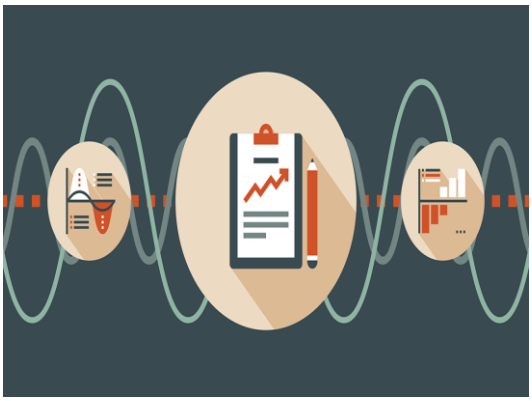
# SALES ANALYTICS: A STATISTICAL APPROACH TOWARDS OPTIMAL SOLUTIONS


- Aishwarya Sutar
- Amrita Das
- Ananya Iyer
- Anushka Yadav
- Apurva Choudhari
- Farheen Tarannum
- Gayatri Lele

# CONTENTS

# MOTIVATION

The motivation behind this project is to help businesses get an analytical approach as to what factors are promoting their sales and by how much. We want to provide insights on the areas where the company is performing well and give them guidance on the other forefronts where possible changes can be made in order to maximize profits and lower the possibility of risk.Business owners review their monthly and yearly financial statements to see the progress of their company. Relying on financial statements alone to measure sales means they might miss out on identifying trends they can use to improve future sales.

Using previous data, we can aid in predicting accurate sales forecasts to get a realistic picture on how much revenue will be earned by the company within a certain time period.We can also use analytics to track whether a particular product is increasing or decreasing in sales. If it's declining, the business can make timely decisions such as to cut prices, market more, or discontinue the product.By providing retailers with the right capabilities to drive business through data, they can have confidence in the decisions they make as they are backed by useful data interpretations.

# ABSTRACT

In this project we have analyzed the sales data provided to us by two shops-a pharmaceutical shop and the other a departmental store using different statistical techniques. We first carried out Exploratory Data Analysis to visualize our data. Further, for Confirmatory Data Analysis we made use of testing of hypothesis to check whether the average monthly sales of the two shops are same.Then, using Time Series Analysis and Markov Chain we have forecasted the future sales for both the shops. These forecasts will help the shops get a clear picture of what their sales will look like in the future. It will help them plan future strategies and decide if they can expand their business, invest elsewhere or maybe not take such risks.Apart from forecasting sales, we have also explored some methods which can help increase profit of small businesses. One such method is shelf space optimization using linear programming wherein products are placed on shelves in such a way that the profit is maximized. We have also made 2 models-the first one is an inventory model which can calculate the amount of units of a particular product to be restocked and the second one is a profit and discount model which can calculate the total profit for a retailer and also the discount he can give on a particular product. These are generalized models and can be used by anyone.Thus with this project we wish to help small businesses by providing them with the capability to drive business through data. With the right capabilities, they can have confidence in the decisions they make as they are backed by useful data interpretations

# KEYWORDS

Sale analysis,Retail space,Inventory,Stock,Lift,LPP- Linear Programming Problems,Shelf Space Optimization,Time Series,Markov Chain,Forecasting,T test,F test,Profit, Loss and Discount models

# TECHNICAL STATEMENT

The main aim of our project is to help small businesses monitor their sales and to pinpoint new opportunities to grow their business and discern areas where there can be possible scope for improvement. By using various techniques, we hope to aid in providing them actionable solutions.

# STATISTICAL STATEMENT

We have done an in depth analysis of sales using statistical tools like Time Series Analysis, Markov Chains and Optimization Techniques. Additionally, mathematical models have been made for profit, discount and inventory to provide insights into the performance of their business.

# DATA COLLECTION FORMAT

We collected our data by visiting two Shops-One is a Pharmaceutical Shop and the other one is Departmental Store. Both the shops are in Ahmednagar. They provided us with daily sales data of their shops from 2014-2018. As sales data is confidential, we will respect their privacy by not revealing the name of the shops.

# SOFTWARES USED

- R studio
- MS Excel
- NetBeans

# ILLUSTRATIVE DATA

## PHARMACEUTICAL SHOP

| YEAR | MONTH | TOTAL SALES |
|---|---|---|
| 2014 | JAN | 230501.1658 |
| 2014 | FEB | 201179.1594 |
| 2014 | MARCH | 220983.6686 |
| 2014 | APRIL | 239061.6488 |
| 2014 | MAY | 194345.6166 |
| 2014 | JUNE | 244296.768 |
| 2014 | JULY | 208531.1403 |
| 2014 | AUG | 281318.8735 |
| 2014 | SEPT | 238520.6214 |
| 2014 | OCT | 226103.0765 |
| 2014 | NOV | 213743.4967 |
| 2014 | DEC | 168210.109 |
| 2015 | JAN | 204315.6272 |
| 2015 | FEB | 185025.0025 |
| 2015 | MARCH | 224669.8568 |
| 2015 | APRIL | 246219.9265 |
| 2015 | MAY | 255989.213 |
| 2015 | JUNE | 195700.9927 |
| 2015 | JULY | 239239.9622 |
| 2015 | AUG | 178039.0151 |
| 2015 | SEPT | 223454.5214 |
| 2015 | OCT | 207323.9851 |
| 2015 | NOV | 207181.8898 |
| 2015 | DEC | 225363.2054 |
| 2016 | JAN | 201159.7234 |
| 2016 | FEB | 242203.3524 |
| 2016 | MARCH | 248997.028 |
| 2016 | APRIL | 240120.9357 |
| 2016 | MAY | 181516.752 |
| 2016 | JUNE | 243468.2318 |
| 2016 | JULY | 226084.0899 |
| 2016 | AUG | 192141.8432 |
| 2016 | SEPT | 211902.9912 |
| 2016 | OCT | 225554.229 |
| 2016 | NOV | 202075.7639 |
| 2016 | DEC | 262196.095 |
| 2017 | JAN | 180159.0937 |
| 2017 | FEB | 244562.9177 |
| 2017 | MARCH | 202912.2226 |
| 2017 | APRIL | 211796.6928 |
| 2017 | MAY | 207191.0003 |
| 2017 | JUNE | 162505.6928 |
| 2017 | JULY | 210320.1717 |
| 2017 | AUG | 204619.9221 |
| 2017 | SEPT | 206784.5517 |
| 2017 | OCT | 216272.745 |
| 2017 | NOV | 150983.3334 |
| 2017 | DEC | 205400.307 |
| 2018 | JAN | 237054.5794 |
| 2018 | FEB | 175655.7091 |
| 2018 | MARCH | 196501.5604 |
| 2018 | APRIL | 209696.1152 |
| 2018 | MAY | 144182.0622 |
| 2018 | JUNE | 227960.8323 |
| 2018 | JULY | 238138.9137 |
| 2018 | AUG | 238072.2487 |
| 2018 | SEPT | 153192.0942 |
| 2018 | OCT | 193527.2983 |
| 2018 | NOV | 189372.0844 |
| 2018 | DEC | 236820.4409 |

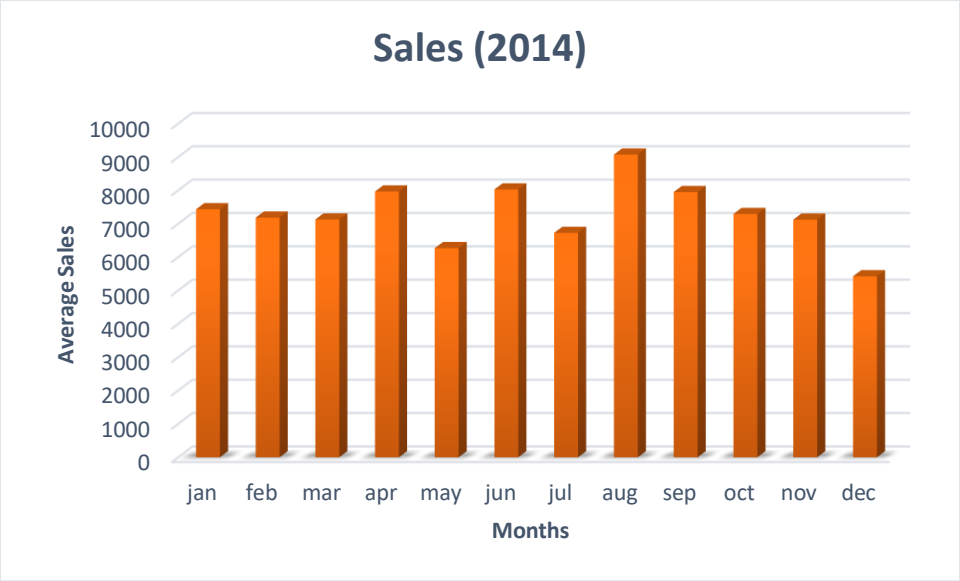
## DEPARTMENTAL STORE

| YEAR | MONTH | TOTAL SALES |
|---|---|---|
| 2014 | JAN | 173463.1 |
| 2014 | FEB | 174288.8355 |
| 2014 | MARCH | 182414.3552 |
| 2014 | APRIL | 188266.3422 |
| 2014 | MAY | 136185.6822 |
| 2014 | JUNE | 164547.9817 |
| 2014 | JULY | 155737.8245 |
| 2014 | AUG | 137796.2637 |
| 2014 | SEPT | 186769.6977 |
| 2014 | OCT | 146345.8604 |
| 2014 | NOV | 192860.0907 |
| 2014 | DEC | 283948.1631 |
| 2015 | JAN | 165300.7315 |
| 2015 | FEB | 231309.7201 |
| 2015 | MARCH | 240306.1141 |
| 2015 | APRIL | 162109.3623 |
| 2015 | MAY | 138800.8232 |
| 2015 | JUNE | 163772.4683 |
| 2015 | JULY | 151420.5401 |
| 2015 | AUG | 165716.0835 |
| 2015 | SEPT | 175109.1714 |
| 2015 | OCT | 267574.5342 |
| 2015 | NOV | 183911.572 |
| 2015 | DEC | 209201.7643 |
| 2016 | JAN | 131017.7926 |
| 2016 | FEB | 237811.1433 |
| 2016 | MARCH | 149096.1971 |
| 2016 | APRIL | 165820.5455 |
| 2016 | MAY | 166206.3897 |
| 2016 | JUNE | 161900.7962 |
| 2016 | JULY | 193069.7626 |
| 2016 | AUG | 173041.2813 |
| 2016 | SEPT | 186189.9536 |
| 2016 | OCT | 135078.0928 |
| 2016 | NOV | 170171.2375 |
| 2016 | DEC | 174220.588 |
| 2017 | JAN | 222948.6129 |
| 2017 | FEB | 157641.2102 |
| 2017 | MARCH | 190351.051 |
| 2017 | APRIL | 137220.8336 |
| 2017 | MAY | 179500.0756 |
| 2017 | JUNE | 186291.1816 |
| 2017 | JULY | 150115.6701 |
| 2017 | AUG | 203036.6988 |
| 2017 | SEPT | 168700.293 |
| 2017 | OCT | 171479.8952 |
| 2017 | NOV | 179859.9293 |
| 2017 | DEC | 217382.3809 |
| 2018 | JAN | 205865.2606 |
| 2018 | FEB | 186029.0845 |
| 2018 | MARCH | 152468.6091 |
| 2018 | APRIL | 192132.6827 |
| 2018 | MAY | 177155.1865 |
| 2018 | JUNE | 161491.3574 |
| 2018 | JULY | 187174.3221 |
| 2018 | AUG | 135206.3394 |
| 2018 | SEPT | 199972.0548 |
| 2018 | OCT | 225642.2227 |
| 2018 | NOV | 136837.3622 |
| 2018 | DEC | 183797.0295 |

# EXPLORATORY DATA ANALYSIS

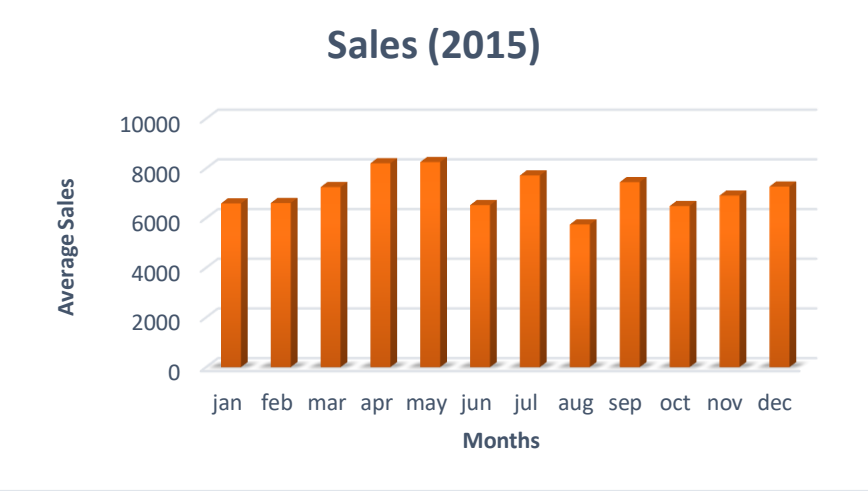
For pharmaceutical shop

**Bar diagram of months v/s average sales:**



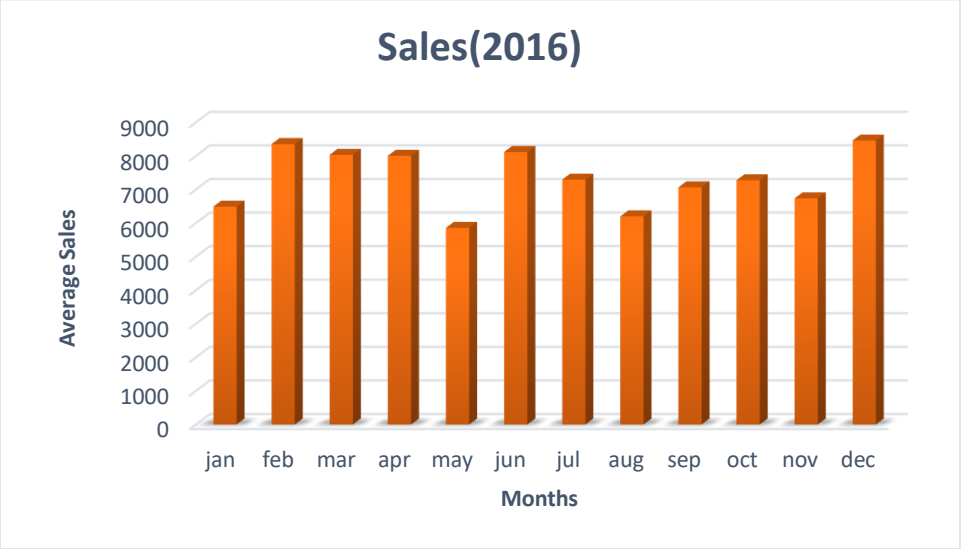

For the year 2014:

- The range of average monthly sales lies in between Rs.5000-9000.
- Sales is maximum in the month of August and minimum in the month of December.




For the year 2015:

- The range of average monthly sales lies in between Rs.5000-8000.
- The sales are maximum in the month of May and minimum in the month of August.

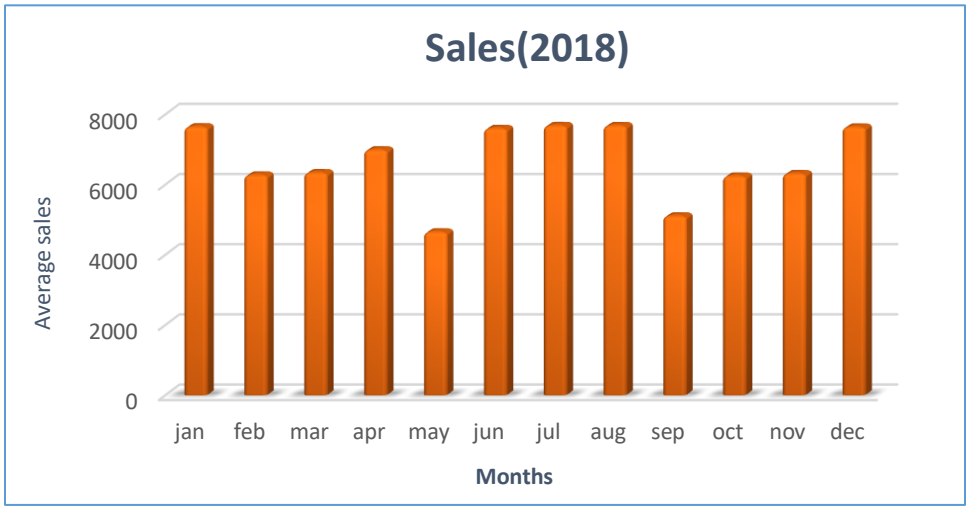



For the year 2016:

- The range of average monthly sales lies in between Rs.5500-8500.
- The sales are maximum in the month of December and minimum in the month of May.


For the year 2017:

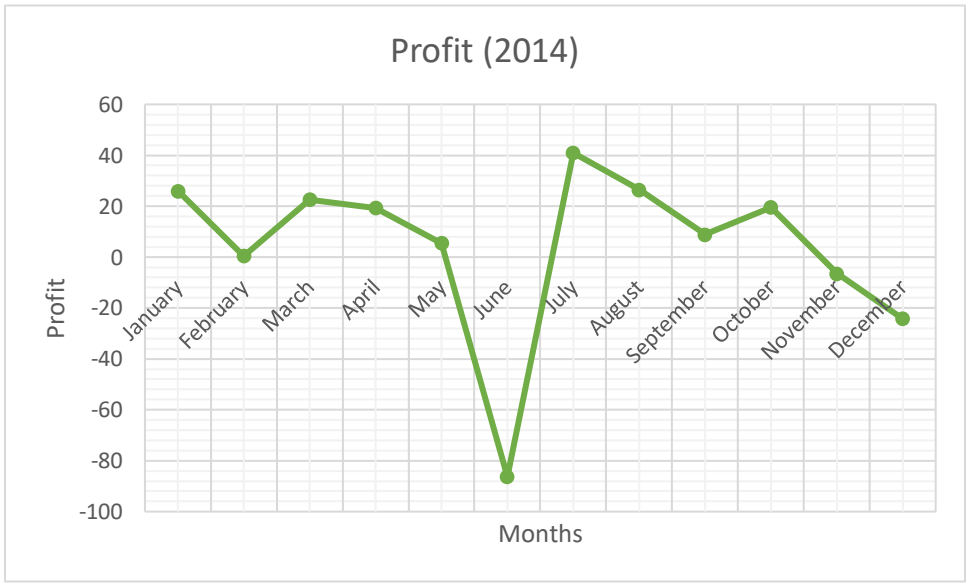
- The range of average monthly sales lies in between Rs.4500-8500.
- The sales are maximum in the month of February and minimum in the month of November.




For the year 2018:

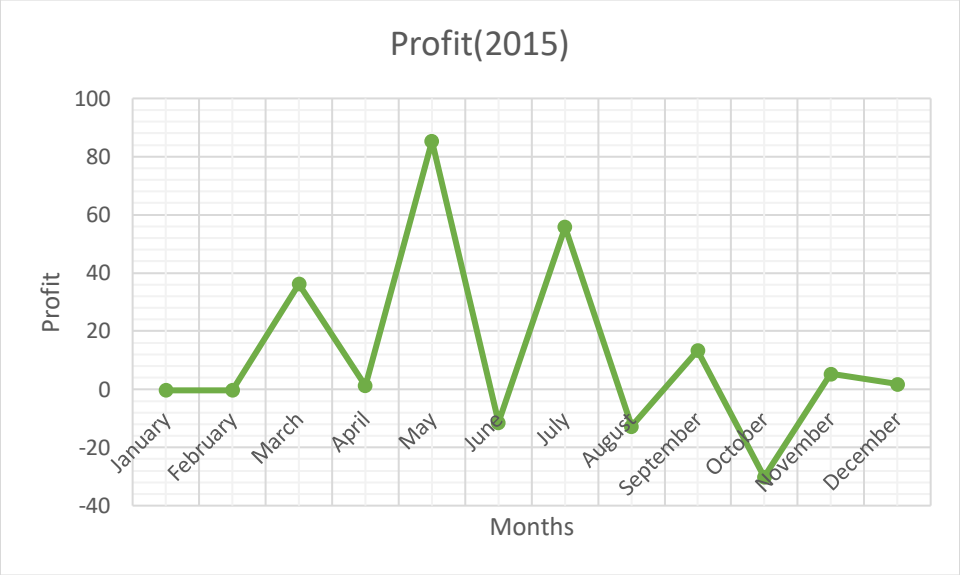
- The range of average monthly sales lies in between Rs.4500-8000.
- The sales are maximum in the month of August and minimum in the month of May.

## Graphical representation of profit:



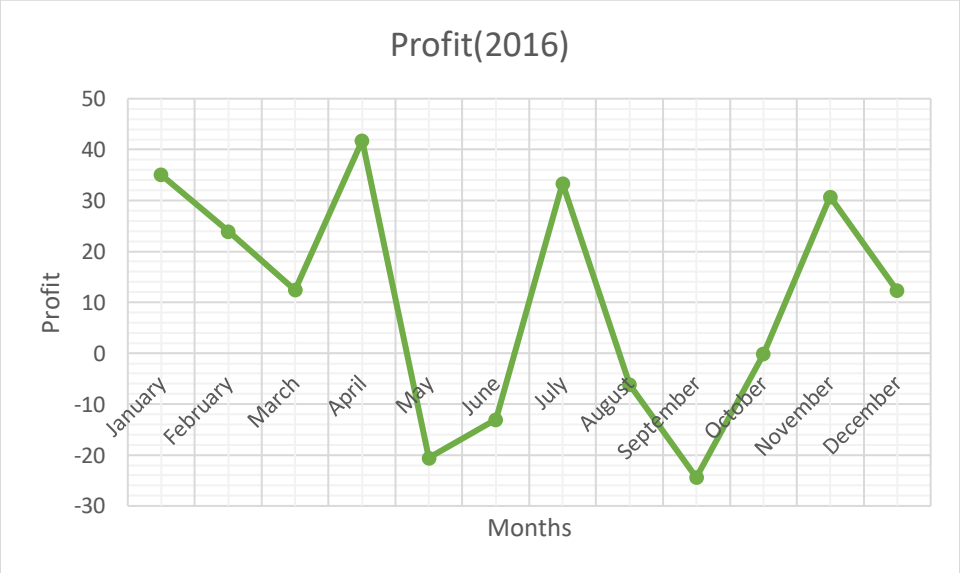

For the year 2014:

- Profit is maximum in the month of July and minimum in the month of June.



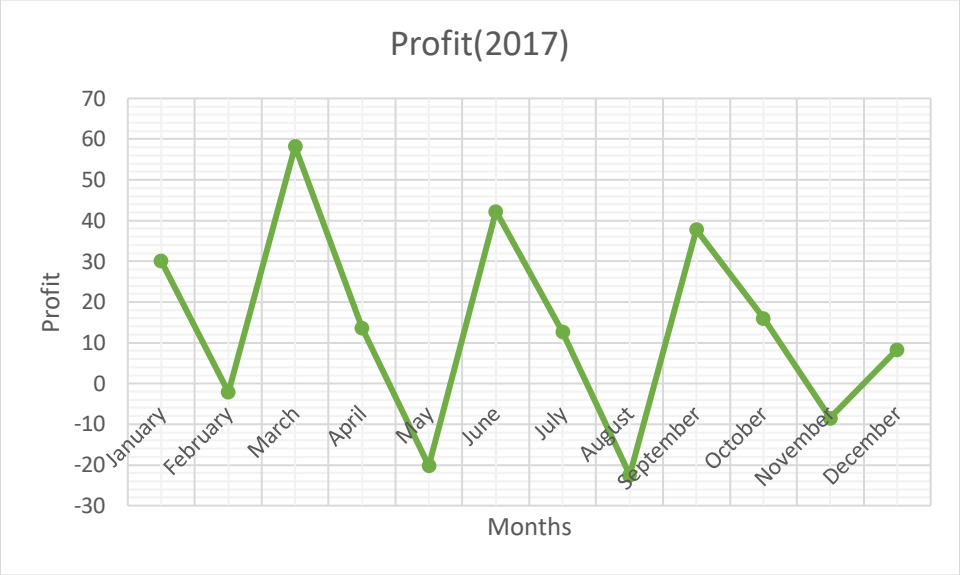

For the year 2015:

- Profit is maximum in the month of May and minimum in the month of October.



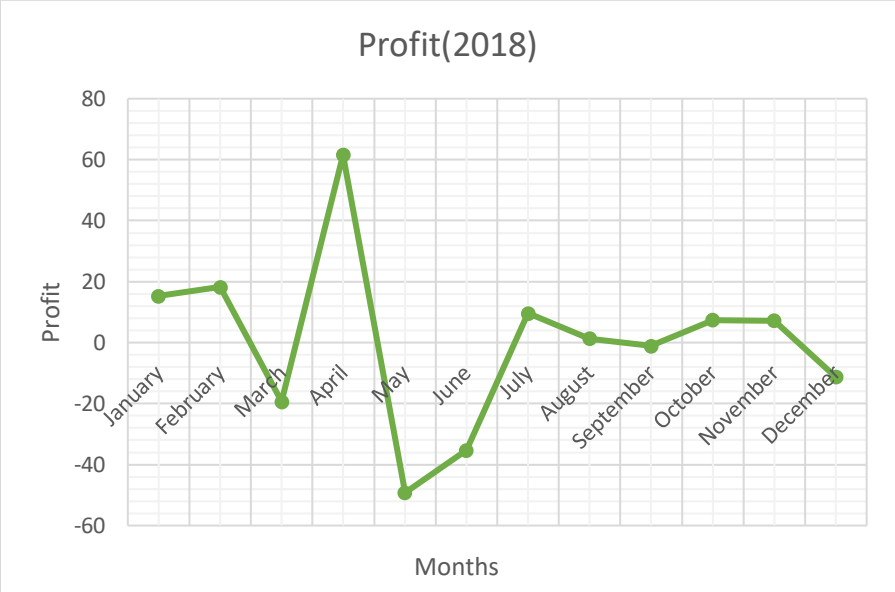

For the year 2016:

- Profit is maximum in the month of April and minimum in the month of September.




For the year 2017:

- Profit is maximum in the month of March and minimum in the month of August.

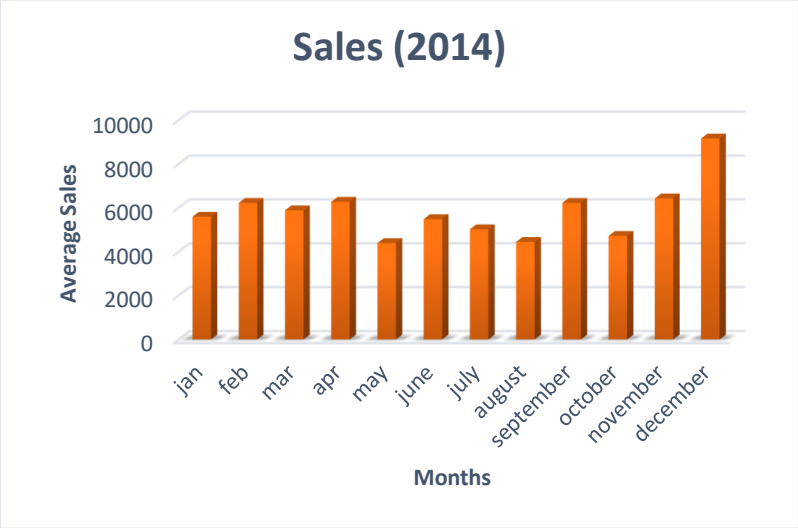



For the year 2018:

- Profit is maximum in the month of April and minimum in the month of May.

## FOR DEPARTMENTAL STORE

### Bar diagram of months v/s average sales:




For the year 2014:

- The range of average monthly sales lies in between Rs.4000-9000.
- Sales is maximum in the month of December and minimum in the month of


For the year 2015:

- The range of average monthly sales lies in between Rs.4000-9000.
- Sales is maximum in the month of October and minimum in the month of May.


For the year 2016:

- The range of average monthly sales lies in between Rs.3500-8000.
- Sales is maximum in the month of February and minimum in the month of January.

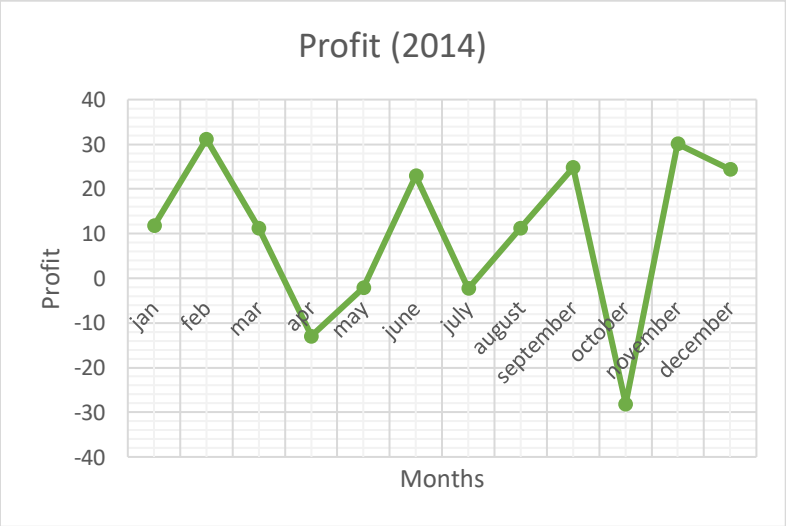

For the year 2017:

- The range of average monthly sales lies in between Rs.4000-7000.
- Sales is maximum in the month of January and minimum in the month of April.


For the year 2018:

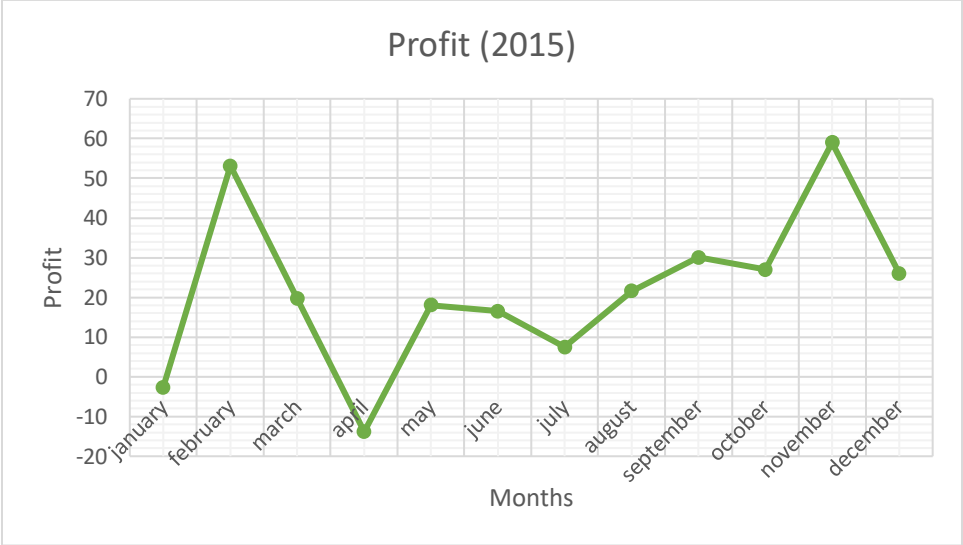
- The range of average monthly sales lies in between Rs.4000-7000.
- Sales is maximum in the month of October and minimum in the month of August.

## Graphical representation of profit



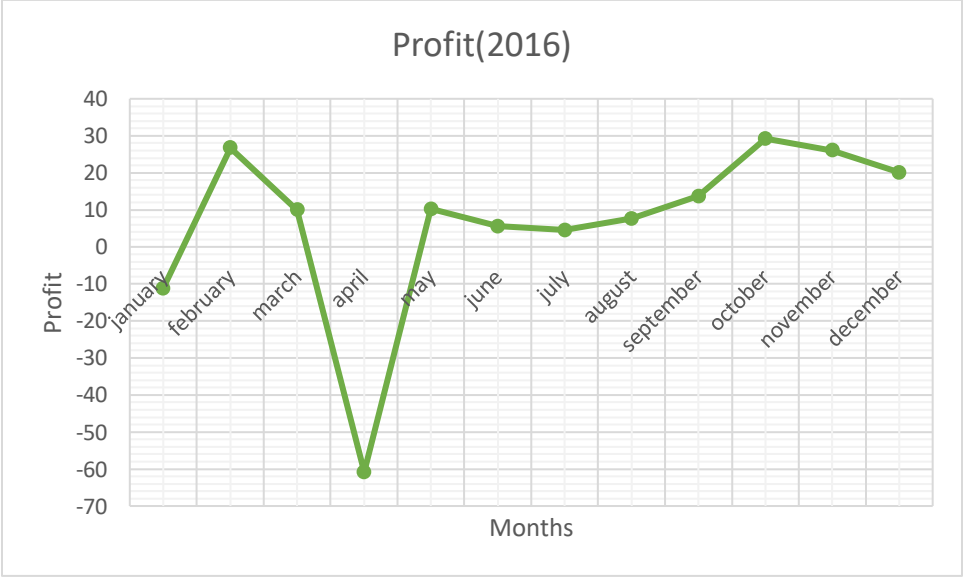

For the year 2014:

Profit is maximum in the month of February and minimum in the month of October.




For the year 2015:

Profit is maximum in the month of November and minimum in the month of April.

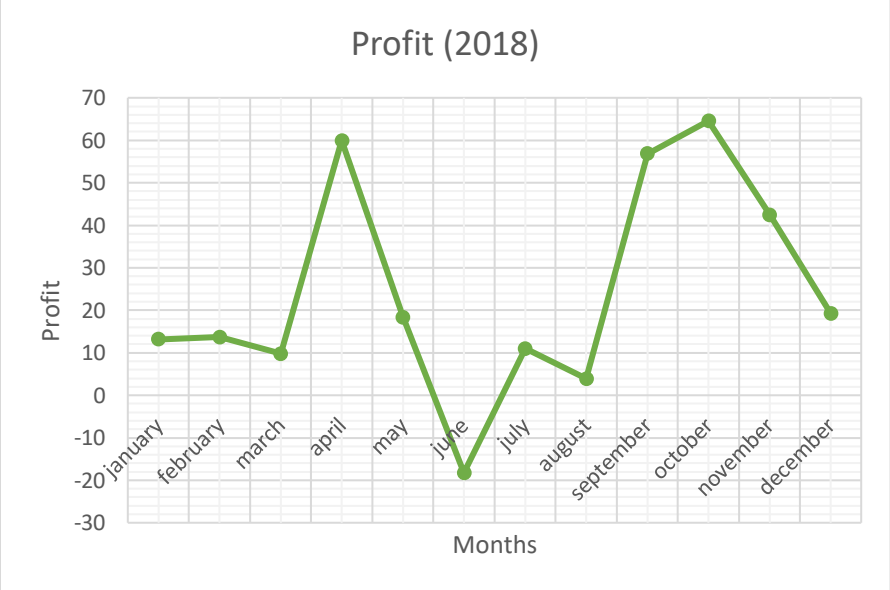



For the year 2016:

Profit is maximum in the month of October and minimum in the month of April.


For the year 2017:

Profit is maximum in the month of November and minimum in the month of July.

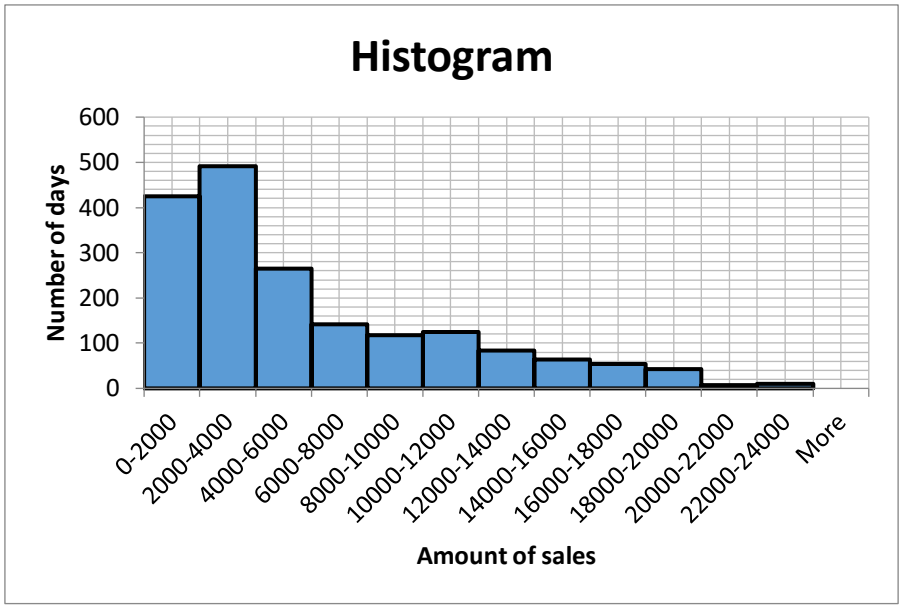



For the year 2018:

Profit is maximum in the month of October and minimum in the month of June.

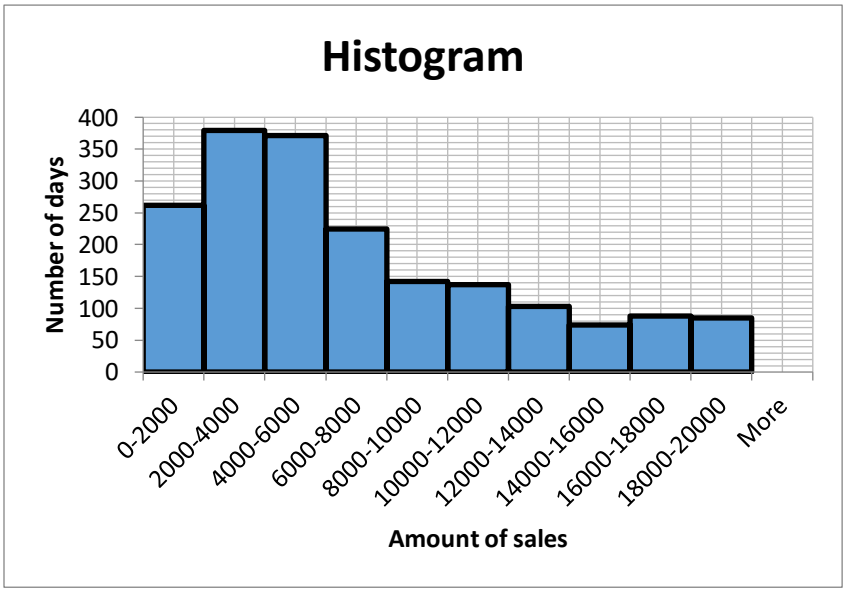
## Histogram of sales v/s days

### Departmental store




For this shop the maximum range of sales is in between Rs.2000-4000 and minimum range of sales is in between Rs.20000-22000

### Pharmaceutical shop




For this shop the maximum range of sales is in between Rs.2000-4000 and minimum range of sales is in between Rs.14000-18000.

# CONFIRMATORY DATA ANALYSIS

## Comparison of monthly sales of the two shops

We wish to compare the average monthly sales of the two shops. To do this we make use of the t test (two sample) for testing if there is significant difference between the average monthly sales of the 2 shops.

## R CODE

```
>a1=c(7435.521476,7184.969979,7128.505438,7968.721626,6269.213439,8143.225601,6726.810978,9074.802372,7950.68738,7293.647631,7124.783225,5426.132547)
>b1=c(6590.826685,6608.035802,7247.414736,8207.330883,8257.716547,6523.366424,7717.418136,5743.194036,7448.484047,6687.870486,6906.062993,7269.78082)
>c1=c(6489.023335,8351.839736,8032.162195,8004.031189,5855.379098,8115.607727,7293.035159,6198.123974,7063.43304,7275.942872,6735.858797,8457.938548)
>d1=c(5811.583669,8734.389916,6545.555569,7059.88976,6683.580653,5416.856427,6784.521666,6600.642649,6892.81839,6976.54016,5032.777781,6625.816356)
>e1=c(7646.921916,6273.418184,6338.760012,6989.870507,4651.034265,7598.694411,7681.900441,7679.749958,5106.403141,6242.816073,6312.402812,7639.369062)
> x=(a1+b1+c1+d1+e1)/5
>a2=c(5595.583869,6224.601268,5884.334037,6275.544739,4393.086521,5484.932723,5023.800789,4445.040765,6225.656589,4720.834205,6428.66969,9159.618163)
>b2=c(5332.281663,8261.061431,7751.810133,5403.645411,4477.44591,5459.082276,4884.533551,5345.680112,5836.972379,8631.436588,6130.385733,6748.444009)
>c2=c(4226.380407,8200.384251,4809.554746,5527.351518,5361.496442,5396.693208,6228.056858,5581.976818,6206.331786,4357.357834,5672.374585,5620.018968)
>d2=c(7191.890738,5630.04322,6140.356485,4574.027788,5790.325019,6209.706053,4842.440971,6549.570928,5623.343099,5531.609522,5995.330978,7012.334868)
>e2=c(6640.814858,6643.895876,4918.34223,6404.422757,5714.683436,5383.045246,6037.881358,4361.494821,6665.735158,7278.781378,4561.245406,5928.936437)
> y=(a2+b2+c2+d2+e2)/5
```

X: Yearly average of monthly sales of Pharmaceutical shop

Y: Yearly average of monthly sales of departmental store

**To check normality assumptions for Pharmaceutical shop**

To test: $H_0$: Yearly average of monthly sales of Pharmaceutical shop is normally distributed Vs

$H_1$: Yearly average of monthly sales of Pharmaceutical shop is not normally distributed

Command

```
> shapiro.test(x)
```

Shapiro-Wilk normality test

data: x

W = 0.96183, p-value = 0.8095

Decision: Here we may accept $H_0$

Conclusion: Yearly average of monthly sales of Pharmaceutical shop is normally distributed.

**To check normality assumptions for Departmental store**

To test: $H_0$: Yearly average of monthly sales of Departmental store is normally distributed Vs

$H_1$: Yearly average of monthly sales of Departmental store is not normally distributed

Command

```
> shapiro.test(y)
```

Shapiro-Wilk normality test

data: y

W = 0.90625, p-value = 0.191

Decision:Here we may accept $H_0$

Conclusion: Yearly average of monthly sales of Departmental store is normally distributed.

**F-test for checking equality of two population variance**

To test: $H_0: \sigma 1^2 = \sigma 2^2$ Vs $H_1: \sigma 1^2 \neq \sigma 2^2$

i.e. Variability in yearly average of monthly sales is same for both shops

Vs Variability in yearly average of monthly sales is not same for both shops

command

```
> var.test(m1,m2)
```

F test to compare two variances

data: x and y

F = 0.41471, num df = 11, denom df = 11, p-value = 0.1599

alternative hypothesis: true ratio of variances is not equal to 1

95 percent confidence interval:

0.1193848 1.4405667

sample estimates:

ratio of variances

0.4147068

Decision: Here we may accept $H_0$

Conclusion: Variability in yearly average of monthly sales is same for both shops.

**t-test for checking equality of two population means**

To test: $H_0: \mu_1 = \mu_2$ Vs $H_1: \mu_1 \neq \mu_2$

i.e. Average of yearly average of monthly sales of Pharmaceutical shop is same as that of Departmental store

Vs Average of yearly average of monthly sales of Pharmaceutical shop is not same as that of Departmental store

Command

```
> t.test(x,y,var.equal=T)
```

Two Sample t-test

data: x and y

t = 5.6328, df = 22, p-value = 1.156e-05

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

95 percent confidence interval:

707.5385 1532.1446

sample estimates:

mean of x mean of y

7002.220 5882.379

Decision: Here we reject $H_0$

Conclusion: Average of yearly average of monthly sales of Pharmaceutical shop is not same as that of Departmental store

# STATISTICAL ANALYSIS AND INTERPRETATION

## Time series analysis for departmental store

The data that we obtained from the departmental store was daily data given for 5 years i.e. from 2014-2018. From the given data, we decided to combine the sales to give monthly sales to do the time series analysis on. The analysis will be done on the 4-year data i.e. from 2014 - 2017, 2018 data will be used to validate the analysis hence done.

**The R code is –**

```
s.q=DEPARTMENTAL_STORE_FINAL_DATA$`TOTAL SALES`
data=c(s.q);data
tss=ts(data,frequency = 12,start=c(2014,1));tss
plot.ts(tss)
tss.dec=decompose(tss)
tss.dec
plot(tss.dec)
f=HoltWinters(tss);f
a=forecast(f,12)
summary(a)
plot(a)
res=tss.dec$random
plot(res)
shapiro.test(res)
res
r=as.vector(res);r
r=r[-c(1,2,3,4,5,6,43,44,45,46,47,48)];r
median(r)
r=r-median(r);r
a=c(0,0,1,0,1,1,0,1,1,0,0,0,0,1,0,1,1,0,0,1,0,1,1,0,1,1,1,0,0,0,1,0,1,0,1,1)
length(a)
a=as.factor(a)
runs.test(a)
```

```
ndiffs(tss)
Box.test(tss,lag=1,type=c("Box-Pierce","Ljung-Box"),fitdf = 0)
adf.test(tss,alternative=c("stationary","explosive"),k=trunc((length(tss)-1)^(1/3)))
kpss.test(tss)
auto.arima(tss)
acf(tss)
pacf(tss)
tss1=diff(log(tss));tss1
Box.test(tss1,lag=1,type=c("Box-Pierce","Ljung-Box"),fitdf = 0)
adf.test(tss1,alternative=c("stationary","explosive"),k=trunc((length(tss1)-1)^(1/3)))
kpss.test(tss1)
auto.arima(tss1)
acf(tss1)
pacf(tss1)
plot(tss1)
```

**FOR DEPARTMENTAL STORE**

```
> s.q=DEPARTMENTAL_STORE_FINAL_DATA$`TOTAL SALES`
> data=c(s.q)
> data
 [1] 173463.1 174288.8 182414.4 188266.3 136185.7 164548.0 155737.8 137796.3
 [9] 186769.7 146345.9 192860.1 283948.2 165300.7 231309.7 240306.1 162109.4
[17] 138800.8 163772.5 151420.5 165716.1 175109.2 267574.5 183911.6 209201.8
[25] 131017.8 237811.1 149096.2 165820.5 166206.4 161900.8 193069.8 173041.3
[33] 186190.0 135078.1 170171.2 174220.6 222948.6 157641.2 190351.1 137220.8
[41] 179500.1 186291.2 150115.7 203036.7 168700.3 171479.9 179859.9 217382.4
```

The **ts()** function will convert the numeric vector into an R time series object. The format is **ts(**vector, **frequency= start=,)** where start are the times of the first observation and frequency is the number of observations per unit time (1=annual, 4=quarterly, 12=monthly, etc.).

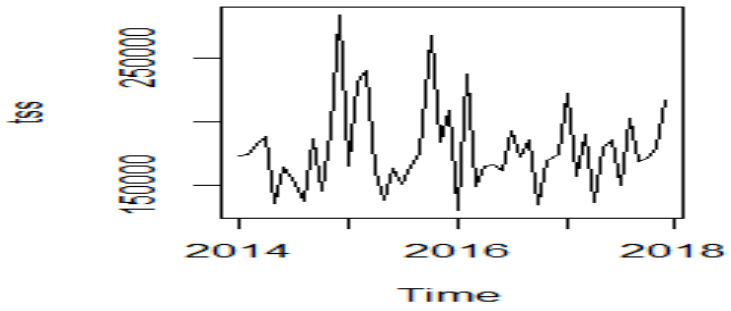
```
> tss=ts(data,frequency = 12,start=c(2014,1)); tss
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
2014 173463.1 174288.8 182414.4 188266.3 136185.7 164548.0 155737.8 137796.3
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2015 | 165300.7 | 231309.7 | 240306.1 | 162109.4 | 138800.8 | 163772.5 | 151420.5 | 165716.1 |
| 2016 | 131017.8 | 237811.1 | 149096.2 | 165820.5 | 166206.4 | 161900.8 | 193069.8 | 173041.3 |
| 2017 | 222948.6 | 157641.2 | 190351.1 | 137220.8 | 179500.1 | 186291.2 | 150115.7 | 203036.7 |

| | Sep | Oct | Nov | Dec |
|---|---|---|---|---|
| 2014 | 186769.7 | 146345.9 | 192860.1 | 283948.2 |
| 2015 | 175109.2 | 267574.5 | 183911.6 | 209201.8 |
| 2016 | 186190.0 | 135078.1 | 170171.2 | 174220.6 |
| 2017 | 168700.3 | 171479.9 | 179859.9 | 217382.4 |

To visualize our time series, we plotted the above time series.

```
> plot.ts(tss)
```




The decompose() function decomposes the time series into seasonal, trend, and irregular components using moving averages. It can be also used to identify whether the time series follows an additive or a multiplicative model. The function first determines the trend component using a moving average and removes it from the time series. Then, the seasonal figure is computed by averaging, for each time unit, over all periods. The seasonal figure is then centered. Finally, the irregular/error component is determined by removing trend and seasonal figure from the original time series.

```
> tss.dec=decompose(tss); tss.dec
```

$x

| | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug |
|---|---|---|---|---|---|---|---|---|
| 2014 | 173463.1 | 174288.8 | 182414.4 | 188266.3 | 136185.7 | 164548.0 | 155737.8 | 137796.3 |
| 2015 | 165300.7 | 231309.7 | 240306.1 | 162109.4 | 138800.8 | 163772.5 | 151420.5 | 165716.1 |
| 2016 | 131017.8 | 237811.1 | 149096.2 | 165820.5 | 166206.4 | 161900.8 | 193069.8 | 173041.3 |
| 2017 | 222948.6 | 157641.2 | 190351.1 | 137220.8 | 179500.1 | 186291.2 | 150115.7 | 203036.7 |

| | Sep | Oct | Nov | Dec |
|---|---|---|---|---|
| 2014 | 186769.7 | 146345.9 | 192860.1 | 283948.2 |
| 2015 | 175109.2 | 267574.5 | 183911.6 | 209201.8 |
| 2016 | 186190.0 | 135078.1 | 170171.2 | 174220.6 |
| 2017 | 168700.3 | 171479.9 | 179859.9 | 217382.4 |

$seasonal

| | Jan | Feb | Mar | Apr | May | Jun | Jul |
|---|---|---|---|---|---|---|---|
| 2014 | -6584.715 | 28418.899 | 12094.176 | -26204.818 | -19921.162 | -9663.693 | -12174.584 |
| 2015 | -6584.715 | 28418.899 | 12094.176 | -26204.818 | -19921.162 | -9663.693 | -12174.584 |
| 2016 | -6584.715 | 28418.899 | 12094.176 | -26204.818 | -19921.162 | -9663.693 | -12174.584 |
| 2017 | -6584.715 | 28418.899 | 12094.176 | -26204.818 | -19921.162 | -9663.693 | -12174.584 |

| | Aug | Sep | Oct | Nov | Dec |
|---|---|---|---|---|---|
| 2014 | -20522.165 | 3437.218 | 4345.840 | 3768.021 | 43006.982 |
| 2015 | -20522.165 | 3437.218 | 4345.840 | 3768.021 | 43006.982 |
| 2016 | -20522.165 | 3437.218 | 4345.840 | 3768.021 | 43006.982 |
| 2017 | -20522.165 | 3437.218 | 4345.840 | 3768.021 | 43006.982 |

$trend

| | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug |
|---|---|---|---|---|---|---|---|---|
| 2014 | NA | NA | NA | NA | NA | NA | 176545.3 | 178581.0 |
| 2015 | 183574.9 | 184558.3 | 185235.8 | 189801.1 | 194479.5 | 190992.2 | 186449.3 | 185291.7 |
| 2016 | 182134.3 | 184174.9 | 184941.8 | 179882.8 | 173789.6 | 171759.5 | 174132.4 | 174622.5 |
| 2017 | 173687.2 | 173147.3 | 173668.4 | 174456.4 | 176376.8 | 178578.9 | NA | NA |

| | Sep | Oct | Nov | Dec |
|---|---|---|---|---|
| 2014 | 183369.0 | 184691.3 | 183710.4 | 183787.1 |
| 2015 | 181762.2 | 178116.4 | 179413.0 | 180476.9 |
| 2016 | 173001.0 | 173528.3 | 172890.6 | 174460.7 |
| 2017 | NA | NA | NA | NA |

$random

| | Jan | Feb | Mar | Apr | May | Jun |
|---|---|---|---|---|---|---|
| 2014 | NA | NA | NA | NA | NA | NA |
| 2015 | -11689.42697 | 18332.50878 | 42976.15586 | -1486.94195 | -35757.47719 | -17556.01230 |
| 2016 | -44531.75455 | 25217.38144 | -47939.75693 | 12142.57073 | 12337.95757 | -195.04136 |
| 2017 | 55846.09126 | -43924.98049 | 4588.51080 | -11030.71904 | 23044.42935 | 17375.96339 |

| | Jul | Aug | Sep | Oct | Nov | Dec |
|---|---|---|---|---|---|---|
| 2014 | -8632.84297 | -20262.59391 | -36.56994 | -42691.31149 | 5381.64793 | 57154.10845 |
| 2015 | -22854.16095 | 946.52690 | -10090.24778 | 85112.27387 | 730.59954 | -14282.08113 |
| 2016 | 31111.91365 | 18940.97674 | 9751.72746 | -42796.05265 | -6487.33774 | -43247.11758 |
| 2017 | NA | NA | NA | NA | NA | NA |

$figure

[1] -6584.715 28418.899 12094.176 -26204.818 -19921.162 -9663.693 -12174.584

[8] -20522.165 3437.218 4345.840 3768.021 43006.982

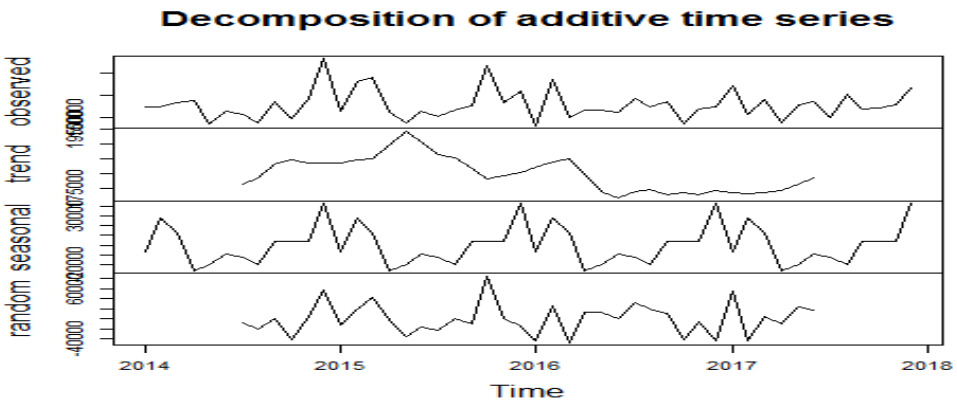
$type

[1] "additive"

attr(,"class")

[1] "decomposed.ts"

> plot(tss.dec)




From the above graph, the following interpretations can be made,

**TREND**- We can interpret that initially the sales trend is initially increasing then towards the beginning of 2015 there is a decrease towards the end of 2015 an increase can be observed, and at the start of 2016 there is a decrease. After mid-2016 the trend remains fairly constant.

**SEASONALITY**- We can interpret that seasonality is present, i.e. sales change in regular intervals of time (monthly).

**RANDOMNESS**- We can guess from the graph that the residuals are random.

For forecasting, we begin by using the Holt Winters Triple Exponential smoothing method.

```
> f=HoltWinters(tss);f
```

Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:

HoltWinters(x = tss)

Smoothing parameters:

**alpha: 0.05452207**

**beta : 0.1003117**

**gamma: 0.4366802**

Coefficients:

| | [,1] |
|---|---|
| a | 187105.68577 |
| b | -30.61519 |
| s1 | -5983.15851 |
| s2 | 14009.23954 |
| s3 | 10525.36725 |
| s4 | -33798.90116 |
| s5 | -24833.19183 |
| s6 | -15206.51370 |
| s7 | -23908.34839 |
| s8 | -8929.79590 |
| s9 | -11335.07963 |
| s10 | -17676.20983 |
| s11 | -8376.12912 |
| s12 | 30269.24698 |

To forecast the future sales for the next 12 months, we have used the forecast() function and then take the summary of the forecast to get the details displayed .

```
> a=forecast(f,12)
```

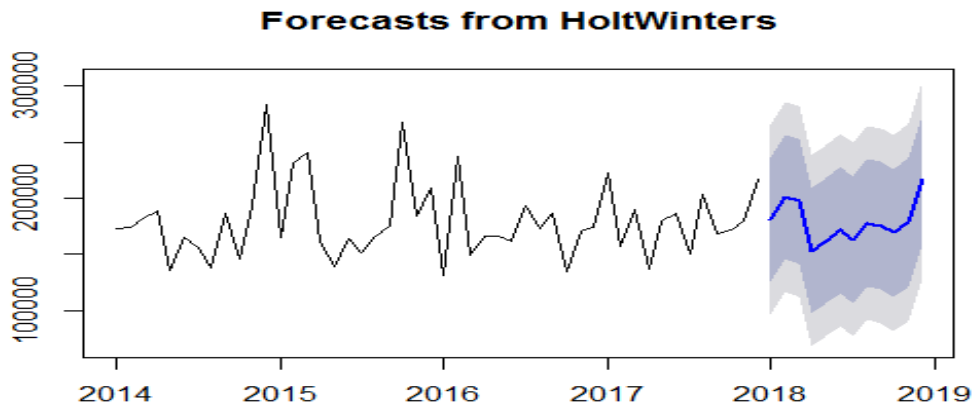
```
> summary(a)
Forecast method: HoltWinters
Model Information:
Holt-Winters exponential smoothing with trend and additive seasonal component.
Call:
HoltWinters(x = tss)
Smoothing parameters:
 alpha: 0.05452207
 beta : 0.1003117
 gamma: 0.4366802
Coefficients:
          [,1]
a   187105.68577
b      -30.61519
s1   -5983.15851
s2   14009.23954
s3   10525.36725
s4  -33798.90116
s5  -24833.19183
s6  -15206.51370
s7  -23908.34839
s8   -8929.79590
s9  -11335.07963
s10 -17676.20983
s11  -8376.12912
s12  30269.24698
Error measures:
                   ME          RMSE     MAE         MPE       MAPE      MASE        ACF1
Training set -6407.931  43137.77  29438.46  -5.601142  16.78836  0.8364316  -0.1292339
Forecasts:
```

(The below table shows the 80% and 95% confidence interval under which the sales will lie for the next 12 months and we will validate the forecasts by comparison with actual data)

| | Lo 80 | Hi 80 | Lo 95 | Hi 95 | actual value |
|---|---|---|---|---|---|
| Jan 2018 | 125646.47 | 236537.4 | 96295.43 | 265888.4 | 205865.261 |
| Feb 2018 | 145508.57 | 256598.8 | 116104.76 | 286002.6 | 186029.085 |
| Mar 2018 | 141875.63 | 253202.8 | 112409.11 | 282669.3 | 152468.609 |
| Apr 2018 | 97381.99 | 208986.7 | 67842.02 | 238526.6 | 192132.683 |
| May 2018 | 106156.54 | 218082.3 | 76531.58 | 247707.3 | 177155.187 |
| Jun 2018 | 115568.81 | 227862.1 | 85846.56 | 257584.4 | 161491.357 |
| Jul 2018 | 106627.93 | 219338.1 | 76795.34 | 249170.7 | 187174.322 |
| Aug 2018 | 121341.43 | 234520.5 | 91384.74 | 264477.2 | 135206.339 |
| Sep 2018 | 118643.80 | 232346.3 | 88548.56 | 262441.6 | 199972.055 |
| Oct 2018 | 111981.79 | 226264.9 | 81732.90 | 256513.8 | 225642.223 |
| Nov 2018 | 120931.30 | 235854.3 | 90513.03 | 266272.6 | 136837.362 |
| Dec 2018 | 159195.31 | 274819.8 | 128591.36 | 305423.7 | 183797.030 |

We can see that the values do lie in the confidence interval. This validates our model.

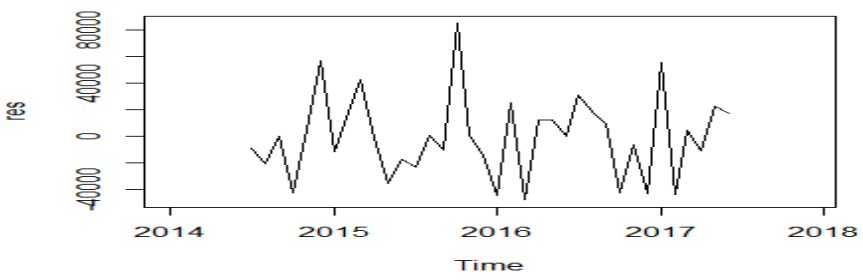
> plot(a)




In the above graph we can see that sales of 2018 are predicted.The confidence bands are also shown.

> res=tss.dec$random

> plot(res)




The plot of residuals is plotted. The residuals are called from the decomposed data. Now we check the normality of the residuals. Normality of residuals are an assumption of running a linear model. If the residuals are normal, it means that the model is valid and the confidence interval and model predi ctions are valid. We check normality by Shapiro Wilkes test.

```
> shapiro.test(res)
```

Shapiro-Wilk normality test

data: res

W = 0.95676, p-value = 0.1705

As the p value is greater than 0.05 we can say that the residuals are normally distributed. Now, we ch eck the randomness of residuals by run test. If the residuals are random i.e. it is just a white noise, we have captured all the components of the time series in our model and it is a good one. The run test pr ocedure is as follows- we take the median and compare it with each value . the values below are put as 0 and above the median as 1. We convert this vector of 0's and 1's as factor and run the test.

```
> res
```

| | Jan | Feb | Mar | Apr | May | Jun |
|---|---|---|---|---|---|---|
| 2014 | NA | NA | NA | NA | NA | NA |
| 2015 | -11689.42697 | 18332.50878 | 42976.15586 | -1486.94195 | - 35757.47719 | -17556.01230 |
| 2016 | -44531.75455 | 25217.38144 | -47939.75693 | 12142.57073 | 12337.95757 | -195.04136 |
| 2017 | 55846.09126 | -43924.98049 | 4588.51080 | -11030.71904 | 23044.42935 | 17375.96339 |
| | Jul | Aug | Sep | Oct | Nov | Dec |
| 2014 | -8632.84297 | -20262.59391 | -36.56994 | -42691.31149 | 5381.64793 | 57154.10845 |
| 2015 | -22854.16095 | 946.52690 | -10090.24778 | 85112.27387 | 730.59954 | -14282.08113 |
| 2016 | 31111.91365 | 18940.97674 | 9751.72746 | -42796.05265 | - 6487.33774 | -43247.11758 |

```
2017      NA                NA                NA                NA                NA                NA
> r=as.vector(res); r
 [1]          NA          NA          NA          NA          NA          NA
 [7]  -8632.84297 -20262.59391    -36.56994 -42691.31149   5381.64793  57154.10845
[13] -11689.42697  18332.50878  42976.15586  -1486.94195 -35757.47719 -17556.01230
[19] -22854.16095    946.52690 -10090.24778  85112.27387    730.59954 -14282.08113
[25] -44531.75455  25217.38144 -47939.75693  12142.57073  12337.95757   -195.04136
[31]  31111.91365  18940.97674   9751.72746 -42796.05265  -6487.33774 -43247.11758
[37]  55846.09126 -43924.98049   4588.51080 -11030.71904  23044.42935  17375.96339
[43]          NA          NA          NA          NA          NA          NA
> r=r[-c(1,2,3,4,5,6,43,44,45,46,47,48)];r
 [1]  -8632.84297 -20262.59391    -36.56994 -42691.31149   5381.64793  57154.10845
 [7] -11689.42697  18332.50878  42976.15586  -1486.94195 -35757.47719 -17556.01230
[13] -22854.16095    946.52690 -10090.24778  85112.27387    730.59954 -14282.08113
[19] -44531.75455  25217.38144 -47939.75693  12142.57073  12337.95757   -195.04136
[25]  31111.91365  18940.97674   9751.72746 -42796.05265  -6487.33774 -43247.11758
[31]  55846.09126 -43924.98049   4588.51080 -11030.71904  23044.42935  17375.96339
> median(r)
[1] -115.8056
> r=r-median(r);r
[1]  -8517.03732 -20146.78826     79.23571 -42575.50584   5497.45358  57269.91410
 [7] -11573.62132  18448.31443  43091.96151  -1371.13630 -35641.67154 -17440.20665
[13] -22738.35530   1062.33255  -9974.44213  85228.07952    846.40519 -14166.27548
[19] -44415.94890  25333.18709 -47823.95128  12258.37638  12453.76322    -79.23571
[25]  31227.71930  19056.78239   9867.53310 -42680.24700  -6371.53209 -43131.31193
[31]  55961.89691 -43809.17484   4704.31645 -10914.91339  23160.23500  17491.76904
> a=c(0,0,1,0,1,1,0,1,1,0,0,0,0,1,0,1,1,0,0,1,0,1,1,0,1,1,1,0,0,0,1,0,1,0,1,1)
> length(a)
[1] 36
> a=as.factor(a)
> runs.test(a)
```

Runs Test

data:  a

Standard Normal = 1.0146, p-value = 0.3103

alternative hypothesis: two.sided

As the p value is greater than 0.5 we say that the residuals are random. Now we will try to make our t ime series stationary. Firstly we will try the differencing method.

```
> ndiffs(tss)
[1] 0
> Box.test(tss,lag=1,type=c("Box-Pierce","Ljung-Box"),fitdf = 0)
```

Box-Pierce test

data:  tss

X-squared = 0.29045, df = 1, p-value = 0.5899

```
> adf.test(tss,alternative=c("stationary","explosive"),k=trunc((length(tss)-1)^(1/3)))
```

Augmented Dickey-Fuller Test

data:  tss

Dickey-Fuller = -3.3399, Lag order = 3, p-value = 0.0765

alternative hypothesis: stationary

```
> kpss.test(tss)
```

KPSS Test for Level Stationarity

data:  tss

KPSS Level = 0.053587, Truncation lag parameter = 3, p-value = 0.1

```
> auto.arima(tss)
```

Series: tss

ARIMA(0,0,0) with non-zero mean

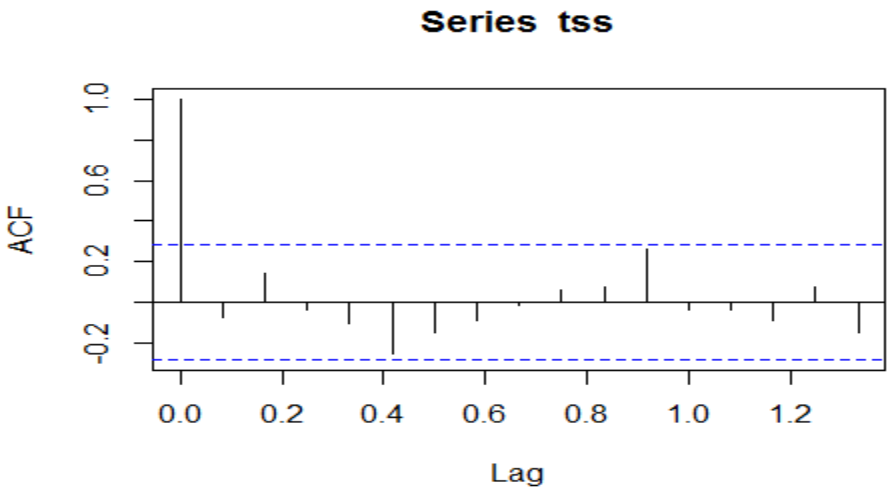
Coefficients:

| | mean |
|---|---|
| | 178860.598 |
| s.e. | 4763.369 |

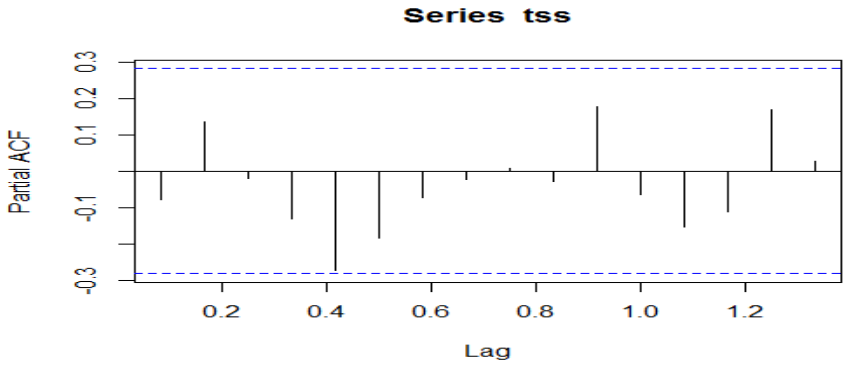
sigma^2 estimated as 1.112e+09:  log likelihood=-567.52

AIC=1139.03   AICc=1139.3   BIC=1142.77

```
> acf(tss)
```




> pacf(tss)




We observe that the ndiffs is 0 i.e. the time series is already stationary. We validated this by Ljung b ox test, adf test, kpss test as well as by ACF and PACF plots. The results are as follows-

1. For Ljung box test:p value is greater than 0.05 i.e. there is insufficient evidence to reject null hypothesis of independence of values on each other.
2. For adf test:p value >0.05 indicates that time series is not stationary.
3. For kpss test: as p value >0.05 this means that only differencing cannot be further used.
4. No ARIMA model can be fit as the time series is not stationary.
5. The ACF and PACF plots do not specify any results to fit ARIMA model.

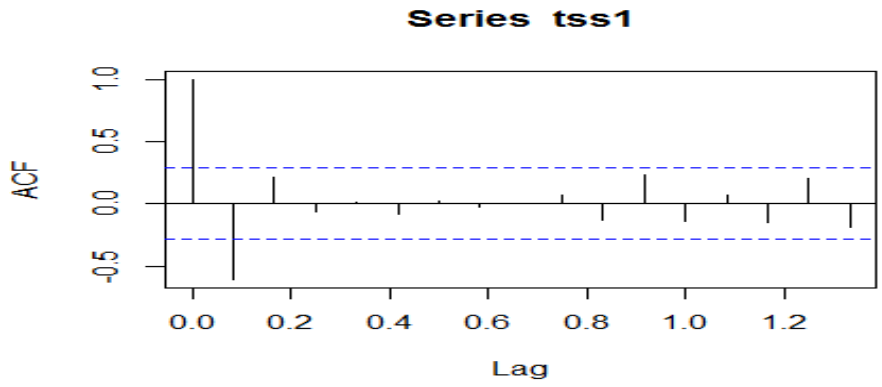
Now because differencing cannot be used we try different transformations.

```
> tss1=diff(log(tss)); tss1
          Jan          Feb          Mar          Apr          May          Jun
2014                0.004749001  0.045566879  0.031576898 -0.323838409  0.189182945
2015 -0.541025267  0.335991162  0.038155994 -0.393642403  0.155231205  0.165438097
2016 -0.467966030  0.596143708 -0.466885127  0.106314437  0.002324175 -0.026246549
2017  0.246619066 -0.346619680  0.188548375 -0.327278452  0.268584077  0.037135313
          Jul          Aug          Sep          Oct          Nov          Dec
2014 -0.055028229 -0.122397737  0.304100050 -0.243903568  0.275992280  0.386826690
2015 -0.078417077  0.090214984  0.055133632  0.423988544 -0.374943105  0.128844110
2016  0.176067809 -0.109521401  0.073237222 -0.320914332  0.230952133  0.023517034
2017 -0.215904811  0.301980614 -0.185263019  0.016342304  0.047712347  0.189479549
> Box.test(tss1,lag=1,type=c("Box-Pierce","Ljung-Box"),fitdf = 0)

        Box-Pierce test

data:  tss1
X-squared = 17.403, df = 1, p-value = 3.024e-05
> adf.test(tss1,alternative=c("stationary","explosive"),k=trunc((length(tss1)-1)^(1/3)))

        Augmented Dickey-Fuller Test

data:  tss1
Dickey-Fuller = -4.1191, Lag order = 3, p-value = 0.0128
alternative hypothesis: stationary
> kpss.test(tss1)
KPSS Test for Level Stationarity
data:  tss1
KPSS Level = 0.052542, Truncation lag parameter = 3, p-value = 0.1
> auto.arima(tss1)
Series: tss1
ARIMA(2,0,0) with zero mean
Coefficients:
         ar1      ar2
      -0.7624  -0.2536
s.e.   0.1409   0.1394
```

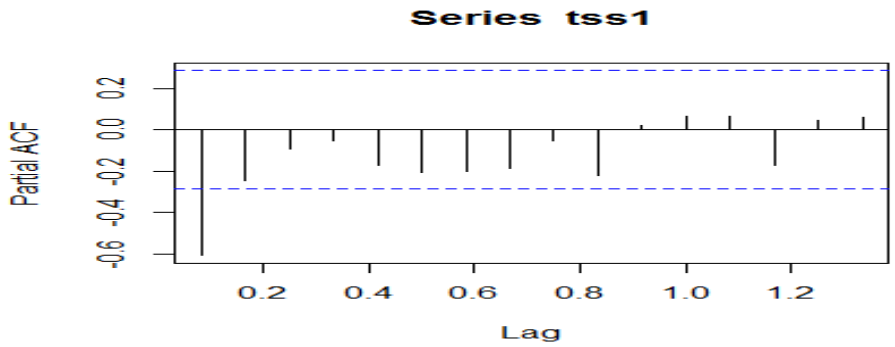
sigma^2 estimated as 0.0405:  log likelihood=9.39

AIC=-12.78  AICc=-12.22  BIC=-7.23

> acf(tss1)



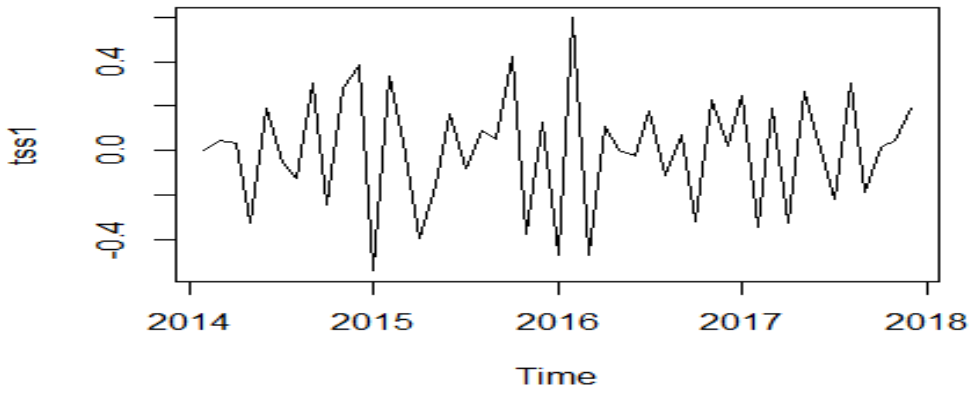

> pacf(tss1)




We used difference of logs and validated this by Ljung box test, adf test, kpss test as well as by ACF and PACF plots. The results are as follows-

1. For Ljung box test: p value is less  than 0.05 i.e we can reject null hypothesis assuming a 5% chance of error. This means the vales are dependent on each other.
2. For adf test: p value <0.05 indicates that time series is stationary.
3. For kpss test: as p value >0.05 this means that only differencing cannot be further used.
4. As now the transformed time series is stationary we can fit ARIMA model by using auto.arima we can go ahead with ar1 or ar2 model.
5. The ACF and PACF plots have aided in determining the MA and AR terms of model in visua l terms

>plot(tss1)




From the plot of tss1 we can see that the trend is stationary with seasonal variations.

## TIME SERIES ANALYSIS FOR PHARMACEUTICAL SHOP

The data that we obtained from the Pharmaceutical shop was daily data given for 5 years i.e. from 2014-2018. From the given data, we decided to combine the sales to give monthly sales to do the time series analysis on. The analysis will be done on the 4 year data i.e. from 2014 - 2017, 2018 data will be used to validate the analysis hence done.

```
s.q=PHARMACEUTICAL_SHOP_FINAL_DATA$`TOTAL SALES`
data=c(s.q);data
tss=ts(data,frequency = 12,start=c(2014,1));tss
plot.ts(tss)
tss.dec=decompose(tss);tss.dec
plot(tss.dec)
f=HoltWinters(tss);f
a=forecast(f,12)
summary(a)
plot(a)
res=tss.dec$random
plot(res)
shapiro.test(res)
```

```
res
r=as.vector(res);r
r=r[-c(1,2,3,4,5,6,43,44,45,46,47,48)];r
median(r)
r=r-median(r);r
a=c(0,1,1,1,1,0,1,0,1,1,1,0,1,0,0,0,1,1,1,1,1,1,0,1,0,0,0,1,0,1,0,1,0,0,0,0)
length(a)
a=as.factor(a)
runs.test(a)
ndiffs(tss)
tss1=diff(tss);tss1
ndiffs(tss1)
Box.test(tss1,lag=1,type=c("Box-Pierce","Ljung-Box"),fitdf = 0)
adf.test(tss1,alternative=c("stationary","explosive"),k=trunc((length(tss1)-1)^(1/3)))
kpss.test(tss1)
auto.arima(tss1)
acf(tss1)
pacf(tss1)
plot(tss1)
```

FOR PHARMACEUTICAL SHOP

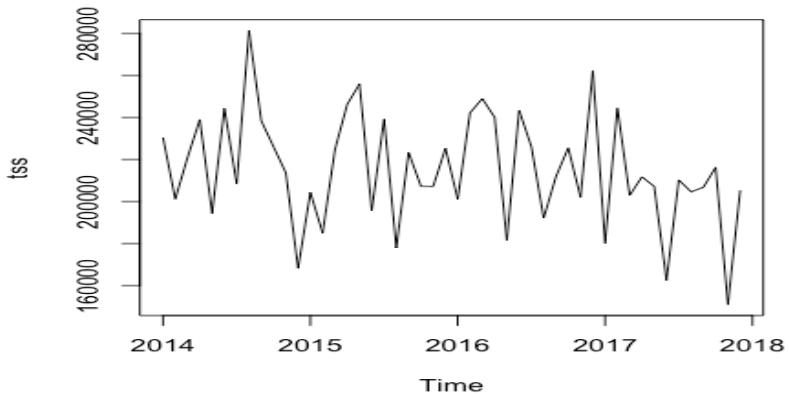
```
> s.q=PHARMACEUTICAL_SHOP_FINAL_DATA$`TOTAL SALES`
> data=c(s.q); data
 [1] 230501.2 201179.2 220983.7 239061.6 194345.6 244296.8 208531.1 281318.9
 [9] 238520.6 226103.1 213743.5 168210.1 204315.6 185025.0 224669.9 246219.9
[17] 255989.2 195701.0 239240.0 178039.0 223454.5 207324.0 207181.9 225363.2
[25] 201159.7 242203.4 248997.0 240120.9 181516.8 243468.2 226084.1 192141.8
[33] 211903.0 225554.2 202075.8 262196.1 180159.1 244562.9 202912.2 211796.7
[41] 207191.0 162505.7 210320.2 204619.9 206784.6 216272.7 150983.3 205400.3
```

The **ts()** function will convert the numeric vector into an R time series object. The format is **ts(**vector, **frequency= start=,)** where start are the times of the first observation and frequency is the number of observations per unit time (1=annual, 4=quarterly, 12=monthly, etc.).

```
> tss=ts(data,frequency = 12,start=c(2014,1)); tss
        Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
2014 230501.2 201179.2 220983.7 239061.6 194345.6 244296.8 208531.1 281318.9
2015 204315.6 185025.0 224669.9 246219.9 255989.2 195701.0 239240.0 178039.0
2016 201159.7 242203.4 248997.0 240120.9 181516.8 243468.2 226084.1 192141.8
2017 180159.1 244562.9 202912.2 211796.7 207191.0 162505.7 210320.2 204619.9
        Sep      Oct      Nov      Dec
2014 238520.6 226103.1 213743.5 168210.1
2015 223454.5 207324.0 207181.9 225363.2
2016 211903.0 225554.2 202075.8 262196.1
2017 206784.6 216272.7 150983.3 205400.3
```

To visualize our time series, we plotted the above time series.

```
> plot.ts(tss)
```




The decompose() function decomposes the time series into seasonal, trend, and irregular components using moving averages. It can be also used to identify whether the time series follows an additive or a multiplicative model. The function first determines the trend component using a moving average and removes it from the time series. Then, the seasonal figure is computed by averaging, for each time unit, over all periods. The seasonal figure is then centered. Finally, the irregular/error component is determined by removing trend and seasonal figure from the original time series.

```
> tss.dec=decompose(tss); tss.dec
$x
        Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
```

```
2014 230501.2 201179.2 220983.7 239061.6 194345.6 244296.8 208531.1 281318.9
2015 204315.6 185025.0 224669.9 246219.9 255989.2 195701.0 239240.0 178039.0
2016 201159.7 242203.4 248997.0 240120.9 181516.8 243468.2 226084.1 192141.8
2017 180159.1 244562.9 202912.2 211796.7 207191.0 162505.7 210320.2 204619.9
          Sep       Oct       Nov      Dec
2014 238520.6 226103.1 213743.5 168210.1
2015 223454.5 207324.0 207181.9 225363.2
2016 211903.0 225554.2 202075.8 262196.1
2017 206784.6 216272.7 150983.3 205400.3
$seasonal
             Jan          Feb       Mar          Apr          May       Jun
2014 -22034.9978 7724.3605 10826.3475 18589.8078 1784.4792 -12201.0651
2015 -22034.9978 7724.3605 10826.3475 18589.8078 1784.4792 -12201.0651
2016 -22034.9978 7724.3605 10826.3475 18589.8078 1784.4792 -12201.0651
2017 -22034.9978 7724.3605 10826.3475 18589.8078 1784.4792 -12201.0651
          Jul         Aug         Sep         Oct          Nov         Dec
2014 4728.1723 -2627.0047 4480.9028 144.9606 -11648.1476 232.1844
2015 4728.1723 -2627.0047 4480.9028 144.9606 -11648.1476 232.1844
2016 4728.1723 -2627.0047 4480.9028 144.9606 -11648.1476 232.1844
2017 4728.1723 -2627.0047 4480.9028 144.9606 -11648.1476 232.1844
$trend
             Jan       Feb       Mar       Apr       May       Jun       Jul       Aug
2014        NA        NA        NA        NA        NA        NA 221141.9 219377.7
2015 221975.2 218951.4 214020.3 212610.1 211554.2 213662.2 2 15912.1 218163.0
2016 219290.9 219330.3 219436.6 219714.9 220261.8 221583.7 222243.4 221466.7
2017 210100.1 209963.1 210269.8 209669.8 207154.2 202658.9        NA        NA
          Sep       Oct       Nov      Dec
2014 218858.2 219310.1 222176.8 222720.5
2015 221559.1 222318.6 218961.5 217848.7
2016 219644.8 216544.4 216434.0 214130.3
2017        NA        NA        NA       NA
```

$random

| | Jan | Feb | Mar | Apr | May | Jun |
|---|---|---|---|---|---|---|
| 2014 | NA | NA | NA | NA | NA | NA |
| 2015 | 4375.4294 | -41650.7605 | -176.8115 | 15020.0142 | 42650.4917 | -5760.1630 |
| 2016 | 3903.8323 | 15148.6465 | 18734.0312 | 1816.1987 | -40529.4946 | 34085.5810 |
| 2017 | -7905.9646 | 26875.4110 | -18183.9226 | -16462.9159 | -1747.7001 | -27952.1209 |

| | Jul | Aug | Sep | Oct | Nov | Dec |
|---|---|---|---|---|---|---|
| 2014 | -17338.9132 | 64568.1509 | 15181.4900 | 6648.0346 | 3214.8182 | -54742.5608 |
| 2015 | 18599.6861 | -37497.0193 | -2585.4839 | -15139.5855 | -131.4287 | 7282.2724 |
| 2016 | -887.4758 | -26697.8345 | -12222.7090 | 8864.8481 | -2710.0924 | 47833.5854 |
| 2017 | NA | NA | NA | NA | NA | NA |

$figure

[1] -22034.9978 7724.3605 10826.3475 18589.8078 1784.4792 -12201.0651

[7] 4728.1723 -2627.0047 4480.9028 144.9606 -11648.1476 232.1844

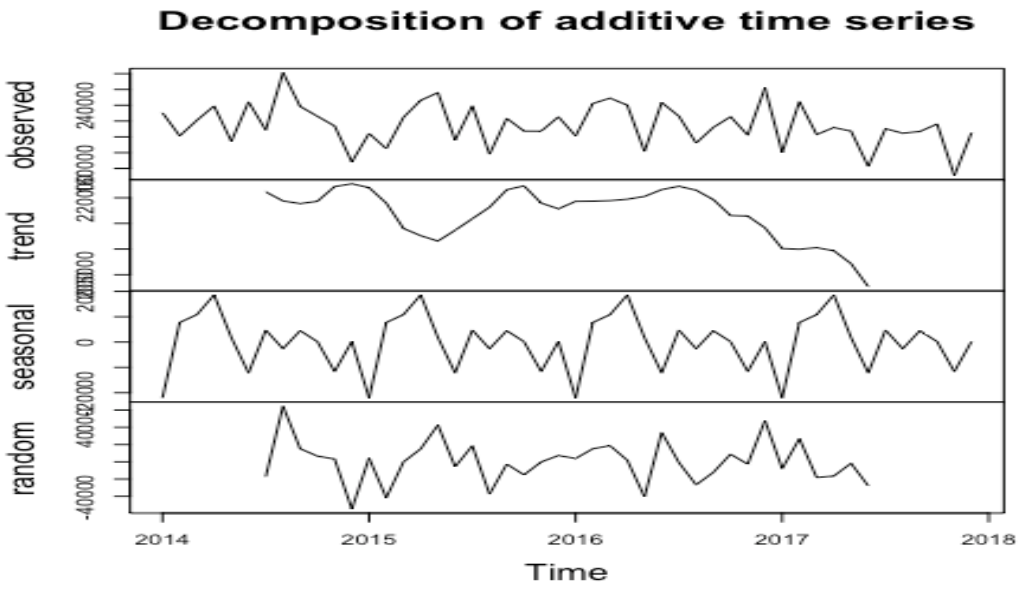
$type

[1] "additive"

attr(,"class")

[1] "decomposed.ts"

> plot(tss.dec)



From the above graph, the following interpretations can be made,

**TREND**- We can interpret that initially the sales trend is decreasing up until 2015, then there is an increase from mid-2015 to 2016. From 2016-2017, it can be seen that the trend is increasing, and after 2017 there is a steady decline.

**SEASONALITY**- We can interpret that seasonality is present, i.e. sales change In regular intervals of time (monthly).

**RANDOMNESS**- We can guess from the graph that the residuals are random.

For forecasting, we begin by using the Holt Winters Triple Exponential smoothing method.

```
> f=HoltWinters(tss);f
```

Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:

HoltWinters(x = tss)

Smoothing parameters:

**alpha: 0**

**beta : 0**

**gamma: 0.6737912**

Coefficients:

| | [,1] |
|---|---|
| a | 193966.2618 |
| b | -815.7163 |
| s1 | -19902.1340 |
| s2 | 31353.4528 |
| s3 | 9960.0756 |
| s4 | 17124.9612 |
| s5 | 3004.3081 |
| s6 | -19188.7064 |
| s7 | 13302.1488 |
| s8 | 988.1866 |
| s9 | 9366.1975 |
| s10 | 17943.8720 |

s11 -30959.3782

s12  19412.0847

> a=forecast(f,12)

> summary(a)

Forecast method: HoltWinters

Model Information:

Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:

HoltWinters(x = tss)

Smoothing parameters:

**alpha: 0**

**beta : 0**

**gamma: 0.6737912**

Coefficients:

[,1]

a  193966.2618

b  -815.7163

s1  -19902.1340

s2  31353.4528

s3  9960.0756

s4  17124.9612

s5  3004.3081

s6  -19188.7064

s7  13302.1488

s8  988.1866

s9  9366.1975

s10  17943.8720

s11 -30959.3782

s12  19412.0847

To forecast the future sales for the next 12 months, we have used the forecast() function and then take the summary of the forecast to get the details displayed .

Error measures:

ME RMSE MAE MPE MAPE MASE ACF1

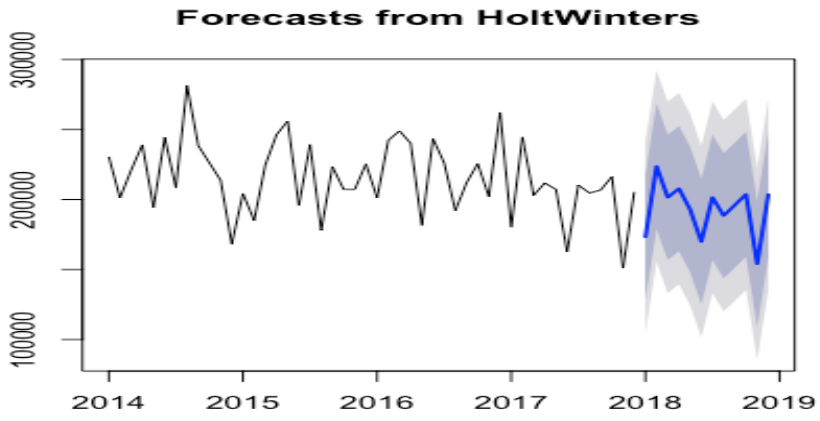
Training 2160.456 34420.88 23656.41 -0.4405699 11.3853 0.8019968 -0.089set

Forecasts:

(The below table shows the 80% and 95% confidence interval under which the sales will lie for the next 12 months,and we will validate with actual values. We can see that the values do lie in the confidence interval, this validates our model.)

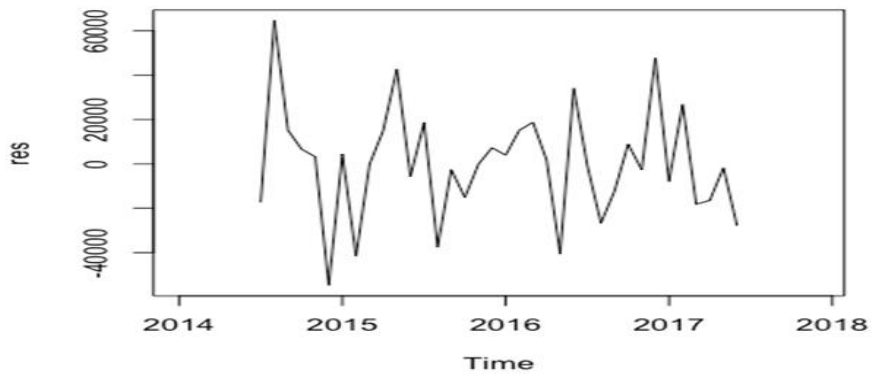
| | Lo 80 | Hi 80 | Lo 95 | Hi 95 | actual value |
|---|---|---|---|---|---|
| Jan 2018 | 128598.8 | 217898.1 | 104962.65 | 241534.2 | 237054.579 |
| Feb 2018 | 179038.6 | 268337.9 | 155402.52 | 291974.0 | 175655.709 |
| Mar 2018 | 156829.5 | 246128.8 | 133193.43 | 269764.9 | 196501.56 |
| Apr 2018 | 163178.7 | 252478.0 | 139542.60 | 276114.1 | 209696.115 |
| May 2018 | 148242.3 | 237541.6 | 124606.23 | 261177.7 | 144182.062 |
| Jun 2018 | 125233.6 | 214532.9 | 101597.50 | 238169.0 | 227960.832 |
| Jul 2018 | 156908.7 | 246208.1 | 133272.64 | 269844.2 | 238138.914 |
| Aug 2018 | 143779.1 | 233078.4 | 120142.96 | 256714.5 | 238072.249 |
| Sep 2018 | 151341.4 | 240640.7 | 127705.25 | 264276.8 | 153192.094 |
| Oct 2018 | 159103.3 | 248402.6 | 135467.21 | 272038.7 | 193527.298 |
| Nov 2018 | 109384.3 | 198683.7 | 85748.24 | 222319.8 | 189372.084 |
| Dec 2018 | 158940.1 | 248239.4 | 135303.99 | 271875.5 | 236820.441 |

> plot(a)



In the above graph we can see sales of 2018 are predicted. The confidence bands are also shown.

```
> res=tss.dec$random
> plot(res)
```




The plot of residuals is plotted. The residuals are called from the decomposed data. Now we check the normality of the residuals. Normality of residuals is an assumption of running a linear model. If the residuals are normal, it means that the model is valid and the confidence interval and model predictions are valid. We check normality by Shapiro Wilkes test.

```
> shapiro.test(res)
```

Shapiro-Wilk normality test

data: res

W = 0.98023, p-value = 0.7531

As the p value is greater than 0.05 we can say that the residuals are normally distributed.

Now we check the randomness of residuals by run test. If the residuals are random i.e. it is just a white noise, we have captured all the components of the time series in our model and it is a good one The run test procedure is as follows- we take the median and compare it with each value. the values below are put as 0 and above the median as 1. We convert this vector of 0's and 1's as factor and run the test.

```
> res
             Jan       Feb          Mar       Apr       May          Jun
2014          NA        NA           NA        NA        NA           NA
2015   4375.4294 -41650.7605  -176.8115 15020.0142 42650.4917 -5760.1630
2016   3903.8323  15148.6465 18734.0312  1816.1987 -40529.4946 34085.5810
2017  -7905.9646  26875.4110 -18183.9226 -16462.9159 -1747.7001 -27952.1209
             Jul         Aug          Sep          Oct          Nov          Dec
2014 -17338.9132   64568.1509   15181.4900   6648.0346    3214.8182   -54742.5608
2015  18599.6861  -37497.0193  -2585.4839   -15139.5855   -131.4287    7282.2724
2016   -887.4758   -26697.8345   -12222.7090   8864.8481   -2710.0924   47833.5854
2017          NA          NA           NA           NA           NA           NA
> r=as.vector(res);r;
 [1]          NA          NA          NA          NA          NA          NA
 [7] -17338.9132  64568.1509  15181.4900   6648.0346   3214.8182 -54742.5608
[13]   4375.4294 -41650.7605   -176.8115  15020.0142  42650.4917  -5760.1630
[19]  18599.6861 -37497.0193  -2585.4839 -15139.5855   -131.4287   7282.2724
[25]   3903.8323  15148.6465  18734.0312   1816.1987 -40529.4946  34085.5810
[31]   -887.4758 -26697.8345 -12222.7090   8864.8481  -2710.0924  47833.5854
[37]  -7905.9646  26875.4110 -18183.9226 -16462.9159  -1747.7001 -27952.1209
[43]          NA          NA          NA          NA          NA          NA
> r=r[-c(1,2,3,4,5,6,43,44,45,46,47,48)]; r
 [1] -17338.9132  64568.1509  15181.4900   6648.0346   3214.8182 -54742.5608
 [7]   4375.4294 -41650.7605   -176.8115  15020.0142  42650.4917  -5760.1630
[13]  18599.6861 -37497.0193  -2585.4839 -15139.5855   -131.4287   7282.2724
[19]   3903.8323  15148.6465  18734.0312   1816.1987 -40529.4946  34085.5810
[25]   -887.4758 -26697.8345 -12222.7090   8864.8481  -2710.0924  47833.5854
[31]  -7905.9646  26875.4110 -18183.9226 -16462.9159  -1747.7001 -27952.1209
> median(r)
[1] -154.1201
> r=r-median(r);r;
[1] -17184.79315  64722.27099  15335.61007  6802.15467   3368.93834 -54588.44068
```

```
 [7]   4529.54948 -41496.64036    -22.69136  15174.13434  42804.61183  -5606.04292
[13]  18753.80621 -37342.89919  -2431.36377 -14985.46544     22.69136   7436.39253
[19]   4057.95240  15302.76660  18888.15126   1970.31884 -40375.37445  34239.70108
[25]   -733.35567 -26543.71441 -12068.58891   9018.96816  -2555.97231  47987.70554
[31]  -7751.84449  27029.53114 -18029.80251 -16308.79580  -1593.57999 -27798.00077
> a=c(0,1,1,1,1,0,1,0,1,1,1,0,1,0,0,0,1,1,1,1,1,1,0,1,0,0,0,1,0,1,0,1,0,0,0,0)
> length(a)
[1] 36
> a=as.factor(a)
> runs.test(a)
Runs Test
data:  a
Standard Normal = 0.018849, p-value = 0.985
alternative hypothesis: two.sided
```

As the p value is greater than 0.5 we say that the residuals are random. Now we will try to make our t ime series stationary. Firstly, we will try the differencing method .

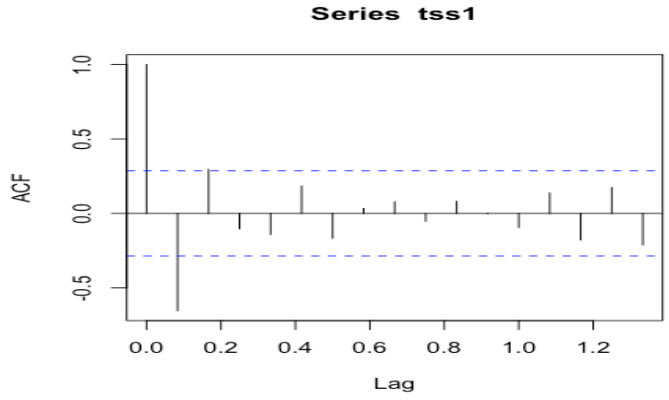
```
> ndiffs(tss)
[1] 0
> tss1=diff(tss); tss1
```

| | Jan | Feb | Mar | Apr | May | Jun |
|---|---|---|---|---|---|---|
| 2014 | | -29322.0064 | 19804.5092 | 18077.9802 | -44716.0322 | 49951.1514 |
| 2015 | 36105.5183 | -19290.6248 | 39644.8544 | 21550.0697 | 9769.2865 | -60288.2203 |
| 2016 | -24203.4820 | 41043.6290 | 6793.6757 | -8876.0924 | -58604.1836 | 61951.4798 |
| 2017 | -82037.0013 | 64403.8239 | -41650.6950 | 8884.4702 | -4605.6925 | -44685.3074 |

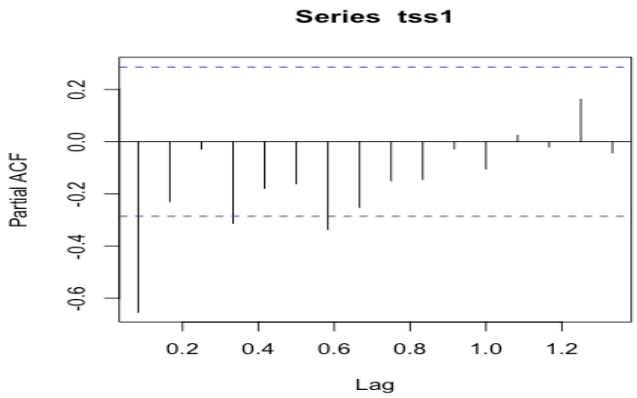
| | Jul | Aug | Sep | Oct | Nov | Dec |
|---|---|---|---|---|---|---|
| 2014 | -35765.6277 | 72787.7332 | -42798.2521 | -12417.5449 | -12359.5798 | -45533.3878 |
| 2015 | 43538.9695 | -61200.9471 | 45415.5063 | -16130.5364 | -142.0953 | 18181.3156 |
| 2016 | -17384.1419 | -33942.2467 | 19761.1480 | 13651.2378 | -23478.4651 | 60120.3311 |
| 2017 | 47814.4788 | -5700.2495 | 2164.6296 | 9488.1933 | -65289.4115 | 54416.9736 |

```
> ndiffs(tss1)
[1] 0
```

```
> Box.test(tss1,lag=1,type=c("Box-Pierce","Ljung-Box"),fitdf = 0)

   Box-Pierce test

data:  tss1

X-squared = 20.096, df = 1, p-value = 7.363e-06

> adf.test(tss1,alternative=c("stationary","explosive"),k=trunc((length(tss1)-1)^(1/3)))

        Augmented Dickey-Fuller Test

data:  tss1

Dickey-Fuller = -5.1676, Lag order = 3, p-value = 0.01

alternative hypothesis: stationary

> kpss.test(tss1)

KPSS Test for Level Stationarity

data:  tss1

KPSS Level = 0.03471, Truncation lag parameter = 3, p-value = 0.1

> auto.arima(tss1)

Series: tss1

ARIMA(0,0,1) with zero mean

Coefficients

          ma1

      -0.9245

s.e.   0.0709

sigma^2 estimated as 742606727:  log likelihood=-547.15

AIC=1098.3   AICc=1098.58   BIC=1102

> acf(tss1)
```



> pacf(tss1)




We observe that the ndiffs is 0 i.e. the time series is already stationary. We validated this by Ljung bo x test, adf test, kpss test as well as by ACF and PACF plots. The results are as follows-

1.For Ljung box test: p value is less than 0.05 i.e. we accept the null hypothesis that the model does n ot show lack of fit.

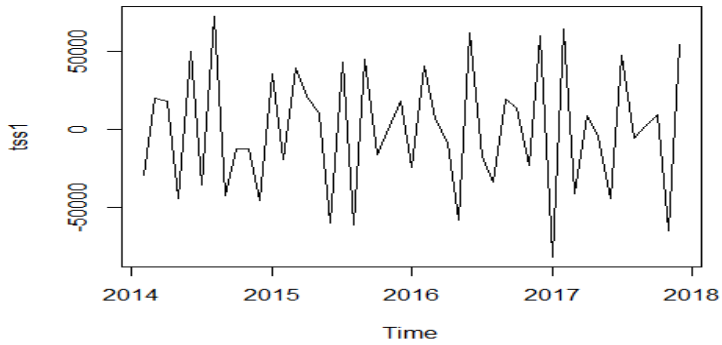
2.For adf test: p value < 0.05 indicates that time series is stationary.

3.For kpss test: as p value >0.05 this means that only differencing cannot be further used.

4. ARIMA model can be fit as time series is stationary. By using auto.arima we can go ahead with ar1 or ar2 model.

5.The ACF and PACF plots have aided in determining the MA and AR terms of model in visual terms

>plot(tss1)

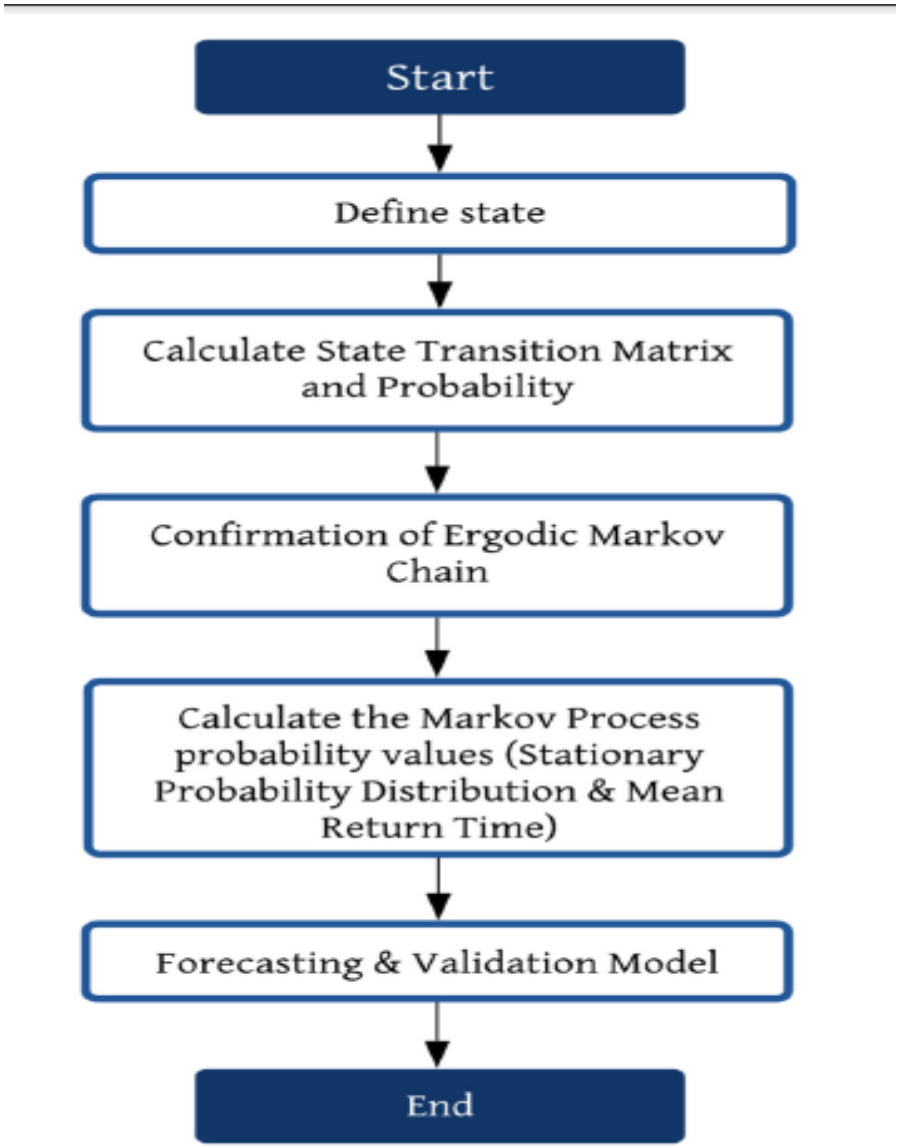



From the plot of tss1 we can see that the trend is stationary with seasonal variations.

# MARKOV CHAIN

## Introduction

Depending upon the condition of sales market, it is important to understand the market dynamics in time. Therefore, a reasonable prediction and analysis of the sales has an important significance. **Sales of a particular shop is a random process changing over time, and many of the times sales on a day is related to the previous day or day before yesterday's sales but the association with long dated is small, this is in accordance with the Markov property**. Therefore, we can apply Markov theory to the analysis and forecast sales.



Data contains the daily sales of a shop in Rs. at the end of the day. Here sales in Rs is a continuous random variable measured at the end of the day which is discrete time point. To form Markov chain (i.e. discrete state space and discrete time point), we classified sales into equidistant classes and hence state space becomes discrete, and then transition probability matrix for each shop is calculated and its stationarity is checked.

## Pharmaceutical shop

State space

s1:Sales in between 100000 to 150000 Rs.

s2: Sales in between 150000 to 200000 Rs.

s3: Sales in between 200000 to 250000 Rs.

s4: Sales in between 250000 to 300000 Rs.

**state space S1 ={1,2,3,4}**

**$X_n$: Class to which sales at the end of nth day belongs**

Transition probability matrix for Pharmaceutical shop

| $X_n$ \ $X_{n+1}$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0.2 | 0.8 | 0 |
| 3 | 0.025 | 0.25 | 0.65 | 0.075 |
| 4 | 0 | 0.666667 | 0.333333 | 0 |

Above matrix is stationary after 8th power.

**Initial Distribution-**

| $X_0$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $P(X_0=x_0)$ | 0 | 0 | 1 | 0 |

**Forecast-**

| | | | |
|---|---|---|---|
| 0.025 | 0.25 | 0.65 | 0.075 |

## Departmental store

State space

s1:Sales in between 100000 to 150000 Rs.

s2: Sales in between 150000 to 200000 Rs.

s3: Sales in between 200000 to 250000 Rs.

s4: Sales in between 250000 to 300000 Rs.

**state space S2 ={1,2,3,4}**

**$X_n$: Class to which sales at the end of nth day belongs**

Transition probability matrix for Departmental store

| $X_n$ \ $X_{n+1}$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 0.9 | 0.1 | 0 |
| 2 | 0.184211 | 0.605263 | 0.157895 | 0.052632 |
| 3 | 0.333333 | 0.444444 | 0.222222 | 0 |
| 4 | 0 | 1 | 0 | 0 |

Above matrix is stationary after $10^{th}$ power.

**Initial Distribution-**

| $X_0$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $P(X_0=x_0)$ | 0 | 1 | 0 | 0 |

**Forecast-**

| | | | |
|---|---|---|---|
| 0.18421053 | 0.60526 | 0.15789 | 0.05263 |

**RESULT:**

| FORECAST | NEXT MONTH'S TOTAL SALES RANGE (IN RS.) |
|---|---|
| PHARMACEUTICAL SHOP | 200000-250000 |
| DEPARTMENTAL STORE | 150000-200000 |

# SHELF OPTIMAZATION USING LINEAR PROGRAMMING

## Introduction

Shelf space maximization is one of key factors behind any marketing strategy for a brand. Here we will explain some challenges in shelf space optimization and then solve an example using excel.

## Lifts Table

Let us assume a retail store with 3 racks, Rack 1, Rack2 and Rack3 with 3, 3 and 4 shelves as shown in the below table. We have to stock products of 3 brands say L'Oreal ,Godrej and Garnier. L'Oreal ,Godrej and Garnier have 3, 3 and 2 products respectively. The numbers that we see in the matrix are the lifts (increase in sales) that we achieve on placing a specific product on a specific rack / shelf (given by corresponding row).

| Rack | Shelf | L'OREAL | | | GODREJ | | | GARNIER | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1)Ebony Black | 2)Light Brown | 3)Deep Plum | 4) Black Brown | 5)Dark Brown | 6)Honey Brown | 7)Darkest Brown | 8)Black Natural |
| 1 | 1 | 950 | 818 | 650 | 848 | 630 | 648 | 842 | 842 |
| | 2 | 691 | 849 | 615 | 700 | 653 | 598 | 563 | 563 |
| | 3 | 427 | 413 | 349 | 347 | 407 | 237 | 465 | 465 |
| 2 | 4 | 345 | 153 | 282 | 301 | 477 | 432 | 313 | 313 |
| | 5 | 464 | 470 | 126 | 392 | 534 | 312 | 326 | 326 |
| | 6 | 281 | 144 | 283 | 200 | 168 | 107 | 148 | 148 |
| 3 | 7 | 238 | 500 | 291 | 434 | 465 | 488 | 544 | 544 |
| | 8 | 138 | 86 | 119 | 149 | 92 | 136 | 119 | 119 |
| | 9 | 127 | 124 | 141 | 54 | 130 | 140 | 75 | 75 |
| | 10 | 70 | 141 | 78 | 106 | 69 | 51 | 58 | 58 |

Now due to a difference in profit margin / inventory cost / demand / expiration date etc. of products, the store wants to optimize the placement of each product on the shelves and maximize the total sales, taking into account the constraints it has got.

## Decision Variables

The decision variables will be in the form of a matrix of the same size as lift (10*8). The matrix will have binary values, 1 indicating a YES for the product/shelf pair and 0 indicating a NO. We will start with a matrix of all 0's and allow the solver to make changes to 1's where required.

## Objective Function

The objective function to be maximized is the Total Sales of all merchandize.

## Constraints

The constraints used here are:

1.One Shelf can have at max one product of any company. (row constraint)

2.The products cannot be marketed more than a particular number of times. This is given in the order of the products as shown in above fig. (Column constraint). The maximum occurrences of the products are as below. These constraints can be attributed to the product type/profit margin/demand or any other rationale applicable at the store.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 2 | 2 | 3 | 1 | 1 |

This boils down to the conditions that Product 1 from X cannot be marketed more than once. Similarly, for the other products the constraints apply.There can be several more constraints applied as per the business understanding of a store and merchandizing best practices.

## Linear Optimization using Excel

CONSTRAINTS:

| SUM OF EACH COLUMN <= | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| BRAND | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 |
| PRODUCT | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 |
| CONSTRAINT | <= | <= | <= | <= | <= | <= | <= | <= |
| VALUE | 1 | 0 | 0 | 2 | 2 | 3 | 1 | 1 |

CONSTRAINTS:

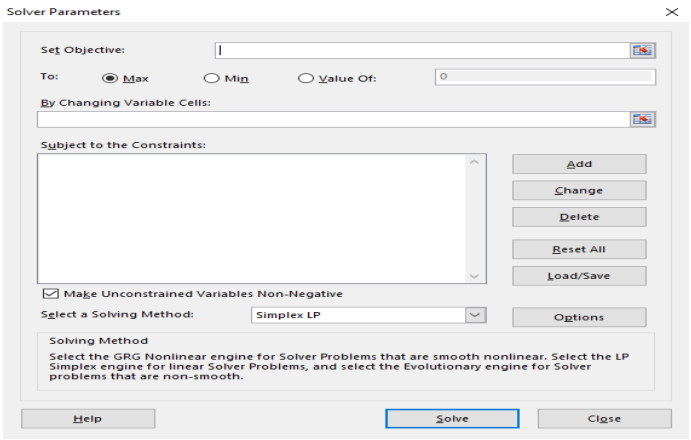
| SUM OF EACH ROW <= | | |
|---|---|---|
| SHELF NO. | CONSTRAINT | VALUE |
| SHELF 1 | <= | 1 |
| SHELF 2 | <= | 1 |
| SHELF 3 | <= | 1 |
| SHELF 4 | <= | 1 |
| SHELF 5 | <= | 1 |
| SHELF 6 | <= | 1 |
| SHELF 7 | <= | 1 |
| SHELF 8 | <= | 1 |
| SHELF 9 | <= | 1 |
| SHELF 10 | <= | 1 |

Constraints can be taken care using the above two tables in excel.

1.Constraint 1 will always hold true when the sum of each column<=the list of the column constraints as shown before.

2.Constraint 2 will always be satisfied when the sum of the rows<=1

Let us go to the Solver in Excel. Go to DATA → Solver. If it's not visible you need to activate it by going to File → Options → Add-Ins




This is how it looks like.

**Set Objective**

The Objective function is given by the sum product of the lift and the decision variable matrix. Select the cell in spreadsheet which indicates this.

**We have to maximize the Profit.**

**Decision variable** is the matrix of same size as the lift. Select all cells representing it.

For **constraints** select the cells that represent the Sum of rows and Sum of columns in the decision variable matrix. Assign them <= inequality. For all rows the sum <=1 and for columns it is given by the list of constraints as given in problem.

Add another constraint to make the decision variables binary integers. (0's and 1's).

Select Simplex LP and run.

The objective function along with the constraints is solved and the maximum sales obtained is 4197.

The decision variable matrix obtained is shown below:

| Rack | Shelf | L'OREAL | | | GODREJ | | | GARNIER | | SUM OF ROWS |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 1)Ebony Black | 2)Light Brown | 3)Deep Plum | 4) Black Brown | 5)Dark Brown | 6)Honey Brown | 7)Darkest Brown | 8)Black Natural | |
| | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | 2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| | 4 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| | 5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 2 | 6 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| | 8 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| | 9 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 3 | 10 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| SUM OF COLUMNS | | 1 | 0 | 0 | 2 | 2 | 3 | 1 | 1 | |

We can decide the optimal allocation of position as follows:

| MAXIMUM SALES OBTAINED | 4197 |
|---|---|

| BRAND | PRODUCT | RACK | SHELF |
|---|---|---|---|
| L'OREAL | 1)Ebony Black | 1 | 1 |
| L'OREAL | 2)Light Brown | N/A | N/A |
| L'OREAL | 3)Deep Plum | N/A | N/A |
| GODREJ | 4) Black Brown | 1 | 2 |
| GODREJ | 4) Black Brown | 2 | 6 |
| GODREJ | 5)Dark Brown | 2 | 4 |
| GODREJ | 5)Dark Brown | 2 | 5 |
| GODREJ | 6)Honey Brown | 3 | 8 |
| GODREJ | 6)Honey Brown | 3 | 9 |
| GODREJ | 6)Honey Brown | 3 | 10 |
| GARNIER | 7)Darkest Brown | 3 | 7 |
| GARNIER | 8)Black Natural | 1 | 3 |

We can notice as there is no advertisement for brand L'Oreal product Light Brown and Deep Plum advertisement is one of the constraints the LPP shows that we do not require to stock these.

But Excel has its limitations and cannot be used for a problems of large size.

### Challenges with Large Datasets

Let us understand what problems arise with large datasets. As in this example we understand that each decision variable can take values 0 or 1 that is 2^1 or 2 possible values. For 2 decision variables the total number of possible combinations can be 2^2 or 4 out of which one/more may give the optimized value of the Objective function. With 80 decision variables in our example, the total combinations is 2^80. This shows that the order of the problem is exponential and not linear. [In language of **Computational Complexity Theory**, exponential time O (2^n)]. Problems of exponential order are very intensive even for the best of computers. As in our example each of the 2^80 combinations will be evaluated to find the optimized solution.

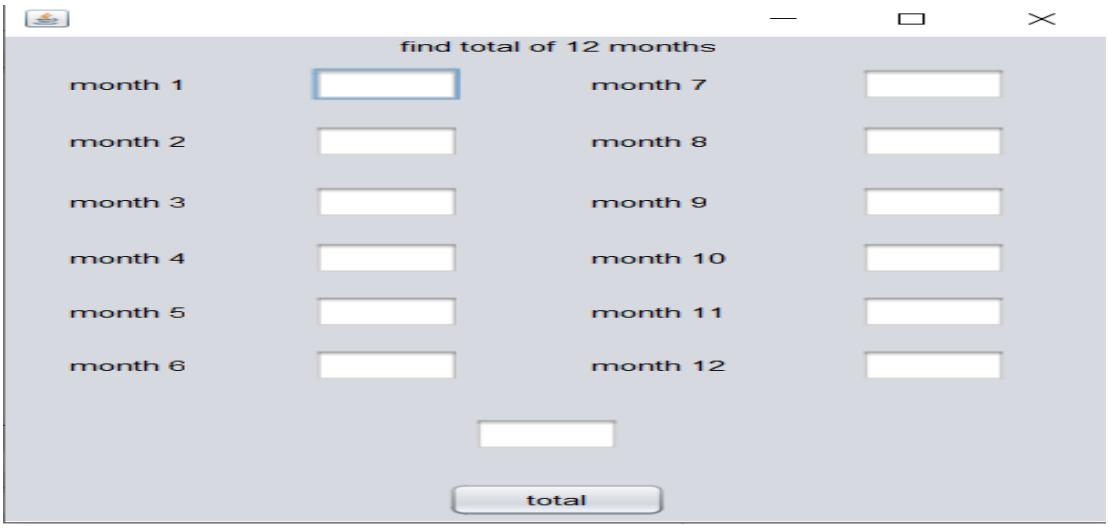
# INVENTORY MODEL (GENERALISED)

The following inventory model is designed with the aim to provide retailers with an efficient and easy to use method to calculate the amount of units of a particular product to be restocked.

Working of the model:

- The user (i.e. retailer) will first enter the month wise quantity of a particular product sold for the last 12 months. The model will compute the total number of units sold.
- Then the user will enter the time period for which he wants to restock (i.e. 15 days,1 month or 3 months),the number of units sold in the last 12 months, the number of units unsold in the previous month and the number of units he wants to stock extra.
- The model will compute the range of units to be stocked.

Quantity sold in previous 12 Months

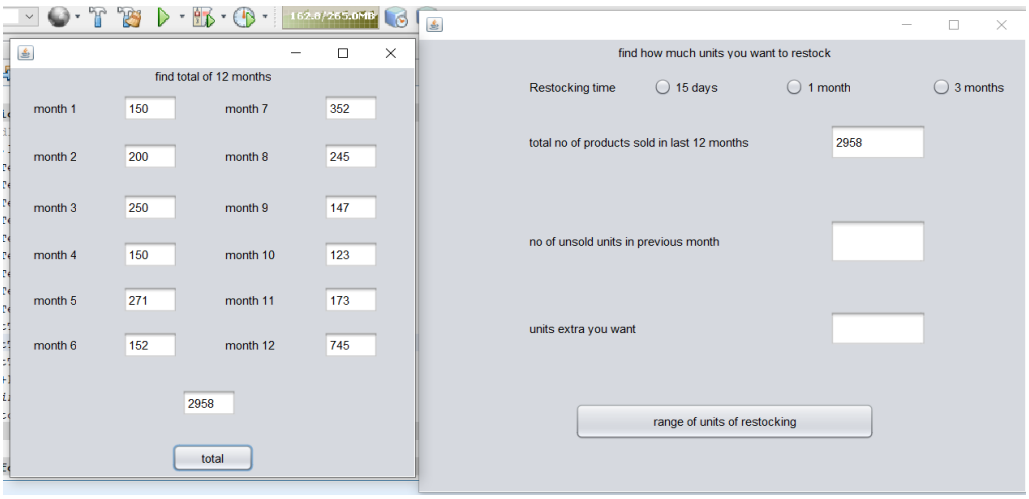



Code:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
   int a,b,c,d,e,f,g,h,i,j,k,l,total;
   a=Integer.parseInt(t1.getText()); b=Integer.parseInt(t2.getText());c=Integer.parseInt(t3.getText());
   d=Integer.parseInt(t4.getText());e=Integer.parseInt(t5.getText());f=Integer.parseInt(t6.getText());
   g=Integer.parseInt(t7.getText());h=Integer.parseInt(t8.getText());i=Integer.parseInt(t9.getText());
 j=Integer.parseInt(t10.getText());k=Integer.parseInt(t11.getText());l=Integer.parseInt(t12.getText());
   total=a+b+c+d+e+f+g+h+i+j+k+l;t13.setText(Integer.toString(total));
```

new eg2(Integer.toString(total)).setVisible(true);}

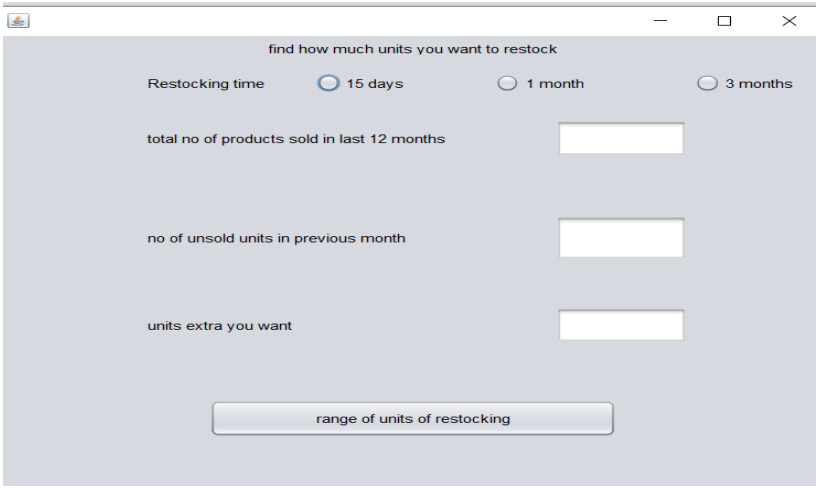
Output :




| | | | | | |
|---|---|---|---|---|---|
| Month 1 | 150 | Month 5 | 271 | Month 9 | 147 |
| Month 2 | 200 | Month 6 | 152 | Month 10 | 123 |
| Month 3 | 250 | Month 7 | 362 | Month 11 | 173 |
| Month 4 | 150 | Month 8 | 245 | Month 12 | 745 |

Total = 2958

Program for restocking :

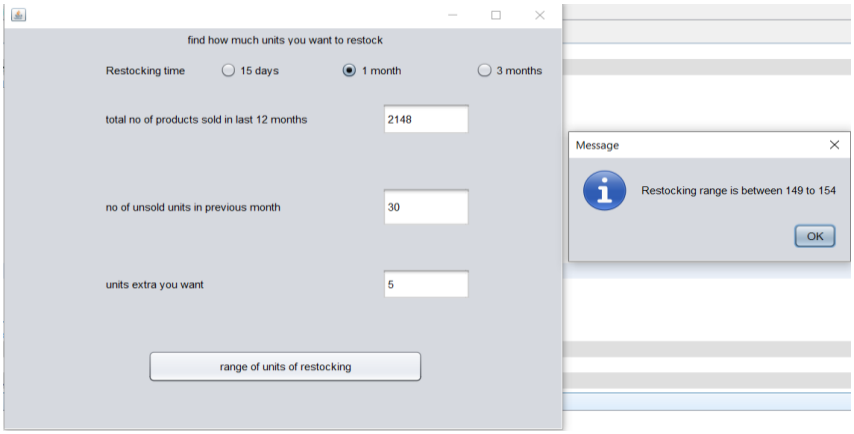



Code :

```
private void
jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
 // TODO add your handling code here:
```

```
int unitsold=0,avg,r11=0,unsoldunit,restock,extra;
unsoldunit=Integer.parseInt(t2.getText());
if(r1.isSelected()==true){r11=unitsold/24;}
else if(r2.isSelected()==true){r11=unitsold/12;}
else if(r3.isSelected()==true){r11=unitsold/4;}
extra=Integer.parseInt(t3.getText());restock=r11-unsoldunit;
JOptionPane.showMessageDialog(this,"Restocking range is between "+Integer.toString(restock)+"
to "+Integer.toString(restock+extra));
System.exit(0);}
```

Output :




Total no of products sold in last 12 months = 2148

No unsold units in previous month = 30

Extra units needed = 5

**Restocking range is between 149 to 154.**

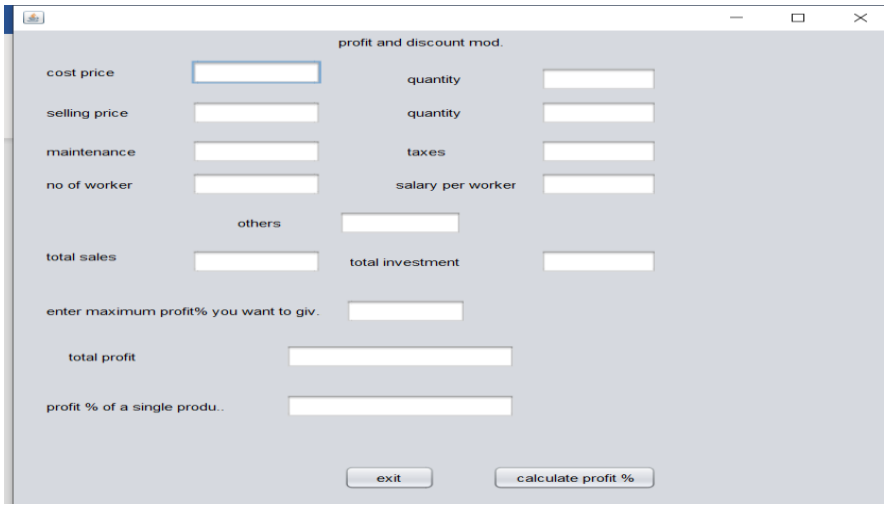
## PROFIT AND DISCOUNT MODEL

Following is a model for calculating the profit of a retailer and the discount he can give on a particular product.

Working of the model:

- The user (i.e. retailer) will enter the cost price and selling price of a single product along with the quantity.
- He will then enter the maintenance costs,amount of taxes that he pays ,the number of workers that he has,the salary he gives to each worker,his total sales and total investment.

- The user will then enter the maximum percentage of profit that he wants on the particular product.
- The model will return the total profit and profit percentage on a single product to the user.The model will also give the the amount of discount that user can give on that product.

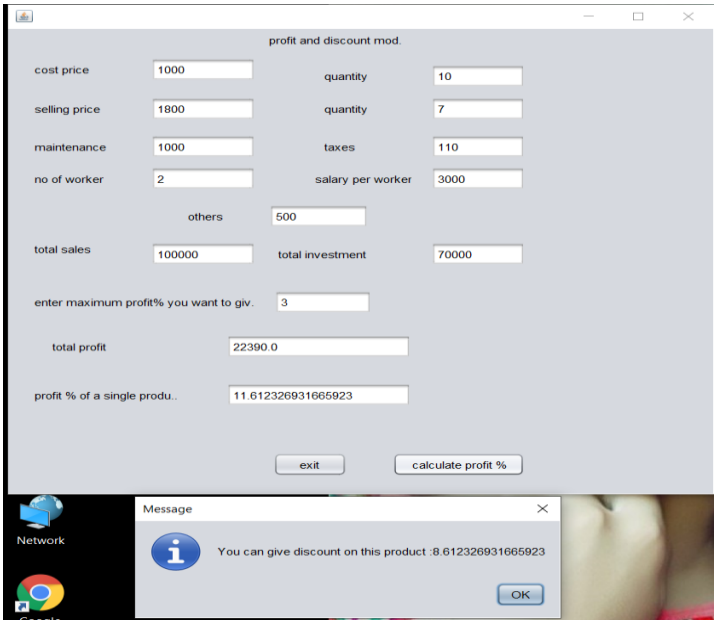



Code :

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        double cp,sp,q1,q2,m,nw,sw,t,o,ts,ti,percent,profitp,aprofit,extra,discount;
cp=Double.parseDouble(t1.getText());q1=Double.parseDouble(t2.getText());
sp=Double.parseDouble(t3.getText());q2=Double.parseDouble(t4.getText());
m=Double.parseDouble(t5.getText());t=Double.parseDouble(t6.getText());
nw=Double.parseDouble(t7.getText());sw=Double.parseDouble(t8.getText());
o=Double.parseDouble(t10.getText());ts=Double.parseDouble(t12.getText());
ti=Double.parseDouble(t13.getText());extra=Double.parseDouble(t01.getText());
profitp=sp*q2-cp*q1; aprofit=ts-ti-m-t-nw*sw-o;
t02.setText(Double.toString(aprofit));percent=((profitp/aprofit)*100);
t11.setText(Double.toString(percent));extra=Double.parseDouble(t01.getText());
discount=percent-extra;
if(percent>extra)
{ JOptionPane.showMessageDialog(null,"You can give discount on this product
:"+Double.toString(discount));}}
```

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        System.exit(0);}
```

Output :




| Cost price | 1000 | Selling price | 1900 | Maintenance | 1000 |
|---|---|---|---|---|---|
| Quantity bought | 10 | Quantity sold | 7 | Taxes | 110 |
| No. of workers | 2 | Salary per worker | 3000 | Others | 500 |
| Total sales | 100000 | Total investment | 70000 | Maximum profit % required | 3 |

Total profit = 22390

Profit percent of a single product = 11.612326931665923

Discount that you can give on this product is 8.612326931665923 %

*All values are in Rupees.

## CONCLUSION

From the following analysis that we have performed, the businesses can plan for the future; whether investments should be done or not, whether their business can be expanded, whether they can afford to take risks, and aid in planning in a strategic manner the way in which they can sell their products. They can create a safety net and avoid serious backlash and going into major losses.

We have mentioned confidence intervals under which they can know the maximum as well as the minimum amount that they will earn in upcoming year. This can help the business decide which approach to use- optimistic, pessimistic or realistic approach.

By an analytic approach that we have used, we wish to give them the confidence in the decisions they make as they are backed by useful data interpretations.

## LIMITATIONS

The information needed for Space Optimization is most times unclear or complex throughout the business. Certain products may play a vital role (being essential to promotions program for instance); others may be duplicates, but provide higher margins etc. Hence, it may become difficult to measure them on a single parameter.

In r software we cannot handle big data sets so we had to convert our daily sales data into monthly total sums. Similarly, for shelf optimization Excel cannot handle very large data sets so we have to take smallest subset like taking only 3 brands and 3 products of each brand.

## REFERENCES


1.https://towardsdatascience.com/detecting-stationarity-in-time-series-data-d29e0a21e638

2.https://www.analyticsvidhya.com/blog/2016/09/a-beginners-guide-to-shelf-space-optimization-using-linear-programming/

3..https://www.statisticshowto.datasciencecentral.com/kpss-test/

4.https://nwfsc-timeseries.github.io/atsa-labs/sec-boxjenkins-kpss.html

5..https://dimensionless.in/beginners-guide-for-time-series-forecasting/

6..https://emanuelaf.github.io/pred-sales.html

7..Nicolas Lanchier- Stochastic Modeling

8..Modelling Financial Time Series Second Edition I- A.J.TAYLOR Laucaster University, UK.

9.Cassandra Data Modeling and Analysis-C.Y. KAN