El modelo epidémico de SIR

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.

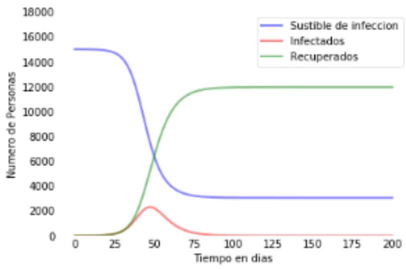
- Beta describe la tasa de contacto efectiva de la enfermedad: un individuo infectado entra en contacto con beta*N otros individuos por unidad de tiempo (de los cuales la fracción que es susceptible a contraer la enfermedad es S/N).
- Gamma es la tasa de recuperación promedio: es decir, 1/ gamma es el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

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\frac{\beta S I}{N},$$
$$\frac{\mathrm{d}I}{\mathrm{d}t} = \frac{\beta S I}{N} - \gamma I,$$
$$\frac{\mathrm{d}R}{\mathrm{d}t} = \gamma I.$$

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 [2]:
```
#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)
```



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.

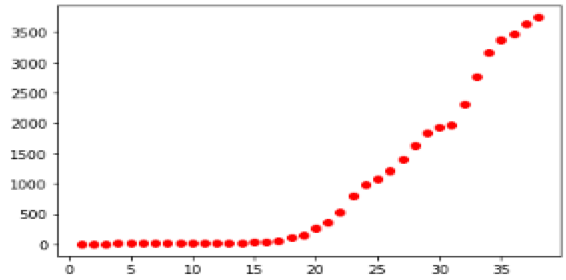
Para ello deben seguir el siguiente tutorial https://www.lewuathe.com/covid-19-dynamics-with-sir-model.html

In [35]:

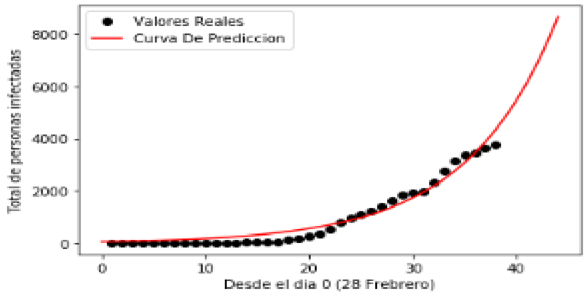
```
# Implementar y explicar la predicion del modelo SIR para el Ecuador

#datos actualizados desde la pagina wikipedia
#https://es.wikipedia.org/wiki/Pandemia_de_enfermedad_por_coronavirus_de_2020_en_Ecuador
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
from scipy.optimize import minimize
import matplotlib.pyplot as plt
from pylab import *
from scipy.stats import expon
matrizdatos=[(1,1),(2,6),(3,6),(4,11),(5,14),(6,14),(7,14),(8,14),(9,15),(10,15),(11,17),(12,17),(13,17),(14,23),(15,28)
,(16,37),(17,58),(18,111),(19,155),(20,260),(21,367),(22,532),(23,789),(24,981),(25,1082),(26,1211),(27,1403),(28,1627)
,(29,1835),(30,1924),(31,1966),(32,2302),(33,2758),(34,3163),(35,3368),(36,3465),(37,3646),(38,3747)]

matrizrecuperdos=[(1,0),(2,0),(3,0),(4,0),(5,0),(6,0),(7,0),(8,0),(9,0),(10,0),(11,0),(12,0),(13,0),(14,0),(15,0)
,(16,0),(17,0),(18,0),(19,0),(20,0),(21,0),(22,3),(23,3),(24,3),(25,3),(26,3),(27,3),(28,3)
,(29,3),(30,3),(31,3),(32,54),(33,58),(34,65),(35,65),(36,100),(37,100),(38,100)]

matrizmuertos=[(1,0),(2,0),(3,0),(4,0),(5,0),(6,0),(7,0),(8,0),(9,0),(10,0),(11,0),(12,0),(13,0),(14,0),(15,0)
,(16,2),(17,2),(18,2),(19,2),(20,2),(21,5),(22,7),(23,14),(24,18),(25,27),(26,28),(27,34),(28,36)
,(29,48),(30,58),(31,60),(32,75),(33,93),(34,120),(35,145),(36,172),(37,180),(38,191)]

x_real= np.zeros(38)
y_real= np.zeros(38)
r_real= np.zeros(38)
d_real= np.zeros(38)

for i in range(38):
    x_real[i]=matrizdatos[i][0]
    y_real[i]=matrizdatos[i][1]
    r_real[i]=matrizrecuperdos[i][1]
    d_real[i]=matrizmuertos[i][1]

plt.scatter(x_real,y_real,label="Datos Reales",color="red")
show()

from scipy.optimize import curve_fit
def func(x, a, b):
    return a * np.exp(-b * x-0)
r=curve_fit(func, x_real, y_real)
print(r)
x_prec=np.array(range(0,45))
r1=func(42,r[0][0],r[0][1])
print(r1)
plt.figure()
plt.plot(x_real, y_real, 'ko', label="Valores Reales")
plt.plot(x_prec,[func(i,r[0][0],r[0][1]) for i in x_prec], 'r-', label="Curva De Prediccion")
plt.legend()
plt.xlabel("Desde el dia 0 (28 Frebrero)")
plt.ylabel("Total de personas infectadas")
plt.show()
```

```python
# 1. Implementar solo teniendo en cuenta los casos confirmados
def loss(point, data, recovered, s_0, i_0, r_0):
    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], [s_0,i_0,r_0], 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))
    alpha = 0.1
    return alpha * l1 + (1 - alpha) * l2
data=(y_real - r_real)

# Total de la poblacion
N = 10000
# Numero Inicial de Infectados
I0 =1
# Numero de Recuperados
R0 = 191
# 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.
optimal = minimize(loss, [4,4], args=(data, r_real, S0, I0, R0), method='L-BFGS-B', bounds=[(0.00000001, 0.4), (0.00000001, 0.
4)])
print(optimal)
beta, gamma = optimal.x
print(beta)
print(gamma)
#beta, gamma = 0.4, 1.0/5
# Una cuadrícula de puntos de tiempo (en días)
t = np.linspace(0, 75, 75)


# 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(x_real,y_real , '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()
```

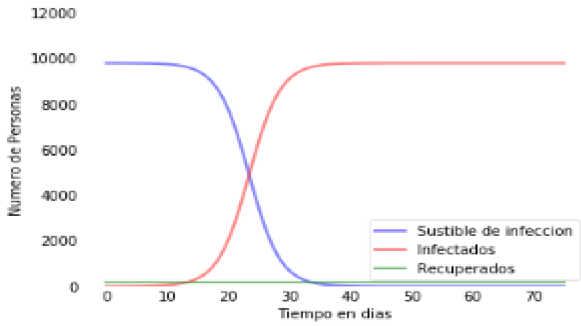
```python
# 2. Implementar teniendo en cuenta los casos confirmados y recuperados.
# Total de la poblacion
data=(y_real)
N = 10000
# Numero Inicial de Infectados
I0 = 1
# Numero de Recuperados
R0 = 191
# Todos los demás, S0, son susceptibles a la infección inicialmente.
S0 = N - I0 - R0
# Tasa de contacto, beta (nivel de repoductividad del virus)
# La tasa de recuperación media, gamma,(1/días) Una persona se recupera en 15 dias.
optimal = minimize(loss, [4,4], args=(data, y_real, S0, I0, R0), method='L-BFGS-B', bounds=[(0.00000001, 0.4), (0.00000001, 0.
4)])
print(optimal)
beta, gamma = optimal.x
print(beta)
print(gamma)
#beta, gamma = 0.4, 1.0/5
# Una cuadrícula de puntos de tiempo (en días)
t = np.linspace(0, 75, 75)

# 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()
```



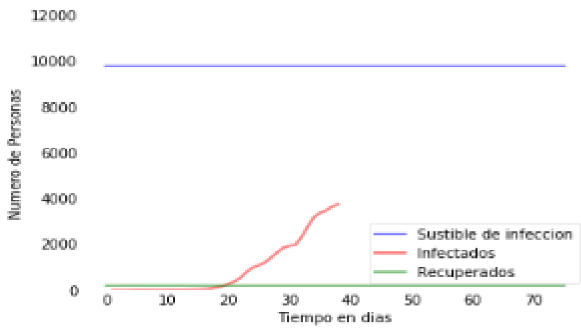

```
(array([55.22188039, -0.11486961]), array([[1.17279857e+02, 6.1248
       [6.12481128e-02, 3.24975488e-05]]))
6876.62357323644
```




```
      fun: 1039.718605383833
 hess_inv: <2x2 LbfgsInvHessProduct with dtype=float64>
      jac: array([9.54969437e-04, 1.32364243e+05])
  message: b'CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH'
     nfev: 27
      nit: 2
   status: 0
  success: True
        x: array([3.99962516e-01, 1.00000000e-08])
0.3999625161512853
1e-08
```




```
      fun: 300.4479276521678
 hess_inv: <2x2 LbfgsInvHessProduct with dtype=float64>
      jac: array([-2.17430090e+02,  1.67801772e-02])
  message: b'CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH'
     nfev: 276
      nit: 38
   status: 0
  success: True
        x: array([4.21191144e-05, 1.22813897e-01])
4.211911436496182e-05
0.12281389670389486
```



Calculos de incidencia

Para obtener metricas de incidencia se debe calcular la tasa de prevalencia, incidencia y la relacion, para esto leer y obtener estos datos con la ultima lectura.

https://www.paho.org/hq/index.php?option=com_content&view=article&id=14402:indicadores-de-salud-aspectos-conceptuales-y-operativos-seccion-2&catid=9894&limitstart=2&Itemid=101&lang=es

In [1]:

```python
# Implementar

def incidencia(nuevos,riesgo,n):
    return (nuevos/riesgo)*pow(10,n)
def prevalencia(casos,poblacion,n):
    return (casos/poblacion)*pow(10,n)
def relacion(incidencia,tiempo):
    return incidencia*tiempo
nuevo=3747-3646
riesgo=16000-3747
incidencia1=incidencia(nuevo,riesgo,3)
prevalencia2 = prevalencia(3747,14483499,3)
relacion3=relacion(incidencia1,28)
print(incidencia1)
print(prevalencia2)
print(relacion3)
```

```
8.242879294866563
0.2587082030385061
230.80062025626378
```

Analisis

El codigo es muy sebcillo denenter pero a al hora de implementar beta y gamma es un poco dificil de determinar ya que el codigo codigo no se especifica muy bien el modelo nos muestra como que el incremento de la poblacion contagiada va ir incrementando de manera muy exponencial.

Conclusiones

Como conclucion podemos decir que el modelo SIR nos muestra la infecion llegara de maneras criticas a nivel, nacional si se sigue llevando de esta manera la infeccion.

Opinion

los grandes creciiemiunto del covid en el pais solo traen un mal presajio para el pais si el crecimiento dara un porcentaje de perdidas humanas y economicas seran.

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