Untitled

Esther_Wairimu_Kamau

1/22/2021

```
#we load our dataset

library("readr")

jr <- read.csv("http://bit.ly/CarreFourSalesDataset")

head(jr,n=3)
```

1.Data loading and exploration

```
##       Date    Sales
## 1 1/5/2019 548.9715
## 2 3/8/2019  80.2200
## 3 3/3/2019 340.5255
```

```
tail(jr,n=3)
```

```
##           Date   Sales
## 998   2/9/2019  33.432
## 999  2/22/2019  69.111
## 1000 2/18/2019 649.299
```

```
#we check the column names we have
# we have only two columns for date and sales

names(jr)
```

```
## [1] "Date"  "Sales"
```

```
#we can also check the no of rows and columns

dim(jr)
```

```
## [1] 1000    2
```

```r
#we can also go ahead and check for the datatypes

str(jr)
```

```
## 'data.frame':    1000 obs. of  2 variables:
##  $ Date : chr  "1/5/2019" "3/8/2019" "3/3/2019" "1/27/2019" ...
##  $ Sales: num  549 80.2 340.5 489 634.4 ...
```

we observe that we have two columns only with one being str(date/time) and the other numerical

```r
#we can go ahead and check for the number missing values per column

colSums(is.na(jr))
```

```
##  Date Sales
##     0     0
```

we observe that we dont have any missing values

```r
#we can go ahead and check for duplicated values
duplicated_rows <- jr[duplicated(jr),]

dim(duplicated_rows)
```

```
## [1] 0 2
```

We observe that we dont have any duplicated values in all columns

```r
uniqueitems <- unique(jr)

dim(uniqueitems)
```

```
## [1] 1000    2
```

We observe that the whole dataset is unque meaning each date is unique and sales for that day too is unique.

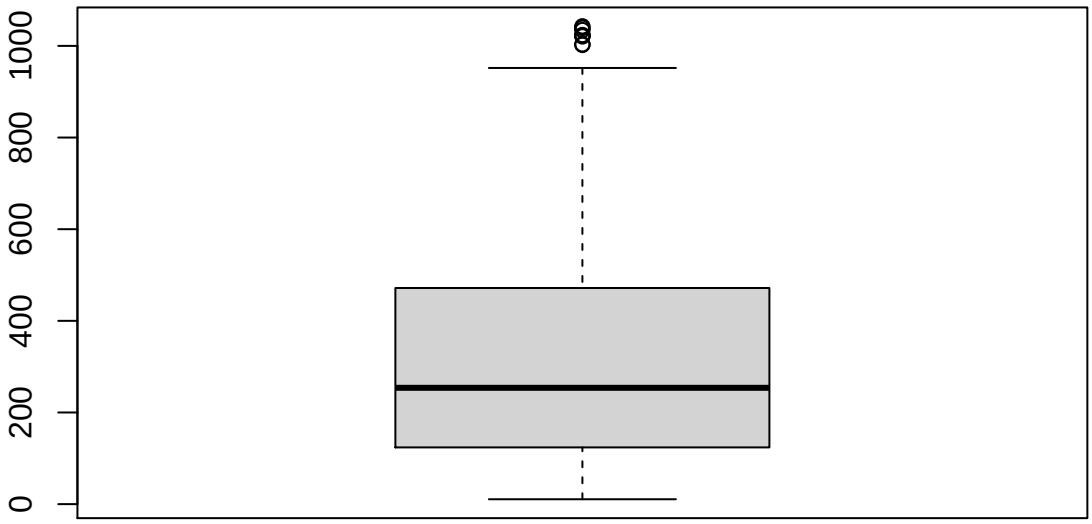
```r
#we can go ahead and check for the outliers
#first we can go ahead and choose the numerical columns only

nums <- subset(jr, select = c(Sales))

colnames(nums)
```

```
## [1] "Sales"
```

```r
boxplot(nums)
```



We observe that we have some outliers in the sales columns.

```
#we can first start by changing the pesky column names
#we will change them to lowercase
# First we Change the type of the loaded dataset to a dataframe

jr1 = as.data.frame(jr)

# Change column names, by making them uniform

colnames(jr1) = tolower(colnames(jr1))

#to confirm the change

names (jr1)
```

2.Tidying the dataset

```
## [1] "date"  "sales"
```

```
#install.packages(tibbletime)

library(tibbletime)
```

```
##
## Attaching package: 'tibbletime'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
jr1$date <- as.Date(jr1$date , format = "%m/%d/%Y")
head(jr1)
```

```
##         date    sales
## 1 2019-01-05 548.9715
## 2 2019-03-08  80.2200
## 3 2019-03-03 340.5255
## 4 2019-01-27 489.0480
## 5 2019-02-08 634.3785
## 6 2019-03-25 627.6165
```

```
unique(jr1$date)
```

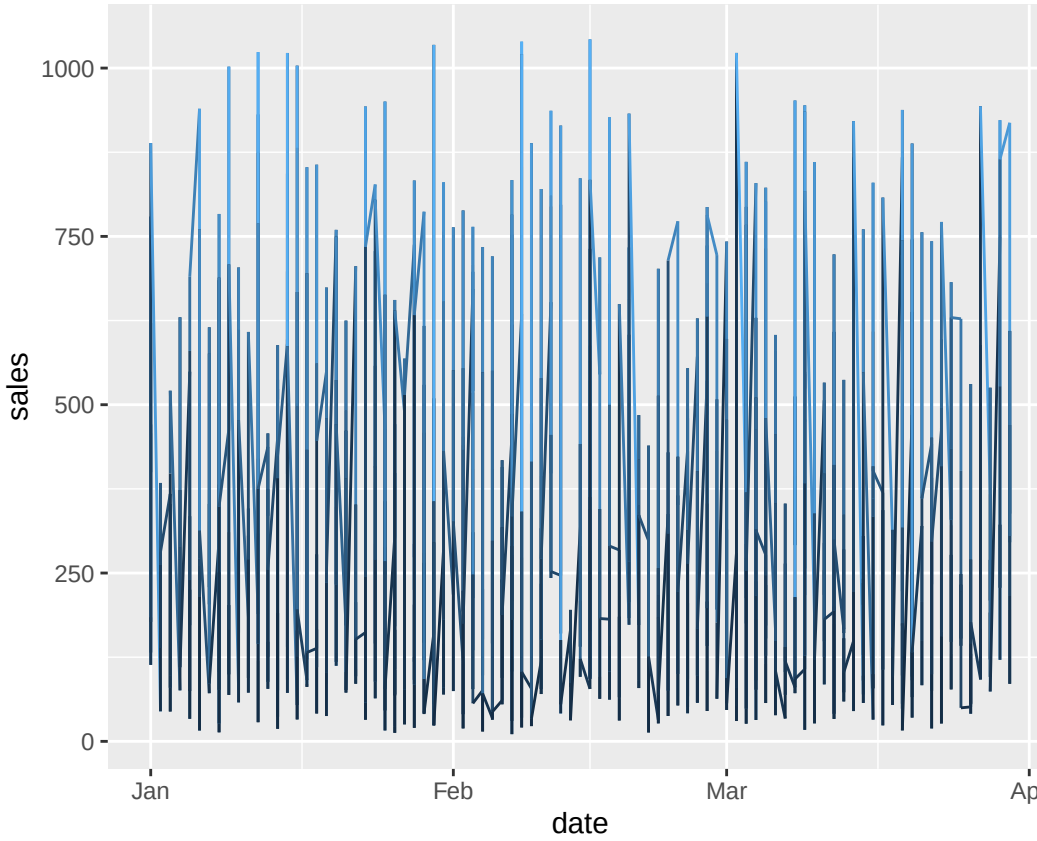
```
##  [1] "2019-01-05" "2019-03-08" "2019-03-03" "2019-01-27" "2019-02-08"
##  [6] "2019-03-25" "2019-02-25" "2019-02-24" "2019-01-10" "2019-02-20"
## [11] "2019-02-06" "2019-03-09" "2019-02-12" "2019-02-07" "2019-03-29"
## [16] "2019-01-15" "2019-03-11" "2019-01-01" "2019-01-21" "2019-03-05"
## [21] "2019-03-15" "2019-02-17" "2019-03-02" "2019-03-22" "2019-03-10"
## [26] "2019-01-25" "2019-01-28" "2019-01-07" "2019-03-23" "2019-01-17"
## [31] "2019-02-02" "2019-03-04" "2019-03-16" "2019-02-27" "2019-02-10"
## [36] "2019-03-19" "2019-02-03" "2019-03-07" "2019-02-28" "2019-03-27"
## [41] "2019-01-20" "2019-03-12" "2019-02-15" "2019-03-06" "2019-02-14"
## [46] "2019-03-13" "2019-01-24" "2019-01-06" "2019-02-11" "2019-01-22"
## [51] "2019-01-13" "2019-01-09" "2019-01-12" "2019-01-26" "2019-01-23"
## [56] "2019-02-23" "2019-01-02" "2019-02-09" "2019-03-26" "2019-03-01"
## [61] "2019-02-01" "2019-03-28" "2019-03-24" "2019-02-05" "2019-01-19"
## [66] "2019-01-16" "2019-01-08" "2019-02-18" "2019-01-18" "2019-02-16"
## [71] "2019-02-22" "2019-01-29" "2019-01-04" "2019-03-30" "2019-01-30"
## [76] "2019-01-03" "2019-03-21" "2019-02-13" "2019-01-14" "2019-03-18"
## [81] "2019-03-20" "2019-02-21" "2019-01-31" "2019-01-11" "2019-02-26"
## [86] "2019-03-17" "2019-03-14" "2019-02-04" "2019-02-19"
```

```
jr1 <- as_tbl_time(jr1 , index= date)
```

```
#Plotting data

library(ggplot2)

ggplot(jr1, aes(x=date, y=sales, color=sales)) + geom_line()
```



3.Exploratory data Analysis
We see some huge spikes at different intervals from Jan to April.

4.Anomally detection method We will now perform anomaly detection using Seasonal Hybrid ESD Test. The technique maps data as a series and captures seasonality while pointing out data which does not follow the seasonality pattern. The AnomalyDetectionTs() function finds the anomalies in the data. It will basically narrow down all the peaks keeping in mind that not more than 10% of data can be anomalies (by default). We can also reduce this number by changing the max_ano ms parameter in the data. We can also specify which kind of anomalies are to be identified using the direction parameter. Here, we are going to specify only positive direction anomalies to be identified. That means that sudden dips in the data are not considered.

```
#Install the devtools package then github packages
#install.packages("devtools")
#install.packages("Rcpp")
#install.packages("anomaly")
#install.packages("anomalize")
#devtools::install_github("twitter/AnomalyDetection")
library(AnomalyDetection)
```

```
#Loading the libraries
library(devtools)
```

```
## Loading required package: usethis
```

```
library(Rcpp)
library(anomaly)
```

```
##
## Attaching package: 'anomaly'
```

```
## The following object is masked from 'package:stats':
##
##     simulate
```

```
library(anomalize)
```

```
## == Use anomalize to improve your Forecasts by 50%! =============================
## Business Science offers a 1-hour course - Lab #18: Time Series Anomaly Detection!
## </> Learn more at: https://university.business-science.io/p/learning-labs-pro </>
```

```
anomaly.detect <- jr1 %>%
 group_by(date) %>%
 summarise(totalsales = sum(sales)) %>%
time_decompose(totalsales, method = "stl", frequency = "7 days", trend = "30 days") %>%  anomalize(rema
plot_anomaly_decomposition()
```

```
## Error in summarise(., totalsales = sum(sales)): could not find function "summarise"
```

```
anomaly.detect
```

```
## Error in eval(expr, envir, enclos): object 'anomaly.detect' not found
```

```
#Apply anomaly detection and plot the results in a more clearer way
#wanted to asceratin the exact points the months have anomalies

jr1 %>%
  group_by(date) %>%
  summarise(totalsales = sum(sales)) %>%
  time_decompose(totalsales, method = "twitter", frequency = "auto", trend = "auto") %>%
  anomalize(remainder, method = "gesd",alpha = 0.05, max_anoms = 0.2) %>%
  time_recompose() %>%
  #Anomaly Visualization
  plot_anomalies(
  time_recomposed =T,
  ncol = 6,
  color_no = "#2c3e50",
  color_yes = "#e31a1c",
  fill_ribbon = "blue",
  size_dots = 3,
  size_circles = 4)
```

```
## Error in summarise(., totalsales = sum(sales)): could not find function "summarise"
```

we observe that the anomalies are in the middle of february am assuming this is during the valentine day periods and near easter holidays in march.And the anomalies are spikes in the sales data.

5.Conclusion

- The sales data seems to contain some anomalies as shown by the red points on the graph above It would be important for the marketing team to check them out to ascertain they are not fraud.

*The spikes in the data in days nearing end month and holidays are indication this is when most people shop and promotions should be done majorly during this times

*Also the management should do auditing for these period just to confirm that theres no fraud that happened during these events.