Data Cleansing : Start

In [1424]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
pd.set_option('display.max_colwidth',300)
pd.set_option('display.max_rows',900)
```

In [1425]:
```
TerrorDTframe=pd.read_csv('global_terrorism_clean.csv')
```

In [1426]:
```
TerrorDTframe.head()
```

Out[1426]:

	date	type	dead	injured	location	details	perpetrator
0	1970-01-13	Shotdown	7	0	Urabá, Colombia	An UH-1 Iroquois helicopter from the Colombian Air Force disappears amidst strange circumstances in the Urabá Antioquia. The PLA was awarded its shotdown. The seven crew members died.[1]	EPL
1	1970-02-08	Bombing	0	0	Belfast, Northern Ireland	A bomb explodes at the home of Ulster Liberal Party MP Sheelagh Murnaghan	Ulster Volunteer Force
2	1970-02-10	Shooting, grenade attacks	1	23	Munich, West Germany	A bus carrying passengers to an El Al airplane at the Munich-Riem Airport, West Germany was attacked by Palestinian terrorists. One person was killed and 23 were wounded in the attack.[2]	PDFLP
3	1970-02-18	Bombing	0	0	County Donegal, Republic of Ireland	A bomb detonated in a TV station that transmitted RTÉ (Mainly Irish broadcaster)	Ulster Volunteer Force
4	1970-02-21	Bombing	47	0	Switzerland	A bomb explodes in the rear of Swissair Flight 330, causing it to crash near Zürich, killing 38 passengers and all 9 crew members. The attack was carried out by Palestinian group PFLP-GC	PFLP-GC

The process of extracting the name of the country from column:"Location":

Now the location column will have only the country names

In [1427]:
```
import pycountry
countryList=[]
for each in pycountry.countries:
    countryList.append(each.name)

```

In [1428]:
```
countryList.append('Iran')
countryList.append('Russia')
countryList.append('Syria')
countryList.append('Gaza City')
countryList.append('Bolivia')
countryList.append('Bosnia')
countryList.append('West Bank')
countryList.append('Rafah')
countryList.append('Ivory Coast')
countryList.append('Kedumim')
countryList.append('Transnistria')
if 'Baghdad' in countryList:
    print('yes')
countryList[0:5]
```

Out[1428]:
```
['Aruba', 'Afghanistan', 'Angola', 'Anguilla', 'Åland Islands']
```

In [1429]:
```python
def ExtractCountry(locName):
    flag=False
    tempstr=''
    for each in countryList:
        if each in locName:
            flag=True
            tempstr = each
    if flag==True:
        return tempstr
    else:
        return ''

```

In [1430]:
```python
TerrorDTframe['location'] = TerrorDTframe.location.apply(lambda x: ExtractCountry(x))
```

The process of extracting the name of the country from Location Ends :

In [1431]:
```python
TerrorDTframe = TerrorDTframe[TerrorDTframe.location.notna()]
```

In [1432]:
```python
TerrorDTframe.rename(columns={'date':'Date','type':'Type','dead':'Dead','injured':'Injured','location':'Location'
                             'details':'Description','perpetrator':'Perpetrators'},inplace=True)
```

Using NLP: Removing the stopwords from the Description column of TerrorDTframe

In [1433]:
```python
import nltk
from nltk.corpus import stopwords
StopWrds = stopwords.words('english')
from nltk.tokenize import word_tokenize as wt
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer
import re
for each in [',','.']:
    StopWrds.append(each)
```

In [1434]:
```python

wnl = WordNetLemmatizer()
ps=PorterStemmer()
def funcNLP(description):
    wrdset= wt(description)
    filtered_description = ''
    filtered_sentence_list = [wnl.lemmatize(elem) for elem in wrdset if elem.lower() not in StopWrds ]
    filtered_description = ' '.join(each for each in filtered_sentence_list)
    filtered_description = re.sub('\[.*\]','',filtered_description)
    filtered_description = re.sub('\(.*\)','',filtered_description)
    return (filtered_description)

```

In [1435]:
```python
TerrorDTframe['Description'] = TerrorDTframe['Description'].apply(lambda x : funcNLP(x))
```

NLP extraction technique ends

Converting all the blank fields to np.nan and then removing the rows containing nan values from the dataframe

In [1436]:
```python
def func(elem):
    if str(elem) == '':
        return np.nan
    else:
        return elem


```

In [1437]:
```python
TerrorDTframe = TerrorDTframe.applymap(lambda x: func(x))
```

In [1438]:
```
TerrorDTframe.isna().sum()
```

Out[1438]:
```
Date              0
Type              3
Dead              0
Injured           0
Location        129
Description      77
Perpetrators    410
dtype: int64
```

In [1439]:
```
TerrorDTframe_notna = TerrorDTframe.dropna(axis=0,how='any')
```

In [1440]:
```
TerrorDTframe_notna.isna().sum()
```

Out[1440]:
```
Date            0
Type            0
Dead            0
Injured         0
Location        0
Description     0
Perpetrators    0
dtype: int64
```

Replacing all the date-time values to Date only.Like from 1970-01-02 to 1970

In [1441]:
```
TerrorDTframe_notna['Date'] = pd.to_datetime(TerrorDTframe_notna['Date'])
TerrorDTframe_notna['Date'] = TerrorDTframe_notna['Date'].dt.year
```

In [1442]:
```
TerrorDTframe_notna.head(1)
```

Out[1442]:

	Date	Type	Dead	Injured	Location	Description	Perpetrators
0	1970	Shotdown	7	0	Colombia	UH-1 Iroquois helicopter Colombian Air Force disappears amidst strange circumstance Urabá Antioquia PLA awarded shotdown seven crew member died	EPL

Extracting the names of the terrorists and keeping only 17 Terrorists

In [1443]:
```
def terror_group_merge(x):
    if 'Islamic State' in str(x).strip().title():
        return 'Islamic State'
    if 'Boko Haram' in str(x).strip().title():
        return 'Boko Haram'
    if 'Al-Shabaab' in str(x).strip().title():
        return 'Al-Shabaab'
    if 'Al-Qaeda' in str(x).strip().title():
        return 'Al-Qaeda'
    if 'Taliban' in str(x).strip().title():
        return 'Taliban'
    else:
        return x
```

In [1444]:
```
TerrorDTframe_notna.Perpetrators = TerrorDTframe_notna.Perpetrators.apply(lambda x : terror_group_merge(x))
Terror_Count = TerrorDTframe_notna.Perpetrators.value_counts().to_frame().reset_index(drop=False)\
                                            .rename(columns={'index':'Perpetrators','Perpetrators':'Perpetrators_(
Terror_Count = Terror_Count[Terror_Count.Perpetrators_Count > 50]
```

In [1445]:
```
terror_Name_List = Terror_Count.Perpetrators
TerrorDTframe_notna = TerrorDTframe_notna[TerrorDTframe_notna.Perpetrators.isin(terror_Name_List)]
TerrorDTframe_notna.shape
```

Out[1445]:
```
(5456, 7)
```

In [1446]:
```
TerrorDTframe_notna.head(1)
```

Out[1446]:

	Date	Type	Dead	Injured	Location	Description	Perpetrators
24	1970	Ambush, Shooting	7	0	Colombia	7 soldier killed ascribed army 's Ricaurte battalion fell ambush FARC rural area Cimitarra	FARC

Keeping only one Type in each row

In [1447]:
```
def extract_One_Name(x):
    if 'Bomb' in x.strip().title():
        return 'Bombing'
    elif 'Shoot' in x.strip().title():
        return 'Shooting'
    elif 'Execut' in x.strip().title():
        return 'Execution'
    elif ',' in str(x):
        return(str(x).split(',')[0])
    else:
        return(x)

```

In [1448]:
```
TerrorDTframe_notna['Type'] = TerrorDTframe_notna.Type.apply(lambda x : extract_One_Name(x))
TerrorDTframe_notna.Type.shape
```

Out[1448]:
```
(5456,)
```

In [1449]:
```
TerrorDTframe_notna = TerrorDTframe_notna.reset_index(drop=True)
TerrorDTframe_notna.shape
```

Out[1449]:
```
(5456, 7)
```

In [1450]:
```
TerrorDTframe_notna.to_csv('C:\\Users\\supratik chanda\\Documents\\All Docs\\FinalTerrorDataFrame.csv',index=Fals
```

In [1644]:
```
TerrorDTframe_notna.head()
```

Out[1644]:

	Date	Type	Dead	Injured	Location	Description	Perpetrators
0	1970	Shooting	7	0	Colombia	7 soldier killed ascribed army 's Ricaurte battalion fell ambush FARC rural area Cimitarra	FARC
1	1971	Shooting	10	0	Colombia	site known San Miguel rural Gaitania member FARC attack patrol 23 soldier assigned Caicedo Battalion 10 uniformed men die action	FARC
2	1972	Bombing	1	0	Canada	Cuban official Sergio Pérez Castillo killed explosion Cuban consulate Montreal	Unknown
3	1972	Bombing	9	130	Ireland	Bloody Friday : Nine killed 130 injured Provisional Irish Republican Army set 22 bomb	PIRA
4	1972	Bombing	9	30	United Kingdom	Claudy bombing ; three car bomb detonated Claudy killing nine people group claimed responsibility	PIRA

Data Cleansing: Stop

Type *Markdown* and LaTeX: α^2

Start of Analyzing DataSet

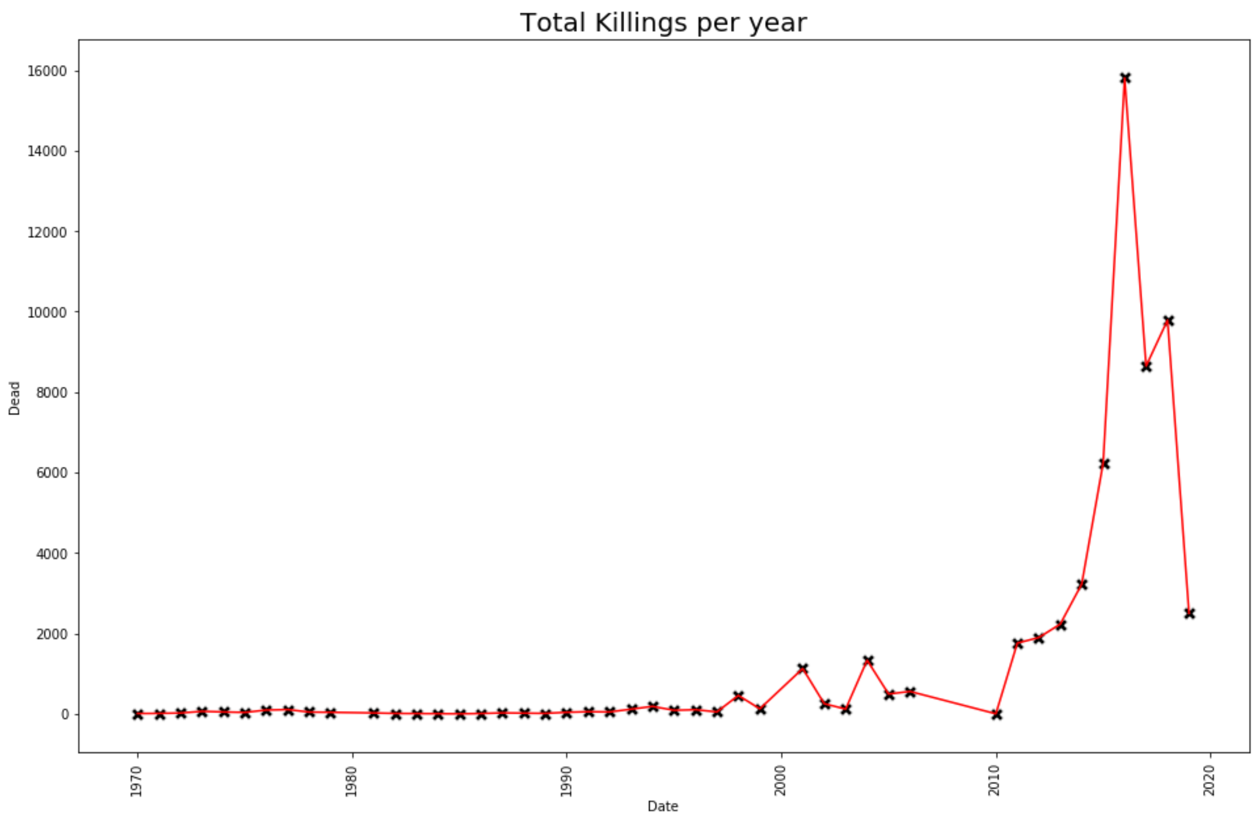
Killing Worldwide

Killings by Year

Let's start by looking to the killings in terror activities. In the following , the size of the areas corresponding to each year is proportional with the number of kills in that year in the terrorist activities. We can easily see that there was a massive increase in killings in terrorist activities in the years from 2012 and in the last 3 years (2017-2019) the volume was significantly higher than in the previous years.

In [1556]:

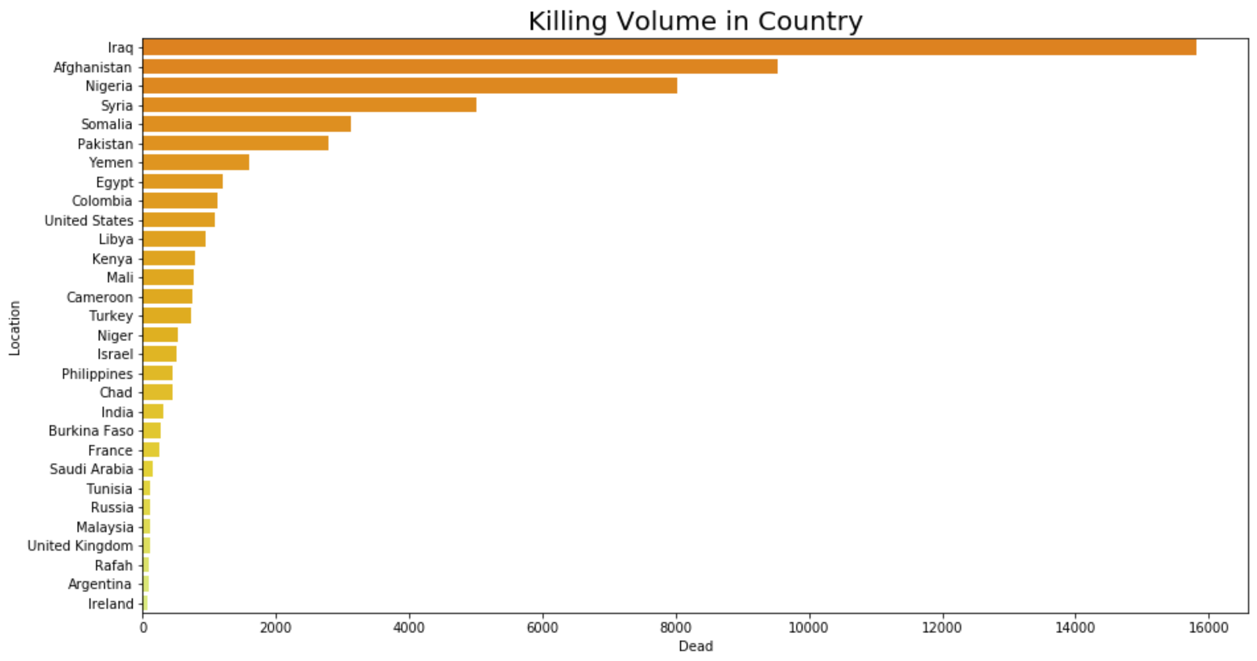
```
killing_worldwide = TerrorDTframe_notna.groupby('Date').agg({'Dead':np.sum})
killing_worldwide =killing_worldwide .reset_index(drop=False)
import squarify
plt.figure(figsize=(16,10))
sns.lineplot(x=killing_worldwide['Date'],y=killing_worldwide['Dead'],color='red')
sns.scatterplot(x=killing_worldwide['Date'],y=killing_worldwide['Dead'],s=100,marker='X',color='black')
#squarify.plot(sizes=killing_worldwide['Dead'], label=killing_worldwide['Date'],alpha=0.4)
#plt.axis('off')
plt.title('Total Killings per year',fontsize=20)
plt.xticks(rotation=90)
plt.show()

```



Killings by country

If we look now to the country split, we see that there are few countries where the number of killed people in terrorist attacks is very large. Some of them have experienced recent massive increase, like Iraq, Nigeria, Syria and Afganistan while others have a long history, like Sri Lanka, Colombia, India, Pakistan.

In [1452]:
```
killing_by_Country = TerrorDTframe_notna.groupby('Location').agg({'Dead':np.sum})
killing_by_Country =killing_by_Country.sort_values(by='Dead',ascending=False).reset_index(drop=False)
killing_by_Country = killing_by_Country.query('Dead != 0')
import squarify
plt.figure(figsize=(15,8))
sns.barplot(x=killing_by_Country['Dead'][0:30], y=killing_by_Country['Location'][0:30],palette='Wistia_r')
#plt.axis('off')
plt.title('Killing Volume in Country',fontsize=20)
#plt.xticks(rotation=90)
plt.show()
```



Killings by Perpetrators and years

In [1453]:
```
TerrorDTframe_notna.Perpetrators.unique()
```

Out[1453]:
```
array(['FARC', 'Unknown', 'PIRA', 'ELN', 'PKK', 'Hamas', 'Al-Qaeda',
       'Taliban', 'Islamist insurgents', 'Abu Sayyaf', 'Al-Shabaab',
       'Boko Haram', 'Islamic State', "New People's Army",
       'Palestinian lone wolf', 'Jaish-e-Mohammed (suspected)', 'CPI'],
      dtype=object)
```

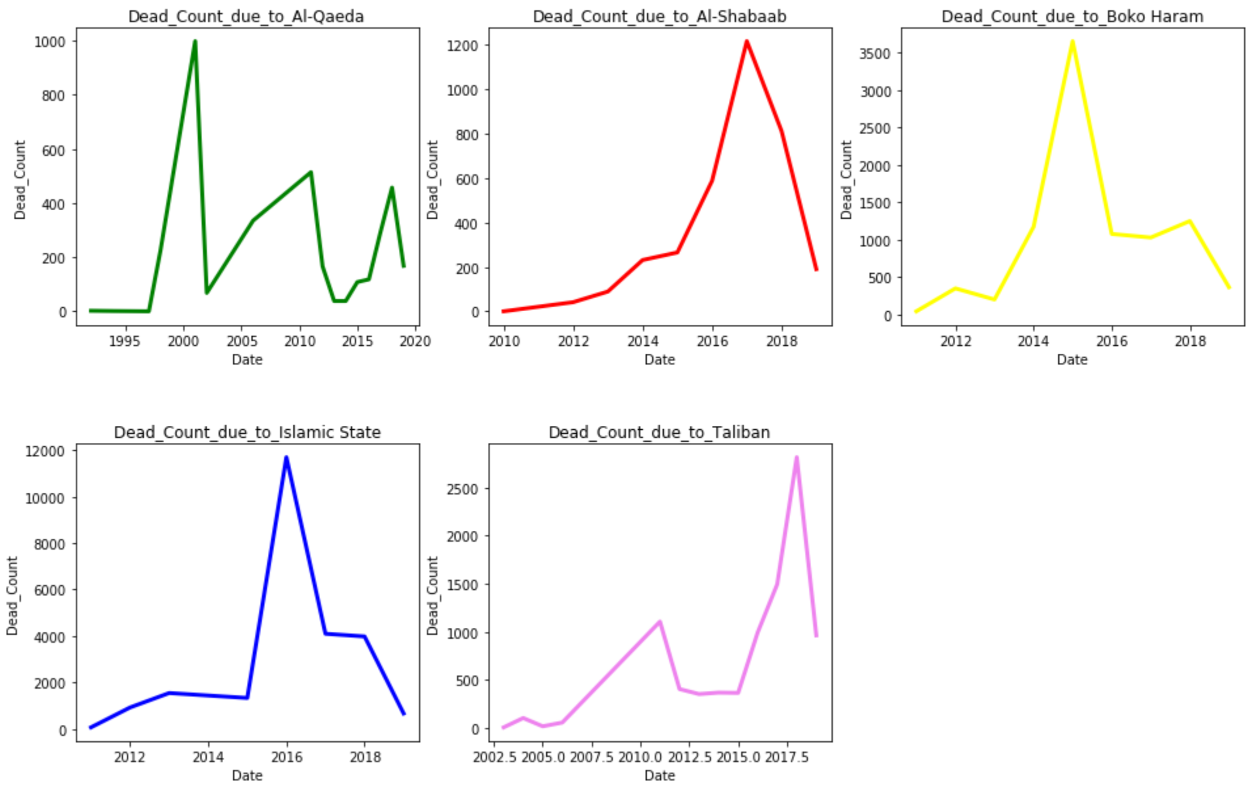
In [1610]:
```
snsDTFrame = TerrorDTframe_notna.loc[:,['Date','Dead','Perpetrators']]
snsDTFrame = snsDTFrame[snsDTFrame.Perpetrators.isin(['Boko Haram','Islamic State',\
                                                      'Taliban','Jaish-e-Mohammed','Al-Shabaab','Al-Qaeda
yearDTFrame=pd.Series()
Dead_Count_DTFrame = pd.Series()
Perpetrators_DTFrame= pd.Series()
snsDTFrame.drop(index=[2474],axis=0,inplace=True)
for key,value in snsDTFrame.groupby(['Perpetrators','Date']):
    yearDTFrame = yearDTFrame.append(pd.Series(value.Date.unique()[0]))
    Dead_Count_DTFrame = Dead_Count_DTFrame.append(pd.Series(value.Dead.sum()))
    Perpetrators_DTFrame = Perpetrators_DTFrame.append(pd.Series(value.Perpetrators.unique()[0]))
OverallDTFrame = pd.concat([yearDTFrame,Dead_Count_DTFrame,Perpetrators_DTFrame],axis=1)
OverallDTFrame = OverallDTFrame.reset_index(drop=True)
OverallDTFrame.columns=['Date','Dead_Count','Perpetrators']
```

In [1612]:
```
OverallDTFrame.head()
```

Out[1612]:

	Date	Dead_Count	Perpetrators
0	1992	2	Al-Qaeda
1	1994	1	Al-Qaeda
2	1997	0	Al-Qaeda
3	1998	224	Al-Qaeda
4	2001	998	Al-Qaeda

In [1645]:
```
i=1
colors=['green','red','yellow','blue','violet']
fig = plt.figure(figsize=(16,10))
fig.subplots_adjust(hspace=0.4)
for key,value in OverallDTFrame.groupby('Perpetrators'):
    ax = fig.add_subplot(2,3,i)
    sns.lineplot(x=value.Date,y=value.Dead_Count,lw=3,color=colors[i-1])
    i+=1
    plt.title('Dead_Count_due_to_'+str(key))
```



American Citizens Killed and Injured

Let’s look now to the numbers of american citizens killed as well as injured worldwide.The leading country for US citizens killed in United States

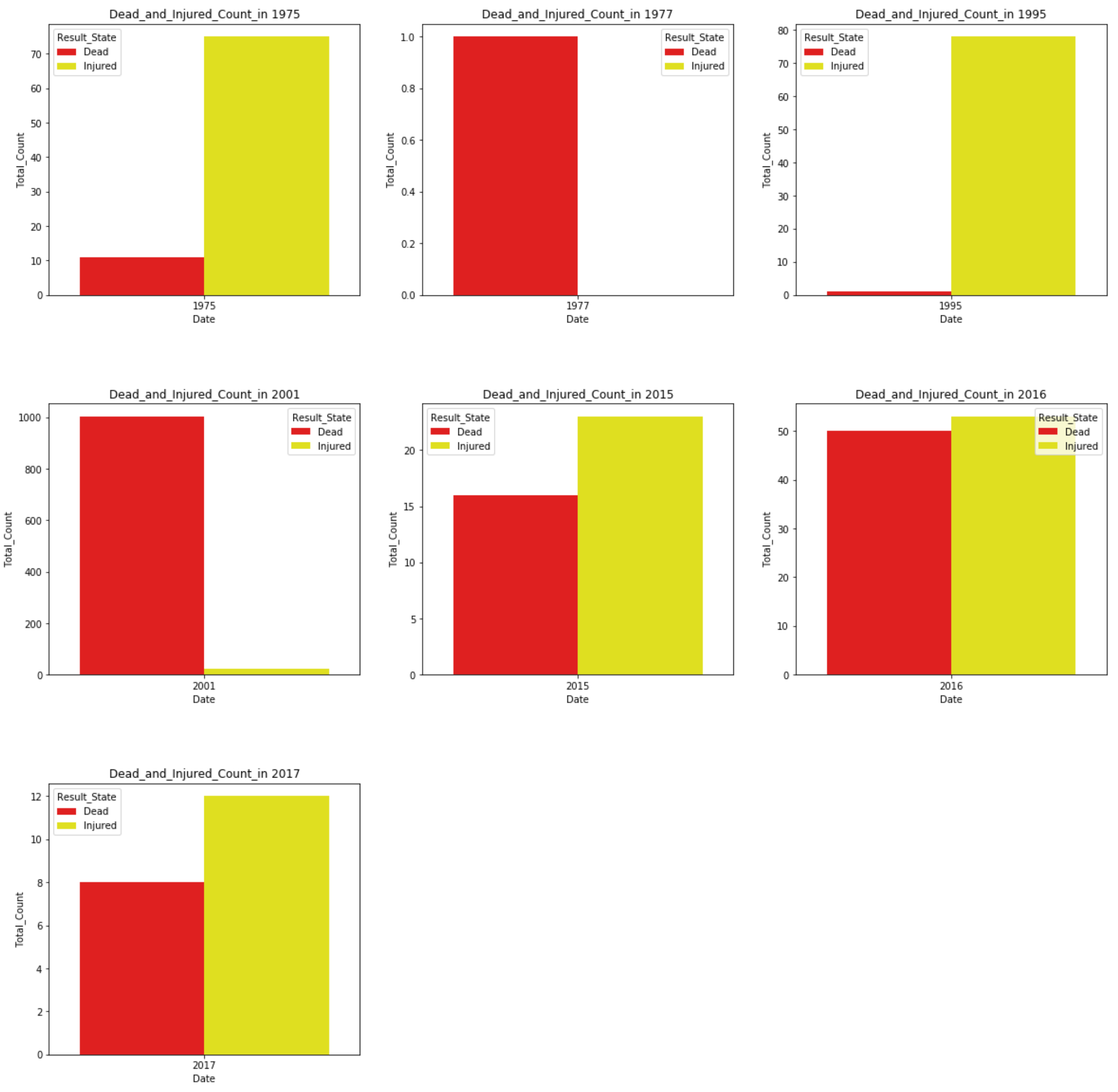
In [1629]:

```
TerrorDT_In_USA = TerrorDTframe_notna[TerrorDTframe_notna.Location =='United States'][['Date','Dead','Injured','L
TerrorDT_In_USA.reset_index(drop=True,inplace=True)
TerrorDT_In_USA_pivot_table = pd.pivot_table(TerrorDT_In_USA,index='Location',columns ='Date',\
                                              values=['Dead','Injured'],aggfunc=np.sum)
TerrorDT_In_USA_pivot_table = TerrorDT_In_USA_pivot_table.T.reset_index(drop=False).rename(columns=\
                                                              {'level_0':'Result_State','United States':'Total_Co
TerrorDT_In_USA_pivot_table
```

Out[1629]:

Location	Result_State	Date	Total_Count
0	Dead	1975	11
1	Dead	1977	1
2	Dead	1995	1
3	Dead	2001	1003
4	Dead	2015	16
5	Dead	2016	50
6	Dead	2017	8
7	Injured	1975	75
8	Injured	1977	0
9	Injured	1995	78
10	Injured	2001	23
11	Injured	2015	23
12	Injured	2016	53
13	Injured	2017	12

In [1641]:
```python
i=1
fig = plt.figure(figsize=(20,20))
fig.subplots_adjust(hspace=0.4)
for key,value in TerrorDT_In_USA_pivot_table.groupby('Date'):
    value = value.reset_index(drop=True)
    #display(value)
    ax = fig.add_subplot(3,3,i)
    sns.barplot(x=value.Date,y=value.Total_Count,lw=3,hue=value.Result_State,palette=['red','yellow'])
    i+=1
    plt.title('Dead_and_Injured_Count_in '+str(key))
```



Indian Citizen killed and injured

In [1576]:
```
TerrorDT_In_India = TerrorDTframe_notna[TerrorDTframe_notna.Location =='India'][['Date','Dead','Injured','Locatio
TerrorDT_In_India.reset_index(drop=True,inplace=True)
TerrorDT_In_India.head()
```

Out[1576]:

	Date	Dead	Injured	Location
0	2003	1	28	India
1	2003	10	70	India
2	2003	4	32	India
3	2004	0	19	India
4	2004	2	39	India

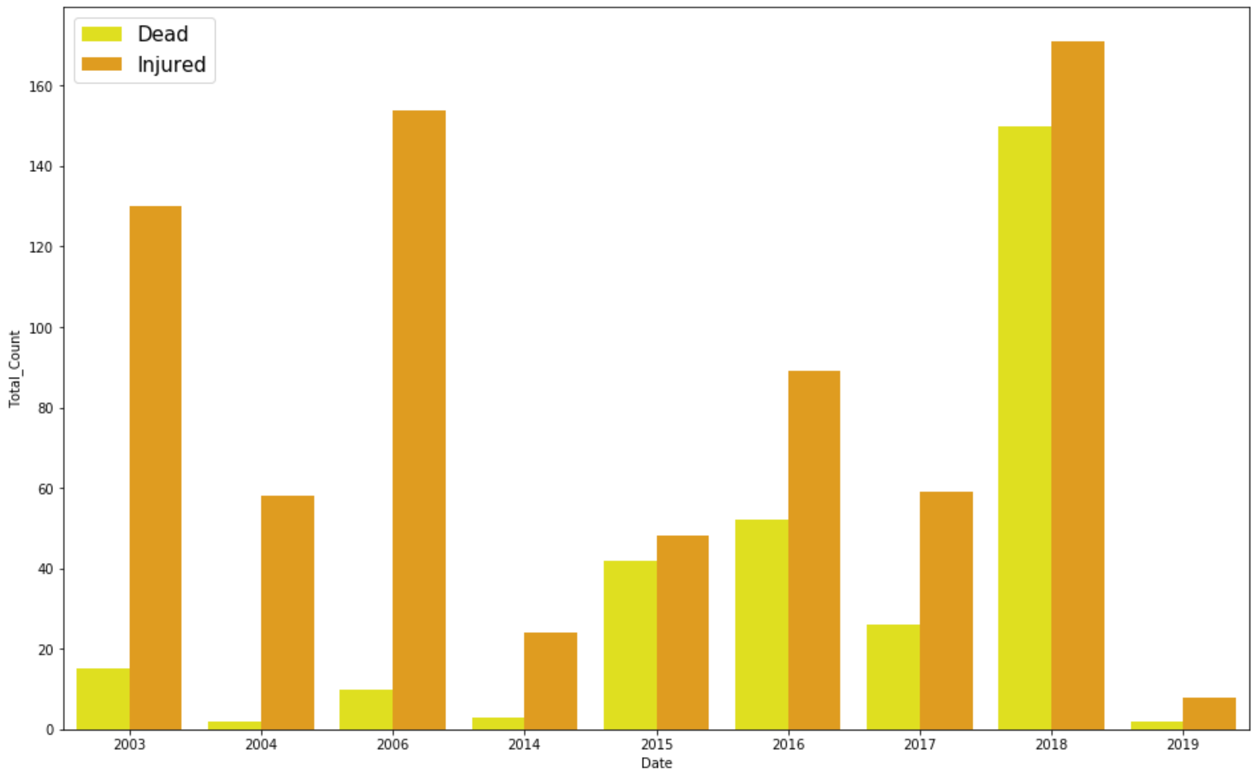
In [1600]:
```
TerrorDT_In_India_pivot_table = pd.pivot_table(TerrorDT_In_India,index='Location',columns ='Date',\
                                                              values=['Dead','Injured'],aggfunc=np.sum)
TerrorDT_In_India_pivot_table = TerrorDT_In_India_pivot_table.T.reset_index(drop=False).rename(columns=\
                                                                                    {'level_0':'Result_State','India':'Total_Count'})
TerrorDT_In_India_pivot_table.head()
```

Out[1600]:

Location	Result_State	Date	Total_Count
0	Dead	2003	15
1	Dead	2004	2
2	Dead	2006	10
3	Dead	2014	3
4	Dead	2015	42

In [1643]:
```
plt.figure(figsize=(16,10))
sns.barplot(x=TerrorDT_In_India_pivot_table.Date, y=TerrorDT_In_India_pivot_table.Total_Count,hue=TerrorDT_In_Ind
plt.legend(loc='best',fontsize=15)
```

Out[1643]:
```
<matplotlib.legend.Legend at 0x1d9a3ed06a0>
```



Type *Markdown* and LaTeX: α^2

Motive Analysis

Motive wordcloud

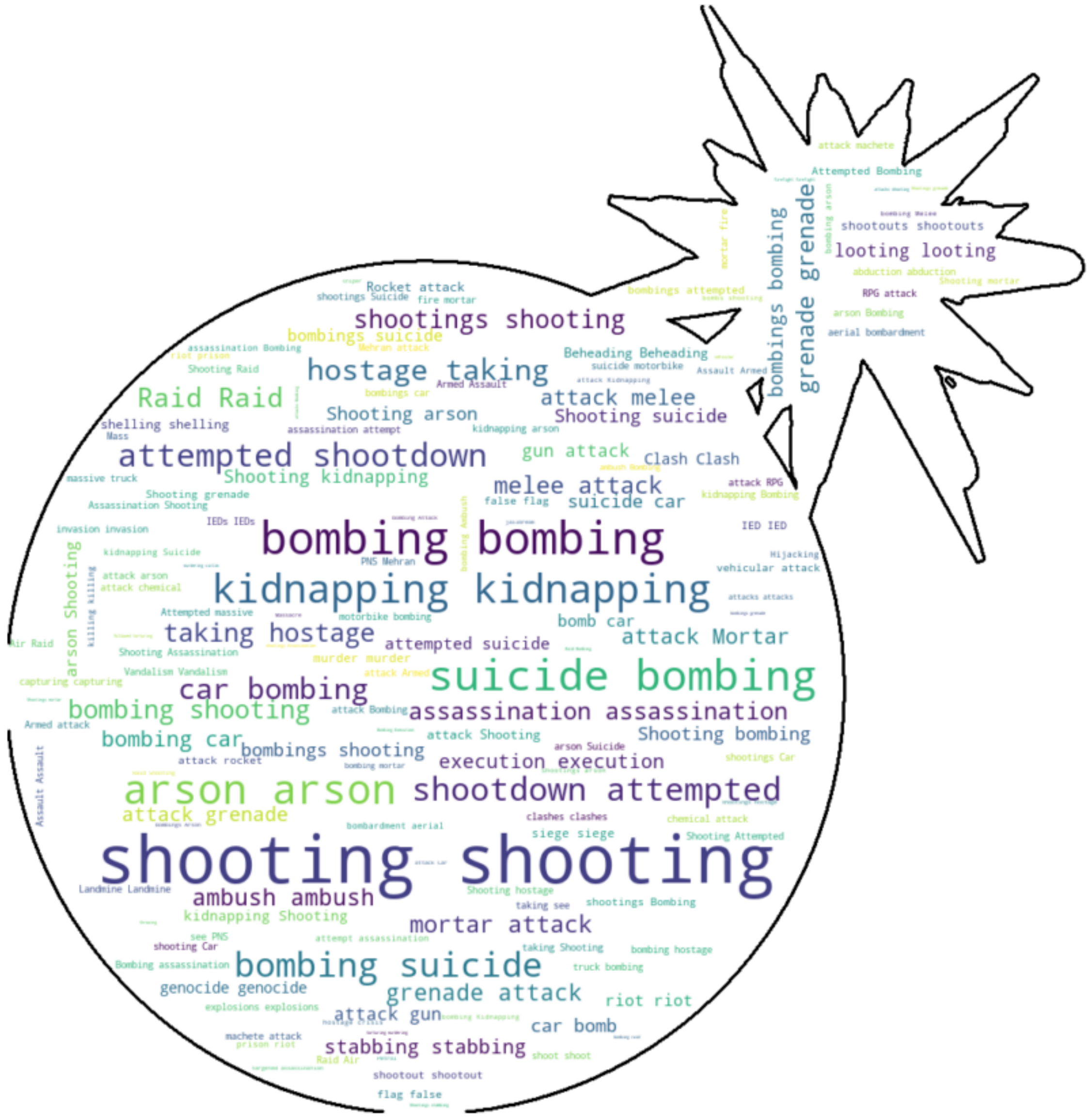
Let’s analyze the motive of the attacks. We will treat two alike words the same irrespective of their case

In [1459]:

```python
motive_Of_Attacks=''
TerrorDTframe_for_motive_analysis=pd.read_csv('global_terrorism_clean.csv')
for each in TerrorDTframe_for_motive_analysis.type:
    eachsplitted = str(each).split(',')
    if len(eachsplitted)> 1:
        for elem in eachsplitted:
            motive_Of_Attacks = motive_Of_Attacks + ' ' + elem
    else:
        motive_Of_Attacks = motive_Of_Attacks + ' ' + elem
```

In [1460]:

```python
from wordcloud import WordCloud
from PIL import Image
def image_generator(image):
    python_mask = np.array(Image.open(image))
    wcObj = WordCloud(background_color='white',mask=python_mask,contour_color='black',contour_width=3)
    wcObj.generate(motive_Of_Attacks)
    fig=plt.figure(figsize=(20,15))
    plt.imshow(wcObj,interpolation='bilinear')
    plt.axis('off')
    plt.savefig('bomb_wordcloud.png')
    #plt.show()
image_generator('bomb.jpg')
```

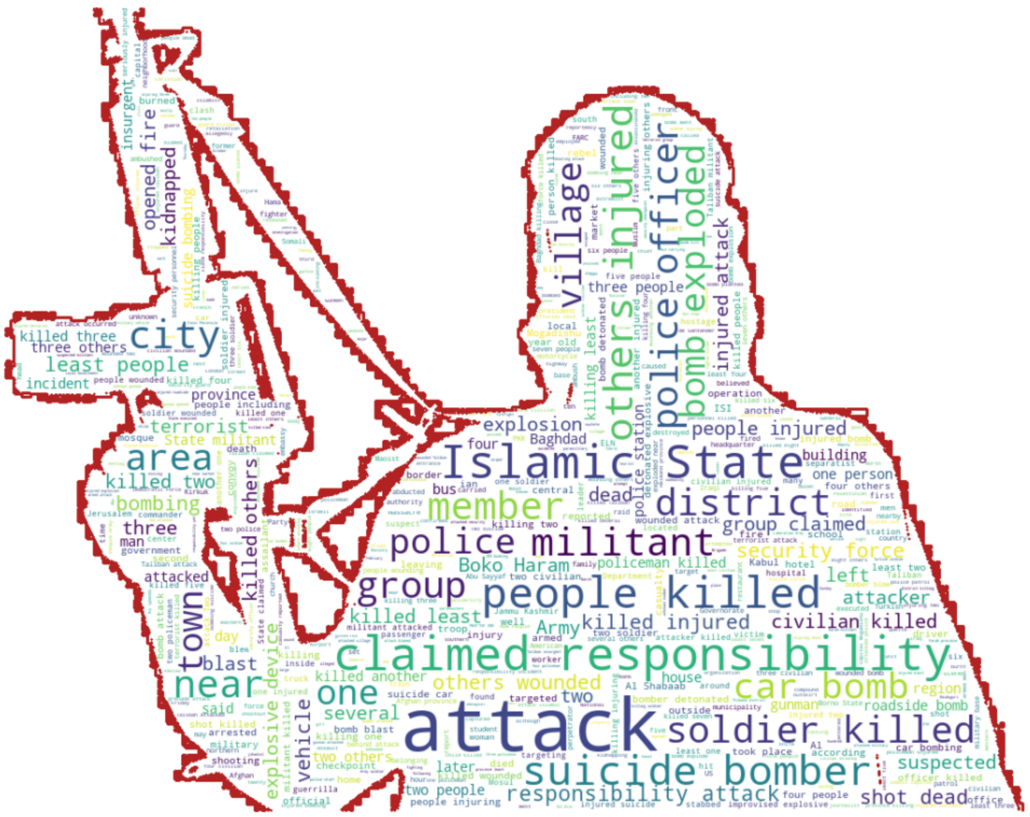


Summary WordCloud

Let’s perform a similar text analysis on the Description field, the field that describes the terrorist event.

In [1461]:

```python
from wordcloud import WordCloud
from PIL import Image
def image_generator(image):
    python_mask = np.array(Image.open(image))
    wcObj = WordCloud(background_color='white',max_words=600,mask=python_mask,contour_color='firebrick',contour_w
    wcObj.generate(strCorpus)
    fig=plt.figure()
    fig.set_figwidth(20)
    fig.set_figheight(20)
    plt.imshow(wcObj,interpolation='bilinear')
    plt.axis('off')
    plt.savefig('wordcloud.png')
    #plt.show()
image_generator('Terrorist.jpg')
```



We can see that the most frequent concepts used are claimed, bomb, responsibility, attack, shot, suicide bomber, militant, Islamic State, opened fire, injured, police.

Type *Markdown* and LaTeX: α^2

Perpetrators Group Analysis

Let’s see now the distribution of events and impact grouped on the perpetrators name.

First we show the perpetrators group based on the number of dead victims.

In [1462]:

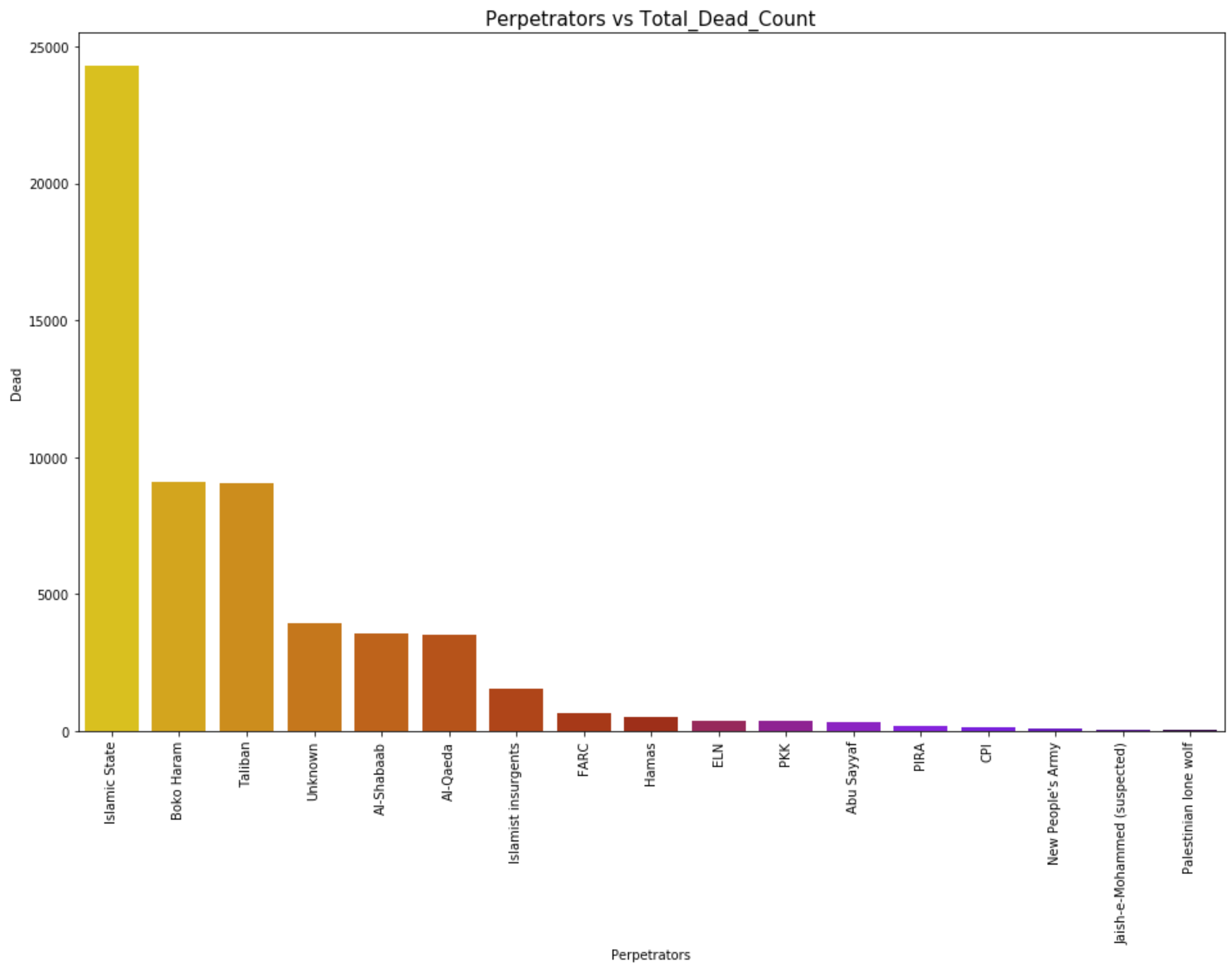
```
Perpetrators = TerrorDTframe_notna.groupby(['Perpetrators']).agg({'Dead':np.sum})
Perpetrators = Perpetrators.sort_values(by=['Dead'],ascending=False).reset_index(drop=False)
Perpetrators
```

Out[1462]:

	Perpetrators	Dead
0	Islamic State	24285
1	Boko Haram	9121
2	Taliban	9034
3	Unknown	3929
4	Al-Shabaab	3578
5	Al-Qaeda	3527
6	Islamist insurgents	1532
7	FARC	671
8	Hamas	505
9	ELN	392
10	PKK	387
11	Abu Sayyaf	323
12	PIRA	170
13	CPI	134
14	New People's Army	109
15	Jaish-e-Mohammed (suspected)	39
16	Palestinian lone wolf	35

In [1463]:

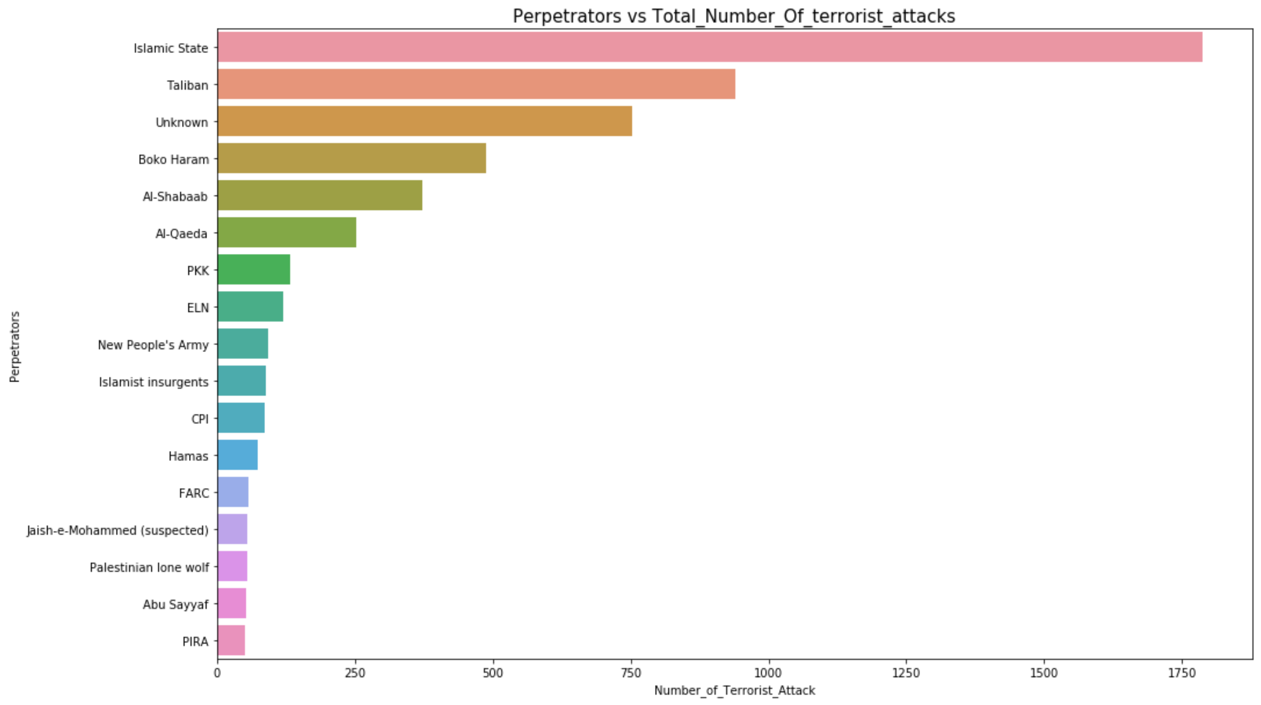
```
plt.figure(figsize=(16,10))
sns.barplot(x=Perpetrators['Perpetrators'],y=Perpetrators['Dead'],palette='gnuplot_r')
plt.xticks(rotation=90)
plt.title('Perpetrators vs Total_Dead_Count',fontsize=15)
plt.show()
```



Type *Markdown* and LaTeX: α^2

Let’s see also the terror groups based on number of terrorist attacks

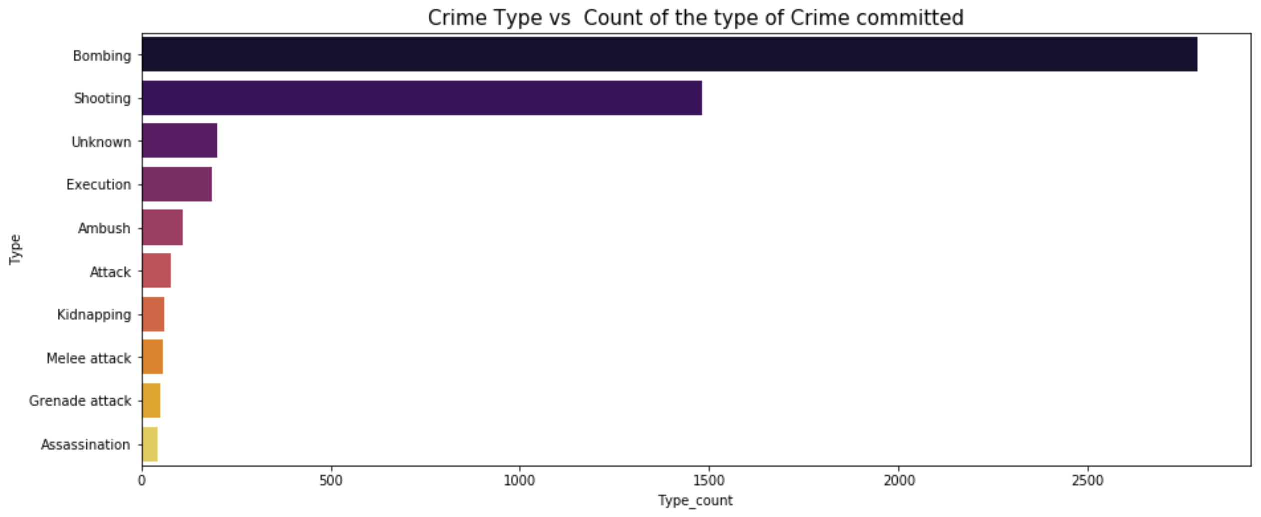
In [1464]:
```
totalTerroristAttack = TerrorDTframe_notna.Perpetrators.value_counts().to_frame().reset_index(drop=False)\
                                                .rename(columns={'index':'Perpetrators','Perpetrators':'Number_of
plt.figure(figsize=(16,10))
sns.barplot(y=totalTerroristAttack.Perpetrators,x=totalTerroristAttack.Number_of_Terrorist_Attack)
plt.title('Perpetrators vs Total_Number_Of_terrorist_attacks',fontsize=15)
plt.show()
```



Type *Markdown* and LaTeX: α^2

Top 10 crimes committed from 1970 to 2019

In [1465]:
```
Attack_count = TerrorDTframe_notna.Type.value_counts().to_frame().reset_index(drop=False)\
                                                            .rename(columns={'index':'Type','Type':'Type_count'})
plt.figure(figsize=(15,6))
sns.barplot(y=Attack_count['Type'][0:10],x=Attack_count['Type_count'][0:10],palette='inferno')
plt.title('Crime Type vs  Count of the type of Crime committed',fontsize=15)
plt.show()
```

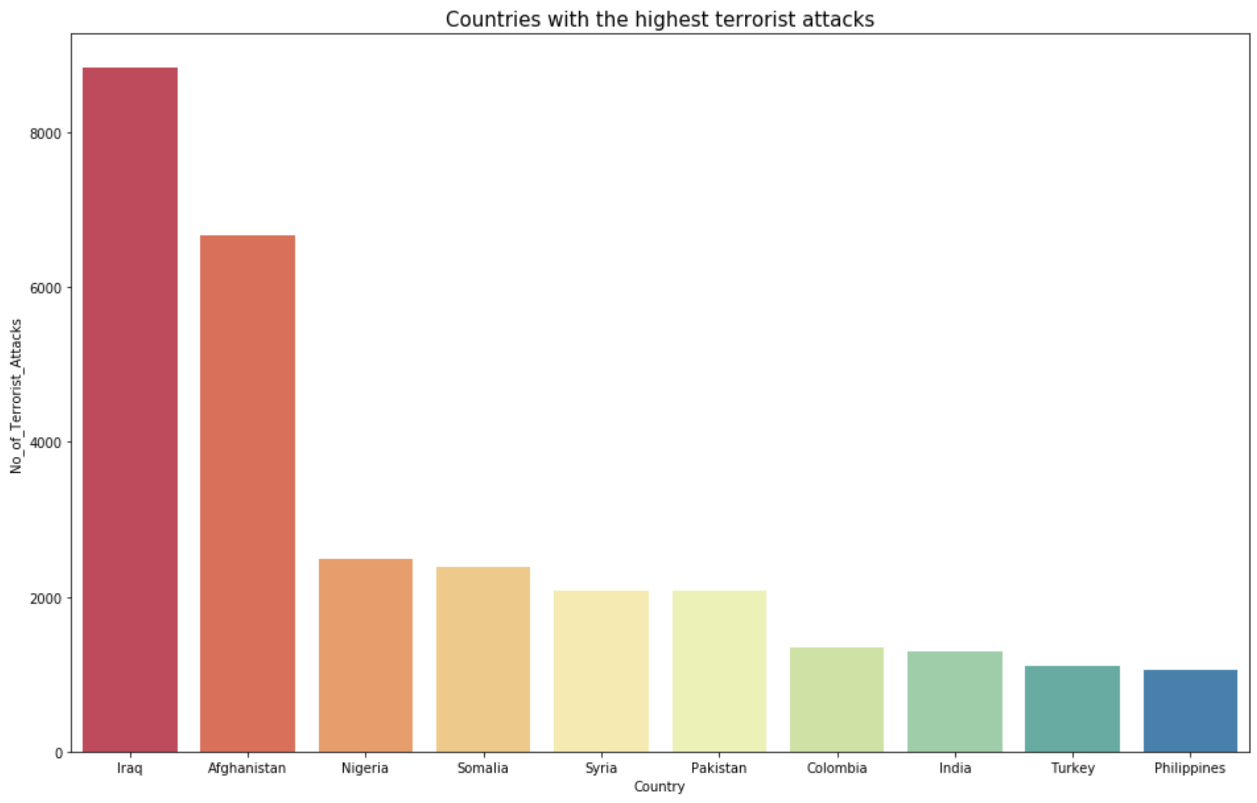


Type *Markdown* and LaTeX: α^2

Top 10 Countries where the most terrorist attack happened

In [1466]:

```python
Country_with_most_terrorist_attacks = TerrorDTframe_notna.groupby(['Location'])
OverallDTFrame=pd.DataFrame()
#Country_with_most_terrorist_attacks = Perpetrators.sort_values(by=['Type'],ascending=False).reset_index(drop=Fals
for key,value in Country_with_most_terrorist_attacks:
    tempDT = pd.DataFrame()
    tempDT = pd.concat([pd.Series(key),pd.Series(value.size)],axis=1)
    OverallDTFrame = pd.concat([OverallDTFrame,tempDT],axis=0)
OverallDTFrame.columns=['Country','No_of_Terrorist_Attacks']
OverallDTFrame = OverallDTFrame.sort_values(by='No_of_Terrorist_Attacks',ascending=False).reset_index(drop=True)
OverallDTFrame.head()
plt.figure(figsize=(16,10))
sns.barplot(x=OverallDTFrame.Country[:10],y=OverallDTFrame.No_of_Terrorist_Attacks[:10],palette='Spectral')
plt.title('Countries with the highest terrorist attacks',fontsize=15)
plt.show()
```



Type *Markdown* and LaTeX: α^2

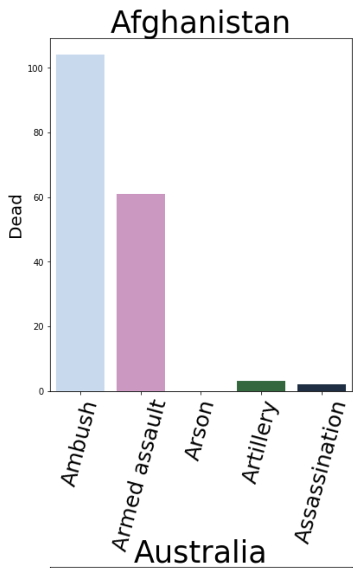
Question 1: What type of terrorist attacks(e.g. shooting,bomb etc.) is the most in which country

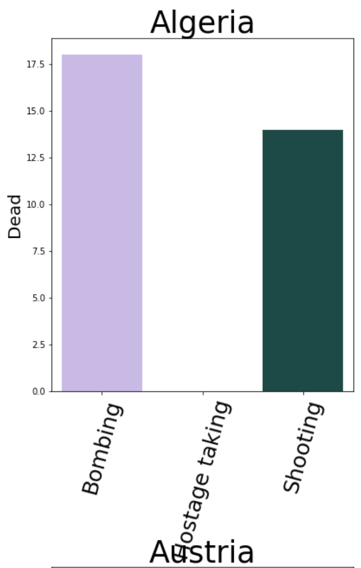
In [1467]:

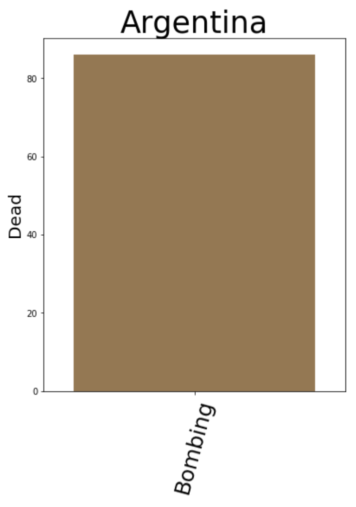
```python
groupByCountry = TerrorDTframe_notna.groupby(['Location','Type']).agg({'Dead':np.sum})
groupByCountry = groupByCountry.reset_index(drop=False)
```

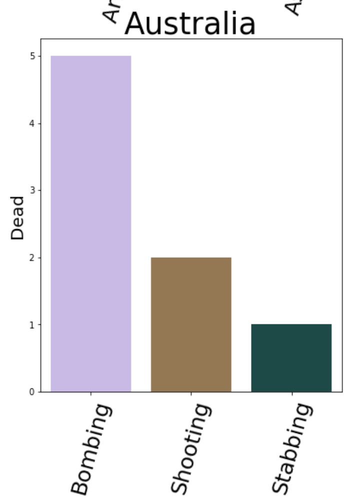
In [1567]:

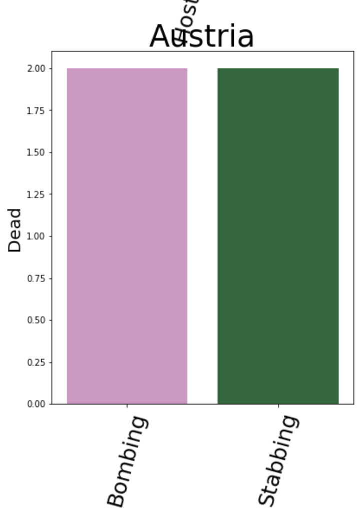
```
i=1
fig=plt.figure(figsize=(25,30))
fig.subplots_adjust(wspace=0.5,hspace=0.5)
for key,value in groupByCountry.groupby('Location'):
    if value.Dead.sum() > 0:
        ax= fig.add_subplot(3,3,i)
        ax = sns.barplot(x=value['Type'][0:5],y=value['Dead'][0:5],palette='cubehelix_r')
        i+=1
        plt.title(key,fontsize=35)
        plt.xticks(rotation=75,fontsize=25)
        plt.ylabel('Dead',fontsize=20)
        plt.xlabel('')
        if i ==10:
            break

```

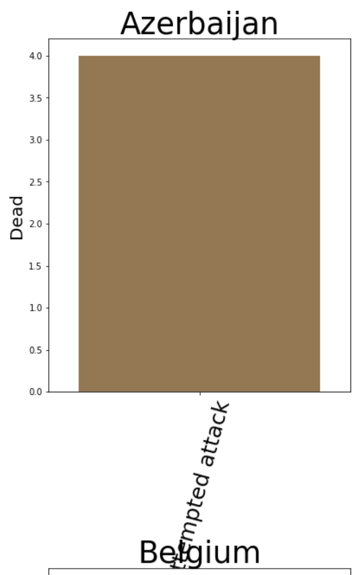


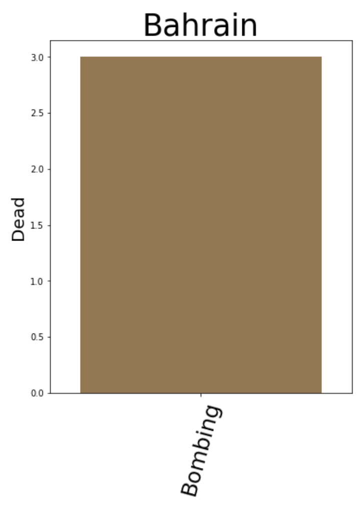


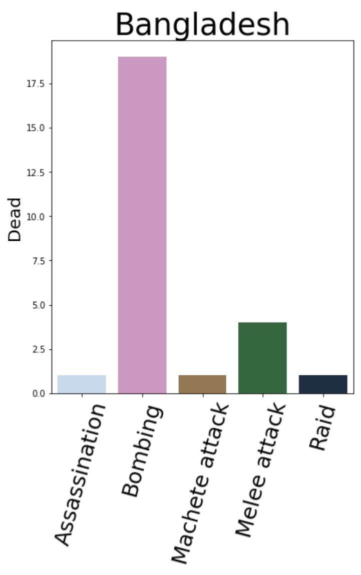


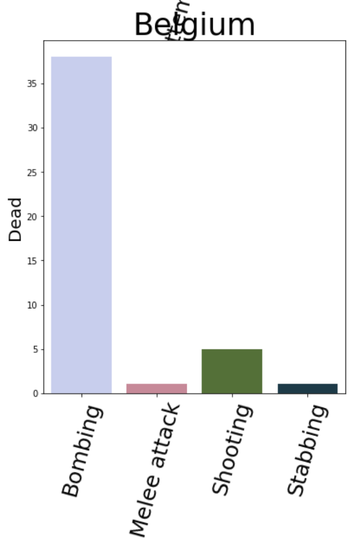












Which terrorist group has caused highest death count in which countries.

In [1469]:
```
import plotly.plotly as py
import pandas as pd
import plotly.figure_factory as ff
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)

df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/2014_world_gdp_with_codes.csv')
df.columns
```

Out[1469]:
```
Index(['COUNTRY', 'GDP (BILLIONS)', 'CODE'], dtype='object')
```

In [1470]:
```
TerrorDTframe_notna.head(1)
```

Out[1470]:

	Date	Type	Dead	Injured	Location	Description	Perpetrators
0	1970	Shooting	7	0	Colombia	7 soldier killed ascribed army 's Ricaurte battalion fell ambush FARC rural area Cimitarra	FARC

In [1471]:
```
k =TerrorDTframe_notna.groupby(['Location','Perpetrators']).agg({'Dead':np.sum})
k.reset_index(drop=False,inplace=True)
```

In [1472]:
```
most_terror_in_country = pd.DataFrame()
for each in TerrorDTframe_notna.Location.unique():
    country_terror = k.groupby('Location').get_group(each).sort_values(by='Dead',ascending=False).reset_index(dro
    most_terror_in_country = pd.concat([most_terror_in_country,country_terror],axis=0)
most_terror_in_country.reset_index(drop=True,inplace=True)
most_terror_in_country.rename(columns={'Location':'COUNTRY'},inplace=True)
most_terror_in_country.head()
```

Out[1472]:

	COUNTRY	Perpetrators	Dead
0	Colombia	FARC	657
1	Canada	Islamic State	1
2	Ireland	PIRA	80
3	United Kingdom	PIRA	90
4	Italy	Unknown	4

In [1473]:
```
df.head()
```

Out[1473]:

	COUNTRY	GDP (BILLIONS)	CODE
0	Afghanistan	21.71	AFG
1	Albania	13.40	ALB
2	Algeria	227.80	DZA
3	American Samoa	0.75	ASM
4	Andorra	4.80	AND

In [1474]:
```
most_terror_in_country_DTFrame = pd.merge(most_terror_in_country,df,how='inner',on='COUNTRY')
most_terror_in_country_DTFrame.head()
```

Out[1474]:

	COUNTRY	Perpetrators	Dead	GDP (BILLIONS)	CODE
0	Colombia	FARC	657	400.1	COL
1	Canada	Islamic State	1	1794.0	CAN
2	Ireland	PIRA	80	245.8	IRL
3	United Kingdom	PIRA	90	2848.0	GBR
4	Italy	Unknown	4	2129.0	ITA

In [1475]:
```
most_terror_in_country_DTFrame['Perpetrators_Name_with_COUNTRY_Name'] = pd.DataFrame(most_terror_in_country_DTFra
most_terror_in_country_DTFrame.head()
```

Out[1475]:

	COUNTRY	Perpetrators	Dead	GDP (BILLIONS)	CODE	Perpetrators_Name_with_COUNTRY_Name
0	Colombia	FARC	657	400.1	COL	FARC::Colombia
1	Canada	Islamic State	1	1794.0	CAN	Islamic State::Canada
2	Ireland	PIRA	80	245.8	IRL	PIRA::Ireland
3	United Kingdom	PIRA	90	2848.0	GBR	PIRA::United Kingdom
4	Italy	Unknown	4	2129.0	ITA	Unknown::Italy

Map showing terrorists responsible for the maximum deaths in the country: Start

In [1569]:
```
data = [dict(
    type='choropleth',
    locations=most_terror_in_country_DTFrame['CODE'],
    z=most_terror_in_country_DTFrame['Dead'],
    text=most_terror_in_country_DTFrame['Perpetrators_Name_with_COUNTRY_Name'],
    colorscale=[[0, "rgb(5, 10, 172)"], [0.4, "rgb(40, 60, 190)"], [0.5, "rgb(70, 100, 245)"],\
                [0.3, "rgb(90, 120, 245)"], [0.5, "rgb(106, 137, 247)"], [1, "rgb(220, 220, 220)"]],
    autocolorscale=False,
    reversescale=True,

    colorbar=dict(
        autotick=True,
        title='Dead_Count'),
)]

layout = dict(
    title='Terrorist who are responsible for the maximum deaths',
    geo=dict(
        showframe=False,
        showcoastlines=False,
    )
)

fig = dict(data=data, layout=layout)
plot(fig,validate=False, filename='d3-world-map-od-max-deaths-by-a-terrorist-group.html')

```

Out[1569]:
```
'file://C:\\Users\\supratik chanda\\Desktop\\Python Tutorial1\\Data Science Final Project\\d3-world-map-od-max-deaths
-by-a-terrorist-group.html'
```

Type *Markdown* and LaTeX: α^2

Which terrorist forces are increasing each year

In [1477]:
```
TerrorDTframe_notna.head()
```

Out[1477]:

	Date	Type	Dead	Injured	Location	Description	Perpetrators
0	1970	Shooting	7	0	Colombia	7 soldier killed ascribed army 's Ricaurte battalion fell ambush FARC rural area Cimitarra	FARC
1	1971	Shooting	10	0	Colombia	site known San Miguel rural Gaitania member FARC attack patrol 23 soldier assigned Caicedo Battalion 10 uniformed men die action	FARC
2	1972	Bombing	1	0	Canada	Cuban official Sergio Pérez Castillo killed explosion Cuban consulate Montreal	Unknown
3	1972	Bombing	9	130	Ireland	Bloody Friday : Nine killed 130 injured Provisional Irish Republican Army set 22 bomb	PIRA
4	1972	Bombing	9	30	United Kingdom	Claudy bombing ; three car bomb detonated Claudy killing nine people group claimed responsibility	PIRA

In [1478]:

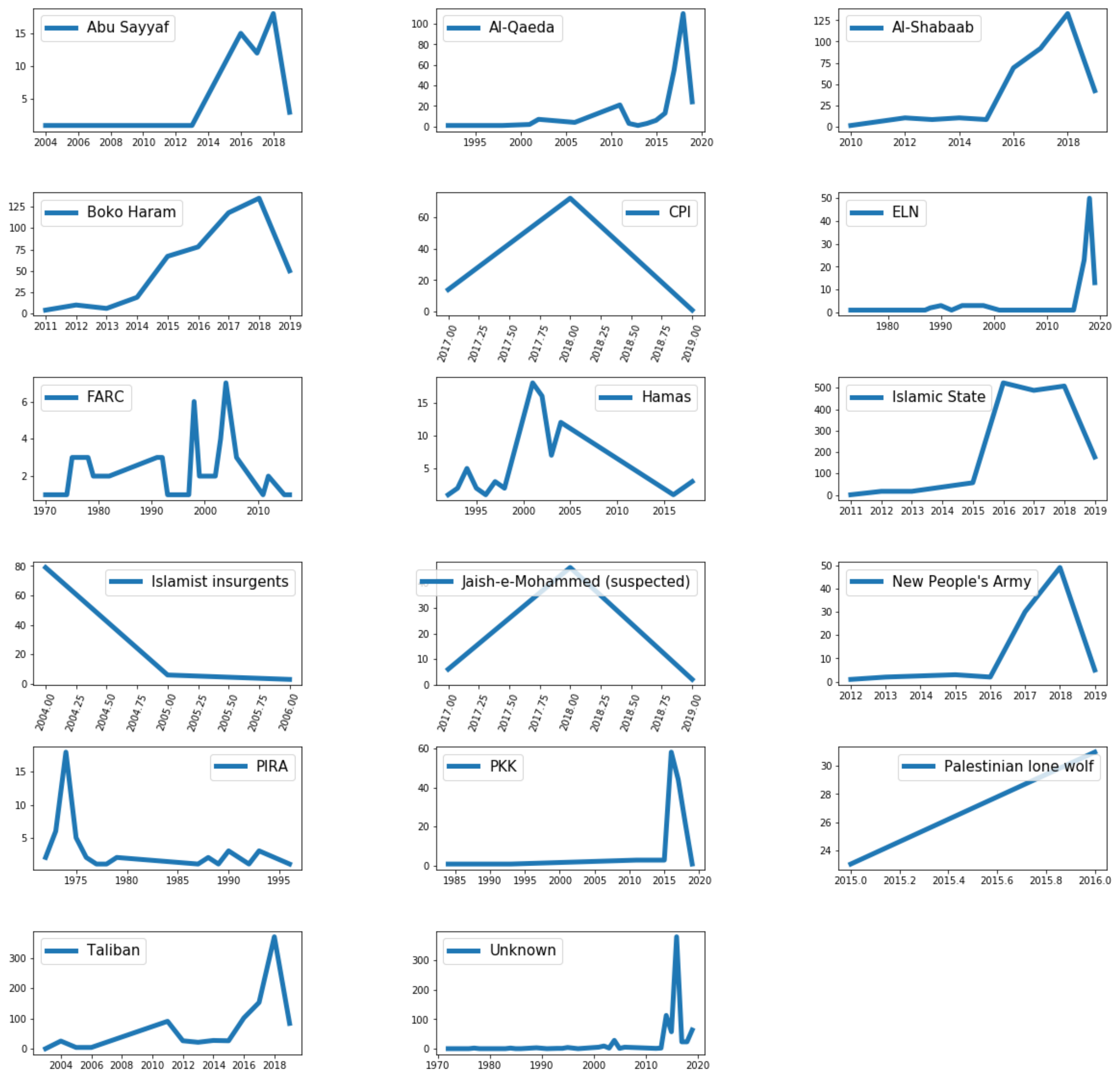
```
group_Of_Perpetrators=pd.DataFrame()
grp_with_highest_attacks = TerrorDTframe_notna.groupby(['Perpetrators','Date'])
for key,value in grp_with_highest_attacks:
    tempDT = pd.concat([pd.Series(key[0]),pd.Series(key[1]),pd.Series(value.Type.size)],axis=1)
    group_Of_Perpetrators = pd.concat([group_Of_Perpetrators,tempDT],axis=0)
group_Of_Perpetrators.columns = ['Perpetrators','Date','Total_Attacks']
group_Of_Perpetrators.reset_index(drop=True,inplace=True)
group_Of_Perpetrators.head()
```

Out[1478]:

	Perpetrators	Date	Total_Attacks
0	Abu Sayyaf	2004	1
1	Abu Sayyaf	2006	1
2	Abu Sayyaf	2011	1
3	Abu Sayyaf	2013	1
4	Abu Sayyaf	2016	15

In [1479]:

```python
fig=plt.figure(figsize=(20,20))
fig.subplots_adjust(wspace=0.5,hspace=0.5)
i=1
for key,value in group_Of_Perpetrators.groupby('Perpetrators'):
        ax=fig.add_subplot(6,3,i)
        ax=plt.plot(value['Date'],value['Total_Attacks'],label=key,lw=5)
        i+=1
        if key in ['Jaish-e-Mohammed (suspected)','Islamist insurgents','CPI']:
            plt.xticks(rotation=70)
        plt.legend(loc='best',fontsize=15)
#plt.show()
```



End

In [1480]:

```python
from sklearn.feature_extraction.text import TfidfVectorizer as tfidf
vectorizer = tfidf(max_features=2000)
X_description = vectorizer.fit_transform(TerrorDTframe_notna.Description)
print(vectorizer.get_feature_names()[0:10])
X_description = X_description.toarray()
```

```
['000', '10', '100', '10th', '11', '12', '120', '13', '130', '14']
```

In [1481]:

```python
dtFrame= pd.DataFrame(X_description)

dtFrame.shape
```

Out[1481]:

```
(5456, 2000)
```

In [1482]:

```python
dtFrame = pd.concat([dtFrame,TerrorDTframe_notna.Type,TerrorDTframe_notna.Location],axis=1)
```

In [1483]:

```python
print(dtFrame.shape)
dtFrame.head()
```

```
(5456, 2002)
```

Out[1483]:

	0	1	2	3	4	5	6	7	8	9	...	1992	1993	1994	1995	1996	1997	1998	1999	Type	Location
0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Shooting	Colombia
1	0.0	0.195708	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Shooting	Colombia
2	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Bombing	Canada
3	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.420788	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Bombing	Ireland
4	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	Bombing	United Kingdom

5 rows × 2002 columns

In [1500]:

```python

X=pd.DataFrame()
X = pd.get_dummies(data = dtFrame,columns=['Type','Location'])

```

In [1501]:

```python
X= pd.concat([X,TerrorDTframe_notna.Perpetrators],axis=1)
X.head()
```

Out[1501]:

	0	1	2	3	4	5	6	7	8	9	...	Location_Transnistria	Location_Tunisia	Location_Turkey	Location_Ukraine	Locat
0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0	0	0	0	
1	0.0	0.195708	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0	0	0	0	
2	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0	0	0	0	
3	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.420788	0.0	...	0	0	0	0	
4	0.0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.000000	0.0	...	0	0	0	0	

5 rows × 2192 columns

In [1502]:

```python
X = X[X.Perpetrators.notna()]
X.shape
```

Out[1502]:

```
(5456, 2192)
```

In [1498]:

```python
X.to_csv('RefinedTerrorData.csv',index=False)
```

Using Logistic Regression

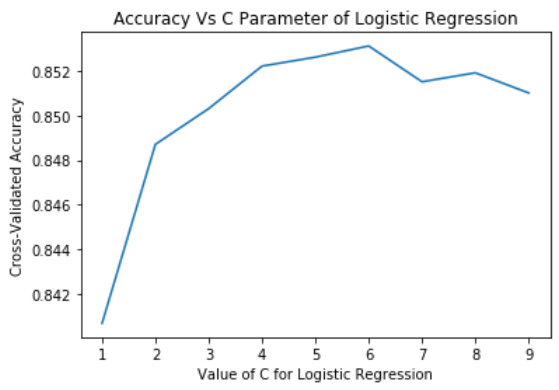
In [1508]:

```python
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV,RandomizedSearchCV,train_test_split
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
lm = LogisticRegression()
```

Using Cross Validation with Different values of C

In [1543]:
```python
from sklearn.model_selection import cross_val_score
train_set,test_set = train_test_split(X,test_size=0.2,random_state=0)
C_List=pd.Series()
Acc_List=pd.Series()
DTFrame=pd.DataFrame()
for i in range(1,10,1):
    lmNew = LogisticRegression(C=i)
    crossval = cross_val_score(lmNew,train_set.iloc[:,:-1],train_set.iloc[:,-1],scoring='accuracy',cv=5)
    #print(i,crossval.mean())
    C_List = C_List.append(pd.Series(i))
    Acc_List = Acc_List.append(pd.Series(round(crossval.mean(),4)))
DTFrame= pd.concat([C_List,Acc_List],axis=1,keys=['C_values','Accuracy_values'])
DTFrame['Accuracy_values'] = DTFrame['Accuracy_values'].apply(lambda x : str(x).replace('$',''))
sns.lineplot(x=C_List,y=Acc_List)
plt.xlabel('Value of C for Logistic Regression')
plt.ylabel('Cross-Validated Accuracy')
plt.title('Accuracy Vs C Parameter of Logistic Regression')
```

Out[1543]:
```
Text(0.5, 1.0, 'Accuracy Vs C Parameter of Logistic Regression')
```



In [1540]:
```python
import warnings
warnings.simplefilter('ignore')
import time
start_time = time.time()
train_set,test_set = train_test_split(X,test_size=0.2,random_state=0)
param_grid = dict(C=[6],penalty=['l1','l2'],random_state=[0,5,16,27])
rdSearchCV = RandomizedSearchCV(lm,param_grid,cv=5).fit(train_set.iloc[:,:-1],train_set.iloc[:,-1])
print('For RandomizedSearchCV (LOGISTIC REGRESSION):')
print('grid best score for train_set: for random_state: ',rdSearchCV.best_score_)
print('grid best parameters for train_set: ',rdSearchCV.best_params_)
print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
For RandomizedSearchCV(LOGISTIC REGRESSION):
grid best score for train_set: for random_state:  0.8531164069660861
grid best parameters for train_set:  {'random_state': 0, 'penalty': 'l2', 'C': 6}
Execution time: 44.846903800964355 ms
```

In [1541]:
```python
print('Test_accuracy: ',rdSearchCV.score(test_set.iloc[:,:-1],test_set.iloc[:,-1]))
```

```
Test_accuracy:  0.8626373626373627
```

In [1527]:

```
from sklearn.metrics import confusion_matrix
lm=LogisticRegression(C=1,penalty='l1',random_state=0)
lm.fit(train_set.iloc[:,:-1],train_set.iloc[:,-1])
cm = confusion_matrix(lm.predict(test_set.iloc[:,:-1]),test_set.iloc[:,-1])
display(cm)
```

```
array([[  9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,  47,   0,   0,   0,   0,   0,   0,   5,   0,   0,   0,   0,
          0,   0,   0,   5],
       [  0,   1,  83,   0,   0,   0,   0,   0,   4,   0,   0,   0,   0,
          0,   0,   0,   4],
       [  0,   1,   0,  93,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   4],
       [  0,   0,   0,   0,  19,   0,   0,   0,   1,   0,   3,   0,   0,
          0,   0,   0,   4],
       [  0,   0,   0,   0,   0,  20,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   1],
       [  0,   0,   0,   0,   0,   0,   8,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  16,   0,   0,   0,   0,   0,
          0,   1,   0,   2],
       [  0,   2,   0,   0,   0,   0,   2,   0, 319,   0,   0,   0,   0,
          0,   0,   3,  37],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,  15,   0,   0,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  10,   0,   0,
          0,   0,   0,   2],
       [  2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  15,   0,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11,
          0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   3,   0,   0,   0,   0,
         29,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   9,   0,   0],
       [  0,   2,   0,   0,   0,   0,   0,   0,  12,   0,   0,   0,   0,
          0,   0, 184,  18],
       [  1,   3,   0,   0,   0,   0,   0,   0,  14,   1,   0,   0,   0,
          0,   0,   4,  63]], dtype=int64)
```

In [1529]:

```
from sklearn.metrics import classification_report as cr
print(cr(lm.predict(test_set.iloc[:,:-1]),test_set.iloc[:,-1]))
```

```
                         precision    recall  f1-score   support

             Abu Sayyaf       0.75      1.00      0.86         9
               Al-Qaeda       0.84      0.82      0.83        57
             Al-Shabaab       1.00      0.90      0.95        92
            Boko Haram       1.00      0.95      0.97        98
                    CPI       1.00      0.70      0.83        27
                    ELN       1.00      0.95      0.98        21
                   FARC       0.80      1.00      0.89         8
                  Hamas       1.00      0.84      0.91        19
         Islamic State       0.89      0.88      0.88       363
    Islamist insurgents       0.94      1.00      0.97        15
Jaish-e-Mohammed (suspected)       0.77      0.83      0.80        12
      New People's Army       1.00      0.88      0.94        17
                   PIRA       1.00      1.00      1.00        11
                    PKK       1.00      0.91      0.95        32
    Palestinian lone wolf       0.90      1.00      0.95         9
                Taliban       0.96      0.85      0.90       216
                Unknown       0.45      0.73      0.56        86

              micro avg       0.87      0.87      0.87      1092
              macro avg       0.90      0.90      0.89      1092
           weighted avg       0.90      0.87      0.88      1092
```