covid19_tweets

February 12, 2021

1 Outliers - Project Fundamentals of Information System

1.1 Ngoc Diem Le, Marina Vicini, Francesco Vinci

1.1.1 ID: Covid19Tweets

Data: http://www.dei.unipd.it/~silvello/FIS2020/covid19Tweets.zip

The following notebook desires to analyze the data of tweets related to Covid-19. These tweets have been collected using Twitter API and a Python Script. Two main points will be implemented. The first is to analyze the trends of COVID-19 subjects following a temporal and spatial distribution that will be achieved through graphs and maps. The second is to investigate the sentiment of the tweets.

Our analysis will follow this structure:

- Cleaning Data
- Analyzing Trends
- Sentiment Analysis
- Dashboard
- References

Cleaning Data

In this first part we want to clean the dataset. We will first import the required libraries, and then we will look at the dataset. After removing unnecessary columns, we want to clean the data related to the dates, the text of the tweets and the location.

- Dates
- Texts
- Locations

[1]:
```
# import some useful libraries

import os
import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
# import data
df = pd.read_csv(r'data/covid19_tweets.csv')
```

[3]:
```
df.head()
```

[3]:
```
          user_name         user_location  \
0                       astroworld
1    Tom Basile           New York, NY
2  Time4fisticuffs      Pewee Valley, KY
3      ethel mertz  Stuck in the Middle
4         DIPR-J&K     Jammu and Kashmir

                                    user_description         user_created  \
0  wednesday addams as a disney princess keepin i...  2017-05-26 05:46:42
1  Husband, Father, Columnist & Commentator. Auth...  2009-04-16 20:06:23
2  #Christian #Catholic #Conservative #Reagan #Re...  2009-02-28 18:57:41
3  #Browns #Indians #ClevelandProud #[]_[] #Cavs ...  2019-03-07 01:45:06
4  Official Twitter handle of Department of Inf...  2017-02-12 06:45:15

   user_followers  user_friends  user_favourites  user_verified  \
0             624           950            18775          False
1            2253          1677               24           True
2            9275          9525             7254          False
3             197           987             1488          False
4          101009           168              101          False

                  date                                               text  \
0  2020-07-25 12:27:21  If I smelled the scent of hand sanitizers toda...
1  2020-07-25 12:27:17  Hey @Yankees @YankeesPR and @MLB - wouldn't it...
2  2020-07-25 12:27:14  @diane3443 @wdunlap @realDonaldTrump Trump nev...
3  2020-07-25 12:27:10  @brookbanktv The one gift #COVID19 has give me...
4  2020-07-25 12:27:08  25 July : Media Bulletin on Novel #CoronaVirus...

                           hashtags              source  is_retweet
0                               NaN   Twitter for iPhone       False
1                               NaN  Twitter for Android       False
2                       ['COVID19']  Twitter for Android       False
3                       ['COVID19']   Twitter for iPhone       False
4  ['CoronaVirusUpdates', 'COVID19']  Twitter for Android       False
```

In the dataset there are many columns that are not useful for our analysis. Therefore, we want to remove them.

[4]:
```
# delete useless columns

del df['user_created']
del df['user_followers']
```

```
del df['user_friends']
del df['user_favourites']
del df['source']
del df['is_retweet']
del df['user_description']
```

[5]:
```
df.head()
```

[5]:
```
          user_name         user_location  user_verified                 date  \
0                       astroworld          False  2020-07-25 12:27:21
1    Tom Basile           New York, NY           True  2020-07-25 12:27:17
2  Time4fisticuffs      Pewee Valley, KY          False  2020-07-25 12:27:14
3      ethel mertz  Stuck in the Middle          False  2020-07-25 12:27:10
4         DIPR-J&K     Jammu and Kashmir          False  2020-07-25 12:27:08

                                                text  \
0  If I smelled the scent of hand sanitizers toda...
1  Hey @Yankees @YankeesPR and @MLB - wouldn't it...
2  @diane3443 @wdunlap @realDonaldTrump Trump nev...
3  @brookbanktv The one gift #COVID19 has give me...
4  25 July : Media Bulletin on Novel #CoronaVirus...

                          hashtags
0                              NaN
1                              NaN
2                      ['COVID19']
3                      ['COVID19']
4  ['CoronaVirusUpdates', 'COVID19']
```

Clean the dates Since all the tweets have been collected in 2020, we will only consider the day and the month. Moreover, we can notice that the data has been collected in a period that spans from July 25th to August 29th.

[6]:
```
#df.date = pd.to_datetime(df.date).dt.date        # type of date is datetime.
 ↪date // format: yyyy-mm-dd
df["date"] = df["date"].apply(lambda x: x[5:10])  # date yyyy-mm-dd hh:mm:ss -->␣
 ↪mm-dd
```

[7]:
```
df.date.unique()
```

[7]:
```
array(['07-25', '07-24', '07-26', '07-27', '07-28', '07-29', '07-30',
       '07-31', '08-01', '08-02', '08-04', '08-06', '08-07', '08-08',
       '08-09', '08-10', '08-11', '08-12', '08-13', '08-14', '08-16',
       '08-17', '08-18', '08-22', '08-30', '08-29'], dtype=object)
```

[8]:
```
df.sort_values(by=['date'], inplace=True)   # sort the df by date
```

[9]:
```
df = df.reset_index(drop=True)
```

[10]:
```
df
```

[10]:

```
                user_name            user_location  user_verified   date  \
0         insightfultroll                      NaN          False  07-24
1       GlobalPandemic.NET                WORLDWIDE          False  07-24
2                Euan Watt      Manchester, England          False  07-24
3       GlobalPandemic.NET                WORLDWIDE          False  07-24
4                    Brian            New York, USA          False  07-24
...                    ...                      ...            ...    ...
179103       RAKAN Sarawak         Kuching, Sarawak          False  08-30
179104           b-yond tv                      NaN          False  08-30
179105  GlobalCapital Asia                     Asia          False  08-30
179106           SV News   Turn On Notifications           False  08-30
179107              Aarush                      NaN          False  08-30

                                                    text  \
0       I'm glad I'm in Canada, You can see why the pa...
1       ALERT: After Times investigation, Newsom says ...
2       Having checked the data of all nation's of the...
3       ALERT: Doctors Post Bikini Photos To Protest S...
4       @CNN But not a word about #COVID19 spread for ...
...                                                   ...
179103  WHO calls for protecting and preserving nature...
179104  #CultureFact: As the world adapts to #WorkFrom...
179105  Crisis Talk - with Thomas Hugger, CEO of Asia ...
179106  Massive anti-mask and anti-#COVID19 lockdown p...
179107  @cbseindia29 Sir We don't want compartment to ...

                                        hashtags
0                                            NaN
1                                            NaN
2                                            NaN
3                                            NaN
4                                  ['COVID19']
...                                          ...
179103                             ['COVID19']
179104  ['CultureFact', 'WorkFromHome', 'COVID19']
179105  ['covid19', 'investing', 'capitalmarkets']
179106                ['COVID19', 'Berlin']
179107                             ['COVID19']

[179108 rows x 6 columns]
```

Clean the texts We want to clean the text of the tweets, removing mentions, retweet count and hyperlinks.

[11]:
```
from nltk import sent_tokenize
df['text'] = df['text'].apply(sent_tokenize)    # create a list of sentences for␣
 ↪each tweet
```

[12]:
```
# clean sentence
def clean_sent(sent):
    sent = sent.lower()
    sent = re.sub('@[A-Za-z0-9]+', '', sent) # remove @mentions
    sent = re.sub('RT[\s]+', '', sent) # remove RT
    sent = re.sub('https?:\/\/\S+', '', sent) # remove hyperlink
    return sent

# clean the whole text
def clean_text(text):
    for i in range(len(text)):
        sent = clean_sent(text[i])
        text[i] = sent
    return text

df['text'] = df['text'].apply(clean_text)
```

[13]:
```
df.head()
```

[13]:

	user_name	user_location	user_verified	date	\
0	insightfultroll	NaN	False	07-24	
1	GlobalPandemic.NET	WORLDWIDE	False	07-24	
2	Euan Watt	Manchester, England	False	07-24	
3	GlobalPandemic.NET	WORLDWIDE	False	07-24	
4	Brian	New York, USA	False	07-24	

	text	hashtags
0	[i'm glad i'm in canada, you can see why the p...	NaN
1	[alert: after times investigation, newsom says...	NaN
2	[having checked the data of all nation's of th...	NaN
3	[alert: doctors post bikini photos to protest ...	NaN
4	[but not a word about #covid19 spread for the...	['COVID19']

Clean Locations We start by formatting the string in the column 'user_location'. We import the library geopy to convert the string into the corresponding country. Note that the computational cost of this step is high, so we saved the output into a dataset with cleaned data.

[14]:
```
import string

def clean_loc(loc):
    try:
        loc = loc.lower()
        loc = re.sub(r'[^\w\s]', '', loc) # remove punctuations
```

```
        loc = re.sub('https?:\/\/\S+', '', loc) # remove hyperlink
        loc = "".join([x for x in loc if x in string.printable]) # remove weird␣
↪character
        #loc = re.
↪sub('[(\U0001F600-\U0001F92F|\U0001F300-\U0001F5FF|\U0001F680-\U0001F6FF|\U0001F190-\U0001F1
↪# remove emoji
        return loc
    except AttributeError:
        return None

df["user_location"] = df["user_location"].apply(clean_loc)
```

[15]:
```
df.head()
```

[15]:
```
             user_name       user_location  user_verified   date  \
0     insightfultroll                None          False  07-24
1  GlobalPandemic.NET           worldwide          False  07-24
2           Euan Watt  manchester england          False  07-24
3  GlobalPandemic.NET           worldwide          False  07-24
4               Brian        new york usa          False  07-24

                                                text     hashtags
0  [i'm glad i'm in canada, you can see why the p...          NaN
1  [alert: after times investigation, newsom says...          NaN
2  [having checked the data of all nation's of th...          NaN
3  [alert: doctors post bikini photos to protest ...          NaN
4  [ but not a word about #covid19 spread for the...  ['COVID19']
```

[16]:
```
# !pip install geopy
```

[17]:
```
from geopy.geocoders import Nominatim
from geopy import exc

# create a geocoder
geolocator = Nominatim(user_agent = 'YOUR_EMAIL_HERE',timeout = 1)
```

[18]:
```
# create a function that gives me the country of the location
# the computational cost of this function is high !

def get_country(loc):

    if type(loc) != str:
        return None

    try:
        loc = geolocator.geocode(loc, language='en')           # loc =␣
↪Location(address, (long, lat))
```

```
    except exc.GeopyError:
        return None

    if type(loc) == type(None):
        return None

    loc = loc.address.split(", ")[-1]           # the country is at the end of␣
 ↪the address
    return loc
```

[19]:
```
# from tqdm import tqdm
# it takes almost 4 hours!

#dict_usloc = {raw_loc : get_country(raw_loc) for raw_loc in tqdm(pd.
 ↪unique(df["user_location"].dropna()))}
```

[20]:
```
# save the dict in a txt file: dict_usloc.txt

#with open('data/dict_usloc.txt', 'w') as outfile:
#    outfile.write(str(dict_usloc))
```

[21]:
```
#import ast

#file = open(r"data/dict_usloc.txt", "r")
#contents = file.read()
#dict_usloc = ast.literal_eval(contents)
```

[22]:
```
# change raw user_location to clean user_location in df

#def loc_raw_to_clean(lraw):

#    if type(lraw) != str:
#        return np.nan
#    return dict_usloc[lraw]

#df['user_location'] = df['user_location'].apply(loc_raw_to_clean)
```

[23]:
```
# save the clean DataFrame as csv file

#df.to_csv(r'df.csv', index= False)
```

[24]:
```
# load cleaned df

df = pd.read_csv('./data/df.csv')
df.head()
```

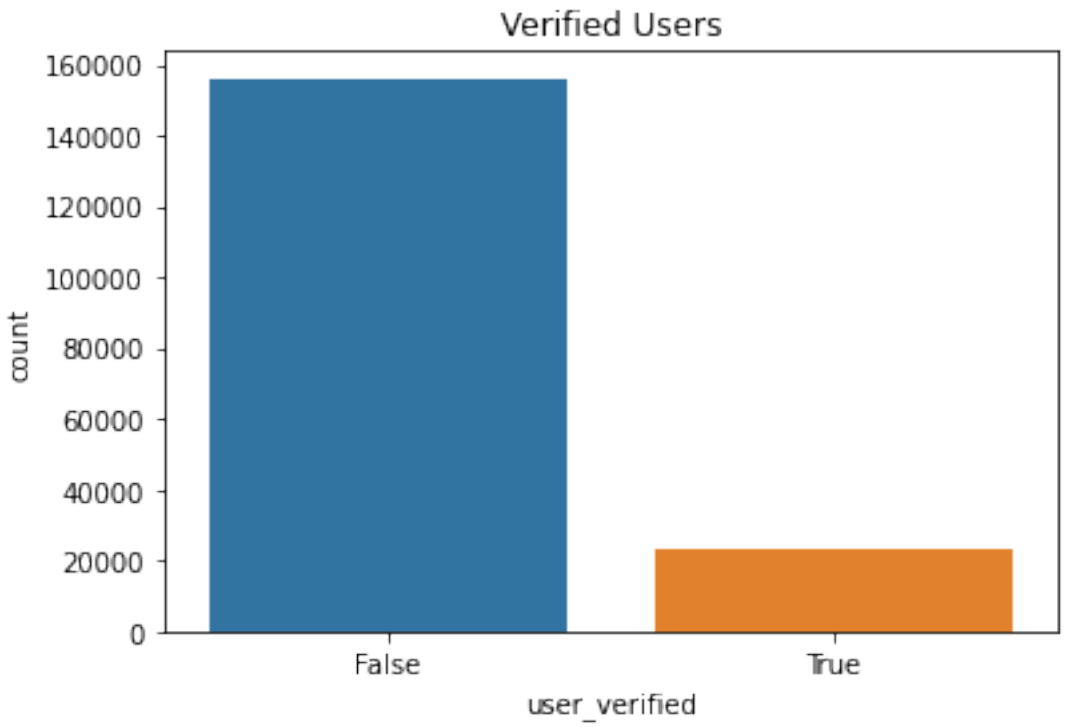
[24]:
```
             user_name   user_location  user_verified   date  \
0     insightfultroll              NaN          False  07-24
1  GlobalPandemic.NET  United Kingdom          False  07-24
2           Euan Watt  United Kingdom          False  07-24
3  GlobalPandemic.NET  United Kingdom          False  07-24
4               Brian   United States          False  07-24

                                                text     hashtags
0  ["i'm glad i'm in canada, you can see why the ...          NaN
1  ['alert: after times investigation, newsom say...          NaN
2  ["having checked the data of all nation's of t...          NaN
3  ['alert: doctors post bikini photos to protest...          NaN
4  [' but not a word about #covid19 spread for th...  ['COVID19']
```

[25]:
```
import ast

df['text'] = df.text.apply(ast.literal_eval)
```

Users verified We want to check how many users are verified, and we want to study the DataFrame by user (verified or not), by dates and by user_location.

[26]:
```
sns.countplot(df['user_verified'])
plt.title('Verified Users')
plt.show()
```

```
/home/francesco/.local/lib/python3.8/site-packages/seaborn/_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```



[27]:
```
def df_usver(usver=True):
    if usver:
        return df[df['user_verified']]
    else:
        return df[~df['user_verified']]
```

[28]:
```
def df_country(country):
    if country not in list(df.user_location):
        raise NameError(f"{country} is not in df['user_location']")
    else:
        return df[df.user_location == country]
```

[29]:
```
def df_dates(d1,d2):
    if re.fullmatch("[0-1][0-9]-[0-3][0-9]",d1) and re.
 ↪fullmatch("[0-1][0-9]-[0-3][0-9]",d2):
        return df[(df.date >= d1) & (df.date <= d2)]
    raise NameError("Dates have to been: 'dd-mm'")
```

Analyzing Trends After an intial cleaning part, we want to analyze the trends of the tweets. We will start by highliting: * The most used words * The frequency of the hashtags * The number of tweets

Most used words In this section, we want to identify the words with the most recurrence in the text of the tweets. We start by tokenizing the text and removing the stopwords. Then we want

to analyze the most used words by users that have been verified and not.

[30]:
```
# tokenization

from nltk.tokenize import TweetTokenizer
tokenizer = TweetTokenizer()

df['text_tokenized'] = df['text'].apply(lambda x: tokenizer.tokenize(" ".join(x).
 ↪lower()))
```

[31]:
```
df.head()
```

[31]:
```
             user_name   user_location  user_verified   date  \
0     insightfultroll              NaN          False  07-24
1  GlobalPandemic.NET  United Kingdom          False  07-24
2           Euan Watt  United Kingdom          False  07-24
3  GlobalPandemic.NET  United Kingdom          False  07-24
4               Brian   United States          False  07-24

                                                text     hashtags  \
0  [i'm glad i'm in canada, you can see why the p...          NaN
1  [alert: after times investigation, newsom says...          NaN
2  [having checked the data of all nation's of th...          NaN
3  [alert: doctors post bikini photos to protest ...          NaN
4  [ but not a word about #covid19 spread for the...  ['COVID19']

                                      text_tokenized
0  [i'm, glad, i'm, in, canada, ,, you, can, see,...
1  [alert, :, after, times, investigation, ,, new...
2  [having, checked, the, data, of, all, nation's...
3  [alert, :, doctors, post, bikini, photos, to, ...
4  [but, not, a, word, about, #covid19, spread, f...
```

[32]:
```
#nltk.download('stopwords')
```

[33]:
```
# remove stopwords
import nltk
from nltk.corpus import stopwords

stopword = nltk.corpus.stopwords.words('english')

def remove_stopwords(text):
    text = [word for word in text if word not in stopword and word.isalnum() and␣
 ↪not word.isnumeric()]
    return text

df['text_tokenized'] = df['text_tokenized'].apply(remove_stopwords)
```

[34]:
```
df.head()
```

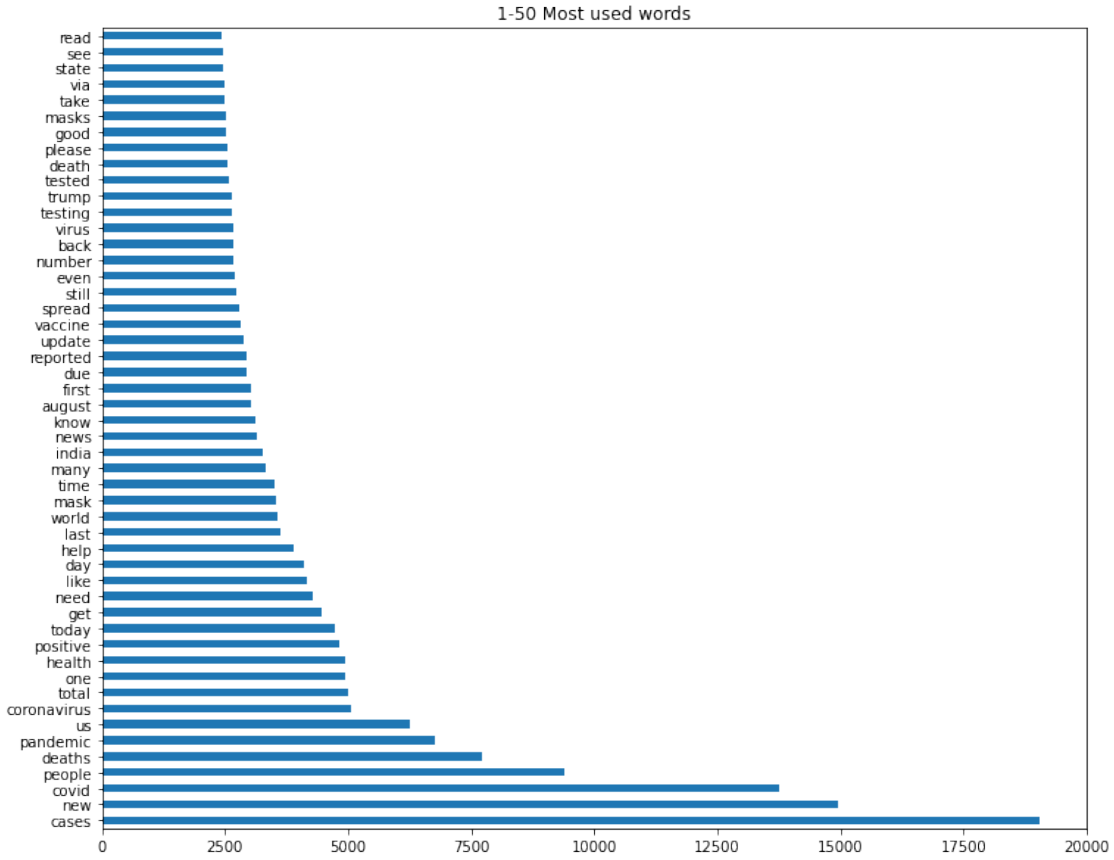
[34]:
```
            user_name   user_location  user_verified   date  \
0     insightfultroll             NaN          False  07-24
1  GlobalPandemic.NET  United Kingdom          False  07-24
2           Euan Watt  United Kingdom          False  07-24
3  GlobalPandemic.NET  United Kingdom          False  07-24
4               Brian   United States          False  07-24

                                                text     hashtags  \
0  [i'm glad i'm in canada, you can see why the p...          NaN
1  [alert: after times investigation, newsom says...          NaN
2  [having checked the data of all nation's of th...          NaN
3  [alert: doctors post bikini photos to protest ...          NaN
4  [ but not a word about #covid19 spread for the...  ['COVID19']

                                     text_tokenized
0  [glad, canada, see, pandemic, raging, united, ...
1  [alert, times, investigation, newsom, says, nu...
2              [checked, data, uk, deaths, england]
3  [alert, doctors, post, bikini, photos, protest...
4  [word, spread, last, days, portland, city, mas...
```

[35]:
```
from nltk import FreqDist

def freq_words(data):
    list_of_all_words = []
    for l in data['text_tokenized'].values:
        list_of_all_words.extend(l)
    fdist_words = FreqDist(list_of_all_words)
    count_words = pd.Series(fdist_words.values(), index = fdist_words.keys())
    count_words = count_words.sort_values(ascending = False)
    return count_words
```

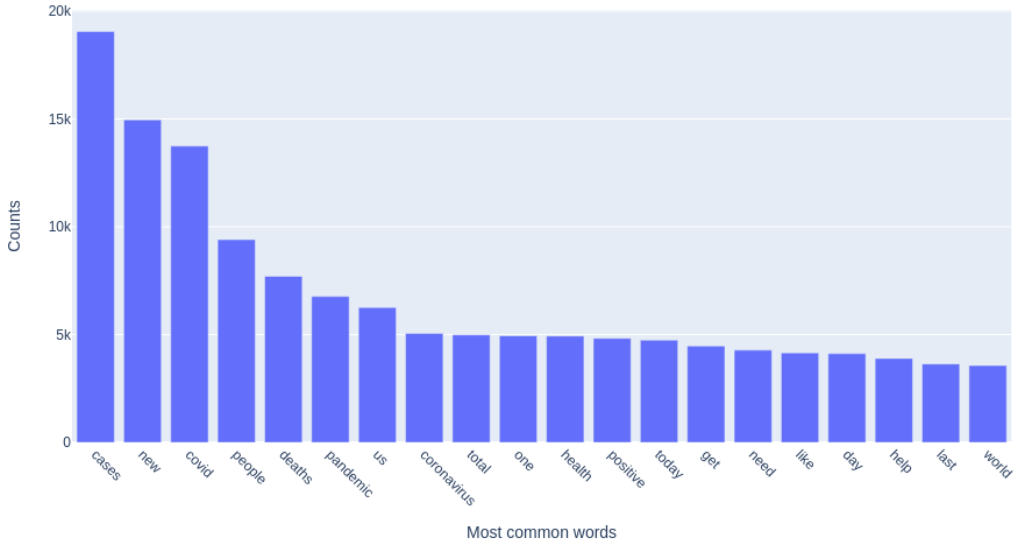
[36]:
```
count_words = freq_words(df)
```

[37]:
```
def barplt_count_words(n,m):
    plt.figure(figsize = (12,10))
    plt.title(f'{n+1}-{m} Most used words')
    count_words[n:m].plot.barh(count_words)
```

[38]:
```
barplt_count_words(0,50)
```



[39]:
```
# Usefull for the final Dashboard
# Barplot for frequency of words
import plotly.express as px

fig_frq_words = px.bar(count_words[0:20],
                       labels = {'index':'Most common words','value':'Counts'})
fig_frq_words.update_layout(showlegend = False)
fig_frq_words.update_xaxes(tickangle=45)
```



Most used words by verified users and not verified

[40]:
```
count_words_usver = freq_words(df_usver())
```

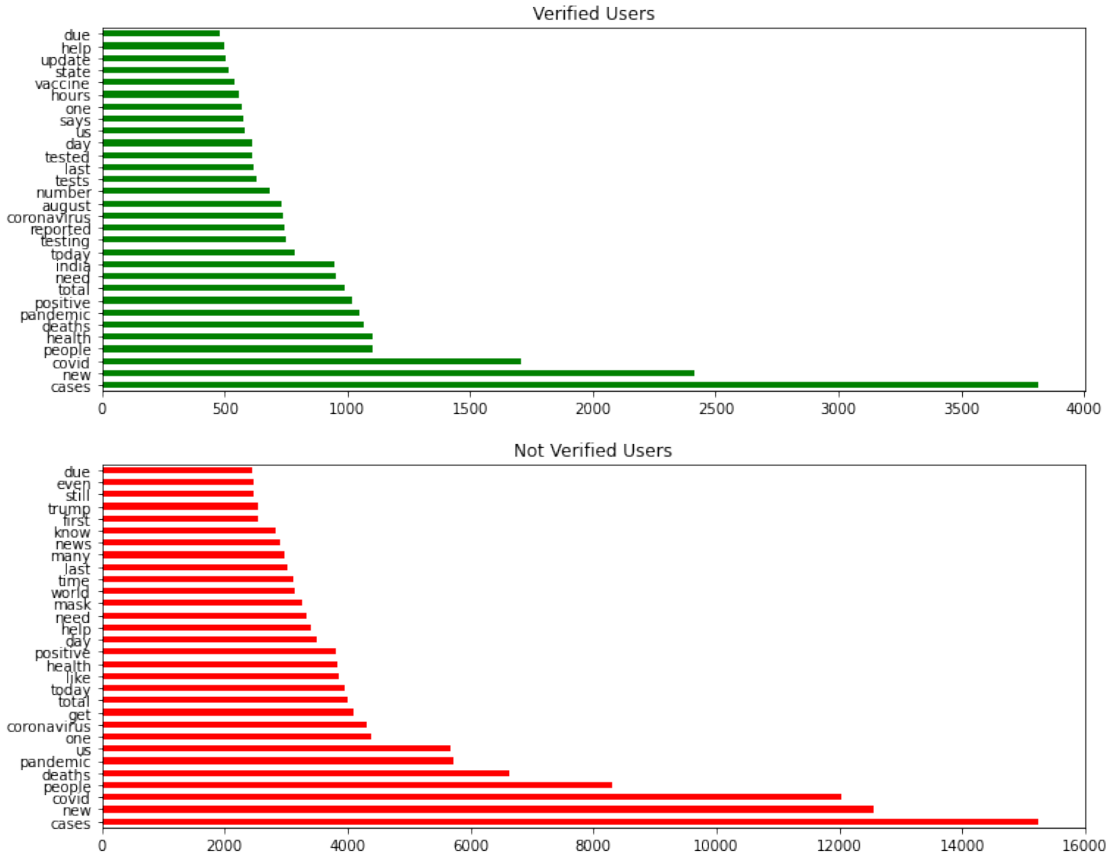
[41]:
```
#count_words_usver[:50]
```

[42]:
```
count_words_usnotver = freq_words(df_usver(False))
```

[43]:
```
#count_words_usnotver[:50]
```

[44]:
```
fig, axes = plt.subplots(2, 1, figsize = (12,10))
count_words_usver[0:30].plot.barh(ax=axes[0], color='g', title = 'Verified␣
↪Users')
count_words_usnotver[0:30].plot.barh(ax=axes[1], color='r', title = 'Not␣
↪Verified Users')
```

[44]:
```
<AxesSubplot:title={'center':'Not Verified Users'}>
```



Frequency of the hashtags We will analyze the frequency of the hashtags through a Word Cloud, with the help of a bar race chart.

[45]:
```
from nltk import FreqDist

def freq_hashtag(data):
    hashtags1 = []
    for x in list(data['hashtags'].dropna()):
        hashtags1.extend(ast.literal_eval(x))  # x is a str object like␣
↪'['hashtag1','hashtag2']'
    hashtags = [x.lower() for x in hashtags1]  # list of all hashtags
    fdist = FreqDist(hashtags)                 # frequencies of hashtags
    return fdist, hashtags

fdist, hashtags = freq_hashtag(df)
```

[46]:
```
fdist
```

[46]:
```
FreqDist({'covid19': 100312, 'coronavirus': 10197, 'pandemic': 1625, 'covid':
1299, 'india': 1193, 'corona': 1162, 'trump': 1101, 'lockdown': 963,
```

```
'coronaviruspandemic': 882, 'covid_19': 828, ...})
```

[47]:
```
def freq_hashtag_nocov(hashtags):
    hashtags_nocovid=[]
    for w in hashtags:
        if not(re.search("covid",w.lower()) or re.search("corona",w.lower())):
            hashtags_nocovid.append(w)
    fdist_nocovid = FreqDist(hashtags_nocovid)
    return fdist_nocovid

fdist_nocovid = freq_hashtag_nocov(hashtags)
```

[48]:
```
l = dict()
for d in df.date.unique():
    l[d] = freq_hashtag_nocov(freq_hashtag(df[df.date == d])[1])
```

[49]:
```
count_words_days = pd.DataFrame(l).T.fillna(0).cumsum()
```

[50]:
```
count_words_days.head()
```

[50]:

	losangeles	newyork	washington	neworleans	italia	china	spain	\
07-24	1.0	2.0	1.0	1.0	1.0	2.0	1.0	
07-25	6.0	13.0	1.0	1.0	3.0	45.0	8.0	
07-26	8.0	16.0	3.0	2.0	4.0	73.0	30.0	
07-27	10.0	24.0	6.0	5.0	6.0	110.0	38.0	
07-28	15.0	26.0	8.0	7.0	8.0	153.0	44.0	

	japan	southkorea	california	...	wi	pandemichoax	\
07-24	2.0	1.0	2.0	...	0.0	0.0	
07-25	7.0	6.0	19.0	...	0.0	0.0	
07-26	11.0	13.0	25.0	...	0.0	0.0	
07-27	19.0	24.0	34.0	...	0.0	0.0	
07-28	29.0	28.0	41.0	...	0.0	0.0	

	distractorandchief	trumpisdangerous	quazergraph	doccom	\
07-24	0.0	0.0	0.0	0.0	
07-25	0.0	0.0	0.0	0.0	
07-26	0.0	0.0	0.0	0.0	
07-27	0.0	0.0	0.0	0.0	
07-28	0.0	0.0	0.0	0.0	

	masonstreasurehunt	oakdalecalifornia	cakepopmaker	culturefact
07-24	0.0	0.0	0.0	0.0
07-25	0.0	0.0	0.0	0.0
07-26	0.0	0.0	0.0	0.0
07-27	0.0	0.0	0.0	0.0
07-28	0.0	0.0	0.0	0.0

```
[5 rows x 36544 columns]
```

[51]:
```
#!pip3 install bar_chart_race
```

[52]:
```
# Supress RuntimeWarning
import warnings
warnings.filterwarnings("ignore", category=RuntimeWarning)      # many␣
 ↪RuntimeWarning

import bar_chart_race as bcr
# This computes in about 4/5 minutes!!
bcr.bar_chart_race(count_words_days,
                   n_bars= 20,
                   #shared_fontdict={'family' : 'Helvetica', 'color' : '.1'},
                   filter_column_colors= True)
```

```
/home/francesco/.local/lib/python3.8/site-
packages/bar_chart_race/_make_chart.py:286: UserWarning:

FixedFormatter should only be used together with FixedLocator

/home/francesco/.local/lib/python3.8/site-
packages/bar_chart_race/_make_chart.py:287: UserWarning:

FixedFormatter should only be used together with FixedLocator
```

[52]:
```
<IPython.core.display.HTML object>
```

[53]:
```
#!pip3 install wordcloud
```

[54]:
```
from wordcloud import WordCloud
```

[55]:
```
def hashtag_wordcloud(fdist):
    wc = WordCloud(width=800, height=400, max_words=50).
 ↪generate_from_frequencies(fdist)  # create the wordcloud
    plt.figure(figsize=(12,10))
    plt.imshow(wc, interpolation="bilinear")
    plt.axis("off")
    plt.show()
```

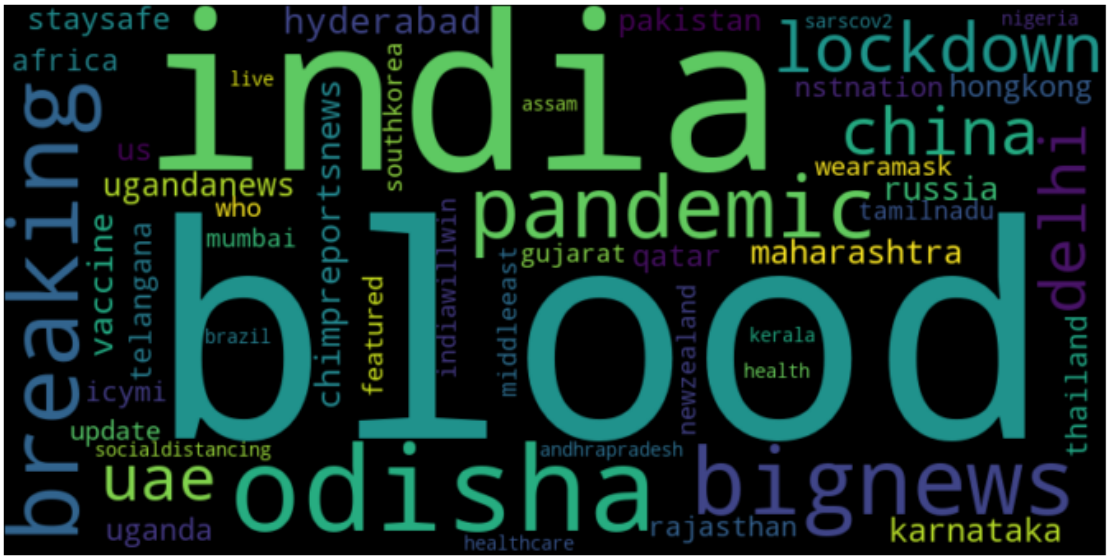
[56]:
```
hashtag_wordcloud(fdist)
```

Let us remove the predominant words “covid” and “corona”, and let us plot again another word-cloud.

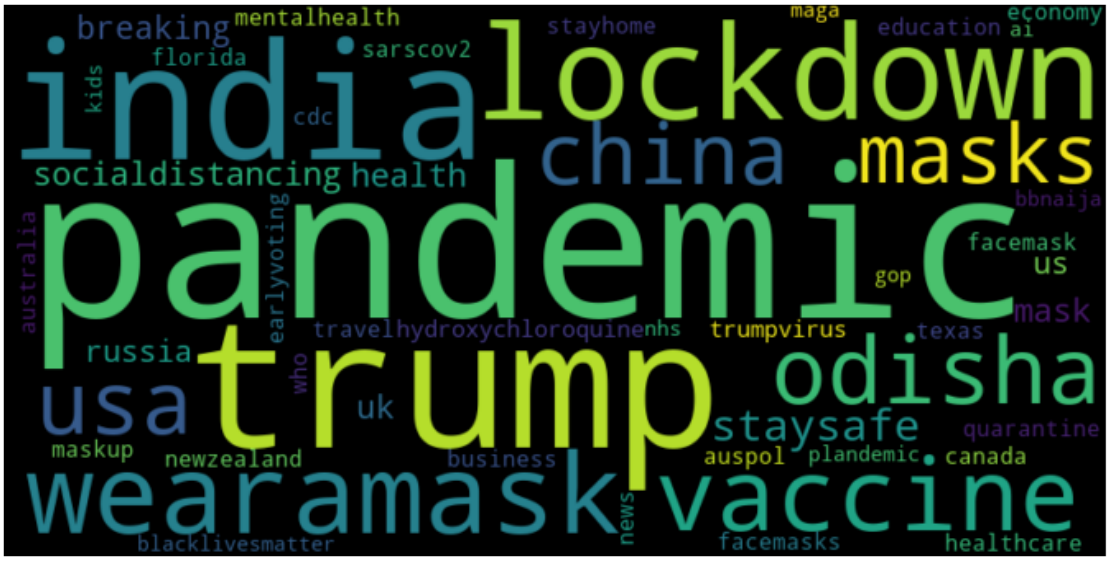
[57]:
```
hashtag_wordcloud(fdist_nocovid)
```

As we have done before, let us plot the wordcloud, classifying them for verification of the user.

[58]:
```
# wordcloud for verified user
hashtag_wordcloud(freq_hashtag_nocov(freq_hashtag(df_usver())[1]))
```

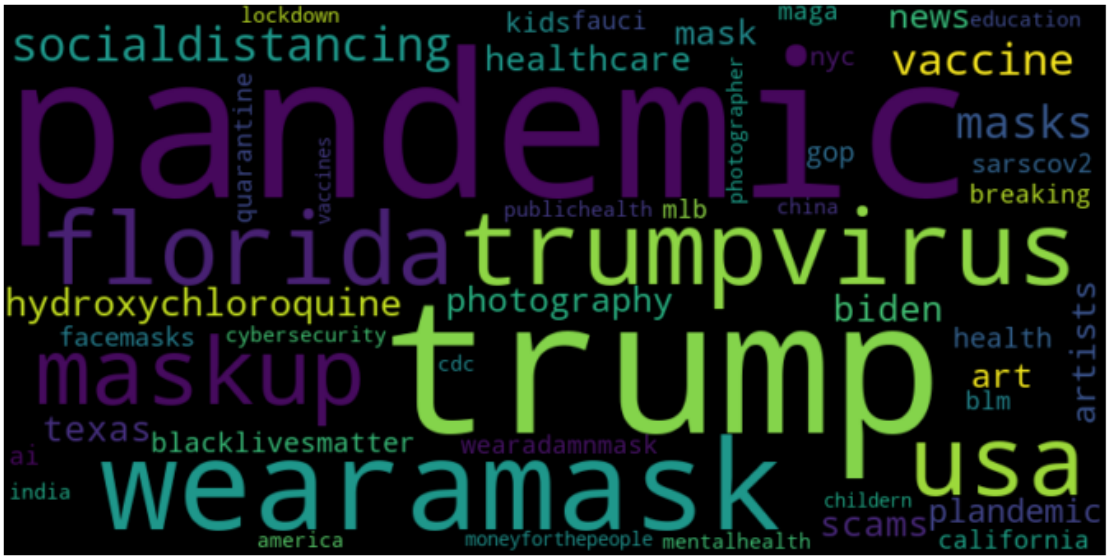


[59]:
```
# wordcloud for not verified user
hashtag_wordcloud(freq_hashtag_nocov(freq_hashtag(df_usver(False))[1]))
```

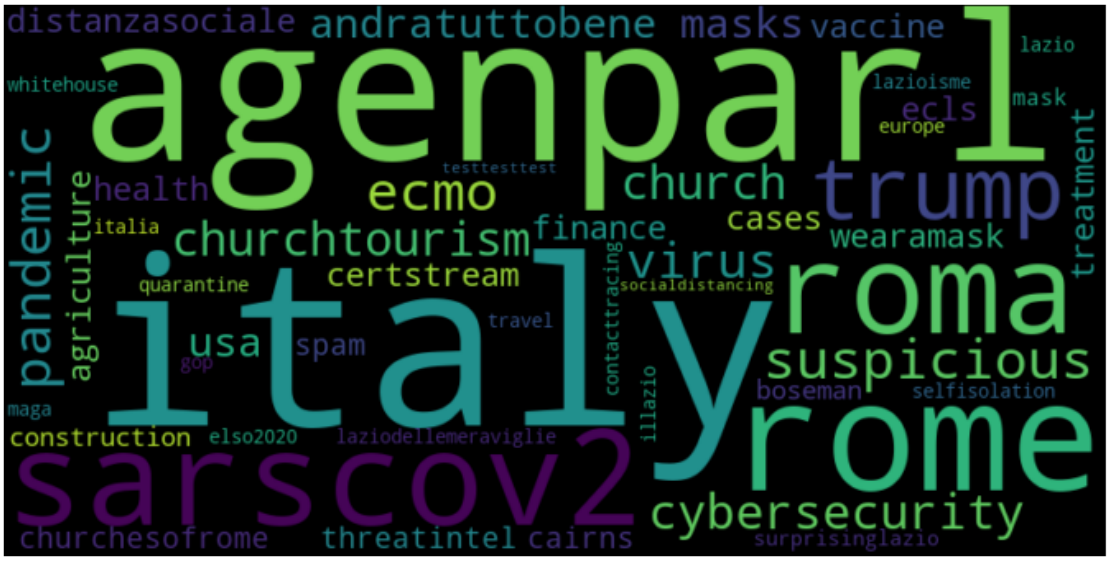


We can also plot a wordcloud following a geographical point of here. Here below, the wordcloud of the most used hashtags are found considering the United States of America and Italy.

[60]:
```
hashtag_wordcloud(freq_hashtag_nocov(freq_hashtag(df_country('United␣
↪States'))[1]))
```



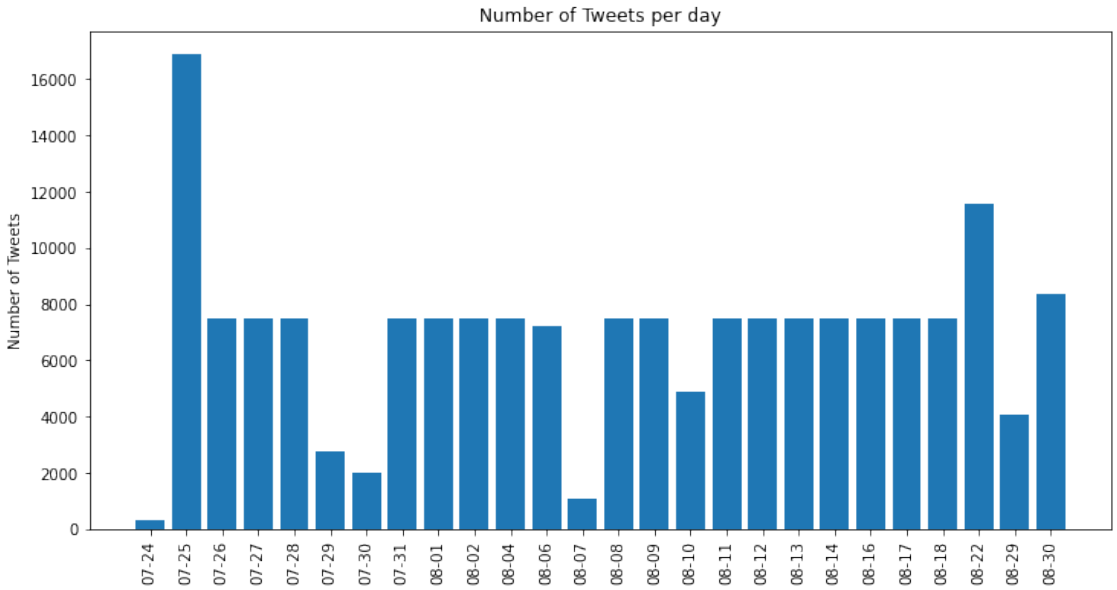
[61]:
```
hashtag_wordcloud(freq_hashtag_nocov(freq_hashtag(df_country('Italy'))[1]))
```



Number of Tweets Now we want to analyze the number of tweets that have been collected. We will first plot them through a bar graph, and then we will plot another bar graph that takes into account that no tweets have been retreived in those days.

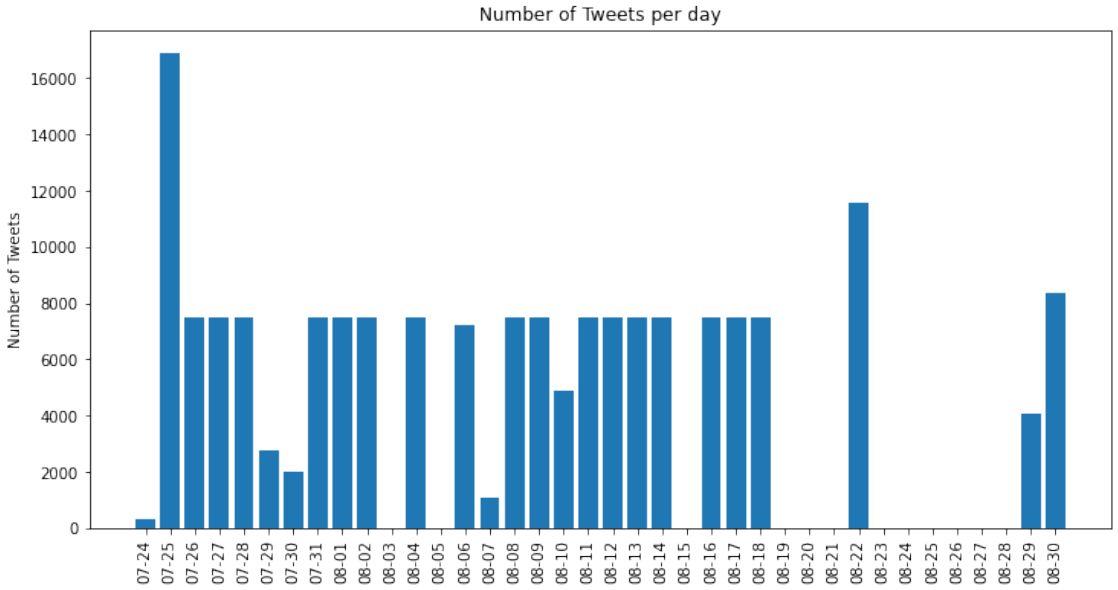
[62]:
```
num_tweet_per_day = {}
for day in df.date.unique():
    num_tweet_per_day[day] = df[df.date == day]['date'].count()
```

[63]:
```
plt.figure(figsize= [12,6])
plt.bar(list(num_tweet_per_day.keys()), list(num_tweet_per_day.values()))
_=plt.xticks(list(num_tweet_per_day.keys()), rotation='vertical')
_=plt.ylabel('Number of Tweets')
_=plt.title('Number of Tweets per day')
```

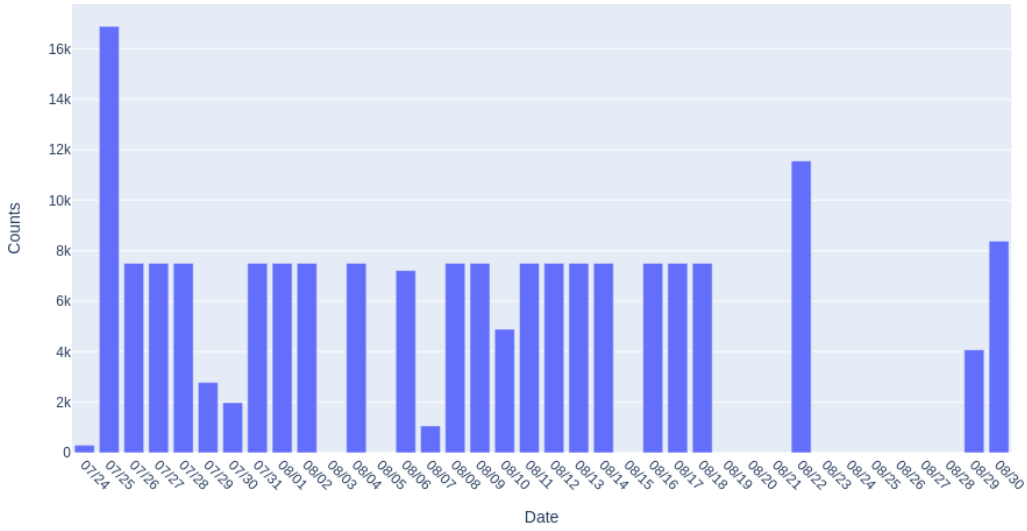


[64]:
```
# Here there are days with no tweets. I check if this is correct in the original␣
 ↪data
plt.figure(figsize= [12,6])

num_tweet_per_day_missing_col = []

#There are all the days including the ones without values
#days = pd.to_datetime(df_countries_t.columns, format = "%m-%d")
days = pd.date_range('07-24-2020', '08-30-2020')
days = [str(day)[5:10] for day in days]

for day in days:
    if day in df.date.values:
        num_tweet_per_day_missing_col.append(num_tweet_per_day[day])
    if day not in df.date.values:
        num_tweet_per_day_missing_col.append(0)


fig_number_of_tweets_per_day = plt.bar(days, num_tweet_per_day_missing_col)
_=plt.xticks(days, rotation='vertical')
_=plt.ylabel('Number of Tweets')
```

```
_=plt.title('Number of Tweets per day')
plt.show()
```



[65]:
```
# Usefull for the final Dashboard
# Barplot for number of tweets per day

days_with_missing_value = []
for day in days:
    days_with_missing_value.append(day.replace('-','/'))

dataframe = pd.DataFrame(list(zip(days_with_missing_value,␣
 ↪num_tweet_per_day_missing_col)), columns = ['Date', 'Counts'])

fig_number_of_tweets_per_day = px.bar(dataframe, x=dataframe['Date'],␣
 ↪y=dataframe['Counts'])
fig_number_of_tweets_per_day.update_xaxes(title_text='Date')
fig_number_of_tweets_per_day.update_yaxes(title_text='Counts')
fig_number_of_tweets_per_day.update_layout(showlegend = False)
fig_number_of_tweets_per_day.update_xaxes(tickangle=45)
```



[66]:
```
country_count_tweets = df.pivot_table(index = 'user_location', columns='date',␣
↪values = 'text', aggfunc = 'count', fill_value = 0).T
country_cumucount_tweets = country_count_tweets.cumsum()
```

[67]:
```
country_cumucount_tweets
```

[67]:

user_location	Afghanistan	Africa	Albania	Algeria	Andorra	Angola	\
date							
07-24	0	0	0	0	0	0	
07-25	9	20	4	1	0	0	
07-26	18	34	5	1	1	0	
07-27	21	38	6	1	1	0	
07-28	27	47	7	2	1	0	
07-29	27	51	7	2	1	0	
07-30	27	53	7	2	1	0	
07-31	27	60	12	4	1	0	
08-01	27	67	12	4	1	0	
08-02	31	72	13	4	1	0	
08-04	33	83	15	4	1	0	
08-06	38	91	15	4	1	0	
08-07	38	93	15	5	1	0	
08-08	41	105	17	5	1	0	
08-09	45	119	19	6	1	0	
08-10	47	122	21	7	1	0	
08-11	51	141	24	8	1	0	
08-12	55	159	28	9	1	0	
08-13	56	185	33	9	1	0	

```
08-14                   56      196        35         9         1        0
08-16                   63      205        39         9         1        0
08-17                   66      213        40        10         1        1
08-18                   66      215        43        10         1        2
08-22                   72      227        48        10         1        2
08-29                   72      227        49        10         1        2
08-30                   73      232        49        10         1        2

user_location  Antarctica  Argentina  Armenia  Asia  ...  United Kingdom  \
date                                                ...
07-24                   0          0        0     0  ...              15
07-25                   2          9        6     3  ...            1722
07-26                   4         11        8     5  ...            2355
07-27                   5         13        9     7  ...            2720
07-28                   5         16       10    10  ...            3021
07-29                   6         16       12    10  ...            3233
07-30                   7         17       12    10  ...            3375
07-31                   8         24       13    12  ...            3853
08-01                   8         27       14    13  ...            4392
08-02                   9         31       14    13  ...            4924
08-04                   9         35       17    14  ...            5409
08-06                   9         44       18    14  ...            5995
08-07                   9         44       19    14  ...            6082
08-08                   9         47       23    16  ...            6899
08-09                   9         48       27    17  ...            7538
08-10                   9         48       29    18  ...            7988
08-11                   9         49       32    20  ...            9233
08-12                   9         50       44    22  ...           10509
08-13                   9         55       48    24  ...           11957
08-14                   9         59       50    27  ...           12350
08-16                  10         65       52    29  ...           12957
08-17                  10         69       55    33  ...           13601
08-18                  11         70       57    34  ...           14262
08-22                  12         78       58    36  ...           15420
08-29                  12         80       59    36  ...           15728
08-30                  12         84       60    40  ...           16389

user_location  United States  Uruguay  Uzbekistan  Vatican City  Venezuela  \
date
07-24                    128        0           0             0          0
07-25                   3667        1           1             2          2
07-26                   5001        2           1             2          3
07-27                   6514        4           1             2          4
07-28                   8169        6           2             2          4
07-29                   9300        6           3             2          4
07-30                  10152        6           3             2          4
07-31                  13333        6           3             2          5
```

08-01	16099	7	3	2	8
08-02	18675	8	3	2	8
08-04	20025	8	4	2	9
08-06	22869	8	6	2	9
08-07	23199	8	6	2	9
08-08	25010	8	7	2	9
08-09	26144	10	7	2	9
08-10	27975	10	7	3	9
08-11	28581	10	7	3	9
08-12	29189	11	8	3	9
08-13	29827	11	8	3	9
08-14	31588	12	9	3	9
08-16	33100	12	9	4	10
08-17	34092	12	9	5	10
08-18	36855	15	10	5	11
08-22	39074	16	10	6	11
08-29	40488	16	10	6	11
08-30	42256	16	10	6	11

user_location	Vietnam	Yemen	Zambia	Zimbabwe
date				
07-24	0	0	0	0
07-25	16	0	6	19
07-26	22	0	10	26
07-27	28	0	12	45
07-28	40	0	12	59
07-29	41	0	12	61
07-30	41	0	12	62
07-31	44	0	13	66
08-01	47	0	14	68
08-02	48	2	15	70
08-04	61	2	17	95
08-06	61	2	17	100
08-07	62	2	17	100
08-08	70	2	20	108
08-09	80	2	24	116
08-10	83	3	26	119
08-11	88	4	30	128
08-12	98	6	37	141
08-13	106	7	40	162
08-14	114	7	42	169
08-16	126	8	45	173
08-17	137	9	47	183
08-18	139	11	49	184
08-22	146	12	52	198
08-29	146	12	52	198
08-30	150	13	53	205

```
[26 rows x 207 columns]
```

[68]:
```
bcr.bar_chart_race(country_cumucount_tweets,
                   title = 'Number of Tweets for country',
                   n_bars= 10,
                   shared_fontdict={'family' : 'Helvetica', 'color' : '.1'},
                   filter_column_colors= True)
```

```
/home/francesco/.local/lib/python3.8/site-
packages/bar_chart_race/_make_chart.py:286: UserWarning:

FixedFormatter should only be used together with FixedLocator

/home/francesco/.local/lib/python3.8/site-
packages/bar_chart_race/_make_chart.py:287: UserWarning:

FixedFormatter should only be used together with FixedLocator
```

[68]:
```
<IPython.core.display.HTML object>
```

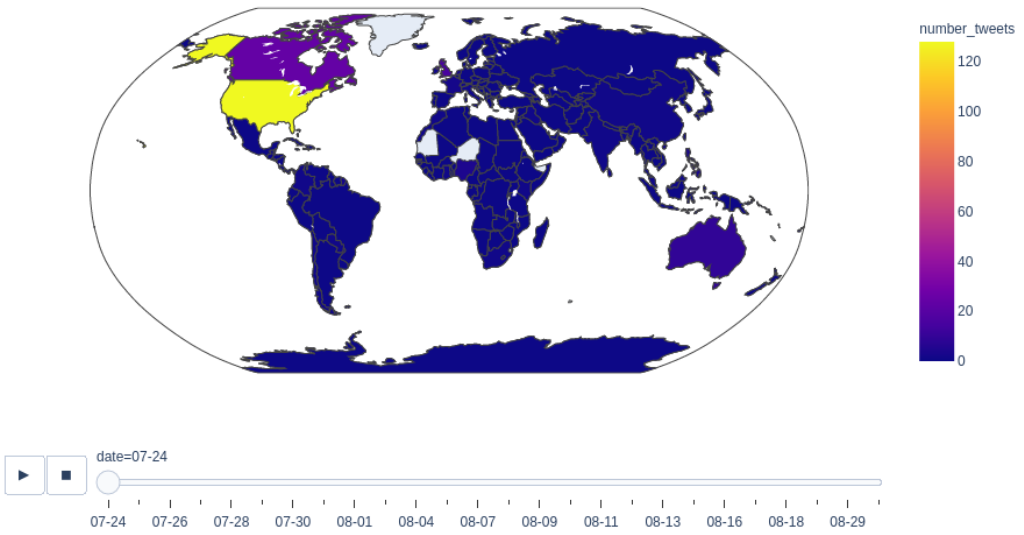
[69]:
```
countries = []
for country in country_count_tweets.columns:
    countries.extend([country]*len(list(df.date.unique())))
```

[70]:
```
dates = list(df.date.unique())*len(pd.Series(countries).unique())
```

[71]:
```
number_tweets = []
for country in pd.Series(countries).unique():
    number_tweets.extend(list(country_count_tweets.loc[:,country].values))
```

[72]:
```
number_of_tweets_per_country = pd.DataFrame({'user_location' :countries,'date':␣
 ↪dates,'number_tweets':number_tweets})
```

[73]:
```
# animation number of tweets
figure_number_of_tweets_per_country = px.choropleth(number_of_tweets_per_country,
                                                    color = 'number_tweets',
                                                    locations = "user_location",
                                                    locationmode = 'country␣
 ↪names',
                                                    animation_frame = 'date',
                                                    projection = 'robinson',
                                                    basemap_visible = True)
figure_number_of_tweets_per_country.show()
```



Sentiment Analysis

In this third section, we want to analyze the sentiment analysis for the texts of the tweets. For each tweet we compute the sentiment score and we save it in a new column called compound. Then we plot the distribution of the sentiments.

- Sentiment Analysis per Country
- Clusters of Countries
- Sentiment Analysis Spatial Temporal distribution

[74]:
```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
```

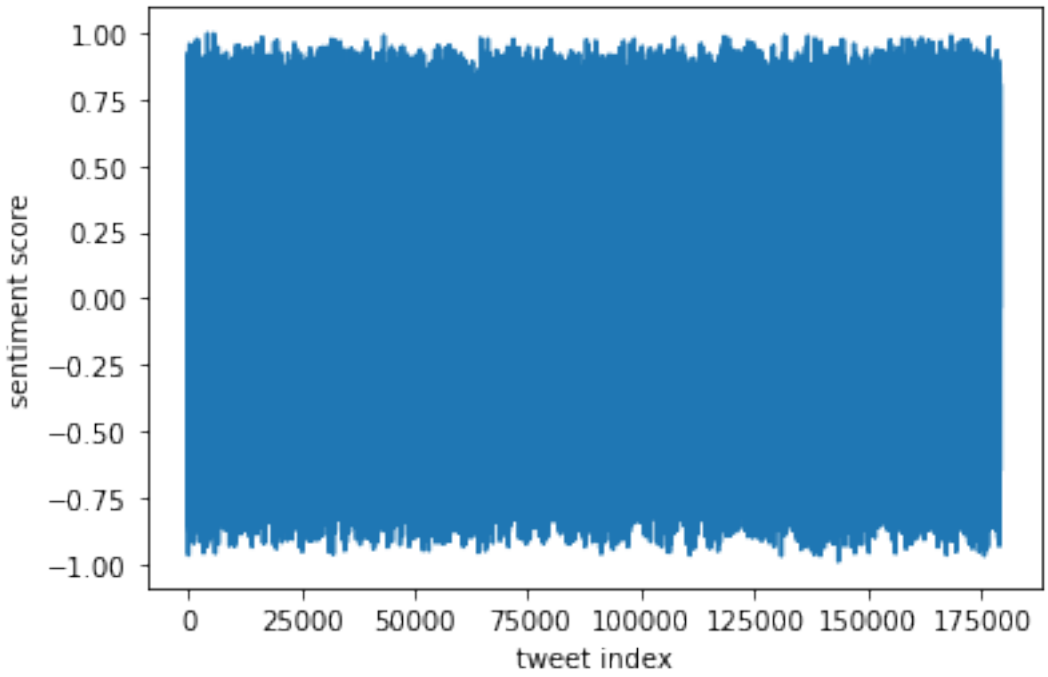
[75]:
```
# create a sentiment analyser
analyzer = SentimentIntensityAnalyzer()
```

[76]:
```
tweets_compound = [] # list of tweets evaluation sentiment  [-1=neg,1=pos]
for text in df.text:
    curr_twt_comp=[]
    for tweet in text:
        sentiment_map = analyzer.polarity_scores(tweet)        # sentiment map:␣
 ↪dict keys:"pos","compound","neg"
        curr_twt_comp.append(sentiment_map['compound'])
    tweets_compound.append(np.array(curr_twt_comp).mean())

df["compound"] = tweets_compound
```

[77]:
```
sum(df.compound == 0)
```

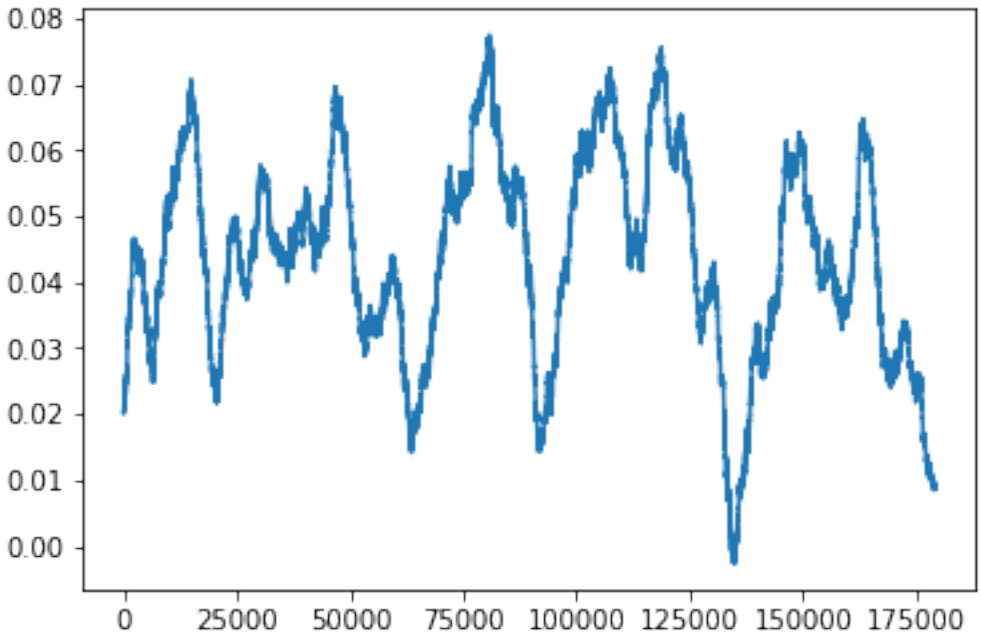
[77]: 59834

[78]:
```
plt.figure()
plt.plot(df.index, df.compound)
plt.ylabel('sentiment score')
plt.xlabel('tweet index')
plt.show()
```



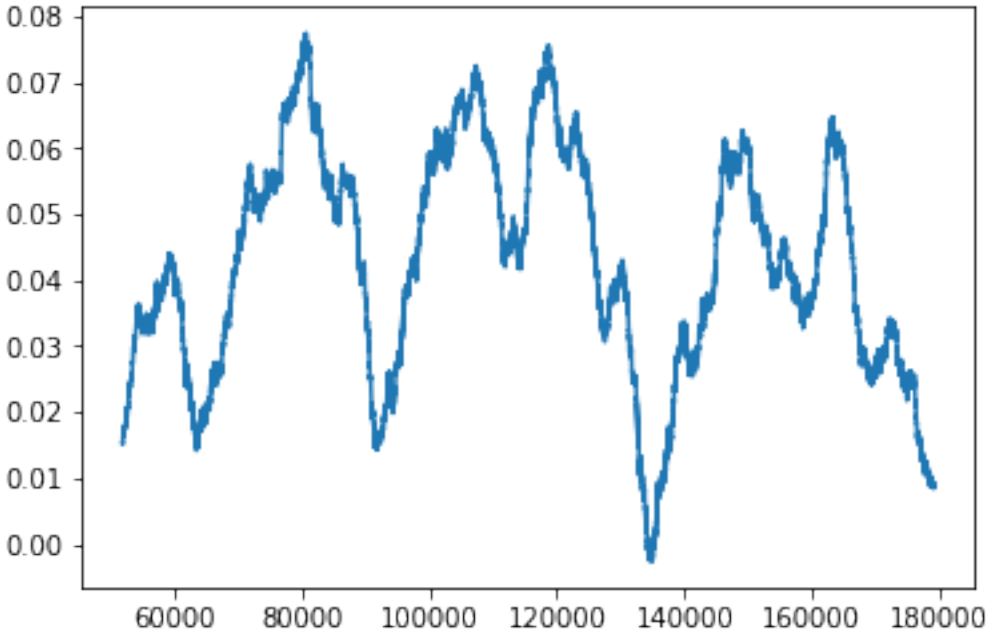
In the plot just above we can see that the plot is not informative at all. Therefore, we want to smooth it.

[79]:
```
def smooth(y, box_pts):
    box = np.ones(box_pts)/box_pts
    # generate values by taking the sum of the products of values within the␣
 ↪input arch and the other signal (ones)
    y_smooth = np.convolve(y, box, mode='same')
    return y_smooth

def plt_compound(data):
    smoothed_signals = smooth(list(data.compound), 4500)
    plt.figure()
    plt.plot(data.index, smoothed_signals)
    plt.show()

plt_compound(df)
```



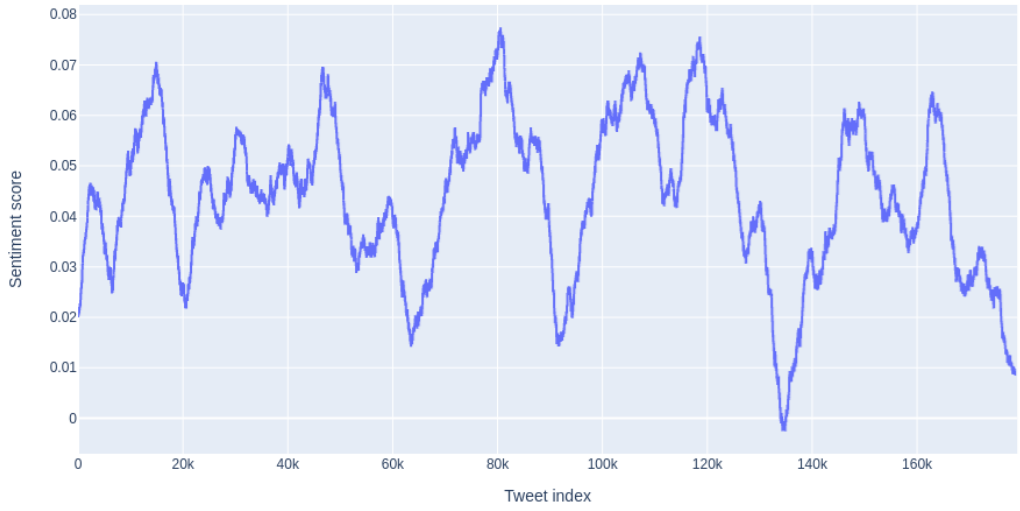
We can also consider a portion of the tweets. For example, in the plot below we want to plot the sentiment score of a certain range of dates.

[80]:
```
plt_compound(df_dates('08-01','09-01'))
```



Below, we see an interactive version of the plot above.

[81]:
```
# Usefull for the final Dashboard

smoothed_signals = smooth(list(df.compound), 4500)
fig_compound = px.line(df, x= df.index, y = smoothed_signals)
fig_compound.update_xaxes(title_text='Tweet index')
fig_compound.update_yaxes(title_text='Sentiment score')
```

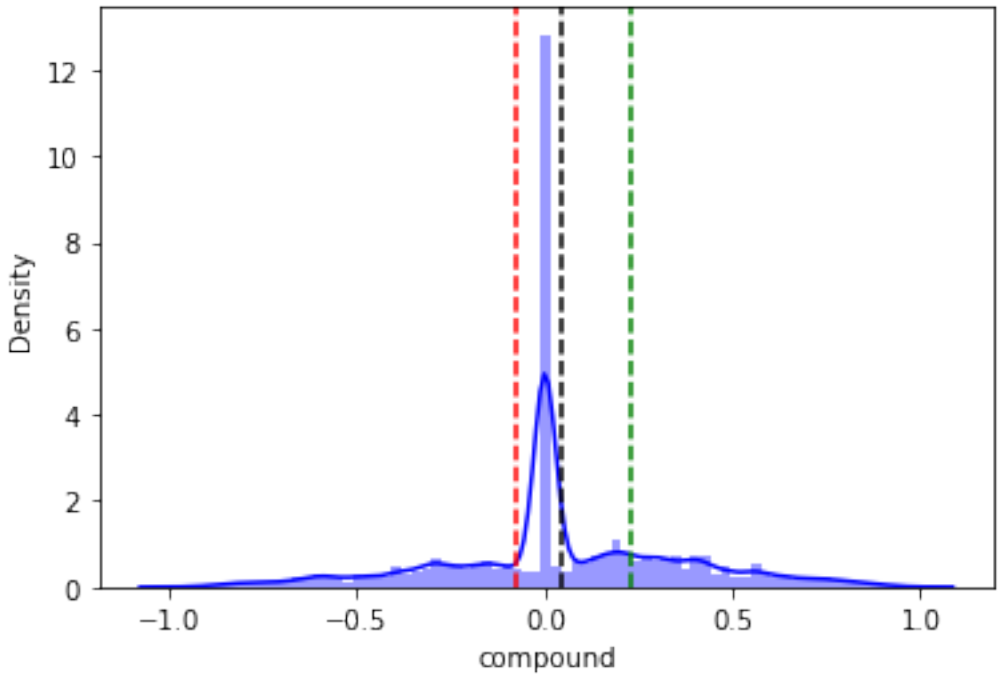


After considering the sentiment score of each tweet, we want to plot the frequency of the compound score. The black line represents the mean, while the red and the green one represent respectively the first and third quantile.

[82]:
```
def freq_compound(data):
    sns.distplot(data.compound, bins=75, color='b')
    plt.axvline(x=np.mean(data.compound), color='black', ls='--')
 ↪# mean of compound
    plt.axvline(x=np.quantile(data.compound, 0.75), color='green', ls='--')
 ↪# third quartile of compound
    plt.axvline(x=np.quantile(data.compound, 0.25), color='red', ls='--')
 ↪# first quartile of compound
    plt.show()

freq_compound(df)
```

```
/home/francesco/.local/lib/python3.8/site-
packages/seaborn/distributions.py:2557: FutureWarning:
```

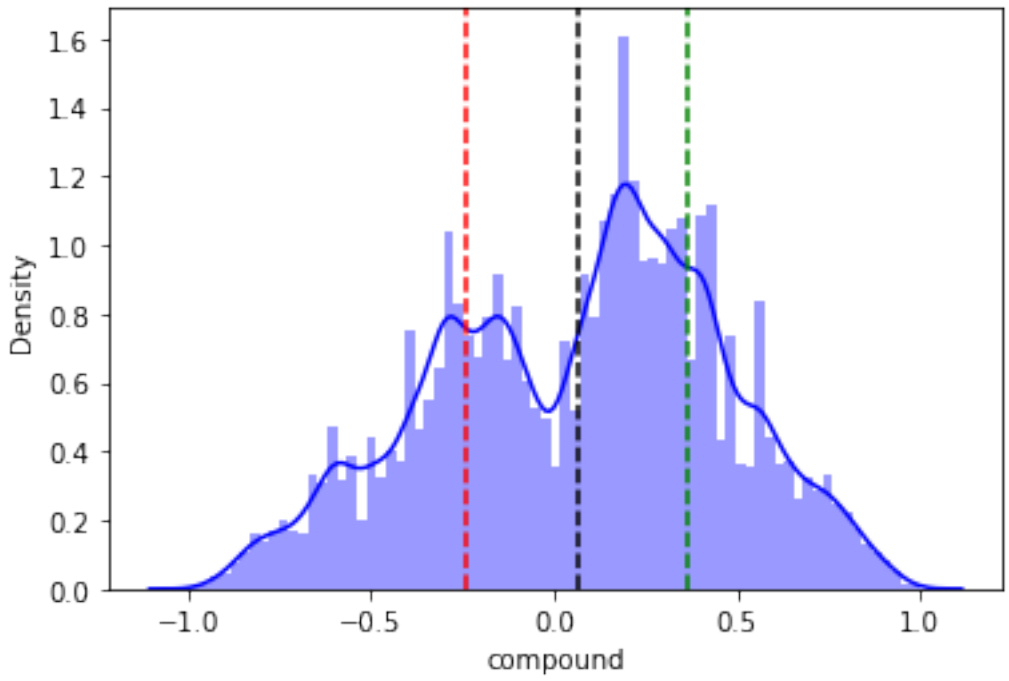
```
`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```



Given that the number of tweets whose compound score is equal to 0 is very high, in order to better visualize the distribution we consider the tweets whose compound score is different from 0.

[83]:
```
freq_compound(df[df.compound != 0])
```

```
/home/francesco/.local/lib/python3.8/site-
packages/seaborn/distributions.py:2557: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```



Furthermore, we can discretize the value of the compound in three classes: negative, positive and neutral, adding this information to a new column of the DataFrame.

- if $compound < -0.05$: the sentiment is negative
- if $compound > 0.05$: the sentiment is positive
- if $-0.05 < compound < 0.05$: the sentiment is neutral

[84]:
```
def discretize_values(values, thr):
    discretized_values = []
    for value in values:
        if value < -thr:
            discretized_values.append("negative")
        elif -thr <= value <= thr:
            discretized_values.append("neutral")
        else:
            discretized_values.append("positive")
    return discretized_values

thr=0.05
discrete_pol_scores = discretize_values(tweets_compound, thr)

df["text_score"] = discrete_pol_scores
```

[85]:
```
df.head()
```

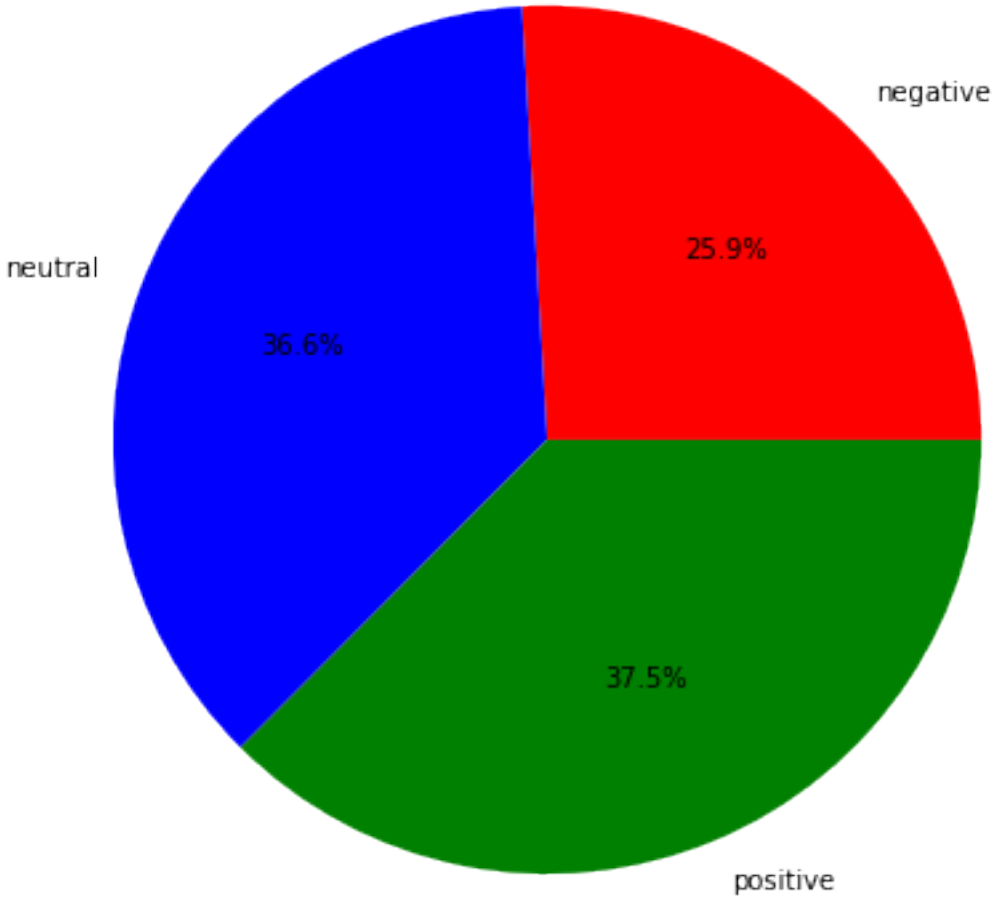
[85]:
```
             user_name   user_location  user_verified   date  \
0     insightfultroll              NaN          False  07-24
1  GlobalPandemic.NET  United Kingdom          False  07-24
2           Euan Watt  United Kingdom          False  07-24
3  GlobalPandemic.NET  United Kingdom          False  07-24
4               Brian   United States          False  07-24

                                                text     hashtags  \
0  [i'm glad i'm in canada, you can see why the p...          NaN
1  [alert: after times investigation, newsom says...          NaN
2  [having checked the data of all nation's of th...          NaN
3  [alert: doctors post bikini photos to protest ...          NaN
4  [ but not a word about #covid19 spread for the...  ['COVID19']

                                     text_tokenized  compound text_score
0  [glad, canada, see, pandemic, raging, united, ...    0.3400   positive
1  [alert, times, investigation, newsom, says, nu...    0.2960   positive
2              [checked, data, uk, deaths, england]    0.0000    neutral
3  [alert, doctors, post, bikini, photos, protest...   -0.4767   negative
4  [word, spread, last, days, portland, city, mas...    0.0000    neutral
```

After having calculated the score for each tweet, we want to summarize these results through a pie chart.

[86]:
```
def freq_discrete(data):
    freq_map = {'negative': 0, 'neutral': 0, 'positive': 0}
    freq_map["positive"] = list(data.text_score).count("positive")
    freq_map["neutral"] = list(data.text_score).count("neutral")
    freq_map["negative"] = list(data.text_score).count("negative")
    print(freq_map)
    plt.figure(figsize=(9,7.5))
    plt.pie(freq_map.values(), labels = freq_map.keys(), autopct='%1.1f%%',␣
 ↪colors=['r', 'b', 'g'] )
```

[87]:
```
freq_discrete(df)
```

```
{'negative': 46439, 'neutral': 65552, 'positive': 67117}
```



In the dataset, certain users tweeted more than others. Therefore, their sentiments are having a higher influence in the chart. Therefore, we want to group the tweets by user for the sentiment analysis.

[88]:
```
# group the dataframe by user_name and do the mean of the compound values for␣
 ↪each user_name

df_byusername = df.groupby(["user_name"])["compound"].mean()
df_byuser = pd.DataFrame({'user_name': df_byusername.index, 'compound':
 ↪df_byusername.values})
df_byuser['text_score'] = discretize_values(df_byuser.compound, thr)
```

[89]:
```
df_byuser.head()
```

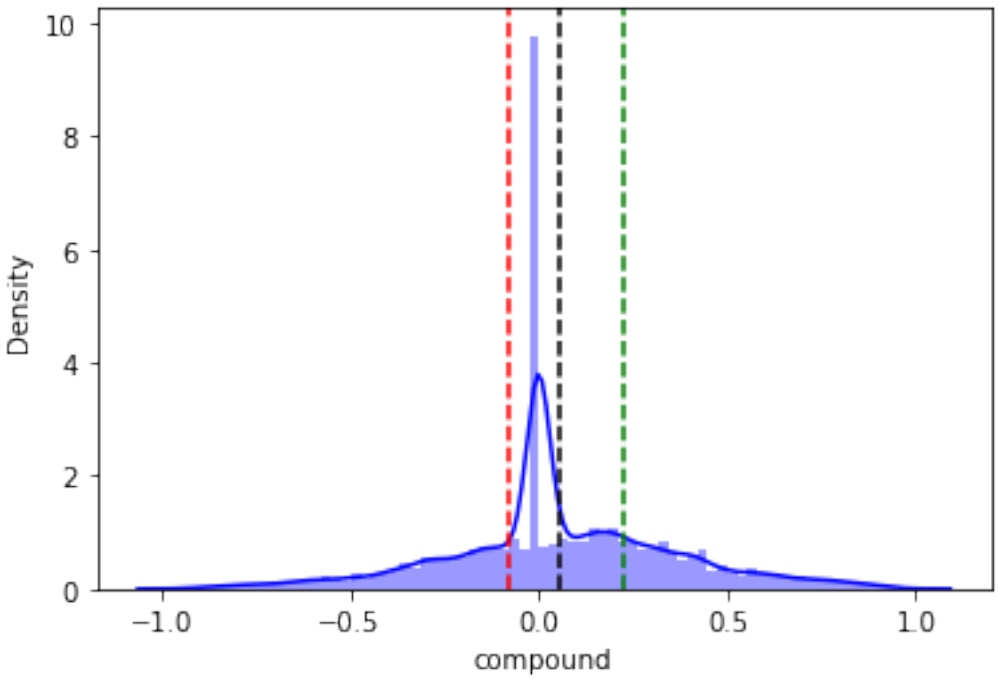
[89]:

	user_name	compound	text_score
0	2Civility	0.0000	neutral
1	Corona	-0.3243	negative
2	!!SEVEN TIMES!! IN-THE-BACK.	0.0000	neutral
3	!F	0.0000	neutral
4	!Gau Khoeb	0.0000	neutral

We repeat the analysis done so far, grouping the data by users. Hence, we plot the distribution of the compound scores and then we plot it considering only the tweets with compound score different than 0. Finally, we plot the a summary of the compund score through a pie chart.

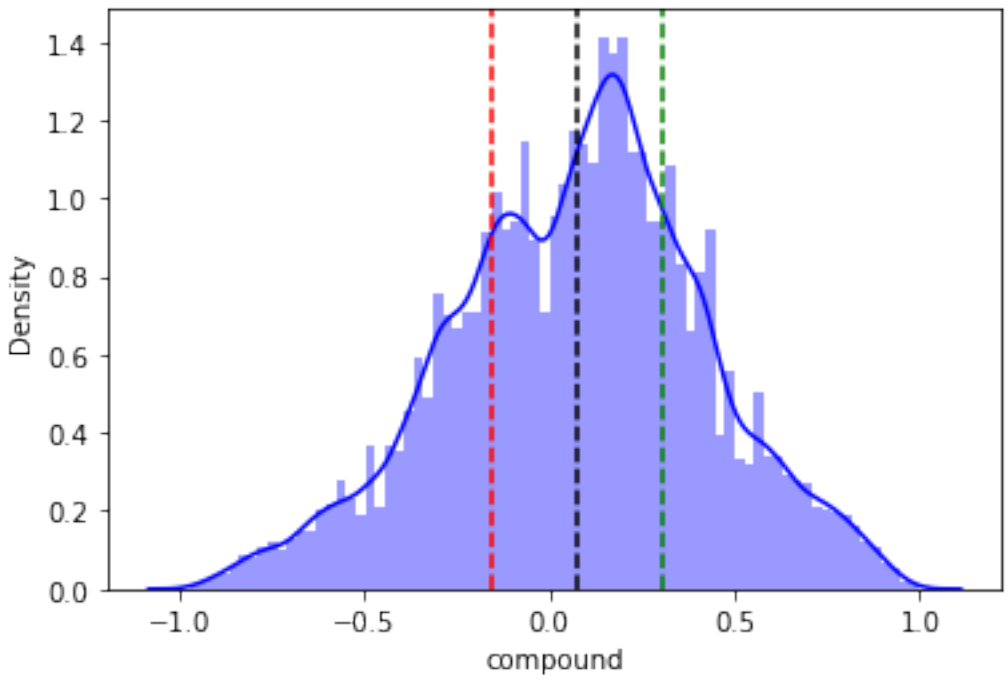
[90]:
```
freq_compound(df_byuser)
freq_compound(df_byuser[df_byuser.compound != 0])
freq_discrete(df_byuser)
```

```
/home/francesco/.local/lib/python3.8/site-
packages/seaborn/distributions.py:2557: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```

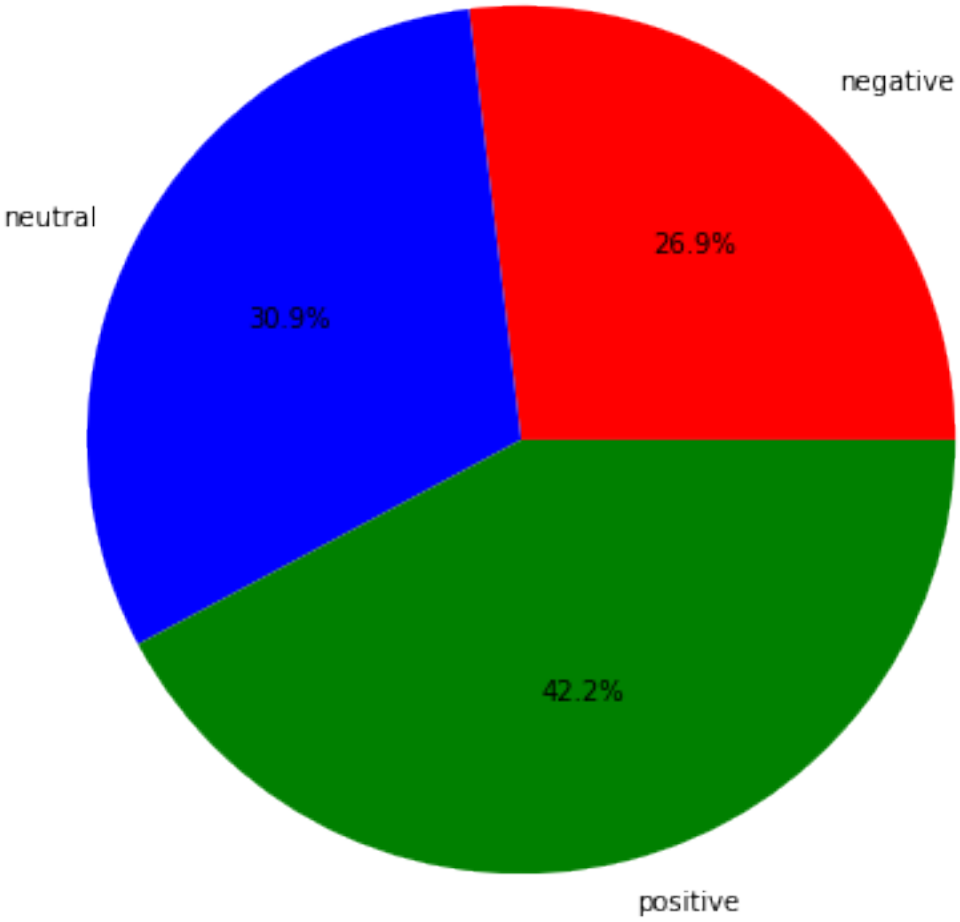



```
/home/francesco/.local/lib/python3.8/site-
packages/seaborn/distributions.py:2557: FutureWarning:
```

```
`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```




```
{'negative': 24837, 'neutral': 28502, 'positive': 38937}
```



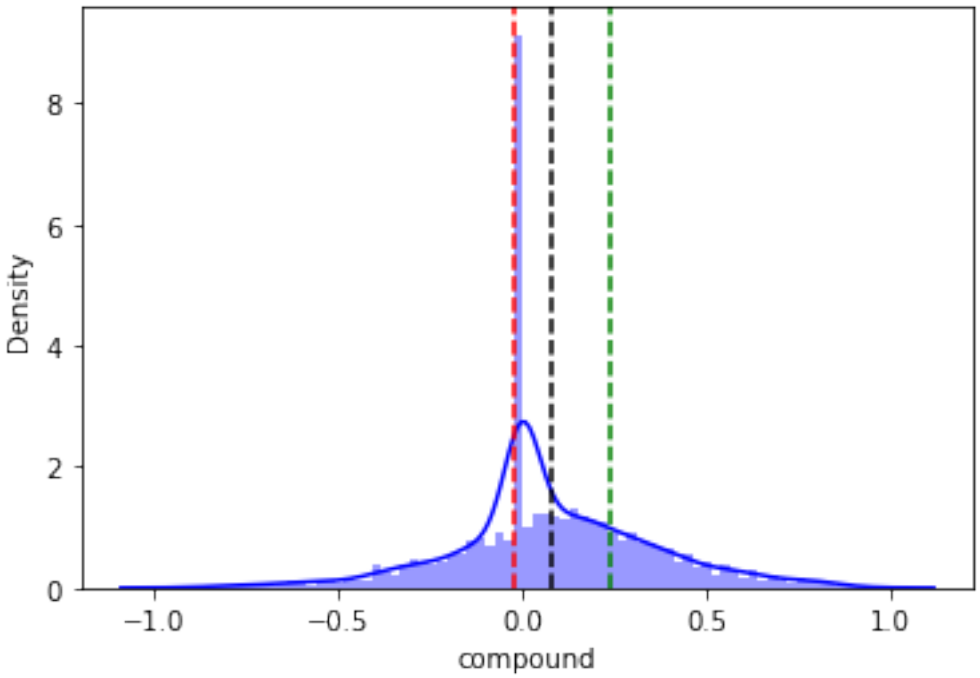
Here we introduce a new classification, repeating the analysis done so far for verified users and not, after having grouped tweets by user.

[91]:
```
df_byusername_ver = df_usver().groupby(["user_name"])["compound"].mean()
df_byuser_ver = pd.DataFrame({'user_name': df_byusername_ver.index, 'compound':
 ↪df_byusername_ver.values})
df_byuser_ver['text_score'] = discretize_values(df_byuser_ver.compound, thr)
```

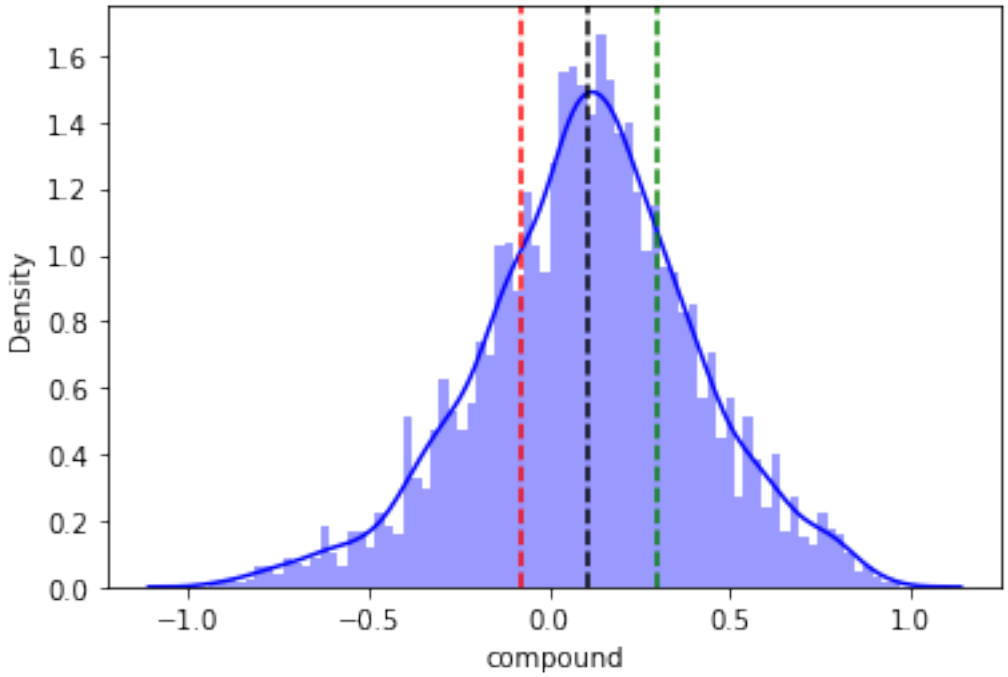
[92]:
```
freq_compound(df_byuser_ver)
freq_compound(df_byuser_ver[df_byuser_ver.compound != 0])
freq_discrete(df_byuser_ver)
```

```
/home/francesco/.local/lib/python3.8/site-
```

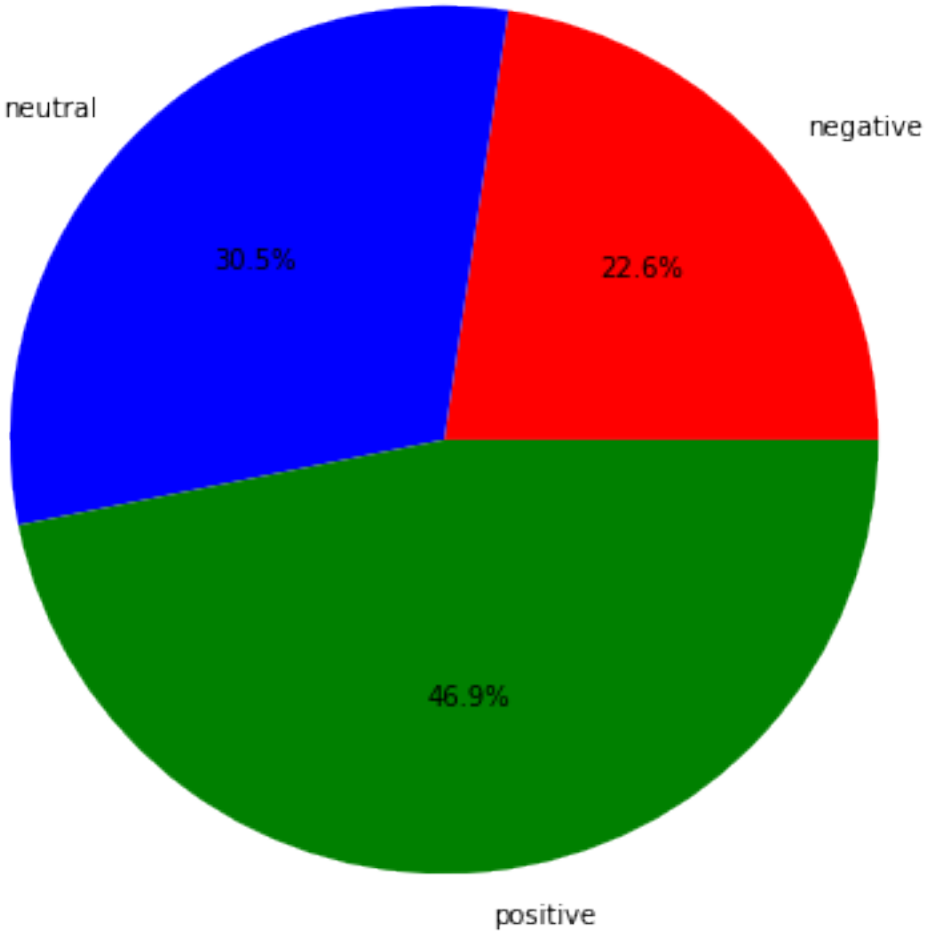
```
packages/seaborn/distributions.py:2557: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```




```
/home/francesco/.local/lib/python3.8/site-
packages/seaborn/distributions.py:2557: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```




```
{'negative': 1409, 'neutral': 1899, 'positive': 2916}
```



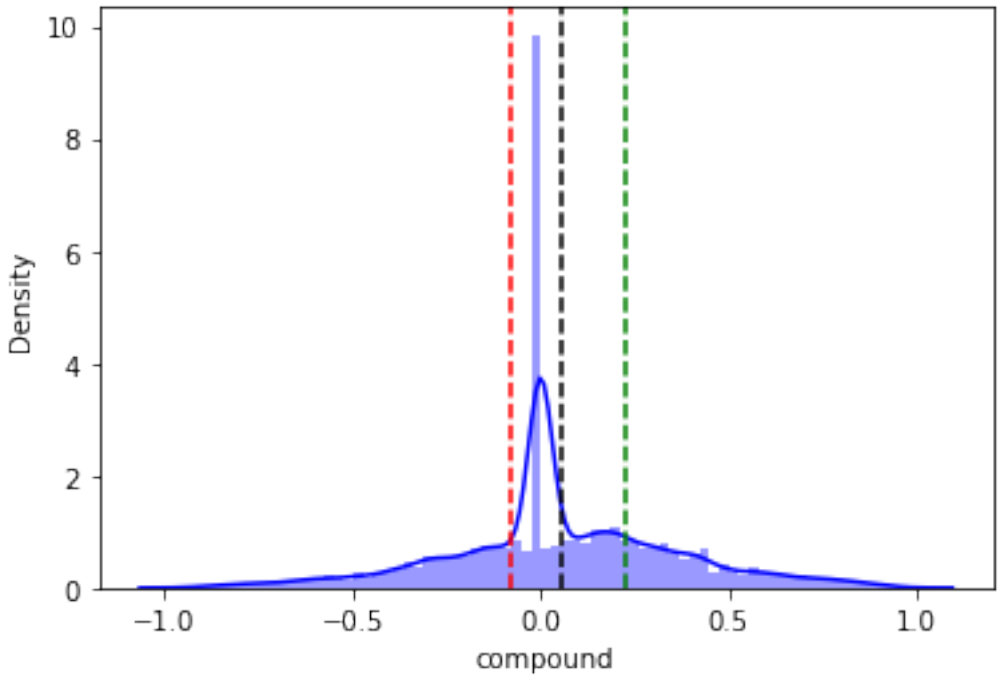
The same procedure is done for not verified users.

[93]:
```
df_byusername_notver = df_usver(False).groupby(["user_name"])["compound"].mean()
df_byuser_notver = pd.DataFrame({'user_name': df_byusername_notver.index,␣
↪'compound':df_byusername_notver.values})
df_byuser_notver['text_score'] = discretize_values(df_byuser_notver.compound,␣
↪thr)
```

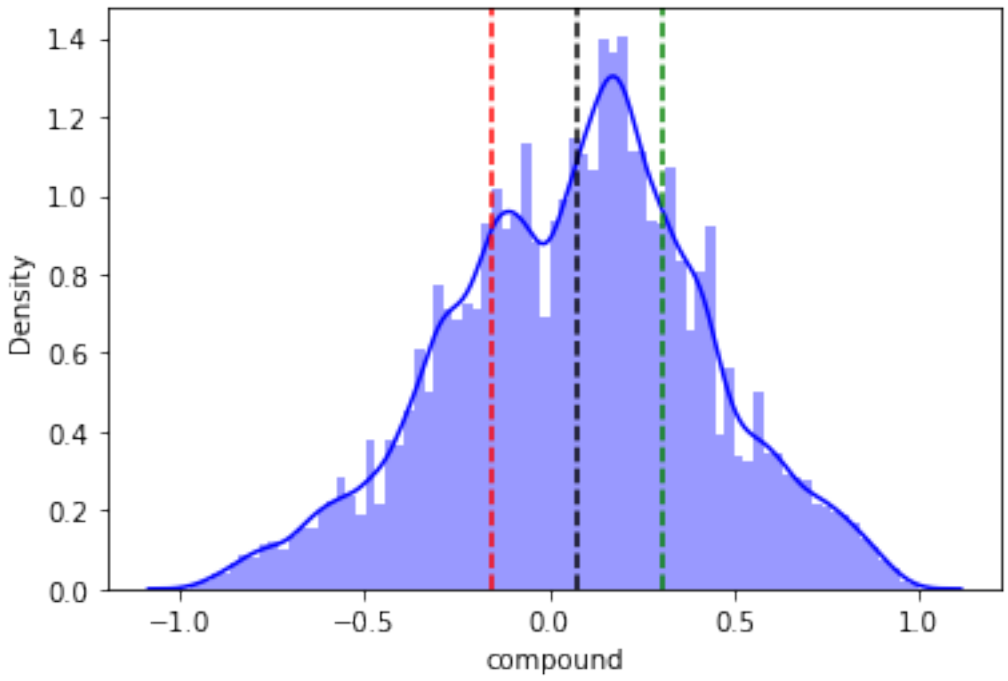
[94]:
```
freq_compound(df_byuser_notver)
freq_compound(df_byuser_notver[df_byuser_notver.compound != 0])
freq_discrete(df_byuser_notver)
```

```
/home/francesco/.local/lib/python3.8/site-
```

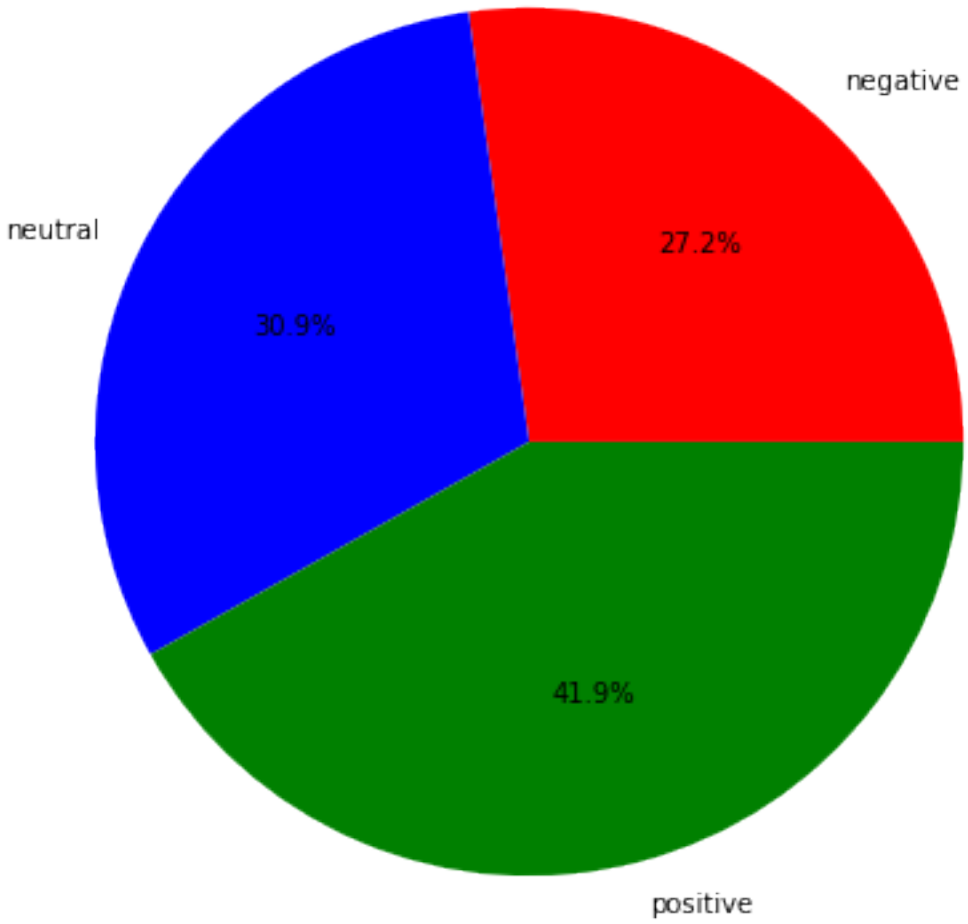
```
packages/seaborn/distributions.py:2557: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```




```
/home/francesco/.local/lib/python3.8/site-
packages/seaborn/distributions.py:2557: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```



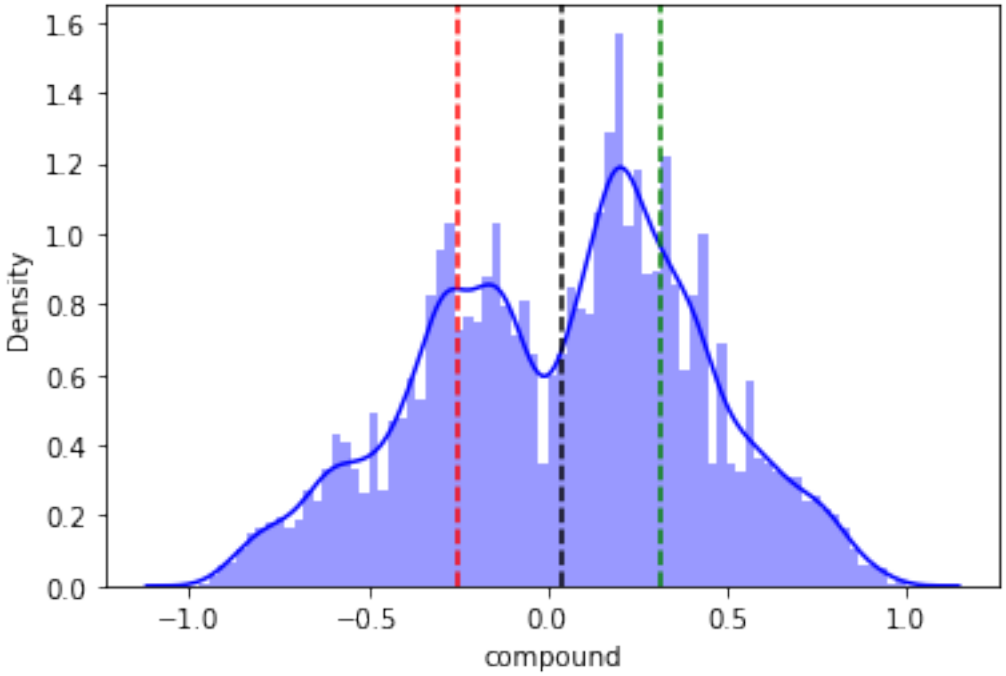

```
{'negative': 23453, 'neutral': 26625, 'positive': 36048}
```



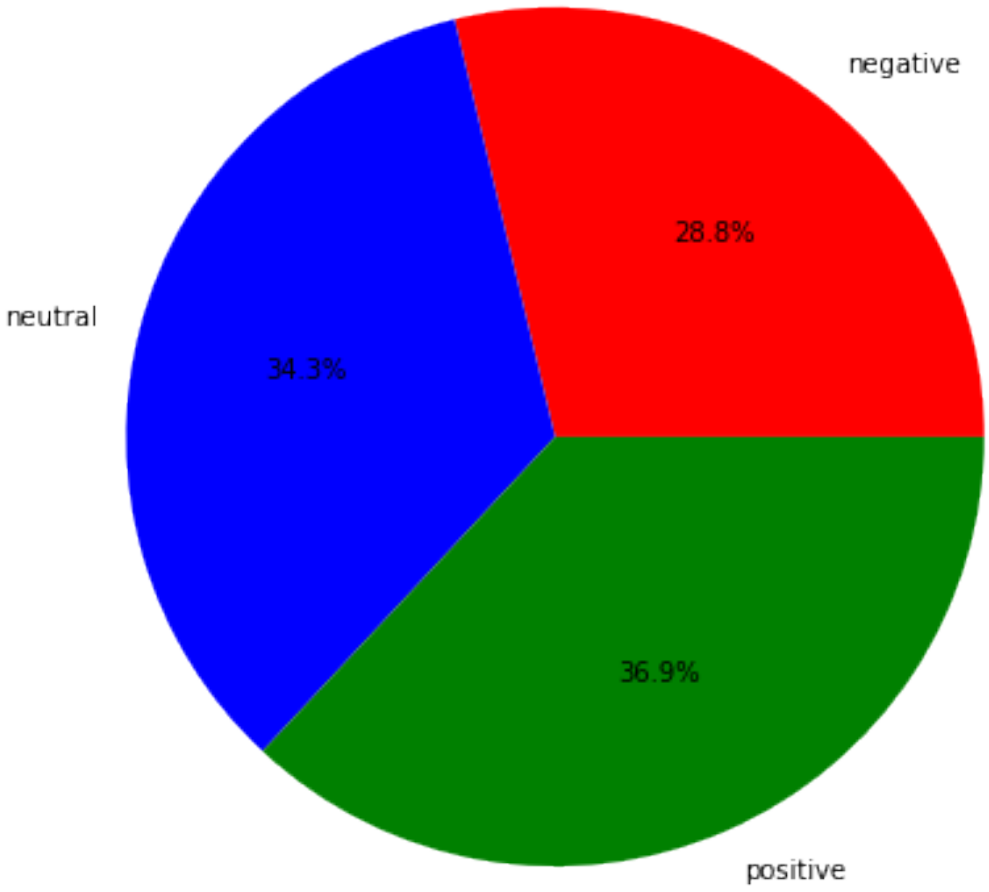
We can plot the frequency of the compound and a pie chart for each country. Here, we will show these plots for the United States (country who sent the most tweets) and for Italy.

[95]:
```
freq_compound(df_country('United States')[df_country('United States').compound !
 ↪= 0])
freq_discrete(df_country('United States'))
```

```
/home/francesco/.local/lib/python3.8/site-
packages/seaborn/distributions.py:2557: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```

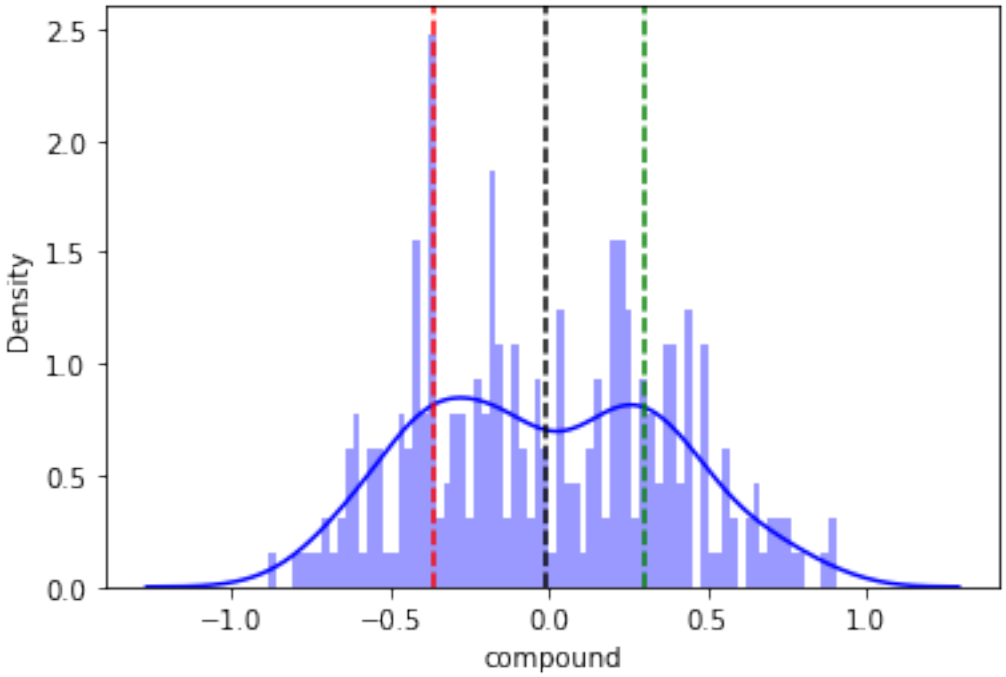



```
{'negative': 12157, 'neutral': 14491, 'positive': 15608}
```

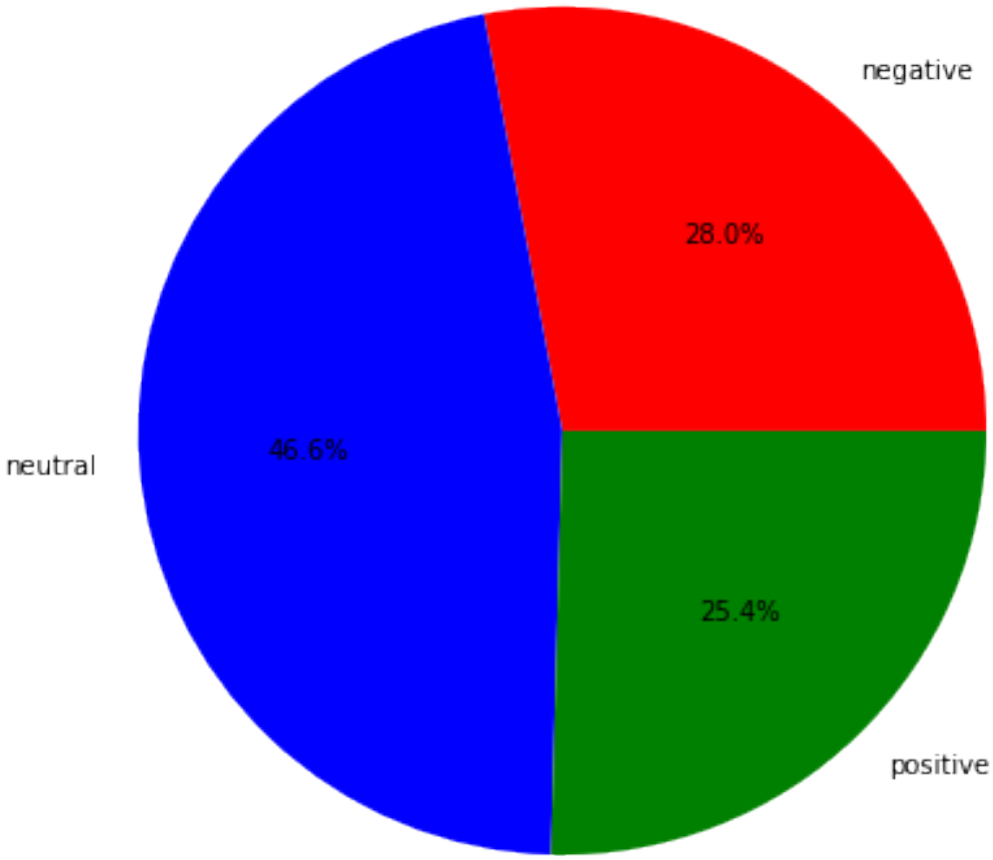


[96]:
```
freq_compound(df_country('Italy')[df_country('Italy').compound != 0])
freq_discrete(df_country('Italy'))
```

```
/home/francesco/.local/lib/python3.8/site-
packages/seaborn/distributions.py:2557: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
```

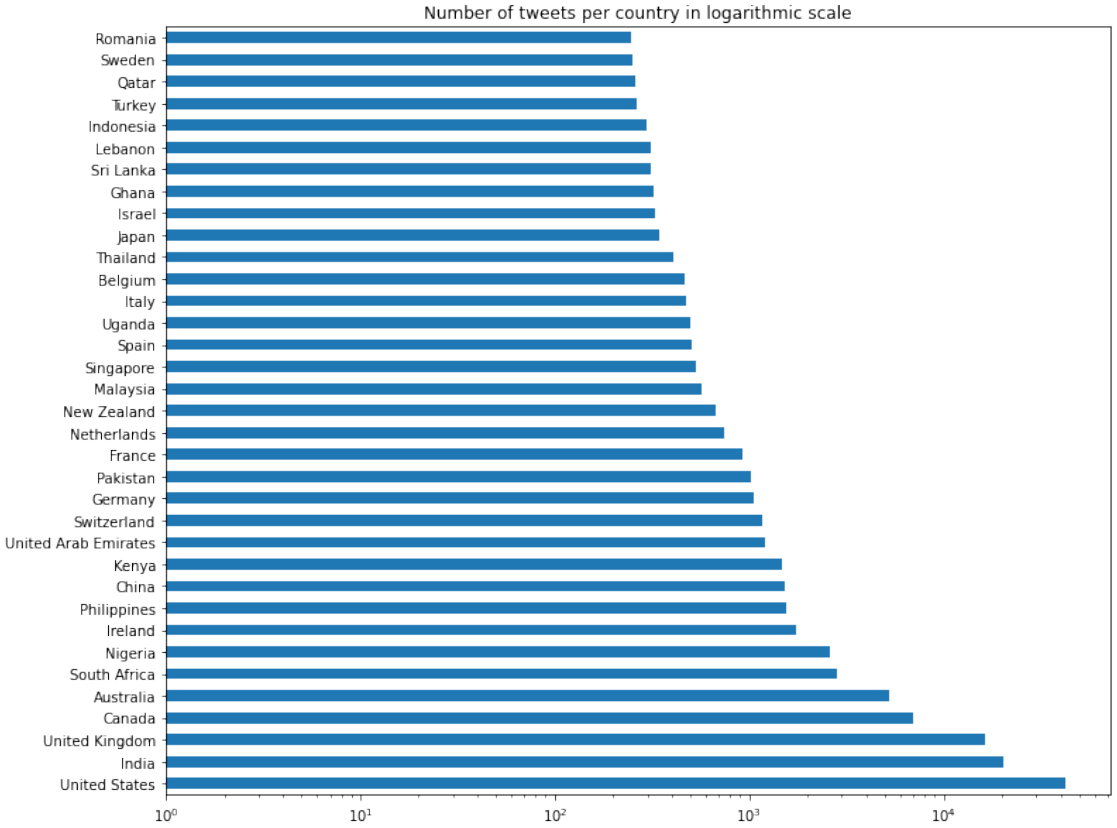



```
{'negative': 132, 'neutral': 220, 'positive': 120}
```



Sentiment Analysis per Country Here we want to consider the sentiment score per country. We will start by plotting the contries who sent the most tweets.

[97]:
```
freq_tweet_country = FreqDist(df.user_location.dropna())
count_country = pd.Series(freq_tweet_country.values(), index =␣
 ↪freq_tweet_country.keys())
count_country = count_country.sort_values(ascending = False)

plt.figure(figsize=(12,10))
count_country[:35].plot.barh(count_country, log = True)
plt.title('Number of tweets per country in logarithmic scale')
plt.show()
```



Then we want to compute the average of the compound score of the tweets sent in that country.

[98]:
```
import pycountry
```

[99]:
```
# get code iso3 for each country

def get_country_iso3(col):
    try:
        country_code =  pycountry.countries.search_fuzzy(col)[0].alpha_3
    except:
        country_code = None
    return country_code
```

[100]:
```
def compound_by_country(data):
    df_bycountry = data.groupby("user_location")["compound"].mean()
    country_iso = pd.Series(df_bycountry.index).apply(get_country_iso3)
    df_byloc = pd.DataFrame({'country': list(df_bycountry.index), 'country_iso3':
 ↪ country_iso, 'compound': df_bycountry.values})
    df_byloc = df_byloc.dropna()
    return df_byloc
```

```
df_bycountry = compound_by_country(df)
```

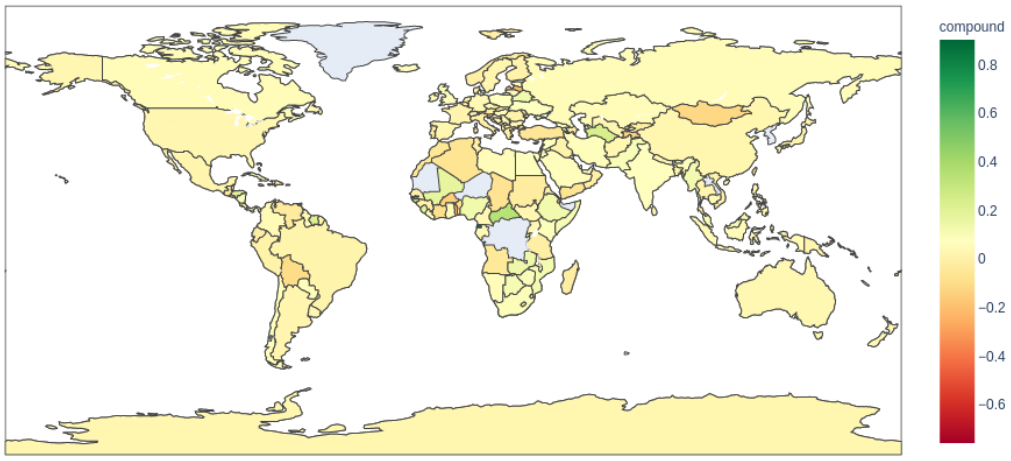
[101]:
```
df_bycountry
```

[101]:
```
          country country_iso3  compound
0     Afghanistan          AFG  0.024940
1          Africa          ZAF  0.044257
2         Albania          ALB -0.037036
3         Algeria          DZA -0.067310
4         Andorra          AND  0.000000
..            ...          ...       ...
202     Venezuela          VEN -0.034734
203       Vietnam          VNM  0.010759
204         Yemen          YEM -0.067965
205        Zambia          ZMB  0.142479
206      Zimbabwe          ZWE  0.092278

[189 rows x 3 columns]
```

Here we want to plot a map of the average sentiment score per country, as described in the dataset above.

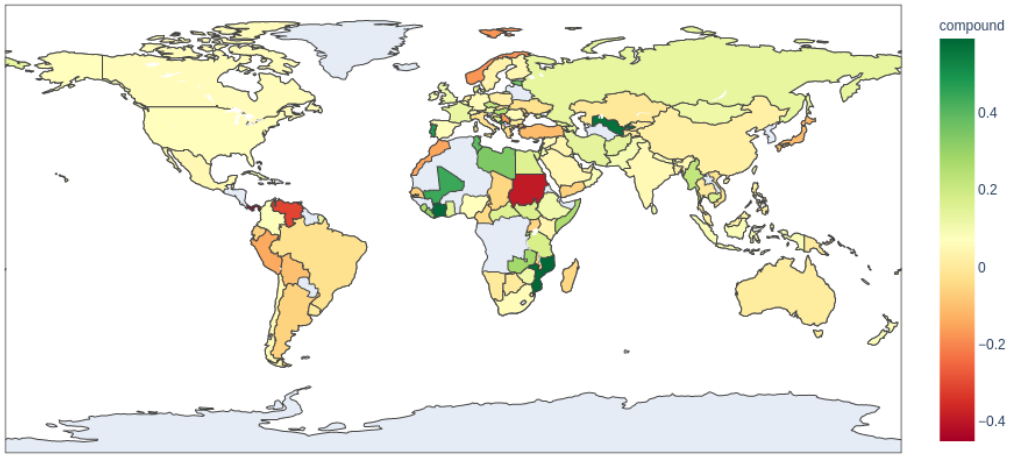
[102]:
```
#!pip install plotly
```

[103]:
```
# create a choropleth map
def choropleth_map_compound(data):
    df_bycountry = compound_by_country(data)
    fig = px.choropleth(df_bycountry ,
                        locations="country_iso3",
                        color="compound",
                        hover_name="country",
                        color_continuous_scale= 'RdYlGn')

    fig.show()

choropleth_map_compound(df)
```



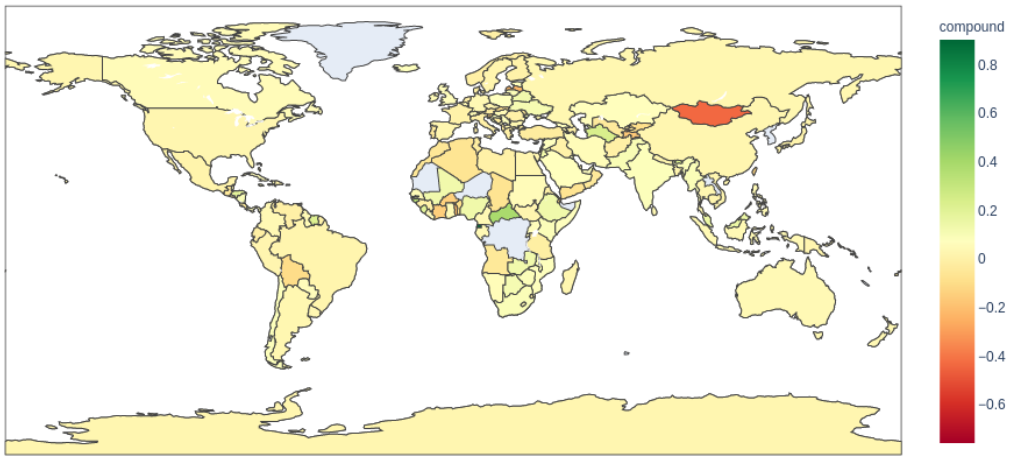
We then can plot a similar map of the sentiment score per country for verified user, and for not verified users.

[104]:
```
choropleth_map_compound(df_usver())
```



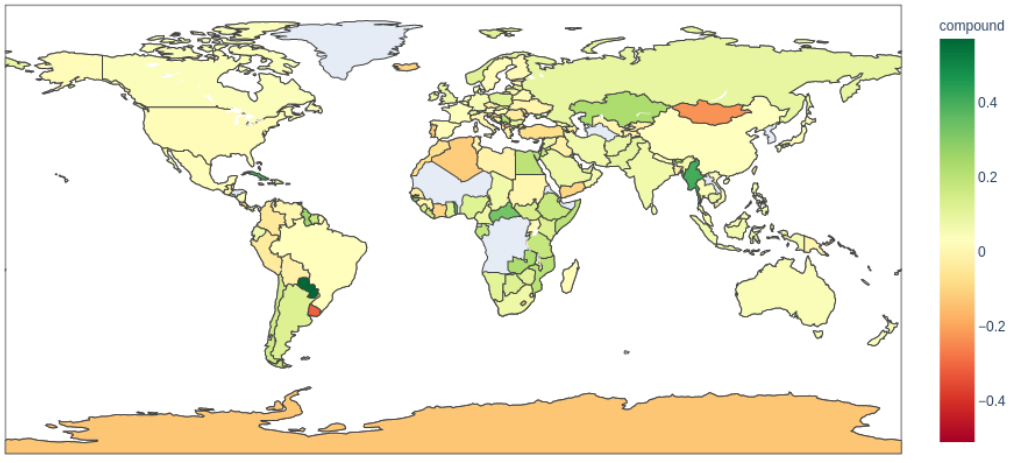
For not verified users.

[105]:
```
choropleth_map_compound(df_usver(False))
```



We also plot a similar map for users in a certain period of time.

[106]:
```
choropleth_map_compound(df_dates("08-10","08-16"))
```



So far, we considered the avarage sentiment score of each country, averaging on the period of days we are considering. Here, we want to consider the individual sentiment score for each day of period we are considering for each country.

[107]:
```
tab = df.pivot_table(index = 'user_location', columns='date', values =␣
↪'compound', aggfunc = 'mean').T  #add margins = True for watching the tot mean
```

[108]:
```
tab
```

[108]:

user_location	Afghanistan	Africa	Albania	Algeria	Andorra	Angola	\
date							
07-24	NaN	NaN	NaN	NaN	NaN	NaN	
07-25	-0.051667	0.034071	0.063450	-0.1027	NaN	NaN	
07-26	-0.025489	0.051713	0.067433	NaN	0.0	NaN	
07-27	0.000000	0.144304	0.153100	NaN	NaN	NaN	
07-28	0.025675	-0.096233	0.493900	0.0000	NaN	NaN	
07-29	NaN	0.022562	NaN	NaN	NaN	NaN	
07-30	NaN	0.354800	NaN	NaN	NaN	NaN	
07-31	NaN	-0.162043	-0.095320	-0.1056	NaN	NaN	
08-01	NaN	-0.089886	NaN	NaN	NaN	NaN	
08-02	-0.039075	0.073360	0.052950	NaN	NaN	NaN	
08-04	0.278700	0.069423	-0.567250	NaN	NaN	NaN	
08-06	0.115560	-0.115969	NaN	NaN	NaN	NaN	
08-07	NaN	0.229500	NaN	0.0000	NaN	NaN	
08-08	-0.136150	0.143463	-0.180600	NaN	NaN	NaN	
08-09	0.108950	0.031710	0.000000	0.0000	NaN	NaN	
08-10	-0.025675	0.212300	0.246950	-0.3592	NaN	NaN	
08-11	0.162025	0.046511	-0.260850	0.0000	NaN	NaN	
08-12	0.123750	0.097060	0.273587	0.0000	NaN	NaN	
08-13	0.000000	0.072527	0.021850	NaN	NaN	NaN	
08-14	NaN	-0.085236	0.076550	NaN	NaN	NaN	
08-16	0.069167	0.023531	-0.192925	NaN	NaN	NaN	
08-17	-0.199800	0.086144	0.000000	0.0000	NaN	-0.30665	
08-18	NaN	0.210750	-0.237700	NaN	NaN	0.21075	
08-22	0.063033	0.134613	-0.053900	NaN	NaN	NaN	
08-29	NaN	NaN	-0.177400	NaN	NaN	NaN	
08-30	0.000000	0.026793	NaN	NaN	NaN	NaN	

user_location	Antarctica	Argentina	Armenia	Asia	...	United Kingdom	\
date					...		
07-24	NaN	NaN	NaN	NaN	...	-0.031704	
07-25	0.000000	-0.054783	0.090383	-0.089300	...	0.068794	
07-26	-0.022042	0.167625	0.000000	0.000000	...	0.045525	
07-27	0.000000	0.765650	0.000000	0.238350	...	0.059901	
07-28	NaN	-0.062283	-0.725800	-0.023100	...	0.034477	
07-29	0.136600	NaN	0.278700	NaN	...	0.083465	
07-30	0.250000	0.000000	NaN	NaN	...	0.072778	
07-31	0.000000	-0.247143	0.000000	0.265375	...	0.084559	
08-01	NaN	0.082233	0.000000	0.202300	...	0.048885	
08-02	0.096463	-0.089775	NaN	NaN	...	0.054625	
08-04	NaN	-0.000575	-0.007400	0.000000	...	0.080610	

```
08-06                NaN   0.102344 -0.025800       NaN  ...        0.076391
08-07                NaN        NaN  0.557400       NaN  ...        0.076381
08-08                NaN  -0.316600  0.244725  0.073867  ...        0.057072
08-09                NaN   0.000000  0.257850  0.268867  ...        0.056244
08-10                NaN        NaN -0.670500  0.709600  ...        0.059886
08-11                NaN   0.557400  0.255000  0.010000  ...        0.099513
08-12                NaN   0.401900 -0.001446  0.217000  ...        0.052193
08-13                NaN   0.044620  0.065100 -0.220200  ...        0.098984
08-14                NaN   0.255575  0.000000  0.199317  ...        0.054875
08-16          -0.133967  -0.016500  0.000000  0.278700  ...        0.045675
08-17                NaN  -0.223087  0.240033  0.186150  ...        0.066510
08-18           0.000000   0.000000  0.345400 -0.735100  ...        0.045596
08-22           0.000000   0.033210  0.000000 -0.431250  ...        0.063488
08-29                NaN   0.022350 -0.179750       NaN  ...        0.041595
08-30                NaN   0.176800  0.000000 -0.316638  ...        0.070806

user_location  United States   Uruguay  Uzbekistan  Vatican City  Venezuela  \
date
07-24               0.035530       NaN         NaN           NaN        NaN
07-25               0.021017  0.000000     0.13085     -0.156225  -0.445500
07-26               0.005225 -0.535800         NaN           NaN   0.000000
07-27               0.021501  0.281250         NaN           NaN  -0.312450
07-28               0.019982  0.354800     0.14800           NaN        NaN
07-29               0.049618       NaN     0.57190           NaN        NaN
07-30               0.025026       NaN         NaN           NaN        NaN
07-31               0.061734       NaN         NaN           NaN   0.273200
08-01               0.038390 -0.127267         NaN           NaN  -0.079608
08-02               0.019736  0.000000         NaN           NaN        NaN
08-04               0.009274       NaN    -0.22020           NaN   0.787000
08-06               0.068404       NaN     0.00000           NaN        NaN
08-07               0.108420       NaN         NaN           NaN        NaN
08-08               0.041819       NaN    -0.61240           NaN        NaN
08-09               0.010567  0.271150         NaN           NaN        NaN
08-10               0.062878       NaN         NaN      0.440400        NaN
08-11               0.017934       NaN         NaN           NaN        NaN
08-12               0.031023 -0.680800     0.00000           NaN        NaN
08-13               0.008958       NaN         NaN           NaN        NaN
08-14               0.015133  0.051467     0.00000           NaN        NaN
08-16              -0.034006       NaN         NaN      0.000000   0.000000
08-17               0.005678       NaN         NaN      0.180600        NaN
08-18               0.052580  0.114933     0.20230           NaN   0.000000
08-22               0.012508 -0.381800         NaN      0.735100        NaN
08-29               0.012444       NaN         NaN           NaN        NaN
08-30              -0.000844       NaN         NaN           NaN        NaN

user_location   Vietnam    Yemen    Zambia  Zimbabwe
date
```

```
07-24              NaN      NaN       NaN       NaN
07-25         0.091141      NaN  0.195750  0.155011
07-26        -0.129717      NaN  0.342154  0.244036
07-27        -0.128700      NaN  0.219683 -0.026407
07-28        -0.134425      NaN       NaN  0.102158
07-29         0.000000      NaN       NaN  0.246950
07-30              NaN      NaN       NaN  0.502300
07-31        -0.069278      NaN  0.000000 -0.165313
08-01         0.000000      NaN  0.000000  0.165525
08-02         0.000000 -0.21355  0.000000 -0.081100
08-04        -0.092162      NaN  0.089983  0.038486
08-06              NaN      NaN       NaN  0.066170
08-07         0.278700      NaN       NaN       NaN
08-08         0.002831      NaN -0.140500  0.240681
08-09         0.107520      NaN  0.135642  0.240105
08-10        -0.331267  0.00000  0.000000  0.259950
08-11        -0.026450 -0.21315  0.317258 -0.053065
08-12        -0.030085 -0.00555  0.306471  0.216269
08-13        -0.004275 -0.75790  0.213200  0.121498
08-14        -0.015050      NaN  0.048500  0.136271
08-16         0.188604  0.37130  0.225700 -0.039700
08-17         0.101042 -0.73510  0.090283 -0.022290
08-18        -0.445050  0.41895 -0.184825  0.064000
08-22         0.182343  0.05160 -0.070658  0.082557
08-29              NaN      NaN       NaN       NaN
08-30         0.292275  0.00000 -0.159100  0.033521

[26 rows x 207 columns]
```

Here we remove from our analysis those columns that have many missing values.

[109]:
```
for col in tab.columns:
    if tab[col].isna().sum() > tab.shape[0]*0.1:
        del tab[col]
```

[110]:
```
tab
```

[110]:
```
user_location    Africa  Australia   Austria   Belgium    Brazil    Canada  \
date
07-24              NaN  -0.096474       NaN       NaN       NaN  0.109123
07-25         0.034071   0.034463  0.051165 -0.030525  0.024029  0.067053
07-26         0.051713   0.019042 -0.035769  0.077965  0.074183  0.035045
07-27         0.144304   0.054700  0.007731  0.039148  0.047531  0.038174
07-28        -0.096233   0.061788 -0.332788  0.025561  0.082582  0.036696
07-29         0.022562  -0.028910       NaN -0.161207  0.198950  0.100157
07-30         0.354800  -0.130175 -0.296000  0.067150 -0.162425  0.078787
07-31        -0.162043   0.106182  0.080219  0.025302  0.167625  0.079610
```

```
08-01         -0.089886  -0.031327  0.029807 -0.184686 -0.048419  0.080573
08-02          0.073360   0.043519 -0.188216  0.001619 -0.051117  0.056657
08-04          0.069423   0.009844 -0.067738  0.040619 -0.105200  0.017822
08-06         -0.115969  -0.000526  0.201190  0.141564 -0.037910  0.077481
08-07          0.229500  -0.224000  0.025800  0.000000  0.363700  0.038110
08-08          0.143463   0.053769 -0.186360 -0.039325  0.020005  0.062349
08-09          0.031710   0.023554  0.012520  0.017132 -0.227610  0.084435
08-10          0.212300  -0.092686  0.025667 -0.078562  0.131990  0.079119
08-11          0.046511   0.043469  0.082150  0.103773  0.111750  0.051653
08-12          0.097060   0.041119 -0.021391  0.100372  0.138067  0.043476
08-13          0.072527   0.087713  0.063567  0.022046 -0.074660  0.039952
08-14         -0.085236   0.047545  0.113450  0.032238 -0.052361  0.038610
08-16          0.023531   0.038395  0.061669 -0.029750  0.049817 -0.009441
08-17          0.086144   0.052970 -0.065950  0.075614  0.061667  0.034629
08-18          0.210750   0.048588  0.087856 -0.038444 -0.108189  0.058927
08-22          0.134613   0.023217  0.047853  0.164032 -0.067461  0.044344
08-29               NaN   0.003110 -0.085700  0.045350 -0.005199  0.020253
08-30          0.026793   0.018644 -0.041293 -0.160650  0.164295 -0.017694

user_location     China  Colombia   Denmark  Ethiopia  ...  South Africa  \
date                                                   ...
07-24         -0.967000  0.877900       NaN       NaN  ...      0.621250
07-25         -0.031915  0.061341 -0.019518  0.095826  ...      0.101835
07-26         -0.025647  0.231250 -0.084489  0.123267  ...      0.079449
07-27          0.040251 -0.087880  0.275680  0.068500  ...      0.131124
07-28          0.039695 -0.110490 -0.076357  0.188767  ...      0.130425
07-29          0.220313  0.135800 -0.187580       NaN  ...      0.138144
07-30          0.112767 -0.394100 -0.542300  0.024550  ...      0.216822
07-31          0.059751  0.075229  0.161875  0.190900  ...      0.054720
08-01          0.022922 -0.040692  0.059940  0.316400  ...      0.012284
08-02          0.040671  0.038773  0.025843  0.300364  ...      0.007679
08-04          0.014607 -0.017570 -0.029629 -0.377700  ...      0.116041
08-06          0.061134  0.107394 -0.083450 -0.076417  ...      0.137448
08-07          0.280425 -0.101150       NaN  0.301300  ...      0.060057
08-08         -0.048727 -0.005708 -0.008828  0.057122  ...      0.010905
08-09         -0.002602  0.508800 -0.144492  0.046117  ...      0.108533
08-10         -0.144318  0.079512 -0.047694  0.159833  ...      0.052228
08-11          0.027633 -0.235333  0.057700  0.137064  ...      0.079248
08-12          0.056492  0.000000 -0.003131  0.287850  ...      0.097585
08-13          0.048895 -0.137750  0.137198  0.065047  ...      0.086276
08-14          0.056641 -0.006133 -0.058513  0.221950  ...      0.078606
08-16          0.006718 -0.042600  0.084067  0.305008  ...      0.048842
08-17          0.065347 -0.170000  0.054780  0.203708  ...      0.103265
08-18         -0.027213  0.154126  0.050933 -0.006125  ...      0.114612
08-22          0.061368  0.008114 -0.091412  0.210233  ...      0.082968
08-29          0.056665  0.059554 -0.044475  0.273000  ...      0.072973
08-30         -0.007000  0.097175 -0.048278  0.072943  ...      0.067332
```

user_location	Spain	Sri Lanka	Sweden	Switzerland	Turkey	Uganda	\
date							
07-24	NaN	-0.340350	NaN	0.000000	NaN	NaN	
07-25	-0.018331	0.137841	-0.004074	0.028496	0.051624	0.002127	
07-26	-0.076297	0.045550	0.053000	0.004469	0.218825	0.151573	
07-27	0.060655	0.057023	0.101200	0.032746	-0.008706	-0.028731	
07-28	-0.034722	0.124542	-0.168845	0.024674	-0.094493	-0.035479	
07-29	0.054025	0.159525	0.035987	0.104812	-0.066929	-0.036767	
07-30	-0.077920	NaN	-0.145433	0.068481	NaN	NaN	
07-31	0.021475	0.000000	-0.008329	0.080869	-0.164260	-0.040200	
08-01	0.238021	-0.148697	0.020442	0.039616	-0.125439	-0.081673	
08-02	0.061712	0.000000	0.067737	-0.028327	-0.014802	0.051546	
08-04	0.085537	0.032097	0.145617	0.003634	-0.185117	0.095806	
08-06	0.088239	0.014800	-0.047915	0.106492	0.140057	0.189378	
08-07	-0.075833	0.579900	0.000000	-0.022020	0.177900	-0.157040	
08-08	-0.021775	0.016233	0.014151	-0.001631	-0.043740	0.034270	
08-09	-0.007462	-0.054639	0.264188	0.007844	-0.186748	0.058737	
08-10	-0.070733	0.198600	-0.019311	0.174112	0.007413	0.010333	
08-11	0.057985	0.060567	0.009240	0.123739	-0.019888	0.030149	
08-12	-0.031678	0.017987	0.129787	0.008339	-0.048826	0.055475	
08-13	0.041850	-0.051390	0.079062	0.103362	-0.160362	-0.001230	
08-14	0.083868	0.025069	0.060542	0.061384	-0.367780	-0.017170	
08-16	-0.002747	-0.018986	-0.144571	0.087131	-0.134706	-0.147355	
08-17	-0.005583	0.033238	0.009550	0.014028	-0.117667	0.091886	
08-18	-0.007847	0.132275	-0.016594	0.038483	-0.060926	-0.019985	
08-22	0.056148	0.018348	-0.048739	0.062568	-0.009066	0.011046	
08-29	0.031552	-0.329850	0.147425	0.195207	-0.195160	0.464350	
08-30	0.104753	0.121476	-0.173131	0.078010	-0.054491	0.078328	

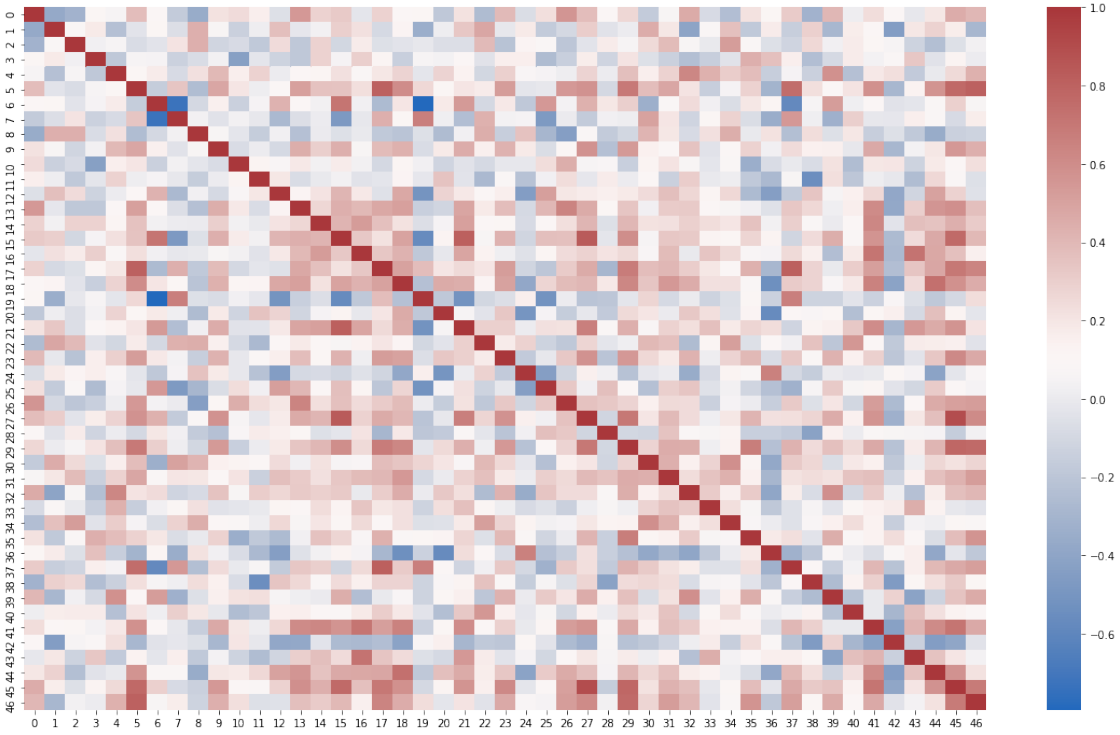
user_location	United Arab Emirates	United Kingdom	United States
date			
07-24	NaN	-0.031704	0.035530
07-25	0.039957	0.068794	0.021017
07-26	0.112713	0.045525	0.005225
07-27	0.089066	0.059901	0.021501
07-28	0.060741	0.034477	0.019982
07-29	0.320807	0.083465	0.049618
07-30	0.131940	0.072778	0.025026
07-31	0.033820	0.084559	0.061734
08-01	0.006577	0.048885	0.038390
08-02	0.046667	0.054625	0.019736
08-04	0.095099	0.080610	0.009274
08-06	0.008128	0.076391	0.068404
08-07	-0.198767	0.076381	0.108420
08-08	0.060759	0.057072	0.041819
08-09	0.066258	0.056244	0.010567

```
08-10                    0.082963        0.059886        0.062878
08-11                    0.078328        0.099513        0.017934
08-12                    0.097337        0.052193        0.031023
08-13                    0.063530        0.098984        0.008958
08-14                    0.097669        0.054875        0.015133
08-16                    0.032770        0.045675       -0.034006
08-17                    0.004860        0.066510        0.005678
08-18                   -0.012404        0.045596        0.052580
08-22                    0.020034        0.063488        0.012508
08-29                    0.113475        0.041595        0.012444
08-30                    0.037222        0.070806       -0.000844

[26 rows x 47 columns]
```

[111]:
```
tab = tab.fillna(0)

sign = np.array(tab).T
```

We plot a similarity matrix to establish the similarity in sentiment scores between each countries.

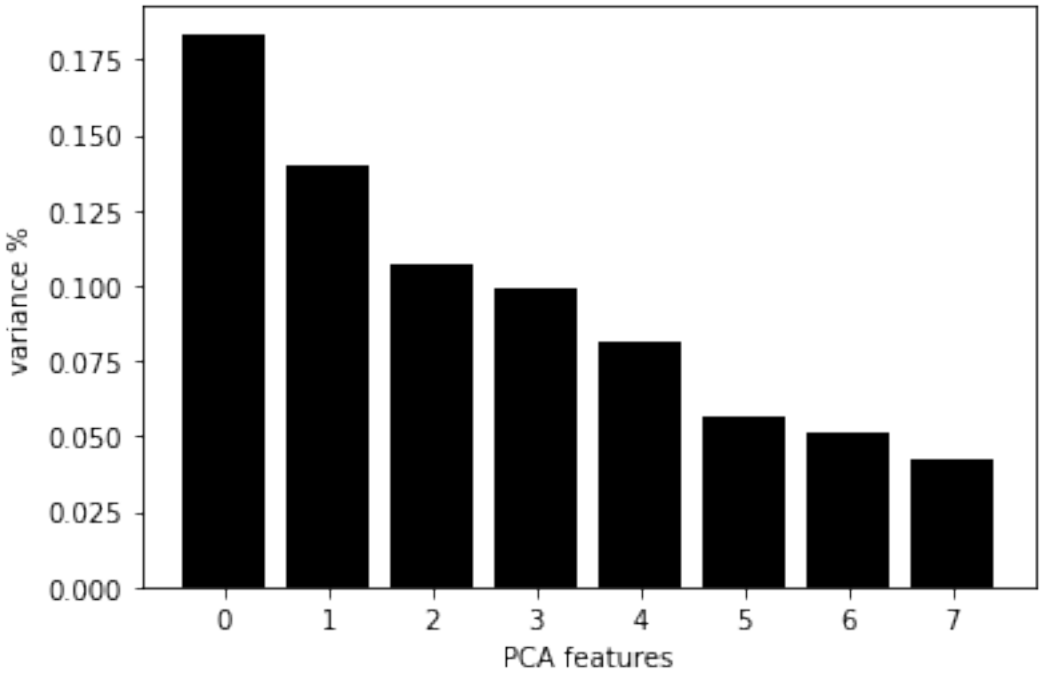
[112]:
```
def get_similarity(a1, a2):
    # cosine similarity/dot product of normalized vectors
    assert len(a1) == len(a2)
    return np.dot(a1, a2)/(np.linalg.norm(a1) * np.linalg.norm(a2))

def compute_sim_matrix(sign):
    # compute similarity matrix across all sentiment arches
    sim_m = np.ones(shape=(sign.shape[0], sign.shape[0]))
    for i in range(sign.shape[0]):
        for j in range(sign.shape[0]):
            sim_m[i, j] = get_similarity(sign[i], sign[j])
    return sim_m


plt.figure(figsize = (20,12))
sim_m = compute_sim_matrix(sign)
ax = sns.heatmap(sim_m, cmap = "vlag")
plt.show()
```



[113]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Create a PCA instance: pca
print(sign.shape)
pca = PCA(n_components=8)
principalComponents = pca.fit_transform(sign)
print(principalComponents.shape)
# Plot the explained variances
features = range(pca.n_components_)
plt.bar(features, pca.explained_variance_ratio_, color='black')
plt.xlabel('PCA features')
plt.ylabel('variance %')
```

```
(47, 26)
(47, 8)
```

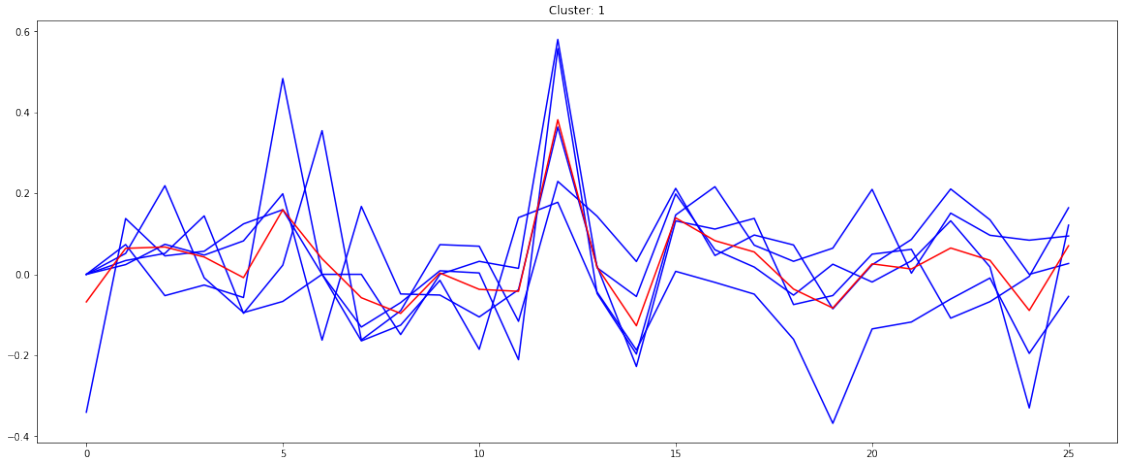
[113]:
```
Text(0, 0.5, 'variance %')
```

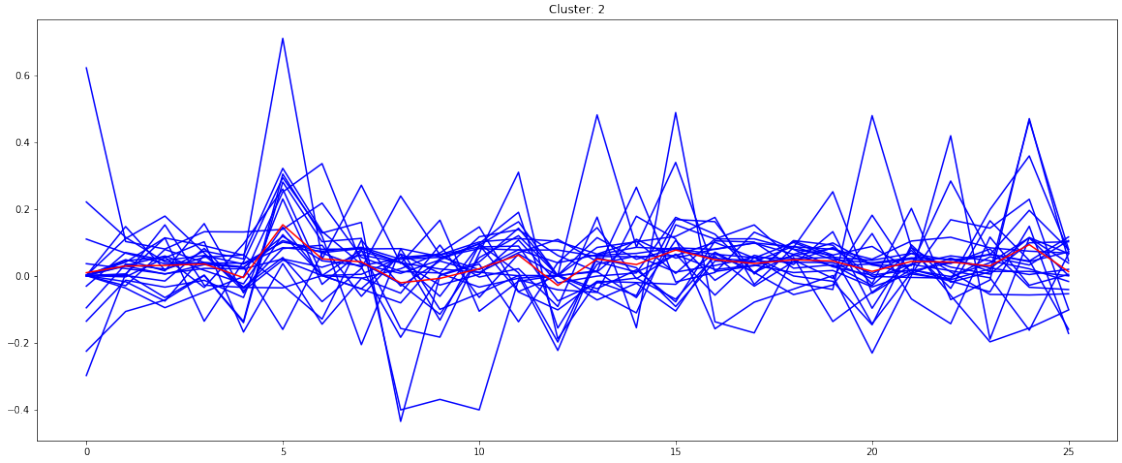


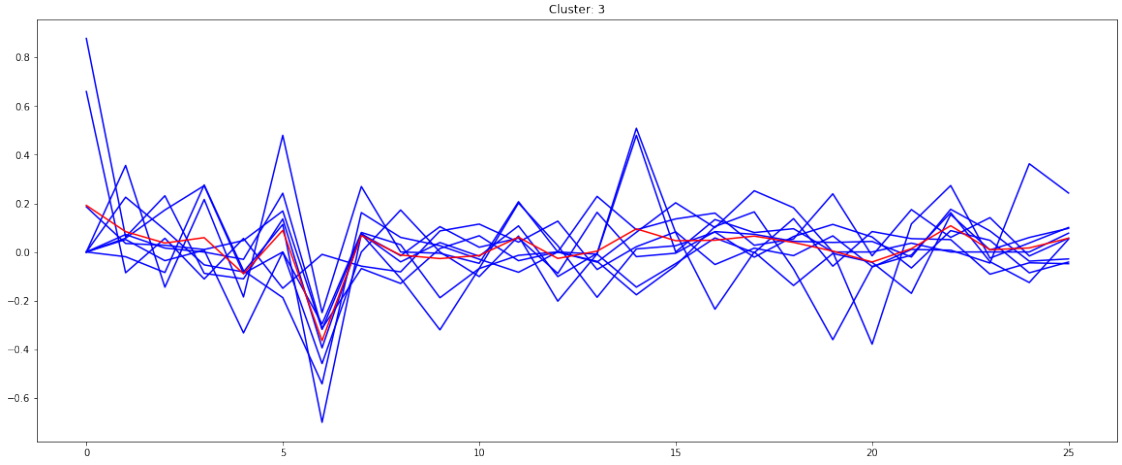
Clusters of Countries We want to analyze how these countries can be grouped in 5 clusters. In the graphs below, the sentiment scores of the countries belonging to that cluster are plotted. The red line represents the mean of the sentiment scores of the countries belonging to that cluster.

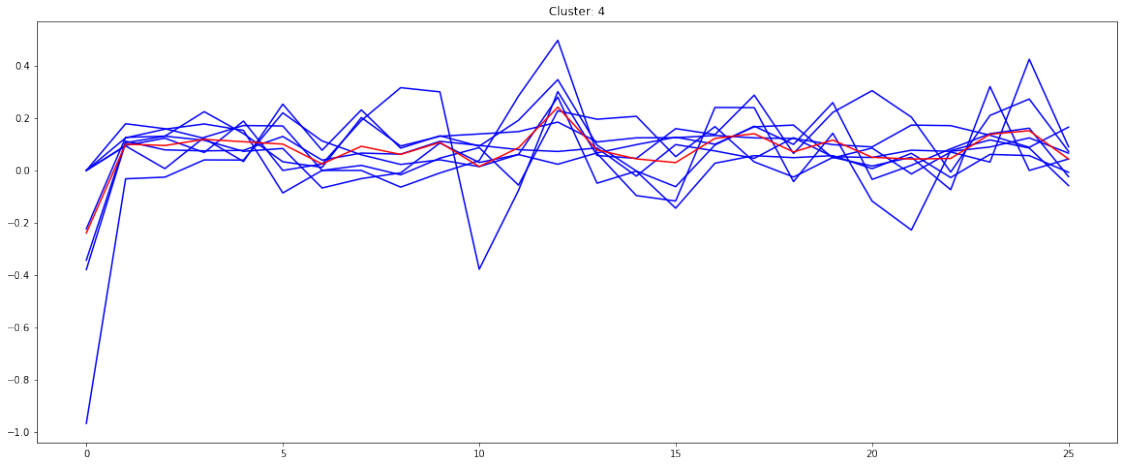
[114]:
```
from sklearn.cluster import KMeans
kmeans_model = KMeans(n_clusters=5, random_state=1).fit(sign) # https://
 ↪scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
k_means_labels = kmeans_model.labels_ # get the index to the cluster assigned to␣
 ↪each text

def group_arcs(sign, labels):
    grouped_sentiment_archs = {}
    for i in range(len(labels)):
        if labels[i] not in grouped_sentiment_archs.keys():
            grouped_sentiment_archs[labels[i]] = []
        grouped_sentiment_archs[labels[i]].append(sign[i])
    return grouped_sentiment_archs

def visualize_arcs_with_averages(grouped_sentiment_archs):
    averaged_clusters = np.array([np.mean(arches, axis=0) for arches in␣
 ↪grouped_sentiment_archs.values()])
    for i, (k, v) in enumerate(grouped_sentiment_archs.items()):
        plt.figure()
        plt.title('Cluster: {}'.format(i+1))
        for arch in v:
```

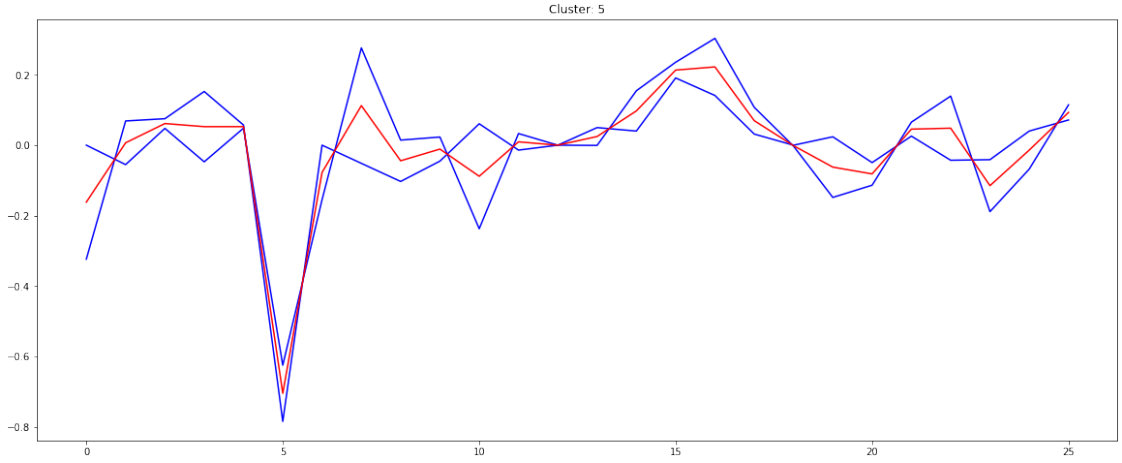
```
            plt.plot(arch, c='b')
        plt.plot(averaged_clusters[i], c='r')
        fig = plt.gcf()
        fig.set_size_inches(20,8)

grouped_sentiment_archs = group_arcs(sign, k_means_labels)
visualize_arcs_with_averages(grouped_sentiment_archs)
```






Cluster: 3
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
5
10
15
20
25


Cluster: 4
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
5
10
15
20
25

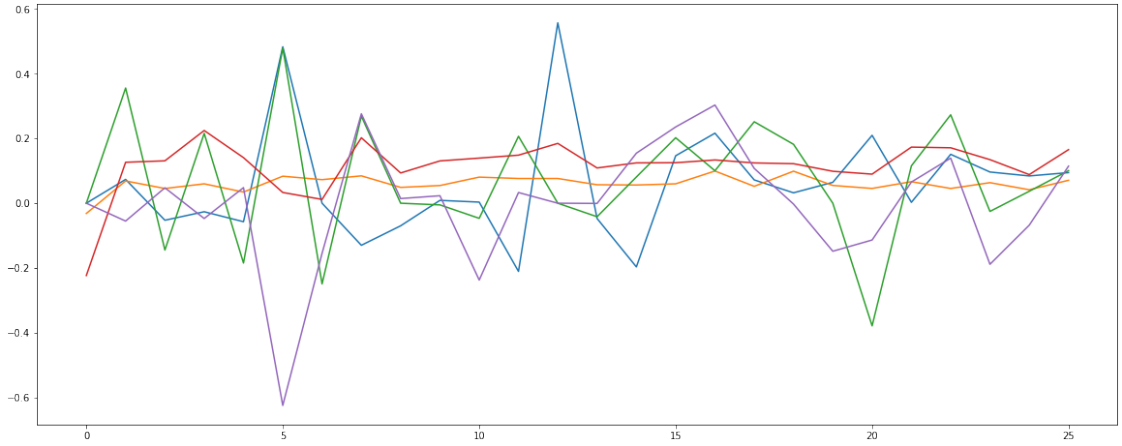

Cluster: 5
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
5
10
15
20
25

[115]:
```
def find_clusters_centers(docs_labels, sim_m):
    doc_groups = {}
    for i in range(len(docs_labels)):
        l = docs_labels[i]
        if l not in doc_groups.keys():
            doc_groups[l] = [i]
        else:
            doc_groups[l].append(i)
    print('documents clusters:\n{}'.format(doc_groups))
    centers_indices = []
    for k, v in doc_groups.items():
        assert len(np.mean(sim_m[v], axis=1)) == len(v)
        print('cluster {} similarity scores={}'.format(k, np.nanmean(sim_m[v],␣
 ↪axis=1)))
        curr_g_center = v[np.argmax(np.nanmean(sim_m[v], axis=1))]
        centers_indices.append(curr_g_center)
    return centers_indices, doc_groups

centers_indices, doc_groups = find_clusters_centers(k_means_labels, sim_m)
print('central documents indices: {}'.format(centers_indices))
```

```
documents clusters:
{0: [0, 4, 32, 39, 42], 3: [1, 3, 5, 10, 11, 12, 13, 14, 16, 17, 18, 20, 25, 26,
28, 37, 38, 40, 41, 43, 44, 45, 46], 1: [2, 7, 8, 19, 22, 30, 31, 33, 34], 2:
[6, 9, 15, 21, 23, 27, 29, 35], 4: [24, 36]}
cluster 0 similarity scores=[ 0.13706272  0.16186741  0.18404766  0.12852422
-0.09912803]
cluster 3 similarity scores=[0.07276489 0.04310862 0.27221498 0.05475413
0.02607478 0.1218317
 0.22703929 0.22616112 0.20195901 0.25997671 0.26963764 0.06827816
 0.06683721 0.18113985 0.02807986 0.15868203 0.08478688 0.1121265
 0.28265038 0.12596308 0.22785246 0.32526778 0.26702539]
cluster 1 similarity scores=[ 0.06331021  0.04126572  0.01246332 -0.04197029
0.12832152  0.18012796
  0.25788986  0.11095931  0.16532803]
cluster 2 similarity scores=[0.08216997 0.19932571 0.26562652 0.25259455
0.22665488 0.30404595
 0.29408895 0.16712258]
cluster 4 similarity scores=[-0.01126403 -0.05327304]
central documents indices: [32, 45, 31, 27, 24]
```

[116]:
```
def plot_arches(arches):
    plt.figure()
    for arch in arches:
        plt.plot(arch)
        fig = plt.gcf()
    fig.set_size_inches(20,8)
```

```
plot_arches(sign[centers_indices])
```



[117]:
```
for i in centers_indices:
    print(tab.columns[i])
```

```
Qatar
United Kingdom
Poland
Nigeria
Mexico
```

After having defined the different clusters behavior, we can create a DataFrame where we highlight in which cluster each country belongs.

[118]:
```
def cluster(i):
    cluster = []
    for j in doc_groups[i]:
        cluster.append(tab.columns[j])
    return cluster

cluster_0 = cluster(0)
cluster_1 = cluster(1)
cluster_2 = cluster(2)
cluster_3 = cluster(3)
cluster_4 = cluster(4)
```

[119]:
```
clusterSeries = pd.Series(index = tab.columns)
for x in cluster_0:
    clusterSeries[x] = 'Cluster 0'
for x in cluster_1:
    clusterSeries[x] = 'Cluster 1'
```

```
for x in cluster_2:
    clusterSeries[x] = 'Cluster 2'
for x in cluster_3:
    clusterSeries[x] = 'Cluster 3'
for x in cluster_4:
    clusterSeries[x] = 'Cluster 4'
```

```
<ipython-input-119-588eb6725158>:1: DeprecationWarning:

The default dtype for empty Series will be 'object' instead of 'float64' in a
future version. Specify a dtype explicitly to silence this warning.
```

[120]:
```
country_iso3_clust = pd.Series(clusterSeries.index).apply(get_country_iso3)
cluster_countries = pd.DataFrame({'country': list(clusterSeries.index),␣
 ↪'country_iso3': country_iso3_clust, 'cluster': clusterSeries.values})
```

[121]:
```
cluster_countries
```

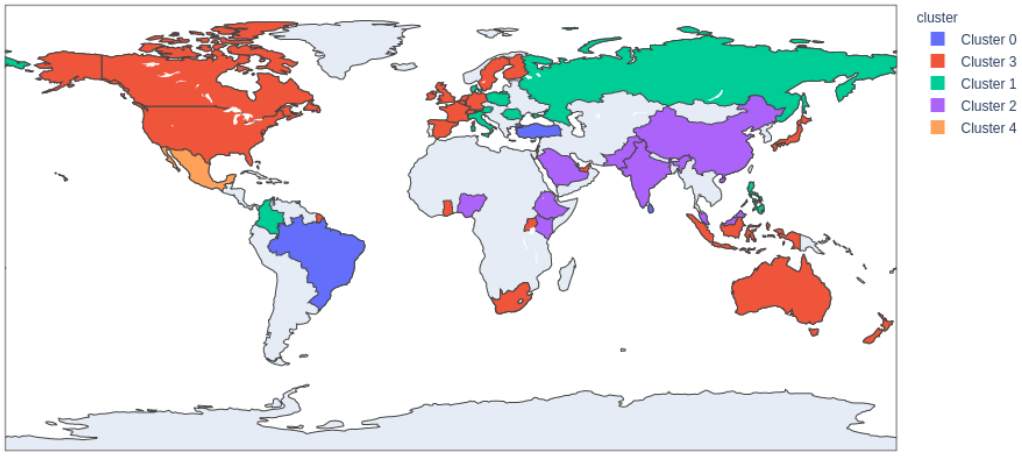
[121]:

	country	country_iso3	cluster
0	Africa	ZAF	Cluster 0
1	Australia	AUS	Cluster 3
2	Austria	AUT	Cluster 1
3	Belgium	BEL	Cluster 3
4	Brazil	BRA	Cluster 0
5	Canada	CAN	Cluster 3
6	China	CHN	Cluster 2
7	Colombia	COL	Cluster 1
8	Denmark	DNK	Cluster 1
9	Ethiopia	ETH	Cluster 2
10	Europe	None	Cluster 3
11	Finland	FIN	Cluster 3
12	France	FRA	Cluster 3
13	Germany	DEU	Cluster 3
14	Ghana	GHA	Cluster 3
15	India	IND	Cluster 2
16	Indonesia	IDN	Cluster 3
17	Ireland	IRL	Cluster 3
18	Israel	ISR	Cluster 3
19	Italy	ITA	Cluster 1
20	Japan	JPN	Cluster 3
21	Kenya	KEN	Cluster 2
22	Luxembourg	LUX	Cluster 1
23	Malaysia	MYS	Cluster 2
24	Mexico	MEX	Cluster 4
25	Netherlands	NLD	Cluster 3
26	New Zealand	NZL	Cluster 3

27	Nigeria	NGA	Cluster 2
28	North America	None	Cluster 3
29	Pakistan	PAK	Cluster 2
30	Philippines	PHL	Cluster 1
31	Poland	POL	Cluster 1
32	Qatar	QAT	Cluster 0
33	Romania	ROU	Cluster 1
34	Russia	RUS	Cluster 1
35	Saudi Arabia	SAU	Cluster 2
36	Singapore	SGP	Cluster 4
37	South Africa	ZAF	Cluster 3
38	Spain	ESP	Cluster 3
39	Sri Lanka	LKA	Cluster 0
40	Sweden	SWE	Cluster 3
41	Switzerland	CHE	Cluster 3
42	Turkey	TUR	Cluster 0
43	Uganda	UGA	Cluster 3
44	United Arab Emirates	ARE	Cluster 3
45	United Kingdom	GBR	Cluster 3
46	United States	USA	Cluster 3

We can finally plot on a map the classification in clusters.

[122]:
```
fig_cluster_countries = px.choropleth(cluster_countries ,
                                      locations="country_iso3",
                                      color="cluster",
                                      hover_name="country")

fig_cluster_countries.show()
```



Sentiment Analysis Spatial Temporal distribution

In this last section of sentiment analysis, we plot an animated map that shows the sentiment score of each country in each day of the period that we are considering.

[123]:
```
df.head()
```

[123]:
```
             user_name   user_location  user_verified   date  \
0     insightfultroll              NaN          False  07-24
1  GlobalPandemic.NET  United Kingdom          False  07-24
2           Euan Watt  United Kingdom          False  07-24
3  GlobalPandemic.NET  United Kingdom          False  07-24
4               Brian   United States          False  07-24

                                                text     hashtags  \
0  [i'm glad i'm in canada, you can see why the p...          NaN
1  [alert: after times investigation, newsom says...          NaN
2  [having checked the data of all nation's of th...          NaN
3  [alert: doctors post bikini photos to protest ...          NaN
4  [ but not a word about #covid19 spread for the...  ['COVID19']

                                     text_tokenized  compound text_score
0  [glad, canada, see, pandemic, raging, united, ...    0.3400   positive
1  [alert, times, investigation, newsom, says, nu...    0.2960   positive
2              [checked, data, uk, deaths, england]    0.0000    neutral
3  [alert, doctors, post, bikini, photos, protest...   -0.4767   negative
4  [word, spread, last, days, portland, city, mas...    0.0000    neutral
```

[124]:
```
df_withloc = df.dropna(subset = ['user_location'])
```

[125]:
```
df_withloc = df_withloc[['user_location','date','compound']]
```

[126]:
```
df_withloc
```

[126]:
```
         user_location   date  compound
1       United Kingdom  07-24    0.2960
2       United Kingdom  07-24    0.0000
3       United Kingdom  07-24   -0.4767
4        United States  07-24    0.0000
5       United Kingdom  07-24   -0.3818
...                ...    ...       ...
179095   United States  08-30   -0.1366
179096   United States  08-30   -0.5945
179099           Japan  08-30    0.0000
179103        Malaysia  08-30    0.4019
```

```
179105            Asia  08-30   -0.3612

[124049 rows x 3 columns]
```

[127]:
```
grouped = df.groupby('user_location')
```

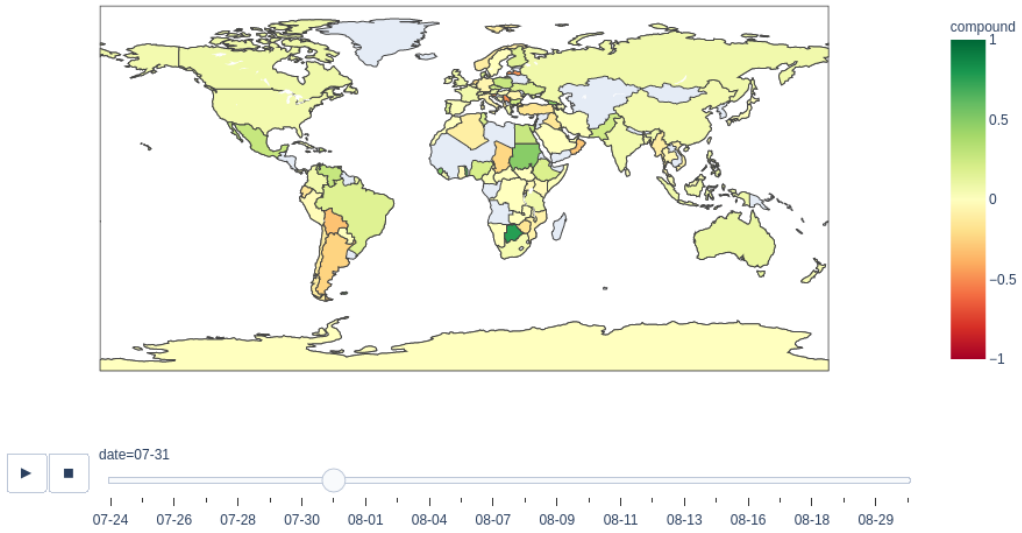
[128]:
```
ls = []
for country in df_withloc.user_location.unique():
    ls.append(grouped.get_group(country).groupby(['user_location', 'date'],␣
 ↪as_index = False)['compound'].mean())
```

[129]:
```
data = pd.concat(ls)

data
```

[129]:
```
                       user_location   date  compound
0                      United Kingdom  07-24 -0.031704
1                      United Kingdom  07-25  0.068794
2                      United Kingdom  07-26  0.045525
3                      United Kingdom  07-27  0.059901
4                      United Kingdom  07-28  0.034477
..                                ...    ...       ...
0                         Cook Islands  08-22 -0.757900
0   Oceanic Point of Inaccessibility  08-22  0.822500
0                    Marshall Islands  08-29  0.458800
1                    Marshall Islands  08-30  0.107200
0                               Palau  08-29  0.410000

[2977 rows x 3 columns]
```

[130]:
```
fig_compound_per_country_animation = px.choropleth(
    data, #Dataframe
    locations= 'user_location', #Spatial coordinates, can give Lat and Lon in␣
 ↪separate params
    locationmode= 'country names', #Type of spatial coordinates
    color= 'compound', #Values to be color coded
    range_color=[-1,1],
    hover_name= 'user_location', #Text to be displayed in Bold upon hover
    animation_frame= 'date', #Data for animation, time-series data
    color_continuous_scale=px.colors.diverging.RdYlGn
)
```

[131]:
```
fig_compound_per_country_animation.show()
```



Dash Board We want to summarize all the results found so far in a more compact and user-friendly way through a Dash Board.

[132]:
```
#!pip3 install dash
```

[133]:
```
#!pip3 install dash_bootstrap_components
```

[134]:
```
import dash
import dash_html_components as html
import dash_core_components as dcc
import dash_bootstrap_components as dbc
from dash.dependencies import Input, Output, State

import plotly.express as px

from io import BytesIO
import base64
from flask import Flask
```

[135]:
```
# List of country
list_country = list(set(df['user_location'].dropna()))
list_country = sorted(list_country)

# List of day
list_day = list(set(df['date']))
list_day = sorted(list_day)
```

```
# Number of users
number_of_users = len(list(set(df['user_name'])))

#Number of tweets
number_of_tweets = len(df['text'])

#Number of countries
number_of_countries = len(list_country)
```

[136]:
```
def hashtag_wordcloud_to_img(data):
    wc = WordCloud(width=800, height=400, max_words=50).
 ↪generate_from_frequencies(data)  # create the wordcloud
    return wc.to_image()
```

[137]:
```
def freq_discrete_df(data):
    keys = ['negative', 'neutral', 'positive']
    values = []
    pos = list(data.text_score).count("positive")
    neu = list(data.text_score).count("neutral")
    neg= list(data.text_score).count("negative")
    values.extend([neg,neu,pos])
    result = list(zip(keys,values))
    result = pd.DataFrame(result, columns = ['keys','values'])
    return result
```

[138]:
```
SIDEBAR_STYLE = {
    'top': '0px',
    'background-color': '#f8f9fa',
    'float':'left'
}

CONTENT_STYLE = {
    'margin-left': '25%',
    'margin-right': '5%',
    'top': '0px',
    'padding': '20px 10px'
}

MAIN_STYLE = {
    'top': '0px',
    'padding': '20px 10px'
}

TEXT_STYLE = {
    'textAlign': 'center',
    'color': '#191970'
}
```

```
CARD_TEXT_STYLE = {
    'textAlign': 'center',
    'color': '#0074D9',
    'font-size': '40px',
    'font-weight': 'bold'
}

CARD_TITLE_STYLE = {
    'textAlign': 'center',
    'color': '#0074D9',
    'font-size': '19px'
}

IMG_STYLE = {
    'textAlign': 'center',
    'max-width': '100%',
    'max-height': '100%',
    'display': 'block',
    'float' : 'right'
}

FIGURE_TITLE = {
  'textAlign': 'center',
  'font-weight': 'bold',
  'display': 'flex',
  'flex-direction': 'column',
  'align-items': 'center'
}

VIDEO_STYLE = {
    'display' : 'block',
    'margin': '0 auto',
    'display': 'flex',
    'flex-direction': 'column',
    'align-items': 'center'
}
```

[139]:
```
controls_tab1 = dbc.FormGroup(
    [
        html.Br(),
        html.P('Wordcloud of hashtags filter:', style={
            'textAlign': 'center'
        }),
        dbc.Card([dbc.RadioItems(
            id='radio_items',
            options=[{
```

```
                'label': 'Normal',
                'value': 'normal'
            },
            {
                'label': 'Without #covid and #corona',
                'value': 'without'
            },
            {
                'label': 'Verified users',
                'value': 'verified'
            },
            {
                'label': 'Not verified users',
                'value': 'nverified'
            },
            {
                'label': 'Country',
                'value': 'country'
            }],
            value = 'normal',
            style={
                'margin': 'auto'
            }
        )]),
        html.Br(),
        html.P('Country', style={
            'textAlign': 'center'
        }),
        dcc.Dropdown(
            id='dropdown_country_1',
            options=[{'label': country,'value': country}
                    for country in list_country],
            value='United States',
            clearable=False
        ),
        html.Br(),
        dbc.Button(
            id='submit_button_1',
            n_clicks=0,
            children='Submit',
            color='primary',
            block=True
        )
    ]
)
```

[140]:
```
trends_first_row = dbc.Row([
    dbc.Col(children =
        [dbc.Card(
            [
                dbc.CardBody(
                    [
                        html.H4(children=['Date'], className='card-title',
                                style=CARD_TITLE_STYLE),
                        html.P(children=['07/24 - 08/30'],␣
↪style=CARD_TEXT_STYLE),
                    ]
                )
            ]
        ),
         dbc.Card(
            [

                dbc.CardBody(
                    [
                        html.H4(children=['Number of users'],␣
↪className='card-title', style=CARD_TITLE_STYLE),
                        html.P(number_of_users, style=CARD_TEXT_STYLE),
                    ]
                ),
            ]
        )
    ], md = 6),
    dbc.Col(children =
        [dbc.Card(
            [
                dbc.CardBody(
                    [
                        html.H4(children=['Number of tweets'],␣
↪className='card-title', style=CARD_TITLE_STYLE),
                        html.P(number_of_tweets, style=CARD_TEXT_STYLE),
                    ]
                ),
            ]
        ),
        dbc.Card(
            [
                dbc.CardBody(
                    [
                        html.H4(children=['Number of countries'],␣
↪className='card-title', style=CARD_TITLE_STYLE),
                        html.P(number_of_countries, style=CARD_TEXT_STYLE),
                    ]
```

```
                ),
            ]
        )
       ], md=6),
    dbc.Col(children =
            [dcc.Graph(id='number_of_tweet_animation', figure =␣
↪figure_number_of_tweets_per_country),
             html.Br(),
             html.P('Number of tweets animation for each country', style =␣
↪FIGURE_TITLE)], md = 12
    )
])
```

[141]:
```
trends_second_row = dbc.Row(
    [
        dbc.Col(children =
            [dcc.Graph(id='number_of_tweets', figure = fig_frq_words),
             html.Br(),
             html.P('Frequency of the most 20 common words in tweets', style =␣
↪FIGURE_TITLE)], md=6
        ),
        dbc.Col(children =
            [dcc.Graph(id='number_of_tweets_per_day', figure =␣
↪fig_number_of_tweets_per_day),
             html.Br(),
             html.P('Number of tweets per day', style = FIGURE_TITLE)], md=6)
    ]
)
```

[142]:
```
sidebar_tab1  = html.Div([controls_tab1], style=SIDEBAR_STYLE)
```

[143]:
```
trends_third_row = dbc.Row([
    dbc.Col(md = 2),
    dbc.Col(sidebar_tab1, md = 2),
    dbc.Col(html.Img(id="image_wc", style = IMG_STYLE), md = 6),
    dbc.Col(md = 2)
])
```

[144]:
```
trends_fourth_row = dbc.Row(
    [
        dbc.Col(children =
            [html.Video(id = 'bcr_hashtags',src='/assets/bar_chart_race_hashtags.
↪mp4',controls=True, style = VIDEO_STYLE),
             html.Br(),
             html.P('Bar chart race for hashtags without #covid and #corona',␣
↪style = FIGURE_TITLE)],
```

```
            md=12
        )
    ]
)
```

[145]:
```
trends_fifth_row = dbc.Row(
    [
        dbc.Col(children =
            [html.Video(id = 'bcr_countries',src='/assets/
 ↪bar_chart_race_countries.mp4',controls=True, style = VIDEO_STYLE),
            html.Br(),
            html.P('Bar chart race for number of tweets per country', style =␣
 ↪FIGURE_TITLE)],
            md=12
        ),
    ]
)
```

[146]:
```
controls_tab2_1 = dbc.FormGroup([
        html.Br(),
        html.P('Sentiment analysis for country', style={
            'textAlign': 'center'
        }),
        html.Br(),
        html.P('Choose a country:', style={
            'textAlign': 'center'
        }),
        dcc.Dropdown(
            id='dropdown_country_2',
            options=[{'label': country,'value': country}
                    for country in list_country],
            value='United States',
            clearable=False
        ),
        html.Br(),
        dbc.Button(
            id='submit_button_2_1',
            n_clicks=0,
            children='Submit',
            color='primary',
            block=True
        )
])
```

[147]:
```
sidebar_tab2_1 = html.Div([controls_tab2_1], style=SIDEBAR_STYLE)
```

[148]:
```
sentiment_analysis_first_row = dbc.Row(children =
    [
        dbc.Col(sidebar_tab2_1, md = 2),
        dbc.Col(dcc.Graph(id = 'sentiment_analysis_country_pie_chart'), md = 5),
        dbc.Col(dcc.Graph(id = 'sentiment_analysis_country_dist_plot'), md = 5)
    ]
)
```

[149]:
```
sentiment_analysis_second_row = dbc.Row(children =
    [
        dbc.Col(children =
            [dcc.Graph(id='sentiment_score', figure = fig_compound),
            html.Br(),
            html.P('Sentiment score for tweets', style = FIGURE_TITLE)], md = 12
        )
    ])
```

[150]:
```
controls_tab2_2 = dbc.FormGroup([
        html.Br(),
        html.P('Average sentiment score per day', style={
            'textAlign': 'center'
        }),
        html.Br(),
        html.P('Choose a day:', style={
            'textAlign': 'center'
        }),
        dcc.Dropdown(
            id='dropdown_day',
            options=[{'label': day,'value': day}
                    for day in list_day],
            value='07-24',
            clearable=False
        ),
        html.Br(),
        dbc.Button(
            id='submit_button_2_2',
            n_clicks=0,
            children='Submit',
            color='primary',
            block=True
        )
])
```

[151]:
```
sidebar_tab2_2 = html.Div([controls_tab2_2], style=SIDEBAR_STYLE)
```

[152]:
```
sentiment_analysis_third_row = dbc.Row(children =
    [   dbc.Col(md=2),
```

```
        dbc.Col(sidebar_tab2_2, md = 2),
        dbc.Col(dcc.Graph(id ='compound_by_country_per_day'), md=6),
        dbc.Col(md=2)
    ])
```

[153]:
```
sentiment_analysis_fourth_row = dbc.Row(children =
    [
        dbc.Col(children =
            [dcc.Graph(id = 'cluster_countries', figure = fig_cluster_countries),
            html.P('Countries clustering', style = FIGURE_TITLE)], md=12
        )
    ])
```

[154]:
```
content = html.Div(
    [
        html.H2('Covid-19 Tweets Analytics Dashboard ', style=TEXT_STYLE),
        html.Hr(),
        html.P('The dataset is collected using Twitter API and a Python script.␣
 ↪A query for this high-frequency hashtag (#covid19) is run daily for a certain␣
 ↪time period, to collect a larger number of tweets samples.', style=TEXT_STYLE),
        dcc.Tabs(id='tab', value = 'tab', children=[
            dcc.Tab(label='Analyzing Trends', value = 'trends', children=[
                html.Br(),
                trends_first_row,
                html.Br(),
                trends_second_row,
                html.Br(),
                trends_third_row,
                html.Br(),
                trends_fourth_row,
                html.Br(),
                trends_fifth_row
            ]),
            dcc.Tab(label='Sentiment Analysis', value = 'sentiment_analysis',␣
 ↪children=[
                sentiment_analysis_first_row,
                html.Br(),
                sentiment_analysis_second_row,
                html.Br(),
                sentiment_analysis_third_row,
                html.Br(),
                sentiment_analysis_fourth_row
            ])
        ])
    ],
    style=MAIN_STYLE
)
```

[155]:
```
server = Flask(__name__)
app = dash.Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP], server =␣
↪server)
app.layout = html.Div([content])
app.title = 'Twitter Sentiment Analysis'
```

[156]:
```
#Video
@server.route('/assets/<path:path>')
def serve_static(path):
    root_dir = os.getcwd()
    return flask.send_from_directory(os.path.join(root_dir, 'assets'), path)
```

[157]:
```
# Wordcloud
@app.callback(Output('image_wc', 'src'),
              [Input('submit_button_1', 'n_clicks'), Input('image_wc', 'id')],
              [State('radio_items', 'value'), State('dropdown_country_1',␣
↪'value')])
def update_word_cloud(n_clicks, b, radio_items_value, dropdown_country_value):
    print(n_clicks)
    print(radio_items_value)
    print(dropdown_country_value)

    img = BytesIO()
    #normal
    if radio_items_value == 'normal':
        hashtag_wordcloud_to_img(fdist).save(img, format='PNG')
        return '{}'.format(base64.b64encode(img.
↪getvalue()).decode())
    #without hashtag covid or corona
    elif radio_items_value == 'without':
        hashtag_wordcloud_to_img(fdist_nocovid).save(img, format='PNG')
        return '{}'.format(base64.b64encode(img.
↪getvalue()).decode())
    #verified user
    elif radio_items_value == 'verified':
        fdist_verified = freq_hashtag_nocov(freq_hashtag(df_usver())[1])
        hashtag_wordcloud_to_img(fdist_verified).save(img, format='PNG')
        return '{}'.format(base64.b64encode(img.
↪getvalue()).decode())
    #not verified user
    elif radio_items_value == 'nverified':
        fdist_nverified = freq_hashtag_nocov(freq_hashtag(df_usver(False))[1])
        hashtag_wordcloud_to_img(fdist_nverified).save(img, format='PNG')
        return '{}'.format(base64.b64encode(img.
↪getvalue()).decode())
    #country
```

```
    elif radio_items_value == 'country':
        fdist_country =␣
↪(freq_hashtag_nocov(freq_hashtag(df_country(dropdown_country_value))[1]))
        hashtag_wordcloud_to_img(fdist_country).save(img, format='PNG')
        return '{}'.format(base64.b64encode(img.
↪getvalue()).decode())
```

[158]:
```
# Sentiment analysis (pie chart)
@app.callback(Output('sentiment_analysis_country_pie_chart', 'figure'),
              [Input('submit_button_2_1', 'n_clicks')],
              [State('dropdown_country_2', 'value')])
def update_pie_chart(n_clicks, dropdown_country_value):
    print(n_clicks)
    print(dropdown_country_value)
    freq_text_score = freq_discrete_df(df_country(dropdown_country_value))
    fig_sentiment_analysis_pie = px.pie(freq_text_score, values = 'values',␣
↪names = 'keys')
    return fig_sentiment_analysis_pie
```

[159]:
```
# Sentiment analysis (dist plot)
@app.callback(Output('sentiment_analysis_country_dist_plot', 'figure'),
              [Input('submit_button_2_1', 'n_clicks')],
              [State('dropdown_country_2', 'value')])
def update_dist_plot(n_clicks, dropdown_country_value):
    print(n_clicks)
    print(dropdown_country_value)
    text_score_by_country = df_country(dropdown_country_value).
↪loc[df_country(dropdown_country_value).compound != 0]
    fig_sentiment_analysis_dist = px.histogram(text_score_by_country, x =␣
↪'compound')
    return fig_sentiment_analysis_dist
```

[160]:
```
df_bycountry = df.groupby(["user_location", 'date'], as_index =␣
↪False)["compound"].mean()
country_iso = pd.Series(df_bycountry['user_location']).apply(get_country_iso3)
df_byloc = pd.DataFrame({'country': list(df_bycountry['user_location']),␣
↪'country_iso3': country_iso, 'date': list(df_bycountry['date']), 'compound':␣
↪df_bycountry['compound']})
df_byloc = df_byloc.dropna()
```

[161]:
```
# Sentiment score for countries per day
@app.callback(Output('compound_by_country_per_day', 'figure'),
              [Input('submit_button_2_2', 'n_clicks')],
              [State('dropdown_day', 'value')])
def update_choropleth_plot(n_clicks, dropdown_day_value):
    print(n_clicks)
```

```
    print(dropdown_day_value)
    df_byloc_by_day = df_byloc.loc[df_byloc['date'] == dropdown_day_value]
    fig_compound_by_country = px.choropleth(df_byloc_by_day,
                    locations="country_iso3",
                    color="compound",
                    hover_name="country",
                    color_continuous_scale= 'RdYlGn', range_color = [-1,1])
    return fig_compound_by_country
```

[ ]:
```
# Run the app
if __name__ == '__main__':
    app.run_server(debug=True, use_reloader=False)
```

```
Dash is running on http://127.0.0.1:8050/

 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
   Use a production WSGI server instead.
 * Debug mode: on
```


References Bird, S., Loper E. & Klein, E. (2009). *Natural Language Processing with Python*. O'Reilly Media Inc.

Hunter, J. D. (2007). *Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95*. DOI:10.1109/MCSE.2007.55

Kim, Ricky (2017). *Another Twitter sentiment analysis with Python-Part 2*. Towards Data Science.

McKinney, W., & others. (2010). *Data structures for statistical computing in python*. In Proceedings of the 9th Python in Science Conference.

Oliphant, T. E. (2006). *A guide to NumPy (Vol. 1)*. Trelgol Publishing USA.

Pandey, P. (2018). *Simplifying Sentiment Analysis using VADER in Python (on Social Media Text)*. Analytics Vidhya

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research.

Plotly Technologies Inc. (2015). *Collaborative data science*. Plotly Technologies Inc., Montréal, QC. https://plot.ly