SIR-Model-en

January 12, 2022

1 The S-I-R Model

Jörn Behrens (joern.behrens@uni-hamburg.de)

1.1 Introduction

In this Python Notebook we explain the S-I-R model. The S-I-R model is a simple mathematicsl model to compute the development of an epidemic. S-I-R stands for

- Susceptible,
- Infectious,
- Recovered/Removed.

We split the population into three groups:

1. S: those individuals not yet infected but susceptible,
2. I: the infected individuals,
3. R: the recovered (or deceased) and therefore not infectious individuals.

Additionally the following assumptions hold:

1. Each individual can become ill only once. After that he/she is either immune (or dead).
2. An individual cannot die without getting ill, i.e. he/she can either be healthy all the time or get infected before recovering or passing away.
3. The total number of individuals is constant, in other words recovered and deceased individuals are counted in the group R.
4. Infected individuals are infectious immediately.
5. Both the infection rate as well as the healing rate are independent of the group sizes and are assumed to be constant.
6. Each group interacts with the same probability with each other.

These assumptions are major simplifications of reality. In spite of this, the model stil allows to understand the mechanics of an epicemic.

1.2 The Model

1.2.1 Differential Equations

The following system of differential equations describes the growth and decay of the three different sets and is derived from considering related proportionalities:

$$\frac{dS}{dt} = -c\frac{S}{N}I \tag{1}$$
$$\frac{dI}{dt} = c\frac{S}{N}I - wI \tag{2}$$
$$\frac{dR}{dt} = wI \tag{3}$$

Furthermore the total number of individuals is constant, therefore

$$N = S + I + R.$$

1.3 Data

In order to solve the above syste of differential equations, we need further information. For example we need numbers for at least two of the groups for an initial time, $S_0 = S(t = 0)$ and $I_0 = I(t = 0)$, where we assume $t = 0$ to be the initial time. We also need to know the two constants c (the infection rate) and w (the healing rate).

In many cases we can assume that the number of recovered is zero at the beginning of our observation period. Knowing the number of infected, we can recall the number of susceptible from the relation $N = S + I + R$.

The infection rate and the healing rate can be derived from data. In our case, we proceed as follows: From the known total numbers of infected (look at the Robert-Koch-Insitute or the World Health Organization WHO) we see that every four to five days the number of infected doubles. So, the infection rate can be assumed to be 2 (doubling) every 4-5 days (our time unit). We set $c \approx \frac{2}{4,5}$.

The healing rate can be derived from the time it takes to recover. After approximately 10-14 days COVID-19 generally cured or deadly. So, we can assume $w \approx \frac{1}{12}$.

1.4 Numerical Solution

Now, we want to solve the S-I-R model numerically. Here we use Python for doing so. First, we implement a function for the right hand side of the above equation system. This function takes as an input three initial conditions *y0* for $S(t = 0)$, $I(t = 0)$ and $R(t = 0)$, the time axis t, the total number of Individuals N and the constants c and w. The output of this function the three quantities $S(t)$, $I(t)$, and $R(t)$ are returned.

[1]:
```
def SIR(y0,t,N,c,w):
    from numpy import size, array, shape, zeros
    # -- initial conditions - S, I, R:
    S = y0[0]
    I = y0[1]
    R = y0[2]
    # -- model equations
    r1 = -c*(S/N)*I
    r2 = c*(S/N)*I -w*I
    r3 = w*I
    # return evaluated right hand side
```

```
    r=zeros(shape(y0))
    r[0] = r1
    r[1] = r2
    r[2] = r3
    return r
```

1.5 Euler's method

One possibility to solve the model is to employ a simple Euler forward time stepping method. However, this method may either be unstable or at least quite inaccurate or inefficient…

[2]:
```
def euler(f,x0,t,N,c,w):
    from numpy import size, array, shape, zeros

    #size of output array=size of t array
    [n,]= shape(t)

    #size of system=size of initial state times sizes of t array
    [m,]= shape(x0)

    # initialize output array
    x=zeros([n,m])

    # initial value
    x[0,:]= x0
    #print(x0)

    # this is just the one line realization of the Euler method
    for i in range(n-1):
        x[i+1,:]= x[i,:]+ (t[i+1]-t[i])*f(x[i,:],t[i],N,c,w)

    return x
```

1.6 Heun's method

A simple Runge-Kutta method is *Heun's Method*, which is given by the following Butcher scheme:

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Therefore we can implement:

[3]:
```
def heun(f,x0,t,N,c,w):
    from numpy import size, array, shape, zeros

    #size of output array=size of t array
    [n,]= shape(t)
```

```
    #size of system=size of initial state times sizes of t array
    [m,]= shape(x0)

    # initialize output array
    x=zeros([n,m])

    # initial value
    x[0,:]= x0

    # this is just the one line realization of the Euler method
    for i in range(n-1):
        dt = (t[i+1]-t[i])
        xh = x[i,:] + dt* f(x[i,:],t[i],N,c,w)
        x[i+1,:]= x[i,:]+ 0.5*dt*(f(x[i,:],t[i],N,c,w) + f(xh,t[i+1],N,c,w))

    return x
```

1.7 Heun's Method of Third Order

A higher order Heun's method is given by the following Butcher Scheme and implemented thereafter:

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{3} & \frac{1}{3} & & \\ \frac{2}{3} & 0 & \frac{2}{3} & \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} \end{array}$$

[4]:
```
def heun3(f,x0,t,N,c,w):
    from numpy import size, array, shape, zeros

    #size of output array=size of t array
    [n,]= shape(t)

    #size of system=size of initial state times sizes of t array
    [m,]= shape(x0)

    # initialize output array
    x=zeros([n,m])

    # initial value
    x[0,:]= x0

    # this is just the one line realization of the Euler method
    for i in range(n-1):
        dt = (t[i+1]-t[i])
        k1 = f(x[i,:],t[i],N,c,w)
        x1 = x[i,:] + dt/3 * k1
        k2 = f(x1,t[i]+dt/3,N,c,w)
```

```
        x2 = x[i,:] + dt*2/3 * k2
        k3 = f(x2,t[i]+dt*2/3,N,c,w)
        x[i+1,:]= x[i,:]+ dt*(.25*k1 + .75*k3)

    return x
```

For the initial conditions we try to generate more or less realistical values for Germany. We are at the beginning of the Pandemic. We will do the computation with one timestep (or value for the three quantities) per day. And we will run the simulation for one year, i.e. 365 days.

Let us consider cases per 100,000 inhabitants, and we take the numbers of March 15, 2020, 15:00, which is our $t = 0$. At that time there were 5.83 infected cases per 100,000 officially registered, thus $I(t = 0) = 5.83$. We derived the infection rate and the healing rate above, so we will run our simulation with the following parameters:

$$N = 100.000 \tag{4}$$
$$I(t = 0) = 5,83 \tag{5}$$
$$R(t = 0) = 0 \tag{6}$$
$$S(t = 0) = N - I(t_0) - R(t_0) = 99.994,17 \tag{7}$$
$$c = \frac{2}{4.5} \tag{8}$$
$$w = \frac{1}{12} \tag{9}$$

[5]:
```
# -- Load required Python modules
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = 9, 6
from scipy.integrate import odeint

# -- define initial data
N = 100000       # total number of individuals
I0 = 5.83        # infected at initial time
R0 = 0.          # recovered at initial time
S0 = N - I0 - R0 # susceptible at initial time

cc = (2/(4.5))   # infection rate
ww = 1/12        # healing rate

# -- define data for equation solver
y0 = [S0, I0, R0]
t  = np.linspace(0,365.,1460) # compute 365 days mit 4 steps per day for smooth␣
 ↪visualization

# -- solve, using euler
SIRsol = euler(SIR, y0, t, N, cc, ww)
```

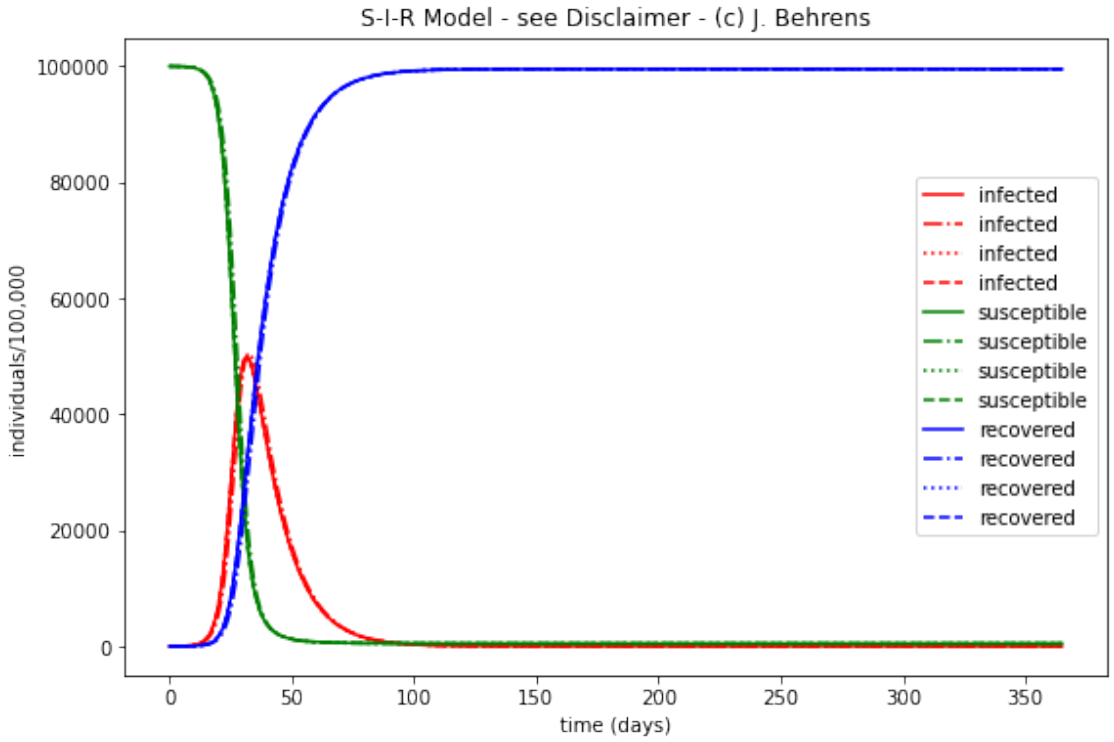
```
Seul = SIRsol[:,0]
Ieul = SIRsol[:,1]
Reul = SIRsol[:,2]

# -- solve, using heun
SIRsol = heun(SIR, y0, t, N, cc, ww)
Sheu = SIRsol[:,0]
Iheu = SIRsol[:,1]
Rheu = SIRsol[:,2]

# -- solve, using heun high order
SIRsol = heun3(SIR, y0, t, N, cc, ww)
She3 = SIRsol[:,0]
Ihe3 = SIRsol[:,1]
Rhe3 = SIRsol[:,2]

# -- solve
SIRsol = odeint(SIR, y0, t, args=(N, cc, ww))
S = SIRsol[:,0]
I = SIRsol[:,1]
R = SIRsol[:,2]

# -- plot the solution graphically
h1 = plt.plot(t,I,'r-',t,Ieul,'r-.',t,Iheu,'r:',t,Ihe3,'r--',label='infected')
h2 = plt.plot(t,S,'g-',t,Seul,'g-.',t,Sheu,'g:
 ↪',t,She3,'g--',label='susceptible')
h3 = plt.plot(t,R,'b-',t,Reul,'b-.',t,Rheu,'b:',t,Rhe3,'b--',label='recovered')
plt.legend(loc='center right')
plt.title('S-I-R Model - see Disclaimer - (c) J. Behrens')
plt.xlabel('time (days)')
plt.ylabel('individuals/100,000');
```



1.8 Error

In order to analyze the error, we take the l^∞-norm of the difference between the Euler solution and the scipy solution.

[6]:
```
Serr = np.linalg.norm((S-Seul),np.inf)
Ierr = np.linalg.norm((I-Ieul),np.inf)
Rerr = np.linalg.norm((R[1:]-Reul[1:]),np.inf)
print('Errors Euler: Rerr= ',Rerr,' Ierr= ',Ierr,' Serr= ',Serr)
Serr = np.linalg.norm((S-Sheu),np.inf)
Ierr = np.linalg.norm((I-Iheu),np.inf)
Rerr = np.linalg.norm((R[1:]-Rheu[1:]),np.inf)
print('Errors Heun: Rerr= ',Rerr,' Ierr= ',Ierr,' Serr= ',Serr)
Serr = np.linalg.norm((S-She3),np.inf)
Ierr = np.linalg.norm((I-Ihe3),np.inf)
Rerr = np.linalg.norm((R[1:]-Rhe3[1:]),np.inf)
print('Errors Heun3: Rerr= ',Rerr,' Ierr= ',Ierr,' Serr= ',Serr)
```

```
Errors Euler: Rerr=  4238.480662993432  Ierr=  5401.49529457949  Serr=
8141.6394804517695
Errors Heun: Rerr=  110.45003249914225  Ierr=  152.6738134649786  Serr=
226.41006705468317
Errors Heun3: Rerr=  2.1825512611831073  Ierr=  2.8896146148908883  Serr=
4.283266190483118
```

1.9 Interpretation of the Model

From the above figure we see that after approx. 10 days nothing really has happened. Only at the 25th day or so, a number of 20,000 infected per 100,000 is reached. The maximum with approx. 50,000 infected occurs at day 40. Extrapolating this number to the population of Germany (of approx. 80 million) leads to a massive number. Assuming that "only" 6% of the infected really need intensive medical care would still mean that approx. 2.4 million individuals would need treatment…

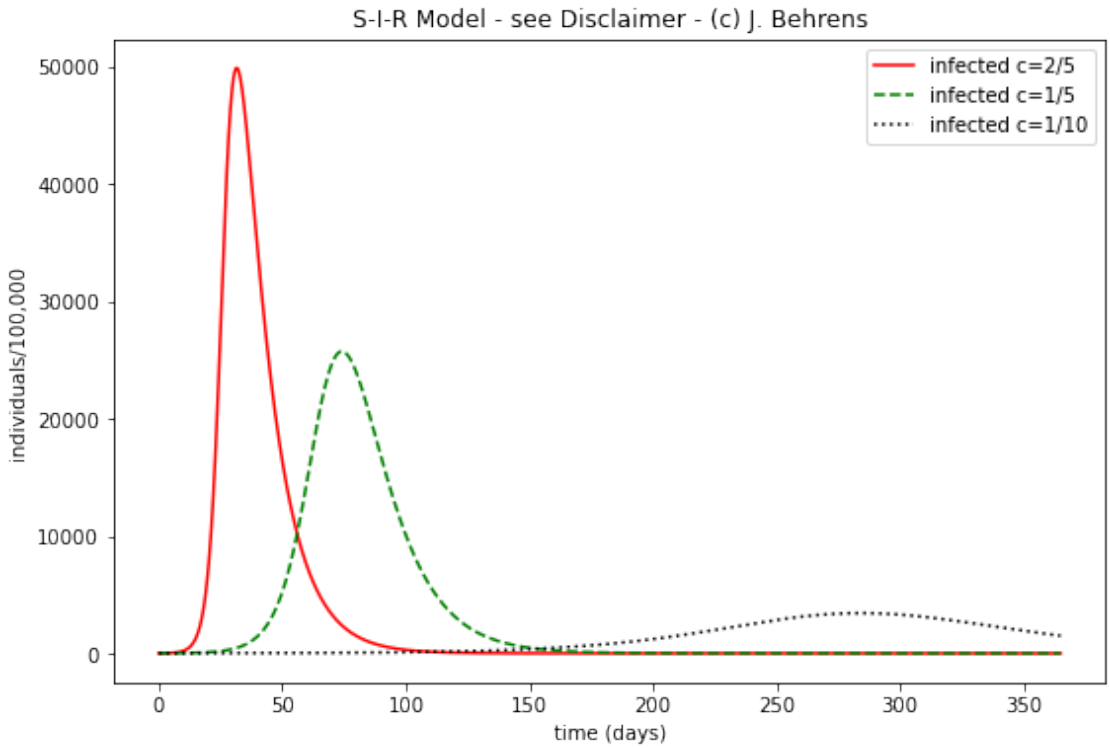
At least, after 100 days the epidemic is over, because at that time almost everybody has been infected and the infected by then have been recovered (or passed away).

So, how can we reduce the number of infected? On the one hand we could try to reduce the number of susceptible. This could be achieved by vaccination. Unfortunately no vaccine exists for the Corona virus yet. So, the only parameter that can be influenced is **the infection rate** c. Reducing c means, isolating the infected. And this is exactly, what the current rules try to achieve.

So, let us assume that the infection rate can be reduced to one half or even one quarter of the original value. Then we can compute our model again and obtain the following results:

[7]:
```
cc_halb = cc*0.5  # half the infection rate
cc_quar = cc*0.25 # a quarter the infection rate
SIRhalb = odeint(SIR, y0, t, args=(N, cc_halb, ww))
I_halb  = SIRhalb[:,1]
SIRquar = odeint(SIR, y0, t, args=(N, cc_quar, ww))
I_quar  = SIRquar[:,1]

# -- plot the result
h1 = plt.plot(t,I,'r-',label='infected c=2/5')
h2 = plt.plot(t,I_halb,'g--',label='infected c=1/5')
h4 = plt.plot(t,I_quar,'k:',label='infected c=1/10')
plt.legend(loc='upper right')
plt.title('S-I-R Model - see Disclaimer - (c) J. Behrens')
plt.xlabel('time (days)')
plt.ylabel('individuals/100,000');
```



1.10 Interpretation of the Modified Model

With this slow down of the spread of the infection we achieve that on the one hand the maximum number of infected is reached after approx 90 days and is only half as large (approx. 25,000) compared to the original model. Assuming again that only 6% need intensive care and extrapolating to the total German population results in approx. 1.2 million sick individuals.

Even more successful would be to decrease the infection rate to a quarter. In this scenario the maximum number of infected after approx. 275 days would be just 3,000 per 100,000, resulting in 144,000 requiring intensive care all over Germany. On the other hand, the epidemic would still not be completely over after one year.

1.11 Warning (Disclaimer)

The data used in this example are not authoritative and the model is largely simplified. The example computations are meant to demonstrate the mathematics behind epidemiologic modeling. They are intendet to understand the mechanics of such simulations. **The values and results are not representative and cannot be used for political decisions, opinion making or political measures and rules.**

The author accepts no liability or responsibility for using these model computations. This worksheet is given to the public for educational purposes only.

1.12 References

1. Robert-Koch-Institut (2020): SARS-CoV-2 Steckbrief zur Coronavirus-Krankheit-2019 (COVID-19), https://www.rki.de/DE/Content/InfAZ/N/Neuartiges_Coronavirus/Steckbrief.html (last access: 19.03.2020).
2. Max-Delbück-Zentrum (2020): COVID-19 Bundesländer in Deutschland v0.004, https://covid19germany.mdc-berlin.de (last access: 19.03.2020).
3. Wikipedia: SIR-Modell, https://de.wikipedia.org/wiki/SIR-Modell (last access: 19.03.2020).

[ ]: