Jingwen Tan

012135461

Virus Recovered Graph (1/22/20-3/30/20)

In [36]:

```
import time
import numpy as np
import pandas as pd
import datetime as DT
import matplotlib.pyplot as plt
from numpy import log as ln
import matplotlib.dates as mdates
from scipy.optimize import curve_fit

rf =pd.read_csv('time_series_covid19_recovered_global.csv') #recovered file

########################### China Recovered Rate ############################

china_start = 40 #row 40,different from confirm sheet & death sheet since Cana
da didn't express its states in this one
china_end = 73  #row 73
ch_recover = rf.iloc[china_start:china_end] #china recovered data in the sheet

#create list for recovered numbers
ch_recoverTotal_y = []
ch_recoverJan = []
ch_recoverFeb = []
ch_recoverMar = []

#create list for time

datetimeTotal = []
datetimeJan = []
datetimeFeb = []
datetimeMar = []

numtimeTotal = []
numtimeJan = []
numtimeFeb = []
numtimeMar = []

Jan = 22 # January started date

Feb = 1 # Feburary started date

Mar = 1 # March started date

while Jan <= 31:

    #code for recovered data:y-axis
    date = str(Jan) #convert to string for following function
    daily_ch_recover = ch_recover['1/'+ date +'/2020'].tolist() # make it as l
ist
    sum_daily_recover = sum(daily_ch_recover) #count the total from all provin
ces
    ch_recoverJan.append(sum_daily_recover) #covert daily sum to an array

    #code for time: x-axis
    presentDate = DT.datetime(2020, 1, Jan)
    convert = mdates.date2num(presentDate)
    datetimeJan.append(presentDate)
```

```
    numtimeJan.append(convert)
    Jan+=1

while Feb <= 29:

    #code for recovered data:y-axis
    date = str(Feb) #convert to string for following function
    daily_ch_recover = ch_recover['2/'+ date +'/2020'].tolist() # make it as l
ist
    sum_daily_recover = sum(daily_ch_recover) #count the total from all provin
ces
    ch_recoverFeb.append(sum_daily_recover) #covert daily sum to an array

    #code for time: x-axis
    presentDate = DT.datetime(2020, 2, Feb)
    convert = mdates.date2num(presentDate)
    datetimeFeb.append(presentDate)
    numtimeFeb.append(convert)
    Feb+=1


while Mar <= 30:

    #code for recovered data:y-axis
    date = str(Mar) #convert to string for following function
    daily_ch_recover = ch_recover['3/'+ date +'/2020'].tolist() # make it as l
ist
    sum_daily_recover = sum(daily_ch_recover) #count the total from all provin
ces
    ch_recoverMar.append(sum_daily_recover) #covert daily sum to an array

    #code for time: x-axis
    presentDate = DT.datetime(2020, 3, Mar)
    convert = mdates.date2num(presentDate)
    datetimeMar.append(presentDate)
    numtimeMar.append(convert)
    Mar+=1

ch_recoverTotal_y = ch_recoverJan + ch_recoverFeb + ch_recoverMar

datetimeTotal = datetimeJan + datetimeFeb + datetimeMar

numtimeTotal = numtimeJan + numtimeFeb + numtimeMar

########################## Italy Recovered Rate ##########################
ItalyinSheet = 131 #located at row 131
it_recover = rf.iloc[ItalyinSheet]

it_recoverTotal_y = []
it_recoverJan = []
it_recoverFeb = []
it_recoverMar = []

Jan_it = 22
Feb_it = 1
Mar_it = 1
```

```
while Jan_it <= 31:
    date = str(Jan_it) #convert to string for following function
    daily_it_recover = it_recover['1/'+ date +'/2020']
    it_recoverJan.append(daily_it_recover) #covert daily sum to an array
    Jan_it+=1


while Feb_it <=29:
    date = str(Feb_it)
    daily_it_recover = it_recover['2/'+ date + '/2020']
    it_recoverFeb.append(daily_it_recover)
    Feb_it+=1


while Mar_it <=30:
    date = str(Mar_it)
    daily_it_recover = it_recover['3/'+ date + '/2020']
    it_recoverMar.append(daily_it_recover)
    Mar_it+=1

it_recoverTotal_y = it_recoverJan + it_recoverFeb + it_recoverMar

########################### Germany Recovered Rate ############################

GermanyinSheet = 112 #located at row 112

ge_recover = rf.iloc[GermanyinSheet]

ge_recoverTotal_y = []
ge_recoverJan = []
ge_recoverFeb = []
ge_recoverMar = []

Jan_ge = 22
Feb_ge = 1
Mar_ge = 1

while Jan_ge <= 31:
    date = str(Jan_ge) #convert to string for following function
    daily_ge_recover = ge_recover['1/'+ date +'/2020']
    ge_recoverJan.append(daily_ge_recover) #covert daily sum to an array
    Jan_ge+=1


while Feb_ge <=29:
    date = str(Feb_ge)
    daily_ge_recover = ge_recover['2/'+ date + '/2020']
    ge_recoverFeb.append(daily_ge_recover)
    Feb_ge+=1


while Mar_ge <=30:
    date = str(Mar_ge)
    daily_ge_recover = ge_recover['3/'+ date + '/2020']
    ge_recoverMar.append(daily_ge_recover)
    Mar_ge+=1
```

```
ge_recoverTotal_y = ge_recoverJan + ge_recoverFeb + ge_recoverMar


########################## Iran Recovered Rate ##########################

IraninSheet = 127 #located at row 127
ir_recover = rf.iloc[IraninSheet]

ir_recoverTotal_y = []
ir_recoverJan = []
ir_recoverFeb = []
ir_recoverMar = []

Jan_ir = 22
Feb_ir = 1
Mar_ir = 1

while Jan_ir <= 31:
    date = str(Jan_ir) #convert to string for following function
    daily_ir_recover = ir_recover['1/'+ date +'/2020']
    ir_recoverJan.append(daily_ir_recover) #covert daily sum to an array
    Jan_ir+=1


while Feb_ir <=29:
    date = str(Feb_ir)
    daily_ir_recover = ir_recover['2/'+ date + '/2020']
    ir_recoverFeb.append(daily_ir_recover)
    Feb_ir+=1


while Mar_ir <=30:
    date = str(Mar_ir)
    daily_ir_recover = ir_recover['3/'+ date + '/2020']
    ir_recoverMar.append(daily_ir_recover)
    Mar_ir+=1

ir_recoverTotal_y = ir_recoverJan + ir_recoverFeb + ir_recoverMar

########################## US Recovered Rate ##########################

USinSheet = 225 #located at row 225
us_recover = rf.iloc[USinSheet]

us_recoverTotal_y = []
us_recoverJan = []
us_recoverFeb = []
us_recoverMar = []

Jan_us = 22
Feb_us = 1
Mar_us = 1

while Jan_us <= 31:
    date = str(Jan_us) #convert to string for following function
    daily_us_recover = us_recover['1/'+ date +'/2020']
    us_recoverJan.append(daily_us_recover) #covert daily number to an array
```

```
    Jan_us+=1


while Feb_us <=29:
    date = str(Feb_us)
    daily_us_recover = us_recover['2/'+ date + '/2020']
    us_recoverFeb.append(daily_us_recover)
    Feb_us+=1


while Mar_us <=30:
    date = str(Mar_us)
    daily_us_recover = us_recover['3/'+ date + '/2020']
    us_recoverMar.append(daily_us_recover)
    Mar_us+=1

us_recoverTotal_y = us_recoverJan + us_recoverFeb + us_recoverMar

########################### Recovered points with curve-fit ###################
#######
plt.figure(1)
figure, axes = plt.subplots(figsize =(20,10))
axes.plot(datetimeTotal,ch_recoverTotal_y,'bo', label = "China")
axes.plot(datetimeTotal,it_recoverTotal_y,'ko', label = "Italy")
axes.plot(datetimeTotal,ge_recoverTotal_y,'go', label = "Germany")
axes.plot(datetimeTotal,ir_recoverTotal_y,'ro', label = "Iran")
axes.plot(datetimeTotal,us_recoverTotal_y,'mo', label = "United States")
ax=plt.gca()

plt.xlabel('Timeline')
plt.ylabel('#Recovered')

plt.xticks(datetimeTotal)
plt.xticks(rotation= 90)

plt.title('Coronavirus in Major Countries: Recovered Points & Curve-Fit')

########################### Recovered Graph curve-fit ##########################
#

#define equation that solve the fitted-curve
def func(x,a,b,c):
    return a*(x**2)+b*x+c


#Best-fit curve calculation

popt_ch, pcov =  curve_fit(func,numtimeTotal,ch_recoverTotal_y)
popt_it, pcov =  curve_fit(func,numtimeTotal,it_recoverTotal_y)
popt_ge, pcov =  curve_fit(func,numtimeTotal,ge_recoverTotal_y)
popt_ir, pcov =  curve_fit(func,numtimeTotal,ir_recoverTotal_y)
popt_us, pcov =  curve_fit(func,numtimeTotal,us_recoverTotal_y)

#curve limit
xFit = np.arange(737446,737514,0.01) # for future lab reminder: 0.01 instead o
f 1 prevent too high horizational line & low horizational data points
```

```
#Best-fit curve graph
axes.plot(xFit, func(xFit,*popt_ch),'b',label = 'China curve-fit: a=%5.3f, b=%
5.3f, c=%5.3f' % tuple(popt_ch) )
axes.plot(xFit, func(xFit,*popt_it),'k',label = 'Italy curve-fit: a=%5.3f, b=%
5.3f, c=%5.3f' % tuple(popt_it) )
axes.plot(xFit, func(xFit,*popt_ge),'g',label = 'Germany curve-fit: a=%5.3f, b
=%5.3f, c=%5.3f' % tuple(popt_ge) )
axes.plot(xFit, func(xFit,*popt_ir),'r',label = 'Iran curve-fit: a=%5.3f, b=%
5.3f, c=%5.3f' % tuple(popt_ir) )
axes.plot(xFit, func(xFit,*popt_us),'m',label = 'US curve-fit: a=%5.3f, b=%5.3
f, c=%5.3f' % tuple(popt_us) )

axes.legend()
########################## Recovered Graph  ##########################
plt.figure(2)
figure, axes = plt.subplots(figsize =(20,10))
axes.plot(datetimeTotal,ch_recoverTotal_y, label = "China")
axes.plot(datetimeTotal,it_recoverTotal_y, label = "Italy")
axes.plot(datetimeTotal,ge_recoverTotal_y, label = "Germany")
axes.plot(datetimeTotal,ir_recoverTotal_y, label = "Iran")
axes.plot(datetimeTotal,us_recoverTotal_y, label = "United States")
ax=plt.gca()

plt.xlabel('Timeline')
plt.ylabel('#Recovered')

plt.xticks(datetimeTotal)
plt.xticks(rotation= 90)

plt.title('Coronavirus in Major Countries: Recovered Graph')
axes.legend()

#Statement
print("China date list: "+'\n')
print(ch_recoverTotal_y )
print("")
print("Italy date list: "+'\n')
print(it_recoverTotal_y )
print("")
print("Germany date list: "+'\n')
print(ge_recoverTotal_y )
print("")
print("Iran date list: "+'\n')
print(ir_recoverTotal_y )
print("")
print("US date list: "+'\n')
print(us_recoverTotal_y )
print("")
print("China Recovered Equation: "+str(popt_ch[0])+"x^2 + "+str(popt_ch[1])+"x
+ "+str(popt_ch[2])+'\n')
print("Italy Recovered Equation: "+str(popt_it[0])+"x^2 + "+str(popt_it[1])+"x
+ "+str(popt_it[2])+'\n')
print("Germany Recovered Equation: "+str(popt_ge[0])+"x^2 + "+str(popt_ge[1])+
"x + "+str(popt_ge[2])+'\n')
print("Iran Recovered Equation: "+str(popt_ir[0])+"x^2 + "+str(popt_ir[1])+"x
 + "+str(popt_ir[2])+'\n')
```

```
print("US Recovered Equation: "+str(popt_us[0])+"x^2 + "+str(popt_us[1])+"x +
  "+str(popt_us[2])+'\n')
```

China date list:

[28, 30, 36, 39, 49, 58, 101, 120, 135, 214, 275, 463, 614, 843, 1115, 1477, 1999, 2596, 3219, 3918, 4636, 5082, 6217, 7977, 9298, 10755, 12462, 14206, 15962, 18014, 18704, 22699, 23187, 25015, 27676, 30084, 32930, 36329, 39320, 42162, 44854, 47450, 50001, 52292, 53944, 55539, 57388, 58804, 60181, 61644, 62901, 64196, 65660, 67017, 67910, 68798, 69755, 70535, 71266, 71857, 72362, 72814, 73280, 73773, 74181, 74720, 75100, 75582, 75923]

Italy date list:

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1, 3, 45, 46, 46, 83, 149, 160, 276, 414, 523, 589, 622, 724, 724, 1045, 1045, 1439, 1966, 2335, 2749, 2941, 4025, 4440, 4440, 6072, 7024, 7024, 8326, 9362, 10361, 10950, 12384, 13030, 14620]

Germany date list:

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 12, 12, 12, 14, 14, 14, 14, 14, 15, 16, 16, 16, 16, 16, 16, 16, 16, 17, 18, 18, 18, 18, 25, 25, 46, 46, 46, 67, 67, 105, 113, 180, 233, 266, 266, 3243, 3547, 5673, 6658, 8481, 9211, 13500]

Iran date list:

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 49, 49, 73, 123, 175, 291, 291, 552, 739, 913, 1669, 2134, 2394, 2731, 2959, 2959, 2959, 2959, 4590, 4590, 5389, 5389, 5710, 6745, 7635, 7931, 7931, 8913, 9625, 10457, 11133, 11679, 12391, 13911]

US date list:

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 5, 5, 5, 5, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 12, 12, 12, 12, 17, 17, 105, 121, 147, 176, 178, 178, 348, 361, 681, 869, 1072, 2665, 5644]

China Recovered Equation: 0.09917858349974548x^2 + -144855.85289396535x + 52887402124.73296

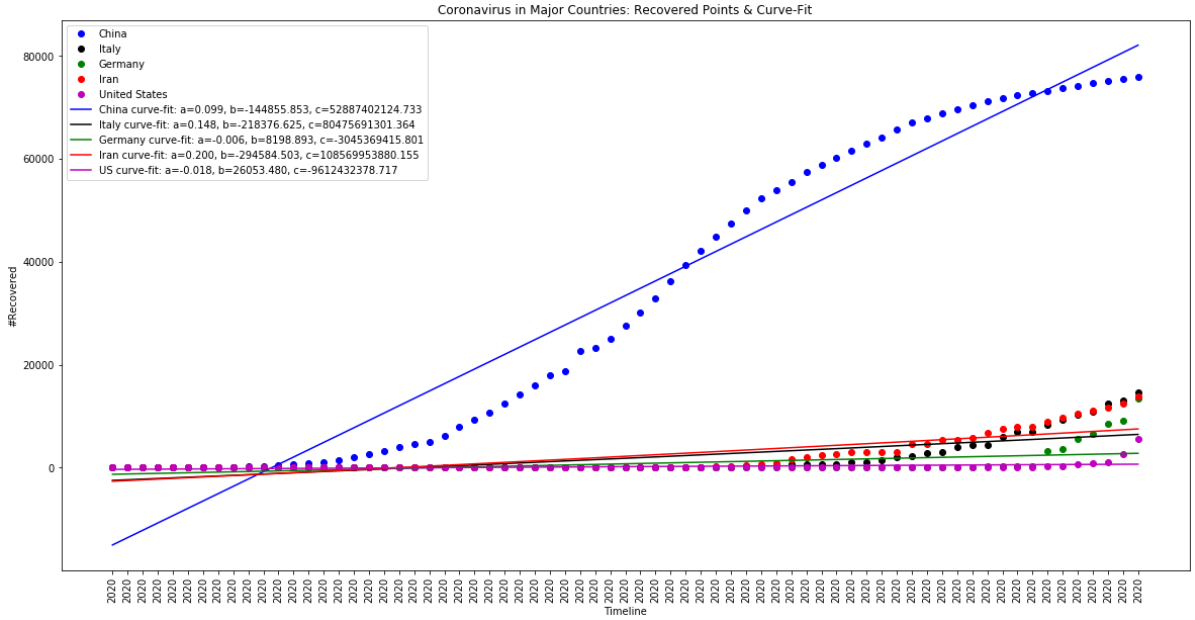
Italy Recovered Equation: 0.14814515232862943x^2 + -218376.6251454343x + 80475691301.36414

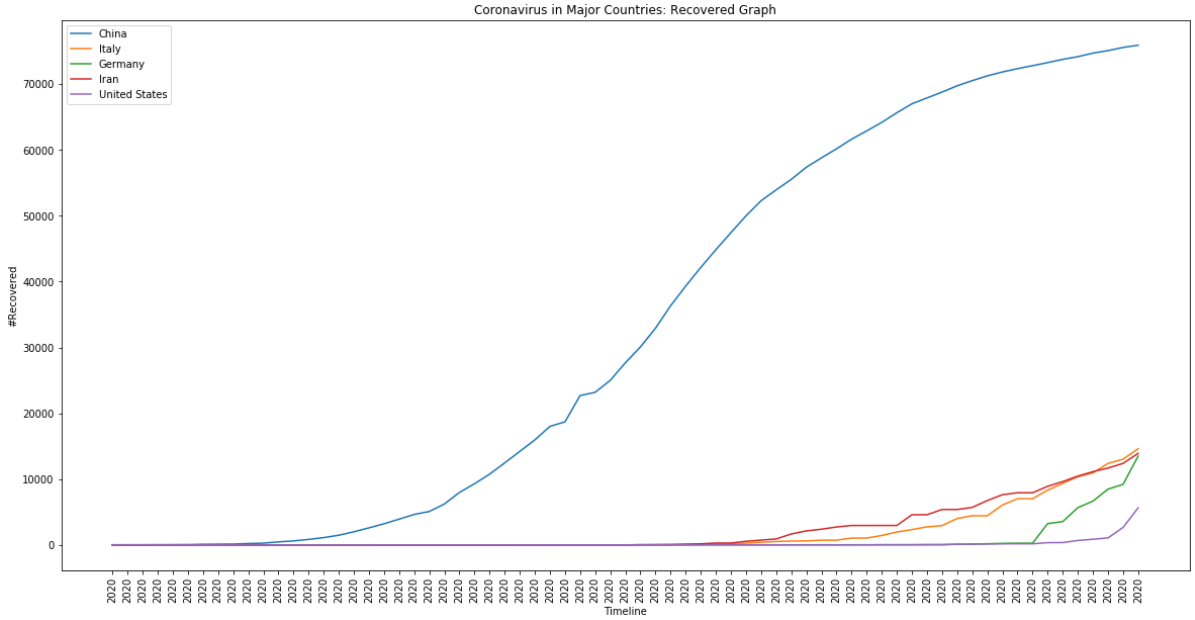
Germany Recovered Equation: -0.005518068339942399x^2 + 8198.892513520792x + -3045369415.800907

Iran Recovered Equation: 0.19982510253445437x^2 + -294584.5027441959x + 108569953880.15509

US Recovered Equation: -0.017653793716876368x^2 + 26053.480135261227x + -9612432378.716787

<Figure size 432x288 with 0 Axes>





Virus Death Graph (1/22/20-3/30/20)

In [34]:

```
import time
import numpy as np
import pandas as pd
import datetime as DT
import matplotlib.pyplot as plt
from numpy import log as ln
import matplotlib.dates as mdates
from scipy.optimize import curve_fit

df =pd.read_csv('time_series_covid19_deaths_global.csv') #death file

########################### China Death Rate ############################

china_start = 49 #row 49
china_end = 82  #row 82
ch_death = df.iloc[china_start:china_end] #china death data in the sheet

#create list for death numbers
ch_deathTotal_y = []
ch_deathJan = []
ch_deathFeb = []
ch_deathMar = []

#create list for time

datetimeTotal = []
datetimeJan = []
datetimeFeb = []
datetimeMar = []

numtimeTotal = []
numtimeJan = []
numtimeFeb = []
numtimeMar = []

Jan = 22 # January started date

Feb = 1 # Feburary started date

Mar = 1 # March started date

while Jan <= 31:

    #code for death data:y-axis
    date = str(Jan) #convert to string for following function
    daily_ch_death = ch_death['1/'+ date +'/2020'].tolist() # make it as list
    sum_daily_death = sum(daily_ch_death) #count the total from all provinces
    ch_deathJan.append(sum_daily_death) #covert daily sum to an array

    #code for time: x-axis
    presentDate = DT.datetime(2020, 1, Jan)
    convert = mdates.date2num(presentDate)
    datetimeJan.append(presentDate)
    numtimeJan.append(convert)
    Jan+=1
```

```
while Feb <= 29:

    #code for death data:y-axis
    date = str(Feb) #convert to string for following function
    daily_ch_death = ch_death['2/'+ date +'/2020'].tolist() # make it as list
    sum_daily_death = sum(daily_ch_death) #count the total from all provinces
    ch_deathFeb.append(sum_daily_death) #covert daily sum to an array

    #code for time: x-axis
    presentDate = DT.datetime(2020, 2, Feb)
    convert = mdates.date2num(presentDate)
    datetimeFeb.append(presentDate)
    numtimeFeb.append(convert)
    Feb+=1


while Mar <= 30:

    #code for death data:y-axis
    date = str(Mar) #convert to string for following function
    daily_ch_death = ch_death['3/'+ date +'/2020'].tolist() # make it as list
    sum_daily_death = sum(daily_ch_death) #count the total from all provinces
    ch_deathMar.append(sum_daily_death) #covert daily sum to an array

    #code for time: x-axis
    presentDate = DT.datetime(2020, 3, Mar)
    convert = mdates.date2num(presentDate)
    datetimeMar.append(presentDate)
    numtimeMar.append(convert)
    Mar+=1

ch_deathTotal_y = ch_deathJan + ch_deathFeb + ch_deathMar

datetimeTotal = datetimeJan + datetimeFeb + datetimeMar

numtimeTotal = numtimeJan + numtimeFeb + numtimeMar

########################## Italy Death Rate ##########################
ItalyinSheet = 137 #located at row 137
it_death = df.iloc[ItalyinSheet]

it_deathTotal_y = []
it_deathJan = []
it_deathFeb = []
it_deathMar = []

Jan_it = 22
Feb_it = 1
Mar_it = 1

while Jan_it <= 31:
    date = str(Jan_it) #convert to string for following function
    daily_it_death = it_death['1/'+ date +'/2020']
    it_deathJan.append(daily_it_death) #covert daily sum to an array
    Jan_it+=1
```

```
while Feb_it <=29:
    date = str(Feb_it)
    daily_it_death = it_death['2/'+ date + '/2020']
    it_deathFeb.append(daily_it_death)
    Feb_it+=1


while Mar_it <=30:
    date = str(Mar_it)
    daily_it_death = it_death['3/'+ date + '/2020']
    it_deathMar.append(daily_it_death)
    Mar_it+=1

it_deathTotal_y = it_deathJan + it_deathFeb + it_deathMar

########################## Germany Death Rate ##########################

GermanyinSheet = 120 #located at row 120

ge_death = df.iloc[GermanyinSheet]

ge_deathTotal_y = []
ge_deathJan = []
ge_deathFeb = []
ge_deathMar = []

Jan_ge = 22
Feb_ge = 1
Mar_ge = 1

while Jan_ge <= 31:
    date = str(Jan_ge) #convert to string for following function
    daily_ge_death = ge_death['1/'+ date +'/2020']
    ge_deathJan.append(daily_ge_death) #covert daily sum to an array
    Jan_ge+=1


while Feb_ge <=29:
    date = str(Feb_ge)
    daily_ge_death = ge_death['2/'+ date + '/2020']
    ge_deathFeb.append(daily_ge_death)
    Feb_ge+=1


while Mar_ge <=30:
    date = str(Mar_ge)
    daily_ge_death = ge_death['3/'+ date + '/2020']
    ge_deathMar.append(daily_ge_death)
    Mar_ge+=1

ge_deathTotal_y = ge_deathJan + ge_deathFeb + ge_deathMar


########################## Iran Death Rate ##########################

IraninSheet = 133 #located at row 133
ir_death = df.iloc[IraninSheet]
```

```
ir_deathTotal_y = []
ir_deathJan = []
ir_deathFeb = []
ir_deathMar = []

Jan_ir = 22
Feb_ir = 1
Mar_ir = 1

while Jan_ir <= 31:
    date = str(Jan_ir) #convert to string for following function
    daily_ir_death = ir_death['1/'+ date +'/2020']
    ir_deathJan.append(daily_ir_death) #covert daily sum to an array
    Jan_ir+=1


while Feb_ir <=29:
    date = str(Feb_ir)
    daily_ir_death = ir_death['2/'+ date + '/2020']
    ir_deathFeb.append(daily_ir_death)
    Feb_ir+=1


while Mar_ir <=30:
    date = str(Mar_ir)
    daily_ir_death = ir_death['3/'+ date + '/2020']
    ir_deathMar.append(daily_ir_death)
    Mar_ir+=1

ir_deathTotal_y = ir_deathJan + ir_deathFeb + ir_deathMar

########################### US Death Rate ############################

USinSheet = 225 #located at row 225
us_death = df.iloc[USinSheet]

us_deathTotal_y = []
us_deathJan = []
us_deathFeb = []
us_deathMar = []

Jan_us = 22
Feb_us = 1
Mar_us = 1

while Jan_us <= 31:
    date = str(Jan_us) #convert to string for following function
    daily_us_death = us_death['1/'+ date +'/2020']
    us_deathJan.append(daily_us_death) #covert daily number to an array
    Jan_us+=1


while Feb_us <=29:
    date = str(Feb_us)
    daily_us_death = us_death['2/'+ date + '/2020']
    us_deathFeb.append(daily_us_death)
```

```
    Feb_us+=1


while Mar_us <=30:
    date = str(Mar_us)
    daily_us_death = us_death['3/'+ date + '/2020']
    us_deathMar.append(daily_us_death)
    Mar_us+=1

us_deathTotal_y = us_deathJan + us_deathFeb + us_deathMar

########################## Death Graph points with curve-fit ################
#########
plt.figure(1)
figure, axes = plt.subplots(figsize =(20,10))
axes.plot(datetimeTotal,ch_deathTotal_y,'bo', label = "China")
axes.plot(datetimeTotal,it_deathTotal_y,'ko', label = "Italy")
axes.plot(datetimeTotal,ge_deathTotal_y,'go', label = "Germany")
axes.plot(datetimeTotal,ir_deathTotal_y,'ro', label = "Iran")
axes.plot(datetimeTotal,us_deathTotal_y,'mo', label = "United States")
ax=plt.gca()

plt.xlabel('Timeline')
plt.ylabel('#Death')

plt.xticks(rotation = 90)
plt.xticks(datetimeTotal)

plt.title('Coronavirus in Major Countries: Death plot & Curve-fit')

########################## Death Graph curve-fit ##########################

#define equation that solve the fitted-curve
def func(x,a,b,c):
    return a*(x**2)+b*x+c


#Best-fit curve calculation

popt_ch, pcov =  curve_fit(func,numtimeTotal,ch_deathTotal_y)
popt_it, pcov =  curve_fit(func,numtimeTotal,it_deathTotal_y)
popt_ge, pcov =  curve_fit(func,numtimeTotal,ge_deathTotal_y)
popt_ir, pcov =  curve_fit(func,numtimeTotal,ir_deathTotal_y)
popt_us, pcov =  curve_fit(func,numtimeTotal,us_deathTotal_y)

#curve limit
xFit = np.arange(737446,737514,0.01) # for future lab reminder: 0.01 instead o
f 1 prevent too high horizational line & low horizational data points

#Best-fit curve graph
axes.plot(xFit, func(xFit,*popt_ch),'b',label = 'China curve-fit: a=%5.3f, b=%
5.3f, c=%5.3f' % tuple(popt_ch) )
axes.plot(xFit, func(xFit,*popt_it),'k',label = 'Italy curve-fit: a=%5.3f, b=%
5.3f, c=%5.3f' % tuple(popt_it) )
axes.plot(xFit, func(xFit,*popt_ge),'g',label = 'Germany curve-fit: a=%5.3f, b
=%5.3f, c=%5.3f' % tuple(popt_ge) )
axes.plot(xFit, func(xFit,*popt_ir),'r',label = 'Iran curve-fit: a=%5.3f, b=%
```

```
5.3f, c=%5.3f' % tuple(popt_ir) )
axes.plot(xFit, func(xFit,*popt_us),'m',label = 'US curve-fit: a=%5.3f, b=%5.3
f, c=%5.3f' % tuple(popt_us) )

axes.legend()

########################## DeathGraph  ###################
plt.figure(2)
figure, axes = plt.subplots(figsize =(20,10))
axes.plot(datetimeTotal,ch_deathTotal_y, label = "China")
axes.plot(datetimeTotal,it_deathTotal_y, label = "Italy")
axes.plot(datetimeTotal,ge_deathTotal_y, label = "Germany")
axes.plot(datetimeTotal,ir_deathTotal_y, label = "Iran")
axes.plot(datetimeTotal,us_deathTotal_y, label = "United States")
ax=plt.gca()


plt.xlabel('Timeline')
plt.ylabel('#Death')

plt.xticks(rotation = 90)
plt.xticks(datetimeTotal)

plt.title('Coronavirus in Major Countries: Death Graph')
axes.legend()

#Equation
print("China date list: "+'\n')
print(ch_deathTotal_y )
print("")
print("Italy date list: "+'\n')
print(it_deathTotal_y )
print("")
print("Germany date list: "+'\n')
print(ge_deathTotal_y )
print("")
print("Iran date list: "+'\n')
print(ir_deathTotal_y )
print("")
print("US date list: "+'\n')
print(us_deathTotal_y )
print("")
print("China Death Equation: "+str(popt_ch[0])+"x^2 + "+str(popt_ch[1])+"x + "
+str(popt_ch[2])+'\n')
print("Italy Death Equation: "+str(popt_it[0])+"x^2 + "+str(popt_it[1])+"x + "
+str(popt_it[2])+'\n')
print("Germany Death Equation: "+str(popt_ge[0])+"x^2 + "+str(popt_ge[1])+"x +
"+str(popt_ge[2])+'\n')
print("Iran Death Equation: "+str(popt_ir[0])+"x^2 + "+str(popt_ir[1])+"x + "+
str(popt_ir[2])+'\n')
print("US Death Equation: "+str(popt_us[0])+"x^2 + "+str(popt_us[1])+"x + "+st
r(popt_us[2])+'\n')
```

China date list:

[17, 18, 26, 42, 56, 82, 131, 133, 171, 213, 259, 361, 425, 491, 563, 633, 71
8, 805, 905, 1012, 1112, 1117, 1369, 1521, 1663, 1766, 1864, 2003, 2116, 223
8, 2238, 2443, 2445, 2595, 2665, 2717, 2746, 2790, 2837, 2872, 2914, 2947, 29
83, 3015, 3044, 3072, 3100, 3123, 3139, 3161, 3172, 3180, 3193, 3203, 3217, 3
230, 3241, 3249, 3253, 3259, 3274, 3274, 3281, 3285, 3291, 3296, 3299, 3304,
3308]

Italy date list:

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 2, 3, 7, 10, 12, 17, 21, 29, 34, 52, 79, 107, 148, 197, 23
3, 366, 463, 631, 827, 827, 1266, 1441, 1809, 2158, 2503, 2978, 3405, 4032, 4
825, 5476, 6077, 6820, 7503, 8215, 9134, 10023, 10779, 11591]

Germany date list:

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 3, 3,
7, 9, 11, 17, 24, 28, 44, 67, 84, 94, 123, 157, 206, 267, 342, 433, 533, 645]

Iran date list:

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 2, 2, 4, 5, 8, 12, 16, 19, 26, 34, 43, 54, 66, 77, 92, 107, 124, 14
5, 194, 237, 291, 354, 429, 514, 611, 724, 853, 988, 1135, 1284, 1433, 1556,
1685, 1812, 1934, 2077, 2234, 2378, 2517, 2640, 2757]

US date list:

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 6, 7, 11, 12, 14, 17, 21, 22, 2
8, 36, 40, 47, 54, 63, 85, 108, 118, 200, 244, 307, 417, 557, 706, 942, 1209,
1581, 2026, 2467, 2978]

China Death Equation: -0.035063183852493865x^2 + 51775.39165191284x + -191132
63264.193935

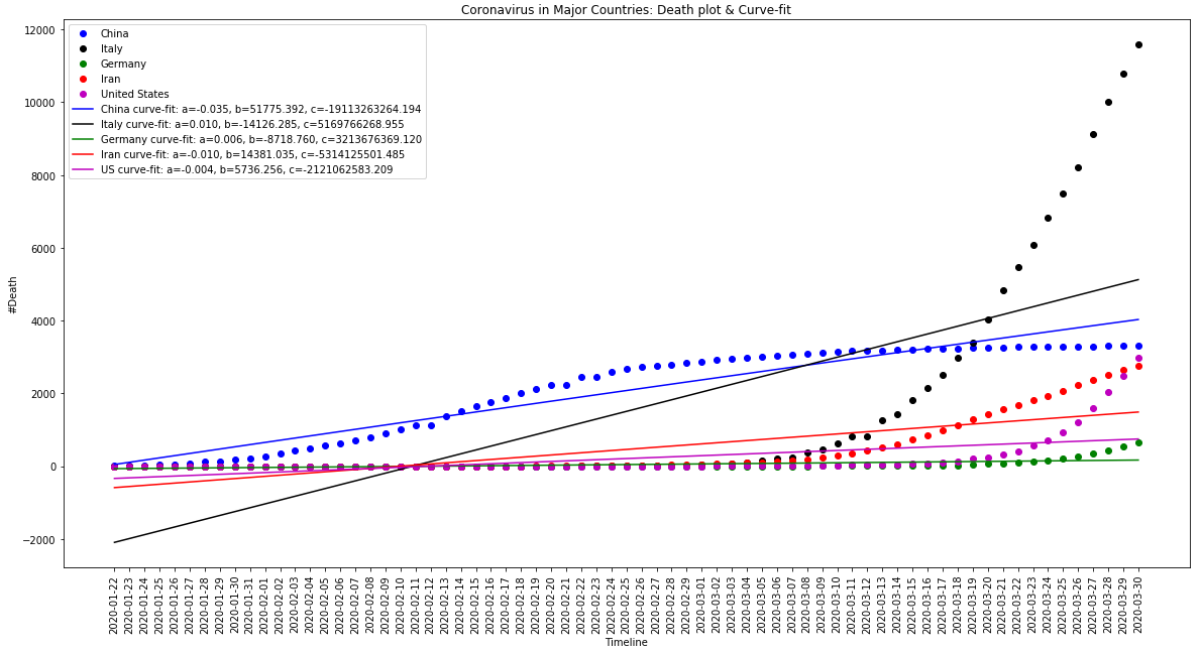
Italy Death Equation: 0.00964940711650058 2x^2 + -14126.285390628718x + 516976
6268.9553

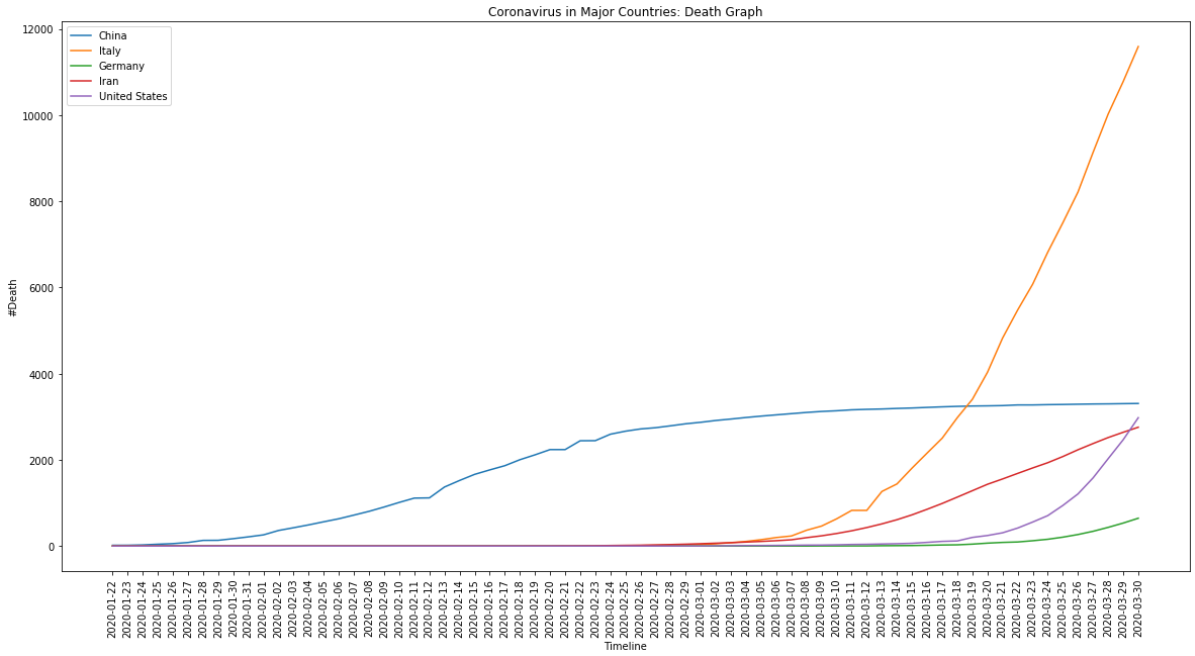
Germany Death Equation: 0.005913535505054295x^2 + -8718.759560106968x + 32136
76369.120166

Iran Death Equation: -0.00972940943159832x^2 + 14381.034802902115x + -5314125
501.48465

US Death Equation: -0.003878288664475946x^2 + 5736.255665651621x + -212106258
3.2093184

<Figure size 432x288 with 0 Axes>





Virus Confirmed Graph (1/22/20-3/30/20)

In [31]:

```
import time
import numpy as np
import pandas as pd
import datetime as DT
import matplotlib.pyplot as plt
from numpy import log as ln
import matplotlib.dates as mdates
from scipy.optimize import curve_fit

cf =pd.read_csv('time_series_covid19_confirmed_global.csv') #confirmed file

########################### China Confirmed Rate ############################

china_start = 49 #row 49
china_end = 82  #row 82
ch_confirm = cf.iloc[china_start:china_end] #china recovered data in the sheet

#create list for Confirm numbers
ch_confirmTotal_y = []
ch_confirmJan = []
ch_confirmFeb = []
ch_confirmMar = []

#create list for time

datetimeTotal = []
datetimeJan = []
datetimeFeb = []
datetimeMar = []

numtimeTotal = []
numtimeJan = []
numtimeFeb = []
numtimeMar = []

Jan = 22 # January started date

Feb = 1 # Feburary started date

Mar = 1 # March started date

while Jan <= 31:

    #code for Confirm data:y-axis
    date = str(Jan) #convert to string for following function
    daily_ch_confirm = ch_confirm['1/'+ date +'/2020'].tolist() # make it as l
ist
    sum_daily_confirm = sum(daily_ch_confirm) #count the total from all provin
ces
    ch_confirmJan.append(sum_daily_confirm) #covert daily sum to an array

    #code for time: x-axis
    presentDate = DT.datetime(2020, 1, Jan)
    convert = mdates.date2num(presentDate)
    datetimeJan.append(presentDate)
    numtimeJan.append(convert)
```

```
    Jan+=1

while Feb <= 29:

    #code for Confirm data:y-axis
    date = str(Feb) #convert to string for following function
    daily_ch_confirm = ch_confirm['2/'+ date +'/2020'].tolist() # make it as l
ist
    sum_daily_confirm = sum(daily_ch_confirm) #count the total from all provin
ces
    ch_confirmFeb.append(sum_daily_confirm) #covert daily sum to an array

    #code for time: x-axis
    presentDate = DT.datetime(2020, 2, Feb)
    convert = mdates.date2num(presentDate)
    datetimeFeb.append(presentDate)
    numtimeFeb.append(convert)
    Feb+=1


while Mar <= 30:

    #code for Confirm data:y-axis
    date = str(Mar) #convert to string for following function
    daily_ch_confirm = ch_confirm['3/'+ date +'/2020'].tolist() # make it as l
ist
    sum_daily_confirm = sum(daily_ch_confirm) #count the total from all provin
ces
    ch_confirmMar.append(sum_daily_confirm) #covert daily sum to an array

    #code for time: x-axis
    presentDate = DT.datetime(2020, 3, Mar)
    convert = mdates.date2num(presentDate)
    datetimeMar.append(presentDate)
    numtimeMar.append(convert)
    Mar+=1

ch_confirmTotal_y = ch_confirmJan + ch_confirmFeb + ch_confirmMar

datetimeTotal = datetimeJan + datetimeFeb + datetimeMar

numtimeTotal = numtimeJan + numtimeFeb + numtimeMar

########################## Italy Confirm Rate ##########################
ItalyinSheet = 137 #located at row 137
it_confirm = cf.iloc[ItalyinSheet]

it_confirmTotal_y = []
it_confirmJan = []
it_confirmFeb = []
it_confirmMar = []

Jan_it = 22
Feb_it = 1
Mar_it = 1

while Jan_it <= 31:
```

```
    date = str(Jan_it) #convert to string for following function
    daily_it_confirm = it_confirm['1/'+ date +'/2020']
    it_confirmJan.append(daily_it_confirm) #covert daily sum to an array
    Jan_it+=1


while Feb_it <=29:
    date = str(Feb_it)
    daily_it_confirm = it_confirm['2/'+ date + '/2020']
    it_confirmFeb.append(daily_it_confirm)
    Feb_it+=1


while Mar_it <=30:
    date = str(Mar_it)
    daily_it_confirm = it_confirm['3/'+ date + '/2020']
    it_confirmMar.append(daily_it_confirm)
    Mar_it+=1

it_confirmTotal_y = it_confirmJan + it_confirmFeb + it_confirmMar

########################### Germany Confirmed Rate ############################

GermanyinSheet = 120 #located at row 120

ge_confirm = cf.iloc[GermanyinSheet]

ge_confirmTotal_y = []
ge_confirmJan = []
ge_confirmFeb = []
ge_confirmMar = []

Jan_ge = 22
Feb_ge = 1
Mar_ge = 1

while Jan_ge <= 31:
    date = str(Jan_ge) #convert to string for following function
    daily_ge_confirm = ge_confirm['1/'+ date +'/2020']
    ge_confirmJan.append(daily_ge_confirm) #covert daily sum to an array
    Jan_ge+=1


while Feb_ge <=29:
    date = str(Feb_ge)
    daily_ge_confirm = ge_confirm['2/'+ date + '/2020']
    ge_confirmFeb.append(daily_ge_confirm)
    Feb_ge+=1


while Mar_ge <=30:
    date = str(Mar_ge)
    daily_ge_confirm = ge_confirm['3/'+ date + '/2020']
    ge_confirmMar.append(daily_ge_confirm)
    Mar_ge+=1

ge_confirmTotal_y = ge_confirmJan + ge_confirmFeb + ge_confirmMar
```

```
########################## Iran Confirm Rate ##########################

IraninSheet = 133 #located at row 133
ir_confirm = cf.iloc[IraninSheet]

ir_confirmTotal_y = []
ir_confirmJan = []
ir_confirmFeb = []
ir_confirmMar = []

Jan_ir = 22
Feb_ir = 1
Mar_ir = 1

while Jan_ir <= 31:
    date = str(Jan_ir) #convert to string for following function
    daily_ir_confirm = ir_confirm['1/'+ date +'/2020']
    ir_confirmJan.append(daily_ir_confirm) #covert daily sum to an array
    Jan_ir+=1


while Feb_ir <=29:
    date = str(Feb_ir)
    daily_ir_confirm = ir_confirm['2/'+ date + '/2020']
    ir_confirmFeb.append(daily_ir_confirm)
    Feb_ir+=1


while Mar_ir <=30:
    date = str(Mar_ir)
    daily_ir_confirm = ir_confirm['3/'+ date + '/2020']
    ir_confirmMar.append(daily_ir_confirm)
    Mar_ir+=1

ir_confirmTotal_y = ir_confirmJan + ir_confirmFeb + ir_confirmMar

########################## US Confirm Rate ##########################

USinSheet = 225 #located at row 225
us_confirm = cf.iloc[USinSheet]

us_confirmTotal_y = []
us_confirmJan = []
us_confirmFeb = []
us_confirmMar = []

Jan_us = 22
Feb_us = 1
Mar_us = 1

while Jan_us <= 31:
    date = str(Jan_us) #convert to string for following function
    daily_us_confirm = us_confirm['1/'+ date +'/2020']
    us_confirmJan.append(daily_us_confirm) #covert daily number to an array
    Jan_us+=1
```

```
while Feb_us <=29:
    date = str(Feb_us)
    daily_us_confirm = us_confirm['2/'+ date + '/2020']
    us_confirmFeb.append(daily_us_confirm)
    Feb_us+=1


while Mar_us <=30:
    date = str(Mar_us)
    daily_us_confirm = us_confirm['3/'+ date + '/2020']
    us_confirmMar.append(daily_us_confirm)
    Mar_us+=1

us_confirmTotal_y = us_confirmJan + us_confirmFeb + us_confirmMar

########################### Confirm Graph plot with curve-fit ################
#########
plt.figure(1)
figure, axes = plt.subplots(figsize =(20,10))
axes.plot(datetimeTotal,ch_confirmTotal_y,'bo', label = "China")
axes.plot(datetimeTotal,it_confirmTotal_y,'ko', label = "Italy")
axes.plot(datetimeTotal,ge_confirmTotal_y,'go', label = "Germany")
axes.plot(datetimeTotal,ir_confirmTotal_y,'ro', label = "Iran")
axes.plot(datetimeTotal,us_confirmTotal_y,'mo', label = "United States")
ax=plt.gca()


plt.xlabel('Timeline')
plt.ylabel('#Confirmed')

plt.xticks(rotation = 90)
plt.xticks(datetimeTotal)

plt.title('Coronavirus in Major Countries: Confirmed point & curve-fit')

########################### Confirmed Graph curve-fit ########################
#

#define equation that solve the fitted-curve
def func(x,a,b,c):
    return a*(x**2)+b*x+c


#Best-fit curve calculation

popt_ch, pcov =  curve_fit(func,numtimeTotal,ch_confirmTotal_y)
popt_it, pcov =  curve_fit(func,numtimeTotal,it_confirmTotal_y)
popt_ge, pcov =  curve_fit(func,numtimeTotal,ge_confirmTotal_y)
popt_ir, pcov =  curve_fit(func,numtimeTotal,ir_confirmTotal_y)
popt_us, pcov =  curve_fit(func,numtimeTotal,us_confirmTotal_y)

#curve limit
xFit = np.arange(737446,737514,0.01) # for future lab reminder: 0.01 instead o
f 1 prevent too high horizational line & low horizational data points
```

```
#Best-fit curve graph
axes.plot(xFit, func(xFit,*popt_ch),'b',label = 'China curve-fit: a=%5.3f, b=%
5.3f, c=%5.3f' % tuple(popt_ch) )
axes.plot(xFit, func(xFit,*popt_it),'k',label = 'Italy curve-fit: a=%5.3f, b=%
5.3f, c=%5.3f' % tuple(popt_it) )
axes.plot(xFit, func(xFit,*popt_ge),'g',label = 'Germany curve-fit: a=%5.3f, b
=%5.3f, c=%5.3f' % tuple(popt_ge) )
axes.plot(xFit, func(xFit,*popt_ir),'r',label = 'Iran curve-fit: a=%5.3f, b=%
5.3f, c=%5.3f' % tuple(popt_ir) )
axes.plot(xFit, func(xFit,*popt_us),'m',label = 'US curve-fit: a=%5.3f, b=%5.3
f, c=%5.3f' % tuple(popt_us) )
axes.legend()


########################## Confirm Graph  ###################
plt.figure(2)
figure, axes = plt.subplots(figsize =(20,10))
axes.plot(datetimeTotal,ch_confirmTotal_y, label = "China")
axes.plot(datetimeTotal,it_confirmTotal_y, label = "Italy")
axes.plot(datetimeTotal,ge_confirmTotal_y, label = "Germany")
axes.plot(datetimeTotal,ir_confirmTotal_y, label = "Iran")
axes.plot(datetimeTotal,us_confirmTotal_y, label = "United States")
ax=plt.gca()


plt.xlabel('Timeline')
plt.ylabel('#Confirmed')

plt.xticks(rotation = 90)
plt.xticks(datetimeTotal)

plt.title('Coronavirus in Major Countries: Confirmed Graph')
axes.legend()
#Statement
print("China date list: "+'\n')
print(ch_confirmTotal_y )
print("")
print("Italy date list: "+'\n')
print(it_confirmTotal_y )
print("")
print("Germany date list: "+'\n')
print(ge_confirmTotal_y )
print("")
print("Iran date list: "+'\n')
print(ir_confirmTotal_y )
print("")
print("US date list: "+'\n')
print(us_confirmTotal_y )
print("")
print("China Death Equation: "+str(popt_ch[0])+"x^2 + "+str(popt_ch[1])+"x + "
+str(popt_ch[2])+'\n')
print("Italy Death Equation: "+str(popt_it[0])+"x^2 + "+str(popt_it[1])+"x + "
+str(popt_it[2])+'\n')
print("Germany Death Equation: "+str(popt_ge[0])+"x^2 + "+str(popt_ge[1])+"x +
"+str(popt_ge[2])+'\n')
print("Iran Death Equation: "+str(popt_ir[0])+"x^2 + "+str(popt_ir[1])+"x + "+
str(popt_ir[2])+'\n')
```

```
print("US Death Equation: "+str(popt_us[0])+"x^2 + "+str(popt_us[1])+"x + "+st
r(popt_us[2])+'\n')
```

China date list:

[548, 643, 920, 1406, 2075, 2877, 5509, 6087, 8141, 9802, 11891, 16630, 1971
6, 23707, 27440, 30587, 34110, 36814, 39829, 42354, 44386, 44759, 59895, 6635
8, 68413, 70513, 72434, 74211, 74619, 75077, 75550, 77001, 77022, 77241, 7775
4, 78166, 78600, 78928, 79356, 79932, 80136, 80261, 80386, 80537, 80690, 8077
0, 80823, 80860, 80887, 80921, 80932, 80945, 80977, 81003, 81033, 81058, 8110
2, 81156, 81250, 81305, 81435, 81498, 81591, 81661, 81782, 81897, 81999, 8212
2, 82198]

Italy date list:

[0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 20, 62, 155, 229, 322, 453, 655, 888, 1128, 1694, 2036, 2502,
3089, 3858, 4636, 5883, 7375, 9172, 10149, 12462, 12462, 17660, 21157, 24747,
27980, 31506, 35713, 41035, 47021, 53578, 59138, 63927, 69176, 74386, 80589,
86498, 92472, 97689, 101739]

Germany date list:

[0, 0, 0, 0, 0, 1, 4, 4, 4, 5, 8, 10, 12, 12, 12, 12, 13, 13, 14, 14, 16, 16,
16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 17, 27, 46, 48, 79, 130, 159,
196, 262, 482, 670, 799, 1040, 1176, 1457, 1908, 2078, 3675, 4585, 5795, 727
2, 9257, 12327, 15320, 19848, 22213, 24873, 29056, 32986, 37323, 43938, 5087
1, 57695, 62095, 66885]

Iran date list:

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 2, 5, 18, 28, 43, 61, 95, 139, 245, 388, 593, 978, 1501, 2336, 2922,
3513, 4747, 5823, 6566, 7161, 8042, 9000, 10075, 11364, 12729, 13938, 14991,
16169, 17361, 18407, 19644, 20610, 21638, 23049, 24811, 27017, 29406, 32332,
35408, 38309, 41495]

US date list:

[1, 1, 2, 2, 5, 5, 5, 5, 5, 7, 8, 8, 11, 11, 11, 11, 11, 11, 11, 11, 12, 12,
13, 13, 13, 13, 13, 13, 13, 13, 15, 15, 15, 51, 51, 57, 58, 60, 68, 74, 98, 1
18, 149, 217, 262, 402, 518, 583, 959, 1281, 1663, 2179, 2727, 3499, 4632, 64
21, 7783, 13677, 19100, 25489, 33276, 43847, 53740, 65778, 83836, 101657, 121
478, 140886, 161807]

China Death Equation: -1.3708501494308643x^2 + 2023244.1593479821x + -7465285
18048.6862

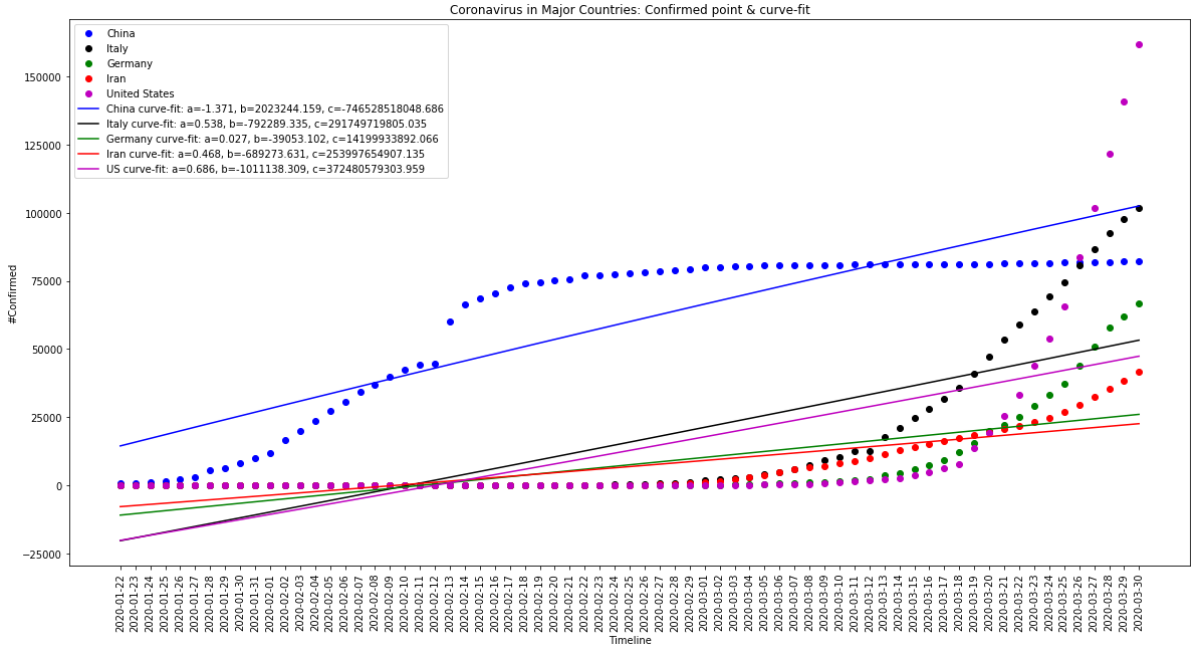
Italy Death Equation: 0.5378936216334474x^2 + -792289.3352965321x + 291749719
805.03467

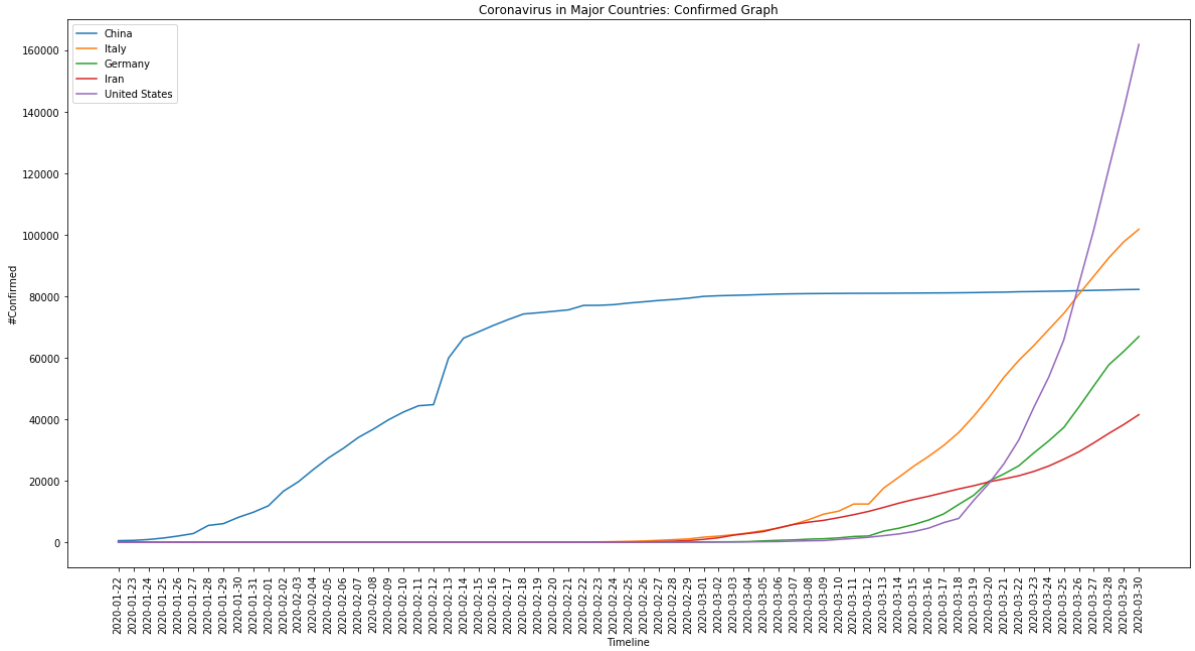
Germany Death Equation: 0.02684607294472084x^2 + -39053.1015415495x + 1419993
3892.065783

Iran Death Equation: 0.467620410379897x^2 + -689273.6308935181x + 25399765490
7.13492

US Death Equation: 0.6862100899407492x^2 + -1011138.3093220154x + 37248057930
3.9586

<Figure size 432x288 with 0 Axes>





Q&A

- What do you deduce about confirmed cases between different countries

China has fastest growing rate among all other countries, but it slow down at the end of Feburary and catch up by US and Italy around 3/25 to 3/26. China has strict stay-home rule that everyone must follows, which is effienct since the cases slow down in a month. However, US has losse stay-home rule as well as not wearing mask recommended for residents by president. It causes people keep gathering without proection.

- What do you deduce about death rate between different countries

Italy is leading the death rate in the world. It has one fifth of the world death. This can due to the medical device shortage that people have to suffer at home instead of hospitalizing with advance ventilator. Iran increase with steady death rate and US is speeding.

- What do you deduce about the recovered cases for each country

China has highest recover rate, and US has the lowest. The other three countries is similar in rate around 3/29.

- Can you share any observation that you have. You have the data! Make the best conclusion out of it!

US seems has the highest chance to get virus, and Italy has the highest possible to dead. To live a longer and happier life, we should fly to China for without any hesitation

In [ ]: