Sales Forecast

Saurabh Mudgal

28 May 2019

## R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

# read the data   
#Set working directory  
  
library("forecast")

## Warning: package 'forecast' was built under R version 3.5.3

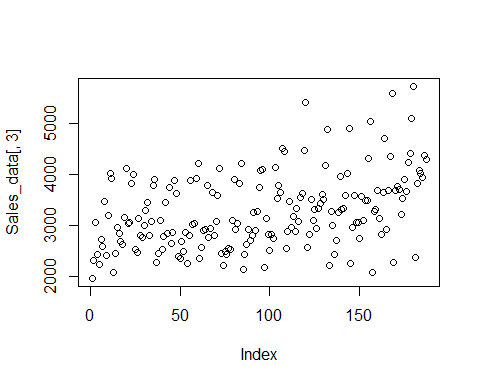
setwd("C://BACP//Module 6 - Time Series Forecasting//Project")  
getwd()

## [1] "C:/BACP/Module 6 - Time Series Forecasting/Project"

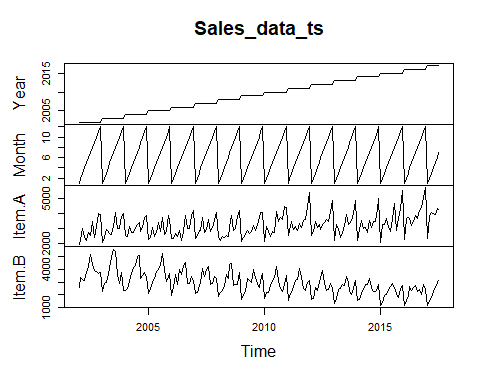
Sales\_data = read.csv("Demand.csv", skip=1)  
attach(Sales\_data)  
  
head(Sales\_data)

## Year Month Item.A Item.B  
## 1 2002 1 1954 2585  
## 2 2002 2 2302 3368  
## 3 2002 3 3054 3210  
## 4 2002 4 2414 3111  
## 5 2002 5 2226 3756  
## 6 2002 6 2725 4216

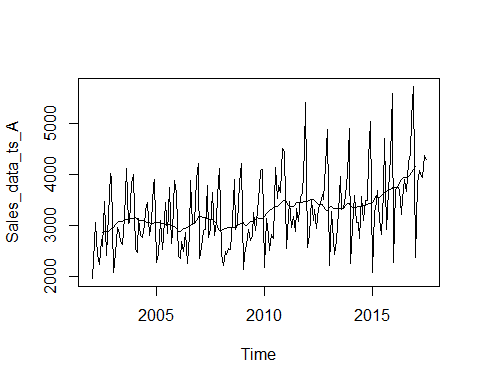
View(Sales\_data)  
  
plot(Sales\_data[,3])



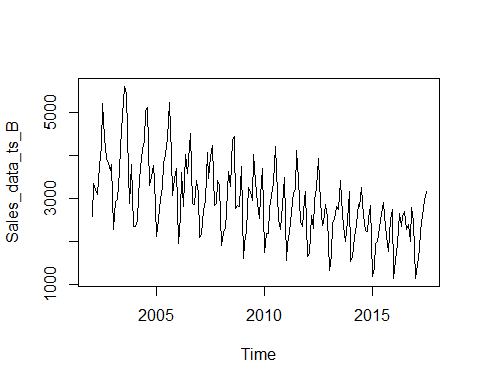
#create time series object item A  
  
Sales\_data\_ts = ts(Sales\_data,frequency = 12, start = c(2002,1))  
plot.ts(Sales\_data\_ts)



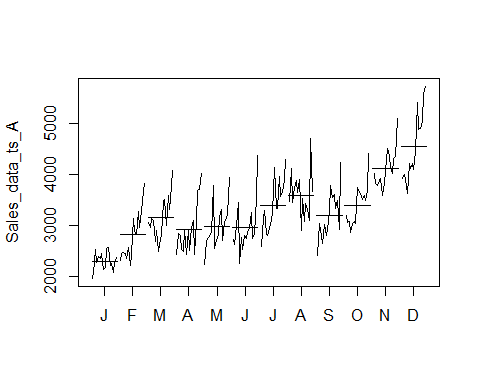
Sales\_data\_ts\_A = ts(Sales\_data[,3], frequency = 12,start = c(2002,1))  
  
library(forecast)  
trend\_A = ma(Sales\_data\_ts\_A,12,centre =T)  
plot(Sales\_data\_ts\_A)  
lines(trend\_A)



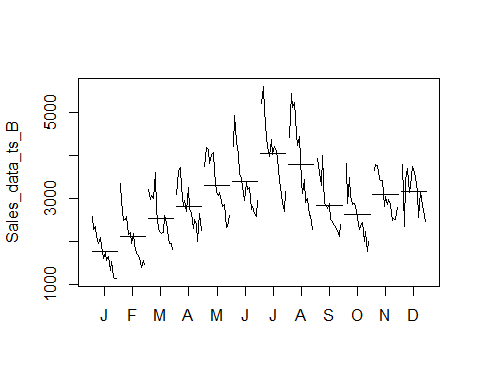
#to get the straight line of the plot.  
#abline(reg=lm(Sales\_data\_ts\_A~time(Sales\_data\_ts\_A)))  
  
#create time series object item B  
  
Sales\_data\_ts\_B = ts(Sales\_data[,4], frequency = 12,start = c(2002,1))  
plot.ts(Sales\_data\_ts\_B)



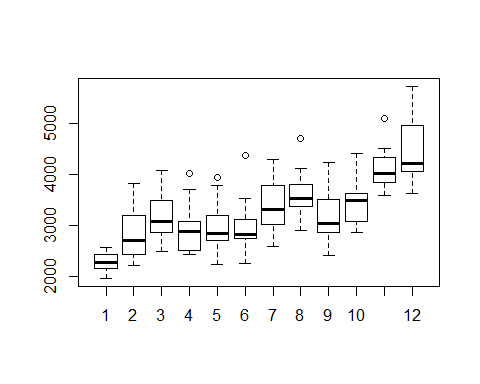
#month plot  
monthplot(Sales\_data\_ts\_A)



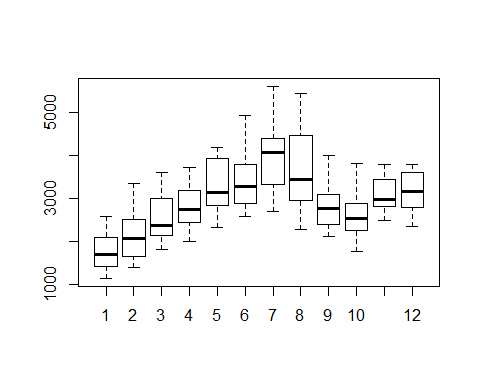
monthplot(Sales\_data\_ts\_B)



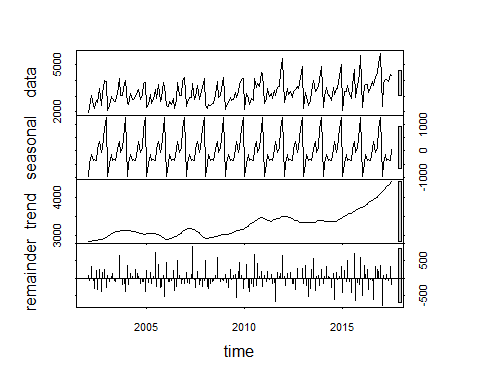
boxplot(Sales\_data\_ts\_A~cycle(Sales\_data\_ts\_A))



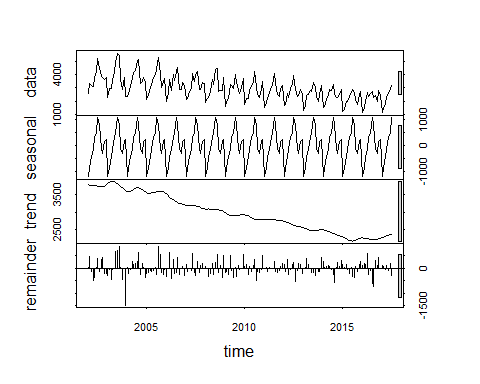
boxplot(Sales\_data\_ts\_B~cycle(Sales\_data\_ts\_B))



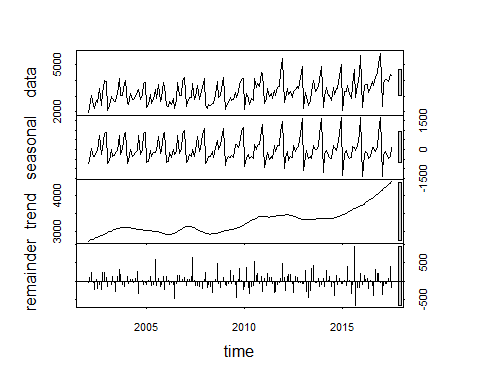
##################################################################  
## Addtitive or Multiplicative   
##################################################################  
##Decompose TS data into seaosonlity ,trend and irregular components   
  
#Sales\_data\_CompA = decompose(Sales\_data\_ts\_A)  
  
#assume seasonality is constant - additive model  
itemA\_Dec = stl(Sales\_data\_ts\_A,s.window = "p")  
itemB\_Dec = stl(Sales\_data\_ts\_B,s.window = "p")  
plot(itemA\_Dec)



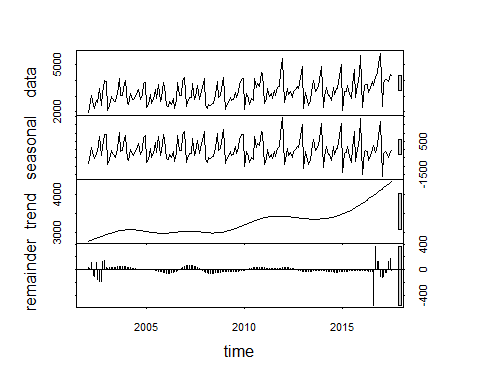
plot(itemB\_Dec)



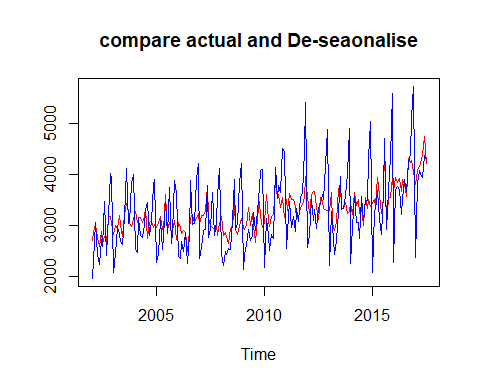
#itemA\_Dec\_D = decompose(Sales\_data\_ts\_A,"multiplicative")  
#plot(itemA\_Dec\_D)  
  
  
  
#assume seasonality is not constant - Multiplicative model  
 itemA\_Dec7 = stl(Sales\_data\_ts\_A,s.window = 7)  
 plot(itemA\_Dec7)



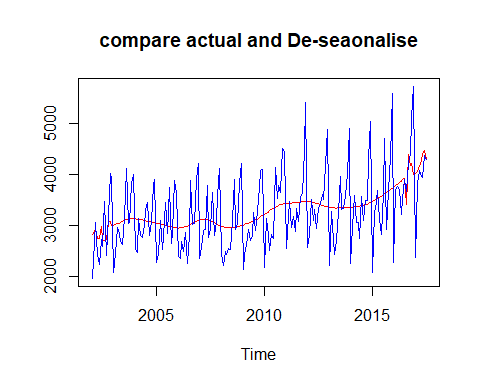
##check for smaller value of window  
 itemA\_Dec3 = stl(Sales\_data\_ts\_A,s.window = 3)  
 plot(itemA\_Dec3)



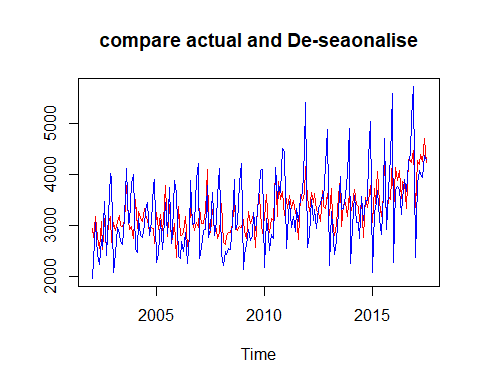
# Deseasonal is sum of trend and remainder .  
 DeseasonRev = (itemA\_Dec7$time.series[,2]+itemA\_Dec7$time.series[,3])  
 ts.plot(DeseasonRev,Sales\_data\_ts\_A, col=c("red", "blue"), main= "compare actual and De-seaonalise")



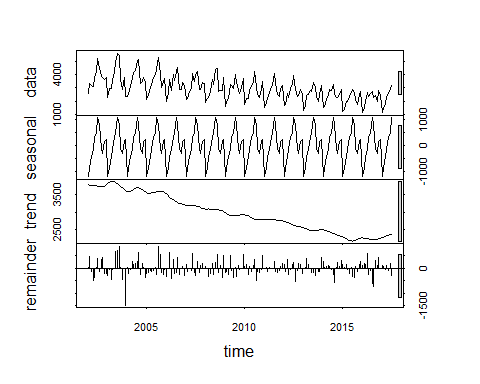
DeseasonRev3 = (itemA\_Dec3$time.series[,2]+itemA\_Dec3$time.series[,3])  
 ts.plot(DeseasonRev3,Sales\_data\_ts\_A, col=c("red", "blue"), main= "compare actual and De-seaonalise")



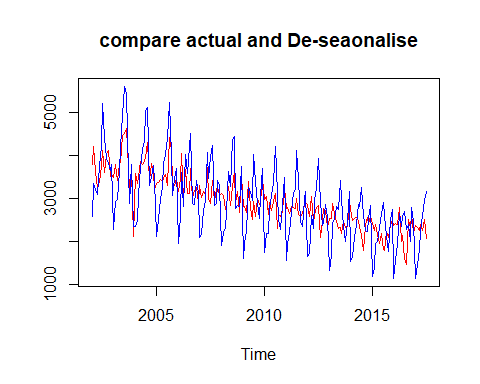
DeseasonRev\_Const = (itemA\_Dec$time.series[,2]+itemA\_Dec$time.series[,3])  
 ts.plot(DeseasonRev\_Const,Sales\_data\_ts\_A, col=c("red", "blue"), main= "compare actual and De-seaonalise")



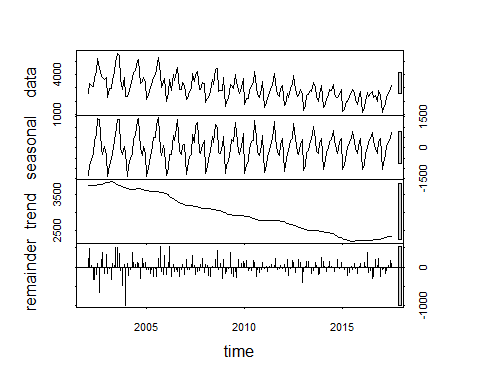
##item B  
itemB\_Dec = stl(Sales\_data\_ts\_B,s.window = "p")  
plot(itemB\_Dec)



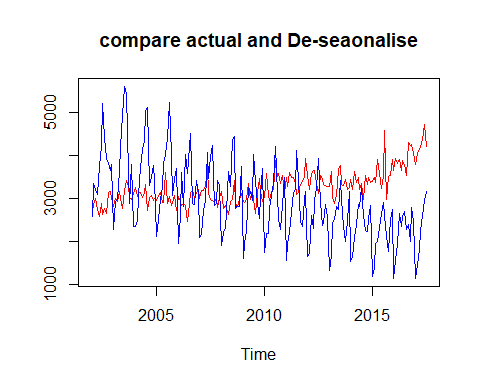
DeseasonRevB\_Const = (itemB\_Dec$time.series[,2]+itemB\_Dec$time.series[,3])  
 ts.plot(DeseasonRevB\_Const,Sales\_data\_ts\_B, col=c("red", "blue"), main= "compare actual and De-seaonalise")



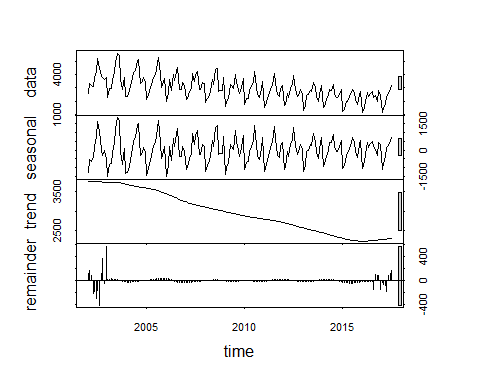
#assume seasonlaity is not constant   
  
 itemB\_Dec7 = stl(Sales\_data\_ts\_B,s.window = 7)  
 plot(itemB\_Dec7)



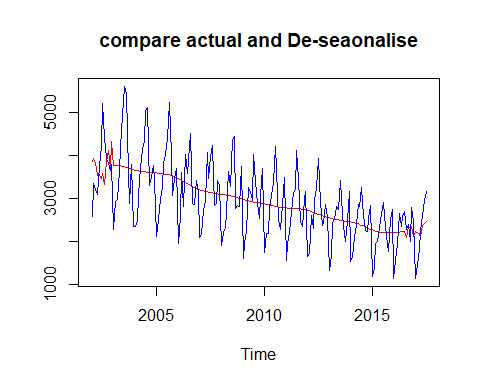
DeseasonRev1 = (itemB\_Dec7$time.series[,2]+itemB\_Dec7$time.series[,3])  
 ts.plot(DeseasonRev,Sales\_data\_ts\_B, col=c("red", "blue"), main= "compare actual and De-seaonalise")



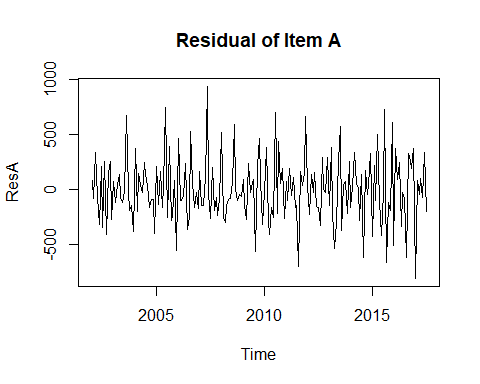
itemB\_Dec3 = stl(Sales\_data\_ts\_B,s.window = 3)  
 plot(itemB\_Dec3)



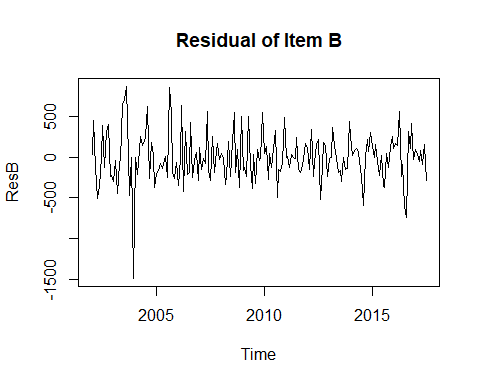
DeseasonRev1B = (itemB\_Dec3$time.series[,2]+itemB\_Dec3$time.series[,3])  
 ts.plot(DeseasonRev1B,Sales\_data\_ts\_B, col=c("red", "blue"), main= "compare actual and De-seaonalise")



### assume multilicative model   
   
 #Sales\_data\_ts\_A\_log = log(Sales\_data\_ts\_A)  
 #itemA\_Dec\_M = stl(Sales\_data\_ts\_A\_log,s.window = "p")  
 #plot(itemA\_Dec\_M)  
 #itemA\_Dec\_M$time.series[1:12,1]  
 #itemA\_Dec\_M\_exp = exp(itemA\_Dec\_M$time.series[1:12,1])  
 #plot(itemA\_Dec\_M\_exp,type= 'l')  
   
   
 ##################################################################  
 #####Residuals ##############  
 ##################################################################  
ResA = itemA\_Dec$time.series[,3]  
ResB = itemB\_Dec$time.series[,3]  
plot(ResA,main = "Residual of Item A")



plot(ResB,main = "Residual of Item B")



#Sales\_data\_CompB = decompose(Sales\_data\_ts\_B)  
#plot(Sales\_data\_CompB)  
  
#check for stationarity####  
library(tseries)

## Warning: package 'tseries' was built under R version 3.5.3

adf.test(Sales\_data\_ts\_A)

## Warning in adf.test(Sales\_data\_ts\_A): p-value smaller than printed p-value

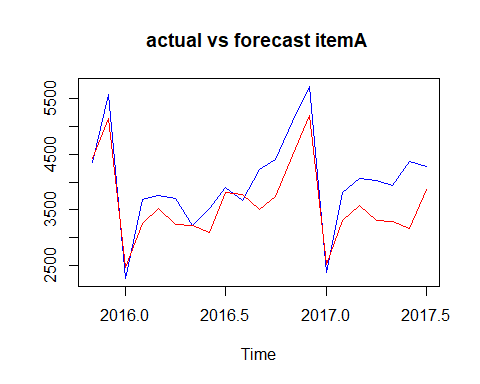
##   
## Augmented Dickey-Fuller Test  
##   
## data: Sales\_data\_ts\_A  
## Dickey-Fuller = -7.8632, Lag order = 5, p-value = 0.01  
## alternative hypothesis: stationary

adf.test(Sales\_data\_ts\_B)

## Warning in adf.test(Sales\_data\_ts\_B): p-value smaller than printed p-value

##   
## Augmented Dickey-Fuller Test  
##   
## data: Sales\_data\_ts\_B  
## Dickey-Fuller = -12.967, Lag order = 5, p-value = 0.01  
## alternative hypothesis: stationary

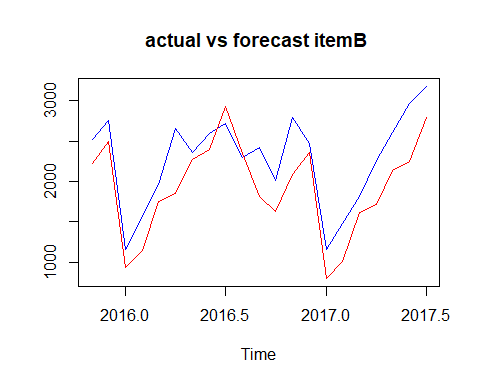
######forecast MOdel ##########  
  
  
# naive decomposition method ##############  
  
##item A #########  
  
#Divide the data into test and hold out sample   
itemA\_Dec\_T <- window(Sales\_data\_ts\_A, start=c(2002,1), end=c(2015,10))  
itemA\_Dec\_HO <- window(Sales\_data\_ts\_A, start=c(2015,11), end=c(2017,7))  
  
#Decompose data itemA  
itemA\_DecF=stl(itemA\_Dec\_T,s.window = 7)  
  
itemA\_Dec\_for = forecast(itemA\_DecF, method="rwdrift", h=21)  
vec1 = cbind(itemA\_Dec\_HO,itemA\_Dec\_for$mean)  
ts.plot(vec1,col=c("blue", "red"), main="actual vs forecast itemA ")



MAPE= mean(abs(vec1[,1]-vec1[,2])/vec1[,1])  
MAPE ##10%

## [1] 0.105762

#item B  
  
#Divide the data into test and hold out sample   
itemB\_Dec\_T <- window(Sales\_data\_ts\_B, start=c(2002,1), end=c(2015,10))  
itemB\_Dec\_HO <- window(Sales\_data\_ts\_B, start=c(2015,11), end=c(2017,7))  
  
#Decompose data itemB  
itemB\_DecF=stl(itemB\_Dec\_T,s.window = 7)  
  
#library("forecast")  
  
itemB\_Dec\_for = forecast(itemB\_DecF, method="rwdrift", h=21)  
vec1 = cbind(itemB\_Dec\_HO,itemB\_Dec\_for$mean)  
ts.plot(vec1,col=c("blue", "red"), main="actual vs forecast itemB ")



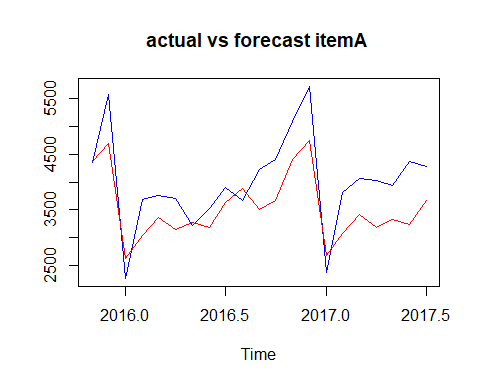
MAPE= mean(abs(vec1[,1]-vec1[,2])/vec1[,1])  
MAPE ##17%

## [1] 0.1694894

#item A  
  
##holt-winter's method   
#divide the item A data into test and hold out sample   
  
itemA\_T <- window(Sales\_data\_ts\_A, start=c(2002,1), end=c(2015,10))  
itemA\_HO <- window(Sales\_data\_ts\_A, start=c(2015,11), end=c(2017,7))  
  
itemA\_FC1= hw(itemA\_T,h=21)  
itemA\_FC1$model

## Holt-Winters' additive method   
##   
## Call:  
## hw(y = itemA\_T, h = 21)   
##   
## Smoothing parameters:  
## alpha = 0.0587   
## beta = 1e-04   
## gamma = 0.0064   
##   
## Initial states:  
## l = 2956.0292   
## b = 3.725   
## s = 1181.297 856.9921 115.8834 -41.8724 330.7963 88.3319  
## -350.0979 -253.7954 -386.6735 -159.4867 -487.1291 -894.2462  
##   
## sigma: 330.8995  
##   
## AIC AICc BIC   
## 2791.968 2796.103 2844.872

vec2 = cbind(itemA\_HO,itemA\_FC1$mean)  
ts.plot(vec2,col=c("blue", "red"), main="actual vs forecast itemA ")



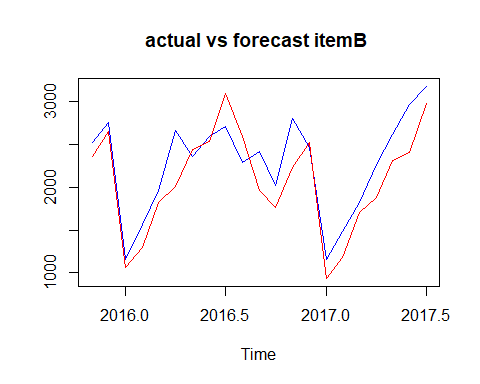
MAPE= mean(abs(vec2[,1]-vec2[,2])/vec2[,1])  
MAPE

## [1] 0.1374336

#forecast for future   
#plot(itemA\_FC1)  
  
  
# item B  
##holt-winter's method   
  
#divide the item B data into test and hold out sample   
  
itemB\_T <- window(Sales\_data\_ts\_B, start=c(2002,1), end=c(2015,10))  
itemB\_HO <- window(Sales\_data\_ts\_B, start=c(2015,11), end=c(2017,7))  
  
#applying holtes-winter method as item B sales has both seasonality and trend .  
  
itemB\_FC1= hw(itemB\_T,h=21)  
itemB\_FC1$model

## Holt-Winters' additive method   
##   
## Call:  
## hw(y = itemB\_T, h = 21)   
##   
## Smoothing parameters:  
## alpha = 1e-04   
## beta = 1e-04   
## gamma = 0.2681   
##   
## Initial states:  
## l = 3946.4018   
## b = -10.3181   
## s = 205.4364 96.3795 -328.4512 -144.3679 894.7028 1189.823  
## 442.1344 426.9279 -161.0965 -445.9498 -914.2891 -1261.249  
##   
## sigma: 334.1357  
##   
## AIC AICc BIC   
## 2795.199 2799.334 2848.103

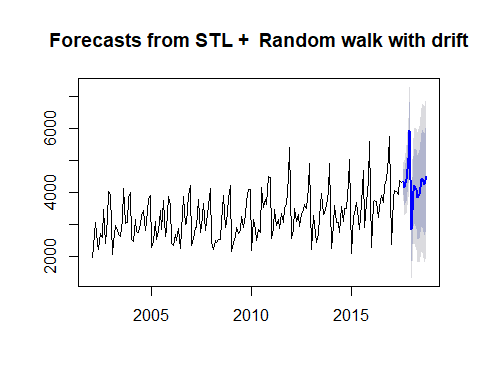
vec2 = cbind(itemB\_HO,itemB\_FC1$mean)  
ts.plot(vec2,col=c("blue", "red"), main="actual vs forecast itemB ")



MAPE= mean(abs(vec2[,1]-vec2[,2])/vec2[,1])  
MAPE

## [1] 0.1211512

##Forecast demand of item A and item B for oct 2017 to Dec 2018.  
  
#item A  
  
itemA\_FC\_Future =stl(Sales\_data\_ts\_A,s.window = 7)  
  
itemA\_Dec\_for = forecast(itemA\_FC\_Future, method="rwdrift", h=15)  
plot(itemA\_Dec\_for)



itemA\_Dec\_for$upper

## 80% 95%  
## Aug 2017 4732.646 4942.897  
## Sep 2017 4704.389 5003.315  
## Oct 2017 5099.361 5467.400  
## Nov 2017 5875.152 6302.347  
## Dec 2017 6823.557 7303.643  
## Jan 2018 3836.522 4365.119  
## Feb 2018 5109.155 5682.997  
## Mar 2018 5360.356 5976.894  
## Apr 2018 5340.544 5997.727  
## May 2018 5160.675 5856.811  
## Jun 2018 5315.517 6049.182  
## Jul 2018 5841.737 6611.719  
## Aug 2018 5953.830 6759.078  
## Sep 2018 5822.948 6662.545  
## Oct 2018 6150.717 7023.852

itemA\_Dec\_for$lower

## 80% 95%  
## Aug 2017 3938.300 3728.050  
## Sep 2017 3575.023 3276.098  
## Oct 2017 3708.876 3340.837  
## Nov 2017 4261.174 3833.980  
## Dec 2017 5009.750 4529.664  
## Jan 2018 1839.431 1310.833  
## Feb 2018 2941.129 2367.287  
## Mar 2018 3031.021 2414.483  
## Apr 2018 2857.650 2200.468  
## May 2018 2530.614 1834.479  
## Jun 2018 2543.662 1809.996  
## Jul 2018 2932.677 2162.695  
## Aug 2018 2911.531 2106.283  
## Sep 2018 2650.878 1811.282  
## Oct 2018 2851.933 1978.798

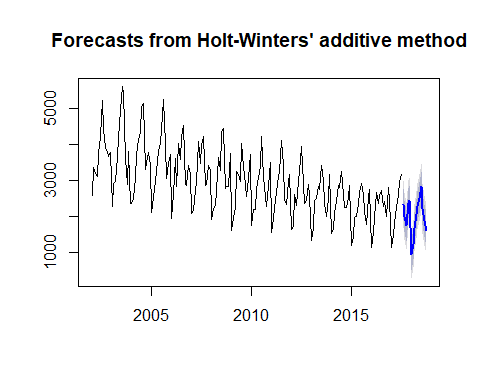
#itmem B  
  
#install.packages("forecast")  
library(forecast)  
itemB\_FC\_Future = hw(Sales\_data\_ts\_B,start=c(2017,10),15)  
itemB\_FC\_Future$upper

## 80% 95%  
## Aug 2017 2772.401 2997.662  
## Sep 2017 2442.400 2667.661  
## Oct 2017 2170.111 2395.372  
## Nov 2017 2791.558 3016.819  
## Dec 2017 2862.313 3087.574  
## Jan 2018 1362.684 1587.945  
## Feb 2018 1665.257 1890.518  
## Mar 2018 2104.628 2329.889  
## Apr 2018 2482.040 2707.301  
## May 2018 2707.459 2932.720  
## Jun 2018 2936.739 3162.000  
## Jul 2018 3257.552 3482.813  
## Aug 2018 2681.414 2919.933  
## Sep 2018 2351.414 2589.934  
## Oct 2018 2079.125 2317.645

#itemb\_FC\_F = forecast::hw(Sales\_data\_ts\_B,15)  
head(itemB\_FC\_Future)

## $model  
## Holt-Winters' additive method   
##   
## Call:  
## hw(y = Sales\_data\_ts\_B, h = 15, start = c(2017, 10))   
##   
## Smoothing parameters:  
## alpha = 1e-04   
## beta = 1e-04   
## gamma = 0.3467   
##   
## Initial states:  
## l = 3954.4932   
## b = -9.6422   
## s = 228.2649 145.7592 -326.3423 -114.366 819.3452 1111.888  
## 440.7485 383.7293 -116.5732 -428.0651 -890.4834 -1253.905  
##   
## sigma: 332.0414  
##   
## AIC AICc BIC   
## 3166.658 3170.279 3221.587   
##   
## $mean  
## Jan Feb Mar Apr May Jun Jul  
## 2017   
## 2018 937.1552 1239.7282 1679.0992 2056.5112 2281.9294 2511.2094 2832.0218  
## Aug Sep Oct Nov Dec  
## 2017 2346.8725 2016.8723 1744.5830 2366.0301 2436.7843  
## 2018 2230.8402 1900.8400 1628.5507   
##   
## $level  
## [1] 80 95  
##   
## $x  
## Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec  
## 2002 2585 3368 3210 3111 3756 4216 5225 4426 3932 3816 3661 3795  
## 2003 2285 2934 2985 3646 4198 4935 5618 5454 3624 2898 3802 2369  
## 2004 2369 2511 3079 3728 4151 4326 5054 5138 3310 3508 3790 3446  
## 2005 2127 2523 3017 3265 3822 4027 4420 5255 4009 3074 3465 3718  
## 2006 1954 2604 3626 2836 4042 3584 4225 4523 2892 2876 3420 3159  
## 2007 2101 2181 2724 2954 4092 3470 3990 4239 2855 2897 3433 3307  
## 2008 1914 2214 2320 2714 3633 3295 4377 4442 2774 2840 2828 3758  
## 2009 1610 1968 2248 3262 3164 2972 4041 3402 2898 2555 3056 3717  
## 2010 1755 2193 2198 2777 3076 3389 4231 3118 2524 2280 2862 3502  
## 2011 1558 1940 2226 2676 3145 3224 4117 3446 2482 2349 2986 3163  
## 2012 1651 1725 2622 2316 2976 3263 3951 2917 2380 2458 2883 2579  
## 2013 1330 1686 2457 2514 2834 2757 3425 3006 2369 2017 2507 3168  
## 2014 1545 1643 2112 2415 2862 2822 3260 2606 2264 2250 2545 2856  
## 2015 1208 1412 1964 2018 2329 2660 2923 2626 2132 1772 2526 2755  
## 2016 1154 1568 1965 2659 2354 2592 2714 2294 2416 2016 2799 2467  
## 2017 1153 1482 1818 2262 2612 2967 3179   
##   
## $upper  
## 80% 95%  
## Aug 2017 2772.401 2997.662  
## Sep 2017 2442.400 2667.661  
## Oct 2017 2170.111 2395.372  
## Nov 2017 2791.558 3016.819  
## Dec 2017 2862.313 3087.574  
## Jan 2018 1362.684 1587.945  
## Feb 2018 1665.257 1890.518  
## Mar 2018 2104.628 2329.889  
## Apr 2018 2482.040 2707.301  
## May 2018 2707.459 2932.720  
## Jun 2018 2936.739 3162.000  
## Jul 2018 3257.552 3482.813  
## Aug 2018 2681.414 2919.933  
## Sep 2018 2351.414 2589.934  
## Oct 2018 2079.125 2317.645  
##   
## $lower  
## 80% 95%  
## Aug 2017 1921.3444 1696.0834  
## Sep 2017 1591.3442 1366.0832  
## Oct 2017 1319.0548 1093.7938  
## Nov 2017 1940.5019 1715.2409  
## Dec 2017 2011.2561 1785.9950  
## Jan 2018 511.6268 286.3657  
## Feb 2018 814.1997 588.9385  
## Mar 2018 1253.5706 1028.3093  
## Apr 2018 1630.9823 1405.7210  
## May 2018 1856.4004 1631.1389  
## Jun 2018 2085.6800 1860.4183  
## Jul 2018 2406.4921 2181.2302  
## Aug 2018 1780.2665 1541.7471  
## Sep 2018 1450.2658 1211.7462  
## Oct 2018 1177.9759 939.4561

plot(itemB\_FC\_Future)



## Including Plots

You can also embed plots, for example:



Note that the echo = FALSE parameter was added to the code chunk to prevent printing of the R code that generated the plot. ..