# Time Series Forecasting Using R: A Starter Pack

## Some basic theoretical ideas needed before we proceed: -

Time Series Data –

A time series is a set of observations on the values that a variable takes at different times. Such data may be collected at regular time intervals, such as daily (e.g., stock prices, weather reports), weekly (e.g., money supply), monthly [e.g., the unemployment rate, the Consumer Price Index (CPI)], quarterly (e.g., GDP), annually (e.g., government budgets), quinquennially, that is, every 5 years (e.g., the census of manufactures), or decennially (e.g., the census of population).

Stochastic Process –

A random or stochastic process is a collection of random variables ordered in time.

Stationary Stochastic Processes –

A stochastic process is said to be stationary if its mean and variance are constant over time and the value of the covariance between two time periods depends only on the distance or gap or lag between the two time periods and not the actual time point at which the covariance is computed.

Nonstationary Stochastic Processes –

A stochastic process is said to be nonstationary if at least one of the characteristics of it from among its mean or variance or the value of the covariance between two time-periods depends on the actual time point at which it is being computed.

Why are stationary time series so important? –

This is because if a time series is nonstationary, we can study its behavior only for the time period under consideration. Each set of time series data will therefore be for a particular episode. As a consequence, it is not possible to generalize it to other time periods. Therefore, for the purpose of forecasting, such (nonstationary) time series may be of little practical value.

Autocorrelation Function (ACF) –

One simple test of stationarity is based on the so-called Autocorrelation function (ACF).  
The ACF at lag k, is given as = (Covaraince at lag k/Variance)

Augmented Dickey-Fuller Test (ADF Test) –

Augmented Dickey-Fuller test (ADF) tests the null hypothesis that a time series sample suffers from Non-Stationarity vs. the alternative hypothesis of Stationarity (Trend-Stationarity).

Exponential Smoothing Methods –

These are essentially methods of fitting a suitable curve to historical data of a given time series. There are a variety of these methods, such as single exponential smoothing, Holt's linear method, and Holt-Winters' method and their variations.

Auto Regressive Model (AR Model) –

The autoregressive model specifies that a time series variable depends linearly on its own previous period values and on a stochastic term. Thus, the model is in the form of a stochastic difference equation. The autoregressive model is not always stationary.

Moving Average Model (MA Model) –

The Moving Average Model is a common approach for modeling univariate time series. The moving-average model specifies that the output variable depends linearly on the current and various past values of a stochastic term. Contrary to the AR model, the finite MA model is always stationary.

Auto Regressive Moving Average Model (ARMA Model) –

It is a combination or AR and MA processes.

Auto Regressive Integrated Moving Average Model (ARIMA Model) –

Auto Regressive Integrated Moving Average (ARIMA) Model is a generalization of an ARMA Model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity.

## In the next portion, we discuss the hands-on techniques required to perform a Time Series Analysis in R.

At first, we need to call up the library named "forecast" required later to perform predictions.

library(forecast)

The next step is setting the working directory of the data set under consideration.

The required path for my system is:-

## "C:\Users\arkad\Documents\STEP UP ANALYTICS\Blog Project\Time Series Using R"

So, we use the function "setwd( )" along with the above relevant path, to set the working directory.

setwd("C:\\Users\\arkad\\Documents\\STEP UP ANALYTICS\\Blog Project\\Time Series Using R")

The name of the required data set in my analysis is "Sales.csv", i.e. a CSV file.

If anyone needs to get access to this data set, get it from the link below.

[link for data set - Sales.csv](https://drive.google.com/file/d/0B7cB2Fb0wrTEdV9hTWV6NmxNYkU/view?usp=sharing)

So we create a space in R to read and store the entire data from the "Sales.csv" file.

Thus we require R to read the required CSV file using the command :- read.csv("sales.csv")

After reading the data, we want R to store it in a space named "data".

So, "data" is the name of the data set in R, henceforth.

data <- read.csv("Sales.csv")

Now we will be using the data set named "data" for our analysis.

By default "data" has been created by R as a Data Frame.

Just to verify we check for the class of "data" using the following command.

class(data)

## [1] "data.frame"

On execution of the above command it shows "data.frame" .

Next we need to look at the contents of the data frame named "data".

Since it is not necessary at all to go through all the rows of records in "data".

So we just want to get an idea by looking at the first few rows of "data".

So we use the function "head( )", which returns the 1st 6 rows of records from "data".

head(data)

## Month.Year Sales  
## 1 Jan-03 25400  
## 2 Feb-03 25440  
## 3 Mar-03 25370  
## 4 Apr-03 25400  
## 5 May-03 25490  
## 6 Jun-03 25240

On execution, we see that "data" contains 2 columns/variables.

The 1st one is "Month.Year" having values like "Jan-03", "Feb-03", etc.

So it is the time variable of our data frame.

The 2nd variable is "Sales", having records like "141", "157", etc.

So this is the variable/series varying over time which we need to analyse.

This series viz. "Sales" is a continuous variable, as evident from its data records.

Next, we plan to rename the column heads of "data" as per our own convenience.

So, we choose to rename the original time variable "Month.Year" as "Date".

And the variable "Sales" is to be renamed as it is.

We use the "names( )" function to rename both the columns of "data".

So we specify as an option while renaming, that we want to start with the 1st column head.

And then proceed on renaming till the 2nd column head.

So, we use "names(data)[c(1:2)]" .

names(data)[c(1:2)] <- c("Date", "Sales")

Again to verify whether the renaming is done successfully we check the content of "data" again.

head(data)

## Date Sales  
## 1 Jan-03 25400  
## 2 Feb-03 25440  
## 3 Mar-03 25370  
## 4 Apr-03 25400  
## 5 May-03 25490  
## 6 Jun-03 25240

On execution, we see the renaming has been successfully done.

Now our next task is to instruct R to convert this data frame into a time series structure.

So, we use the "ts( )" function with its relevant arguments on the data frame "data".

The conversion of the original data frame "data" into Time Series mode is done as follows.

data <- ts(data[,2], start = c(2003,1), frequency = 12)

We see in the above process, while using the function "ts( )", we specify the 2nd column of "data", i.e. "Sales" as the series under consideration and also indicate to R, that the starting point of the series should be "(2003,1)", i.e. the 1st month of the year 2003.

Also the frequency was set to be equal to 12, indicating that all the 12 months of each year are to be considered.

Now just for the sake of knowing the class of "data" after this conversion we do the following.

class(data)

## [1] "ts"

We see that the class has changed to "ts", indicating a Time Series structure.

To understand the differences newly created in the Time Series structure "data", after it has been tabulated in Time Series mode, we return the entire data frame.

data

## Jan Feb Mar Apr May Jun Jul Aug Sep Oct  
## 2003 25400 25440 25370 25400 25490 25240 25240 25280 25420 25280  
## 2004 25280 25240 25340 25360 25390 25270 25350 25420 25530 25790  
## 2005 26530 26640 26930 26750 26740 26700 26810 26860 27230 27400  
## 2006 28070 28440 28650 28930 29350 29490 29660 29700 30140 30500  
## 2007 31330 31640 32110 32450 34020 35670 37140 37460 39940 41360  
## 2008 45830 46320 47310 47820 48060 48770 49360 49800 50900 52120  
## 2009 54080 55700 56560 57840 59940 62070 64010 66270 69350 70740  
## 2010 77000 78750 79280 79200 79870 79960 80320 80400 80450 80730  
## 2011 82820 83370 83170 82960 82770 82850 82290 82800 81540 80020  
## 2012 80880 82160 83090 83490 83980 85370 86570 86970 88920 90410  
## 2013 91780 91590 93270 96050 94130 93190 93030 93170 92930 93890  
## 2014 94820 95790 95330 95300 95650 96210 96910 97310 98940 100140  
## 2015 101240 102470 102600 102810 103000 101960 101960 102140 100220 100140  
## 2016 98270 99790 101690 102550 103910 105830 107170 108640 110430 109200  
## Nov Dec  
## 2003 25410 25370  
## 2004 25980 26130  
## 2005 27510 27650  
## 2006 30740 30760  
## 2007 44300 45840  
## 2008 52190 52800  
## 2009 73080 74770  
## 2010 81500 81900  
## 2011 79650 79940  
## 2012 90090 90440  
## 2013 94370 94450  
## 2014 100350 100570  
## 2015 99630 98910  
## 2016 106900 103690

On execution, we see that the earlier representation of "data" has now been changed.

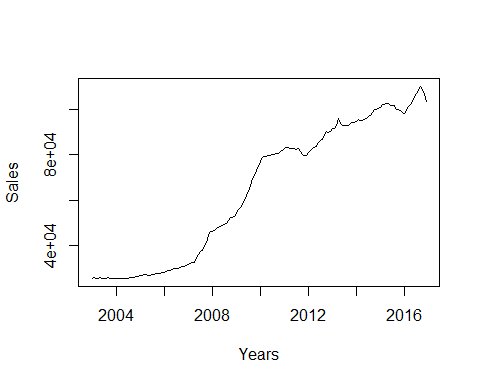
Now, it has a more structured and bi-variately tabulated look.

Next, we go for an initial visual analysis of the series "Sales" from "data".

Actually, this will help us to get a crude idea about the nature of the series "Sales".

So, we plot the "Sales" data points against the corresponding values of the time variable.

plot(data, xlab = "Years", ylab = "Sales")



Here we re-label the X-axis as "Years" and the Y-axis as "Sales".

We see from the plot that the values of the series "Sales" are gradually increasing over time.

Moreover there are some ups and downs in the path of the series "Sales" over time.

Again, these ups and downs gradually increase in magnitude over time, specifically from 2011.

This is quite evident from the heights of the spikes created starting from the year 2011 and on.

So, we can apparently say that this series has an upward rising trend.

Also it shows that some kind of seasonality from the year 2011 till the very end.

During some months of each year starting from 2011, the "Sales" go up drastically, as compared to its values during the other months of that year.

The magnitude of this seasonal change in "Sales" taking place each year also increases over the years.

This is evident from the fact that the heights of the spikes go on increasing from one year to the next and so on.

So, it may be apparent that the series "Sales" taken up here suffers from non-stationarity as well.

But we need to conduct the stationarity test of the series "Sales", in order to get certified.

So we conduct the Augmented Dickey-Fuller (ADF) Test to check for stationarity.

For this we specifically require to call up the R library "tseries" and then conduct the test.

library(tseries)  
adf.test(data, alternative = "stationary")

##   
## Augmented Dickey-Fuller Test  
##   
## data: data  
## Dickey-Fuller = -1.024, Lag order = 5, p-value = 0.9318  
## alternative hypothesis: stationary

Here while conducting the ADF Test on the series "Sales" of "data", we pre-specify the alternative hypothesis of the test to be renamed as "stationary".

So, our ADF Test validates for, Ho: The series is Non-Stationary. vs. H1: The series is stationary.

On execution of this test, we see from the results that ADF has automatically considered 5 to be the appropriate order of lag for the series "Sales" at level.

However, the results of this test show that the p-value = 0.9318 (i.e. > 0.05).

So at a default level of significance of 5 % , this test indicates that the observed p-value is much higher than the critical p-value here.

This indicates that here we have to reject the Alternative Hypothesis, but fail to reject the Null Hypothesis.

Thus it is clear that the series "Sales" suffers from Non-Stationarity at level.

Proceeding with this Non-Stationary series for further analysis is not viable.

Had this series been found to be stationary at level only, then we could have proceeded with it.

So, now we need to go for differencing the series "Sales" and then conducting the ADF Test to check if the differenced series is Stationary or not.

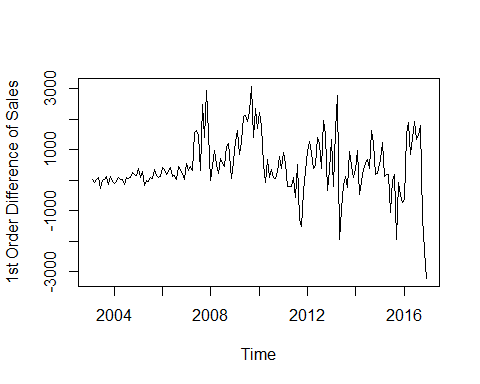
If after taking the 1st difference of the series "Sales", we still find that the ADF Test indicates this 1st differenced series to be Non-Stationary also, then we need to consider the 2nd order difference of the series "Sales" and again check for its Stationarity using ADF Test.

This process should be continued till we obtain stationarity at some higher level of differencing.

So, this time we go for a Time Series plot of the 1st difference of "Sales" from "data".

This will again help us in getting a visual idea.

plot(diff(data), ylab = "1st Order Difference of Sales")



Here, we use "diff(data)" to indicate the 1st order difference of "Sales" from "data", to be considered as the variable for plotting against time.

We re-label the Y-Axis as "1st Order Difference of Sales", just for our convenience.

The graph indicates that the 1st order difference of "Sales" when plotted against time, shows no trend in the Long Run.

However, it exhibits some up and down movement in its path over the years.

Thus some steep spikes are created over time.

Again, the magnitude of fluctuation represented by these spikes increases gradually over the years.

This implies although the mean behaviour of this series is time invariant, but the variation shown by it over the years is certainly time dependent.

So again, it is suspected that this 1st differenced series too may be Non-Stationary.

But, still we conduct the ADF Test to get a strong verification.

So, we first store the 1st order differenced series of "Sales" in a different TS structure.

We have a look at the get up of this new TS structure named as "data1".

Then we conduct the ADF Test on the differenced series of "Sales", which has been separately stored in this new TS structure named "data1".

data1 <- diff(data)  
data1

## Jan Feb Mar Apr May Jun Jul  
## 2003 40.001 -70.000 29.999 90.000 -250.000 0.000  
## 2004 -90.000 -40.001 100.000 20.001 29.998 -119.999 80.000  
## 2005 400.002 109.998 290.001 -180.000 -10.000 -39.999 109.998  
## 2006 420.000 370.001 209.999 280.000 420.000 140.000 170.000  
## 2007 570.000 309.999 470.002 340.000 1569.999 1649.998 1470.001  
## 2008 -9.998 489.998 990.001 509.999 240.001 709.999 590.001  
## 2009 1280.003 1619.999 860.000 1279.999 2099.999 2130.001 1940.002  
## 2010 2230.003 1750.000 529.999 -80.002 670.006 89.996 360.001  
## 2011 919.998 550.003 -200.005 -209.999 -190.002 80.001 -559.997  
## 2012 939.995 1280.007 929.992 400.002 490.005 1390.000 1199.997  
## 2013 1339.997 -190.003 1680.001 2780.006 -1920.006 -939.995 -160.003  
## 2014 370.003 970.001 -459.999 -29.999 349.999 559.997 700.005  
## 2015 670.000 1230.000 130.000 210.000 190.000 -1040.000 0.000  
## 2016 -640.007 1520.004 1899.999 860.000 1360.000 1920.000 1340.000  
## Aug Sep Oct Nov Dec  
## 2003 40.001 139.999 -139.999 129.999 -39.999  
## 2004 70.000 110.001 260.000 189.999 149.999  
## 2005 50.002 369.999 170.000 110.000 140.000  
## 2006 40.001 439.998 360.001 240.000 20.000  
## 2007 320.000 2480.000 1420.002 2939.998 1540.001  
## 2008 439.998 1100.003 1219.997 70.000 610.000  
## 2009 2259.995 3080.001 1390.000 2340.004 1689.995  
## 2010 80.002 49.995 280.006 769.997 400.002  
## 2011 510.002 -1260.002 -1520.004 -369.995 290.000  
## 2012 400.001 1949.997 1490.006 -320.008 350.006  
## 2013 139.999 -239.998 959.999 480.004 79.994  
## 2014 399.994 1630.004 1199.998 210.000 220.000  
## 2015 180.000 -1920.000 -80.000 -510.003 -719.993  
## 2016 1470.000 1790.000 -1230.000 -2300.000 -3210.000

adf.test(data1, alternative = "stationary")

##   
## Augmented Dickey-Fuller Test  
##   
## data: data1  
## Dickey-Fuller = -2.9945, Lag order = 5, p-value = 0.1612  
## alternative hypothesis: stationary

On execution, we again see that the appropraite lag length has been automatically chosen as 5.

The p-value here is 0.1612 (i.e. > 0.05).

This indicates that at 5 % level of significance, we need to reject the Alternative Hypothesis again.

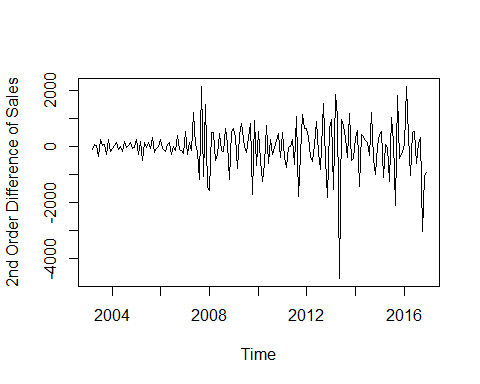
Thus it is evident that this 1st order differenced series of "Sales" is Non-Stationary too.

We cannot proceed with this series even.

So, we go for 2nd order differencing of "Sales" and conduct ADF Test on it, We use the same process as above.

First we plot the 2nd order differenced series of "Sales" and then try to get a visual idea.

plot(diff(data1), ylab = "2nd Order Difference of Sales")



On execution, from the graph we again find that there is no presence of Long Run trend.

However, the spikes/fluctuations are still there showing an increase in their magnitude over time.

So, we suspect presence of time component in the variance, if not in trend.

Again, there is a chance of Non-Stationarity. So we test it.

data2 <- diff(data1)  
data2

## Jan Feb Mar Apr May Jun Jul  
## 2003 -110.001 99.999 60.001 -340.000 250.000  
## 2004 -50.001 49.999 140.001 -79.999 9.997 -149.997 199.999  
## 2005 250.003 -290.004 180.003 -470.001 170.000 -29.999 149.997  
## 2006 280.000 -49.999 -160.002 70.001 140.000 -280.000 30.000  
## 2007 550.000 -260.001 160.003 -130.002 1229.999 79.999 -179.997  
## 2008 -1549.999 499.996 500.003 -480.002 -269.998 469.998 -119.998  
## 2009 670.003 339.996 -759.999 419.999 820.000 30.002 -189.999  
## 2010 540.008 -480.003 -1220.001 -610.001 750.008 -580.010 270.005  
## 2011 519.996 -369.995 -750.008 -9.994 19.997 270.003 -639.998  
## 2012 649.995 340.012 -350.015 -529.990 90.003 899.995 -190.003  
## 2013 989.991 -1530.000 1870.004 1100.005 -4700.012 980.011 779.992  
## 2014 290.009 599.998 -1430.000 430.000 379.998 209.998 140.008  
## 2015 450.000 560.000 -1100.000 80.000 -20.000 -1230.000 1040.000  
## 2016 79.986 2160.011 379.995 -1039.999 500.000 560.000 -580.000  
## Aug Sep Oct Nov Dec  
## 2003 40.001 99.998 -279.998 269.998 -169.998  
## 2004 -10.000 40.001 149.999 -70.001 -40.000  
## 2005 -59.996 319.997 -199.999 -60.000 30.000  
## 2006 -129.999 399.997 -79.997 -120.001 -220.000  
## 2007 -1150.001 2160.000 -1059.998 1519.996 -1399.997  
## 2008 -150.003 660.005 119.994 -1149.997 540.000  
## 2009 319.993 820.006 -1690.001 950.004 -650.009  
## 2010 -279.999 -30.007 230.011 489.991 -369.995  
## 2011 1069.999 -1770.004 -260.002 1150.009 659.995  
## 2012 -799.996 1549.996 -459.991 -1810.014 670.014  
## 2013 300.002 -379.997 1199.997 -479.995 -400.010  
## 2014 -300.011 1230.010 -430.006 -989.998 10.000  
## 2015 180.000 -2100.000 1840.000 -430.003 -209.990  
## 2016 130.000 320.000 -3020.000 -1070.000 -910.000

adf.test(data2, alternative = "stationary")

## Warning in adf.test(data2, alternative = "stationary"): p-value smaller  
## than printed p-value

##   
## Augmented Dickey-Fuller Test  
##   
## data: data2  
## Dickey-Fuller = -4.9091, Lag order = 5, p-value = 0.01  
## alternative hypothesis: stationary

On execution, we see that the appropraite lag order is 5 again.

But this time the p-value = 0.01 (i.e. < 0.05) .

So, at 5 % level of significance, we need to reject the Null Hypothesis.

Thus we fail to reject the Alternative Hypothesis in this case.

So, the 2nd order differenced series of "Sales" is Stationary.

This goes against the usual visual inference that we had made earlier from the graph.

So, this is where testing proves to be necessary for verification.

Thus we can proceed further with this stationary series for further analysis.

But there is another factor we need to take care of.

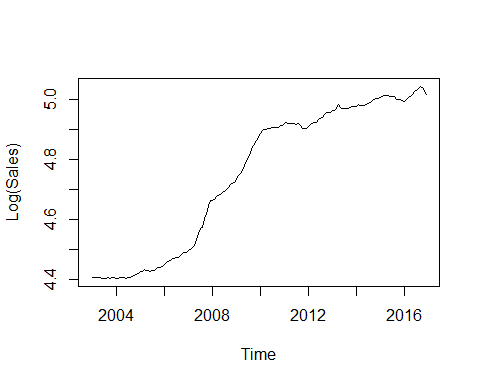
We have seen from all the previous graphs that there were some unevenness in the fluctuations shown by the different ordered differenced series of "Sales" as well as the series "Sales" at level.

This uneven fluctuation/variance if corrected for could improve our results further.

So, we try out with a logarithmic transformation of the series of "Sales" at level.

We plot it first and then try to carry out the further steps as before.

plot(log10(data), ylab = "Log(Sales)")



On execution, we see from the graph that there is a Long term upward trend.

However, the fluctuations/spikes have significantly smoothened up as compared to the graph of the orginal series "Sales" at level.

Next we go for the test of stationarity.

data10 <- log10(data)  
data10

## Jan Feb Mar Apr May Jun Jul  
## 2003 4.404834 4.405517 4.404320 4.404834 4.406370 4.402089 4.402089  
## 2004 4.402777 4.402089 4.403807 4.404149 4.404663 4.402605 4.403978  
## 2005 4.423737 4.425534 4.430236 4.427324 4.427161 4.426511 4.428297  
## 2006 4.448242 4.453930 4.457125 4.461348 4.467608 4.469675 4.472171  
## 2007 4.495960 4.500236 4.506640 4.511215 4.531734 4.552303 4.569842  
## 2008 4.661150 4.665769 4.674953 4.679610 4.681784 4.688153 4.693375  
## 2009 4.733037 4.745855 4.752509 4.762228 4.777717 4.792882 4.806248  
## 2010 4.886491 4.896251 4.899164 4.898725 4.902384 4.902873 4.904824  
## 2011 4.918135 4.921010 4.919967 4.918869 4.917873 4.918293 4.915347  
## 2012 4.907841 4.914660 4.919549 4.921634 4.924176 4.931305 4.937367  
## 2013 4.962748 4.961848 4.969742 4.982497 4.973728 4.969369 4.968623  
## 2014 4.976900 4.981320 4.979230 4.979093 4.980685 4.983220 4.986369  
## 2015 5.005352 5.010597 5.011147 5.012035 5.012837 5.008430 5.008430  
## 2016 4.992421 4.999087 5.007278 5.010936 5.016657 5.024609 5.030073  
## Aug Sep Oct Nov Dec  
## 2003 4.402777 4.405176 4.402777 4.405005 4.404320  
## 2004 4.405176 4.407051 4.411451 4.414639 4.417139  
## 2005 4.429106 4.435048 4.437751 4.439491 4.441695  
## 2006 4.472756 4.479143 4.484300 4.487704 4.487986  
## 2007 4.573568 4.601408 4.616581 4.646404 4.661245  
## 2008 4.697229 4.706718 4.717004 4.717587 4.722634  
## 2009 4.821317 4.841046 4.849665 4.863799 4.873727  
## 2010 4.905256 4.905526 4.907035 4.911158 4.913284  
## 2011 4.918030 4.911371 4.903199 4.901186 4.902764  
## 2012 4.939369 4.948999 4.956216 4.954677 4.956361  
## 2013 4.969276 4.968156 4.972619 4.974834 4.975202  
## 2014 4.988157 4.995372 5.000608 5.001517 5.002468  
## 2015 5.009196 5.000954 5.000608 4.998390 4.995240  
## 2016 5.035990 5.043087 5.038223 5.028978 5.015737

adf.test(data10, alternative = "stationary")

##   
## Augmented Dickey-Fuller Test  
##   
## data: data10  
## Dickey-Fuller = -0.50572, Lag order = 5, p-value = 0.9805  
## alternative hypothesis: stationary

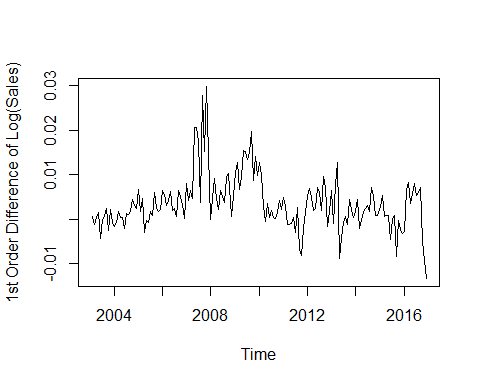
On execution, we see from the test results that the p-value = 0.9805 (i.e. > 0.05) .

So at 5 % level of significance, we have to reject the Alternative Hypothesis.

Thus this series is Non-Stationary and we cannot proceed with this series futher.

We now take differences as before and proceed thereafter.

plot(diff(data10), ylab = "1st Order Difference of Log(Sales)")



The graph shows that apparently there is very little downward trend over time.

The fluctuations/spikes are still there and very haphazard in nature.

We conduct the ADF Test as before, to check for Non-Stationarity.

data11 <- diff(data10)  
data11

## Jan Feb Mar Apr May  
## 2003 6.834074e-04 -1.196640e-03 5.132323e-04 1.536119e-03  
## 2004 -1.543398e-03 -6.877362e-04 1.717260e-03 3.426558e-04 5.134174e-04  
## 2005 6.597873e-03 1.796938e-03 4.702149e-03 -2.912567e-03 -1.623834e-04  
## 2006 6.547277e-03 5.687195e-03 3.195019e-03 4.223807e-03 6.259672e-03  
## 2007 7.974064e-03 4.276066e-03 6.403858e-03 4.574395e-03 2.051959e-02  
## 2008 -9.473276e-05 4.618675e-03 9.184407e-03 4.656615e-03 2.174204e-03  
## 2009 1.040278e-02 1.281850e-02 6.654205e-03 9.718876e-03 1.548845e-02  
## 2010 1.276336e-02 9.759837e-03 2.913074e-03 -4.384709e-04 3.658536e-03  
## 2011 4.851314e-03 2.874591e-03 -1.043126e-03 -1.097953e-03 -9.957987e-04  
## 2012 5.076972e-03 6.819325e-03 4.888286e-03 2.085713e-03 2.541435e-03  
## 2013 6.387485e-03 -9.000086e-04 7.893923e-03 1.275542e-02 -8.769338e-03  
## 2014 1.698003e-03 4.420227e-03 -2.090576e-03 -1.366878e-04 1.592069e-03  
## 2015 2.883686e-03 5.244600e-03 5.506246e-04 8.879984e-04 8.018656e-04  
## 2016 -2.819277e-03 6.666084e-03 8.191220e-03 3.657417e-03 5.721680e-03  
## Jun Jul Aug Sep Oct  
## 2003 -4.280485e-03 0.000000e+00 6.877362e-04 2.398459e-03 -2.398459e-03  
## 2004 -2.057442e-03 1.372722e-03 1.197583e-03 1.875286e-03 4.400527e-03  
## 2005 -6.501253e-04 1.785520e-03 8.092267e-04 5.941617e-03 2.702921e-03  
## 2006 2.066667e-03 2.496374e-03 5.853172e-04 6.386770e-03 5.156606e-03  
## 2007 2.056878e-02 1.753880e-02 3.725874e-03 2.784029e-02 1.517249e-02  
## 2008 6.368980e-03 5.222404e-03 3.854174e-03 9.488465e-03 1.028660e-02  
## 2009 1.516501e-02 1.336610e-02 1.506911e-02 1.972950e-02 8.618590e-03  
## 2010 4.890793e-04 1.950920e-03 4.323598e-04 2.699727e-04 1.508936e-03  
## 2011 4.195628e-04 -2.945435e-03 2.683285e-03 -6.659640e-03 -8.172182e-03  
## 2012 7.129411e-03 6.062121e-03 2.002058e-03 9.629969e-03 7.217044e-03  
## 2013 -4.358724e-03 -7.463048e-04 6.530699e-04 -1.120149e-03 4.463397e-03  
## 2014 2.535227e-03 3.148401e-03 1.788852e-03 7.214451e-03 5.235672e-03  
## 2015 -4.407398e-03 0.000000e+00 7.660267e-04 -8.241455e-03 -3.468113e-04  
## 2016 7.951451e-03 5.464435e-03 5.916526e-03 7.097315e-03 -4.864434e-03  
## Nov Dec  
## 2003 2.227578e-03 -6.841807e-04  
## 2004 3.187788e-03 2.500246e-03  
## 2005 1.740028e-03 2.204545e-03  
## 2006 3.404024e-03 2.824680e-04  
## 2007 2.982318e-02 1.484089e-02  
## 2008 5.828898e-04 5.046626e-03  
## 2009 1.413351e-02 9.928813e-03  
## 2010 4.122640e-03 2.126304e-03  
## 2011 -2.012740e-03 1.578364e-03  
## 2012 -1.539921e-03 1.683996e-03  
## 2013 2.214634e-03 3.679796e-04  
## 2014 9.097898e-04 9.510733e-04  
## 2015 -2.217470e-03 -3.149898e-03  
## 2016 -9.244933e-03 -1.324083e-02

adf.test(data11, alternative = "stationary")

##   
## Augmented Dickey-Fuller Test  
##   
## data: data11  
## Dickey-Fuller = -2.7906, Lag order = 5, p-value = 0.2463  
## alternative hypothesis: stationary

On execution, we see from the results that the p-value = 0.2463 (i.e. > 0.05) .

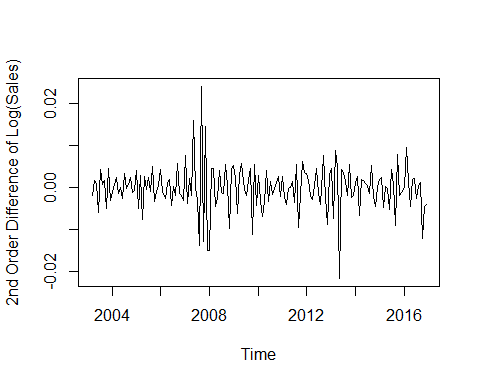
Thus at 5 % level of significance, we reject the Alternative Hypothesis.

We infer that this series is again Non-Stationary.

And we cannot proceed further with this.

Next, we consider the 2nd order difference of the series "Log(Sales)" and proceed further.

plot(diff(data11), ylab = "2nd Order Difference of Log(Sales)")



From the graph, it is apparent that there is no Long Run trend.

And above all the fluctuations have eased down a lot, reducing the unevenness.

We check for Stationarity of this.

data12 <- diff(data11)  
data12

## Jan Feb Mar Apr May  
## 2003 -1.880047e-03 1.709872e-03 1.022887e-03  
## 2004 -8.592168e-04 8.556613e-04 2.404996e-03 -1.374604e-03 1.707616e-04  
## 2005 4.097627e-03 -4.800935e-03 2.905211e-03 -7.614716e-03 2.750184e-03  
## 2006 4.342732e-03 -8.600823e-04 -2.492176e-03 1.028788e-03 2.035865e-03  
## 2007 7.691596e-03 -3.697998e-03 2.127792e-03 -1.829463e-03 1.594520e-02  
## 2008 -1.493563e-02 4.713407e-03 4.565732e-03 -4.527792e-03 -2.482411e-03  
## 2009 5.356159e-03 2.415719e-03 -6.164298e-03 3.064670e-03 5.769571e-03  
## 2010 2.834549e-03 -3.003525e-03 -6.846764e-03 -3.351544e-03 4.097006e-03  
## 2011 2.725010e-03 -1.976723e-03 -3.917717e-03 -5.482681e-05 1.021542e-04  
## 2012 3.498608e-03 1.742354e-03 -1.931039e-03 -2.802574e-03 4.557226e-04  
## 2013 4.703490e-03 -7.287494e-03 8.793931e-03 4.861497e-03 -2.152476e-02  
## 2014 1.330023e-03 2.722224e-03 -6.510802e-03 1.953888e-03 1.728757e-03  
## 2015 1.932613e-03 2.360913e-03 -4.693975e-03 3.373738e-04 -8.613282e-05  
## 2016 3.306212e-04 9.485361e-03 1.525136e-03 -4.533803e-03 2.064263e-03  
## Jun Jul Aug Sep Oct  
## 2003 -5.816604e-03 4.280485e-03 6.877362e-04 1.710723e-03 -4.796919e-03  
## 2004 -2.570859e-03 3.430164e-03 -1.751392e-04 6.777030e-04 2.525242e-03  
## 2005 -4.877419e-04 2.435645e-03 -9.762934e-04 5.132390e-03 -3.238695e-03  
## 2006 -4.193005e-03 4.297072e-04 -1.911057e-03 5.801452e-03 -1.230164e-03  
## 2007 4.918095e-05 -3.029973e-03 -1.381293e-02 2.411441e-02 -1.266780e-02  
## 2008 4.194776e-03 -1.146576e-03 -1.368230e-03 5.634291e-03 7.981339e-04  
## 2009 -3.234331e-04 -1.798919e-03 1.703015e-03 4.660392e-03 -1.111091e-02  
## 2010 -3.169456e-03 1.461841e-03 -1.518560e-03 -1.623871e-04 1.238964e-03  
## 2011 1.415362e-03 -3.364997e-03 5.628720e-03 -9.342925e-03 -1.512542e-03  
## 2012 4.587976e-03 -1.067290e-03 -4.060063e-03 7.627912e-03 -2.412925e-03  
## 2013 4.410614e-03 3.612419e-03 1.399375e-03 -1.773219e-03 5.583546e-03  
## 2014 9.431576e-04 6.131747e-04 -1.359549e-03 5.425599e-03 -1.978779e-03  
## 2015 -5.209263e-03 4.407398e-03 7.660267e-04 -9.007482e-03 7.894644e-03  
## 2016 2.229771e-03 -2.487017e-03 4.520916e-04 1.180789e-03 -1.196175e-02  
## Nov Dec  
## 2003 4.626038e-03 -2.911759e-03  
## 2004 -1.212739e-03 -6.875414e-04  
## 2005 -9.628939e-04 4.645177e-04  
## 2006 -1.752582e-03 -3.121556e-03  
## 2007 1.465068e-02 -1.498228e-02  
## 2008 -9.703709e-03 4.463736e-03  
## 2009 5.514917e-03 -4.204695e-03  
## 2010 2.613704e-03 -1.996336e-03  
## 2011 6.159442e-03 3.591103e-03  
## 2012 -8.756965e-03 3.223916e-03  
## 2013 -2.248763e-03 -1.846655e-03  
## 2014 -4.325882e-03 4.128354e-05  
## 2015 -1.870658e-03 -9.324280e-04  
## 2016 -4.380499e-03 -3.995898e-03

adf.test(data12, alternative = "stationary")

## Warning in adf.test(data12, alternative = "stationary"): p-value smaller  
## than printed p-value

##   
## Augmented Dickey-Fuller Test  
##   
## data: data12  
## Dickey-Fuller = -5.2646, Lag order = 5, p-value = 0.01  
## alternative hypothesis: stationary

On execution, we see that the test results give us a p-value = 0.01 (i.e. < 0.05) .

So, at 5 % level of significance, we fail to reject the Alternative Hypothesis.

This implies that the 2nd Order Difference of the series "Log(Sales)" is Stationary.

Thus we can proceed further with this series.

Till now we have finalised two different statioanry series for further analysis.

They are :- 2nd Order Difference of the series "Sales" and 2nd Order Difference of the series "Log(Sales)"

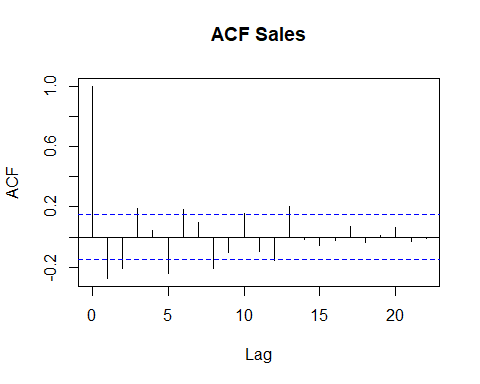
Now we will consider further analysis with these two series separately.

Ultimately we will keep that one only which shows a better fit at the final stage.

Our next task is to find out the most appropraite Model for the above two series.

For that we at first need to create the ACF and PACF plots for both separately.

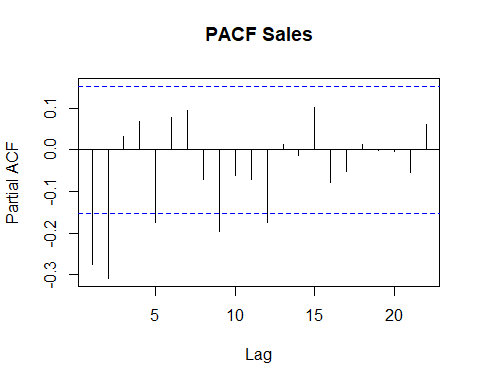
acf(ts(diff(diff(data))), main = "ACF Sales")



We see from the ACF Plot, that the order of Moving Average is expected to be 0.

This is because the last spike which is significantly outside the limits is at lag 0.

pacf(ts(diff(diff(data))), main = "PACF Sales")



We see from the PACF Plot, that the order of Auto Regression is expected to be 2.

This is because the last spike which is significantly outside the limits is at lag 2.

So we expect the model to be ARIMA(2,2,0)

The order of integration was previously determined as 2.

We now fit the ARIMA model as per our expectations.

Here we fit the ARIMA(2,2,0) model on the 1st order difference of the series "Sales".

This is because though the order of integration is 2, i.e. we need to work with the 2nd order difference of the series "Sales", the ARMA process requires one lag lower order of integration while operating.

Thus, we effectively need to model the 1st order differenced values of the series "Sales".

## Fitting appropriate models

ARIMAfit <- arima(diff(data), c(2,2,0))

Next we look at the summary of the fit stored under the name "ARIMAfit".

summary(ARIMAfit)

##   
## Call:  
## arima(x = diff(data), order = c(2, 2, 0))  
##   
## Coefficients:  
## ar1 ar2  
## -0.8322 -0.5793  
## s.e. 0.0634 0.0633  
##   
## sigma^2 estimated as 865420: log likelihood = -1362.55, aic = 2731.1  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set -20.51933 924.6924 648.394 Inf Inf 1.148089 -0.2322592

From the summary results we get the various coefficients of model equation fitted.

We just take a note of the values of the different "Training set error measures" given.

These Error Measures help us to compare between two or more fitted models.

The model with the lowest error values is considered to be the best fit.

However the above ARIMA(2,2,0) model was fit on the basis of rough observations done from the ACF and PACF plots.

They may not be accurate always.

So we fit an auto arima model for better accuracy.

Auto Arima model helps in finding the optimum values of orders of AR and MA and fits the best model possible.

So for running the auto arima we need to call up the library "forecast" at first.

Then we go on to fit the auto arima model.

require(forecast)

ARIMAautofit <- auto.arima(diff(data), approximation = TRUE, trace = TRUE)

##   
## Fitting models using approximations to speed things up...  
##   
## ARIMA(2,0,2)(1,0,1)[12] with non-zero mean : 2709.693  
## ARIMA(0,0,0) with non-zero mean : 2756.933  
## ARIMA(1,0,0)(1,0,0)[12] with non-zero mean : 2707.042  
## ARIMA(0,0,1)(0,0,1)[12] with non-zero mean : 2711.647  
## ARIMA(0,0,0) with zero mean : 2793.519  
## ARIMA(1,0,0) with non-zero mean : 2696.095  
## ARIMA(1,0,0)(0,0,1)[12] with non-zero mean : 2696.215  
## ARIMA(1,0,0)(1,0,1)[12] with non-zero mean : 2707.214  
## ARIMA(2,0,0) with non-zero mean : 2698.12  
## ARIMA(1,0,1) with non-zero mean : 2696.075  
## ARIMA(2,0,2) with non-zero mean : 2698.491  
## ARIMA(1,0,1) with zero mean : 2697.356  
## ARIMA(1,0,1)(1,0,0)[12] with non-zero mean : 2707.149  
## ARIMA(1,0,1)(0,0,1)[12] with non-zero mean : 2695.977  
## ARIMA(1,0,1)(1,0,2)[12] with non-zero mean : 2707.815  
## ARIMA(2,0,1)(0,0,1)[12] with non-zero mean : 2698.429  
## ARIMA(1,0,2)(0,0,1)[12] with non-zero mean : 2696.468  
## ARIMA(0,0,0)(0,0,1)[12] with non-zero mean : 2758.98  
## ARIMA(2,0,2)(0,0,1)[12] with non-zero mean : 2698.638  
## ARIMA(1,0,1)(0,0,1)[12] with zero mean : 2697.436  
## ARIMA(1,0,1)(1,0,1)[12] with non-zero mean : 2706.987  
## ARIMA(1,0,1)(0,0,2)[12] with non-zero mean : 2696.923  
##   
## Now re-fitting the best model(s) without approximations...  
##   
## ARIMA(1,0,1)(0,0,1)[12] with non-zero mean : 2695.91  
##   
## Best model: ARIMA(1,0,1)(0,0,1)[12] with non-zero mean

On running this, we see the most accurate ARIMA model that could be fitted is automatically detected.

The best model fit here is ARIMA(1,0,1)(0,0,1)[12].

So the best fit model indicates AR(1) and MA(1).

The seasonality component is automatically best fitted with order (0,0,1).

Now to get a summary of this best fit model we do the following.

summary(ARIMAautofit)

## Series: diff(data)   
## ARIMA(1,0,1)(0,0,1)[12] with non-zero mean   
##   
## Coefficients:  
## ar1 ma1 sma1 mean  
## 0.7663 -0.2481 -0.1114 412.3128  
## s.e. 0.0980 0.1420 0.0795 166.9744  
##   
## sigma^2 estimated as 575753: log likelihood=-1342.77  
## AIC=2695.54 AICc=2695.91 BIC=2711.13  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE ACF1  
## Training set 1.876461 749.6416 504.2896 NaN Inf 0.5500694 0.02094562

The summary gives us the various coefficients of the best fit equation.

Again we note the values of the different "Training set error measures" given.

These error values are supposed to be the lowest among all possible fits.

To check this claim, we compare the values of RMSE obtained in case of ARIMAfit with that of ARIMAautofit.

ARIMAfit had a RMSE value of 924, while ARIMAautofit has a RMSE value of 749.

Thus it is evident that ARIMAautofit gives us the model with the best fit possible.

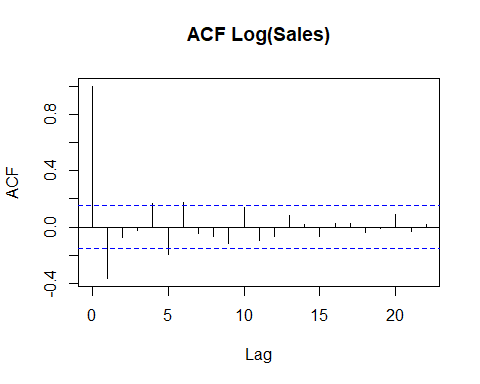
So, till now we have been able to fit the best model for 2nd Order Difference of the series "Sales".

Now the same needs to be done with the 2nd Order Difference of the series "Log(Sales)" .

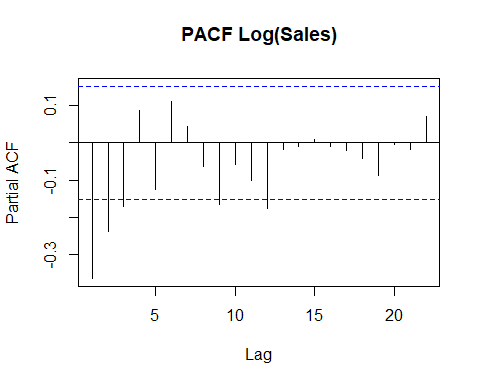
Then we will be comparing between the two Best Fit models in terms of lower RMSE values.

We carry out the ACF and PACF plots to have a visual idea of the best fit possible.

acf(ts(diff(diff(log10(data)))), main = "ACF Log(Sales)")



pacf(ts(diff(diff(log10(data)))), main = "PACF Log(Sales)")



Next we auto fit the arima model on 2nd Order Difference of the series "Log(Sales)"

So we effectively use the 1st Order Difference of the series "Log(Sales)" as the variable.

ARIMAautofit2 <- auto.arima(diff(log10(data)), approximation = TRUE, trace = TRUE)

##   
## Fitting models using approximations to speed things up...  
##   
## ARIMA(2,1,2)(1,0,1)[12] with drift : -1271.978  
## ARIMA(0,1,0) with drift : -1248.794  
## ARIMA(1,1,0)(1,0,0)[12] with drift : -1260.073  
## ARIMA(0,1,1)(0,0,1)[12] with drift : -1282.844  
## ARIMA(0,1,0) : -1250.803  
## ARIMA(0,1,1)(1,0,1)[12] with drift : -1272.231  
## ARIMA(0,1,1) with drift : -1283.515  
## ARIMA(1,1,1) with drift : -1281.146  
## ARIMA(0,1,2) with drift : -1282.21  
## ARIMA(1,1,2) with drift : -1279.153  
## ARIMA(0,1,1) : -1285.505  
## ARIMA(0,1,1)(1,0,0)[12] : -1275.641  
## ARIMA(0,1,1)(0,0,1)[12] : -1284.852  
## ARIMA(0,1,1)(1,0,1)[12] : -1274.276  
## ARIMA(1,1,1) : -1283.198  
## ARIMA(0,1,2) : -1284.237  
## ARIMA(1,1,2) : -1281.222  
##   
## Now re-fitting the best model(s) without approximations...  
##   
## ARIMA(0,1,1) : -1295.637  
##   
## Best model: ARIMA(0,1,1)

This fiting is given a name of "ARIMAautofit2".

The best model that can be fitted in this case is determined automatically to be ARIMA(0,1,1).

ARIMA(0,1,1) model implies this is a purely Moving Average Series.

Note: We have intentionally not performed the manual fitting as it has already been explained in the previous case.

Now we need to get a summary of the latest best fit model named as "ARIMAautofit2".

summary(ARIMAautofit2)

## Series: diff(log10(data))   
## ARIMA(0,1,1)   
##   
## Coefficients:  
## ma1  
## -0.5347  
## s.e. 0.0731  
##   
## sigma^2 estimated as 2.338e-05: log likelihood=649.86  
## AIC=-1295.71 AICc=-1295.64 BIC=-1289.49  
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE MASE  
## Training set -0.0001096626 0.004806686 0.003329635 Inf Inf 0.5568284  
## ACF1  
## Training set 0.02903617

The summary gives us the coefficients of the best fit model equation.

We take a note of the values of the different "Training set error measures".

Here we see that the value of RMSE for this fit is 0.0048 (approx.).

However we cannot compare the 2 best fits represented by ARIMAautofit ~ ARIMA(1,0,1)(0,0,1)[12] and ARIMAautofit2 ~ ARIMA(0,1,1).

This is because ARIMAautofit is a perfect Auto Regressive Moving Average process.

While ARIMAautofit2 is a purely Moving Average Process.

We need to check which one is a better fit graphically after making predictions using them.

Getting the most accurate Predicted Results using a fitted model is always our target.

A fitted model which does not predict accurately is useless.

So, we will use both the models viz. "ARIMAautofit" and "ARIMAautofit2" to make separate predictions.

As ARIMAautofit ~ ARIMA(1,0,1)(0,0,1)[12] , it is expected to give us the better predictive results.

But as ARIMAautofit2 ~ ARIMA(0,1,1) i.e. actually an MA(1) process, it is expected that using the Exponential Smoothing Method would increase the predictive power of this model.

### Prediction using ARIMAautofit ~ ARIMA(1,0,1)(0,0,1)[12]

pred <- predict(ARIMAautofit, n.ahead = 36)  
pred

## $pred  
## Jan Feb Mar Apr May  
## 2017 -1821.998536 -1521.463105 -1127.913509 -691.513134 -499.836912  
## 2018 577.705657 539.053828 509.434819 486.737685 469.344806  
## 2019 419.094244 417.509439 416.294998 415.364368 414.651224  
## Jun Jul Aug Sep Oct  
## 2017 -358.187199 -161.766874 -66.984202 6.921324 386.274926  
## 2018 456.016591 445.803141 437.976545 431.979001 427.383066  
## 2019 414.104740 413.685967 413.365061 413.119149 412.930706  
## Nov Dec  
## 2017 562.325349 704.691036  
## 2018 423.861188 421.162362  
## 2019 412.786302 412.675645  
##   
## $se  
## Jan Feb Mar Apr May Jun Jul  
## 2017 758.7839 854.5981 906.1530 935.1034 951.6934 961.3019 966.8997  
## 2018 976.3260 976.7186 976.9491 977.0844 977.1639 977.2106 977.2380  
## 2019 977.2753 977.2760 977.2764 977.2766 977.2767 977.2768 977.2769  
## Aug Sep Oct Nov Dec  
## 2017 970.1718 972.0881 973.2117 973.8708 974.2577  
## 2018 977.2540 977.2635 977.2690 977.2723 977.2742  
## 2019 977.2769 977.2769 977.2769 977.2769 977.2769

Here we make predictions for the next 36 months, i.e. for a stretch of 3 years in the future.

For this we use the option "n.ahead = 36" .

We store the results of this under the name "pred".

To see the Predicted Values, we use the following.

pred$pred

## Jan Feb Mar Apr May  
## 2017 -1821.998536 -1521.463105 -1127.913509 -691.513134 -499.836912  
## 2018 577.705657 539.053828 509.434819 486.737685 469.344806  
## 2019 419.094244 417.509439 416.294998 415.364368 414.651224  
## Jun Jul Aug Sep Oct  
## 2017 -358.187199 -161.766874 -66.984202 6.921324 386.274926  
## 2018 456.016591 445.803141 437.976545 431.979001 427.383066  
## 2019 414.104740 413.685967 413.365061 413.119149 412.930706  
## Nov Dec  
## 2017 562.325349 704.691036  
## 2018 423.861188 421.162362  
## 2019 412.786302 412.675645

To see the Standard Errors of the Predicted Values we use the following.

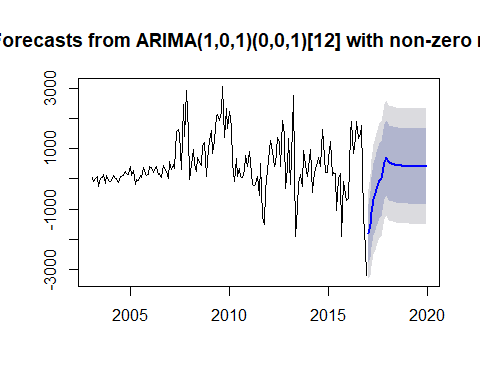
pred$se

## Jan Feb Mar Apr May Jun Jul  
## 2017 758.7839 854.5981 906.1530 935.1034 951.6934 961.3019 966.8997  
## 2018 976.3260 976.7186 976.9491 977.0844 977.1639 977.2106 977.2380  
## 2019 977.2753 977.2760 977.2764 977.2766 977.2767 977.2768 977.2769  
## Aug Sep Oct Nov Dec  
## 2017 970.1718 972.0881 973.2117 973.8708 974.2577  
## 2018 977.2540 977.2635 977.2690 977.2723 977.2742  
## 2019 977.2769 977.2769 977.2769 977.2769 977.2769

Next we go for plotting the observed and the predicted data together, using "ARIMAautofit"

This will help us to get a visual idea about the accuracy of the fit.

plot(forecast(ARIMAautofit, h = 36))



We extend the plot by forecasting the values for the next 36 months i.e. the next 3 years.

The predictions in this case has been done starting from Jan-2017 to Dec-2019 .

The plot shows the predicted values of the series with a blue line.

The predicted line shows an upward trend starting from Jan-2017 till Dec-2017, after which its falls for a while and then flattens out.

The inner purple zone shows the 80 % Confidence Interval of our prediction.

The inner Purple zone added with the outer Grey zone together shows the 95 % Confidence Interval of our prediction.

It can be seen that the Confidence Intervals become wider gradually over the prediction range.

They keep on widening from the Jan-2017 till Dec-2017, after which they become constant through out.

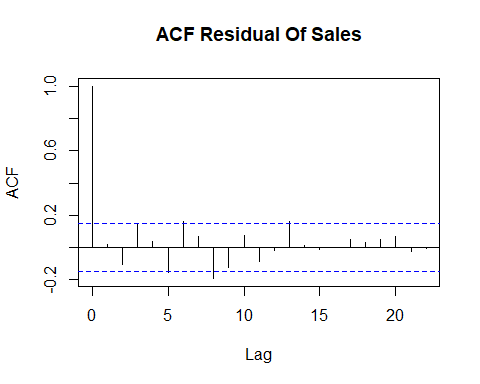
So, it is apparent that the forecast done here is not that accurate after 1 year.

### Residual Analysis

Next we go for a mandatory Residual Analysis of ARIMAautofit.

We look at the ACF and the PACF plots of ARIMAautofit.

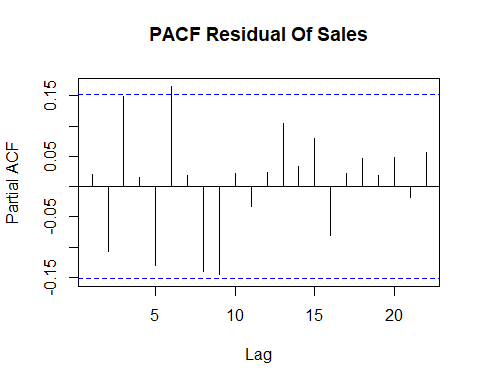
acf(ts(ARIMAautofit$residuals), main = "ACF Residual Of Sales")



The ACF graph shows that the spikes are within the acceptable limits at all lags.

So, there is no Moving Average pattern prevalent in the Residuals.

pacf(ts(ARIMAautofit$residuals), main = "PACF Residual Of Sales")



The PACF graph shows that the spikes are within the acceptable limits at all lags.

So, there is no Auto Regressive pattern prevalent in the Residuals.

Thus, due to the absence of both AR and MA patterns in the Residuals we can say that the Residuals in this case are random in nature, which is actually desired.

### Prediction using ARIMAautofit2 ~ ARIMA(0,1,1).

Here it is evident that ARIMAautofit2 actually ~ MA(1).

Thus it is a pure Moving Average process.

Thus instead of fitting it with an ARIMA model, it is desirable to re-fit it into an MA structure.

For this we will be using Exponential Smoothing.

For our convenience and to save time, we refrain from carryimg out the step by step process of Simple Exponential Smoothing, Double Exponential Smoothing and Triple Exponential Smoothing.

Instead we go for an Automatic Exponential Smoothing using the State Space Model approach.

autoexp\_fit <- ets(data10)

Here we perform an Exponential Smoothing by using the "ets( )" function.

We store the newly fitted model by the name "autoexp\_fit".

Next we go look into the summary of this fit to make some decisions.

summary(autoexp\_fit)

## ETS(M,Ad,N)   
##   
## Call:  
## ets(y = data10)   
##   
## Smoothing parameters:  
## alpha = 0.9999   
## beta = 0.4286   
## phi = 0.9275   
##   
## Initial states:  
## l = 4.4076   
## b = -0.002   
##   
## sigma: 0.001  
##   
## AIC AICc BIC   
## -926.7532 -926.2315 -908.0095   
##   
## Training set error measures:  
## ME RMSE MAE MPE MAPE  
## Training set 0.0004934518 0.004729652 0.003297052 0.01088269 0.0691976  
## MASE ACF1  
## Training set 0.06786506 0.05123158

From the results we get an idea about the values of the different "Training set error measures".

We see that this fit has an RMSE of 0.0047.

We note down this RMSE value for "autoexp\_fit" and compare it with the RMSE values obtained from several other model fits which also have undergone Exponential Smoothing.

The model fit with the lowest RMSE value, obtained after Exponential Smoothing, will be considered as the best fit.

We see that earlier in case of "ARIMAautofit2" which was not fitted after Exponential Smoothing, the RMSE was obtained as to be equal to 0.0048.

But now, under Exponential Smoothing, we find RMSE value for "autoexp\_fit" to be 0.0047.

So, there has been no significant increase in the accuracy of the model in this case.

We go for predicting using this "autoexp\_fit", for the next 36 months.

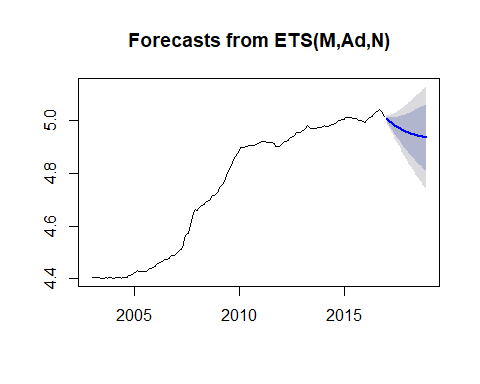
Thus we would be forecasting the values for the next 3 years, i.e. from Jan-2017 till Dec-2019.

pred\_ets <- predict(autoexp\_fit, h = 36)

Next we go for plotting the observed and the forecasted data together, using "autoexp\_fit"

This will help us to get a visual idea about the accuracy of the fit.

plot(forecast(autoexp\_fit))



From the graph, we see that the predicted value of "Sales", given by the deep Blue line, declines gradually from Jan-2017 till Dec-2019.

The inner purple zone shows the 80 % Confidence Interval of our prediction.

The inner Purple zone added with the outer Grey zone together shows the 95 % Confidence Interval of our prediction.

It can be seen that the Confidence Intervals become wider gradually over the prediction range.

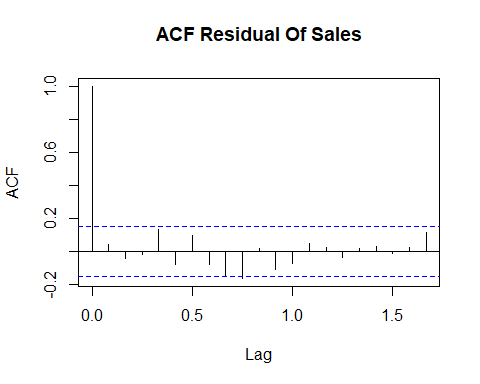
So, it is apparent that the accuracy of the forecast declines gradually with time.

Next we go for a mandatory Residual Analysis of "autoexp\_fit".

We look at the ACF plot of “autoexp\_fit".

Here as this is a purely MA process, no PACF plot is required.

acf(autoexp\_fit$residuals, lag.max = 20, main = "ACF Residual Of Sales")



The ACF graph shows that the spikes are within the acceptable limits at all lags.

So, there is no Moving Average pattern prevalent in the Residuals.

It can be said that the Residuals are random in nature, which is actually desired.

**Thus we show how an analysis of Time Series Data can be done accurately.**

**At last, just to recapitulate we again recall that a proper ARIMA series is better fitted using the "auto.arima( )" process, while in most cases a purely MA series is better fitted using the Exponential Smoothing Method of "ets( )".**

## Thank You.