Part 4: Anomaly Detection

Mildred Kulei

9/10/2021

1. loading the dataset

```
sales_f <- read.csv("http://bit.ly/CarreFourSalesDataset")
head(sales_f,n=3)
```

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

```
#checking the bottom of dataset
tail(sales_f,n=3)
```

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

```
#structure of our dataset
str(sales_f)
```

```
## '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 is of character and sales a numerical datatypes.

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

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

we only have two columns in our dataset.

```r
#checking the no of rows and columns
dim(sales_f)
```

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

we have 1000 records of data and two columns.

2. Data cleaning.

```r
#checking the missing values
colSums(is.na(sales_f))
```

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

no missing values.

```r
#checking for duplicates
duplicated_rows <- sales_f[duplicated(sales_f),]
head(duplicated_rows)
```

```
## [1] Date  Sales
## <0 rows> (or 0-length row.names)
```

no duplicates.

```r
uniqueitems <- unique(sales_f)
dim(uniqueitems)
```

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

the dates and sales are all unique.

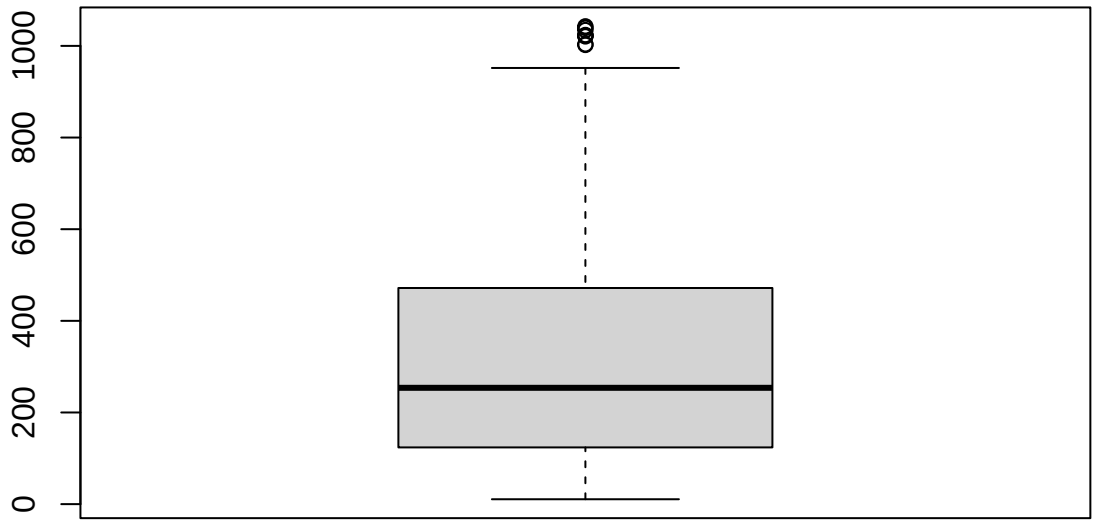
```r
#checking for the outliers
#choose the numerical columns only

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

colnames(nums)
```

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

```r
boxplot(nums)
```



we have a few outliers in sales column, but we will choose to work with it

```
#we will change the column names to lowercase
# First we Change the type of the loaded dataset to a dataframe
sales_f1 = as.data.frame(sales_f)

# Change column names, by making them uniform
colnames(sales_f1) = tolower(colnames(sales_f1))

#to confirm the change
names (sales_f1)
```

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

```
#install.packages(tibbletime)
#'tibbletime' is an extension that allows for the creation of time aware tibbles.
#immediate advantages of this include: the ability to perform time-based subsetting on tibbles

library(tibbletime)
```

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

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

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

```
##         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(sales_f$Date)
```

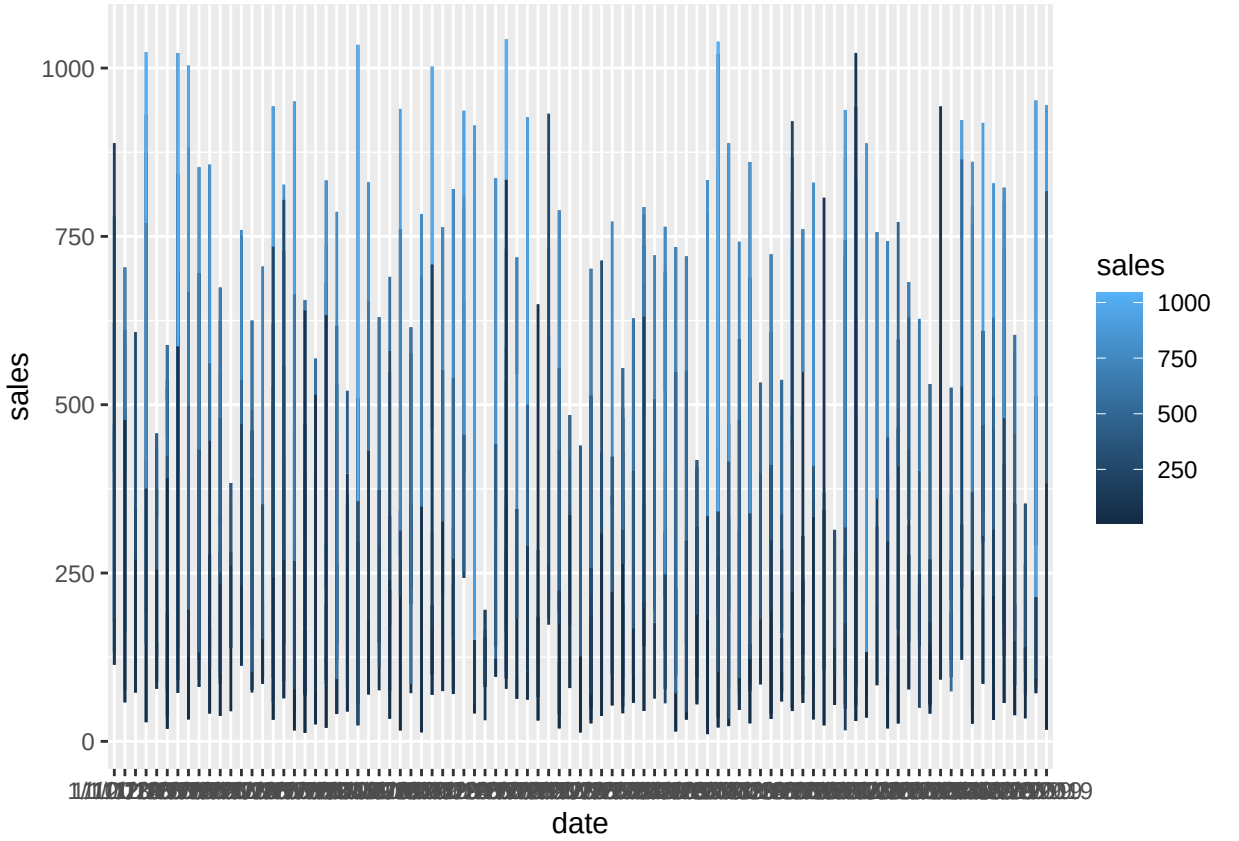
```
##  [1] "2019-01-05" "2019-03-08" "2019-03-03" "2019-01-27" "2019-02-08"
##  [6] "2019-03-25" "2019-02-25" "2019-02-24" "2019-01-10" "2019-02-20"
## [11] "2019-02-06" "2019-03-09" "2019-02-12" "2019-02-07" "2019-03-29"
## [16] "2019-01-15" "2019-03-11" "2019-01-01" "2019-01-21" "2019-03-05"
## [21] "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"
```

```
sales_f <- as_tbl_time(sales_f , index= Date)
```

We observe that the data collected is of month one to three of 2019.

3.Exploratory data Analysis.

```
#Plotting data
library(ggplot2)
ggplot(sales_f1, aes(x=date, y=sales, color=sales)) + geom_line()
```



We see some huge spikes at different intervals from Jan through to March.

```
unique(sales_f1$date)
```

```
##  [1] "1/5/2019"  "3/8/2019"  "3/3/2019"  "1/27/2019" "2/8/2019"  "3/25/2019"
##  [7] "2/25/2019" "2/24/2019" "1/10/2019" "2/20/2019" "2/6/2019"  "3/9/2019"
## [13] "2/12/2019" "2/7/2019"  "3/29/2019" "1/15/2019" "3/11/2019" "1/1/2019"
## [19] "1/21/2019" "3/5/2019"  "3/15/2019" "2/17/2019" "3/2/2019"  "3/22/2019"
## [25] "3/10/2019" "1/25/2019" "1/28/2019" "1/7/2019"  "3/23/2019" "1/17/2019"
## [31] "2/2/2019"  "3/4/2019"  "3/16/2019" "2/27/2019" "2/10/2019" "3/19/2019"
## [37] "2/3/2019"  "3/7/2019"  "2/28/2019" "3/27/2019" "1/20/2019" "3/12/2019"
## [43] "2/15/2019" "3/6/2019"  "2/14/2019" "3/13/2019" "1/24/2019" "1/6/2019"
## [49] "2/11/2019" "1/22/2019" "1/13/2019" "1/9/2019"  "1/12/2019" "1/26/2019"
## [55] "1/23/2019" "2/23/2019" "1/2/2019"  "2/9/2019"  "3/26/2019" "3/1/2019"
## [61] "2/1/2019"  "3/28/2019" "3/24/2019" "2/5/2019"  "1/19/2019" "1/16/2019"
## [67] "1/8/2019"  "2/18/2019" "1/18/2019" "2/16/2019" "2/22/2019" "1/29/2019"
## [73] "1/4/2019"  "3/30/2019" "1/30/2019" "1/3/2019"  "3/21/2019" "2/13/2019"
## [79] "1/14/2019" "3/18/2019" "3/20/2019" "2/21/2019" "1/31/2019" "1/11/2019"
## [85] "2/26/2019" "3/17/2019" "3/14/2019" "2/4/2019"  "2/19/2019"
```

The dataset is of one year only, 2019.

```
library(ggplot2)
library(psych)
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
#Descriptive analysis into measures of central Tendency

describe(sales_f1$sales)
```

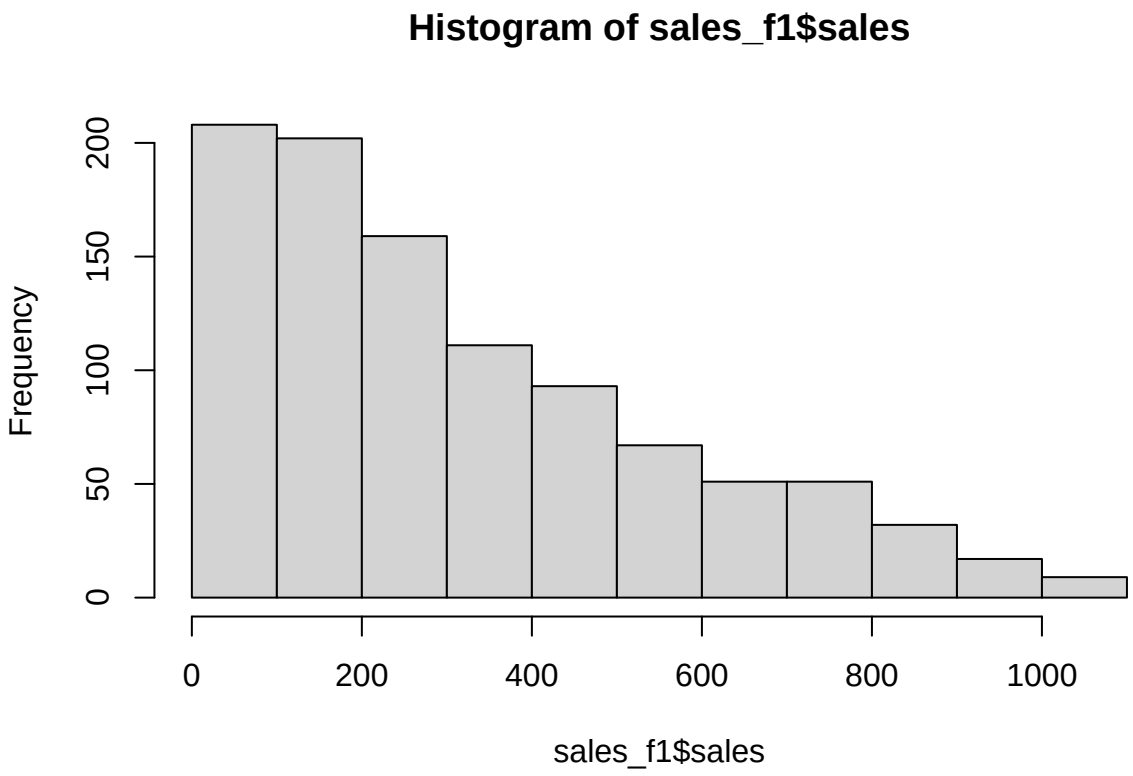
```
##    vars    n   mean     sd median trimmed    mad   min     max   range skew
## X1    1 1000 322.97 245.89 253.85  293.91 233.78 10.68 1042.65 1031.97 0.89
##    kurtosis   se
## X1    -0.09 7.78
```

Maximum sale is 1042.65 and minimum is 10.68 whilemost sale is at 253.85

```
par(mfrow = c())
```

```
## named list()
```

```
hist(sales_f1$sales)
```



as you can confirm most sales are 200 and below.

4. Anomaly detection

```
# Load tidyverse and anomalize
# ---
#
library(tidyverse)
```

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

## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x psych::%+%()    masks ggplot2::%+%()
## x psych::alpha()  masks ggplot2::alpha()
## x dplyr::filter() masks tibbletime::filter(), stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
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 </>
```

```
# Detecting our anomalies
# The default values for time series decompose are method = "stl",
# which is just seasonal decomposition using a Loess smoother (refer to stats::stl()).

# anomalize() -
# We perform anomaly detection on the decomposed data using
# the remainder column through the use of the anomalize() function

# The alpha parameter is by default set to alpha = 0.05,
# but can be adjusted to increase or decrease the height of the anomaly bands,
# making it more difficult or less difficult for data to be anomalous.

# time_recompose()-
# We create the lower and upper bounds around the observed values
# through the use of the time_recompose() function,
#
# plot_anomalies() -
# we now plot using plot_anomaly_decomposition() to visualize out data.

anomaly.detect <- sales_f %>%
 group_by(Date) %>%
 summarise(totalsales = sum(eval(as.symbol("Sales")))) %>%
 ungroup() %>%
time_decompose(totalsales) %>%
anomalize(remainder, method = "gesd", alpha = 0.05, max_anoms = 0.2) %>%
plot_anomaly_decomposition()
```

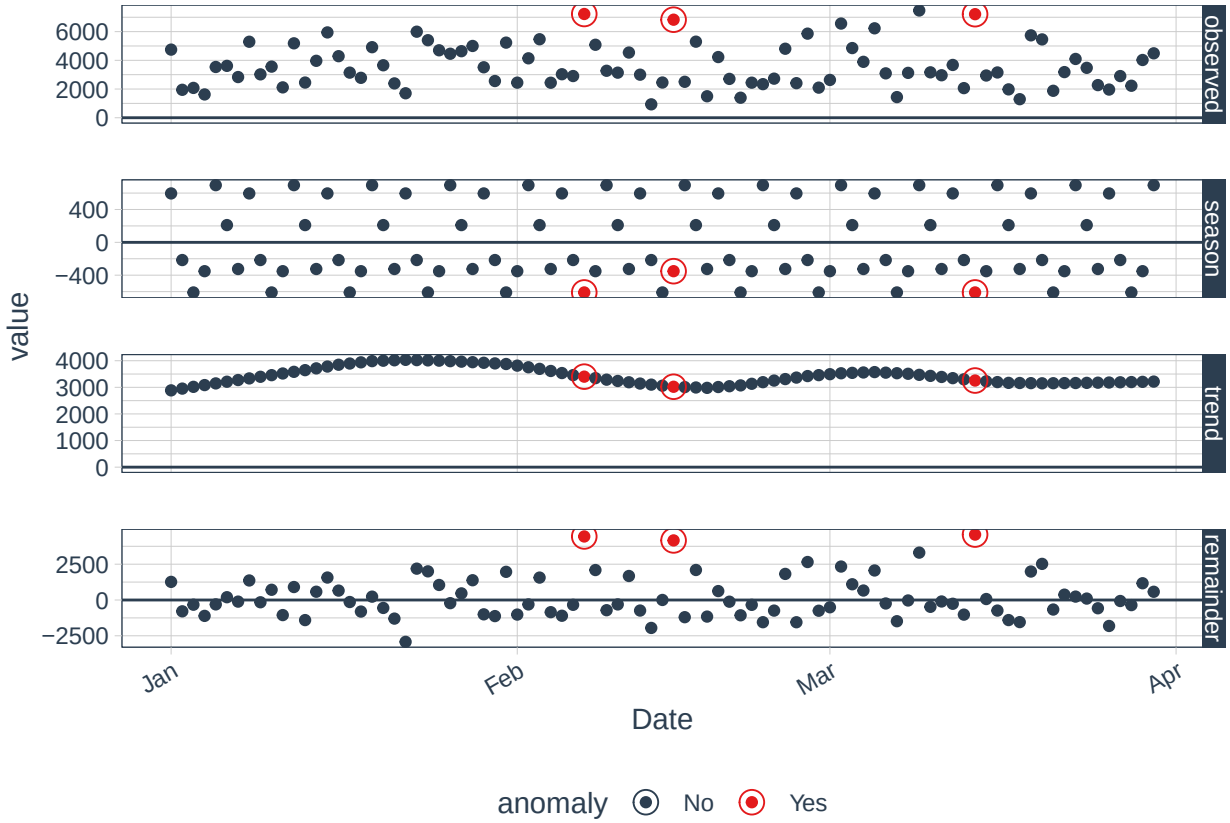
```
## frequency = 7 days

## trend = 30 days

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

## Warning: `type_convert()` only converts columns of type 'character'.
## - `df` has no columns of type 'character'
```

```
anomaly.detect
```



The anomalies as observed are experienced in the month of february, it might be because of the effect valentines' day. also the month of march.

5. Conclusion

The sales dataset contains anomalies shown by the red points in the graph, the marketing team need to check them out to accertain that they are not fraud.

The spikes in the dataset also shows the days where most people shop, this will be beneficial to the marketing team if they add promotions during that days.