Edgar Gerardo Alarcón González

9 de diciembre de 2020

1. Proyecto 2 - Coronavirus

Este proyecto consiste en estimar un modelo SIR con datos de un país sobre el COVID-19.

[1]:
```
# Carga de librerías
import pandas as pd
import numpy as np
from scipy import integrate
from sklearn.metrics import mean_squared_error, mean_squared_log_error,␣
 ↪mean_absolute_error, median_absolute_error
from scipy.integrate import odeint
from scipy.optimize import differential_evolution, minimize
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
# Toolbox especial
import PDEparams as pde
# https://github.com/systemsmedicine/PDE_params
```

1.0.1. Carga de Datos (Rusia)

De acuerdo a Google, la población de Rusia es

$$N = 144{,}5 \text{ Millones}$$

[3]:
```
# Vamos a considerar la población mundial de Rusia
N = 144500000
# Estos son todos los datos con los que contamos
data = pd.read_csv('Russia_COVID.csv')
data.head()
```

[3]:

	Fecha	Tiempo	Infectados_Acumulados	Recuperados_Acumulados	\
0	1/22/20	1	0	0	
1	1/23/20	2	0	0	
2	1/24/20	3	0	0	
3	1/25/20	4	0	0	

```
4  1/26/20       5                       0                       0

   Muertes_Acumulados  Infectados  Recuperados  Muertos
0                   0           0            0        0
1                   0           0            0        0
2                   0           0            0        0
3                   0           0            0        0
4                   0           0            0        0
```

1.1. Defininimos el modelo SIR

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

Suceptible -> Infectado -> Recuperado

$$\beta = \text{Tasa de contacto} \times \text{Probabilidad de Transmisión}$$
$$\gamma = \text{Tasa de Recuperación}$$

En particular, nosotros vamos a considerar:

- $S :=$ Población suceptible.
- $I :=$ Población Infectada.
- $R :=$ Población Recuperada

Nota: Este modelo asumirá que R es la población que no que fue infectada pero ya no puede volver a infectar. Por efectos de cómo tenemos los datos, nosotros vamos a considerar que estos serán los muertos más los recuperados.

[4]:
```
# En realidad nosotros vamos a ocupar lo siguiente
datos = pd.DataFrame({
            # Tiempo
            't': data.Tiempo,
            # Suceptibles
            'S': N - np.cumsum(data.Infectados) -  (np.cumsum(data.Recuperados)␣
 ↪+ np.cumsum(data.Muertos)),
            # Infectados
            'I': data.Infectados,
            # Recuperados (este modelo asume que los recuperados son las␣
 ↪personas que ya no pueden contagiar)
            'R': data.Muertos + data.Recuperados
})
```

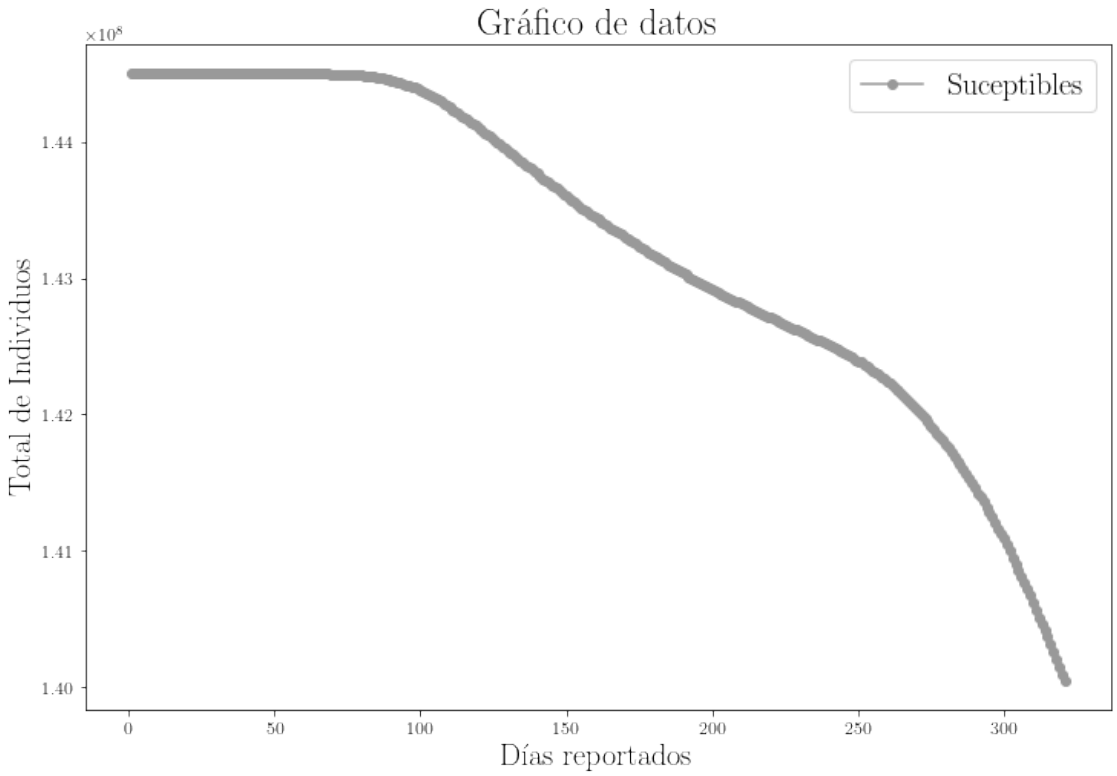
```
datos.describe()
```

[4]:

	t	S	I	R
count	321.000000	3.210000e+02	321.000000	321.000000
mean	161.000000	1.432630e+08	7685.236760	6176.059190
std	92.808944	1.205129e+06	7110.665979	6528.110415
min	1.000000	1.400505e+08	0.000000	0.000000
25%	81.000000	1.424955e+08	1786.000000	197.000000
50%	161.000000	1.434318e+08	6096.000000	5478.000000
75%	241.000000	1.444853e+08	9623.000000	8776.000000
max	321.000000	1.445000e+08	28701.000000	29627.000000

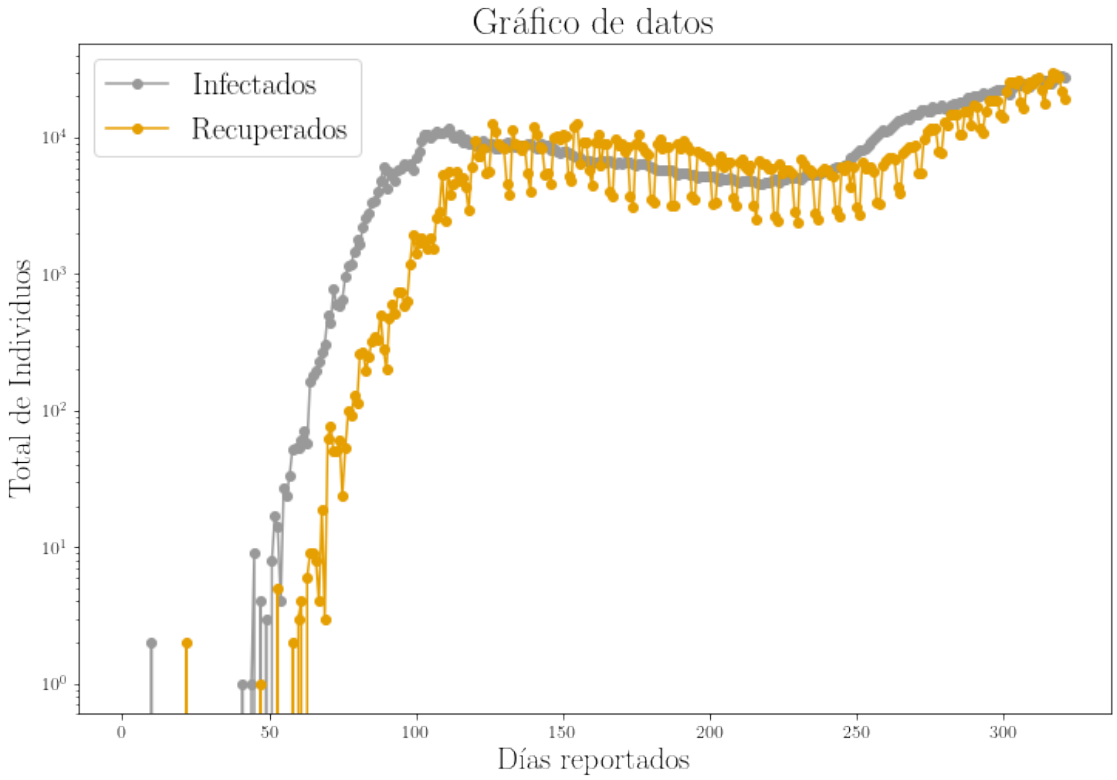
[5]:
```
plt.figure(figsize=(12, 8))
plt.title("Gráfico de datos")
plt.plot(datos.t, datos.S, '-o', label="Suceptibles")
plt.ylabel("Total de Individuos")
plt.xlabel("Días reportados")
plt.legend()
```

[5]: <matplotlib.legend.Legend at 0x1e869101160>



[6]:
```
plt.figure(figsize=(12, 8))
plt.title("Gráfico de datos")
plt.plot(datos.t, datos.I, '-o', label="Infectados")
plt.plot(datos.t, datos.R, '-o', label="Recuperados")
plt.yscale('log')
plt.ylabel("Total de Individuos")
plt.xlabel("Días reportados")
plt.legend()
```

[6]:
```
<matplotlib.legend.Legend at 0x1e8693bb8b0>
```



Definimos el sistema de ecuaciones diferenciales a resolver

[7]:
```
def sistema_SIR(z, t, beta, gamma):

    S, I, R = z

    # dS/dt
    dSdt = - beta * S * I / N
    # dI/dt
    dIdt = beta * S * I / N - gamma * I
    # dR/dt
    dRdt = gamma * I
```

```
    return [dSdt,dIdt,dRdt]
```

Vamos a definir los valores iniciales de nuestros datos. Para esto, consideramos un escenario donde al principio teníamos un infectado, ningún recuperado y una población suceptible de $N - 1$.

[8]:
```
# Suceptible inicial
def S0():
    return N-1
# Infectado inicial
def I0():
    return 1
# Recuperado inicial
def R0():
    return 0
```

Con base en todo lo anterior, ejectuamos nuestro modelo

[9]:
```
modelo_SIR = pde.PDEmodel(datos, sistema_SIR, [S0,I0,R0], bounds=[(0, 10),␣
↪(0,10)],
                          param_names=[r'$\beta$', r'$\gamma$'], nvars=3, ndims=0,␣
↪nreplicates=1, obsidx=None, outfunc=None)
```

[10]:
```
# Mostramos la condición Inicial
modelo_SIR.initial_condition
```

[10]:
```
array([144499999,         1,         0])
```

Encontrando los mejores parámetros

[11]:
```
%%time
modelo_SIR.fit()
```

```
    $\beta$  $\gamma$
0  8.813566  8.768429
Wall time: 2.34 s
```

[12]:
```
modelo_SIR.best_params
```

[12]:
```
    $\beta$  $\gamma$
0  8.813566  8.768429
```

[13]:
```
modelo_SIR.best_error
```

[13]:
```
522391405693.6701
```

Likelihood profiles

[14]:
```
%%time
modelo_SIR.likelihood_profiles(npoints=50)
```

```
HBox(children=(FloatProgress(value=0.0, description='parameters', max=2.0, style=ProgressStyle(d

HBox(children=(FloatProgress(value=0.0, description='values within parameters', max=50.0, style=

HBox(children=(FloatProgress(value=0.0, description='values within parameters', max=50.0, style=

Wall time: 3min
```

[15]:
```
modelo_SIR.result_profiles
```

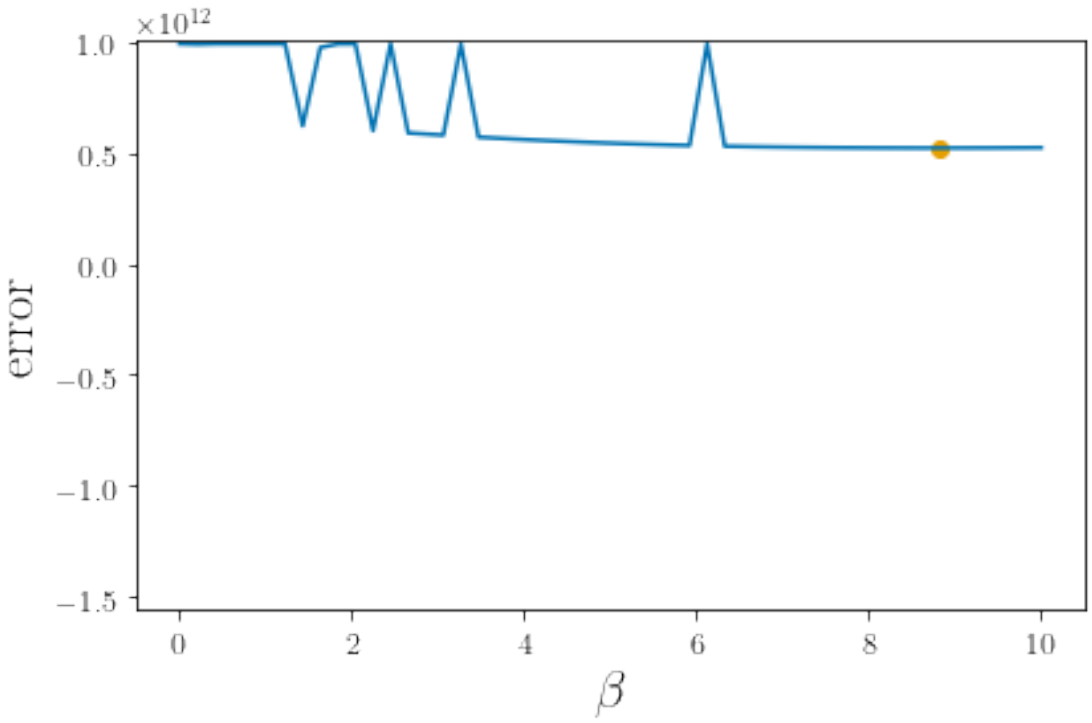
[15]:

	parameter	value	error
0	β	0.000000	9.927122e+11
1	β	0.204082	9.907603e+11
2	β	0.408163	9.926836e+11
3	β	0.612245	9.926905e+11
4	β	0.816327	9.927055e+11
..	...	...	...
95	γ	9.183673	5.226385e+11
96	γ	9.387755	5.228816e+11
97	γ	9.591837	9.926108e+11
98	γ	9.795918	9.926907e+11
99	γ	10.000000	9.926666e+11

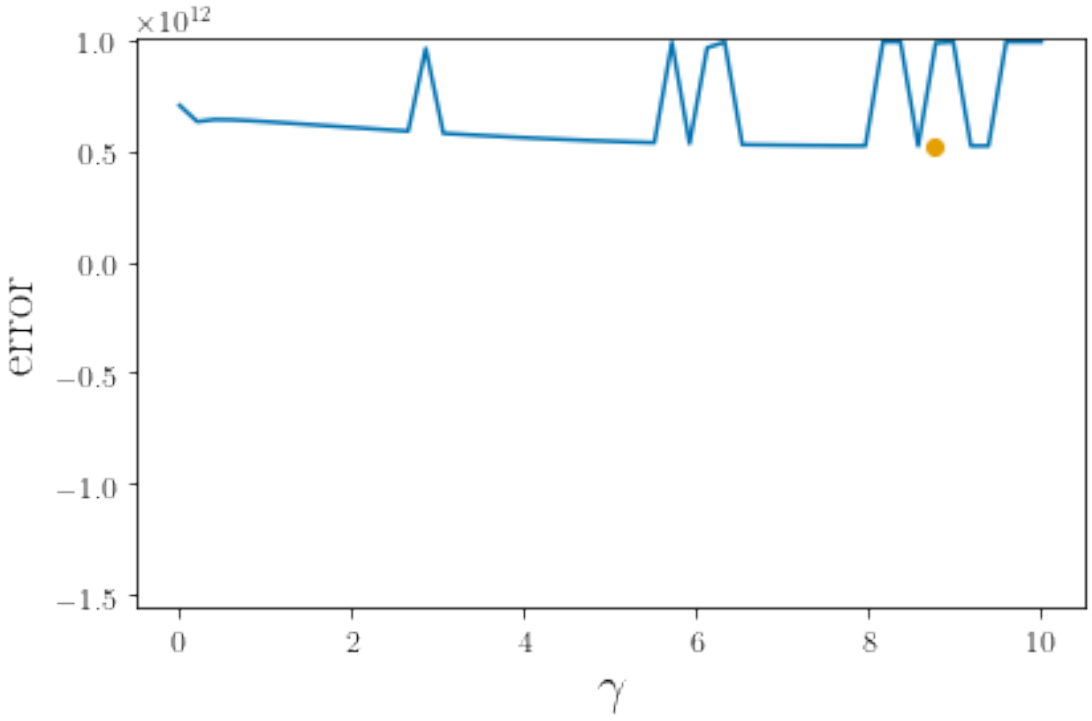
[100 rows x 3 columns]

[16]:
```
modelo_SIR.plot_profiles()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with
a single row if you intend to specify the same RGB or RGBA value for all points.
```




```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with
a single row if you intend to specify the same RGB or RGBA value for all points.
```



1.2. Defininimos el modelo SIRD

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

$\beta =$ Tasa de contacto $\times$ Probabilidad de Transmisión
$\gamma =$ Tasa de Recuperación
$\mu =$ Tasa de Mortalidad

En particular, nosotros vamos a considerar:

- $S :=$ Población suceptible.
- $I :=$ Población Infectada.
- $R :=$ Población Recuperada
- $D :=$ Población Muerta

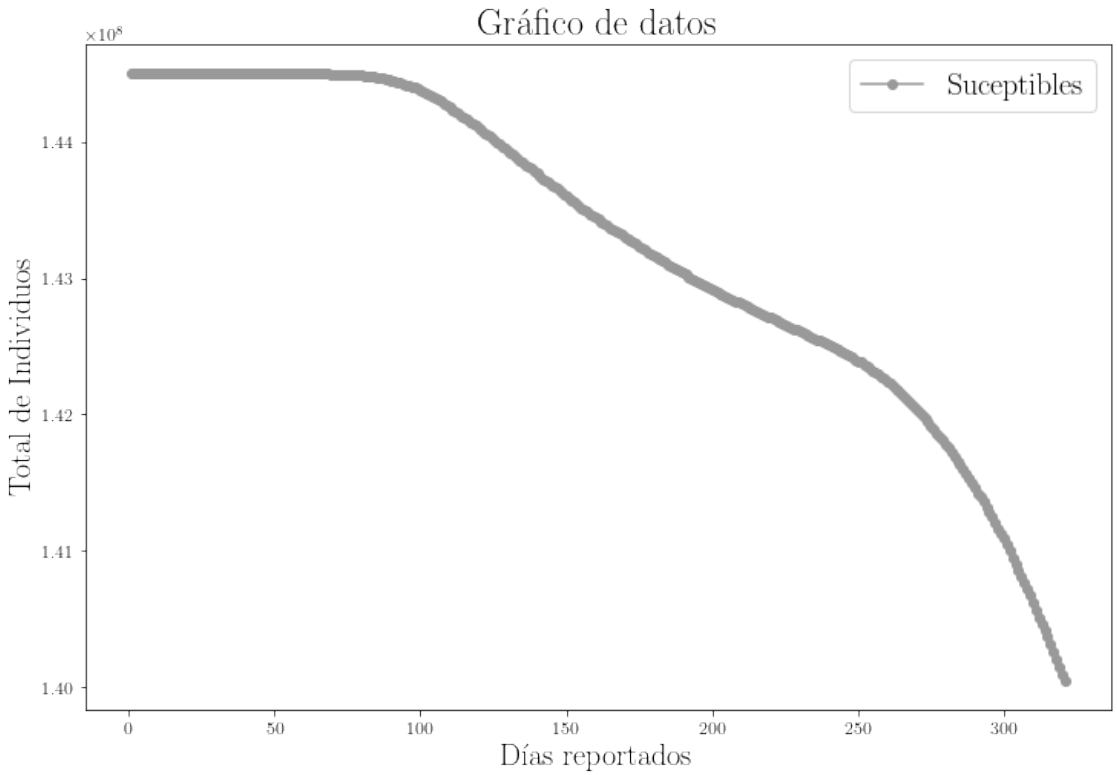
[17]:
```
# En realidad nosotros vamos a ocupar lo siguiente
datos = pd.DataFrame({
            # Tiempo
            't': data.Tiempo,
            # Suceptibles
            'S': N - np.cumsum(data.Infectados) - (np.cumsum(data.Recuperados) +␣
 ↪np.cumsum(data.Muertos)),
            # Infectados
            'I': data.Infectados,
            # Recuperados
            'R': data.Recuperados,
            # Muertos
            'D': data.Muertos
})
datos.describe()
```

[17]:

	t	S	I	R	D
count	321.000000	3.210000e+02	321.000000	321.000000	321.000000
mean	161.000000	1.432630e+08	7685.236760	6041.722741	134.336449
std	92.808944	1.205129e+06	7110.665979	6401.488399	130.640868
min	1.000000	1.400505e+08	0.000000	0.000000	0.000000
25%	81.000000	1.424955e+08	1786.000000	155.000000	18.000000
50%	161.000000	1.434318e+08	6096.000000	5352.000000	114.000000
75%	241.000000	1.444853e+08	9623.000000	8499.000000	175.000000
max	321.000000	1.445000e+08	28701.000000	29084.000000	580.000000

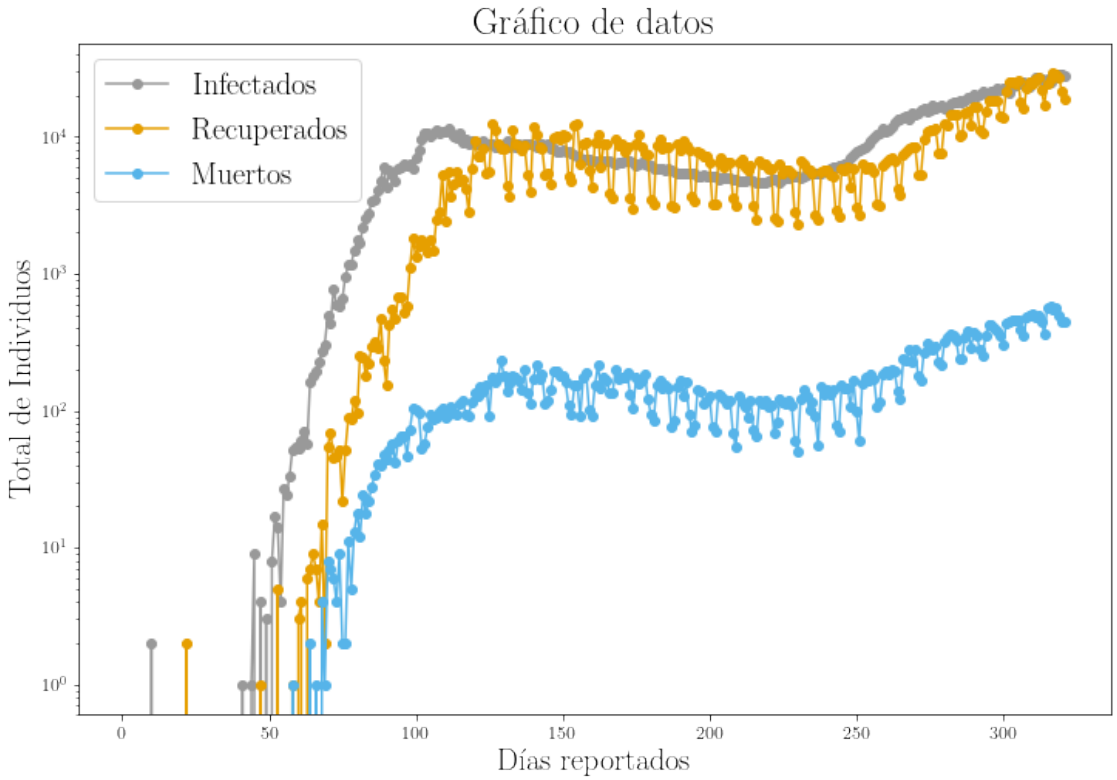
[18]:
```
plt.figure(figsize=(12, 8))
plt.title("Gráfico de datos")
plt.plot(datos.t, datos.S, '-o', label="Suceptibles")
plt.ylabel("Total de Individuos")
plt.xlabel("Días reportados")
plt.legend()
```

[18]:
```
<matplotlib.legend.Legend at 0x1e869aa8a90>
```



[19]:
```
plt.figure(figsize=(12, 8))
plt.title("Gráfico de datos")
plt.plot(datos.t, datos.I, '-o', label="Infectados")
plt.plot(datos.t, datos.R, '-o', label="Recuperados")
plt.plot(datos.t, datos.D, '-o', label="Muertos")
plt.yscale('log')
plt.ylabel("Total de Individuos")
plt.xlabel("Días reportados")
plt.legend()
```

[19]:
```
<matplotlib.legend.Legend at 0x1e869ce0df0>
```



Definimos el sistema de ecuaciones diferenciales a resolver

[20]:
```
def sistema_SIR(z, t, beta, gamma,mu):

    S, I, R, D = z

    # dS/dt
    dSdt = - beta * S * I / N
    # dI/dt
    dIdt = beta * S * I / N - gamma * I - mu * D
    # dR/dt
    dRdt = gamma * I
    # dD/dt
    dDdt = mu * D

    return [dSdt,dIdt,dRdt,dDdt]
```

Vamos a definir los valores iniciales de nuestros datos. Para esto, consideramos un escenario donde al principio teníamos un infectado, ningún recuperado, ningún muerto y una población susceptible de $N - 1$.

[21]:
```
# Suceptible inicial
def S0():
```

```
    return N-1
# Infectado inicial
def I0():
    return 1
# Recuperado inicial
def R0():
    return 0
# Muerto inicial
def D0():
    return 0
```

Con base en todo lo anterior, ejectuamos nuestro modelo

[22]:
```
modelo_SIRD = pde.PDEmodel(datos, sistema_SIR, [S0,I0,R0,D0], bounds=[(0, 10),␣
↪(0,10), (0,10)],
                           param_names=[r'$\beta$', r'$\gamma$', r'$\mu$'],␣
↪nvars=4, ndims=0, nreplicates=1, obsidx=None, outfunc=None)
```

[23]:
```
# Mostramos la condición Inicial
modelo_SIRD.initial_condition
```

[23]:
```
array([144499999,         1,         0,         0])
```

Encontrando los mejores parámetros

[24]:
```
%%time
modelo_SIRD.fit()
```

```
    $\beta$  $\gamma$     $\mu$
0  8.810466  8.765309  5.503376
Wall time: 8.48 s
```

[25]:
```
modelo_SIRD.best_params
```

[25]:
```
    $\beta$  $\gamma$     $\mu$
0  8.810466  8.765309  5.503376
```

[26]:
```
modelo_SIRD.best_error
```

[26]:
```
391858305786.126
```

Likelihood profiles

[27]:
```
%%time
modelo_SIRD.likelihood_profiles(npoints=50)
```

```
HBox(children=(FloatProgress(value=0.0, description='parameters', max=3.0, style=ProgressStyle(d
```

```
HBox(children=(FloatProgress(value=0.0, description='values within parameters', max=50.0, style=

HBox(children=(FloatProgress(value=0.0, description='values within parameters', max=50.0, style=

HBox(children=(FloatProgress(value=0.0, description='values within parameters', max=50.0, style=

Wall time: 10min 54s
```

[28]:
```
modelo_SIRD.result_profiles
```

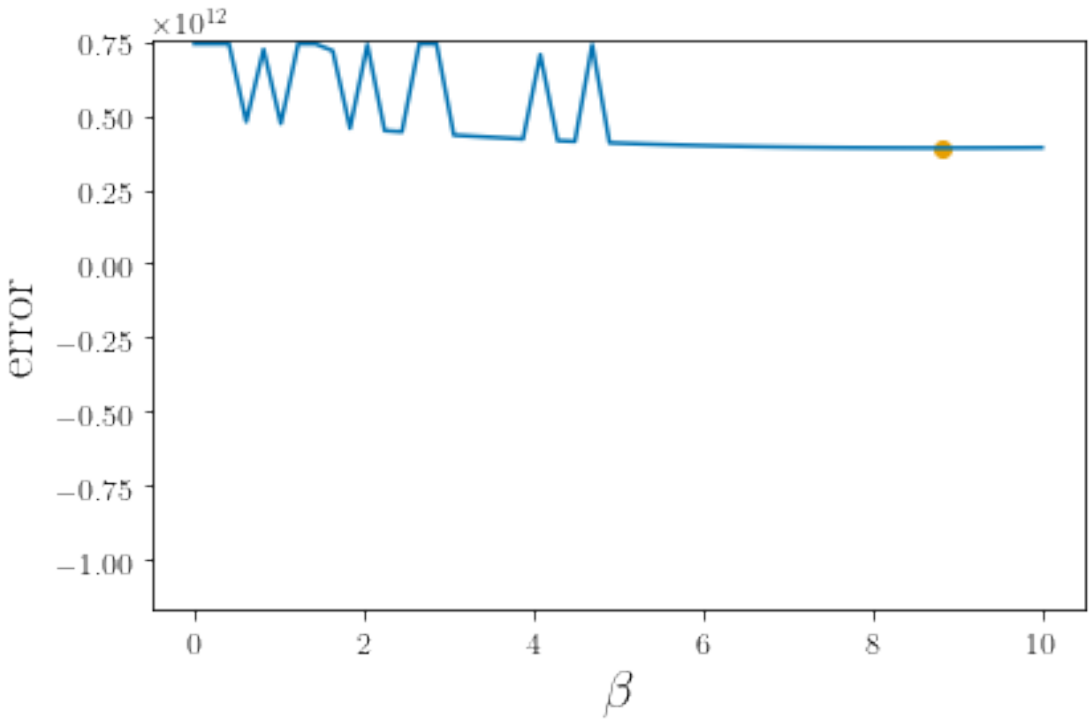
[28]:

	parameter	value	error
0	β	0.000000	7.445334e+11
1	β	0.204082	7.445094e+11
2	β	0.408163	7.445005e+11
3	β	0.612245	4.807974e+11
4	β	0.816327	7.260556e+11
..	...	...	...
145	μ	9.183673	7.428472e+11
146	μ	9.387755	7.443329e+11
147	μ	9.591837	3.918824e+11
148	μ	9.795918	3.918562e+11
149	μ	10.000000	3.918540e+11

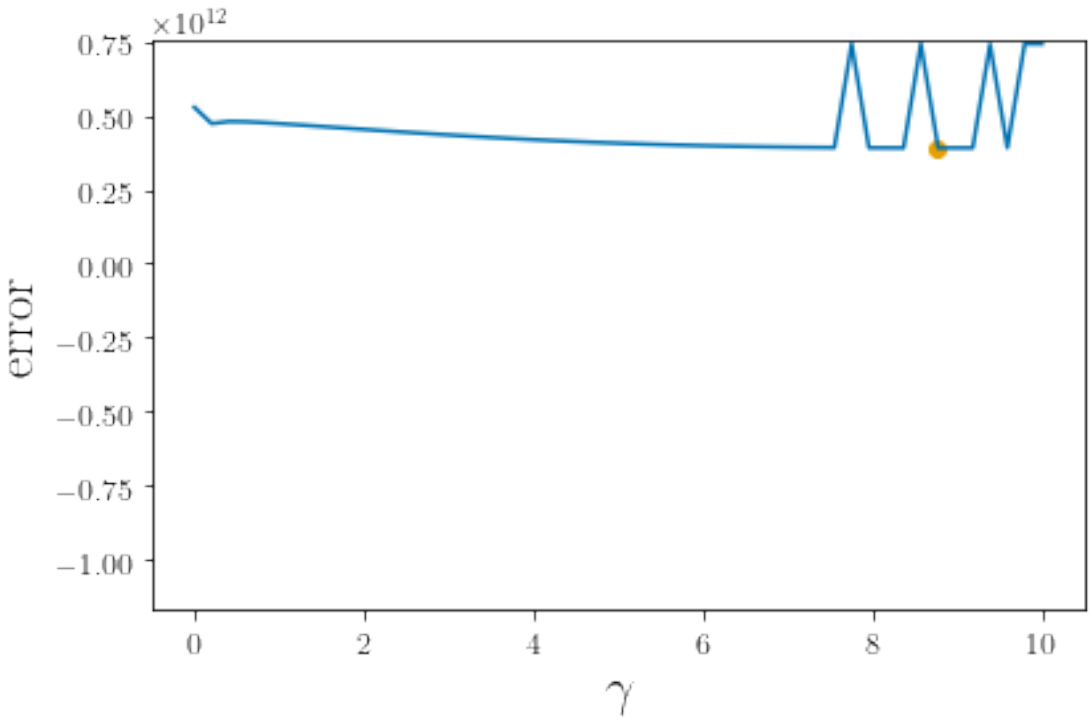
[150 rows x 3 columns]

[29]:
```
modelo_SIRD.plot_profiles()
```

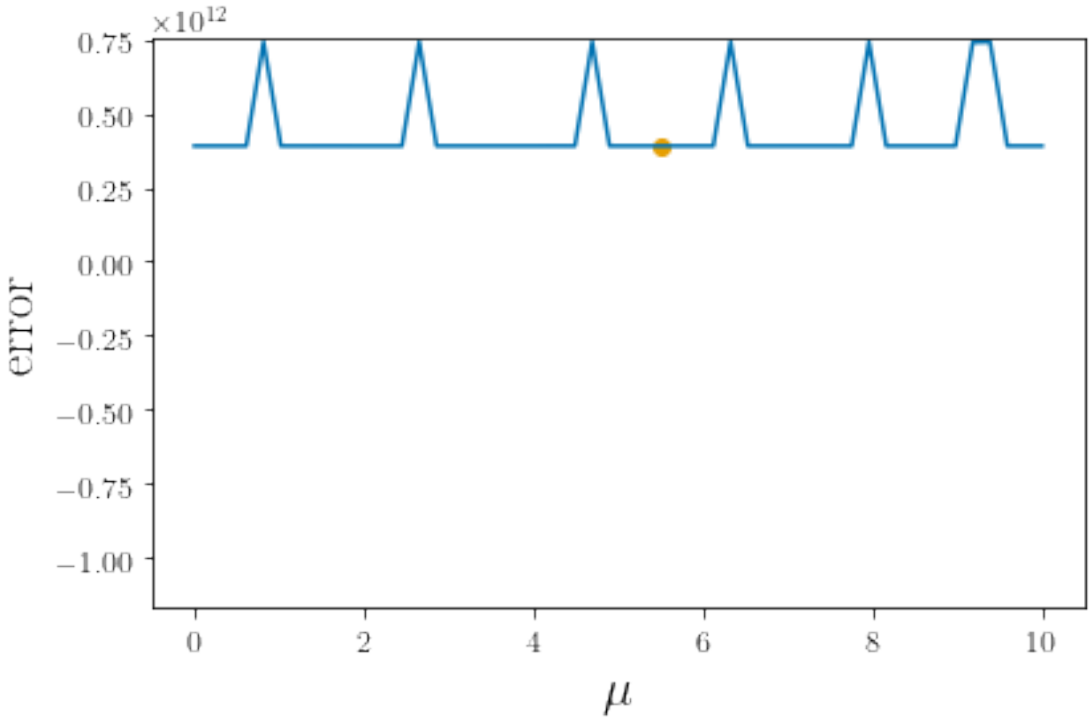
```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with
a single row if you intend to specify the same RGB or RGBA value for all points.
```




```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with
a single row if you intend to specify the same RGB or RGBA value for all points.
```




```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be
avoided as value-mapping will have precedence in case its length matches with
*x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with
a single row if you intend to specify the same RGB or RGBA value for all points.
```



1.2.1. Fuentes

- https://data.humdata.org/dataset/novel-coronavirus-2019-ncov-cases?force_layout=desktop
- https://www.google.com/search?q=poblacion+total+de+rusia&rlz=1C1CHBF_esMX925MX926&oq=pobl 8
- https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology#The_SIRD_model