Anomaly Detection

2022-04-01

```
# Import packages
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 </>
```

```
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
```

```
## Warning: package 'dplyr' 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)
```

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

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

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

```
library(dplyr)
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
```

```
# Load the dataset
data <- fread("http://bit.ly/CarreFourSalesDataset")
# Preview the dataset
head(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
```

```
# Check the datatype of our dataset
str(data)
```

```
## Classes 'data.table' and '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 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
# Check the number of rows and columns in our dataset
dim(data)
```

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

we have 1000 records and 2 variables

```
# Check the summary of our dataset
summary(data)
```

```
##      Date               Sales
##  Length:1000        Min.   :  10.68
##  Class :character   1st Qu.: 124.42
##  Mode  :character   Median : 253.85
##                     Mean   : 322.97
##                     3rd Qu.: 471.35
##                     Max.   :1042.65
```

```r
# Check for missing values
colSums(is.na(data))
```

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

We have no missing values in our dataset

```r
# Check for duplicates
sum(duplicated(data))
```

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

We have no duplicates

```r
# We will convert our date variable to it's correct datatype
data$Date <- as.Date(data$Date, format = "%m/%d/%y")

#Convertion to POCIXct type
data$Date <- as.POSIXct(data$Date)

#changing to tibble

data.tb <- as_tibble(data)
head(data.tb)
```

```
## # A tibble: 6 x 2
##   Date                Sales
##   <dttm>              <dbl>
## 1 2020-01-05 03:00:00 549.
## 2 2020-03-08 03:00:00  80.2
## 3 2020-03-03 03:00:00 341.
## 4 2020-01-27 03:00:00 489.
## 5 2020-02-08 03:00:00 634.
## 6 2020-03-25 03:00:00 628.
```

Workflow of anomalize:

1. Time series decomposition
2. Anomaly detection with remainder
3. Anomaly lower and upper bound transformation

```r
# Using amomalize
anom.data<- data.tb %>%
  time_decompose(Sales, merge = T) %>%
  anomalize(remainder)%>%
  time_recompose()
```

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

```
## Note: Index not ordered. tibbletime assumes index is in ascending order. Results may not be as desire

## frequency = 12 seconds

## Note: Index not ordered. tibbletime assumes index is in ascending order. Results may not be as desire

## trend = 12 seconds

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

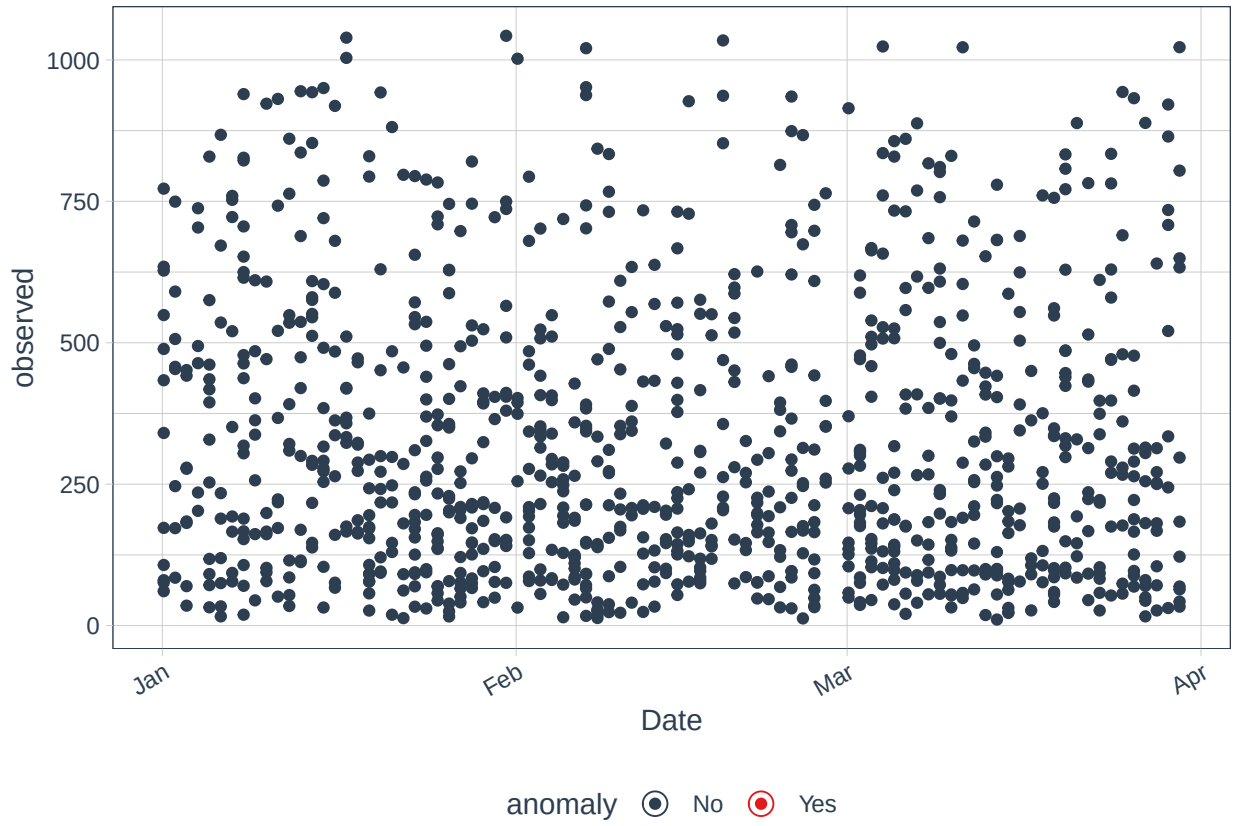
```
anom.data %>% glimpse()
```

```
## Rows: 1,000
## Columns: 11
## $ Date          <dttm> 2020-01-01 03:00:00, 2020-01-01 03:00:00, 2020-01-01 03~
## $ Sales         <dbl> 457.4430, 399.7560, 470.6730, 388.2900, 132.7620, 132.02~
## $ observed      <dbl> 548.9715, 80.2200, 340.5255, 489.0480, 634.3785, 627.616~
## $ season        <dbl> -14.359190, -4.462252, 28.744495, 23.243172, -13.844108,~
## $ trend         <dbl> 445.2248, 445.5012, 445.7776, 435.9271, 426.0767, 416.19~
## $ remainder     <dbl> 118.105886, -360.818930, -133.996555, 29.877684, 222.145~
## $ remainder_l1  <dbl> -917.358, -917.358, -917.358, -917.358, -917.358, -917.3~
## $ remainder_l2  <dbl> 946.1539, 946.1539, 946.1539, 946.1539, 946.1539, 946.15~
## $ anomaly       <chr> "No", "No", "No", "No", "No", "No", "No", "No", "No", "N~
## $ recomposed_l1 <dbl> -486.4924, -476.3191, -442.8360, -458.1877, -505.1254, -~
## $ recomposed_l2 <dbl> 1377.020, 1387.193, 1420.676, 1405.324, 1358.387, 1372.7~
```

remainder_l1 is the lower limit of the remainder and remainder_l2 is the upper limit of the remainder. The anomaly column states whether the observation is an anomaly or not

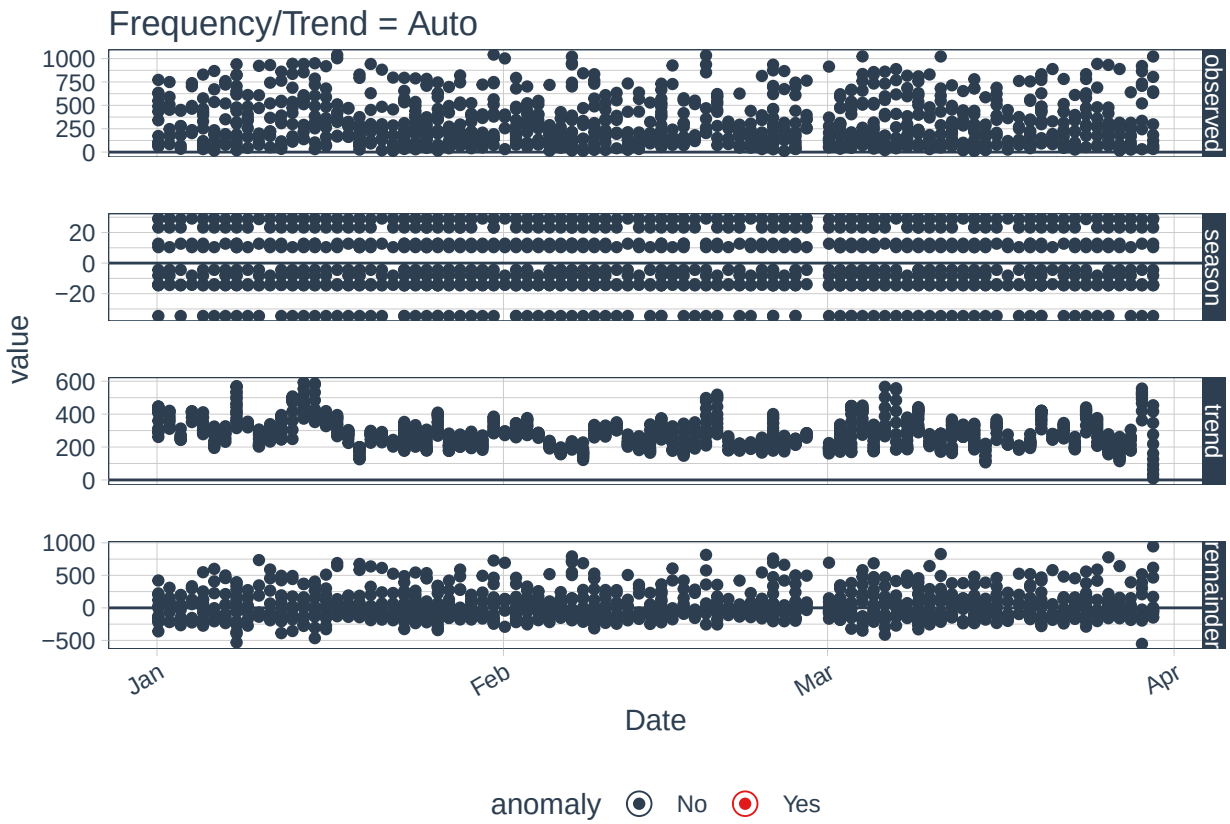
recomposed_l1 is the lower bound of outliers around the observed value recomposed_l2 is the upper bound of outliers around the observed value

```
# View anomalies
anom.data %>%
  plot_anomalies(ncol = 3)
```



No anomalies detected

```
# Trend and Seasonality
anomaly_1 <- anom.data %>%
  plot_anomaly_decomposition()+
  ggtitle("Frequency/Trend = Auto")
anomaly_1
```




```
# Adjusting parameters for anomaly detection
# Alpha adjusts controls the band of the limit, decreasing alpha increases
# size of the band making it difficult for a point to be an anomaly
# max_anoms controls the percentage of data that can be an anomaly
data.tb %>%
  time_decompose(Sales) %>%
  anomalize(remainder,alpha =0.05,max_anoms = 0.2)%>%
  time_recompose()%>%
  plot_anomaly_decomposition()
```

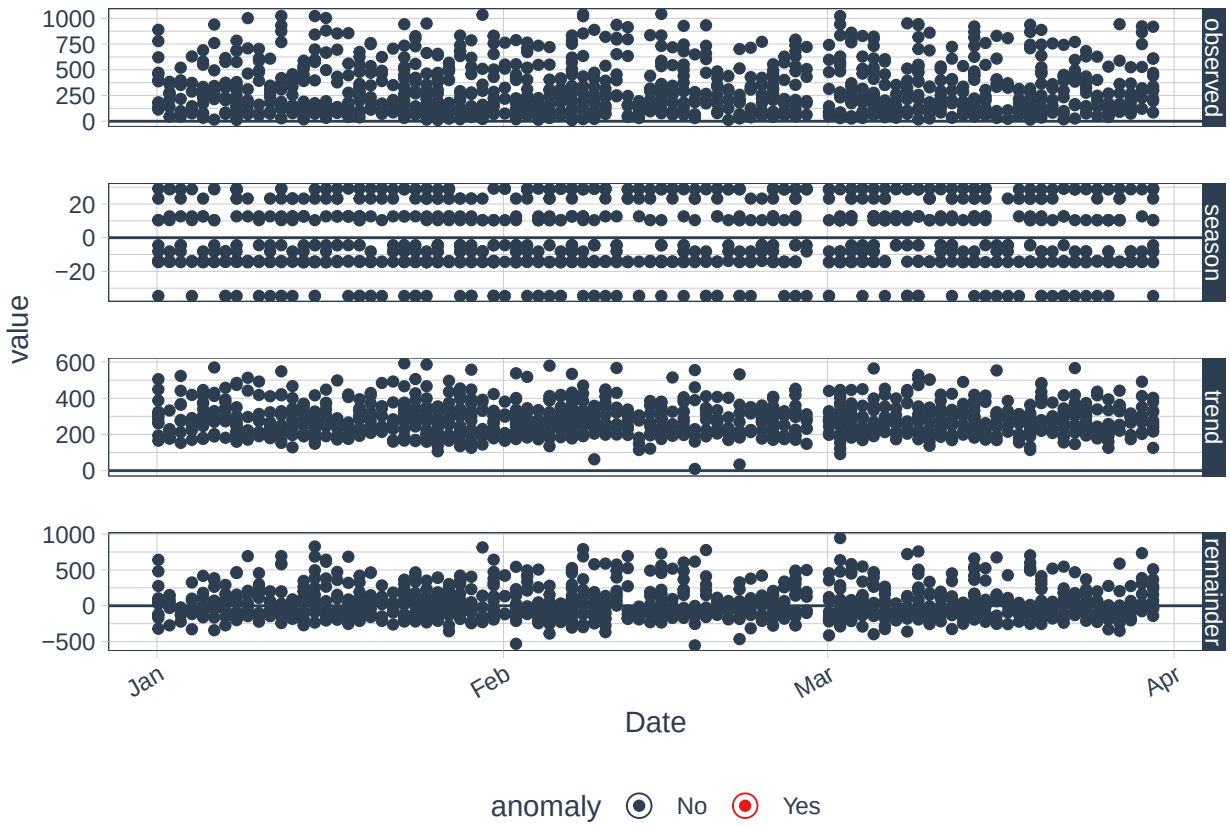
```
## Converting from tbl_df to tbl_time.
## Auto-index message: index = Date
```

```
## Note: Index not ordered. tibbletime assumes index is in ascending order. Results may not be as desire
```

```
## frequency = 12 seconds
```

```
## Note: Index not ordered. tibbletime assumes index is in ascending order. Results may not be as desire
```

```
## trend = 12 seconds
```




```
# Adjusting parameters
data.tb %>%
  time_decompose(Sales) %>%
  anomalize(remainder,alpha =0.3,max_anoms = 0.2)%>%
  time_recompose()%>%
  plot_anomaly_decomposition()
```

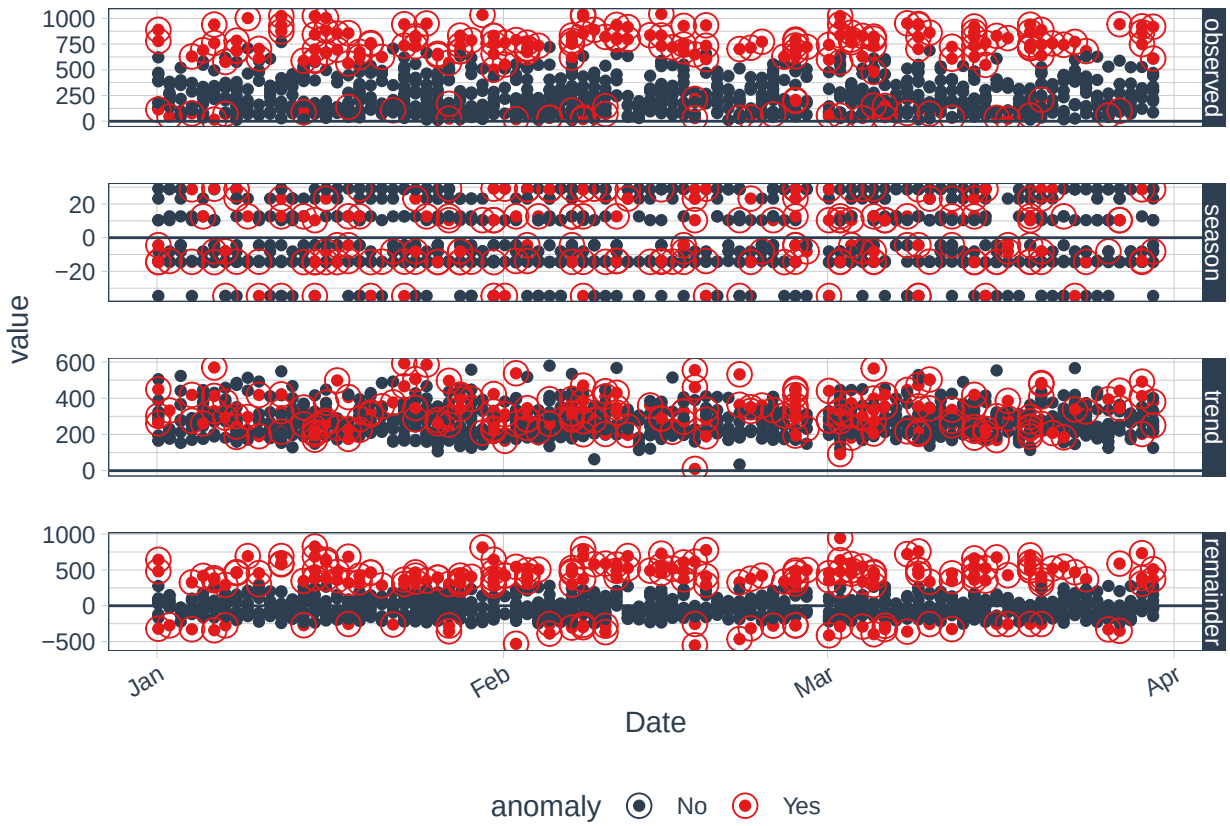
```
## Converting from tbl_df to tbl_time.
## Auto-index message: index = Date

## Note: Index not ordered. tibbletime assumes index is in ascending order. Results may not be as desir

## frequency = 12 seconds

## Note: Index not ordered. tibbletime assumes index is in ascending order. Results may not be as desir

## trend = 12 seconds
```



After increasing alpha from 0.05 to 0.3, we can see significantly more anomalies

```
# Adjusting alpha and max anoms
data.tb %>%
  time_decompose(Sales) %>%
  anomalize(remainder,alpha =0.3,max_anoms = 0.05)%>%
  time_recompose()%>%
  plot_anomaly_decomposition()
```

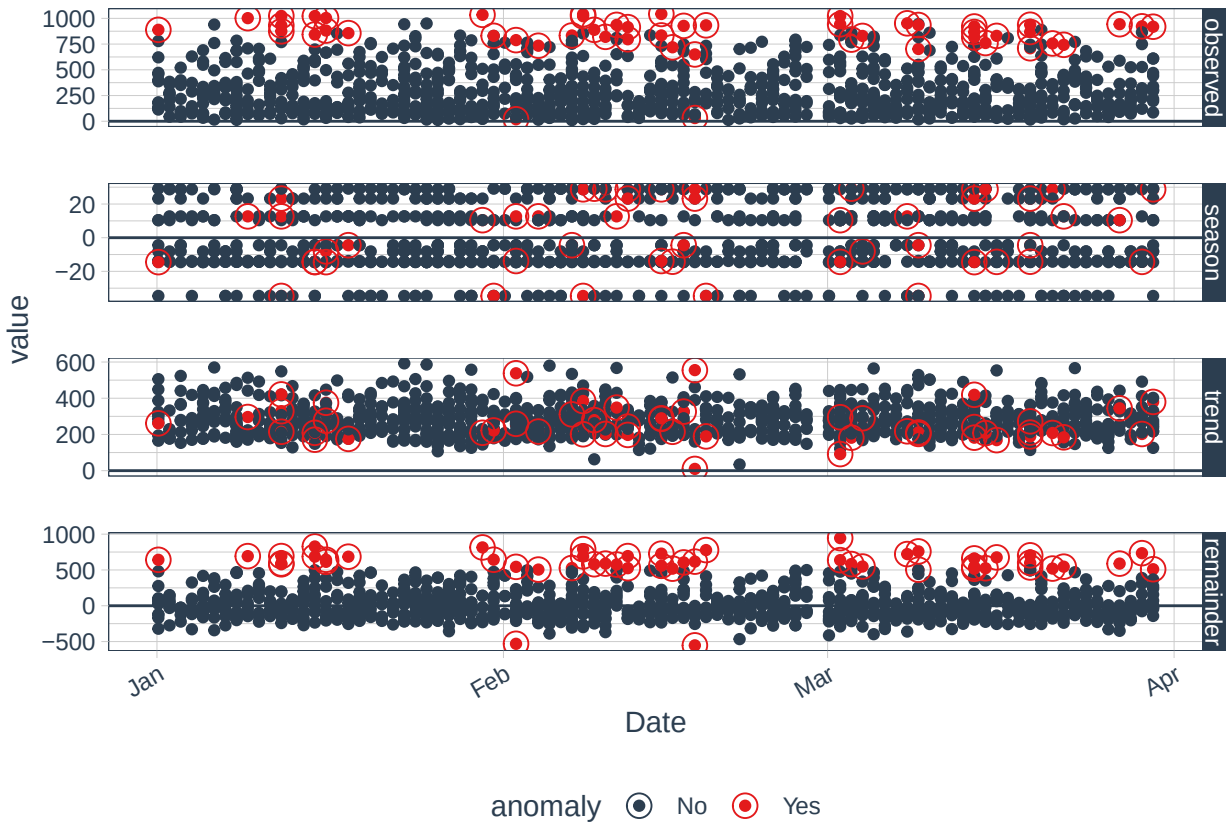
```
## Converting from tbl_df to tbl_time.
## Auto-index message: index = Date

## Note: Index not ordered. tibbletime assumes index is in ascending order. Results may not be as desir

## frequency = 12 seconds

## Note: Index not ordered. tibbletime assumes index is in ascending order. Results may not be as desir

## trend = 12 seconds
```



Once we reduced max_anoms from 0.2 to 0.05 we can view much less anomalies than our previous illustration.

```
# Anomalize implements 2 methods: IQR and GESD
# Let's check how both of them perform
# Using IQR
data.tb %>%
  time_decompose(Sales, method = "stl",frequency = "auto", trend = "auto")%>%
  anomalize(remainder, method="iqr", alpha = 0.05, max_anoms = 0.2)%>%
  plot_anomaly_decomposition()
```

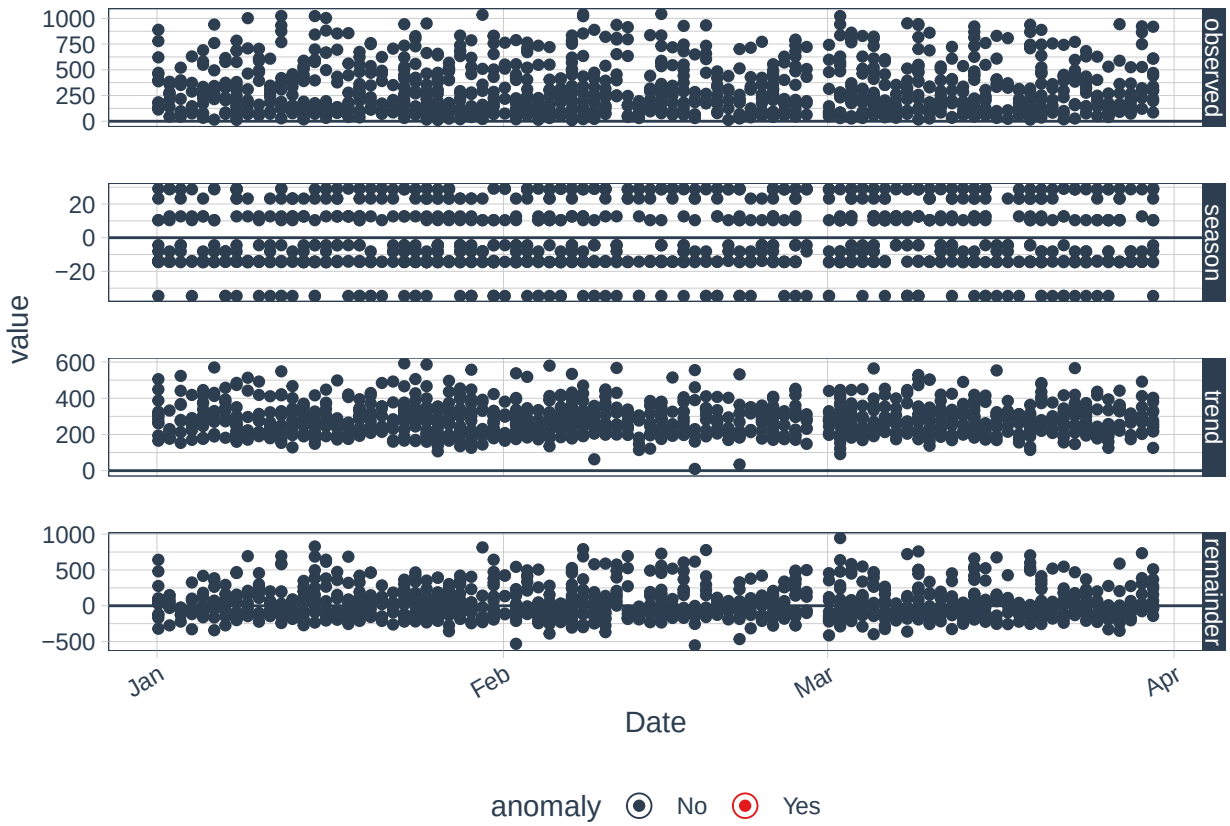
```
## Converting from tbl_df to tbl_time.
## Auto-index message: index = Date

## Note: Index not ordered. tibbletime assumes index is in ascending order. Results may not be as desir

## frequency = 12 seconds

## Note: Index not ordered. tibbletime assumes index is in ascending order. Results may not be as desir

## trend = 12 seconds
```




```
# Using GESD
data.tb %>%
  time_decompose(Sales, method = "stl",frequency = "auto", trend = "auto")%>%
  anomalize(remainder, method="gesd", alpha = 0.05, max_anoms = 0.2)%>%
  plot_anomaly_decomposition()
```

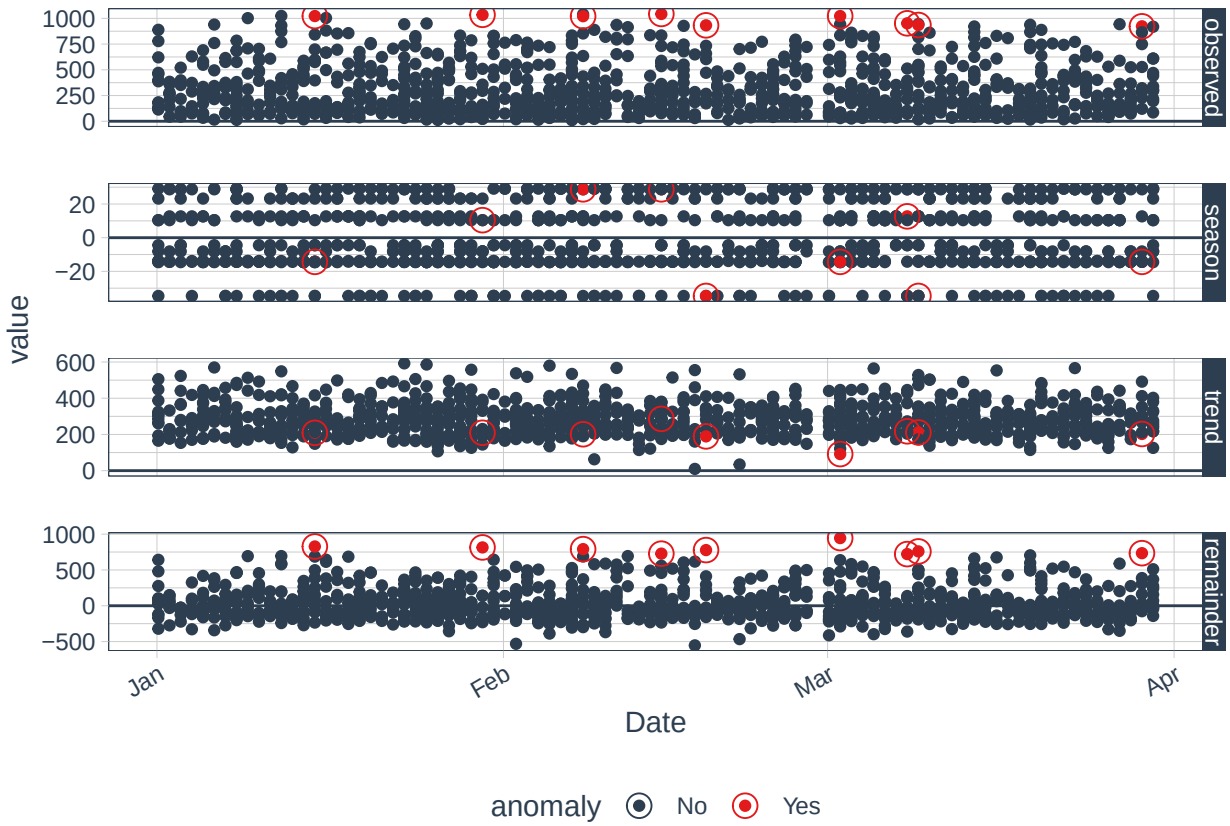
```
## Converting from tbl_df to tbl_time.
## Auto-index message: index = Date

## Note: Index not ordered. tibbletime assumes index is in ascending order. Results may not be as desir

## frequency = 12 seconds

## Note: Index not ordered. tibbletime assumes index is in ascending order. Results may not be as desir

## trend = 12 seconds
```



IQR is faster than GESD but it is not as accurate as GESD. With the same parameters set GESD was able to detect anomalies while IQR was not able to.