PW6 Epidemic models

The SIR epidemic model

A simple mathematical description of the spread of a disease in a population is the so-called SIR model, which divides the (fixed) population of N individuals into three "compartments" which may vary as a function of time, t:

-S(t) are those susceptible but not yet infected with the disease;

-I(t) is the number of infectious individuals;

-R(t) are those individuals who have recovered from the disease and now have immunity to it.

The SIR model describes the change in the population of each of these compartments in terms of two parameters, β and γ. Here, β represents the transmission rate, i.e., the rate of healthy people becoming infected and γ represents the recovery rate, i.e., the rate of infected people becoming withdrawn.

The differential equations describing this model were first derived by Kermack and McKendrick [Proc. R. Soc. A, 115, 772 (1927)]:

$$\frac{dS}{dt} = -\beta S I$$
$$\frac{dI}{dt} = \beta S I - \gamma I,$$
$$\frac{dR}{dt} = \gamma I,$$

The following Python code integrates these equations for a disease characterised by parameters $\beta = 0.0005$, $1/\gamma = 10$ days in a population of $N = 1000$. (perhaps flu in a school). The model is started with a single infected individual on day 0 : $I(0) = 1$. The curves plots $S(t)$, $I(t)$ and $R(t)$ on a same graph.

The program uses the function "odeint" that is a pre-programmed function that integrates the differential equations system. The "odeint" is part of ODEPACK that is a FORTRAN77 library which implements a variety of solvers for ordinary differential equations.

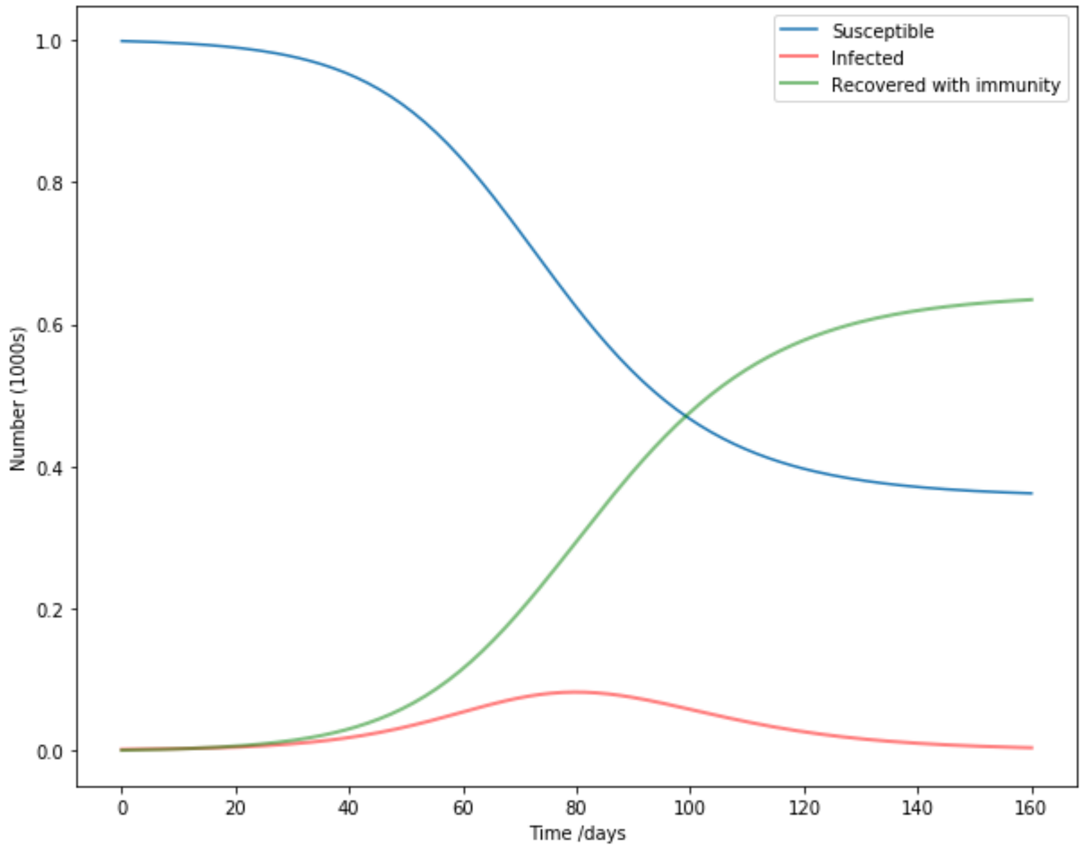
Plot several graphs with different values of R_0, below and above the reproduction number threshold

$$R_0 = \frac{\beta S(0)}{\gamma} = 1$$

. Vary also the other parameter $\gamma = 1/\delta t$, δt being the average duration of the infection and discuss the obtained results. Plot several curves. On each graph, display as a legend the value of R_0 and the values of the modified parameters.

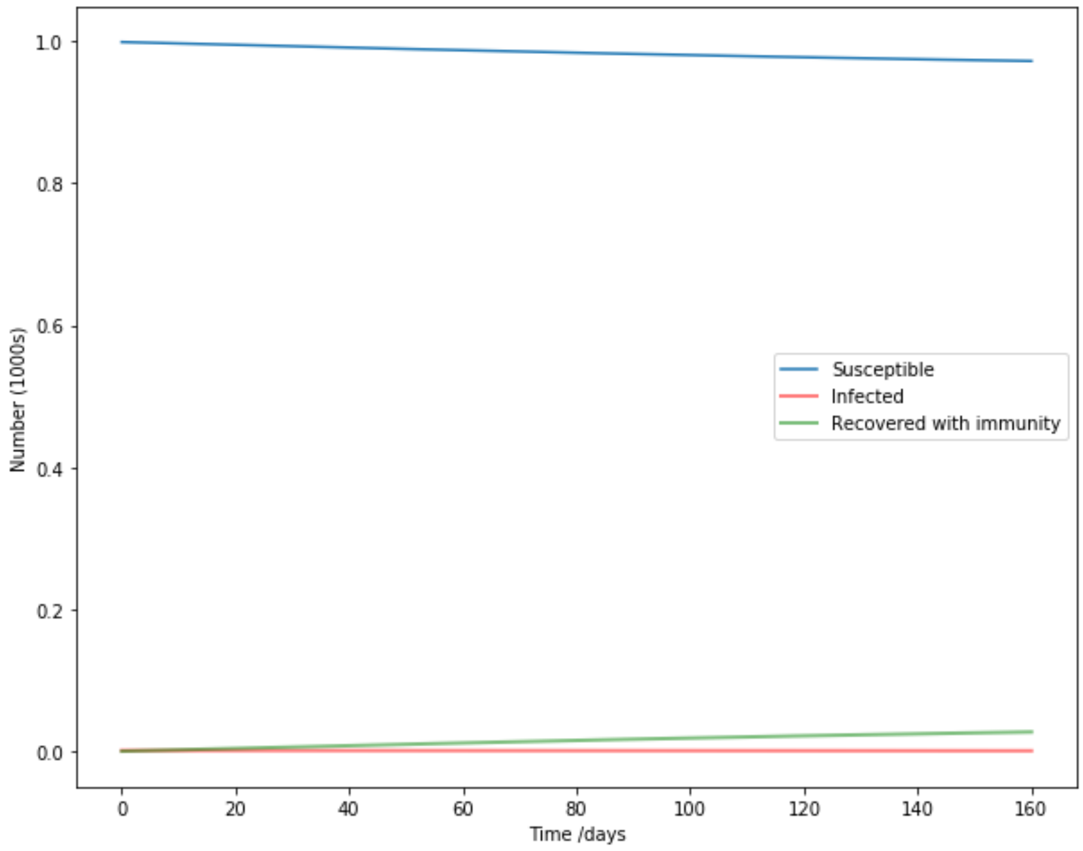
In [8]:
```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

# Total population, N.
N = 1000
# Initial number of infected and recovered individuals, I0 and R
0.
# Be aware that R0 is the number of recovered individuals at t=
0,
# and has nothing to do with R_0 the reproduction number
I0, R0 = 1, 0
# Everyone else, S0, is susceptible to infection initially.
S0 = N - I0 - R0
# Contact rate, beta, and mean recovery rate, gamma, (in 1/day
s).
beta, gamma = 0.0002, 1/8
# A grid of time points (in days)
t = np.linspace(0, 160, 160)

# The SIR model differential equations.
def deriv(y, t, N, beta, gamma):
    S, I, R = y
    dSdt = -beta * S * I
    dIdt = beta * S * I  - gamma * I
    dRdt = gamma * I
    return dSdt, dIdt, dRdt

# Initial conditions vector
y0 = S0, I0, R0
# Integrate the SIR equations over the time grid, t and the solu
tion
# is a table of 3 columns  and 160 rows
ret = odeint(deriv, y0, t, args=(N, beta, gamma))
#print(type(ret),ret.shape)
#to check the size of the table
print(type(ret),ret.shape)
#print(ret)
#print()
#print(ret[:,0])
#It takes the first column of the array
S=ret[:,0]
I=ret[:,1]
R=ret[:,2]
#S, I, R = ret.T is a shorter way of taking the transposition
#print(ret.T)

R_0 = (beta * S0) / gamma

# Plot the data on three separate curves for S(t), I(t) and R(t)
fig = plt.figure(facecolor='w',figsize=[10,8])
fig
plt.plot(t, S/1000,  label='Susceptible')
plt.plot(t, I/1000, 'r', alpha=0.5, lw=2, label='Infected')
plt.plot(t, R/1000, 'g', alpha=0.5, lw=2, label='Recovered with
immunity')

plt.xlabel('Time /days')
plt.ylabel('Number (1000s)')
plt.legend()

plt.show()

print("R_0 = ", R_0)
print("Gamma = ", gamma)
```

```
<class 'numpy.ndarray'> (160, 3)
```



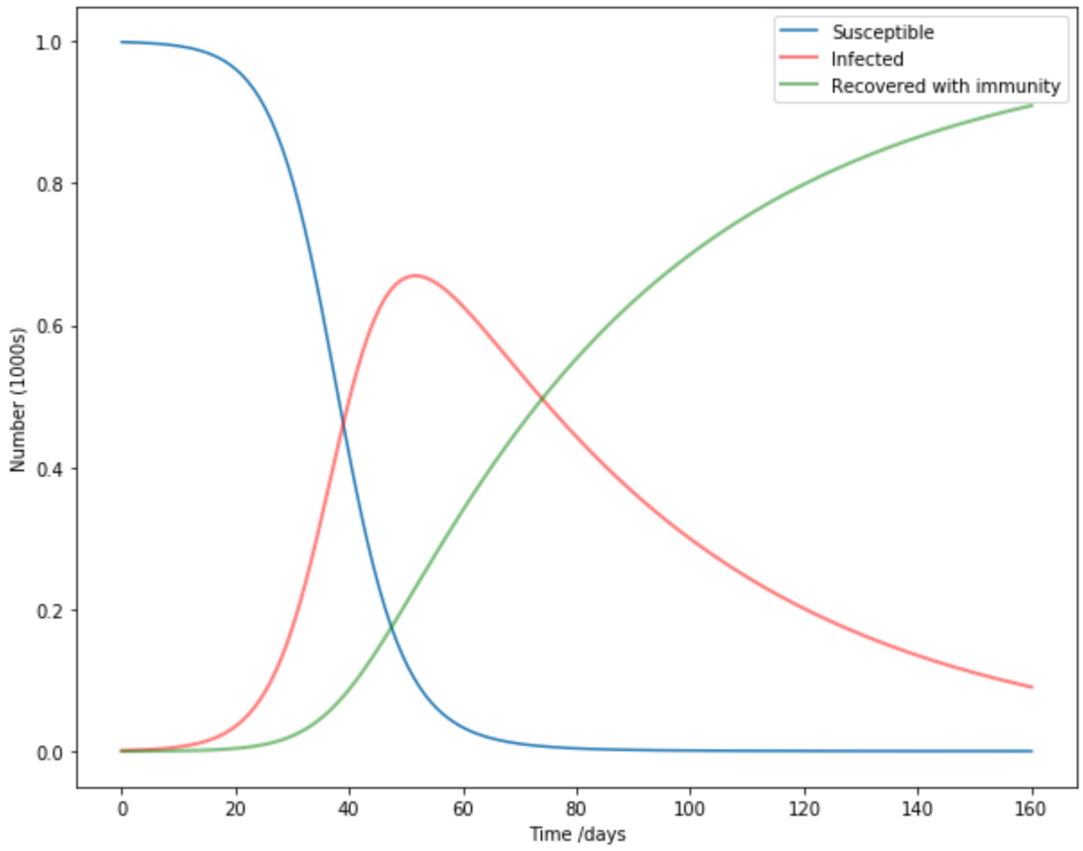
```
R_0 =  1.5984
Gamma =  0.125
```

In [2]:
```
# Total population, N.
N = 1000

# Initial number of infected and recovered individuals, I0 and R
0.
I0, R0 = 1, 0

# Everyone else, S0, is susceptible to infection initially.
S0 = N - I0 - R0

# Contact rate, beta, and mean recovery rate, gamma, (in 1/day
s).
beta, gamma = 0.0002, 1/5

# A grid of time points (in days)
t = np.linspace(0, 160, 160)

# The SIR model differential equations.
def deriv(y, t, N, beta, gamma):
    S, I, R = y
    dSdt = -beta * S * I
    dIdt = beta * S * I  - gamma * I
    dRdt = gamma * I
    return dSdt, dIdt, dRdt

# Initial conditions vector
y0 = S0, I0, R0

# Integrate the SIR equations over the time grid, t and the solu
tion
ret = odeint(deriv, y0, t, args=(N, beta, gamma))

S=ret[:,0]
I=ret[:,1]
R=ret[:,2]

R_0 = (beta * S0) / gamma

# Plot the data on three separate curves for S(t), I(t) and R(t)
fig = plt.figure(facecolor='w',figsize=[10,8])
fig
plt.plot(t, S/1000,  label='Susceptible')
plt.plot(t, I/1000, 'r', alpha=0.5, lw=2, label='Infected')
plt.plot(t, R/1000, 'g', alpha=0.5, lw=2, label='Recovered with
immunity')

plt.xlabel('Time /days')
plt.ylabel('Number (1000s)')
plt.legend()

plt.show()

print("R_0 = ", R_0)
print("Gamma = ", gamma)
```



```
R_0 =  0.999
Gamma =  0.2
```

In [3]:
```
# Total population, N.
N = 1000

# Initial number of infected and recovered individuals, I0 and R
0.
I0, R0 = 1, 0

# Everyone else, S0, is susceptible to infection initially.
S0 = N - I0 - R0

# Contact rate, beta, and mean recovery rate, gamma, (in 1/day
s).
beta, gamma = 0.0002, 1/50

# A grid of time points (in days)
t = np.linspace(0, 160, 160)

# The SIR model differential equations.
def deriv(y, t, N, beta, gamma):
    S, I, R = y
    dSdt = -beta * S * I
    dIdt = beta * S * I  - gamma * I
    dRdt = gamma * I
    return dSdt, dIdt, dRdt

# Initial conditions vector
y0 = S0, I0, R0

# Integrate the SIR equations over the time grid, t and the solu
tion
ret = odeint(deriv, y0, t, args=(N, beta, gamma))

S=ret[:,0]
I=ret[:,1]
R=ret[:,2]

R_0 = (beta * S0) / gamma

# Plot the data on three separate curves for S(t), I(t) and R(t)
fig = plt.figure(facecolor='w',figsize=[10,8])
fig
plt.plot(t, S/1000,  label='Susceptible')
plt.plot(t, I/1000, 'r', alpha=0.5, lw=2, label='Infected')
plt.plot(t, R/1000, 'g', alpha=0.5, lw=2, label='Recovered with
immunity')

plt.xlabel('Time /days')
plt.ylabel('Number (1000s)')
plt.legend()

plt.show()

print("R_0 = ", R_0)
print("Gamma = ", gamma)
```



```
R_0 =  9.99
Gamma =  0.02
```

- We can see that if the reproduction number threshold is below one, the curves of susceptible individuals and people who have recovered with immunity don't intersect. Indeed, when the reproduction number threshold is below one, it is because of gamma. In fact, the more the value of gamma is, the less the average duration of the infection is. Consequently, from a certain stage, the average duration of the infection is too short to infect other people (it is when we obtain, by calculation, a reproduction number threshold below one) so that is why the curves don't intersect.
- Otherwise, the curvesintersect (or almost intersect) when the reproduction number threshold is above one. Indeed, the average duration of the infection is long enough to infect other people. Thus, the curve of people who have recovered increases exponentially and the one of susceptible individuals decreases exponentially as well. Besides, we can see the more the average duration of the infection is, the more the curve of infected individuals increases (and then obviously decreases).

Ref: https://scipython.com/book/chapter-8-scipy/additional-examples/the-sir-epidemic-model/ (https://scipython.com/book/chapter-8-scipy/additional-examples/the-sir-epidemic-model/)

The SIER model

The SEIR model is a bit more elaborate: it takes into account three more assumptions than the SIR model, the demography of the population in particular. The total population N(t) evolves therefore during the time t.

Here is the evolution from the SIR model to the SEIR model:

A new sub-population is added: the non-infectious (exposed) infected persons, who are therefore not contagious, represented by the function $E(t)$

which allows to take into account the incubation period (via α the incubation rate) of a disease. Taking the scheme and system from the SIR model, and adding a term $\pm\alpha E(t)$. The birth rate ν of the population is also considered. People are assumed to be born healthy, so we add a term $\nu N(t)$ Finally, we complete with the addition of the mortality rate μ of the population.

Since a person can die whatever his or her state (S,E, I or$R$), and from a cause unrelated to the epidemic, we therefore remove these people from each line (i.e. $-\mu S(t)$ or $\mu E(t)$, or $\mu I(t)$ or $\mu R(t)$ depending on the subpopulation considered).

The SIER model is given by the following system:

$$\begin{aligned}\frac{dS}{dt} &= -\beta SI + \nu N(t) - \mu S\\ \frac{dE}{dt} &= \beta SI - \alpha E - \mu E\\ \frac{dI}{dt} &= \alpha E - \gamma I - \mu I,\\ \frac{dR}{dt} &= \gamma I - \mu R,\end{aligned}$$

Modify the previous code to integrate the SIER model and plot the four curves as a function of time.

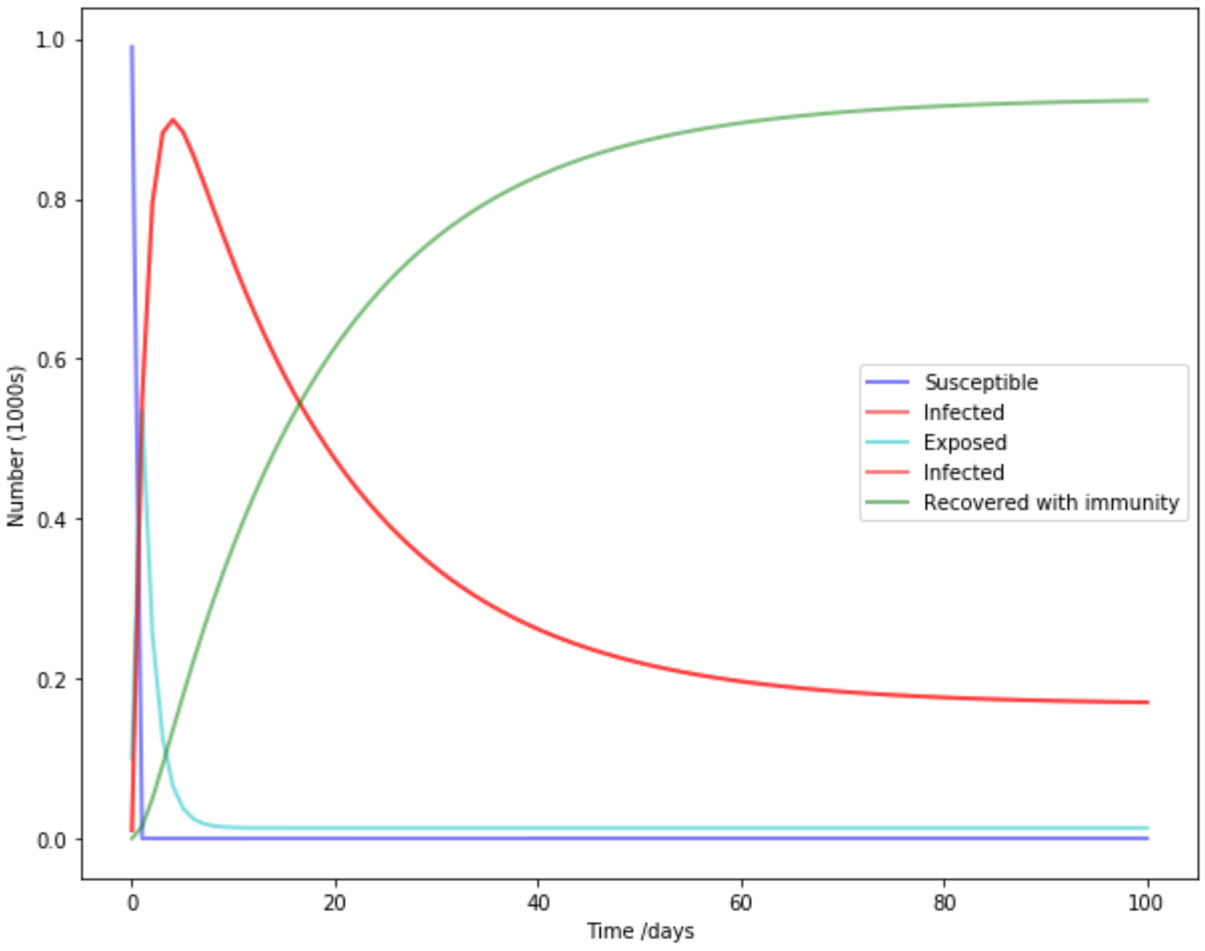
Plot several graphs depending on the value of the reproduction number

$$R_0 = \frac{\alpha}{(\alpha+\mu)}\frac{\beta S(0)}{(\gamma+\mu)}$$

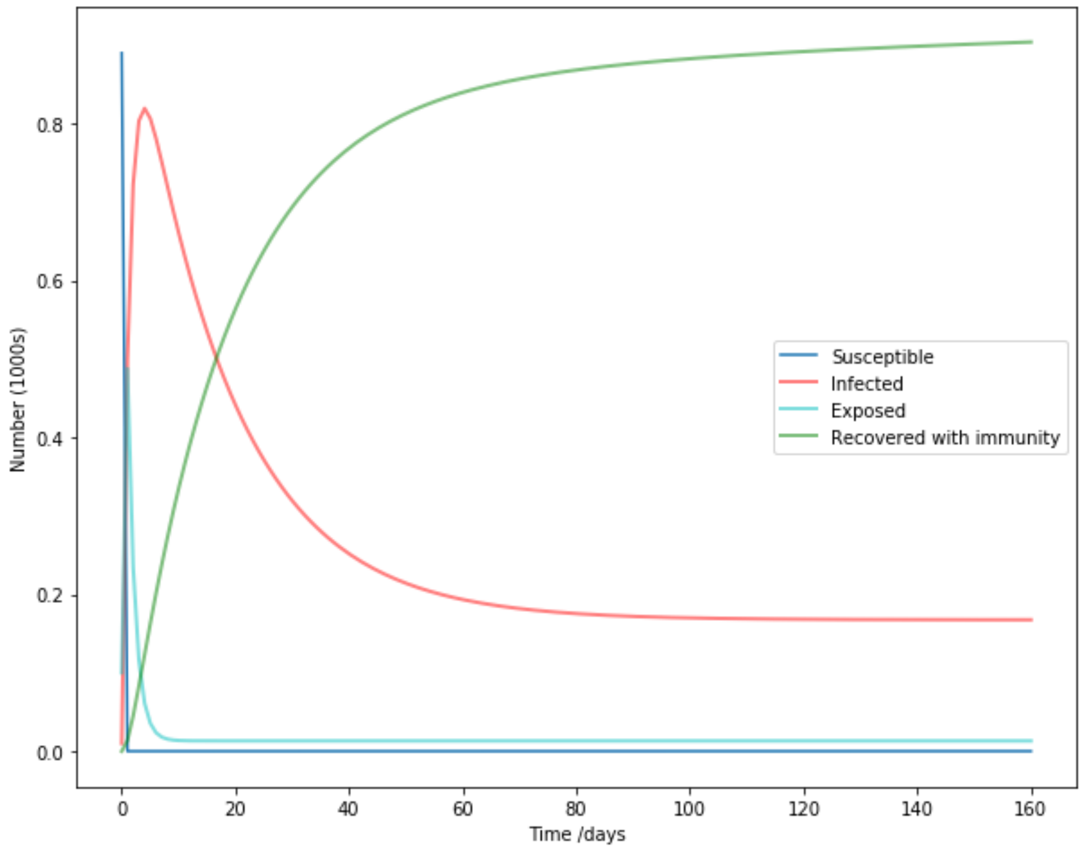
, $R_0 > 1$ and $R_0 < 1$ and varying at least three parameters.

As input datas, you can use: $\alpha = 0.75, \beta = 0.8, \gamma = 0.05, \nu = 0.01, \mu = 0.009$ and $E(0) = 100$, and as previously $N = 1000, I(0) = 10, R(0) = 0$. On each graph, display as a legend the value of R_0 and the values of the modified parameters.

Below is the graph to obtain with those inital values :

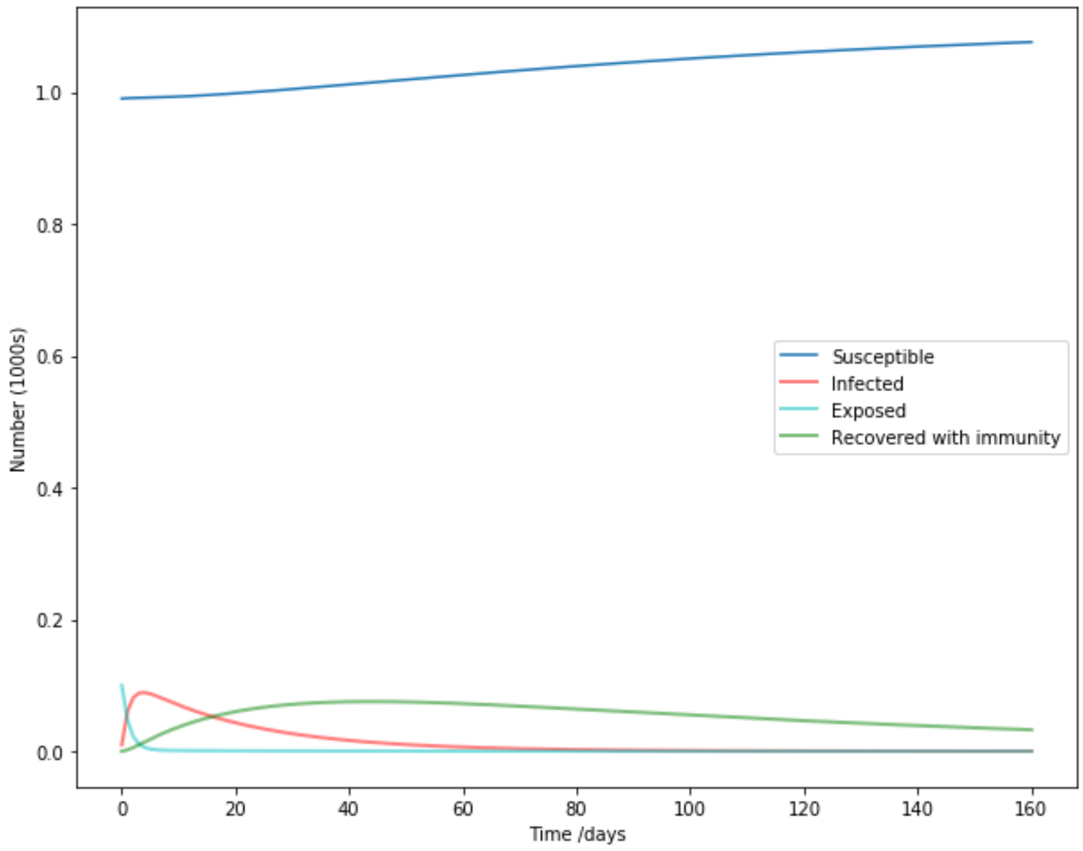


In [4]:
```
# Total population, N.
N = 1000

# Initial number of infected, recovered individuals and infected
persons, who are therefore not contagious, I0, R0 and E0.
I0, R0, E0 = 10, 0, 100

# Everyone else, S0, is susceptible to infection initially.
S0 = N - I0 - R0 - E0

# Incubation rate, alpha, contact rate, beta, mean recovery rat
e, gamma, (in 1/days), birth rate, nu, mortality rate, mu.
alpha, beta, gamma, nu, mu = 0.75, 0.8, 0.05, 0.01, 0.009

# A grid of time points (in days)
t = np.linspace(0, 160, 160)

# The SIER model differential equations.
def deriv(y, t, N, alpha, beta, gamma, nu, mu):
    S, I, E, R = y
    dSdt = -beta * S * I + nu * N - mu * S
    dEdt = beta * S * I - alpha * E - mu * E
    dIdt = alpha * E  - gamma * I - mu * I
    dRdt = gamma * I - mu * R
    return dSdt, dIdt, dEdt, dRdt

# Initial conditions vector
y0 = S0, I0, E0, R0

# Integrate the SIR equations over the time grid, t and the solu
tion
ret = odeint(deriv, y0, t, args=(N, alpha, beta, gamma, nu, mu))

S=ret[:,0]
I=ret[:,1]
E=ret[:,2]
R=ret[:,3]

R_0 = (alpha / (alpha + mu)) * ((beta * S0)/(gamma + mu))

# Plot the data on three separate curves for S(t), I(t) and R(t)
fig = plt.figure(facecolor='w',figsize=[10,8])
fig
plt.plot(t, S/1000,  label='Susceptible')
plt.plot(t, I/1000, 'r', alpha=0.5, lw=2, label='Infected')
plt.plot(t, E/1000, 'c', alpha=0.5, lw=2, label='Exposed')
plt.plot(t, R/1000, 'g', alpha=0.5, lw=2, label='Recovered with
immunity')

plt.xlabel('Time /days')
plt.ylabel('Number (1000s)')
plt.legend()

plt.show()

print("R_0 = ", R_0)
```



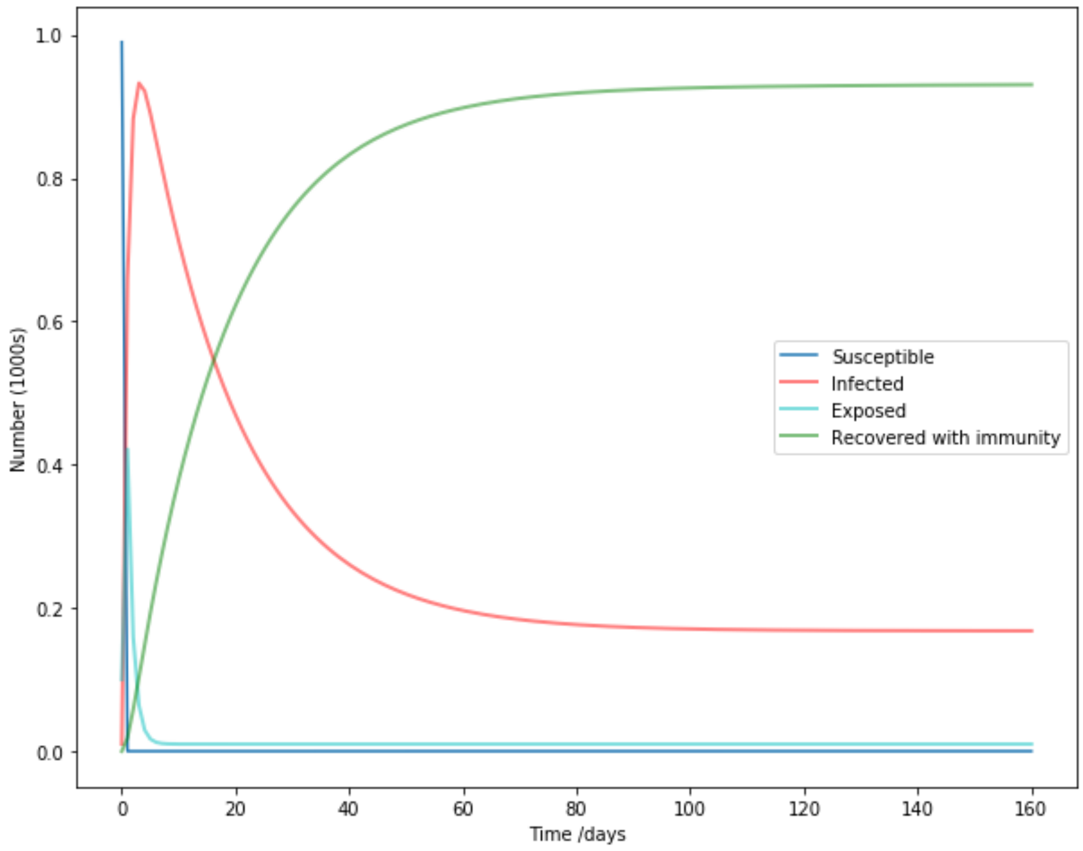
```
R_0 =  11924.700207677362
```

In [5]:
```python
# Total population, N.
N = 1000

# Initial number of infected, recovered individuals and infected
persons, who are therefore not contagious, I0, R0 and E0.
I0, R0, E0 = 10, 0, 100

# Everyone else, S0, is susceptible to infection initially.
S0 = N - I0 - R0

# Incubation rate, alpha, contact rate, beta, mean recovery rat
e, gamma, (in 1/days), birth rate, nu, mortality rate, mu.
alpha, beta, gamma, nu, mu = 0.75, 0.00001, 0.05, 0.01, 0.009

# A grid of time points (in days)
t = np.linspace(0, 160, 160)

# The SIER model differential equations.
def deriv(y, t, N, alpha, beta, gamma, nu, mu):
    S, I, E, R = y
    dSdt = -beta * S * I + nu * N - mu * S
    dEdt = beta * S * I - alpha * E - mu * E
    dIdt = alpha * E  - gamma * I - mu * I
    dRdt = gamma * I - mu * R
    return dSdt, dIdt, dEdt, dRdt

# Initial conditions vector
y0 = S0, I0, E0, R0

# Integrate the SIR equations over the time grid, t and the solu
tion
ret = odeint(deriv, y0, t, args=(N, alpha, beta, gamma, nu, mu))

S=ret[:,0]
I=ret[:,1]
E=ret[:,2]
R=ret[:,3]

R_0 = (alpha / (alpha + mu)) * ((beta * S0)/(gamma + mu))

# Plot the data on three separate curves for S(t), I(t) and R(t)
fig = plt.figure(facecolor='w',figsize=[10,8])
fig
plt.plot(t, S/1000,  label='Susceptible')
plt.plot(t, I/1000, 'r', alpha=0.5, lw=2, label='Infected')
plt.plot(t, E/1000, 'c', alpha=0.5, lw=2, label='Exposed')
plt.plot(t, R/1000, 'g', alpha=0.5, lw=2, label='Recovered with
immunity')

plt.xlabel('Time /days')
plt.ylabel('Number (1000s)')
plt.legend()

plt.show()

print("R_0 = ", R_0)
print("\n")
print("Parameter modified :\n")
print("Beta = ", beta)
```



```
R_0 =  0.1658069270449521

Parameter modified :

Beta =  1e-05
```

In [6]:
```
# Total population, N.
N = 1000

# Initial number of infected, recovered individuals and infected
persons, who are therefore not contagious, I0, R0 and E0.
I0, R0, E0 = 10, 0, 100

# Everyone else, S0, is susceptible to infection initially.
S0 = N - I0 - R0

# Incubation rate, alpha, contact rate, beta, mean recovery rat
e, gamma, (in 1/days), birth rate, nu, mortality rate, mu.
alpha, beta, gamma, nu, mu = 1, 0.8, 0.05, 0.01, 0.009

# A grid of time points (in days)
t = np.linspace(0, 160, 160)

# The SIER model differential equations.
def deriv(y, t, N, alpha, beta, gamma, nu, mu):
    S, I, E, R = y
    dSdt = -beta * S * I + nu * N - mu * S
    dEdt = beta * S * I - alpha * E - mu * E
    dIdt = alpha * E  - gamma * I - mu * I
    dRdt = gamma * I - mu * R
    return dSdt, dIdt, dEdt, dRdt

# Initial conditions vector
y0 = S0, I0, E0, R0

# Integrate the SIR equations over the time grid, t and the solu
tion
ret = odeint(deriv, y0, t, args=(N, alpha, beta, gamma, nu, mu))

S=ret[:,0]
I=ret[:,1]
E=ret[:,2]
R=ret[:,3]

R_0 = (alpha / (alpha + mu)) * ((beta * S0)/(gamma + mu))

# Plot the data on three separate curves for S(t), I(t) and R(t)
fig = plt.figure(facecolor='w',figsize=[10,8])
fig
plt.plot(t, S/1000,  label='Susceptible')
plt.plot(t, I/1000, 'r', alpha=0.5, lw=2, label='Infected')
plt.plot(t, E/1000, 'c', alpha=0.5, lw=2, label='Exposed')
plt.plot(t, R/1000, 'g', alpha=0.5, lw=2, label='Recovered with
immunity')

plt.xlabel('Time /days')
plt.ylabel('Number (1000s)')
plt.legend()

plt.show()

print("R_0 = ", R_0)
print("\n")
print("Parameter modified :\n")
print("Alpha = ", alpha)
```

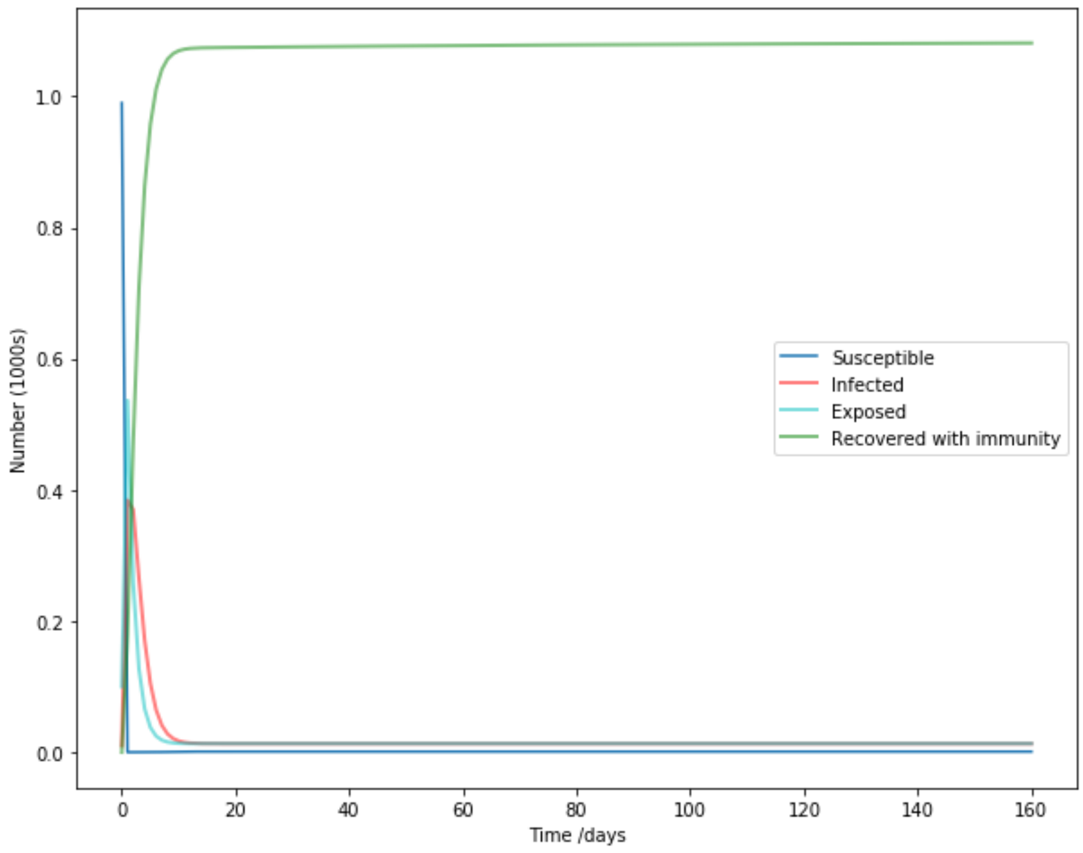


```
R_0 =  13303.99287766038
```

```
Parameter modified :
```

```
Alpha =  1
```

In [7]:
```
# Total population, N.
N = 1000

# Initial number of infected, recovered individuals and infected
persons, who are therefore not contagious, I0, R0 and E0.
I0, R0, E0 = 10, 0, 100

# Everyone else, S0, is susceptible to infection initially.
S0 = N - I0 - R0

# Incubation rate, alpha, contact rate, beta, mean recovery rat
e, gamma, (in 1/days), birth rate, nu, mortality rate, mu.
alpha, beta, gamma, nu, mu = 0.75, 0.8, 0.75, 0.01, 0.009

# A grid of time points (in days)
t = np.linspace(0, 160, 160)

# The SIER model differential equations.
def deriv(y, t, N, alpha, beta, gamma, nu, mu):
    S, I, E, R = y
    dSdt = -beta * S * I + nu * N - mu * S
    dEdt = beta * S * I - alpha * E - mu * E
    dIdt = alpha * E  - gamma * I - mu * I
    dRdt = gamma * I - mu * R
    return dSdt, dIdt, dEdt, dRdt

# Initial conditions vector
y0 = S0, I0, E0, R0

# Integrate the SIR equations over the time grid, t and the solu
tion
ret = odeint(deriv, y0, t, args=(N, alpha, beta, gamma, nu, mu))

S=ret[:,0]
I=ret[:,1]
E=ret[:,2]
R=ret[:,3]

R_0 = (alpha / (alpha + mu)) * ((beta * S0)/(gamma + mu))

# Plot the data on three separate curves for S(t), I(t) and R(t)
fig = plt.figure(facecolor='w',figsize=[10,8])
fig
plt.plot(t, S/1000,  label='Susceptible')
plt.plot(t, I/1000, 'r', alpha=0.5, lw=2, label='Infected')
plt.plot(t, E/1000, 'c', alpha=0.5, lw=2, label='Exposed')
plt.plot(t, R/1000, 'g', alpha=0.5, lw=2, label='Recovered with
immunity')

plt.xlabel('Time /days')
plt.ylabel('Number (1000s)')
plt.legend()

plt.show()

print("R_0 = ", R_0)
print("\n")
print("Parameter modified :\n")
print("Gamma = ", gamma)
```



```
R_0 =  1031.1050008592542
```

```
Parameter modified :
```

```
Gamma =  0.75
```