Anomaly Detection

2022-04-03

Research Question

You are a Data analyst at 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). Your project has been divided into four parts where you'll explore a recent marketing dataset by performing various unsupervised learning techniques and later providing recommendations based on your insights.

Part 4: Anomaly Detection

You have also been requested to check whether there are any anomalies in the given sales dataset. The objective of this task being fraud detection.

Defining the question

i)Specifying the Data Analytic Question

Check whether there are any anomalies in the given sales dataset. The objective of this task being fraud detection.

ii)Defining the Metric for Success

To detect and plot any anomalies in our dataset.

iii) Understanding the Context

The package anomalize does Time Series Anomaly Detection that goes inline with other Tidyverse packages (or packages supporting tidy data) – with one of the most used Tidyverse functionality – compatibility with the pipe %>% operator to write readable and reproducible data pipeline.

Before starting with Time Series forecasting or Modelling, there is Time Series Decomposition where the Time series data is decomposed into Observed, Seasonal, Trend and Remainder components using the tidy_decompose() function. Anomalize() function performs anomaly detection on the decomposed data using the remainder column and creates 3 new columns; remainder_l1 (lower limit), remainder_l2 (upper limit) and anomaly (Yes/No Flag). The default method is method = "iqr", which is fast and relatively accurate at detecting anomalies. We then create the lower and upper bounds around the observed values through the use of the time_recompose() function which recomposes the lower and upper bounds of the anomalies around the observed values and creates new columns: recomposed_l1 (lower limit) and recomposed_l2 (upper limit).

```r
#Load the necessary libraries
library(anomalize) # tidy anomaly detectiom
```

```
## == 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 </>
```

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(tibble)
library(tibbletime) # time series data
```

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

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

```r
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
library(dplyr)
library(timetk)
```

```
##
## Attaching package: 'timetk'
```

```
## The following object is masked from 'package:data.table':
##
##     :=
```

```
library(ggplot2)
```

```
#We load the dataset into our environment.
df4<- read.csv("http://bit.ly/CarreFourSalesDataset")
#Lets preview the head
head(df4)
```

```
##        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
```

```
#Lets preview the tail
tail(df4)
```

```
##           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
```

```
#Check the dimensions of the dataset.
dim(df4)
```

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

1000 observations of 2 variables

```
#Check for missing values in our dataset.
colSums(is.na(df4))
```

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

No null values in our dataset

```
#data structure
str(df4)
```

```
## '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 date column is character type. We will change to datetime type

```
df4$Date <- as.Date(df4$Date, format = "%m/%d/%Y")
df4$Date <- sort(df4$Date, decreasing = FALSE)
```

```
#data structure
str(df4)
```

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

```
df4$Date <- as.POSIXct(df4$Date)
```

```
#Convert the sales into a tibbletime object.
df4 <- as_tibble(df4)
```

Anomaly Detection.

The R 'anomalize' package enables a workflow for detecting anomalies in data. The main functions are:

- time_decompose()- for time series decomposition
- anomalize()- for Anomaly detection of remainder
- time_recompose()- for Anomaly lower and upper bound transformation

Decomposing

```
df4 %>%
    time_decompose(Sales, method = "stl", frequency = "auto", trend = "auto", message = TRUE)
```

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

```
## frequency = 11 seconds
```

```
## trend = 11 seconds
```

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

```
## # A time tibble: 1,000 x 5
## # Index: Date
##    Date                observed season trend remainder
##    <dttm>                 <dbl>  <dbl> <dbl>     <dbl>
##  1 2019-01-01 03:00:00    549.  -25.2   383.      192.
##  2 2019-01-01 03:00:00     80.2  60.7   410.     -391.
```

```
##  3 2019-01-01 03:00:00    341.   22.1   438.    -120.
##  4 2019-01-01 03:00:00    489.   14.8   456.      18.1
##  5 2019-01-01 03:00:00    634.   12.5   474.     148.
##  6 2019-01-01 03:00:00    628.  -37.1   464.     201.
##  7 2019-01-01 03:00:00    434.  -38.0   454.      18.0
##  8 2019-01-01 03:00:00    772.  -18.9   379.     412.
##  9 2019-01-01 03:00:00     76.1  12.4   305.    -241.
## 10 2019-01-01 03:00:00    173.   -9.34  252.     -70.4
## # ... with 990 more rows
```

#The "Sales" column is decomposed into "observed", "season", "trend", and "remainder" columns. The default values for time series decompose are method = "stl", which is just seasonal decomposition using a Loess smoother (refer to stats::stl()). The frequency and trend parameters are automatically set based on the time scale (or periodicity) of the time series using tibbletime based function under the hood.

```
#Anomaly Detection Of Remainder

df4 %>%
    time_decompose(Sales, method = "stl", frequency = "auto", trend = "auto") %>%
    anomalize(remainder, method = "iqr", alpha = 0.05, max_anoms = 0.2)
```

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

## frequency = 11 seconds

## trend = 11 seconds

## # A time tibble: 1,000 x 8
## # Index: Date
##    Date                observed season trend remainder remainder_l1 remainder_l2
##    <dttm>                 <dbl>  <dbl> <dbl>     <dbl>        <dbl>        <dbl>
##  1 2019-01-01 03:00:00    549.  -25.2   383.     192.         -866.         895.
##  2 2019-01-01 03:00:00     80.2  60.7   410.    -391.         -866.         895.
##  3 2019-01-01 03:00:00    341.   22.1   438.    -120.         -866.         895.
##  4 2019-01-01 03:00:00    489.   14.8   456.      18.1        -866.         895.
##  5 2019-01-01 03:00:00    634.   12.5   474.     148.         -866.         895.
##  6 2019-01-01 03:00:00    628.  -37.1   464.     201.         -866.         895.
##  7 2019-01-01 03:00:00    434.  -38.0   454.      18.0        -866.         895.
##  8 2019-01-01 03:00:00    772.  -18.9   379.     412.         -866.         895.
##  9 2019-01-01 03:00:00     76.1  12.4   305.    -241.         -866.         895.
## 10 2019-01-01 03:00:00    173.   -9.34  252.     -70.4        -866.         895.
## # ... with 990 more rows, and 1 more variable: anomaly <chr>
```

#The next step is to perform anomaly detection on the decomposed data, specifically the "remainder" column.We did this using anomalize(), which produces three new columns: "remainder_l1" (lower limit), "remainder_l2" (upper limit), and "anomaly" (Yes/No Flag).

```
#Anomaly Lower and Upper Bounds
df4 %>%
    time_decompose(Sales, method = "stl", frequency = "auto", trend = "auto") %>%
    anomalize(remainder, method = "iqr", alpha = 0.05, max_anoms = 0.2) %>%
    time_recompose()
```

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

## frequency = 11 seconds

## trend = 11 seconds

## # A time tibble: 1,000 x 10
## # Index: Date
##    Date                observed season trend remainder remainder_l1 remainder_l2
##    <dttm>                 <dbl>  <dbl> <dbl>     <dbl>        <dbl>        <dbl>
##  1 2019-01-01 03:00:00    549.  -25.2   383.     192.         -866.         895.
##  2 2019-01-01 03:00:00     80.2  60.7   410.    -391.         -866.         895.
##  3 2019-01-01 03:00:00    341.   22.1   438.    -120.         -866.         895.
##  4 2019-01-01 03:00:00    489.   14.8   456.      18.1        -866.         895.
##  5 2019-01-01 03:00:00    634.   12.5   474.     148.         -866.         895.
##  6 2019-01-01 03:00:00    628.  -37.1   464.     201.         -866.         895.
##  7 2019-01-01 03:00:00    434.  -38.0   454.      18.0        -866.         895.
##  8 2019-01-01 03:00:00    772.  -18.9   379.     412.         -866.         895.
##  9 2019-01-01 03:00:00     76.1  12.4   305.    -241.         -866.         895.
## 10 2019-01-01 03:00:00    173.   -9.34  252.     -70.4        -866.         895.
## # ... with 990 more rows, and 3 more variables: anomaly <chr>,
## #   recomposed_l1 <dbl>, recomposed_l2 <dbl>
```

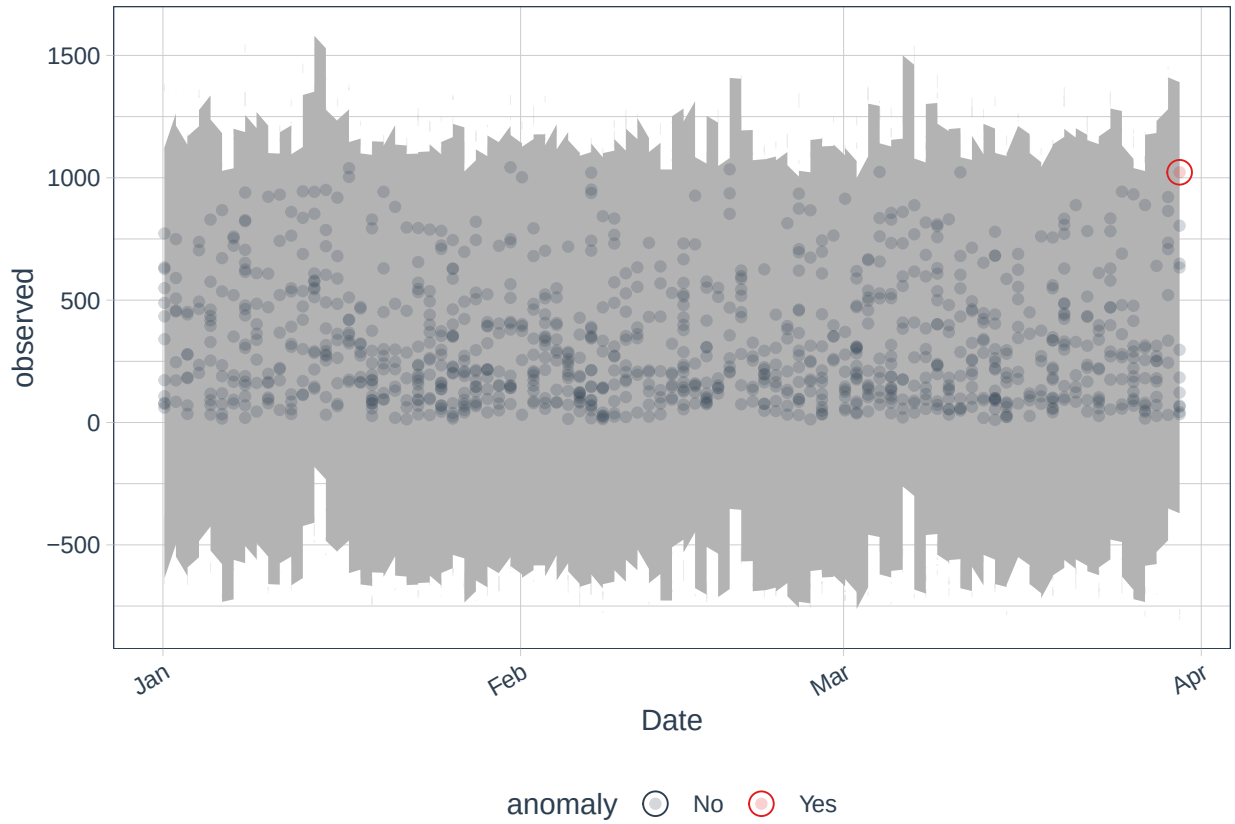
#The last step is to create the lower and upper bounds around the "observed" values. This is the work of time_recompose(), which recomposes the lower and upper bounds of the anomalies around the observed values. Two new columns were created: "recomposed_l1" (lower limit) and "recomposed_l2" (upper limit).

```
#we now plot using plot_anomaly_decomposition() to visualize out data.
df4 %>%
    time_decompose(Sales) %>%
    anomalize(remainder) %>%
    time_recompose() %>%
    plot_anomalies(time_recomposed = TRUE, ncol = 3, alpha_dots = 0.2)
```

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

## frequency = 11 seconds

## trend = 11 seconds
```




```
anomalies = df4 %>%
time_decompose(Sales, method = 'stl', frequency = 'auto', trend = 'auto') %>%
anomalize(remainder, method = 'iqr', alpha = 0.05, max_anoms = 0.2) %>%
time_recompose() %>%
filter(anomaly == 'Yes')
```

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

## frequency = 11 seconds

## trend = 11 seconds
```

```
anomalies
```

```
## # A time tibble: 1 x 10
## # Index: Date
##   Date                observed season trend remainder remainder_l1 remainder_l2
##   <dttm>                 <dbl>  <dbl> <dbl>     <dbl>        <dbl>        <dbl>
## 1 2019-03-30 03:00:00    1022.  -38.0  98.8      962.        -866.         895.
## # ... with 3 more variables: anomaly <chr>, recomposed_l1 <dbl>,
## #   recomposed_l2 <dbl>
```

There are annomalies in the carrefour dataset

```
head(anomalies)
```

```
## # A time tibble: 1 x 10
## # Index: Date
##   Date                observed season trend remainder remainder_l1 remainder_l2
##   <dttm>                 <dbl>  <dbl> <dbl>     <dbl>        <dbl>        <dbl>
## 1 2019-03-30 03:00:00    1022.  -38.0  98.8      962.        -866.         895.
## # ... with 3 more variables: anomaly <chr>, recomposed_l1 <dbl>,
## #   recomposed_l2 <dbl>
```

CONCLUSION

There were more than 10 anomalies in the month of Feruary and March