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

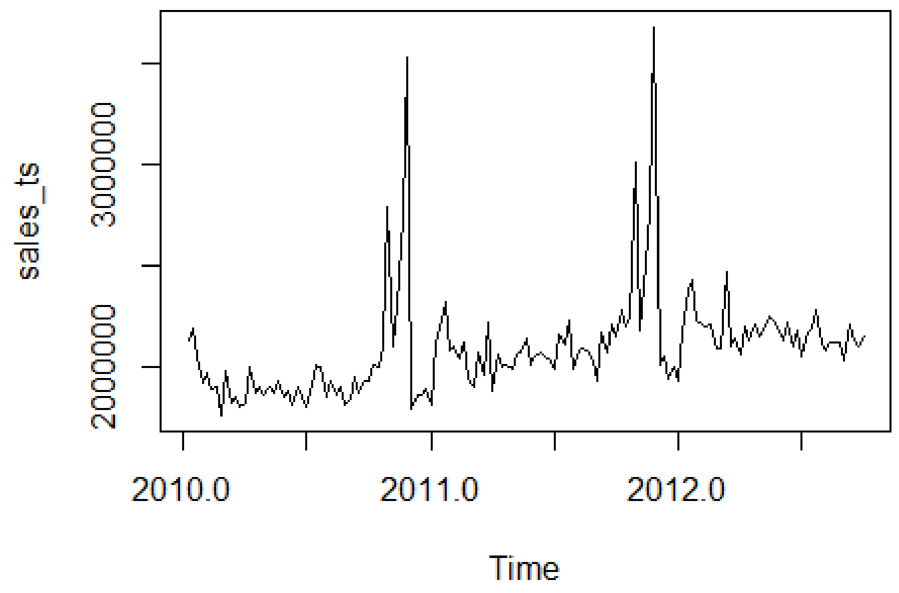
GAURAV MAHAJAN

2022-09-28

LOADING AND PLOTTING DATA USING TIMESERIES FORECASTING

```
library(forecast)

## Warning: package 'forecast' was built under R version 4.2.1

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

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

sales <- df$Sales
head(sales)

## [1] 2135144 2188307 2049860 1925729 1971057 1894324

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
```

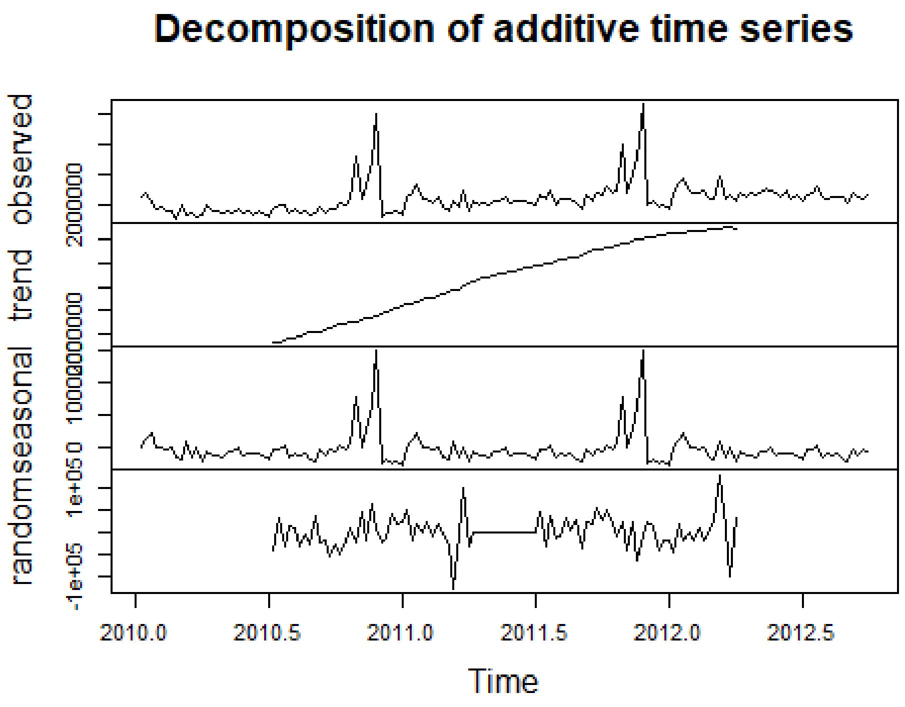
```
##  [10] 1818453 1851520 1802678 1817273 2000626 1875597 1903753 1857534
1903291
##  [19] 1870619 1929736 1846652 1881337 1812208 1898428 1848427 1796638
1907639
##  [28] 2007051 1997181 1848404 1935858 1865821 1899960 1810685 1842821
1951495
##  [37] 1867345 1927610 1933333 2013116 1999794 2097809 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

plot.ts(sales_ts)
```



USING THE DECOMPOSE FUNCTION TO DISPLAY THE LEVEL TREND RANDOM AND SEASONAL COMPONENTS OF THE TIME SERIES PLOT

###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).

```
sales_ts_component<-decompose(sales_ts)
plot(sales_ts_component)
```

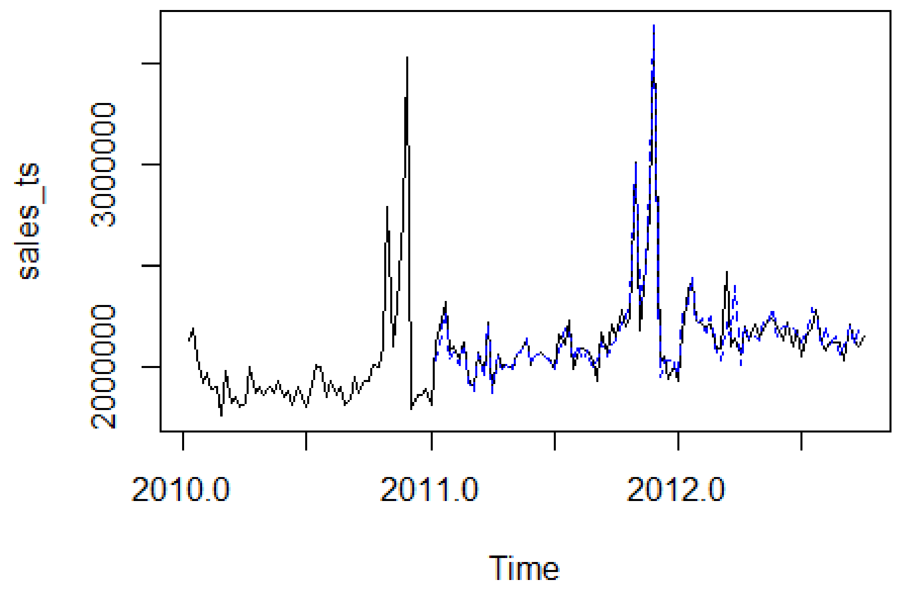


USING EXPONENTIAL SMOOTHING MODELS

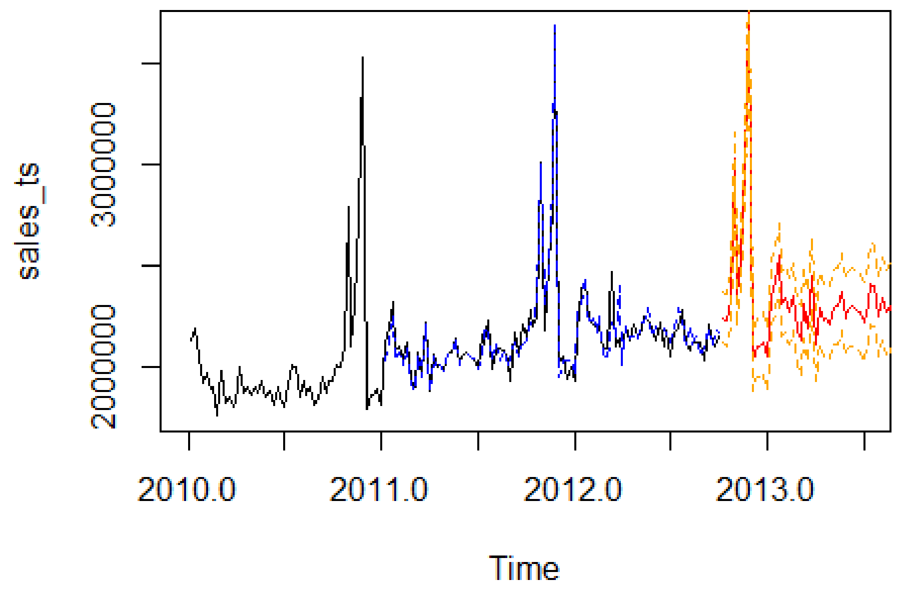
HERE I HAVE USED ONY HOLT WINTERS BUT SET THE PARAMETERS TO ZERO ACCORDING TO THE ORDER OF SMOOTHING REQUIRED.

###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)

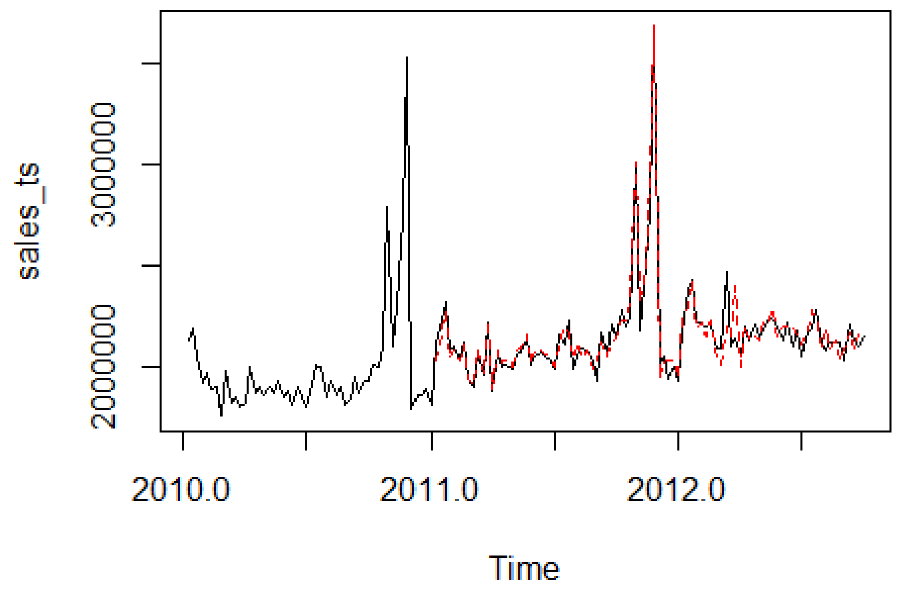
```
Single_exponential_smoothing <- HoltWinters(sales_ts, alpha=0.2, beta=0,
gamma=0)

plot(sales_ts)
lines(Single_exponential_smoothing$fitted[,1], lty=2, col="blue")
```



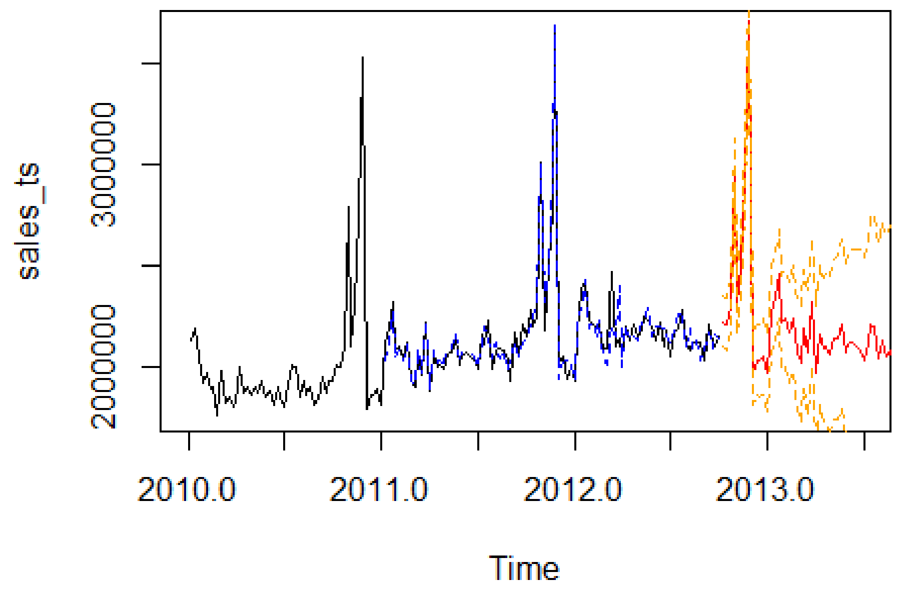
```
Single_exponential_smoothing.pred <- predict(Single_exponential_smoothing, 56
, prediction.interval = TRUE, level=0.95)
#Visually evaluate the prediction
plot(sales_ts , xlim=c(2010,2013.5))
lines(Single_exponential_smoothing$fitted[,1], lty=2, col="blue")
lines(Single_exponential_smoothing.pred[,1], col="red")
lines(Single_exponential_smoothing.pred[,2], lty=2, col="orange")
lines(Single_exponential_smoothing.pred[,3], lty=2, col="orange")
```



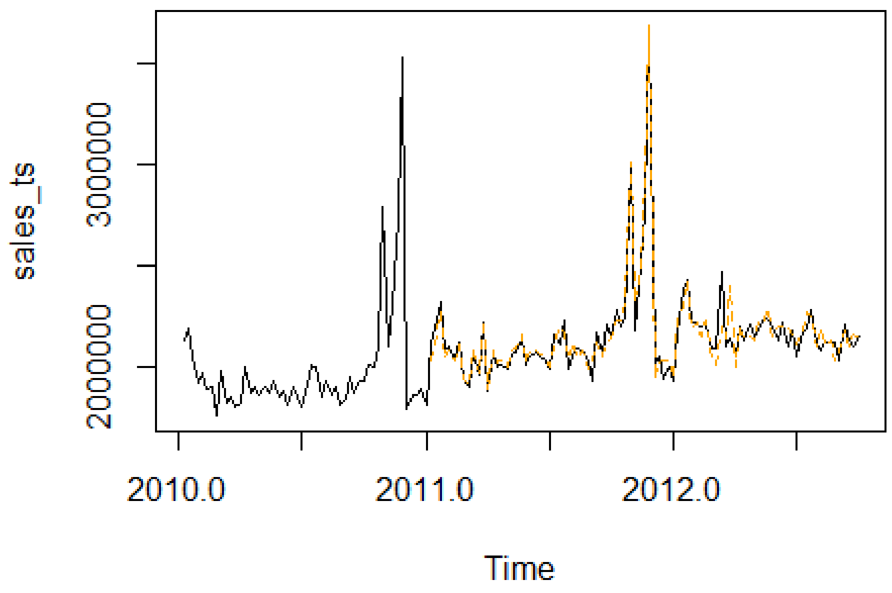
```
Double_exponential_smoothing <- HoltWinters(sales_ts, alpha=0.2, beta=0.1,
gamma=0)

plot(sales_ts)
lines(Double_exponential_smoothing$fitted[,1], lty=2, col="red")
```



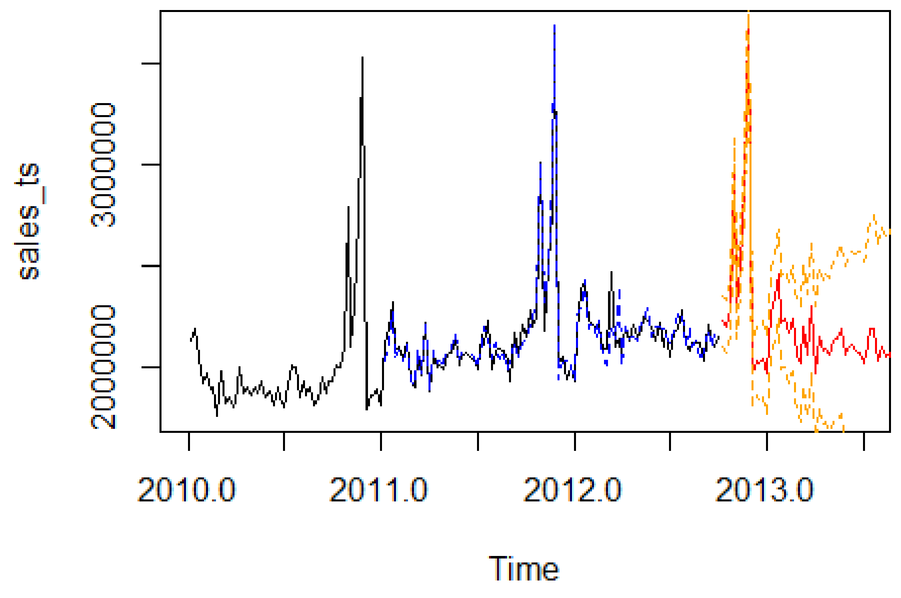
```
Double_exponential_smoothing.pred <- predict(Double_exponential_smoothing, 56
, prediction.interval = TRUE, level=0.95)

plot(sales_ts , xlim=c(2010,2013.5))
lines(Double_exponential_smoothing$fitted[,1], lty=2, col="blue")
lines(Double_exponential_smoothing.pred[,1], col="red")
lines(Double_exponential_smoothing.pred[,2], lty=2, col="orange")
lines(Double_exponential_smoothing.pred[,3], lty=2, col="orange")
```



```
Triple_exponential_smoothing <- HoltWinters(sales_ts, alpha=0.2, beta=0.1,
gamma=0.1)

plot(sales_ts)
lines(Triple_exponential_smoothing$fitted[,1], lty=2, col="orange")
```



```
Triple_exponential_smoothing_pred <- predict(Triple_exponential_smoothing, 56
, prediction.interval = TRUE, level=0.95)
plot(sales_ts , xlim=c(2010,2013.5))
lines(Triple_exponential_smoothing$fitted[,1], lty=2, col="blue")
lines(Triple_exponential_smoothing_pred[,1], col="red")
lines(Triple_exponential_smoothing_pred[,2], lty=2, col="orange")
lines(Triple_exponential_smoothing_pred[,3], lty=2, col="orange")
```



REGRESSION MODEL TO FORECAST THE SALES

###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

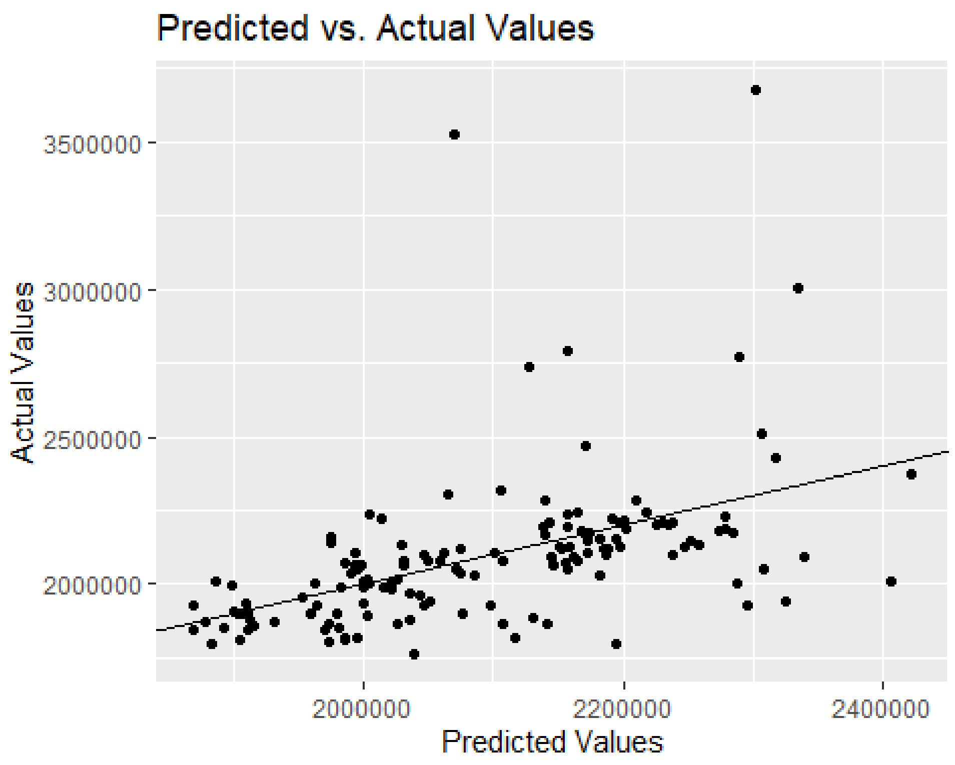
```
lm_pred <- lm(formula=Sales ~ .-X-Date,data=df)
summary(lm_pred)
```

```
##
## Call:
## lm(formula = Sales ~ . - X - Date, 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
```

```
#Plot the predicted values against original
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.1
```

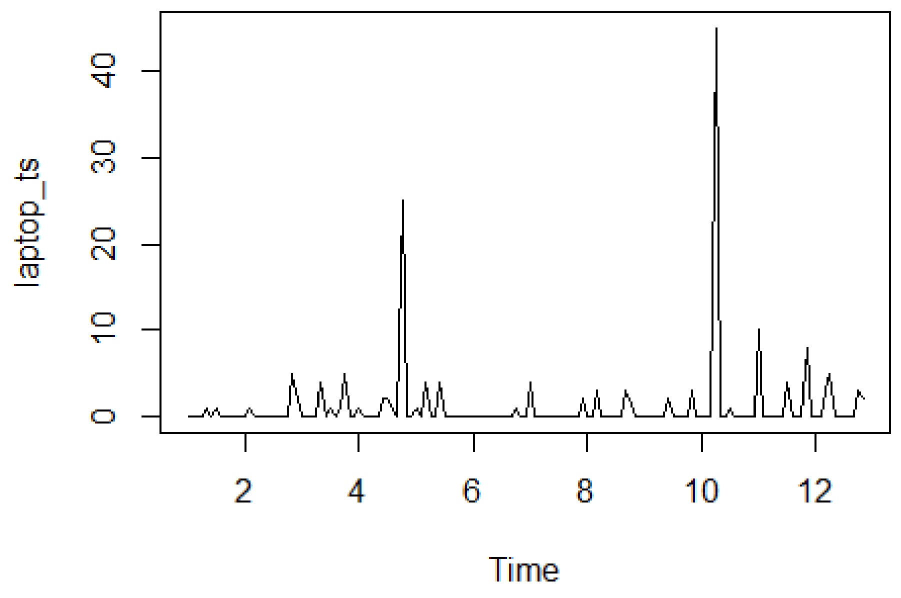
```
ggplot(df, aes(x=predict(lm_pred), y= Sales)) +
geom_point() +
geom_abline(intercept=0, slope=1) +
labs(x='Predicted Values', y='Actual Values', title='Predicted vs. Actual
Values')
```



###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)

```
library(forecast)

laptop <- df$Laptop_Demand
```

```
laptop_ts <- ts(laptop, frequency = 12)


plot(laptop_ts)
```



```
### ARIMA is used to forecast the above timeseries as we are predicting futur
values using the past.
mymodel <- auto.arima(laptop_ts)

forecast_future <- forecast(mymodel, level=c(95), h=143)
forecast_future

##        Point Forecast     Lo 95    Hi 95
## Dec 12       1.132867 -7.676191 9.941926
## Jan 13       1.132867 -7.676191 9.941926
## Feb 13       1.132867 -7.676191 9.941926
## Mar 13       1.132867 -7.676191 9.941926
## Apr 13       1.132867 -7.676191 9.941926
## May 13       1.132867 -7.676191 9.941926
## Jun 13       1.132867 -7.676191 9.941926
## Jul 13       1.132867 -7.676191 9.941926
## Aug 13       1.132867 -7.676191 9.941926
## Sep 13       1.132867 -7.676191 9.941926
## Oct 13       1.132867 -7.676191 9.941926
## Nov 13       1.132867 -7.676191 9.941926
## Dec 13       1.132867 -7.676191 9.941926
## Jan 14       1.132867 -7.676191 9.941926
## Feb 14       1.132867 -7.676191 9.941926
## Mar 14       1.132867 -7.676191 9.941926
## Apr 14       1.132867 -7.676191 9.941926
## May 14       1.132867 -7.676191 9.941926
## Jun 14       1.132867 -7.676191 9.941926
## Jul 14       1.132867 -7.676191 9.941926
## Aug 14       1.132867 -7.676191 9.941926
## Sep 14       1.132867 -7.676191 9.941926
## Oct 14       1.132867 -7.676191 9.941926
## Nov 14       1.132867 -7.676191 9.941926
## Dec 14       1.132867 -7.676191 9.941926
## Jan 15       1.132867 -7.676191 9.941926
## Feb 15       1.132867 -7.676191 9.941926
## Mar 15       1.132867 -7.676191 9.941926
## Apr 15       1.132867 -7.676191 9.941926
## May 15       1.132867 -7.676191 9.941926
## Jun 15       1.132867 -7.676191 9.941926
## Jul 15       1.132867 -7.676191 9.941926
## Aug 15       1.132867 -7.676191 9.941926
## Sep 15       1.132867 -7.676191 9.941926
## Oct 15       1.132867 -7.676191 9.941926
## Nov 15       1.132867 -7.676191 9.941926
## Dec 15       1.132867 -7.676191 9.941926
## Jan 16       1.132867 -7.676191 9.941926
## Feb 16       1.132867 -7.676191 9.941926
## Mar 16       1.132867 -7.676191 9.941926
## Apr 16       1.132867 -7.676191 9.941926
## May 16       1.132867 -7.676191 9.941926
## Jun 16       1.132867 -7.676191 9.941926
## Jul 16       1.132867 -7.676191 9.941926
## Aug 16       1.132867 -7.676191 9.941926
## Sep 16       1.132867 -7.676191 9.941926
## Oct 16       1.132867 -7.676191 9.941926
## Nov 16       1.132867 -7.676191 9.941926
## Dec 16       1.132867 -7.676191 9.941926
## Jan 17       1.132867 -7.676191 9.941926
## Feb 17       1.132867 -7.676191 9.941926
## Mar 17       1.132867 -7.676191 9.941926
## Apr 17       1.132867 -7.676191 9.941926
## May 17       1.132867 -7.676191 9.941926
## Jun 17       1.132867 -7.676191 9.941926
## Jul 17       1.132867 -7.676191 9.941926
## Aug 17       1.132867 -7.676191 9.941926
## Sep 17       1.132867 -7.676191 9.941926
## Oct 17       1.132867 -7.676191 9.941926
## Nov 17       1.132867 -7.676191 9.941926
## Dec 17       1.132867 -7.676191 9.941926
## Jan 18       1.132867 -7.676191 9.941926
## Feb 18       1.132867 -7.676191 9.941926
## Mar 18       1.132867 -7.676191 9.941926
## Apr 18       1.132867 -7.676191 9.941926
## May 18       1.132867 -7.676191 9.941926
## Jun 18       1.132867 -7.676191 9.941926
## Jul 18       1.132867 -7.676191 9.941926
## Aug 18       1.132867 -7.676191 9.941926
## Sep 18       1.132867 -7.676191 9.941926
## Oct 18       1.132867 -7.676191 9.941926
## Nov 18       1.132867 -7.676191 9.941926
## Dec 18       1.132867 -7.676191 9.941926
## Jan 19       1.132867 -7.676191 9.941926
## Feb 19       1.132867 -7.676191 9.941926
## Mar 19       1.132867 -7.676191 9.941926
## Apr 19       1.132867 -7.676191 9.941926
## May 19       1.132867 -7.676191 9.941926
## Jun 19       1.132867 -7.676191 9.941926
## Jul 19       1.132867 -7.676191 9.941926
## Aug 19       1.132867 -7.676191 9.941926
## Sep 19       1.132867 -7.676191 9.941926
## Oct 19       1.132867 -7.676191 9.941926
## Nov 19       1.132867 -7.676191 9.941926
## Dec 19       1.132867 -7.676191 9.941926
## Jan 20       1.132867 -7.676191 9.941926
## Feb 20       1.132867 -7.676191 9.941926
## Mar 20       1.132867 -7.676191 9.941926
## Apr 20       1.132867 -7.676191 9.941926
## May 20       1.132867 -7.676191 9.941926
## Jun 20       1.132867 -7.676191 9.941926
## Jul 20       1.132867 -7.676191 9.941926
## Aug 20       1.132867 -7.676191 9.941926
## Sep 20       1.132867 -7.676191 9.941926
## Oct 20       1.132867 -7.676191 9.941926
## Nov 20       1.132867 -7.676191 9.941926
## Dec 20       1.132867 -7.676191 9.941926
## Jan 21       1.132867 -7.676191 9.941926
## Feb 21       1.132867 -7.676191 9.941926
## Mar 21       1.132867 -7.676191 9.941926
## Apr 21       1.132867 -7.676191 9.941926
## May 21       1.132867 -7.676191 9.941926
## Jun 21       1.132867 -7.676191 9.941926
## Jul 21       1.132867 -7.676191 9.941926
## Aug 21       1.132867 -7.676191 9.941926
## Sep 21       1.132867 -7.676191 9.941926
## Oct 21       1.132867 -7.676191 9.941926
## Nov 21       1.132867 -7.676191 9.941926
## Dec 21       1.132867 -7.676191 9.941926
## Jan 22       1.132867 -7.676191 9.941926
## Feb 22       1.132867 -7.676191 9.941926
## Mar 22       1.132867 -7.676191 9.941926
## Apr 22       1.132867 -7.676191 9.941926
## May 22       1.132867 -7.676191 9.941926
## Jun 22       1.132867 -7.676191 9.941926
## Jul 22       1.132867 -7.676191 9.941926
## Aug 22       1.132867 -7.676191 9.941926
## Sep 22       1.132867 -7.676191 9.941926
## Oct 22       1.132867 -7.676191 9.941926
## Nov 22       1.132867 -7.676191 9.941926
## Dec 22       1.132867 -7.676191 9.941926
## Jan 23       1.132867 -7.676191 9.941926
## Feb 23       1.132867 -7.676191 9.941926
## Mar 23       1.132867 -7.676191 9.941926
## Apr 23       1.132867 -7.676191 9.941926
## May 23       1.132867 -7.676191 9.941926
## Jun 23       1.132867 -7.676191 9.941926
## Jul 23       1.132867 -7.676191 9.941926
## Aug 23       1.132867 -7.676191 9.941926
## Sep 23       1.132867 -7.676191 9.941926
## Oct 23       1.132867 -7.676191 9.941926
## Nov 23       1.132867 -7.676191 9.941926
## Dec 23       1.132867 -7.676191 9.941926
## Jan 24       1.132867 -7.676191 9.941926
## Feb 24       1.132867 -7.676191 9.941926
## Mar 24       1.132867 -7.676191 9.941926
## Apr 24       1.132867 -7.676191 9.941926
## May 24       1.132867 -7.676191 9.941926
## Jun 24       1.132867 -7.676191 9.941926
## Jul 24       1.132867 -7.676191 9.941926
## Aug 24       1.132867 -7.676191 9.941926
## Sep 24       1.132867 -7.676191 9.941926
## Oct 24       1.132867 -7.676191 9.941926
```

###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)

```
library("Metrics")

## Warning: package 'Metrics' was built under R version 4.2.1

##
## Attaching package: 'Metrics'

## The following object is masked from 'package:forecast':
##
##     accuracy

#For regression Model
sprintf("MAPE value for regression model is:
%f",mape(sales_ts,lm_pred$fitted))
```

```
## [1] "MAPE value for regression model is: 0.056019"

sprintf("RMSE value for Regression model is:
%f",rmse(sales_ts,lm_pred$fitted))

## [1] "RMSE value for Regression model is: 232909.197321"

#For Single Exponential Smoothing Model
sprintf("MAPE value for Single Exponential smoothing model is:
%f",mape(sales_ts,Single_exponential_smoothing$fitted))

## [1] "MAPE value for Single Exponential smoothing model is: 0.524194"

sprintf("RMSE value for Single Exponential smoothing model is:
%f",rmse(sales_ts,Single_exponential_smoothing$fitted))

## [1] "RMSE value for Single Exponential smoothing model is: 1543505.407992"

#For Double Exponential Smoothing Model
sprintf("MAPE value for Double Exponential smoothing model is:
%f",mape(sales_ts,Double_exponential_smoothing$fitted))

## [1] "MAPE value for Double Exponential smoothing model is: 0.524445"

sprintf("RMSE value for Double Exponential smoothing model is:
%f",rmse(sales_ts,Double_exponential_smoothing$fitted))

## [1] "RMSE value for Double Exponential smoothing model is: 1543547.820045"

#For Triple Exponential Smoothing Model
sprintf("MAPE value for Triple Exponential smoothing model is:
%f",mape(sales_ts,Triple_exponential_smoothing$fitted))

## [1] "MAPE value for Triple Exponential smoothing model is: 0.524451"

sprintf("RMSE value for Triple Exponential smoothing model is:
%f",rmse(sales_ts,Triple_exponential_smoothing$fitted))

## [1] "RMSE value for Triple Exponential smoothing model is: 1543510.496331"
```