**Assignment 1**

**Title:** Analytical Modeling and Predictive Analytics for BI in Microsoft Excel

**Aim :** Design a business intelligence application using What -if Analysis and Simple Linear Regression using Microsoft Excel.

**Objective**: To implement Analytical Modeling and Predictive Analytics in Microsoft Excel

**Theory:**

* **What if analysis in Microsoft Excel:**

Excel provides many powerful tools to perform complex mathematical calculations, including **what-if analysis**. This feature can help you **experiment** and **answer questions** with your data, even when the data is incomplete.

It includes scenarios, data tables, and goal seek.

1. **Scenarios:**

A **Scenario** is a set of values that Excel saves and can substitute automatically on your worksheet. You can create and save different groups of values as scenarios and then switch between these scenarios to view the different results. After you have all the scenarios you need, you can create a scenario summary report that incorporates information from all the scenarios.

A scenario can have a maximum of 32 different values, but you can create as many scenarios as you want.

1. **Data Tables:**

A data table is a range of cells in which you can change values in some in some of the cells and come up with different answers to a problem. You can create one-variable or two-variable data tables, depending on the number of variables and formulas that you want to test.

* 1. **One-variable data tables**    Use a one-variable data table if you want to see how different values of one variable in one or more formulas will change the results of those formulas.
  2. **Two-variable data tables**    Use a two-variable data table to see how different values of two variables in one formula will change the results of that formula.

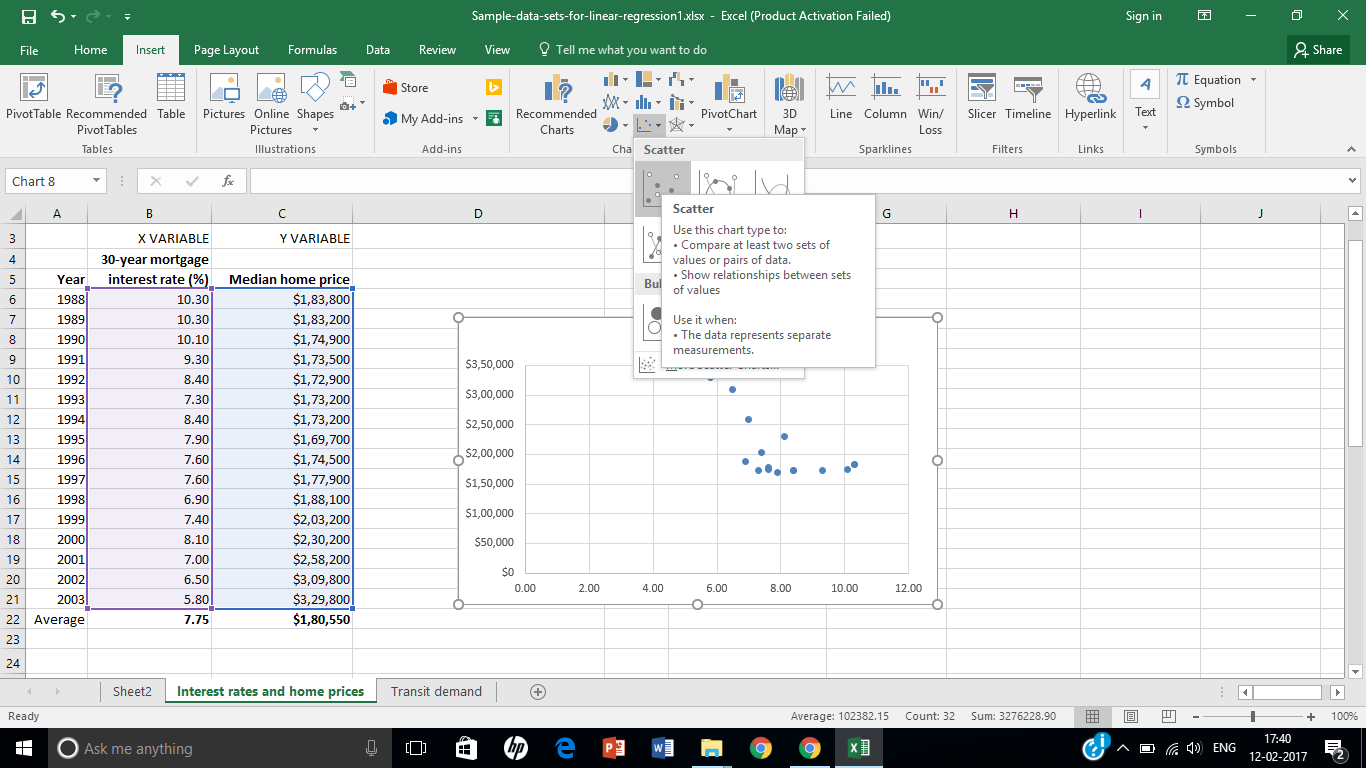
**Data table calculations :** Data tables are recalculated whenever a worksheet is recalculated, even if they have not changed. To speed up calculation of a worksheet that contains a data table, you can change the

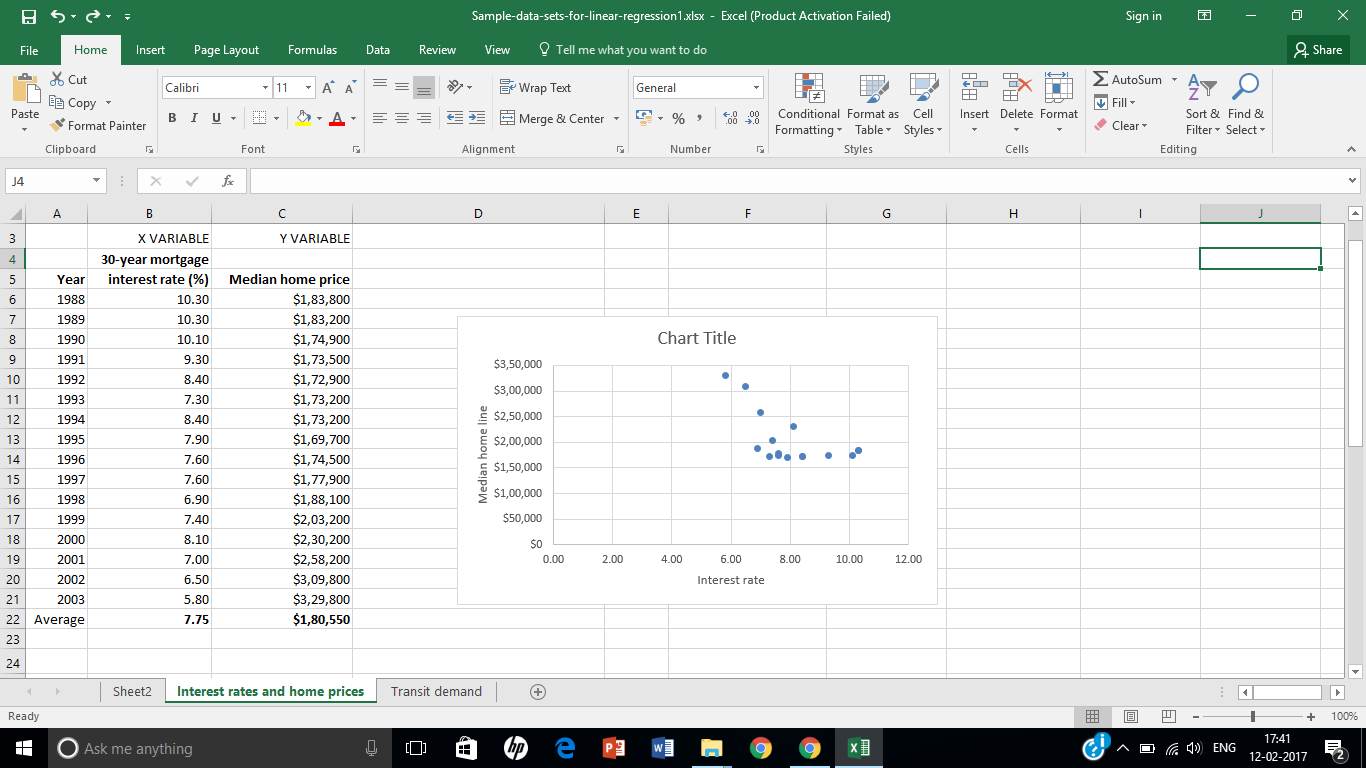
**Calculation** options to automatically recalculate the worksheet but not the data tables. See the section [Speed up calculation in a worksheet that contains data tables](https://support.office.com/en-us/article/Calculate-multiple-results-by-using-a-data-table-e95e2487-6ca6-4413-ad12-77542a5ea50b?ui=en-US&rs=en-US&ad=US" \l "bmspeed).

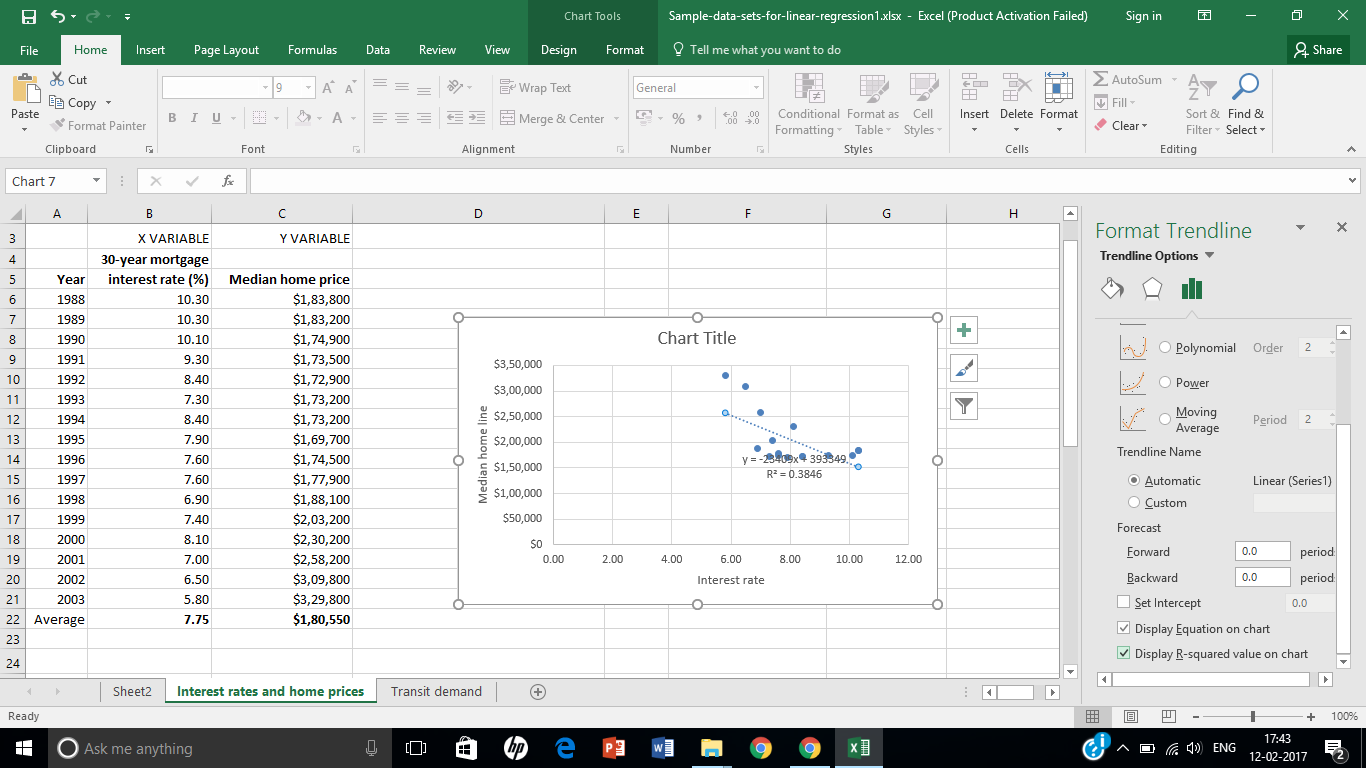
1. **Goal Seek:**

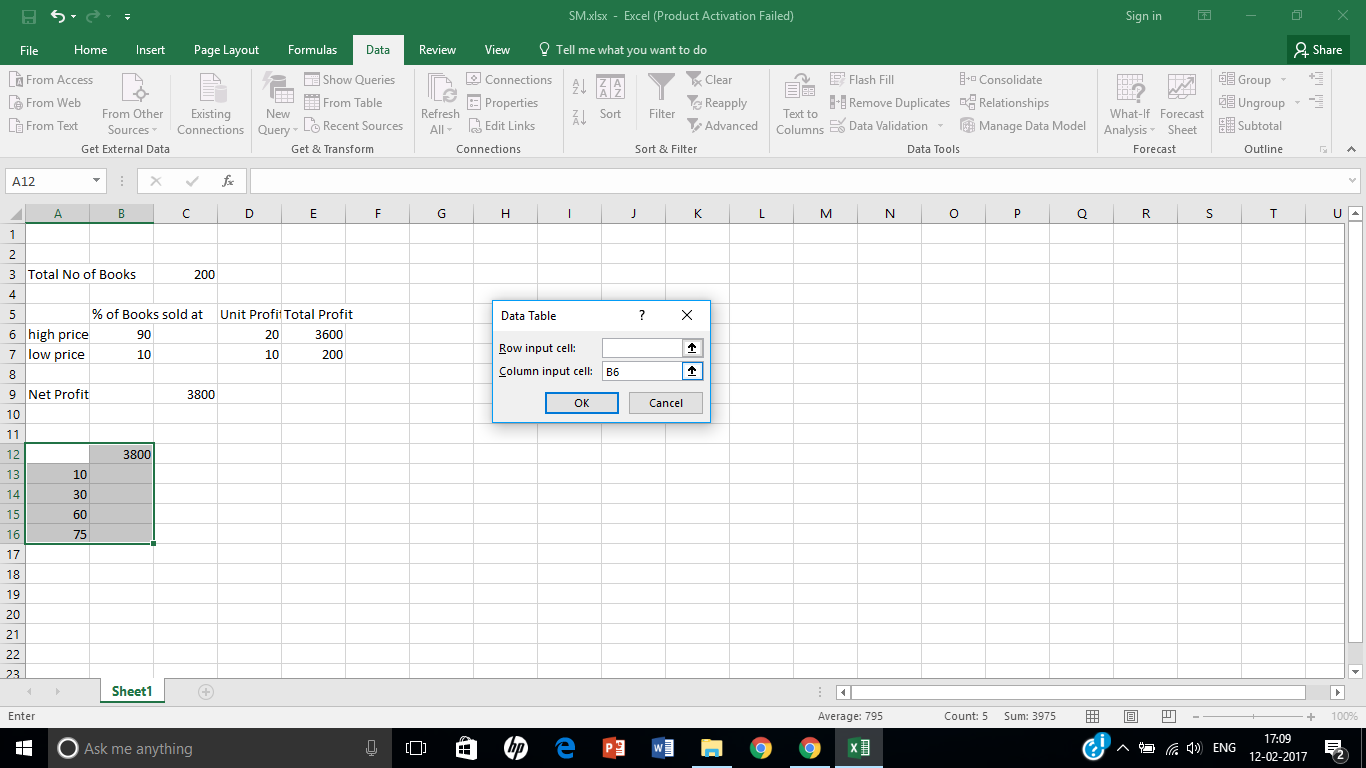
Whenever you create a formula or function in Excel, you put various parts together to calculate a **result**. **Goal Seek** works in the opposite way: It lets you start with the **desired result**, and it calculates the **input value** that will give you that result. We'll use a few examples to show how to use Goal Seek. If you know the result that you want from a formula, but are not sure what input value the formula needs to get that result, use the Goal Seek feature.

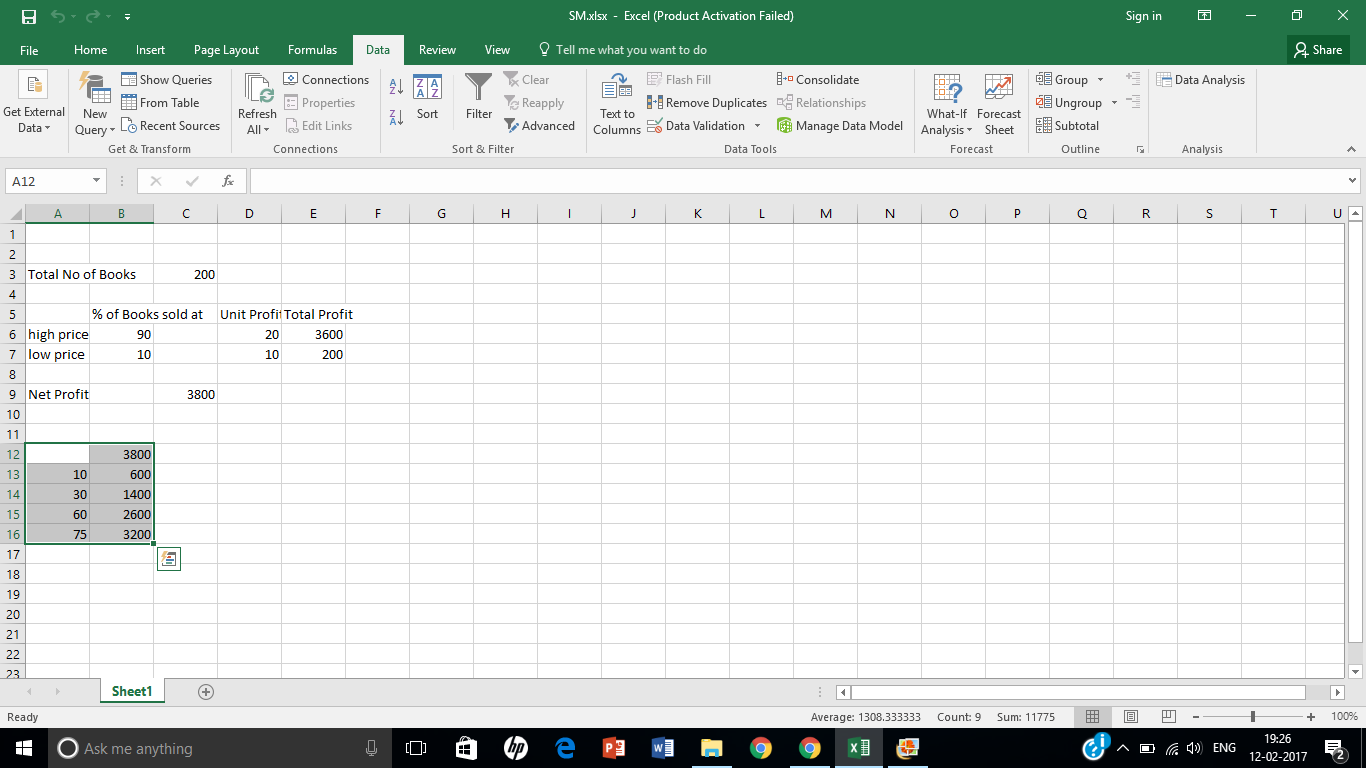
* **Simple Linear Regression in Microsoft Excel:**
* To perform Simple Linear Regression, you have to begin by creating a data table that has the independent and dependent variables.
* Go to the **Tools** Menu and select **Data Analysis**.
* From the **Data Analysis** window select **Regression**.
* The next step is to tell the Regression Wizard the things it needs to know; the location of the Y data, the location of the X data, and the place to put the result of the regression analysis.
* The output display R-square, Y intercept value and slope.

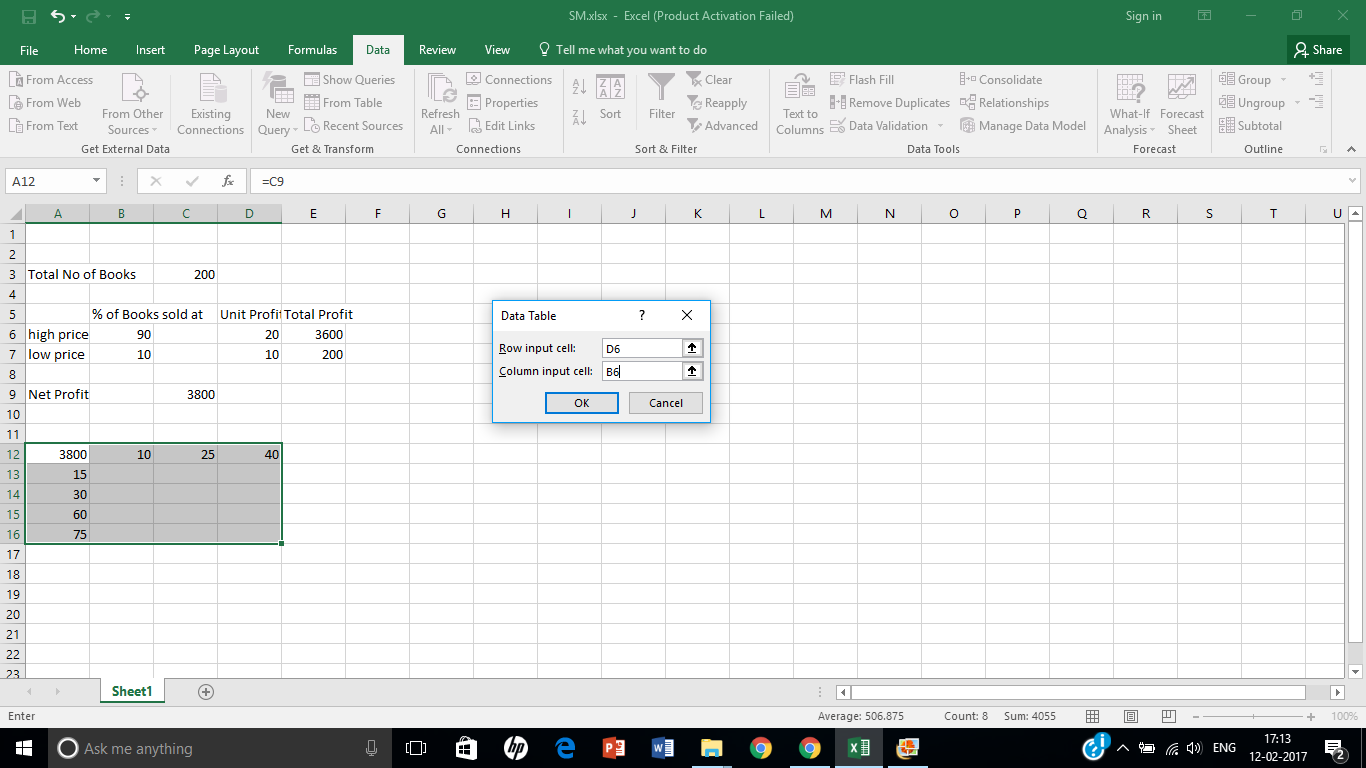
****

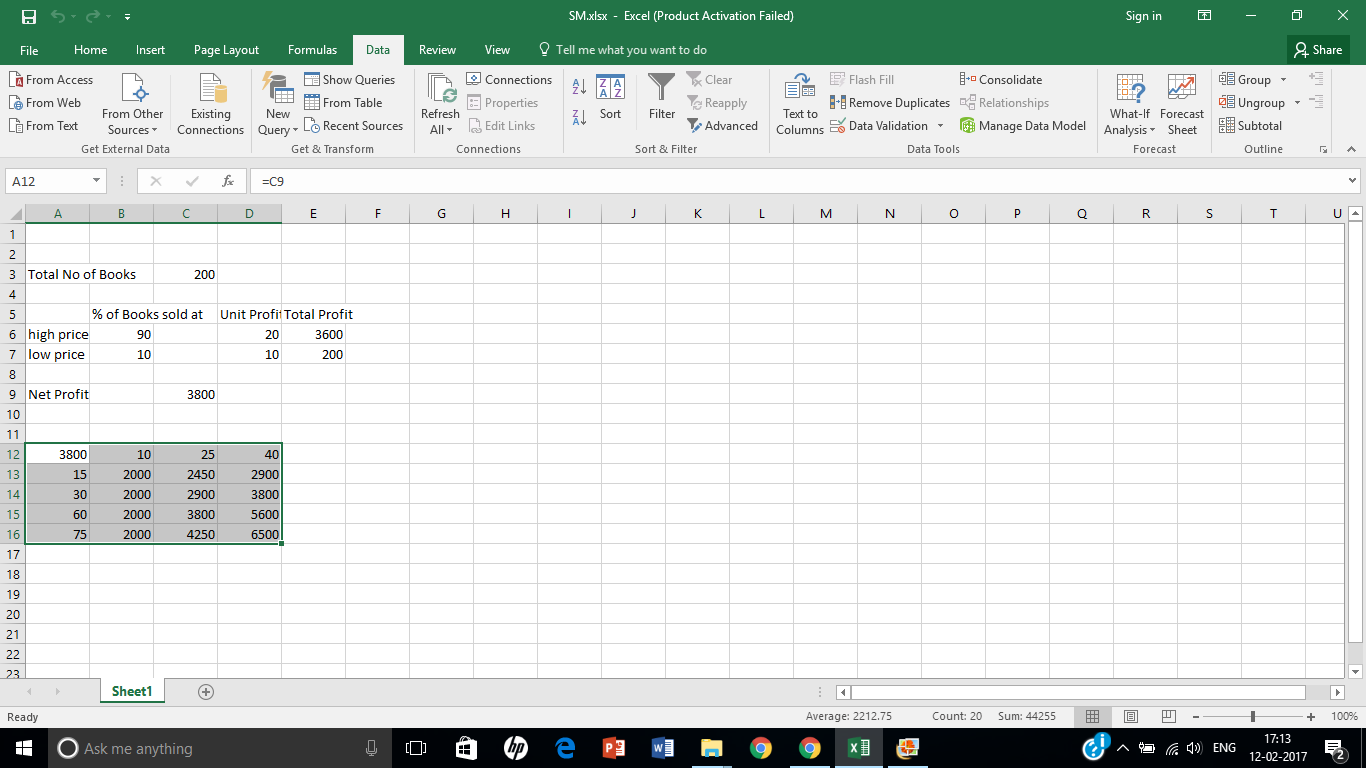
****

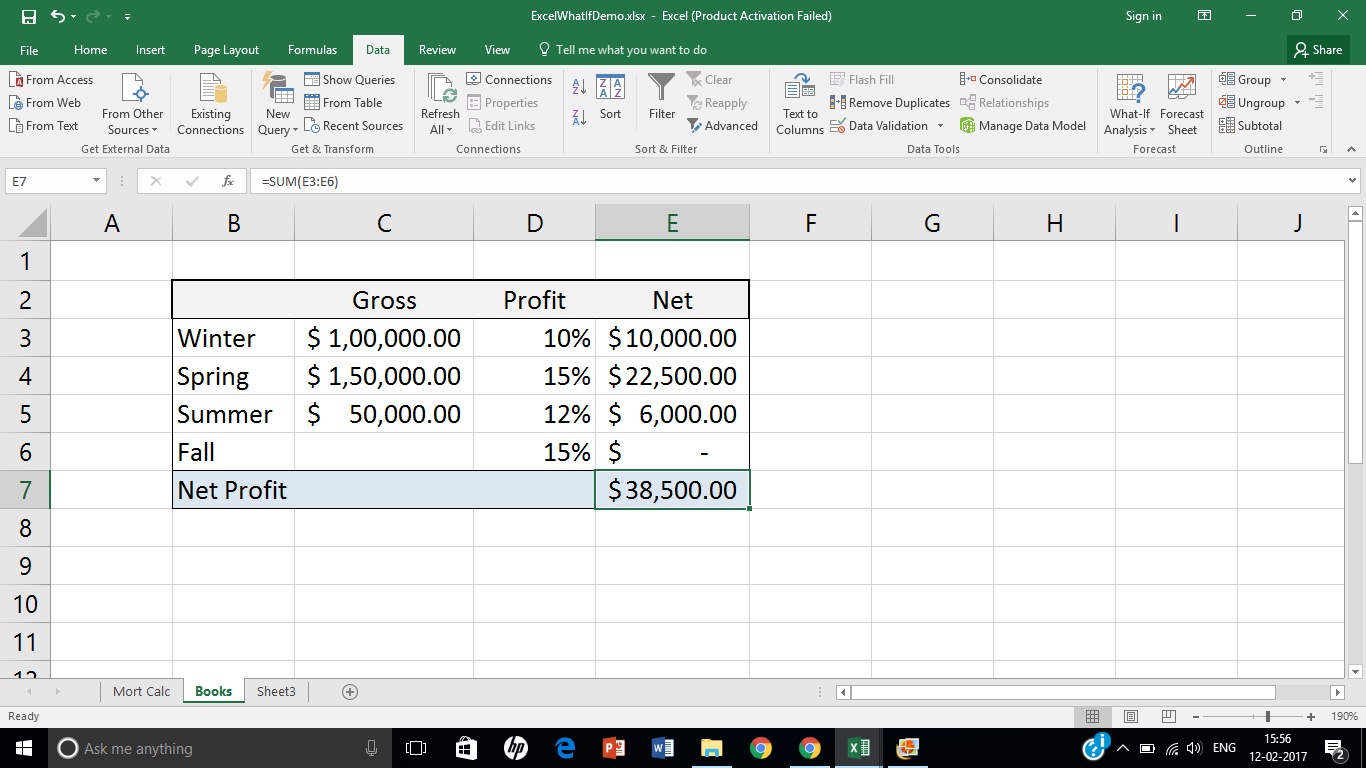
****

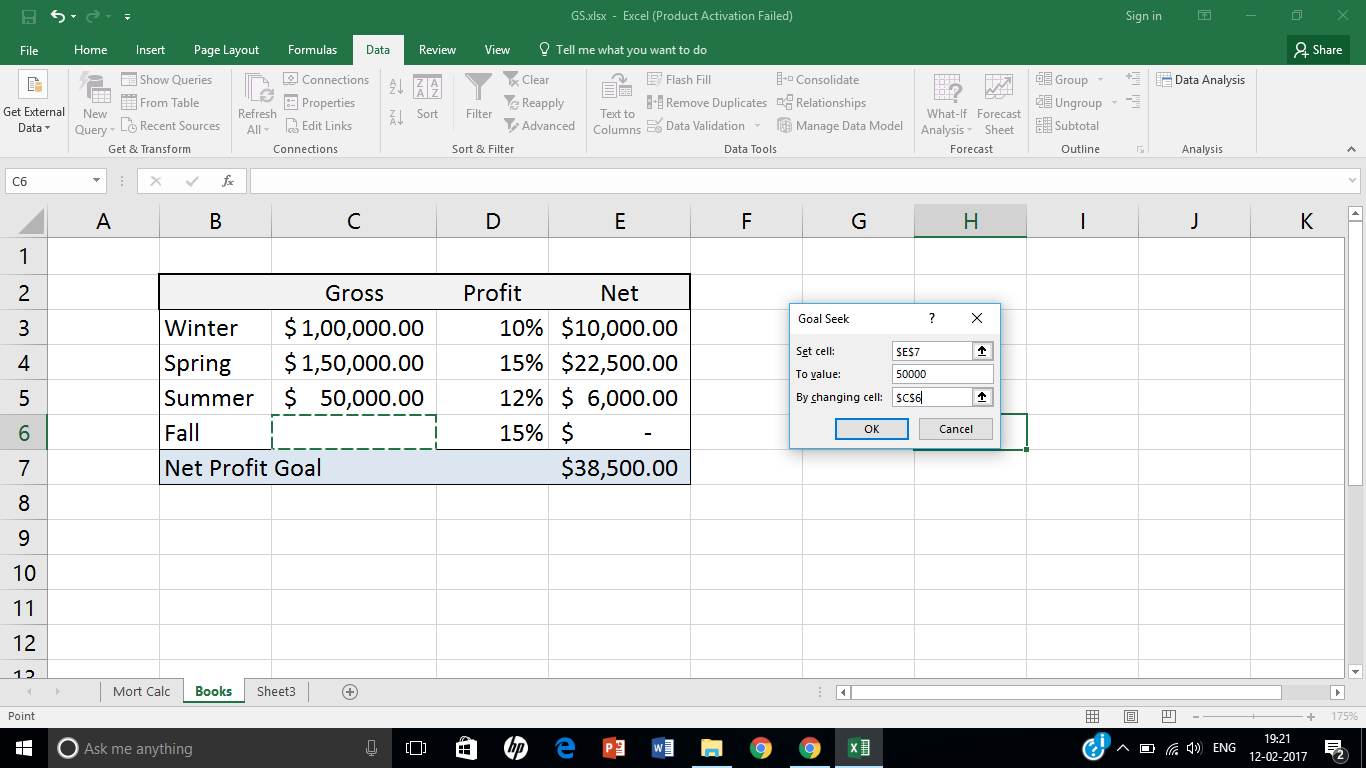
****

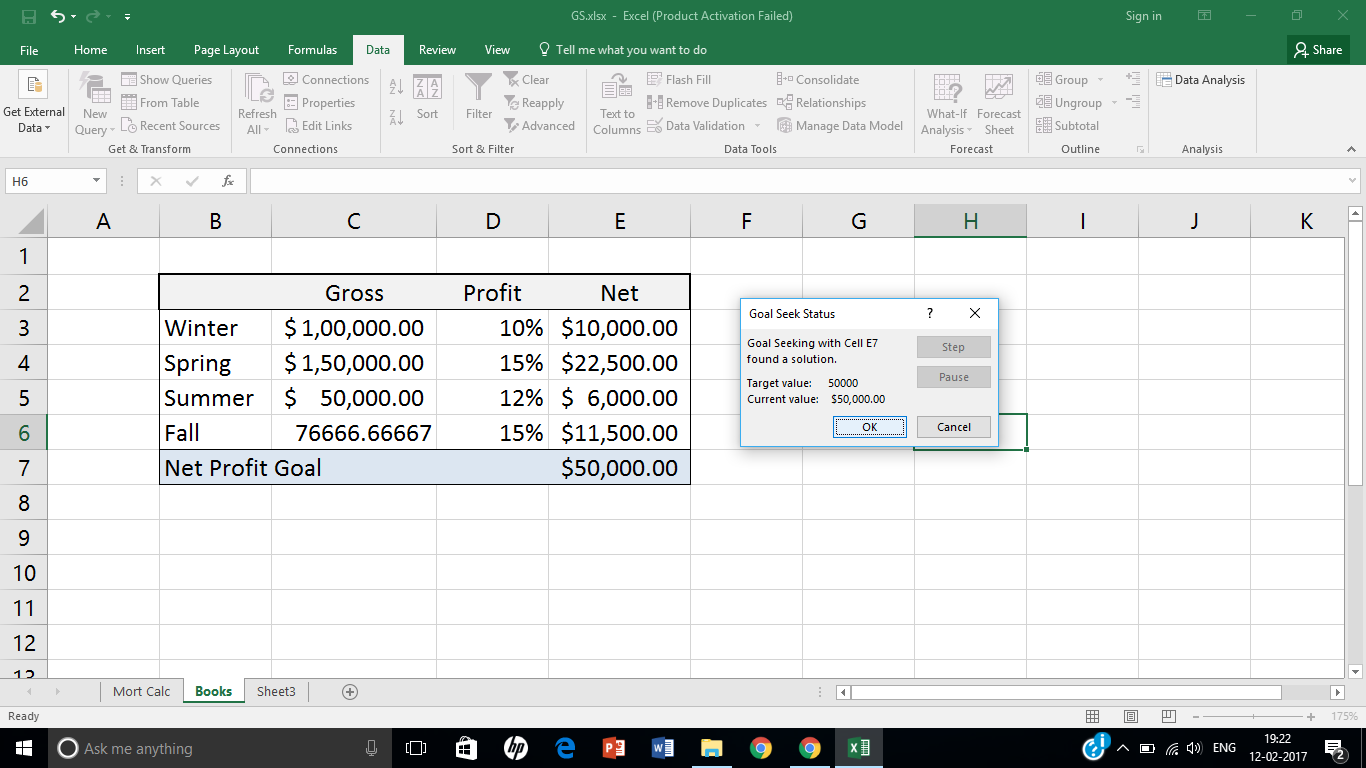
****

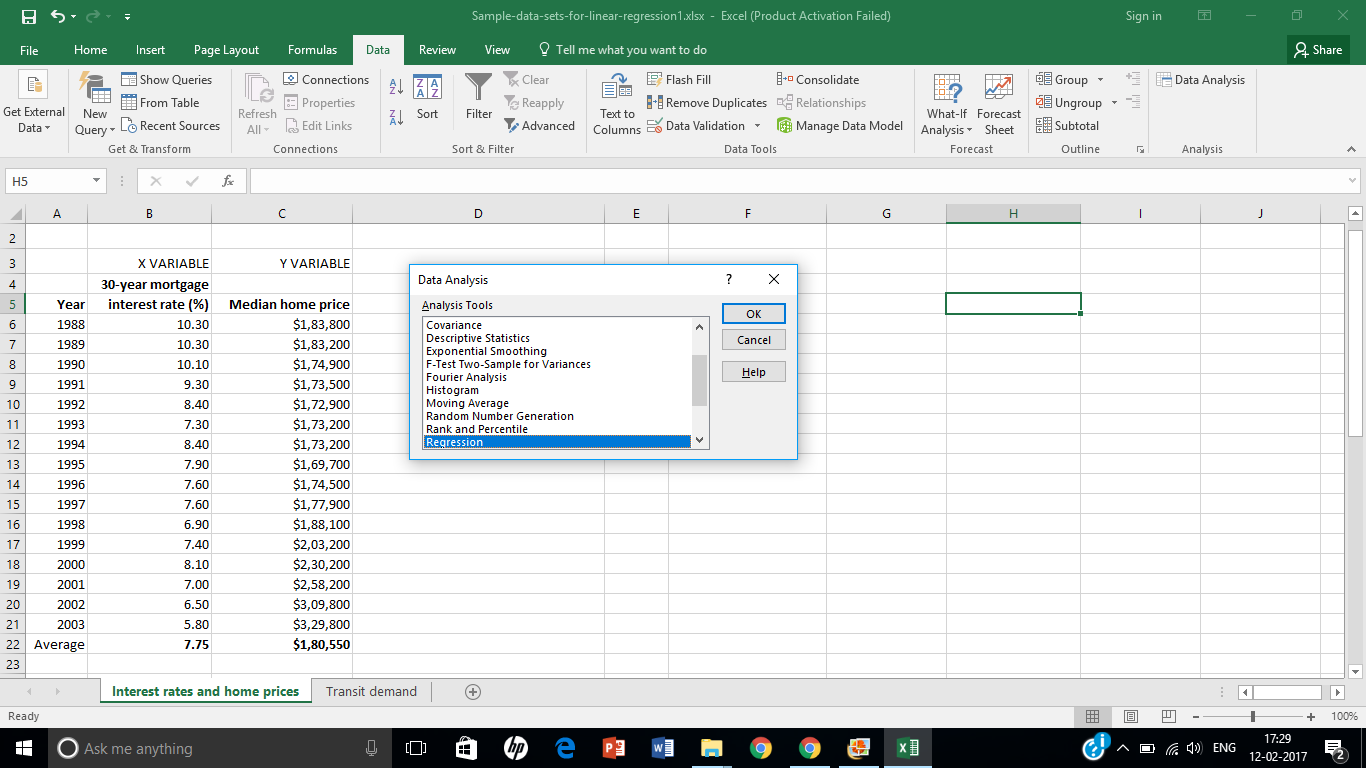
****

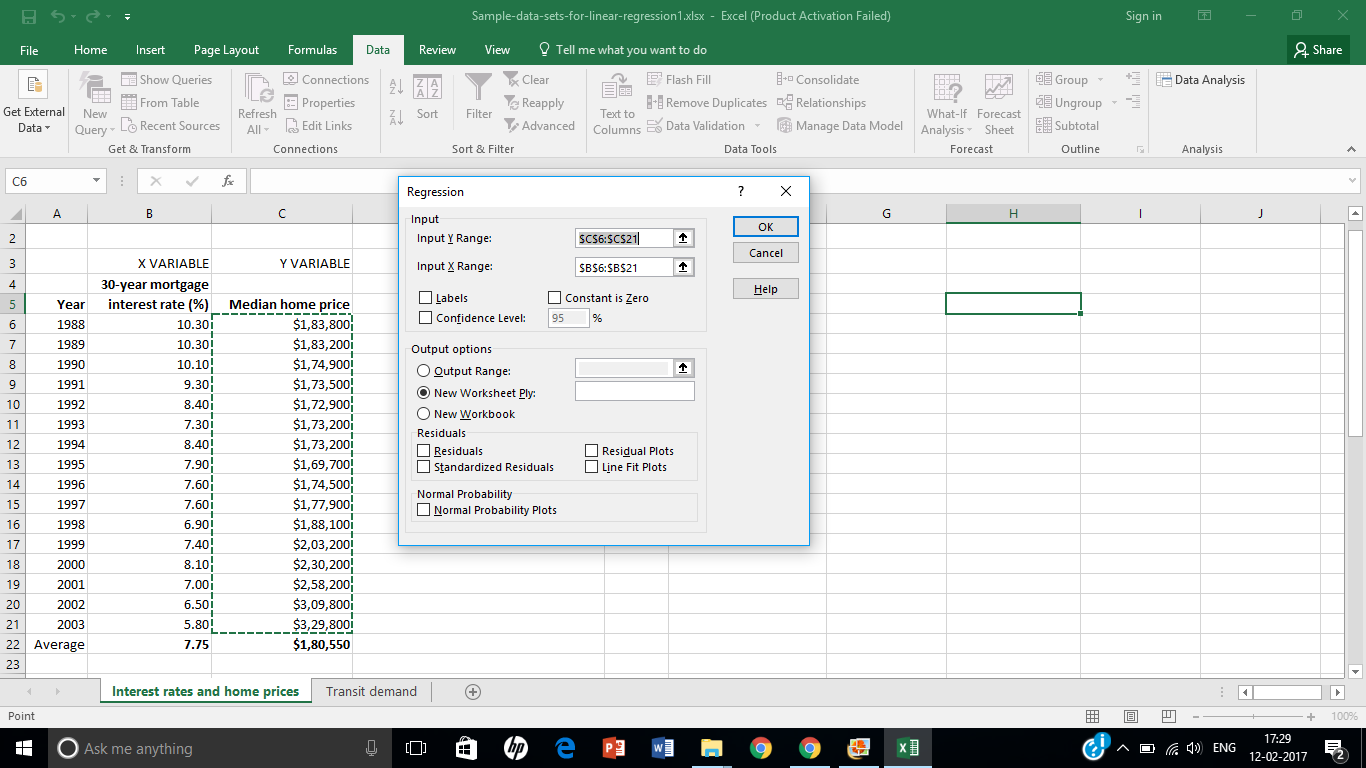
****

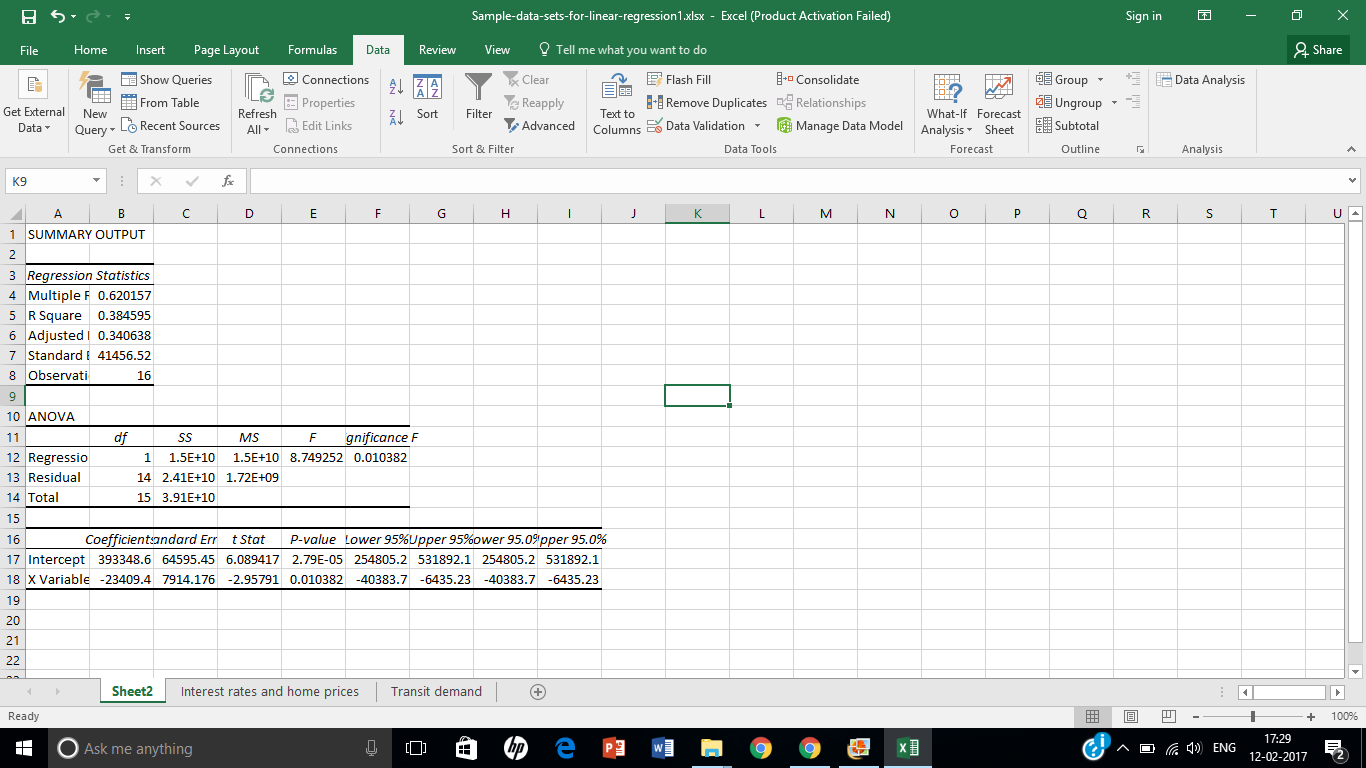
****

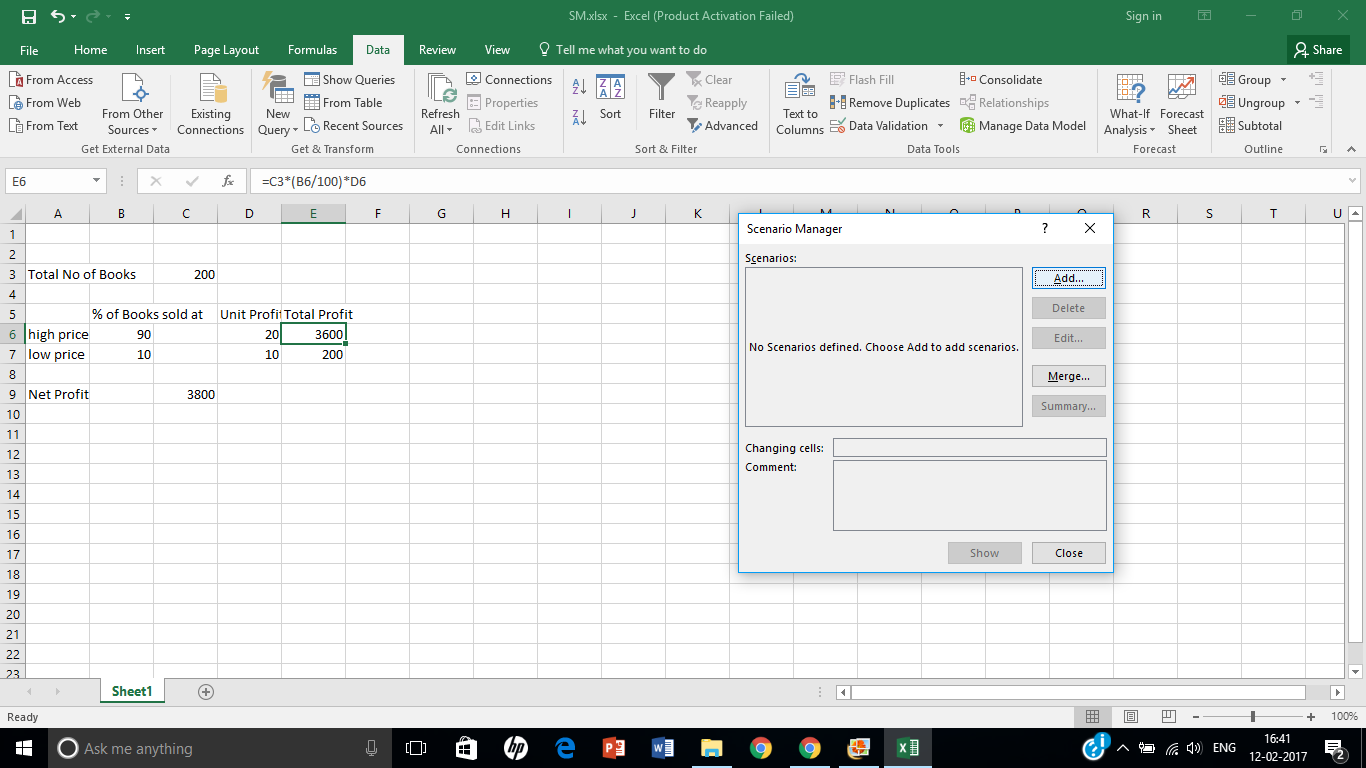
****

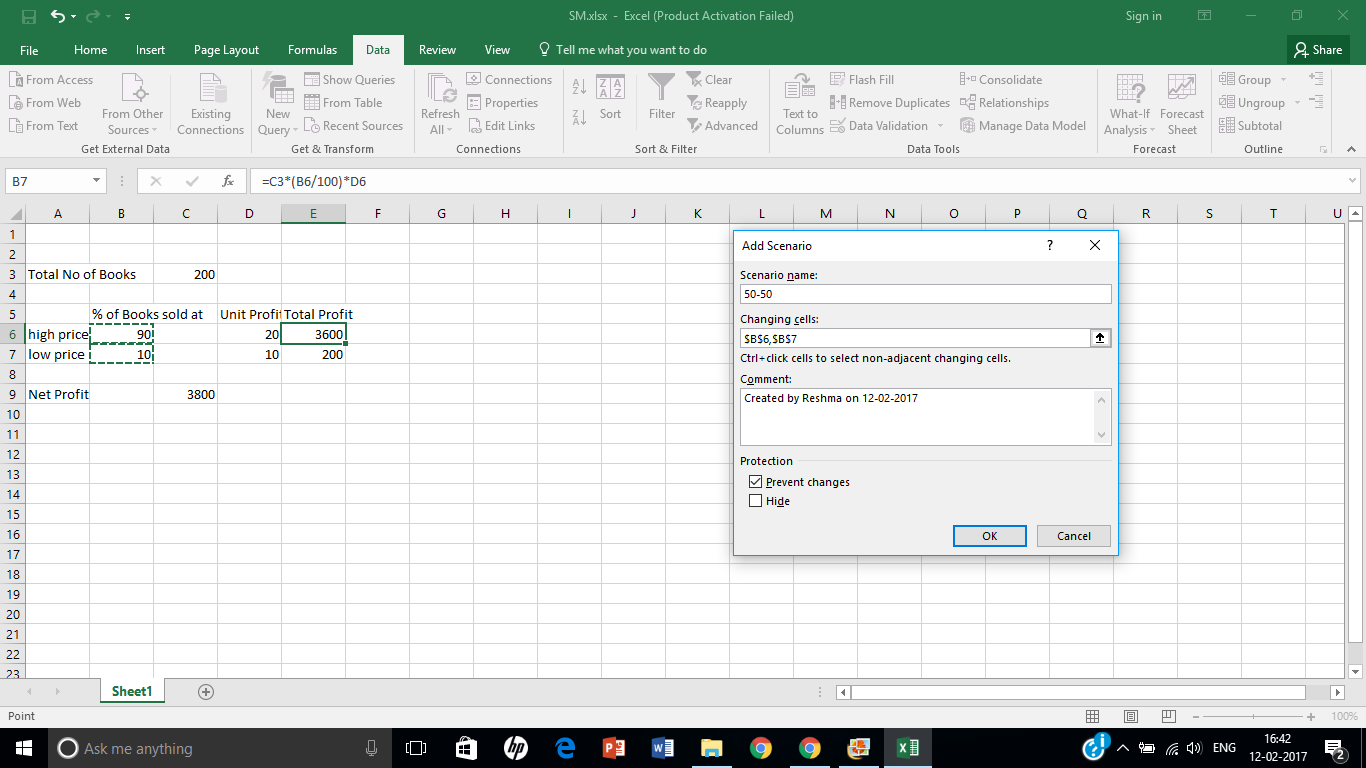
****

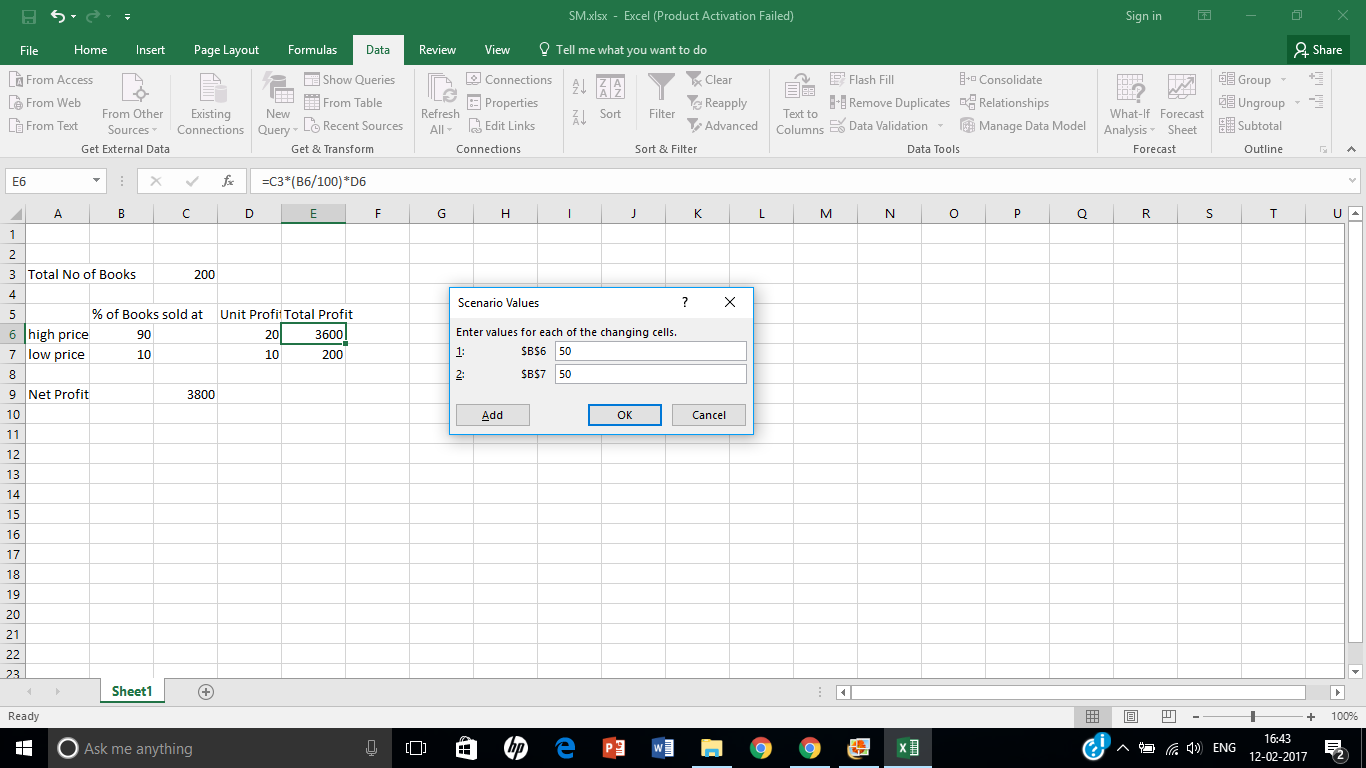
****

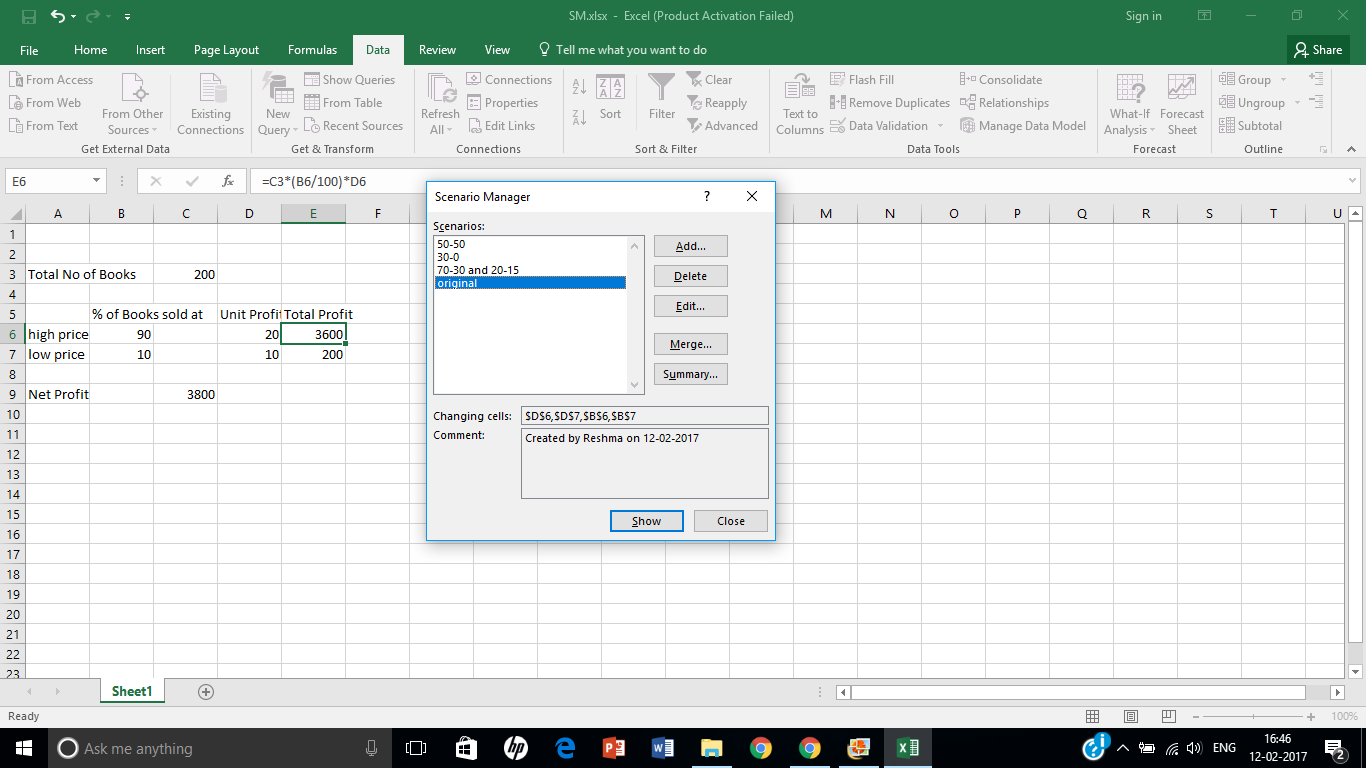
****

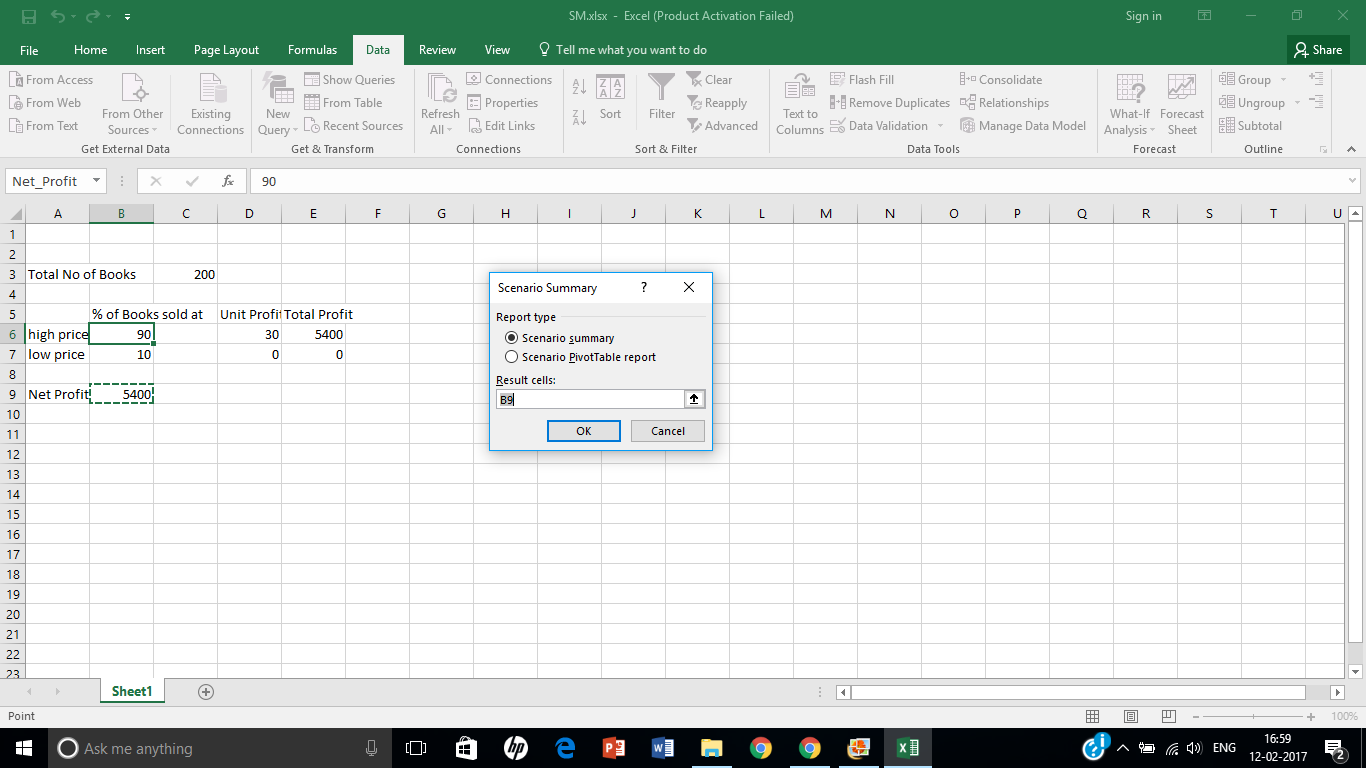
****

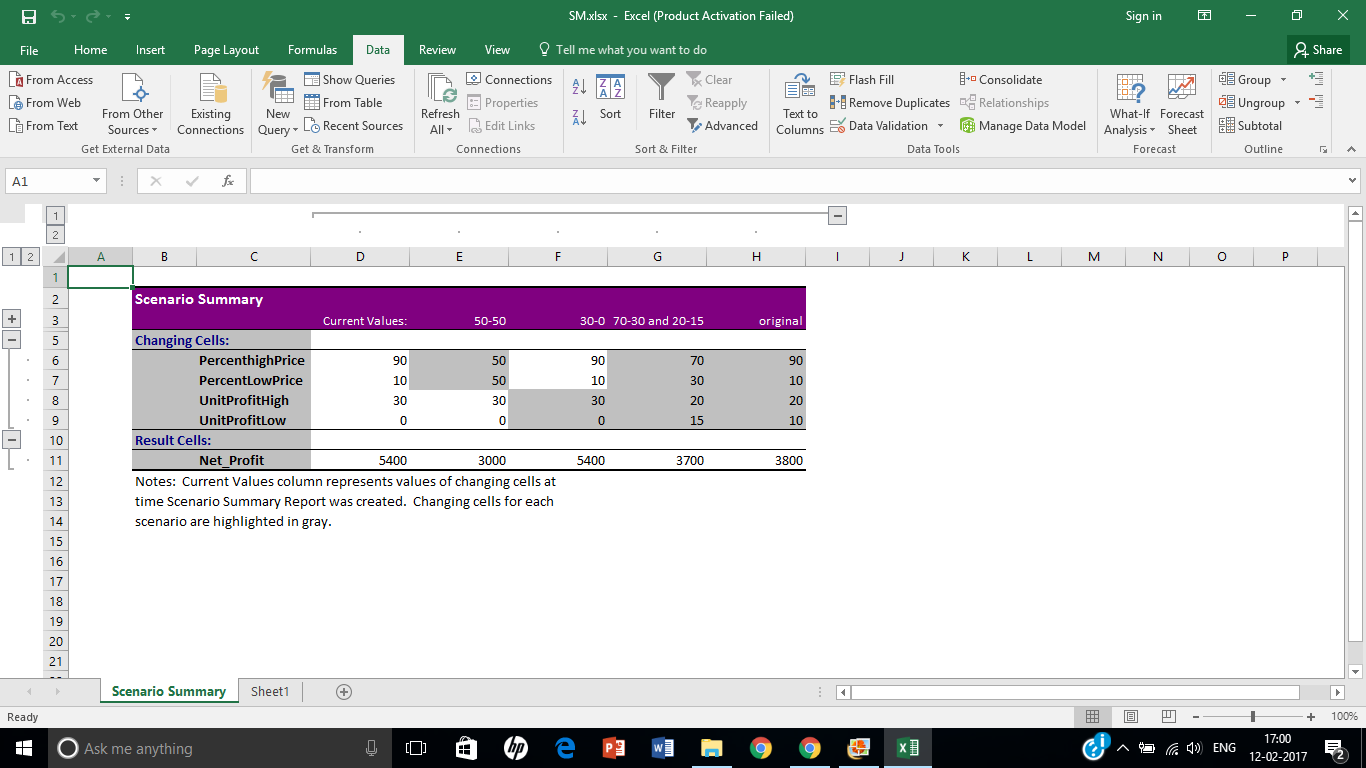
****

****

****

****

****

****

**Assignment 2**

**Title:** BI reporting Tool for BI application

**Aim:** Frame the suitable assignment to perform computing using BI tool effectively. (Generate BI Reports using any BI reporting Tool like Tableau)

**Objective**: To use BI reporting Tool for BI application

**Theory:**

**Business Intelligence Reporting Tools:**

These tools are used to create reports, [data visualizations](http://searchbusinessanalytics.techtarget.com/definition/data-visualization), and charts that can be embedded in Web-based and [rich client](http://whatis.techtarget.com/definition/rich-client) applications. Information gleaned from embedded BI tools such as BIRT may be used in both real-time decision making and to track and analyse historical data or ongoing developments. Examples of reports that can be designed using BIRT include lists, charts, crosstabs, forms and documents, and compound reports with multiple features.

These [business intelligence](http://searchdatamanagement.techtarget.com/definition/business-intelligence) and reporting tools can pull data from many different sources including databases, [Web services](http://searchsoa.techtarget.com/definition/Web-services), and Java objects to create a report. A single report may contain data from multiple sources. The data can be filtered and grouped to create standardized reports or customized based on the [end-user's](http://whatis.techtarget.com/definition/end-user) needs. Because the result may still include too much data for the purposes of decision making, [JavaScript](http://searchsoa.techtarget.com/definition/JavaScript) can be used to create [business-specific logic](http://whatis.techtarget.com/definition/business-logic) that makes reports more useful for business purposes. The resulting information can be presented in a variety of visual or text formats from [Excel](http://searchenterprisedesktop.techtarget.com/definition/Excel) spreadsheets and PowerPoint presentations to [PDF](http://whatis.techtarget.com/definition/Portable-Document-Format-PDF) documents.

**Tableau**

Tableau is a Business Intelligence tool for visually analysing the data. Users can create and distribute interactive and shareable dashboards which depict the trends, variations and density of the data in form of graphs and charts. Tableau can connect to files, relational and Big data sources to acquire and process data. The software allows data blending and real time collaboration, which makes it very unique. It is used by businesses, academic researchers and many governments to do visual data analysis. It is also positioned as a leader Business Intelligence and Analytics Platform in Gartner Magic Quadrant.

As a leading data visualization tool Tableau has many desirable and unique features. Its powerful data discovery and exploration application allows you to answer important questions in seconds. You can use Tableau's drag and drop interface to visualize any data, explore different views, and even combine multiple databases together easily. It does not need any complex scripting. Anyone who understands the business problem can address it with a visualization of the relevant data. When the analysis is finished, sharing with others is as easy as publishing to Tableau Server.

**Tableau Features**

Tableau provides solutions for all kinds of industries, departments and data environments. Below are the unique features which enable tableau handle so many diverse scenarios.

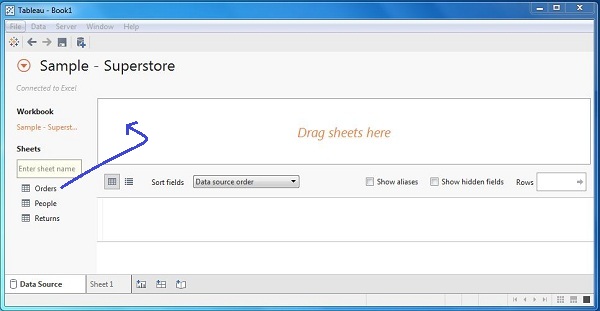
* **Speed of Analysis** - As it does not need high level of programming expertise, any computer user with access to data can start using it to derive value from the data.
* **Self-Reliant** - Tableau does not need a complex software setup. The desktop version which is used by most users is easily installed and contains all the features needed to start and complete data analysis.
* **Visual Discovery** - The user explores and analyses the data by using visual tools like colours, trend lines, charts and graphs. There is very little script to be written as nearly everything is done by drag and drop.
* **Blend Diverse Data Sets** - Tableau allows you to blend different relational, semi-structured and raw data sources in real time, without expensive up-front integration costs. The users don’t need to know the details of how data is stored.
* **Architecture Agnostic** - Tableau works in all kinds of devices where data flows. So the user need not worry about specific hardware or software requirements to use Tableau.
* **Real Time Collaboration** - Tableau can filter, sort, and discuss data on the fly and embed a live dashboard in portals like SharePoint site or Salesforce. You can save your view of data and allow colleagues to subscribe to your interactive dashboards so they see the very latest data just by refreshing their web browser.
* **Centralized Data** - The tableau server provides a centralized location to manage all of the organization’s published data sources. You can delete, change permissions, add tags, and manage schedules in one convenient location. It’s easy to schedule extract refreshes and manage them in the data server. Administrators can centrally define a schedule for extracts on the server for both incremental and full refreshes.

**Explanation of all steps for successful implementation:**

There are three basic steps involved in creating any Tableau data analysis report.

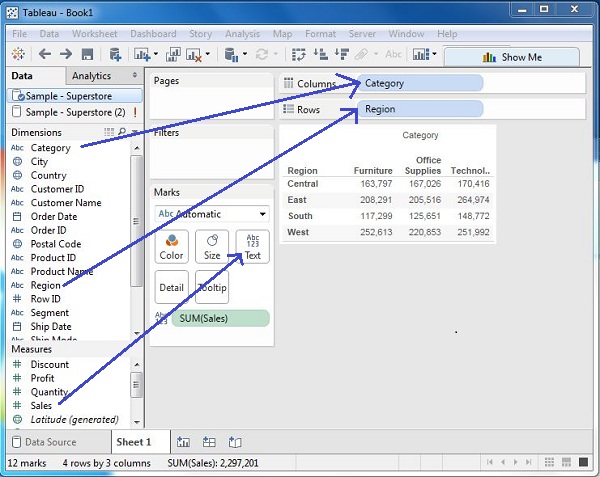
## Connect to a Data Source

One opening Tableau we get the start page Showing various data sources. Under the header Connect, we have options to choose a file or server or saved data source. Under Files we choose excel. Then navigate to the file “Sample – Superstore.xls” as mentioned above. The excel file has three sheets named orders, people and Returns. We choose Orders.



## Choose the Dimensions and Measures

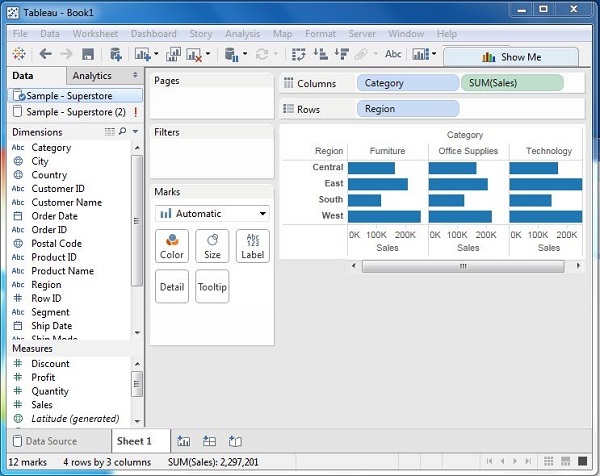
Next we choose the data to be analyzed by deciding on the dimensions and measures. Dimensions are the descriptive data while measures are numeric data. When put together, they help us visualize the performance of the dimensional data with respect to the data which are measures. We choose category and region as the dimensions and sales as the measure. Drag and drop them as shown below. The result shows the total sales in each category for each region.



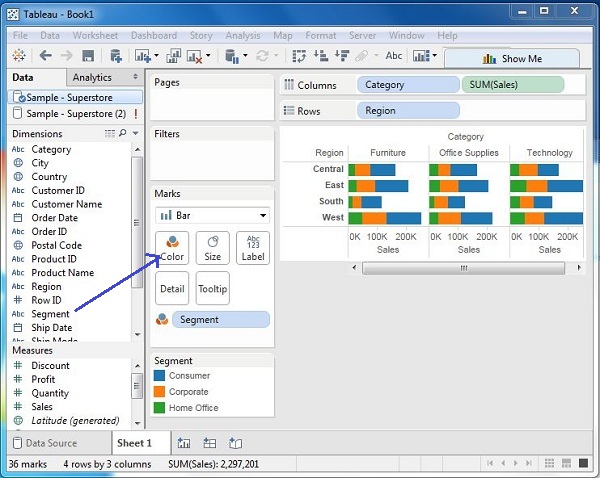
## Apply Visualization Technique

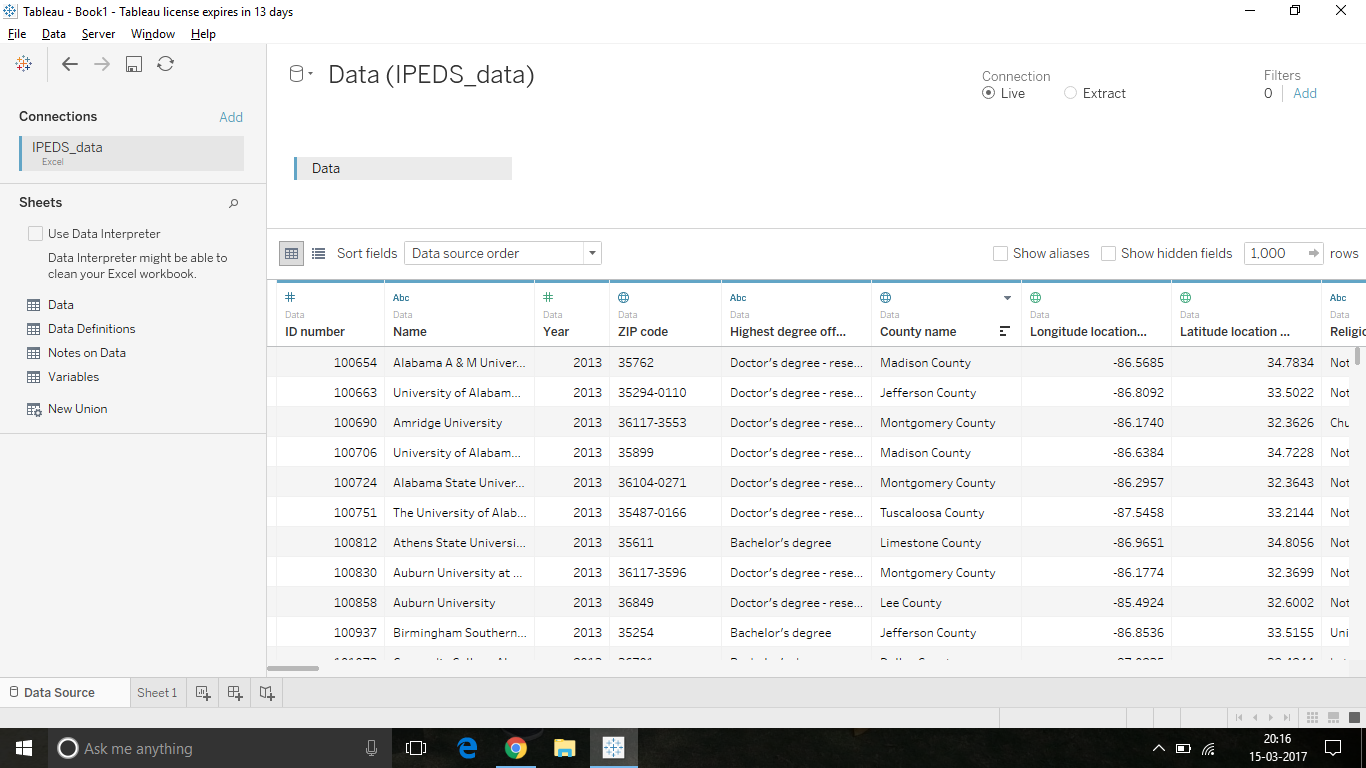
In the previous step we see that the data is available only as numbers. We have to read and calculate each of the values to judge the performance. But we can see them as graphs or charts with different colours to get a quicker judgment.

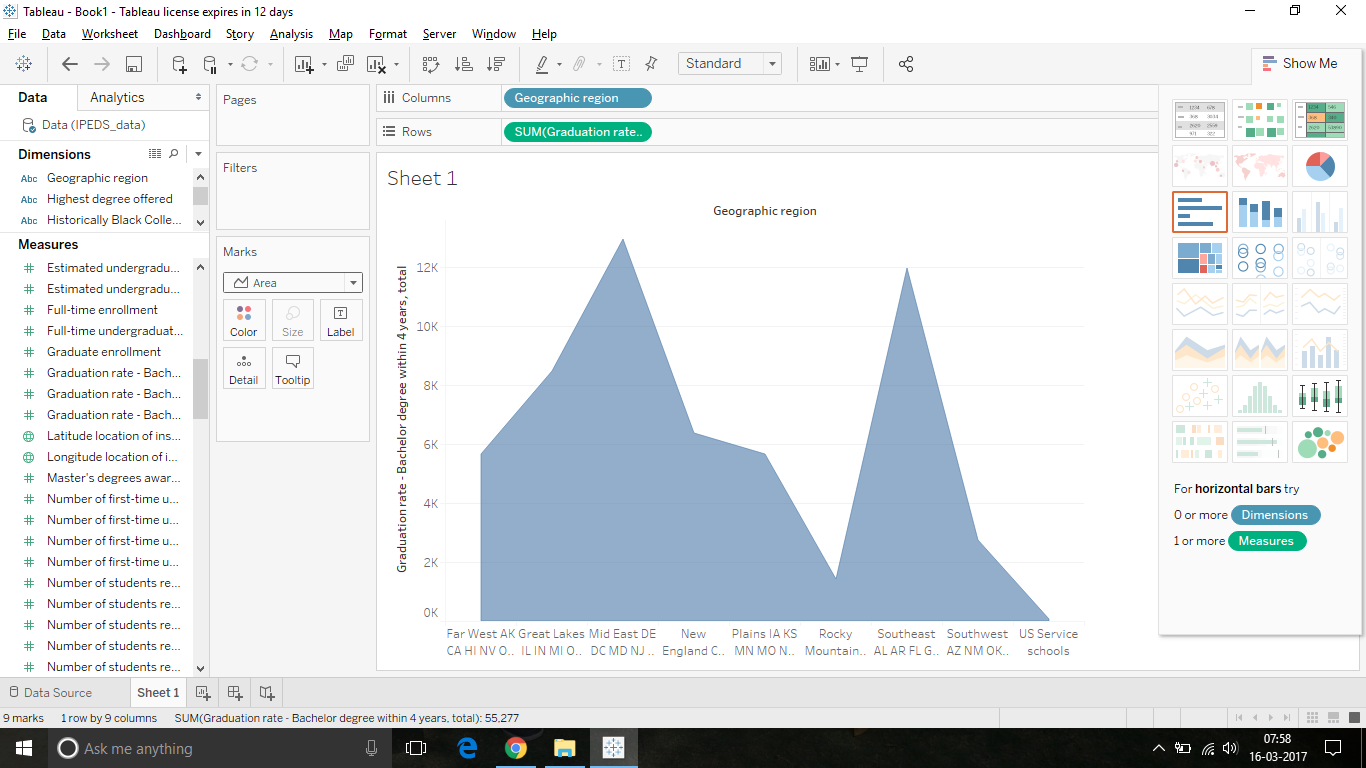
We drag and drop the sum(sales) column from the Marks tab to the Columns shelf. The table showing the numeric values of sales now turns into a bar chart automatically.

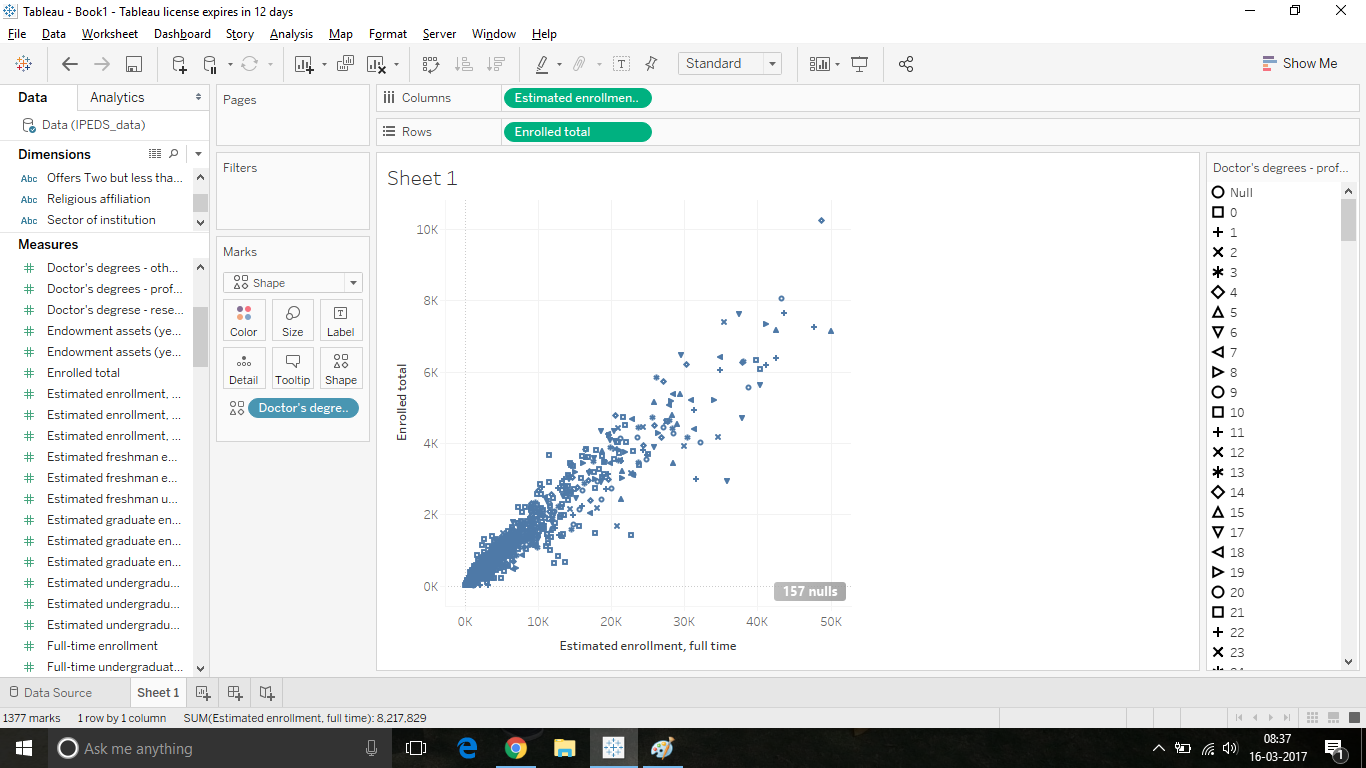


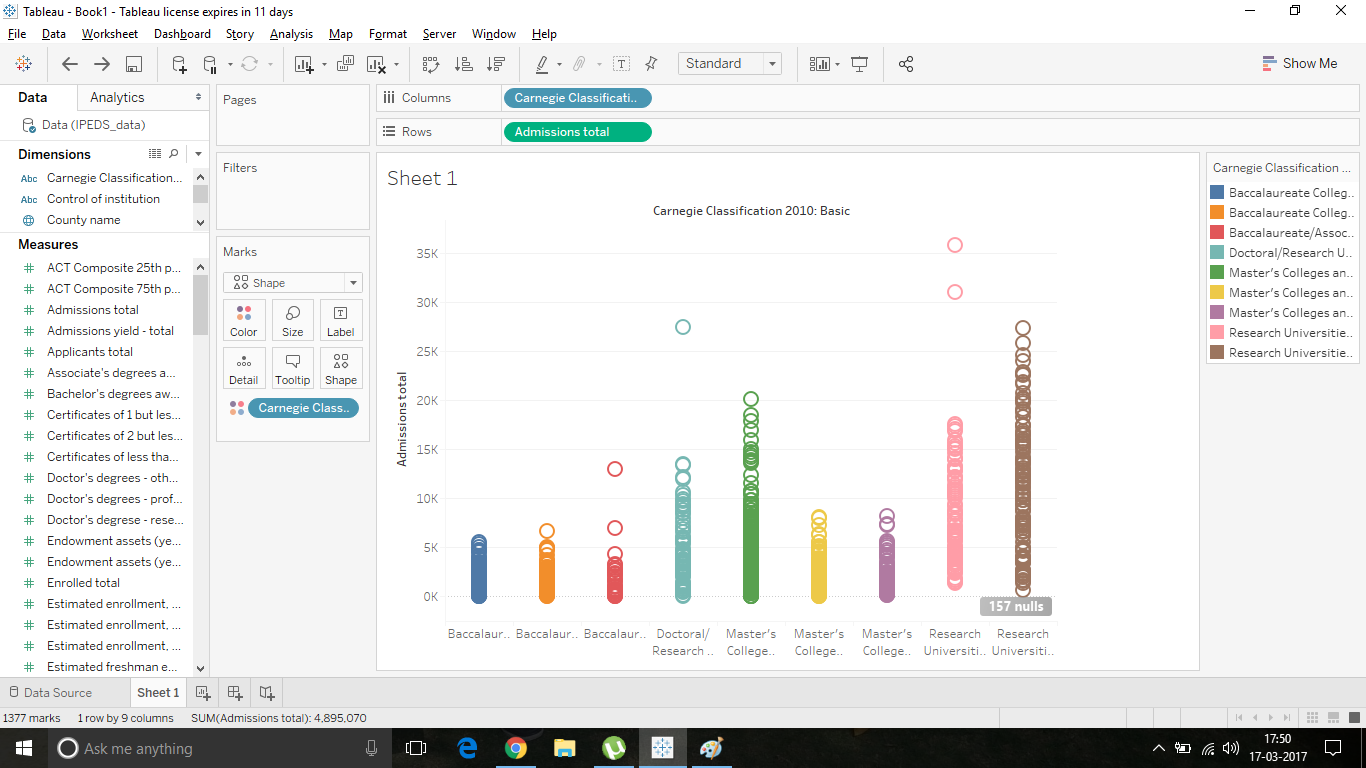
We can apply a further technique of adding another dimension to the existing data and that will add more colours to the existing bar chart as shown below.

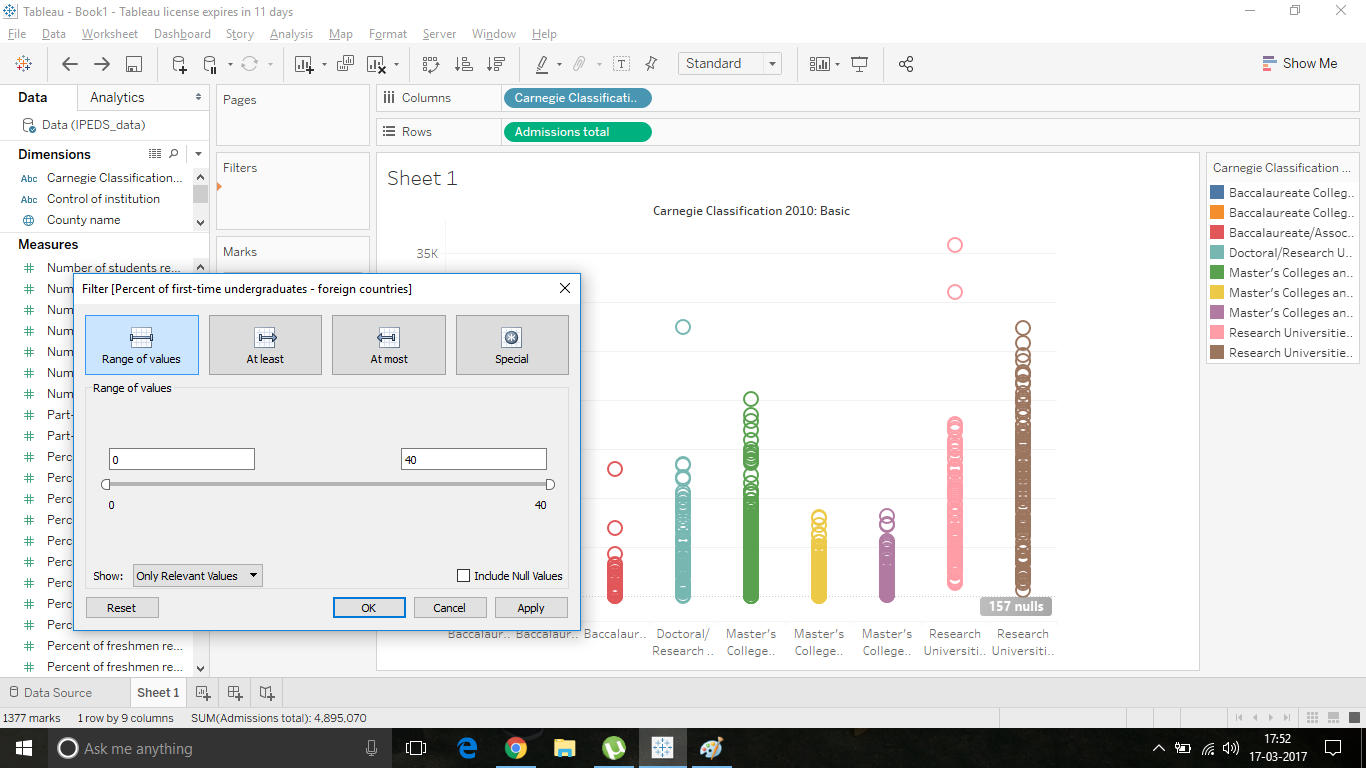


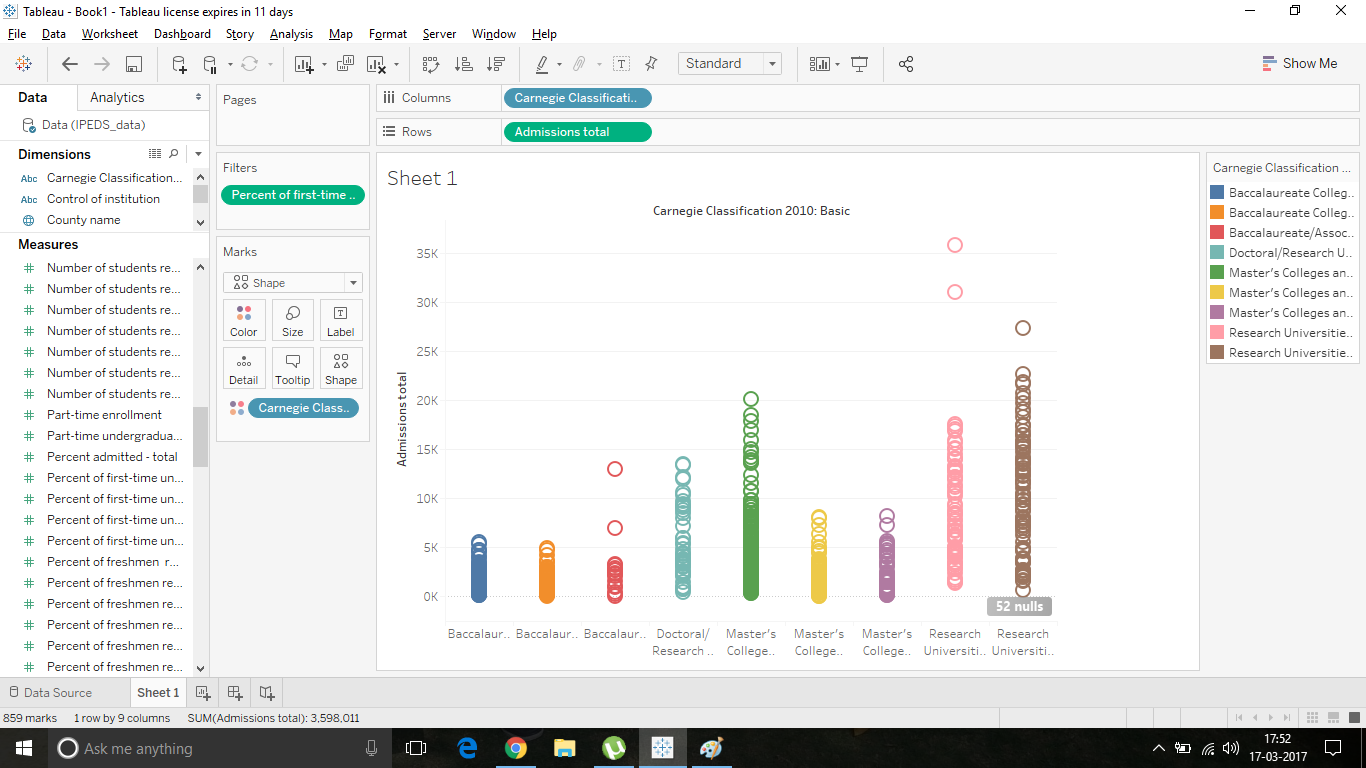
****

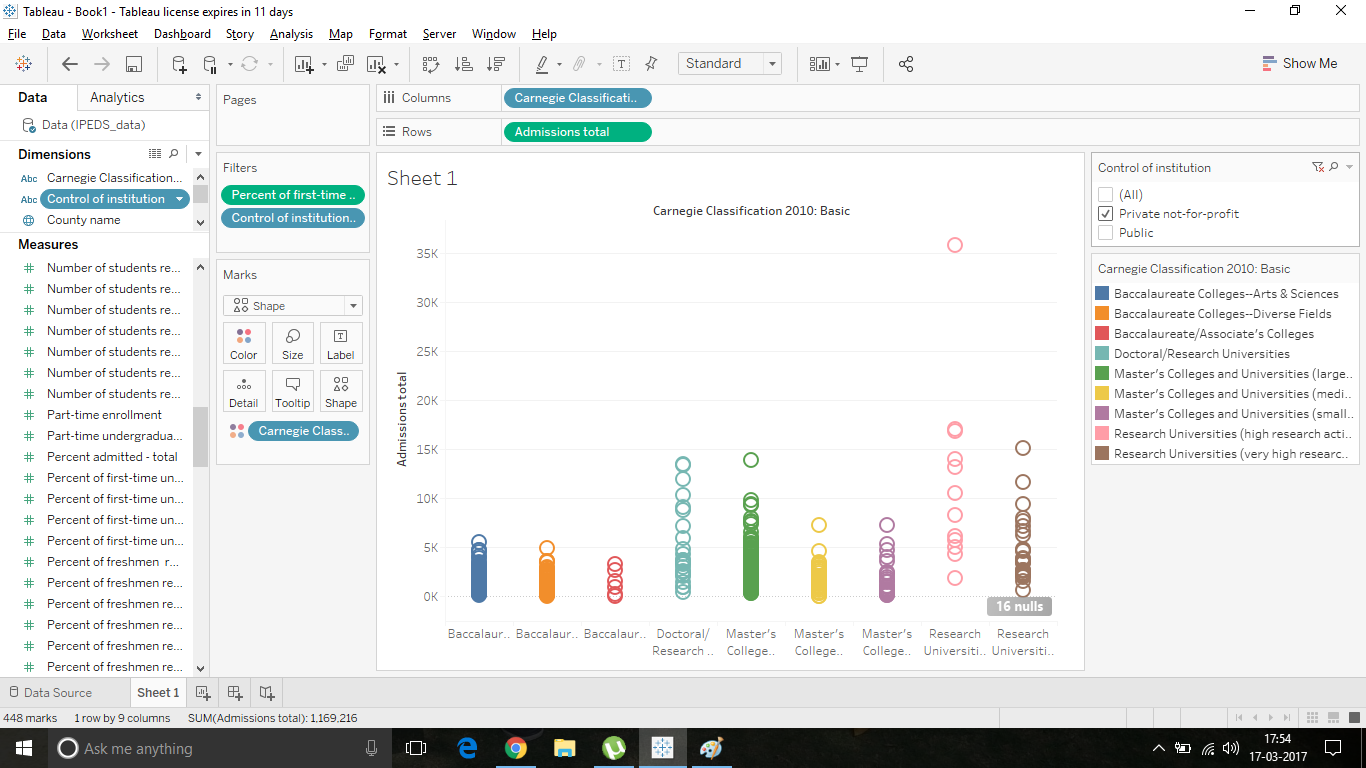
****

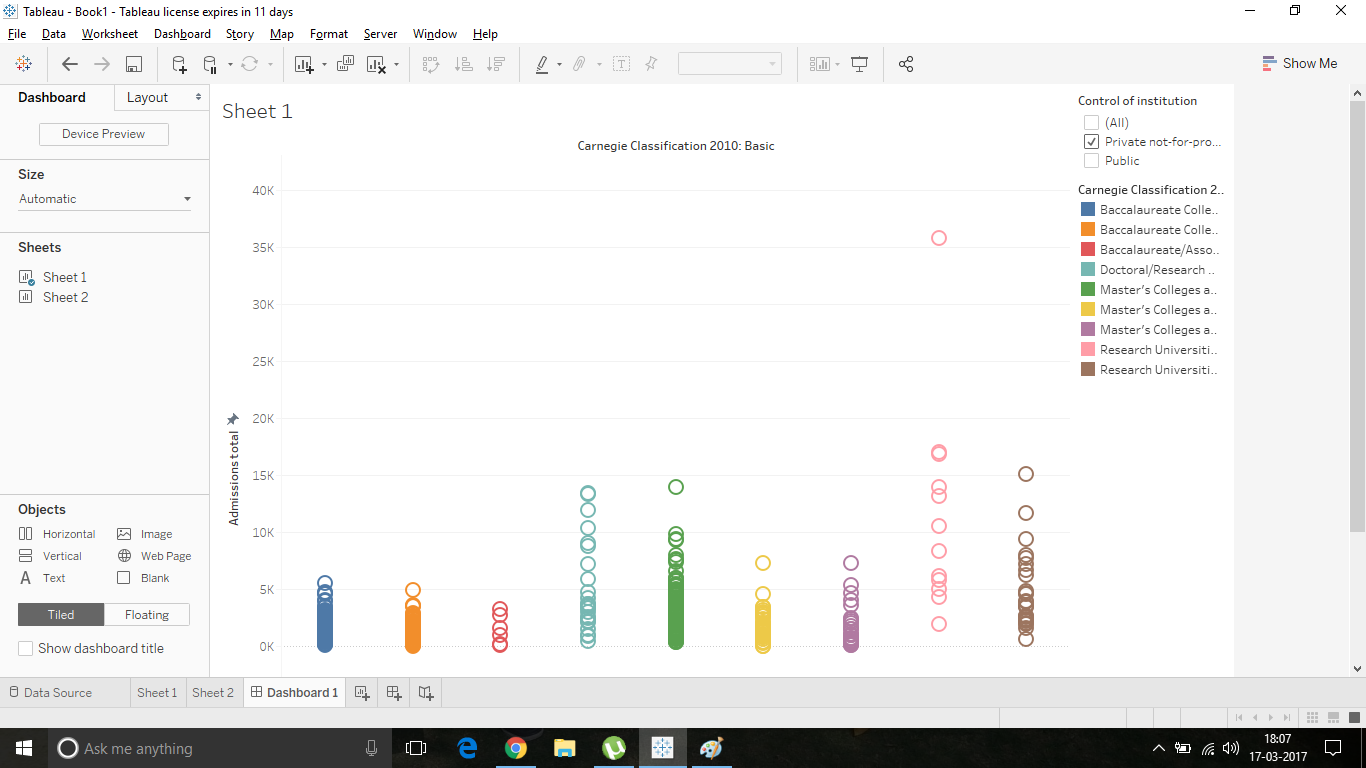
****

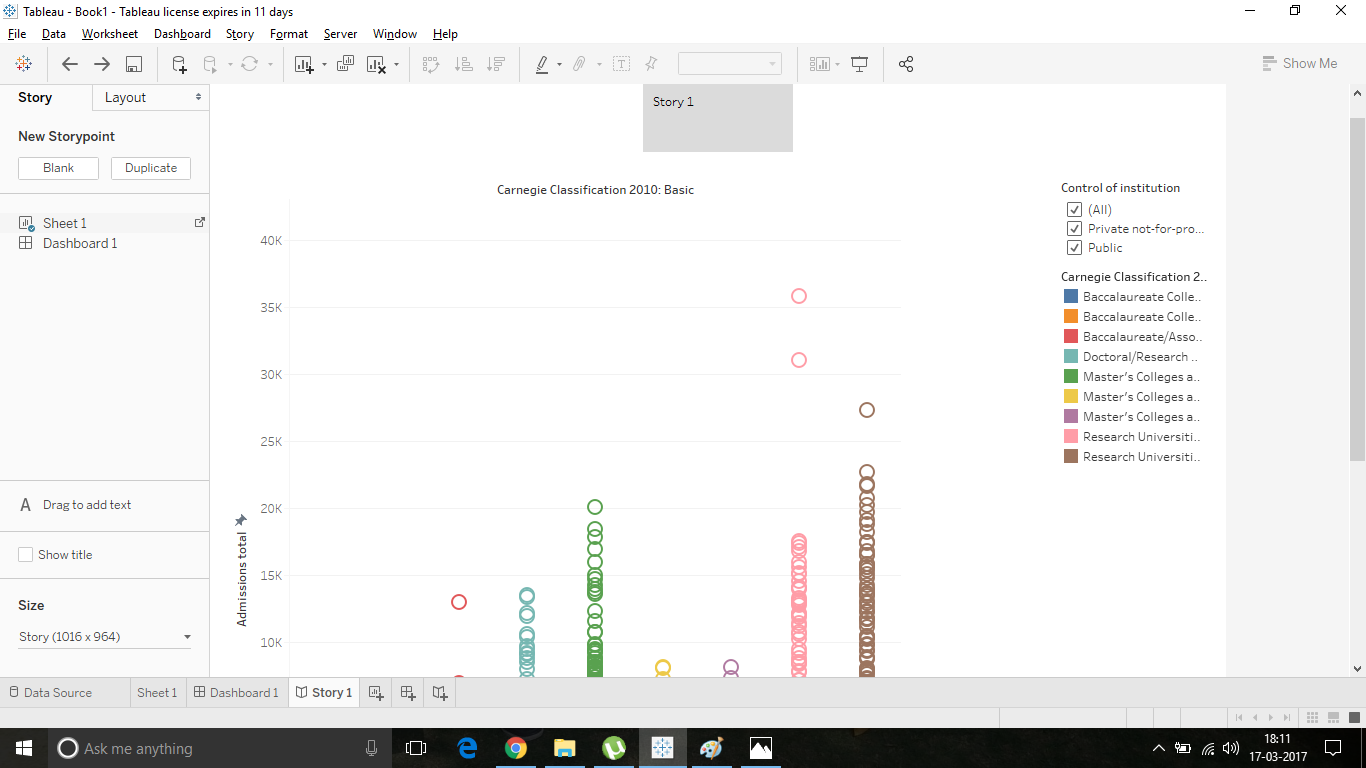
****

****

****

****

****

****

**Assignment 3**

**Title:** Predictive Analytics using R programming

**Aim :** Design a business intelligence application using any BI tool to recommend the combination of share purchases and sales for maximizing the profit. (Predict the cost of the share and use this data to take decision as to buy, sell or hold share for maximizing the profit.)

**Objective**: To implement Predictive Analytics using R programming

**Theory:**

**ARIMA Mod****el technique**

In [statistics](https://en.wikipedia.org/wiki/Statistics) and [econometrics](https://en.wikipedia.org/wiki/Econometrics), and in particular in [time series analysis](https://en.wikipedia.org/wiki/Time_series_analysis), an **autoregressive integrated moving average (ARIMA)** [model](https://en.wikipedia.org/wiki/Mathematical_model) is a generalization of an [autoregressive moving average](https://en.wikipedia.org/wiki/Autoregressive_moving_average) (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](https://en.wikipedia.org/wiki/Forecasting)). ARIMA models are applied in some cases where data show evidence of [non-stationarity](https://en.wikipedia.org/wiki/Stationary_process), where an initial differencing step (corresponding to the ["integrated"](https://en.wikipedia.org/wiki/Order_of_integration) part of the model) can be applied one or more times to eliminate the non-stationarity.[[1]](https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average" \l "cite_note-1)

The AR part of ARIMA indicates that the evolving variable of interest is regressed on its own lagged (i.e., prior) values. The MA part indicates that the [regression error](https://en.wikipedia.org/wiki/Errors_and_residuals_in_statistics) is actually a [linear](https://en.wikipedia.org/wiki/Linear_combination) combination of error terms whose values occurred contemporaneously and at various times in the past. The I (for "integrated") indicates that the data values have been replaced with the difference between their values and the previous values (and this differencing process may have been performed more than once). The purpose of each of these features is to make the model fit the data as well as possible.

Non-seasonal ARIMA models are generally denoted ARIMA(*p*,*d*,*q*) where parameters *p*, *d*, and *q* are non-negative integers, *p* is the order (number of time lags) of the [autoregressive model](https://en.wikipedia.org/wiki/Autoregressive_model), *d* is the degree of differencing (the number of times the data have had past values subtracted), and *q* is the order of the [moving-average model](https://en.wikipedia.org/wiki/Moving-average_model). Seasonal ARIMA models are usually denoted ARIMA(*p*,*d*,*q*)(*P*,*D*,*Q*)*m*, where *m* refers to the number of periods in each season, and the uppercase *P*,*D*,*Q* refer to the autoregressive, differencing, and moving average terms for the seasonal part of the ARIMA model.

When two out of the three terms are zeros, the model may be referred to based on the non-zero parameter, dropping "AR", "I" or "MA" from the acronym describing the model. For example, ARIMA (1,0,0) is AR(1), ARIMA(0,1,0) is I(1), and ARIMA(0,0,1) is MA(1).

ARIMA models can be estimated following the Box Jenkins approach.

Some well-known special cases arise naturally or are mathematically equivalent to other popular forecasting models. For example:

* An ARIMA(0,1,0) model (or I(1) model) is given by **X**t =**X**t+1 + έ— which is simply a [random walk](https://en.wikipedia.org/wiki/Random_walk).
* An ARIMA(0,1,0) with a constant, given by **X**t = **C**+ **X**t+1 + έ — which is a random walk with drift.
* An ARIMA(0,0,0) model is a [white noise](https://en.wikipedia.org/wiki/White_noise) model.
* An ARIMA(0,1,2) model is a Damped Holt's model.
* An ARIMA(0,1,1) model without constant is a [basic exponential smoothing](https://en.wikipedia.org/wiki/Exponential_smoothing) model.[[4]](https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average" \l "cite_note-:0-4)

**Explanation of all the commands of the source code in software tool used for the implementation of the given assignment.**

* **getSymbols(NAME OF STOCK, Source , From Date, To Date) :** This function under the quantmod package is used to obtain information about shares in the stock market from the mentioned source with specified to and from dates.
* **write.csv()**: uploading contents to a csv file from dataframe
* **read.csv()**: uploading contents of csv file to user defined object
* **plot(data for X axis, data for y axis, name of x axis, name of y axis,type of graph to be plotted, name of entire graph) :** This function under ggplot2 package is used to plot contents of data obtained on a graph and display it.
* **adf.test(data) :** Used to perform augmented dickey fuller test on given data as inout parameter to check validity of null hypothesis of unit root
* **arima(training model, order ) :** This function under forecast package is used to form the arima model based on the training model as input parameter with given order.
* **plot(arima,name): function to plot the data of arima model with mentioned name**

**Mathematical Model:**

T ={i,o,f,sc,fc }

where..

i= set of inputs

o=set of outputs

f=set of functions

Sc=set of success cases

Fc= set of failure cases

i= {e}

where e= data required for building ARIMA model

o={o1, o2,o3..}

where,

o1 to on are required predictions with accuracy mentioned.

Sc={Sc1, Sc2}

where

Sc1= Data input is in proper format

Sc2=Accuracy level is high enough

Fc= {Fc1,Fc2}

where

Fc1= Required package is not installed

Fc2=Accuracy is low

**Input**:

Data file

**Output:**

Data prediction

#install quantmod package

library('quantmod')

#get information of Infosys shares from yahoo as xst file using quantmod package

getSymbols("INFY",src="google" , from ="2017-02-14", to = "2017-03-14")

#convert xst file to dataframe

test<-as.data.frame(INFY)

write.csv(test,file="infy1.csv")

data<-read.csv("infy1.csv")

summary(data)

#install ggplot2 package

library(ggplot2, quietly = T)

plot(data$X, data$INFY.Close, xlab = "Dates", ylab = "Adjusted closing price",type = "l", col = "red", main = "Adjusted closing price of INFOSYS for past 1 month")

library(tseries, quietly = T) #time series analysis

adf.test(data$INFY.Close)#dickey-fuller test to convert non-stationary to stationary data

infy\_ret <- 100 \* diff(log(data$INFY.Close)) #function for stationary data

summary(arma(infy\_ret, order = c(2, 2)))

library(tseries, quietly = T)

infy\_ret\_train <- infy\_ret[1:(0.9 \* length(infy\_ret))] #training data

infy\_ret\_test <- infy\_ret[(0.9 \* length(infy\_ret) + 1):length(infy\_ret)] #testing data

fit <- arima(infy\_ret\_train, order = c(2, 0, 2)) #defining arima model

arma.preds <- predict(fit, n.ahead = (length(infy\_ret) - (0.9 \* length(infy\_ret))))$pred

#install forecast package

library(forecast, quietly = T)

arma.forecast <- forecast(fit, h = 25)

summary(arma.forecast)

plot(arma.forecast, main = "ARMA forecasts for INFOSYS returns")

> summary(data)

X INFY.Open INFY.High INFY.Low INFY.Close INFY.Volume

2017-02-14: 1 Min. :14.76 Min. :14.96 Min. :14.70 Min. :14.86 Min. : 2850774

2017-02-15: 1 1st Qu.:15.03 1st Qu.:15.17 1st Qu.:14.95 1st Qu.:15.02 1st Qu.: 4342951

2017-02-16: 1 Median :15.19 Median :15.27 Median :15.07 Median :15.20 Median : 5428752

2017-02-17: 1 Mean :15.18 Mean :15.28 Mean :15.08 Mean :15.19 Mean : 5937413

2017-02-21: 1 3rd Qu.:15.29 3rd Qu.:15.38 3rd Qu.:15.19 3rd Qu.:15.31 3rd Qu.: 7537163

2017-02-22: 1 Max. :15.63 Max. :15.68 Max. :15.51 Max. :15.59 Max. :11206982

(Other) :14

Residuals:

Min 1Q Median 3Q Max

-1.5770 -0.6574 -0.5770 0.1974 1.9812

Coefficient(s):

Estimate Std. Error t value Pr(>|t|)

ar1 1.0627 NA NA NA

ar2 1.5330 NA NA NA

ma1 -1.1381 NA NA NA

ma2 -1.4233 NA NA NA

intercept -0.5244 NA NA NA

Fit:

sigma^2 estimated as 1.11, Conditional Sum-of-Squares = 18.21, AIC = 65.91

Coefficients:

ar1 ar2 ma1 ma2 intercept

-0.418 0.5158 -0.1556 -0.8444 0.1688

s.e. 0.305 0.2664 0.3117 0.2997 0.0784

sigma^2 estimated as 0.8361: log likelihood = -23.73, aic = 59.46

Error measures:

ME RMSE MAE MPE MAPE MASE ACF1

Training set 0.03694301 0.9143609 0.703174 -Inf Inf 0.4476938 0.06660602

Forecasts:

Point Forecast Lo 80 Hi 80 Lo 95 Hi 95

18 -0.067139163 -1.267376 1.133097 -1.902742 1.768464

19 -0.270504682 -1.626390 1.085380 -2.344152 1.803142

20 0.230702275 -1.127133 1.588538 -1.845928 2.307333

21 -0.083703945 -1.472501 1.305093 -2.207686 2.040278

22 0.306252801 -1.084705 1.697211 -1.821034 2.433539

23 -0.018928010 -1.421952 1.384096 -2.164668 2.126812

24 0.318147065 -1.089768 1.726062 -1.835073 2.471367

25 0.009513112 -1.405726 1.424752 -2.154908 2.173934

26 0.312393960 -1.107976 1.732764 -1.859874 2.484662

27 0.026588520 -1.399236 1.452413 -2.154022 2.207199

28 0.302288819 -1.128110 1.732688 -1.885317 2.489895

29 0.039620385 -1.395161 1.474401 -2.154688 2.233928

30 0.291629004 -1.147047 1.730305 -1.908637 2.491895

31 0.050798352 -1.391491 1.493088 -2.154993 2.256589

32 0.281458009 -1.164104 1.727020 -1.929338 2.492254

33 0.060815706 -1.387752 1.509383 -2.154577 2.276209

34 0.272024296 -1.179283 1.723331 -1.947558 2.491607

35 0.069926202 -1.383889 1.523742 -2.153493 2.293345

36 0.263349960 -1.192757 1.719457 -1.963574 2.490274

37 0.078251500 -1.379952 1.536455 -2.151879 2.308382

38 0.255395543 -1.204725 1.715516 -1.977666 2.488457

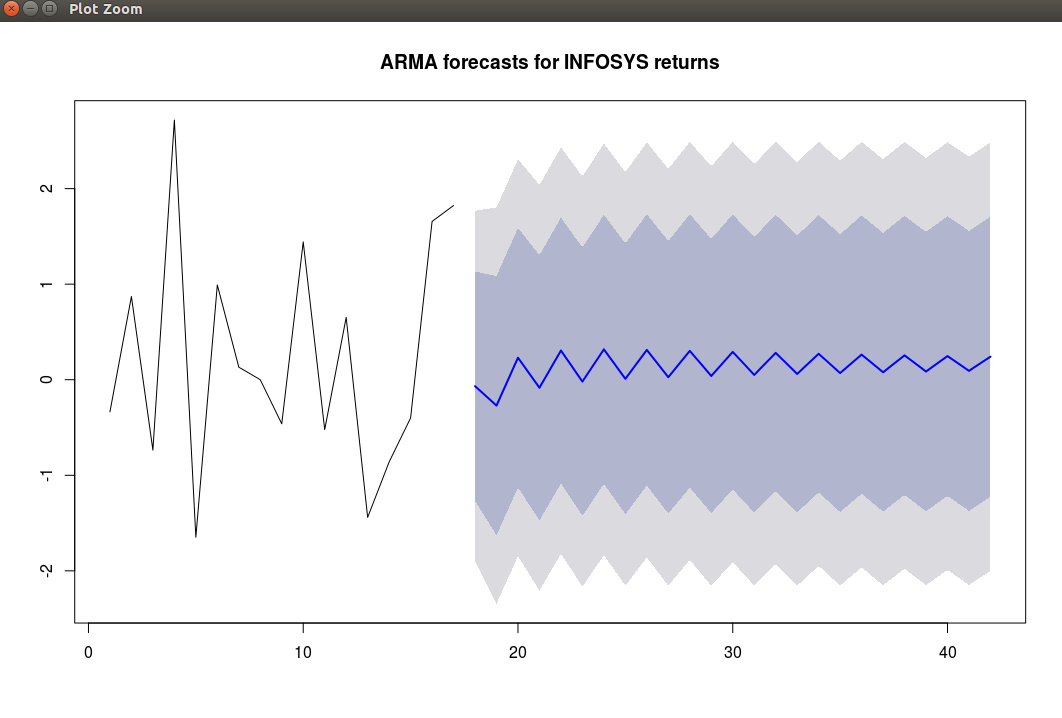
39 0.085870847 -1.376004 1.547745 -2.149873 2.321615

40 0.248107566 -1.215371 1.711586 -1.990090 2.486305

41 0.092847474 -1.372099 1.557794 -2.147595 2.333290

42 0.241432012 -1.224857 1.707721 -2.001064 2.483928

> plot(arma.forecast, main = "ARMA forecasts for INFOSYS returns")



**Stock Market Code:**

#install quantmod package

library('quantmod')

#get information of Infosys shares from yahoo as xst file using quantmod package

getSymbols("INFY",src="google" , from ="2017-02-14", to = "2017-03-14")

#convert xst file to dataframe

test<-as.data.frame(INFY)

write.csv(test,file="infy1.csv")

data<-read.csv("infy1.csv")

summary(data)

#install ggplot2 package

library(ggplot2, quietly = T)

plot(data$X, data$INFY.Close, xlab = "Dates", ylab = "Adjusted closing price",type = "l", col = "red", main = "Adjusted closing price of INFOSYS for past 1 month")

library(tseries, quietly = T) #time series analysis

adf.test(data$INFY.Close)#dickey-fuller test to convert non-stationary to stationary data

infy\_ret <- 100 \* diff(log(data$INFY.Close)) #function for stationary data

summary(arma(infy\_ret, order = c(2, 2)))

library(tseries, quietly = T)

infy\_ret\_train <- infy\_ret[1:(0.9 \* length(infy\_ret))] #training data

infy\_ret\_test <- infy\_ret[(0.9 \* length(infy\_ret) + 1):length(infy\_ret)] #testing data

fit <- arima(infy\_ret\_train, order = c(2, 0, 2)) #defining arima model

arma.preds <- predict(fit, n.ahead = (length(infy\_ret) - (0.9 \* length(infy\_ret))))$pred

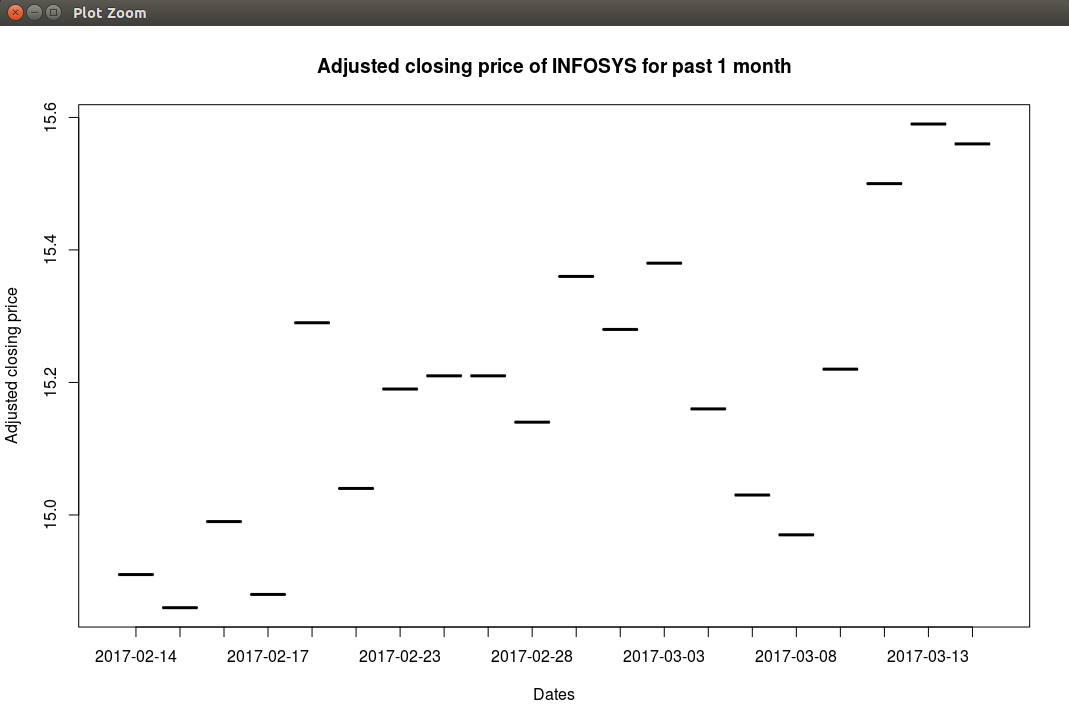
#install forecast package

library(forecast, quietly = T)

arma.forecast <- forecast(fit, h = 25)

summary(arma.forecast)

plot(arma.forecast, main = "ARMA forecasts for INFOSYS returns")

****

**Assignment 4**

**Title:** Association rule mining algorithm of data mining using R programming

**Aim :** Design a business intelligence application using any BI tool for a Mall to identify buying patterns of the customers and generate rules for taking decisions for maximizing the sales for the growth of the business. (Use Association rule mining algorithms to generate strong rules and develop system to recommend product association patterns for maximizing the sales. Make use of any latest technology tools like mongoBD, HDFS, HIVE, PIG, Hadoop, MADlib)

**Objective**: To implement Association rule mining algorithm of data mining using R programming

**Theory:**

**Association Rules Mining Algorithms:**

In general, association rule mining can be viewed as a two step process:

i) Find all frequent itemsets:

By definition, each of these itemsets will occur at least as frequently as a predetermined minimum support count, min sup.

ii) Generate strong association rules from the frequent itemsets:

By definition, these rules must satisfy minimum support and minimum confidence.

**Apriori Algorithm:**

It is by far the most well known association rule algorithm. The Apriori generates the candidate itemsets by joining the large itemsets of the previous pass and deleting those subsets which are small in the previous pass without considering the transactions in the database. By only considering large itemsets of the previous pass, the number of candidate large itemsets is significantly reduced.

The Steps of Apriori algorithm are:

* During the first step, Lk-1 is joined with itself to obtain Ck.
* In the second step, apriori\_gen() deletes all itemsets from the join result, which have some (k-1)–subset that is not in Lk-1. Then, it returns the remaining large k-itemsets.

The frequent item sets determined by Apriori can be used to determine [association rules](https://en.wikipedia.org/wiki/Association_rules) which highlight general trends in the [database](https://en.wikipedia.org/wiki/Database): this has applications in domains such as [market basket analysi](https://en.wikipedia.org/wiki/Market_basket_analysis)s.

**Pseudocode**:

*Ck*: Candidate itemset of size k

*Lk* : frequent itemset of size k

*L1* = {frequent items};

for (*k* = 1; *Lk* !=null; *k*++) do begin

*Ck+1* = candidates generated from *Lk*;

for each transaction *t* in database do

increment the count of all candidates in *Ck+1* that are contained in *t*

*Lk+1* = candidates in *Ck+1* with min\_support

end

return U*k* *Lk*;

**Commands implemented:**

1. The apriori() function from arule package implements the Apriori algorithm to generate frequent itemsets. By default, this function executes all iterations at once.

Itemsets<-apriori(Groceries, parameter = list ( minlen = 1, maxlen = 1, support = 0.02, target= ”frequent itemsets” ))

2. The inspect() function is used to display the top 10 frequent 1-itemsets sorted by their support.

Inspect( head( sort ( Itemsets, by=”support”), 10) )

3. The summary() function applied on Itemsets shows the support of 1-itemsets ranges from 0.02105 to 0.25552.

summary( Itemsets)

4. The summary() of the rules shows the number of rules and ranges of support, confidence and lift.

summary( rules )

5. The plot() function is used to display scatter plot of rules where x axis is the support and y axis is the confidence.

plot( rules )

6. The apply() function returns a vector or array or list of values obtained by applying a function to margins of an array or matrix.

**Apply ( Groceries@data[,10:20], 2, function(r) paste(Groceries@itemInfo[r,"labels"], collapse=", ")**

#Transactions in a Grocery Store

#install.packages('arules')

#install.packages('arulesViz')

library('arules')

library('arulesViz')

#The Groceries Dataset

data(Groceries)

Groceries

summary(Groceries)

class(Groceries)

# display the first 20 grocery labels

Groceries@itemInfo[1:20,]

# display the 10th to 20th transactions

apply(Groceries@data[,10:20], 2,

function(r) paste(Groceries@itemInfo[r,"labels"], collapse=", ")

)

#Frequent Itemset Generation

# frequent 1-itemsets

itemsets <- apriori(Groceries, parameter=list(minlen=1, maxlen=1, support=0.02, target="frequent itemsets"))

summary(itemsets)

inspect(head(sort(itemsets, by = "support"), 10))

# frequent 2-itemsets

itemsets <- apriori(Groceries, parameter=list(minlen=2, maxlen=2, support=0.02, target="frequent itemsets"))

summary(itemsets)

inspect(head(sort(itemsets, by ="support"),10))

# frequent 3-itemsets

itemsets <- apriori(Groceries, parameter=list(minlen=3, maxlen=3, support=0.02, target="frequent itemsets"))

inspect(sort(itemsets, by ="support"))

# frequent 4-itemsets

itemsets <- apriori(Groceries, parameter=list(minlen=4, maxlen=4, support=0.02, target="frequent itemsets"))

inspect(sort(itemsets, by ="support"))

# run Apriori without setting the maxlen parameter

itemsets <- apriori(Groceries, parameter=list(minlen=1, support=0.02,target="frequent itemsets"))

#Rule Generation and Visualization

rules <- apriori(Groceries, parameter=list(support=0.001,confidence=0.6, target = "rules"))

summary(rules)

plot(rules)

plot(rules@quality)

# displays rules with top lift scores

inspect(head(sort(rules, by="lift"), 10))

confidentRules <- rules[quality(rules)$confidence > 0.9]

confidentRules

plot(confidentRules, method="matrix", measure=c("lift", "confidence"),

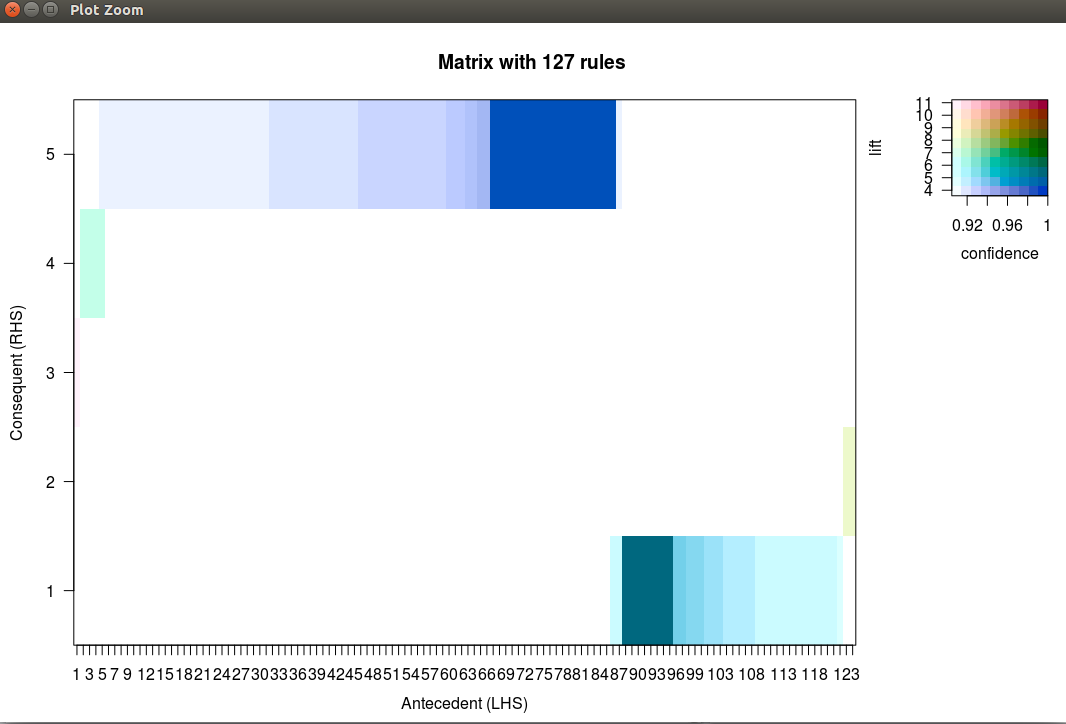
control=list(reorder=TRUE))

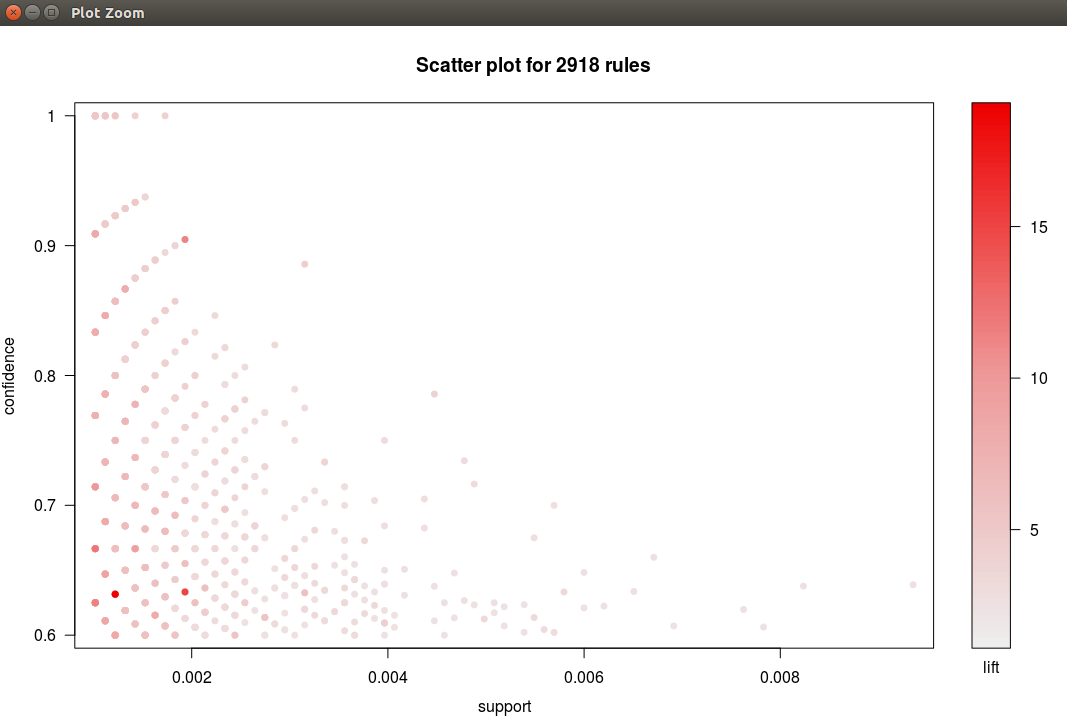
# select the 5 rules with the highest lift

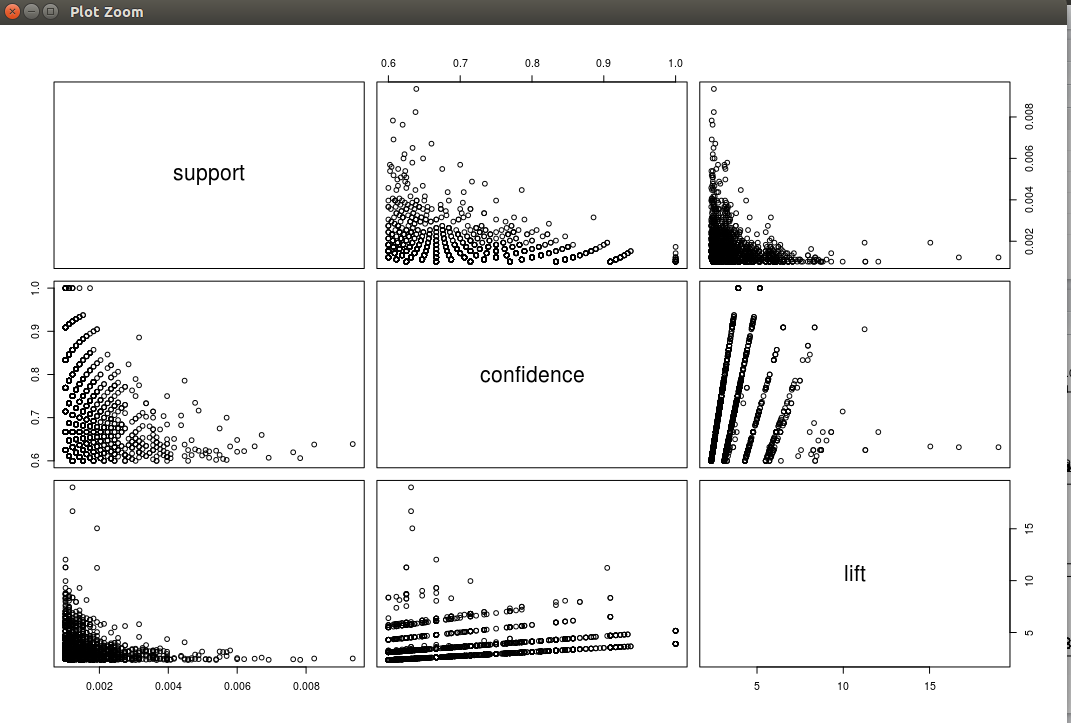
highLiftRules <- head(sort(rules, by="lift"), 5)

plot(highLiftRules, method="graph", control=list(type="items"))

****

****

****

****

**Assignment 5**

**Title:** Data exploration and analytics using R programming

**Aim :** Design a business intelligence application for an Industrial sector for maximizing the profit.

(Industrial applications like Pharmaceutical companies, Product manufacturing, insurance companies, placements companies etc.)

**Objective**: To implement Data exploration and analytics using R programming

**Theory:**

**Regression Technique:**

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

In Linear Regression, these two variables are related through an equation, where exponent (power) of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.

The general mathematical equation for a linear regression is −

y = ax + b

Following is the description of the parameters used −

* **y** is the response variable.
* x is the predictor variable.
* a and b are constants which are called the coefficients.

Multiple regression is an extension of linear regression into relationship between more than two variables. In simple linear relation we have one predictor and one response variable, but in multiple regression we have more than one predictor variable and one response variable.

The general mathematical equation for multiple regression is −

y = a + b1x1 + b2x2 +...bnxn

Following is the description of the parameters used −

* **y** is the response variable.
* a, b1, b2...bn are the coefficients.
* x1, x2, ...xn are the predictor variables

The Logistic Regression is a regression model in which the response variable (dependent variable) has categorical values such as True/False or 0/1. It actually measures the probability of a binary response as the value of response variable based on the mathematical equation relating it with the predictor variables.

The general mathematical equation for logistic regression is −

y = 1/(1+e^-(a+b1x1+b2x2+b3x3+...))

Following is the description of the parameters used −

1. **y** is the response variable.
2. x is the predictor variable.
3. a and b are the coefficients which are numeric constants.

**Problem statement for industrial sector for maximizing the profit:**

A fabricate company, ABC store chain, is selling a new type of grape juice in some of its stores for pilot selling. The marketing team of ABC wants to analyze: Which type of in-store advertisement is more effective?

They have placed two types of ads in stores for testing, one theme is natural production of the juice, the other theme is family health caring;

The Price Elasticity – the reactions of sales quantity of the grape juice to its price change;

The Cross-price Elasticity – the reactions of sales quantity of the grape juice to the price changes of other products such as apple juice and cookies in the same store;

How to find the best unit price of the grape juice which can maximize the profit and the forecast of sales with that price. The marketing team has randomly sampled 30 observations and constructed the following dataset for the analysis. There are 5 variables (data columns) in the dataset.

Variable Description

Sales | Total unit sales of the grape juice in one week in a store

Price | Average unit price of the grape juice in the week

ad\_type | The in-store advertisement type to promote the grape juice.

ad\_type = 0, the theme of the ad is natural production of the juice ad\_type = 1, the theme of the ad is family health caring

price\_apple | Average unit price of the apple juice in the same store in the week

price\_cookies | Average unit price of the cookies in the same store in the week

**Explanation for all the commands of the source code:**

* read.csv(file.choose(),header=T): read the dataset from an existing .csv file.
* head(): list the name of each variable (data column) and the first six rows of the dataset.
* Summary(): basic statistics of the variables.
* Par(): set the 1 by 2 layout plot window
* Boxplot(): boxplot to check if there are outliers.
* Hist(): histogram to explore the data distribution shape.
* Subset(): divide the dataset into sub datasets.
* Mean(): calculate the mean.
* shapiro.test(): check the normality by Shapiro-Wilk test.
* Pairs() and pairs20x(): display the correlation coefficients in pairs.
* Lm(): to fit linear models.
* Vif(): check multicollinearity.
* Optimize(): To get the optimal price to maximize Y,

**Mathematical Model:**

T-(I, O , F, Sc, Fc)

I-set of inputs

O-set of outputs

F- set of functions

Sc-set of success cases

Fc- set of failure cases

I- {d}

Where,

d-dataset of groceries

O={o1,o2,o3,...}

Where.

O1 to On are the association rules based on the dataset

F= {F1,F2,F3}

where,

F1: load()

F2: apriorio()

F3=plot()

Sc={Sc1,Sc2}

where

Sc1=necessary packages are installed

Sc2=support and confidence values are given correctly.

Fc={Fc1,Fc2}

Where.

Fc1= Libraries are not included

Fc2=groceries dataset is not found.

Input: DataSet, Confidence, Support

Output: Association rules

#read the dataset from an existing .csv file

df <- read.csv(file.choose(),header=T)

#list the name of each variable (data column) and the first six rows of the dataset

head(df)

# basic statistics of the variables

summary(df)

#set the 1 by 2 layout plot window

par(mfrow = c(1,2))

# boxplot to check if there are outliers

boxplot(df$sales,horizontal = TRUE, xlab="sales")

# histogram to explore the data distribution shape

hist(df$sales,main="",xlab="sales",prob=T)

lines(density(df$sales),lty="dashed",lwd=2.5,col="red")

#divide the dataset into two sub dataset by ad\_type

sales\_ad\_nature = subset(df,ad\_type==0)

sales\_ad\_family = subset(df,ad\_type==1)

#calculate the mean of sales with different ad\_type

mean(sales\_ad\_nature$sales)

mean(sales\_ad\_family$sales)

#set the 1 by 2 layout plot window

par(mfrow = c(1,2))

# histogram to explore the data distribution shapes

hist(sales\_ad\_nature$sales,main="",xlab="sales with nature production theme ad",prob=T)

lines(density(sales\_ad\_nature$sales),lty="dashed",lwd=2.5,col="red")

hist(sales\_ad\_family$sales,main="",xlab="sales with family health caring theme ad",prob=T)

lines(density(sales\_ad\_family$sales),lty="dashed",lwd=2.5,col="red")

shapiro.test(sales\_ad\_nature$sales)

shapiro.test(sales\_ad\_family$sales)

t.test(sales\_ad\_nature$sales,sales\_ad\_family$sales)

pairs(df,col="blue",pch=20)

pairs20x(df)

sales.reg<-lm(sales~price+ad\_type+price\_apple+price\_cookies,df)

summary(sales.reg)

# plotting the residuals vs. other key model metrics

par(mfrow=c(2,2))

plot(sales.reg)

#check multicollinearity

vif(sales.reg)

f = function(x) -51.24\*x^2 + 1028.84 \* x-3863.2

optimize(f,lower=0,upper=20,maximum=TRUE)

# predict the sales

inputData <- data.frame(price=10,ad\_type=1,price\_apple=7.659,price\_cookies=9.738)

predict(sales.reg,inputData,interval="p")

>df <- read.csv(file.choose(),header=T)

> head(df)

sales price ad\_type price\_apple price\_cookies

1 222 9.83 0 7.36 8.80

2 201 9.72 1 7.43 9.62

3 247 10.15 1 7.66 8.90

4 169 10.04 0 7.57 10.26

5 317 8.38 1 7.33 9.54

6 227 9.74 0 7.51 9.49

> summary(df)

sales price ad\_type price\_apple price\_cookies

Min. :131.0 Min. : 8.200 Min. :0.0 Min. :7.300 Min. : 8.790

1st Qu.:182.5 1st Qu.: 9.585 1st Qu.:0.0 1st Qu.:7.438 1st Qu.: 9.190

Median :204.5 Median : 9.855 Median :0.5 Median :7.580 Median : 9.515

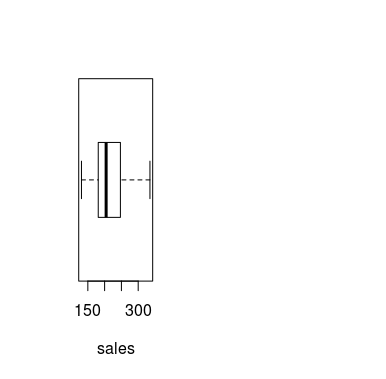
Mean :216.7 Mean : 9.738 Mean :0.5 Mean :7.659 Mean : 9.622

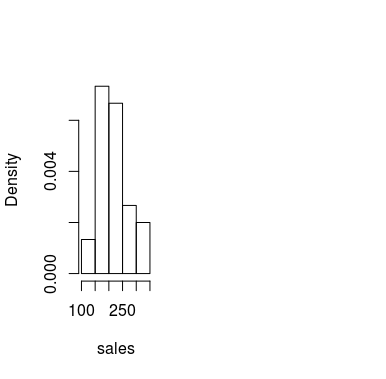
3rd Qu.:244.2 3rd Qu.:10.268 3rd Qu.:1.0 3rd Qu.:7.805 3rd Qu.:10.140

Max. :335.0 Max. :10.490 Max. :1.0 Max. :8.290 Max. :10.580

> par(mfrow = c(1,2))

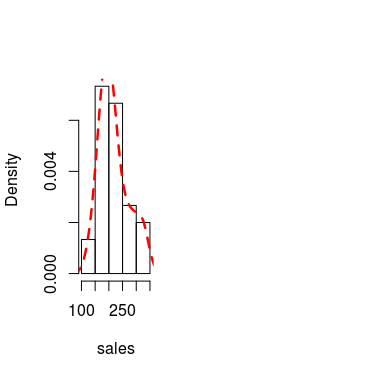
> boxplot(df$sales,horizontal = TRUE, xlab="sales")





> hist(df$sales,main="",xlab="sales",prob=T)

> lines(density(df$sales),lty="dashed",lwd=2.5,col="red")



> sales\_ad\_nature = subset(df,ad\_type==0)

> sales\_ad\_family = subset(df,ad\_type==1)

> mean(sales\_ad\_nature$sales)

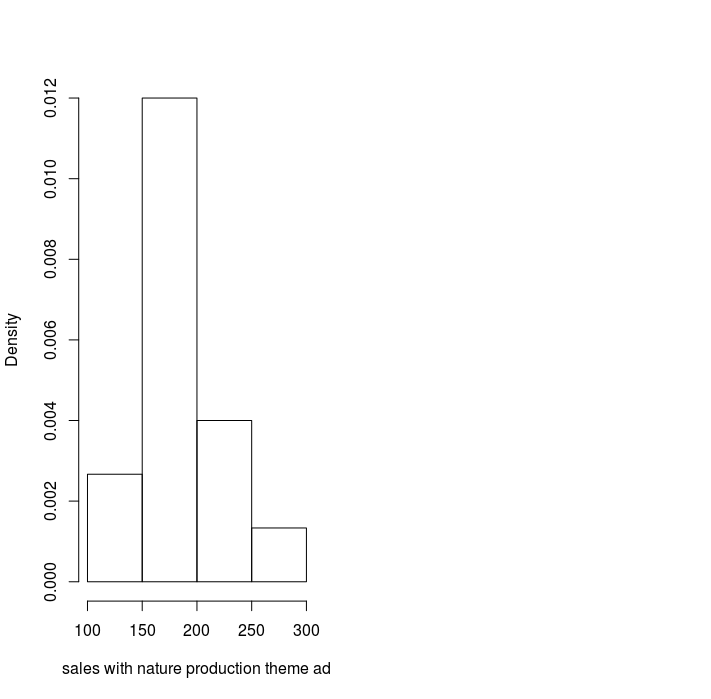
[1] 186.6667

> mean(sales\_ad\_family$sales)

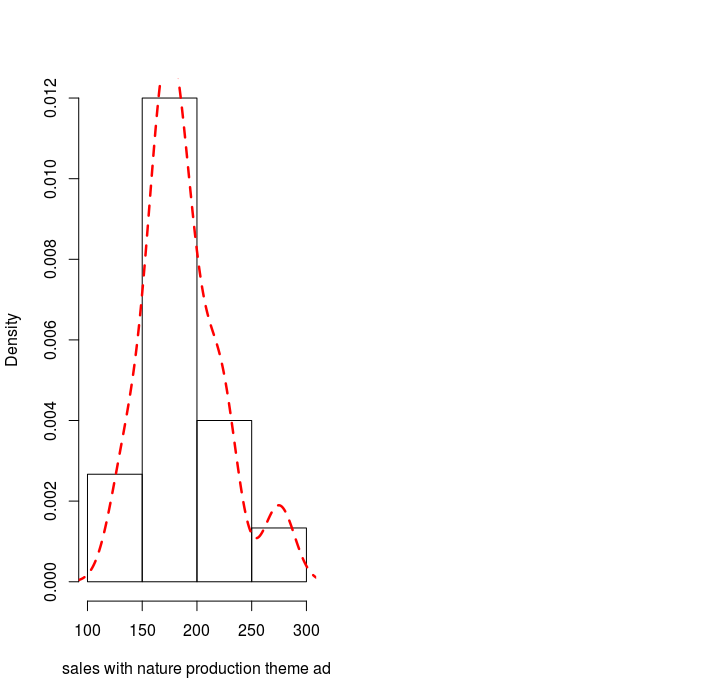
[1] 246.6667

> par(mfrow = c(1,2))

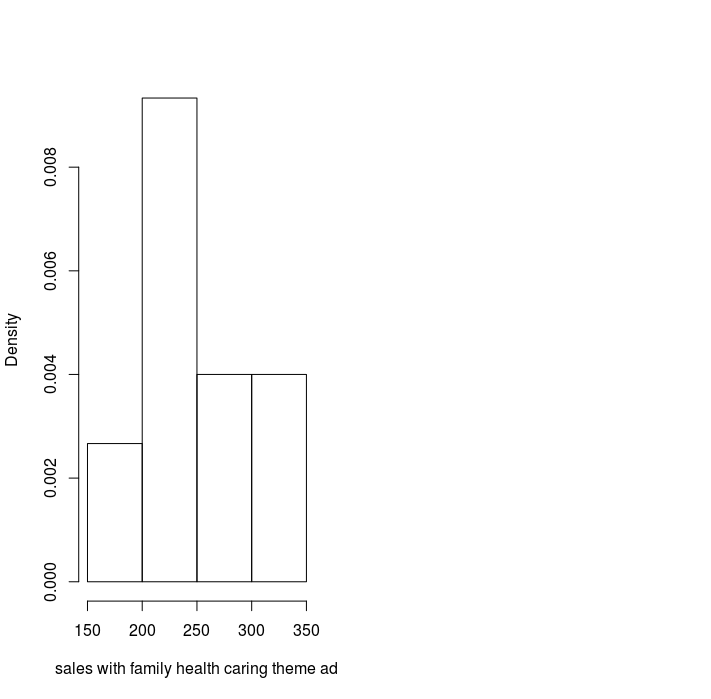
> hist(sales\_ad\_nature$sales,main="",xlab="sales with nature production theme ad",prob=T)



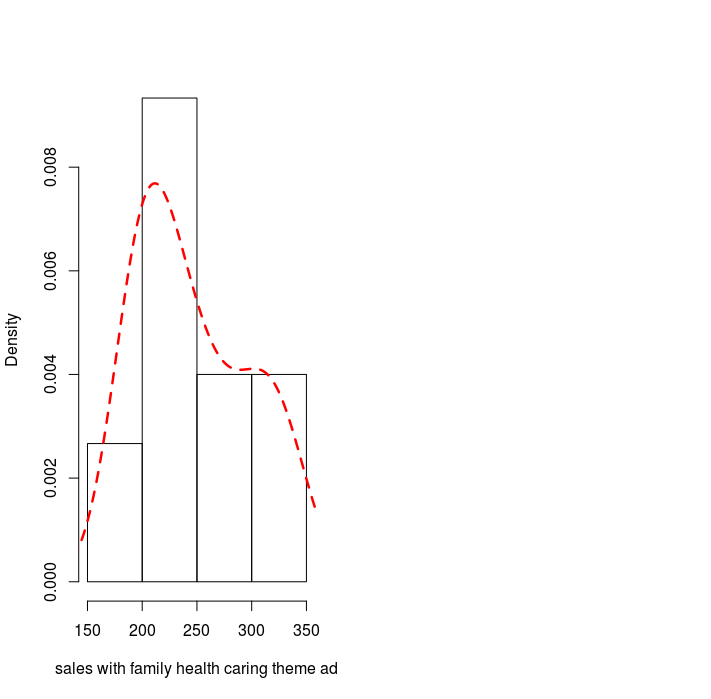
> lines(density(sales\_ad\_nature$sales),lty="dashed",lwd=2.5,col="red")



> hist(sales\_ad\_family$sales,main="",xlab="sales with family health caring theme ad",prob=T)



> lines(density(sales\_ad\_family$sales),lty="dashed",lwd=2.5,col="red")



> shapiro.test(sales\_ad\_nature$sales)

Shapiro-Wilk normality test

data: sales\_ad\_nature$sales

W = 0.94255, p-value = 0.4155

> shapiro.test(sales\_ad\_family$sales)

Shapiro-Wilk normality test

data: sales\_ad\_family$sales

W = 0.89743, p-value = 0.08695

> t.test(sales\_ad\_nature$sales,sales\_ad\_family$sales)

Welch Two Sample t-test

data: sales\_ad\_nature$sales and sales\_ad\_family$sales

t = -3.7515, df = 25.257, p-value = 0.0009233

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

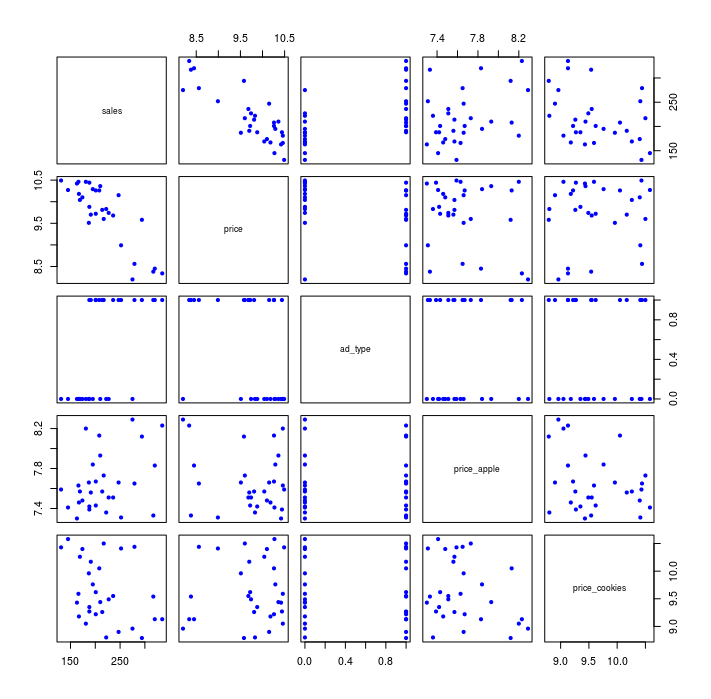
-92.92234 -27.07766

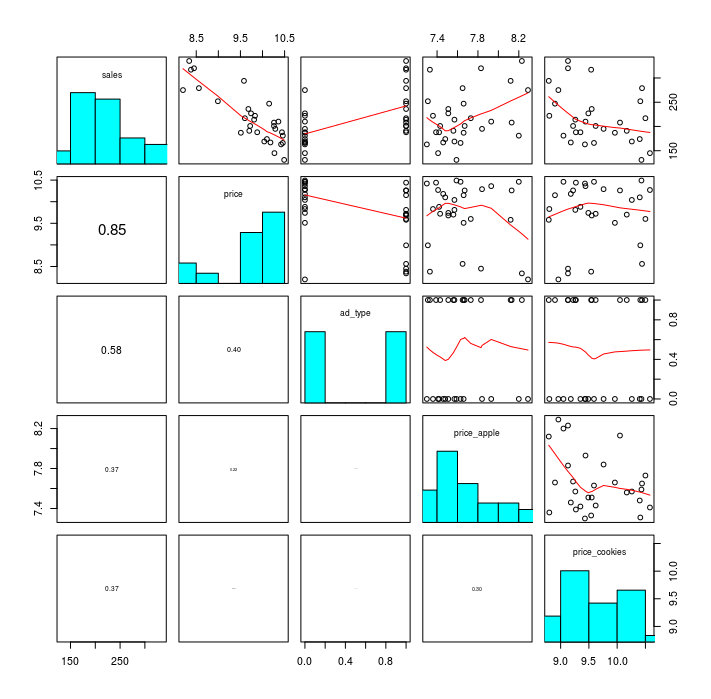
sample estimates:

mean of x mean of y

186.6667 246.6667

> pairs(df,col="blue",pch



 pairs20x(df)

> sales.reg<-lm(sales~price+ad\_type+price\_apple+price\_cookies,df)

> summary(sales.reg)

Call:

lm(formula = sales ~ price + ad\_type + price\_apple + price\_cookies,

data = df)

Residuals:

Min 1Q Median 3Q Max

-36.290 -10.488 0.884 10.483 29.471

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 774.813 145.349 5.331 1.59e-05 \*\*\*

price -51.239 5.321 -9.630 6.83e-10 \*\*\*

ad\_type 29.742 7.249 4.103 0.000380 \*\*\*

price\_apple 22.089 12.512 1.765 0.089710 .

price\_cookies -25.277 6.296 -4.015 0.000477 \*\*\*

---

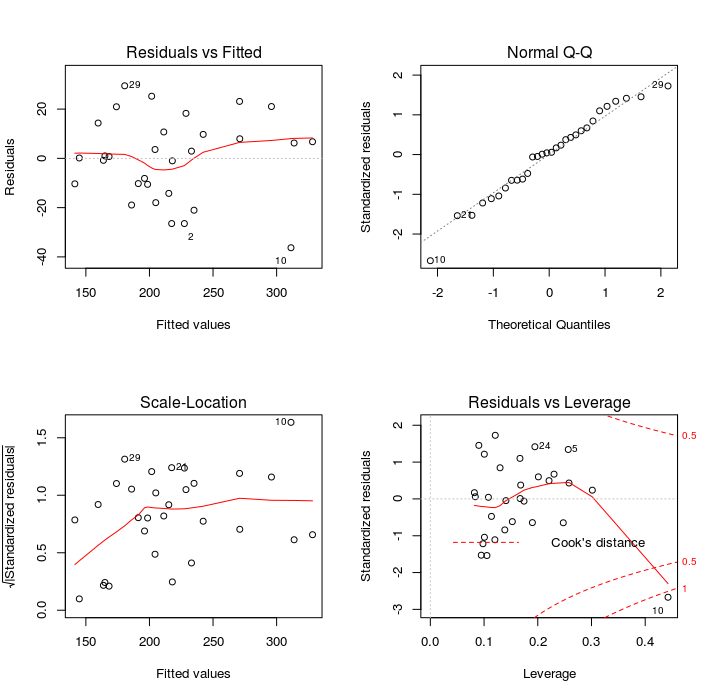
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 18.2 on 25 degrees of freedom

Multiple R-squared: 0.8974, Adjusted R-squared: 0.881

F-statistic: 54.67 on 4 and 25 DF, p-value: 5.318e-12

> par(mfrow=c(2,2))

> plot(sales.reg)

> f = function(x) -51.24\*x^2 + 1028.84 \* x-3863.2

> optimize(f,lower=0,upper=20,maximum=TRUE)

$maximum

[1] 10.03942

$objective

[1] 1301.28

> inputData <- data.frame(price=10,ad\_type=1,price\_apple=7.659,price\_cookies=9.738)

> predict(sales.reg,inputData,interval="p")

fit lwr upr

1 215.1978 176.0138 254.3817