SIR-Modell-de

January 12, 2022

1 Das S-I-R Modell

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

1.1 Einführung

In diesem Python Arbeitsblatt wird das S-I-R Modell erklärt. Das S-I-R Modell ist ein einfaches mathematisches Modell zur Berechnung einer Epidemie. S-I-R steht für die englischen Wörter

- Susceptible (anfällig),
- Infectious (ansteckend),
- Recovered/Removed (genesen/entfernt/rekonvaleszent).

Man teilt also eine Bevölkerung in drei Gruppen:

1. S: die noch nicht infizierten, aber anfälligen,
2. I: die infizierten und daher anstechenden und,
3. R: die genesenen (oder gestorbenen), und daher nicht mehr ansteckenden Personen.

Weiterhin werden zunächst folgende Annahmen getroffen:

1. Jedes Individuum kann die Krankheit nur einmal bekommen und ist danach immun (oder tot).
2. Ein Individuum kann nicht ohne krank zu werden sterben, also kann es entweder immer gesund sein oder sich infizieren, um anschließend wieder zu genesen oder zu sterben.
3. Die Anzahl der Individuen insgesamt ist konstant, das heißt genesene und tote Individuen werden der Gruppe R zugezählt.
4. Infizierte Personen sind sofort ansteckend.
5. Sowohl die Ansteckungsrate als auch die Genesungsrate sind unabhängig von der Anzahl der jeweiligen Gruppen und werden als konstante Faktoren angenommen.
6. Jede der Gruppen agiert miteinander mit derselben Wahrscheinlichkeit.

Diese Annahmen sind starke Vereinfachungen der Wirklichkeit. Trotzdem ist das Modell gut geeignet den Mechanismus einer solchen Epidemie zu verstehen.

1.2 Das Modell

1.2.1 Differentialgleichungen

Das Wachstum und die Abnahme der Mengen S, I und R lässt sich mit Hilfe des folgenden Systems von Differentialgleichungen beschreiben, das man aus Proportionalitätsüberlegungen erhält:

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

Weiterhin gilt - wie in den Annahmen formuliert - dass die Gesamtzahl N der Individuen sich nicht ändert, also

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

1.3 Daten

Um die obigen Gleichungen lösen zu können, bedarf es weiterer Informationen. Wir benötigen beispielsweise die Zahlen in mindestens zwei der Gruppen zum Beginn des Beobachtungszeitraumes, $S_0 = S(t = 0)$ und $I_0 = I(t = 0)$, wobei wir $t = 0$ als den Anfangszeitpunkt nehmen. Auch müssen wir die beiden Konstanten c, die Infektionsrate, und w, die Genesungsrate, kennen.

Oft kann man davon ausgehen, dass die Anzahl der Genesenen zu Beginn eines Zeitraumes gleich Null ist. Kennt man dann die Anzahl der Infizierten, so lässt sich die Anzahl der Anfälligen aus der Gesamtzahl und der Beziehung $N = S + I + R$ herleiten.

Die Infektionsraten und Genesungsraten kann man sich aus Daten herleiten. In unserem Fall gehen wir wie folgt vor. Aus den bekannten Zahlen (beispielsweise des Robert-Koch-Insitutes oder der Weltgesundheitsorganisation WHO) können wir ersehen, dass sich die Zahl der Infizierten etwa alle 4 bis 5 Tage verdoppelt. Die Änderungsrate der Gruppe der Infizierten wäre also 2 (Verdoppelung) pro 4-5 Tage (das ist unsere Zeiteinheit). Wir setzen also $c \approx \frac{2}{4,5}$.

Die Genesungsrate kann man aus der Zeit ableiten, die im Schnitt für die Heilung notwendig ist. Nach etwa 10-14 Tagen ist die Krankheit in der Regel entweder tödlich oder im wesentlichen geheilt. Wir werden also $w \approx \frac{1}{12}$ annehmen.

1.4 Numerische Lösung

Jetzt wollen wir das S-I-R Modell numerisch lösen. Dazu wird hier Python verwendet. Zunächst implementieren wir die Rechte Seite als eine Funktion, die als Eingabe die drei Anfangsbedingungen *y0* für $S(t = 0)$, $I(t = 0)$ und $R(t = 0)$, die Zeitachse t, die Gesamtzahl der Individuen N, und die beiden Änderungsraten c und w annimmt. Als Ausgabe wirderden die drei Größen $S(t)$, $I(t)$ und $R(t)$ zurück gegeben, also die Funktionen für die Anfälligen, die Infizierten und die Genesenen.

[1]:
```
def SIR(y0,t,N,c,w):
    from numpy import size, array, shape, zeros
    # -- zunächst die Anfangsdaten - S, I, R:
    S = y0[0]
```

```
    I = y0[1]
    R = y0[2]
    # -- jetzt die Modellgleichungen
    r1 = -c*(S/N)*I
    r2 = c*(S/N)*I -w*I
    r3 = w*I
    # Rückgabe der berechneten rechten Seite
    r=zeros(shape(y0))
    r[0] = r1
    r[1] = r2
    r[2] = r3
    return r
```

1.5 Euler-Verfahren

Die einfachste Möglichkeit das System zu lösen, ist das Euler-Verfahren anzuwenden. Jedoch ist es eventuell nicht stabil, auf jeden Fall nicht besonders genau und effizient…

[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-Verfahren

Die einfachste Runge-Kutta Methode ist das *HeunVerfahren*, welche durch das Butcher-Schema gegeben ist:

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

Die Implementierung sieht entsprechend wiefolg aus:

[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-Verfahren dritter Ordnung

Ein Verfahren höherer Ordnung, das in der Vorlesung besprochen wurde, ist das durch das folgende Butcher-Schema gegebene Heun-Verfahren dritter Ordnung, das anschließend implementiert wird:

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

[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
```

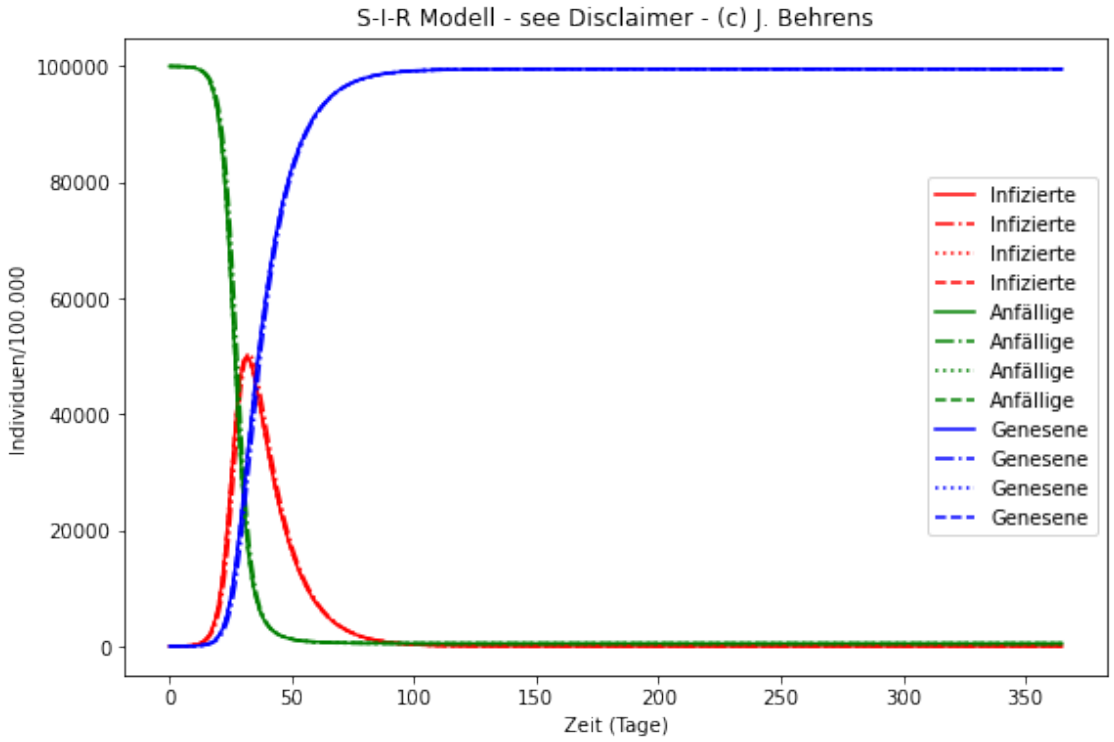
Für die Anfangswerte werden wir versuchen realistische Werte für Deutschland zu nehmen. Wir befinden uns im Anfangsstadium der Infektion. Wir werden mit einem Zeitschritt von einem Tag rechnen. Die Simulation wird sich über ein Jahr, also 365 Tage erstrecken und wir werden für jeden Tag eine Zahl für die jeweiligen Gruppen erhalten.

Wir rechnen mit Fallzahlen umgerechnet auf 100.000 Einwohner nehmen die Zahl der Infizierten vom 15.3.2020, 15:00 als Wert für $t = 0$ in unserer Berechnung. Zu dem Zeitpunkt waren im Bundesdurchschnitt $5,83$ Fälle pro 100.000 gemeldet, also ist $I(t = 0) = 5,83$. Die Infektionsrate und die Genesungsrate hatten wir oben schon besprochen, so dass sich die folgenden Daten (Parameter) für unsere Simulation ergeben:

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

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

# -- Definiere Anfangsdaten
N = 100000       # Gesamtzahl der Individuen
I0 = 5.83        # Infizierte Fälle zum Anfangszeitpunkt
R0 = 0.          # Anzahl Genesener zum Anfangszeitpunkt
S0 = N - I0 - R0 # Anzahl noch Gesunder zum Anfangszeitpunkt
```

```
cc = (2/(4.5))   # Infektionsrate
ww = 1/12        # Genesungsrate

# -- definiere die Daten für den Gleichungslöser
y0 = [S0, I0, R0]
t  = np.linspace(0,365.,1460) # rechne für 365 Tage mit 4 Schritten pro Tag

# -- Löse mit Euler
SIRsol = euler(SIR, y0, t, N, cc, ww)
Seul = SIRsol[:,0]
Ieul = SIRsol[:,1]
Reul = SIRsol[:,2]

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

# -- Löse mit Heun hoher Ordnung
SIRsol = heun3(SIR, y0, t, N, cc, ww)
She3 = SIRsol[:,0]
Ihe3 = SIRsol[:,1]
Rhe3 = SIRsol[:,2]

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

# -- Stelle die Lösung graphisch dar
h1 = plt.plot(t,I,'r-',t,Ieul,'r-.',t,Iheu,'r:',t,Ihe3,'r--',label='Infizierte')
h2 = plt.plot(t,S,'g-',t,Seul,'g-.',t,Sheu,'g:',t,She3,'g--',label='Anfällige')
h3 = plt.plot(t,R,'b-',t,Reul,'b-.',t,Rheu,'b:',t,Rhe3,'b--',label='Genesene')
plt.legend(loc='center right')
plt.title('S-I-R Modell - see Disclaimer - (c) J. Behrens')
plt.xlabel('Zeit (Tage)')
plt.ylabel('Individuen/100.000');
```



1.8 Fehler

Um den Fehler zu untersuchen, berechnen wir die l^∞-Norm der Differenz der eigenen numerischen Verfahren mit der scipy Lösung.

[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('Fehler 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('Fehler 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('Fehler Heun3: Rerr= ',Rerr,' Ierr= ',Ierr,' Serr= ',Serr)
```

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

1.9 Interpretation des Modells

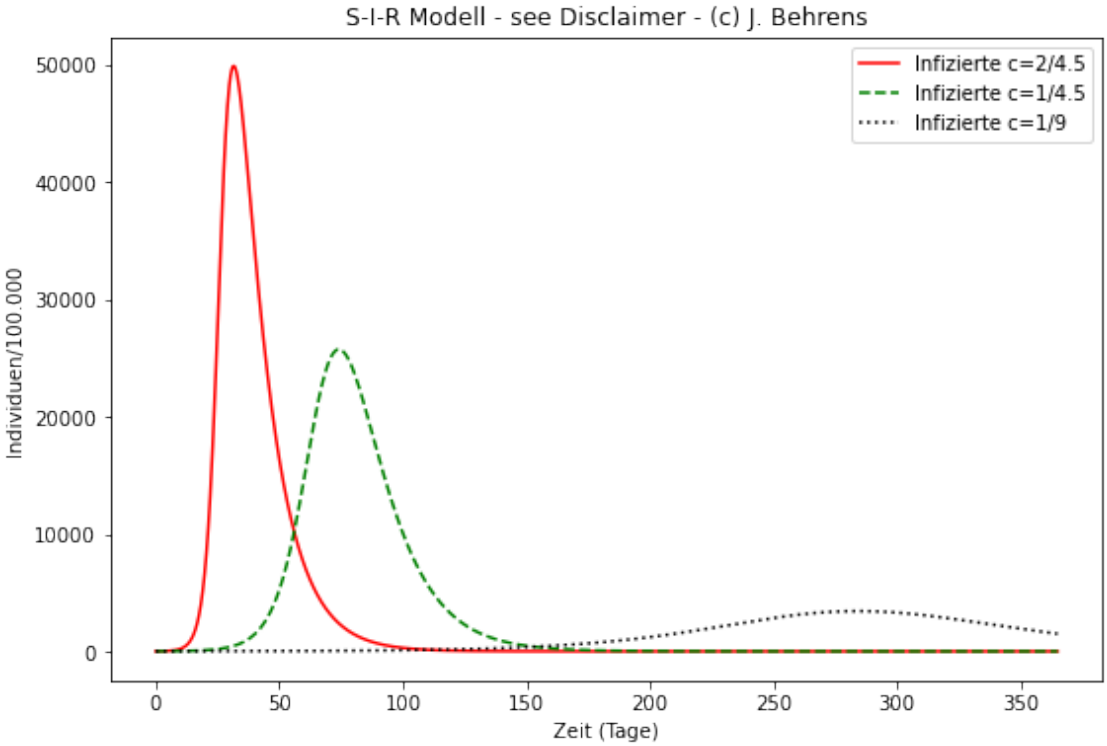
In der Graphik oben erkennt man, dass nach ca. 10 Tagen noch wenig passiert ist. Erst etwa am 25sten Tag erreicht man ca. 20.000 Infizierte pro 100.000 Einwohner und das Maximum mit ca. 50.000 Infizierten ist nach etwa 40 Tagen erreicht. Man kann sich leicht ausrechnen, dass eine solche Zahl hochgerechnet auf die Bundesbevölkerung eine immense Zahl ist. Selbst wenn man weiß dann wohl "nur" 6% der Infizierten tatsächlich intensive medizinische Versorgung benötigen, wären das immer noch etwa 2,4 Millionen Individuen…

Immerhin sieht man auch, dass nach etwa 100 Tagen die Epidemie vorbei ist, weil zu diesem Zeitpunkt sich fast alle angesteckt haben und die Angesteckten inzwischen wieder gesund geworden (oder gestorben) sind.

Wie kann jetzt die Zahl der Infizierten verringert werden? Wir könnten einerseits die Anzahl der Anfälligen senken, indem wir impfen. Leider existiert für Corona noch keine Impfung. Es bleibt also der einzige Parameter, den wir tatsächlich beeinflussen können: **die Infektionsrate**. Die Infektionsrate zu verringern bedeutet, die Infizierten zu isolieren. Und das soll mit den gegenwärtigen Maßnahmen erreicht werden.

Nehmen wir also einmal an, die Infektionsrate könnte durch die einschneidenden Maßnahmen auf etwa die Hälfte oder sogar ein viertel gesenkt werden. Dann können wir eine solche Rechnung nochmals mit dem neuen Wert für den Parameter c durchführen:

[7]:
```
cc_halb = cc*0.5  # Halbierung der Infektionsrate
cc_quar = cc*0.25 # Viertelung der Infektionsrate
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]

# -- Stelle die Lösung graphisch dar
h1 = plt.plot(t,I,'r-',label='Infizierte c=2/4.5')
h2 = plt.plot(t,I_halb,'g--',label='Infizierte c=1/4.5')
h4 = plt.plot(t,I_quar,'k:',label='Infizierte c=1/9')
plt.legend(loc='upper right')
plt.title('S-I-R Modell - see Disclaimer - (c) J. Behrens')
plt.xlabel('Zeit (Tage)')
plt.ylabel('Individuen/100.000');
```



1.10 Interpretation des modifizierten Modells

Mit dieser Verlangsamung der Infektionsausbreitung erreichen wir nun folgendes: Einerseits tritt die größte Zahl infizierte Individuen jetzt erst nach ca. 90 Tagen auf, dafür erreichen wir aber auch nur etwa maximal 25.000 Infizierte pro 100.000 Einwohner. Wieder hochgerechnet und angenommen, dass 6% der Fälle auch intensivmedizinisch versorgt werden müssen bedeutet das eine (gleichzeitige) Anzahl von Kranken von 1,2 Millionen, also eine Halbierung der ersten Zahl! Richtig erfolgreich wären die Maßnahmen, wenn die Infektionsrate auf ein viertel reduziert werden könnte. Dann liegen wir nach etwa 275 Tagen bei gerade einmal etwa 3.000 Infizierten pro 100.000 Einwohner oder etwa 144.000 Fällen in intensiver medizinischer Behandlung. Allerdings wäre in diesem Fall auch nach einem Jahr die Epidemie noch nicht vollkommen vorbei.

1.11 Warnung (Disclaimer)

Die hier verwendeten Daten sind nicht offizielle Daten und das Modell, ist ein sehr stark vereinfachtes Modell. Die hier gezeigten Beispielrechnungen dienen dazu die Mathematik hinter epidemiologischen Modellrechnungen zu veranschaulichen und zu verstehen, wie der Mechanismus solcher Simulationen funktioniert. **Die Werte sind nicht repräsentativ und können nicht für politische Entscheidungen, Meinungsbildung, oder entsprechende Maßnahmen verwendet werden.**

Der Autor übernimmt keinerlei Haftung für die Verwendung dieser Modellrechnungen. Sie werden der Öffentlichkeit lediglich zu didaktischen Zwecken überlassen.

1.12 Referenzen

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 (letzter Zugriff: 19.03.2020).
2. Max-Delbück-Zentrum (2020): COVID-19 Bundesländer in Deutschland v0.004, https://covid19germany.mdc-berlin.de (letzter Zugriff: 19.03.2020).
3. Wikipedia: SIR-Modell, https://de.wikipedia.org/wiki/SIR-Modell (letzter Zugriff: 19.03.2020).

[ ]: