In [ ]:

```
Analysis of Global Events Dataset
1. Introduction
Provide an overview of the research topic, dataset, and objectives.

2. Data Cleaning and Preprocessing
Load the dataset.
Handle missing values.
Remove or impute outliers.
Convert data types if necessary.
Perform any other preprocessing steps.
python
Copy code
# Data cleaning and preprocessing code
import pandas as pd

# Load the dataset
df = pd.read_csv('earthquakeV4.csv')

# Handle missing values
df.dropna(inplace=True)

# Remove outliers
# Code for removing outliers...

# Convert data types
# Code for data type conversion...
3. Exploratory Data Analysis (EDA)
Summarize the dataset using descriptive statistics.
Explore distributions of numerical variables.
Visualize relationships between variables.
Identify any patterns or anomalies.
python
Copy code
# Exploratory data analysis code
import matplotlib.pyplot as plt
import seaborn as sns

# Summary statistics
summary_stats = df.describe()

# Visualize distributions
sns.histplot(df['NumMentions'], bins=20, kde=True)
plt.title('Distribution of NumMentions')
plt.xlabel('NumMentions')
plt.ylabel('Frequency')
plt.show()

# Visualize relationships
sns.scatterplot(x='NumMentions', y='AvgTone', data=df)
plt.title('NumMentions vs. AvgTone')
plt.xlabel('NumMentions')
plt.ylabel('AvgTone')
plt.show()
4. Time Series Analysis
Assuming a time-related column ('EventDate' or 'SQLDATE').
Convert the time-related column to datetime format.
Set the time-related column as the index.
Explore trends, seasonality, and decomposition.
python
Copy code
# Time series analysis code
df['EventDate'] = pd.to_datetime(df['EventDate'])
```

```
df_time_series = df.set_index('EventDate')
plt.figure(figsize=(12, 6))
# Code for time series analysis...
5. Visualization
Create visualizations to represent key insights.
Use plots, charts, and maps as appropriate.
Ensure visualizations are clear and informative.
python
Copy code
# Visualization code using matplotlib, seaborn, or other libraries
# Example code for creating plots and charts...
6. Machine Learning (Optional)
Apply machine learning algorithms for clustering or classification tasks.
Split data into training and testing sets.
Train and evaluate models.
Generate predictions and assess model performance.
python
Copy code
# Machine learning code (if applicable)
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
# Code for machine learning tasks...
7. Conclusion
Summarize key findings, insights, and implications from the analysis.
```

In [5]:

```
# Load the CSV dataset
df = pd.read_csv('earthquakeV4.csv')

df
```

```
/var/folders/lf/hwptlvxd6vv42x9tfj9kdx800000gn/T/ipykernel_26410/128210347
5.py:2: DtypeWarning: Columns (21,24) have mixed types. Specify dtype optio
n on import or set low_memory=False.
  df = pd.read_csv('earthquakeV4.csv')
```

Out[5]:

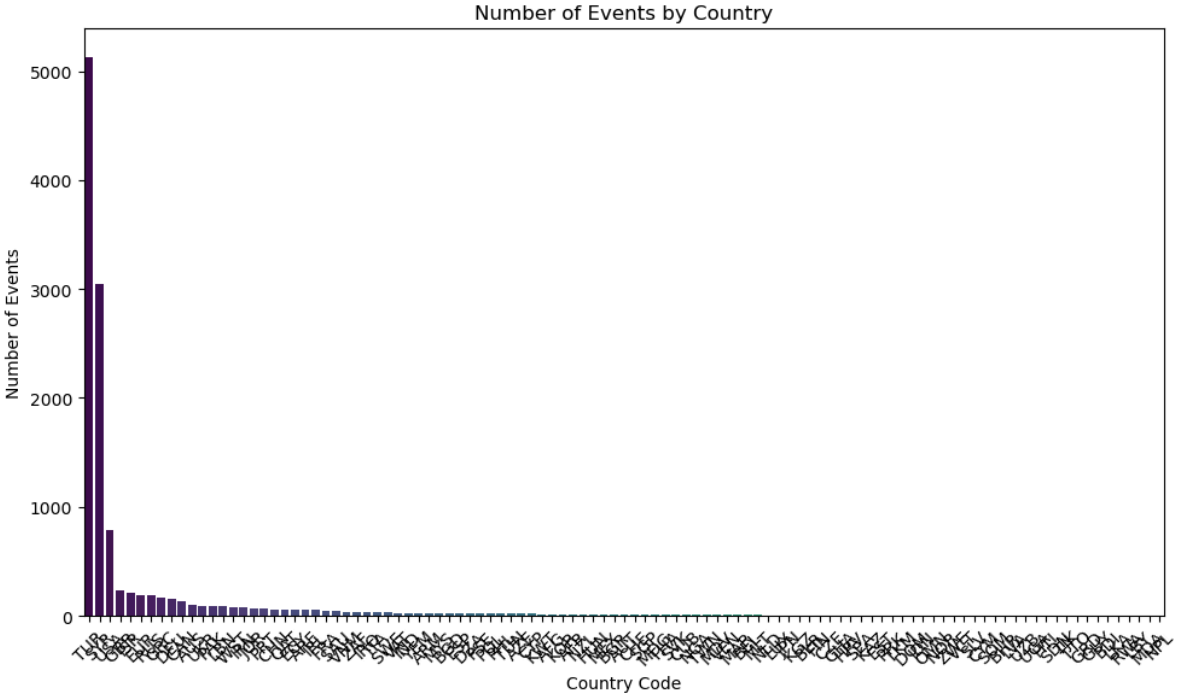
	GLOBALEVENTID	SQLDATE	MonthYear	Year	FractionDate	Actor1Code	A
0	1085521809	20230218	202302	2023	2023.1315	BUS	
1	1085574586	20230219	202302	2023	2023.1342	SYR	
2	1085574598	20230219	202302	2023	2023.1342	TUR	
3	1085569540	20230219	202302	2023	2023.1342	TUR	
4	1085562596	20230218	202302	2023	2023.1315	TUR	
...	...	...	...	...	...	...	
17998	1083807068	20230207	202302	2023	2023.1014	USA	
17999	1083700839	20230207	202302	2023	2023.1014	MNCCANMED	
18000	1083700840	20230207	202302	2023	2023.1014	MNCCANMED	
18001	1083651693	20230206	202302	2023	2023.0986	NGOAGR	INTEF OLIV
18002	1083821637	20230207	202302	2023	2023.1014	IGOUNOHLHWHO	WORI ORG

18003 rows × 61 columns

In [9]:

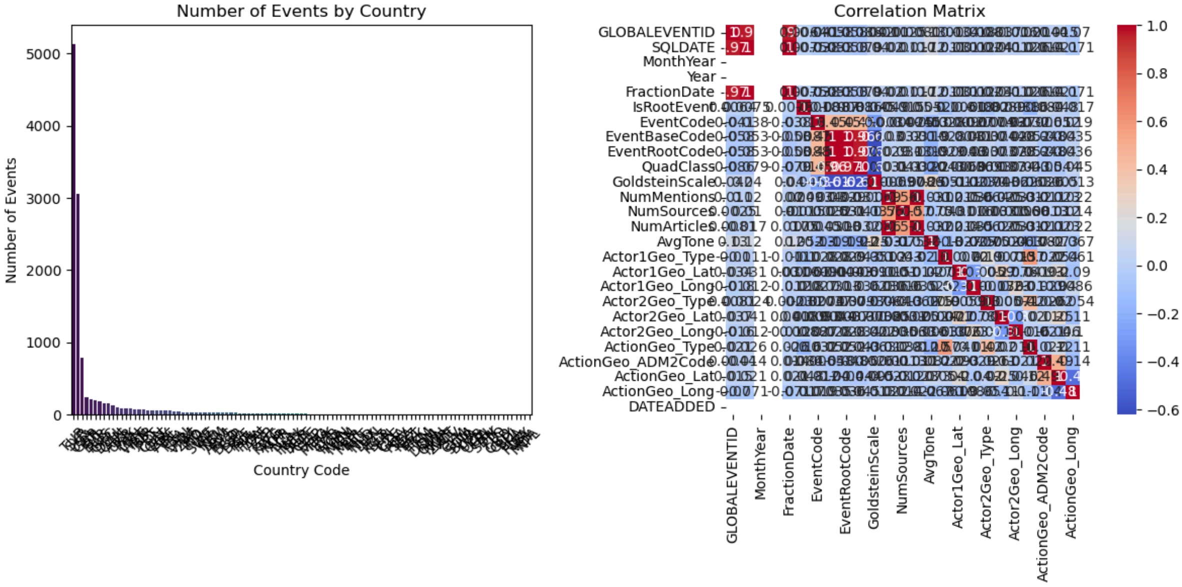
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# Check the first few rows and column names
print(df.head())
print(df.columns)

# Data preprocessing and analysis
# Example: Count the number of events by Actor1CountryCode
events_count = df['Actor1CountryCode'].value_counts()

# Visualization
plt.figure(figsize=(10, 6))
sns.barplot(x=events_count.index, y=events_count.values, palette='viridis')
plt.title('Number of Events by Country')
plt.xlabel('Country Code')
plt.ylabel('Number of Events')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
   GLOBALEVENTID   SQLDATE  MonthYear  Year  FractionDate Actor1Code  \
0     1085521809  20230218     202302  2023     2023.1315        BUS   
1     1085574586  20230219     202302  2023     2023.1342        SYR   
2     1085574598  20230219     202302  2023     2023.1342        TUR   
3     1085569540  20230219     202302  2023     2023.1342        TUR   
4     1085562596  20230218     202302  2023     2023.1315        TUR   

  Actor1Name Actor1CountryCode Actor1KnownGroupCode Actor1EthnicCode  ...  
\
0   INDUSTRY               NaN                  NaN              NaN  ...   
1      SYRIA               SYR                  NaN              NaN  ...   
2   ISTANBUL               TUR                  NaN              NaN  ...   
3     TURKEY               TUR                  NaN              NaN  ...   
4     TURKEY               TUR                  NaN              NaN  ...   

  ActionGeo_Type ActionGeo_FullName ActionGeo_CountryCode ActionGeo_ADM1Cod
e  \
0              1              Syria                    SY                 S
Y   
1              1              Syria                    SY                 S
Y   
2              1              Syria                    SY                 S
Y   
3              1              Syria                    SY                 S
Y   
4              1              Syria                    SY                 S
Y   

  ActionGeo_ADM2Code ActionGeo_Lat ActionGeo_Long ActionGeo_FeatureID  \
0                NaN          35.0           38.0                  SY   
1                NaN          35.0           38.0                  SY   
2                NaN          35.0           38.0                  SY   
3                NaN          35.0           38.0                  SY   
4                NaN          35.0           38.0                  SY   

      DATEADDED                                          SOURCEURL  
0  2.020000e+13  https://www.hampshirechronicle.co.uk/news/2332...  
1  2.020000e+13  https://www.dailyherald.com/news/20230218/port...  
2  2.020000e+13  https://www.dailyherald.com/news/20230218/port...  
3  2.020000e+13  https://www.mynbc5.com/article/turkey-syria-ea...  
4  2.020000e+13  https://www.wmtw.com/article/turkey-syria-eart...  

[5 rows x 61 columns]
Index(['GLOBALEVENTID', 'SQLDATE', 'MonthYear', 'Year', 'FractionDate',
       'Actor1Code', 'Actor1Name', 'Actor1CountryCode', 'Actor1KnownGroupCo
de',
       'Actor1EthnicCode', 'Actor1Religion1Code', 'Actor1Religion2Code',
       'Actor1Type1Code', 'Actor1Type2Code', 'Actor1Type3Code', 'Actor2Cod
e',
       'Actor2Name', 'Actor2CountryCode', 'Actor2KnownGroupCode',
       'Actor2EthnicCode', 'Actor2Religion1Code', 'Actor2Religion2Code',
       'Actor2Type1Code', 'Actor2Type2Code', 'Actor2Type3Code', 'IsRootEven
t',
       'EventCode', 'EventBaseCode', 'EventRootCode', 'QuadClass',
       'GoldsteinScale', 'NumMentions', 'NumSources', 'NumArticles', 'AvgTo
ne',
       'Actor1Geo_Type', 'Actor1Geo_FullName', 'Actor1Geo_CountryCode',
       'Actor1Geo_ADM1Code', 'Actor1Geo_ADM2Code', 'Actor1Geo_Lat',
       'Actor1Geo_Long', 'Actor1Geo_FeatureID', 'Actor2Geo_Type',
       'Actor2Geo_FullName', 'Actor2Geo_CountryCode', 'Actor2Geo_ADM1Code',
       'Actor2Geo_ADM2Code', 'Actor2Geo_Lat', 'Actor2Geo_Long',
       'Actor2Geo_FeatureID', 'ActionGeo_Type', 'ActionGeo_FullName',
       'ActionGeo_CountryCode', 'ActionGeo_ADM1Code', 'ActionGeo_ADM2Code',
       'ActionGeo_Lat', 'ActionGeo_Long', 'ActionGeo_FeatureID', 'DATEADDE
D',
       'SOURCEURL'],
      dtype='object')
```



In [12]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# Data preprocessing and analysis
# Example: Count the number of events by Actor1CountryCode
events_count = df['Actor1CountryCode'].value_counts()

# Example of mathematical modeling: Create a correlation matrix
correlation_matrix = df.corr()

# Example of understanding large group psychological response:
# Analyze the impact of events on public sentiment over time
df['SQLDATE'] = pd.to_datetime(df['SQLDATE'], format='%Y%m%d')
df.set_index('SQLDATE', inplace=True)
events_over_time = df.resample('M').size()

# Visualization
plt.figure(figsize=(12, 6))

# Plotting events count by country
plt.subplot(1, 2, 1)
sns.barplot(x=events_count.index, y=events_count.values, palette='viridis')
plt.title('Number of Events by Country')
plt.xlabel('Country Code')
plt.ylabel('Number of Events')
plt.xticks(rotation=45)

# Plotting correlation matrix
plt.subplot(1, 2, 2)
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')

plt.tight_layout()
plt.show()
```



In [15]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import pearsonr


# Data preprocessing and analysis
# Example: Count the number of events by Actor1CountryCode
events_count = df['Actor1CountryCode'].value_counts()

# Check for constant columns before calculating correlation
constant_columns = [col for col in df.columns if df[col].nunique() == 1]
if constant_columns:
    print(f"Constant columns found: {constant_columns}")
else:
    # Example of statistical analysis: Calculate correlation and p-value bet
    correlation, p_value = pearsonr(df['Year'], df['GLOBALEVENTID'])
    print(f"Correlation: {correlation:.2f}, P-Value: {p_value:.4f}")

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

    # Plotting events count by country
    plt.subplot(1, 2, 1)
    sns.barplot(x=events_count.index, y=events_count.values, palette='virid
    plt.title('Number of Events by Country')
    plt.xlabel('Country Code')
    plt.ylabel('Number of Events')
    plt.xticks(rotation=45)

    # Plotting Year vs Number of Events scatter plot
    plt.subplot(1, 2, 2)
    sns.scatterplot(x='Year', y='GLOBALEVENTID', data=df, alpha=0.5)
    plt.title('Year vs Number of Events')
    plt.xlabel('Year')
    plt.ylabel('Number of Events')

    # Adding correlation and p-value to the plot
    plt.text(2000, 100000, f'Correlation: {correlation:.2f}\nP-Value: {p_va

    plt.tight_layout()
    plt.show()
```

```
Constant columns found: ['MonthYear', 'Year', 'DATEADDED']
```

In [19]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import pearsonr


# Data preprocessing and analysis
# Example: Count the number of events by Actor1CountryCode
events_count = df['Actor1CountryCode'].value_counts()

# Exclude constant columns from the correlation analysis
constant_columns = ['MonthYear', 'Year', 'DATEADDED']
df_filtered = df.drop(columns=constant_columns)

# Check if 'Year' is in the filtered dataset before correlation analysis
if 'Year' in df_filtered.columns:
    # Example of statistical analysis: Calculate correlation and p-value bet
    correlation, p_value = pearsonr(df_filtered['Year'], df_filtered['GLOBAL
    print(f"Correlation: {correlation:.2f}, P-Value: {p_value:.4f}")

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

    # Plotting events count by country
    plt.subplot(1, 2, 1)
    sns.barplot(x=events_count.index, y=events_count.values, palette='virid
    plt.title('Number of Events by Country')
    plt.xlabel('Country Code')
    plt.ylabel('Number of Events')
    plt.xticks(rotation=45)

    # Plotting Year vs Number of Events scatter plot
    plt.subplot(1, 2, 2)
    sns.scatterplot(x='Year', y='GLOBALEVENTID', data=df_filtered, alpha=0.5
    plt.title('Year vs Number of Events')
    plt.xlabel('Year')
    plt.ylabel('Number of Events')

    # Adding correlation and p-value to the plot
    plt.text(2000, 100000, f'Correlation: {correlation:.2f}\nP-Value: {p_va

    plt.tight_layout()
    plt.show()
else:
    print("The 'Year' column is not present in the filtered dataset.")
```

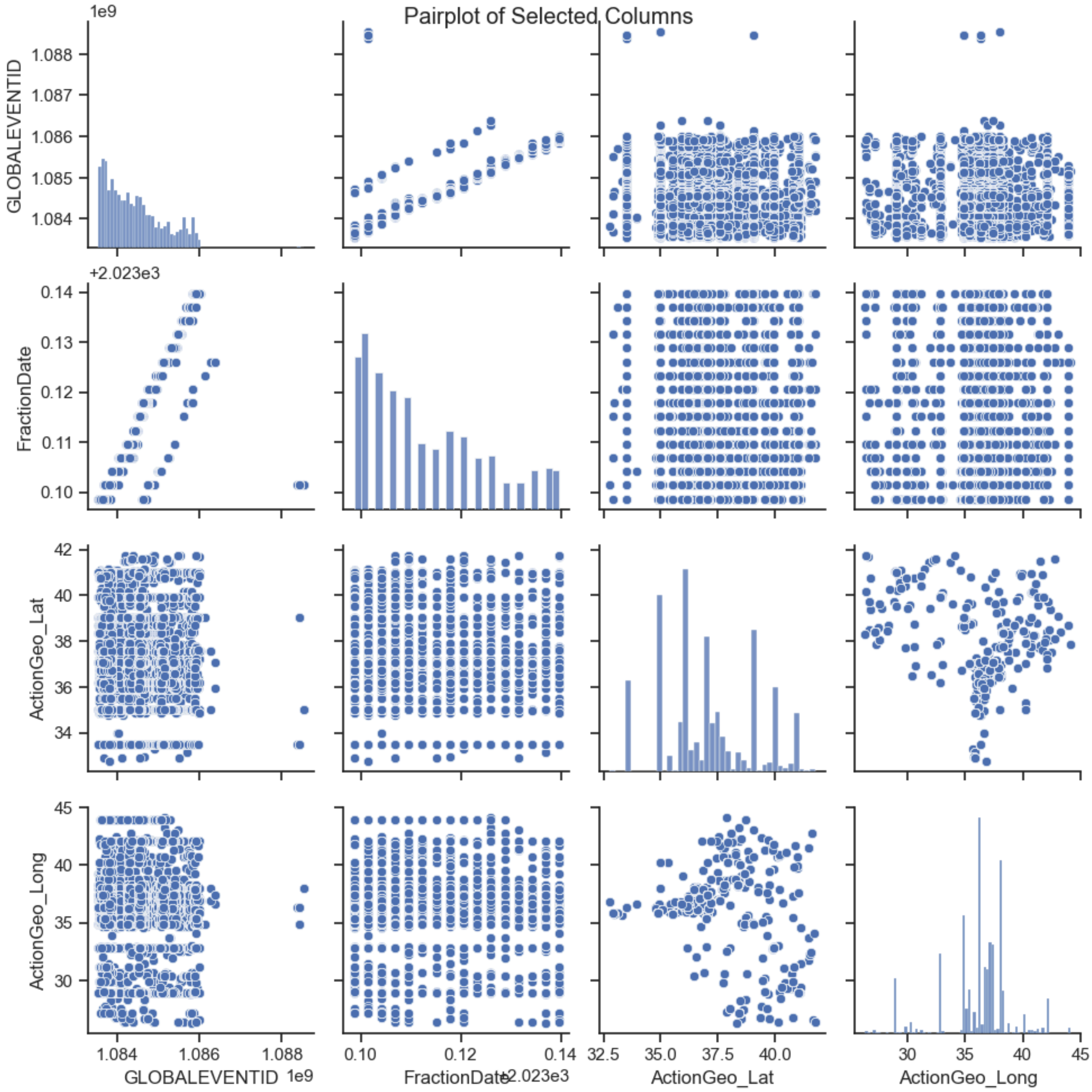
```
The 'Year' column is not present in the filtered dataset.
```

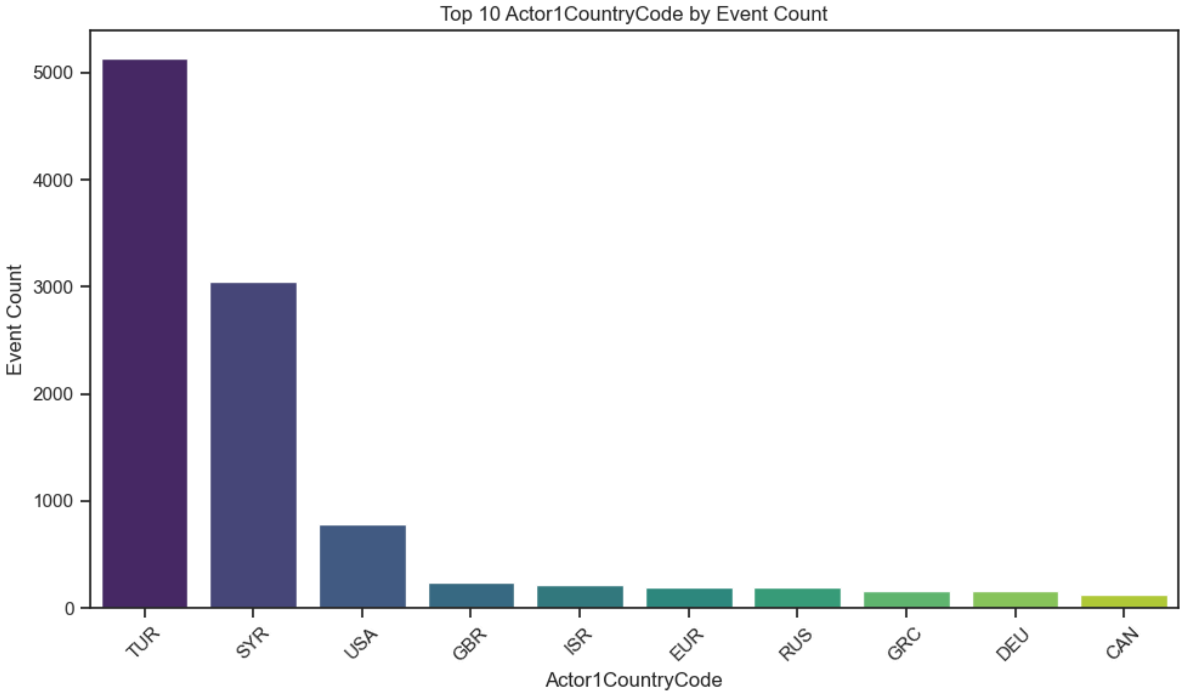
In [24]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import pearsonr


# Data preprocessing and analysis
# Exclude constant columns from the correlation analysis
constant_columns = ['MonthYear', 'DATEADDED']
df_filtered = df.drop(columns=constant_columns)

# Example of statistical analysis: Calculate correlation and p-value between
correlation, p_value = pearsonr(df_filtered['GLOBALEVENTID'], df_filtered['I
print(f"Correlation: {correlation:.2f}, P-Value: {p_value:.4f}")

# Visualization using scatter plot matrix and bar plot
```

```
sns.set(style="ticks")
sns.pairplot(df_filtered[['GLOBALEVENTID', 'FractionDate', 'Actor1Code', 'A
plt.suptitle('Pairplot of Selected Columns')
plt.show()

# Plotting events count by Actor1CountryCode
events_count = df['Actor1CountryCode'].value_counts().head(10)
plt.figure(figsize=(10, 6))
sns.barplot(x=events_count.index, y=events_count.values, palette='viridis')
plt.title('Top 10 Actor1CountryCode by Event Count')
plt.xlabel('Actor1CountryCode')
plt.ylabel('Event Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Correlation: 0.97, P-Value: 0.0000
```





In [29]:

```
df.columns
```

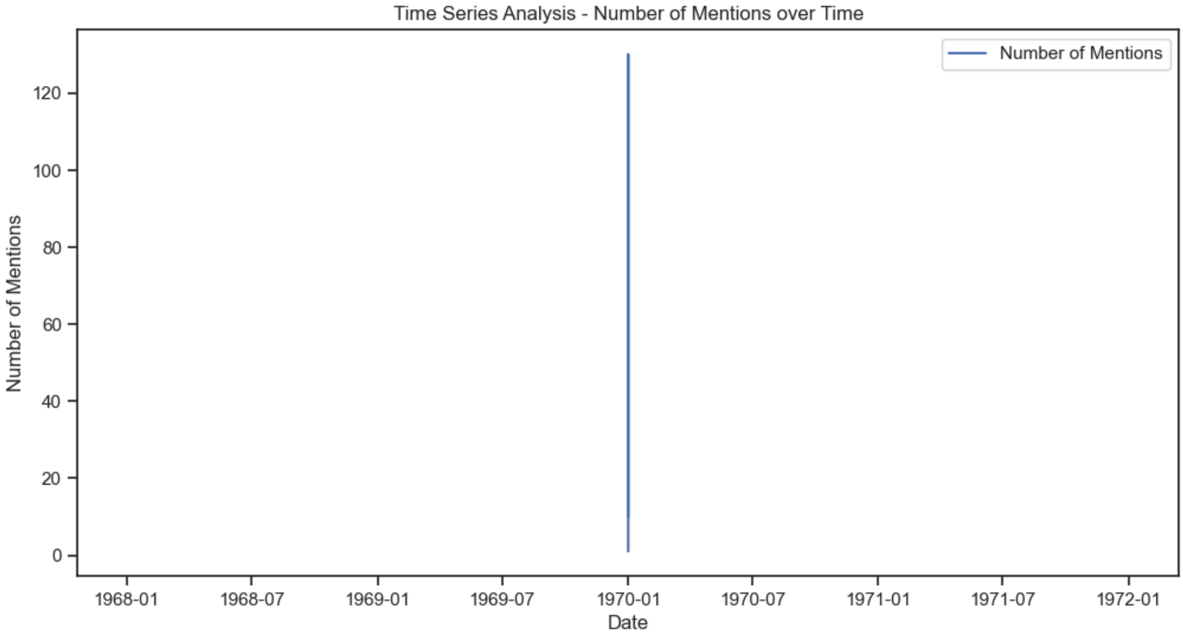
Out[29]:

```
Index(['GLOBALEVENTID', 'MonthYear', 'Year', 'FractionDate', 'Actor1Code',
       'Actor1Name', 'Actor1CountryCode', 'Actor1KnownGroupCode',
       'Actor1EthnicCode', 'Actor1Religion1Code', 'Actor1Religion2Code',
       'Actor1Type1Code', 'Actor1Type2Code', 'Actor1Type3Code', 'Actor2Cod
e',
       'Actor2Name', 'Actor2CountryCode', 'Actor2KnownGroupCode',
       'Actor2EthnicCode', 'Actor2Religion1Code', 'Actor2Religion2Code',
       'Actor2Type1Code', 'Actor2Type2Code', 'Actor2Type3Code', 'IsRootEven
t',
       'EventCode', 'EventBaseCode', 'EventRootCode', 'QuadClass',
       'GoldsteinScale', 'NumMentions', 'NumSources', 'NumArticles', 'AvgTo
ne',
       'Actor1Geo_Type', 'Actor1Geo_FullName', 'Actor1Geo_CountryCode',
       'Actor1Geo_ADM1Code', 'Actor1Geo_ADM2Code', 'Actor1Geo_Lat',
       'Actor1Geo_Long', 'Actor1Geo_FeatureID', 'Actor2Geo_Type',
       'Actor2Geo_FullName', 'Actor2Geo_CountryCode', 'Actor2Geo_ADM1Code',
       'Actor2Geo_ADM2Code', 'Actor2Geo_Lat', 'Actor2Geo_Long',
       'Actor2Geo_FeatureID', 'ActionGeo_Type', 'ActionGeo_FullName',
       'ActionGeo_CountryCode', 'ActionGeo_ADM1Code', 'ActionGeo_ADM2Code',
       'ActionGeo_Lat', 'ActionGeo_Long', 'ActionGeo_FeatureID', 'DATEADDE
D',
       'SOURCEURL'],
      dtype='object')
```

In [32]:

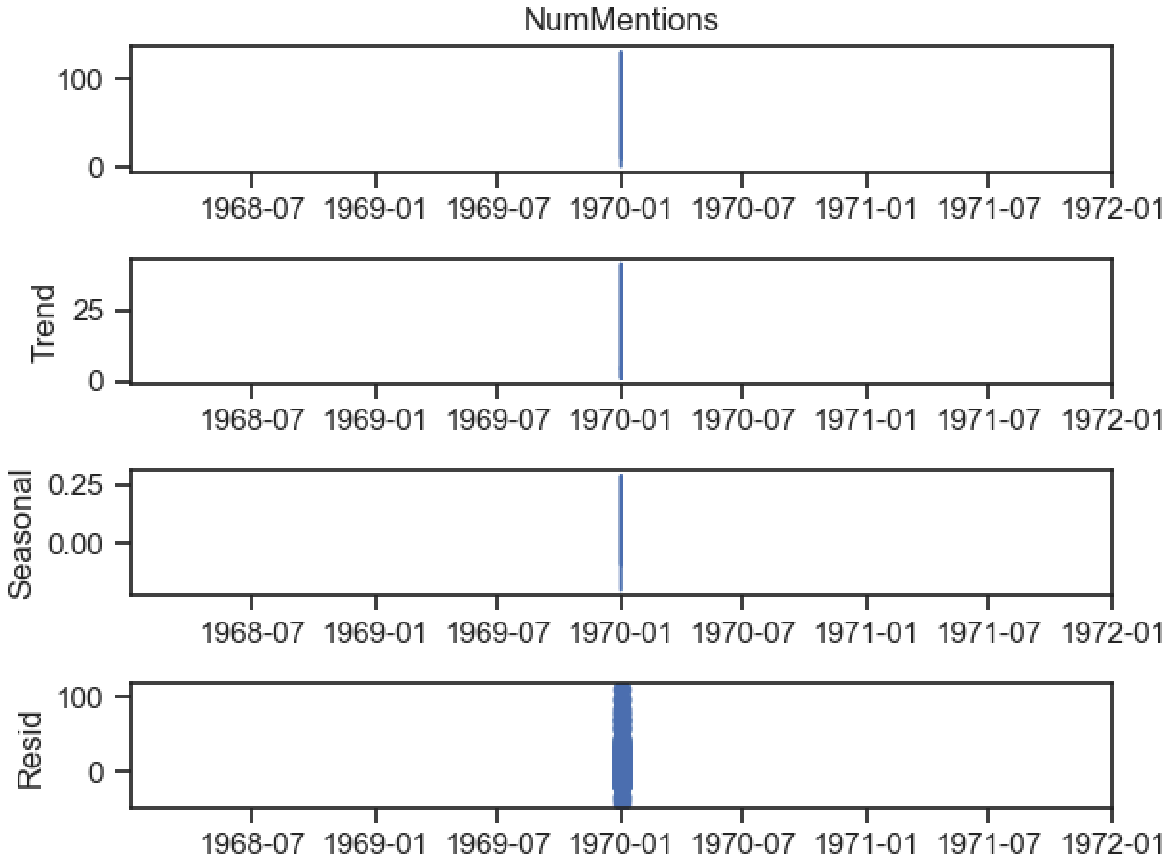
```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
from statsmodels.tsa.seasonal import seasonal_decompose


# Check the column names
print(df.columns)

# Time Series Analysis
# Assuming 'DATEADDED' is the time-related column
df['DATEADDED'] = pd.to_datetime(df['DATEADDED'])
df_time_series = df.set_index('DATEADDED')
plt.figure(figsize=(12, 6))
plt.plot(df_time_series['NumMentions'], label='Number of Mentions')
plt.xlabel('Date')
plt.ylabel('Number of Mentions')
plt.title('Time Series Analysis - Number of Mentions over Time')
plt.legend()
plt.show()

# Correlation Analysis
# Example: Calculate correlation and p-value between NumMentions and NumSou
correlation, p_value = pearsonr(df['NumMentions'], df['NumSources'])
print(f"Correlation: {correlation:.2f}, P-Value: {p_value:.4f}")

# Seasonal Decomposition
result = seasonal_decompose(df_time_series['NumMentions'], model='additive'
result.plot()
plt.show()
```

```
Index(['GLOBALEVENTID', 'MonthYear', 'Year', 'FractionDate', 'Actor1Code',
       'Actor1Name', 'Actor1CountryCode', 'Actor1KnownGroupCode',
       'Actor1EthnicCode', 'Actor1Religion1Code', 'Actor1Religion2Code',
       'Actor1Type1Code', 'Actor1Type2Code', 'Actor1Type3Code', 'Actor2Cod
e',
       'Actor2Name', 'Actor2CountryCode', 'Actor2KnownGroupCode',
       'Actor2EthnicCode', 'Actor2Religion1Code', 'Actor2Religion2Code',
       'Actor2Type1Code', 'Actor2Type2Code', 'Actor2Type3Code', 'IsRootEven
t',
       'EventCode', 'EventBaseCode', 'EventRootCode', 'QuadClass',
       'GoldsteinScale', 'NumMentions', 'NumSources', 'NumArticles', 'AvgTo
ne',
       'Actor1Geo_Type', 'Actor1Geo_FullName', 'Actor1Geo_CountryCode',
       'Actor1Geo_ADM1Code', 'Actor1Geo_ADM2Code', 'Actor1Geo_Lat',
       'Actor1Geo_Long', 'Actor1Geo_FeatureID', 'Actor2Geo_Type',
       'Actor2Geo_FullName', 'Actor2Geo_CountryCode', 'Actor2Geo_ADM1Code',
       'Actor2Geo_ADM2Code', 'Actor2Geo_Lat', 'Actor2Geo_Long',
       'Actor2Geo_FeatureID', 'ActionGeo_Type', 'ActionGeo_FullName',
       'ActionGeo_CountryCode', 'ActionGeo_ADM1Code', 'ActionGeo_ADM2Code',
       'ActionGeo_Lat', 'ActionGeo_Long', 'ActionGeo_FeatureID', 'DATEADDE
D',
       'SOURCEURL'],
      dtype='object')
```



```
Correlation: 0.56, P-Value: 0.0000
```

```
/opt/anaconda3/lib/python3.8/site-packages/statsmodels/tsa/seasonal.py:336:
UserWarning: Attempting to set identical left == right == 0.233796296296296
3 results in singular transformations; automatically expanding.
  ax.set_xlim(xlim)
/opt/anaconda3/lib/python3.8/site-packages/statsmodels/tsa/seasonal.py:336:
UserWarning: Attempting to set identical left == right == 0.233796296296296
3 results in singular transformations; automatically expanding.
  ax.set_xlim(xlim)
/opt/anaconda3/lib/python3.8/site-packages/statsmodels/tsa/seasonal.py:336:
UserWarning: Attempting to set identical left == right == 0.233796296296296
3 results in singular transformations; automatically expanding.
  ax.set_xlim(xlim)
/opt/anaconda3/lib/python3.8/site-packages/statsmodels/tsa/seasonal.py:336:
UserWarning: Attempting to set identical left == right == 0.233796296296296
3 results in singular transformations; automatically expanding.
  ax.set_xlim(xlim)
```



In [37]:

```
print(df.columns)
```

```
Index(['GLOBALEVENTID', 'MonthYear', 'Year', 'FractionDate', 'Actor1Code',
       'Actor1Name', 'Actor1CountryCode', 'Actor1KnownGroupCode',
       'Actor1EthnicCode', 'Actor1Religion1Code', 'Actor1Religion2Code',
       'Actor1Type1Code', 'Actor1Type2Code', 'Actor1Type3Code', 'Actor2Cod
e',
       'Actor2Name', 'Actor2CountryCode', 'Actor2KnownGroupCode',
       'Actor2EthnicCode', 'Actor2Religion1Code', 'Actor2Religion2Code',
       'Actor2Type1Code', 'Actor2Type2Code', 'Actor2Type3Code', 'IsRootEven
t',
       'EventCode', 'EventBaseCode', 'EventRootCode', 'QuadClass',
       'GoldsteinScale', 'NumMentions', 'NumSources', 'NumArticles', 'AvgTo
ne',
       'Actor1Geo_Type', 'Actor1Geo_FullName', 'Actor1Geo_CountryCode',
       'Actor1Geo_ADM1Code', 'Actor1Geo_ADM2Code', 'Actor1Geo_Lat',
       'Actor1Geo_Long', 'Actor1Geo_FeatureID', 'Actor2Geo_Type',
       'Actor2Geo_FullName', 'Actor2Geo_CountryCode', 'Actor2Geo_ADM1Code',
       'Actor2Geo_ADM2Code', 'Actor2Geo_Lat', 'Actor2Geo_Long',
       'Actor2Geo_FeatureID', 'ActionGeo_Type', 'ActionGeo_FullName',
       'ActionGeo_CountryCode', 'ActionGeo_ADM1Code', 'ActionGeo_ADM2Code',
       'ActionGeo_Lat', 'ActionGeo_Long', 'ActionGeo_FeatureID', 'DATEADDE
D',
       'SOURCEURL'],
      dtype='object')
```

In [ ]: