Practica: El modelo epidémico de SIR

Nombre:

Esteban David Rosero Perez

Una descripción matemática simple de la propagación de una enfermedad en una población es el llamado modelo SIR, que divide la población (fija) de N individuos en tres "compartimentos" que pueden variar en función del tiempo, t:

- S(t) son aquellos susceptibles pero aún no infectados con la enfermedad;
- I(t) es el número de individuos infecciosos;
- R(t) son aquellas personas que se han recuperado de la enfermedad y ahora tienen inmunidad.

El modelo SIR describe el cambio en la población de cada uno de estos compartimentos en términos de dos parámetros, beta y gamma.

- Beta describe la tasa de contacto efectiva de la enfermedad: un individuo infectado entra en contacto con beta*N otros individuos por unidad de tiempo (de los cuales la fracción que es susceptible a contraer la enfermedad es S/N).
- Gamma es la tasa de recuperación promedio: es decir, 1/ gamma es el período de tiempo promedio durante el cual una persona infectada puede transmitirlo.

Las ecuaciones diferenciales que describen este modelo fueron derivadas primero por Kermack y McKendrick [Proc. R. Soc. A , 115 , 772 (1927)]:

$$\begin{aligned}\frac{\mathrm{d}S}{\mathrm{d}t} &= -\frac{\beta SI}{N},\\ \frac{\mathrm{d}I}{\mathrm{d}t} &= \frac{\beta SI}{N} - \gamma I,\\ \frac{\mathrm{d}R}{\mathrm{d}t} &= \gamma I.\end{aligned}$$

El siguiente código de Python integra estas ecuaciones para una enfermedad caracterizada por los parámetros beta=0.2, gamma=10 en una población de N=1000 (quizás 'gripe en una escuela) El modelo se inicia con una sola persona infectada el día 0: I(0)=1. Las curvas trazadas de S(t), I(t) y R(t) están diseñadas para verse un poco mejor que los valores predeterminados de Matplotlib.

In [1]:

```
#Importar las librerias.
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

# Total de la poblacion
N = 15000
# Numero Inicial de Infectados
I0 = 1
# Numero de Recuperados
R0 = 0
# Todos los demás, S0, son susceptibles a la infección inicialmente.
S0 = N - I0 - R0
# Tasa de contacto, beta (nivel de repoductividad del virus)
# La tasa de recuperación media, gamma,(1/días) Una persona se recupera en 15 dias.
beta, gamma = 0.4, 1.0/5
# Una cuadrícula de puntos de tiempo (en días)
t = np.linspace(0, 200, 200)


# Las ecuaciones diferenciales del modelo SIR..
def deriv(y, t, N, beta, gamma):
    S, I, R = y
    dSdt = -beta * S * I / N
    dIdt = beta * S * I / N - gamma * I
    dRdt = gamma * I
    return dSdt, dIdt, dRdt

# Vector de condiciones iniciales
y0 = S0, I0, R0
# Integre las ecuaciones SIR en la cuadrícula de tiempo, t. A traves de la funcion odeint()
ret = odeint(deriv, y0, t, args=(N, beta, gamma))
S, I, R = ret.T # Obtenicion de resultados


#  Trace los datos en tres curvas separadas para S (t), I (t) y R (t)
fig = plt.figure(facecolor='w')
ax = fig.add_subplot(111,  axisbelow=True)
ax.plot(t, S, 'b', alpha=0.5, lw=2, label='Sustible de infeccion')
ax.plot(t, I, 'r', alpha=0.5, lw=2, label='Infectados')
ax.plot(t, R, 'g', alpha=0.5, lw=2, label='Recuperados')
ax.set_xlabel('Tiempo en dias')
ax.set_ylabel('Numero de Personas')
ax.set_ylim(0,N*1.2)
ax.yaxis.set_tick_params(length=0)
ax.xaxis.set_tick_params(length=0)
ax.grid(b=True, which='major', c='w', lw=2, ls='-')
legend = ax.legend()
legend.get_frame().set_alpha(0.5)
for spine in ('top', 'right', 'bottom', 'left'):
    ax.spines[spine].set_visible(False)
plt.show()

#Ro = beta/gamma
#print(Ro)
```

```
<Figure size 640x480 with 1 Axes>
```

Generar la prediccion del modelos SIR

Se debe estimar el valor de

- β
- γ

Para ajustar el modelo SIR con los casos confirmados reales (el número de personas infecciosas) del Ecuador.

Para ello deben seguir el siguiente tutorial https://www.lewuathe.com/covid-19-dynamics-with-sir-model.html

In [2]:

```
import pandas as pd
import numpy as np
from datetime import timedelta, datetime
import matplotlib.pyplot as plt
```

In [3]:

```
import scipy.integrate as spi
from scipy.optimize import minimize
from scipy.integrate import solve_ivp
```

Implementar la prediccion del modelo SIR

Para simular el modelo SIR primero debemos declarar las variables iniciales, en este caso vamos a especificar que ya existen 100 contagiados, 0 recuperados y 1000000 de suceptibles. En la variable T especificaremos que la simulacion se va a hacer en 365 dias.

In [4]:

```
# Implementar y explicar la predicion del modelo SIR para el Ecuador
I0=10
R0=0
S0 = 100000
t = 365
y0 = S0,I0,R0
```

In [5]:

```
def load_confirmed(country,START_DATE):
      df = pd.read_csv('time_series_covid19_confirmed_global.csv')
      country_df = df[df['Country/Region'] == country]
      return country_df.iloc[0].loc[START_DATE:]
```

In [6]:

```
def load_recovered(country,START_DATE):
      df = pd.read_csv('time_series_covid19_recovered_global.csv')
      country_df = df[df['Country/Region'] == country]
      return country_df.iloc[0].loc[START_DATE:]
```

Obtenemos la informacion de Ecuador en el dataset, tambien recogemos desde el 13 de abril del 2020 al 13 de abril del 2021. Aqui sacamos la informacion de los confirmados y los recuperados.

In [7]:

```
data_confirmed=load_confirmed('Ecuador','4/13/20');
data_recovered=load_recovered('Ecuador','4/13/20');
```

Procedemos a realizar las funciones, una para los confirmados y recuperados, la otra solo para los confirmados, estas funciones nos ayudan en la optimizacion para poder obtener los mejores resultados de gama y beta.

In [8]:

```
def loss_confirmed_recovered(point, data, recovered):
    size = len(data)
    beta, gamma = point
    def SIR(t, y):
        S = y[0]
        I = y[1]
        R = y[2]
        return [-beta*S*I, beta*S*I-gamma*I, gamma*I]
    solution = solve_ivp(SIR, [0, size], [S0,I0,R0], t_eval=np.arange(0, size, 1), vectorize
d=True)
    l1 = np.sqrt(np.mean((solution.y[1] - data)**2))
    l2 = np.sqrt(np.mean((solution.y[2] - recovered)**2))
    alpha = 0.1
    return alpha * l1 + (1 - alpha) * l2
```

In [9]:

```
def loss_confirmed(point, data):
    size = len(data)
    beta, gamma = point
    def SIR(t, y):
        S = y[0]
        I = y[1]
        R = y[2]
        return [-beta*S*I, beta*S*I-gamma*I, gamma*I]
    solution = solve_ivp(SIR, [0, size], [S0,I0,R0], t_eval=np.arange(0, size, 1), vectorize
d=True)
    return np.sqrt(np.mean((solution.y[1] - data)**2))
```

In [10]:

```
def extend_index(index, new_size):
        values = index.values
        current = datetime.strptime(index[-1], '%m/%d/%y')
        while len(values) < new_size:
            current = current + timedelta(days=1)
            values = np.append(values, datetime.strftime(current, '%m/%d/%y'))
        return values
```

In [11]:

```
def predict(beta, gamma, data):
        predict_range = t
        new_index = extend_index(data.index, predict_range)
        size = len(new_index)
        def SIR(t, y):
            S = y[0]
            I = y[1]
            R = y[2]
            return [-beta*S*I, beta*S*I-gamma*I, gamma*I]
        extended_actual = np.concatenate((data.values, [None] * (size - len(data.values))))
        return new_index, extended_actual, solve_ivp(SIR, [0, size], [S0,I0,R0], t_eval=np.a
range(0, size, 1))
```

Confirmados

Aqui aplicamos las formulas antes creadas en la cual como vamos a utilizar la data de los confirmados, tambien utilizamos la funcion de optimizacion de los confirmados llamada "loss_confirmed", aqui como resultado nos entrega el beta y la gamma, con esta informacion se puede calcular el numero de replicacion del virus para proceder a predecir, utilizando la funcion predict.

In [12]:

```
data = data_confirmed
optimal = minimize(
    loss_confirmed,
    [0.001, 0.001],
    args=(data),
    method='L-BFGS-B',
    bounds=[(0.00000001, 0.4), (0.00000001, 0.4)]
)
beta, gamma = optimal.x
```

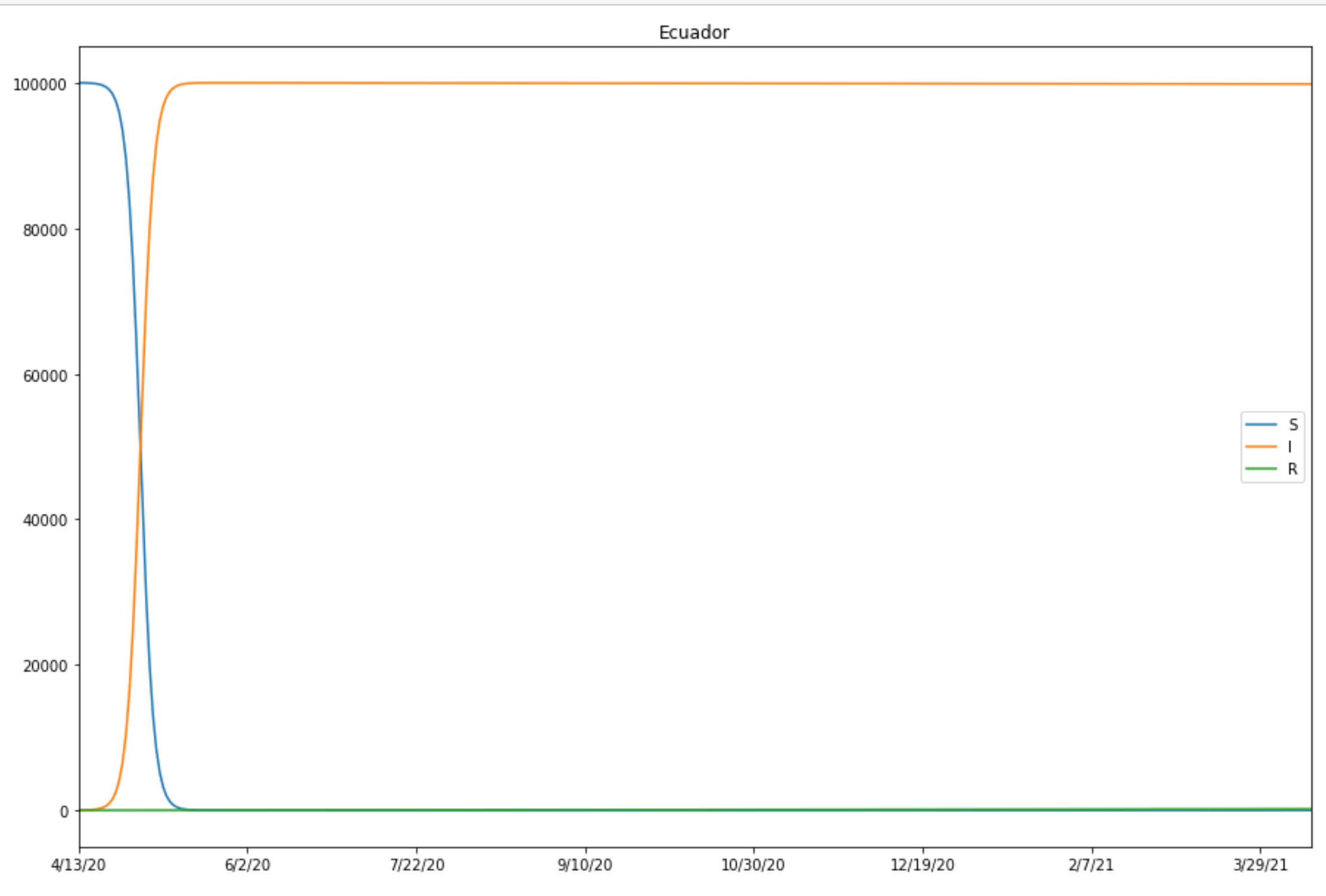
Despues procedemos a realizar la prediccion para los datos de los confirmados, ya obtuvimos la informacion de beta y gamma, con el dataset de los confirmados.

In [13]:

```
new_index, extended_actual, prediction = predict(beta, gamma, data)
```

In [14]:

```
df = pd.DataFrame({
    'S': prediction.y[0],
    'I': prediction.y[1],
    'R': prediction.y[2]
}, index=new_index)
fig, ax = plt.subplots(figsize=(15, 10))
ax.set_title('Ecuador')
df.plot(ax=ax)
fig.savefig(f"ecuador.png")
```



Analisis Confirmados

En la grafica se puede observar que solo se trabajo con los datos de los confirmados,y tambien se visualiza que no existe recuperados y que la curvatura de los suceptibles baja muy rapido, por el cual no se tiene un analisis claro de los resultados.

Confirmados y Recuperados

Aqui aplicamos las formulas antes creadas en la cual como vamos a utilizar la data de los confirmados, tambien utilizamos la funcion de optimizacion de los confirmados y recuperados llamada "loss_confirmed_recovered", aqui como resultado nos entrega el beta y la gamma, con esta informacion se puede calcular el numero de replicacion del virus para proceder a predecir, utilizando la funcion predict.

In [15]:

```
dataConfirmed = data_confirmed
dataRecovered = data_recovered
optimal = minimize(
    loss_confirmed_recovered,
    [0.001, 0.001],
    args=(dataConfirmed, dataRecovered),
    method='L-BFGS-B',
    bounds=[(0.00000001, 0.4), (0.00000001, 0.4)]
)
betaC, gammaC = optimal.x
```

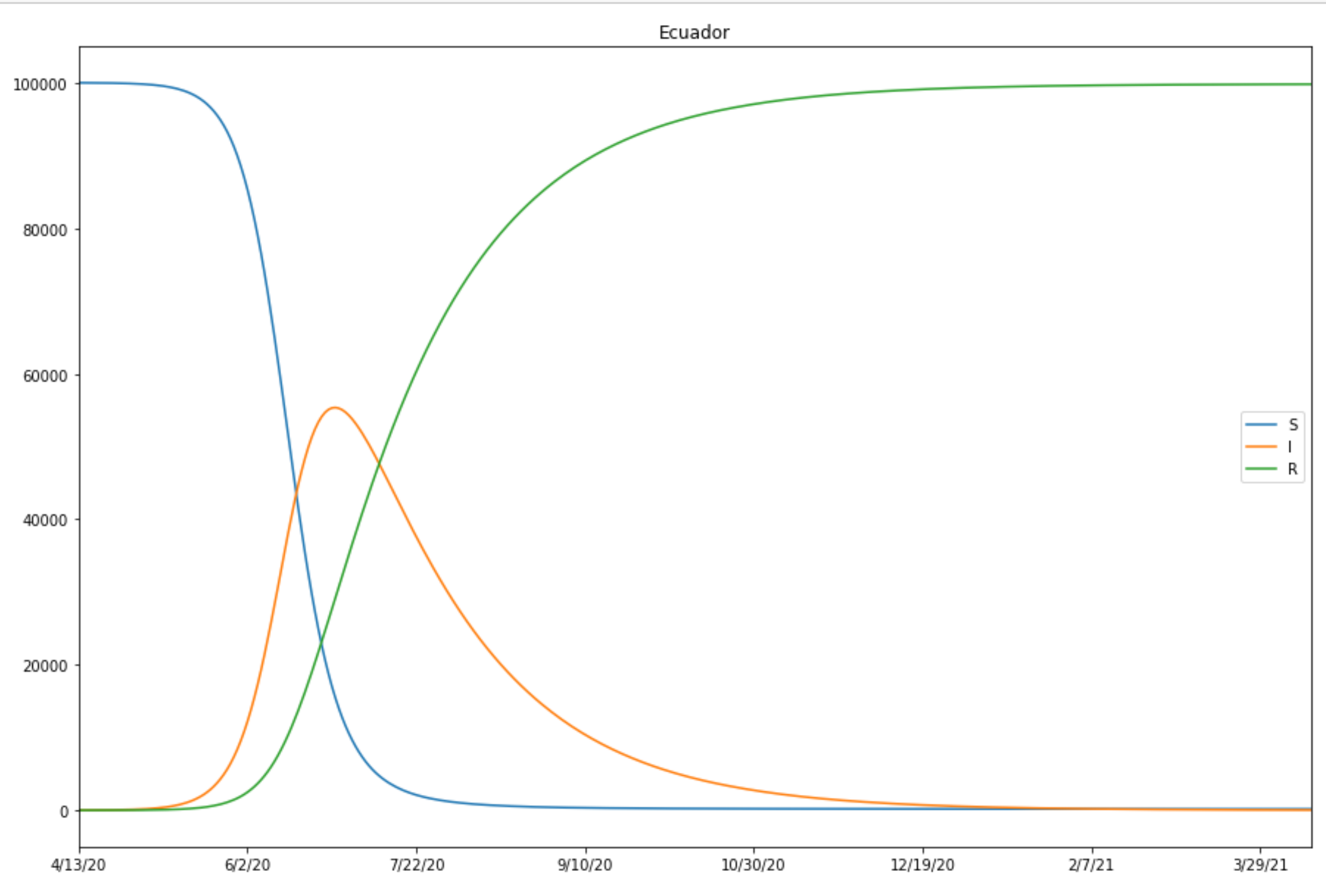
Despues procedemos a realizar la prediccion para los datos de los confirmados y recuperados, ya obtuvimos la informacion de beta y gamma, con el dataset de los confirmados y el dataset de los recuperados.

In [16]:

```
new_indexC, extended_actualC, predictionC = predict(betaC, gammaC, dataConfirmed)
```

In [17]:

```
df = pd.DataFrame({
    'S': predictionC.y[0],
    'I': predictionC.y[1],
    'R': predictionC.y[2]
}, index=new_index)
fig, ax = plt.subplots(figsize=(15, 10))
ax.set_title('Ecuador')
df.plot(ax=ax)
fig.savefig(f"ecuador.png")
```



Analisis Confirmados y Recuperados

En esta grafica se puede obtener una mejor visualizacion de la curvatura de recuperados,suceptibles e infectados, estos resultados se obtuvo por la data de los confirmados y recuperados, en la cual se ve una curvatura mas pronunciada del porcentaje de recuperados, como tambien los suceptibles tienden a caer en picada cuando comienza la curvatura de los recuperados, los infectados tienden a subir a un tope y baja cuando comienza a subir la curvatura de los recuperados, aqui se obtiene un mejor analisis de la visualizacion de la simulacion SIR.

Conclusiones

El modelo SIR nos permite simplificar la modelizacion de las enfermedades infecciosas, como podemos observar se puede detallar los analisis de las personas recuperadas, infectadas y suceptibles, los gobiernos y los sistemas de salud de varios paises han unido fuerzas para la reduccion de contagiados de COVID-19, en la cual apoyan los modelos epidemiologicos al mismo tiempo ellos alimentan la informacion con datos fiables dentro de cada pais.

Referencias:

- https://www.agenciasinc.es/Reportajes/Un-modelo-un-teorema-y-teoria-de-juegos-contra-el-coronavirus
- https://rpubs.com/dsfernandez/422937
- https://towardsdatascience.com/modelling-the-coronavirus-epidemic-spreading-in-a-city-with-python-babd14d82fa2