1/3/2022

# Prob & Stats. Project Fall 2021 Semester 5

## Graphical Analysis and Modelling on Terrorism in Different States Across the Globe and Prediction on Ongoing Terrorism Attacks


| | | |
|---|---|---|
| Muhammad Abdullah | 19F-0916 | SE 5A |
| Mahnoor | 19F-0965 | SE 5A |
| Fareed | 19F-0148 | CS |
| Ali Tahir | 19F-0351 | CS |

## *Introduction:*

This project is based upon dataset of Terrorism throughout different states around the globe. It includes number of incidents occurred and number of people injured/killed in those incidents. It also covers total fatalities upon range of some given years as well as causalities above 100 too. This dataset explains the death percentage according to different years which helps to understand the hierarchy of most to least dangerous countries list.

## *Tool Used:*

Following tools are used in this project:

- PyCharm For Python.
- Pandas For Python Library.
- Numpy For Python Library.
- Skleanr For Python Library.
- Matplotlib for Python Library.
- Excel For CSV Files of Dataset.

## *Prob & Stats. Function Used:*

Following tools are used in this project:

- Frequency Distribution
- Bar Charts
- Line Graphs
- Box Plot
- Histogram
- Pie Charts
- Hypergeometric Distribution
- Linear Regression Modelling

## *Project's Code:*

```
import time
import pandas as pd
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt
from tkinter import ttk
from tkinter import *

df_terrorismInState = pd.read_csv("terrorism1.csv")
df_terrorismInCountry = pd.read_csv("terrorism2.csv")
df_terrorismInYear = pd.read_csv("terrorism3.csv")
df_terrorismMoreThan10 = pd.read_csv("terrorism4.csv")
print(df_terrorismInState.shape)
print(df_terrorismInCountry.shape)
print(df_terrorismInYear.shape)
print(df_terrorismMoreThan10.shape)
splashWindow = Tk()

appWidth = 720
appHeight = 680
screenWidth = splashWindow.winfo_screenwidth()
screenHeight = splashWindow.winfo_screenheight()

xCoordinates = (screenWidth / 2) - (appWidth / 2)
yCoordinates = (screenHeight / 2) - (appHeight / 2)

splashWindow.overrideredirect(True)
splashWindow.geometry(f'{appWidth}x{appHeight}+{int(xCoordinates)}+{int(yCoor
dinates)}')
splashBackground = PhotoImage(file="images/Probability and Statistics.png")
pictureLabel = Label(splashWindow, image=splashBackground)
pictureLabel.place(x=0, y=0, relwidth=1, relheight=1)

progressBar = ttk.Progressbar(splashWindow, orient=HORIZONTAL, length=300,
mode='determinate')
progressBar.place(x=170, y=450, relwidth=0.5, relheight=0.05)


def progress():
    progressBar.start(20)
    for x in range(5):
        progressBar['value'] += 20
        splashWindow.update_idletasks()
        time.sleep(1)


def frequencyDistributionFunc():
    Years = df_terrorismInCountry['iyear']
    Country = df_terrorismInCountry['United Kingdom']
    df = pd.Series({'Year': [Years],
                    'Country': [Country]})
    df.value_counts(sort=False)
    print(df)
```

```
def barCharFunc1():
    plt.bar(df_terrorismInState['Number of Incidents'],
df_terrorismInState['Number Killed'])
    plt.title("Number of Incidents vs Number of People Killed in Terrorism")
    plt.show()


def barCharFunc2():
    plt.bar(df_terrorismInState['Number of Incidents'],
df_terrorismInState['Number Injured'])
    plt.title("Number of Incidents vs Number of People Injured in Terrorism")
    plt.show()


def boxPlotFunc1():
    df_terrorismInState.boxplot(column='Number of Incidents')
    plt.title("Number of Incidents in Respective Countries")
    plt.show()


def boxPlotFunc2():
    df_terrorismInState.boxplot(column='Number Killed')
    plt.title("Number of People Killed in Respective Countries")
    plt.show()


def boxPlotFunc3():
    df_terrorismInState.boxplot(column='Number Injured')
    plt.title("Number of People Injured in Respective Countries")
    plt.show()


def histogramFunc1():
    df_terrorismInState.hist(column=['Number of Incidents'], bins=10)
    plt.title("Number of Incidents Occurred")
    plt.show()


def histogramFunc2():
    df_terrorismInState.hist(column=['Number Killed'], bins=10)
    plt.title("Number of People Killed")
    plt.show()


def histogramFunc3():
    df_terrorismInState.hist(column=['Number Injured'], bins=10)
    plt.title("Number of People Injured")
    plt.show()


def pieChartFunc1():
    pieChartLabels = df_terrorismInYear['iyear']
    plt.pie(df_terrorismInYear['fatalities'], labels=pieChartLabels)
    plt.title("More than 10 People Killed in Respective Years")
    plt.show()
```

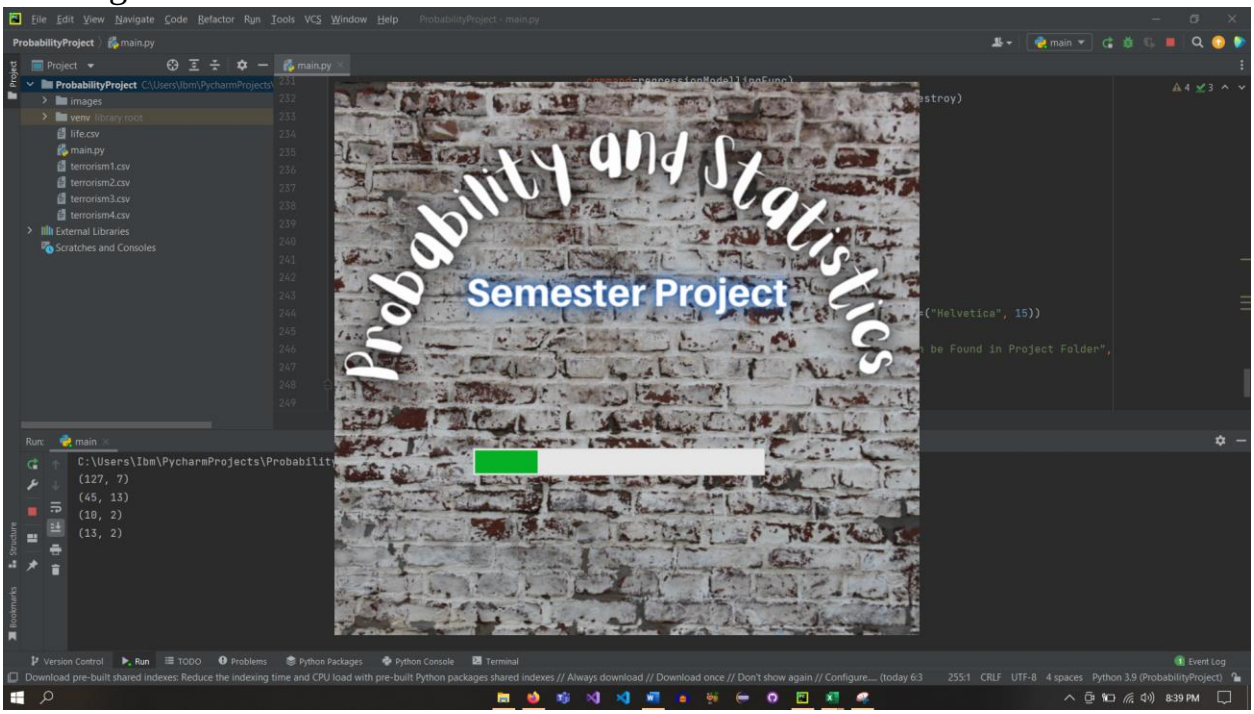
```
def pieChartFunc2():
    pieChartLabels = df_terrorismMoreThan10['Country']
    plt.pie(df_terrorismMoreThan10['Killed More than 100'],
labels=pieChartLabels)
    plt.title("More than 10 People Killed in Respective Years")
    plt.show()


def lineGraphFunc1():
    plt.plot(df_terrorismInState['Number of Incidents'],
df_terrorismInState['Number Killed'])
    plt.title('Number of Incidents vs Number of People Killed')
    plt.xlabel('Total Incidents ')
    plt.ylabel('People Killed')
    plt.show()


def lineGraphFunc2():
    plt.plot(df_terrorismInState['Number of Incidents'],
df_terrorismInState['Number Injured'])
    plt.title('Number of Incidents vs Number of People Injured')
    plt.xlabel('Total Incidents ')
    plt.ylabel('People Injured')
    plt.show()


def hyperGeometricFunc():
    totalCountN = 0
    totalCountries = df_terrorismInState['Country']
    for x in totalCountries:
        totalCountN += 1

    safeCountA = 0
    totalSafeCountries = df_terrorismInState['Number of Incidents']
    for x in totalSafeCountries:
        if x < 50:
            safeCountA += 1

    safeCountn = 0
    totalSafeCountries = df_terrorismInState['Number Killed']
    for x in totalSafeCountries:
        if x < 5:
            safeCountn += 1

    x = 10

    s = np.random.hypergeometric(totalCountN, safeCountA, safeCountn, x)
    plt.hist(s)
    plt.title("Total Countries and Safe to Live Countries upon X = 10")
    plt.show()


def regressionModellingFunc():
    xValue = df_terrorismInState[['Number of Incidents', 'Number Killed',
'Number Injured']]
```

```
    yValue = df_terrorismInState['Number US Killed']
    regression = linear_model.LinearRegression()
    regression.fit(xValue, yValue)

    incidents = 0
    totalSafeCountries = df_terrorismInState['Number of Incidents']
    for x in totalSafeCountries:
        incidents += 1

    killed = 0
    totalSafeCountries = df_terrorismInState['Number Killed']
    for x in totalSafeCountries:
        killed += 1

    injured = 0
    totalSafeCountries = df_terrorismInState['Number Injured']
    for x in totalSafeCountries:
        injured += 1

    print(regression.predict([[incidents, killed, injured]]))


def BarCharFuncCollection():
    barCharFunc1()
    barCharFunc2()


def boxPlotFuncCollection():
    boxPlotFunc1()
    boxPlotFunc2()
    boxPlotFunc3()


def histogramFuncCollection():
    histogramFunc1()
    histogramFunc2()
    histogramFunc3()


def pieChartFuncCollection():
    pieChartFunc1()
    pieChartFunc2()


def lineGraphFuncCollection():
    lineGraphFunc1()
    lineGraphFunc2()


def mainWindow():
    splashWindow.destroy()
    rootWindow = Tk()
    rootWindow.title("Probability Project")

rootWindow.geometry(f'{appWidth}x{appHeight}+{int(xCoordinates)}+{int(yCoordi
nates)}')
```

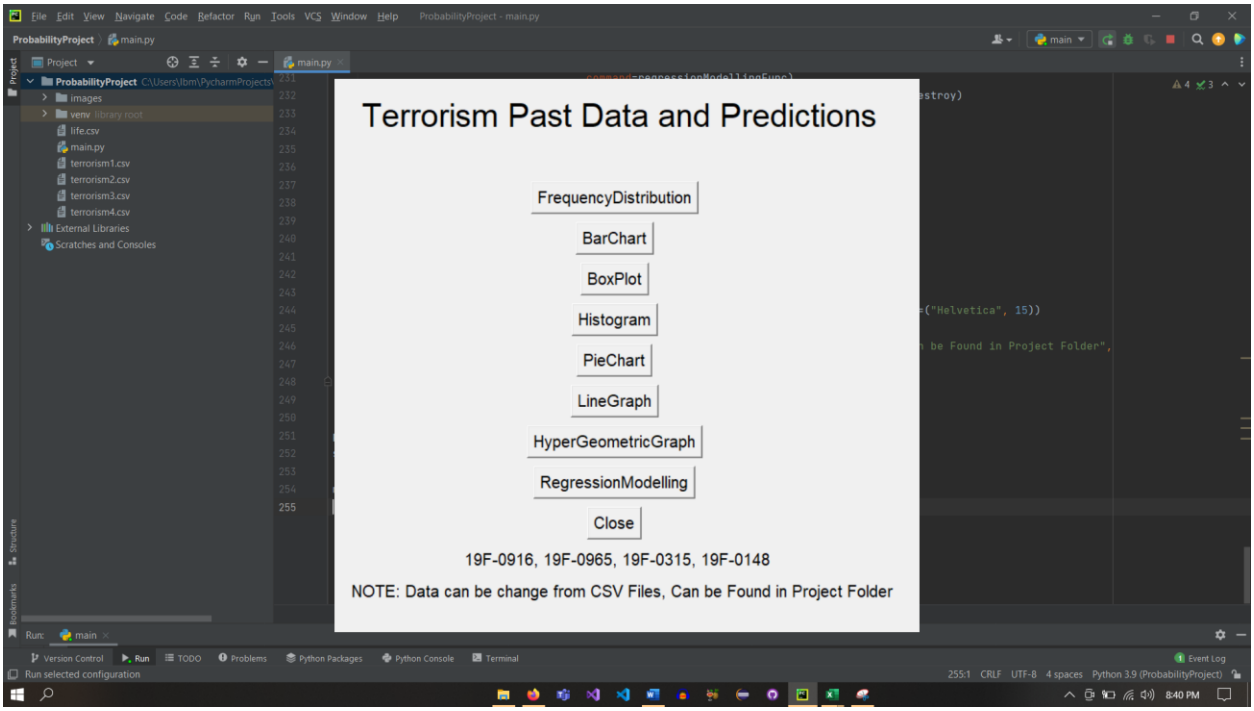
```
    rootWindow.overrideredirect(True)

    mainScreenLabel = Label(rootWindow, text=" Terrorism Past Data and
Predictions", font=("Helvetica", 30))
    mainScreenLabel.grid(row=0, column=0, pady=20, padx=20)
    frame = Frame(rootWindow)
    frame.grid(row=1, column=0, pady=30)

    frequencyDistributionButton = Button(frame, text="FrequencyDistribution",
font=("Helvetica", 15),
                                         command=frequencyDistributionFunc)
    barChartButton = Button(frame, text="BarChart", font=("Helvetica", 15),
command=BarCharFuncCollection)
    boxPlotButton = Button(frame, text="BoxPlot", font=("Helvetica", 15),
command=boxPlotFuncCollection)
    histogramButton = Button(frame, text="Histogram", font=("Helvetica", 15),
command=histogramFuncCollection)
    pieChartButton = Button(frame, text="PieChart", font=("Helvetica", 15),
command=pieChartFuncCollection)
    lineGraphButton = Button(frame, text="LineGraph", font=("Helvetica", 15),
command=lineGraphFuncCollection)
    hyperGeometricButton = Button(frame, text="HyperGeometricGraph",
font=("Helvetica", 15),
                                  command=hyperGeometricFunc)
    regressionModellingButton = Button(frame, text="RegressionModelling",
font=("Helvetica", 15),
                                       command=regressionModellingFunc)
    closeButton = Button(frame, text="Close", font=("Helvetica", 15),
command=rootWindow.destroy)

    frequencyDistributionButton.grid(row=0, column=0, pady=5)
    barChartButton.grid(row=1, column=0, pady=5)
    boxPlotButton.grid(row=2, column=0, pady=5)
    histogramButton.grid(row=3, column=0, pady=5, padx=5)
    pieChartButton.grid(row=4, column=0, pady=5)
    lineGraphButton.grid(row=5, column=0, pady=5, padx=5)
    hyperGeometricButton.grid(row=6, column=0, pady=5, padx=5)
    regressionModellingButton.grid(row=7, column=0, pady=5, padx=5)
    closeButton.grid(row=8, column=0, pady=5)

    mainScreenLabelEnd = Label(frame, text=" 19F-0916, 19F-0965, 19F-0315,
19F-0148", font=("Helvetica", 15))
    mainScreenLabelEnd.grid(row=9, column=0, pady=5)
    mainScreenLabelEnd = Label(frame, text="   NOTE: Data can be change from
CSV Files, Can be Found in Project Folder",
                               font=("Helvetica", 15))
    mainScreenLabelEnd.grid(row=10, column=0, pady=5)


progress()
splashWindow.after(3500, mainWindow)

mainloop()
```

## *Description:*

Loading Screen




Main Screen



## *Frequency distribution:*

```
Year       [[1970, 1971, 1972, 1973, 1974, 1975, 1976, 19
Country    [[20, 110, 368, 210, 234, 245, 264, 103, 81, 1
```

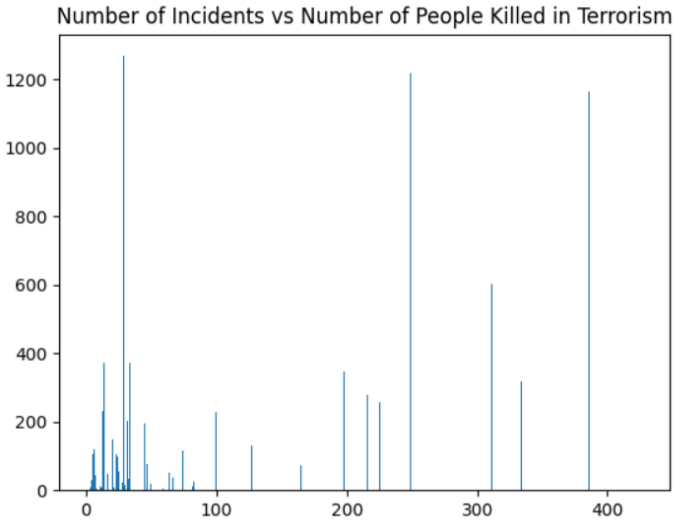
This shows us the years with respective to total number of attacks collectively in United Kingdom. This frequency distribution can be applied on any Country.

## *Bar chart:*

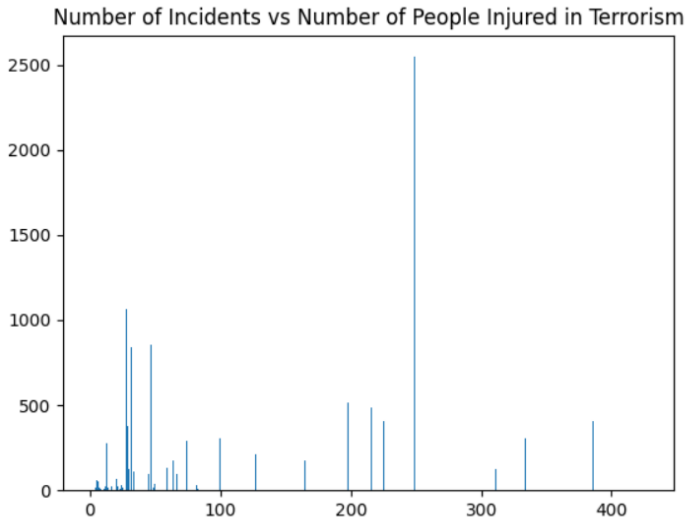
```
def barCharFunc():

    plt.bar(df_terrorismInState['Number of Incidents'], df_terrorismInState['Number Killed'])

    plt.title("Number of Incidents vs Number of People Killed in Terrorism")

    plt.show()
```

**Number of incident vs**

**Number of people killed in terrorism:**




**Number of people injured in terrorism:**




Based on the bar chart above, we can see that our dataset has variation in number of incidents vs number of killed and injured people in different countries due to terrorist attacks.

- In the bar chat at left we see comparison of people killed in terrorism in 1200+

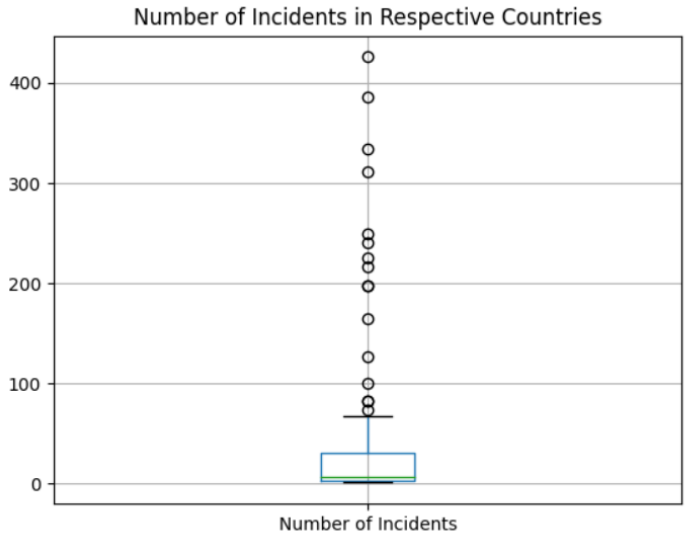
- In the bar chart at right we see people injured in incidents variating and going up to 2500+.
- In conclusion number of incidents have caused deaths at various incidents but number of injured people are more than death at most incidents.

## *Box plot:*

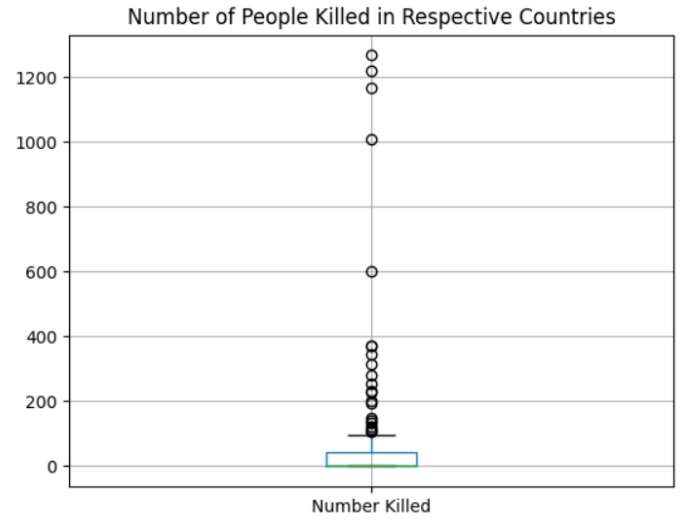
Based on the box plot we can campare all the box plots with each other and observe the distribution of data informeing the number of incidents, number of people injured and number of people killed in respective countries.

**Number of incidents in repective countries:**

- From this plot we identify that our interqurtial range is between 0 to approximatly 85.
- The incidents outliers range from 85 till 400+.

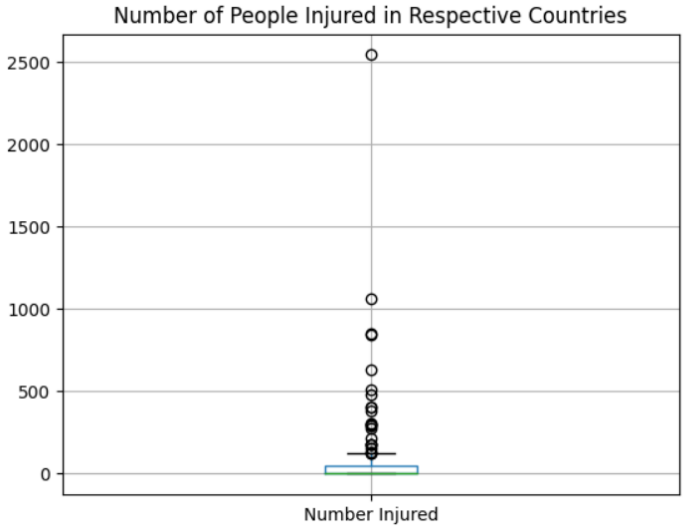


**Number of people killed in repective countries:**




- From this plot we identify that our interqurtial range is between 0 to approximatly 100+.
- The incidents outliers range from 100+ till 1200+.

**Number of people injured in repective countries:**



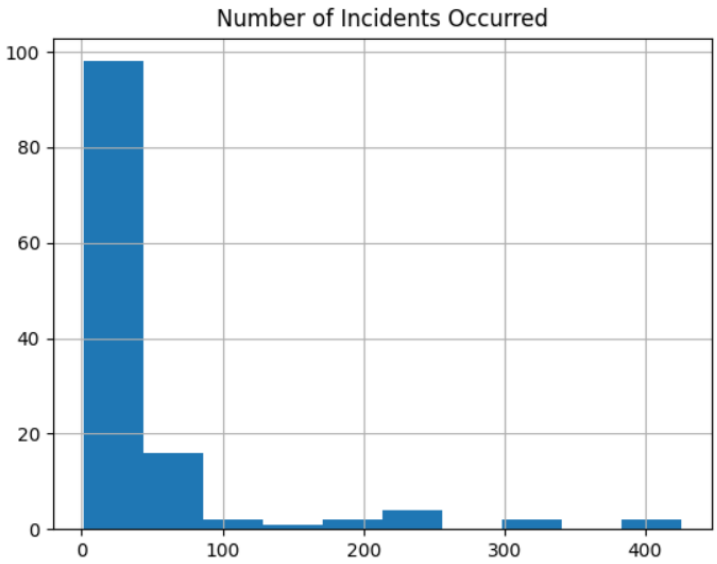

- From this plot we identify that our interqurtial range is between 0 to approximatly 500.
- The incidents outliers range from 500 till 1250 anf going to 2500 for specific country.

```
def boxPlotFunc():
df_terrorismInState.boxplot(column='Number Injured')
plt.title("Number of People Injured in Respective Countries")
plt.show()
```

The interquartile range observed helps us identify outliers in these graphs. We conclude that most incidents, deaths, and injuries occur are outliers means our dataset is dispersed.

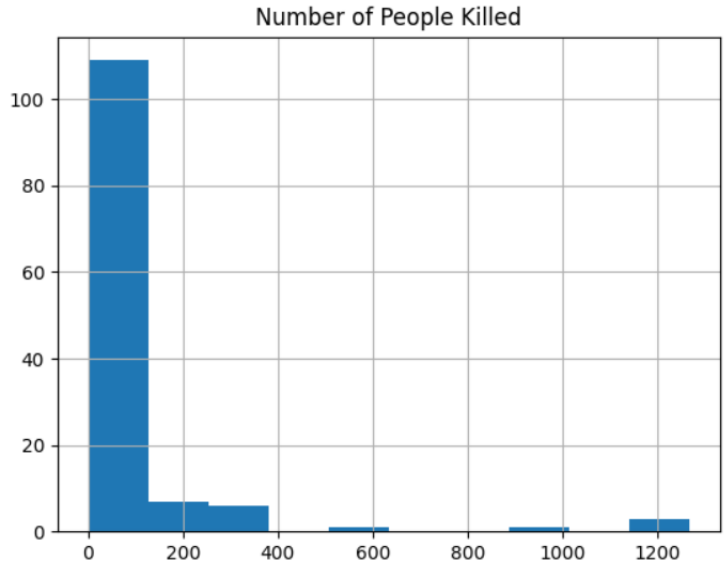
## *Histogram:*

```
def histogramFunc():
    df_terrorismInState.hist(column=['Number of Incidents'], bins=10)
    plt.title("Number of Incidents Occurred")
    plt.show()
```




**Number of incidents occurred**

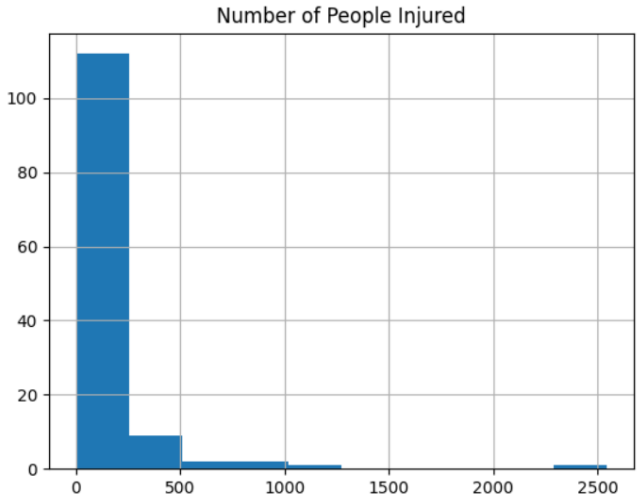
➢ Lesser the amount of vessels are colored, minimum the chances are that the incidents took place, as evident from the graph plotted.




**Number of people killed:**

➢ Lesser the amount of vessels are colored, minimum the chances are that the person were killed in incidents, as

evident from the graph plotted.




**Number of people injured:**

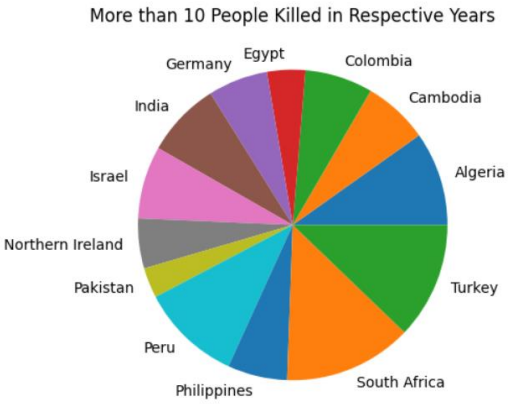
➢Lesser the amount of vessels are colored, minimum the chances are that the person being injured in incidents, as evident from the graph plotted.

## *Pie chart:*

Pie chart makes it easy to display and understand the distribution of people killed in terrorist attacks with respect to the countries and years.

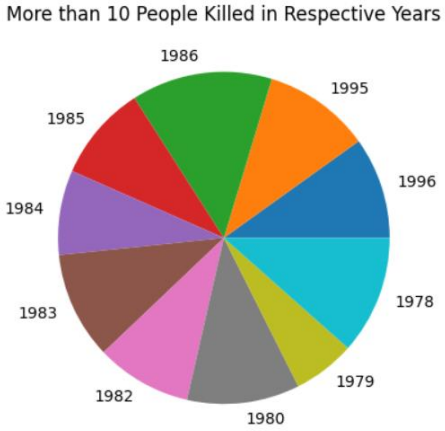
```
def pieChartFunc():
    pieChartLabels = df_terrorismInState['Country']
    plt.pie(df_terrorismInState['Number US Killed'], labels=pieChartLabels)
    plt.title("PieChart for Killed People")
    plt.show()
```

**More than 10 people killed with respective to country:**




➢ The pie chart indicates that if 10 people died in Egypt then visibly we can conclude more than twice people have died in south Africa due to terrorist attack.

**More than 10 people killed with respective to year:**



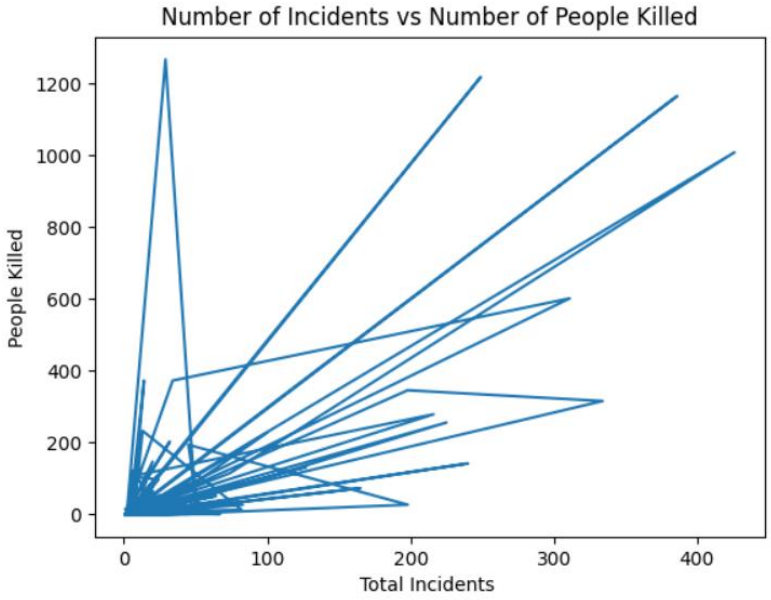

➢ The pie chart indicates that if 10 people died in 1979 then visibly we can conclude more than twice people have died in 1986 due to terrorist attack.

## *Line graph:*

These line graphs are representing information that is changes, displaying the comparison between two variables; incidents and people.

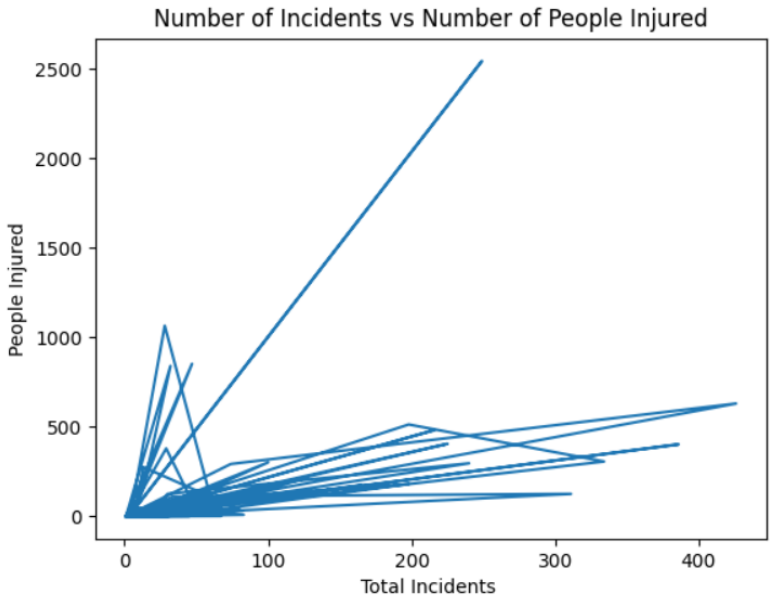
```
def lineGraphFunc():
    plt.plot(df_terrorismInState['Number of Incidents'],
df_terrorismInState['Number Killed'])
    plt.title('Number of Incidents vs Number of People Killed')
    plt.xlabel('Total Incidents ')
    plt.ylabel('People Killed')
    plt.legend()
    plt.show()
```

**Number of incident vs Number of peoplr killed:**




- Thought this line graph we can see how the number of incidents had resulted in killing of people in attacks.
- We can clearly see variation between incidents and number of people killed in them.

**Number of incident vs Number of peoplr injured:**



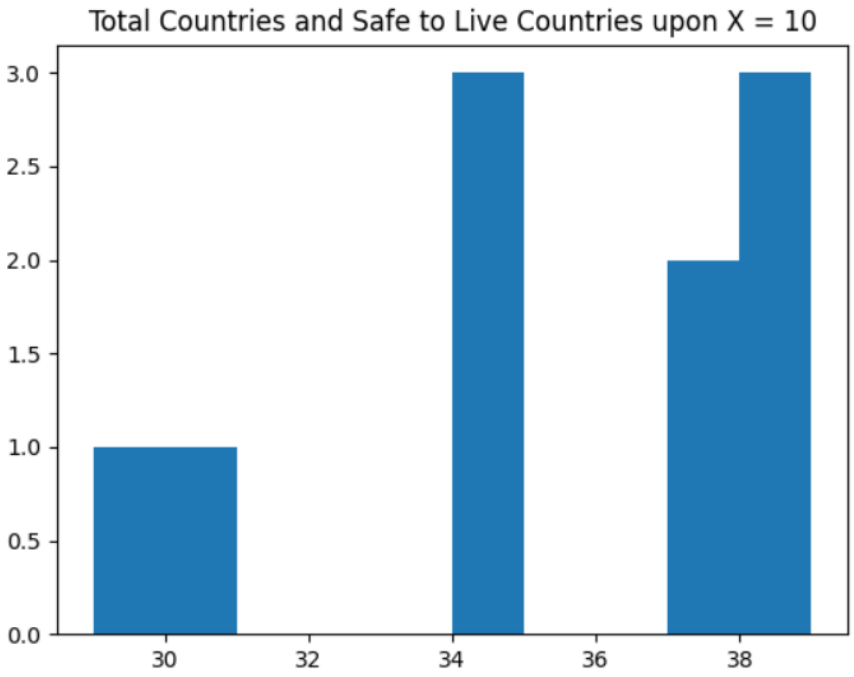

- Thought this line graph we can see how the number of incidents had resulted in injuring of people in attacks.
- We can clearly see that number of injured people alternates at some incidents more than other incidents.

### *Hypergeometric graph:*

We can determine the probability of obtaining a certain number of countries that are safe to live in without replacing from a total number of countries.

```
def hyperGeometricFunc():
    totalCountN = 0
    totalCountries = df_terrorismInState['Country']
    for x in totalCountries:
        totalCountN += 1


    safeCountA = 0
    totalSafeCountries = df_terrorismInState['Number of Incidents']
    for x in totalSafeCountries:
        if x < 50:
            safeCountA += 1


    safeCountn = 0
    totalSafeCountries = df_terrorismInState['Number Killed']
    for x in totalSafeCountries:
        if x < 5:
            safeCountn += 1


    x = 10


    s = np.random.hypergeometric(totalCountN, safeCountA, safeCountn, x)
    plt.hist(s)
    plt.title("Total Countries and Safe to Live Countries upon X = 10")
```

```
plt.show()
```


Total Countries and Safe to Live Countries upon X = 10
3.0
2.5
2.0
1.5
1.0
0.5
0.0
30
32
34
36
38

## *Regression model value:*

```
def regressionModellingFunc():
    xValue = df_terrorismInState[['Number of Incidents', 'Number Killed', 'Number Injured']]
    yValue = df_terrorismInState['Number US Killed']
    regression = linear_model.LinearRegression()
    regression.fit(xValue, yValue)

    incidents = 0
    totalSafeCountries = df_terrorismInState['Number of Incidents']
    for x in totalSafeCountries:
        incidents += 1

    killed = 0
    totalSafeCountries = df_terrorismInState['Number Killed']
    for x in totalSafeCountries:
        killed += 1

    injured = 0
    totalSafeCountries = df_terrorismInState['Number Injured']
    for x in totalSafeCountries:
        injured += 1

    print(regression.predict([[incidents, killed, injured]]))
```

```
[0.50439439]
```

We can predict from the model value that in coming years 50.43% chance of people dying due to terrorist attacks in countries we have collected dataset from.

## *Conclusion:*

For all the data observation we get following conclusion.

- All countries had different incidents at different years resulting in different number of deaths and injured people.
- All this observation showed us alternate in values and helped us in predicting that is there are 50% chance of people dying in terrorist attacks that might take place in countries on high risk.
- This observation also helps us to predict the most volatile countries, facing incidents and causalities over the years.