Task 1

November 24, 2021

1 F06 G05 2D project 2021

In this project, we are tasked with supporting WHO in developing a model to predict the number of deaths in each country. There are several factors that contribute to the deaths caused by covid, but not all data can be accurately transcribed into values that can be used to predict deaths in a nation or region. However, information such as the number of infected patients, a country's wealth, facilities, the number of vaccinated, the number of completely vaccinated, and the human development index can easily acquire and be incorporated in our model. Considering data from the actual world is not perfectly linear, we will look for data that are strongly correlated. We will be able to predict the number of deaths in a specific nation with assumptions made in any given day using this model.

In this project, we will be using 5 external libraries to aid us in building our model. The libraries are mainly pandas, numpy, seaborn, matplotlib and math. This project is tested and working on python v3.8.3. The cell below will install the requirements for this models.

[ ]:
```
pip install -r requirements.txt
```

[ ]:
```
from IPython.display import display
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math
```

From cohort and homework, we will be using similar functions such as: - Maximum and minimum normalisation. - Convertion of x and y axis to numpy array - Data splitting - r^2 score - Mean square error - Prediction

[ ]:
```
def normalize_z(df):
    return (df - df.min(axis=0)) / (df.max(axis=0) - df.min(axis=0))


def normalize_predict(df, dfx):
    return (dfx - df.min(axis=0)) / (df.max(axis=0) - df.min(axis=0))


def prepare_x(df_x):
    xAxis = df_x.to_numpy()
```

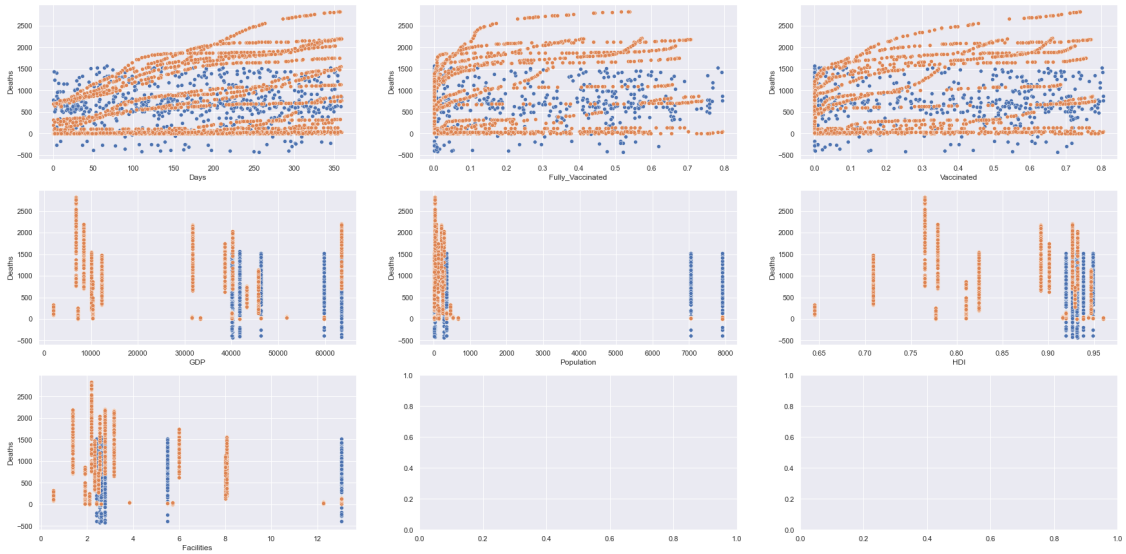
```
    array1 = np.ones((df_x.shape[0], 1))
    return np.concatenate((array1, xAxis), axis=1)


def prepare_y(df_y):
    return df_y.to_numpy()


def predict(df, df_x, beta):
    x = prepare_x(normalize_predict(df, df_x))
    return np.matmul(x, beta)

def split_data(df_x, df_y, random_state=None, test_size=0.5):
    np.random.seed(random_state)
    feature_index = df_x.index

    test_index = np.random.choice(feature_index, size=int(test_size *␣
 ↪len(df_x)), replace=False)
    train_index = [x for x in range(len(feature_index)) if x not in test_index]

    df_x_train = df_x.loc[train_index, :]
    df_x_test = df_x.loc[test_index, :]
    df_y_train = df_y.loc[train_index, :]
    df_y_test = df_y.loc[test_index, :]
    return df_x_train, df_x_test, df_y_train, df_y_test


def r2_score(y, ypred):
    ybar = np.mean(y)
    SStot = np.sum((y - ybar) ** 2)
    SSres = np.sum((y - ypred) ** 2)
    return 1 - SSres / SStot


def mean_squared_error(target, pred):
    n = target.shape[0]
    s = np.sum((target - pred) ** 2)
    return (1 / n) * s


def gradient_descent(X, y, beta, alpha, num_iters):
    m = X.shape[0]
    J_storage = np.zeros((num_iters, 1))
    for n in range(num_iters):
        yp = np.matmul(X, beta)
        error = yp - y
        delta = np.matmul(X.T, error)
```

```
        beta = beta - (alpha/m)*delta
        J_storage[n] = compute_cost(X,y,beta)
    return beta, J_storage


def compute_cost(X, y, beta):
    size = X.shape[0]
    yp = np.matmul(X, beta)
    error = yp - y
    J = (1/(2*size))*np.matmul(error.T, error)
    return J

def NormalisePredict(df, ndf):
    Normalmin = df.min(axis=0)
    Normalmax = df.max(axis=0)
    return (ndf - Normalmin) / (Normalmax - Normalmin)
```

[ ]:
```
dataFrame = pd.read_csv("2Ddata7.csv")
# columns = dataFrame.columns
# print(columns)
# sg = pd.DataFrame.reset_index(dataFrame.loc[(dataFrame['location'] ==␣
 ↪"Singapore"), :])
# dataFrame = pd.DataFrame.reset_index(dataFrame.loc[(pd.
 ↪to_datetime(dataFrame["date"] >= pd.to_datetime("10/19/2020"))), :])
yVal = pd.DataFrame(dataFrame.loc[:, "total_deaths_per_million"].fillna(0))
xVal = pd.DataFrame(columns=[])
#xVal["Days"] = sg["date"].apply(lambda data: (pd.to_datetime(data) - pd.
 ↪to_datetime(sg.loc[0, "date"])).days)
xVal["Location"] = pd.DataFrame(dataFrame["location"].fillna(0))
xVal["Days"] = pd.DataFrame(dataFrame["Days"].fillna(0))
xVal["Fully_Vaccinated"] = pd.
 ↪DataFrame(dataFrame["fully_vaccination_percentage"].fillna(0))
xVal["Vaccinated"] = pd.DataFrame(dataFrame["vaccination_percentage"].fillna(0))
xVal["GDP"] = pd.DataFrame(dataFrame["gdp_per_capita"].fillna(0))
xVal["Population"] = pd.DataFrame(dataFrame["population_density"].fillna(0))
xVal["HDI"] = pd.DataFrame(dataFrame["human_development_index"].fillna(0))
xVal["Facilities"] = pd.DataFrame(dataFrame["hospital_beds_per_thousand"].
 ↪fillna(0))

countryData = {}
countryYData = {}
countries = xVal["Location"].drop_duplicates()
for x in countries:
    countryData[x] = pd.DataFrame.reset_index(xVal.loc[(dataFrame['location']␣
 ↪== x), :])
    countryData[x] = countryData[x][countryData[x].columns[2:-4]]
```

```
    countryYData[x] = pd.DataFrame(pd.DataFrame.reset_index(dataFrame.
 ↪loc[(dataFrame['location'] == x), :])["total_deaths_per_million"])

xVal = xVal[xVal.columns[1:]]
xVal_train, xVal_test, yVal_train, yVal_test = split_data(xVal, yVal, 70, 0.3)
xMVal = normalize_z(xVal_train)

xAxis = prepare_x(xMVal)
yAxis = prepare_y(yVal_train)

iterations = 10500
alpha = 0.05
beta = np.zeros((xAxis.shape[1], 1))

beta, dummy = gradient_descent(xAxis, yAxis, beta, alpha, iterations)
```

[ ]:
```
prediction = pd.DataFrame(predict(xVal_train, xVal_test, beta))
fig, axs = plt.subplots(math.ceil(len(xVal_train.columns)/3), 3, figsize=(30,␣
 ↪15))
for index, x in enumerate(xVal.columns):
    sns.set()
    scat0 = sns.scatterplot(x=xVal_test[x], y=prediction[0], ax=axs[int(index/
 ↪3), index%3,])
    scat1 = sns.scatterplot(x=xVal_test[x],␣
 ↪y=yVal_test["total_deaths_per_million"], ax=axs[int(index/3), index%3,])

    scat0.set_ylabel("Deaths")
    scat0.set_xlabel(x)
plt.show()

print("r^2 Value:", r2_score(prepare_y(yVal_test), prediction))
display(beta)
display(xVal_test.columns)
```




```
r^2 Value: 0    0.341216
dtype: float64
```

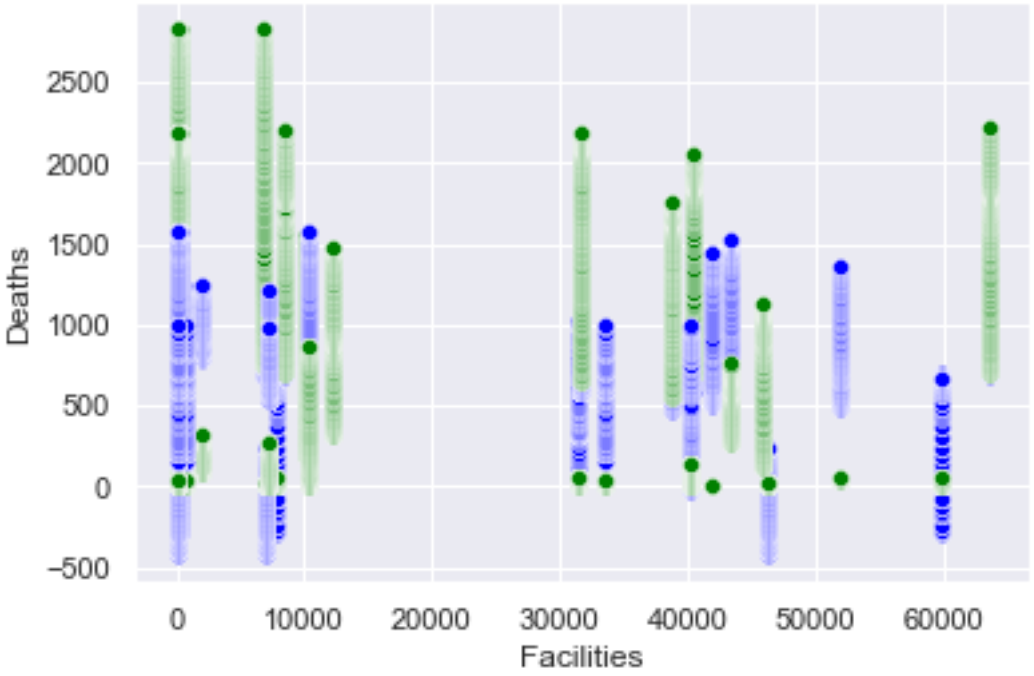
```
array([[1024.76933018],
       [-564.74535452],
       [ 151.31257712],
       [1241.42043476],
       [ 146.88736799],
       [-939.40304397],
       [-330.46341833],
       [-558.67830637]])
```

```
Index(['Days', 'Fully_Vaccinated', 'Vaccinated', 'GDP', 'Population', 'HDI',
       'Facilities'],
      dtype='object')
```

[ ]:
```
for i in countryData:
    prediction = pd.DataFrame(predict(xVal_train, countryData[i], beta))
    print(i)
    for x in xVal.columns:
        sns.set()

        scat0 = sns.scatterplot(x=countryData[i][x], y=prediction[0],␣
↪color="blue")
        scat1 = sns.scatterplot(x=countryData[i][x],␣
↪y=countryYData[i]["total_deaths_per_million"], color="green")

        scat0.set_ylabel("Deaths")
```

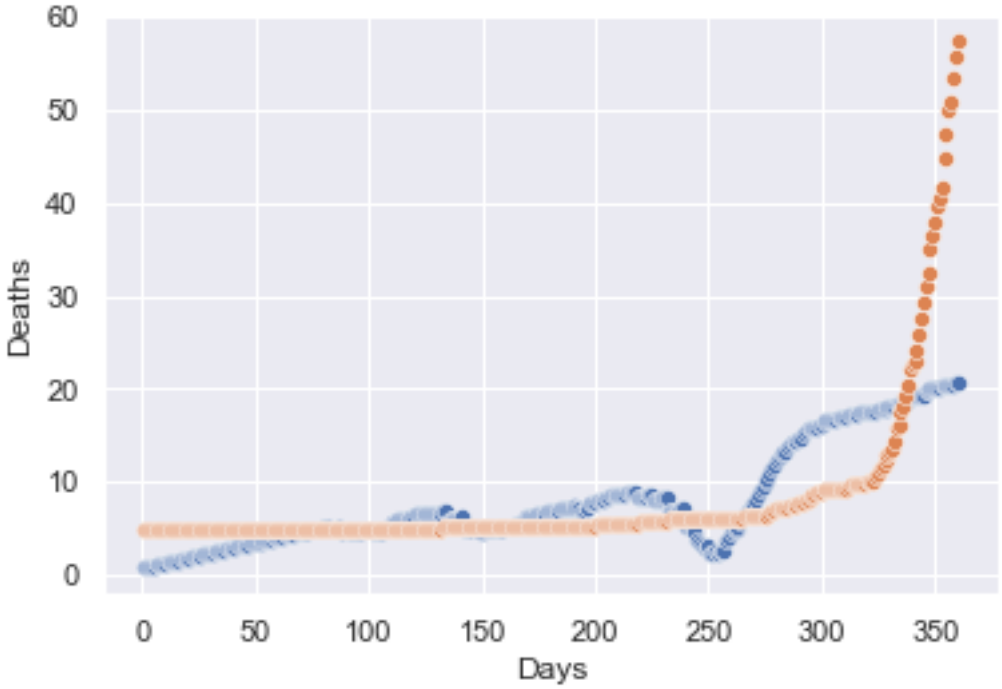
```
        scat0.set_xlabel(x)
        # plt.show()
```

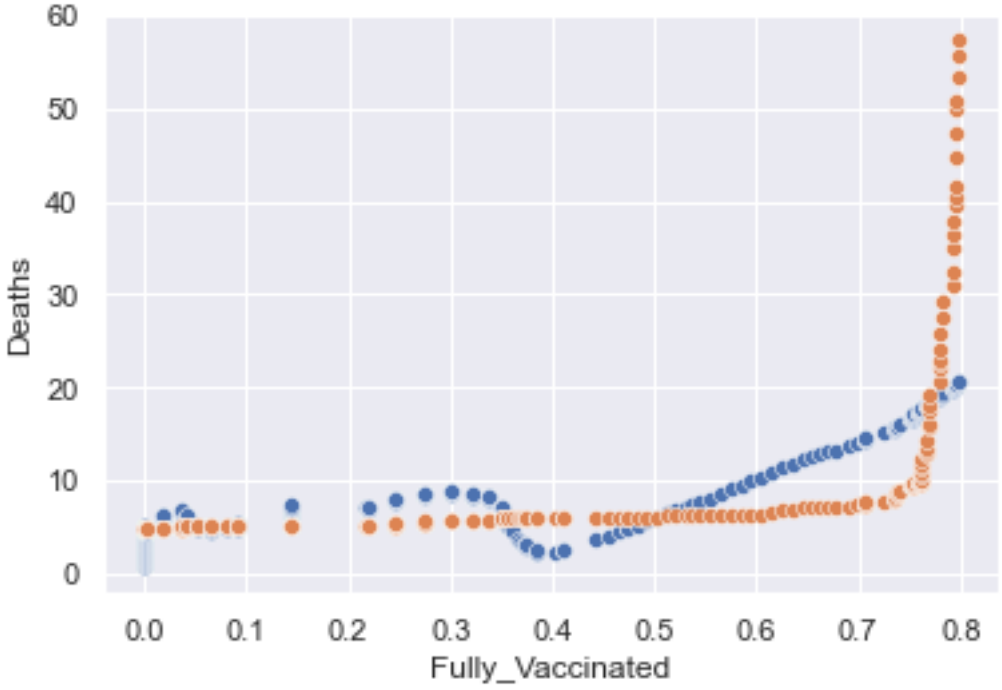
```
Singapore
United States
United Kingdom
Japan
Hong Kong
New Zealand
Australia
India
South Africa
Mexico
Russia
Brazil
South Korea
Germany
Thailand
France
Canada
Italy
Malaysia
Taiwan
```

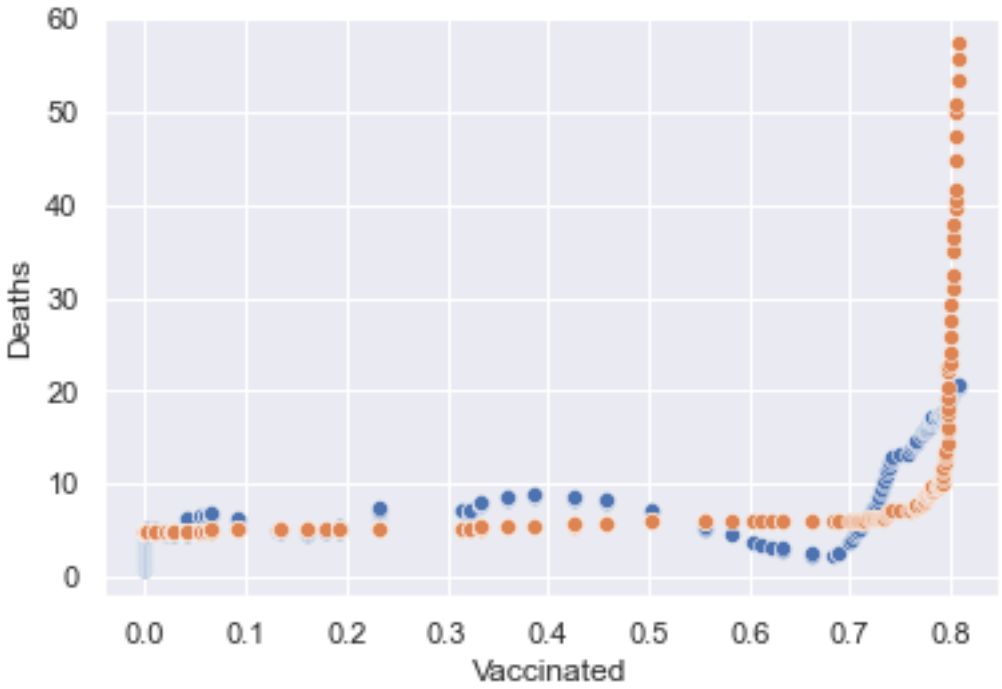


[ ]:
```
for i in countryData:
    xVal_train, xVal_test, yVal_train, yVal_test = split_data(countryData[i],␣
↪countryYData[i], 100, 0.3)
    xMVal = normalize_z(xVal_train)

    xAxis = prepare_x(xMVal)
    yAxis = prepare_y(yVal_train)

    iterations = 10500
    alpha = 0.05
    beta = np.zeros((xAxis.shape[1], 1))

    beta, dummy = gradient_descent(xAxis, yAxis, beta, alpha, iterations)
    # prediction = pd.DataFrame(predict(xVal_train, xVal_test, beta))
    prediction = pd.DataFrame(predict(xVal_train, countryData[i], beta))
    print(i)
    for x in xVal_train.columns:
        sns.set()

        scat0 = sns.scatterplot(x=countryData[i][x], y=prediction[0])
        scat1 = sns.scatterplot(x=countryData[i][x],␣
↪y=countryYData[i]["total_deaths_per_million"])
        # scat0 = sns.scatterplot(x=xVal_test[x], y=prediction[0])
        # scat1 = sns.scatterplot(x=xVal_test[x],␣
↪y=yVal_test["total_deaths_per_million"])

        scat0.set_ylabel("Deaths")
        scat0.set_xlabel(x)
        plt.show()

    print(yVal_test.shape, prediction.shape)
    print("r^2 Value:", r2_score(prepare_y(countryYData[i]), prediction))
    display(beta)
```

Singapore


Deaths
Days
0
10
20
30
40
50
60
0
50
100
150
200
250
300
350

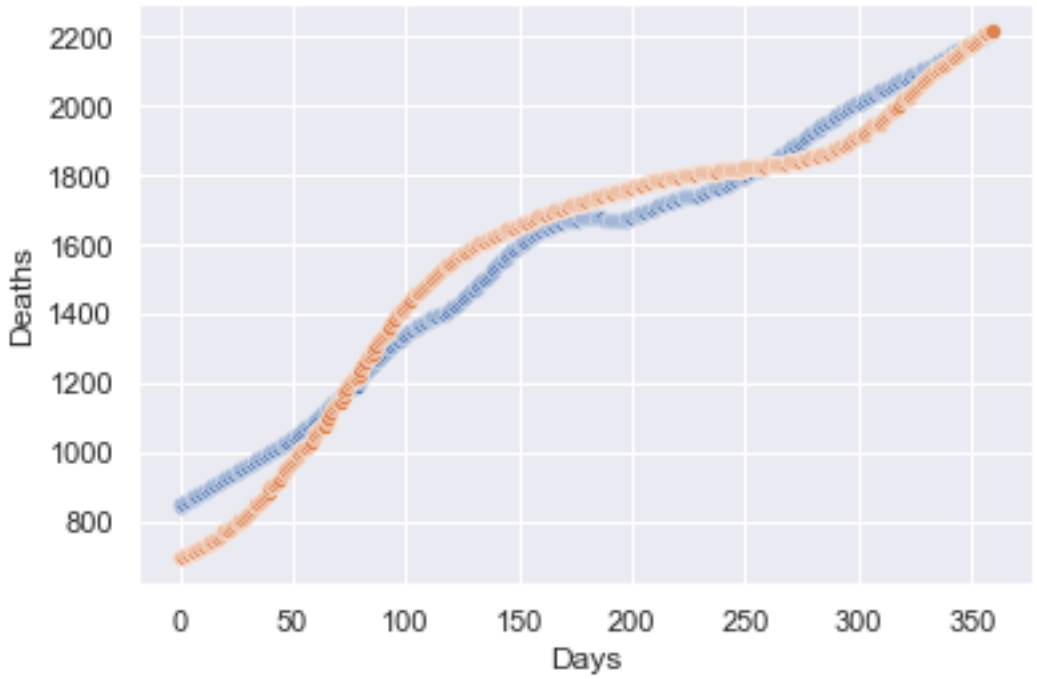

Deaths
Fully_Vaccinated
0
10
20
30
40
50
60
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

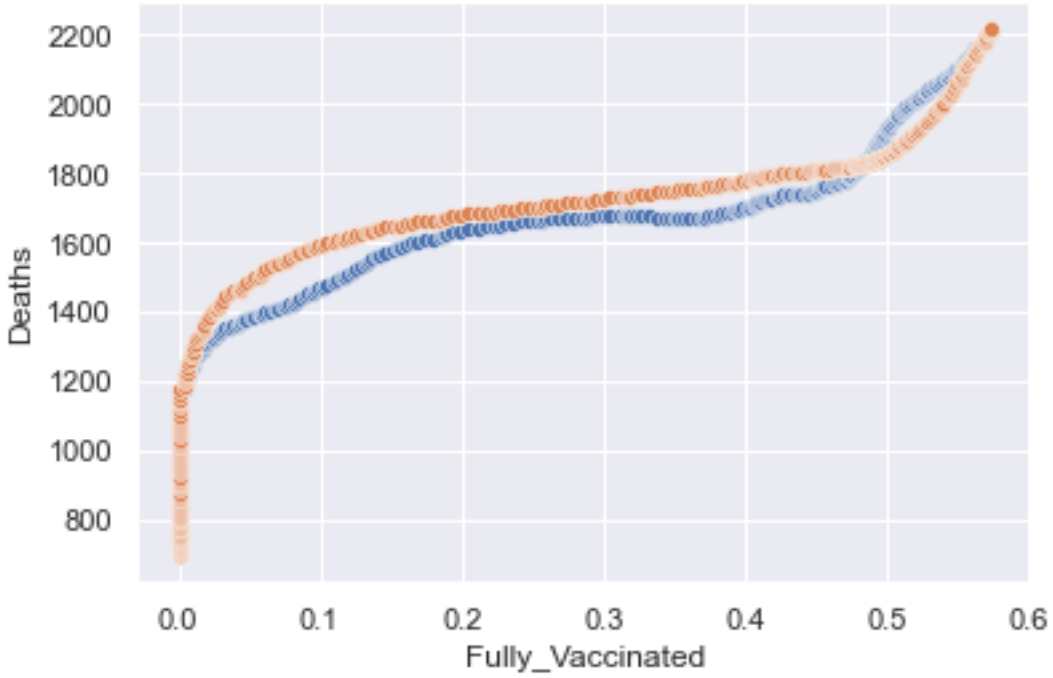


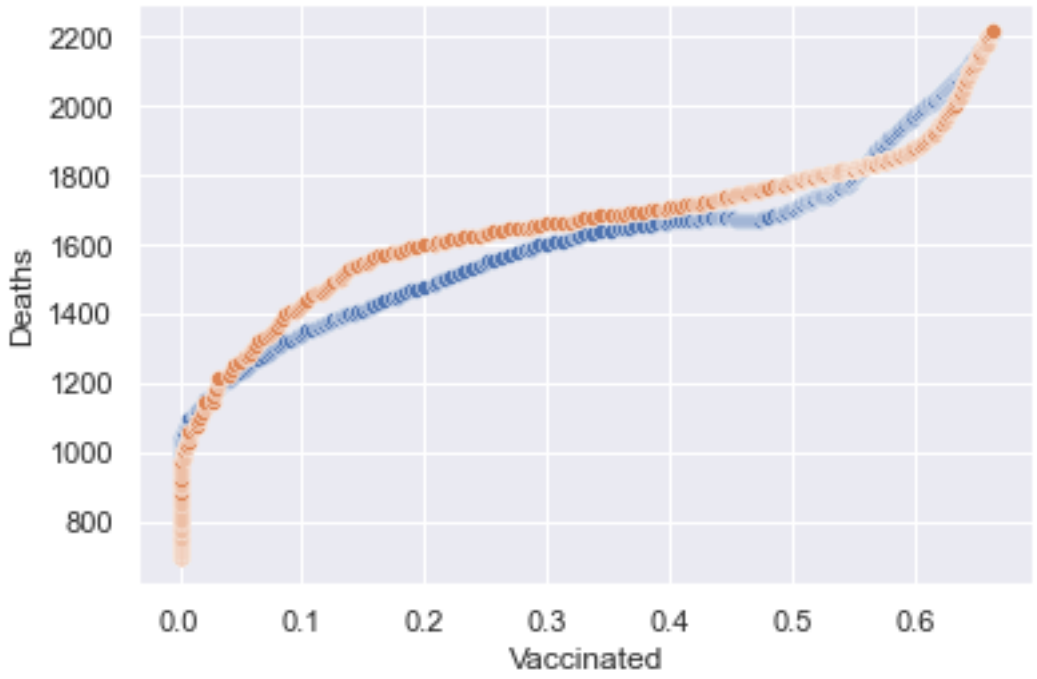

```
(108, 1) (360, 1)
r^2 Value: 0    0.478229
dtype: float64
```

```
array([[  0.74429546],
       [ 19.90024457],
       [ 36.3423769 ],
       [-36.3534448 ]])
```

```
United States
```


2200
2000
1800
1600
1400
1200
1000
800
Deaths
0
50
100
150
200
250
300
350
Days

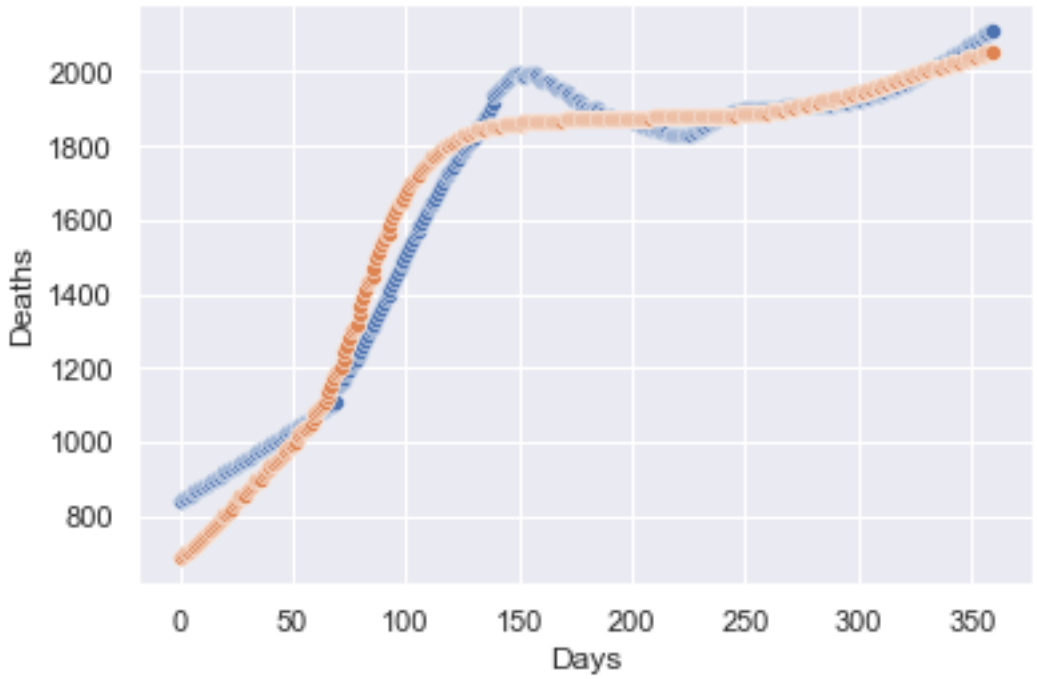

2200
2000
1800
1600
1400
1200
1000
800
Deaths
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Fully_Vaccinated

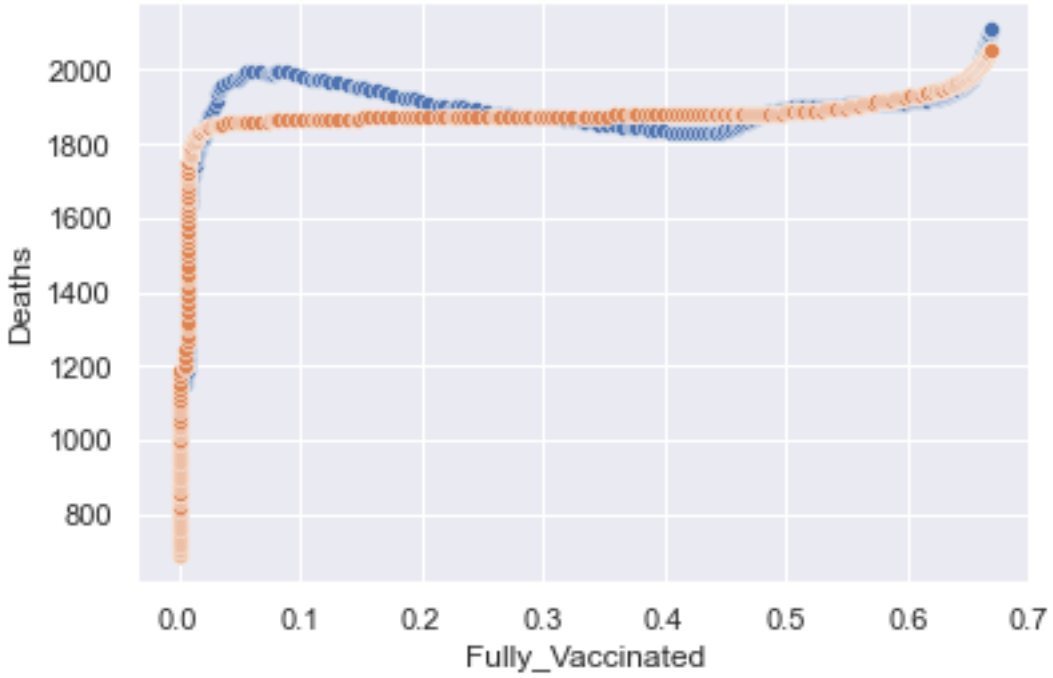


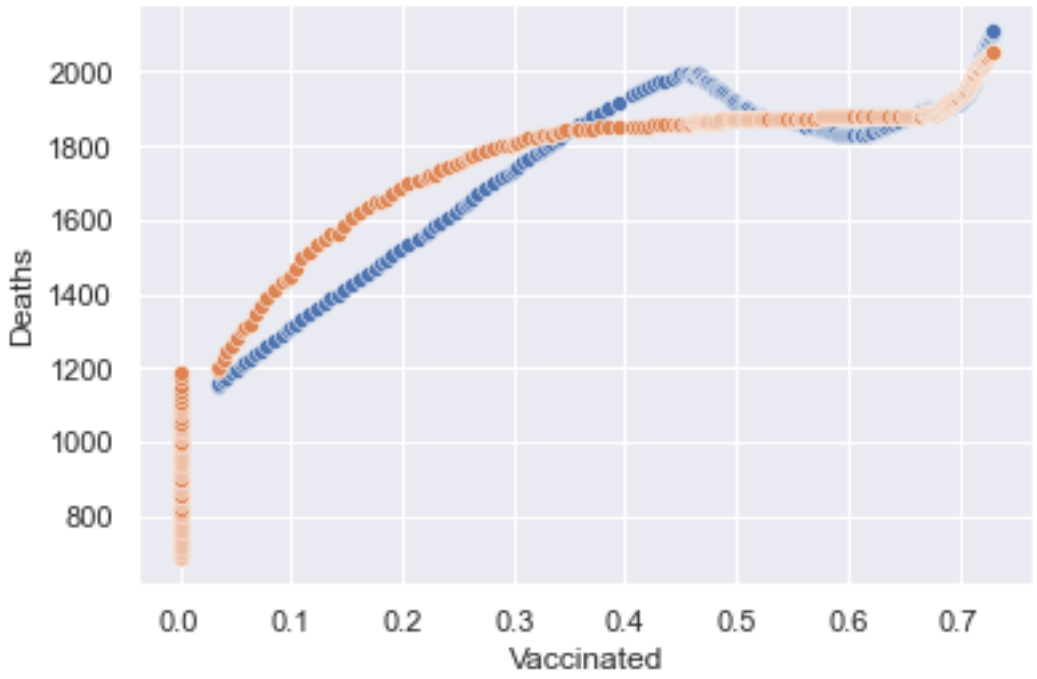

```
(108, 1) (360, 1)
r^2 Value: 0    0.960859
dtype: float64
```

```
array([[  850.26206404],
       [ 1372.38940904],
       [-1140.43154586],
       [ 1132.72331498]])
```

```
United Kingdom
```


Deaths
2000
1800
1600
1400
1200
1000
800
0
50
100
150
200
250
300
350
Days

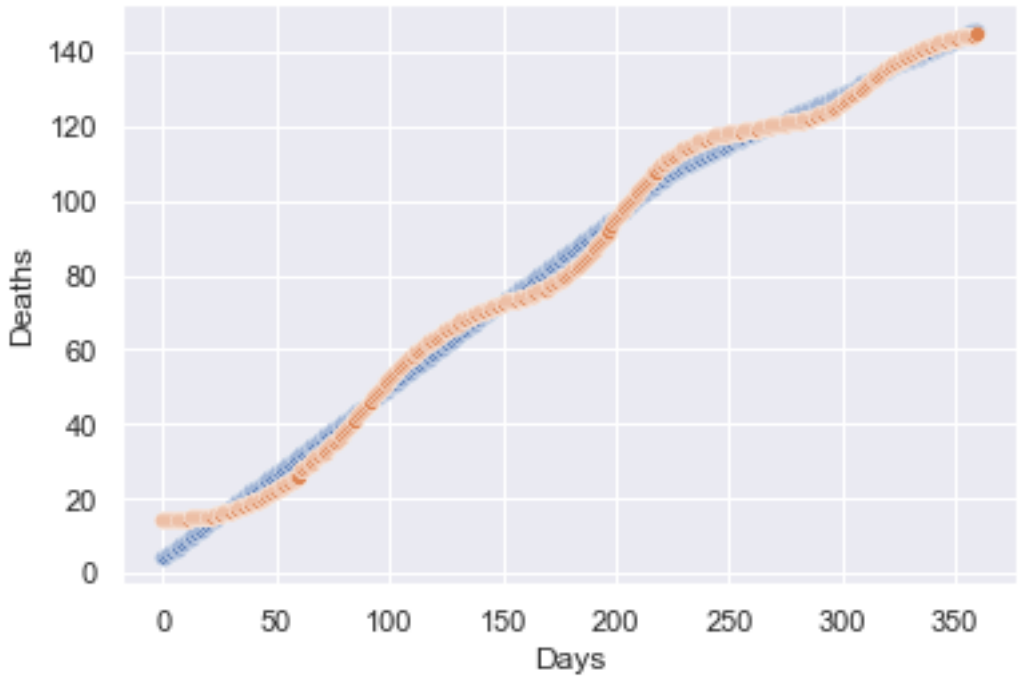

Deaths
2000
1800
1600
1400
1200
1000
800
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Fully_Vaccinated

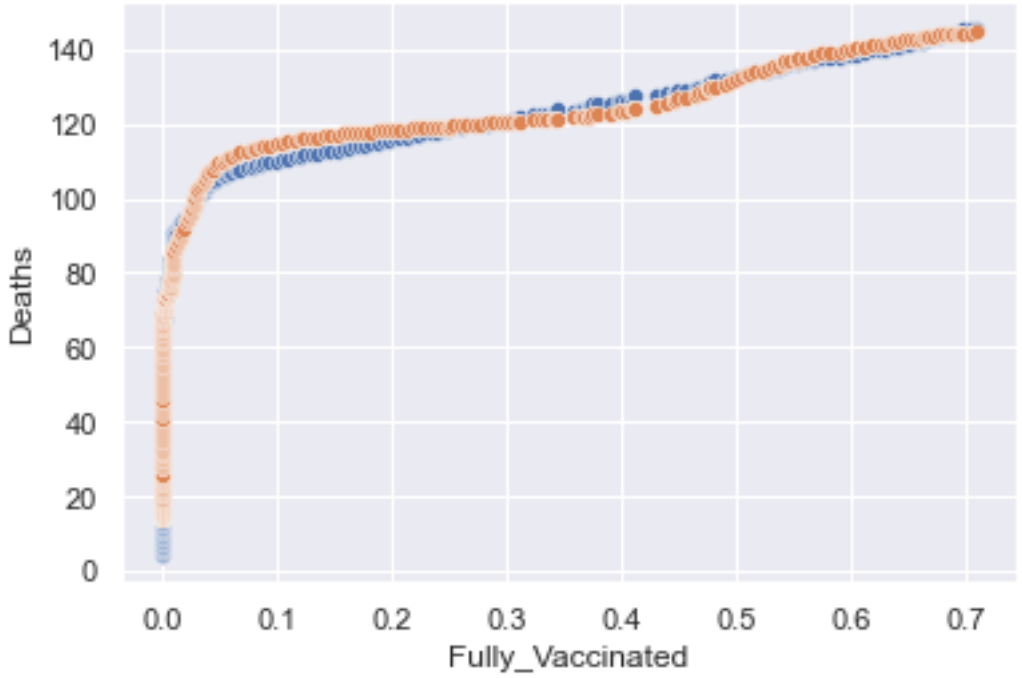


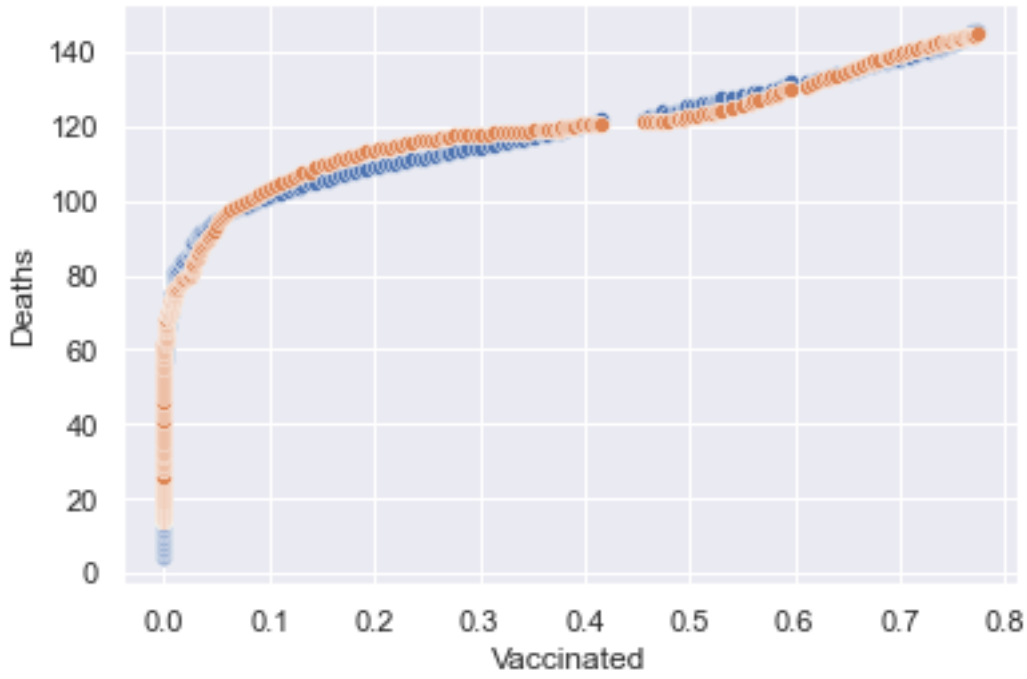

```
(108, 1) (360, 1)
r^2 Value: 0    0.964757
dtype: float64
```

```
array([[  839.34361714],
       [ 1394.53497998],
       [-1229.37388997],
       [ 1109.17159933]])
```

Japan


Deaths
140
120
100
80
60
40
20
0
0
50
100
150
200
250
300
350
Days

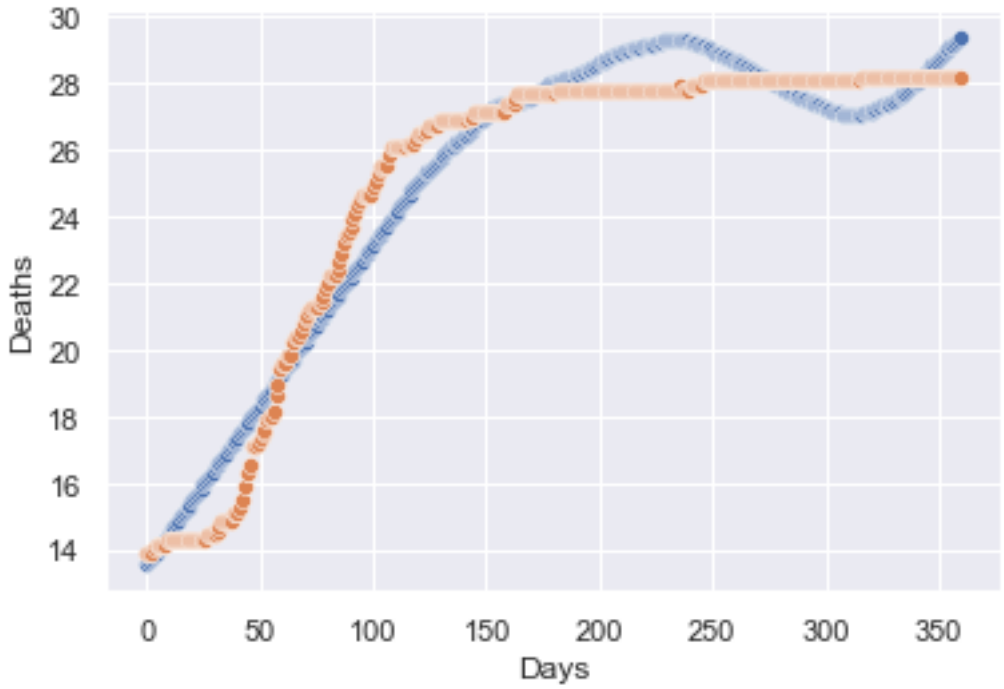

Deaths
140
120
100
80
60
40
20
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Fully_Vaccinated

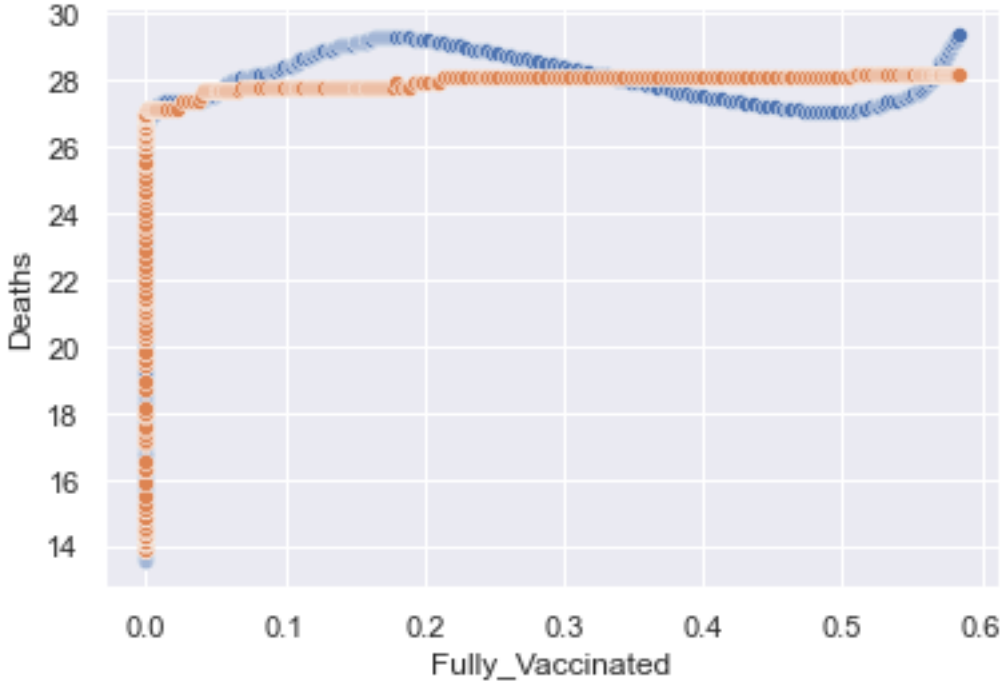


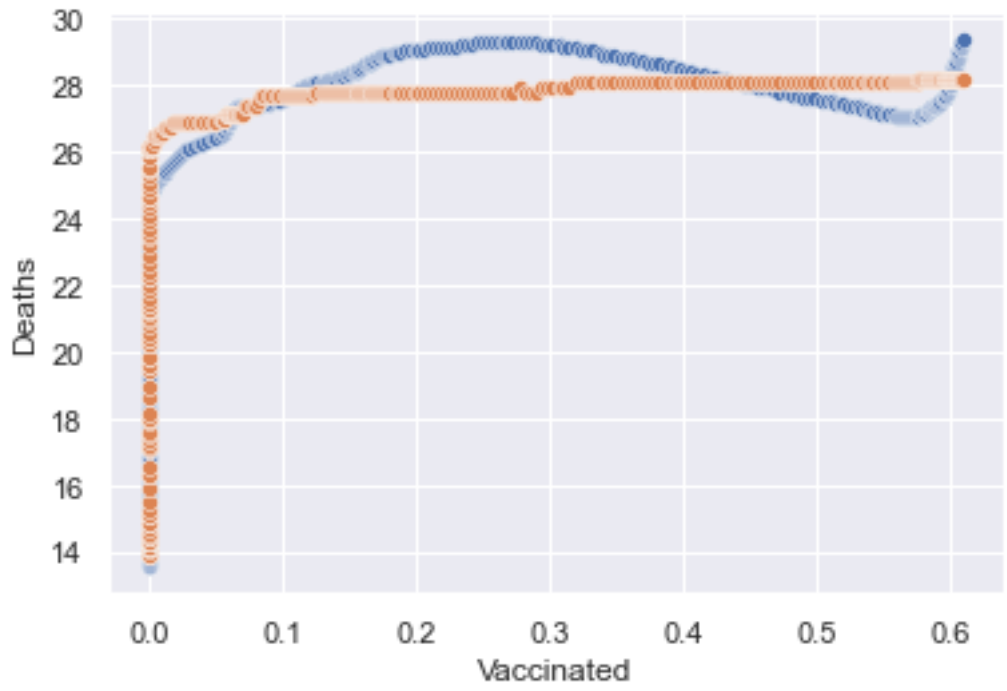

```
(108, 1) (360, 1)
r^2 Value: 0    0.993744
dtype: float64
```

```
array([[  4.06091656],
       [164.59206929],
       [-39.05668677],
       [ 16.22104757]])
```

Hong Kong


30
28
26
24
22
20
18
16
14
Deaths
0
50
100
150
200
250
300
350
Days

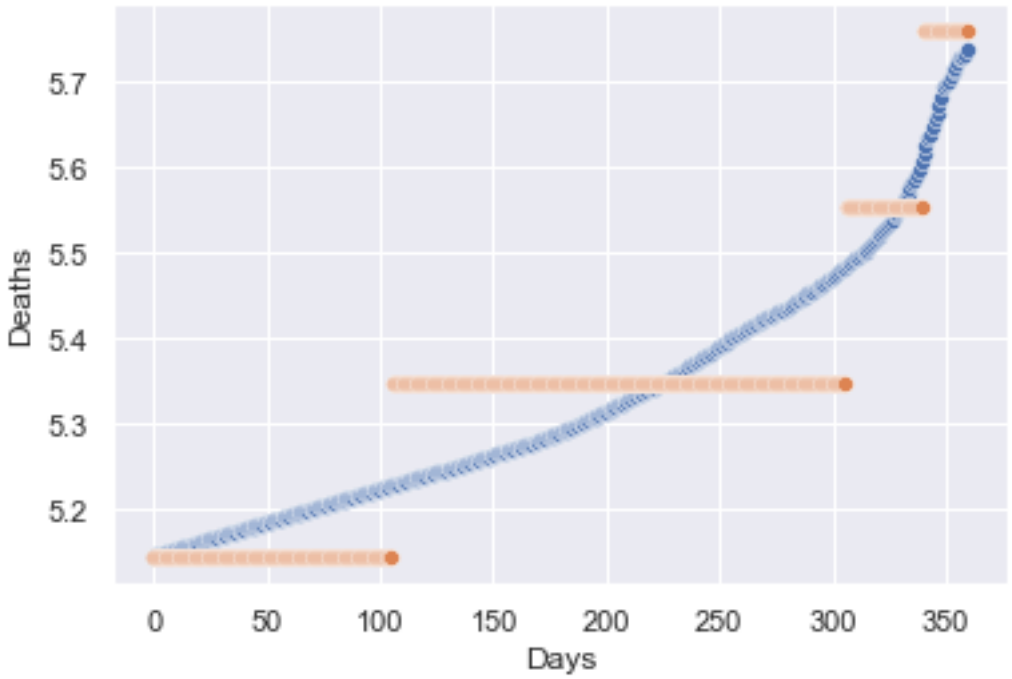

30
28
26
24
22
20
18
16
14
Deaths
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Fully_Vaccinated

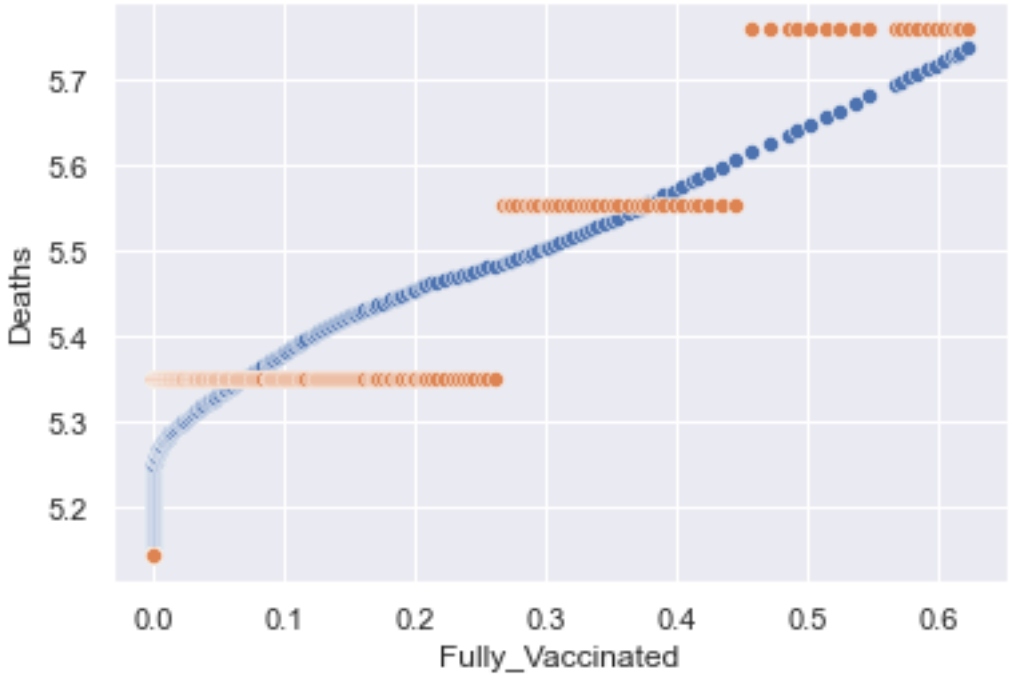


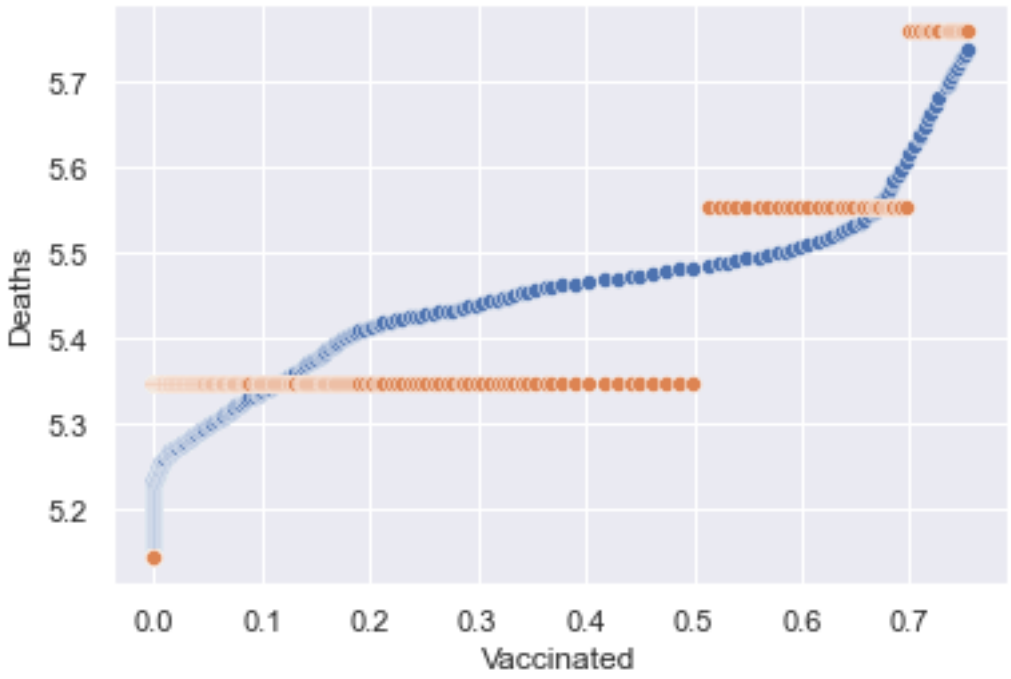

```
(108, 1) (360, 1)
r^2 Value: 0    0.953136
dtype: float64
```

```
array([[ 13.53748939],
       [ 34.51241214],
       [-10.80406392],
       [ -7.83545376]])
```

```
New Zealand
```


Deaths
5.7
5.6
5.5
5.4
5.3
5.2
0
50
100
150
200
250
300
350
Days

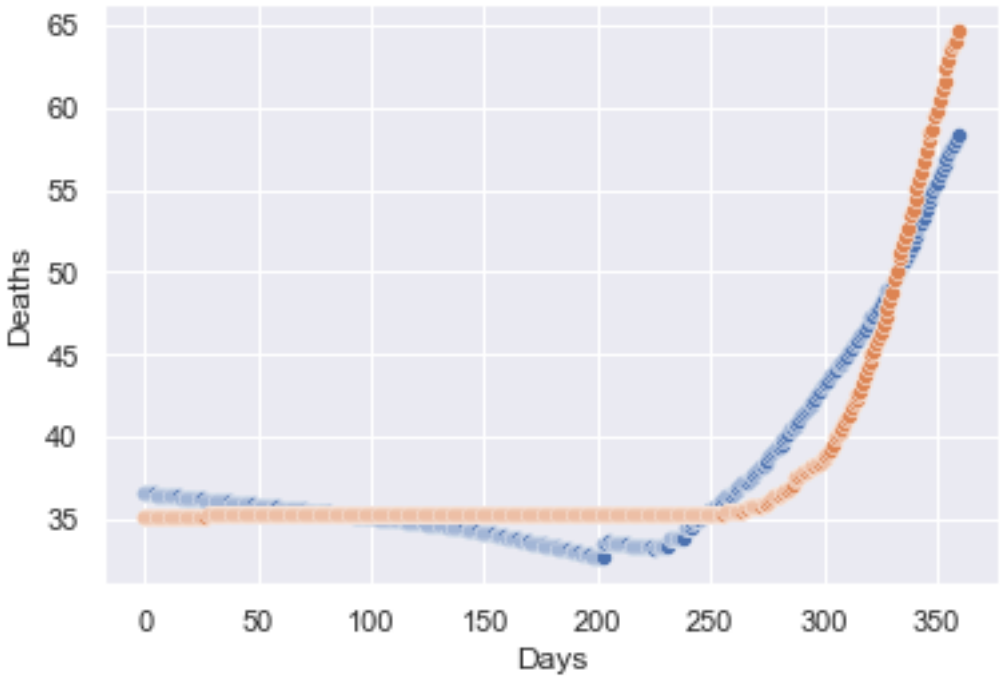

Deaths
5.7
5.6
5.5
5.4
5.3
5.2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Fully_Vaccinated

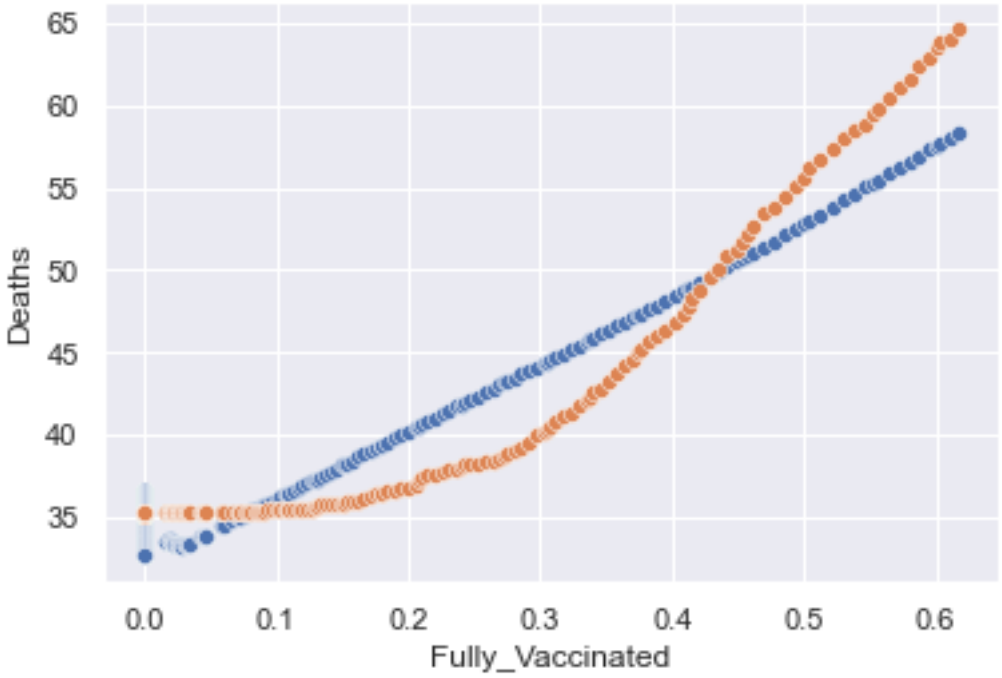


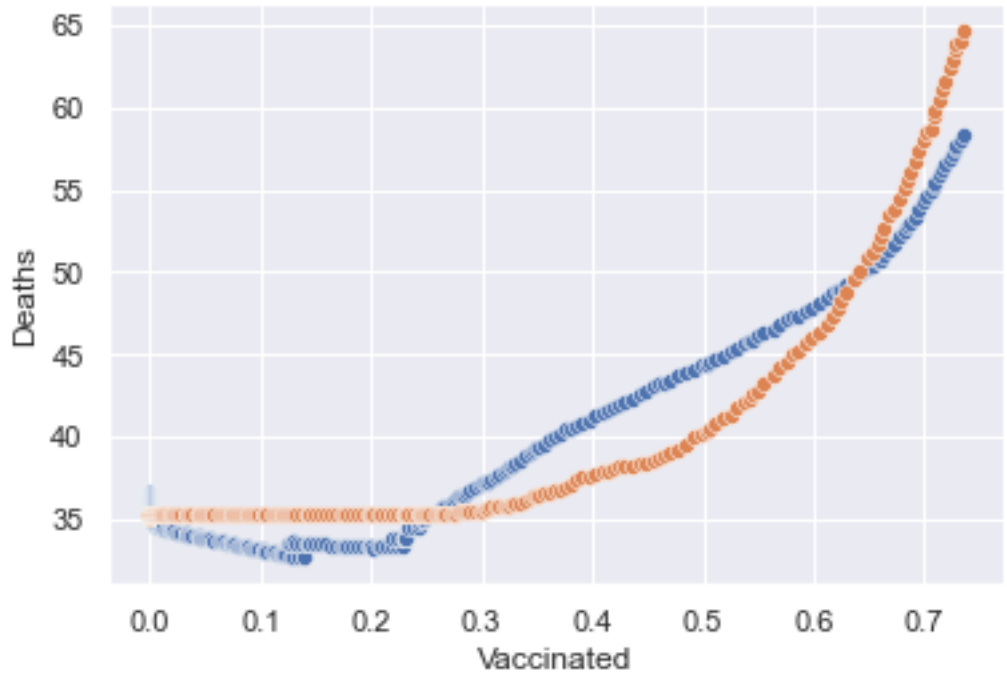

```
(108, 1) (360, 1)
r^2 Value: 0    0.823801
dtype: float64
```

```
array([[ 5.1453454 ],
       [ 0.28149822],
       [ 0.43939731],
       [-0.12924485]])
```

```
Australia
```


Deaths
65
60
55
50
45
40
35
0
50
100
150
200
250
300
350
Days

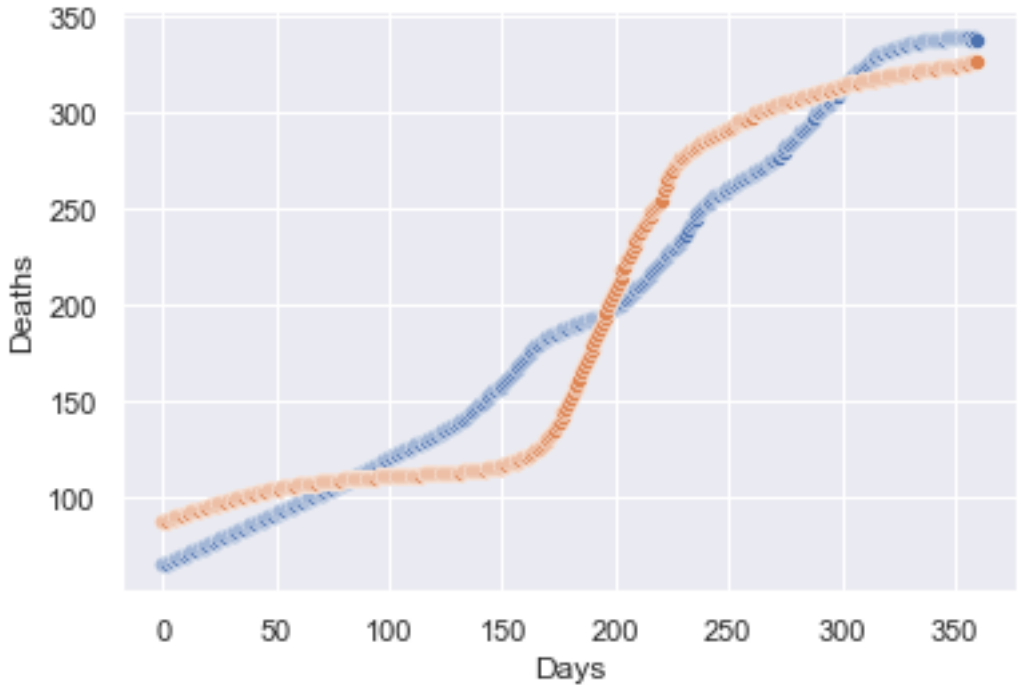

Deaths
65
60
55
50
45
40
35
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Fully_Vaccinated

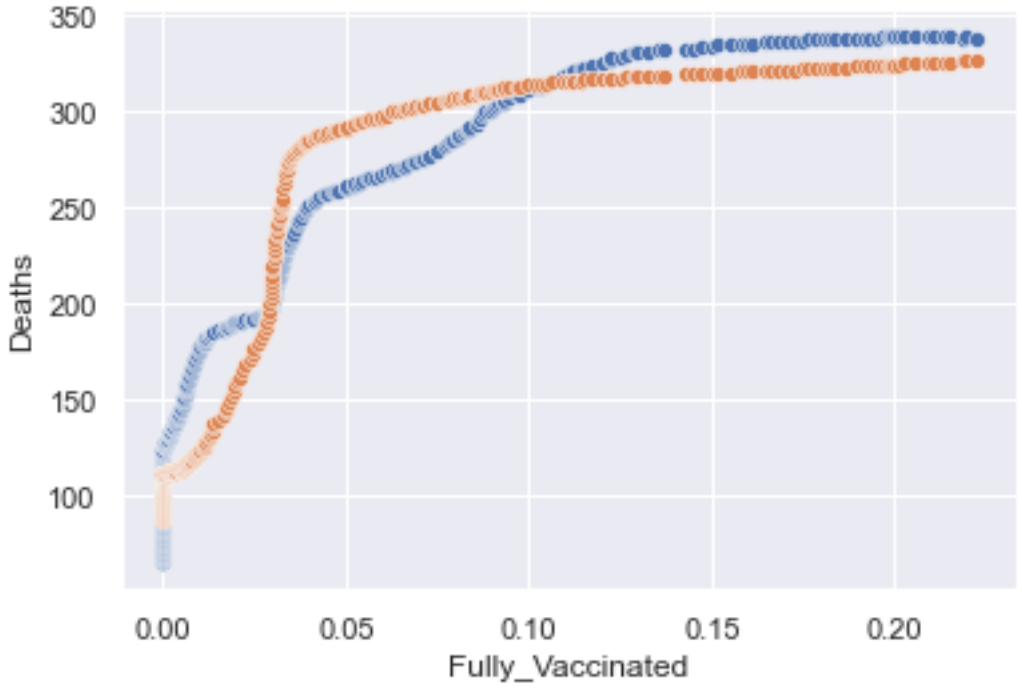


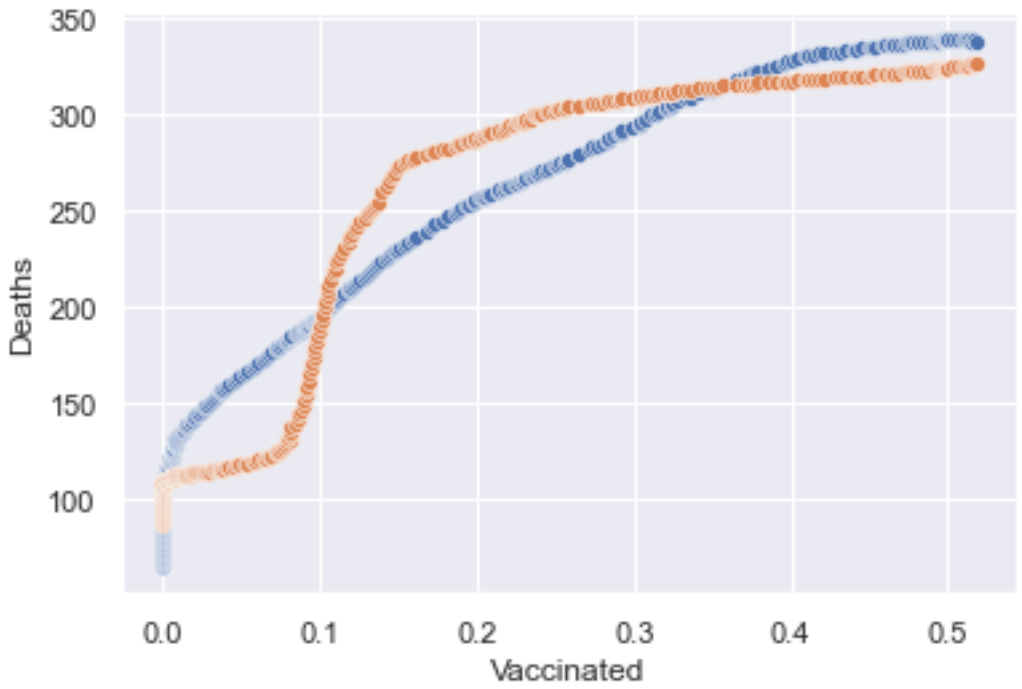

```
(108, 1) (360, 1)
r^2 Value: 0    0.895399
dtype: float64
```

```
array([[36.56268499],
       [-5.27877661],
       [32.16687798],
       [-5.17266243]])
```

```
India
```


350
300
250
200
150
100
Deaths
0
50
100
150
200
250
300
350
Days

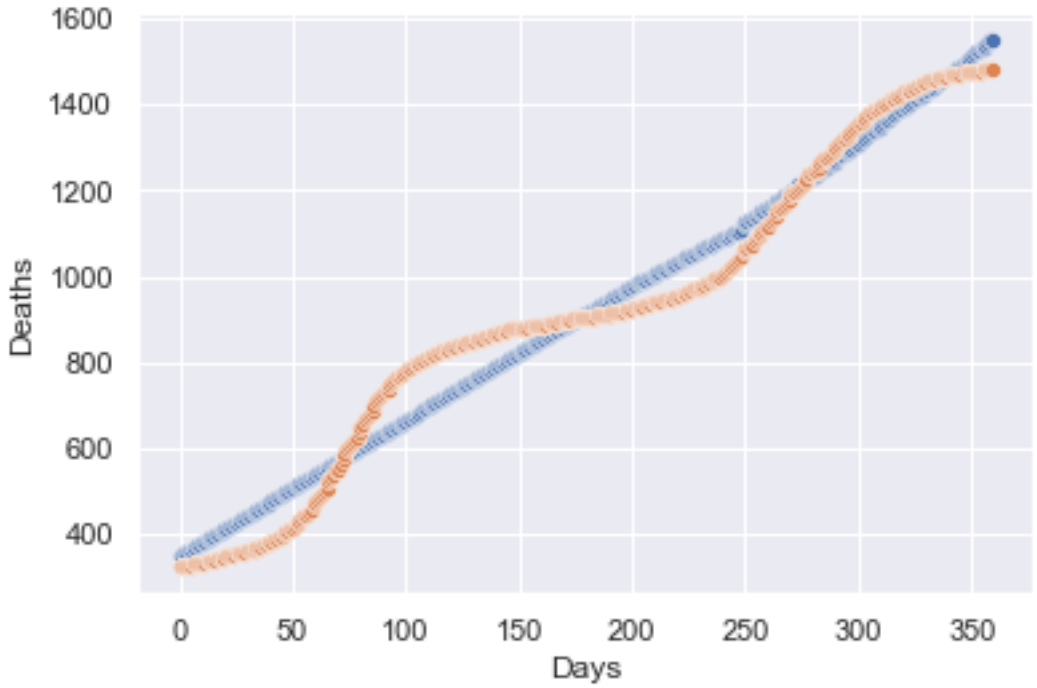

350
300
250
200
150
100
Deaths
0.00
0.05
0.10
0.15
0.20
Fully_Vaccinated

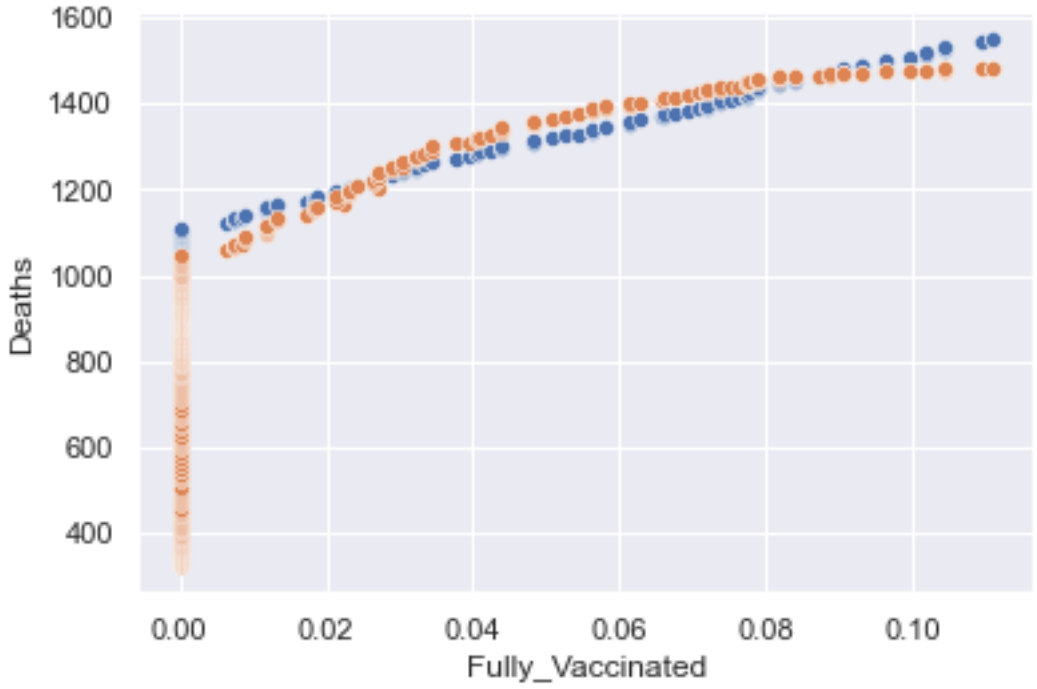


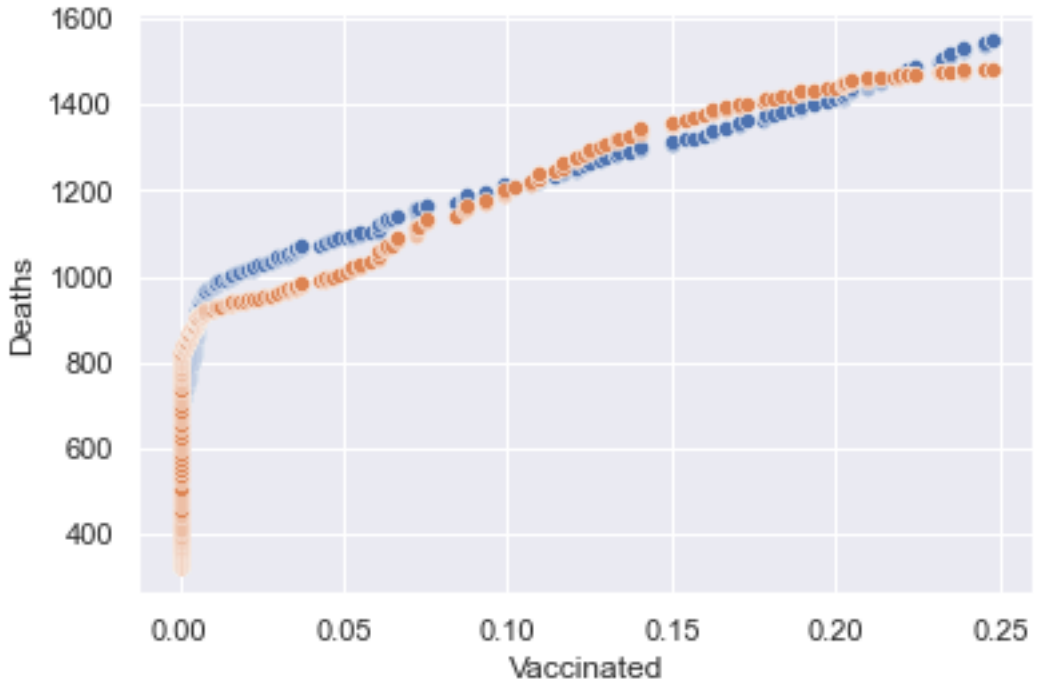

```
(108, 1) (360, 1)
r^2 Value: 0    0.929062
dtype: float64
```

```
array([[  65.38192614],
       [ 191.33290714],
       [-154.59932965],
       [ 236.02880154]])
```

South Africa


1600
1400
1200
1000
800
600
400
Deaths
0
50
100
150
200
250
300
350
Days

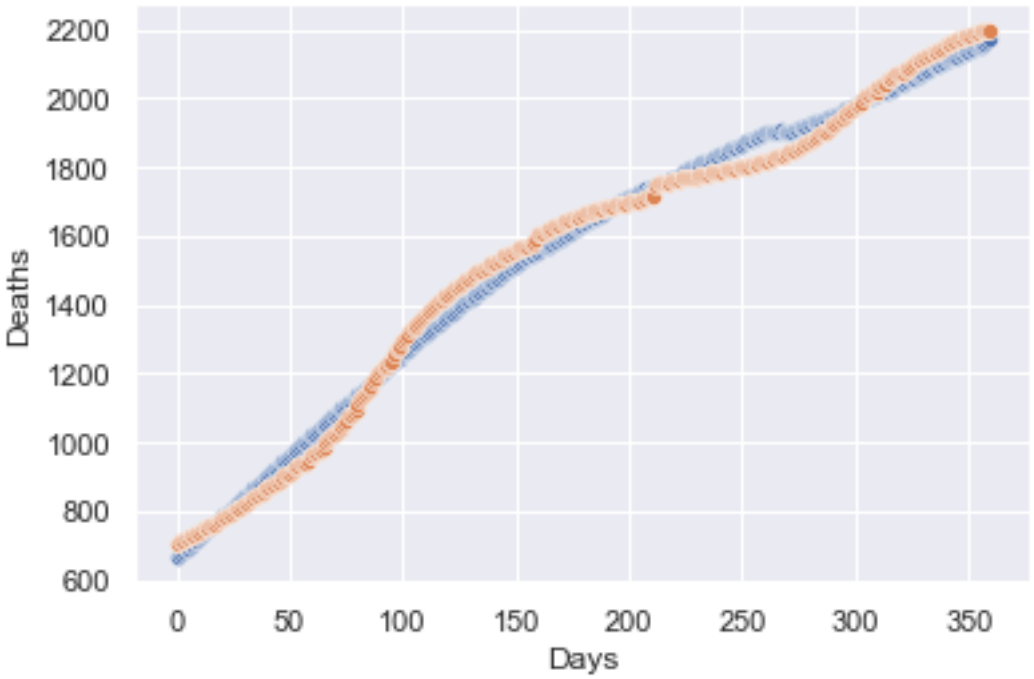

1600
1400
1200
1000
800
600
400
Deaths
0.00
0.02
0.04
0.06
0.08
0.10
Fully_Vaccinated

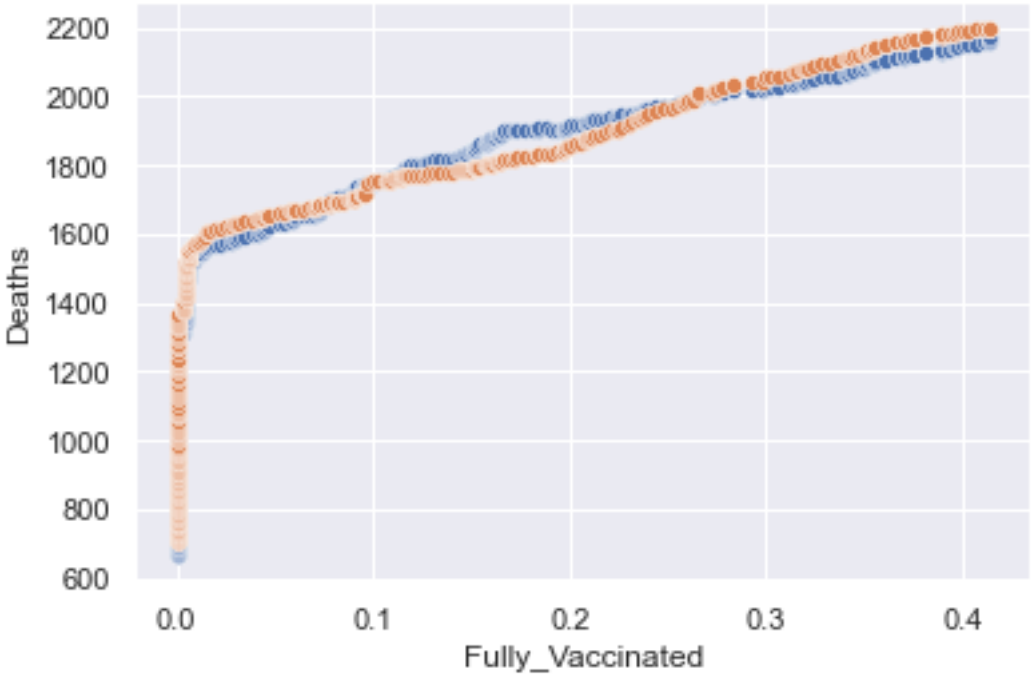


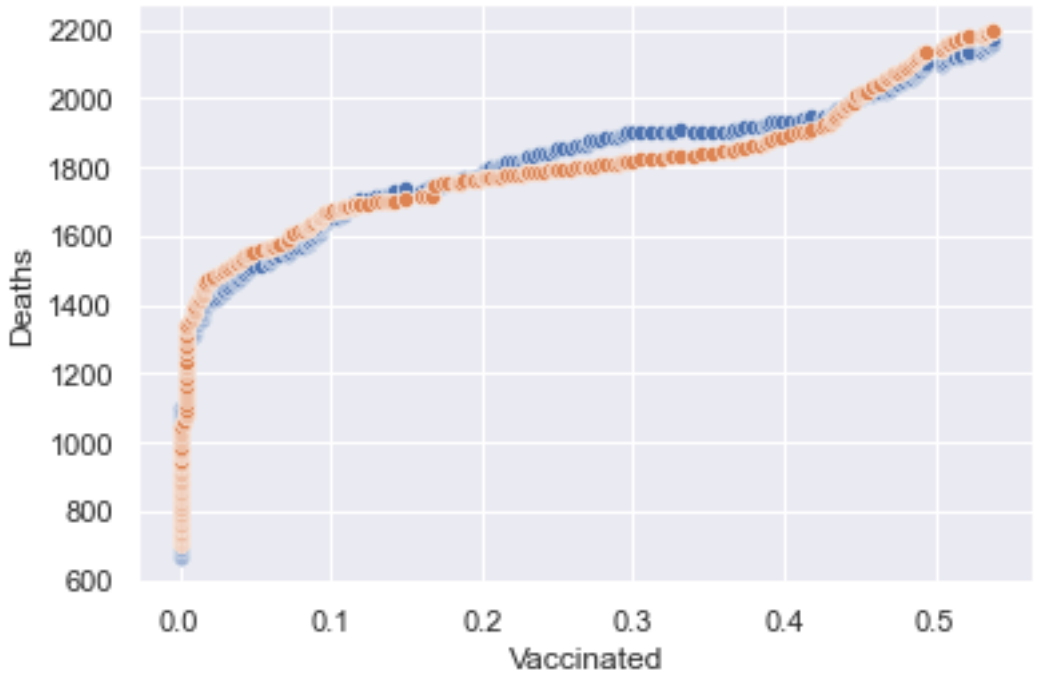

```
(108, 1) (360, 1)
r^2 Value: 0    0.965459
dtype: float64
```

```
array([[ 348.94859458],
       [1132.79149475],
       [ 162.39180287],
       [ -91.4141548 ]])
```

```
Mexico
```


Deaths
2200
2000
1800
1600
1400
1200
1000
800
600
0
50
100
150
200
250
300
350
Days

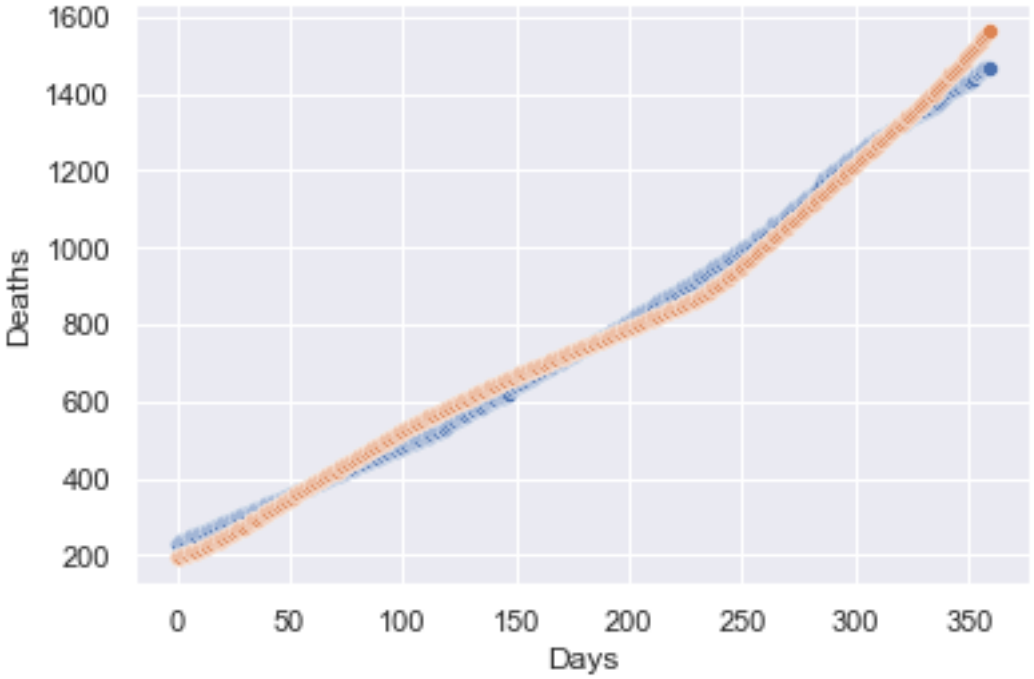

Deaths
2200
2000
1800
1600
1400
1200
1000
800
600
0.0
0.1
0.2
0.3
0.4
Fully_Vaccinated

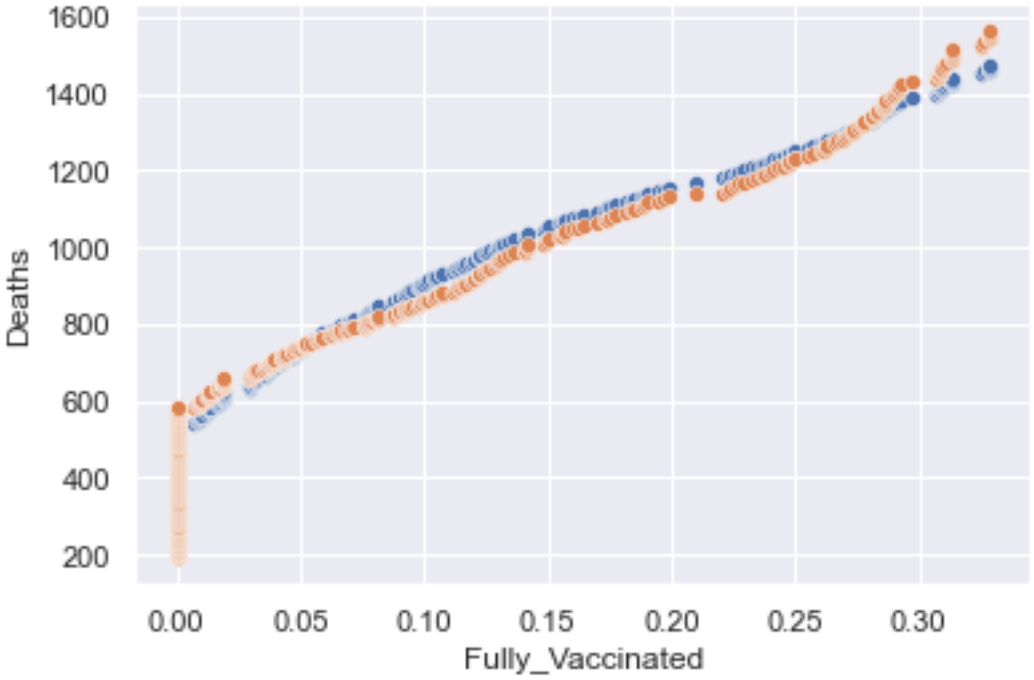


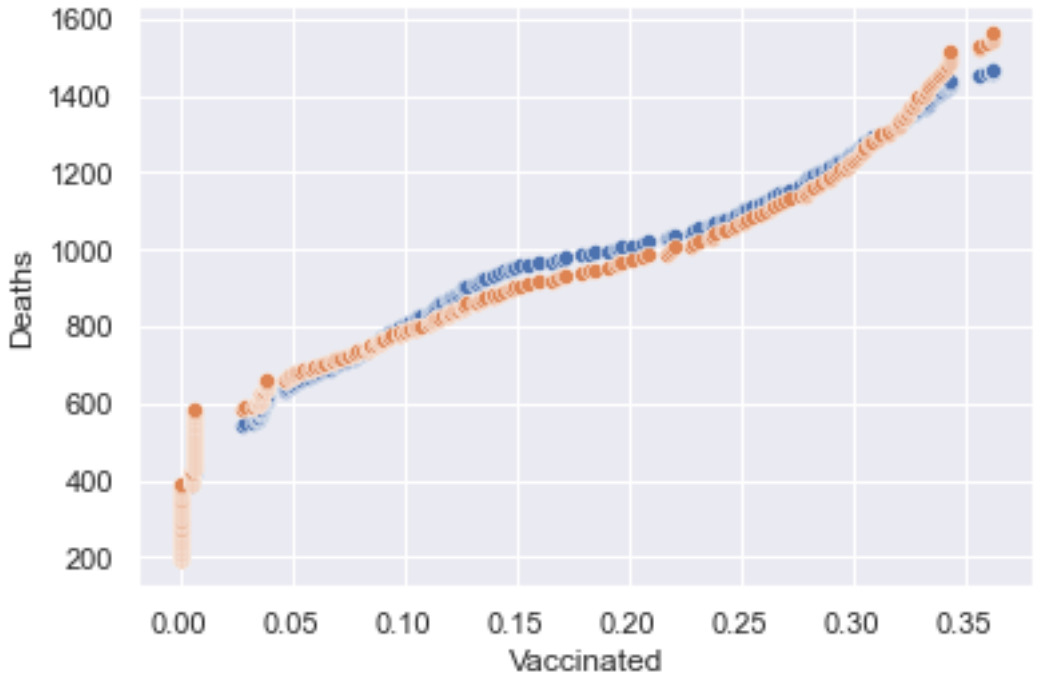

```
(108, 1) (360, 1)
r^2 Value: 0    0.990335
dtype: float64
```

```
array([[ 664.2802851 ],
       [2141.89364098],
       [-239.64939638],
       [-393.67718984]])
```

```
Russia
```


1600
1400
1200
1000
800
600
400
200
Deaths
0
50
100
150
200
250
300
350
Days

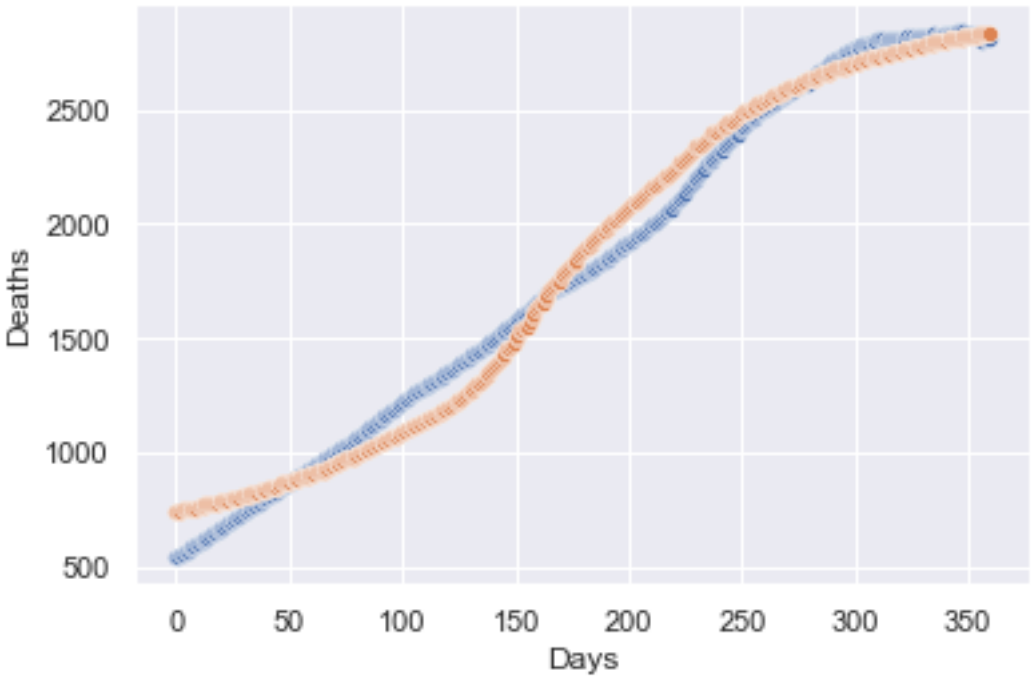

1600
1400
1200
1000
800
600
400
200
Deaths
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Fully_Vaccinated

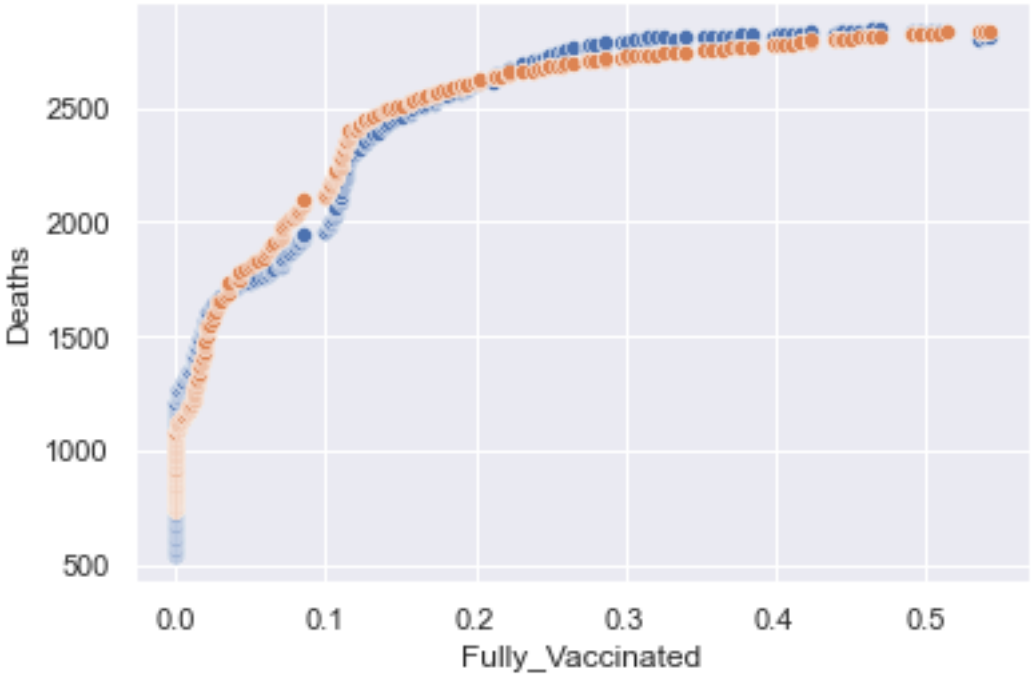


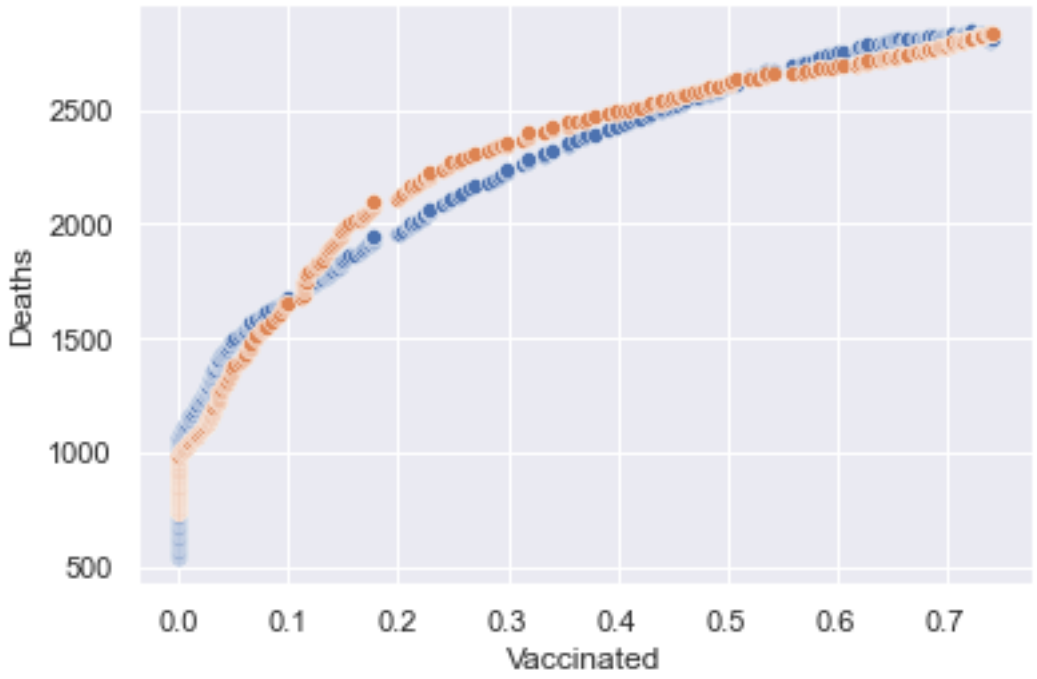

```
(108, 1) (360, 1)
r^2 Value: 0    0.991484
dtype: float64
```

```
array([[229.94177418],
       [907.76989598],
       [299.3575507 ],
       [ 33.63179726]])
```

```
Brazil
```


Deaths
2500
2000
1500
1000
500
0
50
100
150
200
250
300
350
Days

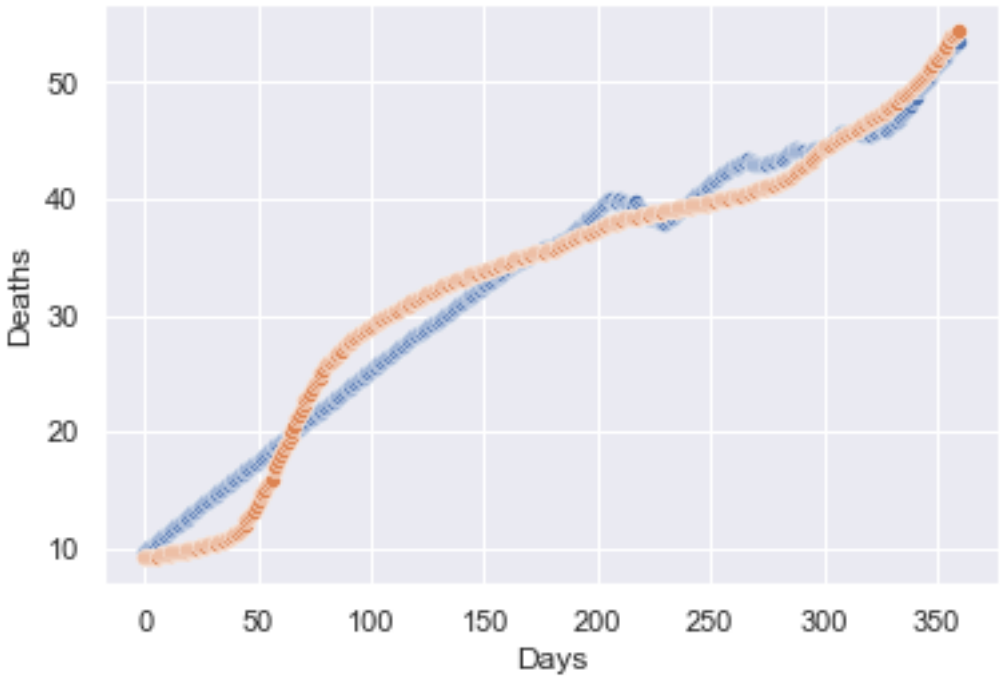

Deaths
2500
2000
1500
1000
500
0.0
0.1
0.2
0.3
0.4
0.5
Fully_Vaccinated

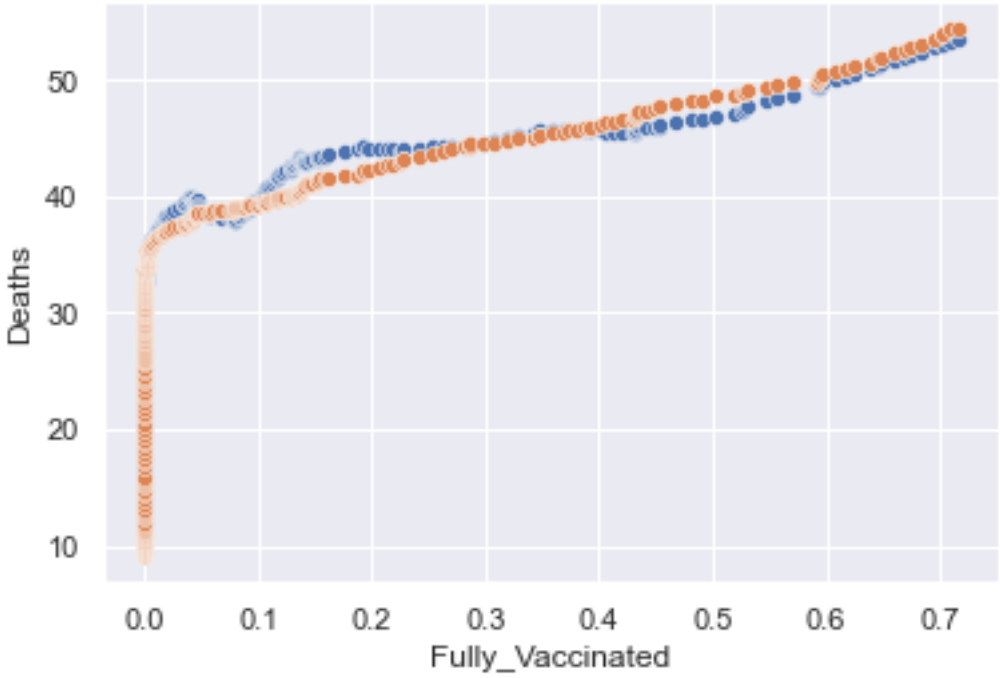


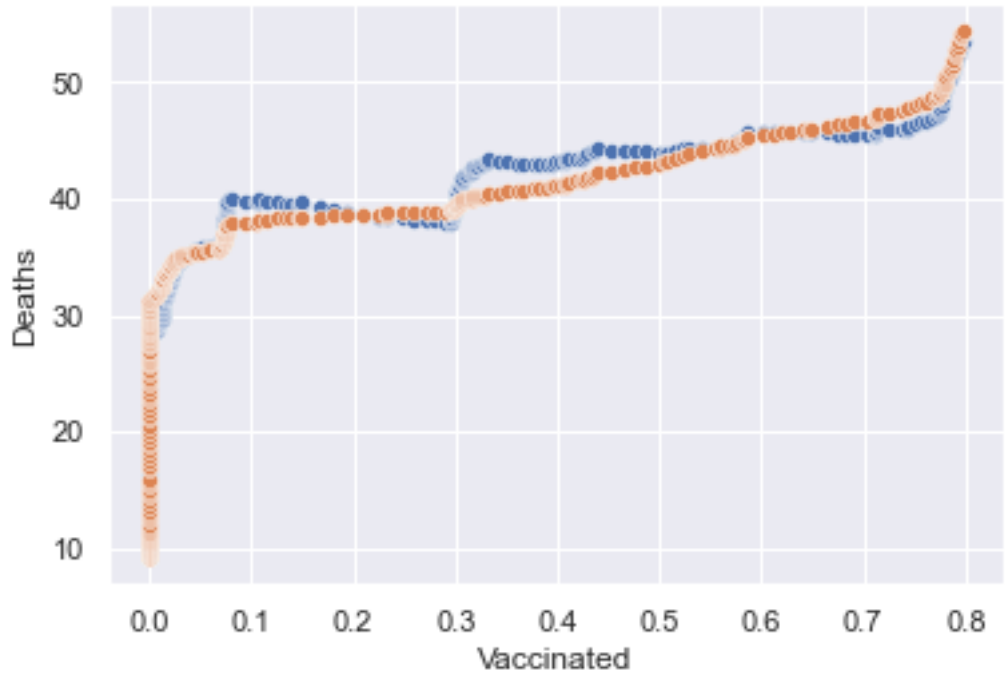

```
(108, 1) (360, 1)
r^2 Value: 0    0.982859
dtype: float64
```

```
array([[  541.35331108],
       [ 2364.52429407],
       [-1018.11193802],
       [  925.33235854]])
```

```
South Korea
```


Deaths
50
40
30
20
10
0
50
100
150
200
250
300
350
Days

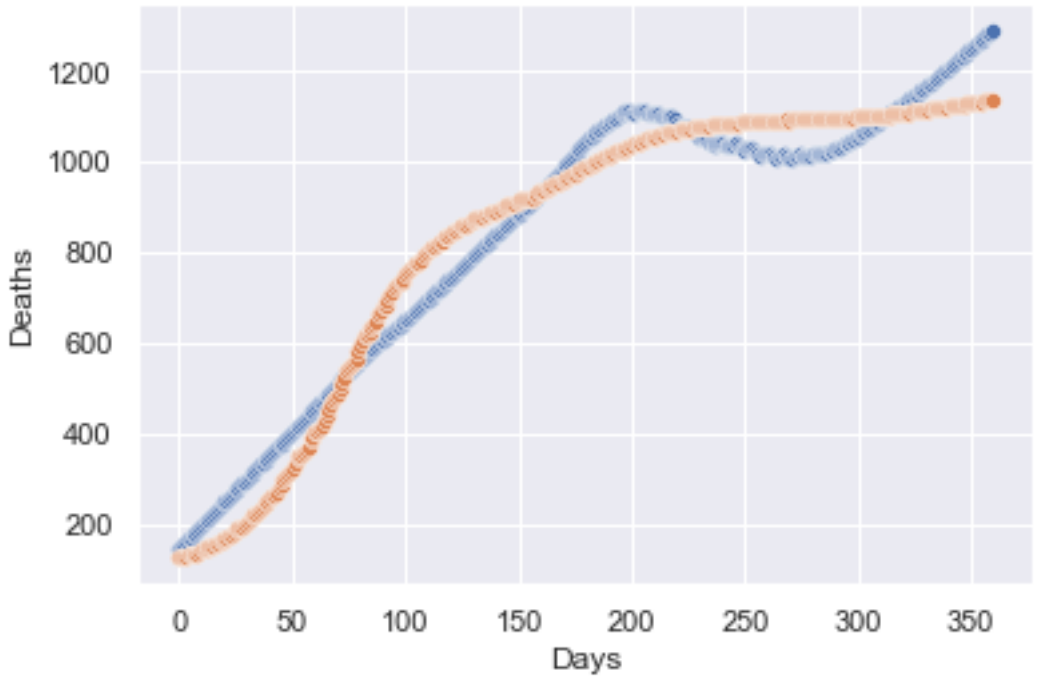

Deaths
50
40
30
20
10
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Fully_Vaccinated

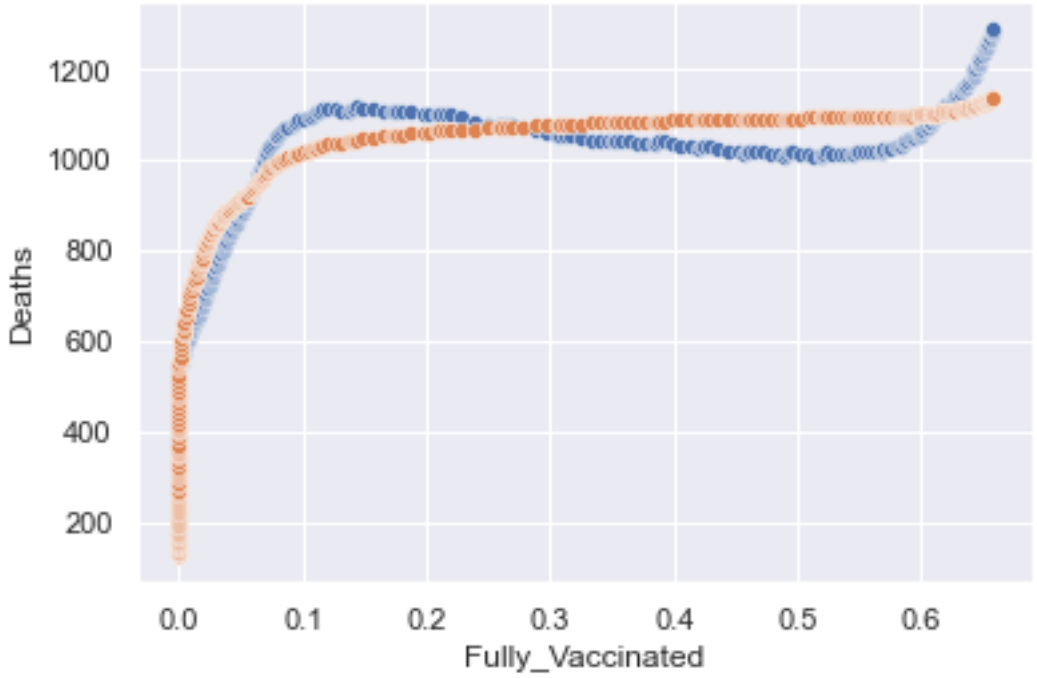


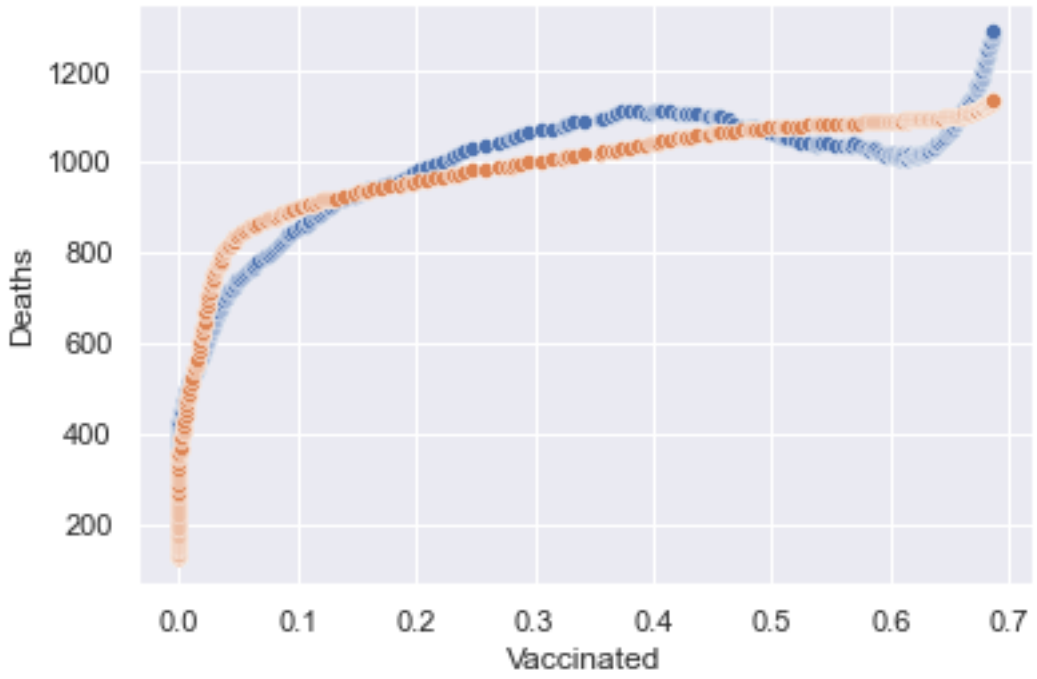

```
(108, 1) (360, 1)
r^2 Value: 0    0.965947
dtype: float64
```

```
array([[  9.75001635],
       [ 55.75971562],
       [ 12.12702956],
       [-24.00065835]])
```

```
Germany
```


Deaths
1200
1000
800
600
400
200
0
50
100
150
200
250
300
350
Days

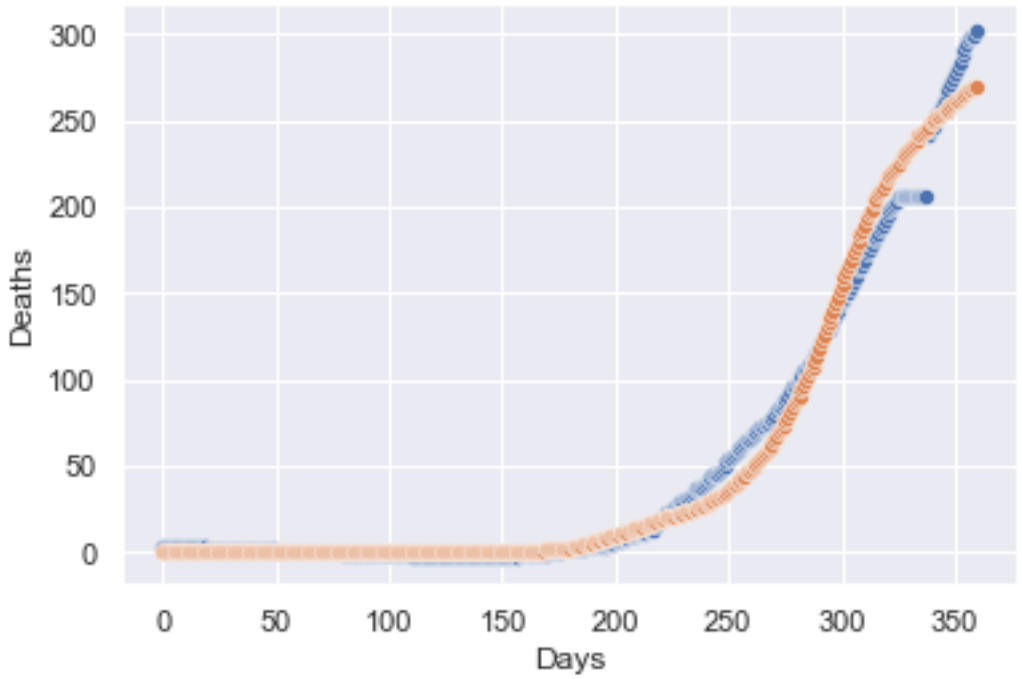

Deaths
1200
1000
800
600
400
200
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Fully_Vaccinated

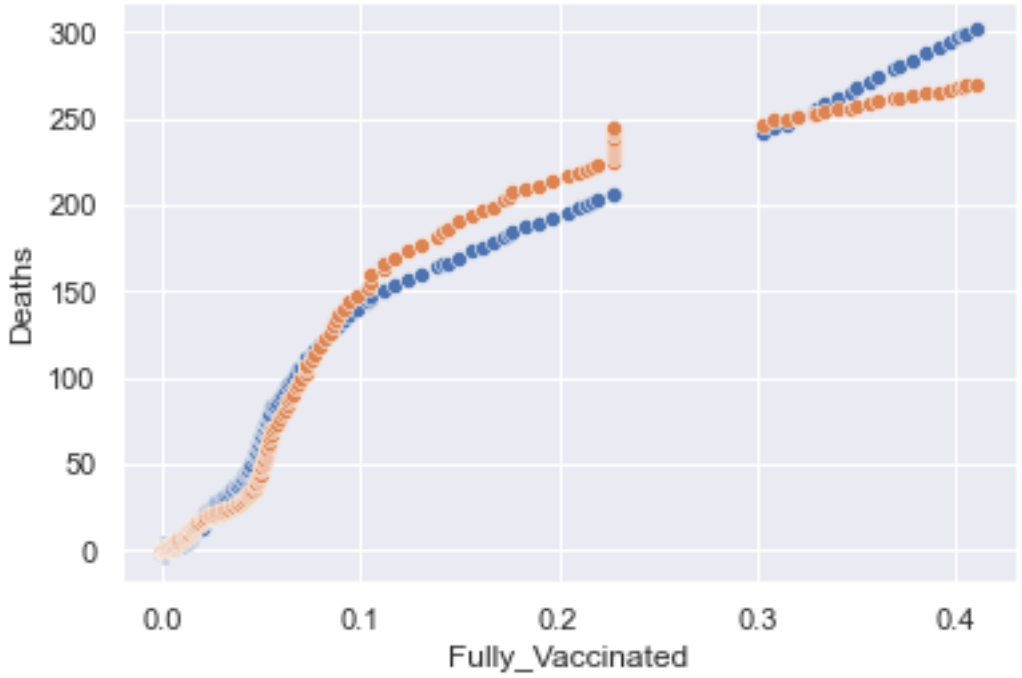


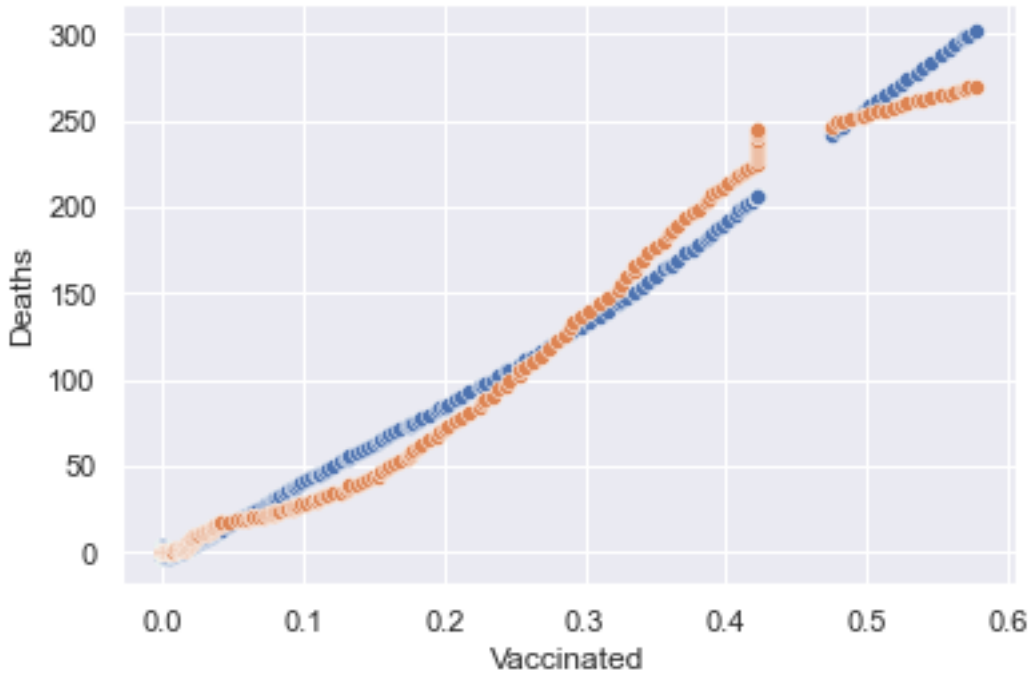

```
(108, 1) (360, 1)
r^2 Value: 0    0.955067
dtype: float64
```

```
array([[ 146.097636  ],
       [1852.3769054 ],
       [-901.86956007],
       [ 194.5415857 ]])
```

```
Thailand
```


Deaths
300
250
200
150
100
50
0
0
50
100
150
200
250
300
350
Days

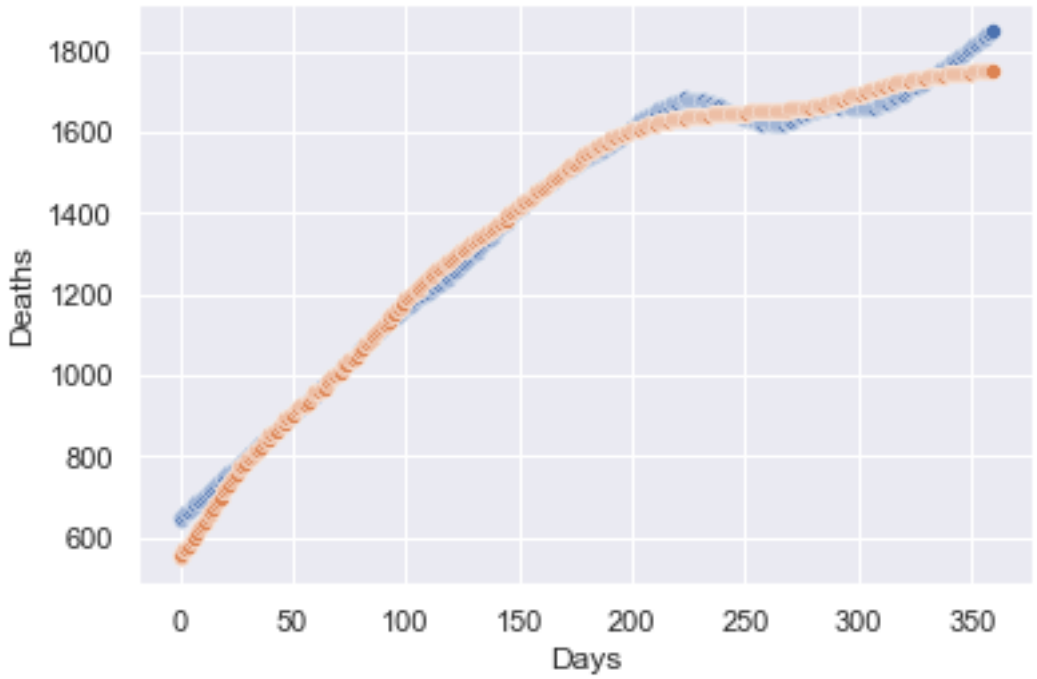

Deaths
300
250
200
150
100
50
0
0.0
0.1
0.2
0.3
0.4
Fully_Vaccinated

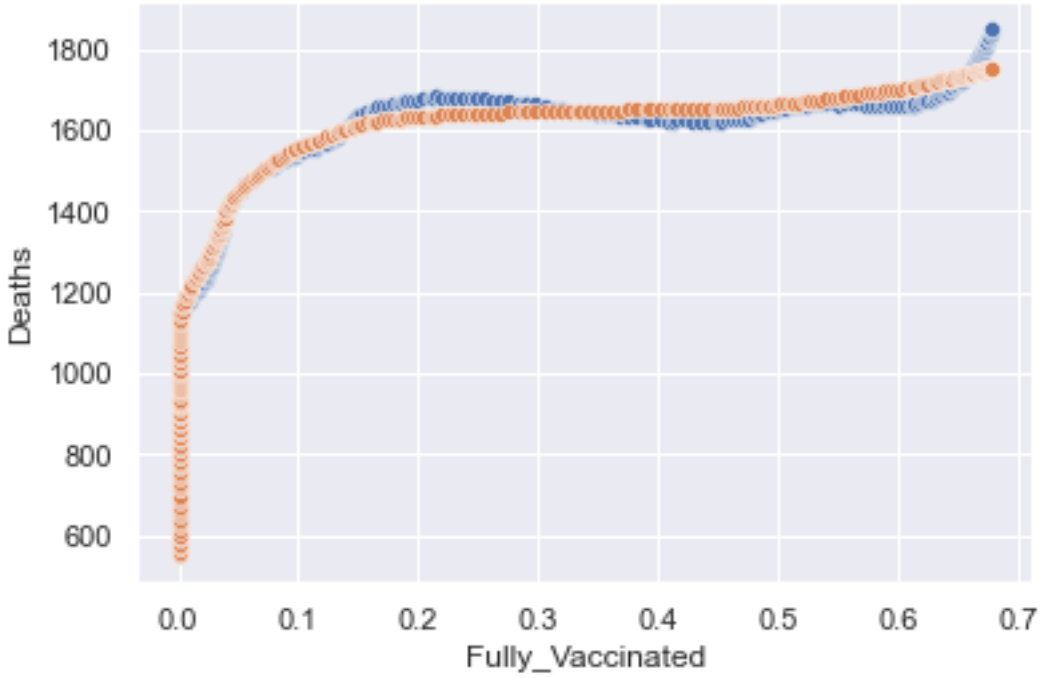


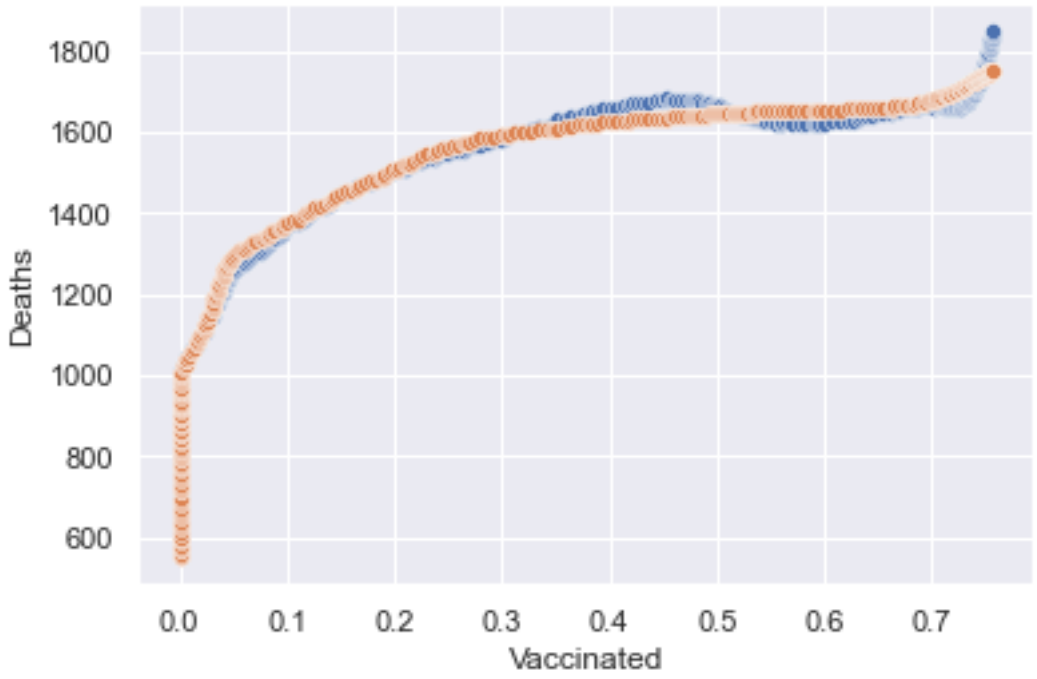

```
(108, 1) (360, 1)
r^2 Value: 0    0.983619
dtype: float64
```

```
array([[  4.04096889],
       [-18.62628353],
       [ 71.26654547],
       [245.5803619 ]])
```

```
France
```


Deaths
1800
1600
1400
1200
1000
800
600
0
50
100
150
200
250
300
350
Days

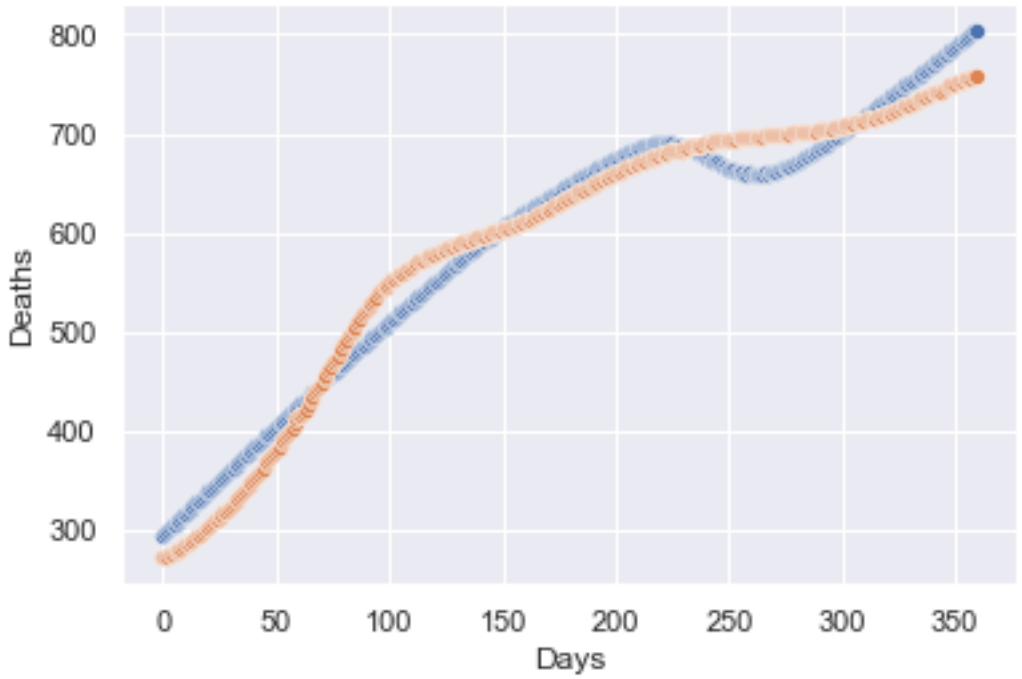

Deaths
1800
1600
1400
1200
1000
800
600
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Fully_Vaccinated

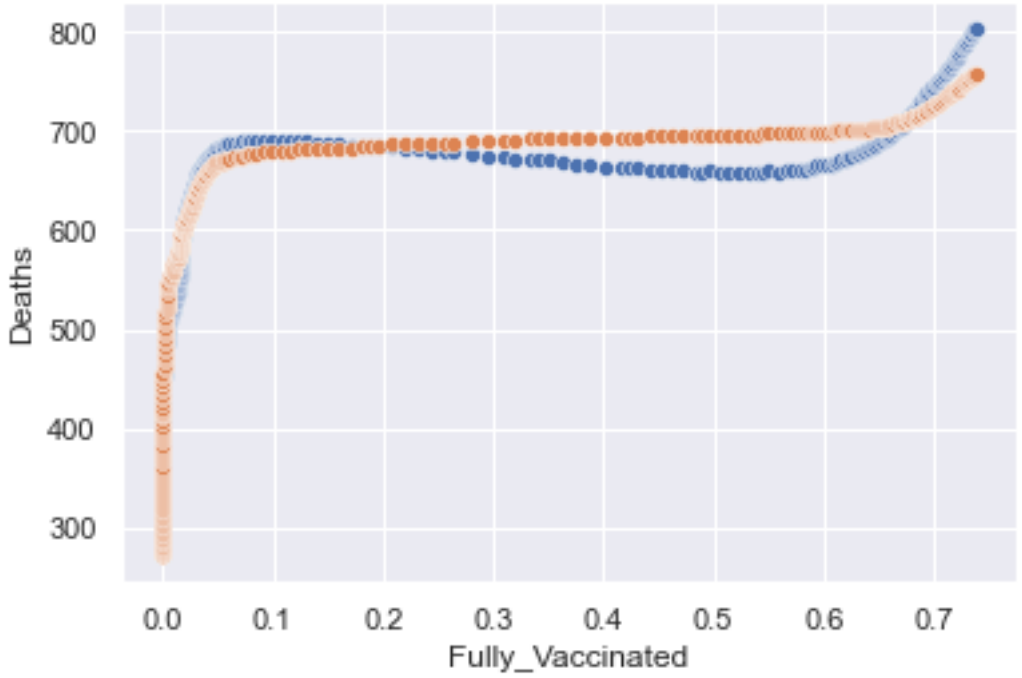


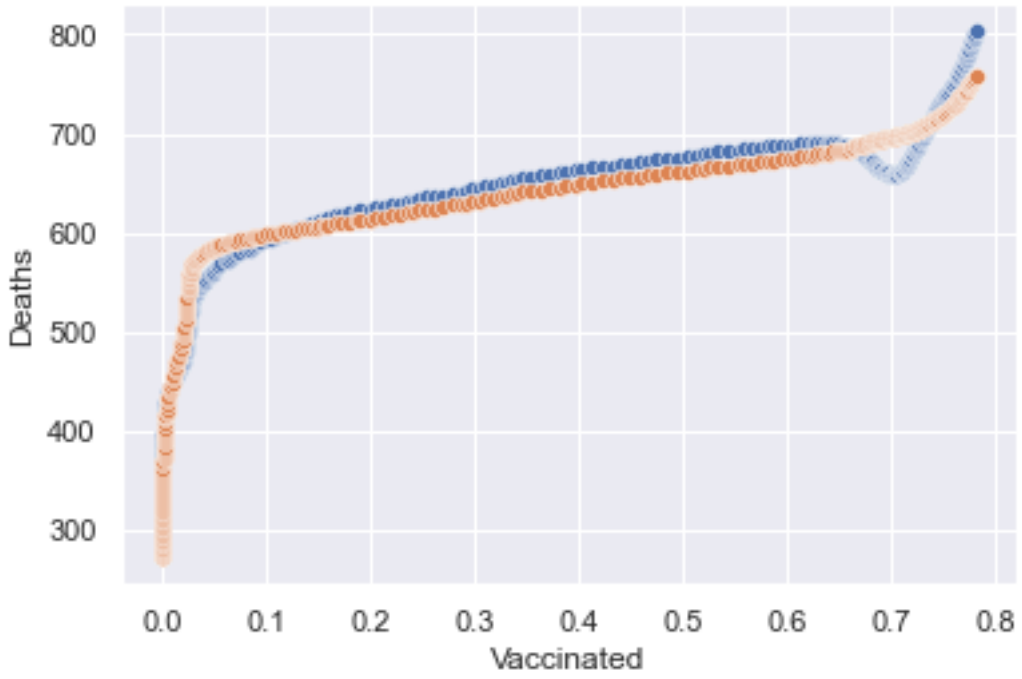

```
(108, 1) (360, 1)
r^2 Value: 0    0.992108
dtype: float64
```

```
array([[ 645.95430131],
       [1853.43645929],
       [-947.56145957],
       [ 301.57588547]])
```

```
Canada
```


800
700
600
500
400
300
Deaths
0
50
100
150
200
250
300
350
Days

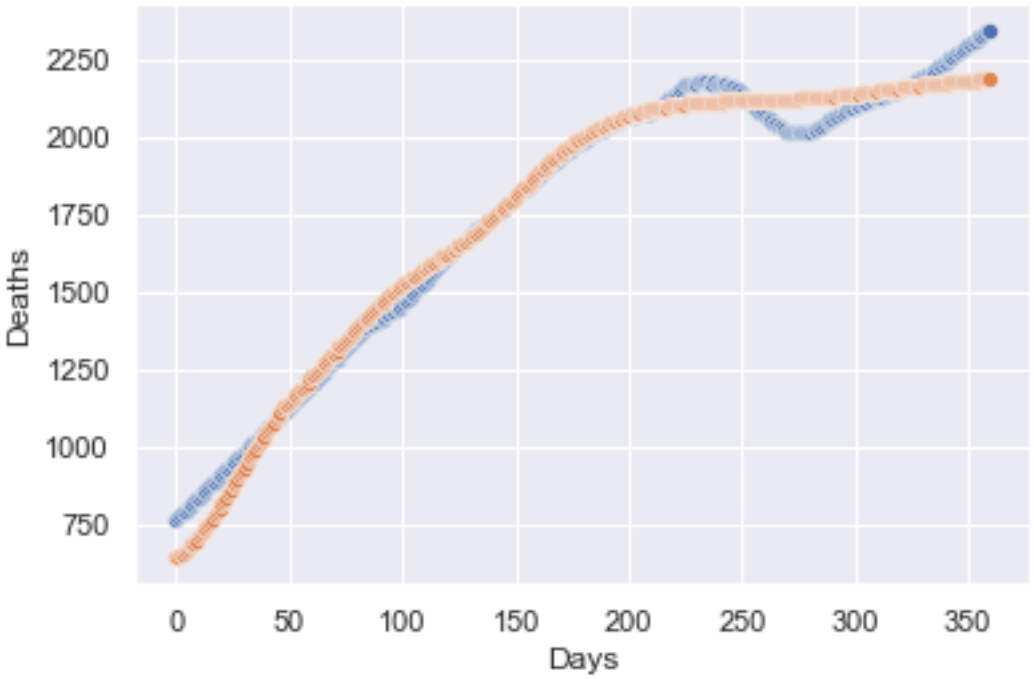

800
700
600
500
400
300
Deaths
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Fully_Vaccinated

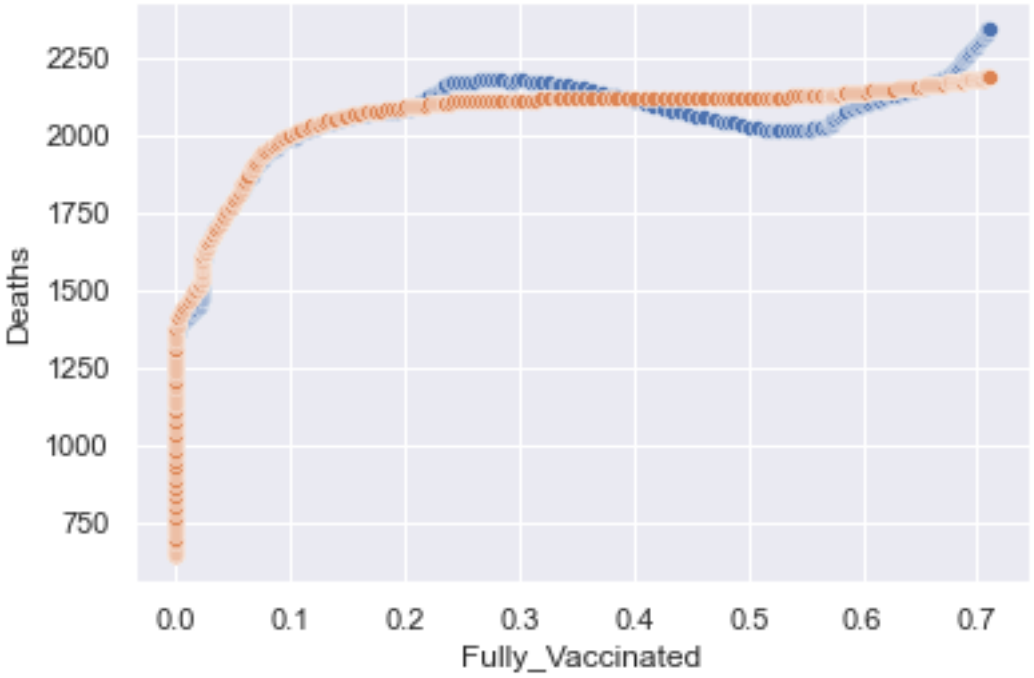


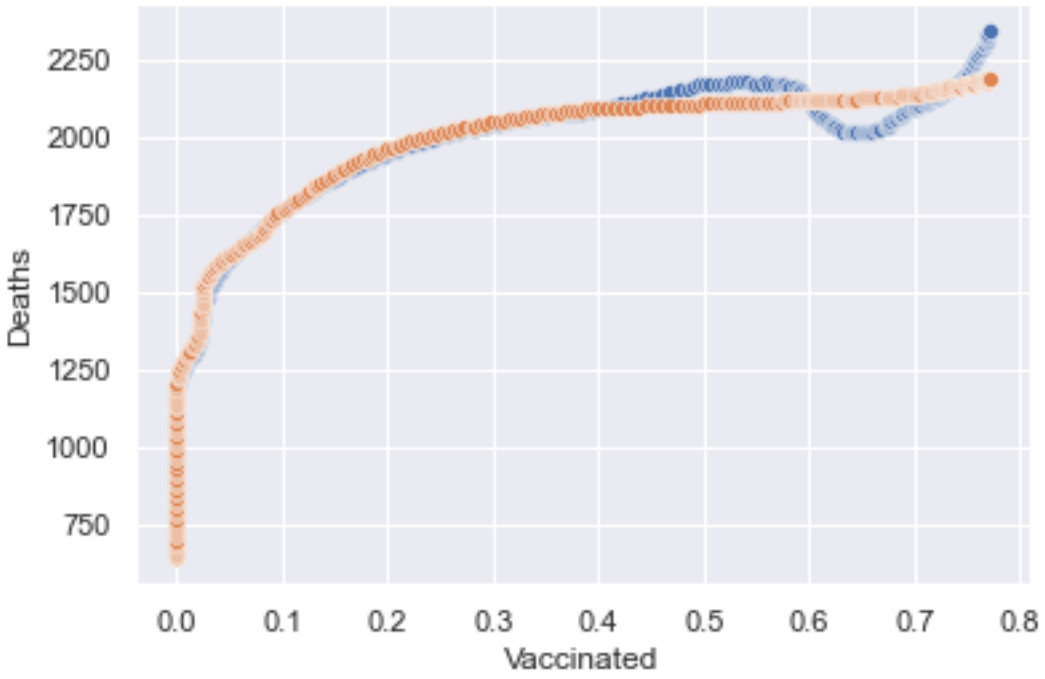

```
(108, 1) (360, 1)
r^2 Value: 0    0.968834
dtype: float64
```

```
array([[ 294.65756354],
       [ 784.62745595],
       [-200.18050882],
       [ -74.41498397]])
```

```
Italy
```


Deaths
2250
2000
1750
1500
1250
1000
750
0
50
100
150
200
250
300
350
Days

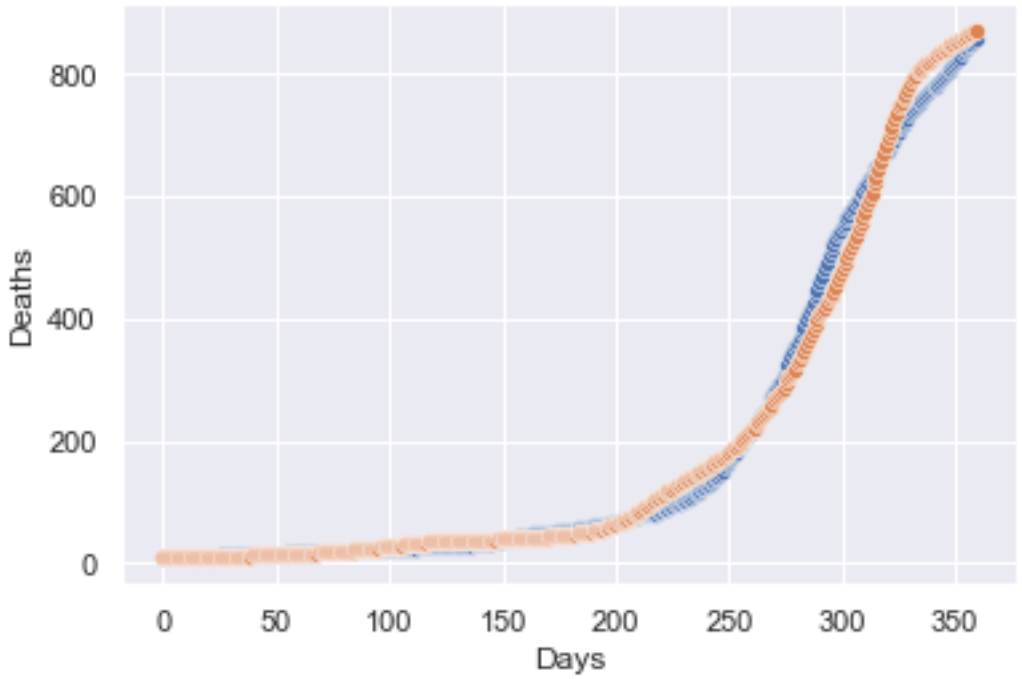

Deaths
2250
2000
1750
1500
1250
1000
750
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Fully_Vaccinated

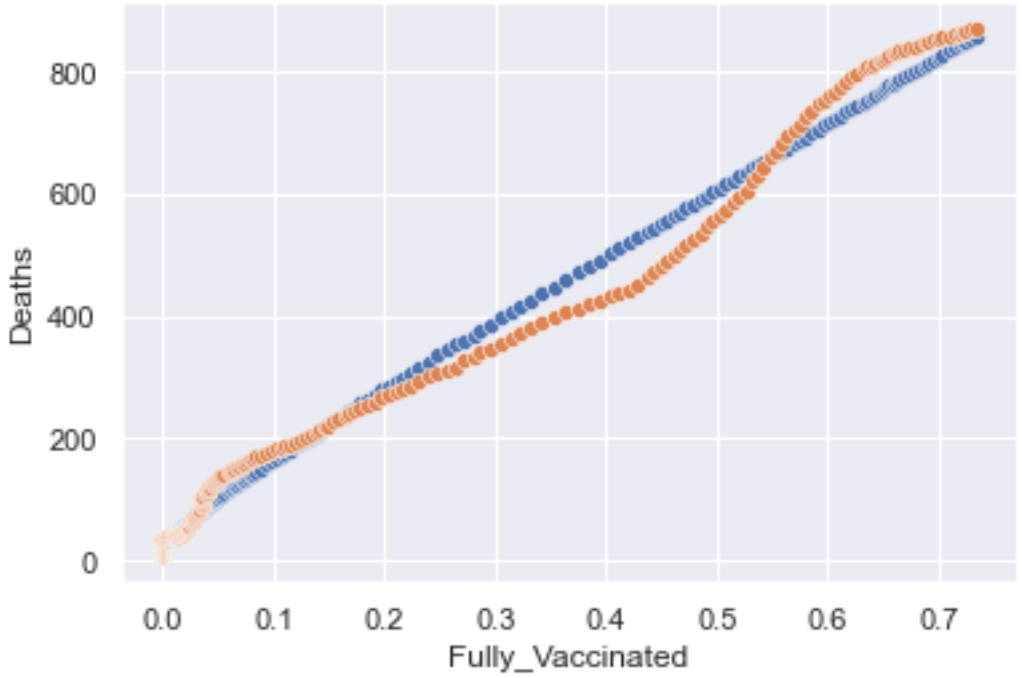


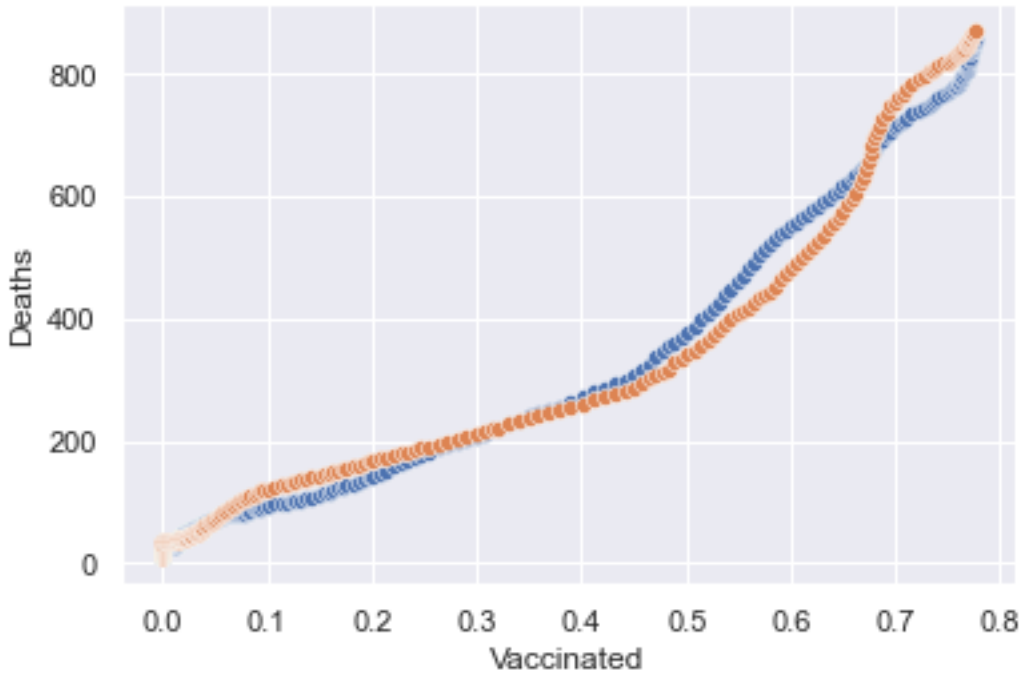

```
(108, 1) (360, 1)
r^2 Value: 0    0.984114
dtype: float64
```

```
array([[  767.62479312],
       [ 2597.27764257],
       [-1393.21312491],
       [  372.31483945]])
```

```
Malaysia
```


800
600
400
200
0
Deaths
0
50
100
150
200
250
300
350
Days

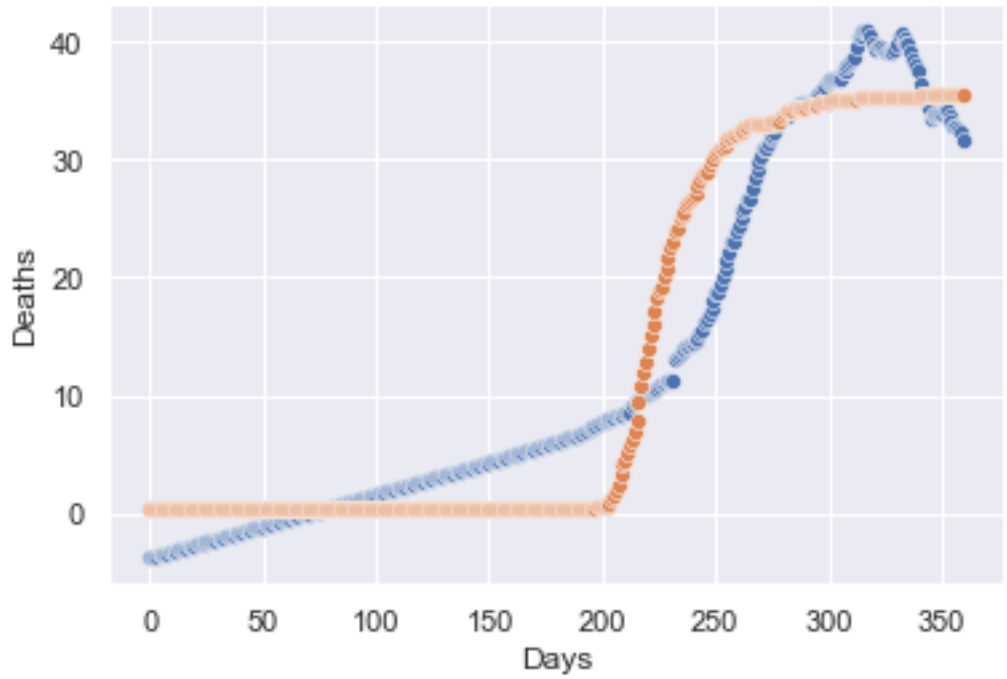

800
600
400
200
0
Deaths
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Fully_Vaccinated

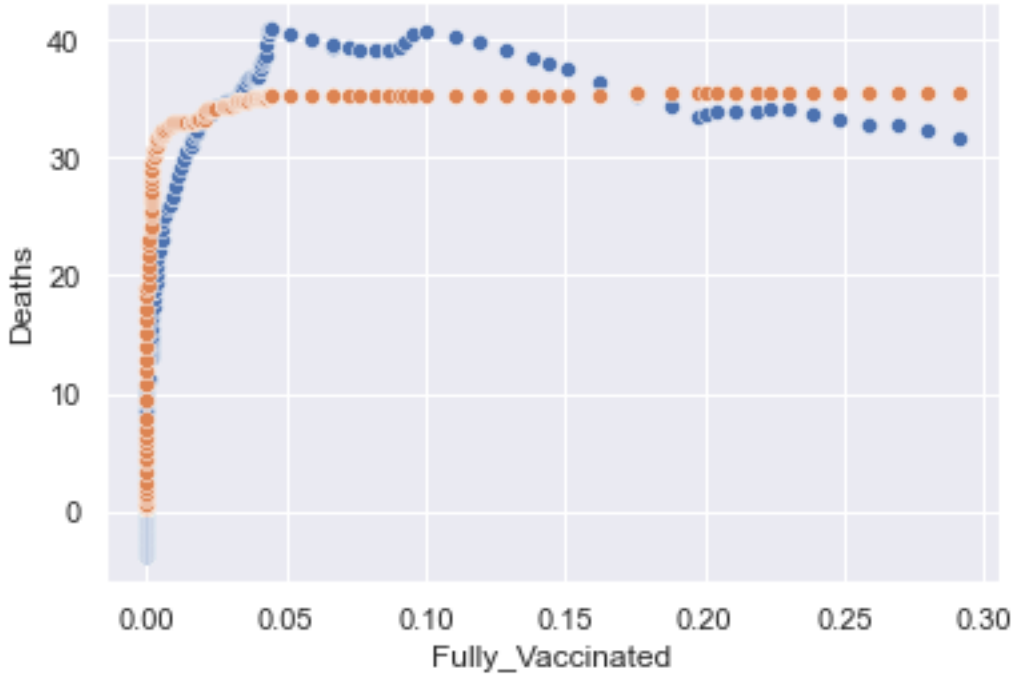


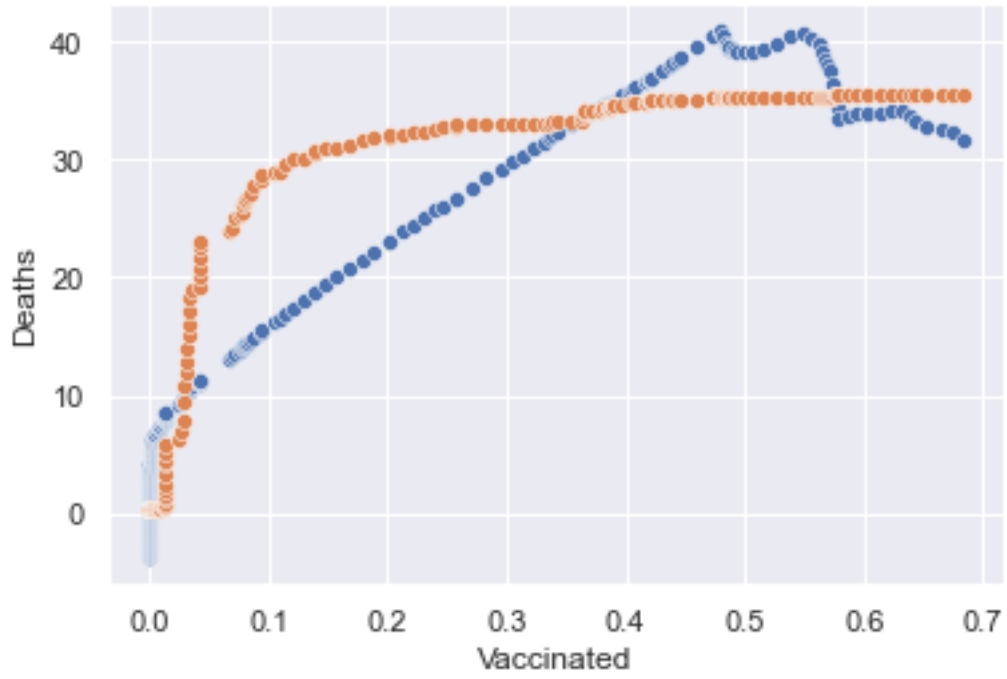

```
(108, 1) (360, 1)
r^2 Value: 0    0.991781
dtype: float64
```

```
array([[  8.39970793],
       [ 53.60552329],
       [737.15737086],
       [ 60.31247107]])
```

```
Taiwan
```


Deaths
40
30
20
10
0
0
50
100
150
200
250
300
350
Days


Deaths
40
30
20
10
0
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Fully_Vaccinated




```
(108, 1) (360, 1)
r^2 Value: 0    0.904165
dtype: float64
```

```
array([[ -3.77034914],
       [ 19.3596279 ],
       [-29.71951667],
       [ 45.86047053]])
```

[ ]: