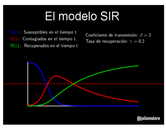
Desarrollo de un modelo SIR para la población de Ecuador

Autor: Bryam David Vega Moreno
Maestro: Diego Quisi
Materia: Simulación
Universidad: Universidad Politécnica Salesiana
Carrera: Ciencias de la computación

Introducción

El modelo SIR es un modelo matemático que permite predecir el comportamiento de una enfermedad infecciosa, a partir de ciertas condiciones iniciales



Este modelo clasifica una población en tres grupos distintos:

- **Susceptible:** Número de personas que son propensas a la enfermedad
- **Infectado:** Número de personas que tienen la enfermedad y que pueden infectar a la gente susceptible
- **Recuperado:** Número de personas que no pueden contraer la enfermedad, porque se han recuperado completamente, o porque son inmunes

Ademas de estos grupos, este modelo depende principalmente de tres parámetros:

- **Tasa de transmisión (β):** Describe que tan rápido se transmite la infección de un individuo a otro
- **Tasa de recuperación (γ):** Describe qué tan rápido un individuo infectado se recupera.
- **Población (N):** Número de población del país.

Para poder resolver este modelo debemos hacer uso de las siguientes formulas:

$$\frac{dS}{dt} = -\beta S \frac{I}{N},$$
$$\frac{dI}{dt} = \beta S \frac{I}{N} - \gamma I,$$
$$\frac{dR}{dt} = \gamma I.$$

Para poder realizar un modelo básico SIR, utilizaremos la librería SciPy el cual contiene un conjunto de paquetes que permiten resolver sistemas de ecuaciones, integrarlas y graficarlas. A continuación mostramos el código

Librerias a importar

Para el análisis de datos

In [173]:
```
import pandas as pd
import numpy as np
from datetime import timedelta, datetime
```

Librerias para visualización de datos

In [475]:
```
import altair as alt
import matplotlib.pyplot as plt
```

Librerias para resolución de ecuaciones diferenciales

In [476]:
```
import scipy.integrate as spi
from scipy.optimize import minimize
from scipy.integrate import solve_ivp
```

Desarrollo del modelo básico SIR

Para empezar a desarrollar el modelo debemos tener en cuenta que necesitaremos crear las variables para nuestro numero de población, número de infectados y número de recuperados, así como también nuestra tasa de transmisión y de recuperación, para ello ponemos los siguientes puntos:

- **N:** Número de población
- **I0:** Número inicial de infectados
- **R0:** Número inicial de recuperados
- **S0:** Número inicial de personas recuperadas que no es mas que N-I0-R0
- **beta:** Tasa de transmisión
- **gamma:** Tasa de recuperación teniendo en cuenta que una persona puede recuperarse en 15 días, es decir 1/15

In [427]:
```
N     = 10000000
I0    = 10000
R0    = 0
S0    = N - I0 - R0
beta  = 0.3
gamma = 1/15
t     = np.linspace(0, 365, 365)
```

Definido nuestras variables, procedemos a realizar las fórmulas para poder resolver nuestro problema de ecuaciones diferenciales, para ello creamos un método denominado **deriv** que contiene las formulas expuestas en la imagen

In [428]:
```
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
```

Después de haber realizado las funciones para resolver nuestra ecuación diferencial y tener los valores iniciales, procedemo a resolver el modelo con las condicionales iniciales, con ello vamos a obtener los valores para 365 como estaba inicializado nuestra variable **t**

In [429]:
```
y0 = S0, I0, R0
ret = spi.odeint(deriv, y0, t, args=(N, beta, gamma))
S, I, R = ret.T
```

Procedemos a crear un pequeño dataframe que nos explica como va variado el SIR a medida que avanza el tiempo, con la finalidad de ver mejor los datos, cabe recalcar que dichos estan normalizados de 0 a 1 ya que estan divididos para el número de población

In [430]:
```
df = pd.DataFrame({'time':t.astype('int'),'susceptible':S/N,'infected':I/N,'recovered':R/N})
df.head(5)
```

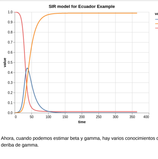
Out[430]:

	time	susceptible	infected	recovered
0	0	0.999375	0.000625	0.000000
1	1	0.999163	0.000790	0.000047
2	2	0.998896	0.000997	0.000107
3	3	0.998558	0.001259	0.000182
4	4	0.998132	0.001591	0.000277

Por ultimo grafícanos los datos utilizando la libreria **altair** la cual nos permite hacer la gráfica un poco más interáctive y con un mejor diseño. Con esta gráfica podemos darnos cuenta que a medida que se aumenta el número de infectados, el número de recuperados también aumenta por lo que por consiguiente, el número de personas susceptibles también tiende a bajar a medida que aumentan los recuperados y bajan las infecciones.

In [431]:
```
alt.Chart(df.melt('time')).mark_line().encode(
    x='time',
    y=alt.Y('value'),
    color='variable'
).properties(
    title='SIR model for Ecuador Example'
).interactive()
```

Out[431]:



Ahora, cuando podemos estimar beta y gamma, hay varios conocimientos derivados de el. Si D, es el promedio de días para recuperarse de infecciones , se deriba de gamma.

$$D = \frac{1}{\gamma}$$

Ademas de ello, podemos estimar la naturaleza de la enfermedad en terminos del poder de la infección.

$$R_0 = \frac{\beta}{\gamma}$$

Se llama número de reproducción básico **R0**. Es el número promedio de personas infectadas de otra persona. Si es alto, la probabilidad de pandemia también es mayor. Si bien antes tenamos como variables fijas beta y gamma, ahora lo que procederemos a hacer es estimar β y γ para ajustar el modelo SIR a los casos confirmados reales (el número de personas infecciosas). A continuación vamos a proceder con el código y la explicación

Simulación del modelo SIR para COVID-19

Lo primero que procedemos a hacer es crear nuestras variables I0,R0,S0 con la finalad de tener datos iniciales, estos datos pueden ir variando para ir simulando los diferentes escenarios. Para los valores tomados, tendremos en cuenta que existían 10 millones susceptibles, mientras que habían 10 infectados y 0 recuperados. De la misma manera tomaremos 365 días a partir del día del primer caso. A continuación mostramos las variables

In [440]:
```
I0=10
R0=0
S0 = 100000
t = 365
y0 = S0,I0,R0
```

Con las variables y condiciones iniciales procedemos a leer los datos que necesitamos para poder realizar nuestro modelo SIR, en este caso utilizamos los dataset de numero de confirmados y numero de recuperados con la finalidad de que se pueda obtener una mejor curva para los recovered puesto que después de un cuarto de experimentos, existía un problema en el número de recuperados. El código de lectura y obtención de datos es el siguiente:

In [440]:
```
def filter_country(df,country,start_date):
    country_df = df[df['Country/Region'] == country]
    return country_df.iloc[0].loc[start_date:]

def load_data(path_confirmed,path_recovered,country,date):
    df_confirmed = filter_country(pd.read_csv(path_confirmed),country,date)
    df_recovered = filter_country(pd.read_csv(path_recovered),country,date)
    return df_confirmed,df_recovered

data_confirmed,data_recovered=load_data('./in/time_series_covid19_confirmed_global.csv','./in/time_series_covid19_recovered_global.csv','Ecuador','3/1/20')
```

Ahora procedemos a realizar una función con el fin de optimizar los valores de beta y gamma con el fin de mejorar dichos valores con respecto a la data que tenemos, para ello utilizamos los datos de confirmed y recovered con el fin de trabajar con los mismos. A continuación mostramos la función que nos permite optimizar dicho proceso:

In [452]:
```
def loss(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), vectorized=True)
    l1 = np.sqrt(np.mean((solution.y[1] - data)**2))
    l2 = np.sqrt(np.mean((solution.y[2] - recovered)**2))
    # Put more emphasis on recovered people
    alpha = 0.1
    return alpha * l1 + (1 - alpha) * l2
```

In [453]:
```
optimal = minimize(
            loss,
            [0.001, 0.001],
            args=(data_confirmed,data_recovered),
            method='L-BFGS-B',
            bounds=[(0.00000001, 0.4), (0.00000001, 0.4)])
```

In [454]:
```
beta,gamma = optimal.x
```

Despues de optimizar dichos valores podemos darnos cuenta que los valores de beta y gamma son los siguientes

In [461]:
```
print("valor gamma: {}".format(gamma))
print("valor beta: {}".format(beta))
```

```
valor gamma: 0.021909976604310948
valor beta: 1.2249257546538603e-06
```

Con dichos valores podemos obtener el valor de R0, a patir de la formula que vimos anteriormente, que no era más que la división entre beta y gamma. Con ello tenemos el siguiente resultado

In [473]:
```
R_0= beta/gamma
print("Numero de reproducción R_0: {}".format(R_0))
```

```
Numero de reproducción R_0: 5.590771464645256e-05
```

Una vez obtenido el valor de R_0 el cual es un indicador para ver el número de reproducción del virus, procedemos a realizar la resolución del modelo SIR, no necesitamos de R_0 en este momento, simplemente es un indicador que nos permite saber como se reproduce el virus en nuestra ciudad. Ahora con ello en mente mostramos el código en donde resolveremos el modelo SIR

In [455]:
```
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

def predict(beta, gamma, data):
    """
    Predict how the number of people in each compartment can be changed through time toward the future.
    The model is formulated with the given beta and gamma.
    """
    predict_range = 150
    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.arange(0, size, 1))
```

In [466]:
```
new_index, extended_actual, prediction = predict(beta, gamma, data)
```

Despues de realizar las predicciones para el modelo SIR procedemos a armar un dataframe para poder visualizar nuestro modelo terminado

In [457]:
```
df = pd.DataFrame(
        {'date':[i for i in range(len(new_index))],
         'susceptible': prediction.y[0],
         'infected': prediction.y[1],
         'recovered': prediction.y[2]})
```

In [458]:
```
df
```

Out[458]:

	date	susceptible	infected	recovered
0	0	100000.000000	10.000000	0.000000
1	1	99988.711346	11.090154	0.198498
2	2	99997.286259	12.229338	0.485403
3	3	99995.715512	13.522228	0.767260
4	4	99993.980185	14.952995	1.076820
...	...	...	...	...
360	360	387.802174	344.493424	99277.612402
361	361	387.769296	337.152823	99295.077880
362	362	387.619913	330.093910	99282.365176
363	363	387.455947	323.306337	99289.537717
364	364	387.304320	316.156827	99306.538853

365 rows × 4 columns

Con este conjunto de datos, podemos obtener la gráfica que se presenta a continuación, nos podemos dar cuenta que el número de infectados llego a un pique de 50000 personas infectadas , sin embargo a medida que los infectados han ido bajando los recuperados han aumentado y por ende los susceptibles bajando por lo que existe lógica en lo planteado, sin embargo, se deben realizar más experimentos con el fin de que la curva de infectados se aplane mejor, sin embargo, no podemos hacer tantas pruebas debido a que la optimización de beta y gamma es costoso computacionalmente.

In [459]:
```
alt.Chart(df.melt('date')).mark_line().encode(
    x='date',
    y=alt.Y('value'),
    color='variable'
).properties(
    title='SIR model for Ecuador Example'
).interactive()
```

Out[459]:

In [ ]:
```

```