ECMM443: Introduction to Data Science

Coursework

Dataset:

The dataset used consists of tweets collected from the Twitter API during the period June 1st to June 30th 2022. These tweets were collected by applying a geographical filter to return only tweets in Europe. The bounding box used is specified by (longitude, latitude) coordinates. The lower-left corner is at (-24.5, 34.8) and the upper-right is at (69.1, 81.9). We assume this box defines "Europe". No keyword or other thematic filters were applied, so the dataset should contain all tweets that Twitter can identify as originating from the specified region, irrespective of their topic/content. The data is available for download as a number of compressed files each covering 1 hour of data. The whole dataset contains millions of tweets.

Part 1: Basic Stats

1. The total number of unique tweets is 15,033,548. Overall, there are 15,040,709 entries in the files. Out of them there are 5,666 duplicate tweets occupying 6,839 places. 322 records do not have tweet ID (appearing as "None" or NULL depending on the method of data extraction)

   Observation: There is a discrepancy between created_at and timestamp_ms. Twitter Documentation mentions that created_at is the time when the tweet was created according to local time. We observe that the minimum created_at value is 2022-05-31 23:00:00, i.e. 1 hour before the start of June 2022. We see that throughout the data, this shift of 1 hour remains between created_at and timestamp_ms (which holds the UTC time). This can be attributed to Daylight Savings time as between March and August, clocks are set back by 1 hour in the UK (*and much of the western hemisphere*), thus falling behind UTC.

   Going forward in this report, we will be using timestamp_ms as our source time.

   timestamp_ms is read as ts, which is then converted to datetime as follows:
   `[datetime.datetime.fromtimestamp(int(dt)/1000) for dt in ts]`

2. Time-series of tweets by day:

   Here is some basic code and time series graph (shown in Fig 1):

```
# Group at day level
twt_daily = twt_df.groupby('ts_day').agg(num_tweets = ('tweet_id','nunique'))

# Plot
fig , ax = plt.subplots(figsize = (20 , 5))
ax.plot(twt_daily['num_tweets'] , color = 'orange');
ax.vlines(
    twt_daily.index ,
    ymin = min(twt_daily['num_tweets']) ,
    ymax = max(twt_daily['num_tweets']) ,
    color = 'blue' ,
    alpha = 0.1 ,
    ls = '--'
);
ax.set_xticks(twt_daily.index);
ax.set_xticklabels(labels = twt_daily.index, rotation = 90);

plt.xlim(min(twt_daily.index) , max(twt_daily.index));
```

Here is the result:

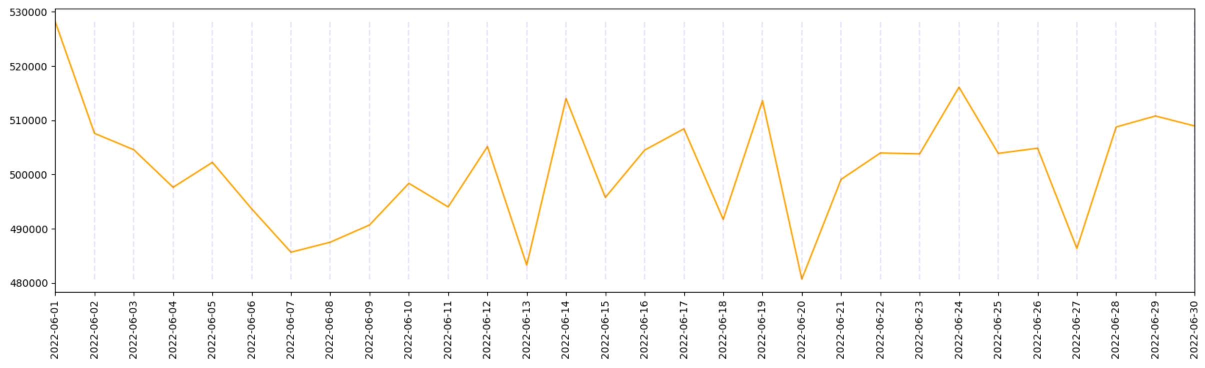


Fig 1: Time Series showing volume of tweets across the month of June'22

Comments:
Tweets seem to be falling slightly on Mondays/ Tuesdays, which sort of makes sense as people resume work after the weekend.

3. Box and Whiskers Plot

Here is some basic code:

```
# Preparing Boxplot
twt_df_grp = twt_df.groupby(['ts_day' , 'weekday']).agg(num_tweets = ('tweet_id' ,
'nunique')).reset_index()
twt_df_grp_wkday = twt_df_grp[twt_df_grp['weekday']==1]
t1 = twt_df_grp_wkday['num_tweets'].tolist()

twt_df_grp_wkend = twt_df_grp[twt_df_grp['weekday']==0]
t2 = twt_df_grp_wkend['num_tweets'].tolist()

week_tweet_dict = {
    'weekday':t1 ,
    'weekend':t2
    }

# Plot
fig , ax = plt.subplots()
ax.boxplot(week_tweet_dict.values());
ax.set_xticklabels(week_tweet_dict.keys());
ax.axhline(np.mean(t1) , ls = '--' , c = 'red' , label = 'weekday mean');
ax.axhline(np.mean(t2) , ls = '--' , c = 'olive' , label = 'weekend mean');
plt.legend(loc = 'best');
plt.title('Tweet Volume Difference');
```

The rather small difference is shown in Fig 2 below.

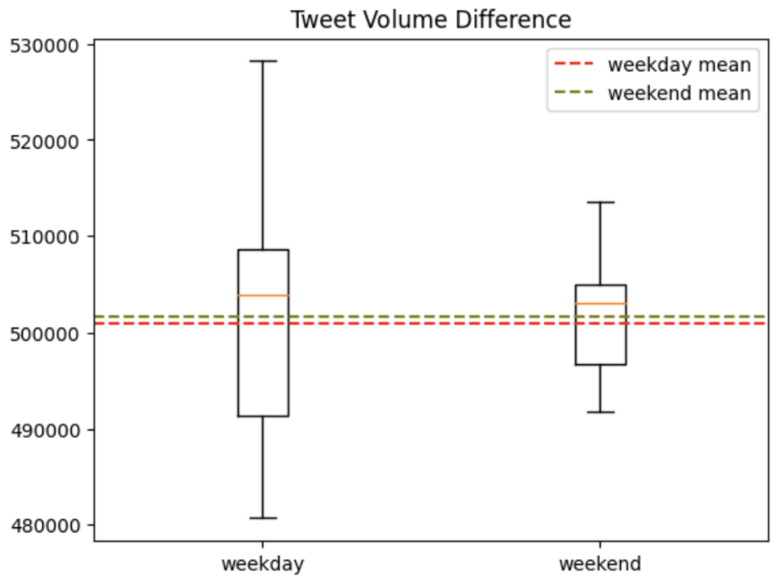


Fig 2: Boxplot showcasing the difference between volume of weekday tweets vs weekend tweets

From preliminary observation, it does not look like there is much difference. Let us test this more robustly. We generate for each user, the mean difference between the average number of their weekday and weekend tweets. In order to perform a t-test, this should approximate a normal.

```
# Standardize
hypo_test_df['avg_weekend_tweets'] = (hypo_test_df['avg_weekend_tweets'] -
np.mean(hypo_test_df['avg_weekend_tweets']))/np.std(hypo_test_df['avg_weekend_tweets'])
hypo_test_df['avg_weekday_tweets'] = (hypo_test_df['avg_weekday_tweets'] -
np.mean(hypo_test_df['avg_weekday_tweets']))/np.std(hypo_test_df['avg_weekday_tweets'])
# We will test if this difference is normal(-ish); that will allow us to conduct t-tests on
the data
hypo_test_df['mean_diff'] = hypo_test_df['avg_weekend_tweets'] -
hypo_test_df['avg_weekday_tweets']
(osm, osr), (slope, intercept, r) = scipy.stats.probplot(hypo_test_df['mean_diff'] , plot =
plt);
print(f'Correlation coefficient is {r}')
```

```
Correlation coefficient is 0.6261623577774496
```

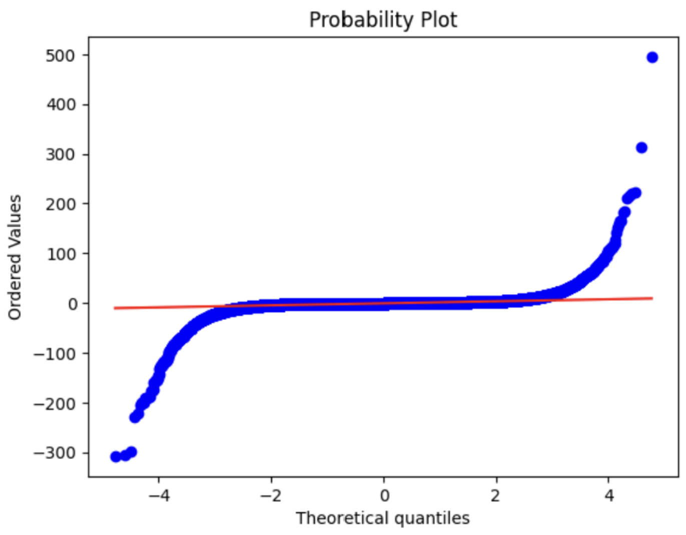


Fig 3: Quantile-quantile plot of our data against a standard normal distribution

This does not look normal. Does this mean we cannot robustly perform a t-test on this? Let us recall that with large N, t-test is not that sensitive to non-normal data, especially when the variances are equal (almost), as shown below.

```
np.var(hypo_test_df['avg_weekend_tweets']) = 0.9999999999999999
np.var(hypo_test_df['avg_week_tweets']) = 1.0000000000000002
```

Moreover, due to such large sample size (over 700,000), we can assume t-test to be valid. <REFERENCE?>

```
%%time
# Simulate CLT
X = np.array(hypo_test_df['mean_diff'])
sample_size = 100
N = 10000
collect = []
for _ in range(N):
    sample = [np.random.choice(X) for i in range(100)]
    sample_mean = np.mean(sample)
    collect.append(sample_mean)

fig , ax = plt.subplots()
freq , bins , patches = ax.hist(collect , bins = 100);
plt.title('Resampled Histogram');
ax.set_xlabel('Mean Diff');
ax.set_ylabel('Occurrences');
```

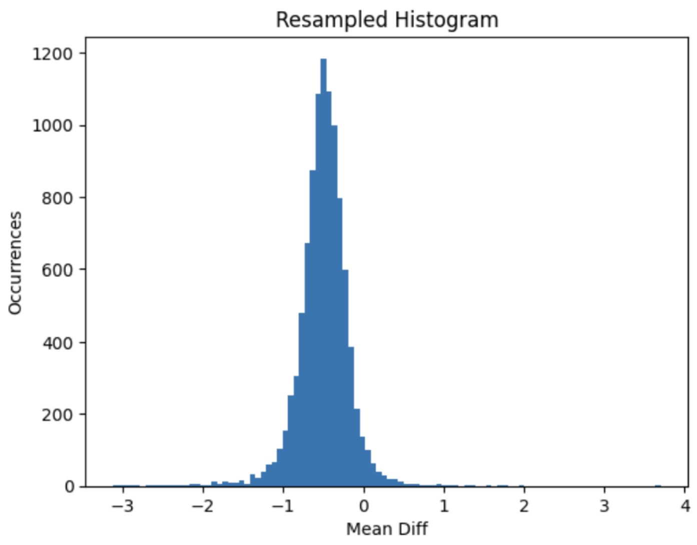


Fig 4: Resampled Histogram of difference between average weekend tweets and average weekday tweets

We will perform a paired t-test.

```
%%time
# Standardize the data
avg_weekday_twt = (np.array(hypo_test_df['avg_weekday_tweets']) -
np.mean(np.array(hypo_test_df['avg_weekday_tweets'])))/np.std(np.array(hypo_test_df['avg_wee
kday_tweets']))
avg_weekdend_twt = (np.array(hypo_test_df['avg_weekend_tweets']) -
np.mean(np.array(hypo_test_df['avg_weekend_tweets'])))/np.std(np.array(hypo_test_df['avg_wee
kend_tweets']))

# Set up the Hypothesis test
# H0: There is no difference between the means; H1: there is difference between means
# Paired T Test - see if u want to implement a two sided t test or a one sample t test?
scipy.stats.ttest_rel(avg_weekday_twt , avg_weekdend_twt, alternative = 'two-sided')
```

Here is the result:

```
Ttest_relResult(statistic=2.654590994409255e-14, pvalue=0.9999999999999788)
```

Such a high p-value says:
1. There is definitely ***not*** enough evidence to ***reject*** the Null Hypothesis that the two means are equal.
2. The two datasets are most likely highly correlated; a correlation test would likely yield a very high coefficient.

```
scipy.stats.pearsonr(avg_weekday_twt , avg_weekdend_twt)
PearsonRResult(statistic=0.7548022023741843, pvalue=0.0)
```

Although not extremely high, the two samples (weekend tweets vs weekday tweets) are correlated to a fair degree (75%). To some extent, this may seem obvious, given we are comparing weekend tweets vs weekday tweets made by the same users

4. Hour and day values are extracted via:

```
twt_df['hour'] = twt_df['ts'].dt.hour
twt_df['day'] = twt_df['ts'].dt.day_of_week
ax.plot(twt_df_hr_only , ls = '--' , label = 'No. of Tweets');
plt.xticks(twt_df_hr_only.index);
ax.grid();
ax.legend(loc = 'best');
ax.set_xlabel('Hours');
ax.set_ylabel('Tweets');
ax.get_yaxis().set_major_formatter(matplotlib.ticker.FuncFormatter(lambda x, p:
format(int(x), ',')))
```

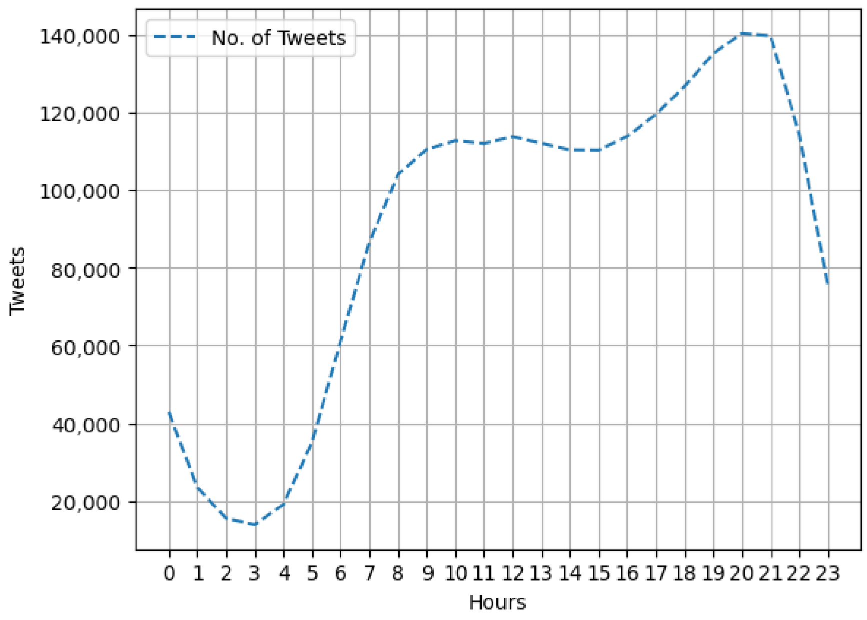


Fig 5: Time-series of tweets by hour, averaged over all days of the week

Comment on Pattern

Starting from 12 midnight, we see a decline in the number of tweets as more and more people go to bed. This number starts picking up after 2 AM - 3AM as people start getting up (recall that there are a few time-zones captured in this data). The number climbs sharply till about 8 AM, which is likely when most users clock in to their daily work/ school, (or start commuting to work/ school etc.). Tweets stay plateaued at this range till around 3 PM, from where it starts hiking upwards again - people are likely leaving work/ school at this time. Number continues to increase until roughly 10 PM at night, which is when we see a sharp decline, likely due to users retiring for the day/ going to bed.

As a bonus, let's look at the pattern over the days separately instead of averaging them.

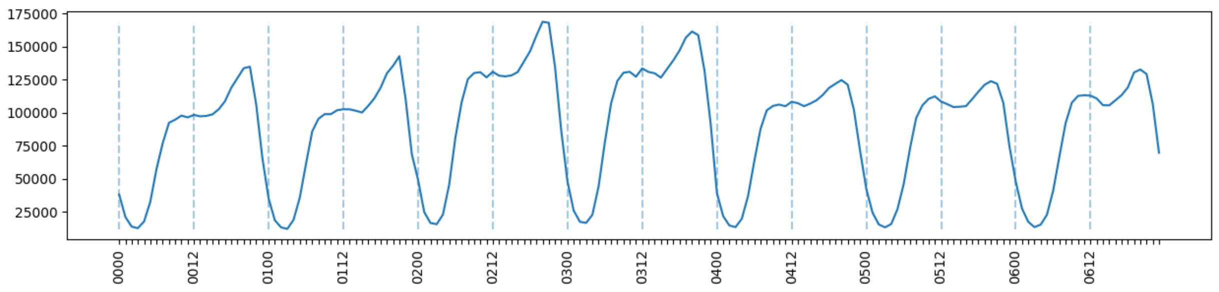


Fig 6: Time-series of tweet volume averaged at hours of a week. All 7 days from Monday to Sunday are represented here.

It seems like although the hourly pattern is mostly preserved in the number of tweets, Wednesdays and Thursdays see on average the highest volume of tweets.

Part 2: Users

1. Number of users against the number of tweets they have made.

```
twt_df_usr = twt_df.groupby('user').agg(num_twt = ('tweet_id' , 'nunique')).reset_index()
fig , ax = plt.subplots(figsize = (10 , 4));
freq , bins , patches = ax.hist(twt_df_usr.num_twt , bins = 250);
ax.set_ylabel('Users');
ax.set_xlabel('Number of tweets');
plt.title('Histogram for Number of Tweets made by  Users');
```

We appear to have a highly right skewed distribution. Likely due to some accounts tweeting at an inordinate rate/ volume.

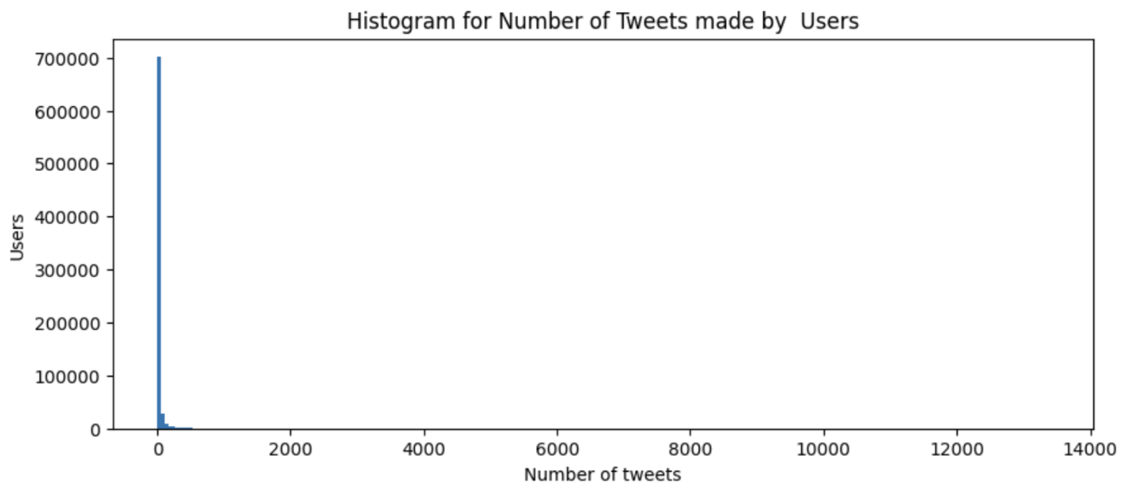


Fig 7: Histogram representing number of tweets made by users

It helps us understand that the majority of users (over 95%) have <100 tweets over the month of June. To quantify this, let us look at quantiles of the distribution.

```
print(f'{90}th quantile -> {np.quantile(twt_df_usr.num_twt , q = 0.9)}')
print(f'{95}th quantile -> {np.quantile(twt_df_usr.num_twt , q = 0.95)}')
print(f'{99}th quantile -> {np.quantile(twt_df_usr.num_twt , q = 0.99)}')
print(f'{100}th quantile -> {np.quantile(twt_df_usr.num_twt , q = 1.00)}')
```

Results:

```
90th quantile -> 37.0
95th quantile -> 76.0
99th quantile -> 289.0
100th quantile -> 13376.0
```

However, there are **some** users who have tweeted over 10,000 times, with the top user (by volume of tweets) having tweeted 13,376 times!
Seeing that there are 30 days in our dataset, this user must have tweeted 13376/(30*24) = 18.6 times per hour, every hour of every day of the month.

Note: Perhaps, such high-volume users are bots. We will return to this notion later!

Log-transforming the data may give us some more tractable insights.

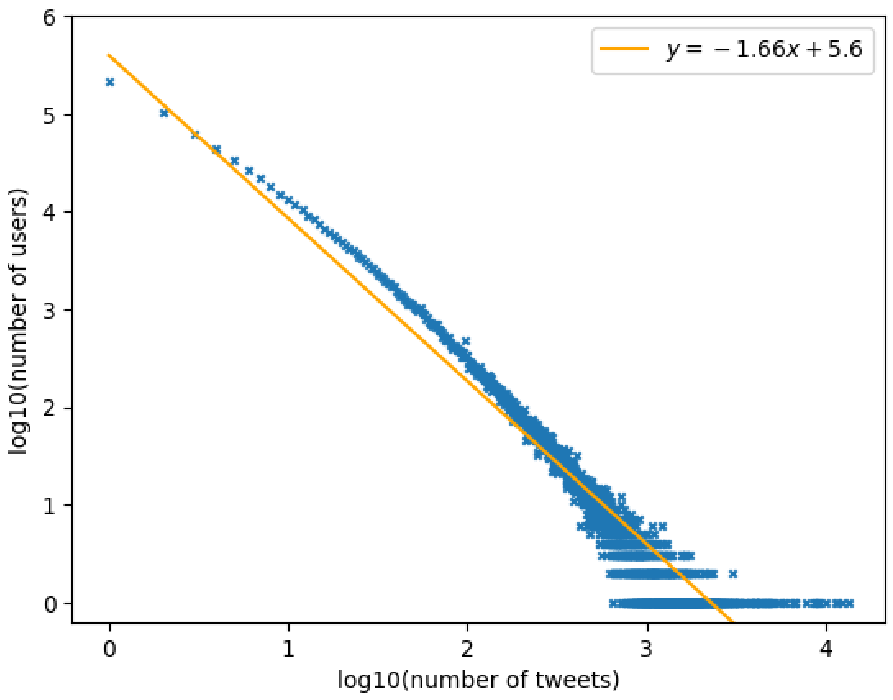


Fig 8: Log10 transforms of both axes – number of tweets, number of users shows a vaguely linear trend

As we can see, the log-transformed data approximately follows a linear pattern. The yellow line is a visual (*not very rigourous*) approximation of the equation the curve follows.

In the labelled equation, `Y = -1.66X + 5.6`, such that `Y = log10(y)` and `X = log10(x)`, where `y = number of users` and `x = number of tweets.`

Therefore, $y = 10^{5.6} x^{-1.666}$. A *power-law* relationship exists between users and tweets. If users doubled their current value, the number of tweets would become $2^{-1.666} = 0.314$ times their current value.

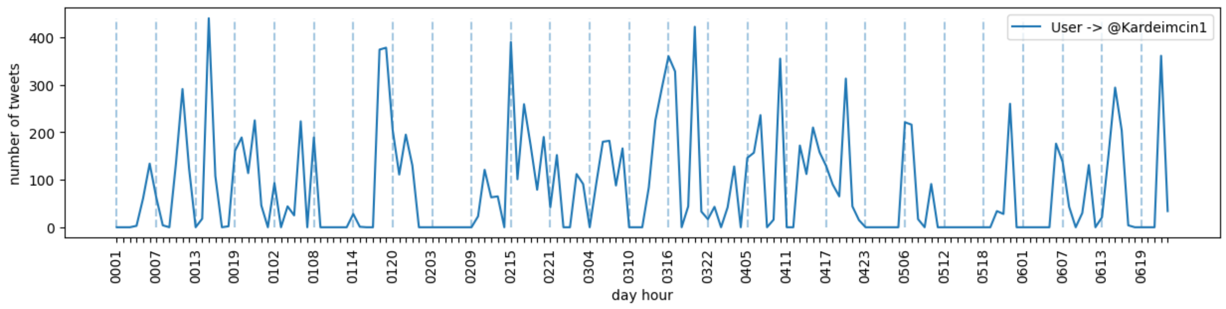
2. Top 5 Users by Total Tweets

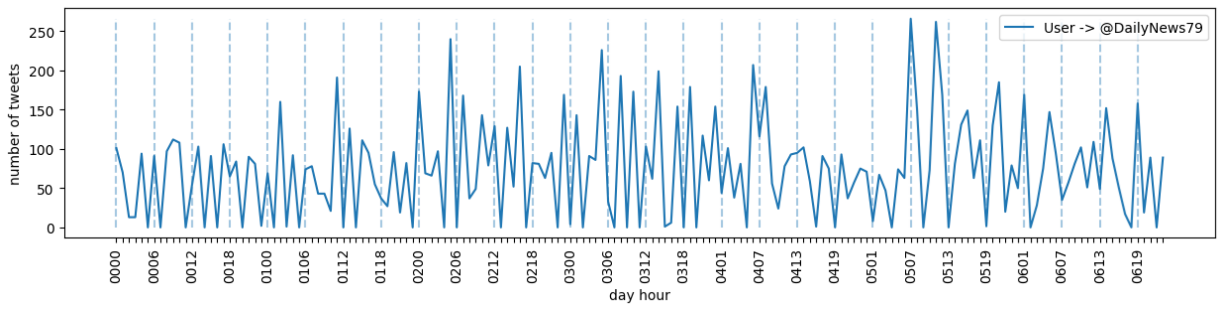
Grouping the number of tweets by user and selecting the top 5, we have the following:

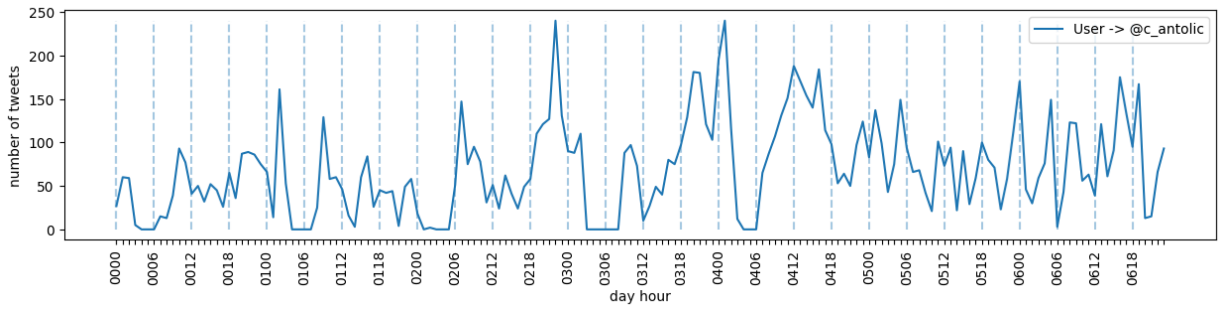
1. Büşra (@Kardeimcin1) – A Turkish account with politically inclined posts

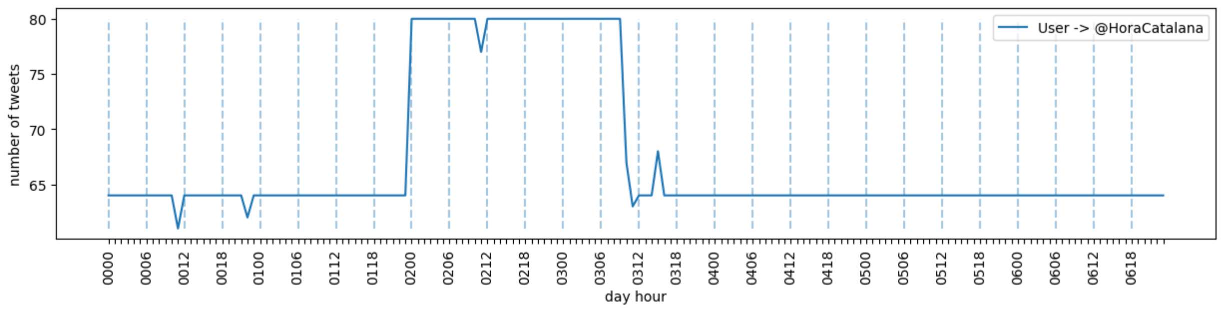
2. Daily News Italy (@DailyNews79) – Twitter handle for Italian news service, which has been suspended at the time of writing this report.
3. Christian Antolic (@c_antolic) – German man tweeting German words – is there some pattern? At the time of writing this report, this account has been suspended.
4. L'hora catalana (@HoraCatalana) – Posts the time in Catalan time system. Appears to be automated, based on frequency.
5. minijob-anzeigen.de (@minijobanzeigen) – German job advert website. The description of the account clearly mentions that it is automated, hence we can say it is a bot (*we have no reason not to trust the description*).

We can explore this further by looking at the patterns of their associated timestamps, i.e. the times when they have been posting content.









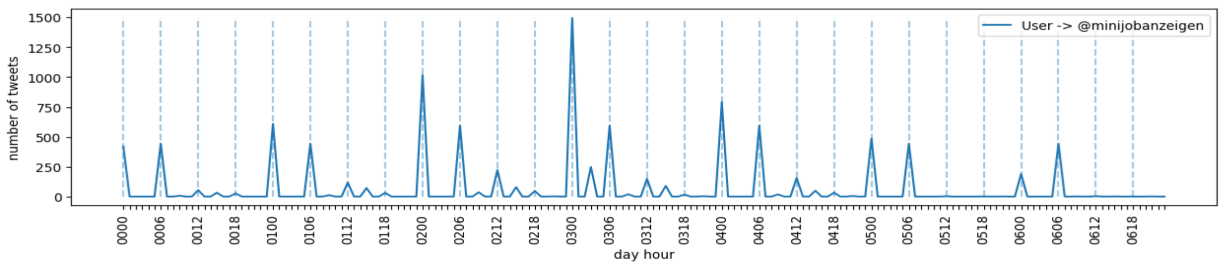


Fig 9: Time-series patterns of tweet volume from high-volume tweeters. All 7 days from Monday to Sunday are captured here at an hour level

Comments:
Studying the top 3 graphs, we cannot find any pattern, which we would expect if it were a human that was making the tweets (*perhaps, something close to Fig 6*). For the fourth figure, we see a very uniform volume of tweets (one can argue that it is far too consistent to be made by a human). The fact that it is the Twitter handle of a news channel can be thought of as indicative of it being a bot. The last figure exhibits some patterns, but it is explicitly stated to be an automated account.

3. Users receiving the most mentions.

   For this, the user_mentions tag inside entities node was used. In case the tweet is truncated, we gathered all user mentions from within the extended_tweet root node

   Some code to extract mentions:

```
%%time
# Make DF
twt_mentions_df = pd.DataFrame(
    list(zip(mentions , tweets)) ,
    columns = ['mentions','tweets']
)
# Get rid of null values
twt_mentions_df = twt_mentions_df[twt_mentions_df['tweets']!='None']
# Get rid of duplicates
twt_mentions_df.drop_duplicates(inplace=True)

# Changing from list representation to an actual list
twt_mentions_df['mentions'] = twt_mentions_df['mentions'].apply(lambda x : json.loads(x))
twt_mentions_df = twt_mentions_df.explode('mentions')
```

Here are the top 5 accounts:

USER ID	***USER***
10228272	YouTube (Company)
68034431	Recep Tayyip Erdogan (President of Turkey)
3131144855	Boris Johnson (Former Prime Minister of the UK)
44196397	Elon Musk (Billionaire)
118787224	Indian politician (MLA* from Siricilla, Telengana**)

Table 1: Highest-volume tweeters

As we would have imagined, these accounts are those belonging to companies, politicians and public figure(s). This would appear to be in line with our expectations of most-mentioned users.

4. We have selected four countries France, Germany, Spain and Portugal to compute patterns of how users from these countries mention (and are mentioned by) each other.

First, we unpivot the *user->mentioned* dataframe.

```
# Changing from list representation to an actual list
twt_mentions_df['mentions'] = twt_mentions_df['mentions'].apply(lambda x : json.loads(x))
twt_mentions_df = twt_mentions_df.explode('mentions')
twt_mentions_df.dropna(inplace = True)
print(f'the length of the df is {len(twt_mentions_df)}')
the length of the df is 16,446,394
```

We can see it has (obviously) increased the number of records. We must now preprocess this data. From the tweet data, we can tie the user to their associated country by checking which country they tweet the most from.

Note: The assumption here is that the country a user tweets most from is their country of residence/ origin for the purposes of this analysis.

Since this gives us a link between user and country, it will also be helpful to locate the associated country of the mentioned user. We need to do this as the raw data does not hold any relation between mentioned user and their country. This also makes it such that any user who has not tweeted during the data capture period will not have an associated country – this is a caveat.

```
%%time
# Trying to get one primary country per user - the country column is only attached to the
user
# Check if primary country adds any value ?
twt_country = twt_mentions_df[['user' ,'country']].drop_duplicates()

twt_mentions_df.drop('country' , axis = 1 , inplace=True)

prim_country = twt_country.groupby(['user','country']).agg(num_occr =
('user','count')).reset_index()

prim_country['max_occr_country'] = prim_country.groupby('user')['num_occr'].rank(method =
"first" , ascending = False)
prim_country = prim_country[prim_country['num_occr'] == prim_country['max_occr_country']]
```

However, we find that the *prim_country* dataframe does not add any value to the analysis as no user is associated to more than one country.

```
prim_country.sort_values('num_occr' , ascending=False).head(2)
```

Here is the result!

user	country	num_occr	max_occr_country
1000001577812463616	Portugal	1	1.0
3317160592	Ireland	1	1.0

Table 2: No user is associated to more than 1 country

Some further preprocessing is required as various countries are called by various names in various languages. For example, Germany is called *Allemagne* in French and *Deutschland* in German! As such, we translate the names of countries in the data to their English counterparts.

```
# Graph the number of cross mentions
fig , axes = plt.subplots(1 , len(country_list) , figsize = (27.5 , 5))

for _ , (ax , user) in enumerate(zip(axes , country_list)):
    df = x_mentions_df_3.loc[user]
    ax.bar(df.index , df.occr);
    ax.set_xlabel(f'Mentions from {user}')
    ax.set_ylabel('No. of Mentions');

# Countries seem to mostly be mentioning themselves
```

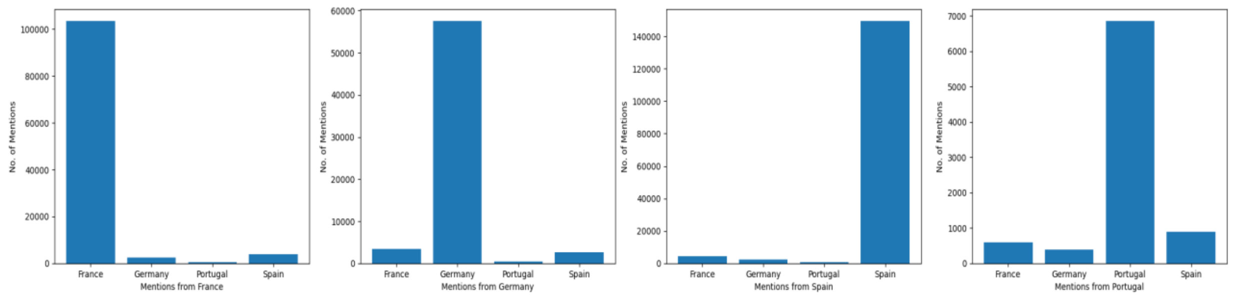


Fig 10: Cross-country mentions

It can be seen that countries mostly mention themselves!

Data is tabulated below in Table 3:

mentioned_country	France	Germany	Portugal	Spain
user_country				
France	103423	2537	550	3784
Germany	3424	57534	493	2642
Portugal	596	384	6849	893
Spain	4311	2194	867	149477

Table 3.1: Tabulation User of Country to Mentioned Country. Read as element i,j = x → Country i mentioned Country j x times.

mentioned_country	France	Germany	Portugal	Spain
user_country				
France	94%	2%	0%	3%
Germany	5%	90%	1%	4%
Portugal	7%	4%	79%	10%
Spain	3%	1%	1%	95%

Table 3.2: Percentage Tabulation of User Country to Mentioned Country

Part 3: Mapping

1. A Map of Europe that displays the use of Twitter across the continent, using only the GPS-tagged tweets.

```
# Use tweets and coordinates to create a mapping dataframe
mapping_df = pd.DataFrame(
    list(zip(tweets , coord)) , columns = ['tweet_id' , 'coordinates']
)
# Removing unnecessary rows
mapping_df = mapping_df[(mapping_df['tweet_id']!="None")&
(mapping_df['coordinates']!="None")].drop_duplicates()
```

```
len(mapping_df)
```

```
701961
```

Having extracted coordinates (longitude and latitude) from the twitter activity data, we overlay a map of Europe [1]against the said activity.

```
%%time

data = {}
pops = {}
shape_file = os.path.join(os.getcwd() , 'world_map\\TM_WORLD_BORDERS_SIMPL-0.3.shp')

with fiona.open(shape_file) as layer:
    for feature in layer:
        country_code = feature['properties']['NAME']
        data[country_code] = shape(feature['geometry'])

patches = []
colours = []

for c in data:
    try:
        for p in data[c].geoms:
            lons , lats = np.array(p.exterior.coords.xy)
            patches.append(PolygonPatch(list(zip(lons , lats)) , closed = True))
    except AttributeError:
        lons , lats = np.array(data[c].exterior.coords.xy)
        patches.append(PolygonPatch(list(zip(lons , lats)) , closed = True))
# Create custom colour map
cmap = LinearSegmentedColormap.from_list('custom blue', [(0,'#ffff00'),(1,'#8B0000')],
N=256)

# Make subplot
fig , ax = plt.subplots(figsize = (15 ,8))

# Hexbin for tweets
ax1 = ax.hexbin(x_coord , y_coord , gridsize = 300 , bins = 'log' , cmap = cmap);
cbar = fig.colorbar(ax1);
cbar.set_label('Number of Tweets');

# Overlay map
p = PatchCollection(patches , edgecolor = 'k' , lw = 1.5 , facecolor = 'white' , alpha =
0.4);
ax.add_collection(p);
```

[1] Shapefile for Europe and neighbouring countries was taken from: https://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units/nuts

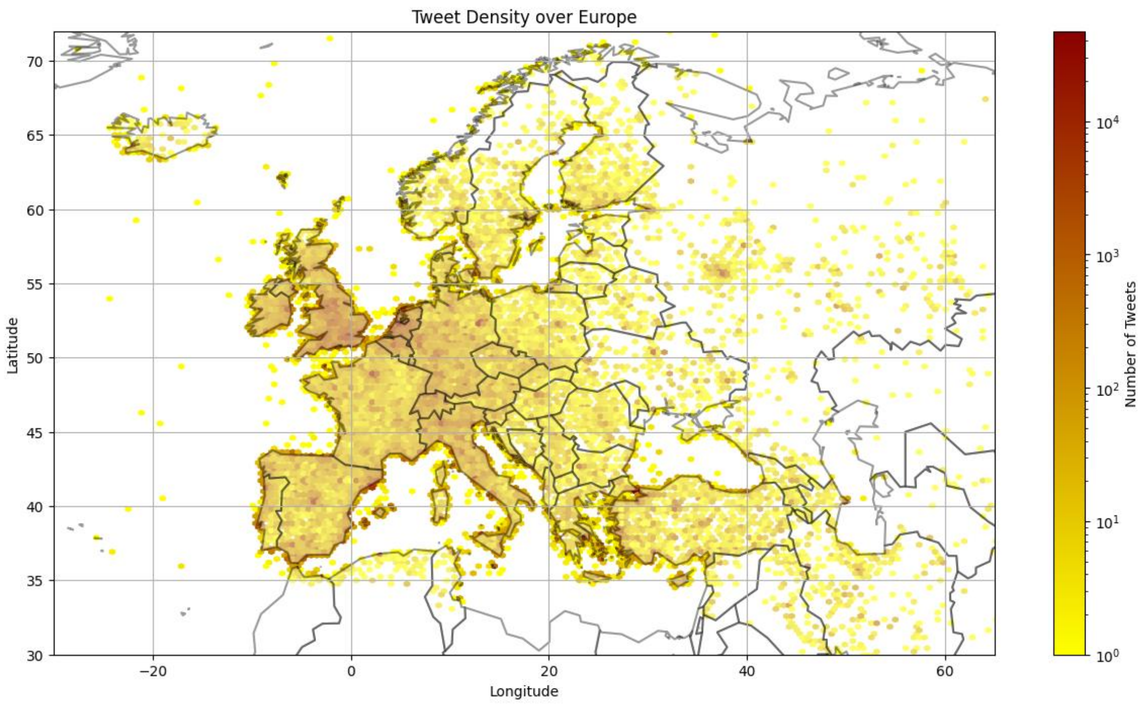


Fig 11: Twitter activity overlaid on top of geographical boundaries. More red signifies higher amount of tweets

2. **Patterns observed:**

1. We can make out the hotspots are centered around places of high population. The further north we go, the lesser tweets show up - indicating (most likely) a drop in population. As we see, the map is bright red over the capitals, especially
Dublin, Paris, Madrid, Berlin, Amsterdam, Copenhagen, Warsaw, Istanbul etc.
2. Large concentrations of tweets also come from big cities which aren't capitals - examples, Barcelona, Malaga
3. We notice that there is a lot more activity along the coastlines, but this is likely due to the presence of big port cities like Nice and Barcelona
4. Weird - Tweets from the middle of oceans and seas - Mediterranean Sea, Atlantic Ocean, Black Sea, Caspian Sea - why? Warrants further investigation.

tweet_id	coord	long	lat	Bot?
1.53334E+18	[-21.27, 46.22]	-21.27	46.222218	Bot
1.53351E+18	[-24.28, 53.92]	-24.28	53.921337	Bot
1.54062E+18	[-21.77, 59.35]	-21.77	59.353892	Bot

5. Maybe, we would have expected a lot more activity in and around Ukraine – but we do not observe it. Perhaps, because of internet outages, they are not being able to connect?
6. Coastal areas – are more hotspots – perhaps because of climate change discussions?
7. Internet availability – good indicator of high tweet volume
8. Small islands of Ibiza and Malta are hotspots!

3. The other tweets have a ‘place’ tag inside which is the ‘bounding box’ tag. We will plot the CDF of the diagonals.

Some preprocessing →

```
length after removal of null tweets = 15,033,548
length after removal of null bounding boxes = 15,026,765, i.e. ~ 0.05% did not have proper bounding boxes
length after removal of non-null exact co-ord = 14,331,587, i.e. ~ 4.62% had exact coordinates
```

In order to get the diagonals, we only need two opposite pairs of points in the bounding box – we have chosen the first and the third.

```
# Construct diagonals
bounding_box_df['diagonal'] = np.sqrt((bounding_box_df['bb1_long'] -
bounding_box_df['bb2_long'])**2 + (bounding_box_df['bb1_lat'] -
bounding_box_df['bb2_lat'])**2)
```

Plot the CDF (Fig 12). For ease of representation, we have chosen a log scale for the y-axis on the right.

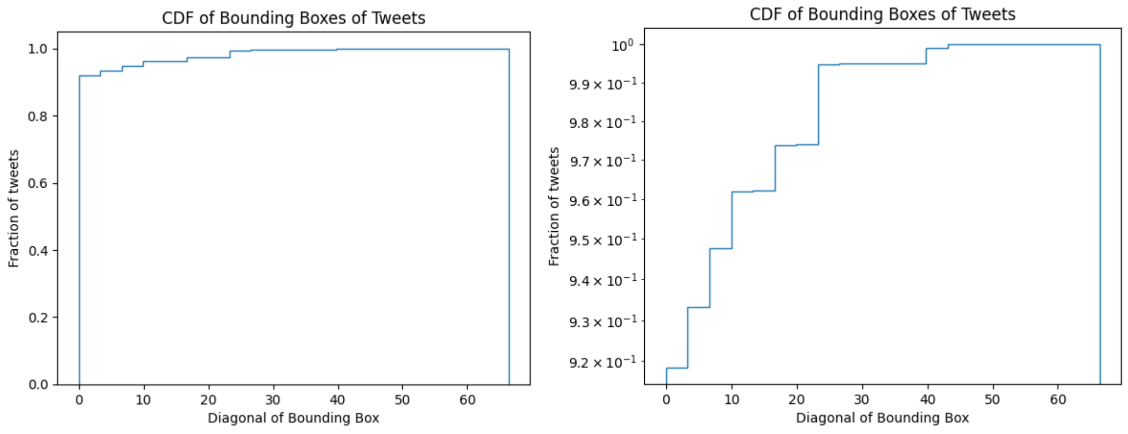


Fig 12.1: CDF of bounding box diagonals represented on i) equal axis and ii) log-transformed y-axis

Comments:

Tweets with HUGE bounding boxes seem to be originating firstly in Greenland & secondly, around Norway and Finland

We can postulate a few reasons for this. From a Euclidean geometrical perspective, it may seem odd that the bounding boxes are so big! But, remember, the closer you move to the poles, the longitudes bunch up together! Hence, they appear larger - it is the same reason why Greenland is disproportionately bigger on flat maps.

Or is the bounding box just really big?

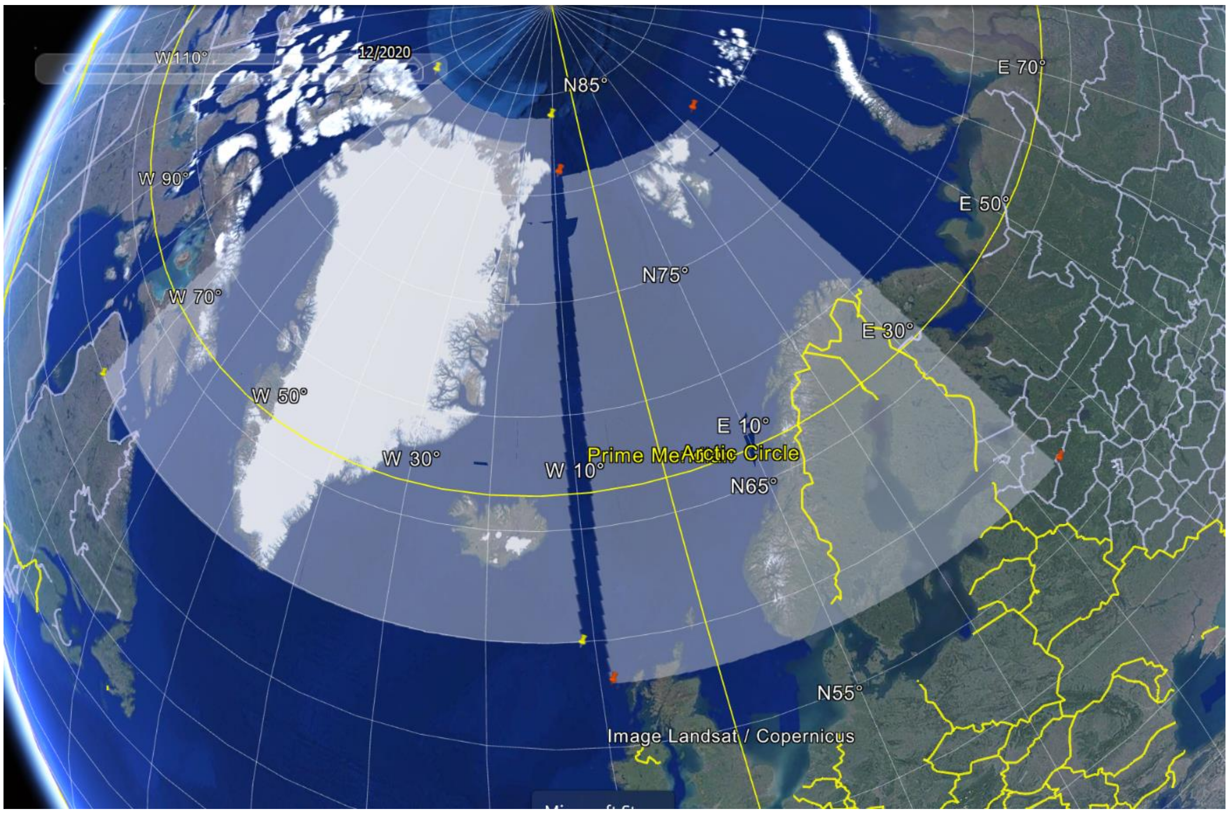


Fig 12.3: Two of the largest bounding boxes in the data

Maybe there's something more interesting at play here – All of the tweets with the largest size of bounding box have **the exact same bounding box** (pale white patch over Greenland, as shown in Fig 12.3). Second largest is that over Scandinavia (as shown in Fig 12.3). Perhaps there are only a few select cell towers in and around the Arctic, where the location data is pinged from. Without further data and exploration however, this is all hyperbolae.

4. We use another additional spatial dataset to compare against the Twitter activity. For this, we look at the greenery coverage[2]* in the city of London - roughly the coordinates are between (51.3 , -0.5) to (51.6, 0.3) in Fig 13. Roadmap of London is overlaid for clarity.

   Note: *OSGB36toWGS84* from the `bng_latlon` library was used to convert Northing-Easting to latitude-longitude

```
# Make subplot
fig , ax = plt.subplots(figsize = (10,6))
p1 = PatchCollection(patches_greenery , edgecolor = 'green' , lw = 0.01 , facecolor
= 'green');
ax.add_collection(p1);

# Scatter for tweets
ax1 = ax.hexbin(x_coord , y_coord , gridsize = 300 , bins = 'log' , cmap = "PuRd");
# ax.scatter(x_coord, y_coord,s=0.2,c='red')
cbar = fig.colorbar(ax1);
cbar.set_label('Number of Tweets');
```

[2] Shape-file for London greenery taken from - https://geospatialwandering.wordpress.com/2015/05/22/open-spaces-shapefile-for-london/comment-page-1/

```
p2 = PatchCollection(patches , edgecolor = 'black' , lw = 0.5 , facecolor =
'white');
ax.add_collection(p2);
```

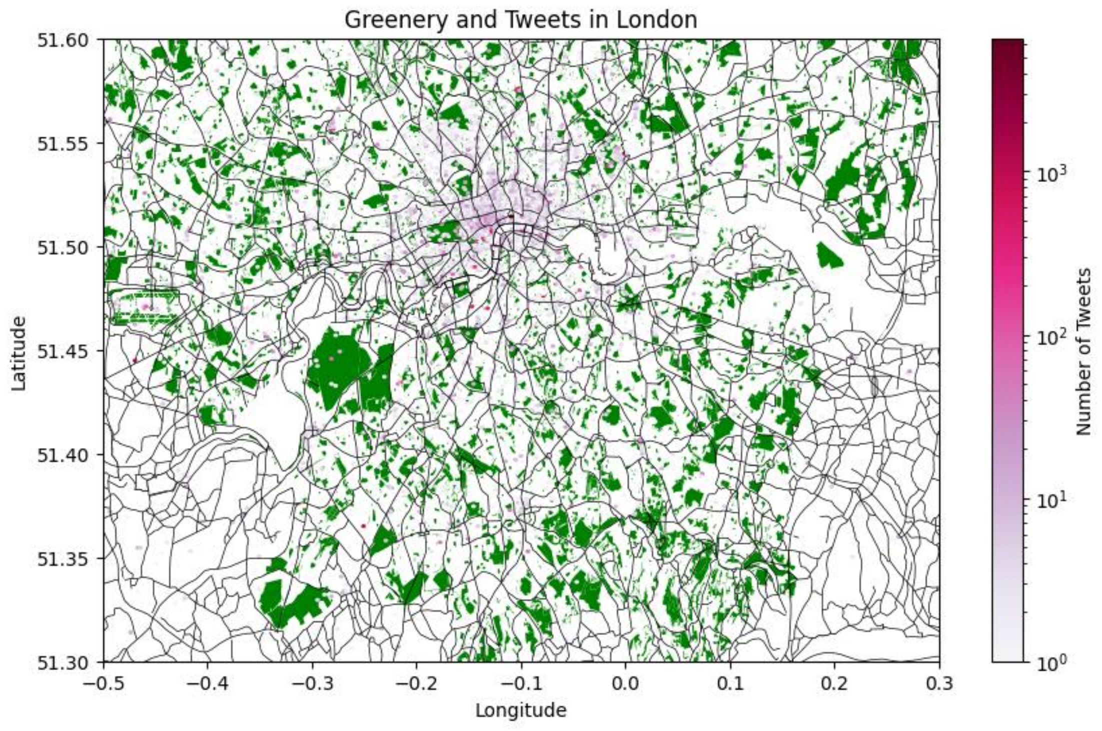


Fig 13: Though not immediately obvious, we can see that areas of greenery generally showcase lower tweet volume.

Apart from one hotspot in Richmond Park (the biggest green patch, slight left and below the centre of the image), other green parts generally show a low amount of twitter usage, if at all. This can perhaps be explained by the confounding variable - population density, to some extent. Parks are generally less crowded than spaces in the city centre.

Part 4: Events

1. We select United Kingdom, France and the Netherlands as our countries of interest. Similar to Part 2, we standardize all the names to their English representation/version.

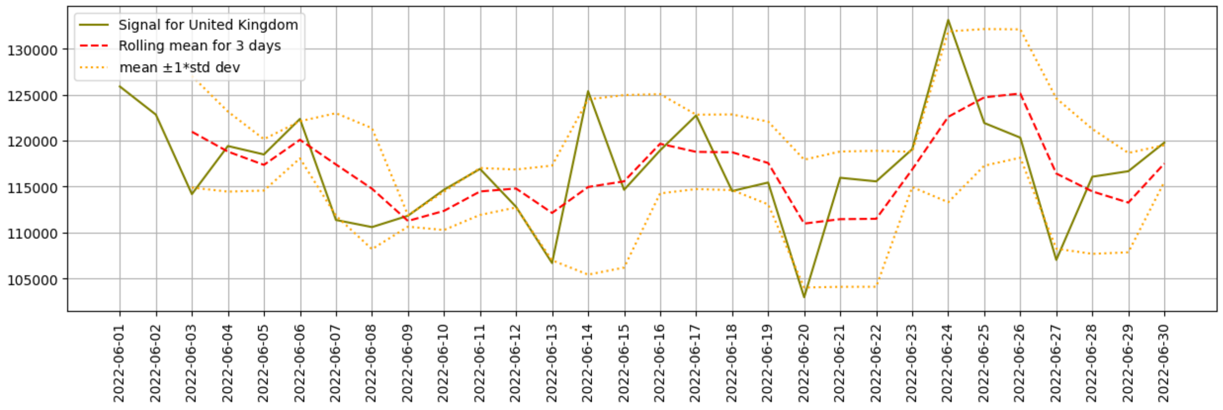
   We put the tweet data of these three countries into three separate dataframes as number of tweets and day. Then we use a technique adjacent to Bollinger bands[3] to determine days of unusual activity. Isolation forests may also be considered, but they seem like overkill for this.

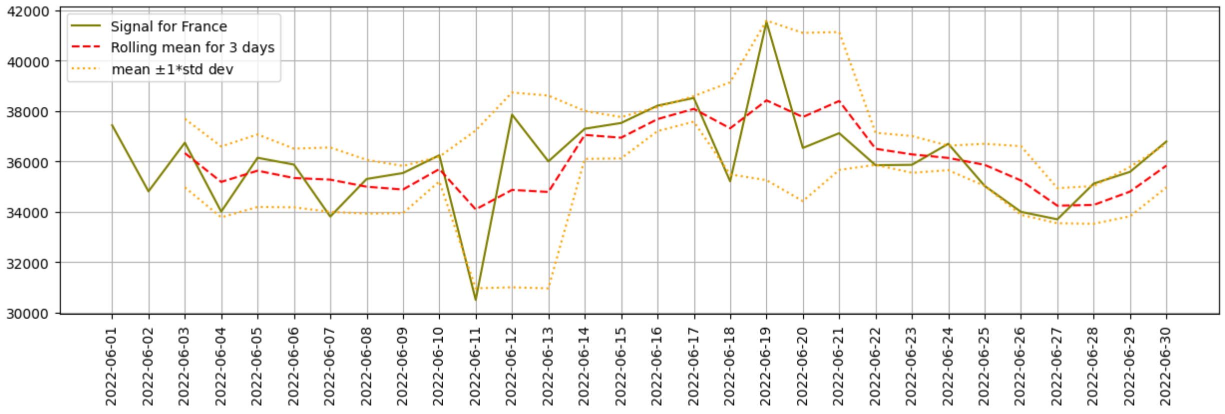
```
# Anomaly Detection using Bollinger Bands-adjacent method
fig , ax = plt.subplots(figsize = (15,4))

rate = 3
tolerance = 1
ax.plot(ts_1 , c = 'olive' , label = f'Signal for {ts_1.name}');
ax.plot(ts_1.rolling(rate).mean() , lw = 1.5 , ls = "--" , c = 'red' , label = f'Rolling
mean for {rate} days');
ax.plot(ts_1.rolling(rate).mean() + tolerance*ts_1.rolling(rate).std() , lw = 1.5 , ls = ":"
, c = 'orange' , label = f'mean $\pm${tolerance}*std dev');
```

[3] Stocks & Commodities V. 10:2 (47-51): Using Bollinger Bands by John Bollinger

```
ax.plot(ts_1.rolling(rate).mean() - tolerance*ts_1.rolling(rate).std() , lw = 1.5 , ls = ":"
, c = 'orange');
ax.legend(loc='upper left');
ax.grid();
ax.set_xticks(ts_1.index);
ax.set_xticklabels(labels = ts_1.index, rotation = 90);
```

Here we see successive outputs for UK (ts_1), France (ts_2) and the Netherlands (ts_3).





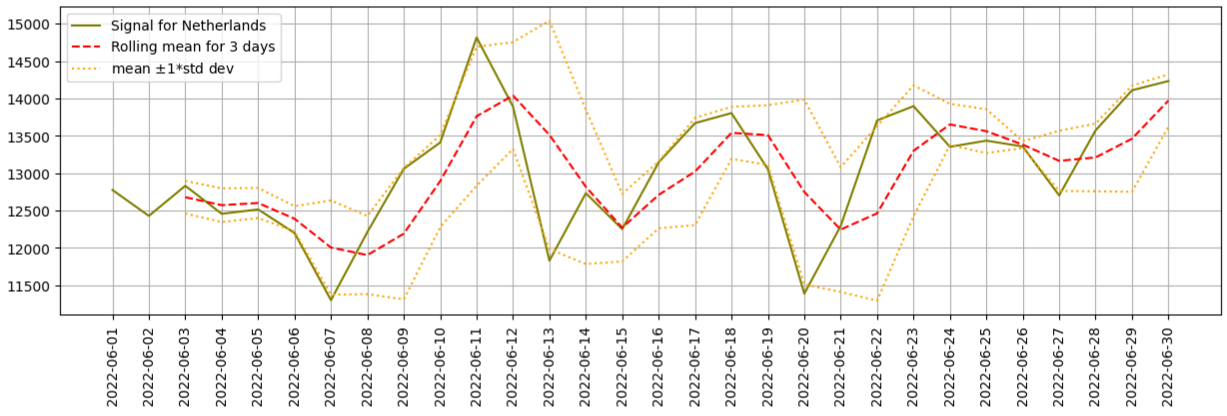


Fig 14: Time-series data on tweet volume in the UK, France and the Netherlands across June’22. We have sought out days of excessive tweeting by checking if the value crosses the threshold of a 3-day rolling mean + 1 standard deviation. As such, from the above graph, the following days stand out as days of 'anomalous' activity
- 24th June in United Kingdom
- 19th June in France
- 11th June in the Netherlands

2. We extract the text from the twitter data to build word clouds from it. We also peer into the ‘extended_tweet’ root node (where it is available) to check for more text if it has been truncated in the root node ‘text’.
   Preprocessing - some stopwords have been removed in the respective languages to cast away uninformative words.

```
from spacy.lang.fr.stop_words import STOP_WORDS as fr_stop
from spacy.lang.en.stop_words import STOP_WORDS as en_stop
from spacy.lang.nl.stop_words import STOP_WORDS as nl_stop

en_lang_stopwords_list = list(en_stop) +
['https','co','t','c','d','l','j','s','good','thank','day','year','week','today','amp','time
','great','like','people','love','new','think','no','yes','going','need','work','right','kno
w','look','got','m','thanks',"I'm",'morning','hope','want','ve','way',"don't",'best','years'
,'come','don','photo','Happy','thing','night','posted','better','X','Friday','let','lot','li
ve','life','looking','Oh','London','world','weekend','lovely']

fr_lang_stopwords_list = list(fr_stop) + ['https','co','t','c','d','l','j','s','Foto']

nl_lang_stopwords_list = list(nl_stop) +
['https','co','t','c','d','l','j','s','i','la','le','a','to','the','you','and','it','dag','v
andaag','goed','Beautiful','that','gaat','Goedemorgen']
```

Define a simple function to plot word clouds.

```
def plot_wordcloud(df , stopwords_list):
    '''
    plots the word cloud given a dataframe and a list of stopwords
    params: df -> country and date dataframe
    params: stopwords_list -> list of stopwords language specific
    returns: None
    '''
    text = ' '.join(df['text'].tolist())
    word_cloud = WordCloud(background_color = 'black' , stopwords = stopwords_list,
collocations = False).generate(text)
    # Plot
    fig , ax = plt.subplots()
    ax.imshow(word_cloud, interpolation='bilinear')
    ax.axis("off")
```

United Kingdom:

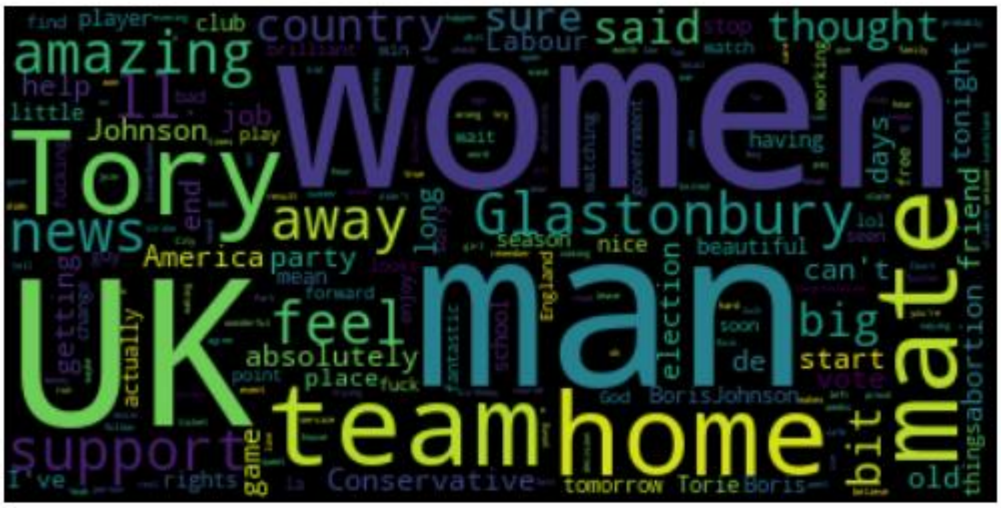


Fig 15: Word cloud showing prominent words in tweets originating in UK on 24th June, 2022

This was around the time Rowe-v-Wade was overturned in the USA. Consequently, we see a lot of solidarity and support from the UK, evinced by prominent mentions of “women”, “America” and “abortion”. Political discussions were also quite widespread on this day with prominent mentions of Boris Johnson “Conservative”, “Labour” and “Tory”.
<SPACE FOR ARTICLES>

France:

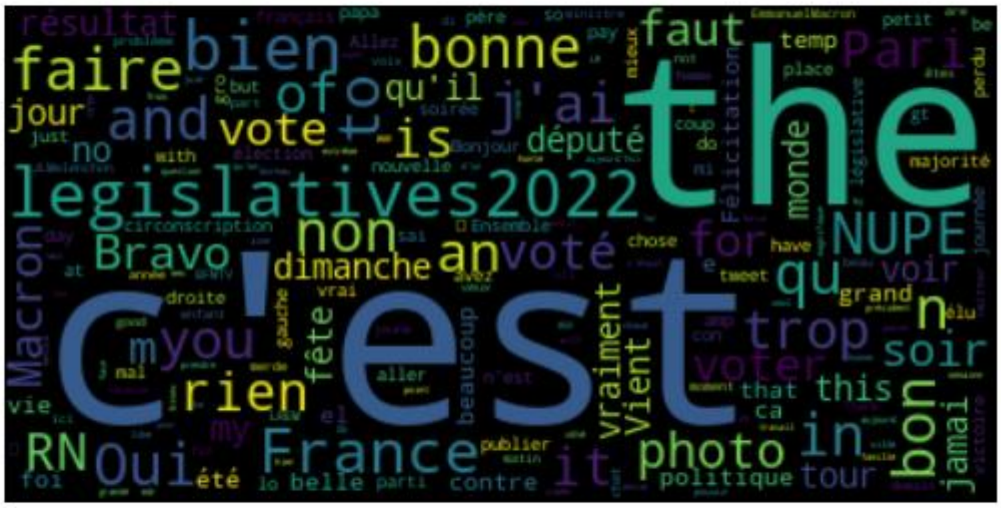


Fig 16: Word cloud showing prominent words in tweets originating in France on 19th June, 2022

Around this day in France, the Legislative Assembly 2022 was being held, marked by mentions of “legislatives2022”, “NUPE” and “Macron”.
<SPACE FOR ARTICLES>

Netherlands:

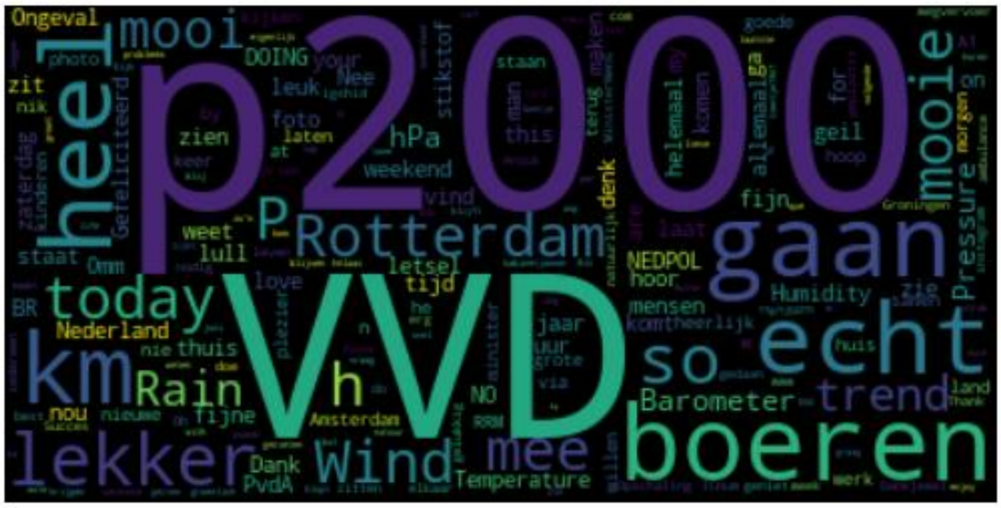


Fig 17: Word cloud showing prominent words in tweets originating in UK on 11th June, 2022

VVD is a political party in the Netherlands, which seems to be mentioned quite often. “Boeren” – the Dutch word for farmer is also seen to appear prominently. This may be in relation to Dutch farmer protests that have been going strong since <>.
However, even more interesting is the word “p2000”. Upon some investigation, it appears to be the Twitter equivalent of an emergency service call-out. <SPACE FOR ARTICLE> It thus stands to reason that this would be the mostly widely tweeted phrase.

Let's try to characterize these days with the use of hashtags – for these, we also reached into the 'extended_tweets' node.

We split up the month's data for each country into that on the day of interest and that on every other day (i.e. apart from the day of interest). Then we get all their hashtags as a list.

A small function helped us visualize the percentage occurrence of said hashtags.

```
def make_hashtag_df(hashtag_list):
    '''
    makes a df of the hashtag occurences. easier to handle for me
    params: hashtag_list_counted -> list containing all occurences of hashtags
    returns: df -> dataframe containing hashtags and occurences
    '''
    hashtag_list_counted = Counter(hashtag_list)
    L = []
    for el in hashtag_list_counted:
        L.append([el , hashtag_list_counted[el]])
    df = pd.DataFrame(L , columns = ['hashtag','occurence']).sort_values('occurence' ,
ascending = False)
    df['%occurence'] = df['occurence']/df['occurence'].sum()
    return df
```

For the Netherlands, on the day of interest, i.e. 11th June, 2022, here are the most common hashtags →

hashtag	occurrence	fraction_occurence
p2000	403	0.07326
NEDPOL	168	0.03054
RRM	94	0.01709

Comparing that to other days,

hashtag	occurrence	fraction_occurence
p2000	11277	0.081272
brugopen	2573	0.018543
ambulance	2317	0.016698

It is interesting to see that ratio of p2000 tweets actually dropped slightly on the day of interest. Hence, the next obvious candidate for what special thing happened that day is #NEDPOL, referring to the football match that day between the Netherlands and Poland.

For the United Kingdom, on the day of interest, i.e. 24th June, 2022, here are the most common hashtags →

hashtag	occurrence	fraction_occurence
LoveIsland	444	0.01015
Glastonbury2022	412	0.00942

RoeVsWade	373	0.00853

Comparing that to other days,

hashtag	occurrence	fraction_occurence
LoveIsland	11947	0.0104
PlatinumJubilee	6478	0.00564
loveisland	3187	0.00278

Though LoveIsland seems to be a consistently predominant topic, the overturning of Roe Vs Wade seems to have been heavily discussed on this particular day. <SPACE FOR ARTICLE>

For France, on the day of interest – 19th June, here are the most common hashtags →

hashtag	occurrence	fraction_occurrence
legislatives2022	1345	0.08898
NUPES	178	0.01178
chat	146	0.00966

Comparing that to other days,

hashtag	occurrence	fraction_occurrence
legislatives2022	8030	0.018948
chat	4085	0.009639
NUPES	3738	0.00882

Although we do not see different hashtags on the day of interest, it is worth noting that the fractional occurrence of "legislatives2022" has increased from 1.8% to 8.8%, likely hinting at the increased discussion around this topic on 19th June.

5. Reflection