PES University, Bangalore

Established under Karnataka Act No. 16 of 2013

UE20CS312 - Data Analytics - Worksheet 3a - Basic Forecasting Techniques

Designed by Vishruth Veerendranath, Dept. of CSE - vishruth@pesu.pes.edu

NAME:Adarsh Subhas Nayak SRN: PES1UG20CS620 SEC: 'K' DATE: 30/9/22

Time Series Data and Basic Forecasting Techniques

Time Series data is any data that is collected at regular time intervals, with changing observations at every time interval. Processing time series data effectively can help gain meaningful insights into how a quantity changes with time.

Forecasting a quantity into the future is an essential task, that predicts future values at any particular time. Forecasts can be made using various techniques like Exponential Smoothing to much more complex techniques such as Recurrent Neural Networks. Let's try to process time-series data and forecast values using basic techniques!

Task

Let's imagine it is 2012 and you are in the market to buy an Orange Ultrabook Laptop for college. But this laptop is rare to find and expensive. You would want to put your Data Analytics skills to use, and predict the best time to buy your laptop, such that you can get it for the best price! You would also like to predict when the Orange Ultrabook would be in stock and when it would have high demand.

An electronics store collected sales data for their store weekly, from *February 2010* to *October 2012*, a period of 143 months. You have gotten your hands on this, and will use it to predict how the prices will change in the future.

Data Dictionary

```
Date - Date on which data was collected (end of the week)
Sales - Weekly sales of the store (in $)
Holiday_Flag - Boolean Flag. 0 for normal week and 1 for holiday season
Temperature - Average temperature during the week
Fuel_Price - Average price during the week (in $/gallon)
CPI - Consumer Price Index
Unemployment - Average percentage of Unemployment in the city
Laptop_Demand - Number of Orange Ultrabook laptops sold during the week
```

Data Ingestion and Preprocessing

- Read the file into a `data.frame` object

```
#reading the csv file.
df <- read.csv('sales.csv')
head(df)
```

```
##   X       Date   Sales Holiday_Flag Temperature Fuel_Price      CPI
## 1 0 05-02-2010 2135144            0       43.76      2.598 126.4421
## 2 1 12-02-2010 2188307            1       28.84      2.573 126.4963
## 3 2 19-02-2010 2049860            0       36.45      2.540 126.5263
## 4 3 26-02-2010 1925729            0       41.36      2.590 126.5523
## 5 4 05-03-2010 1971057            0       43.49      2.654 126.5783
## 6 5 12-03-2010 1894324            0       49.63      2.704 126.6043
##   Unemployment Laptop_Demand
## 1        8.623             0
## 2        8.623             0
## 3        8.623             0
## 4        8.623             0
## 5        8.623             1
## 6        8.623             0
```

```
#print(df)
```

- Pick out the `Sales` column in the `data.frame`.

```
sales <- df$Sales
head(sales)
```

```
## [1] 2135144 2188307 2049860 1925729 1971057 1894324
```

- The `ts` function is used to create the `ts` object in R. Frequency is 52 as it is weekly data. The start is specified like `start= c(y, m, d)` as we are dealing with weekly data. If it was monthly data we can omit the `d` and for yearly data we can omit the `m` as well.(`c` is the combine function in R)

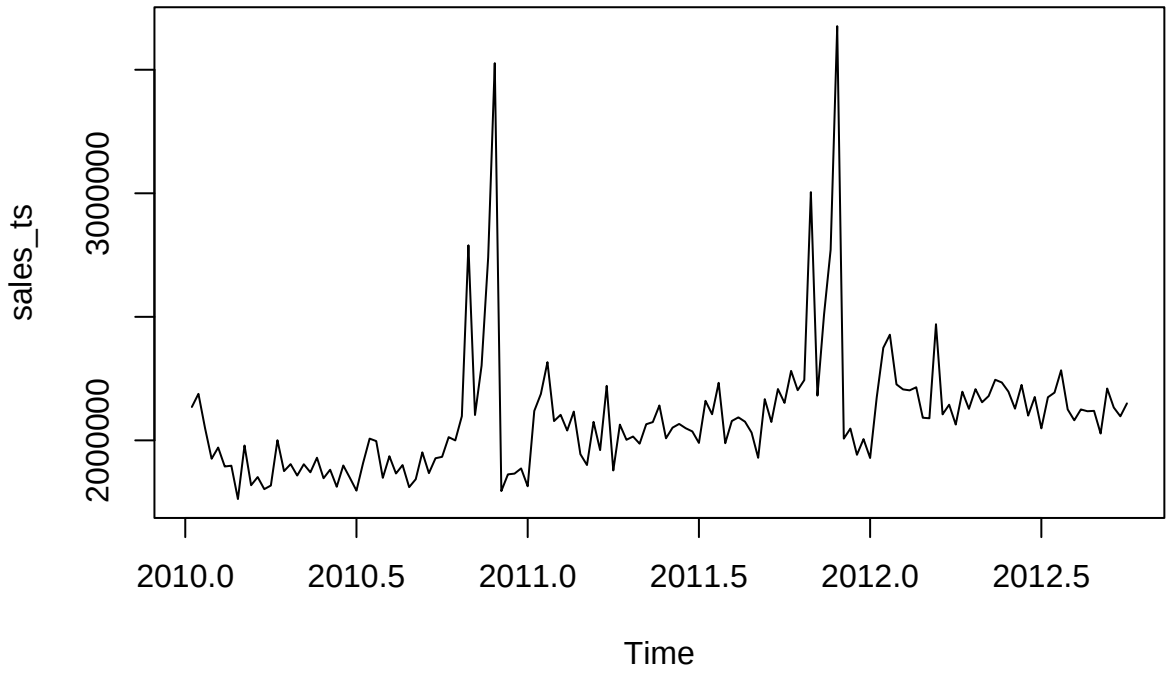
```
sales_ts <- ts(sales, frequency = 52, start=c(2010, 2, 5))
sales_ts
```

```
## Time Series:
## Start = c(2010, 2)
## End = c(2012, 40)
## Frequency = 52
##   [1] 2135144 2188307 2049860 1925729 1971057 1894324 1897429 1762539 1979247
##  [10] 1818453 1851520 1802678 1817273 2000626 1875597 1903753 1857534 1903291
##  [19] 1870619 1929736 1846652 1881337 1812208 1898428 1848427 1796638 1907639
##  [28] 2007051 1997181 1848404 1935858 1865821 1899960 1810685 1842821 1951495
##  [37] 1867345 1927610 1933333 2013116 1999794 2097809 2789469 2102530 2302505
##  [46] 2740057 3526713 1794869 1862476 1865502 1886394 1814241 2119086 2187847
##  [55] 2316496 2078095 2103456 2039818 2116475 1944164 1900246 2074953 1960588
##  [64] 2220601 1878167 2063683 2002362 2015563 1986598 2065377 2073951 2141211
##  [73] 2008345 2051534 2066542 2049047 2036231 1989674 2160057 2105669 2232892
##  [82] 1988490 2078420 2093139 2075577 2031406 1929487 2166738 2074549 2207742
##  [91] 2151660 2281217 2203029 2243947 3004702 2180999 2508955 2771397 3676389
## [100] 2007106 2047766 1941677 2005098 1928721 2173374 2374661 2427640 2226662
```

```
## [109] 2206320 2202451 2214967 2091593 2089382 2470206 2105301 2144337 2064066
## [118] 2196968 2127661 2207215 2154138 2179361 2245257 2234191 2197300 2128363
## [127] 2224499 2100253 2175564 2048614 2174514 2193368 2283540 2125242 2081181
## [136] 2125105 2117855 2119439 2027620 2209835 2133026 2097267 2149594
```

- Visualize the Time-Series of `Sales` column

```
plot.ts(sales_ts)
```



Points

The problems in this worksheet are for a total of 10 points with each problem having a different weightage.

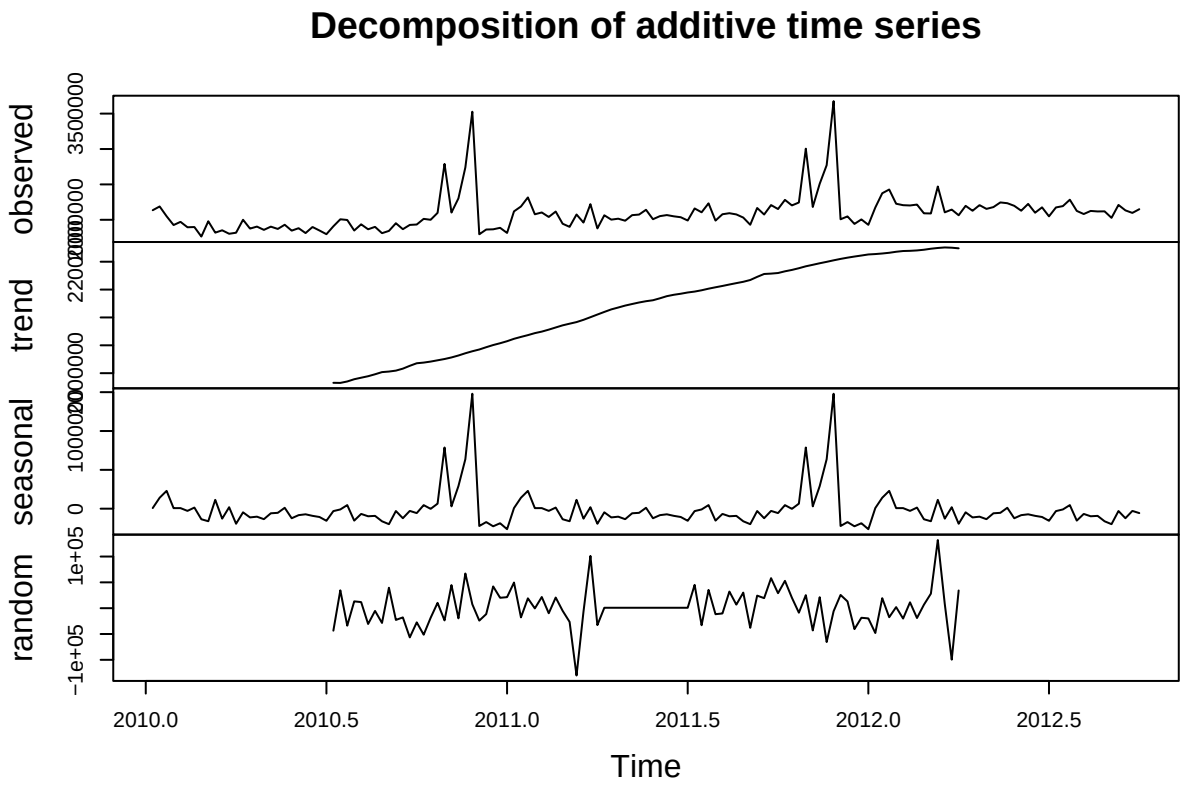
Problem 1 (1 Point)

Decompose the `Sales` column into trend, seasonal and random components. Plot these components as well (Hint: Look at the `decompose` function).

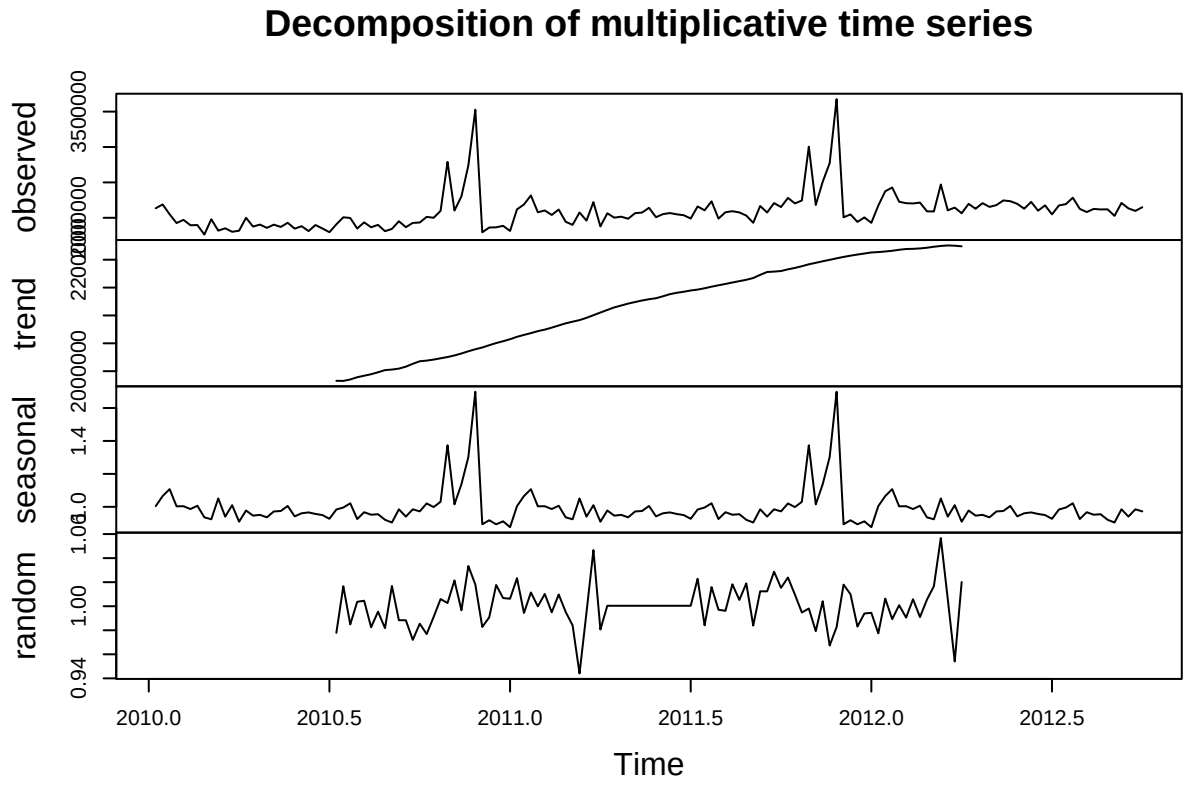
```
#install.packages("forecast")
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
#Decomposing a seasonal time series means separating the time series into these three components
tsA<- decompose(sales_ts,"additive")
plot(tsA)
```



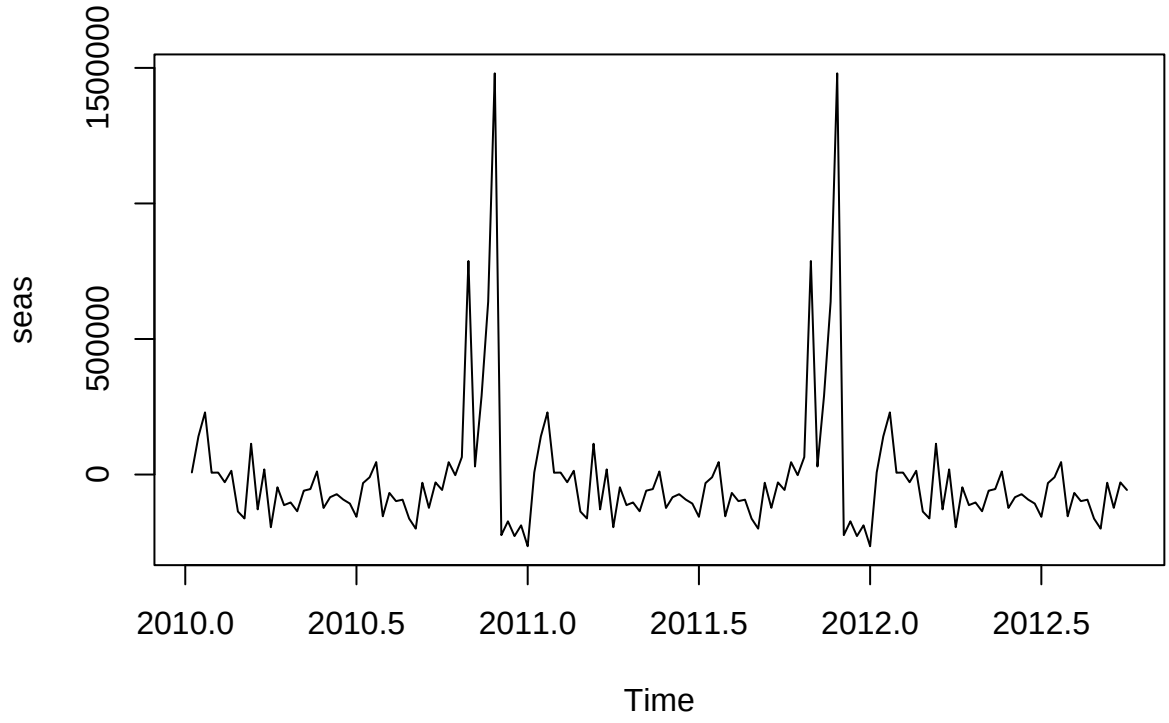

```
library(forecast)
tsM<- decompose(sales_ts,"multiplicative")
plot(tsM)
```



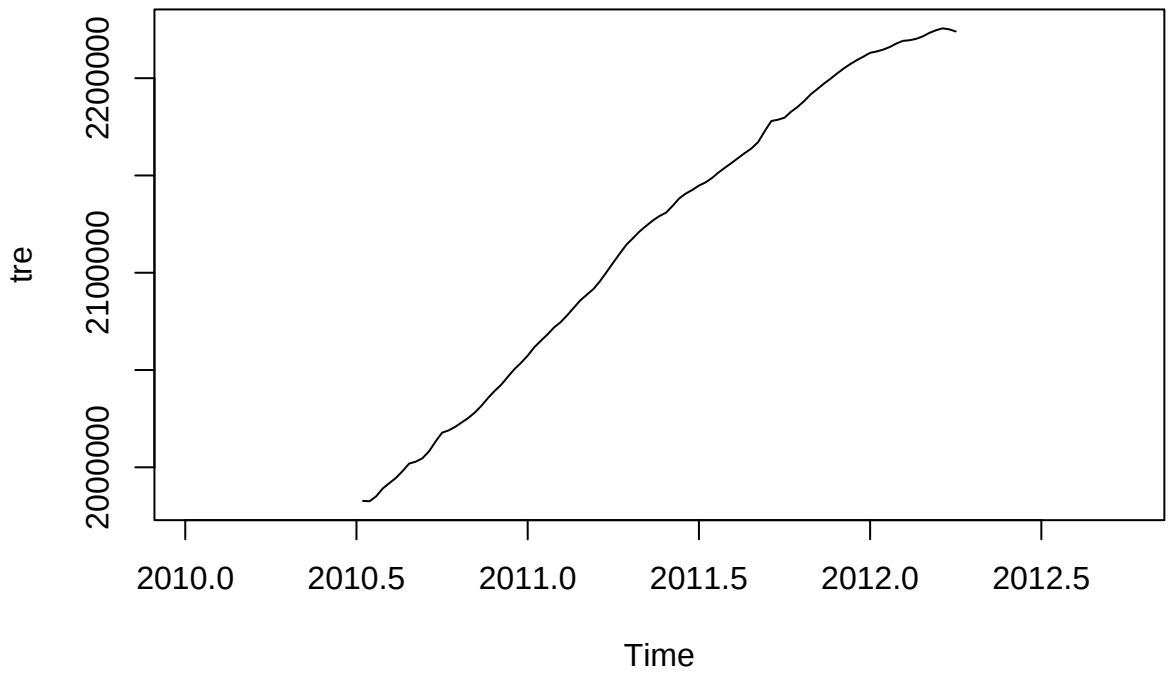
#Observation :

#We choose additive time series over multiplicative time series as the increase in spikes for seasonal component is constant. #Trend is not calculated for first and last year, Periodicity of seasonal component is found to be one year.

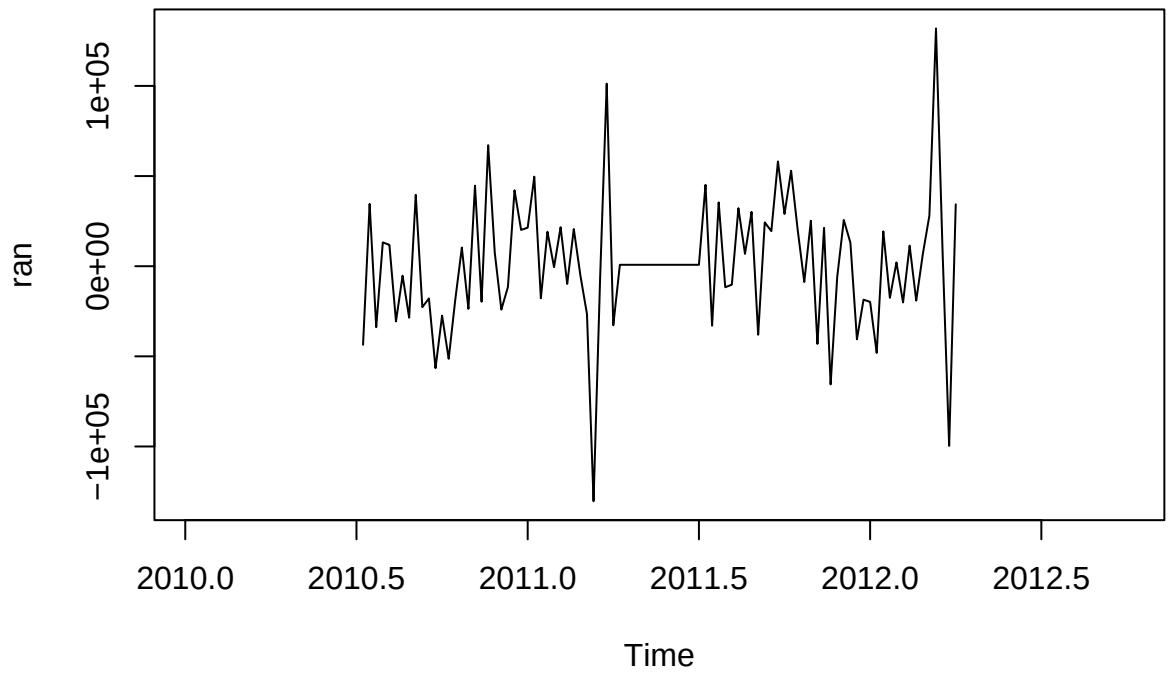
```
seas<- tsA$seasonal
plot(seas)
```




```
tre<-tsA$trend
plot(tre)
```



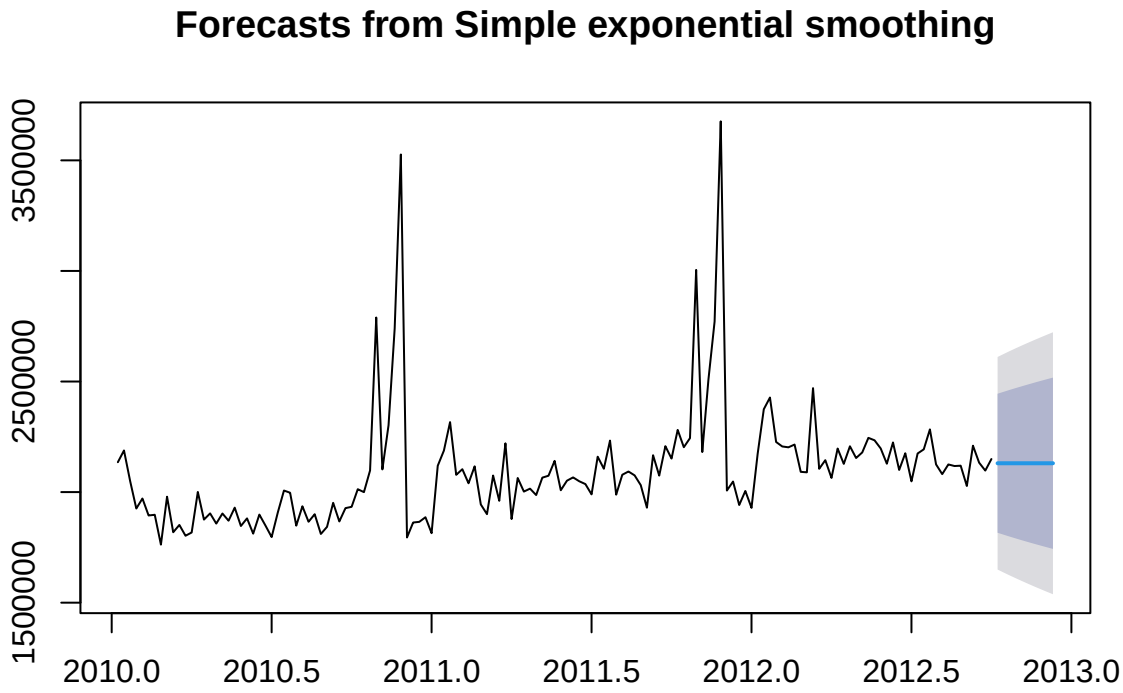

```
ran<-tsA$random
plot(ran)
```



Problem 2 (3 Points)

- Perform forecasts using Single, Double and Triple Exponential Smoothing.
- Plot the forecasts of all three forecasts (different colours) against the true values. (Hint: use `lines`)
- **Use only one function needed for all 3 forecasts**, only changing parameters to get each of the 3 models (Hint: Think about alternate names)

```
#function "ses" gives out the "simple exponential smoothing"
ses<-ses(sales_ts)
#plotting SES
plot(ses)
```



##Simple exponential smoothing does not do well when there is a trend (there will be always bias) , therefore we use double exponential smoothing. ##double exponential smoothing

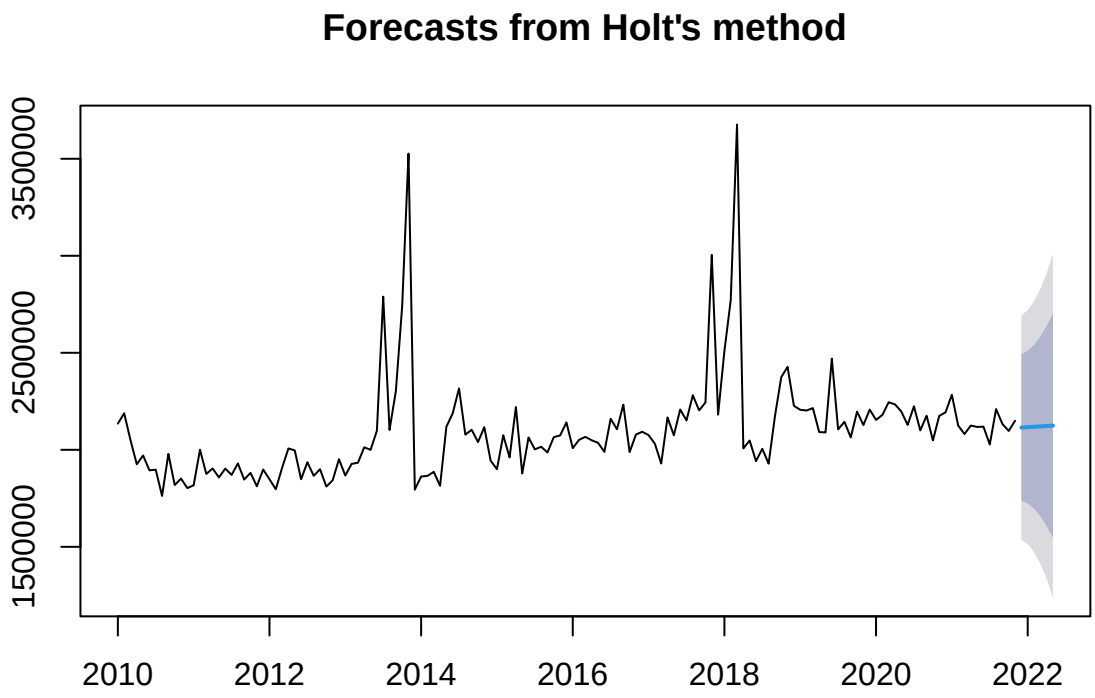
```
#for double exponential smoothing we use HOlt's linear method.
Hts <- ts(sales, frequency = 12, start=2010)
#the alpha -> smoothness parameter for the level (trend-less series), we have a beta parameter which co
HoltModel <- holt(Hts, h=6, alpha=0.2, beta=0.1)

summary(HoltModel)
```

```
##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
## Call:
##  holt(y = Hts, h = 6, alpha = 0.2, beta = 0.1)
##
##   Smoothing parameters:
##     alpha = 0.2
##     beta  = 0.1
##
##   Initial states:
##     l = 2142004.8891
##     b = -57354.9443
```

```
##
##   sigma:  294720.9
##
##      AIC     AICc      BIC
## 4313.452 4313.625 4322.341
##
## Error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE     ACF1
## Training set 4149.404 290569.7 167396.8 -0.6418931 7.757177 0.8326108 0.333307
##
## Forecasts:
##          Point Forecast   Lo 80   Hi 80   Lo 95   Hi 95
## Dec 2021        2114830 1737130 2492530 1537187 2692472
## Jan 2022        2116811 1722481 2511141 1513735 2719887
## Feb 2022        2118793 1696511 2541074 1472969 2764616
## Mar 2022        2120774 1658188 2583360 1413310 2828239
## Apr 2022        2122756 1607642 2637870 1334957 2910555
## May 2022        2124737 1545734 2703740 1239228 3010246
```

```
plot(HoltModel)
```

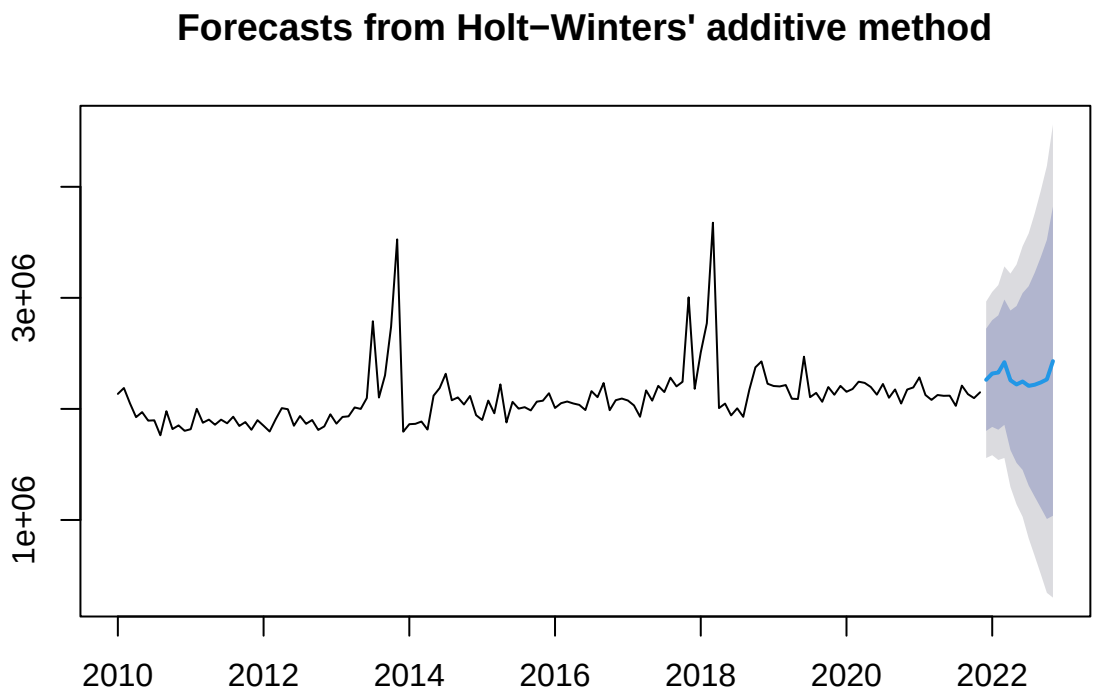


#Inorder to consider seasonal changes as well as trends we use Triple ES. # triple exponential smoothing #lets plot for both additive and multiplicative method.

```r
#
HWModelA <- hw(Hts, h=12, alpha=0.2, beta=0.1, gamma=0.25, seasonal =
"additive")
summary(HWModelA)
```

```
##
## Forecast method: Holt-Winters' additive method
##
## Model Information:
## Holt-Winters' additive method
##
## Call:
##  hw(y = Hts, h = 12, seasonal = "additive", alpha = 0.2, beta = 0.1,
##
##  Call:
##      gamma = 0.25)
##
##   Smoothing parameters:
##     alpha = 0.2
##     beta  = 0.1
##     gamma = 0.25
##
##   Initial states:
##     l = 2231579.9218
##     b = -94825.3161
##     s = -134042.8 90465.38 5880.307 78513.67 38874.26 104318.1
##           30840.1 -2970.316 -30493.82 -822.4503 -26539.91 -154022.5
##
##   sigma:  359295.3
##
##      AIC     AICc      BIC
## 4379.202 4382.483 4420.682
##
## Error measures:
##                     ME     RMSE      MAE        MPE     MAPE     MASE      ACF1
## Training set 7429.997 338598.8 242911.6 -0.4659439 11.43635 1.208212 0.4289343
##
## Forecasts:
##          Point Forecast   Lo 80   Hi 80     Lo 95   Hi 95
## Dec 2021        2262278 1801822 2722733 1558071.7 2966483
## Jan 2022        2318637 1837907 2799366 1583424.2 3053849
## Feb 2022        2327427 1812622 2842232 1540100.7 3114753
## Mar 2022        2420641 1856701 2984582 1558168.9 3283114
## Apr 2022        2257936 1629958 2885913 1297527.1 3218345
## May 2022        2220396 1514531 2926261 1140868.5 3299923
## Jun 2022        2246629 1450427 3042831 1028943.2 3464315
## Jul 2022        2207512 1309919 3105105  834762.4 3580262
## Aug 2022        2217803 1208995 3226610  674964.9 3760640
## Sep 2022        2238446 1109625 3367266  512063.9 3964828
## Oct 2022        2265802 1009003 3522600  343693.6 4187909
## Nov 2022        2430041 1037972 3822110  301055.7 4559027
```

```
plot(HWModelA)
```

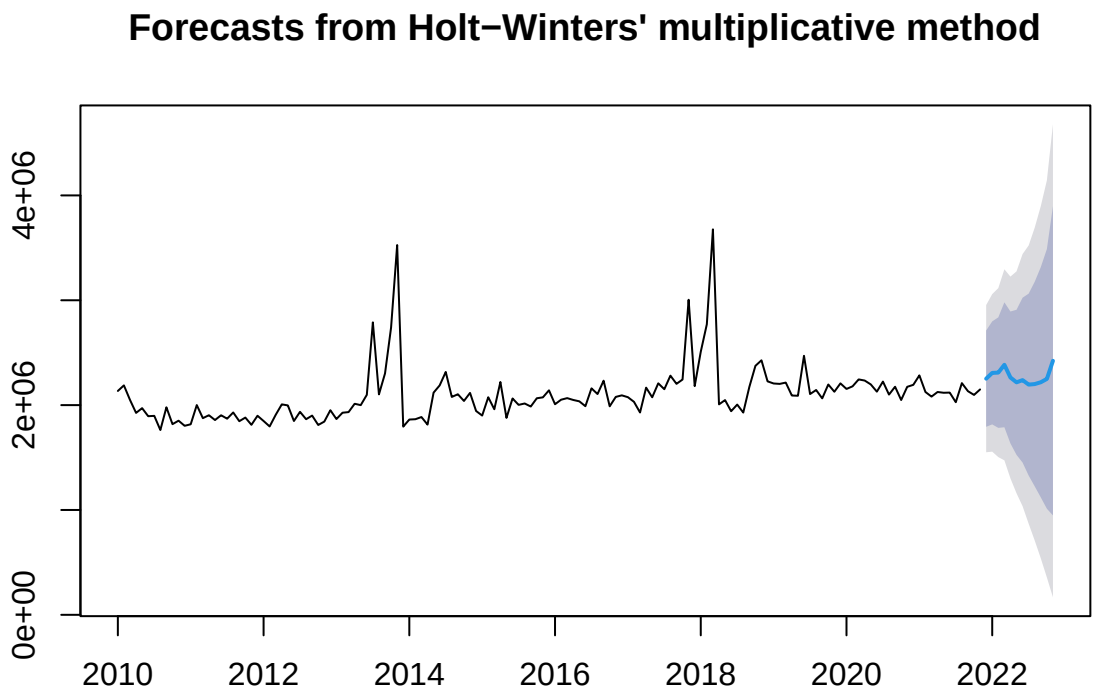



```
HWModelM <- hw(Hts, h=12, alpha=0.2, beta=0.1, gamma=0.25, seasonal =
"multiplicative")
summary(HWModelM)
```

```
##
## Forecast method: Holt-Winters' multiplicative method
##
## Model Information:
## Holt-Winters' multiplicative method
##
## Call:
##  hw(y = Hts, h = 12, seasonal = "multiplicative", alpha = 0.2,
##
##  Call:
##      beta = 0.1, gamma = 0.25)
##
##   Smoothing parameters:
##     alpha = 0.2
##     beta  = 0.1
##     gamma = 0.25
##
##   Initial states:
##     l = 2226020.6955
```

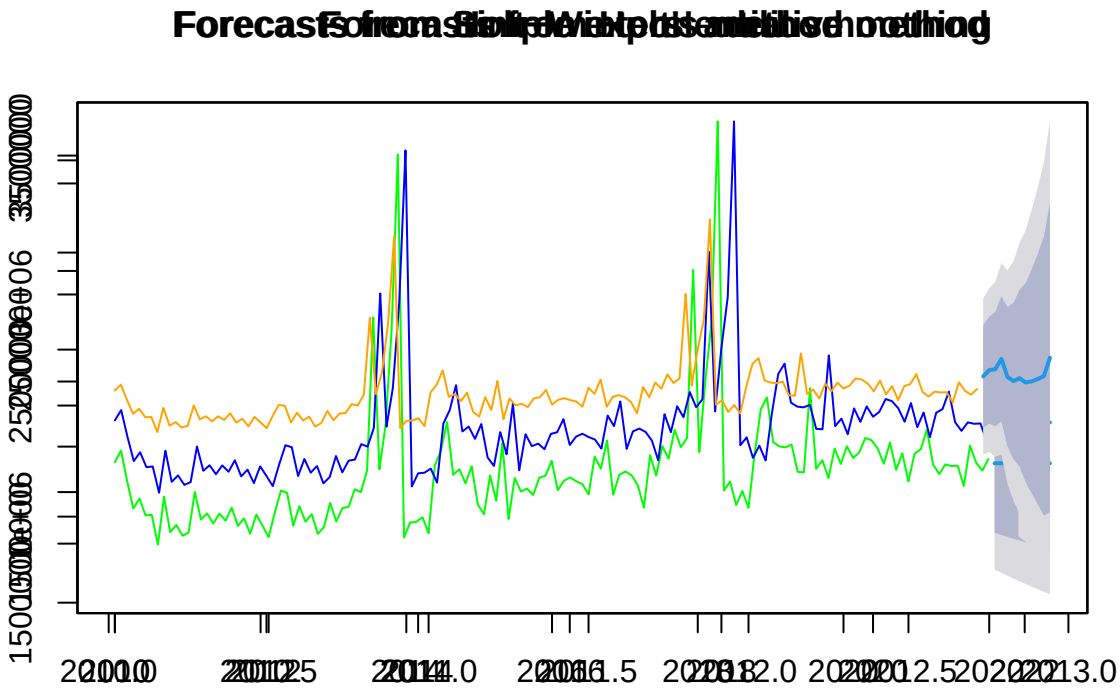
```
##     b = -8819.1106
##     s = 0.7159 1.147 1.0236 1.0133 1.0002 1.0898
##           1.02 1.0111 0.9916 1.0151 1.0144 0.958
##
##   sigma:  0.159
##
##      AIC     AICc      BIC
## 4355.588 4358.870 4397.068
##
## Error measures:
##                       ME     RMSE      MAE       MPE     MAPE     MASE      ACF1
## Training set -6817.147 331420.4 241849.5 -1.135047 11.44502 1.202929 0.4062852
##
## Forecasts:
##          Point Forecast     Lo 80   Hi 80     Lo 95   Hi 95
## Dec 2021        2252164 1793280.0 2711049 1550361.3 2953967
## Jan 2022        2307305 1816164.4 2798445 1556170.6 3058438
## Feb 2022        2310012 1783220.6 2836803 1504354.0 3115670
## Mar 2022        2384443 1788865.6 2980020 1473586.2 3295299
## Apr 2022        2263531 1634468.6 2892592 1301463.4 3225598
## May 2022        2217450 1525716.1 2909183 1159534.4 3275365
## Jun 2022        2238241 1452038.8 3024444 1035848.3 3440635
## Jul 2022        2195615 1327848.2 3063382  868480.2 3522750
## Aug 2022        2200713 1225076.6 3176350  708605.8 3692821
## Sep 2022        2218166 1119881.8 3316450  538485.3 3897847
## Oct 2022        2249983 1011812.3 3488154  356364.2 4143602
## Nov 2022        2422688  948293.2 3897083  167795.8 4677580
```

```
plot(HWModelM)
```



#Plotting all of them in a single graph.

```
#dev.new(width=400, height=250, unit="px")
plot(ses,col="green")
par(new=TRUE)
plot(HoltModel,col="blue")
par(new=TRUE)
plot(HWModelA, col="orange" )
```



Problem 3 (2 Points)

- Forecast `Sales` values by Regression using all other columns. Print summary of regression model.
- Plot the predicted values against original as well. (Hint: Regression model predictions will not be a Time Series, so need to get both to common index/x-axis)
- (Hint: Will not work unless one column is dropped/transformed before including it in the regression. Use the `lm` function to get linear model)

Note: This is Multiple Linear Regression, that is, using all the columns for regression

```
sales_lm <- lm(sales~ Holiday_Flag + Temperature + Fuel_Price + CPI + Unemployment + Laptop_Demand , da
sales_lm%>% summary()
```

```
##
## Call:
## lm(formula = sales ~ Holiday_Flag + Temperature + Fuel_Price +
##     CPI + Unemployment + Laptop_Demand, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -399969  -88853  -26444   41799 1456693
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)   -4640749    6896266  -0.673  0.50213
## Holiday_Flag    104846      80358   1.305  0.19419
## Temperature      -4137       1412  -2.930  0.00398 **
## Fuel_Price     -106728     103421  -1.032  0.30391
## CPI              58100      52430   1.108  0.26976
## Unemployment    -25260      57459  -0.440  0.66091
## Laptop_Demand     3051       4475   0.682  0.49653
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 238800 on 136 degrees of freedom
## Multiple R-squared:  0.2291, Adjusted R-squared:  0.1951
## F-statistic: 6.736 on 6 and 136 DF,  p-value: 2.892e-06
```

```
##obtaining the predicted values.
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
explan_sales<-tibble(df)
head(explan_sales)
```

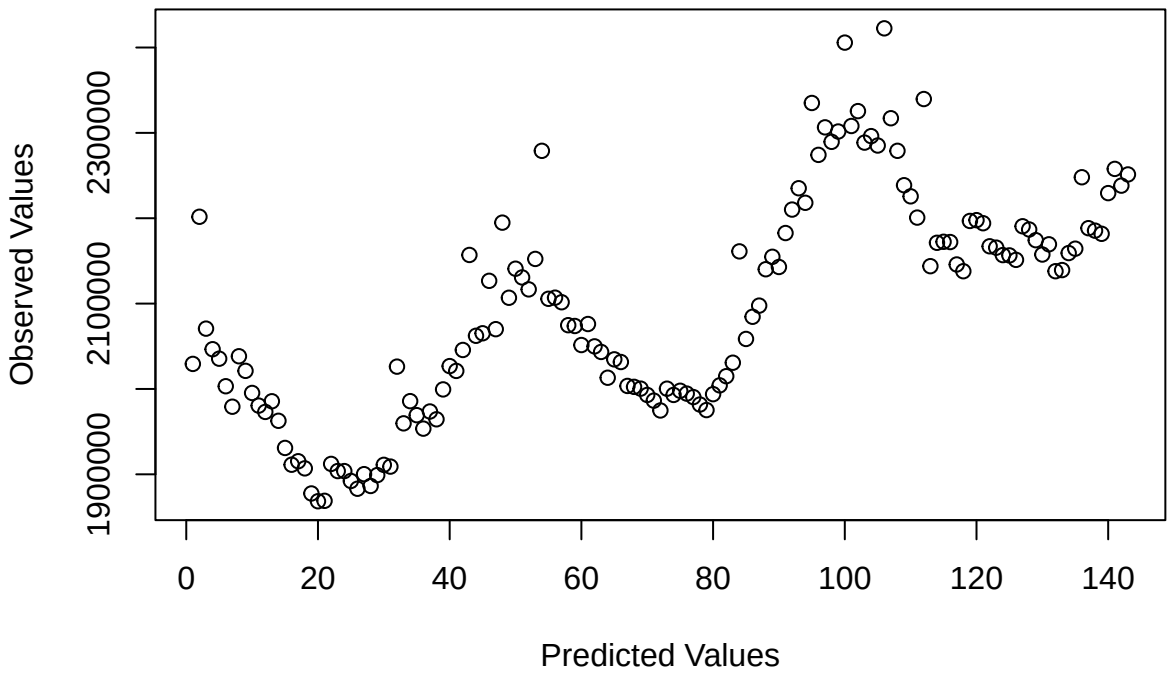
```
## # A tibble: 6 x 9
##       X Date          Sales Holiday_Flag Tempera~1 Fuel_~2   CPI Unemp~3 Lapto~4
##   <int> <chr>         <dbl>        <int>     <dbl>   <dbl> <dbl>   <dbl>   <int>
## 1     0 05-02-2010 2135144.            0      43.8    2.60  126.    8.62       0
## 2     1 12-02-2010 2188307.            1      28.8    2.57  126.    8.62       0
## 3     2 19-02-2010 2049860.            0      36.4    2.54  127.    8.62       0
## 4     3 26-02-2010 1925729.            0      41.4    2.59  127.    8.62       0
## 5     4 05-03-2010 1971057.            0      43.5    2.65  127.    8.62       1
## 6     5 12-03-2010 1894324.            0      49.6    2.70  127.    8.62       0
## # ... with abbreviated variable names 1: Temperature, 2: Fuel_Price,
## #   3: Unemployment, 4: Laptop_Demand
```

```
predict(sales_lm,explan_sales)
```

```
##       1       2       3       4       5       6       7       8       9      10
## 2029339 2201724 2070663 2046525 2035444 2003167 1979191 2038300 2021200 1995288
##      11      12      13      14      15      16      17      18      19      20
## 1980463 1973076 1985568 1962685 1930869 1911242 1915304 1906945 1877541 1868443
##      21      22      23      24      25      26      27      28      29      30
## 1868926 1912179 1903798 1903766 1892207 1883099 1900171 1886323 1899040 1911068
##      31      32      33      34      35      36      37      38      39      40
## 1909040 2026123 1959689 1985700 1969268 1953486 1973592 1964449 1999468 2026775
##      41      42      43      44      45      46      47      48      49      50
```

```
## 2021137 2045652 2156969 2062296 2065204 2126666 2070021 2194838 2106837 2140868
##      51      52      53      54      55      56      57      58      59      60
## 2130628 2116649 2152312 2278981 2105614 2106914 2101571 2074671 2073743 2051427
##      61      62      63      64      65      66      67      68      69      70
## 2076062 2049882 2043349 2013130 2034667 2031535 2003533 2002460 2000353 1992951
##      71      72      73      74      75      76      77      78      79      80
## 1986311 1974780 2000316 1992999 1997870 1994538 1990470 1981750 1975257 1993910
##      81      82      83      84      85      86      87      88      89      90
## 2004195 2014934 2030743 2161166 2058567 2084564 2097621 2140216 2154828 2142869
##      91      92      93      94      95      96      97      98      99     100
## 2182650 2210134 2235133 2218082 2335006 2274212 2306470 2289511 2301596 2405747
##     101     102     103     104     105     106     107     108     109     110
## 2308063 2325482 2288588 2296184 2285195 2422450 2317132 2278955 2238662 2225648
##     111     112     113     114     115     116     117     118     119     120
## 2200740 2339636 2143831 2171218 2172410 2172088 2145837 2137920 2196791 2197729
##     121     122     123     124     125     126     127     128     129     130
## 2194164 2167120 2165520 2156828 2156435 2151235 2190585 2186824 2174292 2157486
##     131     132     133     134     135     136     137     138     139     140
## 2169452 2137910 2139267 2159193 2164308 2248024 2188327 2185438 2181747 2229423
##     141     142     143
## 2257791 2238210 2251303
```

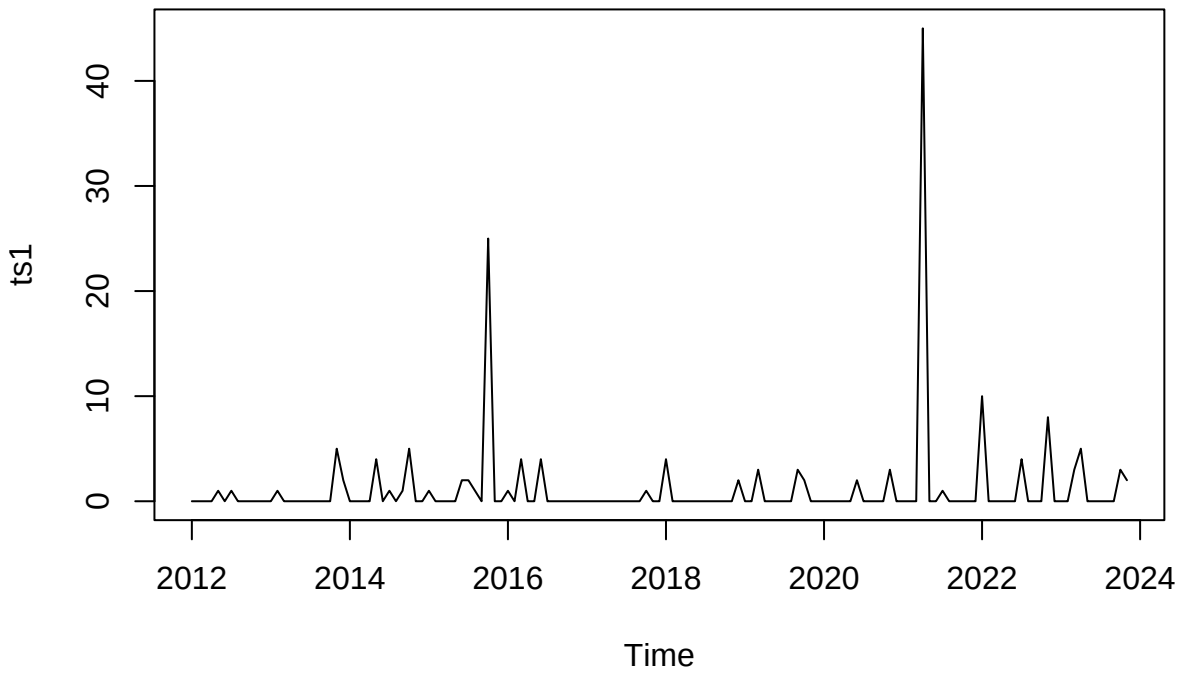
```r
#now let's plot the predicted values against the observed or the original values.
plot(predict(sales_lm), df$sales,
     xlab = "Predicted Values",
     ylab = "Observed Values")
abline(a = 0, b = 1, lwd=2,
       col = "green")
```



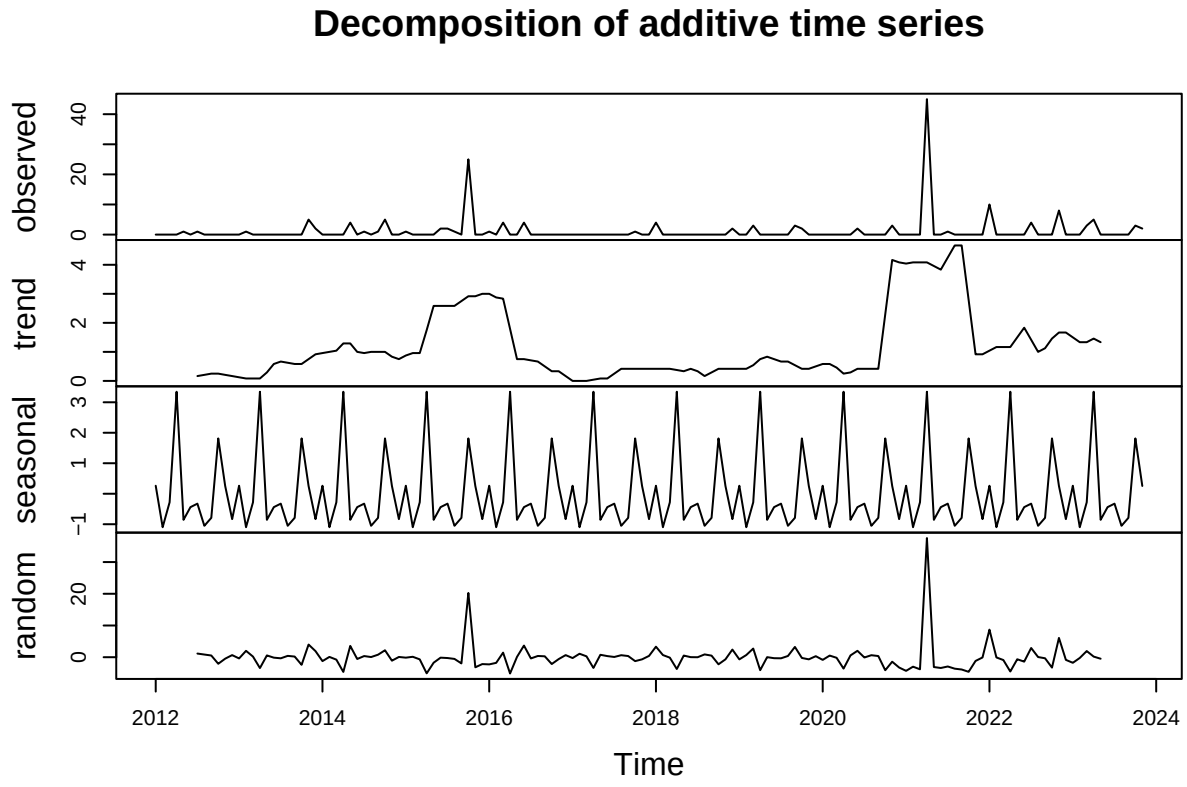
Problem 4 (2 Points)

Plot the `Laptop_Demand` column as a time series. Identify the forecasting required for this type of Time-series, and forecast the values for all 143 weeks (Hint: Look at functions in the `forecast` package)

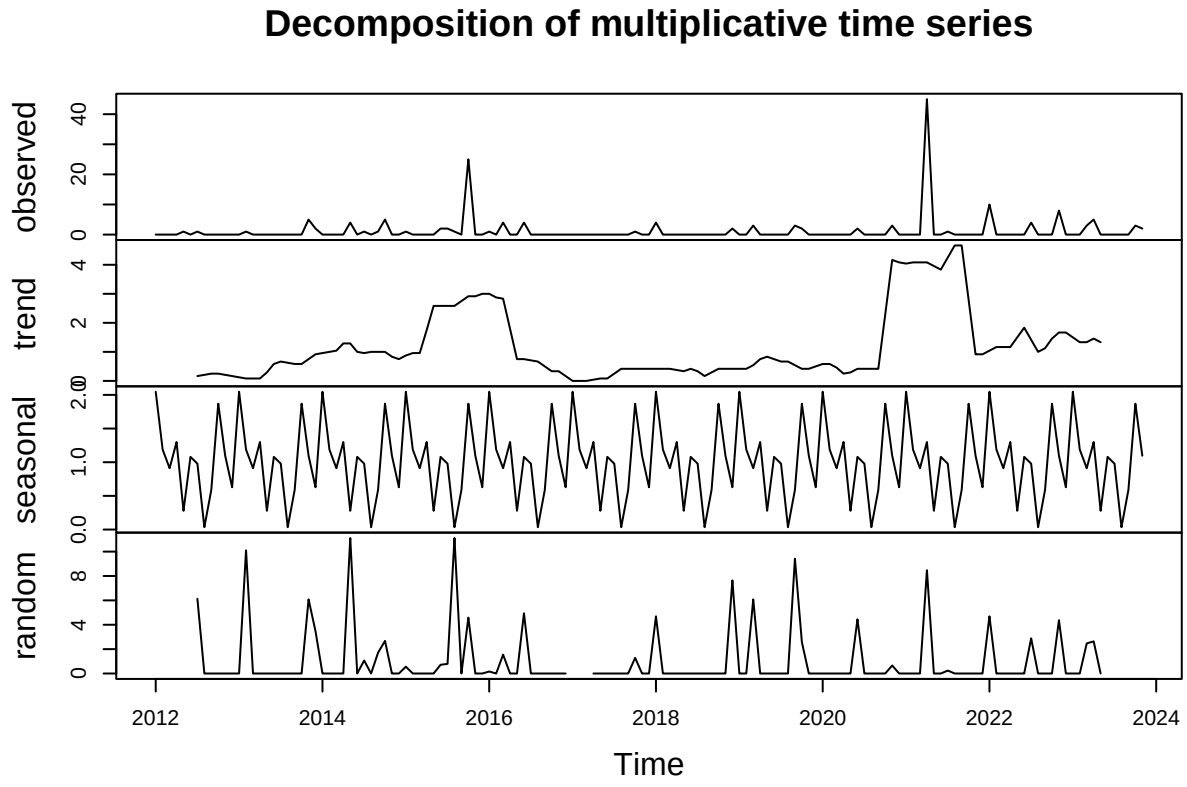
```
ts1 <- ts(df$Laptop_Demand, frequency = 12, start = 2012)
plot(ts1)
```



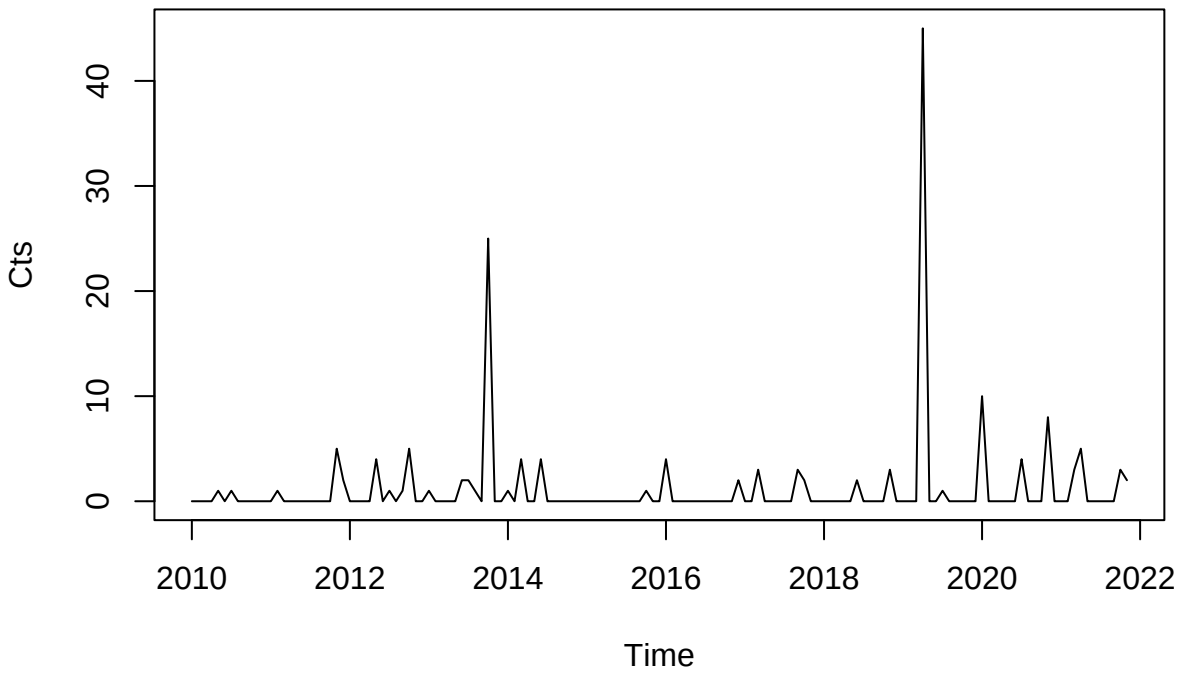

```
tsdA1 <- decompose(ts1, "additive")
plot(tsdA1)
```




```
tsdM1 <- decompose(ts1, "multiplicative")
plot(tsdM1)
```




```
#USING CROSTON MODEL
Cts <- ts(df$Laptop_Demand, frequency = 12, start = 2010)
plot(Cts)
```

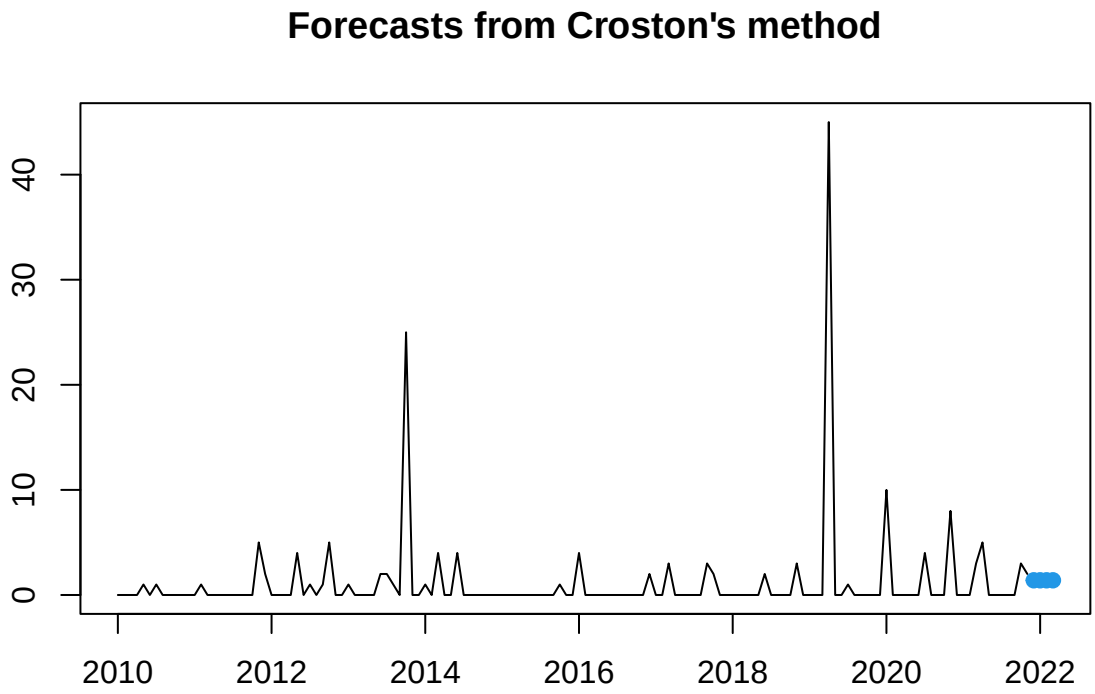



```
CrostonModel <- croston(Cts, h=4, alpha=0.2)
summary(CrostonModel)
```

```
##
## Forecast method: Croston's method
##
## Model Information:
## $demand
##    Point Forecast
## 35       5.214627
## 36       5.214627
## 37       5.214627
## 38       5.214627
##
## $period
##    Point Forecast
## 35       3.731563
## 36       3.731563
## 37       3.731563
## 38       3.731563
##
##
## Error measures:
##                       ME     RMSE      MAE  MPE MAPE      MASE        ACF1
## Training set 0.08649103 4.581462 1.768985 -Inf  Inf 0.8131123 -0.04899203
```

```
##
## Forecasts:
##           Jan      Feb      Mar Apr May Jun Jul Aug Sep Oct Nov      Dec
## 2021                                                          1.397438
## 2022 1.397438 1.397438 1.397438
```

```
plot(CrostonModel)
```



Problem 5 (2 Points)

Evaluate the accuracy of all 3 Exponential Smoothing models (from Problem 2) and Regression models using the MAPE and RMSE metrics. Comment on which is the best Exponential Smoothing method, and if Regression is better than Exponential Smoothing? Provide a reasoning for why the best model is better suited for the `Sales` data (Bonus Point: reasoning for why the 2 other models perform similarly)

```
#lets take a summary of the "Holt Model".
summary(HoltModel)
```

```
##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
```

```
## Call:
##  holt(y = Hts, h = 6, alpha = 0.2, beta = 0.1)
##
##   Smoothing parameters:
##     alpha = 0.2
##     beta  = 0.1
##
##   Initial states:
##     l = 2142004.8891
##     b = -57354.9443
##
##   sigma:  294720.9
##
##      AIC     AICc      BIC
## 4313.452 4313.625 4322.341
##
## Error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE     ACF1
## Training set 4149.404 290569.7 167396.8 -0.6418931 7.757177 0.8326108 0.333307
##
## Forecasts:
##          Point Forecast   Lo 80   Hi 80   Lo 95   Hi 95
## Dec 2021        2114830 1737130 2492530 1537187 2692472
## Jan 2022        2116811 1722481 2511141 1513735 2719887
## Feb 2022        2118793 1696511 2541074 1472969 2764616
## Mar 2022        2120774 1658188 2583360 1413310 2828239
## Apr 2022        2122756 1607642 2637870 1334957 2910555
## May 2022        2124737 1545734 2703740 1239228 3010246
```

```
summary(HWModelA)
```

```
##
## Forecast method: Holt-Winters' additive method
##
## Model Information:
## Holt-Winters' additive method
##
## Call:
##  hw(y = Hts, h = 12, seasonal = "additive", alpha = 0.2, beta = 0.1,
##
##  Call:
##      gamma = 0.25)
##
##   Smoothing parameters:
##     alpha = 0.2
##     beta  = 0.1
##     gamma = 0.25
##
##   Initial states:
##     l = 2231579.9218
##     b = -94825.3161
##     s = -134042.8 90465.38 5880.307 78513.67 38874.26 104318.1
##            30840.1 -2970.316 -30493.82 -822.4503 -26539.91 -154022.5
##
```

```
##   sigma:  359295.3
##
##      AIC     AICc      BIC
## 4379.202 4382.483 4420.682
##
## Error measures:
##                     ME     RMSE      MAE        MPE     MAPE     MASE      ACF1
## Training set 7429.997 338598.8 242911.6 -0.4659439 11.43635 1.208212 0.4289343
##
## Forecasts:
##          Point Forecast   Lo 80   Hi 80     Lo 95   Hi 95
## Dec 2021        2262278 1801822 2722733 1558071.7 2966483
## Jan 2022        2318637 1837907 2799366 1583424.2 3053849
## Feb 2022        2327427 1812622 2842232 1540100.7 3114753
## Mar 2022        2420641 1856701 2984582 1558168.9 3283114
## Apr 2022        2257936 1629958 2885913 1297527.1 3218345
## May 2022        2220396 1514531 2926261 1140868.5 3299923
## Jun 2022        2246629 1450427 3042831 1028943.2 3464315
## Jul 2022        2207512 1309919 3105105  834762.4 3580262
## Aug 2022        2217803 1208995 3226610  674964.9 3760640
## Sep 2022        2238446 1109625 3367266  512063.9 3964828
## Oct 2022        2265802 1009003 3522600  343693.6 4187909
## Nov 2022        2430041 1037972 3822110  301055.7 4559027
```

```
summary(ses)
```

```
##
## Forecast method: Simple exponential smoothing
##
## Model Information:
## Simple exponential smoothing
##
## Call:
##  ses(y = sales_ts)
##
##   Smoothing parameters:
##     alpha = 0.2401
##
##   Initial states:
##     l = 2031178.3537
##
##   sigma:  245212.1
##
##      AIC     AICc      BIC
## 4262.898 4263.071 4271.787
##
## Error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE      ACF1
## Training set 2891.404 243491.3 135145.3 -0.7420791 6.070649 0.9381871 0.1371394
##
## Forecasts:
##          Point Forecast   Lo 80   Hi 80   Lo 95   Hi 95
## 2012.769        2130433 1816181 2444685 1649826 2611040
## 2012.788        2130433 1807254 2453613 1636173 2624694
```

```
## 2012.808        2130433 1798566 2462300 1622886 2637980
## 2012.827        2130433 1790100 2470766 1609939 2650928
## 2012.846        2130433 1781840 2479027 1597306 2663561
## 2012.865        2130433 1773771 2487096 1584965 2675901
## 2012.885        2130433 1765880 2494986 1572898 2687969
## 2012.904        2130433 1758157 2502709 1561086 2699780
## 2012.923        2130433 1750591 2510276 1549514 2711352
## 2012.942        2130433 1743172 2517694 1538169 2722698
```

Conclusion : ##Has least MAPE and RMSE so seems to be the best method