notes

June 12, 2023

1 Twitter Data (Week 2)

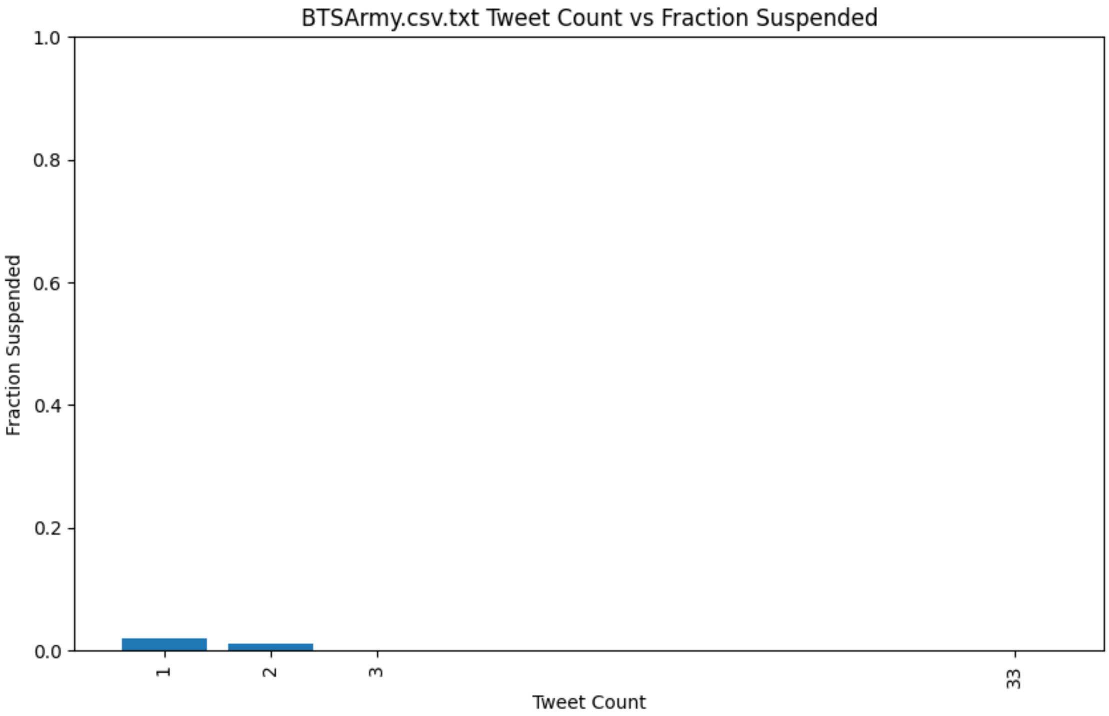
1.0.1 1.1 Fraction Suspended

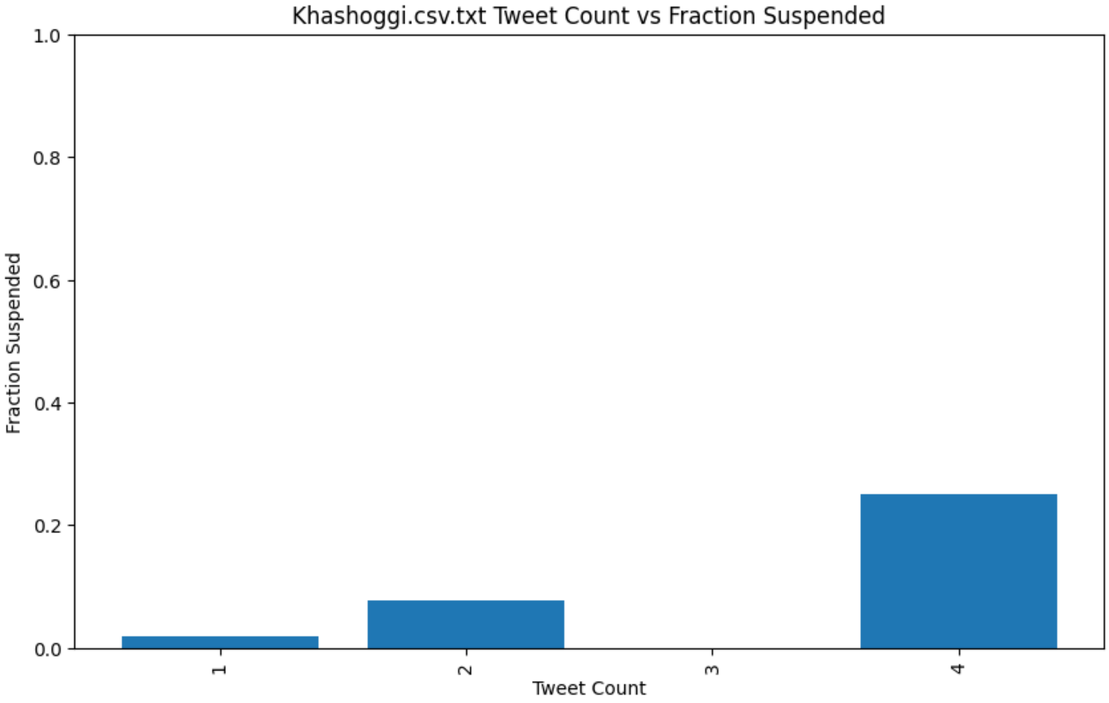
The code below creates bar graphs for the four hashtag datasets, with the x-axis being the number of times an account tweeted about a given hashtag (e.g., #QAnon) and the proportion of those users that were suspended on the y-axis.

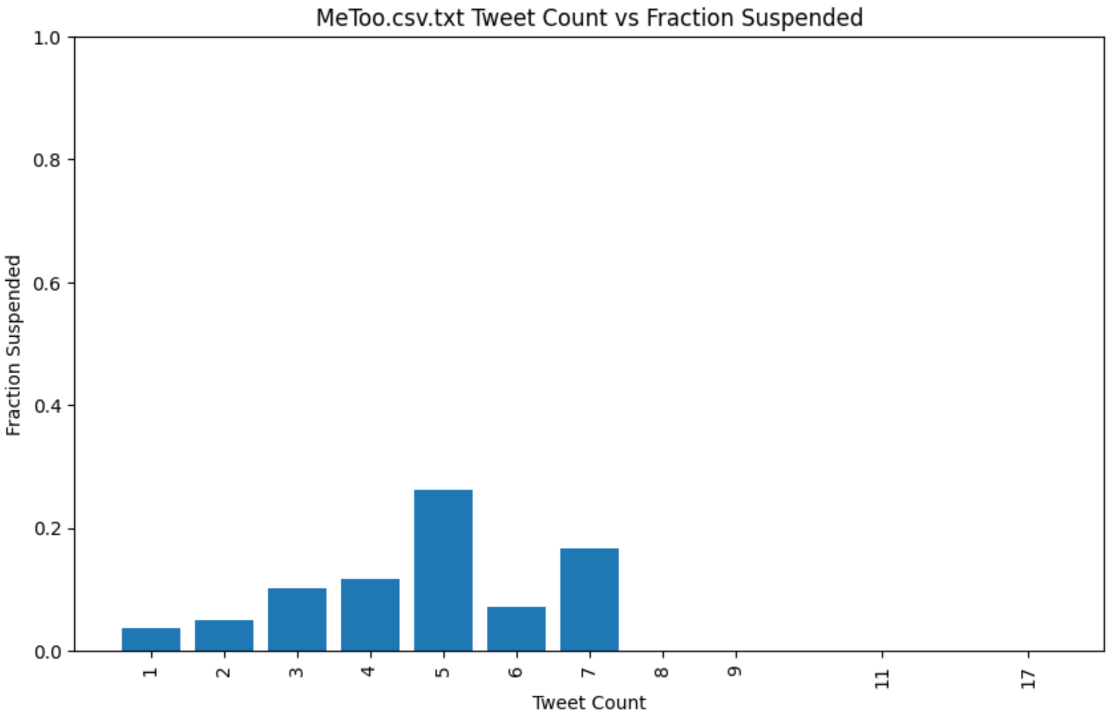
As observed, the datasets lack sufficient data, particularly for BTSArmy and Khashoggi. Only 3570 and 1313 users tweeted once for each hashtag, respectively, with a sharp decline to low single-digit users. I have created an inclusion criterion for the graph, requiring at least three users for a particular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

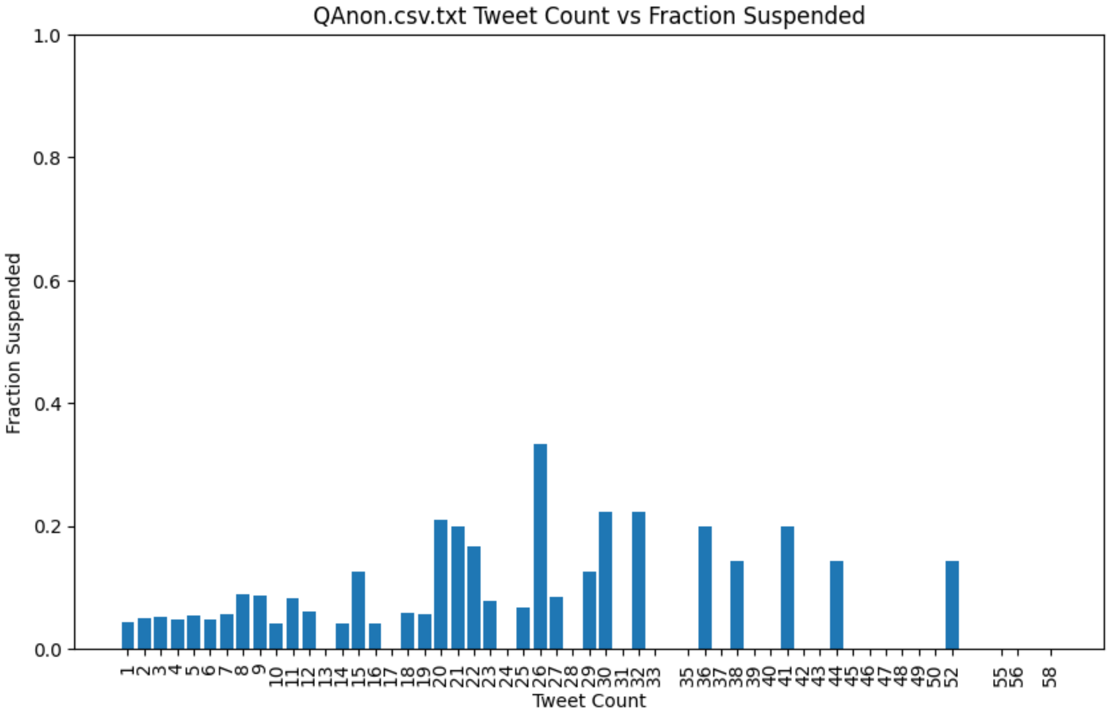
[ ]:

```python
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 2.1 Non-Compliance (Week 2-3)

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.

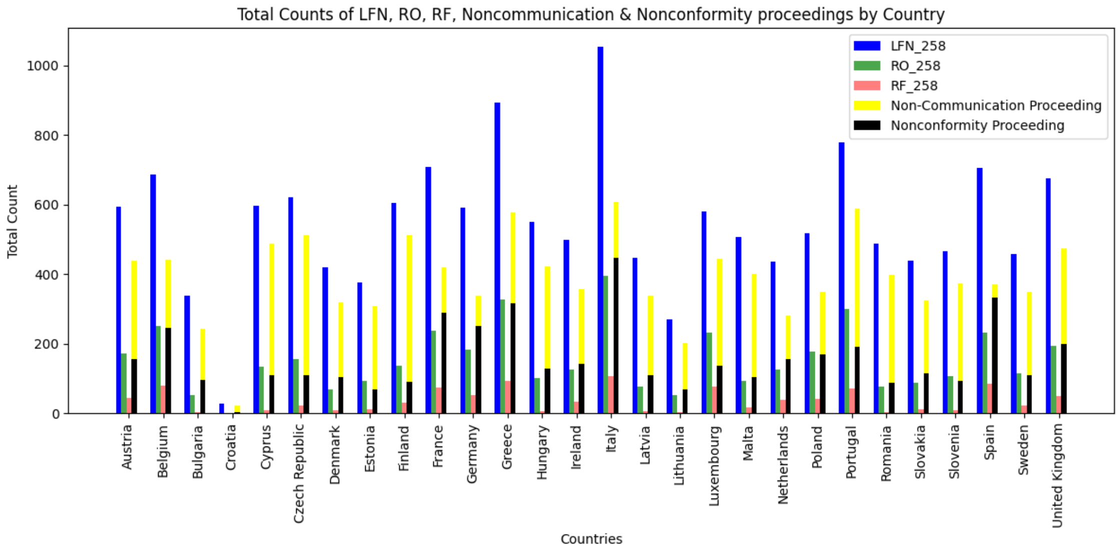
Week 3:

I added the total number of Non-Conformity and Non-Communcation proceedings per country to the graphs as well.

[ ]:

```
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',
    'incorrect': 'sum'
}).reset_index()

grouped_data_2 =  data.groupby('member_state').agg({
    'incorrect': lambda x: x.eq(0).sum()
}).reset_index()

plt.figure(figsize=(12, 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']
incorrect_1_counts = grouped_data['incorrect']
incorrect_0_counts = grouped_data_2['incorrect']

bar_width = 0.15
x_positions = np.arange(len(countries))

# Plot the bars for each category
plt.bar(x_positions - 2 * bar_width, lfn_counts, width=bar_width,␣
  ↪label='LFN_258', color='blue')
```

```
plt.bar(x_positions - bar_width, ro_counts, width=bar_width, label='RO_258', color='green', alpha=0.7)
plt.bar(x_positions, rf_counts, width=bar_width, label='RF_258', color='red', alpha=0.5)
plt.bar(x_positions + bar_width, incorrect_0_counts, width=bar_width, label='Non-Communication Proceeding', color='yellow')
plt.bar(x_positions + bar_width, incorrect_1_counts, width=bar_width, label='Nonconformity Proceeding', color='black')


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

plt.xlabel('Countries')
plt.ylabel('Total Count')
plt.title('Total Counts of LFN, RO, RF, Noncommunication & Nonconformity proceedings by Country')
plt.legend()
plt.tight_layout()
plt.show()
```



2.0.2 Non-Compliance Data Over Time (Week 2-3)

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.

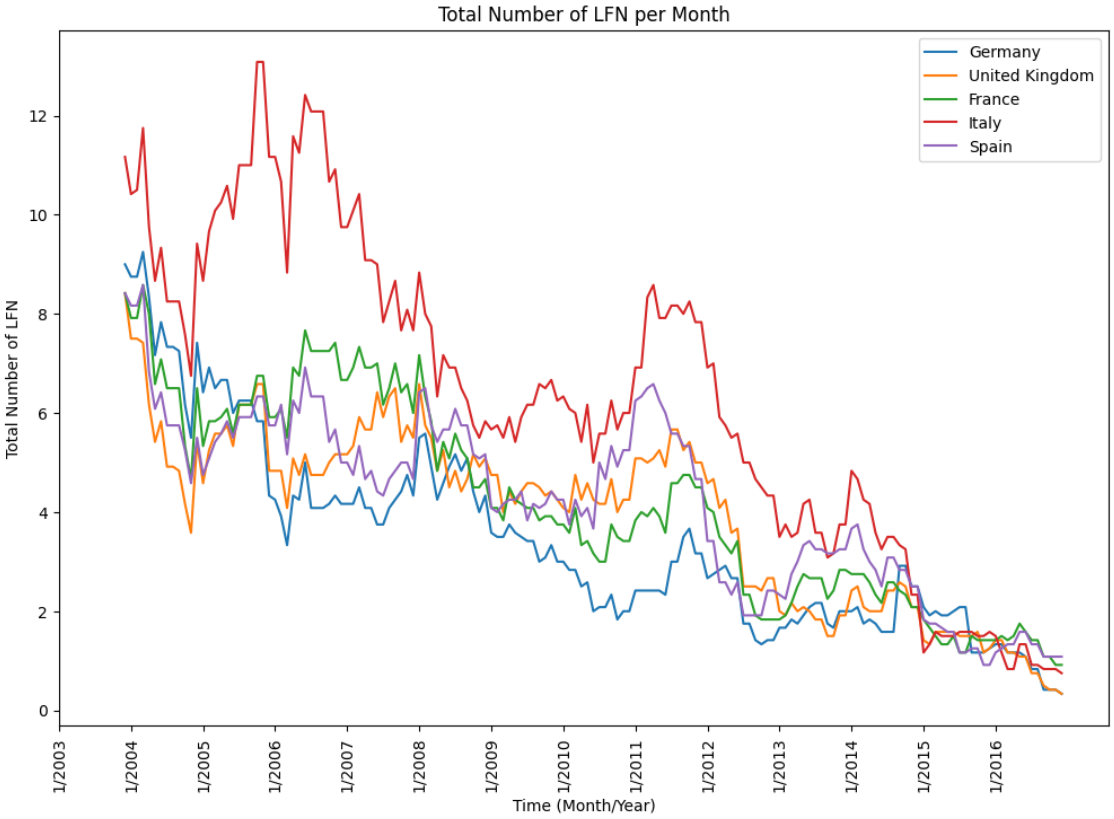
The combination of all countries in a graph resulted in a cluttered and visually confusing representation.

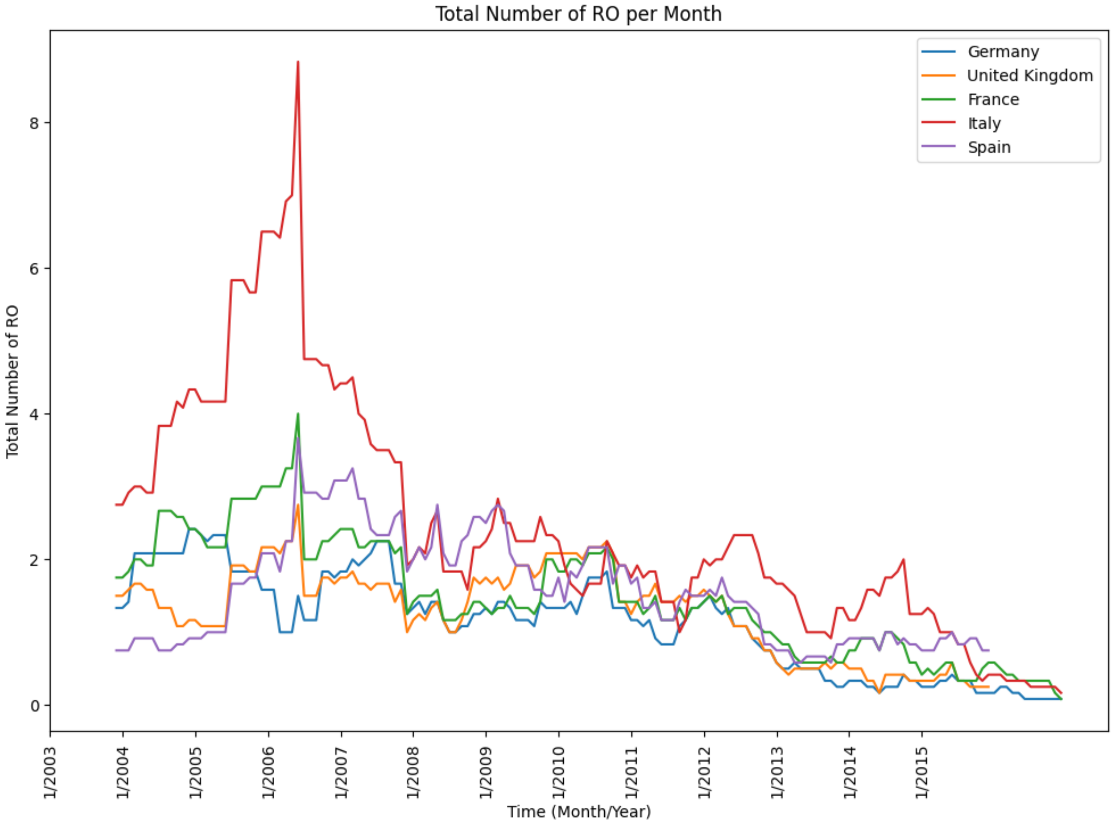
Week 3:

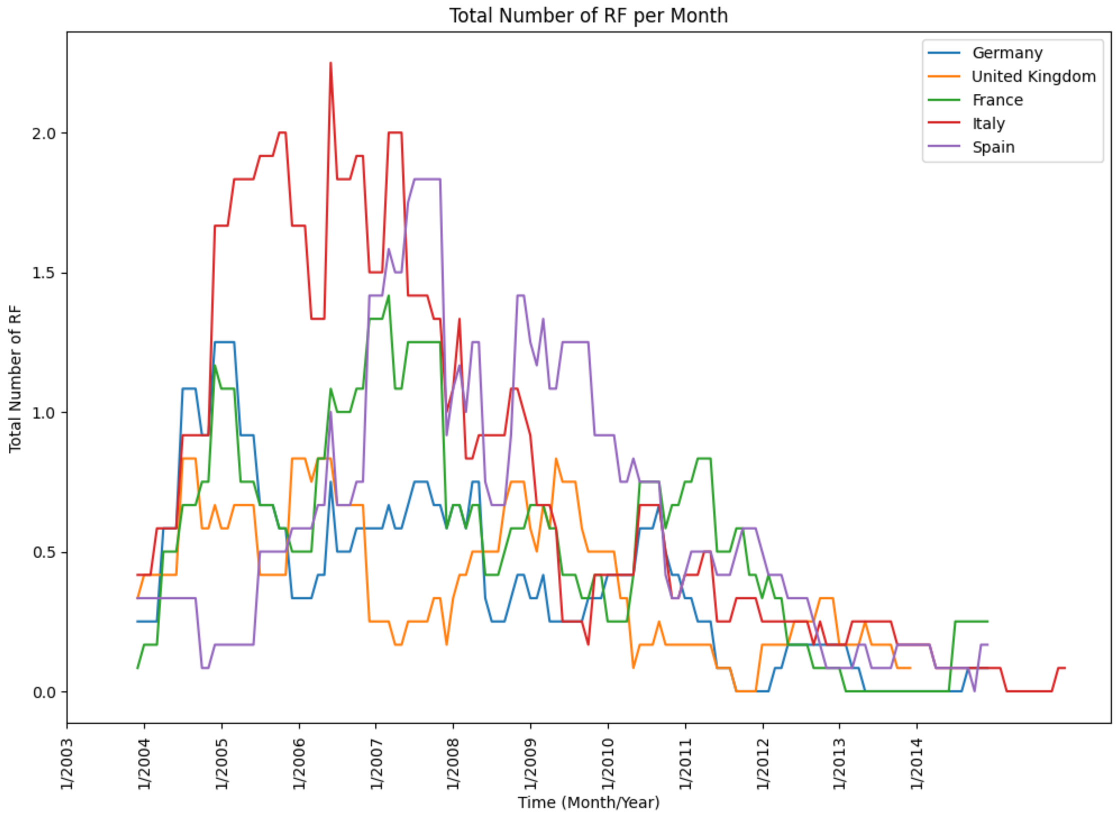
I have plotted trends for the top 5 populated countries for the total no. of LFNs, ROs, and RFs over time.

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

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

     # Top 5 Populated Countries
     countries = ['Germany', 'United Kingdom', 'France', 'Italy', 'Spain']

     dataset = ['date_LFN_258', 'date_RO_258' , 'date_RF_258']

     for date in dataset:
         plt.figure(figsize=(12, 8))
         for country in countries:
             # Filter the data for the current country
             country_data = data[data['member_state'] == country]

             # Remove rows with NA values in the 'date_LFN_258' column
             country_data = country_data.dropna(subset=[date])

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

             # Group the data 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 a value of 0 for 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)

             # Rolling average with a window size of 12 months
             counts = counts.rolling(window=12).mean()

             counts.index = counts.index.map(lambda x: f'{x[1]}/{x[0]}')

             plt.plot(counts.index, counts.values, label=country)
             plt.xticks(rotation=90)
```

```
    plt.xlabel('Time (Month/Year)')
    plt.ylabel('Total Number of ' + date.split('_')[1])
    plt.title('Total Number of ' + date.split('_')[1] + ' per Month')
    plt.legend()
    plt.xticks(range(0, len(counts.index), 12), counts.index[::12], rotation=90)
    #plt.savefig('plot.png')
    # Display the plot
    plt.show()
```




Total Number of RO per Month
Germany
United Kingdom
France
Italy
Spain
Total Number of RO
0
2
4
6
8
1/2003
1/2004
1/2005
1/2006
1/2007
1/2008
1/2009
1/2010
1/2011
1/2012
1/2013
1/2014
1/2015
Time (Month/Year)



2.0.3 Left-Right Ideology (Week 2-3)

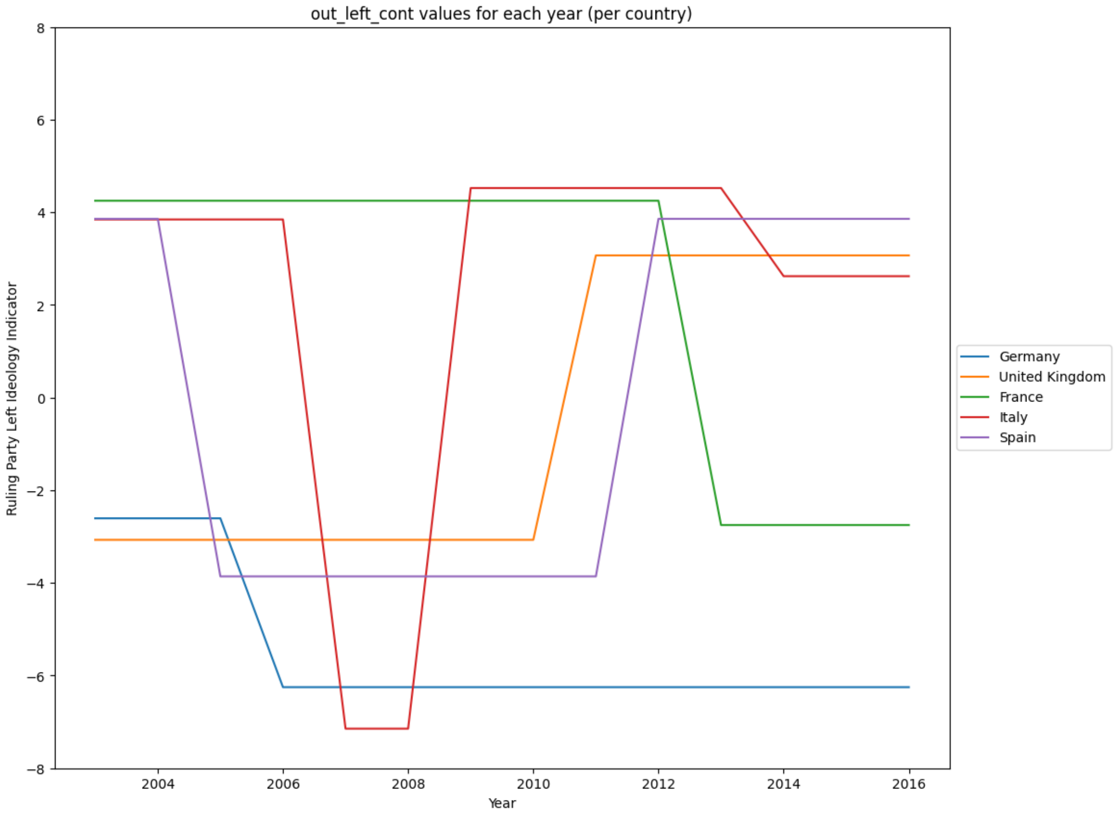
The data set contains a continuous variable called out_left_cont that indicates the difference in right ideology between the ruling and the largest opposition party. Larger values indicate the ruling party is more left-wing compared to the opposition.

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. Therefore, I decided to plot the trends for the top 5 populous countries.

[ ]:

```
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()
```

```python
# Top 5 Populated Countries
countries = ['Germany', 'United Kingdom', 'France', 'Italy', 'Spain']#, 'Poland', 'Romania', 'Netherlands', 'Belgium', 'Czech Republic']

plt.figure(figsize=(12, 10))
for i, country in enumerate(countries):
    country_data = yearly_data[yearly_data['member_state'] == country]
    plt.plot(country_data['year'], country_data['out_left_cont'], label=country)

# Ident to see Data for graph for each country
plt.ylim(-8, 8)
plt.xlabel('Year')
plt.ylabel('Ruling Party Left Ideology Indicator')
plt.title('out_left_cont values for each year (per country)')
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()
```



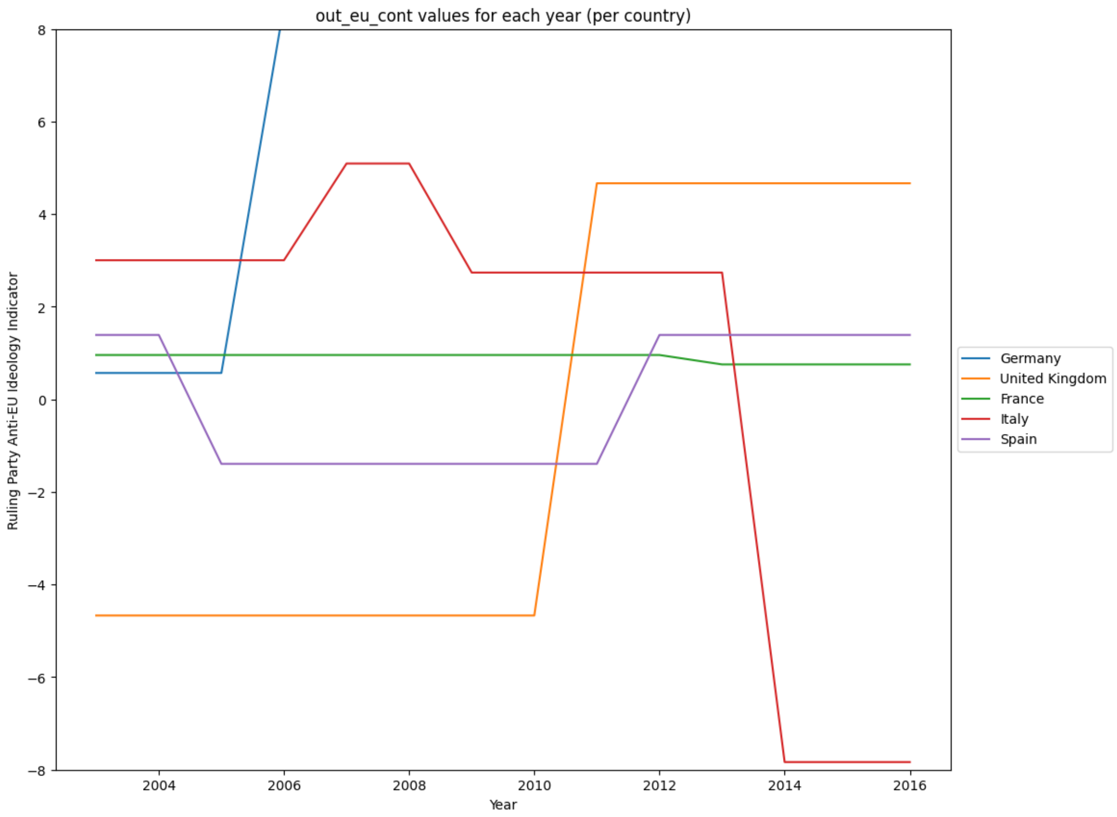
2.0.4 Pro-EU Ideology (Week 3)

The data set contains a continuous variable called out_eu_cont that indicates the difference in EU ideology between the prime minister's party and the largest opposition party in government. Larger values indicate the opposition party is more pro-EU.

Similar to the above graph, I have plotted its change over time for the top 5 countries.

[ ]:

```
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_eu_cont'].first()

# Reset the index
yearly_data = yearly_data.reset_index()

# Top 5 Populated Countries
countries = ['Germany', 'United Kingdom', 'France', 'Italy', 'Spain']#,␣
 ↪'Poland', 'Romania', 'Netherlands', 'Belgium', 'Czech Republic']            ␣
 ↪

plt.figure(figsize=(12, 10))
for i, country in enumerate(countries):
    country_data = yearly_data[yearly_data['member_state'] == country]
    plt.plot(country_data['year'], country_data['out_eu_cont'], label=country)

# Ident to see Data for graph for each country
plt.ylim(-8, 8)
plt.xlabel('Year')
plt.ylabel('Ruling Party Anti-EU Ideology Indicator')
plt.title('out_eu_cont values for each year (per country)')
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.show()
```



2.0.5 Workload (Week 3)

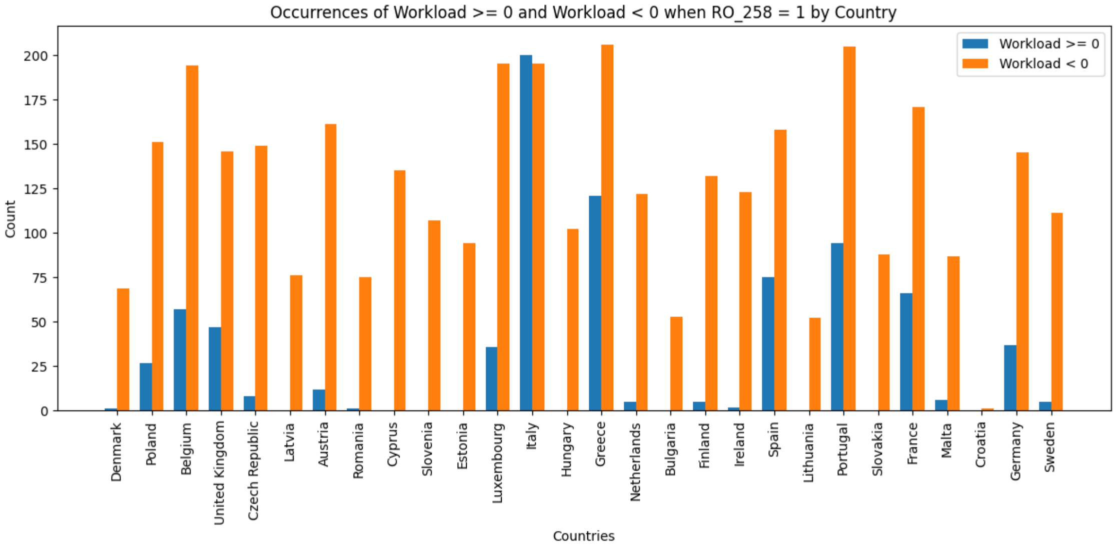
The data set contains a continuous variable called worload, which indicates the number of open infringement proceedings the Commission has against the member state at the time it issues an LFN. Cheruvu's paper interprets that as the worload increases and the Commission hits more resource constraints, the Commission will have higher costs for moving a proceeding from the LFN stage to the RO stage. Unfortunately, it is difficult to prove/map this because the worload variable is calculated at the time of the LFN issue date and not the RO date.

However, if we do map it, there are more ROs when the worload was less than zero (less no. of open proceedings) compared to positive worloads, as you can see from the figure below.

[ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

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

# Filter the data where RO_258 is equal to 1 and worload >= 0
ro258_greater_zero = data[(data['RO_258'] == 1) & (data['worload'] >= 0)]

# Filter the data where RO_258 is equal to 1 and worload < 0
```

```python
ro258_less_zero = data[(data['RO_258'] == 1) & (data['workload'] < 0)]

countries = data['member_state'].unique()

# Calculate the count for workload > 0 and workload < 0 for each country
ro258_greater_zero_count = ro258_greater_zero.groupby('member_state')['RO_258'].count()
ro258_less_zero_count = ro258_less_zero.groupby('member_state')['RO_258'].count()

# Fill missing countries with count 0
ro258_greater_zero_count = ro258_greater_zero_count.reindex(countries, fill_value=0)
ro258_less_zero_count = ro258_less_zero_count.reindex(countries, fill_value=0)

plt.figure(figsize=(12, 6))
bar_width = 0.35
x_positions = np.arange(len(countries))

plt.bar(x_positions - bar_width/2, ro258_greater_zero_count, width=bar_width, label='Workload >= 0')
plt.bar(x_positions + bar_width/2, ro258_less_zero_count, width=bar_width, label='Workload < 0')

plt.xticks(x_positions, countries, rotation=90)
plt.xlabel('Countries')
plt.ylabel('Count')
plt.title('Occurrences of Workload >= 0 and Workload < 0 when RO_258 = 1 by Country')
plt.legend()
plt.tight_layout()
plt.show()
```



[ ]:

```
import pandas as pd

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

# Total number of LFNs
total_lfns = data['LFN_258'].sum()

# Total number of ROs
total_ros = data['RO_258'].sum()

# Elections that overlapped with LFN Date
total_lfn_election = data['LFN_election'].sum()

print("Total Number of LFNs:", total_lfns)
print("Total Number of ROs:", total_ros)
print("Total Number of LFN_election:", total_lfn_election)
```

```
Total Number of LFNs: 15333
Total Number of ROs: 4308
Total Number of LFN_election: 3145
```

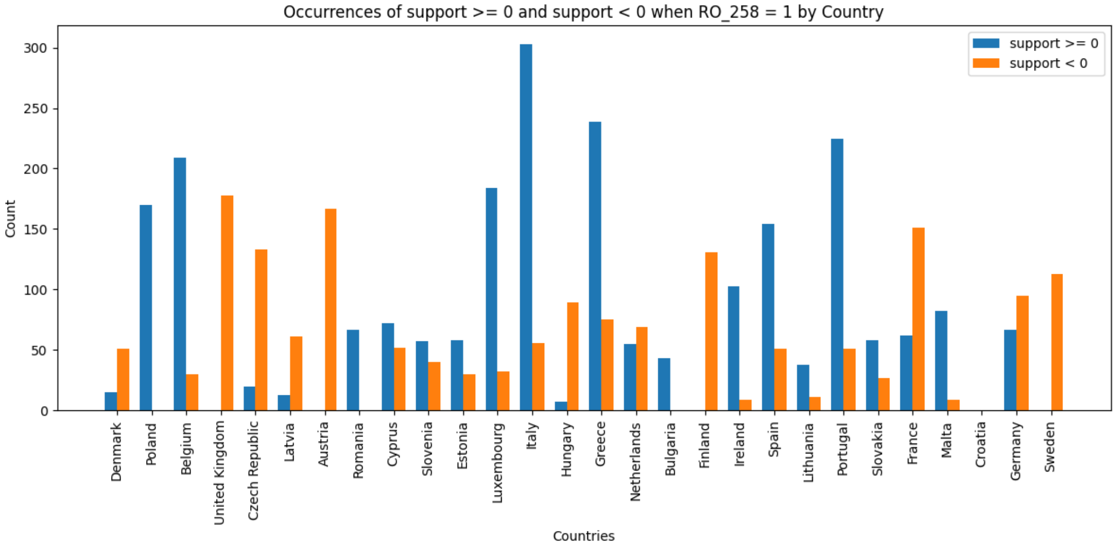
2.0.6 Support Variable (Week 4)

The data set contains a continuous variable called support, which indicates the public opinion of the EU in each member state is another aspect of a member state's cost of compliance. In sum, as support for EU integration decreases, compliance with EU law on average should be more costly for member states.

As observed from the figure below, over 16 of the countries have more ROs issued for the cases when the public supported EU. Interestingly, countries UK, Finland, Austria, and Sweden had public opinions against the EU for cases advanced to RO.

[ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

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

# Filter the data where RO_258 is equal to 1 and support >= 0
ro258_greater_zero = data[(data['RO_258'] == 1) & (data['support'] >= 0)]

# Filter the data where RO_258 is equal to 1 and support < 0
ro258_less_zero = data[(data['RO_258'] == 1) & (data['support'] < 0)]

countries = data['member_state'].unique()

# Calculate the count for support > 0 and support < 0 for each country
ro258_greater_zero_count = ro258_greater_zero.groupby('member_state')['RO_258'].
  ↪count()
ro258_less_zero_count = ro258_less_zero.groupby('member_state')['RO_258'].
  ↪count()

# Fill missing countries with count 0
ro258_greater_zero_count = ro258_greater_zero_count.reindex(countries,␣
  ↪fill_value=0)
ro258_less_zero_count = ro258_less_zero_count.reindex(countries, fill_value=0)

plt.figure(figsize=(12, 6))
bar_width = 0.35
x_positions = np.arange(len(countries))

plt.bar(x_positions - bar_width/2, ro258_greater_zero_count, width=bar_width,␣
  ↪label='support >= 0')
plt.bar(x_positions + bar_width/2, ro258_less_zero_count, width=bar_width,␣
  ↪label='support < 0')

plt.xticks(x_positions, countries, rotation=90)
plt.xlabel('Countries')
plt.ylabel('Count')
plt.title('Occurrences of support >= 0 and support < 0 when RO_258 = 1 by␣
  ↪Country')
plt.legend()
plt.tight_layout()
plt.show()
```



2.0.7 Multiple Election Data for a Case (Week 4)

Check if there are cases where one case has several relevant elections coming up after the LFN is issued and there is a different out_eu_cont value for all of them.

[ ]:

```
import pandas as pd

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

# Top 5 Populated Countries
countries = ['Germany', 'United Kingdom', 'France', 'Italy', 'Spain']
data = data[data['member_state'].apply(lambda x: x in countries)]

# Group the data by country and case_number
grouped_data = data.groupby(['member_state', 'case_number']).size()

# Find the case numbers that were repeated in each country
repeated_case_numbers = grouped_data[grouped_data > 1]

# Loop over the countries and print the repeated case numbers along with their␣
 ↪out_eu_cont values
for country in repeated_case_numbers.index.get_level_values('member_state').
 ↪unique():
    print(country)
    country_data = data[data['member_state'] == country]
    repeated_cases = repeated_case_numbers[country].index
    for case_number in repeated_cases:
        out_eu_cont_values = country_data[country_data['case_number'] ==␣
 ↪case_number]['out_eu_cont'].unique()
```

```
        print(f"Case Number: {case_number}, out_eu_cont Values:␣
↪{out_eu_cont_values}")
    print()
```

```
France
Case Number: 20062110, out_eu_cont Values: [0.9586]

Germany
Case Number: 20034350, out_eu_cont Values: [8.5024]
Case Number: 20042129, out_eu_cont Values: [0.5706 8.5024]
Case Number: 20074866, out_eu_cont Values: [8.5024]

Italy
Case Number: 20015308, out_eu_cont Values: [3.0054 5.0946]
Case Number: 20024801, out_eu_cont Values: [3.0054 5.0946]
Case Number: 20025058, out_eu_cont Values: [3.0054 5.0946]
Case Number: 20034524, out_eu_cont Values: [3.0054 5.0946]
Case Number: 20034616, out_eu_cont Values: [3.0054 5.0946]
Case Number: 20034696, out_eu_cont Values: [3.0054 5.0946]
Case Number: 20034722, out_eu_cont Values: [3.0054 5.0946]
Case Number: 20034762, out_eu_cont Values: [3.0054 5.0946]
Case Number: 20042226, out_eu_cont Values: [3.0054 5.0946]
Case Number: 20045159, out_eu_cont Values: [3.0054 5.0946]
Case Number: 20054347, out_eu_cont Values: [3.0054 5.0946]
Case Number: 20054669, out_eu_cont Values: [3.0054 5.0946]
Case Number: 20062017, out_eu_cont Values: [3.0054 5.0946]
Case Number: 20062104, out_eu_cont Values: [3.0054 5.0946]
Case Number: 20062163, out_eu_cont Values: [3.0054 5.0946]

Spain
Case Number: 20014135, out_eu_cont Values: [ 1.3922 -1.3922]
Case Number: 20022315, out_eu_cont Values: [ 1.3922 -1.3922]
Case Number: 20032211, out_eu_cont Values: [ 1.3922 -1.3922]
Case Number: 20064665, out_eu_cont Values: [-1.3922]
Case Number: 20124154, out_eu_cont Values: [1.3922]
Case Number: 20144186, out_eu_cont Values: [1.3922]
Case Number: 20150416, out_eu_cont Values: [1.3922]
```