Clustering Algorithms for Predicting Politically Unstable Countries

Creating Multiple DataSets according to the year

In [1]:
```
import pandas as pd
```

In [2]:
```
df2006 = pd.read_excel('fsi-2006.xlsx', engine='openpyxl')
df2007 = pd.read_excel('fsi-2007.xlsx', engine='openpyxl')
df2008 = pd.read_excel('fsi-2008.xlsx', engine='openpyxl')
df2009 = pd.read_excel('fsi-2009.xlsx', engine='openpyxl')
df2010 = pd.read_excel('fsi-2010.xlsx', engine='openpyxl')
df2011 = pd.read_excel('fsi-2011.xlsx', engine='openpyxl')
df2012 = pd.read_excel('fsi-2012.xlsx', engine='openpyxl')
df2013 = pd.read_excel('fsi-2013.xlsx', engine='openpyxl')
df2014 = pd.read_excel('fsi-2014.xlsx', engine='openpyxl')
df2015 = pd.read_excel('fsi-2015.xlsx', engine='openpyxl')
df2016 = pd.read_excel('fsi-2016.xlsx', engine='openpyxl')
df2017 = pd.read_excel('fsi-2017.xlsx', engine='openpyxl')
df2018 = pd.read_excel('fsi-2018.xlsx', engine='openpyxl')
df2019 = pd.read_excel('fsi-2019.xlsx', engine='openpyxl')
df2020 = pd.read_excel('fsi-2020.xlsx', engine='openpyxl')
```

Removing unwanted columns (cleansing step)

In [3]:
```
del df2019['Change from Previous Year']
del df2019['Unnamed: 17']
del df2019['Unnamed: 18']
del df2020['Change from Previous Year']
del df2020['Unnamed: 17']
del df2020['Unnamed: 18']
del df2020['Unnamed: 19']
del df2020['Unnamed: 20']
del df2020['Unnamed: 21']
del df2020['Unnamed: 22']
del df2020['Unnamed: 23']
del df2020['Unnamed: 24']
del df2020['Unnamed: 25']
del df2020['Unnamed: 26']
del df2020['Unnamed: 27']
del df2020['Unnamed: 28']
del df2020['Unnamed: 29']
del df2020['Unnamed: 30']
del df2020['Unnamed: 31']
del df2020['Unnamed: 32']
del df2020['Unnamed: 33']
del df2020['Unnamed: 34']
del df2020['Unnamed: 35']
del df2020['Unnamed: 36']
del df2020['Unnamed: 37']
del df2020['Unnamed: 38']
```

In [4]:
```
df2006.dropna(inplace = True)
df2007.dropna(inplace = True)
df2008.dropna(inplace = True)
df2009.dropna(inplace = True)
df2010.dropna(inplace = True)
df2011.dropna(inplace = True)
df2012.dropna(inplace = True)
df2013.dropna(inplace = True)
df2014.dropna(inplace = True)
df2015.dropna(inplace = True)
df2016.dropna(inplace = True)
df2017.dropna(inplace = True)
df2018.dropna(inplace = True)
df2019.dropna(inplace = True)
df2020.dropna(inplace = True)
```

In [5]:
```
df2006.tail()
```

Out[5]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
141	Switzerland	2006-01-01	142nd	18.7	1.0	1.0	2.0	1.2	2.5	2.0
142	Ireland	2006-01-01	143rd	18.6	1.0	1.0	1.0	1.9	3.0	2.0
143	Finland	2006-01-01	144th	18.2	1.0	1.0	1.0	2.2	2.0	2.0
144	Sweden	2006-01-01	144th	18.2	1.0	1.0	1.0	1.2	2.0	2.0
145	Norway	2006-01-01	146th	16.8	1.0	1.0	1.0	1.8	2.0	1.0

In [6]:
```
df2007.tail()
```

Out[6]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
172	Switzerland	2007-01-01	172nd	20.2	1.0	1.0	2.1	1.5	2.6	2.0
173	Ireland	2007-01-01	173rd	19.5	1.0	1.0	1.0	2.1	2.9	2.1
174	Sweden	2007-01-01	174th	19.3	0.9	1.0	1.0	1.3	2.0	2.0
175	Finland	2007-01-01	175th	18.5	0.9	0.7	1.0	2.2	1.9	2.1
176	Norway	2007-01-01	176th	17.1	1.0	1.0	1.0	2.1	2.0	1.1

In [7]:
```
df2008.tail()
```

Out[7]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
172	Switzerland	2008-01-01	173rd	20.3	1.0	1.0	2.6	1.5	2.6	2.0
173	Ireland	2008-01-01	174th	19.9	1.0	1.0	1.0	2.0	3.0	2.0
174	Sweden	2008-01-01	175th	19.8	0.9	1.0	1.3	1.2	2.1	2.0
175	Finland	2008-01-01	176th	18.4	0.9	0.7	1.0	2.0	1.9	2.1
176	Norway	2008-01-01	177th	16.8	1.0	1.0	1.0	1.8	2.0	1.1

In [8]:
```
df2009.tail()
```

Out[8]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
172	Ireland	2009-01-01	173rd	21.6	1.1	1.3	1.0	2.7	3.0	2.0
173	Switzerland	2009-01-01	174th	21.2	1.0	1.0	2.9	2.1	2.6	2.0
174	Sweden	2009-01-01	175th	20.6	1.1	1.3	1.3	1.6	2.3	2.0
175	Finland	2009-01-01	176th	19.2	0.9	0.9	1.2	2.4	1.9	2.1
176	Norway	2009-01-01	177th	18.3	1.1	1.1	1.3	2.3	2.2	1.1

In [9]:
```
df2010.tail()
```

Out[9]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
172	Ireland	2010-01-01	173rd	22.4	1.4	1.5	1.0	3.3	2.8	2.0
173	Switzerland	2010-01-01	174th	21.8	1.2	1.0	3.3	2.4	2.6	1.8
174	Sweden	2010-01-01	175th	20.9	1.3	1.3	1.3	2.2	2.1	1.8
175	Finland	2010-01-01	176th	19.3	1.0	1.0	1.2	3.0	1.7	2.2
176	Norway	2010-01-01	177th	18.7	1.2	1.1	1.3	2.6	2.4	1.2

In [10]:
```
df2011.tail()
```

Out[10]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
172	Denmark	2011-01-01	173rd	23.8	1.5	1.0	3.3	2.5	1.7	2.1
173	Switzerland	2011-01-01	174th	23.2	1.4	1.0	3.5	2.4	2.8	2.1
174	Sweden	2011-01-01	175th	22.8	2.3	1.8	1.3	1.9	2.2	2.0
175	Norway	2011-01-01	176th	20.4	1.2	1.2	1.3	2.9	2.1	1.5
176	Finland	2011-01-01	177th	19.7	1.0	1.2	1.7	2.8	1.3	2.5

In [11]:
```
df2012.tail()
```

Out[11]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
173	Norway	2012-01-01	173rd	23.9	3.0	1.2	3.6	2.4	1.8	1.5
174	Switzerland	2012-01-01	174th	23.3	1.7	1.0	3.2	2.2	2.6	2.2
175	Denmark	2012-01-01	175th	23.0	1.8	1.0	3.0	2.2	1.8	2.2
176	Sweden	2012-01-01	176th	21.3	2.5	1.8	1.0	1.6	1.9	1.8
177	Finland	2012-01-01	177th	20.0	1.3	1.2	1.4	2.9	1.3	2.6

In [12]:
```
df2013.tail()
```

Out[12]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
173	Denmark	2013-01-01	174th	21.9	1.5	1.4	3.4	1.9	1.6	1.9
174	Norway	2013-01-01	175th	21.5	2.7	1.1	3.6	1.9	1.5	1.6
175	Switzerland	2013-01-01	175th	21.5	1.4	1.0	3.5	2.3	2.3	2.1
176	Sweden	2013-01-01	177th	19.7	2.2	1.8	1.0	1.7	1.7	1.7
177	Finland	2013-01-01	178th	18.0	1.0	1.1	1.4	3.2	1.0	2.3

In [13]:
```
df2014.tail()
```

Out[13]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
173	Switzerland	2014-01-01	174th	23.3	1.7	1.0	3.5	2.2	2.4	2.4
174	Norway	2014-01-01	175th	23.0	2.8	1.1	3.7	2.0	1.7	1.9
175	Denmark	2014-01-01	176th	22.8	1.8	1.4	3.4	2.2	1.8	2.0
176	Sweden	2014-01-01	177th	21.4	2.4	1.8	1.0	2.0	1.8	1.8
177	Finland	2014-01-01	178th	18.7	1.3	1.1	1.3	3.5	1.3	2.2

In [14]:
```
df2015.tail()
```

Out[14]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
173	Luxembourg	2015-01-01	174th	22.1	2.0	3.4	3.1	1.5	1.5	2.1
174	Denmark	2015-01-01	175th	21.4	1.5	1.4	3.6	2.5	2.1	1.9
175	Norway	2015-01-01	176th	20.9	2.5	1.1	3.7	1.7	2.0	1.6
176	Sweden	2015-01-01	177th	20.2	2.1	1.8	1.3	2.3	1.8	1.5
177	Finland	2015-01-01	178th	17.7	1.4	1.1	1.6	3.8	1.0	2.3

In [15]:
```
df2016.tail()
```

Out[15]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
173	Switzerland	2016-01-01	174th	21.8	1.3	1.0	3.8	2.5	2.4	1.8
174	Denmark	2016-01-01	175th	21.5	1.4	1.4	4.6	2.2	1.8	1.6
175	New Zealand	2016-01-01	176th	21.3	1.4	1.1	3.8	3.8	2.5	1.8
176	Norway	2016-01-01	177th	21.2	2.2	1.1	3.8	1.7	1.7	1.3
177	Finland	2016-01-01	178th	18.8	1.4	1.1	2.0	3.7	1.2	2.0

In [16]:
```
df2017.tail()
```

Out[16]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
173	Sweden	2017-01-01	174th	22.1	2.1	1.8	1.5	1.8	1.8	1.5
174	Denmark	2017-01-01	175th	21.5	1.7	1.4	4.4	2.0	1.6	1.9
175	Switzerland	2017-01-01	176th	21.1	1.1	1.0	3.6	2.3	2.2	2.1
176	Norway	2017-01-01	177th	20.5	2.0	1.1	3.6	2.2	1.5	1.6
177	Finland	2017-01-01	178th	18.7	1.7	1.1	1.8	3.5	1.0	2.3

In [17]:
```
df2018.tail()
```

Out[17]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
173	Iceland	2018-01-01	174th	20.300000	0.8	1.8	1.1	3.20000	0.9	2.600000
174	Denmark	2018-01-01	175th	19.785177	1.4	1.4	4.4	1.73386	1.3	1.972119
175	Switzerland	2018-01-01	176th	19.209526	1.4	1.0	3.6	2.00000	1.9	1.800000
176	Norway	2018-01-01	177th	18.258722	1.8	1.1	3.4	2.00000	1.0	1.400000
177	Finland	2018-01-01	178th	17.934252	2.2	1.4	1.5	3.20000	0.7	2.341068

In [18]:
```
df2019.tail()
```

Out[18]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
173	Australia	2019-01-01	174th	19.7	2.7	1.7	3.3	1.6	1.6	1.0
174	Denmark	2019-01-01	175th	19.5	1.3	1.4	4.3	1.6	1.2	1.9
175	Switzerland	2019-01-01	176th	18.7	1.1	1.0	3.3	1.9	1.8	1.7
176	Norway	2019-01-01	177th	18.0	2.1	1.1	3.3	1.9	1.0	1.3
177	Finland	2019-01-01	178th	16.9	2.5	1.4	1.2	2.9	0.7	2.0

In [19]:
```
df2020.tail()
```

Out[19]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
173	Iceland	2020-01-01	174th	17.800000	1.0	1.8	0.7	2.80000	1.0	2.2
174	Denmark	2020-01-01	175th	17.213587	1.6	1.4	4.0	1.30000	0.9	1.6
175	Switzerland	2020-01-01	176th	17.094086	1.4	1.0	3.0	1.60000	1.5	1.4
176	Norway	2020-01-01	177th	16.191210	1.8	1.1	3.6	1.69121	0.7	1.0
177	Finland	2020-01-01	178th	14.626666	2.8	1.4	0.9	2.60000	0.5	1.7

Concatenating the Data

In [20]:
```
dataframes = [df2006, df2007, df2008, df2009, df2010, df2011, df2012, df2013, df2014,
              df2017, df2018, df2019, df2020]
```

In [21]:
```
fsi6_20_df = pd.concat(dataframes)
```

In [22]:
```
fsi6_20_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2633 entries, 0 to 177
Data columns (total 16 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Country                           2633 non-null   object
 1   Year                              2633 non-null   datetime64[ns]
 2   Rank                              2633 non-null   object
 3   Total                             2633 non-null   float64
 4   C1: Security Apparatus            2633 non-null   float64
 5   C2: Factionalized Elites          2633 non-null   float64
 6   C3: Group Grievance               2633 non-null   float64
 7   E1: Economy                       2633 non-null   float64
 8   E2: Economic Inequality           2633 non-null   float64
 9   E3: Human Flight and Brain Drain  2633 non-null   float64
 10  P1: State Legitimacy              2633 non-null   float64
 11  P2: Public Services               2633 non-null   float64
 12  P3: Human Rights                  2633 non-null   float64
 13  S1: Demographic Pressures         2633 non-null   float64
 14  S2: Refugees and IDPs             2633 non-null   float64
 15  X1: External Intervention         2633 non-null   float64
dtypes: datetime64[ns](1), float64(13), object(2)
memory usage: 349.7+ KB
```

In [24]:
```
fsi6_20_df[fsi6_20_df.Country.eq('India')]
```

Out[24]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
92	India	2006-01-01	93rd	70.400000	4.500000	5.7	6.9	5.000000	8.5	7.10000
109	India	2007-01-01	109th	70.800000	5.000000	5.6	7.0	4.600000	8.9	7.10000
97	India	2008-01-01	98th	72.900000	6.600000	6.0	7.0	4.600000	8.9	6.90000
86	India	2009-01-01	87th	77.800000	7.100000	6.0	7.3	5.000000	8.9	6.70000
78	India	2010-01-01	79th	79.200000	7.600000	6.2	7.8	5.100000	8.7	6.50000
75	India	2011-01-01	76th	79.300000	7.800000	6.8	8.2	5.400000	8.5	6.20000
78	India	2012-01-01	78th	78.000000	7.500000	6.8	7.9	5.500000	8.4	5.90000
78	India	2013-01-01	79th	77.500000	7.800000	6.8	8.2	5.400000	8.1	5.40000
80	India	2014-01-01	81st	76.900000	7.900000	6.8	7.8	5.700000	7.8	5.20000
68	India	2015-01-01	69th	79.300000	7.900000	7.3	8.3	5.600000	7.5	5.50000
69	India	2016-01-01	70th	79.600000	7.600000	7.3	8.5	5.300000	7.2	5.80000
71	India	2017-01-01	72nd	77.900000	7.400000	7.3	8.3	5.100000	7.0	6.10000
71	India	2018-01-01	72nd	76.308510	7.100000	7.3	8.3	5.021057	6.7	6.39822
73	India	2019-01-01	74th	74.400000	7.200000	7.3	8.0	5.300000	6.4	6.10000
67	India	2020-01-01	68th	75.307433	6.907433	7.3	8.5	5.600000	6.1	5.80000

In [25]:
```
# Basic Data cleaning checks
fsi6_20_df.isna().sum()
```

Out[25]:
```
Country                             0
Year                                0
Rank                                0
Total                               0
C1: Security Apparatus              0
C2: Factionalized Elites            0
C3: Group Grievance                 0
E1: Economy                         0
E2: Economic Inequality             0
E3: Human Flight and Brain Drain    0
P1: State Legitimacy                0
P2: Public Services                 0
P3: Human Rights                    0
S1: Demographic Pressures           0
S2: Refugees and IDPs               0
X1: External Intervention           0
dtype: int64
```

In [26]:
```
fsi6_20_df.info()
```

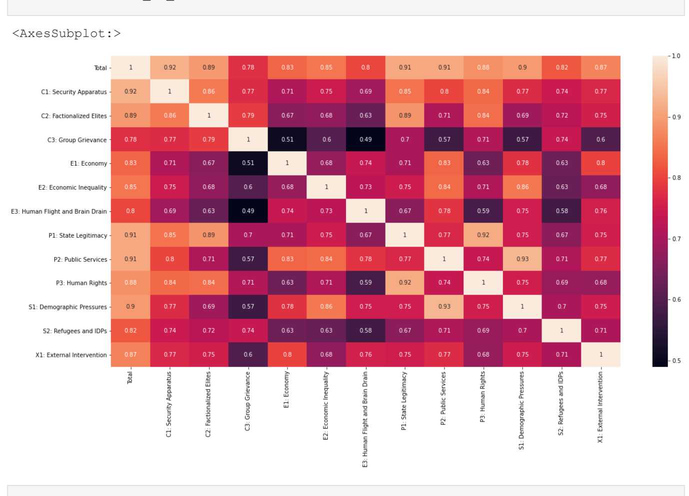
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2633 entries, 0 to 177
Data columns (total 16 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Country                           2633 non-null   object
 1   Year                              2633 non-null   datetime64[ns]
 2   Rank                              2633 non-null   object
 3   Total                             2633 non-null   float64
 4   C1: Security Apparatus            2633 non-null   float64
 5   C2: Factionalized Elites          2633 non-null   float64
 6   C3: Group Grievance               2633 non-null   float64
 7   E1: Economy                       2633 non-null   float64
 8   E2: Economic Inequality           2633 non-null   float64
 9   E3: Human Flight and Brain Drain  2633 non-null   float64
 10  P1: State Legitimacy              2633 non-null   float64
 11  P2: Public Services               2633 non-null   float64
 12  P3: Human Rights                  2633 non-null   float64
 13  S1: Demographic Pressures         2633 non-null   float64
 14  S2: Refugees and IDPs             2633 non-null   float64
 15  X1: External Intervention         2633 non-null   float64
dtypes: datetime64[ns](1), float64(13), object(2)
memory usage: 349.7+ KB
```

In [27]:
```
# For Visualisation
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Visualization (Heat Map)

In [28]:
```
plt.figure(figsize = (20,10))
sns.heatmap(fsi6_20_df.corr(),annot = True)
```

Out[28]:
```
<AxesSubplot:>
```



In [29]:
```
# Preparing Data for making country cluster
fsi6_20_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2633 entries, 0 to 177
Data columns (total 16 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Country                           2633 non-null   object
 1   Year                              2633 non-null   datetime64[ns]
 2   Rank                              2633 non-null   object
 3   Total                             2633 non-null   float64
 4   C1: Security Apparatus            2633 non-null   float64
 5   C2: Factionalized Elites          2633 non-null   float64
 6   C3: Group Grievance               2633 non-null   float64
 7   E1: Economy                       2633 non-null   float64
 8   E2: Economic Inequality           2633 non-null   float64
 9   E3: Human Flight and Brain Drain  2633 non-null   float64
 10  P1: State Legitimacy              2633 non-null   float64
 11  P2: Public Services               2633 non-null   float64
 12  P3: Human Rights                  2633 non-null   float64
 13  S1: Demographic Pressures         2633 non-null   float64
 14  S2: Refugees and IDPs             2633 non-null   float64
 15  X1: External Intervention         2633 non-null   float64
dtypes: datetime64[ns](1), float64(13), object(2)
memory usage: 349.7+ KB
```

In [30]:
```
df_cluster = fsi6_20_df.copy()
```

In [31]:
```
df_cluster.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2633 entries, 0 to 177
Data columns (total 16 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Country                           2633 non-null   object
 1   Year                              2633 non-null   datetime64[ns]
 2   Rank                              2633 non-null   object
 3   Total                             2633 non-null   float64
 4   C1: Security Apparatus            2633 non-null   float64
 5   C2: Factionalized Elites          2633 non-null   float64
 6   C3: Group Grievance               2633 non-null   float64
 7   E1: Economy                       2633 non-null   float64
 8   E2: Economic Inequality           2633 non-null   float64
 9   E3: Human Flight and Brain Drain  2633 non-null   float64
 10  P1: State Legitimacy              2633 non-null   float64
 11  P2: Public Services               2633 non-null   float64
 12  P3: Human Rights                  2633 non-null   float64
 13  S1: Demographic Pressures         2633 non-null   float64
 14  S2: Refugees and IDPs             2633 non-null   float64
 15  X1: External Intervention         2633 non-null   float64
dtypes: datetime64[ns](1), float64(13), object(2)
memory usage: 349.7+ KB
```

In [32]:
```
df_cluster.shape
```

Out[32]:
```
(2633, 16)
```

In [33]:
```
# Lowering the country names for normalizing them
df_cluster['Country'] = df_cluster['Country'].str.lower()
```

In [34]:
```
df_cluster['Year'] = df_cluster['Year'].apply(str)
```

In [35]:
```
df_cluster.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2633 entries, 0 to 177
Data columns (total 16 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Country                           2633 non-null   object
 1   Year                              2633 non-null   object
 2   Rank                              2633 non-null   object
 3   Total                             2633 non-null   float64
 4   C1: Security Apparatus            2633 non-null   float64
 5   C2: Factionalized Elites          2633 non-null   float64
 6   C3: Group Grievance               2633 non-null   float64
 7   E1: Economy                       2633 non-null   float64
 8   E2: Economic Inequality           2633 non-null   float64
 9   E3: Human Flight and Brain Drain  2633 non-null   float64
 10  P1: State Legitimacy              2633 non-null   float64
 11  P2: Public Services               2633 non-null   float64
 12  P3: Human Rights                  2633 non-null   float64
 13  S1: Demographic Pressures         2633 non-null   float64
 14  S2: Refugees and IDPs             2633 non-null   float64
 15  X1: External Intervention         2633 non-null   float64
dtypes: float64(13), object(3)
memory usage: 349.7+ KB
```

In [36]:
```
df_cluster.shape
```

Out[36]:
```
(2633, 16)
```

Now we are going to join the Country and year column to make a unique country name to do our further analysis

In [38]:
```
df_cluster['Country_Year']= df_cluster['Country'].str.cat(df_cluster['Year'], sep ="_
```

In [39]:
```
df_cluster.head()
```

Out[39]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
0	sudan	2006-01-01 00:00:00	1st	112.3	9.8	9.1	9.7	7.5	9.2	9.1
1	congo democratic republic	2006-01-01 00:00:00	2nd	110.1	9.8	9.6	9.1	8.1	9.0	8.0
2	cote d'ivoire	2006-01-01 00:00:00	3rd	109.2	9.8	9.8	9.8	9.0	8.0	8.5
3	iraq	2006-01-01 00:00:00	4th	109.0	9.8	9.7	9.8	8.2	8.7	9.1
4	zimbabwe	2006-01-01 00:00:00	5th	108.9	9.4	8.5	8.5	9.8	9.2	9.0

In [41]:
```
df_cluster.tail()
```

Out[41]:

	Country	Year	Rank	Total	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain
173	iceland	2020-01-01 00:00:00	174th	17.800000	1.0	1.8	0.7	2.80000	1.0	2.2
174	denmark	2020-01-01 00:00:00	175th	17.213587	1.6	1.4	4.0	1.30000	0.9	1.6
175	switzerland	2020-01-01 00:00:00	176th	17.094086	1.4	1.0	3.0	1.60000	1.5	1.4
176	norway	2020-01-01 00:00:00	177th	16.191210	1.8	1.1	3.6	1.69121	0.7	1.0
177	finland	2020-01-01 00:00:00	178th	14.626666	2.8	1.4	0.9	2.60000	0.5	1.7

Deleting the unwanted columns

In [42]:
```
del df_cluster['Country']
```

In [43]:
```
del df_cluster['Year']
```

In [44]:
```
del df_cluster['Rank']
```

In [45]:
```
del df_cluster['Total']
```

In [46]:
```
df_cluster.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2633 entries, 0 to 177
Data columns (total 13 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   C1: Security Apparatus            2633 non-null   float64
 1   C2: Factionalized Elites          2633 non-null   float64
 2   C3: Group Grievance               2633 non-null   float64
 3   E1: Economy                       2633 non-null   float64
 4   E2: Economic Inequality           2633 non-null   float64
 5   E3: Human Flight and Brain Drain  2633 non-null   float64
 6   P1: State Legitimacy              2633 non-null   float64
 7   P2: Public Services               2633 non-null   float64
 8   P3: Human Rights                  2633 non-null   float64
 9   S1: Demographic Pressures         2633 non-null   float64
 10  S2: Refugees and IDPs             2633 non-null   float64
 11  X1: External Intervention         2633 non-null   float64
 12  Country_Year                      2633 non-null   object
dtypes: float64(12), object(1)
memory usage: 288.0+ KB
```

In [47]:
```
df_cluster['Country_Year'] = df_cluster['Country_Year'].apply(lambda x: x.split('-')[
```

In [48]:
```
df_cluster.head()
```

Out[48]:

	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain	P1: State Legitimacy	P2: Public Services	P3: Human Rights
0	9.8	9.1	9.7	7.5	9.2	9.1	9.5	9.5	9.8
1	9.8	9.6	9.1	8.1	9.0	8.0	9.0	9.0	9.5
2	9.8	9.8	9.8	9.0	8.0	8.5	10.0	8.5	9.4
3	9.8	9.7	9.8	8.2	8.7	9.1	8.5	8.3	9.7
4	9.4	8.5	8.5	9.8	9.2	9.0	8.9	9.5	9.5

Scaling the Data

In [49]:
```
from sklearn.preprocessing import PowerTransformer
```

In [50]:
```
df_cluster.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2633 entries, 0 to 177
Data columns (total 13 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   C1: Security Apparatus            2633 non-null   float64
 1   C2: Factionalized Elites          2633 non-null   float64
 2   C3: Group Grievance               2633 non-null   float64
 3   E1: Economy                       2633 non-null   float64
 4   E2: Economic Inequality           2633 non-null   float64
 5   E3: Human Flight and Brain Drain  2633 non-null   float64
 6   P1: State Legitimacy              2633 non-null   float64
 7   P2: Public Services               2633 non-null   float64
 8   P3: Human Rights                  2633 non-null   float64
 9   S1: Demographic Pressures         2633 non-null   float64
 10  S2: Refugees and IDPs             2633 non-null   float64
 11  X1: External Intervention         2633 non-null   float64
 12  Country_Year                      2633 non-null   object
dtypes: float64(12), object(1)
memory usage: 288.0+ KB
```

In [51]:
```
col = df_cluster.select_dtypes(include=['float64']).columns
col
```

Out[51]:
```
Index(['C1: Security Apparatus', 'C2: Factionalized Elites',
       'C3: Group Grievance', 'E1: Economy', 'E2: Economic Inequality',
       'E3: Human Flight and Brain Drain', 'P1: State Legitimacy',
```

```
       'P2: Public Services', 'P3: Human Rights', 'S1: Demographic Pressures',
       'S2: Refugees and IDPs', 'X1: External Intervention'],
      dtype='object')
```

In [52]:
```python
scaler = PowerTransformer(method = 'box-cox')
df_cluster[col] = scaler.fit_transform(df_cluster[col])
```

In [53]:
```python
df_cluster.head()
```

Out[53]:

	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain	P1: State Legitimacy	P2: Public Services	P3: Human Rights	Demog Pre
0	1.805800	1.232558	1.800606	0.931980	1.646909	1.866460	1.448098	1.553094	1.723301	1.6
1	1.805800	1.483375	1.506146	1.253989	1.515454	1.234896	1.196108	1.356130	1.586870	1.5
2	1.805800	1.585064	1.849714	1.741162	0.883755	1.518950	1.705347	1.158779	1.541520	1.2
3	1.805800	1.534123	1.849714	1.307879	1.321432	1.866460	0.949555	1.079726	1.677761	1.2
4	1.623831	0.938196	1.212015	2.178072	1.646909	1.808051	1.146357	1.553094	1.586870	1.6

Clustering of Countries with respect to year

Hopkins Statistics calculation

In [55]:
```python
from sklearn.neighbors import NearestNeighbors
from random import sample
from numpy.random import uniform
import numpy as np
from math import isnan

def hopkins(X):
    d = X.shape[1]
    #d = len(vars) # columns
    n = len(X) # rows
    m = int(0.1 * n)
    nbrs = NearestNeighbors(n_neighbors=1).fit(X.values)

    rand_X = sample(range(0, n, 1), m)

    ujd = []
    wjd = []
    for j in range(0, m):
        u_dist, _ = nbrs.kneighbors(uniform(np.amin(X,axis=0),np.amax(X,axis=0),d).res
        ujd.append(u_dist[0][1])
        w_dist, _ = nbrs.kneighbors(X.iloc[rand_X[j]].values.reshape(1, -1), 2, return
        wjd.append(w_dist[0][1])

    H = sum(ujd) / (sum(ujd) + sum(wjd))
    if isnan(H):
        print(ujd, wjd)
        H = 0

    return H
```

In [56]:
```python
#Let's check the Hopkins measure
hopkins(df_cluster.drop(['Country_Year'],axis=1))
```

Out[56]: 0.8845462494801902

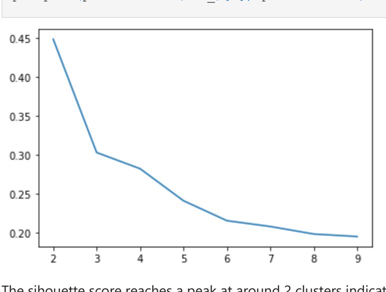
In [57]:
```python
df_cluster_var = df_cluster.drop(['Country_Year'],axis=1)
```

K-means clustering

Silhouette score

In [59]:
```python
#Let's check the silhouette score first to identify the ideal number of clusters
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
sse_ = []
for k in range(2, 10):
    kmeans = KMeans(n_clusters=k).fit(df_cluster_var)
    sse_.append([k, silhouette_score(df_cluster_var, kmeans.labels_)])
```

In [60]:
```python
plt.plot(pd.DataFrame(sse_)[0], pd.DataFrame(sse_)[1]);
```



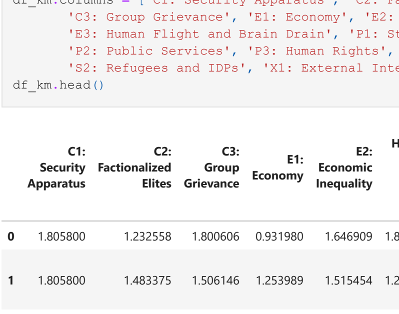
The sihouette score reaches a peak at around 2 clusters indicating that it might be the ideal number of clusters.

Elbow curve

In [61]:
```python
#Let's use the elbow curve method to identify the ideal number of clusters.
ssd = []
for num_clusters in list(range(1,10)):
    model_clus = KMeans(n_clusters = num_clusters, max_iter=50)
    model_clus.fit(df_cluster_var)
    ssd.append(model_clus.inertia_)

plt.plot(ssd)
```

Out[61]: [<matplotlib.lines.Line2D at 0x190668b4448>]



A distinct elbow is formed at around 2-4 clusters. We will go ahead with 2 clusters referring to stable and unstable countries.

K-Means Clustering

In [62]:
```python
#K-means with k=2 clusters
model_clus5 = KMeans(n_clusters = 2, max_iter=50)
model_clus5.fit(df_cluster_var)
```

Out[62]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=50,
       n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [63]:
```python
df_cluster.columns
```

Out[63]:
```
Index(['C1: Security Apparatus', 'C2: Factionalized Elites',
       'C3: Group Grievance', 'E1: Economy', 'E2: Economic Inequality',
       'E3: Human Flight and Brain Drain', 'P1: State Legitimacy',
       'P2: Public Services', 'P3: Human Rights', 'S1: Demographic Pressures',
       'S2: Refugees and IDPs', 'X1: External Intervention', 'Country_Year'],
      dtype='object')
```

In [64]:
```python
dat4 = df_cluster
dat4.index = pd.RangeIndex(len(dat4.index))
df_km = pd.concat([dat4, pd.Series(model_clus5.labels_)], axis=1)
df_km.columns = ['C1: Security Apparatus', 'C2: Factionalized Elites',
       'C3: Group Grievance', 'E1: Economy', 'E2: Economic Inequality',
       'E3: Human Flight and Brain Drain', 'P1: State Legitimacy',
       'P2: Public Services', 'P3: Human Rights', 'S1: Demographic Pressures',
       'S2: Refugees and IDPs', 'X1: External Intervention', 'Country_Year','ClusterII
df_km.head()
```

Out[64]:

	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain	P1: State Legitimacy	P2: Public Services	P3: Human Rights	Demog Pre
0	1.805800	1.232558	1.800606	0.931980	1.646909	1.866460	1.448098	1.553094	1.723301	1.6
1	1.805800	1.483375	1.506146	1.253989	1.515454	1.234896	1.196108	1.356130	1.586870	1.5
2	1.805800	1.585064	1.849714	1.741162	0.883755	1.518950	1.705347	1.158779	1.541520	1.2
3	1.805800	1.534123	1.849714	1.307879	1.321432	1.866460	0.949555	1.079726	1.677761	1.2
4	1.623831	0.938196	1.212015	2.178072	1.646909	1.808051	1.146357	1.553094	1.586870	1.6

In [65]:
```python
df_km['ClusterID'].value_counts()
```

Out[65]:
```
0    1805
1     828
Name: ClusterID, dtype: int64
```

In [66]:
```python
l = df_km.columns[0:-2]
l
```

Out[66]:
```
Index(['C1: Security Apparatus', 'C2: Factionalized Elites',
       'C3: Group Grievance', 'E1: Economy', 'E2: Economic Inequality',
       'E3: Human Flight and Brain Drain', 'P1: State Legitimacy',
       'P2: Public Services', 'P3: Human Rights', 'S1: Demographic Pressures',
       'S2: Refugees and IDPs', 'X1: External Intervention'],
      dtype='object')
```

In [67]:
```python
from itertools import combinations

def rSubset(lis, r):
    return list(combinations(lis, r))
```

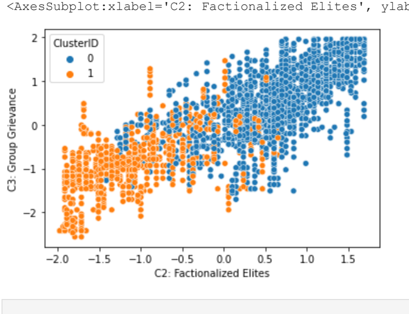
In [68]:
```python
l2 = rSubset(l, 2)
l2
```

Out[68]:
```
[('C1: Security Apparatus', 'C2: Factionalized Elites'),
 ('C1: Security Apparatus', 'C3: Group Grievance'),
 ('C1: Security Apparatus', 'E1: Economy'),
 ('C1: Security Apparatus', 'E2: Economic Inequality'),
 ('C1: Security Apparatus', 'E3: Human Flight and Brain Drain'),
 ('C1: Security Apparatus', 'P1: State Legitimacy'),
 ('C1: Security Apparatus', 'P2: Public Services'),
 ('C1: Security Apparatus', 'P3: Human Rights'),
 ('C1: Security Apparatus', 'S1: Demographic Pressures'),
 ('C1: Security Apparatus', 'S2: Refugees and IDPs'),
 ('C1: Security Apparatus', 'X1: External Intervention'),
 ('C2: Factionalized Elites', 'C3: Group Grievance'),
 ('C2: Factionalized Elites', 'E1: Economy'),
 ('C2: Factionalized Elites', 'E2: Economic Inequality'),
 ('C2: Factionalized Elites', 'E3: Human Flight and Brain Drain'),
 ('C2: Factionalized Elites', 'P1: State Legitimacy'),
 ('C2: Factionalized Elites', 'P2: Public Services'),
 ('C2: Factionalized Elites', 'P3: Human Rights'),
 ('C2: Factionalized Elites', 'S1: Demographic Pressures'),
 ('C2: Factionalized Elites', 'S2: Refugees and IDPs'),
 ('C2: Factionalized Elites', 'X1: External Intervention'),
 ('C3: Group Grievance', 'E1: Economy'),
 ('C3: Group Grievance', 'E2: Economic Inequality'),
 ('C3: Group Grievance', 'E3: Human Flight and Brain Drain'),
 ('C3: Group Grievance', 'P1: State Legitimacy'),
 ('C3: Group Grievance', 'P2: Public Services'),
 ('C3: Group Grievance', 'P3: Human Rights'),
 ('C3: Group Grievance', 'S1: Demographic Pressures'),
 ('C3: Group Grievance', 'S2: Refugees and IDPs'),
 ('C3: Group Grievance', 'X1: External Intervention'),
 ('E1: Economy', 'E2: Economic Inequality'),
 ('E1: Economy', 'E3: Human Flight and Brain Drain'),
 ('E1: Economy', 'P1: State Legitimacy'),
 ('E1: Economy', 'P2: Public Services'),
 ('E1: Economy', 'P3: Human Rights'),
 ('E1: Economy', 'S1: Demographic Pressures'),
 ('E1: Economy', 'S2: Refugees and IDPs'),
 ('E1: Economy', 'X1: External Intervention'),
 ('E2: Economic Inequality', 'E3: Human Flight and Brain Drain'),
 ('E2: Economic Inequality', 'P1: State Legitimacy'),
 ('E2: Economic Inequality', 'P2: Public Services'),
 ('E2: Economic Inequality', 'P3: Human Rights'),
 ('E2: Economic Inequality', 'S1: Demographic Pressures'),
 ('E2: Economic Inequality', 'S2: Refugees and IDPs'),
 ('E2: Economic Inequality', 'X1: External Intervention'),
 ('E3: Human Flight and Brain Drain', 'P1: State Legitimacy'),
 ('E3: Human Flight and Brain Drain', 'P2: Public Services'),
 ('E3: Human Flight and Brain Drain', 'P3: Human Rights'),
 ('E3: Human Flight and Brain Drain', 'S1: Demographic Pressures'),
 ('E3: Human Flight and Brain Drain', 'S2: Refugees and IDPs'),
 ('E3: Human Flight and Brain Drain', 'X1: External Intervention'),
 ('P1: State Legitimacy', 'P2: Public Services'),
 ('P1: State Legitimacy', 'P3: Human Rights'),
 ('P1: State Legitimacy', 'S1: Demographic Pressures'),
 ('P1: State Legitimacy', 'S2: Refugees and IDPs'),
 ('P1: State Legitimacy', 'X1: External Intervention'),
 ('P2: Public Services', 'P3: Human Rights'),
 ('P2: Public Services', 'S1: Demographic Pressures'),
 ('P2: Public Services', 'S2: Refugees and IDPs'),
 ('P2: Public Services', 'X1: External Intervention'),
 ('P3: Human Rights', 'S1: Demographic Pressures'),
 ('P3: Human Rights', 'S2: Refugees and IDPs'),
 ('P3: Human Rights', 'X1: External Intervention'),
 ('S1: Demographic Pressures', 'S2: Refugees and IDPs'),
 ('S1: Demographic Pressures', 'X1: External Intervention'),
 ('S2: Refugees and IDPs', 'X1: External Intervention')]
```

In [69]:
```python
# for i in range(0, len(l2)):
#     for couple in l2:
#         plt.figure(i)
#         sns.scatterplot(x=couple[0],y=couple[1],hue='ClusterID',legend = False, dat
```

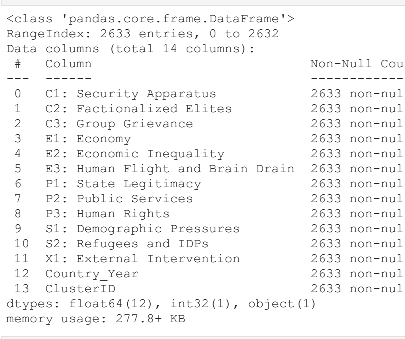
In [70]:
```python
#Let's do some further visualizations.
#We'll be visualising the clusters on the basis of the original columns.
sns.scatterplot(x='C1: Security Apparatus',y='C2: Factionalized Elites',hue='ClusterII
```

Out[70]: <AxesSubplot:xlabel='C1: Security Apparatus', ylabel='C2: Factionalized Elites'>



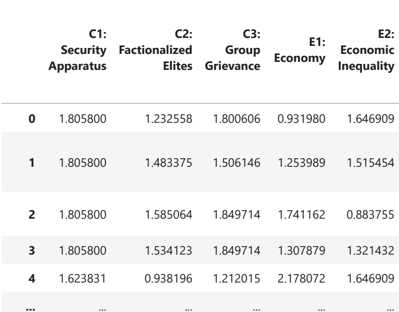
In [71]:
```python
sns.scatterplot(x='C1: Security Apparatus',y='C3: Group Grievance',hue='ClusterID',le
```

Out[71]: <AxesSubplot:xlabel='C1: Security Apparatus', ylabel='C3: Group Grievance'>



In [72]:
```python
# 'C2: Factionalized Elites', 'C3: Group Grievance'
sns.scatterplot(x='C2: Factionalized Elites',y='C3: Group Grievance',hue='ClusterID',
```

Out[72]: <AxesSubplot:xlabel='C2: Factionalized Elites', ylabel='C3: Group Grievance'>



In [73]:
```python
# Testing of clusters
df_km.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2633 entries, 0 to 2632
Data columns (total 14 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   C1: Security Apparatus            2633 non-null   float64
 1   C2: Factionalized Elites          2633 non-null   float64
 2   C3: Group Grievance               2633 non-null   float64
 3   E1: Economy                       2633 non-null   float64
 4   E2: Economic Inequality           2633 non-null   float64
 5   E3: Human Flight and Brain Drain  2633 non-null   float64
 6   P1: State Legitimacy              2633 non-null   float64
 7   P2: Public Services               2633 non-null   float64
 8   P3: Human Rights                  2633 non-null   float64
 9   S1: Demographic Pressures         2633 non-null   float64
 10  S2: Refugees and IDPs             2633 non-null   float64
 11  X1: External Intervention         2633 non-null   float64
 12  Country_Year                      2633 non-null   object
 13  ClusterID                         2633 non-null   int32
dtypes: float64(12), int32(1), object(1)
memory usage: 277.8+ KB
```

In [74]:
```python
#let's take a look at those countries clusters and try to make sense if the clusterin
df_km[df_km['ClusterID']==0]
```

Out[74]:

	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain	P1: State Legitimacy	P2: Public Services	P3: Human Rights
0	1.805800	1.232558	1.800606	0.931980	1.646909	1.866460	1.448098	1.553094	1.723301
1	1.805800	1.483375	1.506146	1.253989	1.515454	1.234896	1.196108	1.356130	1.586870
2	1.805800	1.585064	1.849714	1.741162	0.883755	1.518950	1.705347	1.158779	1.541520
3	1.805800	1.534123	1.849714	1.307879	1.321432	1.866460	0.949555	1.079726	1.677761
4	1.623831	0.938196	1.212015	2.178072	1.646909	1.808051	1.146357	1.553094	1.586870
...	...	...	...	...	...	...	...	...	...
2562	-0.229935	0.235992	-0.495715	0.242802	-1.330117	0.415343	0.065873	-0.924903	0.171026
2563	-0.357464	-0.655585	-1.221944	-0.174777	-0.259457	0.956110	-1.229572	0.524376	-0.503813
2564	0.185439	0.510865	1.604264	-1.031671	-1.046637	-1.219800	0.869081	-1.452790	1.225902
2565	-0.647239	0.190994	-0.252748	-0.891113	-1.212202	-0.050550	0.863755	-0.721188	0.759435
2566	-0.693391	-0.575821	-0.883584	0.158208	-1.171692	2.250740	-0.760879	-0.541911	-0.944350

1805 rows × 14 columns

In [75]:
```python
df_km[df_km['ClusterID']==1]
```

Out[75]:

	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain	P1: State Legitimacy	P2: Public Services	P3: Human Rights
101	-0.940885	-0.962467	-0.301373	0.085581	-0.244328	-0.101006	-0.108589	-0.115270	-0.462435
102	-0.101627	-0.962467	-0.980364	-0.739345	-0.140088	0.153946	-0.108589	-0.115270	-0.420941
103	-0.734922	-1.035967	0.234283	0.033338	-0.396938	-0.349106	-0.911479	-0.599419	-0.545075
104	-0.101627	0.464481	-0.738268	-1.486703	-0.244328	-0.822766	0.154688	-1.047645	0.257136
105	-1.538203	-1.178815	-0.495715	-0.891113	0.184343	1.234896	-0.403288	0.484565	-0.627237
...	...	...	...	...	...	...	...	...	...
2628	-1.911508	-1.713897	-2.559611	-1.486703	-2.089230	-1.583400	-1.934678	-2.009007	-2.019486
2629	-1.690672	-1.818909	-0.980364	-2.195662	-2.106957	-1.802855	-1.952453	-1.917325	-1.783127
2630	-1.765462	-1.913370	-1.462937	-2.058144	-1.985466	-1.871059	-1.969098	-1.858115	-1.887005
2631	-1.614885	-1.890860	-1.173672	-2.015829	-2.139030	-1.998549	-1.969098	-1.837933	-2.051228
2632	-1.223941	-1.818909	-2.465485	-1.583889	-2.166162	-1.767762	-1.952453	-2.040701	-2.082270

828 rows × 14 columns

In [76]:
```python
df_km[df_km['ClusterID']==1].count()
```

Out[76]:
```
C1: Security Apparatus              828
C2: Factionalized Elites            828
C3: Group Grievance                 828
E1: Economy                         828
E2: Economic Inequality             828
E3: Human Flight and Brain Drain    828
P1: State Legitimacy                828
P2: Public Services                 828
P3: Human Rights                    828
S1: Demographic Pressures           828
S2: Refugees and IDPs               828
X1: External Intervention           828
Country_Year                        828
ClusterID                           828
dtype: int64
```

In [77]:
```python
df_km[df_km['ClusterID']==0].count()
```

Out[77]:
```
C1: Security Apparatus              1805
C2: Factionalized Elites            1805
C3: Group Grievance                 1805
E1: Economy                         1805
E2: Economic Inequality             1805
E3: Human Flight and Brain Drain    1805
P1: State Legitimacy                1805
P2: Public Services                 1805
P3: Human Rights                    1805
S1: Demographic Pressures           1805
S2: Refugees and IDPs               1805
X1: External Intervention           1805
Country_Year                        1805
ClusterID                           1805
dtype: int64
```

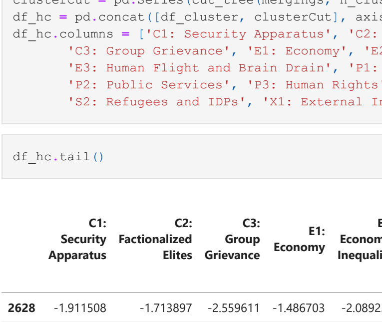
In [78]:
```python
# So cluster 1 has the countries in respective years which are prone to conflict
df_km[df_km['ClusterID']==1].Country_Year
```

Out[78]:
```
101        romania _2006
102       bulgaria_2006
103        croatia _2006
104         kuwait _2006
105          ghana _2006
               ...
2628       iceland_2020
2629       denmark_2020
2630   switzerland_2020
2631        norway_2020
2632       finland_2020
Name: Country_Year, Length: 828, dtype: object
```

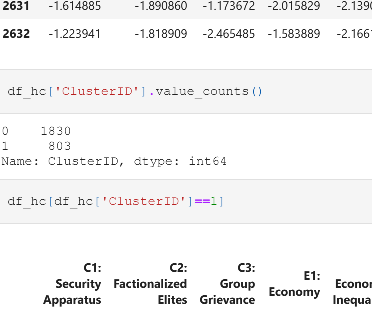
Hierarchical Clustering

In [79]:
```python
# To perform Hierarchical clustering
from scipy.cluster.hierarchy import linkage
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import cut_tree
```

In [80]:
```python
#Let's try hierarchical clustering to see if it works well
#First we'll try the single linkage procedure.
mergings = linkage(df_cluster_var, method = "single", metric='euclidean')
dendrogram(mergings)
plt.show()
```



In [81]:
```python
#Let's try complete linkage method
mergings = linkage(df_cluster_var, method = "complete", metric='euclidean')
dendrogram(mergings)
plt.show()
```



In [82]:
```python
#Okay now we are seeing some good clusters here. Let's see if they make sense if we e
clusterCut = pd.Series(cut_tree(mergings, n_clusters = 2).reshape(-1,))
df_hc = pd.concat([df_cluster, clusterCut], axis=1)
df_hc.columns = ['C1: Security Apparatus', 'C2: Factionalized Elites',
       'C3: Group Grievance', 'E1: Economy', 'E2: Economic Inequality',
       'E3: Human Flight and Brain Drain', 'P1: State Legitimacy',
       'P2: Public Services', 'P3: Human Rights', 'S1: Demographic Pressures',
       'S2: Refugees and IDPs', 'X1: External Intervention', 'Country_Year', 'Cluster
```

In [83]:
```python
df_hc.tail()
```

Out[83]:

	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain	P1: State Legitimacy	P2: Public Services	P3: Human Rights
2628	-1.911508	-1.713897	-2.559611	-1.486703	-2.089230	-1.583400	-1.934678	-2.009007	-2.019486
2629	-1.690672	-1.818909	-0.980364	-2.195662	-2.106957	-1.802855	-1.952453	-1.917325	-1.783127
2630	-1.765462	-1.913370	-1.462937	-2.058144	-1.985466	-1.871059	-1.969098	-1.858115	-1.887005
2631	-1.614885	-1.890860	-1.173672	-2.015829	-2.139030	-1.998549	-1.969098	-1.837933	-2.051228
2632	-1.223941	-1.818909	-2.465485	-1.583889	-2.166162	-1.767762	-1.952453	-2.040701	-2.082270

In [84]:
```python
df_hc['ClusterID'].value_counts()
```

Out[84]:
```
0    1830
1     803
Name: ClusterID, dtype: int64
```

In [85]:
```python
df_hc[df_hc['ClusterID']==1]
```

Out[85]:

	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain	P1: State Legitimacy	P2: Public Services	P3: Human Rights
101	-0.940885	-0.962467	-0.301373	0.085581	-0.244328	-0.101006	-0.108589	-0.115270	-0.462435
103	-0.734922	-1.035967	0.234283	0.033338	-0.396938	-0.349106	-0.911479	-0.599419	-0.545075
104	-0.101627	0.464481	-0.738268	-1.486703	-0.244328	-0.822766	0.154688	-1.047645	0.257136
107	-0.442030	-0.615849	-0.931984	-0.637610	-0.396938	-1.658836	-0.108589	-0.115270	0.343600
108	-1.538203	-0.962467	-0.738268	-0.122879	0.296306	-0.349106	-0.683600	-0.639973	-0.910757
...	...	...	...	...	...	...	...	...	...
2628	-1.911508	-1.713897	-2.559611	-1.486703	-2.089230	-1.583400	-1.934678	-2.009007	-2.019486
2629	-1.690672	-1.818909	-0.980364	-2.195662	-2.106957	-1.802855	-1.952453	-1.917325	-1.783127
2630	-1.765462	-1.913370	-1.462937	-2.058144	-1.985466	-1.871059	-1.969098	-1.858115	-1.887005
2631	-1.614885	-1.890860	-1.173672	-2.015829	-2.139030	-1.998549	-1.969098	-1.837933	-2.051228
2632	-1.223941	-1.818909	-2.465485	-1.583889	-2.166162	-1.767762	-1.952453	-2.040701	-2.082270

803 rows × 14 columns

In [86]:
```python
df_hc[df_hc['ClusterID']==0]
```

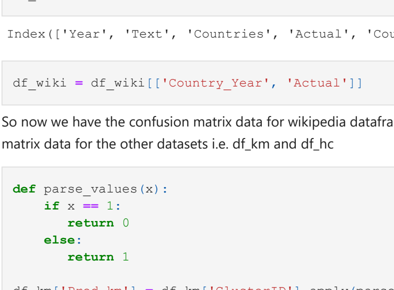
Out[86]:

	C1: Security Apparatus	C2: Factionalized Elites	C3: Group Grievance	E1: Economy	E2: Economic Inequality	E3: Human Flight and Brain Drain	P1: State Legitimacy	P2: Public Services	P3: Human Rights
0	1.805800	1.232558	1.800606	0.931980	1.646909	1.866460	1.448098	1.553094	1.723301
1	1.805800	1.483375	1.506146	1.253989	1.515454	1.234896	1.196108	1.356130	1.586870
2	1.805800	1.585064	1.849714	1.741162	0.883755	1.518950	1.705347	1.158779	1.541520
3	1.805800	1.534123	1.849714	1.307879	1.321432	1.866460	0.949555	1.079726	1.677761
4	1.623831	0.938196	1.212015	2.178072	1.646909	1.808051	1.146357	1.553094	1.586870
...	...	...	...	...	...	...	...	...	...
2570	0.497386	-1.108099	-1.703245	0.242802	-1.088904	1.749837	-1.020818	0.264027	-0.164009
2573	-0.284196	-0.117187	-0.786726	0.019375	-1.649600	1.291293	-0.278686	-0.691634	-0.830426
2575	-0.877109	-1.248065	-0.980364	-0.122879	0.240089	-0.250717	-1.515068	0.484565	-0.429596
2577	-0.817654	0.651355	-0.301373	-0.329923	-0.734998	-0.913563	-1.020818	-1.333398	-1.244544
2579	-0.276400	-0.371420	-1.173672	-0.226584	-0.916415	1.098137	-0.760879	-0.924903	-1.417396

1830 rows × 14 columns

In [87]:
```python
# The above mentioned cluster is the cluster we need to focus on
#Let's visualise the clusters here. We don't see the countries properly clustered here
#as that would give a skewed result
df_vis = df_hc[df_hc['ClusterID'] == 0]
sns.scatterplot(x = 'E2: Economic Inequality',y = 'P3: Human Rights',data = df_vis)
```

Out[87]: <AxesSubplot:xlabel='E2: Economic Inequality', ylabel='P3: Human Rights'>



So now we have df_km the dataframe of K-Means and df_hc the dataframe of hierarchical clustering. Now we will compare it with the actual data from wikipedia (data has been extracted in a separate jupyter notebook)

In [88]:
```python
df_wiki = pd.read_excel('riotsListWiki2.xlsx')
```

In [89]:
```python
df_wiki.columns
```

Out[89]: Index(['Year', 'Text', 'Countries', 'Actual'], dtype='object')

In [90]:
```python
df_wiki['Countries'] = df_wiki['Countries'].str.lower()
```

In [91]:
```python
df_wiki['Year'] = df_wiki['Year'].astype(str)
```

In [92]:
```python
df_wiki['Country_Year'] = df_wiki[['Countries', 'Year']].apply(lambda x: '_'.join(x),
```

In [93]:
```python
df_wiki.columns
```

Out[93]: Index(['Year', 'Text', 'Countries', 'Actual', 'Country_Year'], dtype='object')

In [94]:
```python
df_wiki = df_wiki[['Country_Year', 'Actual']]
```

So now we have the confusion matrix data for wikipedia dataframe ready, now we will create the confusion matrix data for the other datasets i.e. df_km and df_hc

In [95]:
```python
def parse_values(x):
    if x == 1:
        return 0
    else:
        return 1

df_km['Pred_km'] = df_km['ClusterID'].apply(parse_values)
```

In [96]:
```python
def parse_values(x):
    if x == 0:
        return 1
    else:
        return 0

df_hc['Pred_hc'] = df_hc['ClusterID'].apply(parse_values)
```

In [97]:
```python
df_km.columns
```

Out[97]:
```
Index(['C1: Security Apparatus', 'C2: Factionalized Elites',
       'C3: Group Grievance', 'E1: Economy', 'E2: Economic Inequality',
       'E3: Human Flight and Brain Drain', 'P1: State Legitimacy',
       'P2: Public Services', 'P3: Human Rights', 'S1: Demographic Pressures',
       'S2: Refugees and IDPs', 'X1: External Intervention', 'Country_Year',
       'ClusterID', 'Pred_km'],
      dtype='object')
```

In [98]:
```python
df_km = df_km[['Country_Year','Pred_km']]
```

In [99]:
```python
df_hc = df_hc[['Country_Year','Pred_hc']]
```

In [100…
```python
df_res = df_km.merge(df_hc, on='Country_Year', how='left')
```

In [101…
```python
df_res
```

Out[101…

	Country_Year	Pred_km	Pred_hc
0	sudan_2006	1	1
1	congo democratic republic_2006	1	1
2	cote d'ivoire_2006	1	1
3	iraq_2006	1	1
4	zimbabwe_2006	1	1
...	...	...	...
2810	iceland_2020	0	0
2811	denmark_2020	0	0
2812	switzerland_2020	0	0
2813	norway_2020	0	0
2814	finland_2020	0	0

2815 rows × 3 columns

In [102…
```python
df_cm = df_wiki.merge(df_res, on='Country_Year', how='left')
```

In [103…
```python
df_cm = df_cm.dropna(how='any',axis=0)
```

In [104…
```python
df_cm.reset_index()
```

Out[104…

	index	Country_Year	Actual	Pred_km	Pred_hc
0	1	afghanistan_2006	1	1.0	1.0
1	2	somalia_2006	1	1.0	1.0
2	3	lebanon_2006	1	1.0	1.0
3	4	turkey_2006	1	1.0	1.0
4	5	pakistan_2006	1	1.0	1.0
...	...	...	...	...	...
363	388	libya_2020	1	1.0	1.0
364	389	sweden_2020	1	0.0	0.0
365	391	nigeria_2020	1	1.0	1.0
366	392	afghanistan_2020	1	1.0	1.0
367	393	indonesia_2020	1	1.0	1.0

368 rows × 5 columns

In [105…
```python
df_cm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 368 entries, 1 to 393
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Country_Year  368 non-null    object
 1   Actual        368 non-null    int64
 2   Pred_km       368 non-null    float64
 3   Pred_hc       368 non-null    float64
dtypes: float64(2), int64(1), object(1)
memory usage: 14.4+ KB
```

In [106…
```python
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
# Scores on k-means set.
print(classification_report(df_cm['Actual'],df_cm['Pred_km']))
print(confusion_matrix(df_cm['Actual'],df_cm['Pred_km']))
print('Accuracy Score on k-means:  ',accuracy_score(df_cm['Actual'],df_cm['Pred_km']))
```

```
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00         0
         1.0       1.00      0.70      0.82       368

    accuracy                           0.70       368
   macro avg       0.50      0.35      0.41       368
weighted avg       1.00      0.70      0.82       368

[[  0   0]
 [111 257]]
Accuracy Score on k-means:   0.6983695652173914
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: Un
definedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 in label
s with no true samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [107…
```python
# Scores on hc set.
print(classification_report(df_cm['Actual'],df_cm['Pred_hc']))
print(confusion_matrix(df_cm['Actual'],df_cm['Pred_hc']))
print('Accuracy Score on the Hierarchical set:  ',accuracy_score(df_cm['Actual'],df_cr
```

```
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00         0
         1.0       1.00      0.70      0.83       368

    accuracy                           0.70       368
   macro avg       0.50      0.35      0.41       368
weighted avg       1.00      0.70      0.83       368

[[  0   0]
 [109 259]]
Accuracy Score on the Hierarchical set:   0.7038043478260869
```

In [108…
```
from sklearn import metrics
```

Homogeneity, Completeness and V-Measure Scores

In [109…
```
labels_true = df_cm['Actual']
```

In [110…
```
labels_predKM = df_cm['Pred_km']
```

In [111…
```
labels_predHC = df_cm['Pred_hc']
```

In [122…
```
metrics.homogeneity_completeness_v_measure(labels_true, labels_predKM)
```

Out[122…
```
(1.0, 1.4507043439045496e-15, 2.901408687809095e-15)
```

In [123…
```
metrics.homogeneity_completeness_v_measure(labels_true, labels_predHC)
```

Out[123…
```
(1.0, 0.0, 0.0)
```

Fowlkes-Mallows scores

In [124…
```
metrics.fowlkes_mallows_score(labels_true, labels_predKM)
```

Out[124…
```
0.7599690882171529
```

In [125…
```
metrics.fowlkes_mallows_score(labels_true, labels_predHC)
```

Out[125…
```
0.762847548298654
```

Contingency Matrix

In [127…
```
from sklearn.metrics.cluster import contingency_matrix
```

In [128…
```
contingency_matrix(labels_true, labels_predKM)
```

Out[128…
```
array([[111, 257]])
```

In [129…
```
contingency_matrix(labels_true, labels_predHC)
```

Out[129…
```
array([[109, 259]])
```