TIME SERIES FORECASTING

**PROJECT OBJECTIVE**:

The aim is to predict the sales or forecast the demand of the consumable goods for the timeframe given as period October 2017 to December 2018. Here we use R programming to achieve this with the help of Time series forecasting.

The data is in the xlsx format and by the name “Demand” shows monthly demand of two different types of consumable items in a certain store from January 2002 to September 2017.

The following steps are to be applied in the time series forecasting:

* We convert the dataset into a time series.
* Combine the plots of both the item A and item B and study the trend, seasonality are draw insights from them
* Decompose the time series into seasonal, trend and random we must choose the type of model which is additive or multiplicative
* Decompose time series using STL Plot data for demand vs de seasoned demand
* Divide data into training and testing data
* Build random walk model
* Use ACF and PACF to test the autocorrelation and check the stationary data. To test the assumptions for stationary time series. In case of non-stationary, the data is to be converted to stationary.
* Forecast data on final hold out dataset

**DATA**

The data provided is the demand in numbers for the time 2002 to the year 2017. The data is for two items namely Item A and Item B. The data is given for 1 years and we need to use forecasting techniques to perform analysis on the data. The plots will explain the trends and the seasonality.

**ASSUMPTIONS**

The assumptions are made accordingly for hypothesis testing here it is apparent that the test that the data is non-root and non-stationary. We want the alternative hypothesis to be true. Since we need to use stationary data.

**EXPLORATORY ANALYSIS STEP BY STEP APPROACH**

Packages in R

The necessary packages to be installed are “xlsx”. Use the command install.packages(“xlsx”). Since we need to read values of the excel file which are in xls format we need this package which is not present by default.

Environment setup and data import

To set up a working directory

A working directory is set up to make the import and export of files easier. The data relevant to the project are stored in the location which makes data handling simpler if all the data is stored in the same location retrieving them is not a difficult task. We use the following command

getwd(“file location”)

setwd to set the file location that’s chosen.

Import and read the dataset

The R-studio is used as a tool to explore the data and the cardio dataset is in .xls format which is extension or spreadsheet file format. The command used to import data is new<-read.xls(“filename”) or we can use a command to choose the dataset file directly from the directory using new<-read.xls(file.choose()).

**Variable identification**

The variables are the parameters given in the function on which the analysis is carried out. The parameters are called features.

The R function used for variable identification are as given below:

Dim- this function is used to find out the dimensions i.e. number of rows and columns in the dataset.

Head-this function is used to get the head values which are the starting values by default only 10 are displayed but it can be increased as per specification.

Tail- this function is used to read and focus only on the ending values or last few values by default only 10 are displayed but it can be increased as per specification.

Head and tail are used to check if the data provided in the particular dataset behaves similarly throughout the table or is random.

View-it gives the tabulated from of the data as shown in the dataset in the output window.

Names-this function gives only the names, or the parameters specified in the dataset arranged as a row.

Summary-it is a generic function used to give summaries of the objects in the dataset to be analyzed with different parameters included such as minimum, maximum, median and quantiles depending on the class of the variable individually for all the features. Result of model fitting functions.

Str-the function str displays the structure of an arbitrary R object. It is a diagnostic function and an alternate to the summary function which displays the class of the object and its name.

summary(Demand[,3:4])

Item A Item B

Min. :1954 Min. :1153

1st Qu.:2748 1st Qu.:2362

Median :3134 Median :2876

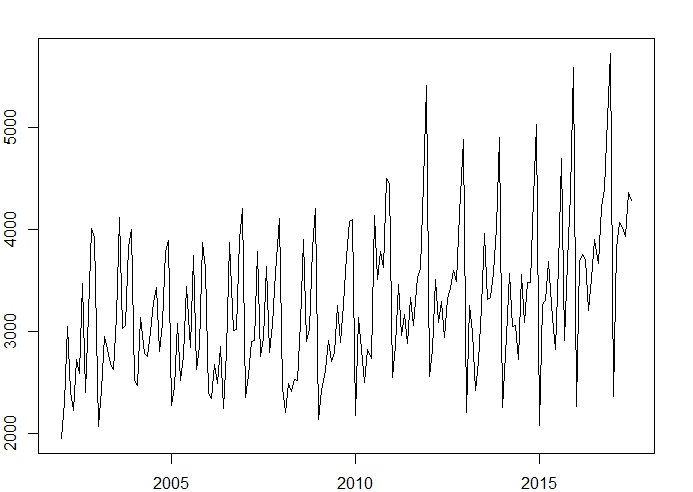
Mean :3263 Mean :2962

3rd Qu.:3741 3rd Qu.:3468

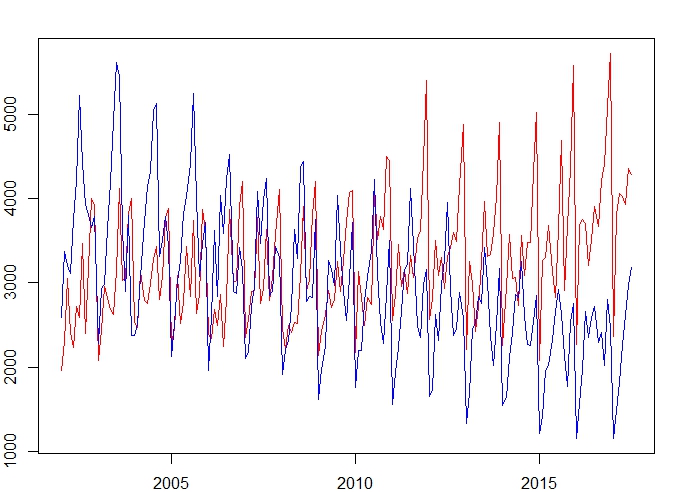
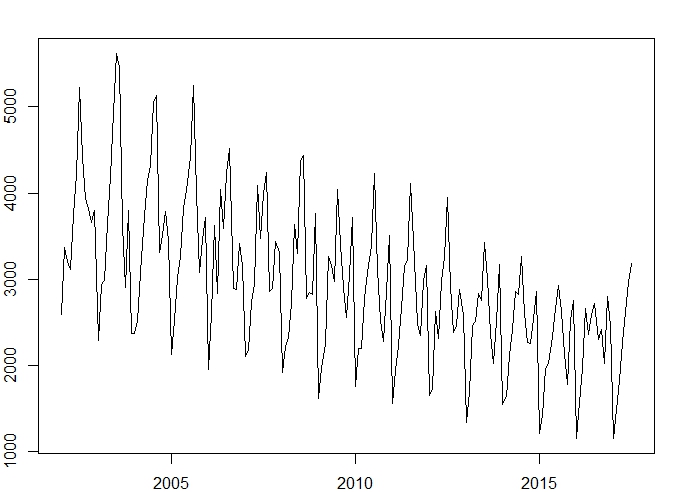
Max. :5725 Max. :5618

Here we observe that the minimum value, average and maximum value of Item A is greater than Item B. This shows that by the dataset the demand of Item A is of greater value than Item B. Item A is in more demand than Item B over the span of 15 years.

If we convert Item A into a time series, we get the following plot. It looks like there is an upwards trend and seasonality can also be seen throughout the data.



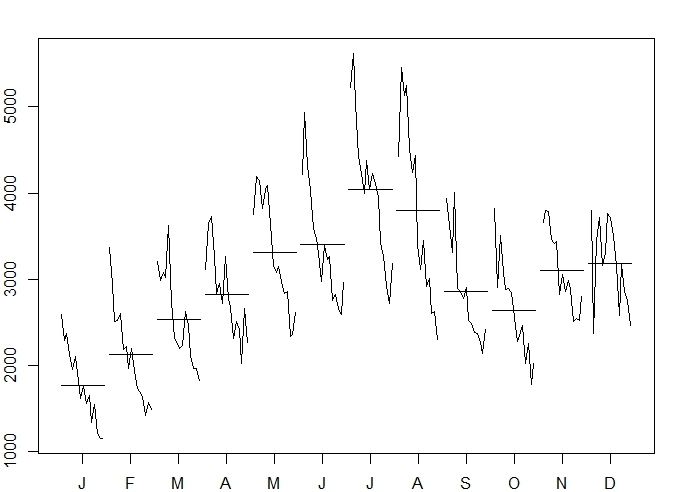
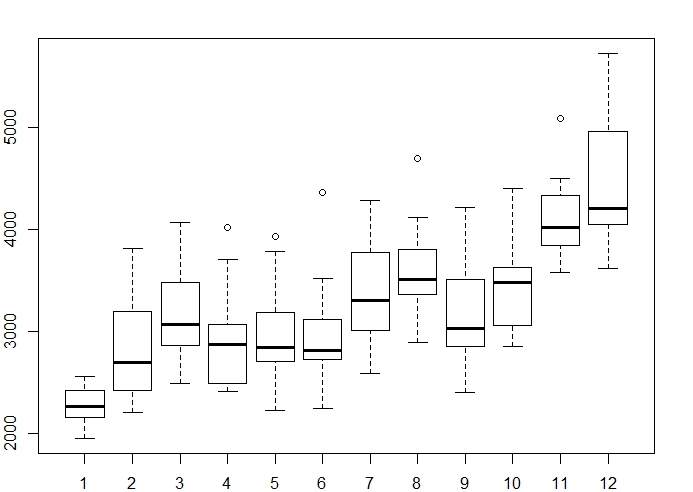
If we convert Item B into a time series, we get the following plot. It looks like there is an downwards trend and seasonality can also be seen throughout the data.



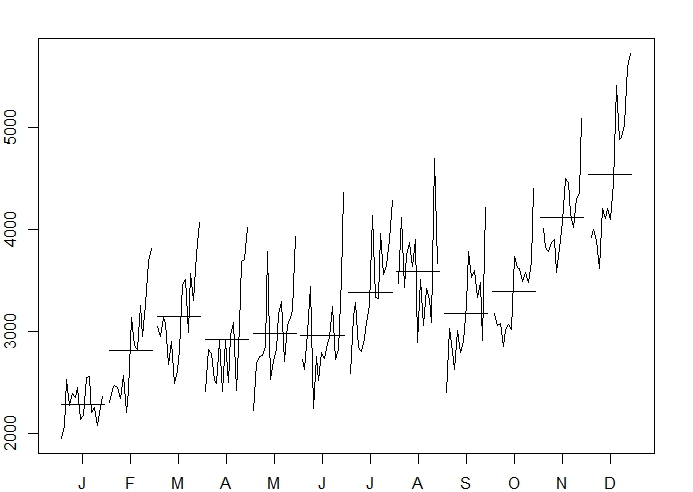
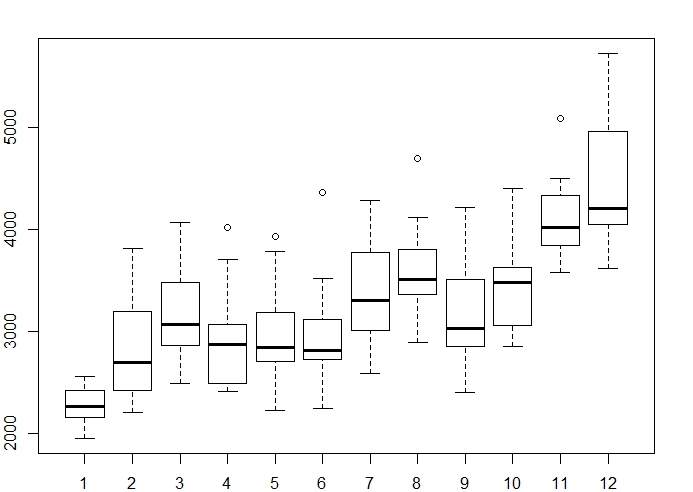
If we compare both the Item A and Item B time series, we get the result of such a combination. Here Item A demand seems to be increasing whereas the demand of Item B seems to be decreasing. They are not cyclic.

**DECOMPOSISTION**

On decomposing the time series, we get different series namely the trend, seasonality and random. Here, we get a timeseries which is constant over time period hence we use an additive model



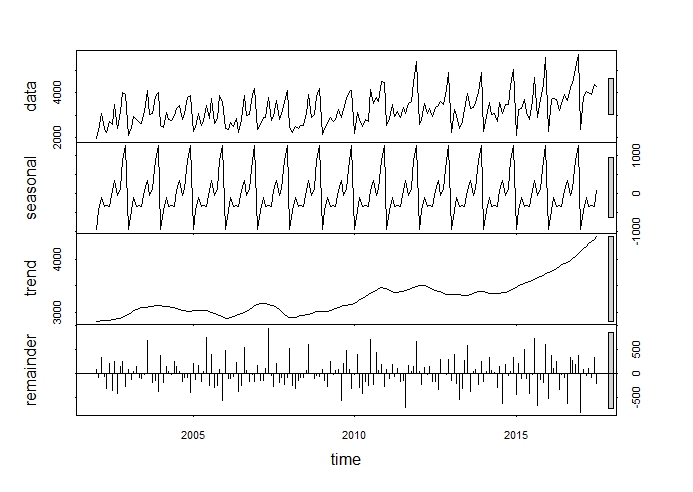
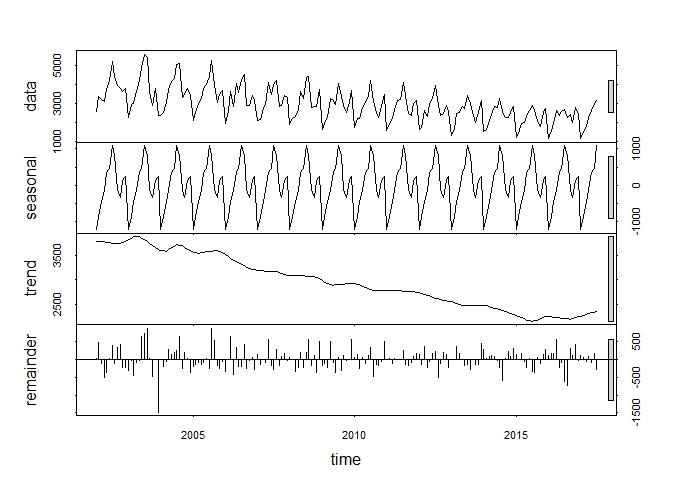
Here we have the plots for Item A and monthly plots give the monthly variations of the data for each for us to understand the data better and boxplots give the outliers and other representation of values.

\

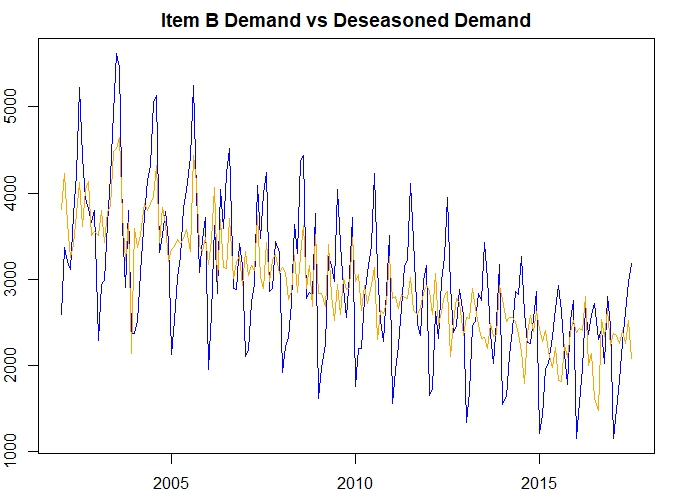
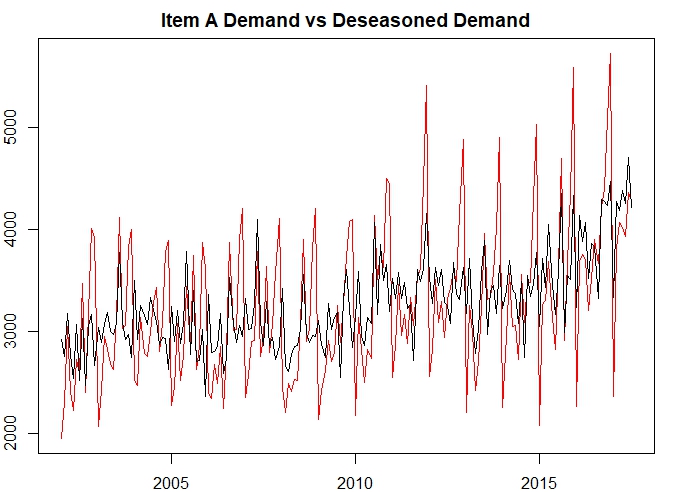
Here we have the plots for Item B and monthly plots give the monthly variations of the data for each for us to understand the data better and boxplots give the outliers and other representation of values. From, these plots we decide that the model is additive, and we see that the variations are constant over time.

Decomposition using stl: STL- Seasonal and trend decomposition using Loess

The plots are decomposed into four different categories when isolated they are split into: Seasonal. Trend, remainder and data. Data represents the actual data. The representation on the same scale makes it easier to dive deep.

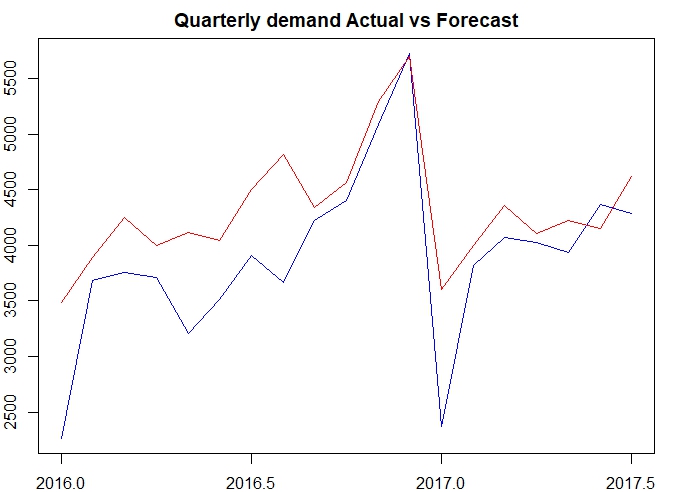
This method is used for both the item A and item B. The observations remain the same as mentioned above the trends are clearly visible. 

The plots are for item A an item B we combine time series of seasoned and de-seasoned demand. The increasing and decreasing trend of demand is clearly evident. The remaining residual values are also included.



The dataset has been divided into training and testing dataset. There are two sets of training and testing datasets for Item A and Item B.

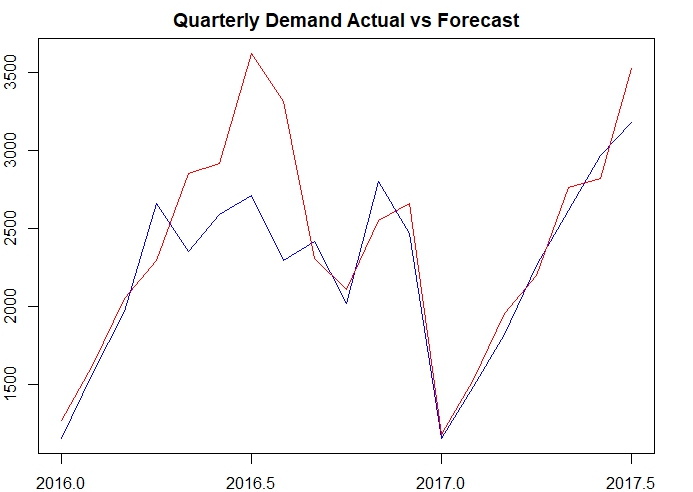
After applying the forecasting techniques, we plot the actual values with the forecast values. It shows that the forecast values are higher than the actual values.



There are two types of tests that are conducted which are MAPE and Box-Ljung test to be conducted on both the Items.

MAPEA=0.1408798

In, the box test, we reject Ha the alternate hypothesis: residuals are not independent.



After applying the forecasting techniques, we plot the actual values with the forecast values. It shows that the forecast values are higher than the actual values. Its shows that at some portions the forecasted data is little higher or lower compared to the actual data.

There is always some difference in the actual and forecasted data it is very difficult to precisely forecast the future values, but we can approximately get a estimation.

MAPEB=0.1082608

In, the box test we reject Ha the alternate hypothesis: residuals are not independent.

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

Holt winter method is for three smoothing equations with trend, seasonality and level which are given by gamma, beta and alpha respectively.

## Call:

## HoltWinters(x = as.ts(DataATrain), seasonal = "additive")

## Smoothing parameters:

## alpha: 0.1241357

## beta : 0.03174654

## gamma: 0.3636975

Coefficients:

[,1]

## a 3753.348040

## b 7.663395

## s1 -1250.098605

## s2 -438.592232

## s3 -224.017731

## s4 -407.395313

## s5 -507.668223

## s6 -667.267246

## s7 63.659702

## s8 197.909330

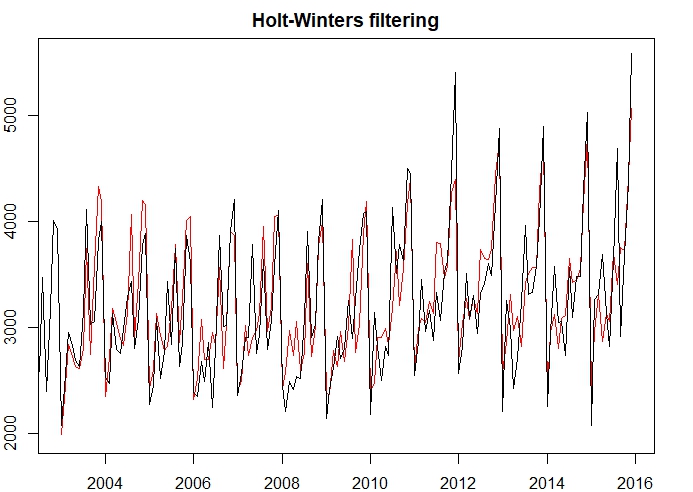
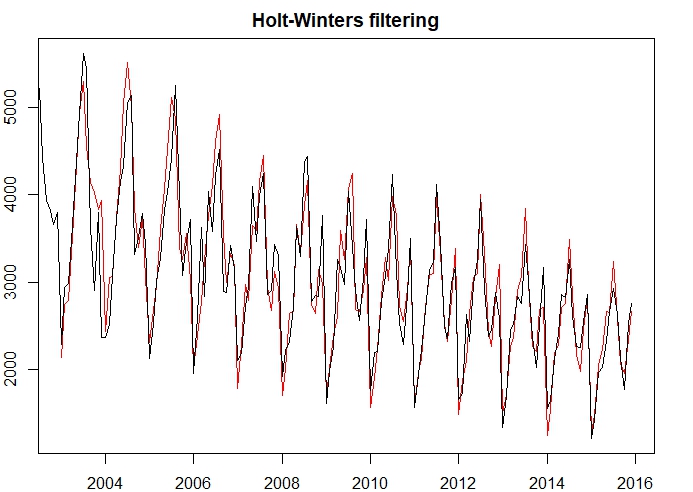
## s9 -301.525945

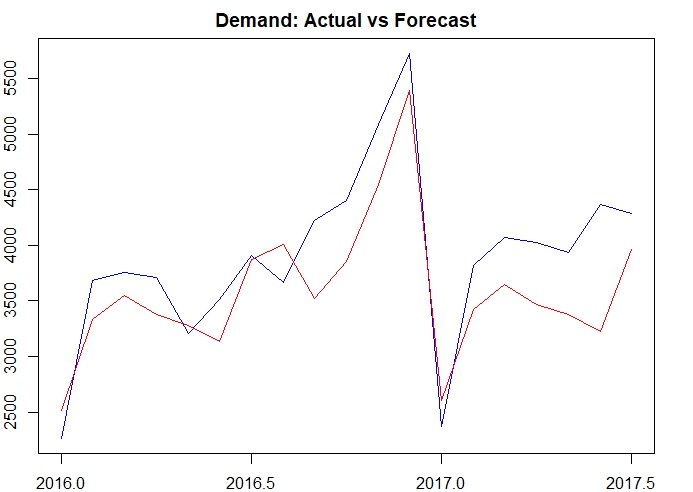
## s10 25.272325

## s11 712.529546

## s12 1545.291998

For the particular values of smoothing parameters: we can value of s form s1 to s12.





**ARIMA Model is used to check for stationary time series**

Dickey-Fuller test

Statistical tests make strong assumptions about your data. They can only be used to inform the degree to which a null hypothesis can be accepted or rejected.

The result must be interpreted for a given problem to be meaningful. Nevertheless, they can provide a quick check and confirmatory evidence that your time series is stationary or non-stationary.

Null Hypothesis (Ho): Stationary and time series does not have a unit root.

Alternate Hypothesis (Ha): Time dependent. Non-stationary and time series has a unit root.

p-value > 0.05: Accept 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.

**ACF and PACF**

ACF and PACF Autocorrelation function and partial autocorrelation function is used to test if data is stationary and if it’s is not it should be converted from non-stationary to stationary.

The function ACF is used to compute an estimate of the autocorrelation function of a time series. The time series can be multi variate. Function PACF computes an estimate of the partial autocorrelation function of a time series. The time series can be multi variate.

**ACF**

* Autocorrelation of different orders gives inside information regarding time series. It will help determine order p of the series
* Significant autocorrelations imply observations of long past influences current observation
* Maximum lag at which to calculate the ACF: Default is 10∗log10(N/m)
* where N is the number of observations and m the number of series.
* Will be automatically limited to one less than the number of observations in the series

**PACF**

Partial autocorrelation adjusts itself for the intervening periods.

If we apply ACF and PACF applied to the item A and item B. We get plots for which we see actual vs forecast. The box-Ljung test is conducted to see if the residuals are independent or not.

We use that to fit train data and residuals in one of the actual vs forecast data we can see that actual data follows the forecast data and on the hold out set we see that the actual data is different from the forecast data.

Forecasted values are as given below for the timeframe given.

**FCASTA**

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95

## Aug 2017 4320.211 3879.991 4760.431 3646.953 4993.469

## Sep 2017 4169.513 3725.551 4613.476 3490.531 4848.495

## Oct 2017 4428.791 3981.385 4876.197 3744.542 5113.040

## Nov 2017 5102.669 4652.091 5553.246 4413.570 5791.767

## Dec 2017 5879.220 5425.721 6332.719 5185.653 6572.787

## Jan 2018 2819.535 2363.343 3275.727 2121.849 3517.221

## Feb 2018 3990.984 3532.307 4449.660 3289.498 4692.469

## Mar 2018 4181.449 3720.480 4642.419 3476.458 4886.441

## Apr 2018 4081.089 3618.003 4544.174 3372.860 4789.317

## May 2018 3888.336 3423.296 4353.376 3177.118 4599.554

## Jun 2018 4029.525 3562.679 4496.370 3315.545 4743.504

## Jul 2018 4390.292 3921.777 4858.807 3673.760 5106.823

## Aug 2018 4407.590 3900.778 4914.402 3632.487 5182.693

## Sep 2018 4257.019 3747.019 4767.019 3477.041 5036.997

## Oct 2018 4516.419 4003.480 5029.358 3731.946 5300.892

## Nov 2018 5190.414 4674.763 5706.065 4401.794 5979.034

## Dec 2018 5967.079 5448.925 6485.232 5174.632 6759.526

## Jan 2019 2907.502 2387.039 3427.966 2111.522 3703.483

## Feb 2019 4079.056 3556.458 4601.654 3279.812 4878.301

**FACSTB**

## Point Forecast Lo 80 Hi 80 Lo 95 Hi 95

## Oct 2017 1784.7949 1373.5500 2196.040 1155.8501 2413.740

## Nov 2017 2436.4019 2025.1570 2847.647 1807.4571 3065.347

## Dec 2017 2429.8611 2018.6162 2841.106 1800.9163 3058.806

## Jan 2018 965.2270 553.9834 1376.471 336.2842 1594.170

## Feb 2018 1278.2300 866.9865 1689.474 649.2873 1907.173

## Mar 2018 1693.3730 1282.1294 2104.617 1064.4302 2322.316

## Apr 2018 2088.9240 1677.6805 2500.168 1459.9813 2717.867

## May 2018 2342.6726 1931.4290 2753.916 1713.7298 2971.615

## Jun 2018 2587.5703 2176.3268 2998.814 1958.6276 3216.513

## Jul 2018 2903.9519 2492.7084 3315.196 2275.0092 3532.895

## Aug 2018 2267.6121 1814.7093 2720.515 1574.9570 2960.267

## Sep 2018 1945.3845 1492.4816 2398.287 1252.7293 2638.040

## Oct 2018 1663.8271 1210.9242 2116.730 971.1719 2356.482

## Nov 2018 2293.2590 1840.3561 2746.162 1600.6038 2985.914

## Dec 2018 2325.0672 1872.1643 2777.970 1632.4120 3017.722

## Jan 2019 843.6106 390.7094 1296.512 150.9580 1536.263

## Feb 2019 1150.9134 698.0123 1603.815 458.2608 1843.566

**Conclusion**

Here in the given dataset we have noticed it has both trend and seasonality.

There seems to be an upward trend for item A and downward trend for item B. For both items there are few months with high variation in seasonality and for Item A there are few outliers.

Since the seasonality was not following the trend pattern we have opted for additive seasonality. We have performed the three models:

**1. Random Walk with Drift 2. Holt Winters 3. ARIMA model.**

MAPE and box-Ljung tests for each of the values:

1. **Random Walk with Drift**

Item A# 0.1408798 (14%), p-value < 2.2e-16

Item B# 0.1082608 (10.8%), p-value = 2.931e-13

1. **Holt-Winters**

Item A# 0.1160528 (11.6%), p-value = 0.8188

Item B# 0.1867152 (18.6%), p-value = 0.873

1. **ARIMA**

Item A# 0.0733376 (7%), p-value = 0.0809

Item B# 0.07654621 (7%), p-value = 0.09177

From the MAPE values observed the ARIMA model provided the lowest values and we selected the model for the Forecasting.

**SOURCE CODE**

library(readxl)

Demand <- read\_excel("C:/Users/Saurabh/Downloads/Demand.xlsx",skip=1)

View(Demand)

str(Demand)

summary(Demand[,3:4])

dm<-ts(Demand,start=c(2002,1),end=c(2017,7),f=12)

plot(dm)

ita<-ts(Demand[,3],start=c(2002,1),end=c(2017,7),f=12)

plot(ita)

itb<-ts(Demand[,4],start=c(2002,1),end=c(2017,7),f=12)

plot(itb)

ts.plot(ita,itb, gpars=list(col=c("red","blue")),xlab="Year",ylab="Demand")

legend("topleft",colnames(Demand[,3:4]),col=1:ncol(Demand),lty=1.9,cex=.45)

monthplot(ita)

boxplot(ita)

boxplot(ita~cycle(ita))

monthplot(itb)

boxplot(itb)

boxplot(ita~cycle(itb))

itas<-stl(ita[,1], s.window="p")

plot(itas)

itbs<-stl(ita[,1], s.window="p")

plot(itbs)

series <- c('Deseasoned', 'Actual')

Deseason\_Ita <- (itas$time.series[,2]+itas$time.series[,3])

ts.plot(ita, Deseason\_Ita, col=c("red", "black"), main="Item A Demand vs Deseasoned Demand")

series<-c('Deseasoned','Actual')

Deseason\_Itb<-(itbs$time.series[,2]+itbs$time.series[,3])

ts.plot(itb, Deseason\_Itb, col=c("blue", "orange"), main="Item B Demand vs Deseasoned Demand")

TrainA <- window(ita, start=c(2002,1), end=c(2015,12), frequency=12)

TestA <- window(ita, start=c(2016,1), frequency=12)

TrainB <- window(itb, start=c(2002,1), end=c(2015,12), frequency=12)

TestB <- window(itb, start=c(2016,1), frequency=12)

ItemATrn <- stl(TrainA[,1], s.window="p")

ItemBTrn <- stl(TrainB[,1], s.window="p")

install.packages(forecast)

library(forecast)

fcst.ItA.stl <- forecast(ItemATrn, method="rwdrift", h=19)

fcst.ItB.stl <- forecast(ItemBTrn, method="rwdrift", h=19)

VecA<- cbind(TestA,fcst.ItA.stl$mean)

VecB<- cbind(TestB,fcst.ItB.stl$mean)

ts.plot(VecA, col=c("blue", "red"),xlab="year", ylab="demand", main="Quarterly demand Actual vs Forecast")

MAPEA <- mean(abs(VecA[,1]-VecA[,2])/VecA[,1])

MAPEA

Box.test(fcst.ItA.stl$residuals, lag=10, type="Ljung-Box")

ts.plot(VecB, col=c("blue", "red"),xlab="year", ylab="demand", main="Quarterly Demand Actual vs Forecast")

MAPEB<-mean(abs(VecB[,1]-VecB[,2])/VecB[,1])

MAPEB

Box.test(fcst.ItB.stl$residuals, lag=10, type="Ljung-Box")

hwA <- HoltWinters(as.ts(TrainA),seasonal="additive")

hwA

plot(hwA)

hwB <- HoltWinters(as.ts(TrainB),seasonal="additive")

hwB

plot(hwB)

hwAForecast <- forecast(hwA, h=19)

VecA1 <- cbind(TestA,hwAForecast)

par(mfrow=c(1,1), mar=c(2, 2, 2, 2), mgp=c(3, 1, 0), las=0)

ts.plot(VecA1[,1],VecA1[,2], col=c("blue","red"),xlab="year", ylab="demand", main="Demand: Actual vs Forecast")

Box.test(hwAForecast$residuals, lag=20, type="Ljung-Box")

install.packages("MLmetrics")

library(MLmetrics)

MAPE(VecA1[,1],VecA1[,2])

hwB <- HoltWinters(as.ts(TrainB),seasonal="additive")

hwB

plot(hwB)

hwBForecast <- forecast(hwB, h=19)

VecB1 <- cbind(TestB,hwBForecast)

par(mfrow=c(1,1), mar=c(2, 2, 2, 2), mgp=c(3, 1, 0), las=0)

ts.plot(VecB1[,1],VecB1[,2], col=c("blue","red"),xlab="year", ylab="demand", main="Demand: Actual v

Box.test(hwBForecast$residuals, lag=20, type="Ljung-Box")

MAPE(VecB1[,1],VecB1[,2])

library(tseries)

adf.test(ita)

diff\_dem\_ItA <- diff(ita)

plot(diff\_dem\_ItA)

adf.test(diff(ita))

library(tseries)

adf.test(itb)

diff\_dem\_ItB<- diff(itb)

plot(diff\_dem\_ItB)

adf.test(diff(itb))

acf(ita,lag=15

acf(diff\_dem\_ItA, lag=15)

acf(itb,lag=15)

acf(diff\_dem\_ItB, lag=15)

acf(ita,lag=50)

acf(diff\_dem\_ItA, lag=50)

pacf(ita)

pacf(diff\_dem\_ItA)

acf(itb,lag=50)

acf(diff\_dem\_ItB, lag=50)

pacf(itb)

pacf(diff\_dem\_ItB)

ItA.arima.fit.train <- auto.arima(TrainA, seasonal=TRUE)

ItA.arima.fit.trainplot(ItA.arima.fit.train$residuals)

plot(ItA.arima.fit.train$residuals)

plot(ItA.arima.fit.train$x,col="blue")

lines(ItA.arima.fit.train$fitted,col="red",main="Demand A: Actual vs Forecast")

MAPE(ItA.arima.fit.train$fitted,ItA.arima.fit.train$x)

acf(ItA.arima.fit.train$residuals)

pacf(ItA.arima.fit.train$residuals)

Box.test(ItA.arima.fit.train$residuals, lag = 10, type = c("Ljung-Box"), fitdf = 0)

ArimafcastA <- forecast(ItA.arima.fit.train, h=19)

VecA2 <- cbind(TestA,ArimafcastA)

par(mfrow=c(1,1), mar=c(2, 2, 2, 2), mgp=c(3, 1, 0), las=0)

ts.plot(VecA2[,1],VecA2[,2], col=c("blue","red"),xlab="year", ylab="demand", main="Demand A: Actual vs Forecast")

ItB.arima.fit.train <- auto.arima(itb, seasonal=TRUE)

ItB.arima.fit.train

plot(ItB.arima.fit.train$residuals)

plot(ItB.arima.fit.train$x,col="blue")

lines(ItB.arima.fit.train$fitted,col="red", main="Demand B: Actual vs Forecast")

MAPE(ItB.arima.fit.train$fitted,ItB.arima.fit.train$x)

acf(ItB.arima.fit.train$residuals)

pacf(ItB.arima.fit.train$residuals)

Box.test(ItB.arima.fit.train$residuals, lag = 10, type = c("Ljung-Box"), fitdf = 0)

ArimafcastB <- forecast(ItB.arima.fit.train, h=19)

VecB2 <- cbind(TestB,ArimafcastB)

par(mfrow=c(1,1), mar=c(2, 2, 2, 2), mgp=c(3, 1, 0), las=0)

ts.plot(VecB2[,1],VecB2[,2], col=c("blue","red"),xlab="year", ylab="demand", main="Demand B: Actual vs Forecast")

ItA.arima.fit <- auto.arima(ita, seasonal=TRUE)

fcastA <- forecast(ItA.arima.fit, h=19)

plot(fcastA)

fcastA

ItB.arima.fit <- auto.arima(itb, seasonal=TRUE)

fcastB <- forecast(ItB.arima.fit, h=19)

plot(fcastB)

fcastB