Google Algorithm Anomalies

R Moseley

2022-07-20

Description:

The goal of the markdown is to show data wrangling, visualization, interpretive skills & storytelling abilities.

In early 2018, a website was impacted by a Google algorithm update that disrupted search rankings and ultimately caused significantly fewer non-paid search engine traffic to the website. I was asked to complete a deep dive & impact analysis. I have also completed a deck to present the findings to a theoretical executive board.

Focus of this task includes:

- Describe the timeline of the Google algorithm update
- Quantify the traffic impact of the Google algorithm update
- Highlight performance & activity of noteworthy segments
- Prepare a breakdown of any methodologies in the appendix

Data

- 4.3M records, daily & segmented aggregated session volume data

Data Dictionary

- session_date (calendar date)
- search_engine (applicable search engine used prior to entry)
- mkt_channel (traffic channel)
- is_paid_traffic (true/false if resulting from paid traggic campaigns)
- platform_type (device category)
- entry_page (entry point page name)
- page_rollup (entry point page group)
- session_count (aggregate entry sessions metric)

Load Packages

```r
library(dplyr)
library(anomalize)
library(tidyverse)
library(zoo)
library(TTR)
```

These are the packages required for doing the preliminary investigation.

Load the Data

```r
traffic_data <- read.csv("~/Documents/GitHub/projects/storytelling/site-traffic-anomalies/traffic_data.
```

Here I am loading the data to my environment.

Explore the Data

Initially, I will back up the data in case it needs to be loaded again due to any changes.

```r
traffic_data$date <- as.Date(traffic_data$session_date,format="%Y-%m-%d")
traffic_data_backup<-traffic_data
traffic_data<-traffic_data_backup
```

Next, it would be beneficial to check the head and see the type of data we are dealing with.

```r
head(traffic_data,5)
```

```
##   session_date search_engine mkt_channel is_paid_traffic  platform_type
## 1   2018-05-06         other         sem            TRUE Laptop/Desktop
## 2   2018-05-06    duckduckgo     organic           FALSE    Smartphones
## 3   2018-05-06         other      direct           FALSE    Smartphones
## 4   2018-05-20         other      direct           FALSE Laptop/Desktop
## 5   2018-05-20        google         sem            TRUE    Smartphones
##    browser_type                          entry_page      page_rollup
## 1       Unknown                      new_model_core   new model page
## 2 Mobile Safari                      new_model_core   new model page
## 3 Mobile Safari mobile_car_reviews_features_article mobile editorial
## 4          Edge                           home_page        home page
## 5 Chrome Mobile  mobile_advice_safety_article_index mobile editorial
##   sesssion_count       date
## 1            211 2018-05-06
## 2            102 2018-05-06
## 3             26 2018-05-06
## 4           1111 2018-05-20
## 5             38 2018-05-20
```

Looks good so far.

Prepare data for Anomaly Detection

```
df_unpaid_daily_goog<-traffic_data_backup %>%
  filter(is_paid_traffic ==FALSE)%>%
  filter(search_engine =='google')%>%
  group_by(date) %>%
  mutate(session_count_sum=sum(sesssion_count))

df_unpaid_daily_goog<-df_unpaid_daily_goog[order(as.Date(df_unpaid_daily_goog$date, format="%Y-%m-%d"))

df_unpaid_daily_goog_short <-df_unpaid_daily_goog%>%
  select(date,session_count_sum)%>%
  distinct()

df <- as_tibble(df_unpaid_daily_goog_short)
```

We are preparing the data to find the anomaly using time decomposition package

```
df_anomalized <- df %>%
  time_decompose(session_count_sum, merge = TRUE) %>%
  anomalize(remainder) %>%
  time_recompose()
```

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

## frequency = 7 days

## trend = 90.5 days

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

In the time decomposition, we can see this obvious detail that the frequency would be 7 days, as in there would be a weekly pattern in the data.

...

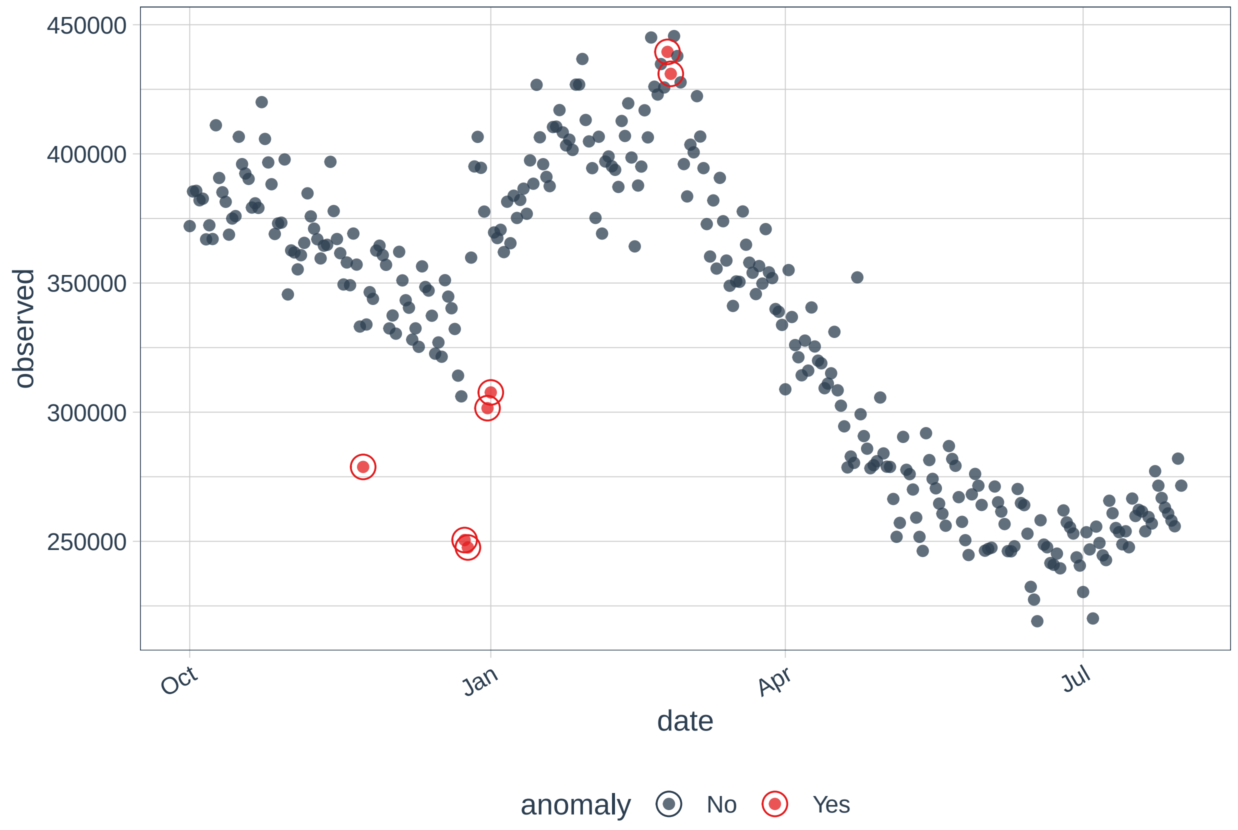
```
df_anomalized %>% glimpse()
```

```
## Rows: 304
## Columns: 11
## Index: date
## $ date              <date> 2017-10-01, 2017-10-02, 2017-10-03, 2017-10-04, 201~
## $ session_count_sum <int> 372039, 385453, 385688, 382033, 382651, 366910, 3723~
## $ observed          <dbl> 372039, 385453, 385688, 382033, 382651, 366910, 3723~
## $ season            <dbl> -10274.992, 17589.622, 6199.648, 2094.484, -1223.248~
## $ trend             <dbl> 391645.7, 391035.6, 390425.5, 389815.4, 389205.3, 38~
## $ remainder         <dbl> -9331.663, -23172.193, -10937.136, -9876.888, -5331.~
## $ remainder_l1      <dbl> -55140.31, -55140.31, -55140.31, -55140.31, -55140.3~
## $ remainder_l2      <dbl> 53149.47, 53149.47, 53149.47, 53149.47, 53149.47, 53~
## $ anomaly           <chr> "No", "No", "No", "No", "No", "No", "No", "No", "No"~
## $ recomposed_l1     <dbl> 326230.4, 353484.9, 341484.8, 336769.6, 332841.8, 32~
## $ recomposed_l2     <dbl> 434520.1, 461774.7, 449774.6, 445059.4, 441131.5, 43~
```

Here we can see some detail on these results, now we can prepare the visualization.

Visualize the Outliers

```r
#determining data outliers in the complete time series
unpaid_goog_anomaly<-df_anomalized %>% plot_anomalies(ncol = 3, alpha_dots = 0.75)

unpaid_goog_anomaly
```



We can see that there are two possible times were there were Google algorithm changes, one of the times is at the end of December, and one looks like February-March. However, these December dates are not in the determined anomaly date, so they need to be removed.

```r
#save results
ggsave("unpaid_goog_anomaly.png")
```

```
## Saving 6.5 x 4.5 in image
```

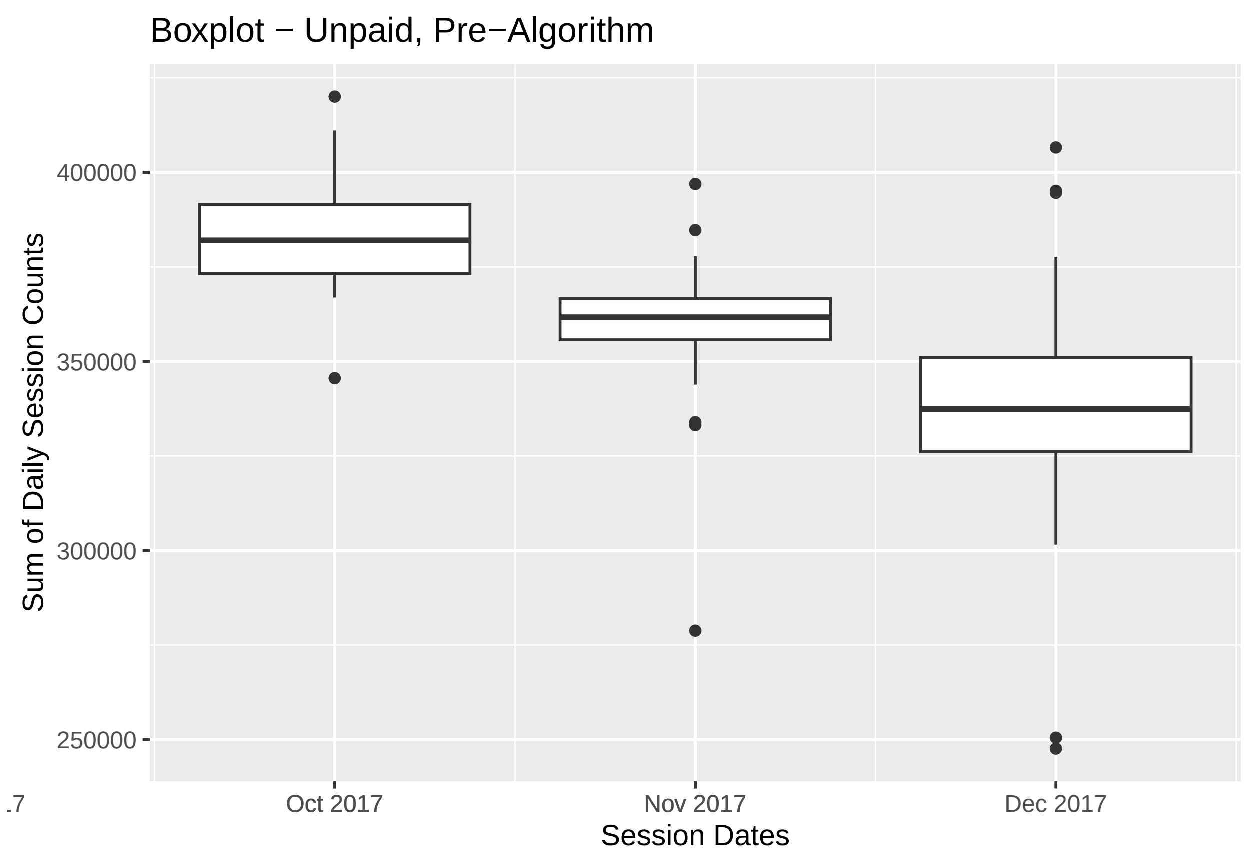
I am saving this for later, what a interesting image.

Finding Pre-2018 Outliers

```r
# Find the outliers pre-2018 ---------
unpaid_daily_goog_pre_18<-traffic_data  %>%
  filter(is_paid_traffic ==FALSE) %>%
  filter(search_engine =='google') %>%
  #filter(date >'2018-01-01')%>%
  filter(date <'2018-01-01')%>%
  group_by(date) %>%
  mutate(session_count_sum=sum(sesssion_count))

#select specfic columns for display
unpaid_daily_goog_pre_18_short<-unpaid_daily_goog_pre_18 %>%
  select(date,session_count_sum) %>%
  distinct()

#clear up date/month
unpaid_daily_goog_pre_18_short$year_month<-as.yearmon(ymd(unpaid_daily_goog_pre_18_short$date), "%Y %m"
```

Here, I filter the data for those before 2018, to remove the outliers present. But first, I am examining the data from this time.

```r
# Find the outliers pre-2018 ---------
#create a boxplot to explore doistribution of session counts by months
unpaid_daily_goog_pre_18_boxplot<-unpaid_daily_goog_pre_18_short %>%
  ggplot(aes(x = year_month, y = session_count_sum)) +
  geom_boxplot(aes(group=year_month)) +
  ggtitle("Boxplot - Unpaid, Pre-Algorithm") +
  xlab("Session Dates") + ylab("Sum of Daily Session Counts")

unpaid_daily_goog_pre_18_boxplot
```



This shows that the session counts have slowed down towards the end of the year. Also, you can see that these outlier points in December skew the ranges, and need to be removed.

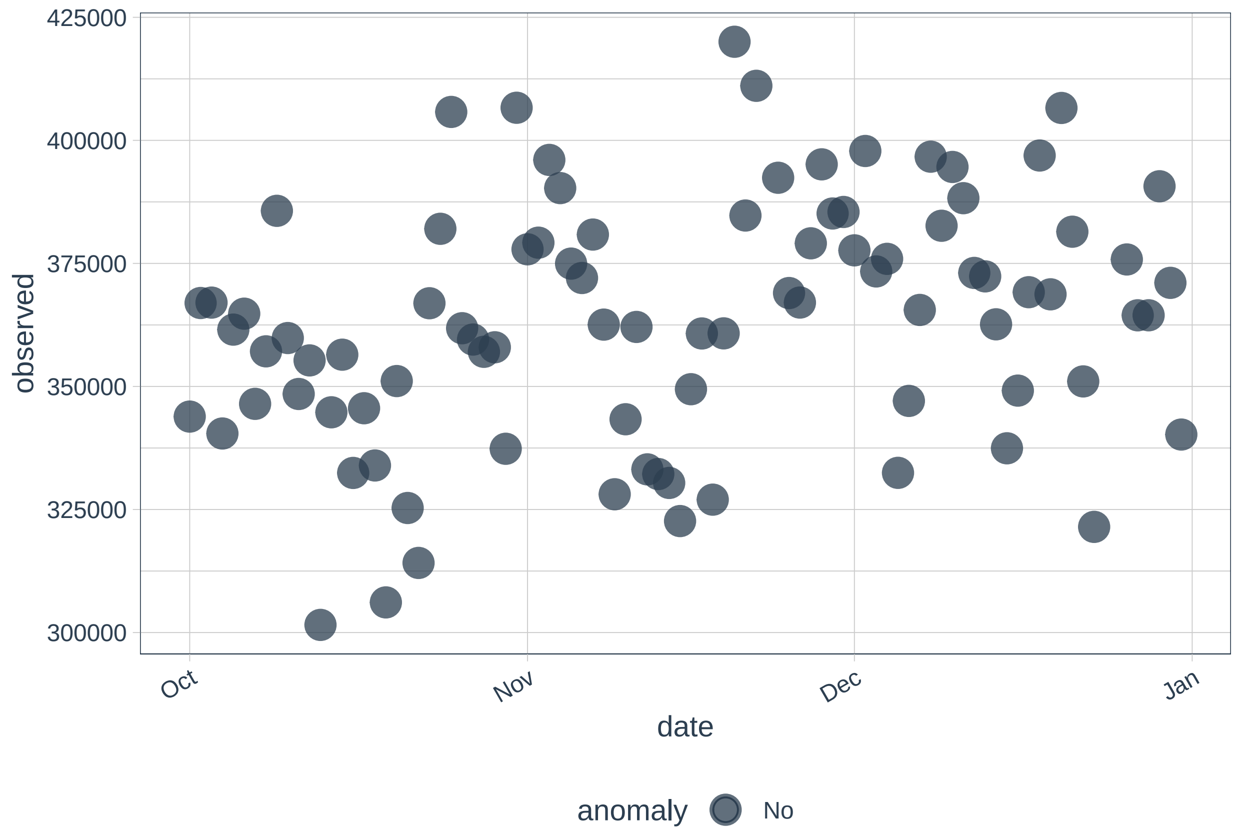
```
#save results
ggsave("unpaid_daily_goog_pre_18_boxplot.png", dpi = 300, height = 9, width = 12, unit = 'in')
```

Saving the boxplot output!

Remove The Pre-2018 Outliers

Here I am removing the outliers, then verifying that the remaining data pre-2018 contains no anomalies.

```
#plot session counts before the anomalies
pre_algo_goog_clean<-df_anomalized %>% plot_anomalies(ncol = 3, alpha_dots = 0.75,
                                                      size_dots=5)
pre_algo_goog_clean
```



```r
ggsave("pre_algo_goog_clean.png", dpi = 300, height = 9, width = 12, unit = 'in')
```

Looks good! There are no more anomalies left pre-2018. Saving this image.

```r
# Look at all data again no anomalies---------

#remove original days
df_unpaid_daily_goog_short_no_outliers<- df_unpaid_daily_goog_short[!df_unpaid_daily_goog_short$date %i

#still has outliers, run again
df <- as_tibble(df_unpaid_daily_goog_short_no_outliers)


df_anomalized <- df %>%
  time_decompose(session_count_sum, merge = TRUE) %>%
  anomalize(remainder) %>%
  time_recompose()
```

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

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

```
## trend = 89.5 days
```

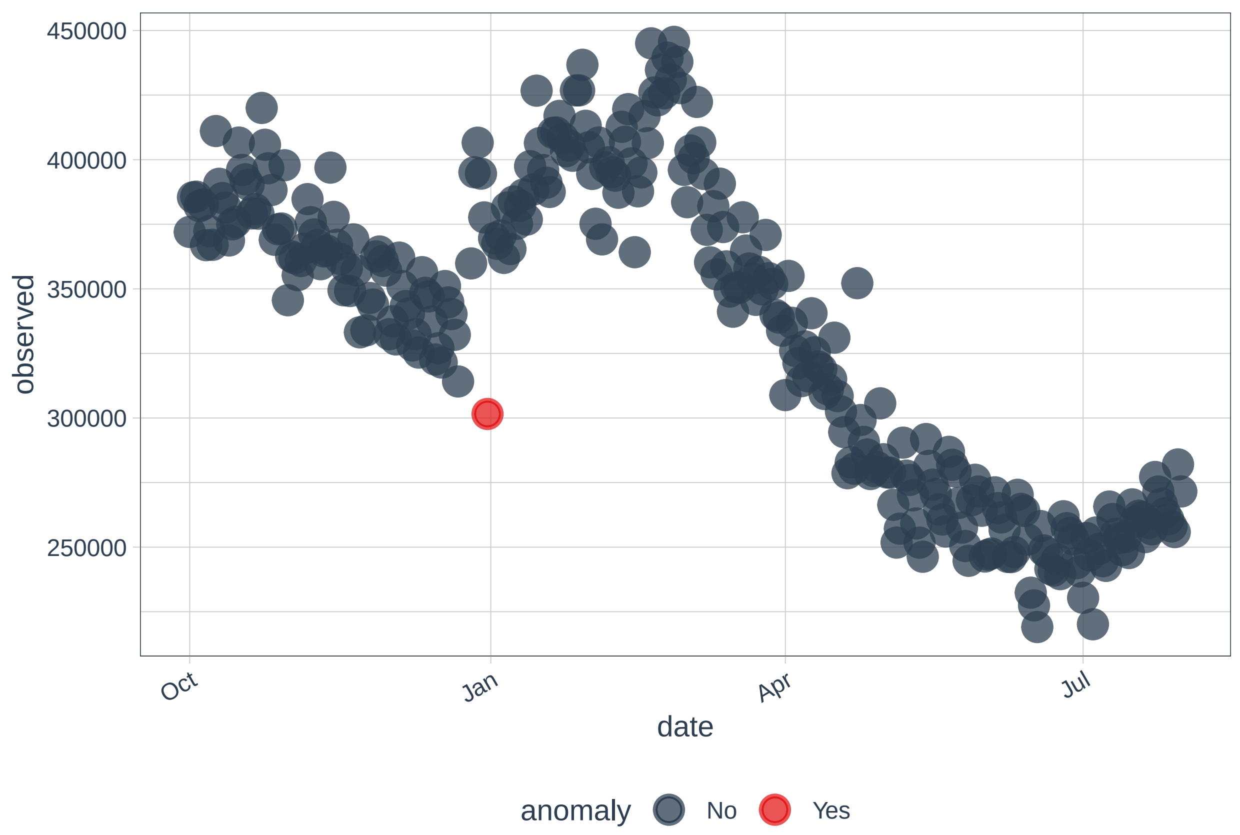
This process might have to be repeated to take out additional anomalies.

```
dates_to_remove<-df_anomalized[df_anomalized$anomaly=='Yes',]
df_unpaid_daily_goog_short_no_outliers<- df_unpaid_daily_goog_short_no_outliers[!df_unpaid_daily_goog_sh

df <- as_tibble(df_unpaid_daily_goog_short_no_outliers)


df_anomalized <- df %>%
  time_decompose(session_count_sum, merge = TRUE) %>%
  anomalize(remainder) %>%
  time_recompose()
```

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

## frequency = 7 days

## trend = 88.5 days
```

We see the weekly frequency represented here again.

```
#look into the results

pre_algo_non_goog_anomaly<-df_anomalized %>% plot_anomalies(ncol = 3, alpha_dots = 0.75,
                                                          size_dots=5)
pre_algo_non_goog_anomaly
```



This anomaly accounts for Christmas day, where there would not be that many people online. This date should be excluded, and we can keep searching for the anomaly date.

```
dates_to_remove<-df_anomalized[df_anomalized$anomaly=='Yes',]
df_unpaid_daily_goog_short_no_outliers<- df_unpaid_daily_goog_short_no_outliers[!df_unpaid_daily_goog_sh
df <- as_tibble(df_unpaid_daily_goog_short_no_outliers)

df_anomalized <- df %>%
  time_decompose(session_count_sum, merge = TRUE) %>%
  anomalize(remainder) %>%
  time_recompose()
```

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

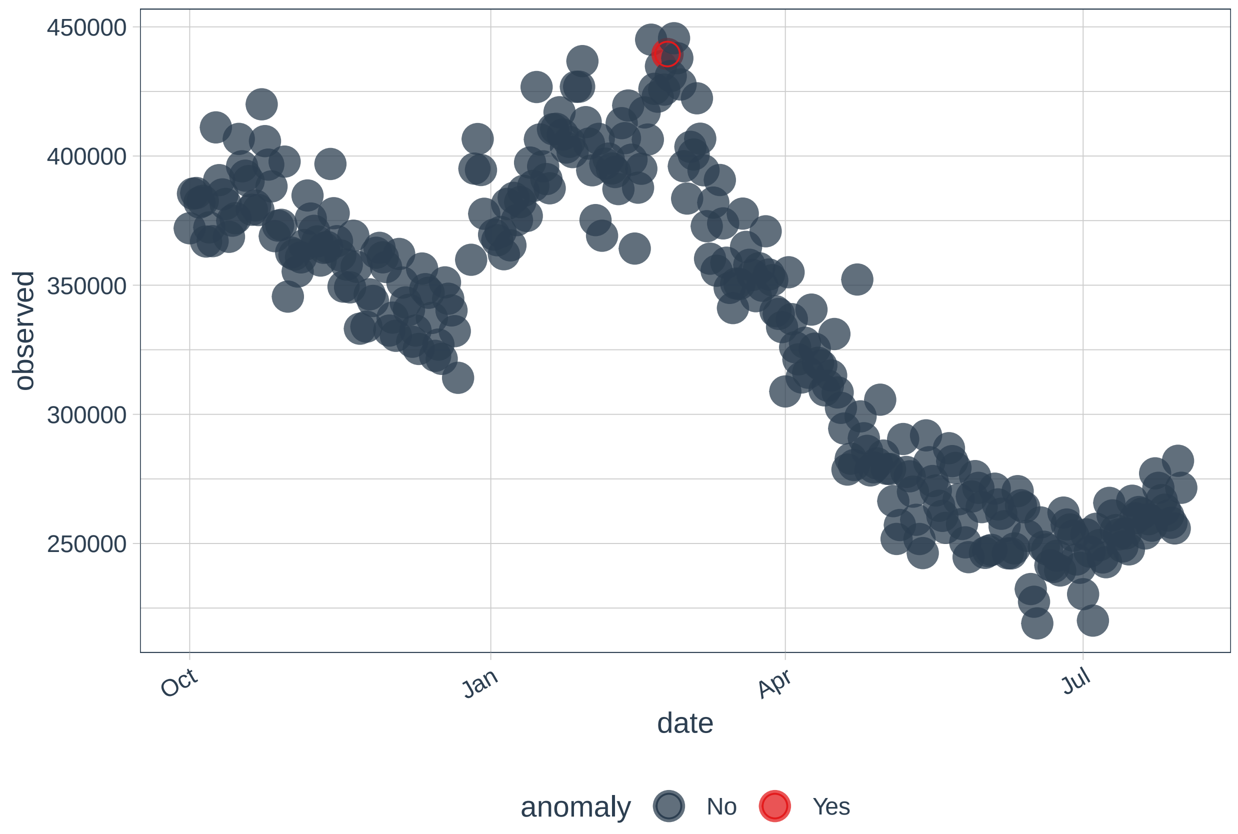
```
## frequency = 7 days
```

```
## trend = 88 days
```

Here I am removing Christmas day from the data.

Finding the Anomaly Date

```r
algorithm_identification<-df_anomalized %>% plot_anomalies(ncol = 3, alpha_dots = 0.75,
                                                           size_dots=5)
df_anomalized_identified<-df_anomalized
algorithm_identification
```



```r
ggsave("algorithm_date.png", dpi = 300, height = 9, width = 12, unit = 'in')
```

Here! We can see this point as the possible date where the algorithm change could have occured. Let's see what day it is.

```r
####finding real anomaly
algorithm_date<-df_anomalized[df_anomalized$anomaly=='Yes',]
usable_dates<-df_anomalized[df_anomalized$anomaly=='No',]

algorithm_date
```

```
## # A time tibble: 1 x 11
## # Index:         date
##   date       session_count_sum observed season   trend remainder remainder_l1
##   <date>                 <int>    <dbl>  <dbl>   <dbl>     <dbl>        <dbl>
## 1 2018-02-24            439493   439493 -2649. 386128.    56015.      -56115.
## # i 4 more variables: remainder_l2 <dbl>, anomaly <chr>, recomposed_l1 <dbl>,
## #   recomposed_l2 <dbl>
```

Anomaly date was Feb 24 2018. We can see that after this date, the traffic searches decreased and never went back up. In the future, we can estimate how things would have progressed if there had been no Google algorithm change.