What is Polynomial Regression?

Polynomial Regression is another algorithm used to characterize signals and generate a synthetic history. Polynomial regression is similar to Auto regression in that it relies on historic inputs to make predictions. Likewise Polynomial Regression is very similar to linear regression as it draws a line between the data points. The major difference is that linear regression will generate a straight line through the data points which often leaves significant outliers in the data whereas the polynomial regression will generate a line that curves based on the data better fitting the data presented.

Disease Example

Polynomial regression has a wide variety of applications, but the most common application in polynomial regression is with predicting the infection rate of diseases. It has been used in numerous cases both small and world wide. The function to compute this is found below.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_n x^n + \epsilon$$

The Result of the algorithm would be the predicted infection rate of a given disease.

Polynomial Regression in Raven

For Raven polynomial regression has a clear connection. We find that often the signals originating from reactors vary significantly due to multiple factors. Polynomial regression provides us with a clearer prediction with fewer errors than a linear model would in this case. Polynomial regression can be used to accurately characterize the signals originating from the reactor and generate synthetic histories from the parameters.

Code

Below is a coded demonstration of Polynomial regression using datasets from the CDC database. These datasets are focused on the infectious rates of the Chikungunya virus, and the number of smokers per day.

In [1]:
```
!pip install statsmodels
!pip install matplotlib

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.tsa.stattools as sm
```

```
Requirement already satisfied: statsmodels in c:\users\admin\miniconda3\lib\site-packages (0.12.2)
Requirement already satisfied: pandas>=0.21 in c:\users\admin\miniconda3\lib\site-packages (from statsmodels) (1.2.4)
Requirement already satisfied: numpy>=1.15 in c:\users\admin\miniconda3\lib\site-packages (from statsmodels) (1.20.2)
Requirement already satisfied: scipy>=1.1 in c:\users\admin\miniconda3\lib\site-packages (from statsmodels) (1.6.2)
Requirement already satisfied: patsy>=0.5 in c:\users\admin\miniconda3\lib\site-packages (from statsmodels) (0.5.1)
Requirement already satisfied: pytz>=2017.3 in c:\users\admin\miniconda3\lib\site-packages (from pandas>=0.21->statsmodels) (2021.1)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\admin\miniconda3\lib\site-packages (from pandas>=0.21->statsmodels) (2.8.1)
Requirement already satisfied: numpy>=1.15 in c:\users\admin\miniconda3\lib\site-packages (from statsmodels) (1.20.2)
Requirement already satisfied: six in c:\users\admin\miniconda3\lib\site-packages (from patsy>=0.5->statsmodels) (1.15.0)
Requirement already satisfied: numpy>=1.15 in c:\users\admin\miniconda3\lib\site-packages (from statsmodels) (1.20.2)
Requirement already satisfied: six in c:\users\admin\miniconda3\lib\site-packages (from patsy>=0.5->statsmodels) (1.15.0)
Requirement already satisfied: numpy>=1.15 in c:\users\admin\miniconda3\lib\site-packages (from statsmodels) (1.20.2)
Requirement already satisfied: matplotlib in c:\users\admin\miniconda3\lib\site-packages (3.4.1)
Requirement already satisfied: cycler>=0.10 in c:\users\admin\miniconda3\lib\site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\admin\miniconda3\lib\site-packages (from matplotlib) (1.3.1)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\admin\miniconda3\lib\site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: numpy>=1.16 in c:\users\admin\miniconda3\lib\site-packages (from matplotlib) (1.20.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\admin\miniconda3\lib\site-packages (from matplotlib) (8.2.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\admin\miniconda3\lib\site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: six in c:\users\admin\miniconda3\lib\site-packages (from cycler>=0.10->matplotlib) (1.15.0)
Requirement already satisfied: six in c:\users\admin\miniconda3\lib\site-packages (from cycler>=0.10->matplotlib) (1.15.0)
```

In [16]:
```
df = pd.read_csv("Chikungunya_Virus.csv")
```

In [17]:
```
df = pd.read_csv("localcases_0.csv")
```

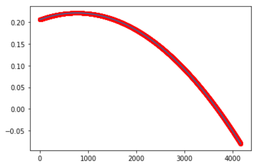
In [18]:
```
data = pd.read_csv('localcases.csv')
```

In [20]:
```
x = df.x
y = df.y

dat = pd.DataFrame({'x': x, 'y': y})
dat.to_csv('localcases.csv', index=False)

dat = pd.read_csv('localcases_0.csv')
# print(dat)

plt.plot(x, y, 'o', alpha=0.5, color='red')
plt.plot(dat.x, dat.y)
plt.show()
```



In [6]:
```
df = pd.read_csv("combined_smokers__everyday.csv")
```

In [7]:
```
df = pd.read_csv("regression_A_0.csv")
```

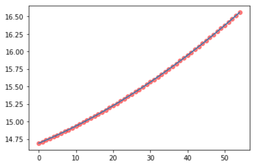
In [8]:
```
data = pd.read_csv('samples_0.csv')
```

In [21]:
```
x = data.x
y = data.y

dat = pd.DataFrame({'x': x, 'y': y})
dat.to_csv('regression_A_0.csv', index=False)

dat = pd.read_csv('samples_0.csv')
# print(dat)

plt.plot(x, y, 'o', alpha=0.5, color='red')
plt.plot(dat.x, dat.y)
plt.show()
```



Polynomial Regression in Raven Example

Here we see the same polynomial regression applied to Raven. The first Example shows the smoking predictions.

In [10]:
```
df = pd.read_csv("regression_A_0.csv")
```

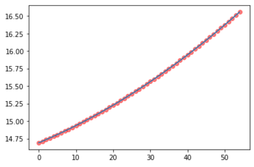
In [11]:
```
data = pd.read_csv('samples_0.csv')
```

In [22]:
```
x = data.x
y = data.y

dat = pd.DataFrame({'x': x, 'y': y})
dat.to_csv('regression_A_0.csv', index=False)

dat = pd.read_csv('samples_0.csv')
# print(dat)

plt.plot(x, y, 'o', alpha=0.5, color='red')
plt.plot(dat.x, dat.y)
plt.show()
```



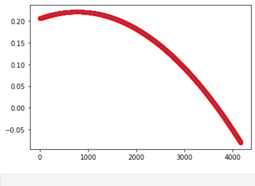
In [23]:
```
data = pd.read_csv('localcases_0.csv')
# data
```

In [24]:
```
x = data.x
y = data.y

dat = pd.DataFrame({'x': x, 'y': y})
dat.to_csv('localcases.csv', index=False)

dat = pd.read_csv('localcases_0.csv')
# print(dat)

plt.plot(x, y, 'o', alpha=0.5, color='red')
plt.plot(dat.x, dat.y)
plt.show()
```



In [ ]:
```
```