UE20CS312 - Data Analytics

Worksheet 3a - Basic Forecasting Techniques

PES University

'SUNDEEP A, Dept. of CSE - PES1UG20CS445'

2022-09-29

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!

Prerequisites

- Revise the following concepts
  - Components of Time-Series Data
  - Single, Double and Triple Exponential Smoothing
  - Regression (Refer to worksheets and slides from Unit 2)
  - Croston's Forecasting
  - Time-Series Accuracy Metrics

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.

The data for the following tasks can be downloaded from this Github Repository.

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

```
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
```

- Pick out the `Sales` column in the `data.frame`. Most of our time-series analysis will be done on this column.

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

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

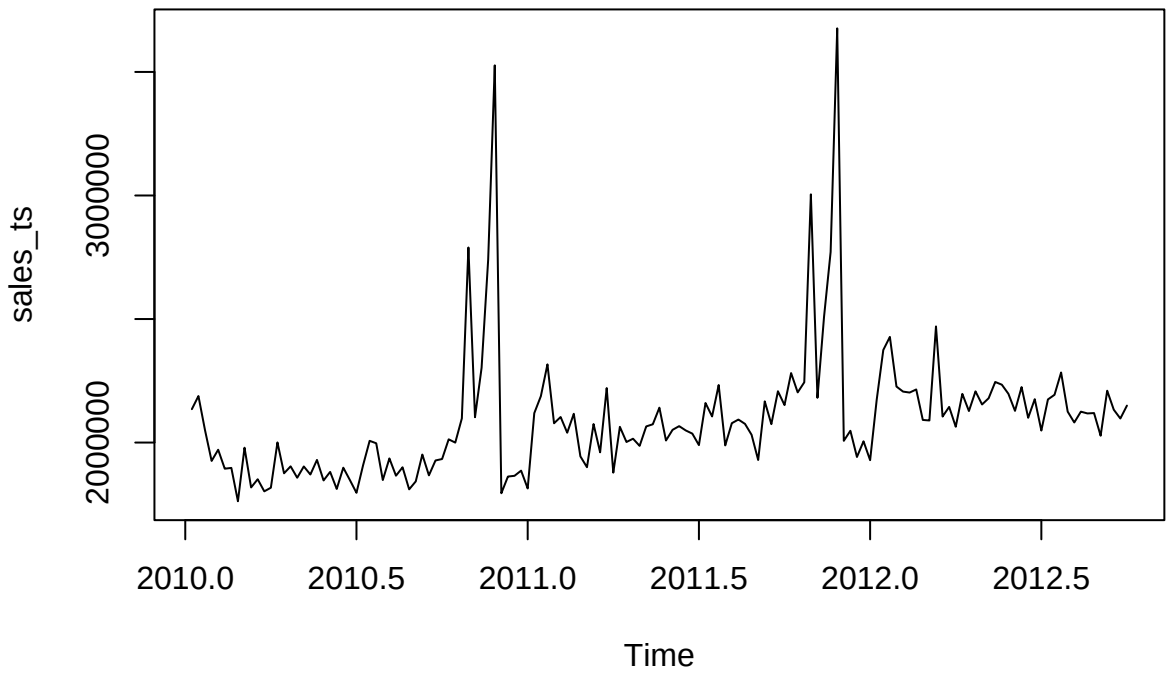
- The `ts` function is used to create the `ts` object in R. Frequency is 52 as it is weekly data. The start is specified like `start= c(y, m, d)` as we are dealing with weekly data. If it was monthly data we can omit the `d` and for yearly data we can omit the `m` as well.(`c` is the combine function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

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

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

- Visualize the Time-Series of `Sales` column

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



Points

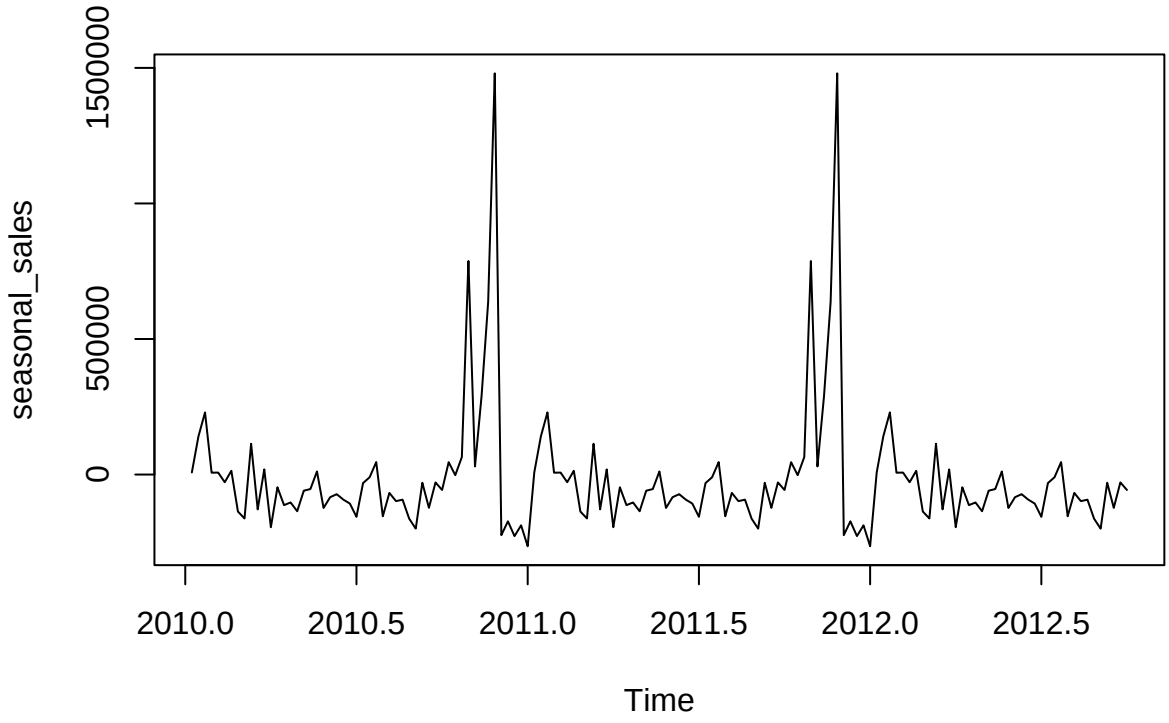
The problems in this worksheet are for a total of 10 points with each problem having a different weightage.

- *Problem 1*: 1 point
- *Problem 2*: 3 points
- *Problem 3*: 2 points
- *Problem 4*: 2 point
- *Problem 5*: 2 points

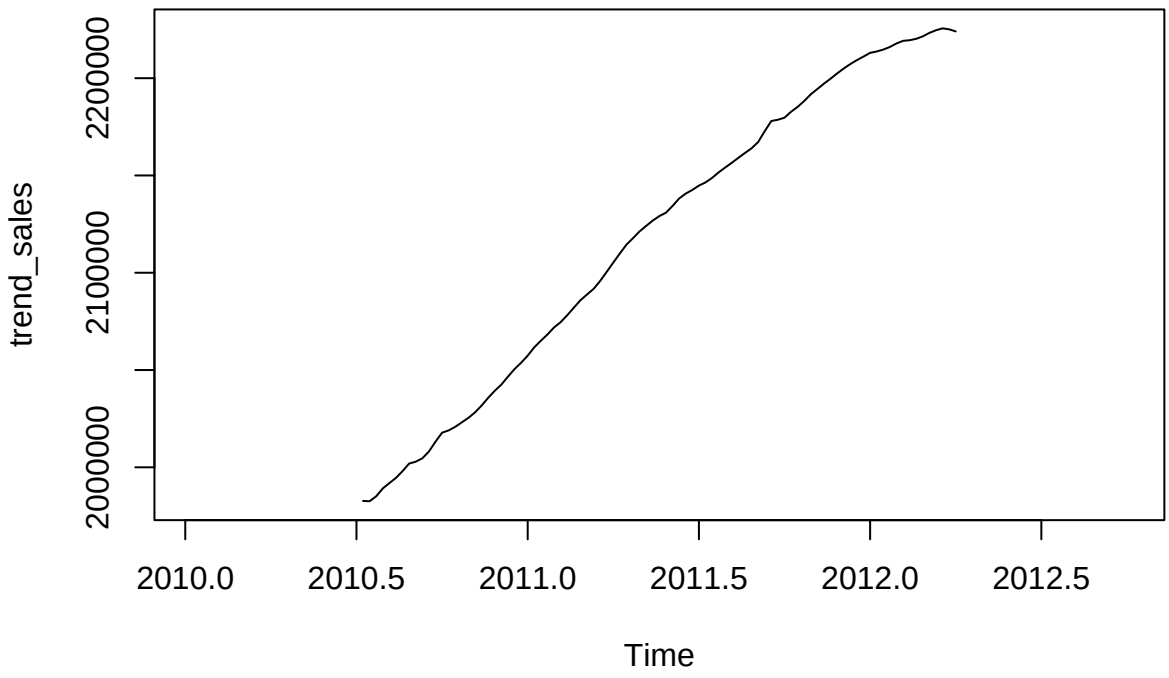
Problem 1 (1 Point)

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

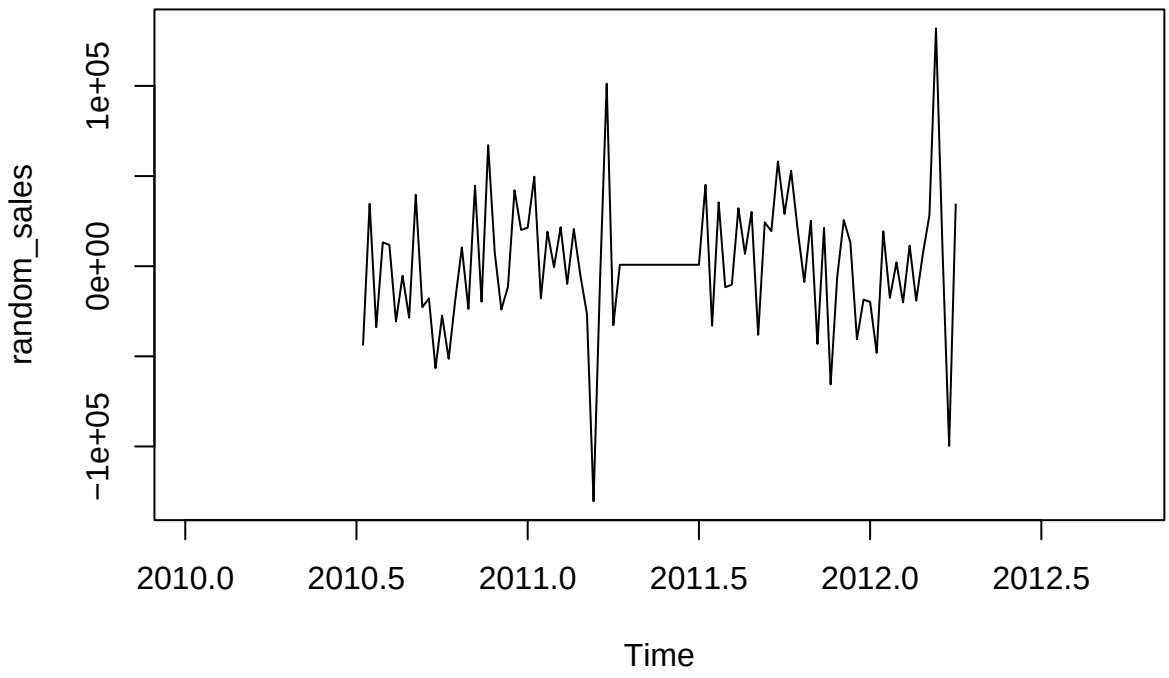
```
decomp = decompose(sales_ts)
#seasonal Component
seasonal_sales = decomp$seasonal
plot(seasonal_sales)
```




```
#Trend Component
trend_sales = decomp$trend
plot(trend_sales)
```




```
#Random Component
random_sales = decomp$random
plot(random_sales)
```



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)

```
library(fpp2)
```

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

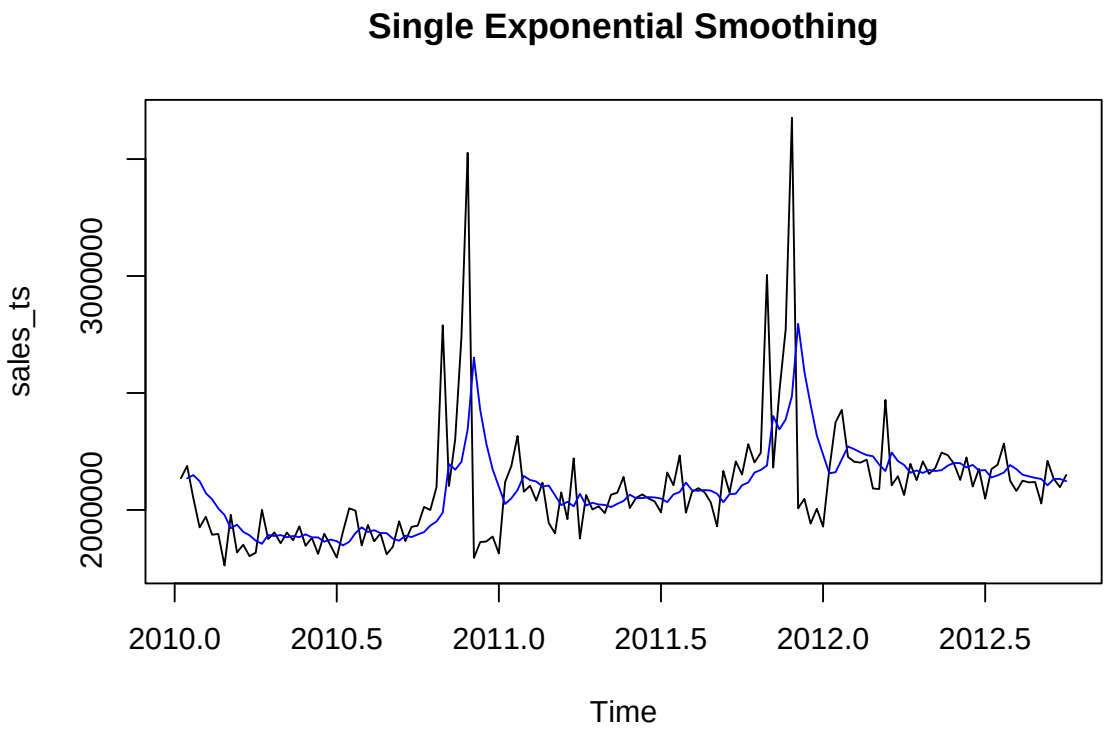
```
## -- Attaching packages --------------------------------------------- fpp2 2.4 --
```

```
## v ggplot2   3.3.6      v fma       2.4
## v forecast  8.17.0     v expsmooth 2.3
```

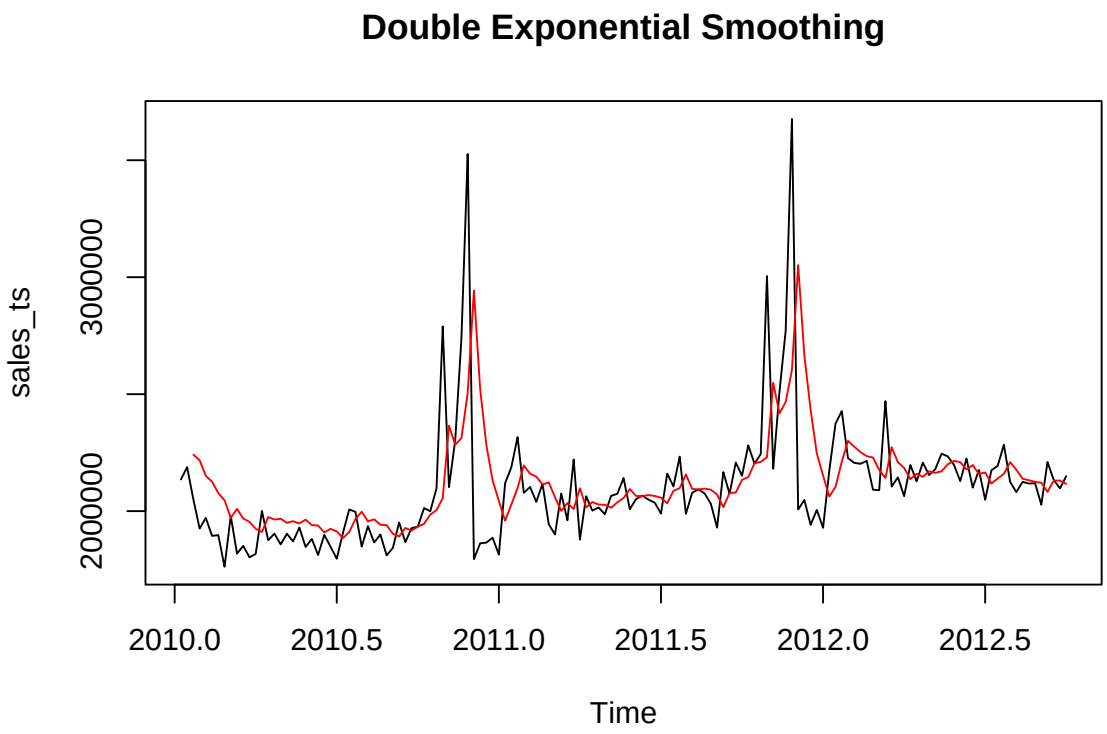
```
##
```

```
#Performing Single Exponential Smoothing
Single_exponential_smoothing <- HoltWinters(sales_ts,beta=FALSE, gamma=FALSE , start = 2010)
#Performing  Double Exponential Smoothing
Double_exponential_smoothing <- HoltWinters(sales_ts,gamma=FALSE , start = 2010)
#Performing  Triple Exponential Smoothing
Triple_exponential_smoothing <- HoltWinters(sales_ts)
```

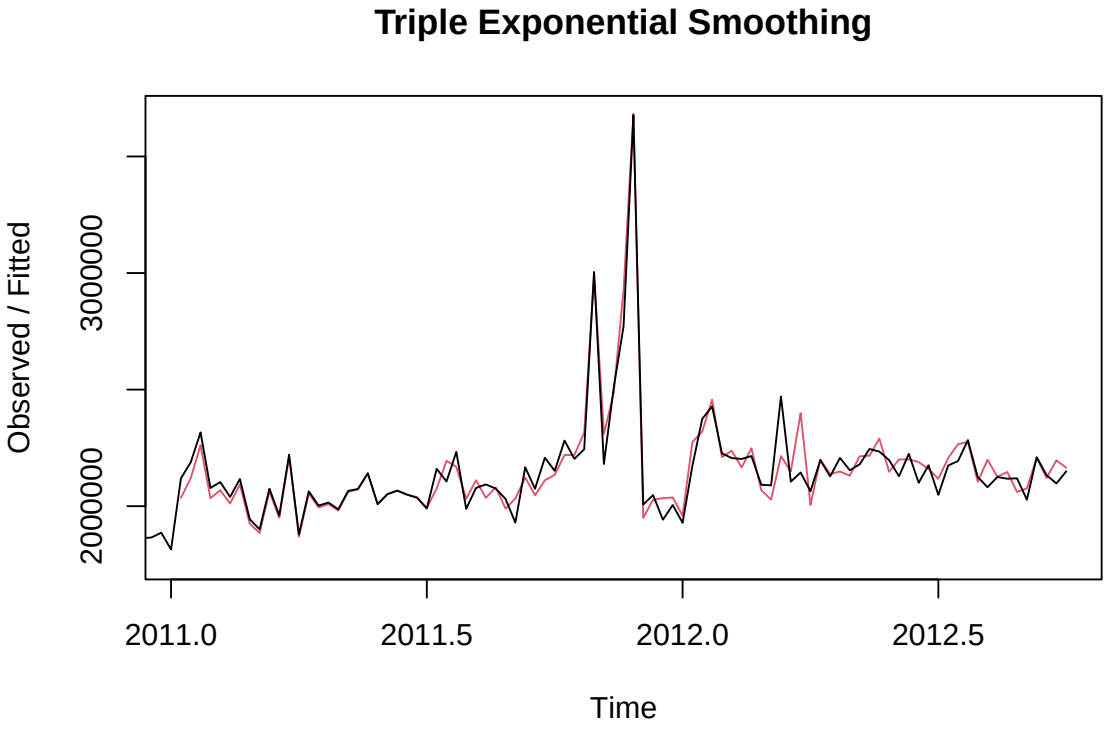
```
plot(sales_ts,main="Single Exponential Smoothing")
lines(Single_exponential_smoothing$fitted[,1], col="blue")
```




```
plot(sales_ts,main="Double Exponential Smoothing")
lines(Double_exponential_smoothing$fitted[,1],col="red")
```




```
plot(Triple_exponential_smoothing,main="Triple Exponential Smoothing")
```



Problem 3 (2 Points)

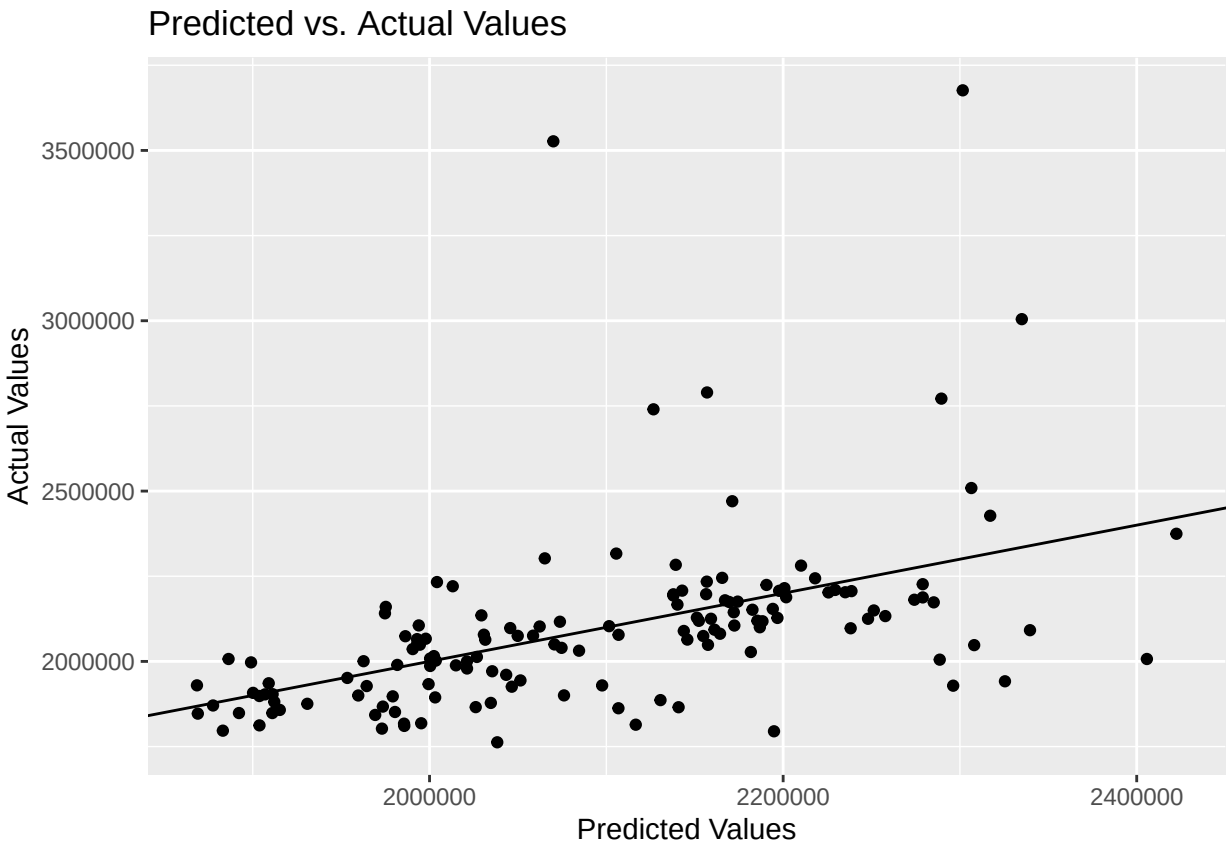
- Forecast `Sales` values by Regression using all other columns. Print summary of regression model.
- Plot the predicted values against original as well. (Hint: Regression model predictions will not be a Time Series, so need to get both to common index/x-axis)
- (Hint: Will not work unless one column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

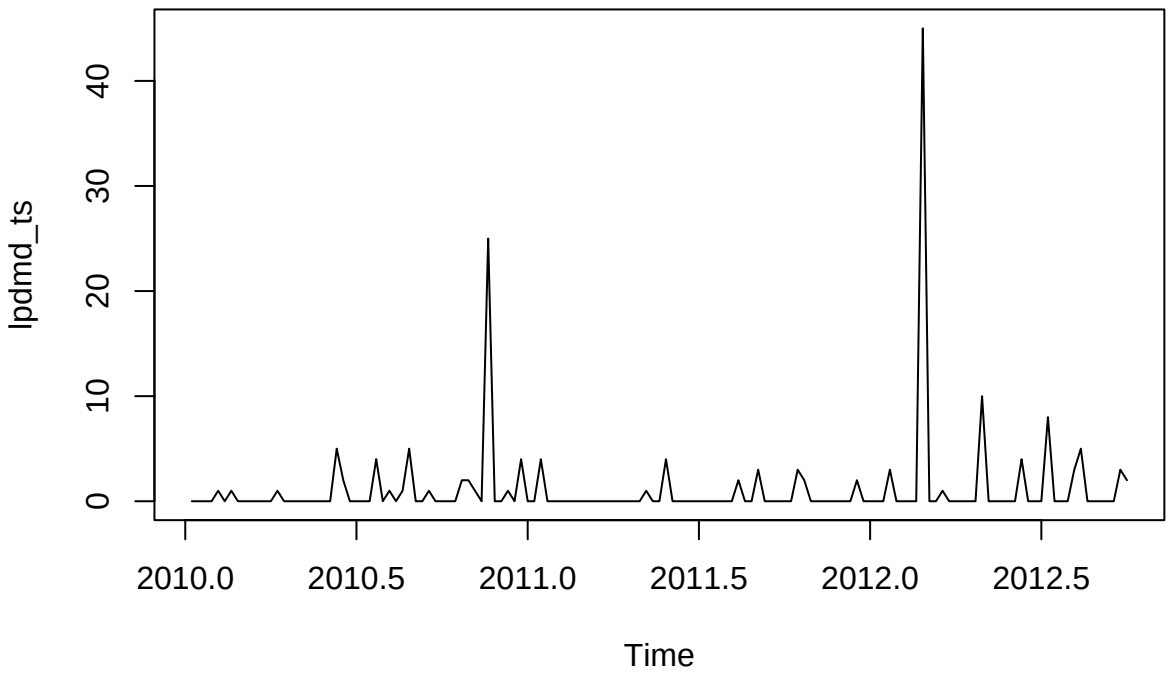
```
#Plot the predicted values against original
library(ggplot2)
ggplot(df, aes(x=predict(lm_pred), y= Sales)) +
  geom_point() +
  geom_abline(intercept=0, slope=1) +
  labs(x='Predicted Values', y='Actual Values', title='Predicted vs. Actual Values')
```



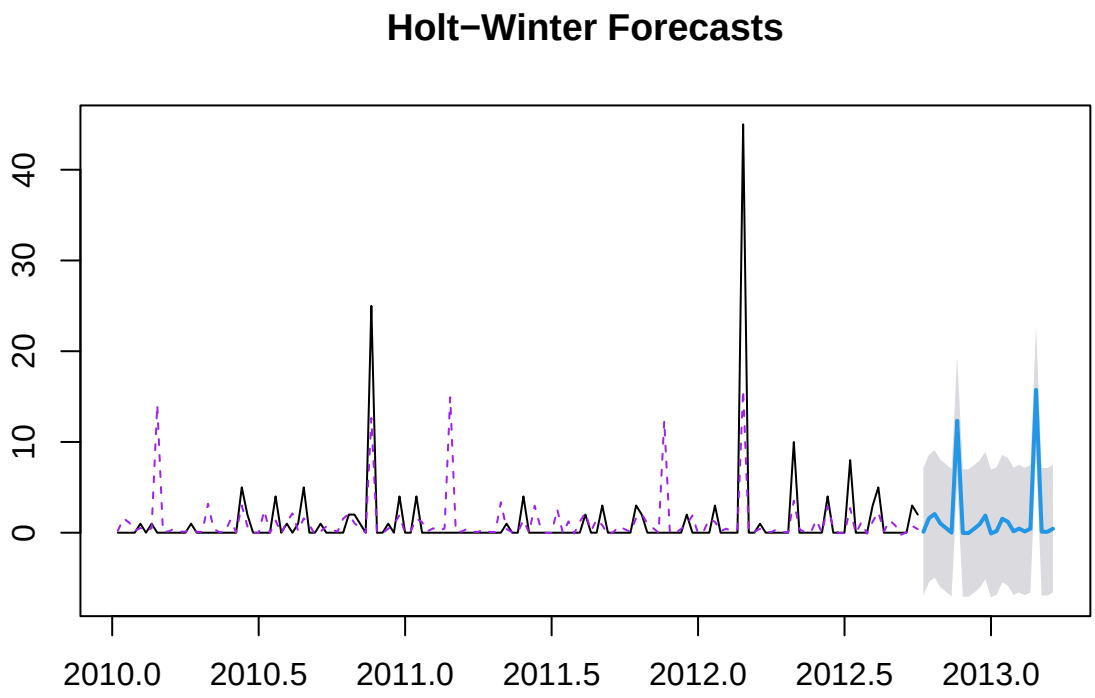
Problem 4 (2 Points)

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")
#Plotting the Laptop_Demand as a time Series
lpdmd_ts <- ts(df$Laptop_Demand, frequency = 52, start=c(2010, 2, 5))
plot.ts(lpdmd_ts)
```




```
#Forecasting required for this type of time-series is Holt-Winter Forecasting.
lpdmd_forecast <- forecast(lpdmd_ts, h=24, level=95)
plot(lpdmd_forecast,main='Holt-Winter Forecasts')
lines(lpdmd_forecast$fitted, lty=2, col="purple")
```



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")
```

```
##
## 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_smo
```

```
## [1] "MAPE value for Single Exponential smoothing model is: 0.061437"
```

```
sprintf("RMSE value for Single Exponential smoothing model is: %f",rmse(sales_ts,Single_exponential_smo
```

```
## [1] "RMSE value for Single Exponential smoothing model is: 244672.518751"
```

```
#For Double Exponential Smoothing Model
sprintf("MAPE value for Double Exponential smoothing model is: %f",mape(sales_ts,Double_exponential_smo
```

```
## [1] "MAPE value for Double Exponential smoothing model is: 0.376055"
```

```
sprintf("RMSE value for Double Exponential smoothing model is: %f",rmse(sales_ts,Double_exponential_smo
```

```
## [1] "RMSE value for Double Exponential smoothing model is: 1230068.199944"
```

```
#For Triple Exponential Smoothing Model
sprintf("MAPE value for Triple Exponential smoothing model is: %f",mape(sales_ts,Triple_exponential_smo
```

```
## [1] "MAPE value for Triple Exponential smoothing model is: 0.523961"
```

```
sprintf("RMSE value for Triple Exponential smoothing model is: %f",rmse(sales_ts,Triple_exponential_smo
```

```
## [1] "RMSE value for Triple Exponential smoothing model is: 1542982.632181"
```

Analysis:

Since, both MAPE and RMSE are error metrics, lower the error metrics (i.e. MAPE and RMSE), better is the prediction of the model.

Since Regression Model has the **lowest MAPE and RMSE** among all the models. So we can say that Regression model is the Best Model.

In the Exponential Models, Single Regression Model had the Lowest MAPE and RMSE values . So, we can say that Single Exponential Smoothing Model is the Best model among the Exponential Smoothing Models.