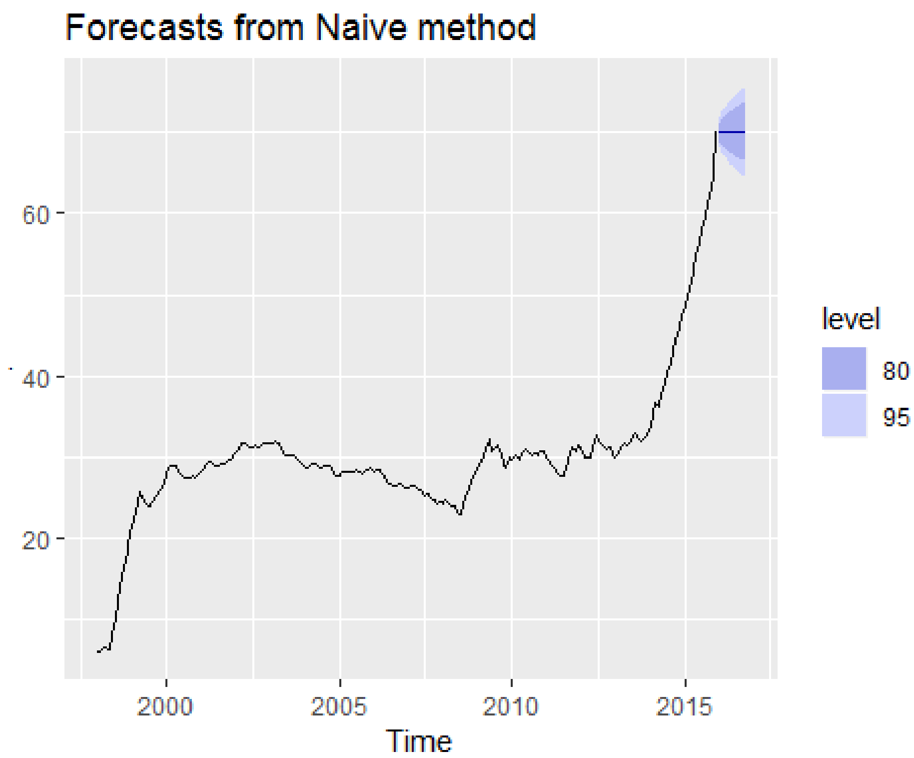
Moving average of 7 for the stationary data

```
plot(count_ts, main="Moving Average Method", ylab="currency value",
xlab="Time")
lines(ma(count_ts,7),col="blue")
lines(ma(count_ts,25), col="green")
lines(ma(count_ts,59), col="red")
```

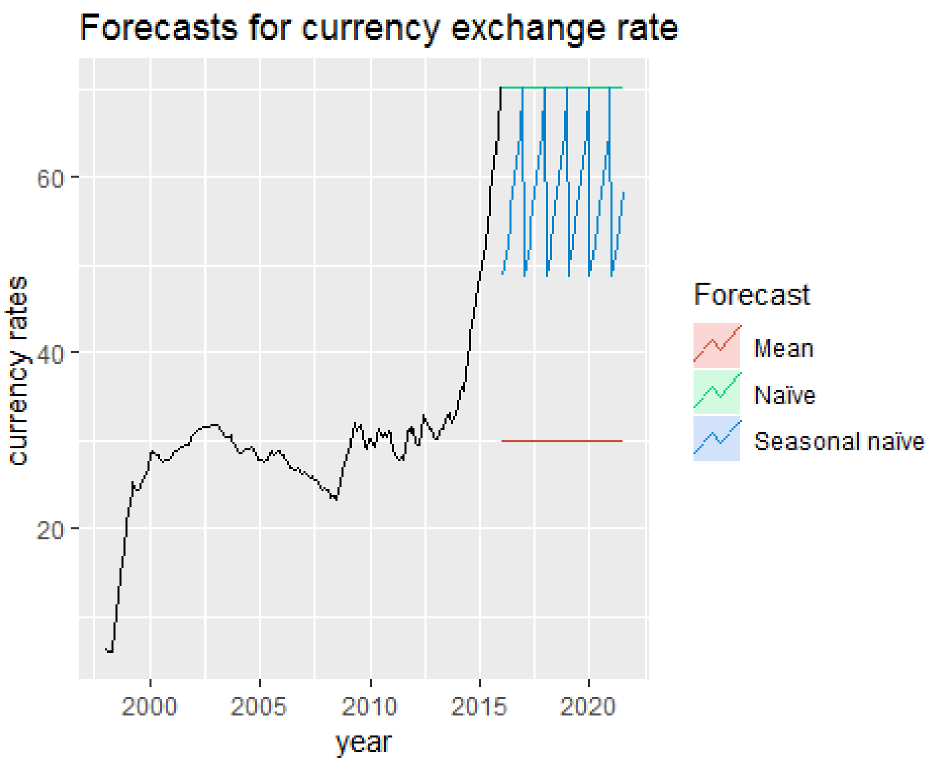


Naive forecasts of seasonally adjusted data

```
fit <- stl(count_ts, t.window=30, s.window="periodic", robust=TRUE)
fit %>% seasadj() %>% naive() %>%
# ploting the graph without seasonality
autoplot()
```

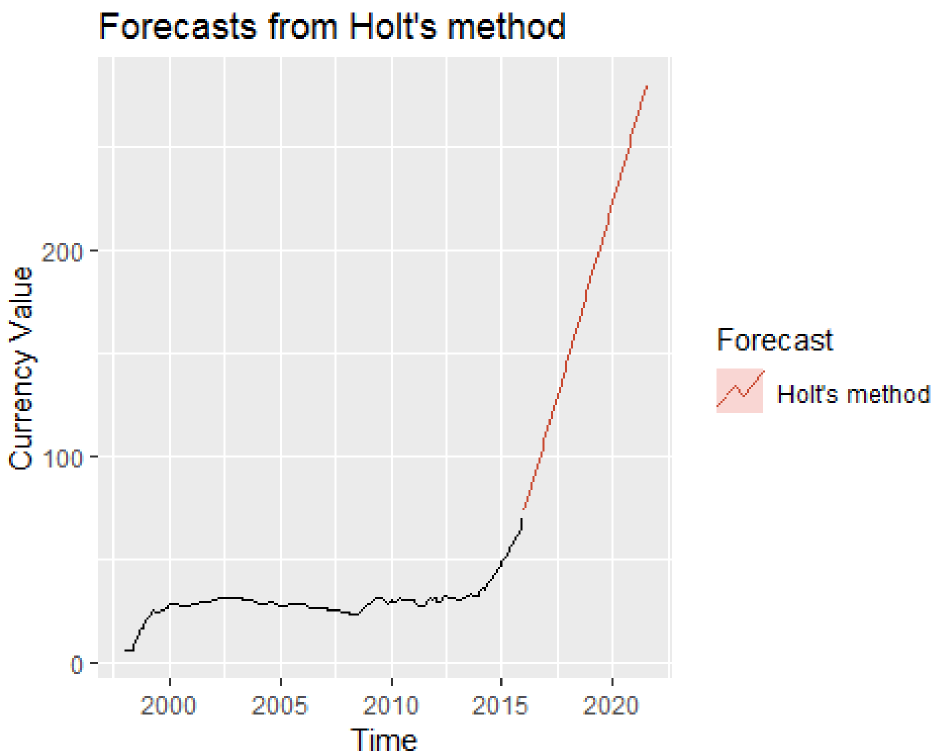



```
library(ggplot2)

#divide the data into training and testing
train <- deseasonal_ts[1:150]
test <- deseasonal_ts[150:216]


fit1 <- meanf(count_ts,h=67)
fit2 <- rwf(count_ts,h=67)
fit3 <- snaive(count_ts,h=67)
autoplot(window(count_ts, start=1998)) +
autolayer(fit1, series="Mean", PI=FALSE) +
autolayer(fit2, series="Naïve", PI=FALSE) +
autolayer(fit3, series="Seasonal naïve", PI=FALSE) +
ylab("currency rates") + xlab("year") +
ggtitle("Forecasts for currency exchange rate") +
guides(colour=guide_legend(title="Forecast"))
```

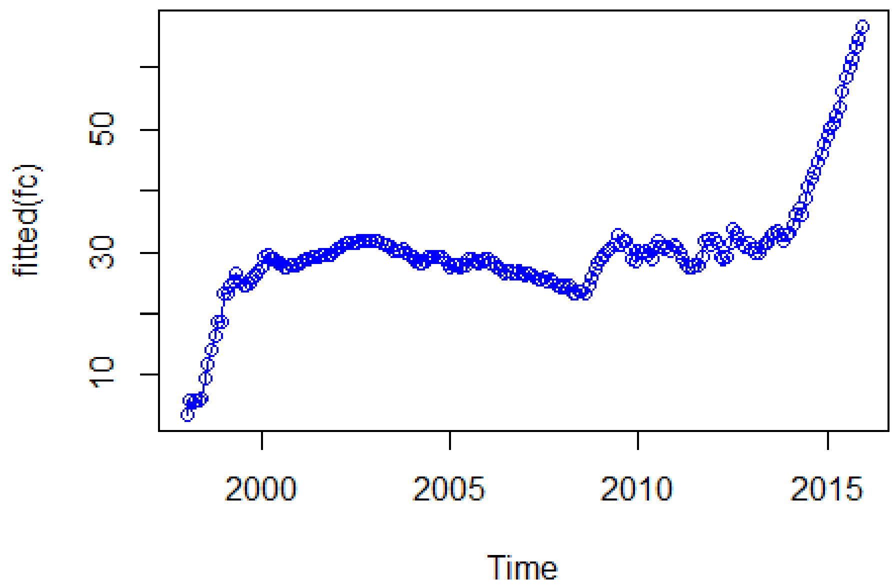



```
accuracy(fit1, test)

##                          ME      RMSE      MAE       MPE     MAPE      MASE
## Training set 1.842341e-16  8.896925 4.914027 -13.50684 23.74663  8.449778
## Test set     7.221067e+00 12.891380 7.566082  14.23475 15.44650 13.010045
##                   ACF1
## Training set 0.9314482
## Test set            NA

accuracy(fit2, test)

##                       ME       RMSE        MAE         MPE       MAPE
## Training set   0.2972487  0.8824197  0.5815569    1.044885   2.068504
## Test set     -32.9360934 34.6241264 32.9360934 -100.450309 100.450309
##                  MASE     ACF1
## Training set  1.00000 0.549123
## Test set     56.63434       NA

accuracy(fit3, test)

##                       ME      RMSE       MAE        MPE     MAPE      MASE
## Training set   2.747478  6.327891  3.846758   7.482198 11.52544  6.614585
## Test set     -19.879916 23.581694 21.746094 -63.377021 66.38235 37.392890
##                   ACF1
## Training set 0.9452067
## Test set            NA
```

```
## Fit a model by using Holt method and find the fitted values (estimation of original values)
fc <- holt(count_ts, h=67)
autoplot(count_ts) +
autolayer(fc, series="Holt's method", PI=FALSE) +
ggtitle("Forecasts from Holt's method") + xlab("Time") +
ylab("Currency Value") + guides(colour=guide_legend(title="Forecast"))
```




```
plot(fitted(fc))
lines(fitted(fc), col = "blue", type = "o")
```




```
fitted(fc)

##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 1998  3.536353  5.872186  5.642468  5.803458  5.936461  6.108069  9.317140
## 1999 23.104476 23.412559 24.638446 25.239072 26.694645 25.101015 24.614057
## 2000 27.409035 29.049309 29.449095 28.743394 28.790663 28.309539 28.193766
## 2001 28.120343 28.592884 28.840408 28.853197 29.037568 29.203446 29.276477
## 2002 30.416573 30.889087 31.153413 31.374475 31.403986 31.422300 31.594141
## 2003 31.919892 31.866792 31.606874 31.312828 31.025659 30.684093 30.151131
## 2004 29.127857 28.404019 28.123282 28.311000 28.676828 29.066595 29.090509
## 2005 27.500684 27.719589 27.835857 27.423635 27.759576 27.972438 28.770183
## 2006 28.893507 28.236420 28.072963 27.647179 27.281334 26.689852 26.773280
## 2007 26.089994 26.483118 26.223012 25.964247 25.627962 25.714869 25.881057
## 2008 24.410545 24.353889 24.437525 23.363056 23.231185 23.629935 23.557423
## 2009 29.012954 29.597989 30.045806 30.444570 31.157613 32.869919 31.289627
## 2010 30.139576 29.801105 30.281237 29.397204 28.935845 30.835127 31.787468
## 2011 30.862074 29.891147 28.767051 27.800121 27.547433 27.550702 27.792371
## 2012 32.001767 31.529423 29.306106 28.897218 29.245568 31.400276 33.924247
## 2013 30.443889 29.877326 29.958127 30.912345 31.596443 31.509328 32.845778
## 2014 33.041164 34.294201 36.221031 36.992035 36.019515 38.579598 40.726595
## 2015 48.816353 50.112463 50.945946 52.256264 53.620012 56.268083 58.581657
##            Aug       Sep       Oct       Nov       Dec
## 1998 11.657845 14.014312 16.245620 18.615669 18.665528
## 1999 24.418226 24.943943 25.911023 26.124907 26.765129
## 2000 27.657337 27.576442 27.739531 27.865089 27.814998
## 2001 29.362780 29.479581 29.552139 29.672078 30.007151
## 2002 31.674031 31.679368 31.738927 31.789256 31.927183
```

```
## 2003 30.121604 30.225824 30.616172 29.931444 29.558147
## 2004 29.139399 29.307376 29.265293 29.014037 28.344180
## 2005 28.926765 28.524878 28.355072 28.639316 28.905860
## 2006 26.743803 26.586082 26.658625 26.856633 26.480291
## 2007 25.367777 25.572241 25.145342 24.607440 24.127359
## 2008 23.183237 24.464455 25.912213 27.265719 28.171810
## 2009 31.815090 31.927662 30.515530 28.775423 28.360503
## 2010 30.721933 30.341692 30.940718 30.209883 31.198748
## 2011 27.788064 29.057827 31.711756 32.066781 31.132782
## 2012 32.983167 32.022761 31.253052 30.865959 31.348803
## 2013 33.243699 33.433079 32.662691 31.904958 32.881291
## 2014 41.908742 43.035599 44.714199 45.812306 47.527169
## 2015 60.010293 61.412163 63.385228 64.801326 66.744771
```

```
fc$model
```

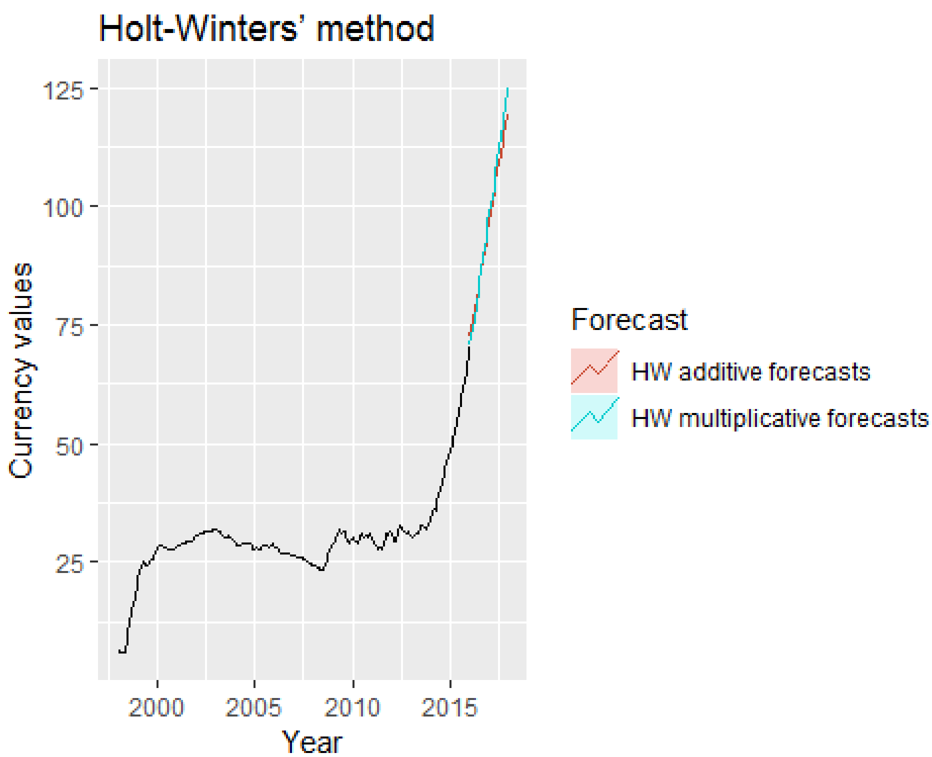
```
## Holt's method
##
## Call:
##  holt(y = count_ts, h = 67)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.4063
##
##   Initial states:
##     l = 5.0579
##     b = -1.5215
##
##   sigma:  0.6822
##
##      AIC     AICc      BIC
## 1001.783 1002.069 1018.659
```

```
accuracy(fc,test)
```

```
##                        ME        RMSE         MAE          MPE       MAPE
## Training set    0.05298339   0.6758088   0.4394268    0.4846092   1.834466
## Test set     -139.28644808 148.6694513 139.2864481 -371.6019886 371.601989
##                    MASE       ACF1
## Training set   0.755604 0.08076586
## Test set     239.506133         NA
```

```
## Fit a model by using Holt-Winters’ method with additive seasonality and
find the fitted values (estimation of original values)

fit1 <- hw(count_ts,seasonal="additive")
fit2 <- hw(count_ts,seasonal="multiplicative")
autoplot(count_ts) +
autolayer(fit1, series="HW additive forecasts", PI=FALSE) +
autolayer(fit2, series="HW multiplicative forecasts", PI=FALSE) +
```

```
xlab("Year") +
ylab("Currency values") +
ggtitle("Holt-Winters’ method") +
guides(colour=guide_legend(title="Forecast"))
```




```
fit1$model

## Holt-Winters' additive method
##
## Call:
##  hw(y = count_ts, seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.9367
##     beta  = 0.1324
##     gamma = 0.0627
##
##   Initial states:
##     l = 4.1638
##     b = 0.9608
##     s = -0.1025 -0.1144 -0.2646 -0.306 -0.2623 -0.019
##            0.0483 0.3358 0.0594 -0.0092 0.0261 0.6084
##
##   sigma:  0.7346
##
##      AIC     AICc      BIC
## 1045.170 1048.261 1102.549
```

```r
fit2$model

## Holt-Winters' multiplicative method
##
## Call:
##  hw(y = count_ts, seasonal = "multiplicative")
##
##   Smoothing parameters:
##     alpha = 0.4989
##     beta  = 0.2011
##     gamma = 1e-04
##
##   Initial states:
##     l = 4.4614
##     b = 0.3362
##     s = 1.011 0.999 0.9999 1.0075 1.0023 1.0021
##            1.0032 0.995 0.9877 0.9908 0.9975 1.0039
##
##   sigma:  0.049
##
##      AIC     AICc      BIC
## 1323.179 1326.270 1380.559

# since AIC of additive is less. It is good
```

Fit the ARIMA model.

ARIMA stands for auto-regressive integrated moving average and is specified by these three order parameters: (p, d, q).

AR(p), is referring to the use of past values in the regression equation for the series Y. The auto-regressive parameter p specifies the number of lags used in the model.

I(d) The d represents the degree of differencing

MA(q) component represents the error of the model as a combination of previous error terms. The order q determines the number of terms to include in the model

auto.arima() command choose the best ARIMA model

```r
#fitting the model
model1<-auto.arima(deseasonal_ts, seasonal = F)
model1

## Series: deseasonal_ts
## ARIMA(1,2,1)
##
## Coefficients:
##          ar1      ma1
##       0.3208  -0.8394
## s.e.  0.0975   0.0587
##
```

```
## sigma^2 estimated as 0.3857:  log likelihood=-201.07
## AIC=408.14   AICc=408.26   BIC=418.24
```

Evaluate the model for forecast.

we would expect no significant autocorrelations present in the model residuals.

```
forecast(model1,h = 20)

##          Point Forecast     Lo 80     Hi 80    Lo 95     Hi 95
## Jan 2016       72.94141  72.14546  73.73736 71.72410  74.15871
## Feb 2016       75.55951  74.13684  76.98219 73.38372  77.73530
## Mar 2016       78.00299  75.98585  80.02014 74.91804  81.08795
## Apr 2016       80.39045  77.79180  82.98910 76.41616  84.36474
## May 2016       82.75994  79.57984  85.94003 77.89641  87.62346
## Jun 2016       85.12365  81.35444  88.89287 79.35914  90.88817
## Jul 2016       87.48552  83.11503  91.85601 80.80143  94.16961
## Aug 2016       89.84679  84.86032  94.83327 82.22064  97.47295
## Sep 2016       92.20788  86.58930  97.82646 83.61500 100.80075
## Oct 2016       94.56890  88.30134 100.83646 84.98349 104.15431
## Nov 2016       96.92990  89.99610 103.86370 86.32557 107.53424
## Dec 2016       99.29090  91.67349 106.90831 87.64108 110.94072
## Jan 2017      101.65189  93.33352 109.97027 88.93004 114.37375
## Feb 2017      104.01289  94.97632 113.04946 90.19265 117.83313
## Mar 2017      106.37388  96.60205 116.14571 91.42916 121.31861
## Apr 2017      108.73488  98.21093 119.25882 92.63989 124.82986
## May 2017      111.09587  99.80318 122.38856 93.82520 128.36654
## Jun 2017      113.45686 101.37905 125.53468 94.98544 131.92829
## Jul 2017      115.81786 102.93878 128.69694 96.12100 135.51471
## Aug 2017      118.17885 104.48261 131.87510 97.23225 139.12545
```

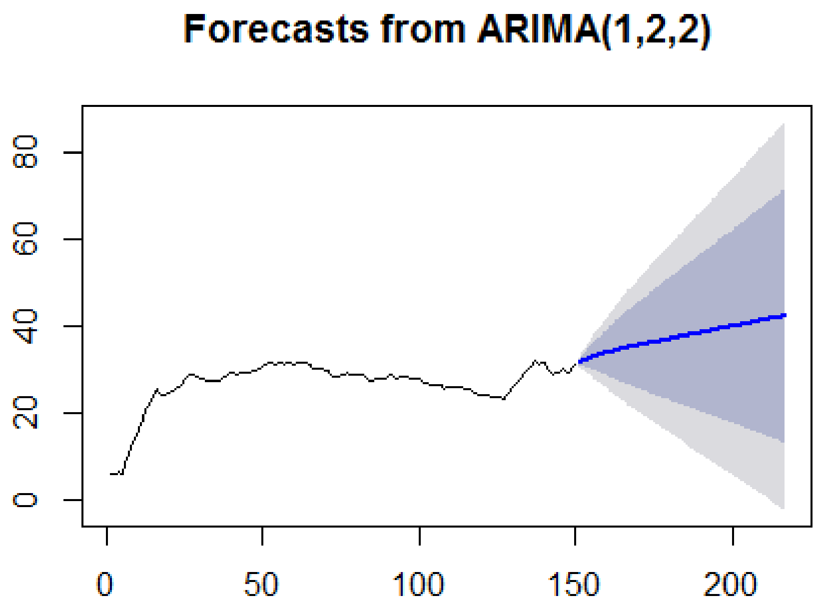
Lets divide the data and find the accuracy

Take the training set as all the values except the last 10 (test data) and find the accuracy.

```
#divide the data into training and testing
train <- deseasonal_ts[1:150]
test <- deseasonal_ts[150:216]

# find the best fit
fit <- auto.arima(train)
fit

## Series: train
## ARIMA(1,2,2)
##
## Coefficients:
##          ar1      ma1     ma2
##       0.8254  -1.4065  0.4175
## s.e.  0.1335   0.1599  0.1332
##
```

```
## sigma^2 estimated as 0.2879:  log likelihood=-117.1
## AIC=242.2   AICc=242.48   BIC=254.19
```

```r
# fit the model and predict the next 67 values
testest<- forecast(fit,h = 67)
plot(testest)
```




```r
# find the accuracy of fitted and actual
accuracy(testest,test)
```

```
##                       ME      RMSE       MAE         MPE      MAPE
## Training set -0.03513822 0.5275399 0.3566533  0.06092997  1.535543
## Test set     -0.38850643 8.5036860 6.7091207 -5.95947078 16.961706
##                   MASE        ACF1
## Training set  0.796456 -0.03842399
## Test set     14.982390          NA
```