R Notebook

R Programming - Anomaly Detection

EXAMPLE

The following example has been borrowed from the following article: https://www.business-science.io/code-tools/2018/04/08/introducing-anomalize.html

```
# Find the anomalies on the following given time series dataset.
```

```
# Installing anomalize package
# install.packages("anomalize")
```

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

```
## Warning: package 'tidyverse' was built under R version 4.0.5
```

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

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## Warning: package 'ggplot2' was built under R version 4.0.4
```

```
## Warning: package 'tibble' was built under R version 4.0.4
```

```
## Warning: package 'tidyr' was built under R version 4.0.4
```

```
## Warning: package 'readr' was built under R version 4.0.4
```

```
## Warning: package 'purrr' was built under R version 4.0.4
```

```
## Warning: package 'dplyr' was built under R version 4.0.4
```

```
## Warning: package 'stringr' was built under R version 4.0.4
```

```
## Warning: package 'forcats' was built under R version 4.0.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(anomalize)
```

```
## Warning: package 'anomalize' was built under R version 4.0.5
```

```
## == 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
# Collect our time series data
tidyverse_cran_downloads
```

```
## # A tibble: 6,375 x 3
## # Groups:   package [15]
##    date       count package
##    <date>     <dbl> <chr>
##  1 2017-01-01   873 tidyr
##  2 2017-01-02  1840 tidyr
##  3 2017-01-03  2495 tidyr
##  4 2017-01-04  2906 tidyr
##  5 2017-01-05  2847 tidyr
##  6 2017-01-06  2756 tidyr
##  7 2017-01-07  1439 tidyr
##  8 2017-01-08  1556 tidyr
##  9 2017-01-09  3678 tidyr
## 10 2017-01-10  7086 tidyr
## # ... with 6,365 more rows
```

```r
# Detecting our anomalies

# We now use the following functions to detect and visualize anomalies;
# We decomposed the "count" column into "observed", "season", "trend", and "remainder" columns.
# The default values for time series decompose are method = "stl", which is just seasonal decomposition
# The frequency and trend parameters are automatically set based on the time scale (or periodicity) of

# 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
# which procides 3 new columns; "remainder_l1" (lower limit),
# "remainder_l2" (upper limit), and "anomaly" (Yes/No Flag).
# The default method is method = "iqr", which is fast and relatively
# accurate at detecting anomalies.
# 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.
# The max_anoms parameter is by default set to a maximum of max_anoms = 0.2
# for 20% of data that can be anomalous.

# time_recompose()-
# 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 create new columns created: "recomposed_l1" (lower limit)
# and "recomposed_l2" (upper limit).
#
# plot_anomalies() -
# we now plot using plot_anomaly_decomposition() to visualize out data.

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

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

CHALLENGE

```
# Find the anomalies on the following given time series dataset.

library('tibbletime')
```

```
## Warning: package 'tibbletime' was built under R version 4.0.5
```

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

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

```
library('dplyr')
library('tidyverse')

logs_path <- "http://bit.ly/LogsDataset"

# Grouping by server and converting to tibbletime
security_access_logs <- read_csv(logs_path) %>%
  group_by(server) %>%
  as_tbl_time(date)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   date = col_date(format = ""),
##   count = col_double(),
##   server = col_character()
## )
```

```
security_access_logs
```

```
## # A time tibble: 198 x 3
## # Index:  date
## # Groups: server [3]
##    date       count server
##    <date>     <dbl> <chr>
##  1 2017-05-22     7 SERVER-549521
##  2 2017-05-23     9 SERVER-549521
##  3 2017-05-24    12 SERVER-549521
##  4 2017-05-25     4 SERVER-549521
##  5 2017-05-26     4 SERVER-549521
##  6 2017-05-30     2 SERVER-549521
##  7 2017-05-31    10 SERVER-549521
##  8 2017-06-01    14 SERVER-549521
##  9 2017-06-02    12 SERVER-549521
## 10 2017-06-05     7 SERVER-549521
## # ... with 188 more rows
```

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