# Anomaly Detection

Jenipher Mawia

11/13/2020

## 1. Problem Definition

- Detect anomalies in the sales data.
- Implement the solution using unsupervised learning techniques for anomaly detection

## 2. Defining the metrics for success

This project will be considered a success if anomalies that indicate fraud are detected.

## 3. The Context

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). The main task here is to detect fraud in any of the sales made.

## 4. Experimental Design taken

The following is the order in which I went about to achieve the objectives of this project:

- Data Sourcing and Understanding
- Checking the data (head, shape(number of records), datatypes)
- Data Type Conversion
- Anomaly detection
- Plotting anomalies

## 5. Data Sourcing and Understanding.

The data was provided by Moringa School and can be downloaded here

```
# load libraries
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse
1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ---------------------------------------
tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x 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 Anomaly
Detection!
## </> Learn more at:
https://university.business-science.io/p/learning-labs-pro </>
```

### Reading the data

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

## 6. Checking the data

checking the top of the data

```
head(sales_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 data types

```
str(sales_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 ...
```

Checking the dimensions of the data

```
dim(sales_data)
```

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

There are two columns and 1000 entries in the data of types character and numeric.

The date column shoudl be converted to its appropriate format.

## 7. Data Type Conversion

### Change the data type of the "Date" column

```
# convert the datatype of the date column
sales_data$Date <- anytime::anydate(sales_data$Date)
str(sales_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 ...
```

Extract the year, month and day from the date column

```
sales_data$year <- as.numeric(format(sales_data$Date, format="%Y"))
sales_data$month <- as.numeric(format(sales_data$Date, format="%m"))
sales_data$day <- as.numeric(format(sales_data$Date, format="%d"))
#check the datatypes
head(sales_data)
```

```
##         Date    Sales year month day
## 1 2019-01-05 548.9715 2019     1   5
## 2 2019-03-08  80.2200 2019     3   8
## 3 2019-03-03 340.5255 2019     3   3
## 4 2019-01-27 489.0480 2019     1  27
## 5 2019-02-08 634.3785 2019     2   8
## 6 2019-03-25 627.6165 2019     3  25
```

```
# change the class of the data to tibble to be able to use the anomalize
function
anomaly <- as_tibble(sales_data)
head(anomaly)
```

```
## # A tibble: 6 x 5
##   Date       Sales  year month   day
##   <date>     <dbl> <dbl> <dbl> <dbl>
## 1 2019-01-05 549.   2019     1     5
## 2 2019-03-08  80.2  2019     3     8
```

```
## 3 2019-03-03 341.   2019     3     3
## 4 2019-01-27 489.   2019     1    27
## 5 2019-02-08 634.   2019     2     8
## 6 2019-03-25 628.   2019     3    25
```

## 8. Checking for anomalies

```
# check for anomalies in the Sales column using the anomalize() function
sales_anomalies <- anomalize(anomaly, Sales, method = "iqr", alpha = 0.1,
max_anoms = 0.2, verbose = FALSE)

head(sales_anomalies)
```

```
## # A tibble: 6 x 8
##   Date       Sales  year month   day Sales_l1 Sales_l2 anomaly
##   <date>     <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl> <chr>
## 1 2019-01-05 549.   2019     1     5    -396.     992. No
## 2 2019-03-08  80.2  2019     3     8    -396.     992. No
## 3 2019-03-03 341.   2019     3     3    -396.     992. No
## 4 2019-01-27 489.   2019     1    27    -396.     992. No
## 5 2019-02-08 634.   2019     2     8    -396.     992. No
## 6 2019-03-25 628.   2019     3    25    -396.     992. No
```

```
# check for unique values in the column "anomaly"
unique(sales_anomalies$anomaly)
```

```
## [1] "No"  "Yes"
```

Some transactions have been detected as anomalies in the data. These could be signs of fraud.

### Converting the anomaly column to factor

```
# convert the anomaly column to factor
sales_anomalies_data<- sales_anomalies
sales_anomalies_data$anomaly <- as.factor(sales_anomalies_data$anomaly)
str(sales_anomalies_data)
```

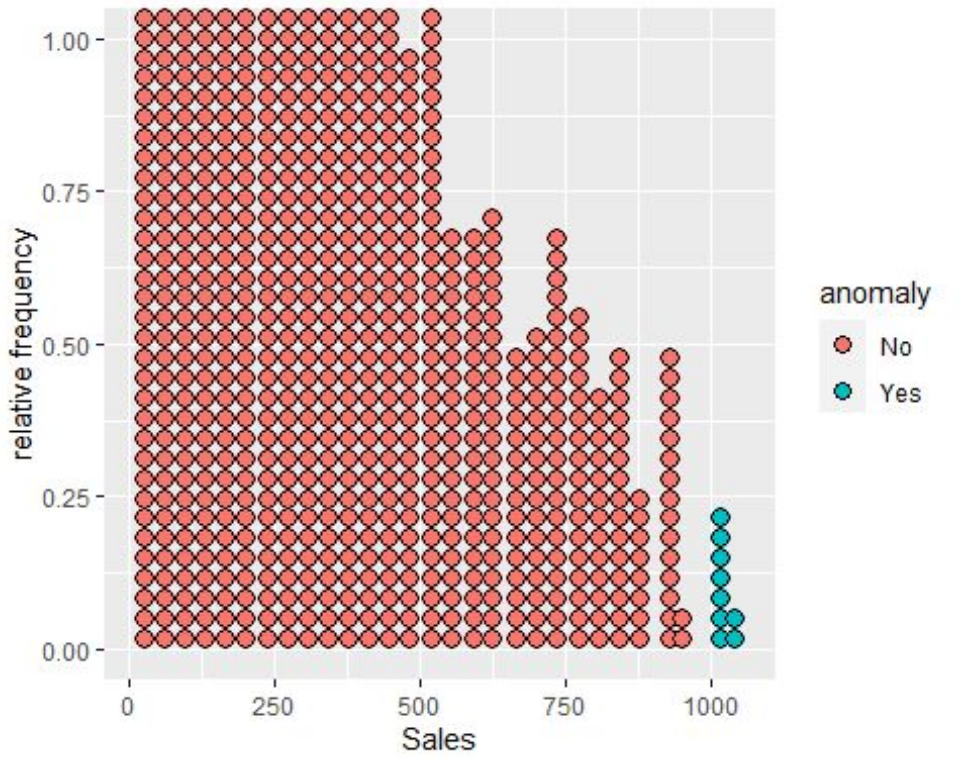
```
## tibble [1,000 x 8] (S3: tbl_df/tbl/data.frame)
##  $ Date    : Date[1:1000], format: "2019-01-05" "2019-03-08" ...
##  $ Sales   : num [1:1000] 549 80.2 340.5 489 634.4 ...
##  $ year    : num [1:1000] 2019 2019 2019 2019 2019 ...
##  $ month   : num [1:1000] 1 3 3 1 2 3 2 2 1 2 ...
##  $ day     : num [1:1000] 5 8 3 27 8 25 25 24 10 20 ...
##  $ Sales_l1: num [1:1000] -396 -396 -396 -396 -396 ...
##  $ Sales_l2: num [1:1000] 992 992 992 992 992 ...
##  $ anomaly : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "names")= chr [1:1000] "25%" "25%" "25%" "25%" ...
```

## 9. Plotting the anomalies

```
#plot(sales_anomalies_data$Sales, color = "anomaly")

sales_anomalies_data %>%
  ggplot() +
  aes(x = Sales, anomaly = ..count../nrow(sales_anomalies_data), fill =
anomaly) +
  geom_dotplot() +
  ylab("relative frequency")

## `stat_bindot()` using `bins = 30`. Pick better value with `binwidth`.
```




From the plot we can see that anomalies were detected at sales greater than 1000