CS 210 Project Step 2

Data Exploration, Hypothesis Testing and Single Linear Regression

1-Data Exploration

1.0.1 Shape, Data Types and NaN Values

```
import numpy as np
import pandas as pd
import csv
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.formula.api import ols
from scipy import stats
import seaborn as sns
from sklearn.linear_model import LinearRegre
plt.style.use('seaborn-whitegrid')
%matplotlib inline
from scipy.stats import ttest_ind
```

```
In [3]: df.shape
Out[3]: (181691, 135)
```

```
In [4]: df.columns.values
Out[4]: array(['eventid', 'iyear', 'imonth', 'iday', 'approxdate', 'extended',
               'resolution', 'country', 'country_txt', 'region', 'region_txt',
               'provstate', 'city', 'latitude', 'longitude', 'specificity',
               'vicinity', 'location', 'summary', 'crit1', 'crit2', 'crit3',
               'doubtterr', 'alternative', 'alternative_txt', 'multiple',
               'success', 'suicide', 'attacktype1', 'attacktype1_txt',
               'attacktype2', 'attacktype2_txt', 'attacktype3', 'attacktype3_txt',
               'targtype1', 'targtype1_txt', 'targsubtype1', 'targsubtype1_txt',
               'corp1', 'target1', 'natlty1', 'natlty1_txt', 'targtype2',
               'targtype2_txt', 'targsubtype2', 'targsubtype2_txt', 'corp2',
               'target2', 'natlty2', 'natlty2_txt', 'targtype3', 'targtype3_txt',
               'targsubtype3', 'targsubtype3_txt', 'corp3', 'target3', 'natlty3',
               'natlty3_txt', 'gname', 'gsubname', 'gname2', 'gsubname2',
               'gname3', 'gsubname3', 'motive', 'guncertain1', 'guncertain2',
               'guncertain3', 'individual', 'nperps', 'nperpcap', 'claimed',
               'claimmode', 'claimmode_txt', 'claim2', 'claimmode2',
               'claimmode2_txt', 'claim3', 'claimmode3', 'claimmode3_txt',
               'compclaim', 'weaptype1', 'weaptype1_txt', 'weapsubtype1',
               'weapsubtype1_txt', 'weaptype2', 'weaptype2_txt', 'weapsubtype2',
               'weapsubtype2_txt', 'weaptype3', 'weaptype3_txt', 'weapsubtype3',
               'weapsubtype3_txt', 'weaptype4', 'weaptype4_txt', 'weapsubtype4',
               'weapsubtype4_txt', 'weapdetail', 'nkill', 'nkillus', 'nkillter',
               'nwound', 'nwoundus', 'nwoundte', 'property', 'propextent',
               'propextent_txt', 'propvalue', 'propcomment', 'ishostkid',
               'nhostkid', 'nhostkidus', 'nhours', 'ndays', 'divert',
               'kidhijcountry', 'ransom', 'ransomamt', 'ransomamtus',
               'ransompaid', 'ransompaidus', 'ransomnote', 'hostkidoutcome',
               'hostkidoutcome_txt', 'nreleased', 'addnotes', 'scite1', 'scite2',
               'scite3', 'dbsource', 'INT_LOG', 'INT_IDEO', 'INT_MISC', 'INT_ANY',
               'related'], dtype=object)
```

```
df.dtypes
```

```
eventid          int64
iyear            int64
imonth           int64
iday             int64
approxdate      object
extended         int64
```

```
propextent        float64
propextent_txt     object
propvalue         float64
propcomment        object
ishostkid         float64
nhostkid          float64
nhostkidus        float64
nhours            float64
```

```
resolution                  object
country                      int64
country_txt                 object
region                       int64
region_txt                  object
provstate                   object
city                        object
latitude                   float64
longitude                  float64
specificity                float64
vicinity                     int64
location                    object
summary                     object
crit1                        int64
crit2                        int64
crit3                        int64
doubtterr                  float64
alternative                float64
alternative_txt             object
multiple                   float64
success                      int64
suicide                      int64
attacktype1                  int64
attacktype1_txt             object
ndays                      float64
divert                      object
kidhijcountry               object
ransom                     float64
ransomamt                  float64
ransomamtus                float64
ransompaid                 float64
ransompaidus               float64
ransomnote                  object
hostkidoutcome             float64
hostkidoutcome_txt          object
nreleased                  float64
addnotes                    object
scite1                      object
scite2                      object
scite3                      object
dbsource                    object
INT_LOG                      int64
INT_IDEO                     int64
INT_MISC                     int64
INT_ANY                      int64
related                     object
Length: 135, dtype: object
```

```
df.isnull().sum()
```

```
]: eventid                      0
   iyear                        0
   imonth                       0
   iday                         0
   ...
   propextent              117626
   propextent_txt          117626
   propvalue               142702
   propcomment             123732
   ishostkid                  178
   nhostkid                168119
```

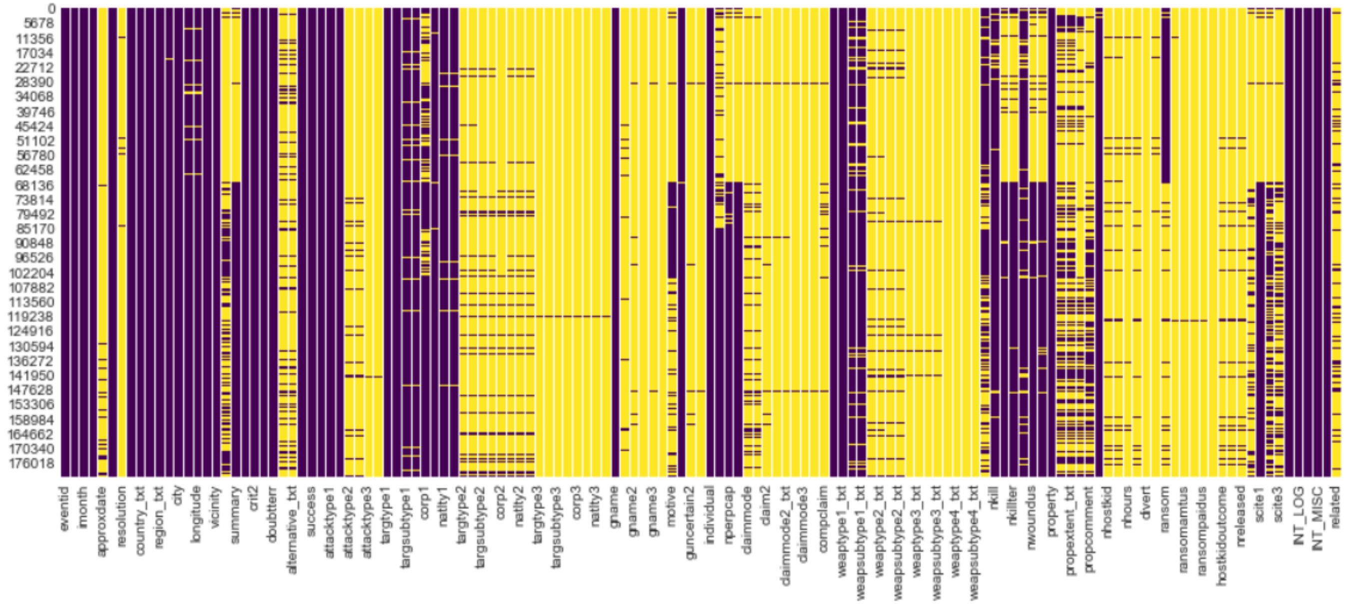
```
approxdate           172452
extended                  0
resolution           179471
country                   0
country_txt               0
region                    0
region_txt                0
provstate               421
city                    434
latitude               4556
longitude              4557
specificity               6
vicinity                  0
location             126196
summary               66129
crit1                     0
crit2                     0
crit3                     0
doubtterr                 1
alternative          152680
alternative_txt      152680
multiple                  1
success                   0
suicide                   0
attacktype1               0
attacktype1_txt           0
```

```
nhostkidus           168174
nhours               177628
ndays                173567
divert               181367
kidhijcountry        178386
ransom               104310
ransomamt            180341
ransomamtus          181128
ransompaid           180917
ransompaidus         181139
ransomnote           181177
hostkidoutcome       170700
hostkidoutcome_txt   170700
nreleased            171291
addnotes             153402
scite1                66191
scite2               104758
scite3               138175
dbsource                  0
INT_LOG                   0
INT_IDEO                  0
INT_MISC                  0
INT_ANY                   0
related              156653
Length: 135, dtype: int64
```

1.0.2 This graph visualizes that the dataset has several NaN values (Yellow ones)

```python
plt.figure(figsize=(16,6))
sns.heatmap(df.isnull(),cmap='viridis',cbar=False)
```

]: <matplotlib.axes._subplots.AxesSubplot at 0x29bb68e15c0>



1.1.1 Data Cleaning

To start with, we have eliminated most of the unjustifiable columns since they consist of NaN values also we are not going to base our hypothesis on those variables

```python
df.rename(columns={'eventid':'Event_ID','iyea
df=df[['Event_ID','Year','Month','Day','Count
```

```
#Number of Terrorists participating in Attack
df.head(10)
```

Country	Region	city	AttackType	Killed	Wounded	Target	Group	Target_type	Weapon_type	CostOfDamage	TerroristNumber
Dominican Republic	Central America & Caribbean	Santo Domingo	Assassination	1.0	0.0	Julio Guzman	MANO-D	Private Citizens & Property	Unknown	NaN	NaN
Mexico	North America	Mexico city	Hostage Taking (Kidnapping)	0.0	0.0	Nadine Chaval, daughter	23rd of September Communist League	Government (Diplomatic)	Unknown	NaN	7.0
Philippines	Southeast Asia	Unknown	Assassination	1.0	0.0	Employee	Unknown	Journalists & Media	Unknown	NaN	NaN
Greece	Western Europe	Athens	Bombing/Explosion	NaN	NaN	U.S. Embassy	Unknown	Government (Diplomatic)	Explosives	NaN	NaN
Japan	East Asia	Fukouka	Facility/Infrastructure Attack	NaN	NaN	U.S. Consulate	Unknown	Government (Diplomatic)	Incendiary	NaN	NaN
United States	North America	Cairo	Armed Assault	0.0	0.0	Cairo Police Headquarters	Black Nationalists	Police	Firearms	NaN	-99.0
Uruguay	South America	Montevideo	Assassination	0.0	0.0	Juan Maria de Lucah/Chief of Directorate of in...	Tupamaros (Uruguay)	Police	Firearms	NaN	3.0
United States	North America	Oakland	Bombing/Explosion	0.0	0.0	Edes Substation	Unknown	Utilities	Explosives	22500.0	-99.0
United States	North America	Madison	Facility/Infrastructure Attack	0.0	0.0	R.O.T.C. offices at University of Wisconsin, M...	New Year's Gang	Military	Incendiary	60000.0	1.0
United States	North America	Madison	Facility/Infrastructure Attack	0.0	0.0	Selective Service Headquarters in Madison Wisc...	New Year's Gang	Government (General)	Incendiary	NaN	1.0

From df.head(10) we can observe that TerroristNumber has some invalid values

```
df = df[pd.notnull(df['TerroristNumber'])]
df.index = pd.RangeIndex(len(df.index))
df = df[df.TerroristNumber != -99]
df.index = pd.RangeIndex(len(df.index))
df = df[df.TerroristNumber != 0]
df.index = pd.RangeIndex(len(df.index))
df = df[pd.notnull(df['Killed'])]
df.index = pd.RangeIndex(len(df.index))
```

```
df.shape
```

```
(27350, 16)
```

```
df.isnull().sum()
```

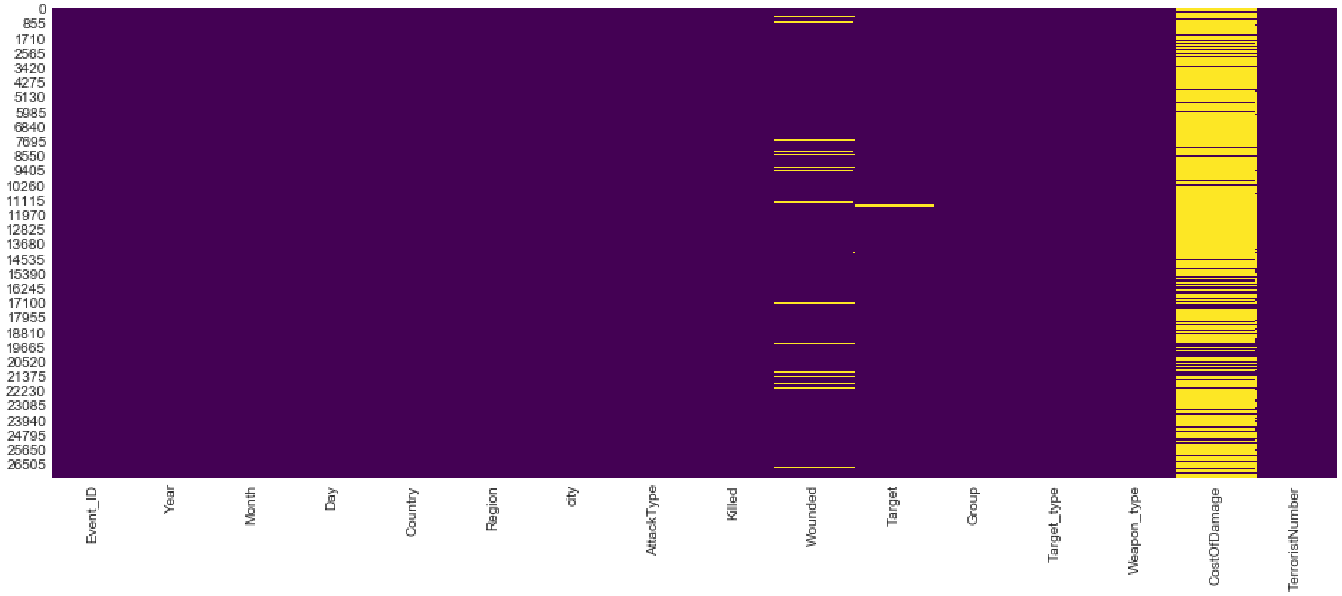
```
Event_ID              0
Year                  0
Month                 0
Day                   0
Country               0
Region                0
city                 42
AttackType            0
Killed                0
Wounded             941
Target              119
Group                 0
Target_type           0
Weapon_type           0
CostOfDamage      21707
TerroristNumber       0
dtype: int64
```

We can observe that CostofDamage is mostly consists of NaN values however we need to use those rows since they include valid information about Dataset, It will be handled in Hypothesis Testing .

This graph visualizes the NaN values same as above however this graph is mostly purple since we have deleted most of NaN values

```
plt.figure(figsize=(16,6))
sns.heatmap(df.isnull(),cmap='viridis',cbar=False)
```

```
]: <matplotlib.axes._subplots.AxesSubplot at 0x29bb6b52f28>
```

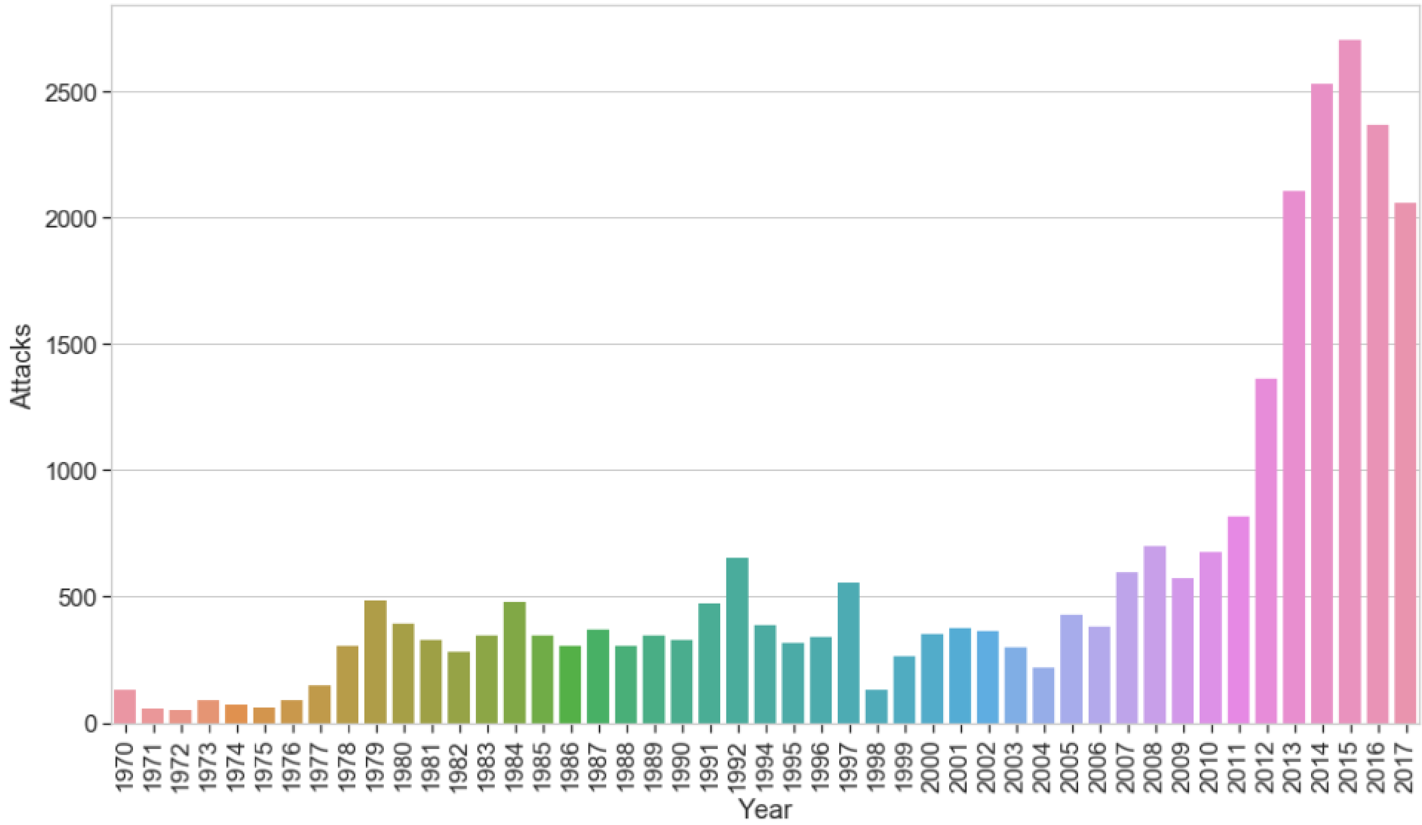


1.2.1 Data Visualization

Number of Total Attacks Between 1970 – 2017

First graph demonstrate the distribution of terror attacks year by year. Years are on the X axis and number of attacks are on the Y axis. A peak can be seen in 1992 and then a fall after it. Also there is an increase between years 2011 and 2015. After 2015 a fall is precisely shown.

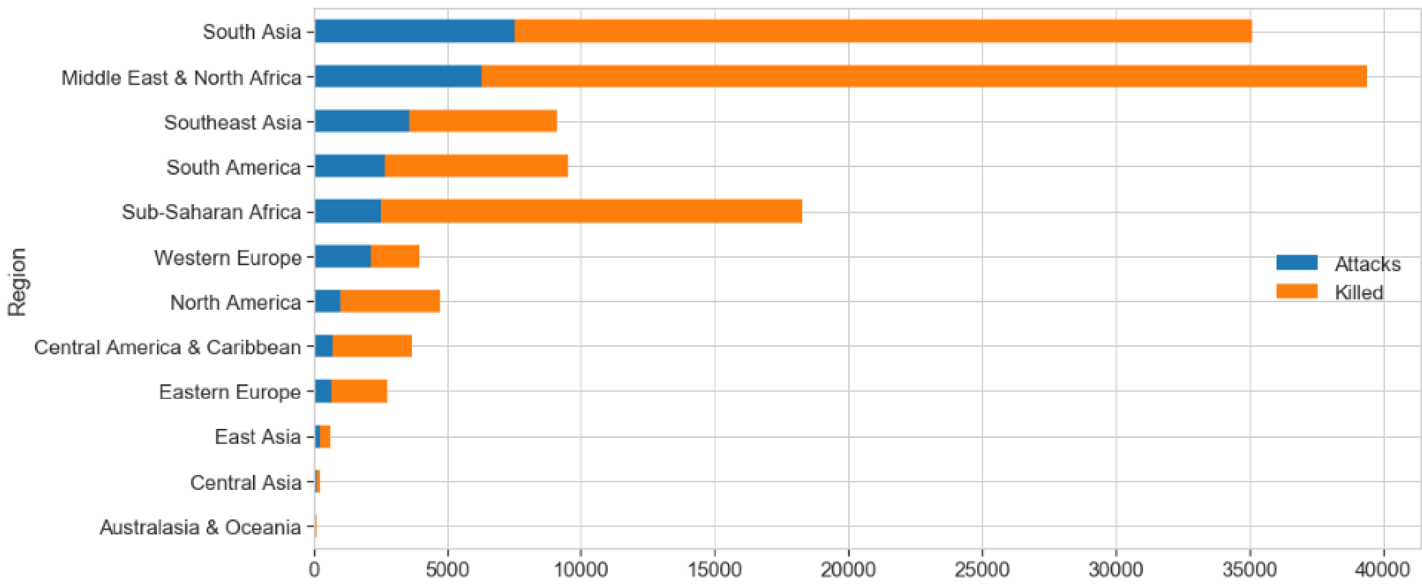
```
sns.set_context(context='notebook',font_scale
plt.figure(figsize=(16,9))
v1=df['Year'].value_counts().to_frame().reset
sns.barplot(data=v1,x='Year',y='Attacks',ci=N
plt.xticks(rotation=90)
```



Most Affected Regions by Terror Attacks

The most attacks occured in South Asia and then Middle East and North Africa (MENA). But if we look at the number of killed people MENA is greater than South Asia. The number of killed people per attack is greater in MENA. Also in Sub-Saharan Africa the ratio is very high.

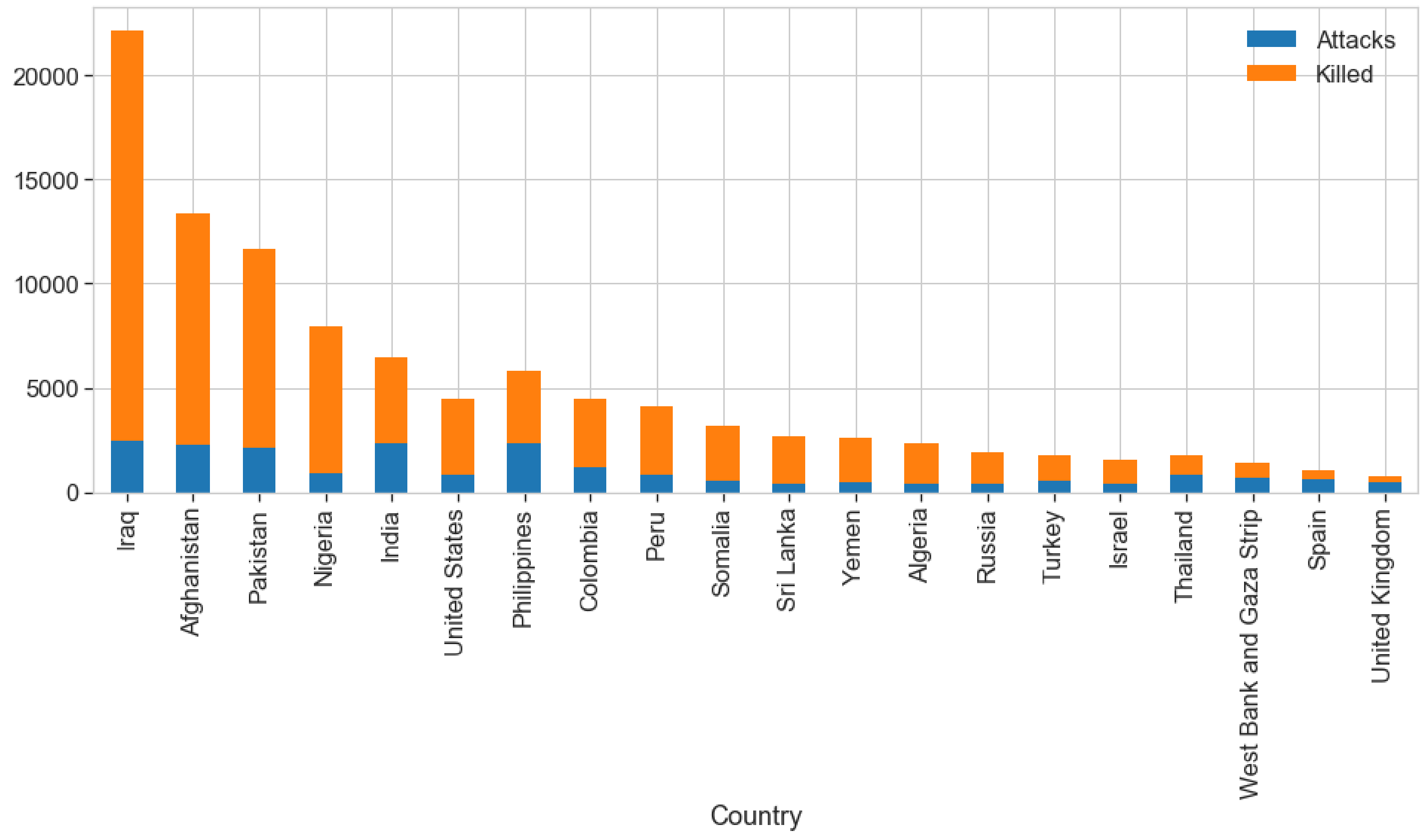
```
df['Region'].value_counts(ascending=True).to_
plt.legend(loc=5)
```



Most Affected Countries by Terror Attacks

Iraq is the number one country according to the graph. Number of attacks in Iraq, Afghanistan and Pakistan are close to each other. But number of killed people in Iraq is largely greater than others. Turkey is in the top 15.

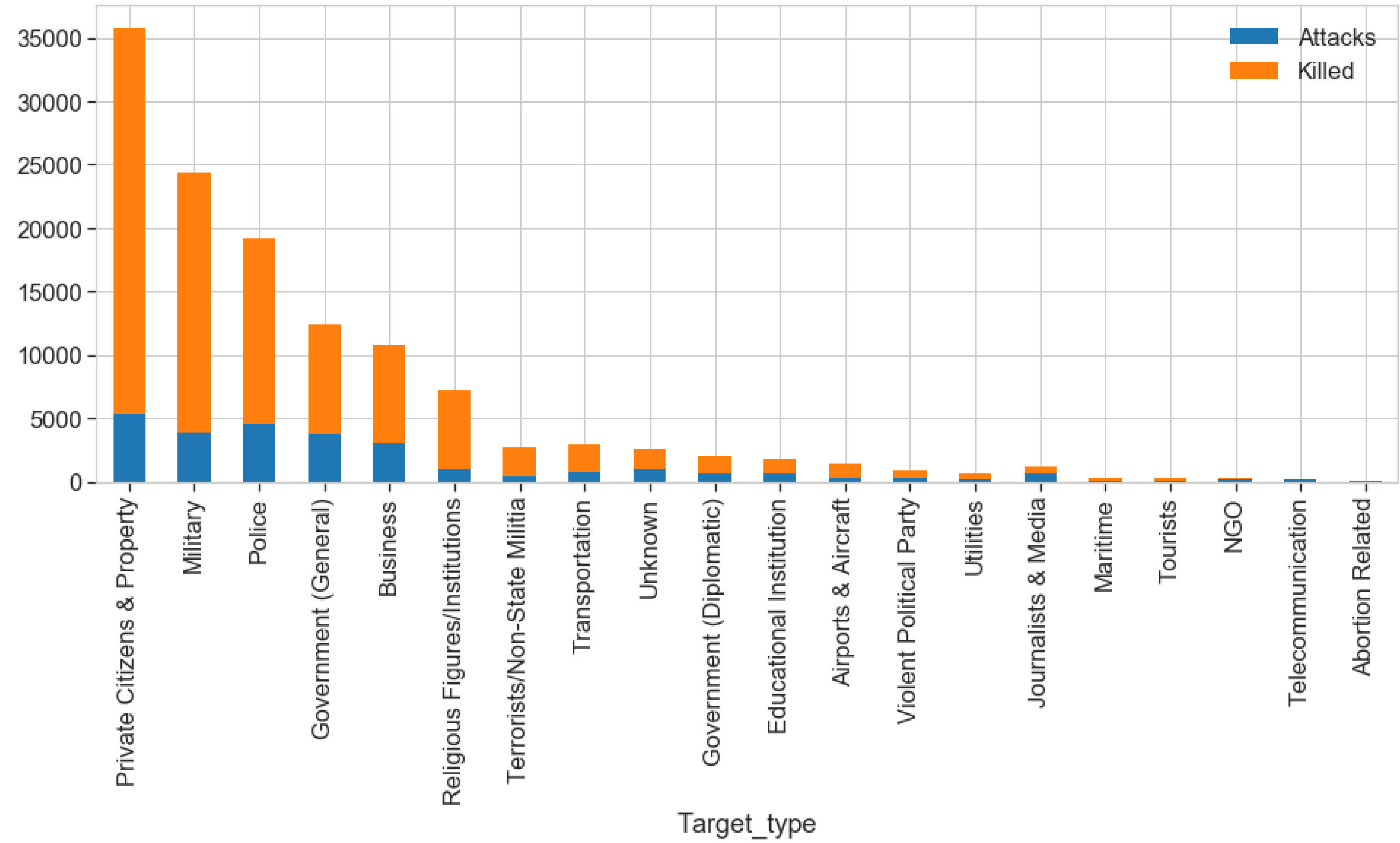
```
df[df['Country'].isin(df['Country'].value_cou
```



Most Affected Target Types

It can be seen that civils are the largest target grouop for terror attacks and they have the most number of killed people with more than 35000 people. Military, police and government people should also be taken into the consideration with its high number of killed people.

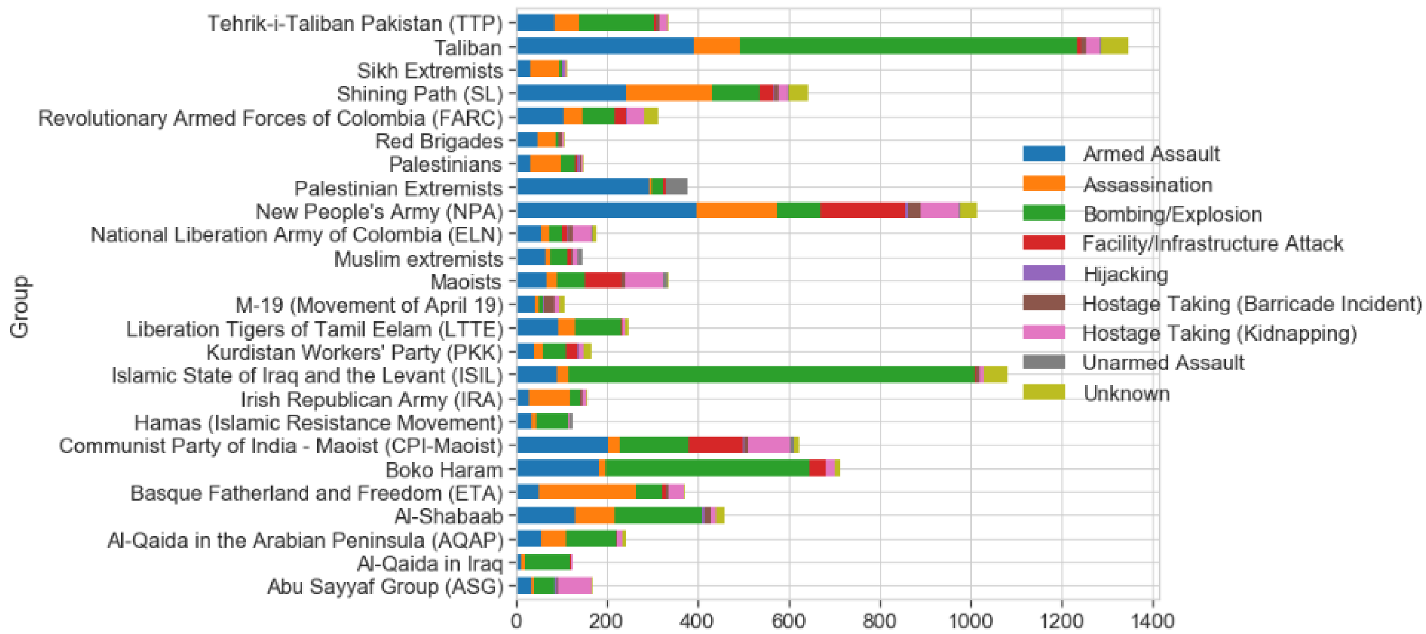
```
df[df['Target_type'].isin(df['Target_type'].v
```



Distribution of Attack Types of Terror Groups

Here we can see how terror groups attacks generally. Taliban and ISIL mostly use Bombing when attacking. This distribution can show the method of terror groups and terror groups can be categorized by their attack types. Most terror groups have their own ideologies and they want to be known by thier own features. It shows us they choose spesific methods and use it repeatedly.

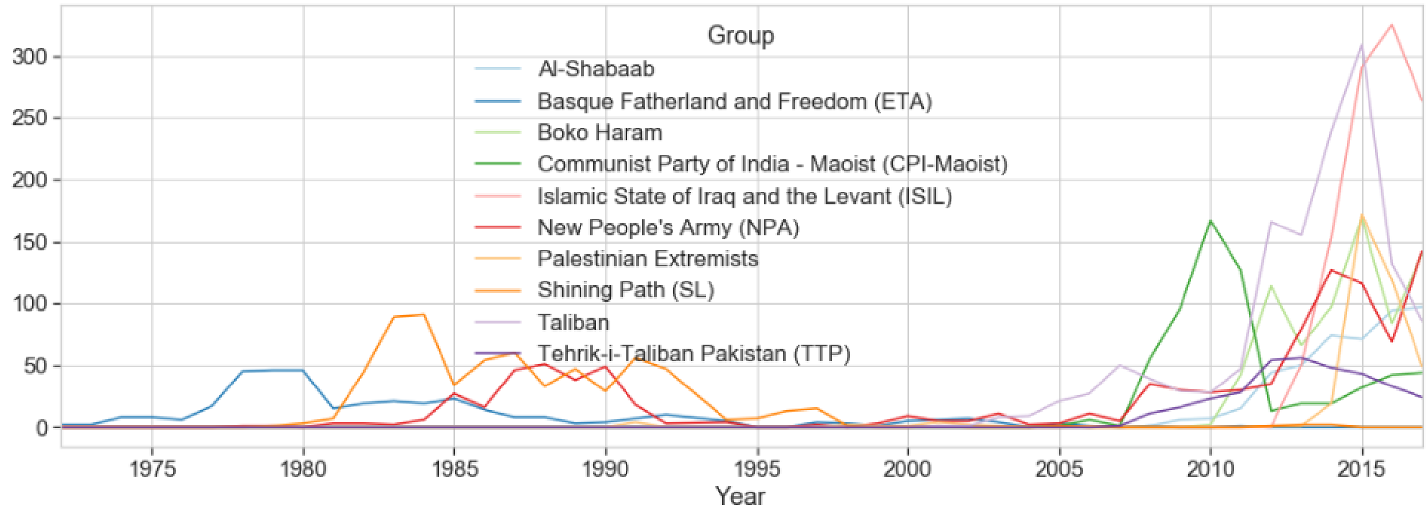
```
v1=df[df['Group'].isin(df['Group'].value_coun
v2=v1[v1['Target_type'].isin(v1['Target_type'
pd.crosstab(v1['Group'],v1['AttackType']).plo
plt.legend(loc=9,bbox_to_anchor=(1.1,0.8))
```



Number of Attacks by Terrror Groups Between 1970 – 2017

Their number of attacks over the years are shown in the graph. Also the time periods when terror gropus are active can be seen here. Espeacially between 1980 and 1995 terror attacks are increased. And then a fall can be seen. After 2007 some new terror groups are emerged and total number of attacks are increased in world. Some terror groups are very active in some short periods. They gain popularity and then disappear after a while. ISIL can be an example to that. It emerged in 2010s. They have a peak in 2016 and they started to disappear after that. Also there are terror groups which maintain their activities more than 20 years like TTP and SL.

```
pd.crosstab(df[df['Group'].isin(df['Group'].v
```



Number of attacks Top5 Country

```
pd.value_counts(df['AttackType'])[0:5]
```

```
Bombing/Explosion                      9339
Armed Assault                          8279
Assassination                          4535
Hostage Taking (Kidnapping)            1747
Facility/Infrastructure Attack         1611
Name: AttackType, dtype: int64
```

Most used Weapons in Attacks

```python
pd.value_counts(df['Weapon_type'])[0:5]
```

```
Firearms       12630
Explosives     10067
Unknown         1733
Incendiary      1463
Melee           1223
Name: Weapon_type, dtype: int64
```

Most Targeted Groups

```python
pd.value_counts(df['Target_type'])[0:5]
```

```
Private Citizens & Property    5423
Police                         4520
Military                       3930
Government (General)           3799
Business                       3050
Name: Target_type, dtype: int64
```

Number of Kills per Attack

```
[29]:  df['Killed'].sum()/df['Killed'].count()
Out[29]: 3.66
```

Number of Unique Weapon Types

```
[87]:  df['Weapon_type'].nunique()
Out[87]: 12
```

Number of Unique Target Types

```
[88]:  df['Target_type'].nunique()
Out[88]: 22
```

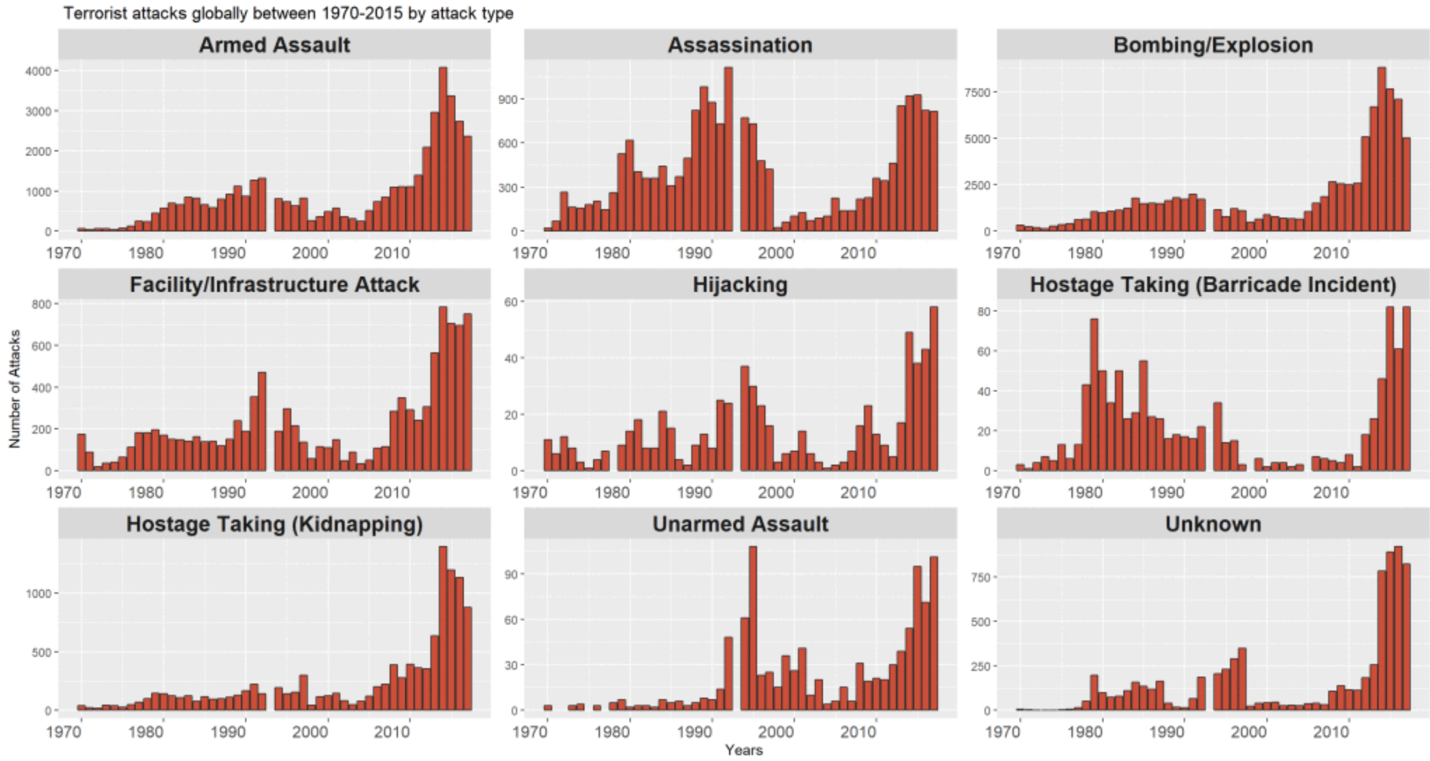
Visualizing Terrorist attacks in Turkey and World

```
GT <- read.csv("master1.csv")
TIN = GT[which(GT$country_txt=='Turkey'),]
TIN[TIN==""] <- NA #empty cells become NA
library(ggplot2)
```

```
library(grid)
library(leaflet)
library(dplyr)
```

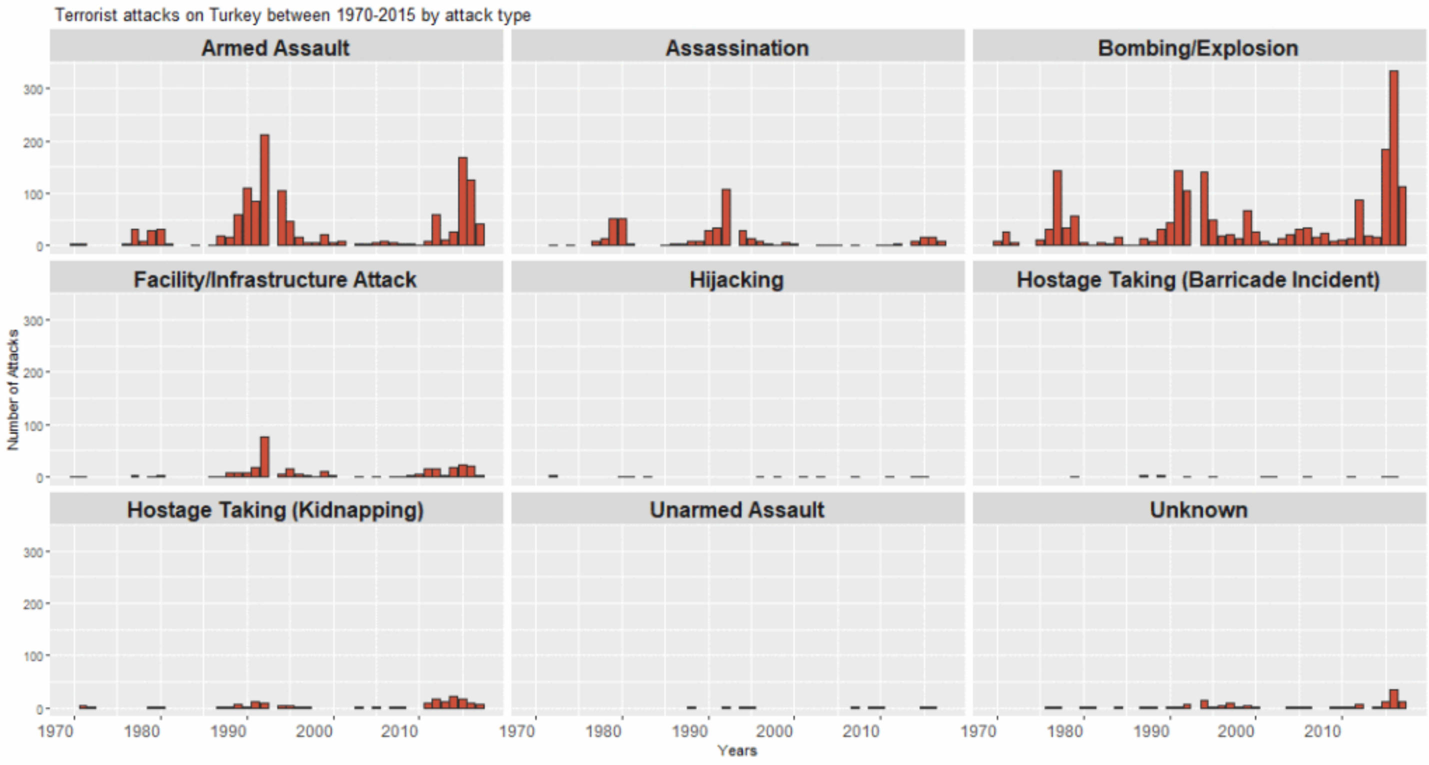
1.3.0 let's have a look at terrorist attacks globally by attack type

```
ggplot(GT, aes(x = iyear))+ labs(title =" Ter
  geom_bar(colour = "grey19", fill = "tomato3
  theme(axis.text.x = element_text(hjust = 1,
```



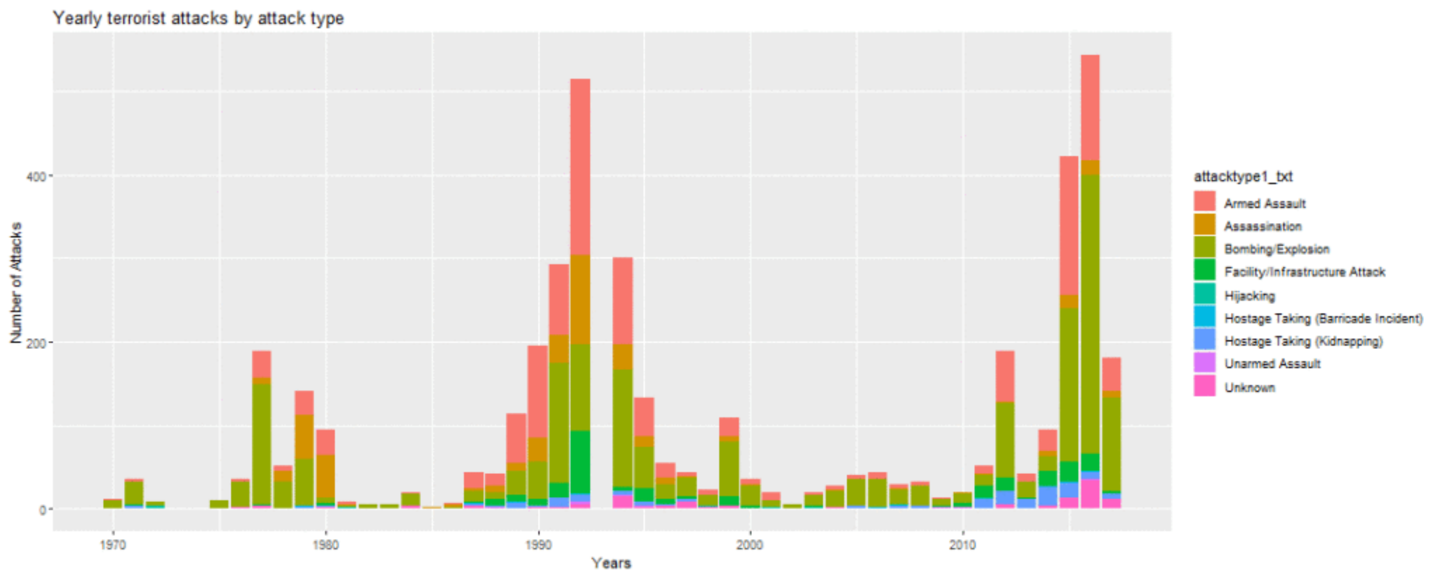
Terrorist attacks on Turkey between 1970-2015 by ATTACK type

```
ggplot(TIN,aes(x = iyear))+ labs(title =" Ter
  geom_bar(colour = "grey19", fill = "tomato3
  theme(strip.text = element_text(size = 16,
```



1.3.1 Yearwise terrorist attacks by ATTACK type in Turkey

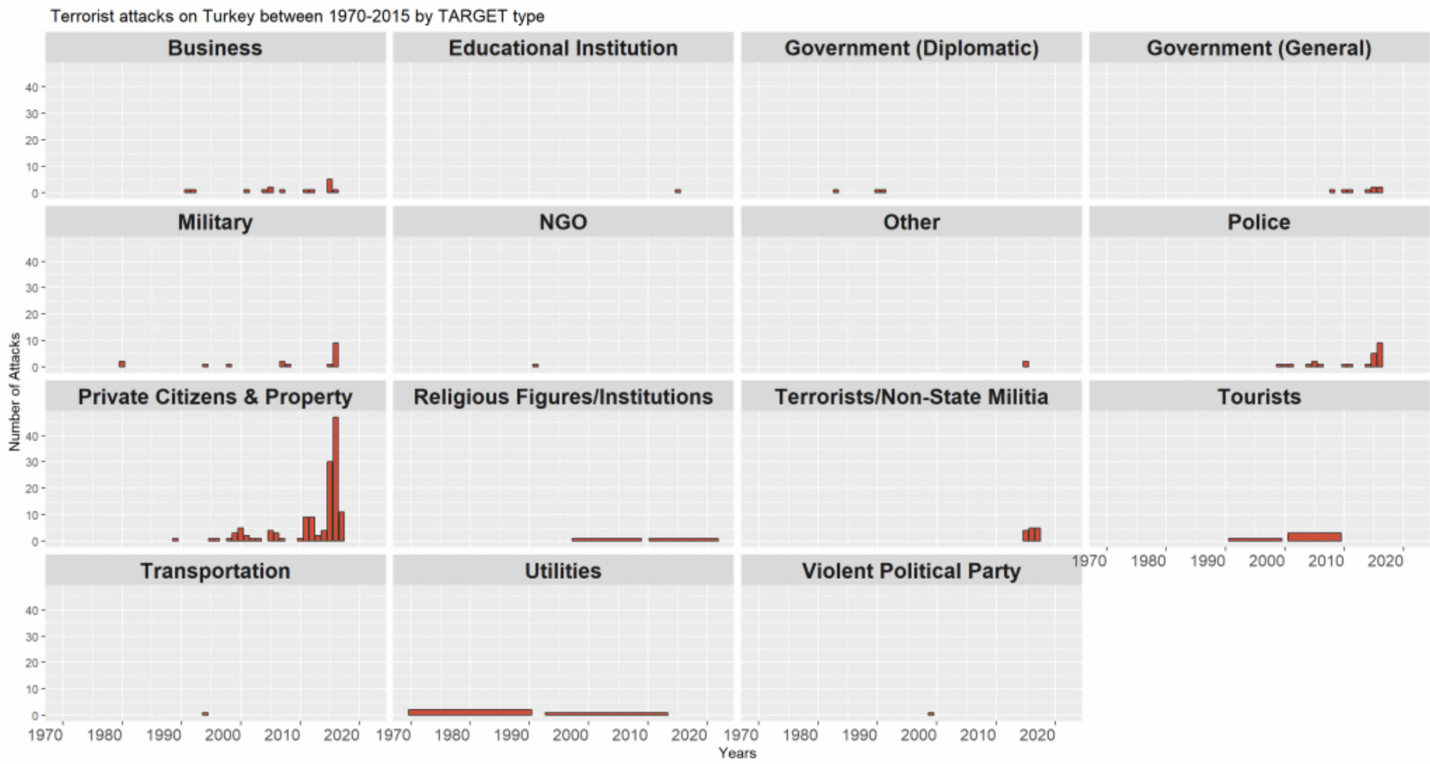
```
ggplot(data=TIN, aes(x=iyear,fill=attacktype1
    labs(x = "Years", y = "Number of Attacks"
```



Terrorist attacks in Turkey by TARGET type

```
TINclean = TIN[which(TIN$targsubtype2_txt !='

ggplot(TINclean, aes(x = iyear))+ labs(title
  geom_bar(colour = "grey19", fill = "tomato3
  theme(strip.text = element_text(size = 16,
```

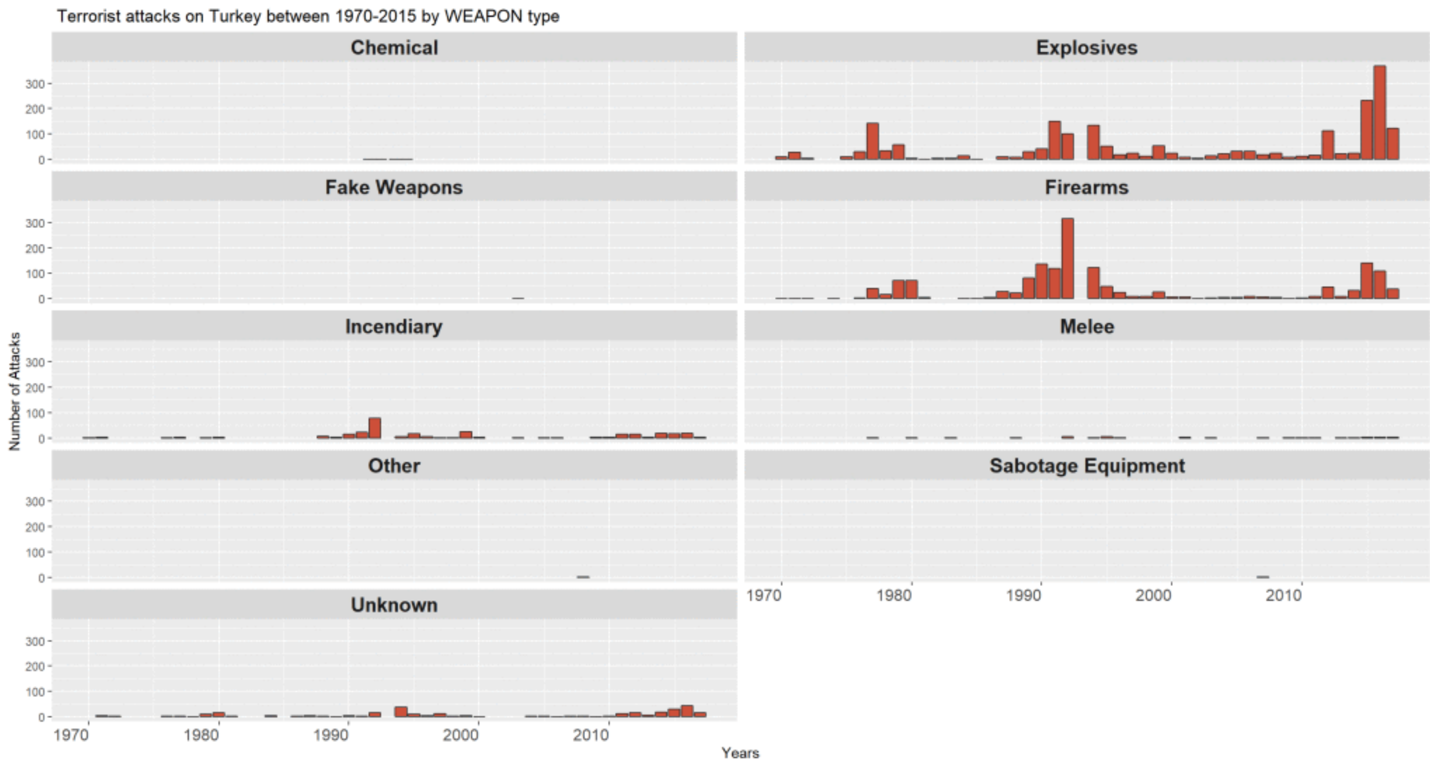


Yearwise terrorist attacks globally by TARGET type

```
ggplot(GT, aes(x=iyear,fill=targtype1_txt)) +
    labs(x = "Years", y = "Number of Attacks"
```

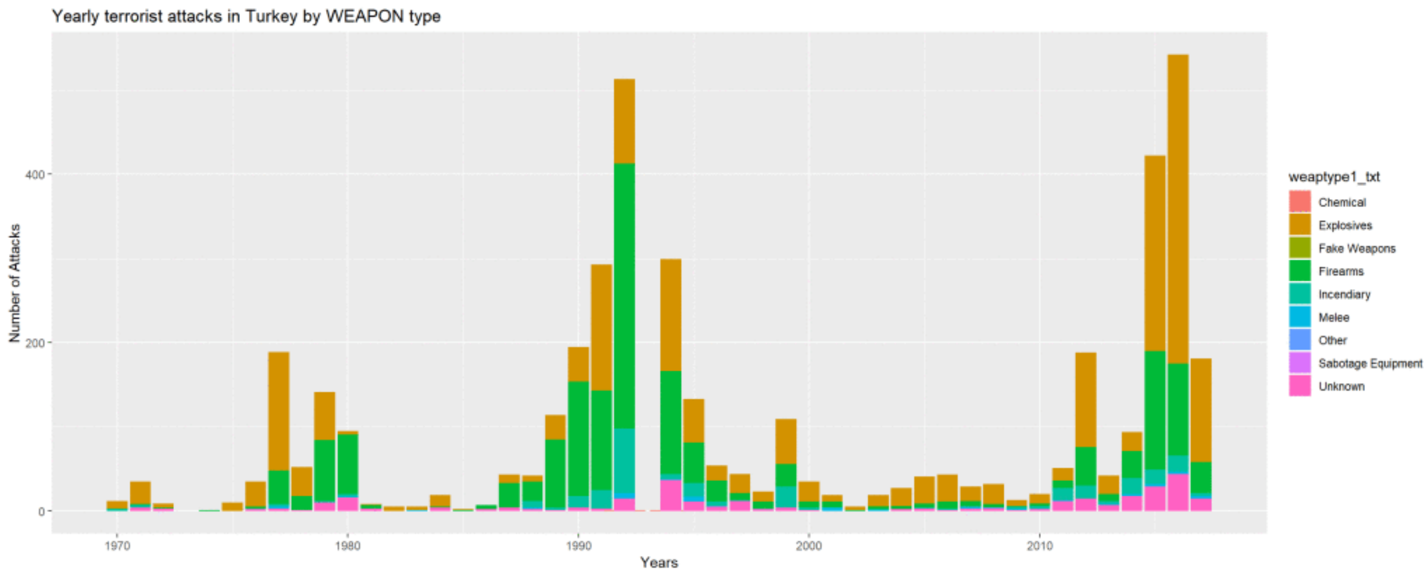
Terrorist attacks in Turkey by weapon type

```
ggplot(TIN, aes(x = iyear))+ labs(title =" Te
  geom_bar(colour = "grey19", fill = "tomato3
  facet_wrap(~weaptype1_txt, ncol = 2) + them
```



Yearwise terrorist attacks in Turkey by WEAPON type

```
ggplot(data=TIN, aes(x=iyear,fill=weaptype1_t
    geom_bar() + ggtitle("Yearly terrorist at
    labs(x = "Years", y = "Number of Attacks"
```



---

2. Hypothesis Testing

2.1 Does the weapon type effect the number of killed people?

```
der=[]
for i in range(0,27350):
    if df['Weapon_type'][i]=="Explosives" :
```

```
            der.append(1)
        else:
            der.append(0)

df["ExpOrNot"]=der
ert1=df[df["ExpOrNot"] == 1]["Killed"]
ert2=df[df["ExpOrNot"] == 0]["Killed"]
stats.ttest_ind(ert1, ert2, equal_var=False)
```

```
Ttest_indResult(statistic=15.11521942446915, pvalue=2.22852726319430
```

Result:Before the test we can observe there are 12 types of Weapon Types and we need to convert them into 2 samples where these are weapon type is Explosive and others. After the conversion we can group them with their Kills. From the T-test we conduct the P value as 2.2285272631943068e-51 where it is much smaller than the significance level which is 0.05. Since pvalue of this test is smaller than 0.05 we can reject our null hypothesis. Test score is 15.1152 which indicates that the groups are 15 times as different from each other as they are within each other. Since our tscore is large we can say that the groups are different, not similiar.

2.2 Does the target type effect the number of killed people?

```
die=[]
for i in range(0,27350):
    if df['Target_type'][i]=="Private Citize
        die.append(1)
    else:
        die.append(0)

df["CitizenorNot"]=die
ert1=df[df["CitizenorNot"] == 1]["Killed"]
ert2=df[df["CitizenorNot"] == 0]["Killed"]
stats.ttest_ind(ert1, ert2, equal_var=False)
```

```
Ttest_indResult(statistic=5.845911101009332, pvalue=5.3142661566395
```

Result:Before the test we can observe there are 22 types of Target Types and we need to convert them into 2 samples where these are target type is Private Citizens & Property and others. After the conversion we can group them with their Kills. From the T-test we conduct the P value as 5.314266156639527e-09 where it is much smaller than the significance level which is 0.05. Since pvalue of this test is smaller than 0.05 we can reject our null hypothesis. Test score is 5.8459 which indicates that the groups are 5 times as different from each other as they are within each other

2.3 Does the weapon type effect the cost of damage ?

```
df1 = df.copy()
df1 = df1[pd.notnull(df1['CostOfDamage'])]
df1.index = pd.RangeIndex(len(df1.index))

das=[]
for i in range(0,5643):
    if df1['Weapon_type'][i]=="Firearms" :
```

```
            das.append(1)
        else:
            das.append(0)

df1["fireEffect"]=das
ert1=df1[df1["fireEffect"] == 1]["CostOfDama
ert2=df1[df1["fireEffect"] == 0]["CostOfDama
stats.ttest_ind(ert1, ert2, equal_var=False)
```

```
Ttest_indResult(statistic=-1.7963519561643368, pvalue=0.07251786941
```

Result:Before the test we can observe there are 12 types of Weapon and we need to convert them into 2 samples where these are target type is Firearms and others. After the conversion we can group them with their Cost of damage. From the T-test we conduct the P value as 0.07251786941718862 where it is bigger than the significance level which is 0.05. Since pvalue of this test is bigger than 0.05, the result is not statistically significant which indicates weak evidence against the null hypothesis so we fail to reject the null hypothesis. Test score is neagtive which indicates that mean of cost of damage with “Firearms” does not equal to mean of distance covered with other types of weapons.

3. Single Linear Regression

3.1- Regression between Terrorist Attack Counts and Population growth

We’ll get a summary first,

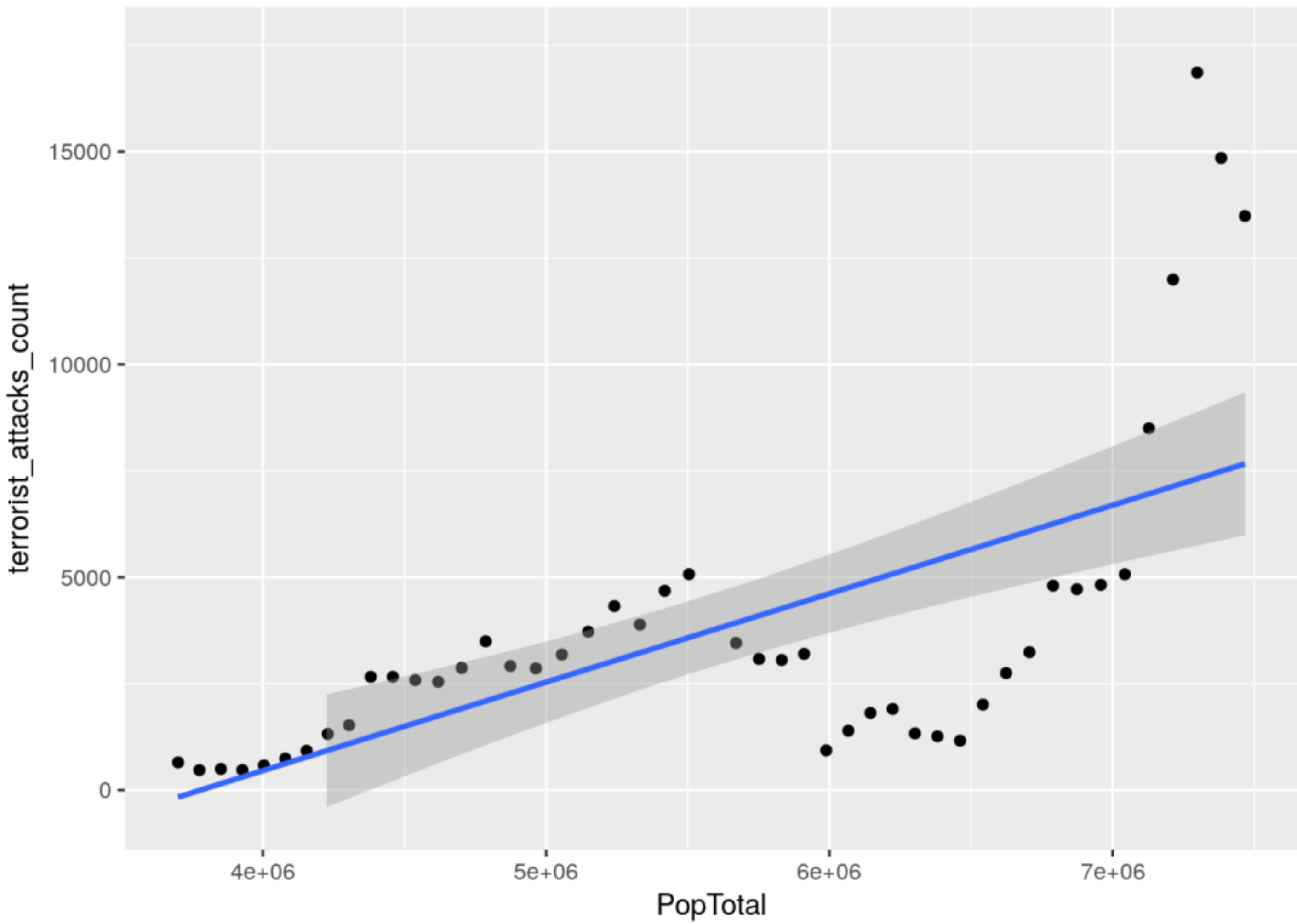
```
##
## Call:
## (formula = terrorist_attacks_count ~ PopT
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4413.0 -1713.2    365.4    994.9   9544.8
##
## Coefficients:
##                Estimate Std. Error t value
## (Intercept) -7.857e+03  2.132e+03  -3.686
## PopTotal     2.079e-03  3.757e-04   5.534
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '
##
## Residual standard error: 2884 on 44 degre
## Multiple R-squared:  0.4104, Adjusted R-s
## F-statistic: 30.62 on 1 and 44 DF,  p-val
```

$$\begin{cases} H_0: & \text{variance-terrorist attacks properly explained by population growth} \\ H_a: & \text{variance-terrorist attacks not properly explained by populationgrowth} \end{cases}$$

```
#Adjusted R-squared: 0.397 with p<0.05.

ggplot(data=GT, aes(x=PopTotal, y=terrorist_a
  geom_point() +
  geom_smooth(method="lm",formula= y ~ x)+
  scale_y_continuous(limits = c(-500, 17500))
```



The Linear Model is significant but explains the variance in terrorist attacks count quite poorly with an adjusted R-squared of 0.397 ($p<0.05$). This means we REJECT the Null Hypothesis because the variance is not properly explained by population growth, meaning there are other factors that leads to these changes.

3.2- Regression between Number of Terrorist Participating in Attacks and Number of Kills

```
df3 = df.copy()

#they can not be nan or negative number it do
#to be more precise we narrow the value for N
df3 = df3[df3['TerroristNumber'] > 0]

df3 = df3[df3['TerroristNumber'] < 1000]

df3 = df3[df3['Killed'] < 1000]
```

In [66]:
```python
df3.shape
```
Out[66]: (27270, 18)

In [68]:
```python
new = df3[['TerroristNumber', 'Killed']].copy()
```

Mean x = 14.85 ,,, Mean y = 3.5

In [69]:
```python
new.describe()
```
Out[69]:

	TerroristNumber	Killed
count	27270.000000	27270.000000
mean	14.859553	3.532453
std	49.019456	11.168726
min	1.000000	0.000000
25%	1.000000	0.000000
50%	2.000000	1.000000
75%	6.000000	3.000000
max	800.000000	588.000000

```python
x = new['TerroristNumber']
```

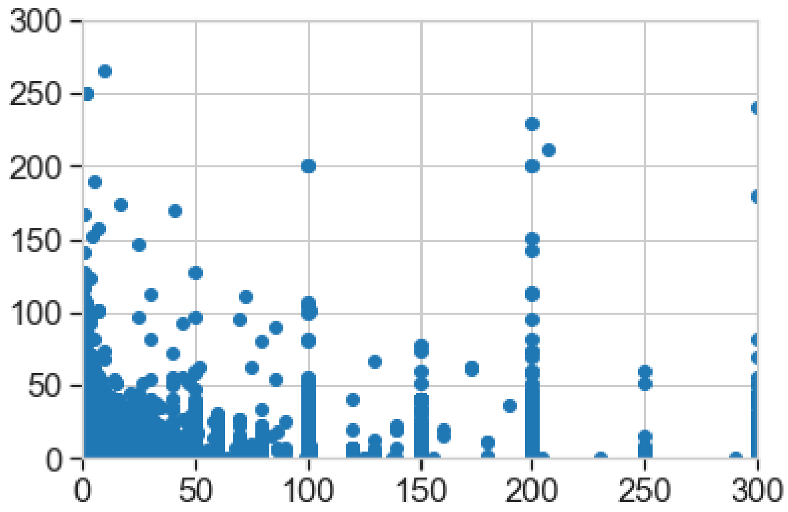
```
y = new['Killed']
slope, intercept,x_value,p_value,std_err = s

plt.scatter(x,y)

plt.axis([0,300,0,300])
plt.plot()
plt.show()


#Prediction
newX = 15.2
newY = newX*slope*intercept

print(newY)
```




```
1.8457229150621142
```

Given prediction x=15.2 that expected people to be killed is y = 4.2

This is how x and y look now:

```
nperps_real = new["TerroristNumber"].values
nkills_real = new["Killed"].values

x = np.array(nperps_real).reshape((-1, 1))
y = np.array(nkills_real)
print(x)
```

```
[[ 7.]
 [ 3.]
 [ 1.]
 ...
 [ 1.]
 [12.]
 [ 3.]]
```

```
print(y)
```

```
[0. 0. 0. ... 0. 0. 8.]
```

Now, we have two arrays: the input x and output y. We should call .reshape() on x because this array is required to be two-dimensional, or to be more precise, to have one column and as many rows as necessary. That’s exactly what the argument (-1, 1) of .reshape() specifies.

As you can see, x has two dimensions, and x.shape is (n, 1), while y has a single dimension, and y.shape is (n,)

```
In [76]: model = LinearRegression()

In [77]: model.fit(x, y)

Out[77]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                  normalize=False)

In [78]: print('intercept:', model.intercept_)

intercept: 2.913036894496056

In [79]: print('slope:', model.coef_)

slope: [0.04168472]

In [80]: y_pred = model.predict(x)
         print('predicted response:', y_pred, sep='\n')

predicted response:
[3.20482996 3.03809107 2.95472162 ... 2.95472162 3.41325358 3.03809107]

In [81]: y_pred = model.predict(x)
         print('predicted response:', y_pred, sep='\n')

predicted response:
[3.20482996 3.03809107 2.95472162 ... 2.95472162 3.41325358 3.03809107]

In [82]: r_sq = model.score(x, y)
         print('coefficient of determination:', r_sq)

coefficient of determination: 0.03347215001648096
```

Since R^2 analysis's result is very close to 0 there is not any corelation between

The total number of terrorists participating in the incident and The number of total confirmed fatalities for the incident

Alper Bingöl, Baran Gayretli, Ege Arıkan, Poyraz Özmen

Global Terrorism Problem 27 April 2019 Genel

Leave a comment