Experiment No. 1

Objective: Create a blog and link it with Google Analytics using R scripts.

1. Open blogger --https://www.blogger.com/about/?r=1-null_user
2. Create blog
3. http://hostel453.blogspot.com/
4. https://analytics.google.com
5. Create new account in https
6. Generate track id UA-121837656-1 automatically generated
7. gtack.js
8. Go to blogger- setting-others
9. https://www.blogger.com/blogger.g?blogID=3433789405044560733#othersettings
10. Enter track id
11. Now go to blogger home - https://analytics.google.com/analytics/web/#/report-home/a121837656w179817195p178062548
12. Go to real time
13. https://console.developers.google.com/apis/dashboard
14. Developer.facebook.com
15. Dev.twitter.com
16. Enable analytics api
17. Ooth protocols
18. Add credentials to your project
20. Find out what kind of credentials you need
21. Calling Analytics API from a web server
22. Create an OAuth 2.0 client ID
23. Created OAuth client 'Web client 1'
24. Set up the OAuth 2.0 consent screen

Summary of Steps

1. Install R
2. Install R Studio
3. Save the R Script
4. install.packages(c("googleAnalyticsR","ggplot2"))
5. Authorize googleAnalyticsR to access your data
6. Find your Google Analytics viewID
7. Set your date range dynamically or statically
8. Run page view query
9. View the page view query data View(df1)
10. Run sessions by date query
11. View the sessions data View(df2)
12. Create a line graph of Google Analytics sessions using ggplot2
13. Create a bar graph of Google Analytics sessions by month by year using ggplot2

Google Analytics using R Scripts

1. Download and Install R

R is a system for statistical computation and graphics. It provides, among other things, a programming language, high level graphics, interfaces to other languages and debugging facilities. Download and install R for Windows, Mac and Linux here: https://cran.r-project.org/

2. Download and Install R Studio

R Studio is where the magic happens. It is an IDE (Interactive Development Environment). This is the software user interface that you'll be working in and where you'll run your scripts. Download R studio here: https://www.rstudio.com/products/rstudio/download/

3. Save the R Script

Launch R Studio and in the top menu go to File > New File > New R Script. This will open a blank window in the top left pane of R Studio. Copy and paste the code below into the blank window. Save the R Script.

```
library(googleAnalyticsR)

library(ggplot2)

Authorize Google Analytics R- this will open a webpage

#You must be logged into your Google Analytics account on your web browser

ga_auth()

#Use the Google Analytics Management API to see a list of Google Analytics accounts you have
access to

  my_accounts <- google_analytics_account_list()

  #set date variables for dynamic date
  range

  View(my_accounts)

  #Use my_accounts to find the viewId. Make sure to replace this with your viewId.

  my_id <- 94579701

  start_date <- "60daysAgo"

  end_date <- "yesterday"

  #Page View Query

  df1 <- google_analytics_4(my_id, date_range = c("2016-12-10", "2017-02-07"), metrics =
  c("pageviews"), dimensions = c("pagePath"))

  #Session Query - Uses start_date and end_date
```

```
df2 <- google_analytics_4(my_id, date_range = c(start_date, end_date), metrics = c("sessions"),
dimensions = c("date"))

#graph sessions by date

ggplot(data=df2, aes(x=date, y=sessions)) + geom_line(stat="identity")
```

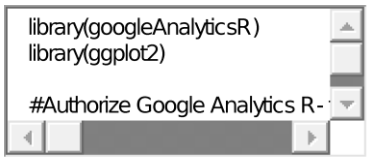
4. Install googleAnalyticsR and ggplot2 R packages

Packages can be thought of as add-on to R that make it easier to perform your specific task. Much like an Excel plug-in like Analytics Edge or the Google Analytics Sheets add-on. We will be using googleAnalyticsR to connect with and pull Google Analytics data and ggplot2 to visualize the Google Analytics data. In R Studio in the bottom left console paste the code shown below and press enter on your keyboard to install the two packages. Be patient as this may take some time.

```
> install.packages(c("googleAnalyticsR","ggplot2"))
```

5. Authorize Google Analytics R to access your account data

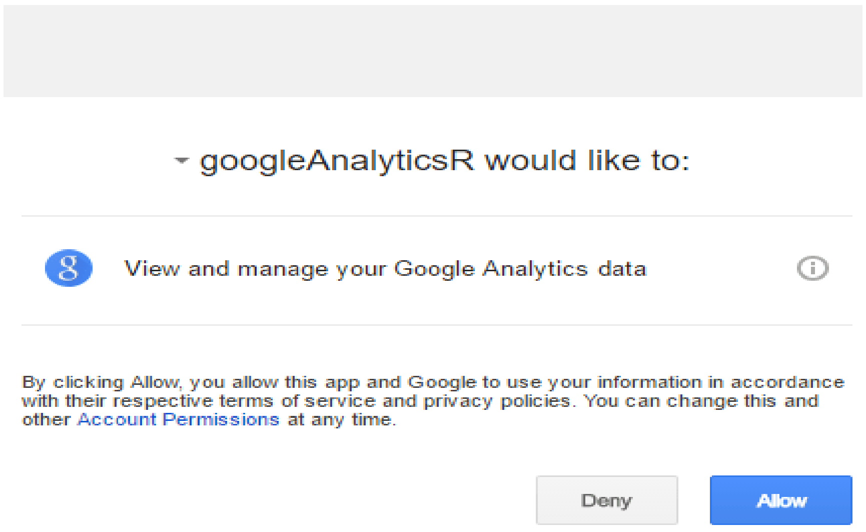
Highlight the code shown below and click run in the upper right corner of the top left pane called the source pane.



```
library(googleAnalyticsR)
library(ggplot2)

#Authorize Google Analytics R-
```

```
library(googleAnalyticsR)

library(ggplot2)



#Authorize Google Analytics R- this will open a webpage

#You must be logged into your Google Analytics account on your web
browser

ga_auth()
```

ga_auth() will open your default web browser to a window that will ask you to allow googleAnalyticsR to read your data. Make sure you are logged into the Google Account with access to the Google Analytics data you'd like to query. See the *16 second mark of the video* to watch how this authentication process work. Once you've authenticated, googleAnalyticsR will remember and you won't have to reauthenticate on subsequent queries.

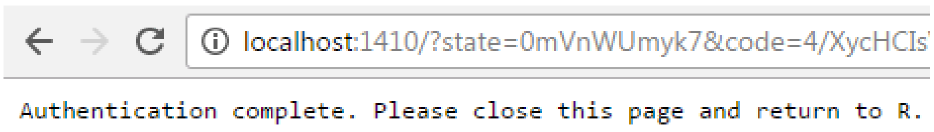


googleAnalyticsR would like to:

View and manage your Google Analytics data

By clicking Allow, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.

googleAnalyticsR requesting authorization in your web browser

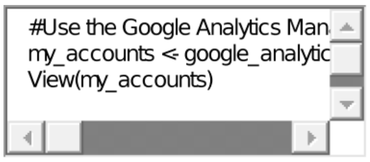


```
localhost:1410/?state=0mVnWUmyk7&code=4/XycHCIs
Authentication complete. Please close this page and return to R.
```

successful authentication

6. Find your Google Analytics viewID

Highlight the code shown below and click run in the source pane.



```
#Use the Google Analytics Man
my_accounts <- google_analytic
View(my_accounts)
```

#Use the Google Analytics Management API to see a list of Google Analytics accounts you have access to

my_accounts <- google_analytics_account_list()

View(my_accounts)

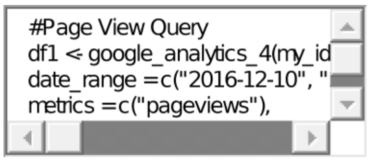
This will open the my_accounts dataframe where you will see a table of all the Google Analytics accounts with the corresponding view and viewId that you have access to. Find the the Google Analytics view that you'd like to pull data on and copy the viewId. Paste the numeric viewId on line 13 for my_id replacing my viewId 94579701. Note you can also find your viewId in the Google Analytics admin section of the web user interface.

7. Set your query date range dynamically or statically

Set the date range for your Google Analytics R queries. In the example on rows 16 and 17 I've used dynamic dates to set start_date and end date. You can use dynamic dates like yesterday and 60daysAgo or you can use static date like 2017-01-01 YYYY-MM-DD format.

8. Run the page views query

Highlight the code shown below and click run in the source pane. This will run your page view by pagePath query. The queries use Google Analytics Reporting API v4.



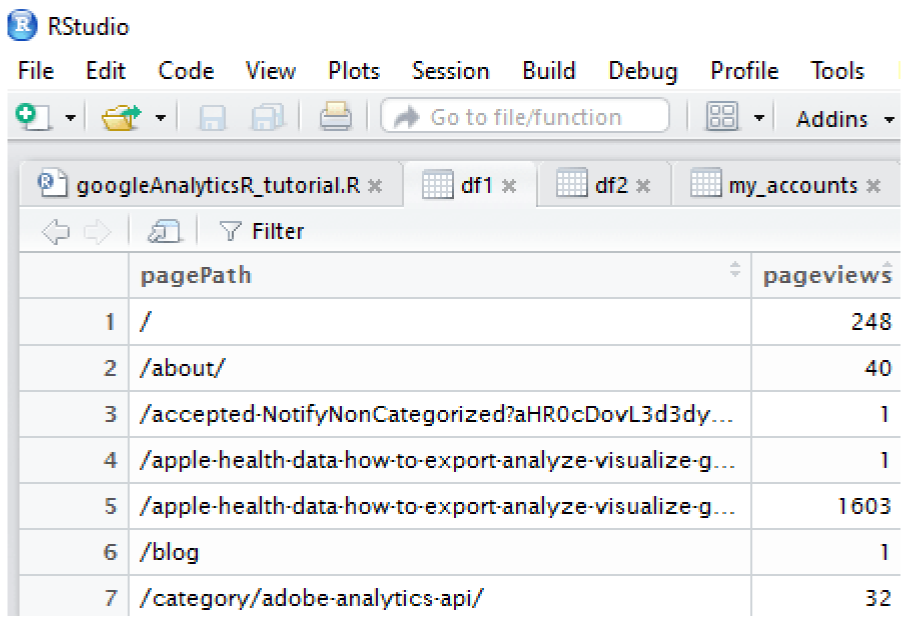
```
#Page View Query
df1 <- google_analytics_4(my_id
date_range =c("2016-12-10", "
metrics =c("pageviews"),
```

```
#Page View Query

df1 <- google_analytics_4(my_id,

date_range = c("2016-12-10", "2017-02-07"),

metrics = c("pageviews"), dimensions = ("pagePath"))
```

Note that I've use static dates for the date_range and I've chosen pageviews for the metric and pagePath for the dimension.

9. View the page view data

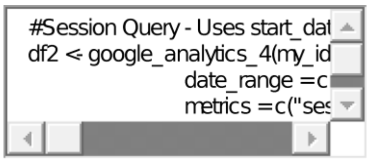
To view the page view data in a data frame you can find df1 in the top right pane under the Environment Tab. Click on df1 and this will open the data frame as a data table in the top left Source pane.


RStudio
File Edit Code View Plots Session Build Debug Profile Tools
Go to file/function
Addins
googleAnalyticsR_tutorial.R
df1
df2
my_accounts
Filter

	pagePath	pageviews
1	/	248
2	/about/	40
3	/accepted-NotifyNonCategorized?aHR0cDovL3d3dy...	1
4	/apple-health-data-how-to-export-analyze-visualize-g...	1
5	/apple-health-data-how-to-export-analyze-visualize-g...	1603
6	/blog	1
7	/category/adobe-analytics-api/	32

10. Run the session query

Highlight the code shown below and click run in the source pane. This will run your sessions by date query.



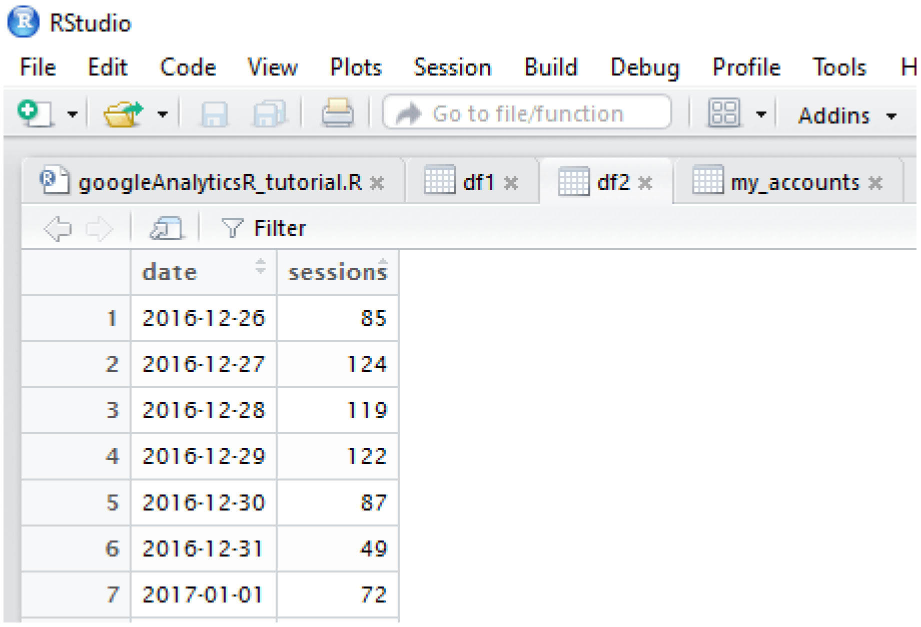
```
#Session Query - Uses start_dat
df2 <- google_analytics_4(my_id
                   date_range = c
                   metrics = c("ses
```

#Session Query - Uses start_date and end_date

df2 <- google_analytics_4(my_id, date_range = c(start_date, end_date),metrics = c("sessions"), dimensions = c("date"))

11. View the sessions data

To view the sessions data in a data frame you can find df2 in the top right pane under the Environment Tab. Click on df2 and this will open the data frame as a data table in the top left source pane. *See the 35 second mark of the video* to view this step.

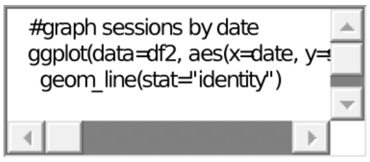


RStudio
File Edit Code View Plots Session Build Debug Profile Tools H
Go to file/function
Addins
googleAnalyticsR_tutorial.R
df1
df2
my_accounts
Filter

	date	sessions
1	2016-12-26	85
2	2016-12-27	124
3	2016-12-28	119
4	2016-12-29	122
5	2016-12-30	87
6	2016-12-31	49
7	2017-01-01	72

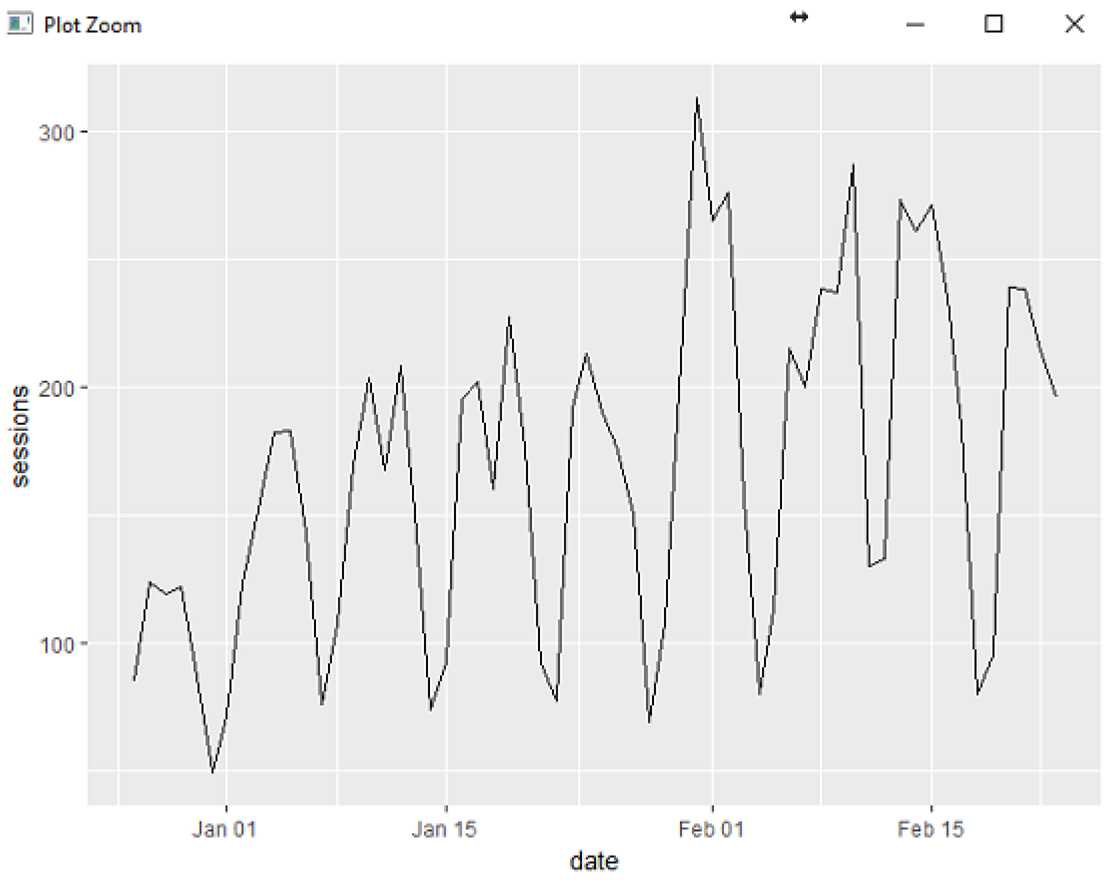
12. Create a line graph using ggplot2

Highlight the code shown below and click run in the source pane. This will create a line graph showing session by date in the bottom right pane under the Plots tab.



```
#graph sessions by date
ggplot(data=df2, aes(x=date, y=
  geom_line(stat="identity")
```

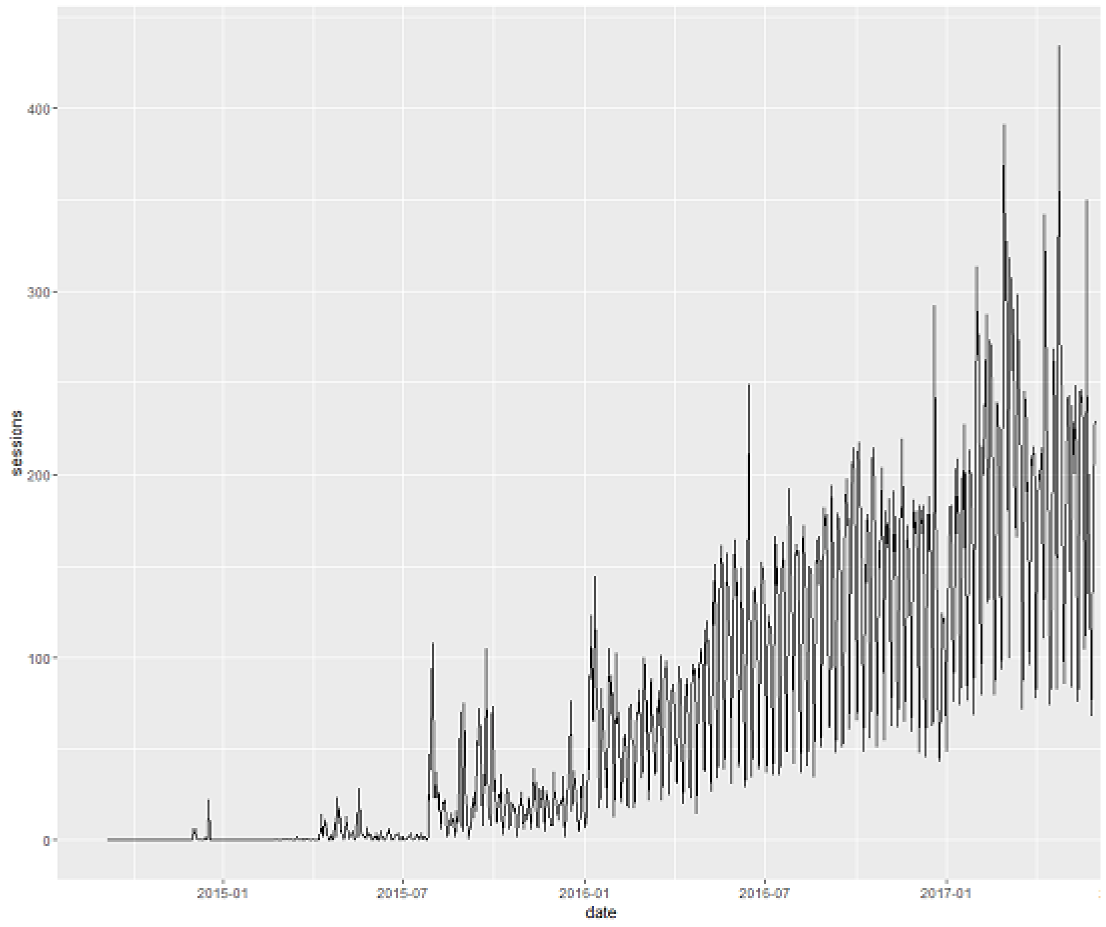
```
#graph sessions by date

ggplot(data=df2, aes(x=date, y=sessions)) + geom_line(stat="identity"
```



13. Create a bar graph of Google Analytics session by month by year using ggplot2

This is a bonus step. A reader asked me how to graph session data by month instead of session by date using ggplot2. He sent me a graph like the one below. Which you can see shows the trend over time but is impossible to answer questions like: Are my Google Analytics sessions up for this month year over year? The steps below will show how to answer this question using a bar graph data visualization in R.

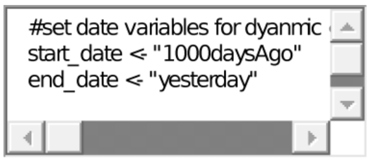
To create the daily line graph over a long period of time (in this case Google Analytics sessions data for the last 1000 days) go to row 16 of the R code and set start_date to 1000daysAgo instead of set 60daysAgo then rerun the sessions query shown in step 8. Then rerun the graph sessions by day code shown in step 12.



Add the code below the ###New Code Starts here### line to your R script and run it to create a bar graph showing Google Analytics sessions data by month by year like the graph below.

Let's walk through the script and what it is doing. First you need to add a month and year column to your dataframe using the date column from your query. Then you need to include the dplyr package to group the data by month and by year using the new columns.

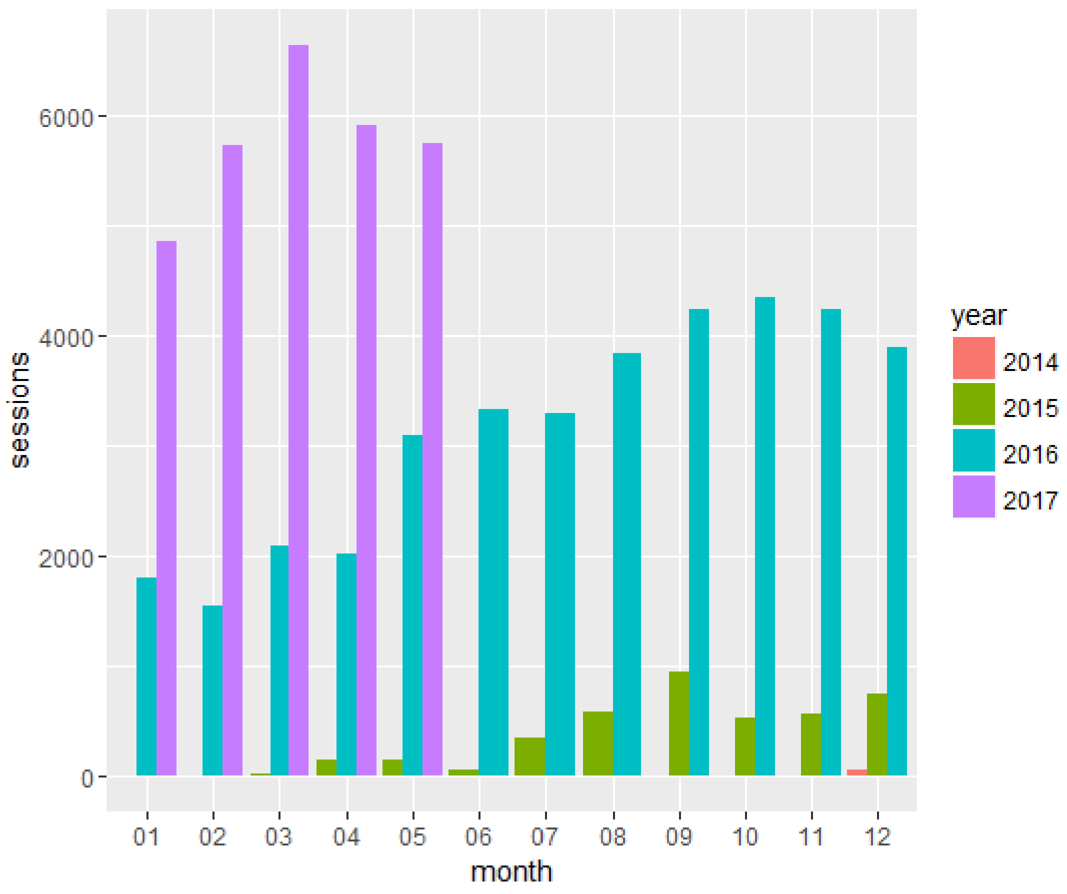
The code starting on line 43 below takes our dataframe df2 and groups the data by year and month. Then it sums the yearly and monthly sessions data. The print(n=100) on row 47 will print out the yearly and monthly aggregated session data to the R console. Then the ggplot2 portion of the code starting on line 49 will create a bar graph with months on the x axis and session on the y axis and fill of year creating a different color bar graph for each year.



```
#set date variables for dyanmic
start_date <- "1000daysAgo"
end_date <- "yesterday"
```

```
#set date variables for dyanmic date range

start_date <- "1000daysAgo"

end_date <- "yesterday"
```

```
#Session Query - Uses start_date and end_date

df2 <- google_analytics_4(my_id, date_range = c(start_date, end_date), metrics = c("sessions") dimensions =
c("date"))
```



Specific Software / Hardware used (if any): R Studio/Rscripts, Google Analytics

Experiment No. 2

Objective: Facebook metric analysis using Facebook Insights.

Facebook Page Insights vs. Facebook Audience Insights

Facebook Page Insights and Facebook Audience Insights are both Facebook analytics tools. There is a little bit of overlap between the two of them, but they serve very different purposes. The key to how to use each is in the name of the tool:

- **Facebook Page Insights** gives you detailed analytics for your *Facebook Page*, so you can track what works, learn how people interact with your content, and improve your results over time.
- **Facebook Audience Insights** helps you understand your *Facebook audience* so you can better target ads and create more relevant content.

In this post, we walk you through how to use Facebook Page Insights. Looking for information on how to track and analyze your Facebook audience? We've got a whole blog post about how to set up and use Facebook Audience Insights to get you started.

How to use Facebook analytics

First things first. To access Facebook Page Insights, go to your Facebook Page and click **Insights** in the top menu. If you don't see Insights in the menu, click **More** to bring it up.

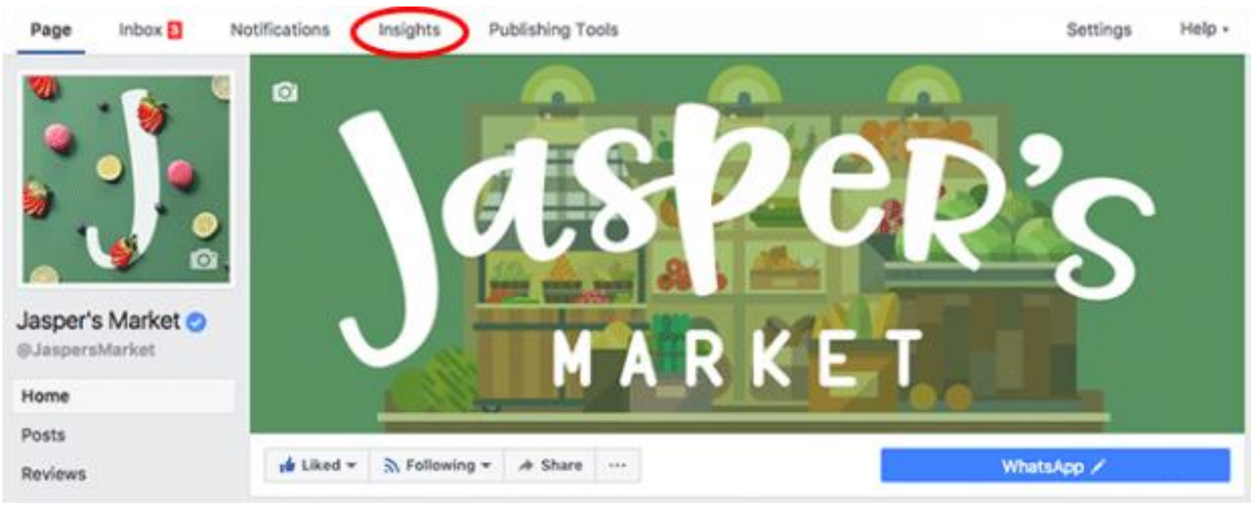


Image via Facebook

You'll be taken straight to your Overview, which you can access again at any time by clicking **Overview** in the left-hand menu.

Overview: The big picture

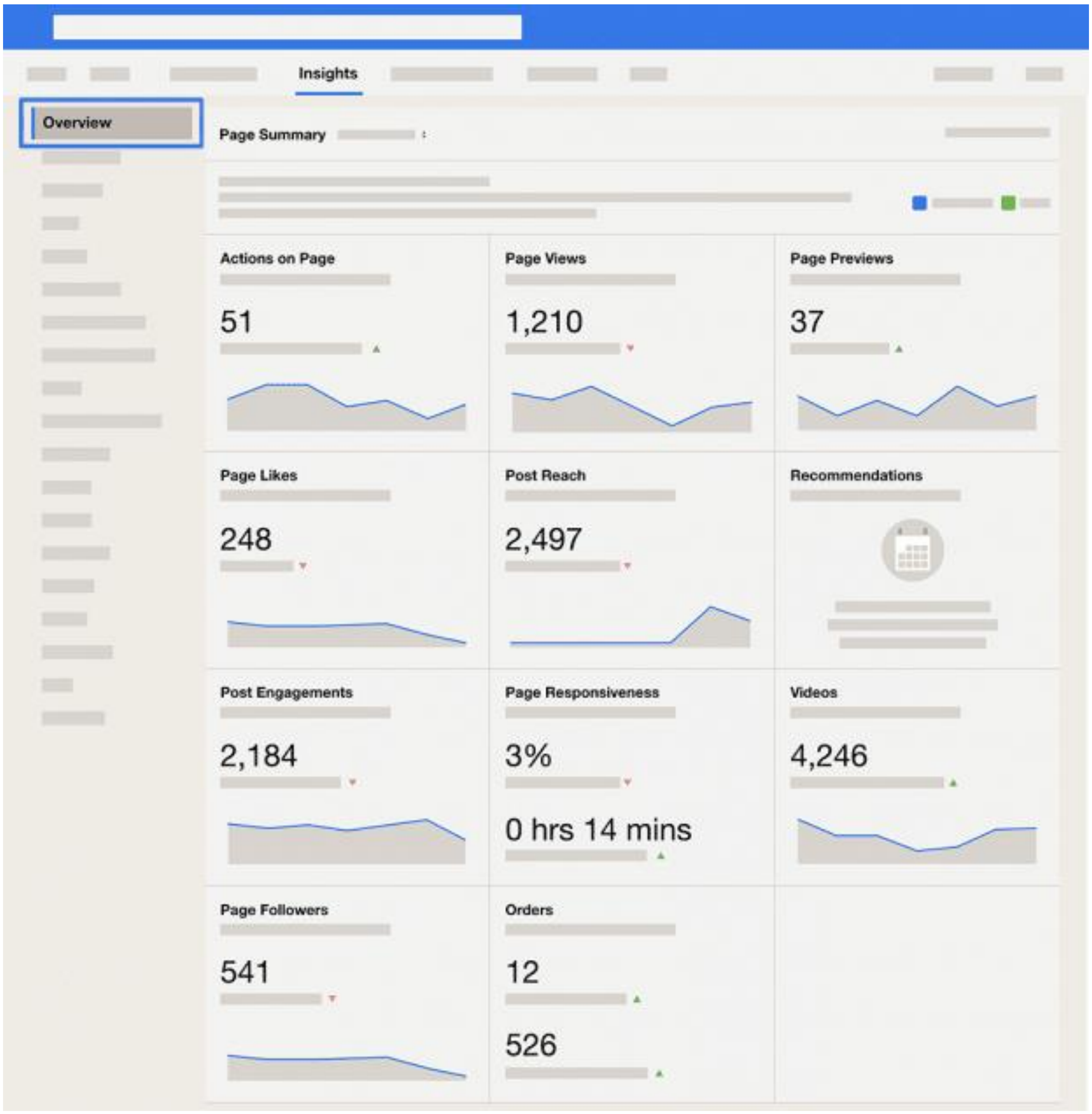


Image via [Facebook]

The Overview gives you a bird's-eye view of everything that's happening with your Facebook Page. You can choose to view data for the last seven or 28 days, for the current day, or for the previous day.

The Overview is broken down into three sections, starting with the **Page Summary.** Here, you'll see a set of graphs with top-level metrics for a number of categories:

- **Actions on Page:** The combined total clicks for your contact information and call-to-action button
- **Page views:** Total views of your Facebook Page, including by people not logged into Facebook.
- **Page Previews:** The number of times people hovered their mouse over your Page information to see a preview of your Page.

- **Page Likes:** The number of new likes.
- **Post reach:** The number of people who saw your posts in their timeline.
- **Story reach:** The number of people who saw your Stories.
- **Recommendations:** The number of people who recommended your Page.
- **Post engagement:** A combined total of post likes, comments, shares, and other engagements.
- **Responsiveness:** An evaluation of how often and how fast you respond to messages.
- **Videos:** The number of video views of three seconds of more.
- **Page followers:** The number of new followers
- **Orders:** Your orders and earnings.

You can click on any of these charts to get more detailed information, or click on the corresponding item in the left-hand menu.

The second section is called **Your 5 Most Recent Posts**. This gives you the reach and engagement numbers for your latest posts. There's also an option to boost your best performing posts right from this screen.

Finally, you'll see a section called **Pages to Watch**. In this section, you can manually add in Pages you want to compare to your own. For example, you could add a competitor's page to see how you measure up. This can be a good way to benchmark your results against the major players in your industry.

Directly under the page summary, you'll see a box with the heading **Ad Results Have Moved.** All tracking for Facebook ads has moved to a separate ad analytics page called Ad Center. You can find it in the top navigation menu of your Facebook Page by clicking **More**.

Posts: Detailed analysis

The Posts section of your Facebook page analytics dashboard gives you tons of important information about your posts and the activity on your Page, divided into three tabs:

- When Your Fans Are Online
- Post Types
- Top Posts From Pages You Watch

It's a bit confusing, because only the information in the box at the top of the screen changes when you click on one of these tabs. The information in the section below this top box, called All Posts Published, stays the same.

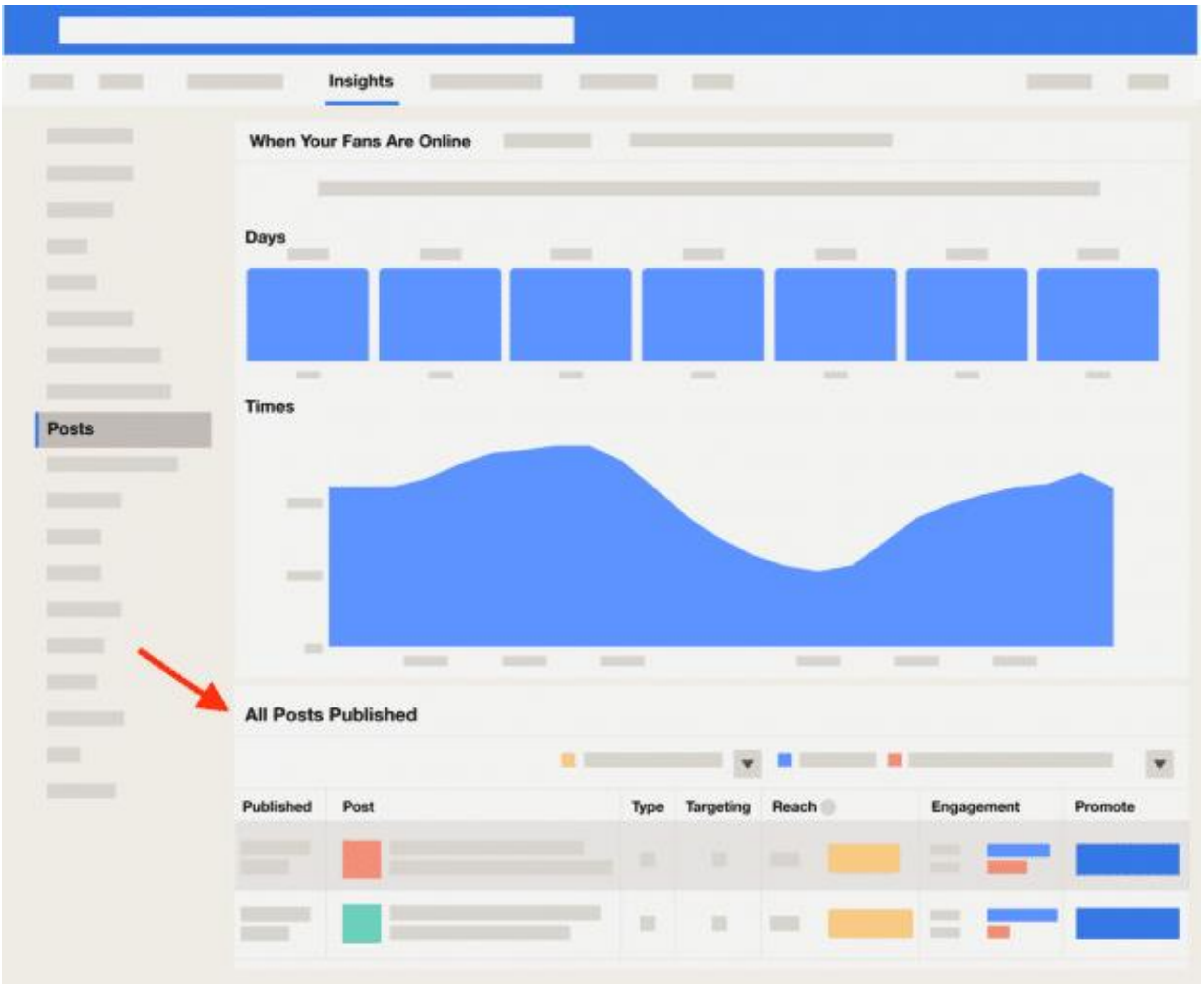


Image via [Facebook]

We'll start with the All Posts Published section before moving on to what you can learn from the individual tabs at the top of the page.

All posts published

Here, you can review the results for all the posts you've published on your Page. For each post, you can see:

- **Type:** Was it a link post? A photo post? A video? This column can help you understand which types of posts appeal most to your audience.
- **Targeting:** Was it a public post? Did you target a specific audience? When comparing results, make sure you take the targeted group into account.
- **Reach:** By default, this column shows the number of people who say your post. To see the breakdown of paid versus organic reach, hover your mouse over the yellow bar. At the top of the All Posts Published section, there's a drop-down menu that you can use to change what's displayed in this column. Your other options are to view impressions (organic vs. paid) or reach among fans vs. non-fans.
- **Engagement:** Here you see the number of clicks each post got, as well as the combined number of reactions, comments, and shares.

There's also an option to boost your posts right from this screen.

Experiment No. 3

Objective: Text Mining Twitter Data With TidyText in R

After completing this tutorial, you will be able to:

- Use the `tidytext` package in `R` to filter social media data by date.
- Use the `tidytext` package in `R` to text mine social media data.

What You Need

You will need a computer with internet access to complete this lesson.

Download Week 13 Data (~80 MB)

In the previous lesson you learned the basics of preparing social media data for analysis and using `tidytext` to analyze tweets. In this lesson you will learn to use `tidytext` to text mine tweets and filter them by date.

The structure of twitter data is complex. In this lesson you will only work with the text data of tweets even though there is much more information that you could analyze.

```
# json support
library(rjson)
library(jsonlite)

# plotting and pipes - tidyverse!
library(ggplot2)
library(dplyr)
library(tidyr)
# text mining library
library(tidytext)
library(tm)
# coupled words analysis
library(widyr)
# plotting packages
library(igraph)
library(ggraph)

options(stringsAsFactors = FALSE)
# create file path
json_file <- "~/Documents/earth-analytics/data/week-
13/boulder_flood_geolocated_tweets.json"
# import json file line by line to avoid syntax errors
# this takes a few seconds
boulder_flood_tweets <- stream_in(file(json_file))
## opening fileconnectionoldClass input connection.
##
 Found 500 records...
 Found 1000 records...
 Found 1500 records...
 Found 2000 records...
 Found 2500 records...
 Found 3000 records...
 Found 3500 records...
 Found 4000 records...
 Found 4500 records...
 Found 5000 records...
 Found 5500 records...
 Found 6000 records...
 Found 6500 records...
 Found 7000 records...
 Found 7500 records...
 Found 8000 records...
 Found 8500 records...
 Found 9000 records...
 Found 9500 records...
 Found 10000 records...
 Found 10500 records...
 Found 11000 records...
 Found 11500 records...
 Found 12000 records...
 Found 12500 records...
 Found 13000 records...
 Found 13500 records...
 Found 14000 records...
 Found 14500 records...
 Found 15000 records...
 Found 15500 records...
 Found 16000 records...
 Found 16500 records...
 Found 17000 records...
 Found 17500 records...
 Found 18000 records...
 Found 18500 records...
 Found 18821 records...
 Imported 18821 records. Simplifying...
## closing fileconnectionoldClass input connection.


## Found 18000 records...
## Found 18500 records...
## Found 18821 records...
## Imported 18821 records. Simplifying...
```

First, create a new dataframe that contains:

- The date
- The twitter handle (username)
- The tweet text

for each tweet in the imported json file.

```
# view the first 6 usernames
head(boulder_flood_tweets$user$screen_name)
## [1] "lilcakes3209"    "coloradowx"      "ChelseaHider"    "jpreyes"
## [5] "BoulderGreenSts" "RealMatSmith"
# create new df with the tweet text & usernames
tweet_data <- data.frame(date_time = boulder_flood_tweets$created_at,
                         username = boulder_flood_tweets$user$screen_name,
                         tweet_text = boulder_flood_tweets$text)
head(tweet_data)
##                        date_time        username
## 1 Tue Dec 31 07:14:22 +0000 2013    lilcakes3209
## 2 Tue Dec 31 18:49:31 +0000 2013      coloradowx
## 3 Mon Dec 30 20:29:20 +0000 2013    ChelseaHider
## 4 Mon Dec 30 23:02:29 +0000 2013         jpreyes
## 5 Wed Jan 01 06:12:15 +0000 2014 BoulderGreenSts
## 6 Mon Dec 30 21:05:27 +0000 2013    RealMatSmith
```

```
##
tweet_text
## 1                        Boom bitch get out the way! #drunk #islands
#girlsnight  #BJs #hookah #zephyrs #boulder #marines…
http://t.co/uYmu7c4o0x
## 2     @WeatherDude17 Not that revved up yet due to model inconsistency.
I'd say 0-2" w/ a decent chance of &gt;1" #snow #COwx #weather #Denver
## 3
Story of my life! \U0001f602 #boulder http://t.co/ZMfNKEl0xD
## 4 We're looking for the two who came to help a cyclist after a hit-and-
run at 30th/Baseline ~11pm Dec 23rd #Boulder #CO http://t.co/zyk3FkB4og
## 5                                                              Happy New
Year #Boulder !!!! What are some of your New Years resolutions this year?
## 6                                                  @simon_Says_so Nearly 60
degrees in #Boulder today. Great place to live. http://t.co/cvAcbpDQTC
```

Next, clean up the data so that you can work with it. According to the incident report, the Colorado flood officially started September 09 2013 and ended on the 24th.

Let’s filter the data to just that time period. To do this, you need to:

1. convert the date column to a R date / time field.
2. filter by the dates when the flood occured.

```
#format = "%Y-%m-%d %H:%M:%s"
# flood start date sept 13 - 24 (end of incident)
start_date <- as.POSIXct('2013-09-13 00:00:00')
end_date <- as.POSIXct('2013-09-24 00:00:00')

# cleanup
flood_tweets <- tweet_data %>%
  mutate(date_time = as.POSIXct(date_time, format = "%a %b %d %H:%M:%S
+0000 %Y")) %>%
  filter(date_time >= start_date & date_time <= end_date )

min(flood_tweets$date_time)
## [1] "2013-09-13 00:00:18 MDT"
max(flood_tweets$date_time)
## [1] "2013-09-23 23:52:53 MDT"
```

Explore Common Words

Next let’s explore the content of the tweets using some basic text mining approaches. Text mining refers to looking for patters in blocks of text.

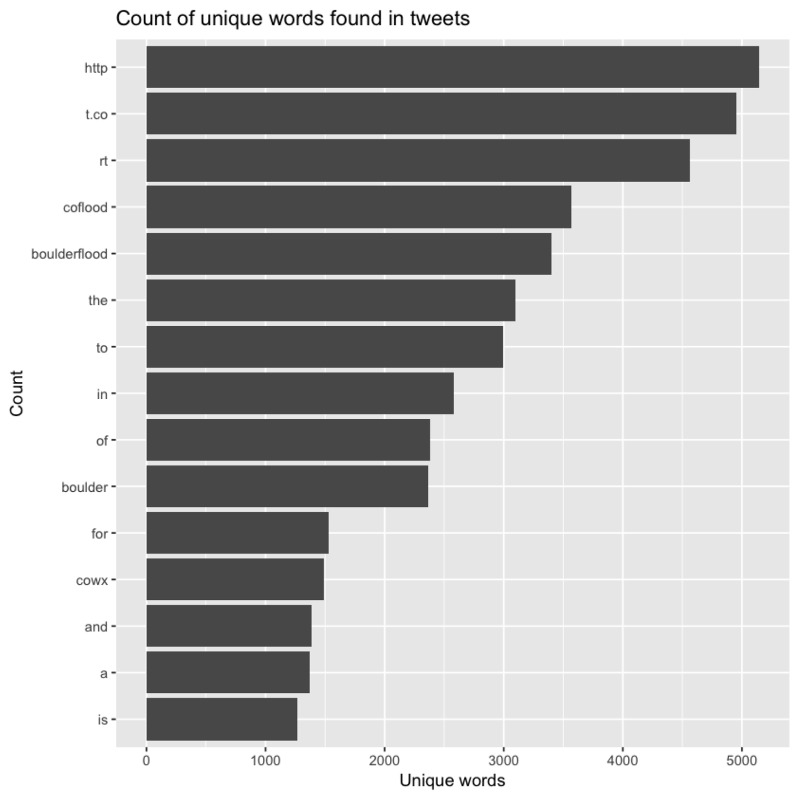
Generate a list of the most popular words found in the tweets during the flood event. To do this, you use pipes to:

1. select the `tweet_text` column.
2. stack the words so you can group and count them.

```
# get a list of words
flood_tweet_messages <- flood_tweets %>%
  dplyr::select(tweet_text) %>%
  unnest_tokens(word, tweet_text)

head(flood_tweet_messages)
##             word
## 1       download
## 1.1 centurylink
## 1.2   community
## 1.3       flood
## 1.4      impact
## 1.5      report
```

Next, plot the top 15 words found in the tweet text. What do you notice about the plot below?

```
# plot the top 15 words
flood_tweet_messages %>%
  count(word, sort = TRUE) %>%
  top_n(15) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(x = word, y = n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip() +
      labs(x = "Count",
      y = "Unique words",
      title = "Count of unique words found in tweets")
## Selecting by n
```



Remove Stop Words

The plot above contains "stop words". If you recall from the previous lesson, these are words like **and**, **in** and **of** that aren't useful to us in an analysis of commonly used words. Rather, you'd like to focus on analysis on words that describe the flood event itself.

As you did in the previous lesson, you can remove those words using the list of `stop_words` provided by the `tm` package.

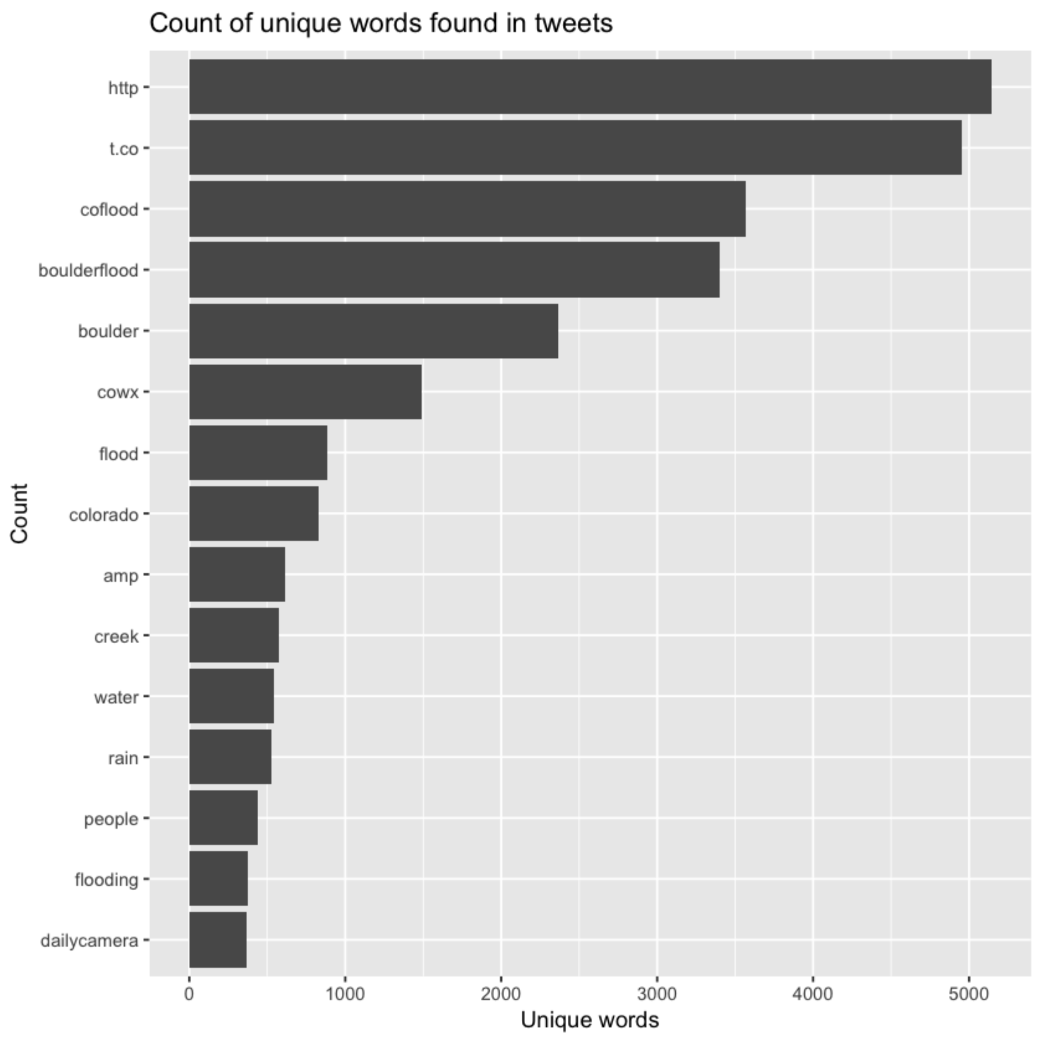
```
data("stop_words")
# how many words do you have including the stop words?
nrow(flood_tweet_messages)
## [1] 151536
```

```
flood_tweet_clean <- flood_tweet_messages %>%
  anti_join(stop_words) %>%
  filter(!word == "rt")
## Joining, by = "word"

# how many words after removing the stop words?
nrow(flood_tweet_clean)
## [1] 95740
```

Notice that before removing the stop words, you have 151536 rows or words in your data. After removing the stop words you have, 95740 words.

Once you’ve removed the stop words, you can plot the top 15 words again.

```
# plot the top 15 words -- notice any issues?
flood_tweet_clean %>%
  count(word, sort = TRUE) %>%
  top_n(15) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(x = word, y = n)) +



  geom_col() +
  xlab(NULL) +
  coord_flip() +
      labs(x = "Count",
      y = "Unique words",
      title = "Count of unique words found in tweets")
## Selecting by n
```



Finally, notice that http is the top word in the plot above. Let's remove all links from your data using a regular expression. Then you can recreate all of the cleanup that you performed above, using one pipe.

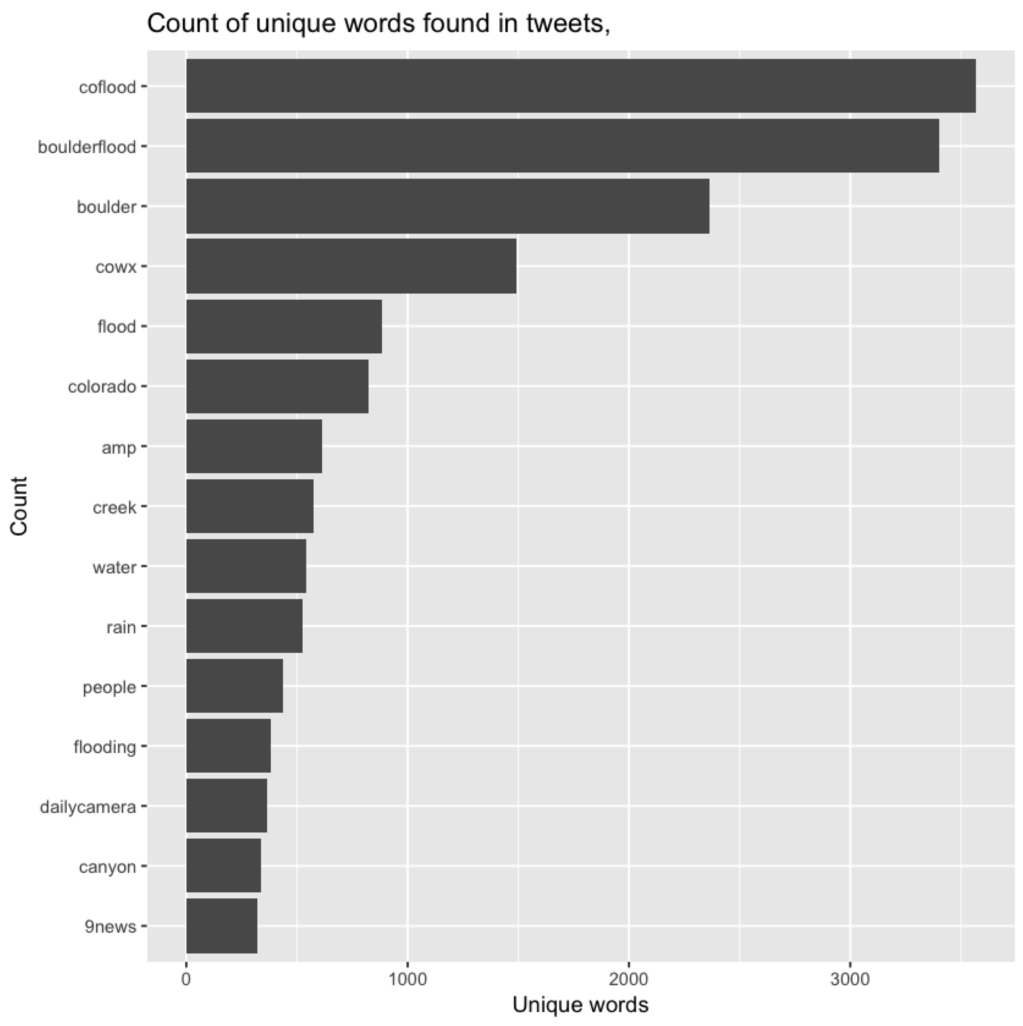
To remove all url's you can use the expression below.

```
gsub("\\s?(f|ht)(tp)(s?)(://)([^\\.]*)[\\.|/](\\S*)", "", tweet_text
```

```
# cleanup
flood_tweet_clean <- tweet_data %>%
  mutate(date_time = as.POSIXct(date_time, format = "%a %b %d %H:%M:%S
+0000 %Y"),
         tweet_text = gsub("\\s?(f|ht)(tp)(s?)(://)([^\\.]*)[\\.|/](\\S*)",
                           "", tweet_text)) %>%
  filter(date_time >= start_date & date_time <= end_date ) %>%
  dplyr::select(tweet_text) %>%
  unnest_tokens(word, tweet_text) %>%
  anti_join(stop_words) %>%
  filter(!word == "rt") # remove all rows that contain "rt" or retweet
```

```
## Joining, by = "word"
# plot the top 15 words -- notice any issues?
```

```
flood_tweet_clean %>%
  count(word, sort = TRUE) %>%
  top_n(15) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(x = word, y = n)) +
  geom_col() +
  xlab(NULL) +
  coord_flip() +
      labs(x = "Count",
      y = "Unique words",
      title = "Count of unique words found in tweets, ")
## Selecting by n
```



Paired Word Analysis

As you did in the previous text mining introductory lesson, you can do a paired words analysis to better understand which words are most often being used together.

Do perform this analysis you need to reclean and organization your data. In this case you will create a data frame with 2 columns which contain word pairs found in the data and a 3rd column that has the count of how many time that word pair is found in the data. Examples of word pairs includes:

- boulder flood
- colorado flood

To begin, clean up the data by removing stop words

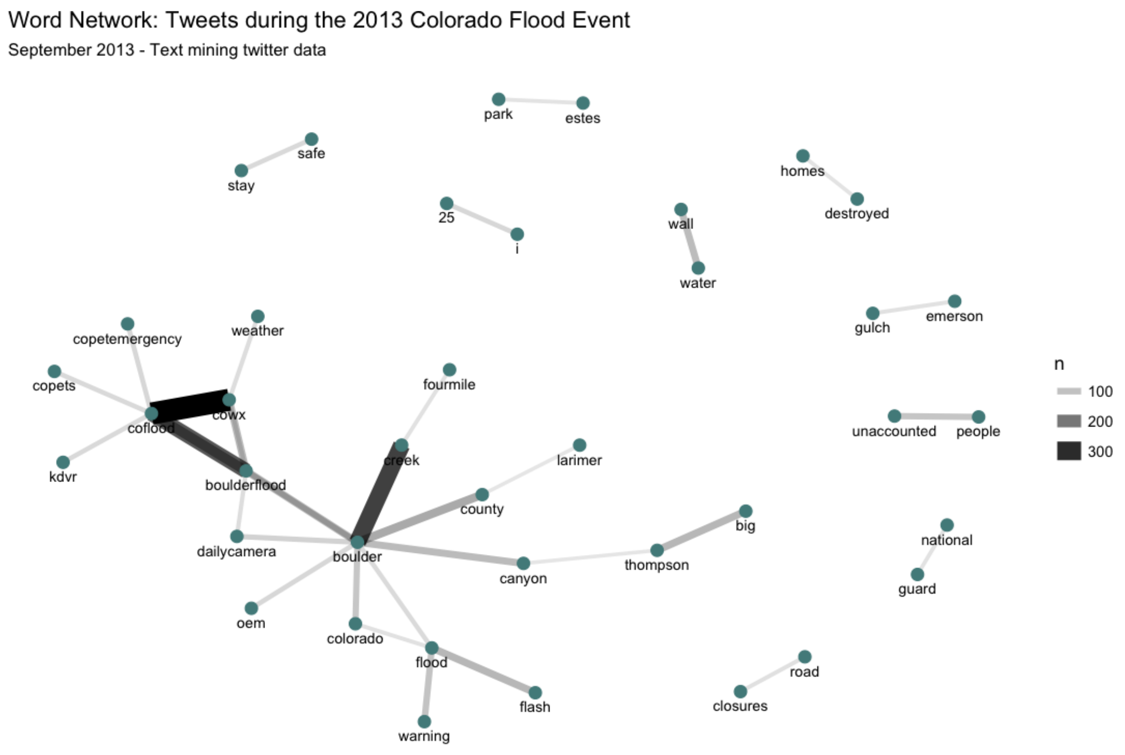
```
flood_tweets_paired <- flood_tweets %>%
  dplyr::select(tweet_text) %>%
  mutate(tweet_text = removeWords(tweet_text, stop_words$word)) %>%
  mutate(tweet_text = gsub("\\brt\\b|\\bRT\\b", "", tweet_text)) %>%
  mutate(tweet_text = gsub("http://*", "", tweet_text)) %>%
  unnest_tokens(paired_words, tweet_text, token = "ngrams", n = 2)

flood_tweets_paired %>%
  count(paired_words, sort = TRUE)
## # A tibble: 57,506 x 2
##             paired_words     n
##                    <chr> <int>
##  1         cowx coflood   358
##  2        boulder creek   273
##  3 boulderflood coflood   242
##  4         coflood cowx   159
##  5 coflood boulderflood   151
##  6       boulder county   130
##  7         big thompson   113
##  8 boulder boulderflood   113
##  9          flash flood   113
## 10    boulderflood cowx   110
## # ... with 57,496 more rows
```

Separate the words into columns and count the unique combinations of words.

```
flood_tweets_separated <- flood_tweets_paired %>%
  separate(paired_words, c("word1", "word2"), sep = " ")

# new bigram counts:
flood_word_counts <- flood_tweets_separated %>%
  count(word1, word2, sort = TRUE)
flood_word_counts
## # A tibble: 57,506 x 3
##            word1        word2     n
##            <chr>        <chr> <int>
##  1          cowx      coflood   358
##  2       boulder        creek   273
##  3 boulderflood      coflood   242
##  4       coflood         cowx   159
##  5       coflood boulderflood   151
##  6       boulder       county   130
##  7           big     thompson   113
##  8       boulder boulderflood   113
##  9         flash        flood   113
## 10 boulderflood         cowx   110
## # ... with 57,496 more rows
```

Finally, plot the word network.

```
# plot climate change word network
flood_word_counts %>%
        filter(n >= 50) %>%
        graph_from_data_frame() %>%
        ggraph(layout = "fr") +
        geom_edge_link(aes(edge_alpha = n, edge_width = n)) +
        geom_node_point(color = "darkslategray4", size = 3) +
        geom_node_text(aes(label = name), vjust = 1.8, size = 3) +
        labs(title = "Word Network: Tweets during the 2013 Colorado Flood
Event",
             subtitle = "September 2013 - Text mining twitter data ",



             x = "", y = "") +
        theme_void()
```



Specific Software / Hardware used (if any): R Studio

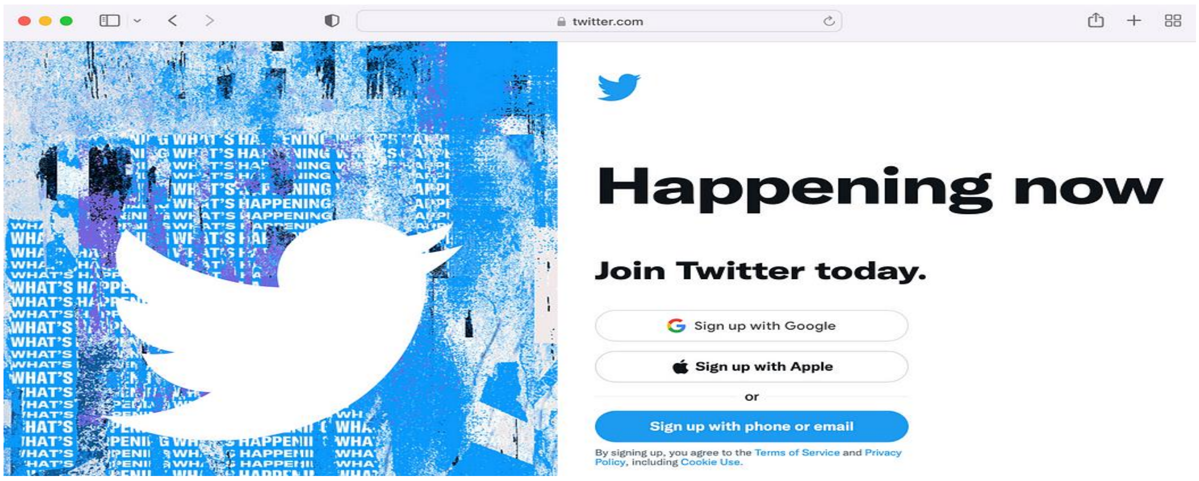
Experiment No. 4

Objective: Analyze and Download Twitter Data Using Tweepy.

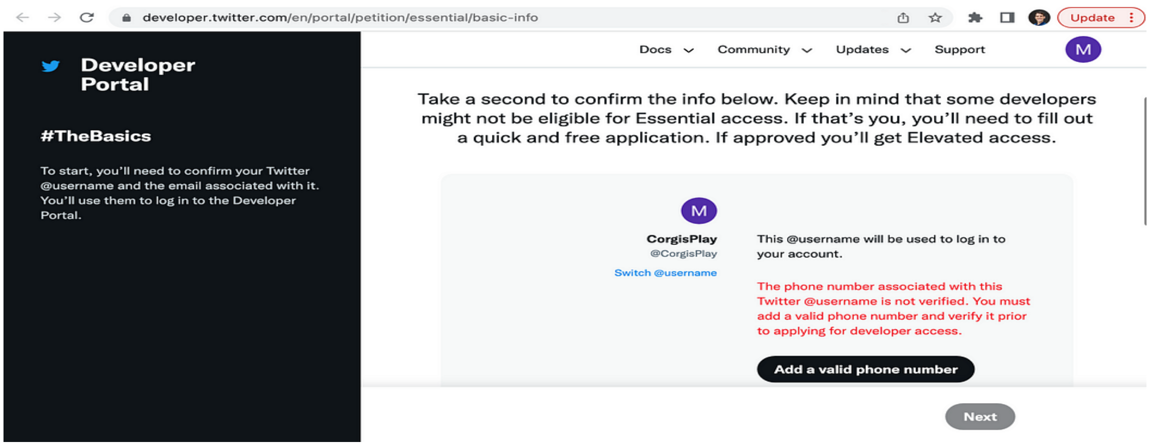
The Twitter API allows you to do many things including retrieve tweet data. In order to access this data, you need a developer account. Using the Twitter API should be an easy thing, but sometimes pictures and simple code can save you some frustration.

How to Setup a Twitter Developer Account

1.) Create a twitter account if you do not have one.


twitter.com
Happening now
Join Twitter today.
Sign up with Google
Sign up with Apple
or
Sign up with phone or email
By signing up, you agree to the Terms of Service and Privacy Policy, including Cookie Use.

2.) On the twitter developer account page, you will be asked to answer a few questions. For example, I was asked for a phone number, country, and use case. The next step is to read and agree to the developer agreement.


developer.twitter.com/en/portal/petition/essential/basic-info
Update
Developer Portal
Docs
Community
Updates
Support
M
Next

#TheBasics

To start, you'll need to confirm your Twitter @username and the email associated with it. You'll use them to log in to the Developer Portal.

Take a second to confirm the info below. Keep in mind that some developers might not be eligible for Essential access. If that's you, you'll need to fill out a quick and free application. If approved you'll get Elevated access.

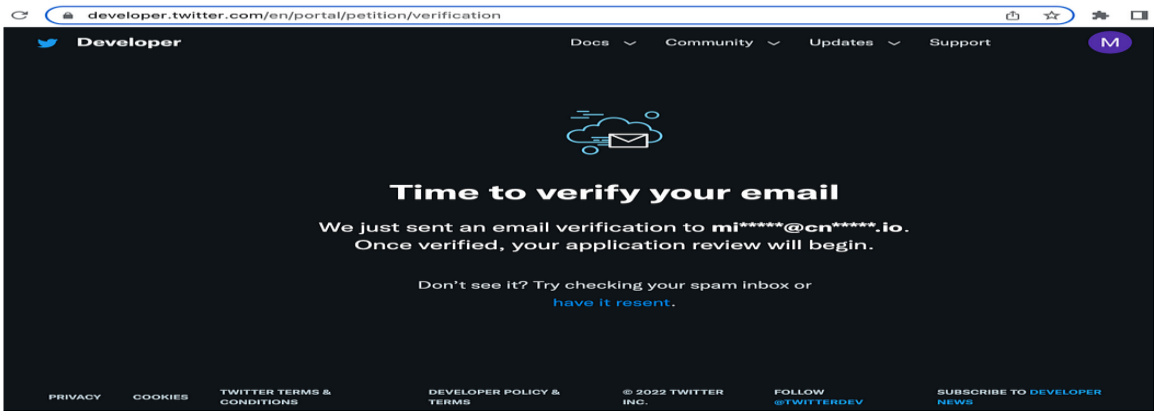
CorgisPlay
@CorgisPlay
Switch @username

This @username will be used to log in to your account.

The phone number associated with this Twitter @username is not verified. You must add a valid phone number and verify it prior to applying for developer access.

Add a valid phone number

3.) Verify your email.



developer.twitter.com/en/portal/petition/verification

Developer Docs Community Updates Support

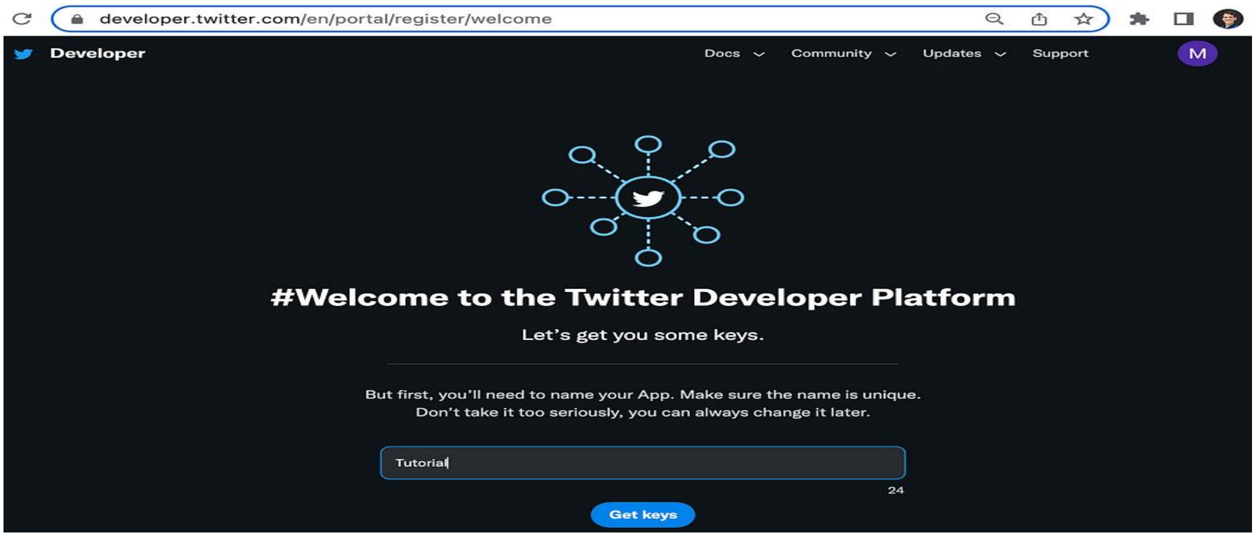
Time to verify your email

We just sent an email verification to mi*****@cn*****.io.
Once verified, your application review will begin.

Don't see it? Try checking your spam inbox or have it resent.

PRIVACY COOKIES TWITTER TERMS & CONDITIONS DEVELOPER POLICY & TERMS © 2022 TWITTER INC. FOLLOW @TWITTERDEV SUBSCRIBE TO DEVELOPER NEWS

4.) After verifying your email, you will be sent to a welcome screen. Name your app and click on Get keys.



developer.twitter.com/en/portal/register/welcome

Developer Docs Community Updates Support

#Welcome to the Twitter Developer Platform

Let's get you some keys.

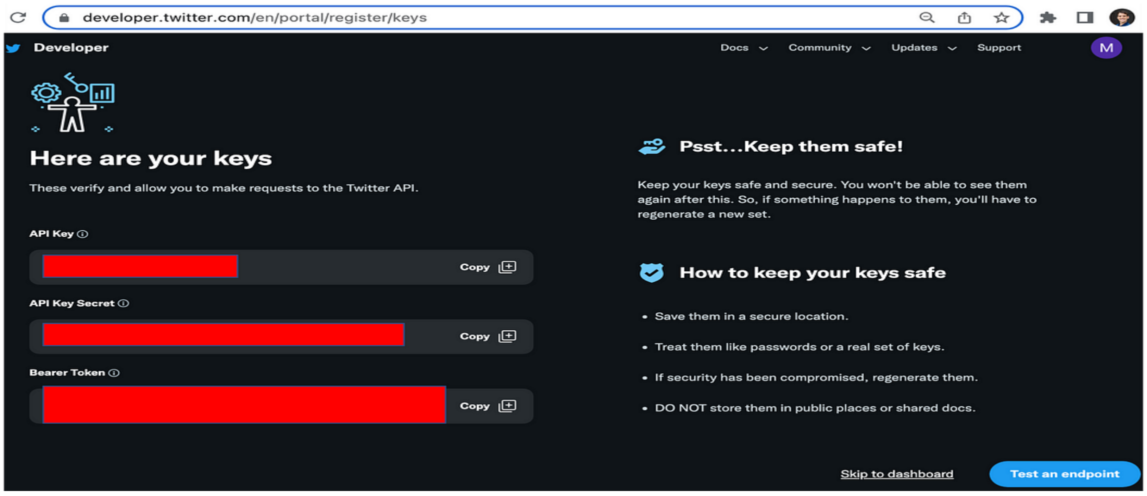
But first, you'll need to name your App. Make sure the name is unique.
Don't take it too seriously, you can always change it later.

Tutorial

24

Get keys

5.) You now have access to your keys. Make sure to save your information to a secure location. You will need them to access data using the twitter api. This is enough information for OAuth 2.0.



developer.twitter.com/en/portal/register/keys

Developer Docs Community Updates Support

Here are your keys

These verify and allow you to make requests to the Twitter API.

API Key

[redacted] Copy

API Key Secret

[redacted] Copy

Bearer Token

[redacted] Copy

Psst...Keep them safe!

Keep your keys safe and secure. You won't be able to see them again after this. So, if something happens to them, you'll have to regenerate a new set.

How to keep your keys safe

- Save them in a secure location.
- Treat them like passwords or a real set of keys.
- If security has been compromised, regenerate them.
- DO NOT store them in public places or shared docs.

Skip to dashboard Test an endpoint

Using tweepy to Access Twitter Data

```
pip install tweepy
```

Search for Tweets from the Last 7 Days

The code below will search and return tweets for the last 7 days with a maximum of 100 tweets per request. This particular code searches for tweets (not retweets) in English that contain the hashtag #petday.

```
#OAuth2.0
Version
         import tweepy


         #Put your Bearer Token in the parenthesis below
         client = tweepy.Client(bearer_token='Change this')

         """
         If you don't understand search queries, there is an excellent introduction to it
         here:
         https://github.com/twitterdev/getting-started-with-the-twitter-api-v2-for-
         academic-research/blob/main/modules/5-how-to-write-search-queries.md
         """

         # Get tweets that contain the hashtag #petday
         # -is:retweet means I don't want retweets
         # lang:en is asking for the tweets to be in english
         query = '#petday -is:retweet lang:en'
         tweets = client.search_recent_tweets(query=query,
         tweet_fields=['context_annotations', 'created_at'], max_results=100)

         """
         What context_annotations are:
         https://developer.twitter.com/en/docs/twitter-api/annotations/overview
         """
         for tweet in tweets.data:
             print(tweet.text)
             if len(tweet.context_annotations) > 0:
                 print(tweet.context_annotations)
```

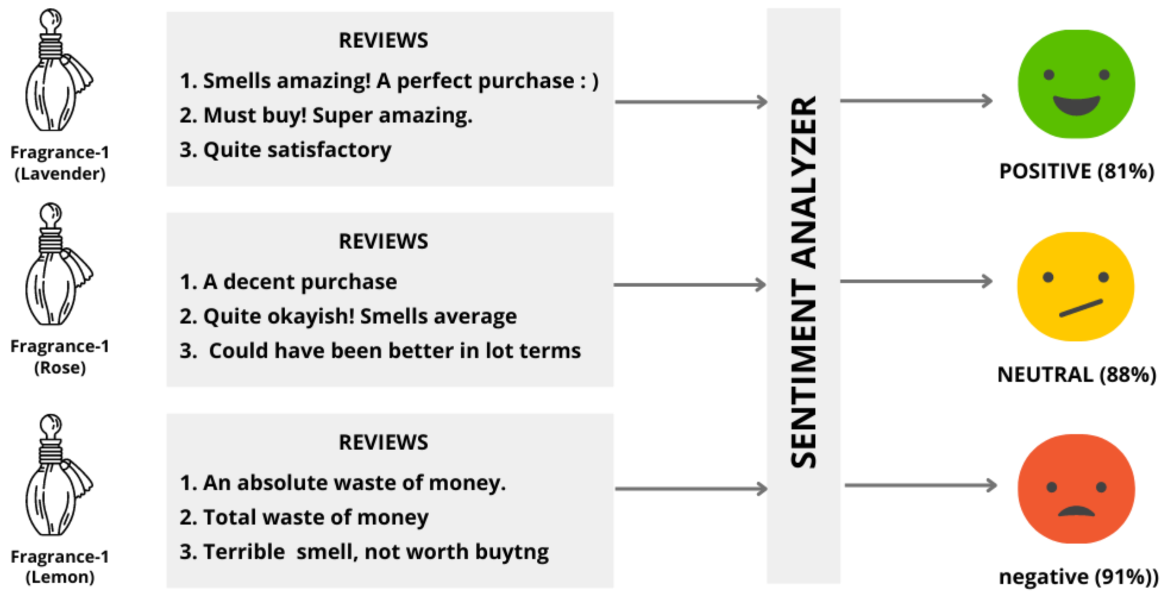
Note that in order to get tweets older than just the last 7 days, you will need to use the search_all_tweets method which is ONLY available if you upgrade to the academic research product track or other elevated access levels.

Specific Software / Hardware used (if any): Python

Experiment No. 5

Objective: Implementing sentiment analysis application in R/Python.

Sentiment Analysis is a use case of Natural Language Processing (NLP) and comes under the category of text classification. To put it simply, Sentiment Analysis involves classifying a text into various sentiments, such as positive or negative, Happy, Sad or Neutral, etc. Thus, the ultimate goal of sentiment analysis is to decipher the underlying mood, emotion, or sentiment of a text. This is also known as Opinion Mining.



We will use the Trip Advisor Hotel Reviews Kaggle dataset for this analysis, so make sure to have it downloaded before you start to be able to code along.

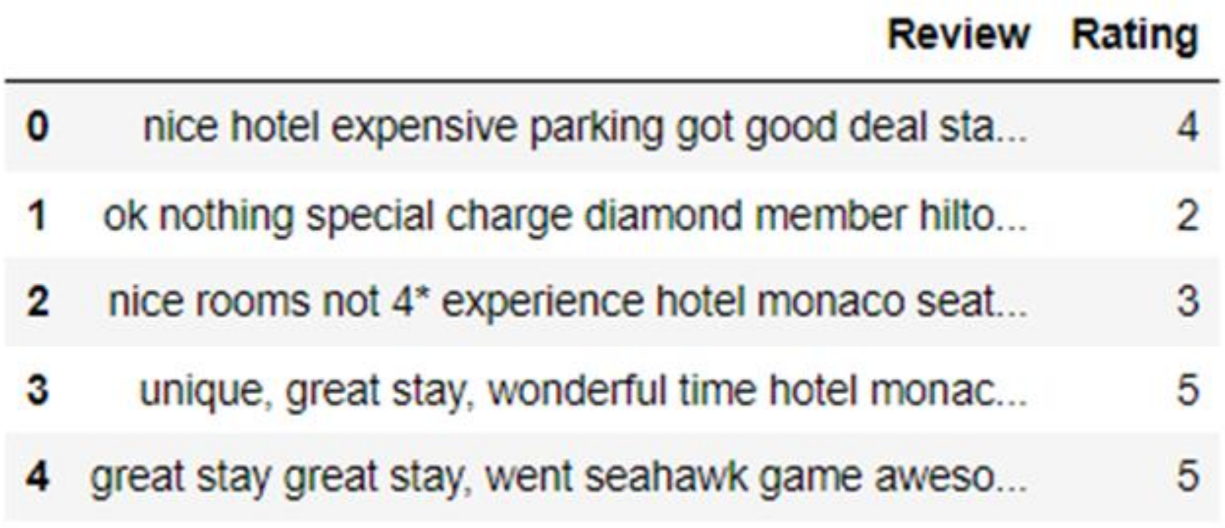
Step 1: Python Pre-Requisites

Make sure to have the following libraries installed as well: NumPy, pandas, Matplotlib, seaborn, Regex, and scikit-learn.

Step 2: Reading the Dataset

```
import pandas as pd
df = pd.read_csv('tripadvisor_hotel_reviews.csv')
df.head()
```

This dataset only has 2 variables: “Review” which contains guests’ impressions of the hotel and “Rating”



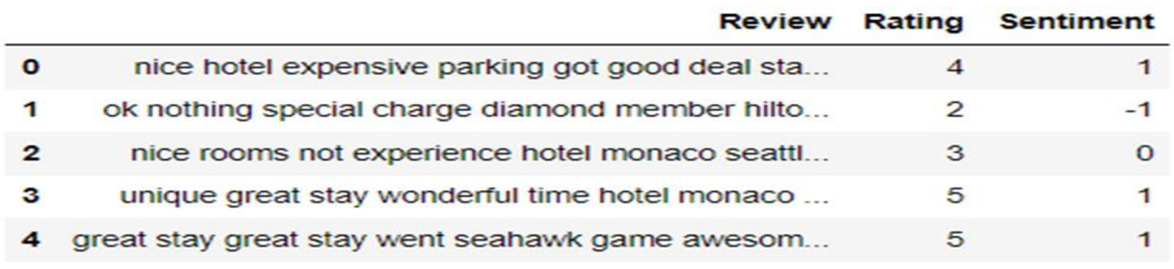
	Review	Rating
0	nice hotel expensive parking got good deal sta...	4
1	ok nothing special charge diamond member hilto...	2
2	nice rooms not 4* experience hotel monaco seat...	3
3	unique, great stay, wonderful time hotel monac...	5
4	great stay great stay, went seahawk game aweso...	5

Step 3: Data Preprocessing

As we already know the TripAdvisor dataset has 2 variables – user reviews and ratings, which range from 1 to 5. We will use “Ratings” to create a new variable called “Sentiment.” In it, we will add 2 categories of sentiment as follows:

- 0 to 1 will be encoded as -1 as they indicate negative sentiment
- 3 will be labeled as 0 as it has a neutral sentiment
- 4 and 5 will be labeled as +1 as they indicate positive sentiment

```
import numpy as np

def create_sentiment(rating):

    if rating==1 or rating==2:
        return -1 # negative sentiment
    elif rating==4 or rating==5:
        return 1 # positive sentiment
    else:
        return 0 # neutral sentiment

df['Sentiment'] = df['Rating'].apply(create_sentiment)
```

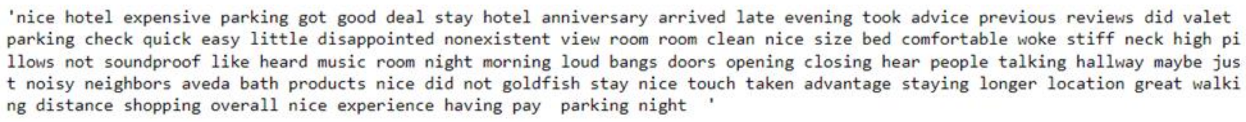


	Review	Rating	Sentiment
0	nice hotel expensive parking got good deal sta...	4	1
1	ok nothing special charge diamond member hilto...	2	-1
2	nice rooms not experience hotel monaco seattl...	3	0
3	unique great stay wonderful time hotel monaco ...	5	1
4	great stay great stay went seahawk game awesom...	5	1

Notice that we have a new column called "Sentiment" – this will be our target variable. We will train a machine learning model to predict the sentiment of each review.

First, however, we need to preprocess the "Review" column in order to remove punctuation, characters, and digits. The code looks like this:

```
from sklearn.feature_extraction.text import re

def clean_data(review):

    no_punc = re.sub(r'[^\w\s]', '', review)
    no_digits = ''.join([i for i in no_punc if not
i.isdigit()])

    return(no_digits)

df['Review'] = df['Review'].apply(clean_data)
df['Review'][0]
```



```
'nice hotel expensive parking got good deal stay hotel anniversary arrived late evening took advice previous reviews did valet
parking check quick easy little disappointed nonexistent view room room clean nice size bed comfortable woke stiff neck high pi
llows not soundproof like heard music room night morning loud bangs doors opening closing hear people talking hallway maybe jus
t noisy neighbors aveda bath products nice did not goldfish stay nice touch taken advantage staying longer location great walki
ng distance shopping overall nice experience having pay  parking night  '
```

Step 4: TF-IDF Transformation

Now, we need to convert this text data into a numeric representation so that it can be ingested into the ML model. We will do this with the help of scikit-learn's TF-IDF Vectorizer package.

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(strip_accents=None,
                        lowercase=False,
                        preprocessor=None)

X = tfidf.fit_transform(df['Review'])
```

Step 5: Building and Evaluating the Machine Learning Model

We can now train our algorithm on the review data to classify its sentiment into 3 categories:

- Positive
- Negative
- Neutral

perform a train-test split:

```
from sklearn.model_selection import train_test_split
y = df['Sentiment'] # target variable
X_train, X_test, y_train, y_test = train_test_split(X,y)
```

Now, fit a logistic regression classifier on the training dataset and use it to make predictions on the test data:

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(solver='liblinear')
lr.fit(X_train,y_train) # fit the model
preds = lr.predict(X_test) # make predictions
```

Finally, evaluate the performance:

```
from sklearn.metrics import accuracy_score
accuracy_score(preds,y_test) # 0.86
```

Complete Python Code:

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
df = pd.read_csv('tripadvisor_hotel_reviews.csv')
def create_sentiment(rating):

    res = 0 # neutral sentiment

    if rating==1 or rating==2:
        res = -1 # negative sentiment
    elif rating==4 or rating==5:
        res = 1 # positive sentiment

    return res
df['Sentiment'] = df['Rating'].apply(create_sentiment)
def clean_data(review):

    no_punc = re.sub(r'[^\w\s]', '', review)
    no_digits = ''.join([i for i in no_punc if not
i.isdigit()])

    return(no_digits)
df['Review'] = df['Review'].apply(clean_data)
tfidf = TfidfVectorizer(strip_accents=None,
                        lowercase=False,
                        preprocessor=None)
X = tfidf.fit_transform(df['Review'])
y = df['Sentiment']
X_train, X_test, y_train, y_test = train_test_split(X,y)
lr = LogisticRegression(solver='liblinear')
lr.fit(X_train,y_train)
preds = lr.predict(X_test)
accuracy_score(preds,y_test)
```

Specific Software / Hardware used (if any): Python

References (if any): Datasets from Kaggle

<u>**Experiment No. 6**</u>

Objective: Develop a dashboard and reporting tool based on real time social media data.

A social media dashboard monitors your social media performance metrics like engagement, subscriber or follower count, and audience insights. Social media dashboards bring together metrics from platforms like Facebook, Twitter, and YouTube to display your social media marketing performance in a single view. When you track your metrics on a social media dashboard, you have quick access to insights that will help you make smart, data-driven marketing decisions.

Use Of social media dashboard

A social media monitoring dashboard displays all of your metrics in a single view. Use your social media metrics to shape your marketing strategy, engage with your audience, increase your conversion rates, and generate revenue. Social media dashboards allow you to gain insight with a single glance and share your performance with your team so you can stay on top of your social media strategies.

How do I create a social media dashboard?

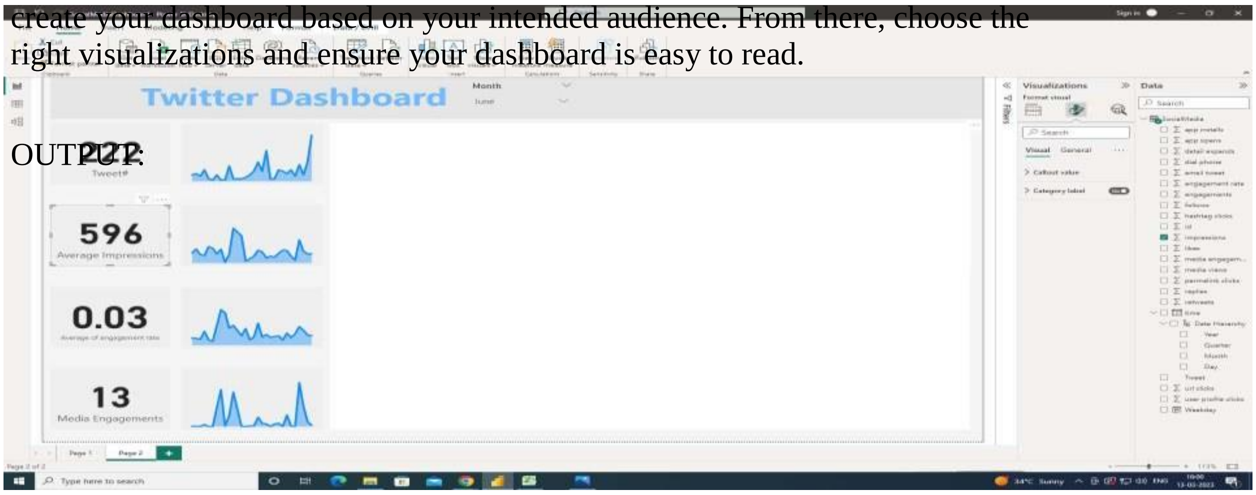
Before you start building a social media dashboard, ask yourself these questions.

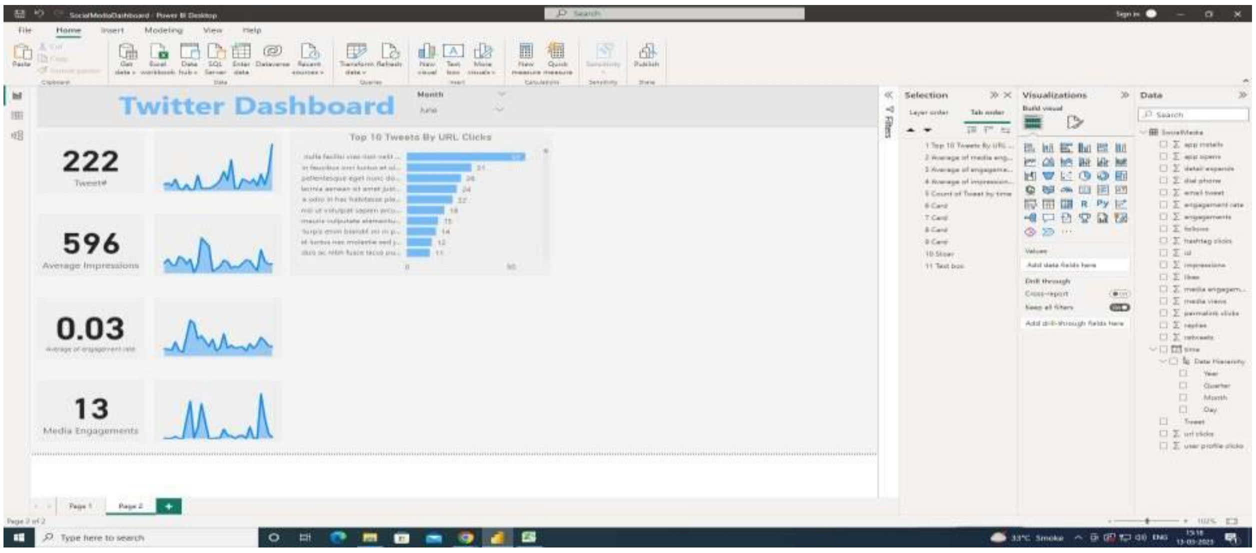
- Who is the audience for the marketing dashboard?
- What information do they need?
- What do they already know about the marketing metrics?
- What is their level of experience with marketing data?

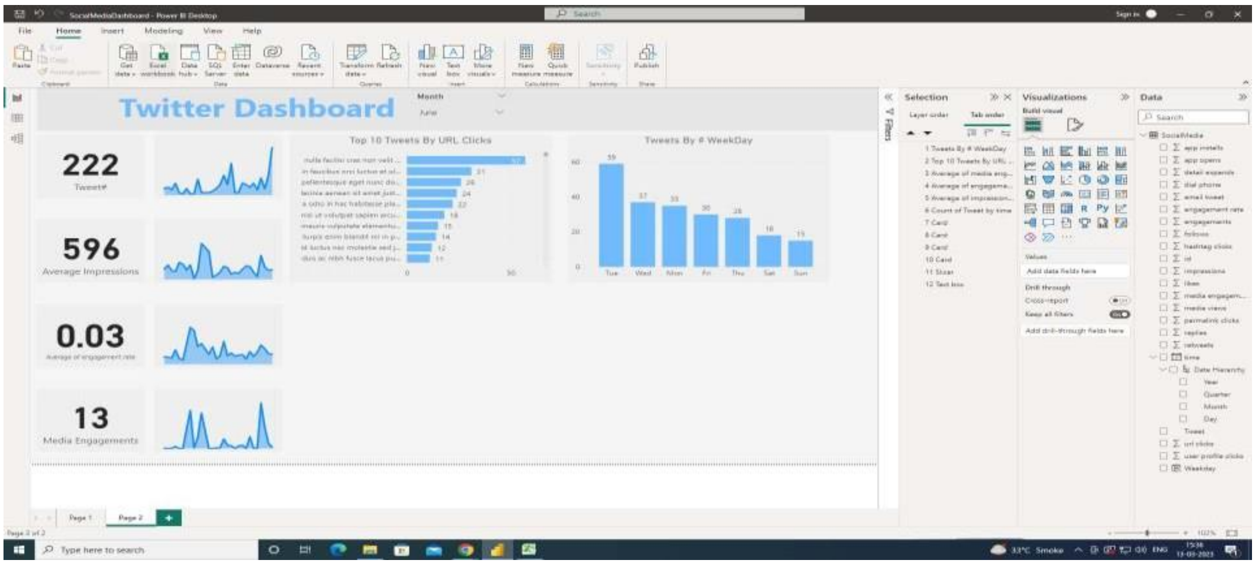
These questions will help you narrow the scope of your dashboard and identify the top social media marketing metrics to track.

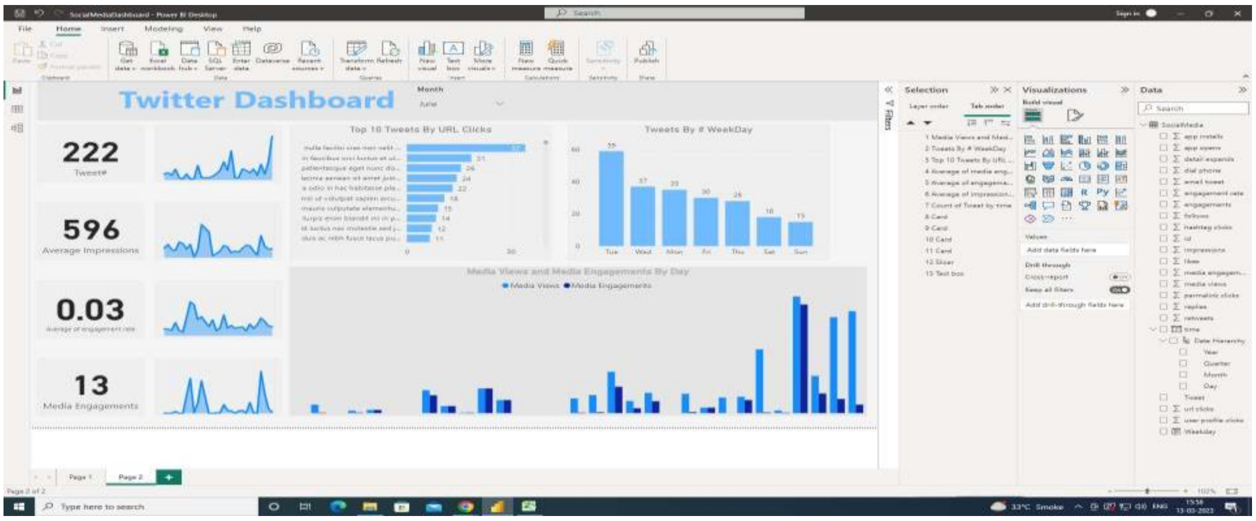
A well-designed social media dashboard will communicate a clear message, so create your dashboard based on your intended audience. From there, choose the right visualizations and ensure your dashboard is easy to read.

OUTPUT:




Twitter Dashboard
Month
June
Top 10 Tweets By URL Clicks
222
Tweet#
596
Average Impressions
0.03
Average of engagement rate
13
Media Engagements
Selection
Visualizations
Data
Page 1
Page 2


Twitter Dashboard
Month
June
Top 10 Tweets By URL Clicks
Tweets By # WeekDay
222
Tweet#
596
Average Impressions
0.03
Average of engagement rate
13
Media Engagements
Tue
Wed
Mon
Fri
Thu
Sat
Sun
Selection
Visualizations
Data
Page 1
Page 2


Twitter Dashboard
Month
June
Top 10 Tweets By URL Clicks
Tweets By # WeekDay
Media Views and Media Engagements By Day
● Media Views ● Media Engagements
222
Tweet#
596
Average Impressions
0.03
Average of engagement rate
13
Media Engagements
Selection
Visualizations
Data
Page 1
Page 2

Experiment No. 7

Objective: Analyze competitor activities using social media data.

SimilarWeb is a tool that estimates the total amount of traffic different websites get. It allows you to see competitors' top traffic sources, broken down into six major categories, including referring sites, social traffic, and top search keywords.

Key Features:

Traffic Overview

Even if the "Traffic Overview" section were the only thing SimilarWeb offered, it would still be worth using.

Total Visits

The "Total Visits" graph lets you see approximately how much traffic a given site has gotten over the past month. And a trendline of that site's traffic over the past 6 months.

Engagement Metrics

You can also see engagement metrics like average visit duration, page views, and bounce rate. **Traffic Sources**

It shows the breakdown of what percentage of a site's traffic comes from each of these channels: Direct, Referrals, Search, Social, Email, and Display Advertising. This is incredibly useful information you can use to help determine where you should focus your content marketing efforts.

Referrals

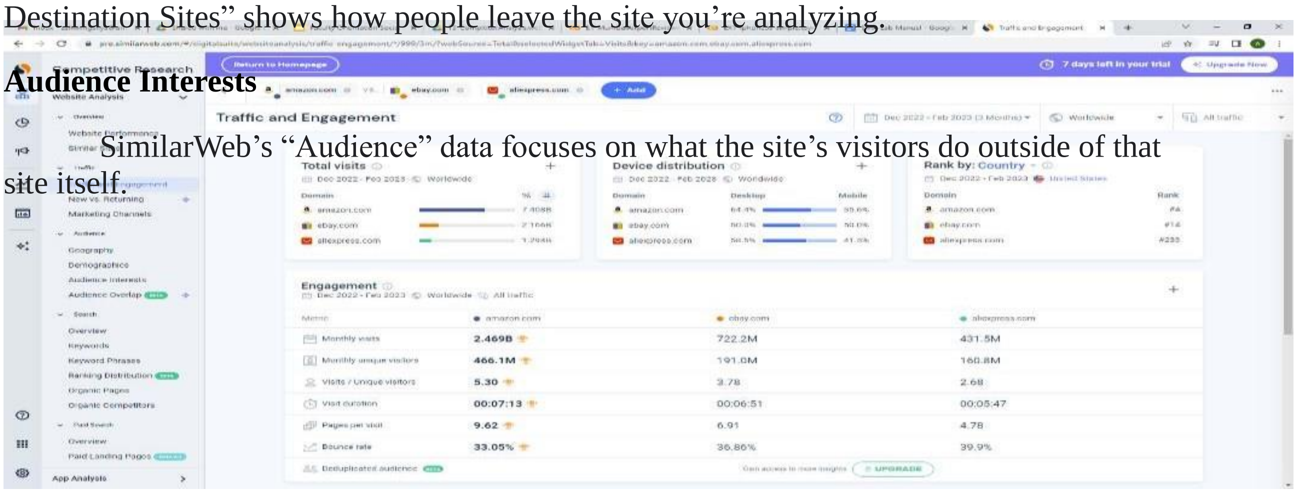
Well, you can see the % of the site's overall traffic that is coming from referral traffic.

Top Destination Sites

While "Top Referring Sites" shows where the site's visitors are coming from, "Top Destination Sites" shows how people leave the site you're analyzing.

Audience Interests

SimilarWeb's "Audience" data focuses on what the site's visitors do outside of that site itself.



Traffic and Engagement

Dec 2022 - Feb 2023 (3 Months) · Worldwide · All traffic

Total visits · Dec 2022 - Feb 2023 · Worldwide

Domain	%
amazon.com	7.408B
ebay.com	2.166B
aliexpress.com	1.294B

Device distribution · Dec 2022 - Feb 2023 · Worldwide

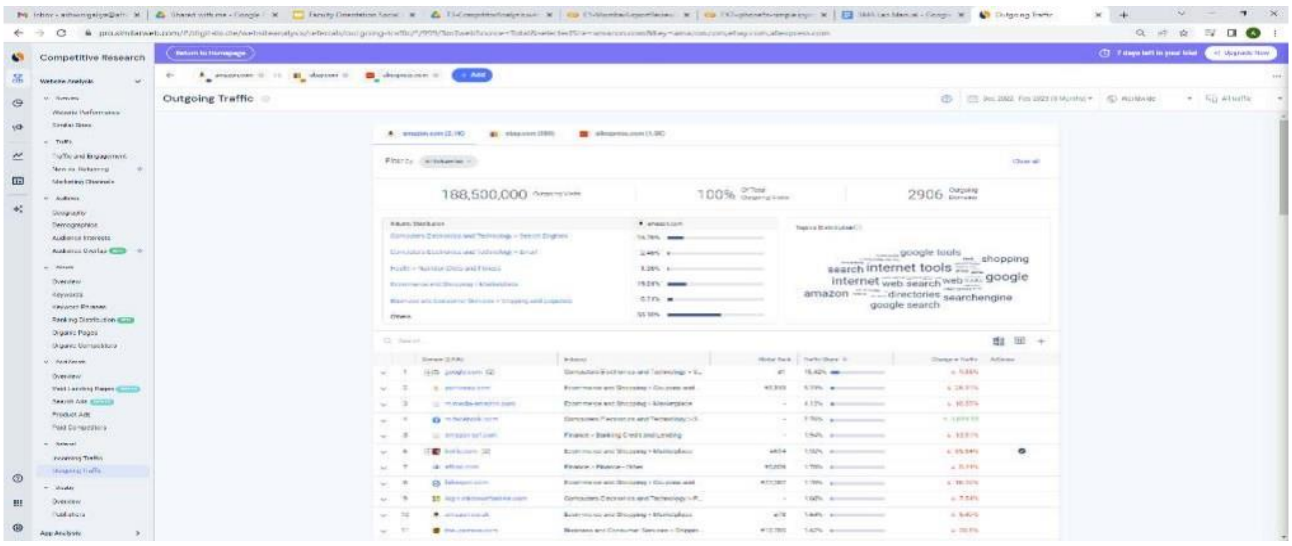
Domain	Desktop	Mobile
amazon.com	64.4%	35.6%
ebay.com	50.0%	50.0%
aliexpress.com	58.5%	41.5%

Rank by: Country · Dec 2022 - Feb 2023 · United States

Domain	Rank
amazon.com	#4
ebay.com	#14
aliexpress.com	#235

Engagement · Dec 2022 - Feb 2023 · Worldwide · All traffic

Metric	amazon.com	ebay.com	aliexpress.com
Monthly visits	2.469B	722.2M	431.5M
Monthly unique visitors	466.1M	191.0M	160.8M
Visits / Unique visitors	5.30	3.78	2.68
Visit duration	00:07:13	00:06:51	00:05:47
Pages per visit	9.62	6.91	4.78
Bounce rate	33.05%	36.86%	39.9%
Deduplicated audience			



Competitive Research

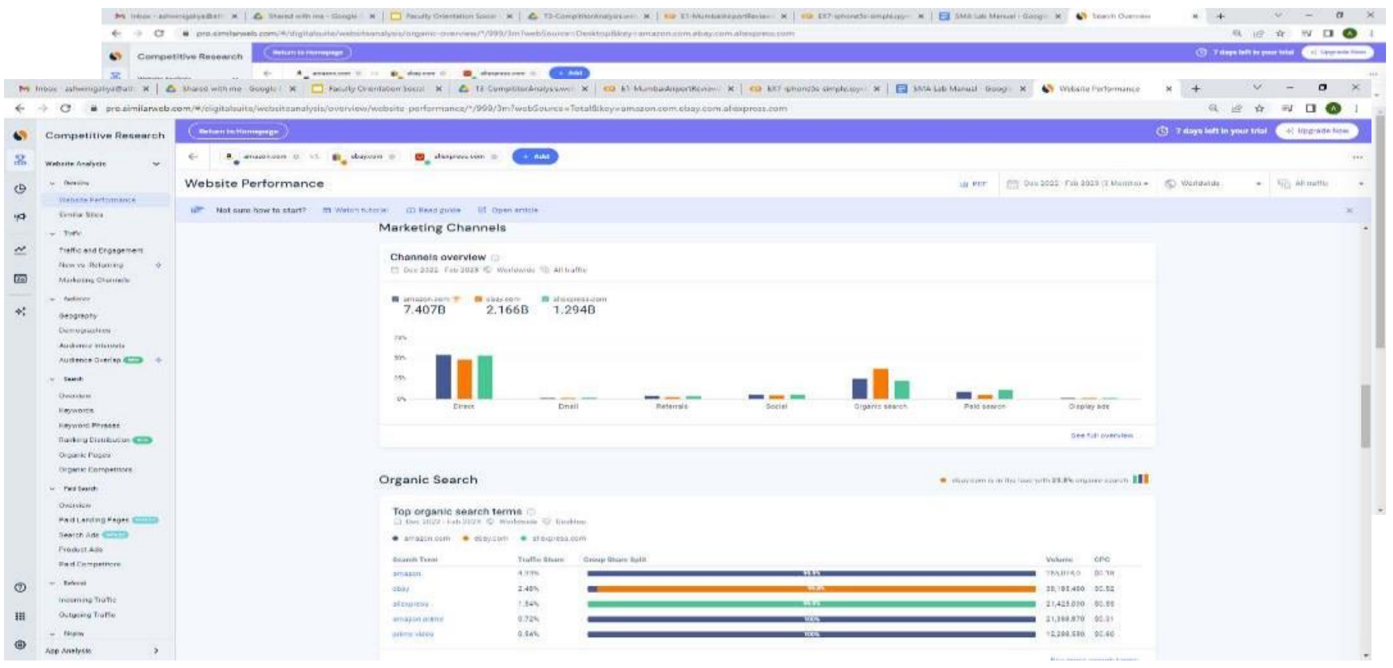
Outgoing Traffic

amazon.com | ebay.com | aliexpress.com

188,500,000 Outgoing Visits | 100% Of Total Outgoing Visits | 2906 Outgoing Domains

Industry Distribution	amazon.com
Computers Electronics and Technology > Search Engines	14.78%
Computers Electronics and Technology > Email	2.48%
Health > Nutrition Diets and Fitness	1.28%
Ecommerce and Shopping > Marketplace	15.24%
Business and Consumer Services > Shipping and Logistics	0.71%
Others	55.18%

	Domain	Industry	Global Rank	Traffic Share	Change in Traffic
1	google.com	Computers Electronics and Technology > S...	#1	15.42%	[illegible]
2	[illegible]	Ecommerce and Shopping > Coupons and ...	[illegible]	[illegible]	[illegible]
3	m.media-amazon.com	Ecommerce and Shopping > Marketplace	-	4.13%	[illegible]
4	m.facebook.com	Computers Electronics and Technology > S...	-	[illegible]	[illegible]
5	amazon.[illegible].com	Finance > Banking Credit and Lending	-	[illegible]	[illegible]
6	[illegible]	Ecommerce and Shopping > Marketplace	[illegible]	[illegible]	[illegible]
7	[illegible]	Finance > Finance - Other	[illegible]	[illegible]	[illegible]
8	[illegible]	Ecommerce and Shopping > Coupons and ...	[illegible]	[illegible]	[illegible]
9	[illegible]	Computers Electronics and Technology > P...	-	[illegible]	[illegible]
10	amazon.co.uk	Ecommerce and Shopping > Marketplace	[illegible]	[illegible]	[illegible]
11	[illegible]	Business and Consumer Services > Shippin...	[illegible]	[illegible]	[illegible]


Website Performance

Marketing Channels

Channels overview

Dec 2022 - Feb 2023 · Worldwide · All traffic

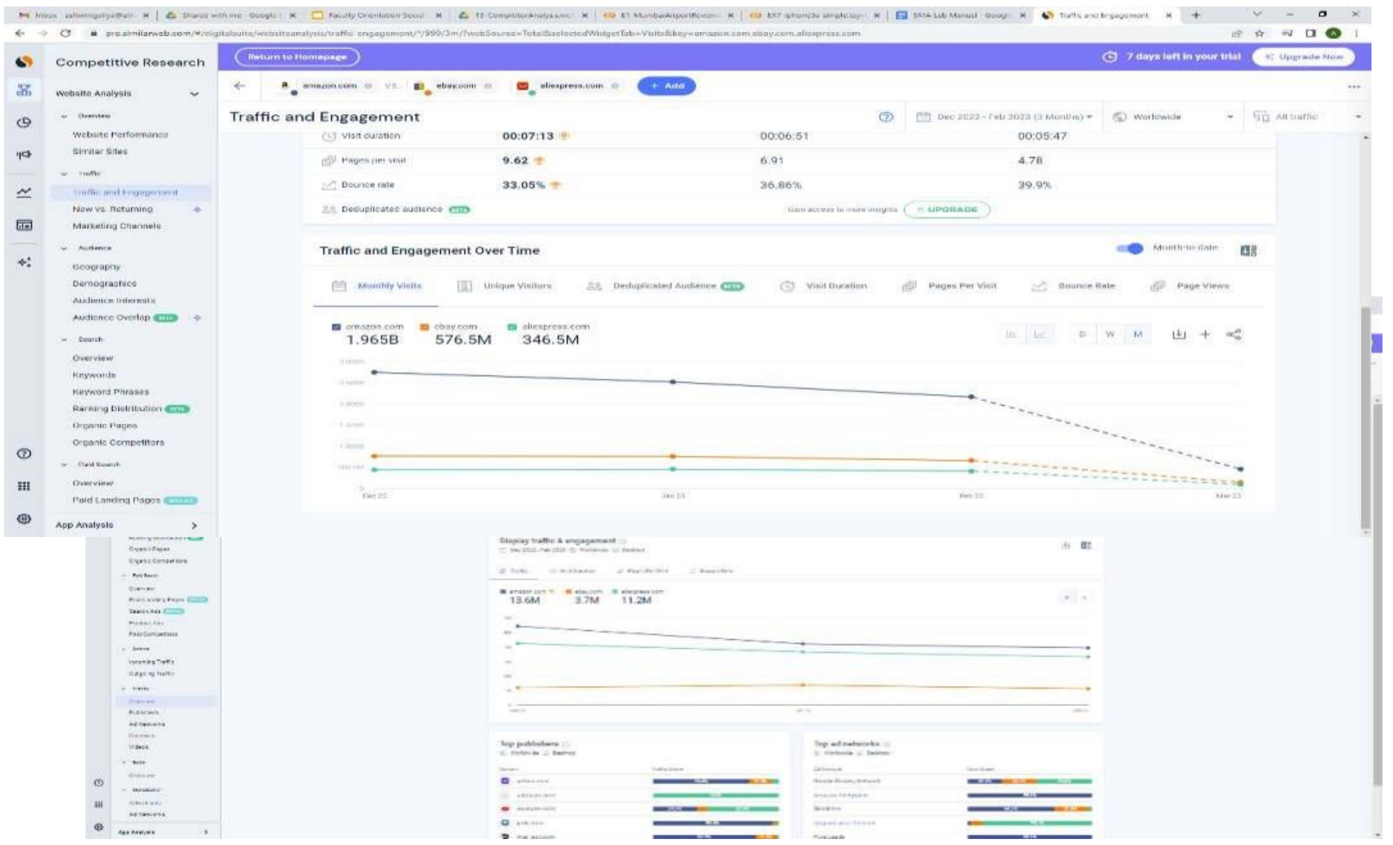
amazon.com	ebay.com	aliexpress.com
7.407B	2.166B	1.294B

Organic Search

Top organic search terms

Dec 2022 - Feb 2023 · Worldwide · Desktop

Search Term	Traffic Share	Group Share Split	Volume	CPC
amazon	4.77%	[illegible]	[illegible]	$0.78
ebay	2.40%	[illegible]	28,193,400	$0.52
aliexpress	1.54%	[illegible]	21,425,030	$0.88
amazon prime	0.72%	100%	21,388,870	$0.31
prime video	0.54%	100%	12,288,580	$0.60


Traffic and Engagement

Dec 2022 - Feb 2023 (3 Months) · Worldwide · All traffic

	amazon.com	ebay.com	aliexpress.com
Visit duration	00:07:13	00:06:51	00:05:47
Pages per visit	9.62	6.91	4.78
Bounce rate	33.05%	36.86%	39.9%

Traffic and Engagement Over Time

Monthly Visits

amazon.com	ebay.com	aliexpress.com
1.965B	576.5M	346.5M

Specific Software / Hardware used (if any): SimilarWeb tool

Experiment No. 8

Objective: Link a website to Google Analytics and Use Web browser to Analyze the statistics of a web site using Google Analytics

Google Analytics is a free website analytics dashboard that provides a wealth of insights about your website and its visitors, including those who find you through social media.
For instance, you can track:

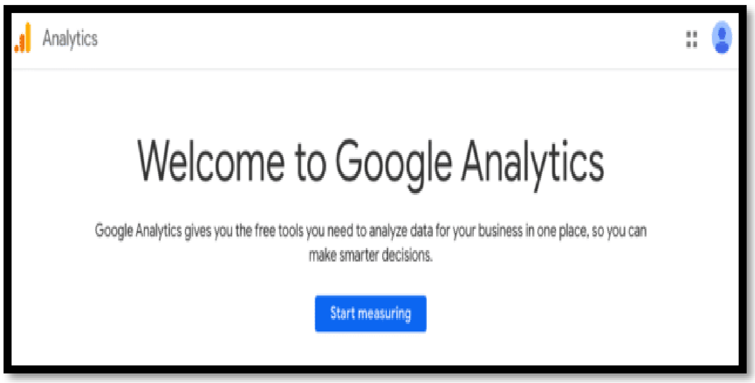
- Total traffic to your site and traffic sources (including social networks)
- Individual page traffic
- Number of leads converted and where those leads come from
- Whether your traffic comes from mobile or desktop.

When you add Google Analytics to your overall social media analytics and reporting strategy, you get even more insights into how social media is working for your business. That's because Google Analytics social media reports allow you to:

- Discover which social media platforms give you the most traffic
- Calculate the ROI(Return of Investment) of your social media campaigns
- See what content works best with each social media platform
- See how many sales conversions your business gets from social media

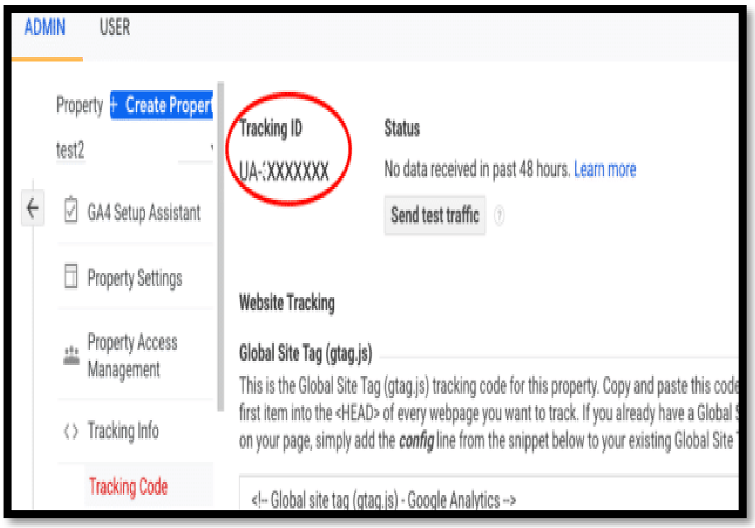
Using Google Analytics to track social media: 5 simple steps

Step 1: Create a Google Analytics account
1. Create a Google Analytics account by clicking the **Start measuring** button to sign up on the GA page.
2. Enter your account name and choose your data sharing settings. When you're ready, click **Nex**t.
3. This is where you have to pay attention to get the Universal Analytics tracking code. Under **Property name**, enter the name of your website or business (not your URL). Choose your time zone and currency. Then, click **Show advanced options**.



Google Analytics gives you the free tools you need to analyze data for your business in one place, so you can make smarter decisions.

4. Switch the toggle on for **Create a Universal Analytics property**. Enter your website URL. Leave the radio button selected for **Create both a Google Analytics 4 and a Universal Analytics property**. Double-check the settings, then click **Next**.
5. On the next screen, you can enter information about your business, but you don't have to. Once you've entered as much detail as you'd like, click **Create**, then accept the **Terms of Service Agreement** in the pop-up box.
6. In the bottom left corner of the Google Analytics dashboard, click **Admin**. Select the account and property you're looking for. In the Property column, click **Tracking Info**.
7. Click **Tracking code** to get your tracking ID.

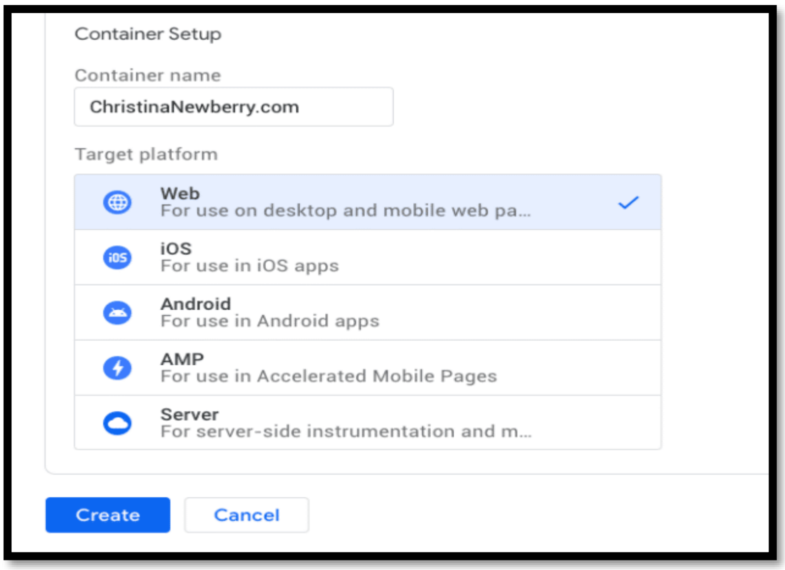


No data received in past 48 hours. Learn more

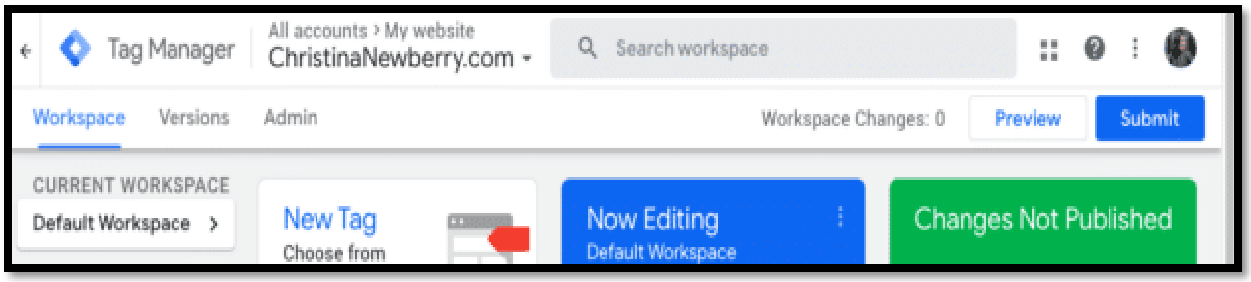
This is the Global Site Tag (gtag.js) tracking code for this property. Copy and paste this code first item into the <HEAD> of every webpage you want to track. If you already have a Global S on your page, simply add the *config* line from the snippet below to your existing Global Site

```
<!-- Global site tag (gtag.js) - Google Analytics -->
```

Step 2: Set up Google Tag Manager
Google Tag Manager allows you to send data to Google Analytics without coding knowledge.
1. Create an account on the Google Tag Manager dashboard. Choose a good account name, the country your business is in, and whether or not you want to share your data with Google to enable benchmarking.
2. Scroll down to the **Container Setup** section. A container holds all the macros, rules, and tags needed to track data for your website. Enter a name you'd like for your container and choose **Web** as your Target platform, then click **Create**. Review the **Terms of Service** in the pop-up and click **Yes**.

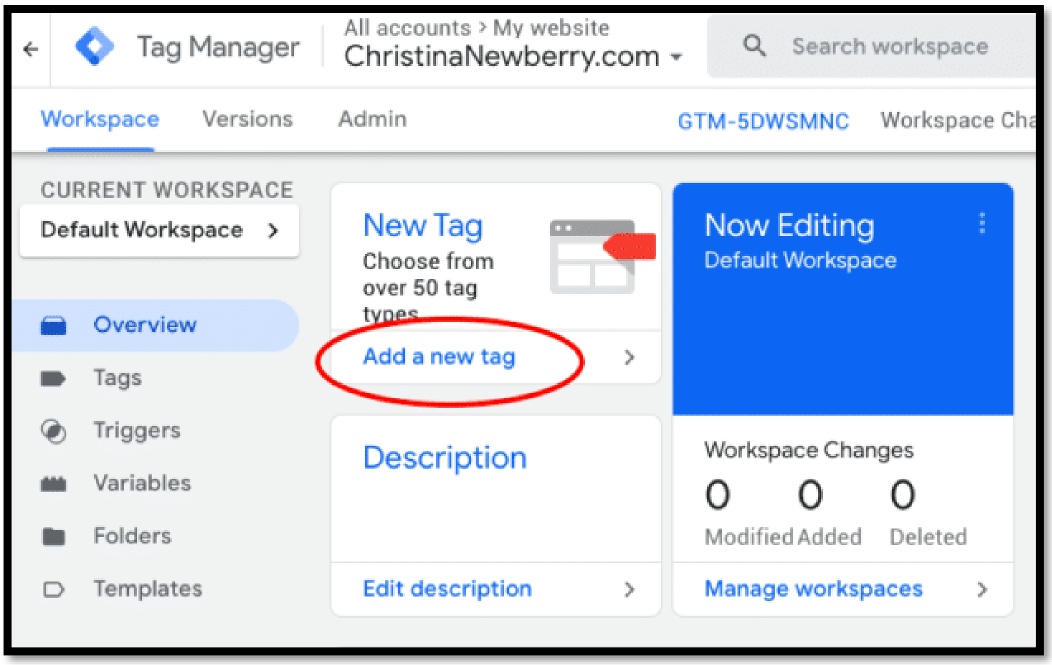


3. Copy and paste the code from the Install Google Tag Manager pop-up box onto your website. The first snippet goes in the <head> section of your page, and the second in the <body> section. The code has to go on every page of your website, so it's best if you can add it to the templates of your content management system (CMS).
4. Once you've added the code to your website, return to the Tag Manager workspace and click **Submit** on the top right of the screen.

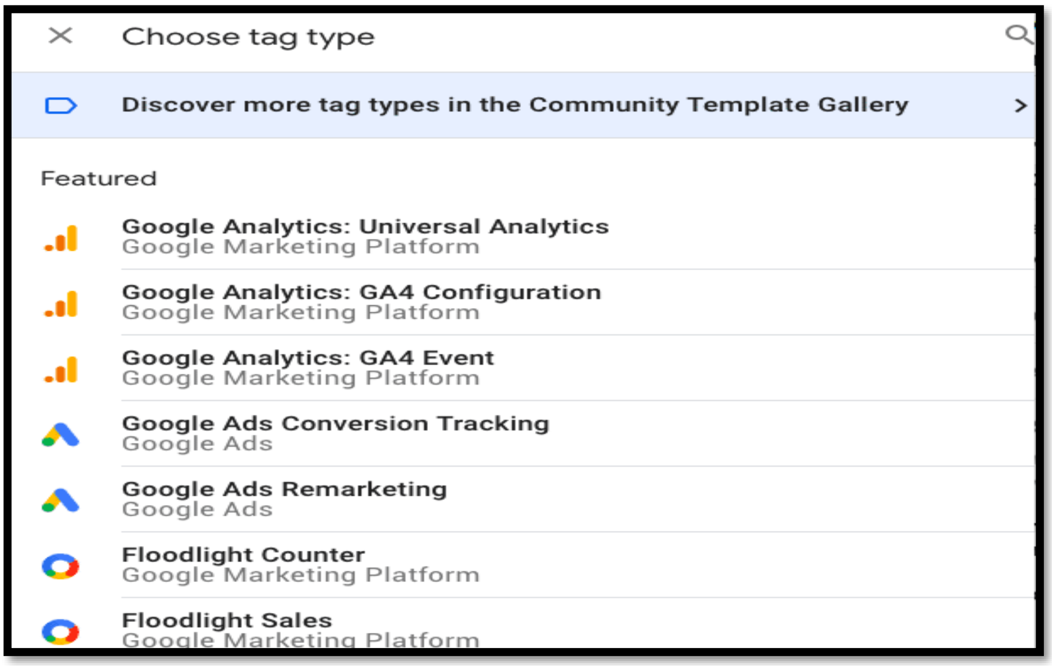


Step 3: Create your analytics tags
Now it's time to merge Google Tag Manager with Google Analytics.
1. Go to your Google Tag Manager workspace and click **Add a new tag**.
There are two areas of the tag you'll be able to customize:

- Configuration. Where the data collected by the tag will go.
- Triggering. What type of data you want to collect.



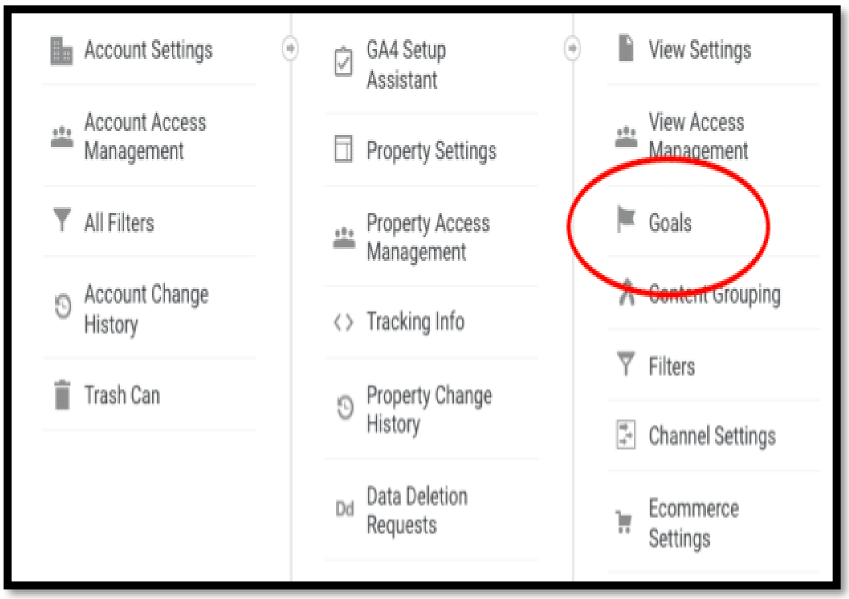
2. Click **Tag Configuration** and choose **Google Analytics: Universal Analytics**.
3. Choose the type of data you want to track and then choose **New Variable…** from the dropdown menu under **Google Analytics Settings**.
4. Head back to the **Triggering** section to select the data you want to send to Google Analytics. Select **All Pages** to send data from all your web pages, then click **Add**.
Click **Save** and voila! You have a new Google Tag tracking and sending data to Google Analytics.


Choose tag type
Discover more tag types in the Community Template Gallery
Featured
Google Analytics: Universal Analytics
Google Marketing Platform
Google Analytics: GA4 Configuration
Google Marketing Platform
Google Analytics: GA4 Event
Google Marketing Platform
Google Ads Conversion Tracking
Google Ads
Google Ads Remarketing
Google Ads
Floodlight Counter
Google Marketing Platform
Floodlight Sales
Google Marketing Platform

Step 4: Add social media to Google Analytics goals

Google Analytics uses “goals” to track your website’s key performance indicators. Before you add your Google Analytics social media goals, think about what kinds of metrics will have the most impact on your social media reporting and overall business objectives.
The SMART goal-setting framework can be very helpful on this front.

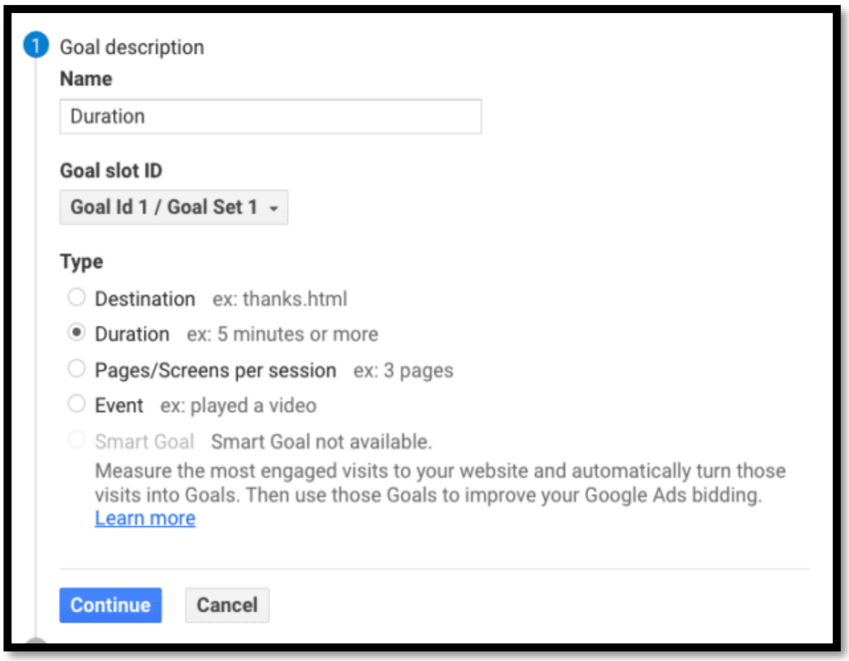
1. Go to your Google Analytics dashboard and click the **Admin** button on the bottom left corner. In the **View** column, click on **Goals**.



You can also see the different types of goals Google Analytics can track for you. They are:

- **Destination**. e.g. if your goal was for your user to reach a specific web page.
- **Duration**. e.g. if your goal was for users to spend a specific amount of time on your site.
- **Pages/Screens per session**. e.g. if your goal was to have users go to a specific number of pages.
- **Event**. e.g. if your goal was to get users to play a video or click on a link.

Choose your settings, then click **Continue**. On the next screen, you can get even more specific with your goals, like choosing exactly how long users need to spend on your site in order to consider it a success.
Save the goal and Google Analytics will start to track it for you.


1 Goal description

Name

Duration

Goal slot ID

Goal Id 1 / Goal Set 1

Type

Destination ex: thanks.html

[✓ Duration] ex: 5 minutes or more

Pages/Screens per session ex: 3 pages

Event ex: played a video

Smart Goal Smart Goal not available.

Measure the most engaged visits to your website and automatically turn those visits into Goals. Then use those Goals to improve your Google Ads bidding.

Learn more

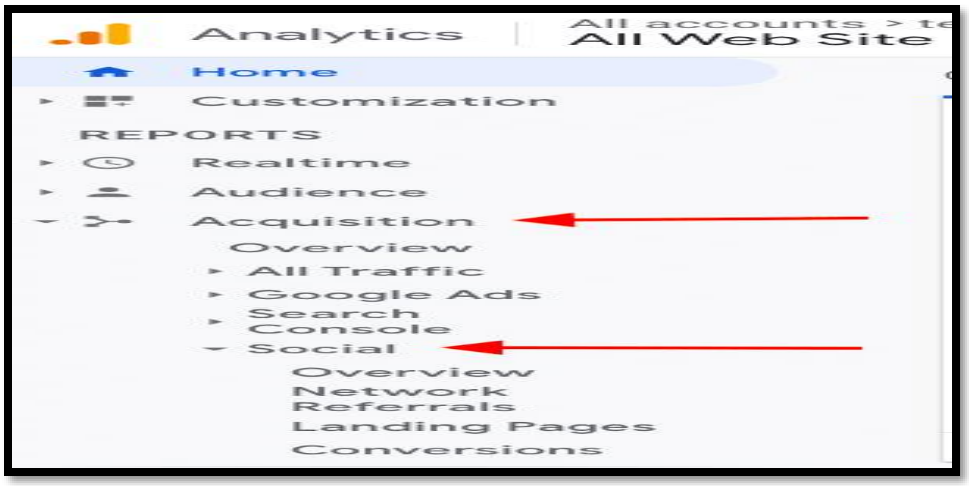
Continue Cancel

Step 5: Pull your Google Analytics social media reports

Google Analytics Universal Analytics currently allows you to view six social analytics reports. These reports showcase the ROI and impact of your social media campaigns.

From your Google Analytics dashboard, click the **down arrows** next to **Acquisitions** and then **Social**.

From here, we'll be able to take a look at the six big Google Analytics social media reports.



1. Overview report

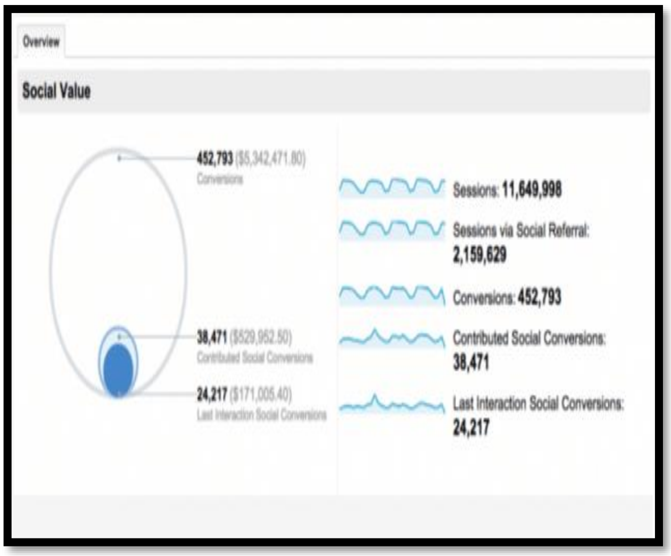
This report gives digital marketers a quick overview of how many people convert via social media platforms. It compares the value of all goal completions with those from social referrals.

2. Network referrals

This report provides engagement metrics from individual social networks. This can help you identify your best performing content on each network. For example, if you're looking for specific Google analytics Facebook referral data, this is the report to check.

3. Landing pages

Here you can see engagement metrics for individual URLs. You’ll also be able to track the originating social network of each URL.


Overview
Social Value
452,793 ($5,342,471.80)
Conversions
38,471 ($529,952.50)
Contributed Social Conversions
24,217 ($171,005.40)
Last Interaction Social Conversions
Sessions: 11,649,998
Sessions via Social Referral: 2,159,629
Conversions: 452,793
Contributed Social Conversions: 38,471
Last Interaction Social Conversions: 24,217

4. Conversions

The Google Analytics social conversions report shows the total number of conversions from each social network as well as their monetary value. So, for example, this is where you can see Google Analytics Instagram conversion data.

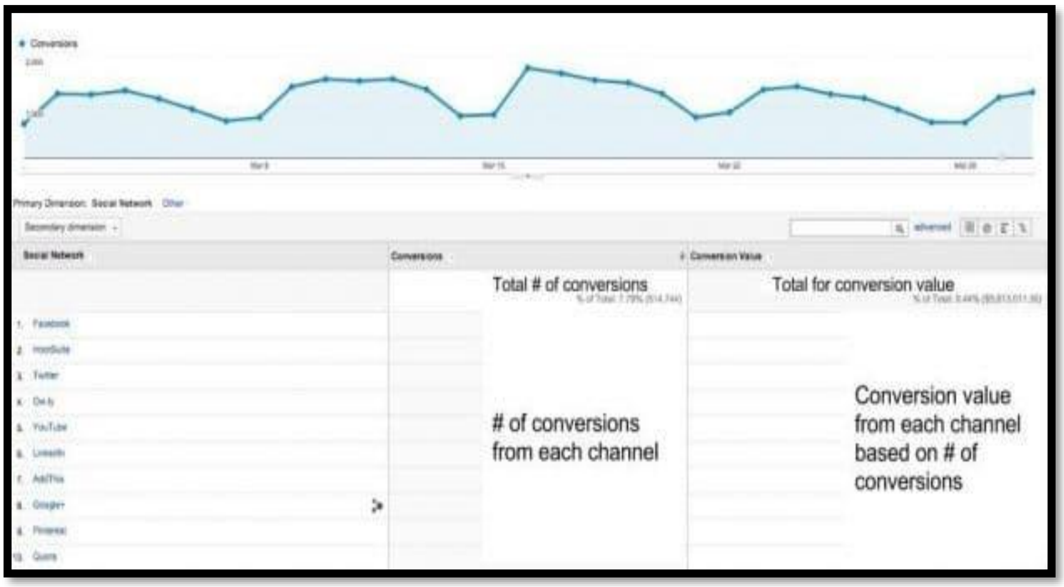
5. Plugins

You know those social share buttons on your website? The Google Analytics social plugins report shows how often those buttons are clicked and for which content. This report includes metrics and data showing which piece of content on your site is shared the most — and which social media networks it’s being shared on — directly from your website.

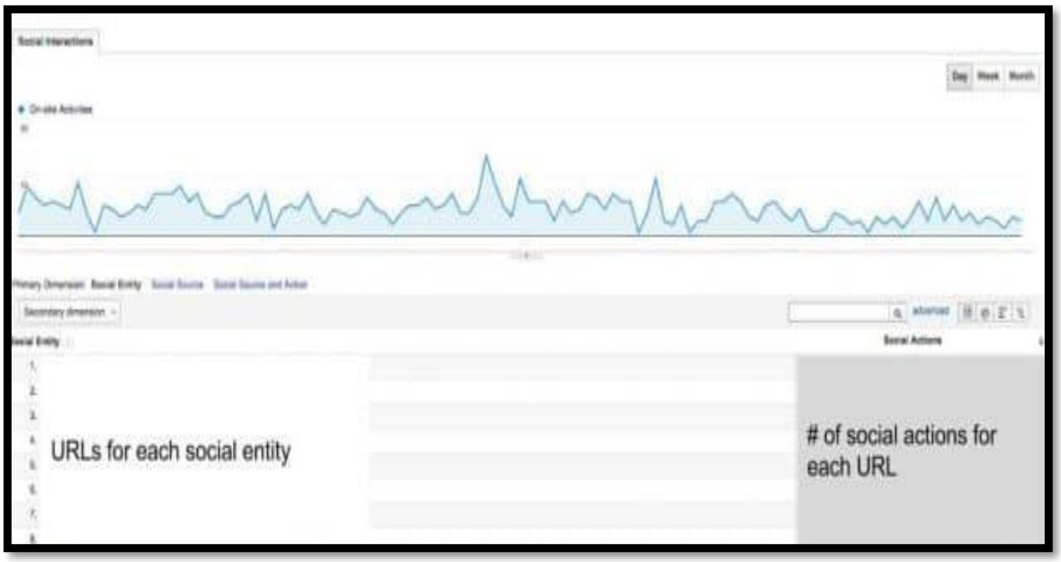
6. Users flow

This report shows digital marketers a “graphical representation of the paths users took through your site from the source through the various pages and where along the paths they exited your site,” according to Google.

For example, if you’re running a campaign that promotes a specific product, you’ll be able to find whether users entered your site via a product page and if they continued to other parts of your site.



Social Network	Conversions	Conversion Value
	Total # of conversions	Total for conversion value
1. Facebook	# of conversions from each channel	Conversion value from each channel based on # of conversions
2. HootSuite		
3. Twitter		
4. Del.ly		
5. YouTube		
6. LinkedIn		
7. AddThis		
8. Google+		
9. Pinterest		
10. Quora		



Social Entity	Social Actions
1.	
2.	
3.	
4. URLs for each social entity	# of social actions for each URL
5.	
6.	
7.	
8.	

Experiment No. 9

Objective: Exploratory Data Analysis and visualization of Social Media Data for business.

Exploratory Data Analysis (EDA) is usually the first step when you have data in hand and want to analyze it.

- In EDA, there is no hypothesis and no model. You are finding patterns and truth from the data.
- EDA is crucial for data science projects because it can: ○ Help you gain intuition about the data;
  - Make comparisons between distributions;
  - Check if the data is on the scale you expect;
  - Find out where data is missing or if there are outliers;
  - Summarize data, calculate the mean, min, max, and variance.
- The basic tools of EDA are plots, graphs, and summary statistics.

Twitter Data Analytics using Python Steps

- Twitter Data extraction using snscrape Python library
- Twitter Data Cleaning and Preprocessing using Python
- Twitter Data Visualization
- Twitter Data Sentiment Analysis using Textblob

Program Code

1.Scrape Twitter Data for Union Budget 2023

```
!pip install snscrape

import pandas as pd

import snscrape.modules.twitter as sntwitter
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
import nltk
nltk.download('stopwords')

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer
import string

import re
import textblob

from textblob import
TextBlob import os

from wordcloud import WordCloud, STOPWORDS
from wordcloud import ImageColorGenerator
import warnings

%matplotlib inline

os.system("snscrape --jsonl --max-results 5000 --since 2023-01-31 twitter-
search 'Budget 2023 until:2023-02-07'>text-query-tweets.json")

tweets_df = pd.read_json("text-query-tweets.json" ,lines=True)
tweets_df.head(5)

tweets_df.to_csv()
```

2. Data Loading

```
df1 = tweets_df[[ 'date', 'rawContent' , 'renderedContent' , 'user' , 'replyCount'
,'retweetCount' , 'likeCount' , 'lang' , 'place' , 'hashtags' , 'viewCount']].copy()
df1.head()

df1.shape
```

3. Twitter Data Cleaning, Preprocessing and Exploratory Data Analysis

```
df1=df1.drop_duplicates("renderedContent")

df1.shape df1.head df1.info

df1.date.value_counts()
```

```
plt.figure(figsize=(17, 5))

sns.heatmap(df1.isnull(), cbar=True, yticklabels=False)
plt.xlabel("Column_Name", size=14, weight="bold")
plt.title("Places of missing values in column",size=17)

plt.show()

import plotly.graph_objects as go

Top_Location_Of_tweet= df1['place'].value_counts().head (10)
```

Twitter Data Cleaning and Preprocessing

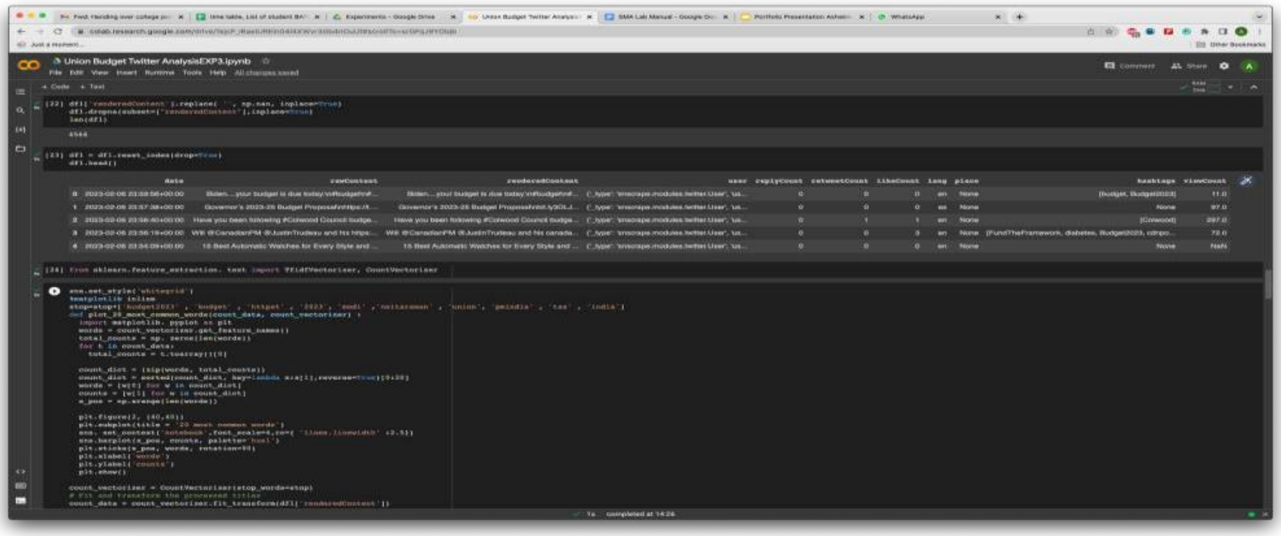
```
from nltk. corpus import stopwords

stop = stopwords.words('english')

df1['renderedContent'].apply(lambda x: [item for item in x if item not in stop])


df1.shape

!pip install tweet-preprocessor

#Remove unnecessary characters

punct  =  ['%','/',':','\\','&amp','&',';','?']


def remove_punctuations(text):

for punctuation in punct:

   text = text.replace(punctuation,'')
 return text

df1['renderedContent'] = df1['renderedContent'].apply

(lambda x: remove_punctuations(x))

df1['renderedContent'].replace( '', np.nan, inplace=True)
df1.dropna(subset=["renderedContent"],inplace=True)
len(df1)
```

```
df1 = df1.reset_index(drop=True)
df1.head()

from sklearn.feature_extraction. text import TfidfVectorizer,
CountVectorizer sns.set_style('whitegrid')

%matplotlib inline

stop=stop+['budget2023' , 'budget' , 'httpst' , '2023', 'modi' ,'nsitaraman' ,
'union', 'pmindia' , 'tax' , 'india']

def plot_20_most_common_words(count_data, count_vectorizer) :


 import matplotlib. pyplot as plt

 words = count_vectorizer.get_feature_names()

 total_counts = np.
 zeros(len(words)) for t in
 count_data:

   total_counts = t.toarray()[0]


 count_dict = (zip(words, total_counts))

 count_dict = sorted(count_dict, key=lambda
 x:x[1],reverse=True)[0:20] words = [w[0] for w in count_dict]

 counts = [w[1] for w in count_dict]

 x_pos = np.arange(len(words))


 plt.figure(2, (40,40))

 plt.subplot(title = '20 most common words')

 sns. set_context('notebook',font_scale=4,rc={ 'lines.linewidth' :2.5})
 sns.barplot(x_pos, counts, palette='husl')

 plt.xticks(x_pos, words, rotation=90)
 plt.xlabel('words')
 plt.ylabel('counts')
 plt.show()


count_vectorizer = CountVectorizer(stop_words=stop)
# Fit and transform the processed titles

count_data = count_vectorizer.fit_transform(df1['renderedContent'])
# print(count_vectorizer)

# print(count_data)

# Visualise the 20 most common words
plot_20_most_common_words(count_data,count_vectorizer)
plt.savefig( 'saved_figure.png')

import cufflinks as cf
cf.go_offline()

cf.set_config_file(offline=False, world_readable=True)

def get_top_n_bigram(corpus, n=None) :

 vec = CountVectorizer(ngram_range=(2, 4), stop_words="english").fit(corpus)
 bag_of_words = vec.transform(corpus)

 sum_words = bag_of_words.sum(axis=0)

 words_freq =[(word, sum_words[0, idx]) for
word, idx in vec.vocabulary_.items()]

 words_freq =sorted(words_freq, key = lambda x: x[1],
 reverse=True) return words_freq[:n]

common_words = get_top_n_bigram(df1['renderedContent'] , 8) mydict={}

for word, freq in common_words:

 bigram_df = pd.DataFrame(common_words,columns = ['ngram', 'count'])

bigram_df.groupby( 'ngram'
).sum()['count'].sort_values(ascending=False).sort_values().plot.barh(title =
'Top 8 bigrams',color='orange' , width=.4, figsize=(12,8),stacked = True)
```

```
def get_subjectivity(text):

 return TextBlob(text).sentiment.subjectivity
def get_polarity(text):

 return TextBlob(text).sentiment.polarity

df1['subjectivity']=df1[ 'renderedContent'].apply(get_subjectivity)
df1[ 'polarity' ]=df1[ 'renderedContent'].apply(get_polarity)
df1.head()

df1['textblob_score'] =df1[ 'renderedContent'].apply

(lambda x: TextBlob(x).sentiment.polarity)

neutral_threshold=0.05

df1['textblob_sentiment']=df1[ 'textblob_score'].apply

(lambda c:'positive' if c >= neutral_threshold else ('Negative' if c <= -
(neutral_threshold) else 'Neutral' ) )

textblob_df =  df1[['renderedContent','textblob_sentiment','likeCount']]
textblob_df

textblob_df["textblob_sentiment"].value_counts()
textblob_df["textblob_sentiment"].value_counts().plot.barh(title =
'Sentiment Analysis',color='orange' , width=.4, figsize=(12,8),stacked
= True)
df_positive=textblob_df[textblob_df['textblob_sentiment']=='positive
' ] df_very_positive=df_positive[df_positive['likeCount']>0]
df_very_positive.head()

df_negative=textblob_df[textblob_df['textblob_sentiment']=='Ne
gative' ] df_negative
df_neutral=textblob_df[textblob_df['textblob_sentiment']=='Neu
tral' ] df_neutral

from wordcloud import WordCloud, STOPWORDS

from PIL import Image
#Creating the text variable

positive_tw =" ".join(t for t in df_very_positive.renderedContent)
```

```
# Creating word _ cloud with text as argument in . generate() rtpthod
word_cloud1 = WordCloud(collocations = False, background_color = 'white')
.generate(positive_tw)

# Display the generated Word Cloud

plt. imshow(word_cloud1, interpolation='bilinear')
plt.axis('off')

plt.show()

#Creating the text variable

negative_tw =" ".join(t for t in df_negative.renderedContent)

# Creating word _ cloud with text as argument in . generate() rtpthod
word_cloud2 = WordCloud(collocations = False, background_color = 'white')
.generate(negative_tw)

# Display the generated Word Cloud

plt. imshow(word_cloud2, interpolation='bilinear')
plt.axis('off')

plt.show()

#Creating the text variable

neutral_tw =" ".join(t for t in df_neutral.renderedContent)

# Creating word _ cloud with text as argument in . generate() rtpthod
word_cloud2 = WordCloud(collocations = False, background_color = 'white')
.generate(neutral_tw)

# Display the generated Word Cloud

plt. imshow(word_cloud2,
interpolation='bilinear') plt.axis('off')

plt.show()
```

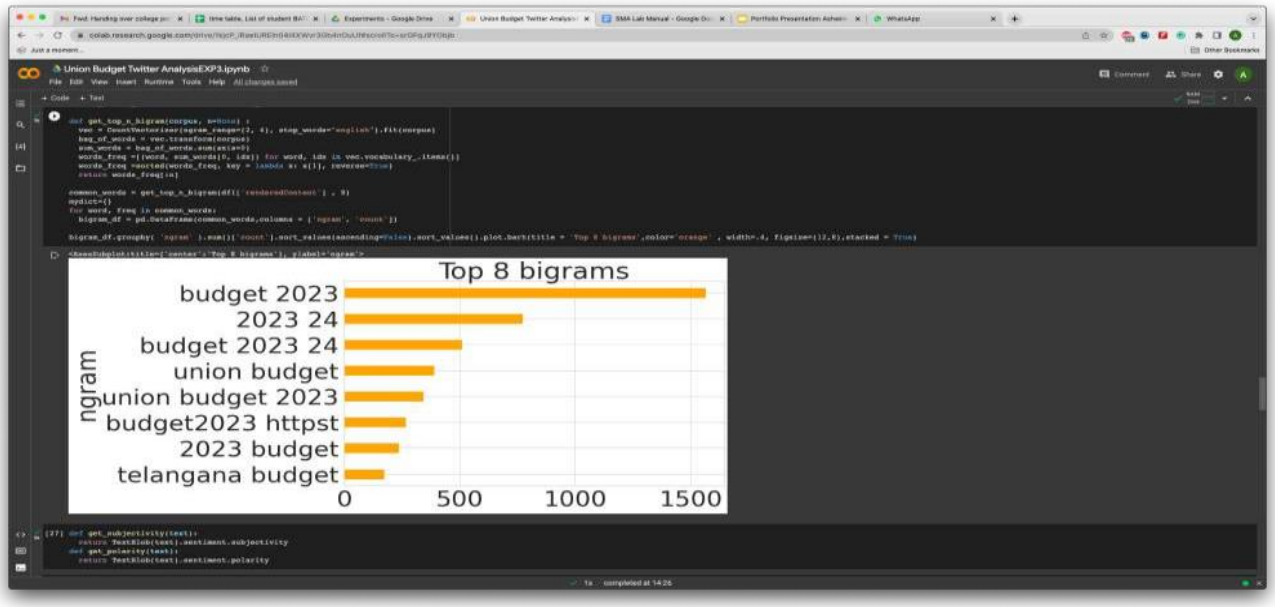
OUTPUT



```
[22] df1['renderedContent'].replace('', np.nan, inplace=True)
     df1.dropna(subset=['renderedContent'],inplace=True)
     len(df1)

     4566

[23] df1 = df1.reset_index(drop=True)
     df1.head()
```

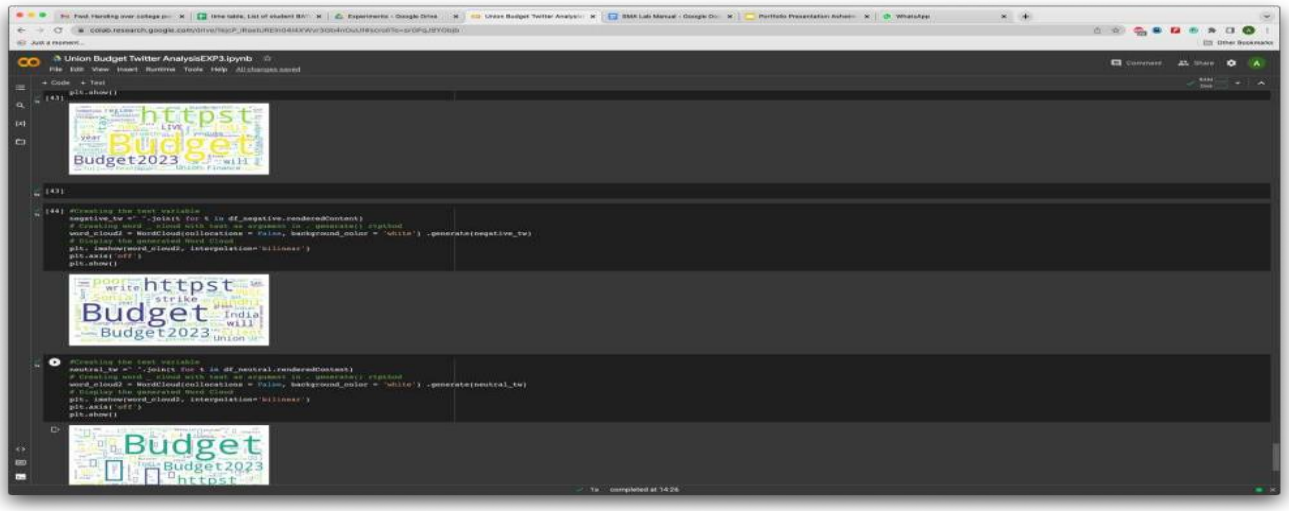
	date	rawContent	renderedContent	user	replyCount	retweetCount	likeCount	lang	place	hashtags	viewCount
0	2023-02-06 23:59:56+00:00	Biden.....your budget is due today.\n#budget\n#...	Biden.....your budget is due today.\n#budget\n#...	{'_type': 'snscrape.modules.twitter.User', 'us...	0	0	0	en	None	[budget, Budget2023]	11.0
1	2023-02-06 23:57:38+00:00	Governor's 2023-25 Budget Proposal\n\nhttps://t....	Governor's 2023-25 Budget Proposal\nbit.ly/3OLJ...	{'_type': 'snscrape.modules.twitter.User', 'us...	0	0	0	en	None	None	97.0
2	2023-02-06 23:56:40+00:00	Have you been following #Colwood Council budge...	Have you been following #Colwood Council budge...	{'_type': 'snscrape.modules.twitter.User', 'us...	0	1	1	en	None	[Colwood]	287.0
3	2023-02-06 23:56:19+00:00	Will @CanadianPM @JustinTrudeau and his https:...	Will @CanadianPM @JustinTrudeau and his canada...	{'_type': 'snscrape.modules.twitter.User', 'us...	0	0	3	en	None	[FundTheFramework, diabetes, Budget2023, cdnpo...	72.0
4	2023-02-06 23:34:09+00:00	15 Best Automatic Watches for Every Style and ...	15 Best Automatic Watches for Every Style and ...	{'_type': 'snscrape.modules.twitter.User', 'us...	0	0	0	en	None	None	NaN

```
[24] from sklearn.feature_extraction. text import TfidfVectorizer, CountVectorizer

sns.set_style('whitegrid')
%matplotlib inline
stop=stop+['budget2023' , 'budget' , 'httpst' , '2023', 'modi' ,'nsitharaman' , 'union', 'pmindia' , 'tax' , 'india']
def plot_20_most_common_words(count_data, count_vectorizer) :
  import matplotlib. pyplot as plt
  words = count_vectorizer.get_feature_names()
  total_counts = np. zeros(len(words))
  for t in count_data:
    total_counts = t.toarray()[0]

  count_dict = (zip(words, total_counts))
  count_dict = sorted(count_dict, key=lambda x:x[1],reverse=True)[0:20]
  words = [w[0] for w in count_dict]
  counts = [w[1] for w in count_dict]
  x_pos = np.arange(len(words))

  plt.figure(2, (40,40))
  plt.subplot(title = '20 most common words')
  sns. set_context('notebook',font_scale=4,rc={ 'lines.linewidth' :2.5})
  sns.barplot(x_pos, counts, palette='husl')
  plt.xticks(x_pos, words, rotation=90)
  plt.xlabel('words')
  plt.ylabel('counts')
  plt.show()

count_vectorizer = CountVectorizer(stop_words=stop)
# Fit and transform the processed titles
count_data = count_vectorizer.fit_transform(df1['renderedContent'])
```



```
def get_top_n_bigram(corpus, n=None) :
  vec = CountVectorizer(ngram_range=(2, 4), stop_words='english').fit(corpus)
  bag_of_words = vec.transform(corpus)
  sum_words = bag_of_words.sum(axis=0)
  words_freq =[(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
  words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
  return words_freq[:n]

common_words = get_top_n_bigram(df1['renderedContent'] , 8)
mydict={}
for word, freq in common_words:
  bigram_df = pd.DataFrame(common_words,columns = ['ngram', 'count'])

bigram_df.groupby( 'ngram' ).sum()['count'].sort_values(ascending=False).sort_values().plot.barh(title = 'Top 8 bigrams',color='orange' , width=.4, figsize=(12,8),stacked = True)

<AxesSubplot:title={'center':'Top 8 bigrams'}, ylabel='ngram'>
```

```
[27] def get_subjectivity(text):
       return TextBlob(text).sentiment.subjectivity
     def get_polarity(text):
       return TextBlob(text).sentiment.polarity
```



```
     plt.show()
[43]

[43]

[44] #Creating the text variable
     negative_tw =" ".join(t for t in df_negative.renderedContent)
     # Creating word _ cloud with text as argument in . generate() method
     word_cloud2 = WordCloud(collocations = False, background_color = 'white') .generate(negative_tw)
     # Display the generated Word Cloud
     plt. imshow(word_cloud2, interpolation='bilinear')
     plt.axis('off')
     plt.show()

#Creating the text variable
neutral_tw =" ".join(t for t in df_neutral.renderedContent)
# Creating word _ cloud with text as argument in . generate() method
word_cloud2 = WordCloud(collocations = False, background_color = 'white') .generate(neutral_tw)
# Display the generated Word Cloud
plt. imshow(word_cloud2, interpolation='bilinear')
plt.axis('off')
plt.show()
```