Untitled

Anomaly Detection

```r
# Installing the required library
#

library('tibbletime')
```

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

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

```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 tibbletime::filter(), stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
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 </>
```

```r
sales <- read.csv('http://bit.ly/CarreFourSalesDataset')
head(sales)
```

```
##        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
```

```r
# Checking the dimensions of the dataset
dim(sales)
```

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

```r
#Checking the structure of the dataset
str(sales)
```

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

```r
#Changing the date column to its appropriate data type.
sales$Date <- as.Date(sales$Date, "%m/%d/%Y")
str(sales)
```

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

```r
# find the number of missing values in each column using is.na() and colSums() functions
colSums(is.na(sales))
```

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

No missing values

```r
# totalling the sales based on their common shared dates
sales_tot <- aggregate(sales$Sales, by = list(Date = sales$Date), FUN = sum)

head(sales_tot)
```

```
##         Date        x
## 1 2019-01-01 4745.181
## 2 2019-01-02 1945.503
## 3 2019-01-03 2078.128
## 4 2019-01-04 1623.688
## 5 2019-01-05 3536.684
## 6 2019-01-06 3614.205
```

```r
#Changing the column name x to total sales.
names(sales_tot)[2] <- 'Total_sales'
head(sales_tot)
```

```
##         Date Total_sales
## 1 2019-01-01    4745.181
## 2 2019-01-02    1945.503
## 3 2019-01-03    2078.128
## 4 2019-01-04    1623.688
## 5 2019-01-05    3536.684
## 6 2019-01-06    3614.205
```

```r
# Converting data frame to a tibble time (tbl_time)
# tbl_time have a time index that contains information about which column
# should be used for time-based subsetting and other time-based manipulation,
sales= tbl_time(sales, Date)
class(sales)
```

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

```r
# time_decompose() - this function would help with time series decomposition.
# anomalize() -
# We perform anomaly detection on the decomposed data using
# the remainder column through the use of the anomalize() function
# We 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.
# we now plot using plot_anomaly_decomposition() to visualize out data.

#options(repr.plot.width = 20, repr.plot.height = 20)

#tidyverse_cran_downloads %>%
#   time_decompose(count) %>%
#    anomalize(remainder) %>%
#    time_recompose() %>%
#    plot_anomalies(time_recomposed = TRUE, ncol = 3, alpha_dots = 0.5)
```