ANALOMY DETECTION

GRIFFIN

2022-08-03

##RESEARCH QUESTION##

Carrefour Kenya and are currently undertaking a project that will inform the marketing department on the most relevant marketing strategies that will result in the highest no. of sales (total price including tax).This project is aimed at doing analysis on the dataset provided by carrefour and create insights on how to achieve highest sales.

##METRIC FOR SUCCESS##

Be able to detect and do away with anomalies in our dataset

##THE CONTEXT##

Carre Four is an International chain of retail supemarkets in the world, It was set up in Kenya in the year 2016 and has been performing well over the years.Carrefour ensures customer satisfaction and everyday convenience while offering unbeatable value for money with a vast array of more than 100,000 products, shoppers can purchase items for their every need, whether home electronics or fresh fruits from around the world, to locally produced items. This project is aimed at creating insights from existing and current trends to develop marketing strategies that will enable the marketing team achieve higher sales.

##EXPERIMENTAL DESIGN##

1. Loading libraries
2. Load data
3. Data cleaning
4. Anomaly detection
5. Conclusion
6. Recommendation

Loading the libraries

```
#pkg <- c('tidyverse','tibbletime','anomalize','timetk')
#install.packages(pkg)
#install.packages("anomalize")
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyve
rse 1.3.2 ──
```

```
## ✔ ggplot2 3.3.6     ✔ purrr   0.3.4
## ✔ tibble  3.1.7     ✔ dplyr   1.0.9
## ✔ tidyr   1.2.0     ✔ stringr 1.4.0
## ✔ readr   2.1.2     ✔ forcats 0.5.1
## ── Conflicts ────────────────────────────────────────── tidyverse_co
nflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()

library(tibbletime)

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

library(anomalize)

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

library(timetk)
```

Loading the data

reading the data

```
data<- read.csv("C://moringa//GROUP WORK//Supermarket_Sales_Forecasting
 - Sales.csv")
```

Previewing the dataset

```
#cheaking the head
head(data)

##         Date    Sales
## 1  1/5/2019 548.9715
## 2  3/8/2019  80.2200
## 3  3/3/2019 340.5255
## 4 1/27/2019 489.0480
## 5  2/8/2019 634.3785
## 6 3/25/2019 627.6165

#checking the tail
tail(data)
```

```
##            Date     Sales
## 995  2/18/2019   63.9975
## 996  1/29/2019   42.3675
## 997   3/2/2019 1022.4900
## 998   2/9/2019   33.4320
## 999  2/22/2019   69.1110
## 1000 2/18/2019  649.2990
```

```
#checking the data structure
str(data)
```

```
## '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 ...
```

our data has 1000 records and 2 variables which are character and numeric # Data cleaning **Checking missing values**

```
#checking missing values
colSums((is.na(data)))
```

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

our data has no missing values **checking duplicates**

```
#checking duplicates
duplicates<- data[duplicated(data)]
duplicates
```

```
## data frame with 0 columns and 1000 rows
```

our data has no duplicates **checking datatype**

```
#checking datatype
str(data)
```

```
## '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 ...
```

```
#date column is a character thus need to cconvert it to date type
data$Date <- as.Date(data$Date,"%m/%d/%Y")
str(data)
```

```
## 'data.frame':    1000 obs. of  2 variables:
##  $ Date : Date, format: "2019-01-05" "2019-03-08" ...
##  $ Sales: num  549 80.2 340.5 489 634.4 ...
```

```
head(data)
```

```
##          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
```

```
#as. POSIXct stores both a date and time with an associated time zone.
The default time zone selected, is the time zone that your computer is
set #to which is most often your local time zone
data$Date <- as.POSIXct(data$Date)

# Convert df to a tibble
#because time_decompose require data to be tibble
data <- as_tibble(data)
class(data)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
data
```

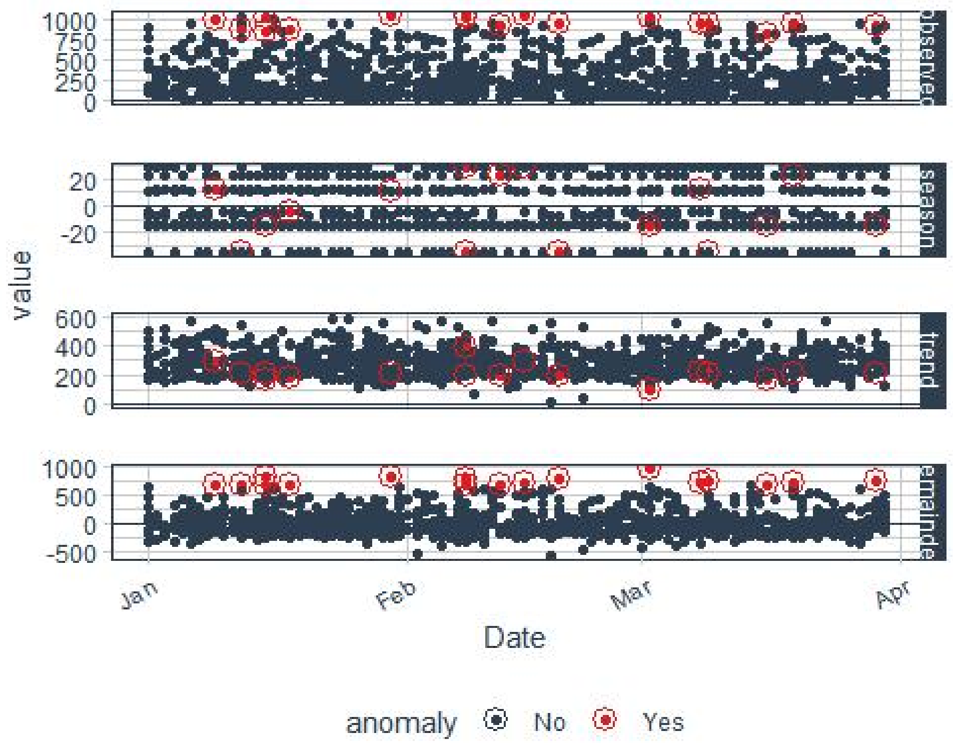
```
## # A tibble: 1,000 × 2
##    Date                Sales
##    <dttm>              <dbl>
##  1 2019-01-05 03:00:00 549.
##  2 2019-03-08 03:00:00  80.2
##  3 2019-03-03 03:00:00 341.
##  4 2019-01-27 03:00:00 489.
##  5 2019-02-08 03:00:00 634.
##  6 2019-03-25 03:00:00 628.
##  7 2019-02-25 03:00:00 434.
##  8 2019-02-24 03:00:00 772.
##  9 2019-01-10 03:00:00  76.1
## 10 2019-02-20 03:00:00 173.
## # … with 990 more rows
## # ℹ Use `print(n = ...)` to see more rows
```

implementing the solution

The R ‘anomalize’ package enables a workflow for detecting anomalies in data. The main functions are time_decompose(), anomalize(), and time_recompose(). We will use time_decompose() function in anomalize package. We will use stl method which uses the loess smoothing to extracts seasonality

```
data %>%
time_decompose(Sales, method = 'stl', frequency = 'auto', trend = 'auto
') %>%
anomalize(remainder, method = 'gesd', alpha = 0.1, max_anoms = 0.5) %>%
plot_anomaly_decomposition()
```

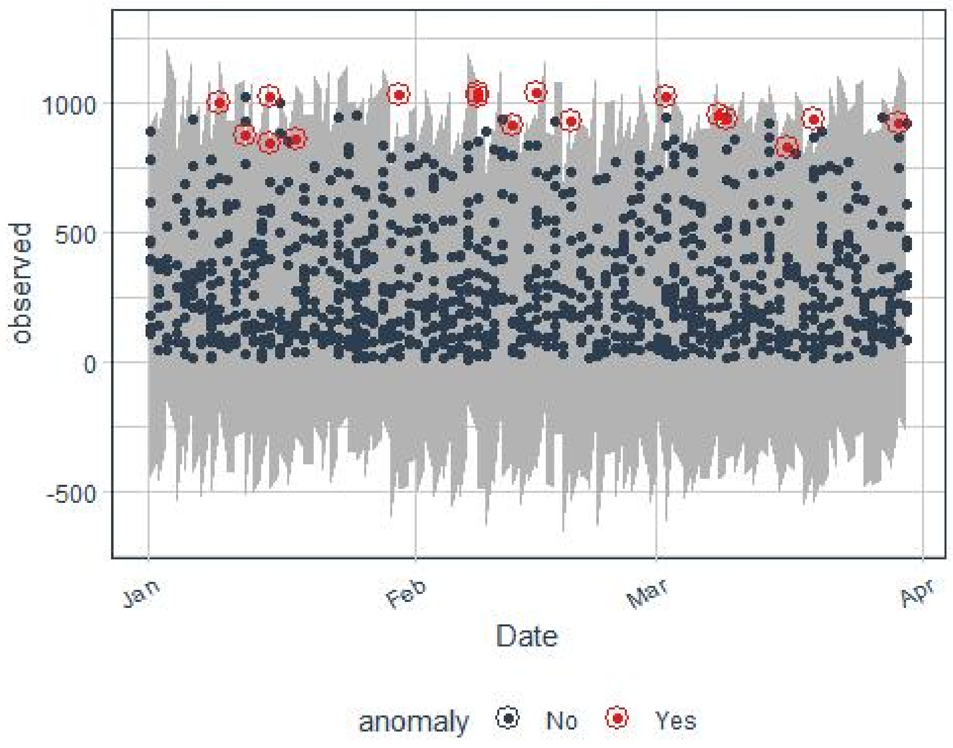
```
## Converting from tbl_df to tbl_time.
## Auto-index message: index = Date

## Note: Index not ordered. tibbletime assumes index is in ascending or
der. Results may not be as desired.

## frequency = 12 seconds

## Note: Index not ordered. tibbletime assumes index is in ascending or
der. Results may not be as desired.

## trend = 12 seconds

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



time_decompose() functions which produces four columns -The overall observed data. -The seasonal or cyclic trend. -The long-term trend. The default is a span of 3 months. -It is used for analyzing the outliers. The red points indicate anomalies according to the anomalize function

Recomposing create lower and upper bounds around the observed values with time_recompose. It recomposes the season, trend, remainder_l1 and remainder_l2 into new limits that are -recomposed_l1 : The lower bound of outliers around the observed values. -recomposed_l2 : The upper bound of outliers around the observed values

```
data %>%
time_decompose(Sales, method = 'stl', frequency = 'auto', trend = 'auto
') %>%
anomalize(remainder, method = 'gesd', alpha = 0.1, max_anoms = 0.1) %>%
time_recompose() %>%
plot_anomalies(time_recomposed = TRUE)

## Converting from tbl_df to tbl_time.
## Auto-index message: index = Date

## Note: Index not ordered. tibbletime assumes index is in ascending or
der. Results may not be as desired.

## frequency = 12 seconds

## Note: Index not ordered. tibbletime assumes index is in ascending or
der. Results may not be as desired.

## trend = 12 seconds
```



- The plot and shows the anomalies

- check actual anomalies values

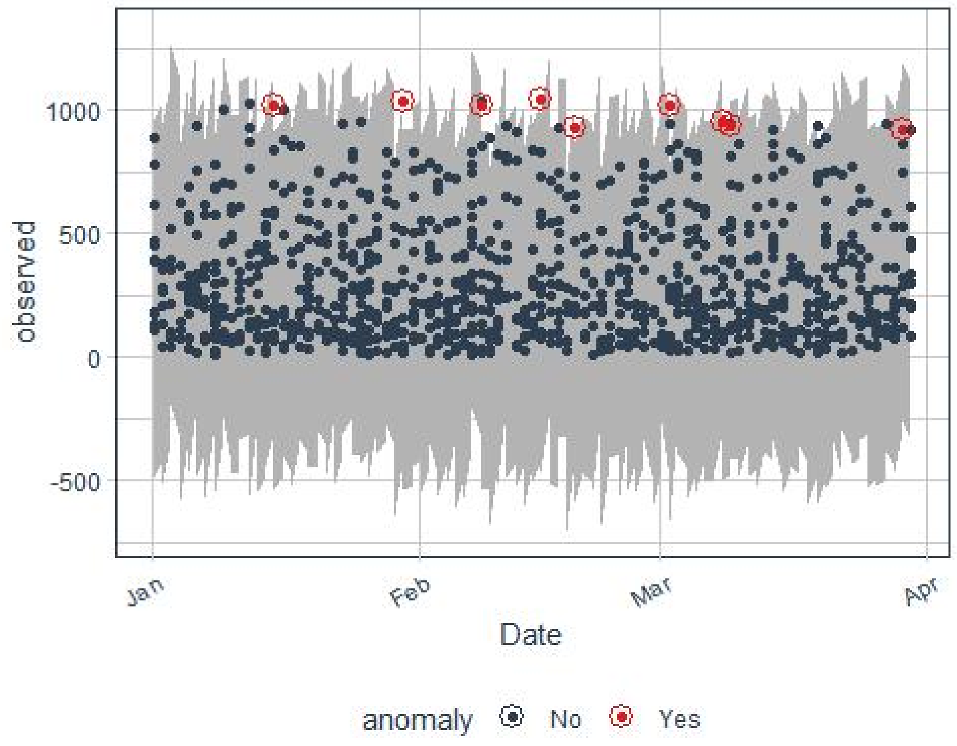
```
anomalies = data %>%
time_decompose(Sales, method = 'stl', frequency = 'auto', trend = 'auto
') %>%
anomalize(remainder, method = 'gesd', alpha = 0.05, max_anoms = 0.1)
 %>%
time_recompose() %>%
filter(anomaly == 'Yes')

## Converting from tbl_df to tbl_time.
## Auto-index message: index = Date

## Note: Index not ordered. tibbletime assumes index is in ascending or
der. Results may not be as desired.

## frequency = 12 seconds

## Note: Index not ordered. tibbletime assumes index is in ascending or
der. Results may not be as desired.

## trend = 12 seconds

anomalies

## # A time tibble: 9 × 10
## # Index: Date
##   Date                observ… season trend remain… remain… remain… a
nomaly
##   <dttm>                <dbl>  <dbl> <dbl>   <dbl>   <dbl>   <dbl> <
chr>
## 1 2019-03-29 03:00:00    923.  -14.4 203.     734.   -738.    714. Y
es
## 2 2019-02-15 03:00:00   1043.   28.7 286.     728.   -738.    714. Y
es
## 3 2019-02-08 03:00:00   1021.   28.7 201.     791.   -738.    714. Y
es
## 4 2019-03-08 03:00:00    952.   12.7 217.     722.   -738.    714. Y
es
## 5 2019-01-30 03:00:00   1034.   10.4 209.     815.   -738.    714. Y
es
## 6 2019-03-09 03:00:00    935.  -34.6 211.     759.   -738.    714. Y
es
## 7 2019-01-15 03:00:00   1022.  -14.4 209.     828.   -738.    714. Y
es
## 8 2019-02-19 03:00:00    932.  -34.6 189.     778.   -738.    714. Y
es
## 9 2019-03-02 03:00:00   1022.  -14.4  91.6    945.   -738.    714. Y
es
## # … with 2 more variables: recomposed_l1 <dbl>, recomposed_l2 <dbl>
```

Adjusting Alpha and Max Anoms Aplha The alpha and max_anoms are the two parameters that control the anomalize() function. we used alpha = 0.1 amd we then try to reduce it to alpha=0.05 and see what happens

```
data %>%
time_decompose(Sales, method = 'stl', frequency = 'auto', trend = 'auto
') %>%
anomalize(remainder, method = 'gesd', alpha = 0.05, max_anoms = 0.1)
```

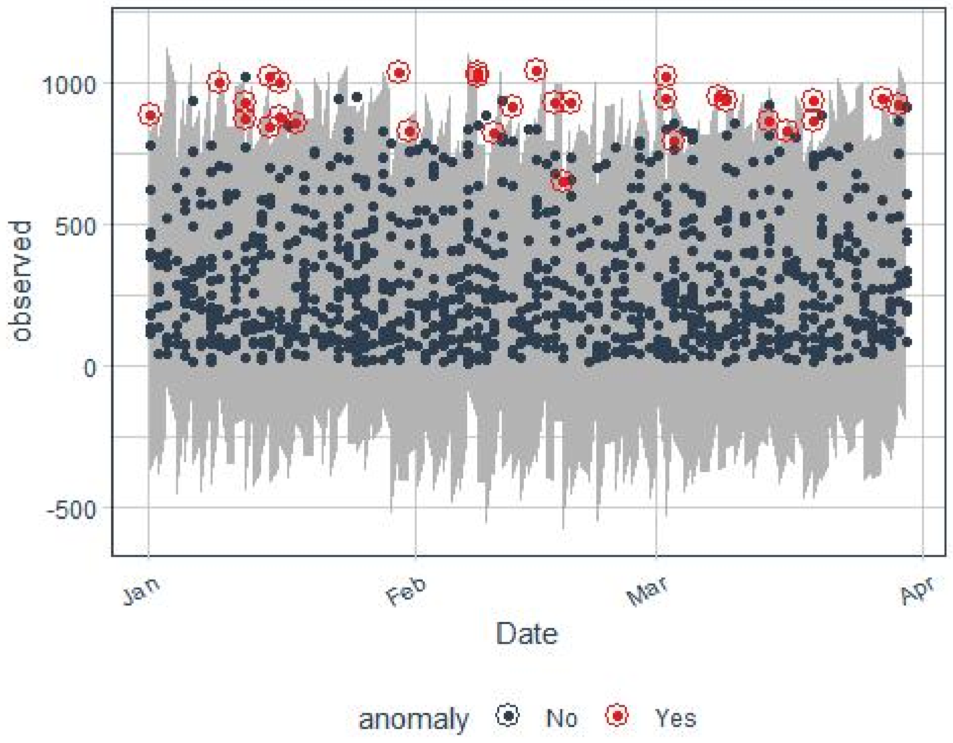
```
 %>%
time_recompose() %>%
plot_anomalies(time_recomposed = TRUE)
```

```
## Converting from tbl_df to tbl_time.
## Auto-index message: index = Date

## Note: Index not ordered. tibbletime assumes index is in ascending or
der. Results may not be as desired.

## frequency = 12 seconds

## Note: Index not ordered. tibbletime assumes index is in ascending or
der. Results may not be as desired.

## trend = 12 seconds
```

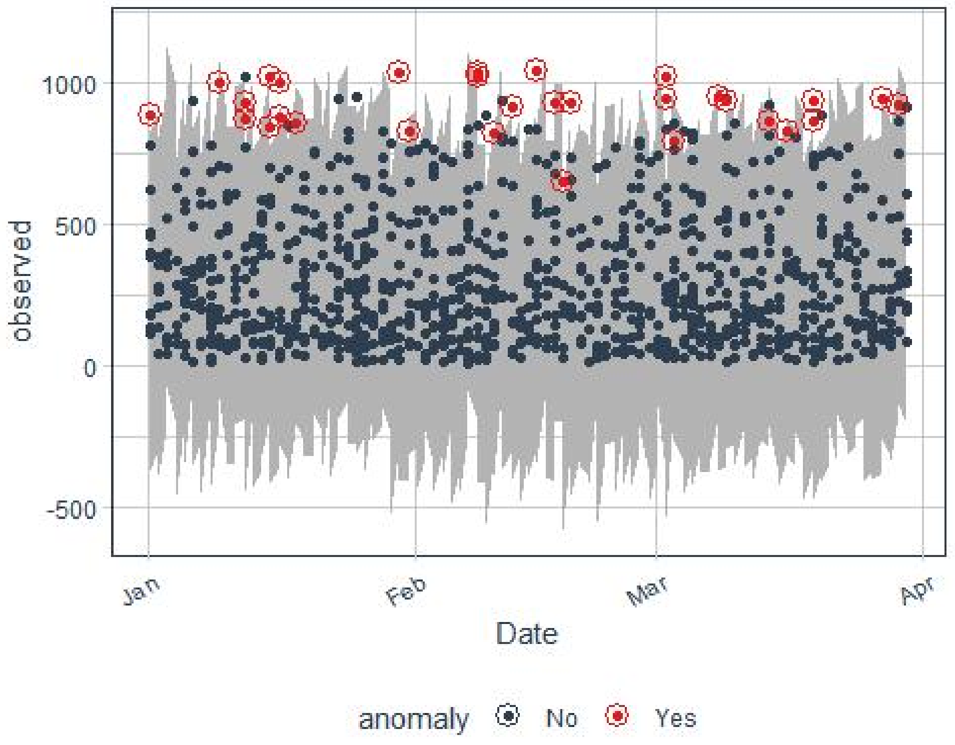


If we decrease alpha, it increases the bands making it more difficult to be an outlier. Max Anoms The max_anoms parameter is used to control the maximum percentage of data that can be an anomaly. we used max_anoms as 0.1 and we now try to increase it to 0.2 and see what happens and we use alpha as 0.5 where nearly eveything is anomally

```
data %>%
time_decompose(Sales, method = 'stl', frequency = 'auto', trend = 'auto
') %>%
anomalize(remainder, method = 'gesd', alpha = 0.5, max_anoms = 0.2) %>%
time_recompose() %>%
plot_anomalies(time_recomposed = TRUE)

## Converting from tbl_df to tbl_time.
## Auto-index message: index = Date

## Note: Index not ordered. tibbletime assumes index is in ascending or
der. Results may not be as desired.

## frequency = 12 seconds

## Note: Index not ordered. tibbletime assumes index is in ascending or
der. Results may not be as desired.

## trend = 12 seconds
```



```
data %>%
time_decompose(Sales, method = 'stl', frequency = 'auto', trend = 'auto
') %>%
anomalize(remainder, method = 'gesd', alpha = 0.5, max_anoms = 0.05)
 %>%
time_recompose() %>%
plot_anomalies(time_recomposed = TRUE)

## Converting from tbl_df to tbl_time.
## Auto-index message: index = Date
## Note: Index not ordered. tibbletime assumes index is in ascending or
der. Results may not be as desired.

## frequency = 12 seconds

## Note: Index not ordered. tibbletime assumes index is in ascending or
der. Results may not be as desired.

## trend = 12 seconds
```



using a lower value for the max_anoms means less number of anomalies will be detected # 5. CONCLUSION## ftom our investigation we find that the data has some anomalies # 6. RECOMMENDATION## The carrefour should investigate to find out the cause of anomalies