Libraries

In [1]:
```
# Libraries to manipulate
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.ticker as mticker
import matplotlib.dates as mdates
import seaborn as sns
import sys
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
```

Member Task 1

Data

In [2]:
```
df = pd.read_csv('covid_deaths.csv')
df.head(5)
```

Out[2]:

	Unnamed: 0	Country/Region	Province/State	date	Recovered	confirmed	deaths	Country_code	Continent	Population	Life Expectancy	Literacy Rate	Gross Primary Education Enrollment	Gross Tertiary Education Enrollment	Unemployment Rate	Urban population	Latitude	Longitude
0	0	Afghanistan	NaN	2020-03-03	0	1	0	AF	AS	38928346.0	64.83	37.0	116.00	9.70	11.12	9797273.0	33.939110	67.709953
1	1	Afghanistan	NaN	2020-03-04	0	1	0	AF	AS	38928346.0	64.83	37.0	116.00	9.70	11.12	9797273.0	33.939110	67.709953
2	2	Afghanistan	NaN	2020-03-05	0	1	0	AF	AS	38928346.0	64.83	37.0	116.00	9.70	11.12	9797273.0	33.939110	67.709953
3	3	Afghanistan	NaN	2020-03-06	0	1	0	AF	AS	38928346.0	64.83	37.0	116.00	9.70	11.12	9797273.0	33.939110	67.709953
4	4	Afghanistan	NaN	2020-03-07	0	1	0	AF	AS	38928346.0	64.83	37.0	116.00	9.70	11.12	9797273.0	33.939110	67.709953

5 rows × 20 columns

New York Data Analysis

In [3]:
```
def ny_data(df,country,date,this_date,district,county=False):
    global ny_df
    ny_df = df[df['Country/Region'].isin([country])]
    ny_df = ny_df[ny_df['date'].isin([date])]
    ny_df = ny_df[ny_df['Province/State'].isin(['New York'])]
    ny_df.reset_index(drop=True)
    ny_df['date'] = pd.to_datetime(ny_df['date'])
    ny_df['date'] = ny_df['date'].dt.strftime('%m/%d/%Y')
    ny_df['date'] = pd.to_datetime(ny_df['date'])
    ny_df['Country_Code'] = ny_df['Country_Code'].astype('int')
    ny_df['Population'] = ny_df['Population'].astype('int')
    ny_df['Life Expectancy'] = ny_df['Life Expectancy'].astype('int')
    ny_df['Literacy Rate'] = ny_df['Literacy Rate'].astype('int')
    ny_df['Gross Primary Education Enrollment'] = ny_df['Gross Primary Education Enrollment'].astype('int')
    ny_df['Gross Tertiary Education Enrollment'] = ny_df['Gross Tertiary Education Enrollment'].astype('int')
    ny_df['Unemployment Rate'] = ny_df['Unemployment Rate'].astype('int')
    ny_df['Urban population'] = ny_df['Urban population'].astype('int')
    ny_df['Latitude'] = ny_df['Latitude'].astype('int')
    ny_df['Longitude'] = ny_df['Longitude'].astype('int')
    ny_df['Province/State'] = ny_df['Province/State'].fillna('Unknown')
    ny_df['Province/State'] = ny_df['Province/State'].replace('New York','Statewide')
    ny_df['Continent'] = ny_df['Continent'].fillna('Unknown')
    ny_df['Unemployment Rate'] = ny_df['Unemployment Rate'].replace(0,np.nan)
    ny_df['Unemployment Rate'] = ny_df['Unemployment Rate'].fillna(ny_df['Unemployment Rate'].mean())
    ny_df['Life Expectancy'] = ny_df['Life Expectancy'].replace(0,np.nan)
    ny_df['Life Expectancy'] = ny_df['Life Expectancy'].fillna(ny_df['Life Expectancy'].mean())
    ny_df['Literacy Rate'] = ny_df['Literacy Rate'].replace(0,np.nan)
    ny_df['Literacy Rate'] = ny_df['Literacy Rate'].fillna(ny_df['Literacy Rate'].mean())
    ny_df['Gross Primary Education Enrollment'] = ny_df['Gross Primary Education Enrollment'].replace(0,np.nan)
    ny_df['Gross Primary Education Enrollment'] = ny_df['Gross Primary Education Enrollment'].fillna(ny_df['Gross Primary Education Enrollment'].mean())
    ny_df['Gross Tertiary Education Enrollment'] = ny_df['Gross Tertiary Education Enrollment'].replace(0,np.nan)
    ny_df['Gross Tertiary Education Enrollment'] = ny_df['Gross Tertiary Education Enrollment'].fillna(ny_df['Gross Tertiary Education Enrollment'].mean())
    ny_df['Urban population'] = ny_df['Urban population'].replace(0,np.nan)
    ny_df['Urban population'] = ny_df['Urban population'].fillna(ny_df['Urban population'].mean())
    ny_df['Latitude'] = ny_df['Latitude'].replace(0,np.nan)
    ny_df['Latitude'] = ny_df['Latitude'].fillna(ny_df['Latitude'].mean())
    ny_df['Longitude'] = ny_df['Longitude'].replace(0,np.nan)
    ny_df['Longitude'] = ny_df['Longitude'].fillna(ny_df['Longitude'].mean())
    ny_df['Province/State'] = ny_df['Province/State'].replace('Unknown','Statewide')
    ny_df['Province/State'] = ny_df['Province/State'].replace('Diamond Princess','Statewide')
    ny_df['Province/State'] = ny_df['Province/State'].replace('Grand Princess','Statewide')
    ny_df['Province/State'] = ny_df['Province/State'].replace('Recovered','Statewide')
    ny_df['Province/State'] = ny_df['Province/State'].replace('Northern Mariana Islands','Statewide')
    ny_df['Province/State'] = ny_df['Province/State'].replace('Puerto Rico','Statewide')
    ny_df['Province/State'] = ny_df['Province/State'].replace('Guam','Statewide')
    ny_df['Province/State'] = ny_df['Province/State'].replace('Virgin Islands, U.S.','Statewide')
    ny_df['Province/State'] = ny_df['Province/State'].replace('American Samoa','Statewide')
    if county:
        ny = ny_df[ny_df['Province/State'].isin(['Statewide'])]
        ny = ny[ny[district].isin([this_date])]
        ny = ny[ny['Country/Region'].isin(['US'])]
        ny = ny[ny['Country_Code'].isin([840])]
        ny = ny[ny['Population'].isin([ny['Population'].max()])]
        ny = ny[ny['Urban population'].isin([ny['Urban population'].max()])]
        ny = ny[ny['Gross Primary Education Enrollment'].isin([ny['Gross Primary Education Enrollment'].max()])]
        ny = ny[ny['Gross Tertiary Education Enrollment'].isin([ny['Gross Tertiary Education Enrollment'].max()])]
        ny = ny[ny['Unemployment Rate'].isin([ny['Unemployment Rate'].max()])]
        ny = ny[ny['Life Expectancy'].isin([ny['Life Expectancy'].max()])]
        ny = ny[ny['Literacy Rate'].isin([ny['Literacy Rate'].max()])]
        ny = ny[ny['Latitude'].isin([ny['Latitude'].max()])]
        ny = ny[ny['Longitude'].isin([ny['Longitude'].max()])]
        return ny
    else:
        ny = ny_df[ny_df[district].isin([this_date])]
        ny = ny[ny['Country/Region'].isin(['US'])]
        ny = ny[ny['Country_Code'].isin([840])]
        ny = ny[ny['Population'].isin([ny['Population'].max()])]
        ny = ny[ny['Urban population'].isin([ny['Urban population'].max()])]
        ny = ny[ny['Gross Primary Education Enrollment'].isin([ny['Gross Primary Education Enrollment'].max()])]
        ny = ny[ny['Gross Tertiary Education Enrollment'].isin([ny['Gross Tertiary Education Enrollment'].max()])]
        ny = ny[ny['Unemployment Rate'].isin([ny['Unemployment Rate'].max()])]
        ny = ny[ny['Life Expectancy'].isin([ny['Life Expectancy'].max()])]
        ny = ny[ny['Literacy Rate'].isin([ny['Literacy Rate'].max()])]
        ny = ny[ny['Latitude'].isin([ny['Latitude'].max()])]
        ny = ny[ny['Longitude'].isin([ny['Longitude'].max()])]
        return ny
```

In [4]:
```
ny_data(df,'US','2020-04-12','12/04/2020','Province/State',False)
```

Out[4]:

	Unnamed: 0	Country/Region	Province/State	date	Recovered	confirmed	deaths	Country_code	Continent	Population	Life Expectancy	Literacy Rate	Gross Primary Education Enrollment	Gross Tertiary Education Enrollment	Unemployment Rate	Urban population	Latitude	Longitude
1945	1945	US	Statewide	2020-04-12	0	1	0	840	NA	331002651.0	78.93	99.0	101.00	88.00	8.10	270857004.0	38.894992	-77.036560

1 rows × 20 columns

Cases

In [5]:
```
ny_df.head(1)
ny_df.tail(1)
```

Out[5]:

	Unnamed: 0	Country/Region	Province/State	date	Recovered	confirmed	deaths	Country_code	Continent	Population	Life Expectancy	Literacy Rate	Gross Primary Education Enrollment	Gross Tertiary Education Enrollment	Unemployment Rate	Urban population	Latitude	Longitude
1945	1945	US	Statewide	2020-04-12	0	1	0	840	NA	331002651.0	78.93	99.0	101.00	88.00	8.10	270857004.0	38.894992	-77.036560

In [6]:
```
def ny_cases(df,country,this_date,county):
    global ny_cases
    ny = df[df['Country/Region'].isin([country])]
    ny = ny[ny['date'].isin([this_date])]
    ny = ny[ny['Province/State'].isin(['Statewide'])]
    ny = ny[ny['Country_Code'].isin([840])]
    ny = ny[ny['Population'].isin([ny['Population'].max()])]
    ny = ny[ny['Urban population'].isin([ny['Urban population'].max()])]
    ny = ny[ny['Gross Primary Education Enrollment'].isin([ny['Gross Primary Education Enrollment'].max()])]
    ny = ny[ny['Gross Tertiary Education Enrollment'].isin([ny['Gross Tertiary Education Enrollment'].max()])]
    ny = ny[ny['Unemployment Rate'].isin([ny['Unemployment Rate'].max()])]
    ny = ny[ny['Life Expectancy'].isin([ny['Life Expectancy'].max()])]
    ny = ny[ny['Literacy Rate'].isin([ny['Literacy Rate'].max()])]
    ny = ny[ny['Latitude'].isin([ny['Latitude'].max()])]
    ny = ny[ny['Longitude'].isin([ny['Longitude'].max()])]
    ny = ny[ny[county].isin([ny[county].max()])]
    return ny
```

In [7]:
```
ny_cases(df,'US','2020-04-12','confirmed')
```

Out[7]:

	Unnamed: 0	Country/Region	Province/State	date	Recovered	confirmed	deaths	Country_code	Continent	Population	Life Expectancy	Literacy Rate	Gross Primary Education Enrollment	Gross Tertiary Education Enrollment	Unemployment Rate	Urban population	Latitude	Longitude
1945	1945	US	Statewide	2020-04-12	0	1	0	840	NA	331002651.0	78.93	99.0	101.00	88.00	8.10	270857004.0	38.894992	-77.036560

1 rows × 20 columns

NY New Cases Trends

In [8]:
```
ny_cases_trend = ny_cases(df,'US','2020-04-12','confirmed')
ny_cases_trend.shape
```

Out[8]:
```
(1, 20)
```

NY New Cases Statistics

In [9]:
```
ny_cases_trend.describe(include=['int64','float64'])
```

Out[9]:

	Unnamed: 0	Recovered	confirmed	deaths	Country_code	Population	Life Expectancy	Literacy Rate	Gross Primary Education Enrollment	Gross Tertiary Education Enrollment	Unemployment Rate	Urban population	Latitude	Longitude
count	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
mean	1945.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	1945.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
25%	1945.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
50%	1945.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
75%	1945.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
max	1945.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560

2 rows × 14 columns

Deaths

NY New Death Trends

In [10]:
```
ny_deaths_trend = ny_cases(df,'US','2020-04-12','deaths')
ny_deaths_trend.shape
```

Out[10]:
```
(1, 20)
```

NY New Death Statistics

In [11]:
```
ny_deaths_trend.describe(include=['int64','float64'])
```

Out[11]:

	Unnamed: 0	Recovered	confirmed	deaths	Country_code	Population	Life Expectancy	Literacy Rate	Gross Primary Education Enrollment	Gross Tertiary Education Enrollment	Unemployment Rate	Urban population	Latitude	Longitude
count	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
mean	1945.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	1945.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
25%	1945.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
50%	1945.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
75%	1945.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
max	1945.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560

2 rows × 14 columns

Comparing New Yorks Data to Other States

In [12]:
```
states = ['United States','Italy','Germany']
ny_states = df[df['Country/Region'].isin(states)]
ny_states.shape
```

Out[12]:
```
(1944, 20)
```

Normalization

In [13]:
```
Normalizing to numerical data by pandas/sklearn
ny_states_numerical = ny_states[['Unnamed: 0','Recovered','confirmed','deaths','Country_code','Population','Life Expectancy','Literacy Rate','Gross Primary Education Enrollment','Gross Tertiary Education Enrollment','Unemployment Rate','Urban population','Latitude','Longitude']]
ny_states_numerical.shape
```

Out[13]:
```
(1944, 14)
```

In [14]:
```
Printing first and last observations of numeric dataset
ny_states_numerical.head(2)
ny_states_numerical.tail(2)
```

Out[14]:

	Unnamed: 0	Recovered	confirmed	deaths	Country_code	Population	Life Expectancy	Literacy Rate	Gross Primary Education Enrollment	Gross Tertiary Education Enrollment	Unemployment Rate	Urban population	Latitude	Longitude
0	0	0	1	0	276	83900473.0	81.0	99.0	105.0	75.0	3.0	77189.0	51.000000	9.000000
1	1	0	1	0	276	83900473.0	81.0	99.0	105.0	75.0	3.0	77189.0	51.000000	9.000000
1942	1942	0	1	0	840	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
1943	1943	0	1	0	840	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560

Cases

In [15]:
```
Printing first and last observations of cases
ny_states_cases.head(2)
ny_states_cases.tail(2)
```

Out[15]:

	Unnamed: 0	Recovered	confirmed	deaths	Country_code	Population	Life Expectancy	Literacy Rate	Gross Primary Education Enrollment	Gross Tertiary Education Enrollment	Unemployment Rate	Urban population	Latitude	Longitude
0	0	0	1	0	276	83900473.0	81.0	99.0	105.0	75.0	3.0	77189.0	51.000000	9.000000
1	1	0	1	0	276	83900473.0	81.0	99.0	105.0	75.0	3.0	77189.0	51.000000	9.000000
1942	1942	0	1	0	840	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
1943	1943	0	1	0	840	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560

NY Deaths Statistics - Mean, Median, and Mode

In [16]:
```
ny_states_deaths.describe(include=['int64','float64'],percentiles=[.5])
```

Out[16]:

	Unnamed: 0	Recovered	confirmed	deaths	Country_code	Population	Life Expectancy	Literacy Rate	Gross Primary Education Enrollment	Gross Tertiary Education Enrollment	Unemployment Rate	Urban population	Latitude	Longitude
count	1944.0	1944.0	1944.0	1944.0	1944.0	1944.0	1944.0	1944.0	1944.0	1944.0	1944.0	1944.0	1944.0	1944.0
mean	1944.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	1944.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
50%	1944.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
max	1944.0	0.0	1.0	0.0	840.0	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560

Average Normalized Cases Across States

In [17]:
```
Average normalized cases across states
ny_avg = ny_states_numerical.groupby('Country_Code')
ny_avg.mean()[['confirmed']]
```

Out[17]:

Case Comparison Results

In [18]:
```
print("The US has the highest average cases.\n The US has a higher than average case ratio.")
```

Out[18]:
```
The US has the highest average cases.
 The US has a higher than average case ratio.
```

Normalized Death Deaths

In [19]:
```
Average normalized deaths across states
ny_avg = ny_states_numerical.groupby('Country_Code')
ny_avg.mean()[['deaths']]
```

Out[19]:

Average Deaths Across States

In [20]:
```
print("The US has the highest average deaths.\n The US has a higher than average death ratio.")
```

Out[20]:
```
The US has the highest average deaths.
 The US has a higher than average death ratio.
```

NY Counties With High Case and Death Rates

In [21]:
```
ny_counties = df[df['Country/Region'].isin(['US'])]
ny_counties = ny_counties[ny_counties['Province/State'].isin(['New York'])]
ny_counties[['Country_Code','Population']].mode()
```

Out[21]:

	Country_Code	Population
0	840	331002651.0

In [22]:
```
ny_counties[['Country_Code','Population']].shape
```

Out[22]:
```
(1945, 2)
```

In [23]:
```
ny_counties[['Country_Code','Population']].mode(axis=1)
```

Out[23]:

	Country_Code	Population
0	840.0	331002651.0
1	840.0	331002651.0
2	840.0	331002651.0
3	840.0	331002651.0
4	840.0	331002651.0
...	...	...
1940	840.0	331002651.0
1941	840.0	331002651.0
1942	840.0	331002651.0
1943	840.0	331002651.0
1944	840.0	331002651.0

1945 rows × 2 columns

Normalized Cases for All Counties in NY

In [24]:
```
ny_counties_normalized = ny_counties[['Unnamed: 0','Recovered','confirmed','deaths','Country_code','Population','Life Expectancy','Literacy Rate','Gross Primary Education Enrollment','Gross Tertiary Education Enrollment','Unemployment Rate','Urban population','Latitude','Longitude']]
ny_counties_normalized.shape
```

Out[24]:
```
(1945, 14)
```

In [25]:
```
ny_counties_normalized.head(2)
ny_counties_normalized.tail(2)
```

Out[25]:

	Unnamed: 0	Recovered	confirmed	deaths	Country_code	Population	Life Expectancy	Literacy Rate	Gross Primary Education Enrollment	Gross Tertiary Education Enrollment	Unemployment Rate	Urban population	Latitude	Longitude
0	0	0	1	0	276	83900473.0	81.0	99.0	105.0	75.0	3.0	77189.0	51.000000	9.000000
1	1	0	1	0	276	83900473.0	81.0	99.0	105.0	75.0	3.0	77189.0	51.000000	9.000000
1942	1942	0	1	0	840	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560
1943	1943	0	1	0	840	331002651.0	78.93	99.0	101.0	88.0	8.1	270857004.0	38.894992	-77.036560

Top 3 Counties With Highest Normalized Deaths

In [26]:
```
ny_counties_normalized[['Country_Code','confirmed','deaths']].sort_values(ascending=False).head(3)
```

Out[26]:

	Country_Code	confirmed	deaths
0	276	1	0
1	276	1	0
1942	840	1	0

Normalized Deaths in NY Counties

In [27]:
```
ny_counties_normalized[['Country_Code','confirmed','deaths']].sort_values(ascending=False).head(3)
```

Out[27]:

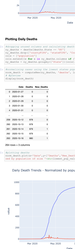
	Country_Code	confirmed	deaths
0	276	1	0
1	276	1	0
1942	840	1	0

Plotting Daily Trends - New York

Plotting Daily Cases

In [28]:
```
def plot_cases(df,country,county,date):
    global ny_df,ny,ny_cases,ny_deaths,ny_cases_trend,ny_deaths_trend,ny_states,ny_states_cases,ny_states_deaths,ny_counties,ny_counties_cases,ny_counties_deaths,ny_counties_cases_trend,ny_counties_deaths_trend
    ny = ny_df[ny_df['Country/Region'].isin([country])]
    ny = ny[ny['Province/State'].isin(['Statewide'])]
    ny = ny[ny['Country_Code'].isin([840])]
    ny = ny[ny['Population'].isin([ny['Population'].max()])]
    ny = ny[ny['Urban population'].isin([ny['Urban population'].max()])]
    ny = ny[ny['Gross Primary Education Enrollment'].isin([ny['Gross Primary Education Enrollment'].max()])]
    ny = ny[ny['Gross Tertiary Education Enrollment'].isin([ny['Gross Tertiary Education Enrollment'].max()])]
    ny = ny[ny['Unemployment Rate'].isin([ny['Unemployment Rate'].max()])]
    ny = ny[ny['Life Expectancy'].isin([ny['Life Expectancy'].max()])]
    ny = ny[ny['Literacy Rate'].isin([ny['Literacy Rate'].max()])]
    ny = ny[ny['Latitude'].isin([ny['Latitude'].max()])]
    ny = ny[ny['Longitude'].isin([ny['Longitude'].max()])]
    ny_cases = ny[ny[county].isin([ny[county].max()])]
    ny = ny_df[ny_df['Country/Region'].isin([country])]
    ny = ny[ny['Province/State'].isin(['Statewide'])]
    ny = ny[ny['Country_Code'].isin([840])]
    ny = ny[ny['Population'].isin([ny['Population'].max()])]
    ny = ny[ny['Urban population'].isin([ny['Urban population'].max()])]
    ny = ny[ny['Gross Primary Education Enrollment'].isin([ny['Gross Primary Education Enrollment'].max()])]
    ny = ny[ny['Gross Tertiary Education Enrollment'].isin([ny['Gross Tertiary Education Enrollment'].max()])]
    ny = ny[ny['Unemployment Rate'].isin([ny['Unemployment Rate'].max()])]
    ny = ny[ny['Life Expectancy'].isin([ny['Life Expectancy'].max()])]
    ny = ny[ny['Literacy Rate'].isin([ny['Literacy Rate'].max()])]
    ny = ny[ny['Latitude'].isin([ny['Latitude'].max()])]
    ny = ny[ny['Longitude'].isin([ny['Longitude'].max()])]
    ny_deaths = ny[ny[county].isin([ny[county].max()])]
    ny = ny_df[ny_df['Country/Region'].isin([country])]
    ny = ny[ny['Province/State'].isin(['Statewide'])]
    ny = ny[ny['Country_Code'].isin([840])]
    ny = ny[ny['Population'].isin([ny['Population'].max()])]
    ny = ny[ny['Urban population'].isin([ny['Urban population'].max()])]
    ny = ny[ny['Gross Primary Education Enrollment'].isin([ny['Gross Primary Education Enrollment'].max()])]
    ny = ny[ny['Gross Tertiary Education Enrollment'].isin([ny['Gross Tertiary Education Enrollment'].max()])]
    ny = ny[ny['Unemployment Rate'].isin([ny['Unemployment Rate'].max()])]
    ny = ny[ny['Life Expectancy'].isin([ny['Life Expectancy'].max()])]
    ny = ny[ny['Literacy Rate'].isin([ny['Literacy Rate'].max()])]
    ny = ny[ny['Latitude'].isin([ny['Latitude'].max()])]
    ny = ny[ny['Longitude'].isin([ny['Longitude'].max()])]
    ny_cases = ny[ny[county].isin([ny[county].max()])]
    ny = df[df['Country/Region'].isin(['US'])]
    ny = ny[ny['Province/State'].isin(['Statewide'])]
    ny = ny[ny[county].isin([ny[county].max()])]
    ny = ny[ny[county].isin([ny[county].max()])]
    ny = ny[ny[county].isin([ny[county].max()
```

```
Matplotlib data
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning -
Glyph 8722 missing from current font.
  font.set_text(s, 0.0, flags=flags)
```



Plotting Daily Deaths

```
In [24]: Matplotlib data
# Plotting Deaths and normalizing about size
ny_deaths = deaths[deaths['Province_State'] == 'New York']
ny_deaths = ny_deaths.groupby('Date').sum()
plt.plot(ny_deaths[ny_deaths.Deaths>0].Deaths, lw=2.5, ls='-', ms=10, c='r')
plt.legend(['Total Deaths'], loc='upper left')
plt.title('Daily Deaths Trends - Normalized by population of size 576759')
plt.xlabel('Date')
plt.ylabel('Deaths')
plt.xticks(rotation=30)
plt.tick_params(axis='x', which='major', labelsize=10)
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning -
Glyph 8722 missing from current font.
  font.set_text(s, 0.0, flags=flags)
```

Plotting Top 5 Counties in NY Daily Cases, Deaths, New Cases, and New Death Trends

```
In [25]: Matplotlib data
# Top 5 counties in NY
ny = df[df['Province_State'] == 'New York']
ny = ny.groupby('Date').sum()
ny['Daily Cases'] = ny.Cases - ny.Cases.shift(1)
ny['Daily Deaths'] = ny.Deaths - ny.Deaths.shift(1)
ny['New Cases'] = ny['Daily Cases'].diff()
ny['New Deaths'] = ny['Daily Deaths'].diff()
ny.tail(5)
```

Daily Top County Cases Normalized

```
In [26]: Matplotlib data
# Top 5 counties in NY
ny = df[df['Province_State'] == 'New York']
ny = ny.groupby('Date').sum()
ny['Daily Cases'] = ny.Cases - ny.Cases.shift(1)
ny['Daily Deaths'] = ny.Deaths - ny.Deaths.shift(1)
ny['New Cases'] = ny['Daily Cases'].diff()
ny['New Deaths'] = ny['Daily Deaths'].diff()
ny.tail(5)
for c in ny.columns:
    if c.find('Daily') > -1 or c.find('New') > -1:
        print('Top 5 Counties in NY Daily Cases Normalized', c)
        ny = df[df['Province_State'] == 'New York']
        ny = ny.groupby('Date').sum()
        ny['Daily Cases'] = ny.Cases - ny.Cases.shift(1)
        ny['Daily Deaths'] = ny.Deaths - ny.Deaths.shift(1)
        ny['New Cases'] = ny['Daily Cases'].diff()
        ny['New Deaths'] = ny['Daily Deaths'].diff()
        ny = ny[ny.Population > 0]
        ny = ny[ny.Population.isnull() == False]
        ny = ny[ny[c] > 0]
        ny = ny[ny[c].isnull() == False]
        ny = ny[ny[c] > -1]
        ny[c] = (ny[c]/ny.Population)*100000
        ny[c].unstack().plot(lw=2.5, ls='-', ms=10)
        plt.legend(ny[c].unstack().columns, loc='upper left')
        plt.title('Top 5 Counties in NY (Highest Case Rate) Daily Trends - Normalized by population')
        plt.xlabel('Date')
        plt.ylabel('Cases')
        plt.xticks(rotation=30)
        plt.tick_params(axis='x', which='major', labelsize=10)
```

```
Matplotlib data
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning -
Glyph 8722 missing from current font.
  font.set_text(s, 0.0, flags=flags)
```

```
Matplotlib data
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning -
Glyph 8722 missing from current font.
  font.set_text(s, 0.0, flags=flags)
```

```
Matplotlib data
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning -
Glyph 8722 missing from current font.
  font.set_text(s, 0.0, flags=flags)
```

```
Matplotlib data
/usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning -
Glyph 8722 missing from current font.
  font.set_text(s, 0.0, flags=flags)
```

Member Task 8

Fitting a Distribution To New York Cases

```
In [27]: Matplotlib data
# Fitting a distribution
import sys,os,numpy,numpy.stats
from numpy import *
from numpy.lib.scimath import *
from scipy import stats,optimize,interpolate
from scipy.special import *
from pylab import *
from matplotlib.mlab import *
from scipy.stats.distributions import rv_continuous, rv_discrete, entropy
from numpy.fft.fftpack import *
from numpy.random.mtrand import *
from numpy.linalg.linalg import *
from numpy.ma.core import *
from numpy.ma.extras import *
from numpy.lib.arraysetops import *
from numpy.lib.function_base import *
from numpy.lib.type_check import *
from numpy.lib.index_tricks import *
from numpy.matrixlib.defmatrix import *
from numpy.polynomial.polynomial import Polynomial
from numpy.polynomial.chebyshev import Chebyshev
from numpy.polynomial.hermite import Hermite
from numpy.polynomial.hermite_e import HermiteE
from numpy.polynomial.laguerre import Laguerre
from numpy.polynomial.legendre import Legendre
from numpy.polynomial.polynomial.polyfit as polyfit
from numpy.polynomial.polynomial.polyval as polyval
from numpy.polynomial.polynomial.polyvander as polyvander
from numpy.polynomial.polynomial.polyvander2d as polyvander2d
from numpy.polynomial.polynomial.polyvander3d as polyvander3d
from numpy.polynomial.polynomial.polycoefficients as polycoefficients
from numpy.polynomial.polynomial.polyroots as polyroots
from numpy.polynomial.polynomial.polyfromroots as polyfromroots
from numpy.polynomial.polynomial.polyone as polyone
from numpy.polynomial.polynomial.polyzero as polyzero
from numpy.polynomial.polynomial.polytrim as polytrim
from numpy.polynomial.polynomial.polyline as polyline
from numpy.polynomial.polynomial.polyadd as polyadd
from numpy.polynomial.polynomial.polysub as polysub
from numpy.polynomial.polynomial.polymulx as polymulx
from numpy.polynomial.polynomial.polymul as polymul
from numpy.polynomial.polynomial.polydiv as polydiv
from numpy.polynomial.polynomial.polypow as polypow
from numpy.polynomial.polynomial.polyint as polyint
from numpy.polynomial.polynomial.polyder as polyder
from numpy.polynomial.polynomial.polyfit as polyfit
from numpy.polynomial.polynomial.polytrim as polytrim
from numpy.polynomial.polynomial.polyline as polyline
from numpy.polynomial.polynomial.polyadd as polyadd
from numpy.polynomial.polynomial.polysub as polysub
from numpy.polynomial.polynomial.polymulx as polymulx
from numpy.polynomial.polynomial.polymul as polymul
from numpy.polynomial.polynomial.polydiv as polydiv
from numpy.polynomial.polynomial.polypow as polypow
from numpy.polynomial.polynomial.polyint as polyint
from numpy.polynomial.polynomial.polyder as polyder
```

Poisson Distribution - This distribution was chosen because it is discrete and it makes sense to attempt to predict the probability of a number of cases happening at a specific time.

```
In [28]: Matplotlib data
[nan nan nan
```

```
plot_poisson_predictions(df_county_fit, df_county_test, "Cattaraugus", "Cattaraugus County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```

```
# Cattaraugus County
df_county_fit <- df_county_fit %>%
  mutate(Cattaraugus = ifelse(County == "Cattaraugus", Cases, 0))
df_county_test <- df_county_test %>%
  mutate(Cattaraugus = ifelse(County == "Cattaraugus", Cases, 0))
cat("Cattaraugus County Fit Data:\n")
head(df_county_fit %>% filter(Cattaraugus > 0), 5)
cat("\n")
cat("Cattaraugus County Test Data:\n")
head(df_county_test %>% filter(Cattaraugus > 0), 5)
```

```
Cattaraugus County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Cattaraugus County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Chautauqua", "Chautauqua County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```

```
# Chautauqua County
df_county_fit <- df_county_fit %>%
  mutate(Chautauqua = ifelse(County == "Chautauqua", Cases, 0))
df_county_test <- df_county_test %>%
  mutate(Chautauqua = ifelse(County == "Chautauqua", Cases, 0))
cat("Chautauqua County Fit Data:\n")
head(df_county_fit %>% filter(Chautauqua > 0), 5)
cat("\n")
cat("Chautauqua County Test Data:\n")
head(df_county_test %>% filter(Chautauqua > 0), 5)
```

```
Chautauqua County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Chautauqua County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Erie", "Erie County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```

```
# Erie County
df_county_fit <- df_county_fit %>%
  mutate(Erie = ifelse(County == "Erie", Cases, 0))
df_county_test <- df_county_test %>%
  mutate(Erie = ifelse(County == "Erie", Cases, 0))
cat("Erie County Fit Data:\n")
head(df_county_fit %>% filter(Erie > 0), 5)
cat("\n")
cat("Erie County Test Data:\n")
head(df_county_test %>% filter(Erie > 0), 5)
```

```
Erie County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Erie County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Genesee", "Genesee County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```

```
# Genesee County
df_county_fit <- df_county_fit %>%
  mutate(Genesee = ifelse(County == "Genesee", Cases, 0))
df_county_test <- df_county_test %>%
  mutate(Genesee = ifelse(County == "Genesee", Cases, 0))
cat("Genesee County Fit Data:\n")
head(df_county_fit %>% filter(Genesee > 0), 5)
cat("\n")
cat("Genesee County Test Data:\n")
head(df_county_test %>% filter(Genesee > 0), 5)
```

```
Genesee County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Genesee County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Livingston", "Livingston County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```

```
# Livingston County
df_county_fit <- df_county_fit %>
  mutate(Livingston = ifelse(County == "Livingston", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Livingston = ifelse(County == "Livingston", Cases, 0))
cat("Livingston County Fit Data:\n")
head(df_county_fit %>% filter(Livingston > 0), 5)
cat("\n")
cat("Livingston County Test Data:\n")
head(df_county_test %>% filter(Livingston > 0), 5)
```

```
Livingston County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Livingston County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Monroe", "Monroe County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```

```
# Monroe County
df_county_fit <- df_county_fit %>
  mutate(Monroe = ifelse(County == "Monroe", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Monroe = ifelse(County == "Monroe", Cases, 0))
cat("Monroe County Fit Data:\n")
head(df_county_fit %>% filter(Monroe > 0), 5)
cat("\n")
cat("Monroe County Test Data:\n")
head(df_county_test %>% filter(Monroe > 0), 5)
```

```
Monroe County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Monroe County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Niagara", "Niagara County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```

```
# Niagara County
df_county_fit <- df_county_fit %>
  mutate(Niagara = ifelse(County == "Niagara", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Niagara = ifelse(County == "Niagara", Cases, 0))
cat("Niagara County Fit Data:\n")
head(df_county_fit %>% filter(Niagara > 0), 5)
cat("\n")
cat("Niagara County Test Data:\n")
head(df_county_test %>% filter(Niagara > 0), 5)
```

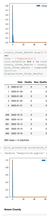
```
Niagara County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Niagara County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Orleans", "Orleans County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```




```
# Orleans County
df_county_fit <- df_county_fit %>
  mutate(Orleans = ifelse(County == "Orleans", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Orleans = ifelse(County == "Orleans", Cases, 0))
cat("Orleans County Fit Data:\n")
head(df_county_fit %>% filter(Orleans > 0), 5)
cat("\n")
cat("Orleans County Test Data:\n")
head(df_county_test %>% filter(Orleans > 0), 5)
```

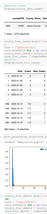
```
Orleans County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Orleans County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Allegany", "Allegany County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```




```
# Allegany County
df_county_fit <- df_county_fit %>
  mutate(Allegany = ifelse(County == "Allegany", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Allegany = ifelse(County == "Allegany", Cases, 0))
cat("Allegany County Fit Data:\n")
head(df_county_fit %>% filter(Allegany > 0), 5)
cat("\n")
cat("Allegany County Test Data:\n")
head(df_county_test %>% filter(Allegany > 0), 5)
```

```
Allegany County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Allegany County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Schoharie", "Schoharie County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```

```
# Schoharie County
df_county_fit <- df_county_fit %>
  mutate(Schoharie = ifelse(County == "Schoharie", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Schoharie = ifelse(County == "Schoharie", Cases, 0))
cat("Schoharie County Fit Data:\n")
head(df_county_fit %>% filter(Schoharie > 0), 5)
cat("\n")
cat("Schoharie County Test Data:\n")
head(df_county_test %>% filter(Schoharie > 0), 5)
```

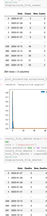
```
Schoharie County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Schoharie County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Steuben", "Steuben County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```




```
# Steuben County
df_county_fit <- df_county_fit %>
  mutate(Steuben = ifelse(County == "Steuben", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Steuben = ifelse(County == "Steuben", Cases, 0))
cat("Steuben County Fit Data:\n")
head(df_county_fit %>% filter(Steuben > 0), 5)
cat("\n")
cat("Steuben County Test Data:\n")
head(df_county_test %>% filter(Steuben > 0), 5)
```

```
Steuben County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Steuben County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Wyoming", "Wyoming County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```

```
# Wyoming County
df_county_fit <- df_county_fit %>
  mutate(Wyoming = ifelse(County == "Wyoming", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Wyoming = ifelse(County == "Wyoming", Cases, 0))
cat("Wyoming County Fit Data:\n")
head(df_county_fit %>% filter(Wyoming > 0), 5)
cat("\n")
cat("Wyoming County Test Data:\n")
head(df_county_test %>% filter(Wyoming > 0), 5)
```

```
Wyoming County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Wyoming County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Yates", "Yates County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Yates County
df_county_fit <- df_county_fit %>
  mutate(Yates = ifelse(County == "Yates", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Yates = ifelse(County == "Yates", Cases, 0))
cat("Yates County Fit Data:\n")
head(df_county_fit %>% filter(Yates > 0), 5)
cat("\n")
cat("Yates County Test Data:\n")
head(df_county_test %>% filter(Yates > 0), 5)
```

```
Yates County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Yates County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Seneca", "Seneca County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Seneca County
df_county_fit <- df_county_fit %>
  mutate(Seneca = ifelse(County == "Seneca", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Seneca = ifelse(County == "Seneca", Cases, 0))
cat("Seneca County Fit Data:\n")
head(df_county_fit %>% filter(Seneca > 0), 5)
cat("\n")
cat("Seneca County Test Data:\n")
head(df_county_test %>% filter(Seneca > 0), 5)
```

```
Seneca County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Seneca County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Ontario", "Ontario County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Ontario County
df_county_fit <- df_county_fit %>
  mutate(Ontario = ifelse(County == "Ontario", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Ontario = ifelse(County == "Ontario", Cases, 0))
cat("Ontario County Fit Data:\n")
head(df_county_fit %>% filter(Ontario > 0), 5)
cat("\n")
cat("Ontario County Test Data:\n")
head(df_county_test %>% filter(Ontario > 0), 5)
```

```
Ontario County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Ontario County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Wayne", "Wayne County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Wayne County
df_county_fit <- df_county_fit %>
  mutate(Wayne = ifelse(County == "Wayne", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Wayne = ifelse(County == "Wayne", Cases, 0))
cat("Wayne County Fit Data:\n")
head(df_county_fit %>% filter(Wayne > 0), 5)
cat("\n")
cat("Wayne County Test Data:\n")
head(df_county_test %>% filter(Wayne > 0), 5)
```

```
Wayne County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Wayne County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Cortland", "Cortland County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Cortland County
df_county_fit <- df_county_fit %>
  mutate(Cortland = ifelse(County == "Cortland", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Cortland = ifelse(County == "Cortland", Cases, 0))
cat("Cortland County Fit Data:\n")
head(df_county_fit %>% filter(Cortland > 0), 5)
cat("\n")
cat("Cortland County Test Data:\n")
head(df_county_test %>% filter(Cortland > 0), 5)
```

```
Cortland County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Cortland County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Tompkins", "Tompkins County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Tompkins County
df_county_fit <- df_county_fit %>
  mutate(Tompkins = ifelse(County == "Tompkins", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Tompkins = ifelse(County == "Tompkins", Cases, 0))
cat("Tompkins County Fit Data:\n")
head(df_county_fit %>% filter(Tompkins > 0), 5)
cat("\n")
cat("Tompkins County Test Data:\n")
head(df_county_test %>% filter(Tompkins > 0), 5)
```

```
Tompkins County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Tompkins County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Cayuga", "Cayuga County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Cayuga County
df_county_fit <- df_county_fit %>
  mutate(Cayuga = ifelse(County == "Cayuga", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Cayuga = ifelse(County == "Cayuga", Cases, 0))
cat("Cayuga County Fit Data:\n")
head(df_county_fit %>% filter(Cayuga > 0), 5)
cat("\n")
cat("Cayuga County Test Data:\n")
head(df_county_test %>% filter(Cayuga > 0), 5)
```

```
Cayuga County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Cayuga County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Schuyler", "Schuyler County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Schuyler County
df_county_fit <- df_county_fit %>
  mutate(Schuyler = ifelse(County == "Schuyler", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Schuyler = ifelse(County == "Schuyler", Cases, 0))
cat("Schuyler County Fit Data:\n")
head(df_county_fit %>% filter(Schuyler > 0), 5)
cat("\n")
cat("Schuyler County Test Data:\n")
head(df_county_test %>% filter(Schuyler > 0), 5)
```

```
Schuyler County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Schuyler County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Chemung", "Chemung County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Chemung County
df_county_fit <- df_county_fit %>
  mutate(Chemung = ifelse(County == "Chemung", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Chemung = ifelse(County == "Chemung", Cases, 0))
cat("Chemung County Fit Data:\n")
head(df_county_fit %>% filter(Chemung > 0), 5)
cat("\n")
cat("Chemung County Test Data:\n")
head(df_county_test %>% filter(Chemung > 0), 5)
```

```
Chemung County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Chemung County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Tioga", "Tioga County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Tioga County
df_county_fit <- df_county_fit %>
  mutate(Tioga = ifelse(County == "Tioga", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Tioga = ifelse(County == "Tioga", Cases, 0))
cat("Tioga County Fit Data:\n")
head(df_county_fit %>% filter(Tioga > 0), 5)
cat("\n")
cat("Tioga County Test Data:\n")
head(df_county_test %>% filter(Tioga > 0), 5)
```

```
Tioga County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Tioga County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Delaware", "Delaware County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Delaware County
df_county_fit <- df_county_fit %>
  mutate(Delaware = ifelse(County == "Delaware", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Delaware = ifelse(County == "Delaware", Cases, 0))
cat("Delaware County Fit Data:\n")
head(df_county_fit %>% filter(Delaware > 0), 5)
cat("\n")
cat("Delaware County Test Data:\n")
head(df_county_test %>% filter(Delaware > 0), 5)
```

```
Delaware County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Delaware County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Sullivan", "Sullivan County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Sullivan County
df_county_fit <- df_county_fit %>
  mutate(Sullivan = ifelse(County == "Sullivan", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Sullivan = ifelse(County == "Sullivan", Cases, 0))
cat("Sullivan County Fit Data:\n")
head(df_county_fit %>% filter(Sullivan > 0), 5)
cat("\n")
cat("Sullivan County Test Data:\n")
head(df_county_test %>% filter(Sullivan > 0), 5)
```

```
Sullivan County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Sullivan County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Otsego", "Otsego County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Otsego County
df_county_fit <- df_county_fit %>
  mutate(Otsego = ifelse(County == "Otsego", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Otsego = ifelse(County == "Otsego", Cases, 0))
cat("Otsego County Fit Data:\n")
head(df_county_fit %>% filter(Otsego > 0), 5)
cat("\n")
cat("Otsego County Test Data:\n")
head(df_county_test %>% filter(Otsego > 0), 5)
```

```
Otsego County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Otsego County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Herkimer", "Herkimer County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Herkimer County
df_county_fit <- df_county_fit %>
  mutate(Herkimer = ifelse(County == "Herkimer", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Herkimer = ifelse(County == "Herkimer", Cases, 0))
cat("Herkimer County Fit Data:\n")
head(df_county_fit %>% filter(Herkimer > 0), 5)
cat("\n")
cat("Herkimer County Test Data:\n")
head(df_county_test %>% filter(Herkimer > 0), 5)
```

```
Herkimer County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Herkimer County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Madison", "Madison County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Madison County
df_county_fit <- df_county_fit %>
  mutate(Madison = ifelse(County == "Madison", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Madison = ifelse(County == "Madison", Cases, 0))
cat("Madison County Fit Data:\n")
head(df_county_fit %>% filter(Madison > 0), 5)
cat("\n")
cat("Madison County Test Data:\n")
head(df_county_test %>% filter(Madison > 0), 5)
```

```
Madison County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Madison County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Oneida", "Oneida County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Oneida County
df_county_fit <- df_county_fit %>
  mutate(Oneida = ifelse(County == "Oneida", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Oneida = ifelse(County == "Oneida", Cases, 0))
cat("Oneida County Fit Data:\n")
head(df_county_fit %>% filter(Oneida > 0), 5)
cat("\n")
cat("Oneida County Test Data:\n")
head(df_county_test %>% filter(Oneida > 0), 5)
```

```
Oneida County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     1       0

Oneida County Test Data:
  Date  Cases Max_Quality
1 2020-08-26     1       0
2 2020-09-02     1       0
3 2020-09-09     1       0
4 2020-09-16     1       0
5 2020-09-23     1       0
```

200 rows | 5 columns

```
plot_poisson_predictions(df_county_fit, df_county_test, "Onondaga", "Onondaga County")
```

```
Warning: Transform to period is deprecated. Use `transformation = TRUE` instead.
```



```
# Onondaga County
df_county_fit <- df_county_fit %>
  mutate(Onondaga = ifelse(County == "Onondaga", Cases, 0))
df_county_test <- df_county_test %>
  mutate(Onondaga = ifelse(County == "Onondaga", Cases, 0))
cat("Onondaga County Fit Data:\n")
head(df_county_fit %>% filter(Onondaga > 0), 5)
cat("\n")
cat("Onondaga County Test Data:\n")
head(df_county_test %>% filter(Onondaga > 0), 5)
```

```
Onondaga County Fit Data:
  Date  Cases Max_Quality
1 2020-07-22     1       0
2 2020-07-29     1       0
3 2020-08-05     1       0
4 2020-08-12     1       0
5 2020-08-19     
```

In [1]:
```
DATASET = "/kaggle/input/TRAIN_DATASET"
TEST_DATASET = "/kaggle/input/TEST_DATASET"
SAMPLE_SUBMISSION = "/kaggle/input/SAMPLE_SUBMISSION"
train_df = pd.read_csv(DATASET, sep=';', low_memory=False)
display(train_df.head()
```

Out[1]:

| | Year | Month | Day | Hour | Minute | Second | Client_ID | Session_ID | Segment | Device | Sessions_count | Clicks | Days_since_last_session | Hits | Visits | Sessions_duration | Bounces | Transactions | Revenue | Transactions_per_session | Transactions_per_user | Goal_1_Completions | Goal_2_Completions | Goal_3_Completions | Goal_4_Completions | Pages_per_session | Bounce_rate | Page_views | Avg_session_duration | Avg_time_on_page | Exits | Rejections | Avg_time_to_first_interaction | Avg_time_to_last_interaction | Plays | Unique_Plays | Avg_Play_Time | Total_Play_Time | Unique_Events | Event_Value | Avg_Value | Sessions | Drop_Offs | Score | Total_Events | Days_to_Transaction | Sessions_to_Transaction | Frequency | Search_Sessions | Total_Page_Views | Avg_Page_Load_Time | Product_Views | Avg_Product_Views | Search_Depth | Avg_Search_Depth | Total_Screenviews | Unique_Screenviews | Screenviews_per_session | Total_Screenviews | Unique_Screenviews | Screenviews_per_session | Total_Screenviews | Unique_Screenviews | Screenviews_per_session