research_data

May 31, 2023

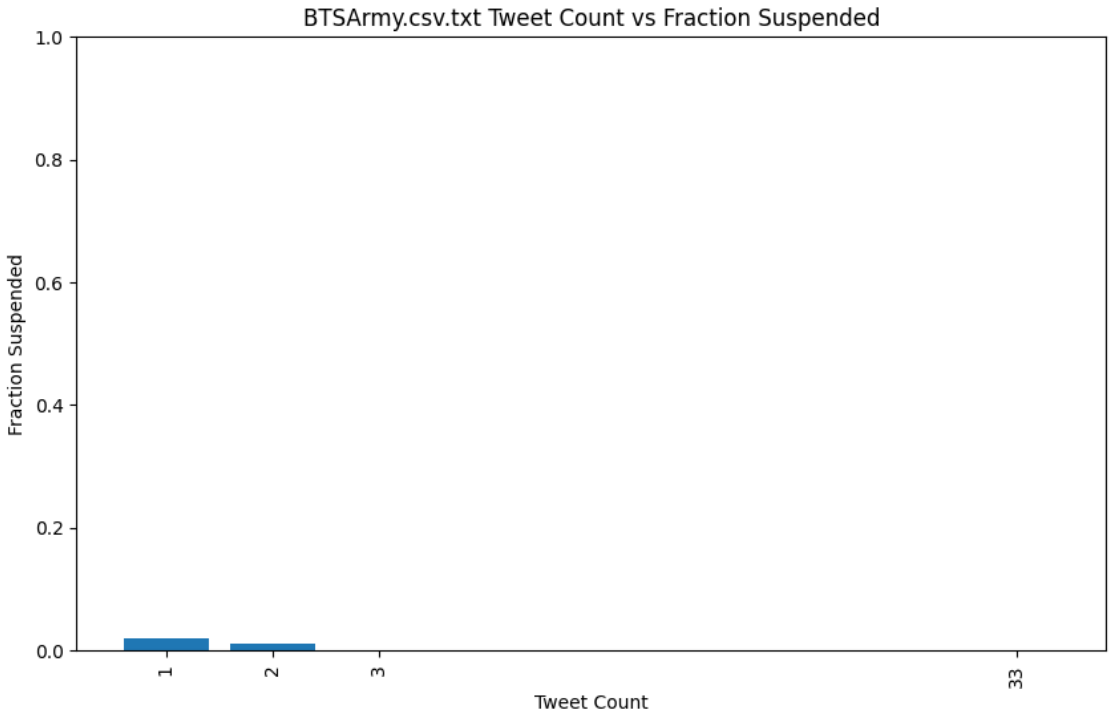
1 Twitter Data

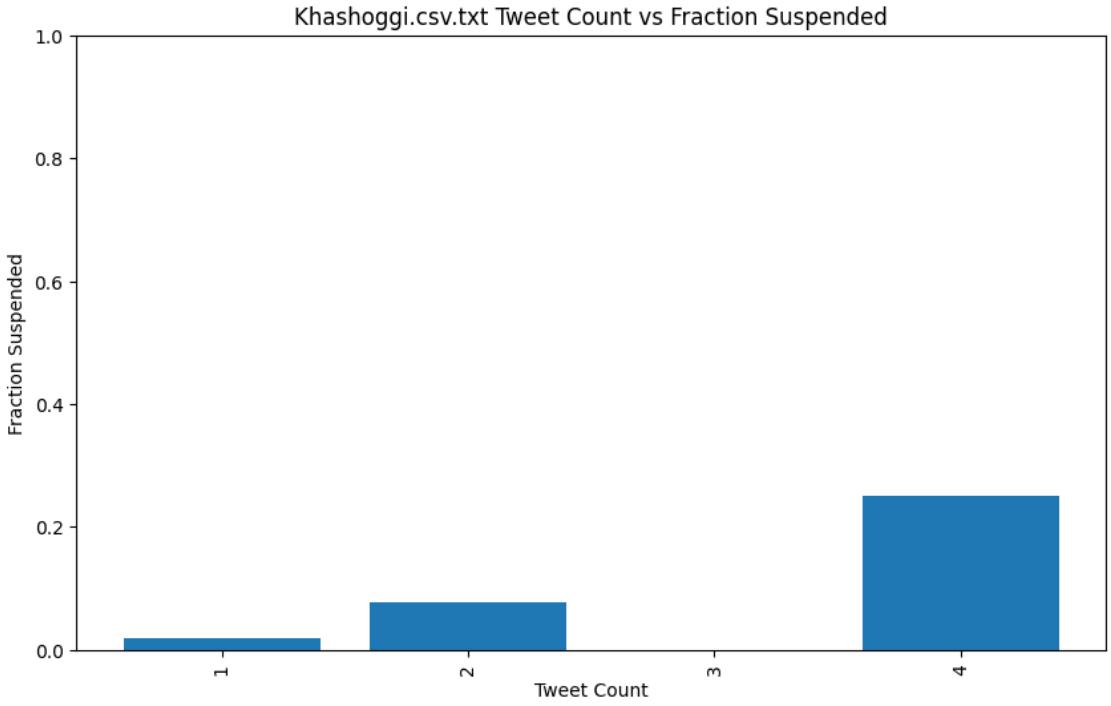
The code below creates bar graphs for the four hashtag datasets, with the x-axis being the number of times an account tweeted about a given hashtag (e.g., #QAnon) and the proportion of those users that were suspended on the y-axis.

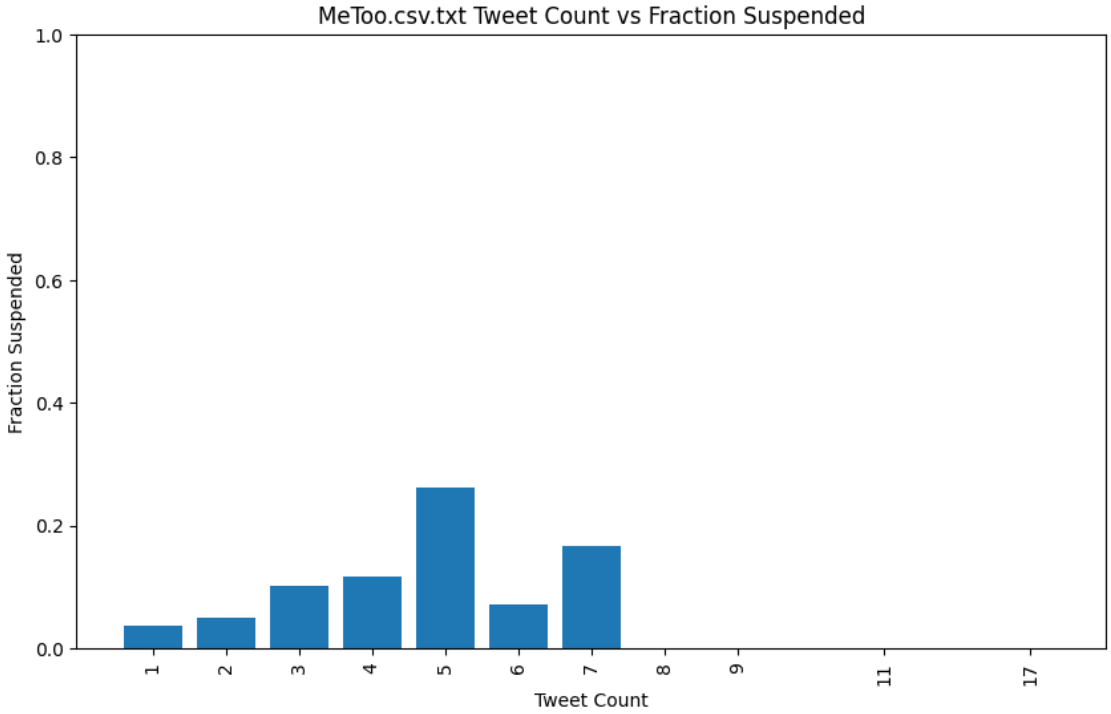
As observed, the datasets lack sufficient data, particularly for BTSArmy and Khashoggi. Only 3570 and 1313 users tweeted once for each hashtag, respectively, with a sharp decline to low single-digit users. I have created an inclusion criterion for the graph, requiring at least three users for a particular tweet count to be included.

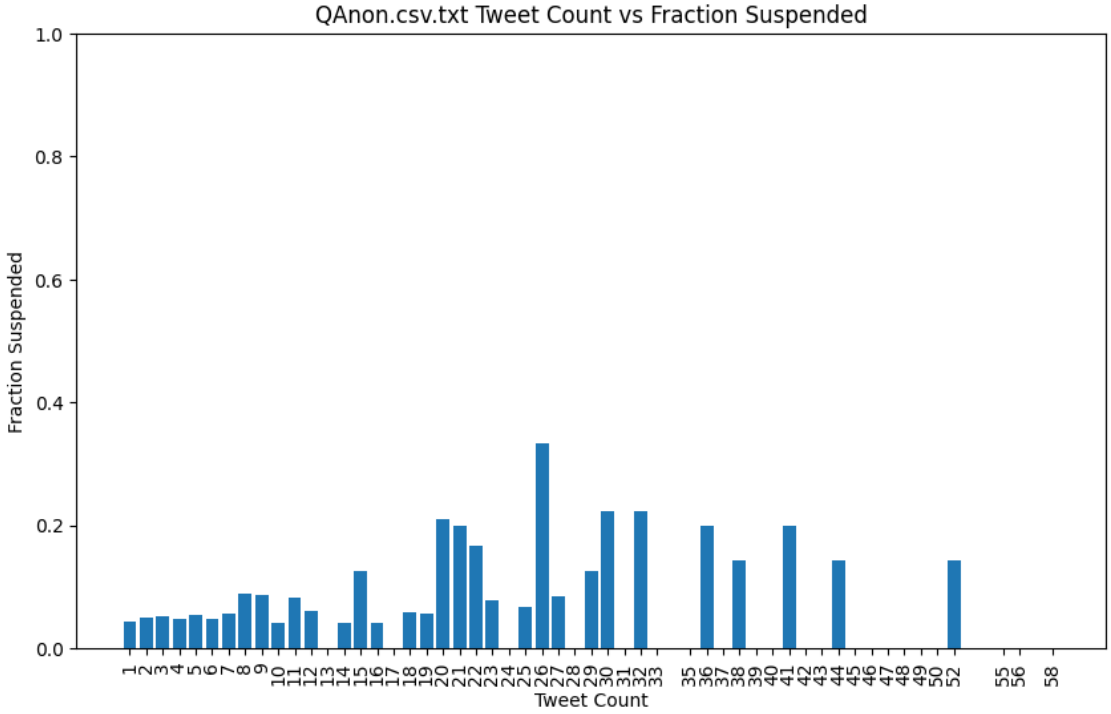
[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

files = ['./Twitter_Data/BTSArmy.csv.txt', './Twitter_Data/Khashoggi.csv.txt',␣
↪'./Twitter_Data/MeToo.csv.txt', './Twitter_Data/QAnon.csv.txt']

for file in files:
    # Read the CSV file
    data = pd.read_csv(file)

    # Group by total_tweets and aggregate the total no. of users and suspended␣
↪accounts
    grouped_data = data.groupby('user_id').agg({'tweet_id': 'count',␣
↪'suspended': 'max'}).reset_index()
    grouped_data.columns = ['user_id', 'total_tweets', 'suspended']

    # Sort the data by total_tweets in ascending order
    grouped_data = grouped_data.sort_values(by='total_tweets', ascending=True)

    # Save
    output_file = file.replace('.csv.txt', '_Tweet_By_User.csv.txt')
    grouped_data.to_csv(output_file, index=False)

    # Group by total_tweets and aggregate the total no. of users and suspended␣
↪accounts
    grouped_tweet_count = grouped_data.groupby('total_tweets').agg({'user_id':␣
↪'count', 'suspended': 'sum'}).reset_index()
```

```
    grouped_tweet_count.columns = ['tweet_count', 'total_users',␣
↪'total_suspended']

    # Fraction of suspended accounts
    grouped_tweet_count['fraction_suspended'] =␣
↪grouped_tweet_count['total_suspended'] / grouped_tweet_count['total_users']

    # Save
    output_file = file.replace('.csv.txt', '_Tweet_By_Count.csv.txt')
    grouped_tweet_count.to_csv(output_file, index=False)

    # print tweet count data
    #print(" Tweet count data of " + file.split("Data/")[1] + "\n")
    #print(grouped_tweet_count.to_string(index=False))


    # Read the new file
    data = pd.read_csv(output_file)
    data = data[data['total_users'] >= 3] # inclusion criteria
    plt.figure(figsize=(10, 6))

    # plot the bar
    plt.bar(data.index, data['fraction_suspended'])

    # x-axis with ticks
    plt.xticks(data.index, data['tweet_count'], rotation=90)
    plt.ylim(0, 1)
    plt.xlabel('Tweet Count')
    plt.ylabel('Fraction Suspended')
    plt.title(f'{file.split("Data/")[1]} Tweet Count vs Fraction Suspended')
    plt.show()
```


BTSArmy.csv.txt Tweet Count vs Fraction Suspended
Fraction Suspended
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
33
Tweet Count


Khashoggi.csv.txt Tweet Count vs Fraction Suspended
Fraction Suspended
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
Tweet Count


MeToo.csv.txt Tweet Count vs Fraction Suspended
Fraction Suspended
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
5
6
7
8
9
11
17
Tweet Count

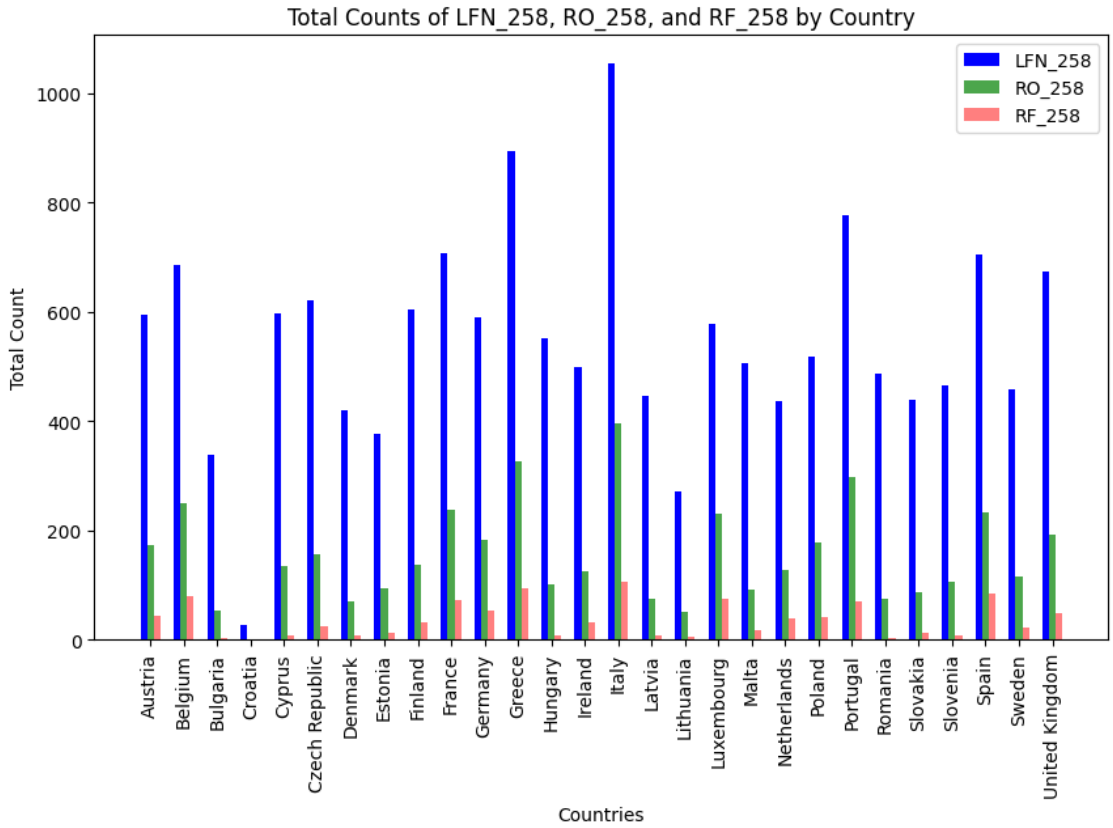

QAnon.csv.txt Tweet Count vs Fraction Suspended
Fraction Suspended
1.0
0.8
0.6
0.4
0.2
0.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 52 55 56 58
Tweet Count

2 EU Data

2.0.1 Non-Compliance

Non-Compliance of a country can be determined by the number of LFN, RO, and RFs. The code below plots a graph that maps the total no. of LFNs, RO, and RF for each country. The highest no. of LFNs recorded were for Italy, Greece, and Portugal.

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

data = pd.read_csv('./EU_Data/eucommission.csv')

# Group by member_state
grouped_data = data.groupby('member_state').agg({
    'LFN_258': 'sum',
    'RO_258': 'sum',
    'RF_258': 'sum'
}).reset_index()

plt.figure(figsize=(10, 6))

# Get the countries
countries = grouped_data['member_state']

lfn_counts = grouped_data['LFN_258']
ro_counts = grouped_data['RO_258']
rf_counts = grouped_data['RF_258']

bar_width = 0.2
x_positions = np.arange(len(countries))

# Plot the bars for each category# Set the y-axis range from 0 to 150
plt.bar(x_positions - bar_width, lfn_counts, width=bar_width, label='LFN_258',␣
 ↪color='blue')
plt.bar(x_positions, ro_counts, width=bar_width, label='RO_258', color='green',␣
 ↪alpha=0.7)
plt.bar(x_positions + bar_width, rf_counts, width=bar_width, label='RF_258',␣
 ↪color='red', alpha=0.5)

plt.xticks(x_positions, countries, rotation=90)

plt.xlabel('Countries')
plt.ylabel('Total Count')
plt.title('Total Counts of LFN_258, RO_258, and RF_258 by Country')
plt.legend()
plt.show()
```



2.1 Non-Compliance Data Over Time

I plotted the number of LFNs a country receives over time. To improve visualization, I implemented a rolling window approach. I used a fixed window size of 6 months and calculated the average LFNs within that window for every month.

Note that the below graphs are different for each country as their combination would result in a cluttered and visually confusing representation.

[ ]:
```
import pandas as pd
from datetime import datetime
import matplotlib.pyplot as plt

data = pd.read_csv('./EU_Data/eucommission.csv')

# Get the countries in the dataset
countries = data['member_state'].unique()

for country in countries:
    # Filter the data for the current country
    country_data = data[data['member_state'] == country]
```

```
    country_data = country_data.dropna(subset=['date_LFN_258'])

    # convert date str to obj
    date_lfn_country = country_data['date_LFN_258'].apply(lambda x: datetime.
↪strptime(x, '%m/%d/%y'))

    # Group by month and year and count the number of occurrences
    counts = date_lfn_country.groupby([date_lfn_country.dt.year,␣
↪date_lfn_country.dt.month]).count()

    # assign 0 to all the missing months
    all_months = pd.MultiIndex.from_product([range(counts.index.
↪get_level_values(0).min(), counts.index.get_level_values(0).max() + 1),
                                             range(1, 13)],
                                             names=['Year', 'Month'])
    counts = counts.reindex(all_months, fill_value=0)

    # Calculate the rolling average with a window size of 6 months
    counts = counts.rolling(window=6, min_periods=1).mean()

    # Convert the index to string of month/year format
    counts.index = counts.index.map(lambda x: f'{x[1]}/{x[0]}')

    # Plot the data
    plt.figure(figsize=(14, 6))
    plt.plot(counts.index, counts.values, label=country)
    plt.xticks(rotation=90)

    # Remove Ident to see Data for all countries in one graph
    plt.ylim(0, 40)
    plt.xlabel('Time (Month/Year)')
    plt.ylabel('Total Number of LFN')
    plt.title('Total Number of LFN vs Time')
    plt.legend()

    plt.xticks(range(0, len(counts.index), 12), counts.index[::12], rotation=90)

    plt.show()
```

2.1.1 Left-Right Ideology

The data set contains a continuous variable called out_left_cont that indicates the difference in right ideology between the ruling and the largest opposition party. Larger values indicate the ruling party is more left-wing compared to the opposition.

I have plotted graphs to observe the change of the variable over time for each country. Again, I tried putting all countries on the same graph but their combination resulted in a cluttered and visually confusing representation.

[ ]:
```python
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv('./EU_Data/eucommission.csv')

# Group the data by member_state and year, and select the out_left_cont column
yearly_data = data.groupby(['member_state', 'year'])['out_left_cont'].first()

# Reset the index
yearly_data = yearly_data.reset_index()

for i, country in enumerate(yearly_data['member_state'].unique()):
    country_data = yearly_data[yearly_data['member_state'] == country]
    plt.figure(figsize=(12, 6))
    plt.plot(country_data['year'], country_data['out_left_cont'], label=country)

    # Remove Ident to see Data for all countries in one graph
    plt.ylim(-8, 8)
    plt.xlabel('Year')
    plt.ylabel('out_left_cont')
    plt.title('out_left_cont values for each year (per country)')
    plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
    plt.show()
```