Nielsen - Facebook page analysis

Archit Gupta

20 September 2016

Contents

Facebook is not only used by individuals but also by many businesses. Most of the businesses create a Facebook page to advertise about their company and to display their products, offers, and related content. Hence, it becomes really important to check what kinds of posts are liked by the user, perform experimentation with different content, and identify the content that results in a follower's engagement.

Let's see how to get the contents of a page. First, we need to get the name of the page. We can get this from the URL of the page(https://www.facebook.com/nielsen). Let's take Nielsen as an example.

Unlike Facebook friends' data, it is not necessary to like the page in order to pull the data. We can pull the data from any page provided the page, as well as the content, is public. The code is as follows:

```
## Loading required package: httr
```

```
## Loading required package: rjson
```

```
## Loading required package: httpuv
```

```
##
## Attaching package: 'Rfacebook'
```

```
## The following object is masked from 'package:methods':
##
##     getGroup
```

```
# Facebook Page data
page<- getPage("nielsen", token, n = 500, reactions = TRUE)
```

```
## 25 posts 50 posts 75 posts 100 posts 125 posts 150 posts 175 posts 200 posts 225 posts 250 posts 275
```

```
head(page, n=2)
```

```
##                              id likes_count     from_id from_name
## 4 71760418978_10153390014048979          30 71760418978   Nielsen
## 2 71760418978_10153389636278979           9 71760418978   Nielsen Make sure you<U+0092>re getting co
##   shares_count love_count haha_count wow_count sad_count angry_count
## 4           11          0          0         0         0           0
## 2            0          0          0         0         0           0
```

The success of a Facebook post is usually measured by the number of people who liked it and the number of people who commented about it, as it tells us how much engagement was triggered by the post as well as the number of times it has been shared by different people on their Facebook profile. All these details can be obtained from the preceding function. With Version 1.0 of the Facebook Graph API, we were able to search for a post with a keyword, but that feature has been deprecated with Version 2.0. Hence, now we need to know the ID or the name of the post if we want to get the details of a particular post. The commands are as follows:

```
#Let's get the detail about the post which had the maximum number of likes.
# Page with maximum likes
page[which.max(page$likes_count), ]
```

```
##                                 id likes_count     from_id from_name
## 271 71760418978_10153991682823979         429 71760418978   Nielsen "This acquisition significantly
```

Out of the 100 posts that we have pulled from the nielsen page, we got the post that had the maximum number of likes. From the output, we see that the most popular post had about 210 likes:

Now, let's see what kind of details are captured and what kind of analysis can be performed that would help us to measure the impact.

Trending topics

The concept of trending topics is quite popular. We can see the trending topics in news websites, Twitter, and so on. But how can we identify the trending topic for a particular Facebook page or a group of Facebook pages? Let's see how it can be done in detail.

Trend analysis

We will see how to learn which posts are doing well in recent times. After selecting the page that we are planning to do some analysis for, we will filter the posts' data based on a time range. Let's consider the same **nielsen** page and filter the recent data and see the posts that were popular:

```
# Most trending posts
page<- getPage("nielsen", token, n = 500)
```

```
head(page, n=2)
```

```
##                               id likes_count     from_id from_name
## 4 71760418978_10153390014048979          30 71760418978   Nielsen
## 2 71760418978_10153389636278979           9 71760418978   Nielsen Make sure you<U+0092>re getting co
##   shares_count love_count haha_count wow_count sad_count angry_count
## 4           11          0          0         0         0           0
## 2            0          0          0         0         0           0
```

We pull the interactions, that is, messages posted in a page using the getPage function. In the following code, we are filtering the data. We are pulling the data that was posted after April 1, 2015. Then, we order the post based on the number of likes, and we use the head function to display the top posts and their details. The code is as follows:

```
pageRecent<- page[which(page$created_time> "2015-04-01"), ]
top<- pageRecent[order(- pageRecent$likes),]
head(top, n=5)
```

```
##                                 id likes_count     from_id from_name
## 271 71760418978_10153991682823979         429 71760418978   Nielsen
## 373 71760418978_10154255132393979         373 71760418978   Nielsen                          Thank yo
## 11  71760418978_10153396289538979         297 71760418978   Nielsen
## 237 71760418978_10153913686868979         290 71760418978   Nielsen We<U+0092>re blushing! We<U+0092
## 24  71760418978_10153420369493979         282 71760418978   Nielsen
##
## 271                                                                                        http://n
## 373                                                                                       http://nl
```

```
## 11              https://www.facebook.com/nielsen/photos/a.409857393978.187232.71760418978/1015339628
## 237 http://sites.nielsen.com/newscenter/nielsen-awarded-top-service-provider-by-consumer-goods-techn
## 24                                                    https://storify.com/Nielsen/2015-nielsen-gl
```

In this case, we are considering the number of likes for the post as a proxy for a good post. By using the minus symbol, we are sorting the post in descending order. Hence, the top post will come at the top.

In order to check the trend for the top posts, we will obtain the preceding details on a daily basis and combine them using the function rbind, and finally, filter the posts based on the post ID as well as the date to check the trend for the posts. This will be helpful to know if there has been any spike in the popularity of a post.

If we increase the granularity of the analysis by repeating the preceding exercise more frequently, say every hour. We would be able to see the patterns of interaction. For example, we would be able to see the time when the interactions of the followers are highest.

Influencers

Having seen the details of the post, let's see how to learn about the people who comment and like these posts and to check if there is anyone who is more influential. For doing such an analysis, first we need to pull the data about the user interaction in a particular post.

Based on a single post

Let's take the most recent post and pull all the user comments using the function getPost. For each of those comments, let's see how many people liked it using the following code:

```
post_id<- head(page$id, n = 1) ## ID of most recent post
post<- getPost(post_id
               , token
               , n = 1000
               , likes = TRUE
               , comments = TRUE)
head(post$comments, n=2)
```

```
##             from_id             from_name
## 1 10152263367671749 Nicholas P. Schiavone Why Nielsen TV Ratings Are Inaccurate & Unreliable, \nPlus
```

```
samppost<- post$comments
```

The preceding command will copy all the comments of a particular post. After that, in order to check the user who had the maximum likes, we can write a query using the function sqldf. First, we need to import the package sqldf and write a query using sqldf. Since we are interested in only those users who had the most number of likes, we are pulling only the username and the number of likes for the various comments by the user. Then, we will arrange them in descending order to know the top users of this post in the nielsen page.

```
library(sqldf)
comments <- post$comments
influentialusers<- sqldf("select from_name, sum(likes_count) as totlikes
                         from comments
                         group by from_name")
head(influentialusers)
```

```
##               from_name totlikes
## 1 Nicholas P. Schiavone        0
```

```
influentialusers$totlikes<- as.numeric(influentialusers$totlikes)

# Sorting the users based on the number of likes they received
top<- influentialusers[order(- influentialusers$totlikes),]
head(top, n=10)
```

```
##               from_name totlikes
## 1 Nicholas P. Schiavone        0
```

Based on multiple posts

Based on a single post or a few comments, can we come to a conclusion on if a particular user is influential or not? In order to identify the overall favorite user, we will first download the comments posted in all the posts. Combine them using the rbind function, and finally write a SQL query using the sqldf function with groupby"user name".

For the Facebook page nielsen, let's check the most influential person. We will first download the top 100 posts from the page. Convert them into a matrix so that it becomes easy to access the comments based on their position, and initialize the data frame allcomments as null. The code is as follows:

```
post_id<- head(page$id, n = 100)
head(post_id, n=10)
```

```
##  [1] "71760418978_10153390014048979" "71760418978_10153389636278979" "71760418978_10153389805123979"
```

```
post_id<- as.matrix(post_id)
allcomments<- ""
```

In the for loop, we will traverse post by post and append all the comments of the posts to the data frame allcomments. The following for loop might take some time because we are consolidating thousands of comments. Finally, we will sort the users based on the number of likes they got for their comments. Hence, we get to know the most influential person in the page. The code is as follows:

```
# Collecting all the commments from all the 100 posts
for (i in 1:nrow(post_id))
{
# Get upto 1000 comments for each post
post<- getPost(post_id[i,], token, n = 1000,likes = TRUE, comments = TRUE)
comments<- post$comments
# Append the comments to a single data frame
allcomments<- rbind(allcomments, comments)
}
```

Once we have consolidated all the comments, we use the sqldf function to aggregate the likes based on user. To know how many users have commented in the posts and in total how many liked their comments, the code is as follows:

```
# Consolidating the like for each user.
influentialusers<- sqldf("select from_name, sum(likes_count) as totlikes
                         from allcomments
                         group by from_name")
influentialusers$totlikes<- as.numeric(influentialusers$totlikes)
top<- influentialusers[order(- influentialusers$totlikes),]
head(top, n=20)
```

```
##                         from_name totlikes
## 4                     Ana G Molina        1
```

```
## 7                    Ashraf Sadek        1
## 9                       Blanca SD        1
## 17                 Faby MartÃnez        1
## 19       Habiba Magroun Ep Yazidi        1
## 20                  Heather Barrow        1
## 22                  High Risk Hope        1
## 24                   HÆ°Æ¡ng Giang        1
## 36                     Nara Campos        1
## 1                                         0
## 2  Adeparua Olabode Colin Carrick        0
## 3                 Albert Chambliss        0
## 5                  Armando Estevez        0
## 6                     Ashish Vijay        0
## 8                   Beatriz Burgos        0
## 10                 Brett Nicholson        0
## 11     Carolina Fuentealba Burgos        0
## 12                Carolyn Broadbelt        0
## 13                  Cathy Dougherty        0
## 14                     Celso Cepeda        0
```

In the same way, we can do this for a group of related pages. Getting to know the most influential person is a very useful task, that is very helpful for executing marketing campaigns and making them successful.

Measuring CTR performance for a page

The performance of a page can be measured by the user activity in the page and the user's interaction in the posts published in the page. Let's measure the performance for a page. When we say measuring the performance, it is by means of counting the user interaction through likes, comments, and shares.

In order to come up with a trend, we will need the timestamp data. Then, we need to consolidate the data on a monthly basis so that we can draw the time-series performance chart.

First, we need to convert the Facebook date format into R-supported date format. The following code is a function to convert the date format. We need to pass the date timestamp data to this function. The function will return the same date timestamp in R-supported format so that we can perform date operation. The code is as follows:

```
format.facebook.date<- function(datestring) {
  date<- as.POSIXct(datestring,
  format = "%Y-%m-%dT%H:%M:%S+0000", tz = "GMT")
}
```

Then, we need to aggregate the data on a monthly basis. The aggregate.metric function will aggregate the required data on a monthly basis. We will pass the likes, comments, and shares count data to this function. We would have already converted the date to the required format using the previous function. The code is as follows:

```
aggregate.metric<- function(metric) {
  m<- aggregate(page[[paste0(metric, "_count")]]
                , list(month = page$month)
                , mean)
  m$month<- as.Date(paste0(m$month, "-15"))
  m$metric<- metric
  return(m)
}
```

Then, let's see how our R code makes use of the preceding functions and to plot the performance trend. Finally, we use the ggplot function to plot the trend and save the plot using the function ggsave. The quality of image can be adjusted using the dpi parameter.

Use the getPage function to extract all the posts from the Facebook page nielsen The number of pages to be retrieved can be altered using the parameter n. In this case, we are downloading the top 500 posts. The command is as follows:

```
page<- getPage("nielsen", token, n = 500)
```

```
## 25 posts 50 posts 75 posts 100 posts 125 posts 150 posts 175 posts 200 posts 225 posts 250 posts 275
```

We are passing the date timestamp to the function format.facebook.date created by us. So it will be converted to a format that would be supported by the aggregate function which will be used next. The code is as follows:

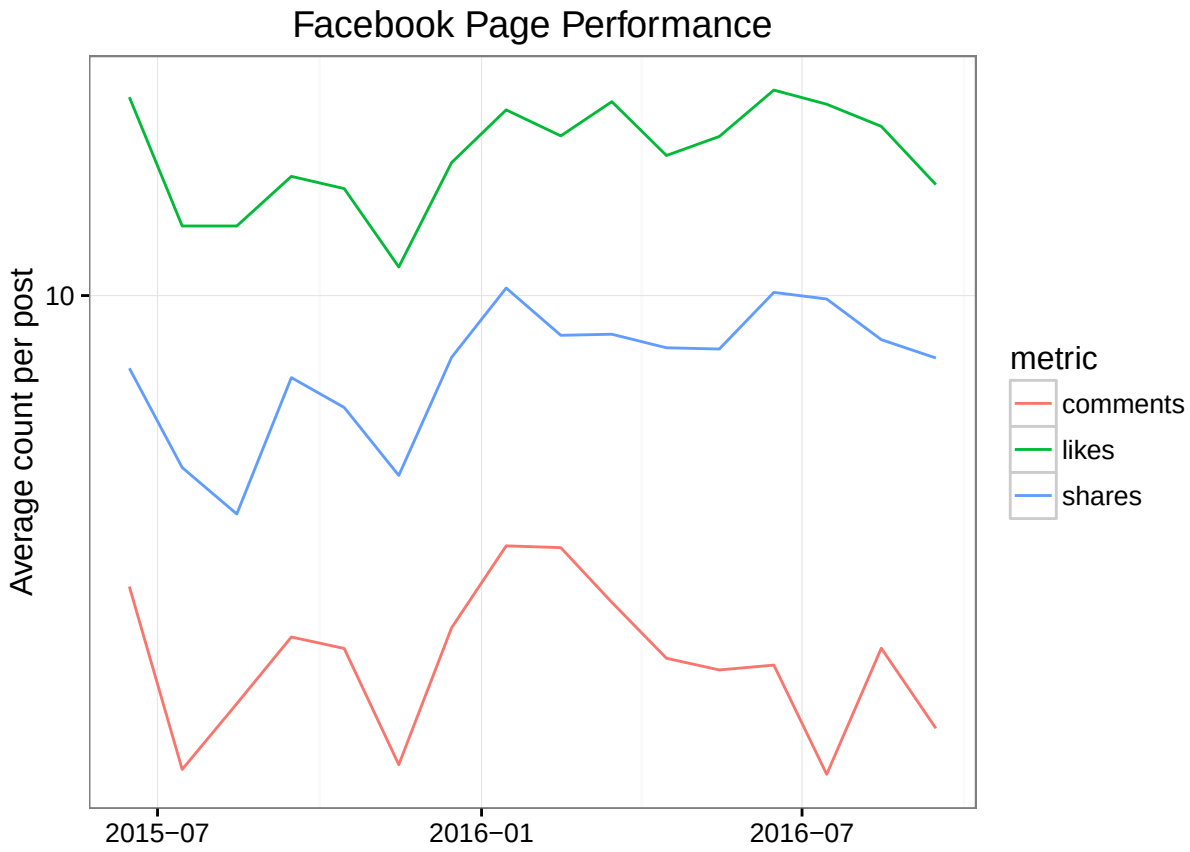
```
page$datetime<- format.facebook.date(page$created_time)
page$month<- format(page$datetime, "%Y-%m")
```

We are using the aggregate function to aggregate the number of likes, comments, and shares on a monthly basis. The code is as follows:

```
df.list<- lapply(c("likes", "comments", "shares")
                 ,aggregate.metric)
df<- do.call(rbind, df.list)
```

We need the ggplot2 and scales packages in order to make the plot. Hence, we need to load those packages using the library function. We plot the graph using the ggplot function and define a few parameters to make the graph readable. The code is as follows:

```
library(ggplot2)
library(scales)
ggplot(df, aes(month, x, group = metric)) +
  geom_line(aes(color = metric)) +
  scale_x_date(breaks = waiver()) +
  scale_y_log10("Average count per post", breaks =c(10, 100, 1000, 10000, 50000)) +
  theme_bw() +
  theme(axis.title.x = element_blank()) +
  ggtitle("Facebook Page Performance")
```




```
ggsave(file="C:/Users/GuptAr04/Desktop/3973-format-trend.png",
dpi=500)
```

Hence, we can now see the performance trend chart. The same data can be used to plot the chart on an hourly, quarterly, or yearly basis with little modification to the existing code. Similarly, different metrics could be defined, computed, and the trend could be plotted.