Anomaly

Oppy

2022-04-01

```
knitr::opts_chunk$set(error = TRUE)
```

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

```
## Warning: package 'tidyverse' was built under R version 4.1.3
```

```
## -- 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
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```
## -- 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)
```

```
## Warning: package 'anomalize' was built under R version 4.1.3
```

```
## == 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(vctrs)
```

```
##
## Attaching package: 'vctrs'

## The following object is masked from 'package:dplyr':
##
##     data_frame

## The following object is masked from 'package:tibble':
##
##     data_frame
```

```r
library(tidyr)
library(dplyr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
df <- read_csv("http://bit.ly/CarreFourSalesDataset")
```

```
## Rows: 1000 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Date
## dbl (1): Sales
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
head(df)
```

```
## # A tibble: 6 x 2
##   Date      Sales
##   <chr>     <dbl>
## 1 1/5/2019  549.
## 2 3/8/2019   80.2
## 3 3/3/2019  341.
## 4 1/27/2019 489.
## 5 2/8/2019  634.
## 6 3/25/2019 628.
```

Checking the datatypes

```r
#checking the datatypes
str(df)
```

```
## spec_tbl_df [1,000 x 2] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Date : chr [1:1000] "1/5/2019" "3/8/2019" "3/3/2019" "1/27/2019" ...
##  $ Sales: num [1:1000] 549 80.2 340.5 489 634.4 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Date = col_character(),
##   ..   Sales = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Change the date column datatype

```r
library(lubridate)
#changing the datatype
df$Date<- as.Date(df$Date,"%m/%d/%y")

#Confirm the changes made
df
```

```
## # A tibble: 1,000 x 2
##    Date       Sales
##    <date>     <dbl>
##  1 2020-01-05 549.
##  2 2020-03-08  80.2
##  3 2020-03-03 341.
##  4 2020-01-27 489.
##  5 2020-02-08 634.
##  6 2020-03-25 628.
##  7 2020-02-25 434.
##  8 2020-02-24 772.
##  9 2020-01-10  76.1
## 10 2020-02-20 173.
## # ... with 990 more rows
```

Checking for null and duplicate values

```r
colSums(is.na(df))
```

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

```r
#no need for this sine there is no null values to be removed.
df1 <- df[complete.cases(df),]
df1[complete.cases(df),]
```

```
## # A tibble: 1,000 x 2
##    Date       Sales
##    <date>     <dbl>
```

```
##  1 2020-01-05 549.
##  2 2020-03-08  80.2
##  3 2020-03-03 341.
##  4 2020-01-27 489.
##  5 2020-02-08 634.
##  6 2020-03-25 628.
##  7 2020-02-25 434.
##  8 2020-02-24 772.
##  9 2020-01-10  76.1
## 10 2020-02-20 173.
## # ... with 990 more rows
```

```r
colSums(is.na(df1))
```

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

```r
head(df1)
```

```
## # A tibble: 6 x 2
##   Date       Sales
##   <date>     <dbl>
## 1 2020-01-05 549.
## 2 2020-03-08  80.2
## 3 2020-03-03 341.
## 4 2020-01-27 489.
## 5 2020-02-08 634.
## 6 2020-03-25 628.
```

Confirming the changes

```r
sum(is.na(df1))
```

```
## [1] 0
```

```r
# group and tally the number of transactions per day
dff <- df1 %>% group_by(Date) %>% tally()
colnames(dff) <- c('transactionDate', 'totalCount')
head(dff)
```

```
## # A tibble: 6 x 2
##   transactionDate totalCount
##   <date>               <int>
## 1 2020-01-01              12
## 2 2020-01-02               8
## 3 2020-01-03               8
## 4 2020-01-04               6
## 5 2020-01-05              12
## 6 2020-01-06               9
```

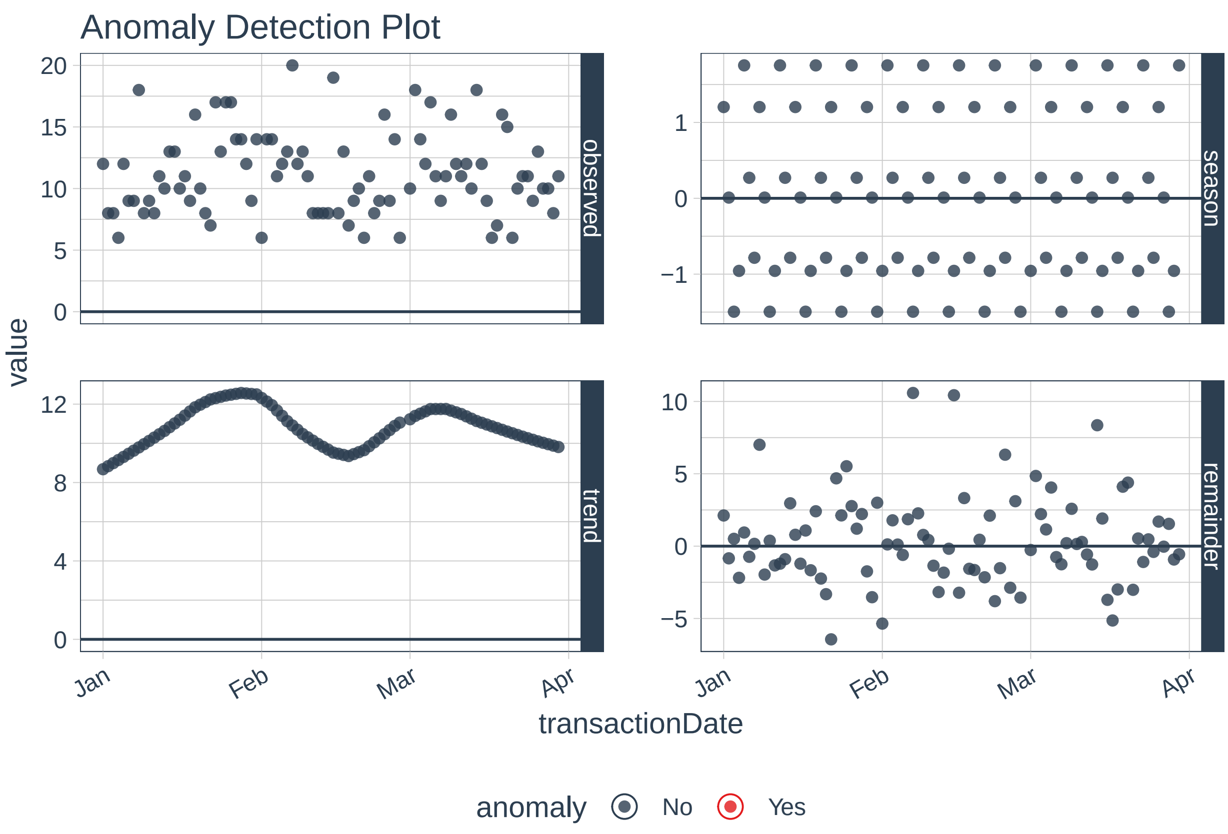
```r
dff %>%
    time_decompose(totalCount, merge = TRUE) %>%
    anomalize(remainder) %>%
    plot_anomaly_decomposition(ncol = 2, alpha_dots = .8) +
    ggtitle("Anomaly Detection Plot")
```

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

## frequency = 7 days

## trend = 30 days

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



Parameter Tuning

We will use the max anoms and alpha parameters for tuning

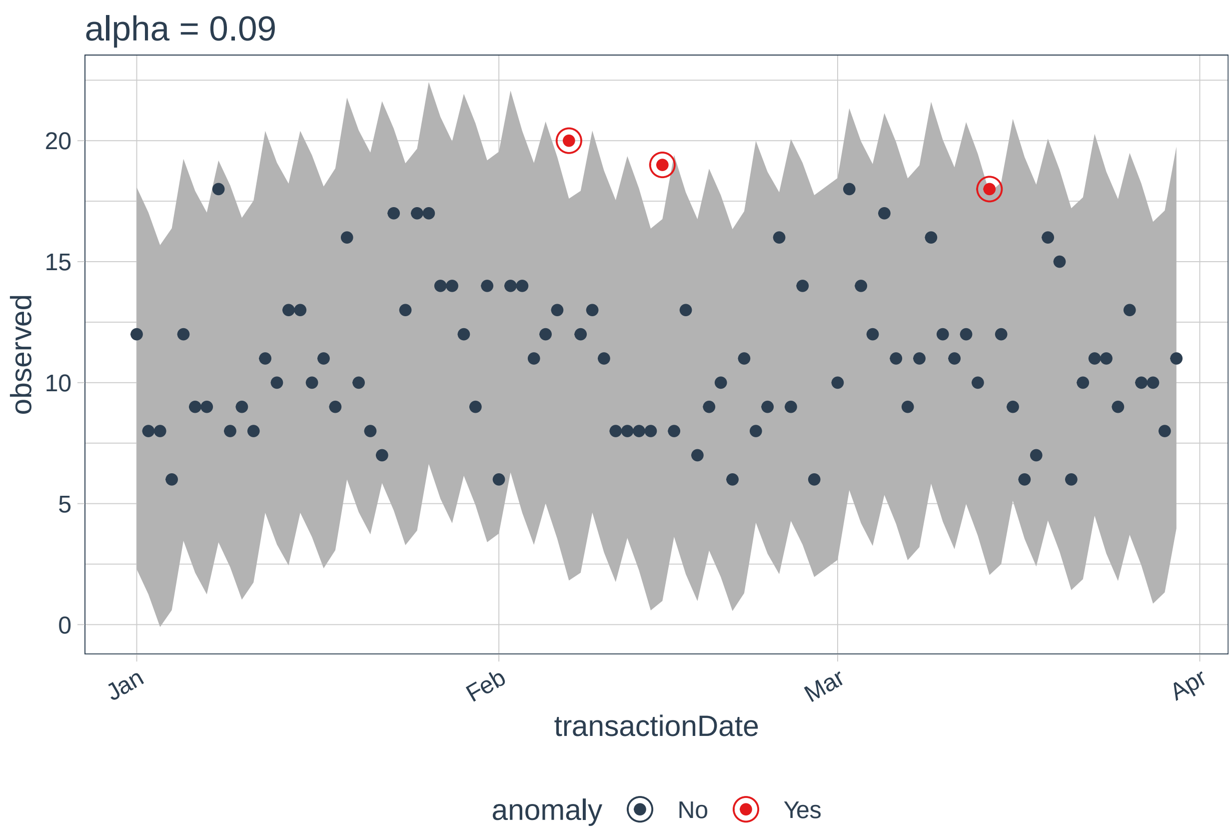
```r
#visualize the anomalies using the plot_anomalies() function.
dff %>%
 time_decompose(totalCount)%>%
  anomalize(remainder, alpha = 0.09, max_anoms = 0.10)%>%
```

```
  time_recompose()%>%
  plot_anomalies(time_recompose = T)+
  ggtitle("alpha = 0.09")
```

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

## frequency = 7 days

## trend = 30 days
```

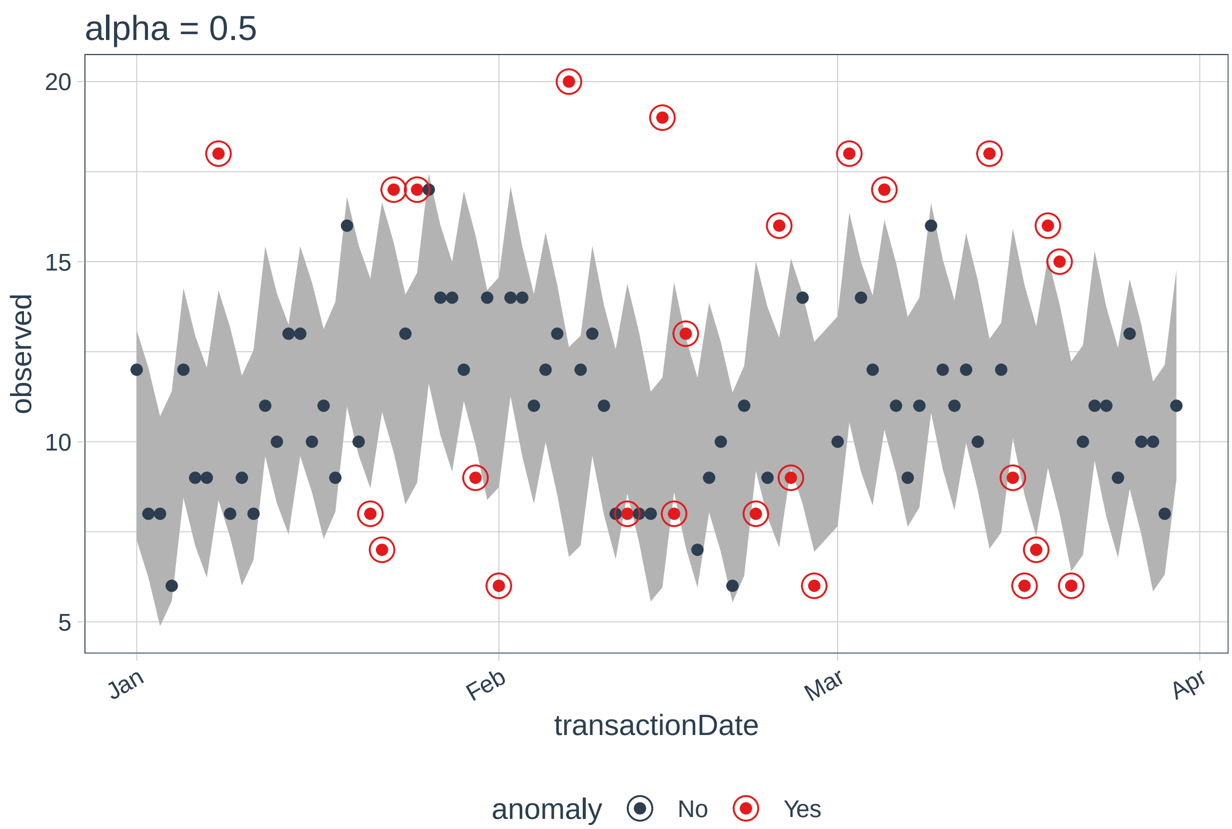



```
#Tuning the Alpha parameter
dff%>%
  time_decompose(totalCount)%>%
  anomalize(remainder, alpha = 0.5, max_anoms = 0.5)%>%
  time_recompose()%>%
  plot_anomalies(time_recompose = T)+
  ggtitle("alpha = 0.5")
```

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

## frequency = 7 days
```

```
## trend = 30 days
```



When the alpha level and max_anoms are increased, more anomalies are observed

In IQR a distribution is taken and 25% and 75% inner quartile range to establish the distribution of the remainder. Limits are set by default to a factor of 3 times above, and below the inner quartile range, any remainder beyond the limit is considered as an anomaly.

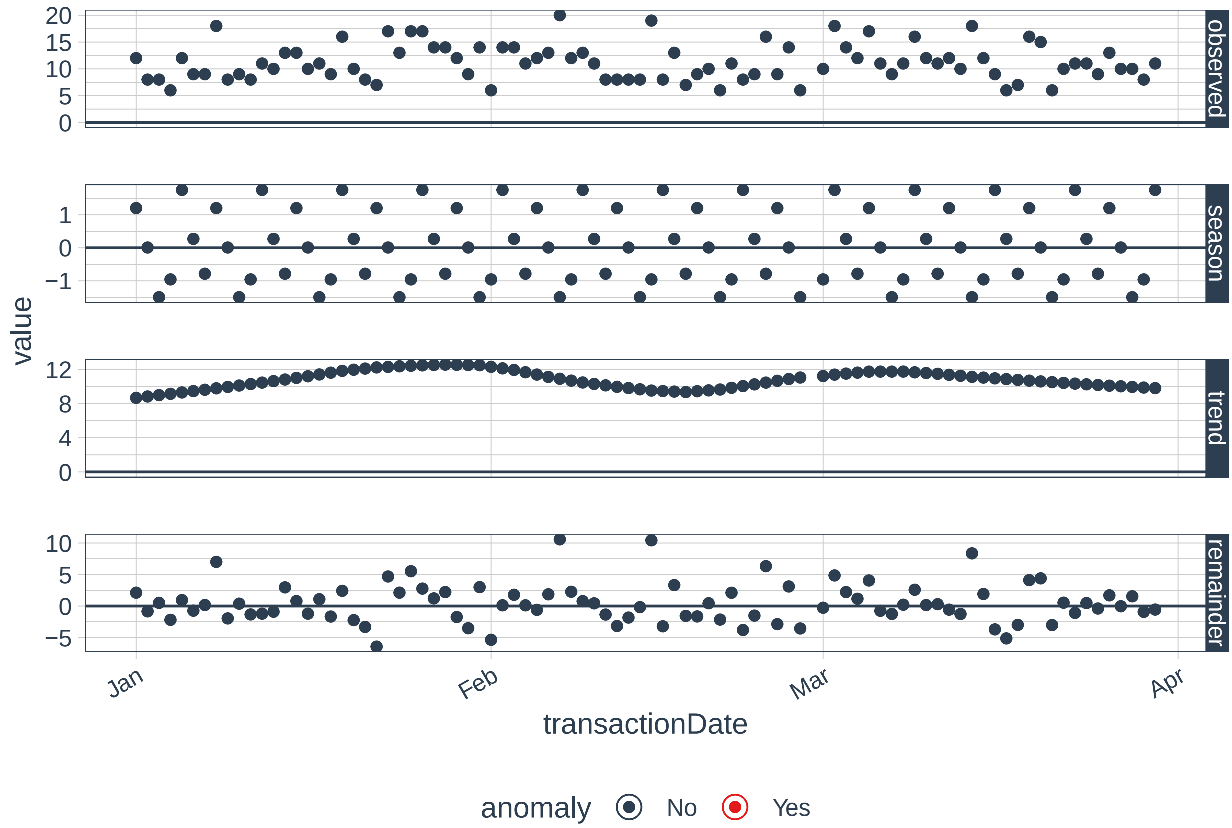
```
iqr<-dff %>%
    time_decompose(totalCount, method = "stl") %>%
    anomalize(remainder, method = "iqr")
```

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

## frequency = 7 days

## trend = 30 days
```

```
plot_anomaly_decomposition(iqr)
```



In GESD anomalies are progressively evaluated removing the worst offenders and recalculating the test statistics and critical values, or more simply you can say that a range is recalculated after identifying the anomalies in an iterative way.

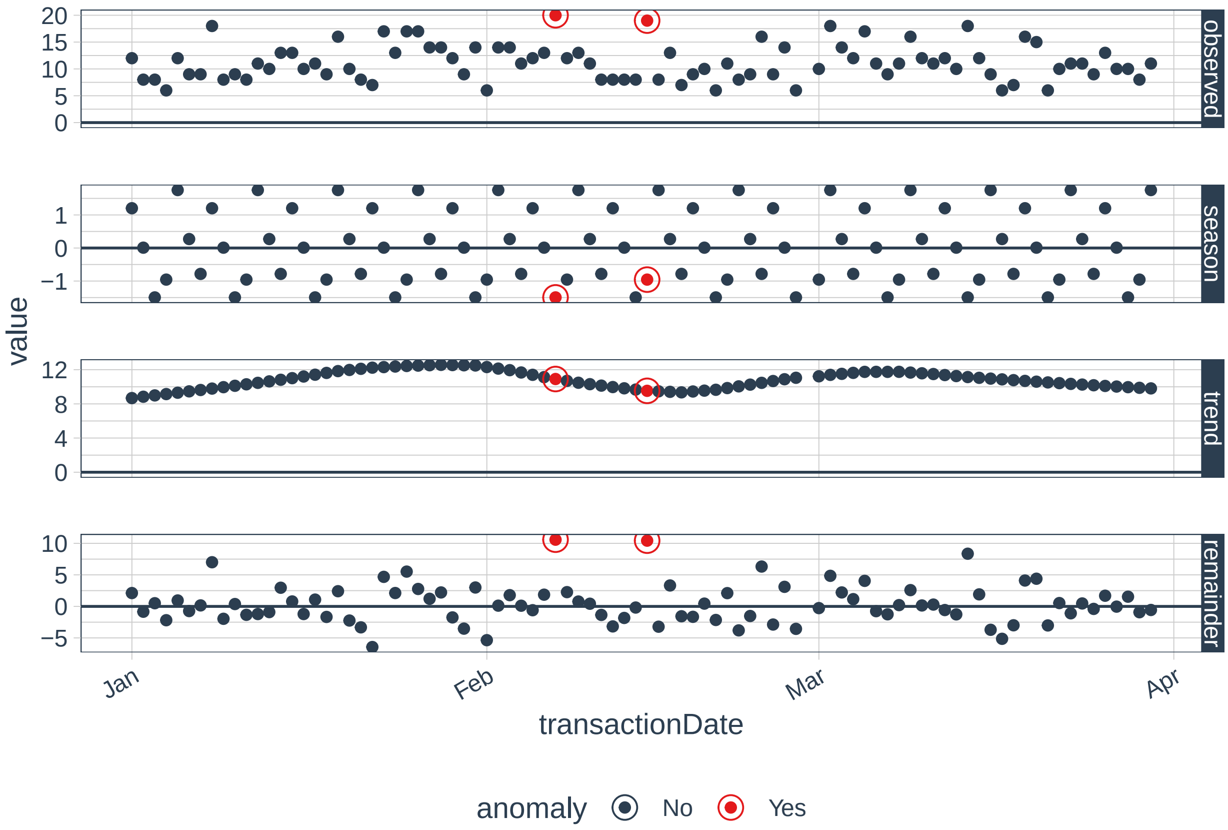
```
gesd <-dff %>%
    time_decompose(totalCount, method = "stl") %>%
    anomalize(remainder, method = "gesd")
```

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

```
## frequency = 7 days
```

```
## trend = 30 days
```

```
plot_anomaly_decomposition(gesd)
```



GESD is more accurate since it detects more anomalies than IQR with the same hyperparameters