**Модель эпидемии (SIR-модель)**

Предположим, что некая популяция, состоящая из 𝑁 особей, подразделяется в каждый момент времени 𝑡 на три группы:

• 𝑆(𝑡) (susceptible) — здоровые особи, которые находятся в группе риска и могут подхватить инфекцию;

• 𝐼(𝑡) (infected) — инфицированные особи; являются переносчиками инфекции;

• 𝑅(𝑡) (removed) — «выбывшие» особи, к которым относятся выздоровевшие особи, приобретшие иммунитет к данной болезни, а также умершие.

Таким образом, 𝑆(𝑡) + 𝐼(𝑡) + 𝑅(𝑡) = 𝑁. (1)

Классическая модель эпидемии (SIR-модель) имеет вид

𝑆˙ = −𝛽𝑆𝐼

˙𝐼 = 𝛽𝑆𝐼 – 𝛾𝐼

𝑅˙ = 𝛾𝐼 (2)

Здесь параметры 𝛽 и 𝛾 имеют смысл коэффициентов «заболеваемости» и «выбывания» соответственно. Что касается начальных условий, то естественно предположить, что изначально в популяции нет особей с иммунитетом к болезни, т. е. 𝑅(0) = 0. Уравнения (2) носят название системы Кермака — Маккендрика (W. O. Kermack and A. G. McKendrick, 1927).

В моем случае было проведено исследование простой модели SIR(входные параметры задаются случайным образом)

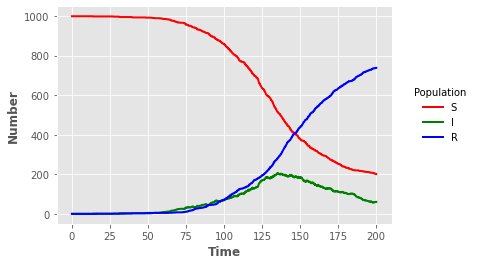
infection = np.random.binomial(S,ifrac) - случайный параметр, биноминальное распределение

от S до ifrac recovery = np.random.binomial(I,rfrac) - случайный параметр, биноминальное распределение, от I до ifrac.

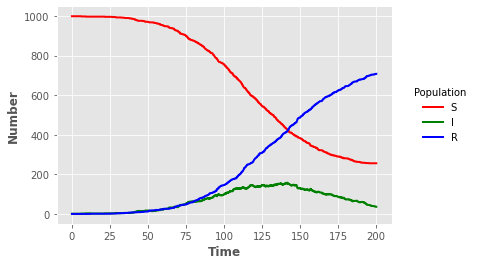
Т е при каждой итерации зараженные(I) и выздоровевшие(R) получаются случайным образом из остатков S и I.

Чтобы каждый раз получать идентичные последовательности чисел, используется функция np.random.seed(), она фиксирует число, служащее начальной точкой для запуска алгоритма генерации случайных чисел. В качестве аргумента функции указывают любое целое число (не важно, какое именно).

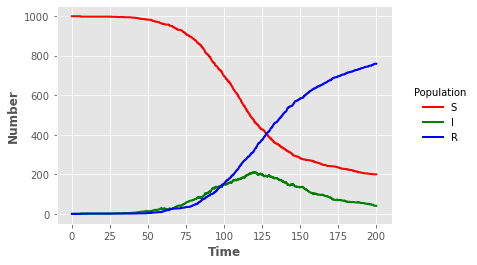
При начальном числе алгоритма генерации случайных чисел 69:



30:



180



Код на Python 3:

def sir(u,parms,t):

bet,gamm,iota,N,dt=parms

S,I,R,Y=u

lambd = bet\*(I+iota)/N

ifrac = 1.0 - math.exp(-lambd\*dt)

rfrac = 1.0 - math.exp(-gamm\*dt)

infection = np.random.binomial(S,ifrac)

recovery = np.random.binomial(I,rfrac)

return [S-infection,I+infection-recovery,R+recovery,Y+infection]

def simulate():

parms = [0.1, 0.05, 0.01, 1000.0, 0.1]

tf = 200

tl = 2001

t = np.linspace(0,tf,tl)

S = np.zeros(tl)

I = np.zeros(tl)

R = np.zeros(tl)

Y = np.zeros(tl)

u = [999,1,0,0]

S[0],I[0],R[0],Y[0] = u

for j in range(1,tl):

u = sir(u,parms,t[j])

S[j],I[j],R[j],Y[j] = u

return {'t':t,'S':S,'I':I,'R':R,'Y':Y}