Introduction to Data Science - Coursework

November 24, 2021

Contents

List of Figures

1 Dataset Description

According the coursework handout, the dataset consists of tweets extracted using Twitter API from the European region specified by a bounding box specified by the coordinates (-24.5, 24.8) and (69.1,81.9) for the left-lower corner and upper-right corner respectively. The extraction of tweets was carried out for the period of June without any other filters. The resulting zip file is of 8.2GB of size, and 30GB when unzipped which makes it hard and slow to work with.

2 Tasks

2.1 Part 1 - Basic Stats

2.1.1 Loading Data

The folder contains 720 (24hx30 days) files. Each file corresponds to an extraction worth of one hour of tweets. In order start processing the data, it was first converted to 720 smaller CSV files that contains only the needed bits of information. The documentation of twitter JSON objects is provided on Twitter (2021). The function ***saveCSV()*** was built in order to access each individual json file, get only the data needed for the rest of the tasks, and save it in a csv file. The function take as argument the directory that contains the data extracted from twitter and the target directory where it will save the csv files. For this function we have used json, zipfile, and pandas libraries. First, using the below line, we accessed the downloaded twitter directory

```
general_archive = zipfile.ZipFile(zip_file_path,'r')
```

Next, using a for loop, we iterated over all the files contained in ***general_archive***, extracted them, and saved each line in a list.

```
for f in general_archive.namelist():

        archive = zipfile.ZipFile(f, 'r')
        # construct the path of all files based on the pattern of the
    above path
        file_path ="geoEurope/"+f[16:36]+".json"

        # access each file individually
        file = archive.open(file_path)
        df = pd.DataFrame()
        tweet = []
        for s in file.readlines():
            tweet.append(json.loads(s.decode("utf-8")))
```

After we obtained the list of JSON objects in the file, we used ***json_normalize()***, a pandas function, to unwrap the JSON object at multiple levels, the code is shown below. We only keep:

- **id_str:** the unique ID of the tweet.
- **created_at:** date and time at which the tweet was created.

- **text:** the content of the tweet
- **coordinates.coordinates:** the coordinates for the geo-tagged tweets. The actual country can be derived later using reverse geocoding.
- **user.id_str:** a unique ID for the user
- **user.screen_name:** a unique name of the user, this is the name that appears on the screen.
- **entities.user_mentions:** this is the list of users mentioned in that specific tweet.

```
df = pd.json_normalize(tweet, meta ='user')[['id_str','created_at','
    text','coordinates.coordinates','user.id_str','user.screen_name','
    entities.user_mentions']]
```

The last object, ***entities.user_mentions*** is extracted as as JSON object as well, it contains other additional information such as ***media, symbols, and urls*** (the list is non exhaustive). The only attribute we need is ***id_str***. Using the following code, we unwrapped the data in that object, and created a list of user ids mentioned using only ***id_str*** attribute.

```
# convert lists of json objects of user_mentions to a simple list
    containing all mentionned users
data_mentions = df['entities.user_mentions'].apply(lambda x: pd.
    json_normalize(x) if len(x)> 0 else None)

for label, content in data_mentions.iteritems():
    if content is None:
        pass
    else:
        df['entities.user_mentions'].loc[label] = [int(x) for x in
    list(content['id_str'])]
            print('finished normalizing')
```

The dataset contains some missing tweets that are ignored by catching the exception and print an error message. The resulting directory contains 720 CSV files of total size 3GB (reduced by 10 times). Loading the CSV files by creating a DataFrame then append the new CSV file is memory and time consuming since pandas needs to create a copy of the previous DataFrame each time it needs to append a DataFrame to it. A better way to do it is to load all csv files in a list then convert it to a DataFrame. The following code snippet describes how the above steps were implemented.

```
li = []
for i in range(1,721):
    file_path = 'reduced_data/csv'+str(i)+'.csv'
    try:
        df = pd.read_csv(file_path, engine = 'python')
        li.append(df)
        print('Loading file: ', i)
    except: print('Error....',i)
df = pd.concat(li, axis=0)
```

```
df.drop(labels=['Unnamed: 0'], axis=1, inplace=True)
df.reset_index(inplace=True)
del li
```

Running ***info()*** on ***df*** shows that there are **13,861,538 tweets** excluding empty tweets. The dataset contains duplicates as well, we can drop them by calliing:

```
df.drop_duplicates(inplace=True)
```

We also looked at the NaNs in our dataset; some of the columns does not really matter if they contain NaNs, for example, if the text column contains some NaNs, it will not matter when we are plotting the number of tweets per day or building a map. However, we checked for rows that are entirely NaN and removed them (at this step there was only one since we removed duplicates before). Also, we checked the format of the date and removed those rows with wrong data in the date column. We could for example take the data points that have the date attribute as NaN and fill it with the date of the next/previous rows to preserve all rows that are NaNs for our timeseries analysis. However, the total number of removed tweets is 4,375, it represents almost 0.3% of the data. Finally, we are left with **13,857,163 tweets**.

2.1.2 Number of tweets by day

We noticed from our data that it contains 1 hour worth of tweets for June 25th, however that is out of scope of our analysis as it does not represent the entire day. In order to get the number of tweets per day, we resample the data per day and count the number of data points we have per day using the following line:

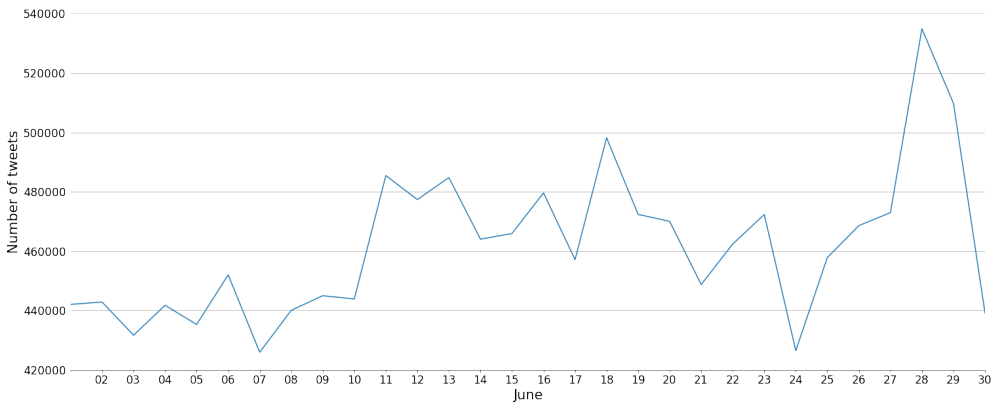
```
daily_tweets = df.loc['2021-06-01':]['created_at'].resample('D').count
    ()
```

In order to plot ***daily_tweets***, we have written a function ***plot_daily()*** (that can be used later as well) which used ***plt.plot()*** from ***matplotlib*** library. The function takes as argument the resampled DataFrame, xlabel, ylabel, path to the saved image, and both min and max xlim components. The result is shown in figure 1. We notice a low number of tweets in the first and third weeks. Whereas there is a relatively large number of tweets in the second and last week of June. In the 7th and 24th of June the number of tweets decreased to around 425k tweets and increased in the 28th to 534k in four days.

2.1.3 Whisker-and-Box Diagram

In order to plot the box-and-whisker diagram, we used the same variable ***daily_tweets***. In order to distinguish between weekends and weekdays, we use the ***weekday*** attribute for the date index of our data. If the weekday is less than 5, then it is a weekday. Otherwise, it is a weekend. The below code snippet shows how ***boxplot()*** method was used to create the plot. Other lines of code were added for the styling of the diagram.

Figure 1: Number of tweets per day in the month of June

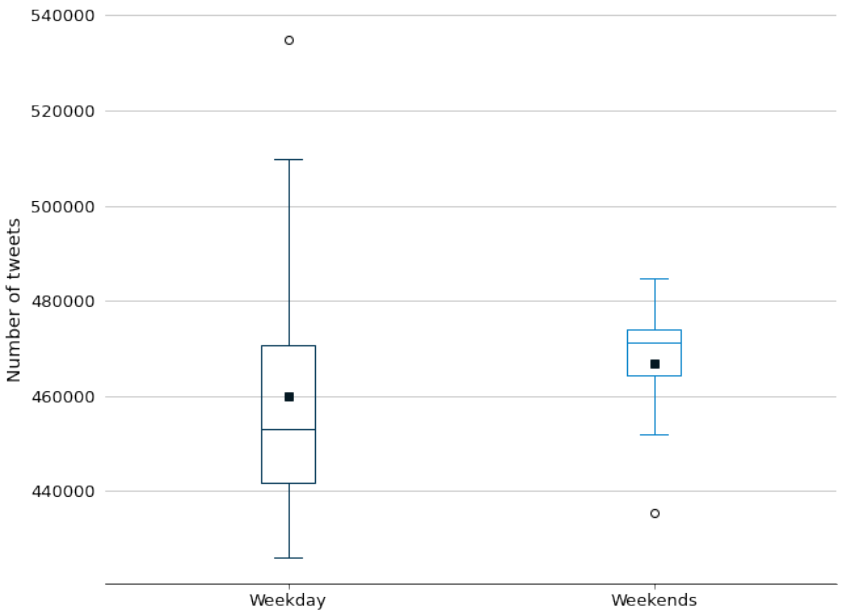



```
mean_color = {"markerfacecolor":"#051923", "markeredgecolor":"#051923"
    ,"marker":'s'}

ax.boxplot(daily_tweets[daily_tweets.index.weekday < 5],positions
    =[0], showmeans=True,labels=['Weekday'], boxprops=weekday_color,
    medianprops=weekday_color, whiskerprops=weekday_color,capprops=
    weekday_color, meanprops=mean_color)
ax.boxplot(daily_tweets[daily_tweets.index.weekday > 4], positions =
    [1], showmeans=True, labels=['Weekends'], boxprops=weekend_color,
    medianprops=weekend_color, whiskerprops=weekend_color,capprops=
    weekend_color, meanprops=mean_color)
```

The above code results in a diagram that contains two box plots for both the number of tweets during weekends and during weekdays (figure 2).

From the box plot we can see that the distribution of the number of tweets during weekdays is right skewed whereas for the weekends is left skewed. We can also see from the plots that the number of tweets for weekdays is more spread than weekends. During weekdays most of the days receive around 445k and 480k tweets whereas during weekends most of the days receive around 465k to 470k. The diagrams show that the number of tweets during weekdays range from 420,000 to 515,000 whereas for weekends the number is between 455,000 to 485,000 approximately. However, 75% of the numbers of tweets during weekdays lie below 470,000 while 50% of the number of tweets during weekends lies above 475,000. We can clearly see that the average number of tweets during weekends is higher than during weekdays. In fact, the difference in the central tendency of both for weekends and weekdays is even bigger. The average number of tweets during weekdays is pulled to a bigger value due to the outlier on the 28th of June, 2021 and the average number of tweets during weekends is pulled down due to the outlier on the 7th of June, 2021 which brought them closer to each others. Overall, it is more likely to receive a larger number of tweets during weekends compared to weekdays.

Figure 2: Number of tweets during weekdays and weekends



2.1.4 Average number of tweets during weekdays and weekends

In order to get the number of tweets per hour, we resample ***df*** by hour ('H'), then count the number of rows. We use the ***weekday*** attribute on the index of the resulting Dataframe to test if the date in the index is a weekday or weekend as shown in the following code snippet.

```
hourly_tweets = df.loc['2021-06-01':]['created_at'].resample('H').
    count()
weekday_h_tweets = hourly_tweets[hourly_tweets.index.weekday < 5]
weekend_h_tweets = hourly_tweets[hourly_tweets.index.weekday > 4]
```

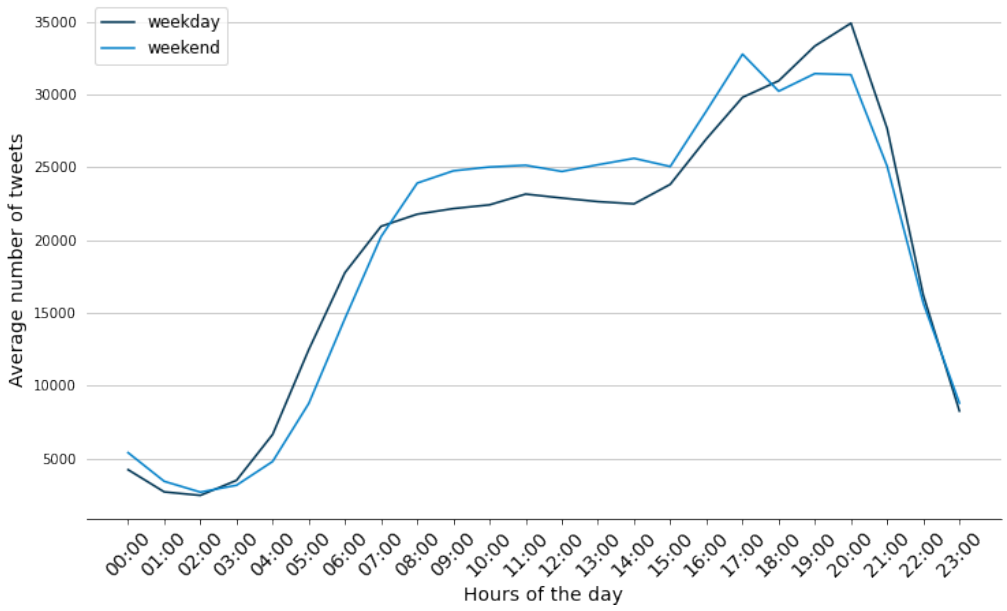
In order to get the average number of tweets per hour, we first group both ***weekday_h_tweets*** and ***weekend_h_tweets*** by hour and user the build in ***mean()*** method.

```
avg_weekend_h_tweets = weekend_h_tweets.groupby(weekend_h_tweets.index
    .hour,axis=0).mean()
avg_weekday_h_tweets = weekday_h_tweets.groupby(weekday_h_tweets.index
    .hour,axis=0).mean()
```

By plotting the previous result, we get figure 3.

From the graph we notice that the two plots intersect four times. Keeping the slight time difference in mind, from 7AM to 6PM the average number of tweets during weekends is higher than the average number of tweets during weekdays, that is probably due to the fact most people will be working or studying. However, after 6PM to 11PM, we notice that the average number of tweets during weekends falls below weekday, that

Figure 3: Average number of tweets per hour during weekdays and weekends



is most likely caused by the fact that people usually go out during weekends at night, while during workdays people may be staying at home at that time after a long day of hard work an studies. During weekdays, people are most likely to sleep early, as a result, the plot of weekdays falls below the plot of weekends until early morning where people are most likely to sleep during weekends.

2.2 Part 2 - Mapping

2.2.1 Plotting the Map of Europe

In the initial twitter dataset, some of the country and city labels were written in various languages and characters. In addition, the coordinates are even more precise in describing the location of the tweet, so we preferred to discard the city and country columns in order to save some memory and processing time. In this section we plotted two maps. the first one is a choropleth map that indicates the number of tweets in each European country using a color scale, whereas the other indicates the specific location of each tweet in using the coordinates (scatter map). We first discarded the NaNs in the ***coordinates.coordinates*** column in ***df*** then converted the string of formated "[long,lat]" to two separate float columns of longitude and latitude that, in turn, are converted to ***coordinates*** dataframe as shown below.

```
'''
convert the coordinates column in df to a list of long,lat pairs then
    converted to a dataframe
'''
coordinates = [x.split(', ') for x in [x[1:-1] for x in df['
    coordinates.coordinates'].dropna()]]
```

```
coordinates = pd.DataFrame(coordinates, columns=['long','lat'])
coordinates.set_index(df['coordinates.coordinates'].dropna().index,
    inplace=True)
coordinates = coordinates.astype(float)
```

Using ***reverse_geocode*** library (PyPI, 2020), we can fetch the city and country from the coordinates. After getting the list of countries and cities, we add them as two separate columns to ***coordinates.***

```
reversed_coord = reverse_geocode.search(coordinates=coordinates[
    coordinates.columns[::-1]])
coordinates['country'] = [rc['country'] for rc in reversed_coord]
coordinates['city'] = [rc['city'] for rc in reversed_coord]
```

We downloaded a **geojson** file from (Carto, 2021) to be used along with **plotly** in order to plot the aforementioned maps. The geojson file will be used to map country names to their country code and continent. That will allow us to filter only for European countries and use the country code with **plotly.express** to plot both maps. The following code shows how both the country codes and continent data were extracted the geojson file and mapped to ***coordinates*** dataframe.

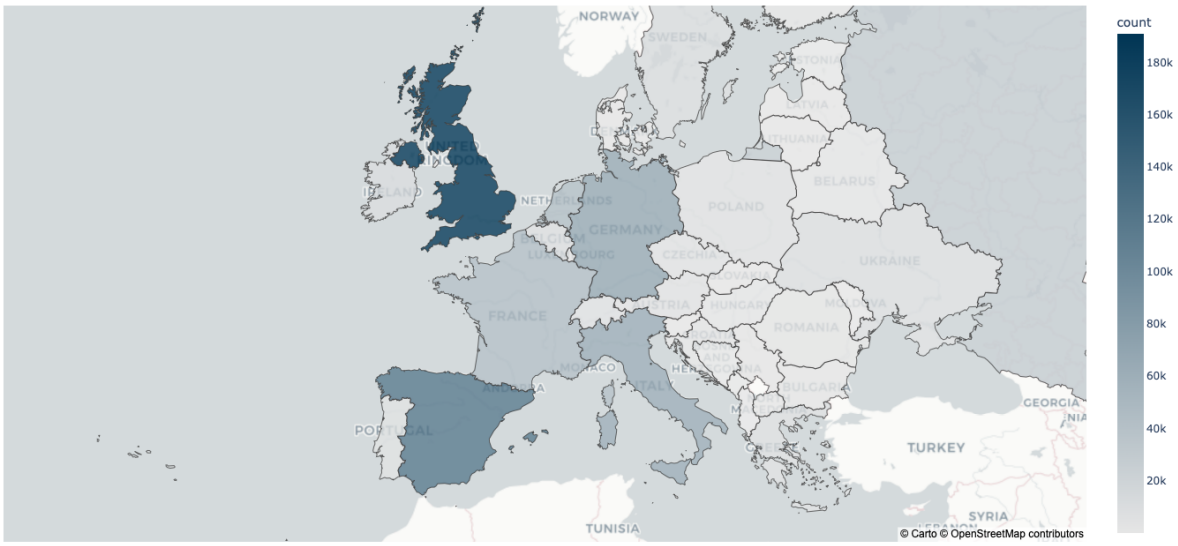
```
european_countries = json.load(open('world_countries_geojson.geojson')
    )

country_name_code_map = {}
for feature in european_countries['features']:
    feature['id'] = feature['properties']['iso_a3']
    country_name_code_map[feature['properties']['name_long']] =
    feature['id']
```

Some of the countries, when fetched using ***reverse_geocoding***, were named differently from the ***geojson*** file. Those countries where changed so that the mapping could be done successfully. An example of that is "Svalbard and Jan Mayen" which was changed to "Norway". The mapping of continents to the corresponding countries was done in a similar manner to the mapping of country codes. The resulting ***coordinates*** dataframe contains 6 columns: lat, long, country, country code, city, and continent, as shown in figure 4.

Figure 4: Coordinates DataFrame

created_at	long	lat	country	city	continent	country_code
2021-05-31 23:00:00+00:00	13.435000	52.481388	Germany	Berlin Treptow	Europe	DEU
2021-05-31 23:00:01+00:00	8.987778	44.497500	Italy	Piccarello	Europe	ITA
2021-05-31 23:00:01+00:00	30.524574	50.450861	Ukraine	Kyiv	Europe	UKR
2021-05-31 23:00:01+00:00	24.031343	49.841809	Ukraine	Lviv	Europe	UKR
2021-05-31 23:00:02+00:00	1.893817	41.545639	Spain	Olesa de Montserrat	Europe	ESP

Figure 5: Number of tweets per European country



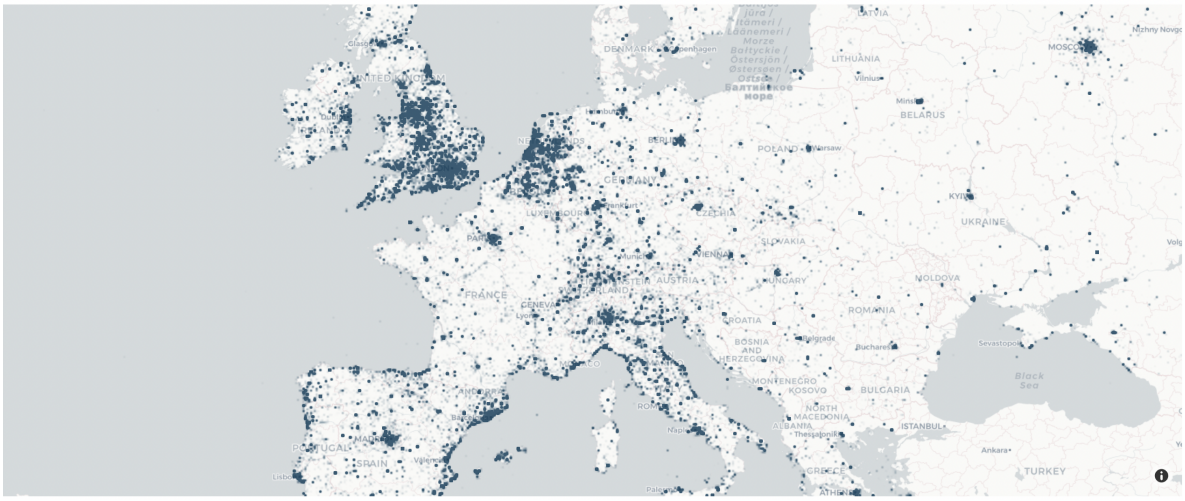
Finally, we filtered to only European countries and counted the number of tweets per country.

```
tweets_euro_country = coordinates[coordinates['continent']=='Europe']
count_tweets_per_euro_country = pd.DataFrame(coordinates[coordinates['
    continent']=='Europe']['country_code'].value_counts())
```

The resulting map plot is show in figure 5

Taking into account only geo-tagged tweets, from the choropleth map we notice that the highest number of tweets comes from the UK, Spain, Germany, Italy, and France. Although the UK has less population than Germany, it has considerably a larger number of tweets. While the restrictions started to reduce in the UK and from the 7th of June 2021, international cruises were allowed to dock in Spain (Molly, 2021), in France, on June 9th, there was still some quarantine restrictions and people were not allowed to go out after 11PM (Vie publique.fr, 2021) and was loosened a bit until June 30, where traveling resumed as well, this may explain the lower activity of on Twitter during that period. By plotting a scatter map, we get a better view on the locations of tweets.

Figure 6: Scatter Map of tweets in European countries



By plotting a more detailed map using the coordinates (figure 6), we notice that most tweets come from larger cities in those countries; for instance, for the UK, we have a high density in London, Manchester, Liverpool, and Birminghham. While in the other countries, most tweets come from the capital, for example in France we notice that most tweets come from Paris, for Spain we see a large number of tweets from Madrid, and Germany from Berlin. We can clearly see that there is a large number that is coming from Spanish coasts. Finally, we notice that tweets in the UK are more spread, that is probably due to the absence of lockdown which allowed people to travel freely across the UK.

2.3 Part 3 - Users

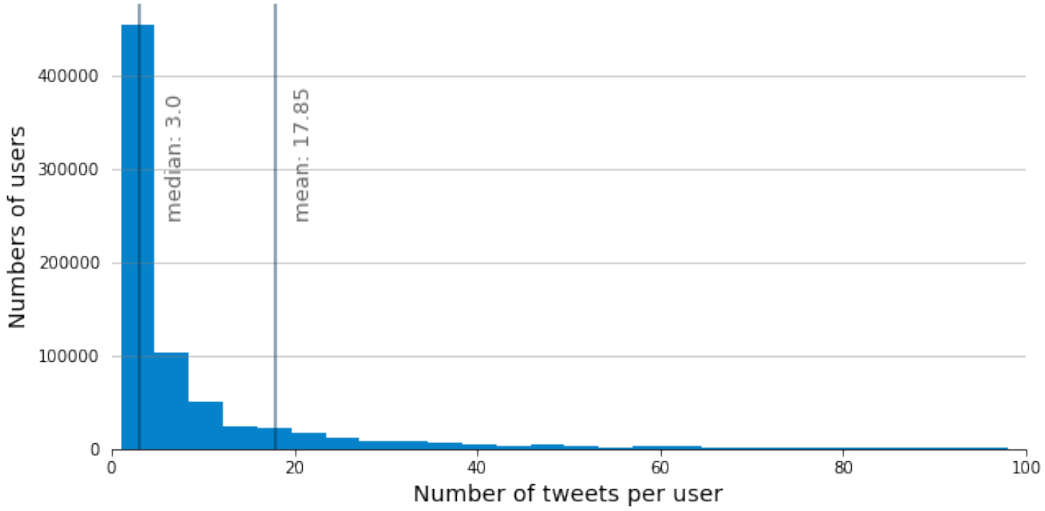
2.3.1 Number of users vs Number of tweets

By checking the unique user IDs and unique user screen names, we found out that there are 776,366 unique IDs whereas there are 743,939 screen names, more than one user ID is attributed to a screen name. We want to use an attribute that describes a user uniquely, so we counted the number of tweets based on the user IDs as follows:

```
1 tweets_per_user = df['user.id_str'].value_counts()
```

We limited the x-axis to 100 for the clarity of the plot. From figure 7, we notice that the plot is right skewed. The median is 3 posts per user and the mean is almost 18. We can clearly see that almost 450k users have one to five tweets and 150k have between 5 to 12 tweets. Which means, almost 77.28% of all the users we have in our data have tweeted one to 12 tweets in June.

Figure 7: Number of users vs number of tweets



2.3.2 Top users

We can get the top 5 user by numbers of tweet using the following line of code:

```
index_list_top_users = [x for x in tweets_per_user.iloc[:5].index]
```

The resulting user IDs are the following:

1. 1384110594.0
2. 1.2116064333990953e+18
3. 1.382496899744301e+18
4. 9.741889409734861e+17
5. 161262801.0

By matching those user IDs with the screen names we get the following list: **'HoraCatalana', 'RadioTeddyMusic', 'casimiroperezc1', 'Maria70221974'**, **'AyferGl02976871'**. Most of the top users are definitely automated accounts. The main reason is that most of those accounts are posting a high number of tweets in such a short timeframe. If we take the first top user as an example "HoraCatalana", by looking at the numbers of tweets posted each hour by this account we find that it is 16 most of the time each hour of the day (for 24 hours) without stopping. Which confirms that this is an automated account (all the account is doing is posting the current time in Spanish).

2.3.3 Most mentioned users

User mentions are stored as list in the column ***entities.user_mentions*** in ***df*** dataframe. For some reason, some of the user mentions objects we not converted to a list of IDs and stayed as JSON objects. By filtering empty lists we were left with 7,631,379. By removing the rows that were not converted to lists of IDs (1,724,386 - it was taking too much time to convert to lists - 433 rows in 12 hours), we are left with 5,906,993 rows. We first convert the user mentions column into a 1D array using the following function.

```
def mention_per_user_df(df,mentions_column, users_column):
    i = 0
    mentions_list = []
    user_list = []
    for c in df[mentions_column]:
        count += 1
        for m in ast.literal_eval(c):
            mentions_list.append(m)
            user_list.append(df.iloc[i][users_column])
        i += 1

    user_mention_dict = {'user': user_list, 'mention': mentions_list}

    return pd.DataFrame(user_mention_dict)
```

The above code results in a DataFrame that contains in each row, a user id and one single user mention () which is shown in figure 8.

Figure 8: User and mentioned user DataFrame

	user	mention
0	1.199788e+09	1186645124
1	1.199788e+09	922162684325498880
2	1.199788e+09	516260791
3	1.346758e+18	1365364671914651648
4	1.346758e+18	327149170

By counting how many times a specific user was mentionned, we end up having the following 5 account: **10228272**, **3131144855**, **68034431**, **335141638**, and **98149527** being mentioned **13787**, **10216**, **10007**, **9465**, and **9226** respectively. By looking the frequency of mentions for those users we can tell that those accounts are of famous figures or famous organizations. From virtualfollow.com, we can confirm that those accounts are of ***Youtube***, ***Boris Johnson***, ***Recep Tayyip Erdoğan*** (the president of Turkey), ***BTS*** (famous Korean band), and ***Sedat Peker*** (a turkish mafia boss (Wikipedia, 2021). Another way to confirm that those accounts are of famous figures/organizations, we saw in how many tweets per user those accounts were mentioned and we noticed that it is mainly 1 to 4 times by so many users.

2.3.4 Mentions in other Countries

For this part, have only selected tweets that had mentions. We assigned also created a DataFrame mapping each users to their country (that is limited to only tweets that were geo-tagged).

```
target_countries_users = coordinates[coordinates['country'].isin(['
    Italy','Germany','France', 'United Kingdom', 'Turkey'])][['country'
    ,'user']]
target_countries_users = target_countries_users.drop_duplicates()
```

The resulting DataFrame looked as follows:

Figure 9: Users location DataFrame

	country	user
0	Germany	1.092190e+09
4	Italy	2.858864e+09
11	United Kingdom	4.643842e+09
19	United Kingdom	1.029522e+08
22	United Kingdom	2.612048e+09
...	...	...
13854721	United Kingdom	8.144977e+07
13854728	France	2.731477e+09
13855154	United Kingdom	1.352436e+09
13855980	France	3.060088e+09
13856069	Germany	3.320024e+09

In order to count how many users were mentioned from other countries ('Italy','Germany','France', 'United Kingdom', 'Turkey'), we created a function ***mentions_coutry_matrix()*** that takes as argument the list of the target countries, the DataFrame containing the pairs of users and their respective countries (***target_countries_users***), and the DataFrame that contains pairs of users and the user mentioned (***user_per_mention_df***). The function returns a dataframe that is of shape 5x5 containing the frequency of a user in the country in the row mentioning a user from a country in the column. The function will check if we have country information in ***target_counutries_users*** for every user mentioned in ***user_per_mention***. If we have the information, it will check as well the information responsible for the mention and increment the corresponding well in the matrix. The resulting matrix is shown in figure 10

Figure 10: Matrix for users mentioning users in other countries

	Italy	Germany	France	United Kingdom	Turkey
Italy	1532	213	219	212	7
Germany	202	1136	205	207	12
France	198	210	523	193	14
United Kingdom	204	188	217	2294	23
Turkey	8	6	4	10	567

We can see that most users mention other users from the same country. If we take for example the accounts that were mentioned the most, Boris Johnson for instance, it is more likely that he will be mentioned by a UK user rather than users from other countries. Some of the users that mentioned accounts from a different country may be mentioning accounts of famous figures or accounts of famous places they visited.

2.4 Part 4 - Events

For this part, Germany, United Kingdom, and Italy were chosen.

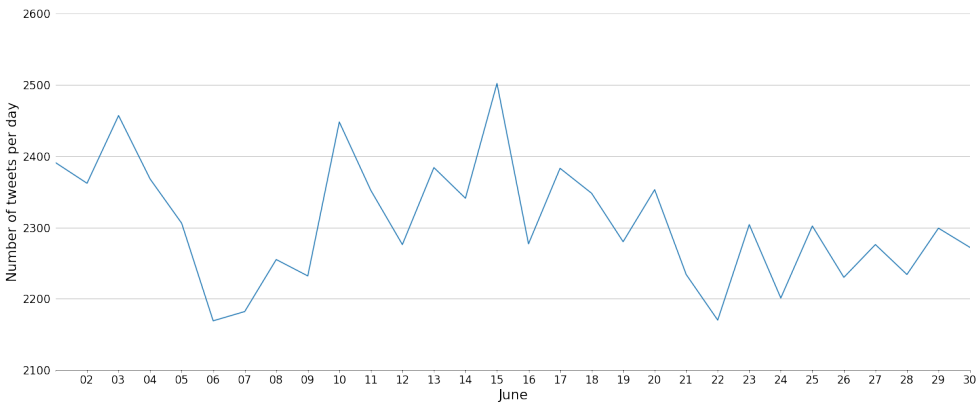
2.4.1 Germany

Germany has seen one of the largest amounts of tweets during June 2021 after Spain and the United Kingdom reaching 69,216 (geo-taged only) tweets. In order to plot the daily number of tweets in Germany, ***tweets_euro_country*** was used (derived from ***coordinates*** dataframe by filtering the continent only to Europe) and sampled by day as shown in the following code snippet.

```
daily_tweets_germany = tweets_euro_country[tweets_euro_country['
    country'] == 'Germany'].resample('D').count()['city']
```

Using ***plot_daily()***, we plot the number of tweets per day in Germany (figure 11).

Figure 11: Number of tweets per day in Germany



We can see from the plot that there is a significant increase on the 15th of June in the numbers of tweets, that may mainly due to the EURO2020 football match where Germany played against France and lost. We can also see from some of the tweets that they contain #EURO2020 referring to the game and some of the people went out with their friends and watched the game in bars and restaurants.

- Pomeczowe memy zacne! #EURO2020 #POLSVK
- @Cristiano nie pije @CocaCola to i sa rezultaty, tylko woda nie mylić z wóda @LaczyNasPilka @pzpn_pl @BoniekZibi #EURO2020
- #EURO2020 (@ Salon zur wilden Renate in Berlin) https://t.co/suNz8AIag4

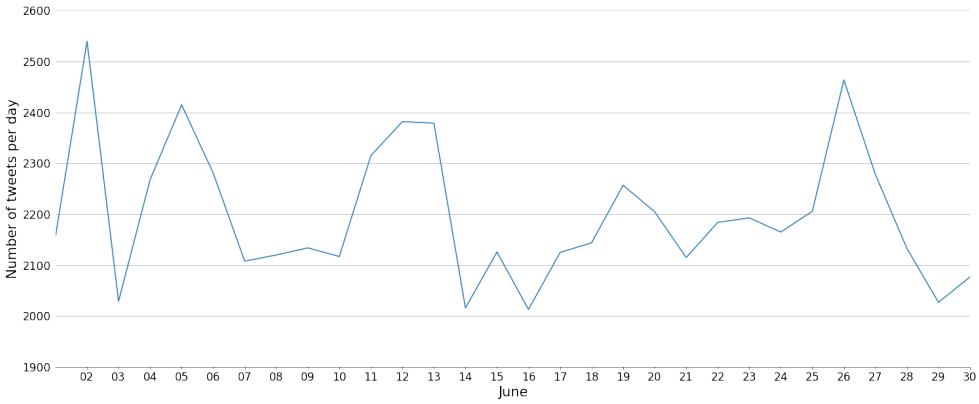
DW as well wrote an article about the game in the following link:https://p.dw.com/p/3uzhk In order to have a better visualization of which words were mostly used during that day, we generated a word-cloud using wordcloud library. From the word-cloud in figure 12, we notice that the most recurring word is EURO2020.

Figure 12: Wordcloud for German Tweets - June 15th 2021

2.4.2 Italy

Similarly to Germany, we can get the tweets for Italy using the same line of code only testing with the country name Italy. The resulting plot is the following (figure 13)

Figure 13: Number of tweets per day in Italy



Although Italy has seen a large number of tweets in some days of June, the 2nd of June has seen a large amount of tweets with a severe decrease in the following day. That is due to the fact that the 2nd of June is a public Holiday called Republic Day. Most of people that day were on a day-off to celebrate. However, the following day was a still a weekday, people had to go back to work/study which explains the decrease in the number of tweets. If the next day was a weekend, we might expect a slightly higher number. Some examples of tweets that day are:

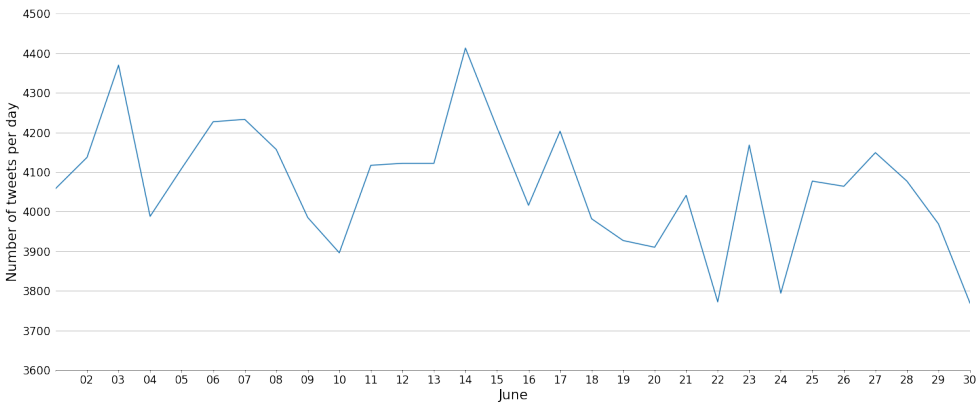
- #2giugno #festadellarepubblica #stato #italia #freccetricolori #wlitalia... https://t.co/sWZ2s0dWEg
- W l'italia e W la Repubblica giorno e #buonagiornata #2giugno20221 @Ospedaletti https://t.co/jMLt3LOu0o
- Buona Festa della Repubblica @Centro Storico Nardò https://t.co/FB0FaEJlYH
- Acaba de publicar una foto en Porta camollia Siena https://t.co/clovaBNqf4
- Buona festa della repubblica italiana a tutti #2giugno #festadellarepubblicaitaliana #75anniversario @Settimo T... https://t.co/9pp5nUTo5E

These tweets and posts (following the links) clearly confirms the statement above.

2.4.3 Spain

We can get the tweets from spain in the same way as we did with Italy and Germany. Figure 14 shows the number of tweets per day for Spain during the month of June.

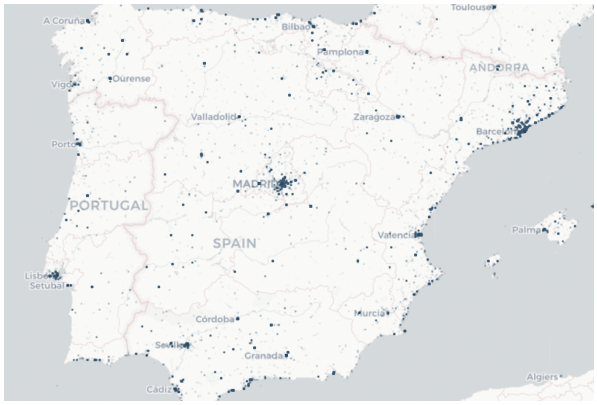
Figure 14: Number of tweets per day in Spain



Although the 14th of June was a weekday, the number of tweets in during that day reached the peak (compared to the other days in the month of June). On June 7th, the Spanish government has announced the ease of COVID-19 restrictions where they allowed EU citizens to travel to Spain using the EU-approved rapid test and tourists from non-EU countries using vaccination certificates (Screteria de Estado de Turismo, 2021). Starting from the 7th of June 2021, international cruises we allowed to dock in Spain which increased the number of tourists especially in coastal areas. The following figure shows the location and density of tweets on the Spanish map before 12th of June (we cannot expect more tourists during the same day as the restrictions ease as people require more procedure to get to Spain, in addition, comparing a week worth of tweets to the remaining month is not a fair comparison) and after. We can clearly see that there is a higher density in both the Capital Madrid and coastal areas including Barcelona after the 12th of June, as more cruises started to dock.

Figure 15: Location of tweets in Spain before and after restrictions ease

(a) Density of tweets before 12 June



(b) Density of tweets after 12 June

2.5 Reflection

1. The way Twitter works makes it easier to browse for data especially using hashtags. Twitter API makes it even easier as well for researchers to get the data needed according to specific criteria such as the geographical location. Although the data in twitter is more reliable than other social media, Twitter user demographics varies highly from region/country to another. For example, in North Africa, most people use Instagram, Facebook, and other social media mainly for entertainment purposes and very few people use Twitter. In that region, most people who use twitter are very likely to have at least a Bachelor's degree and the usage purpose may be for sports news, politics, or economics. A statista report – "Social media usage in the United Kingdom (UK)" (Statista, 2021b), reported that in the February 2018, 56% of twitter users were male and that did not vary a lot in the previous years while another report showed that in Spain only 37.2% of users are females (Statista, 2021c), this illustrates that Twitter data is skewed and differs from region to another not only in terms of users' gender, but also online activity, age, habits, and other personal attributes which varies from country to another. Having this in mind, we need to be careful about the assumptions we make in our analysis and think carefully if that category actually represents the entire population which we are trying to study.

2. Twitter data may be biased in such a way it will massively affect the study of the human behaviour. As an example, we have seen previously that a huge number of data points were originating from different non-human sources that were programmed to automatically generate a tweet, scientists must identify those accounts so it will not affect the study. Another issue could be fan accounts for example; fan accounts tend to imitate a specific celebrity/place and replicate information that exists in the original account or even spread fake news. Public figures accounts could be biased as well, that is due to the high likelihood of someone from PR trying to post on behalf of that person. Brid-Aine Parnell published in her article on Forbes that Juergen Pfeffer from Carnegie Mellon University reported that most social media companies filter using data streaming algorithms that may change in a second which will induce other biases that scientists are not aware of (Parnell, 2014). They also reported that a large number of fake accounts was deleted by twitter in 2013. Most fake accounts were created to influence public opinion and the behaviour of users online for commercial or political purposes, scientists should know how to apply the appropriate filter to eliminate bots and fake accounts based on activity patterns and analyse public's response to those accounts.

3. While researchers were using questionnaires, surveys, and interviews to gather data, Twitter nowadays offers a better way for them to get data at larger scale with only few clicks and words to write. However, legal restrictions and ethical concerns must be taken into accounts whenever any research is done on social media data. There is a very interesting Social Media Ethics Framework developed by the University of Aberdeen that addresses several aspects of this subject. This framework takes into account three major aspects: the legal aspect, privacy and risk, and re-use and publication (Dr. Leanne Townsen, Prof. Claire Wallace,

2016). Although researchers read terms and conditions and users may agree on them (mostly without reading), one should be aware that these can change regularly. Different regulations may also apply on different users on Twitter, for example the interaction of several users from different countries, take an EU based user commenting on a non-EU based user. Here, the GDPR applies on the EU based user, the researcher does not need to comply with GDPR when analysing the non-EU user's data. In addition, it is not because the data is public that it would be ethical to use it, it depends on which context and the aim of the research, the risk of harm induced on the users among the participants pool should be assessed in terms of anonymity and exposure or having children accounts included in the research (to name a few). Finally, researchers should think carefully whether they can publish their findings in book, articles, scientific journals and the way they report it risking to breach users' privacy or expose them.

4. Taking out fake accounts and bots, users on Twitter express themselves genuinely as opposed to other social media. Twitter can be used to analyse users' sentiments regarding lockdown policies and COVID-19 restrictions based on their posts and see if people are coping with that. Hashtags, also, can be released by governments and healthcare organizations, in order to quickly identify infected individuals. When combined with geographical data of the tweet, we will get a more accurate information of where the user was infected (or where/when the user tested positive). This will allow healthcare organizations to act faster and make the right decisions at the right time, however this does not mean that all Twitter users will comply with this and will not consent to share their location. In addition, statista has reported that twitter is the 15th most used social media worldwide in October 2021 (ranked by number of active users) (Statista, 2021a) with a major difference compared to Facebook. This indicates that there is a large portion of user categories that is not on Twitter and may not be represented by those who are on Twitter. This may encourage the usage of multiple social media platforms to study the effectiveness of lockdown policies.

References

Carto (2021). world countries geojson.

Dr. Leanne Townsen, Prof. Claire Wallace (2016). Social Media Research: A Guide to Ethics. Technical report.

Molly (2021). Situation in Spain with Coronavirus 2021.

Parnell, B.-A. (2014). Scientists Warn About Bias In The Facebook And Twitter Data Used In Millions Of Studies.

PyPI (2020). reverse-geocode.

Screteria de Estado de Turismo (2021). NEW HEALTH REQUIREMENTS FOR ENTRY INTO SPAIN AS FROM 7 JUNE. Technical report.

Statista (2021a). Global social networks ranked by number of users 2021.

Statista (2021b). Social media usage in the United Kingdom (UK). Technical report.

Statista (2021c). Twitter user share in Spain in 2021, by gender.

Twitter (2021). Tweet object.

Vie publique.fr (2021). Covid-19 : un déconfinement en quatre étapes.

Wikipedia (2021). Sedat Peker.