Computational Environments and Toolchains

Topic 04 : Implementing Models

Lecture 03 : SIR Models

Kieran Murphy and David Power

Computing and Mathematics, SETU (Waterford).
(kieran.murphy@setu.ie, david.power@setu.ie)

Autumn Semester, 2025/26

Outline

- Types of epidemic models: Component/Stochastic/Agent-based models
- SIR and related models
- Parameters dependence

Outline

1. Introduction 2

2. SIR 4
2.1. Parameter Dependence 7
2.2. Python Implementation 15

3. SIS 26
3.1. Parameter Dependence 29

4. SIRD 31
4.1. Parameter Dependence 34

Epidemic Modeling

Epidemic models describe and predict the spread of infectious diseases within a population.

Compartmental Models

Divide the population into distinct “compartments” representing different stages of the disease. Common compartments include Susceptible, Infectious, Recovered, and sometimes Exposed or Dead.

Stochastic models

Incorporate randomness, recognising that disease transmission is not purely deterministic and that random events (e.g., super-spreader events) can affect the course of an epidemic.

Examples: Agent-based models, Monte Carlo simulations, Markov chains.

Network Models

Network models represent individuals as nodes and interactions as edges in a network, simulating disease spread through social or spatial contacts.

`en.wikipedia.org/wiki/Compartmental_models_in_epidemiology`

Outline

1. Introduction 2

2. SIR 4
2.1. Parameter Dependence 7
2.2. Python Implementation 15

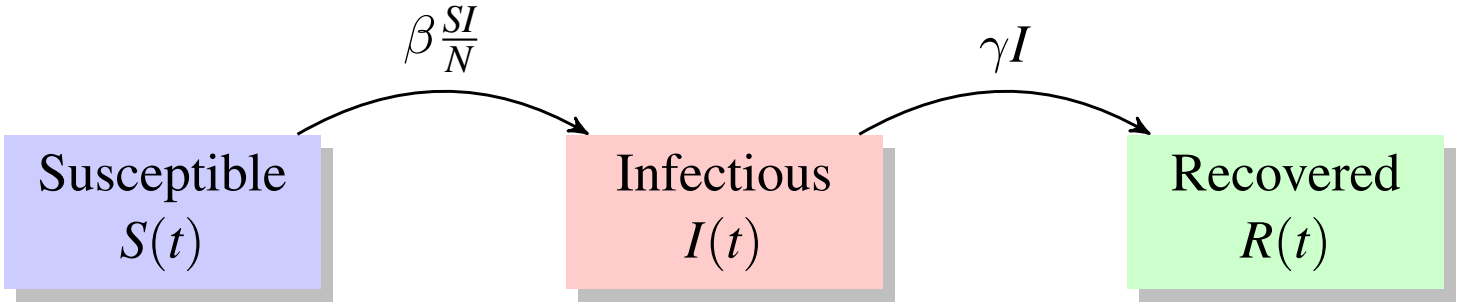
3. SIS 26
3.1. Parameter Dependence 29

4. SIRD 31
4.1. Parameter Dependence 34

SIR Model

The population of N individuals is divided into three compartments:

- S individuals susceptible to be infected;
- I individuals infected;
- R individuals recovered from the disease (and now have acquired immunity to it).

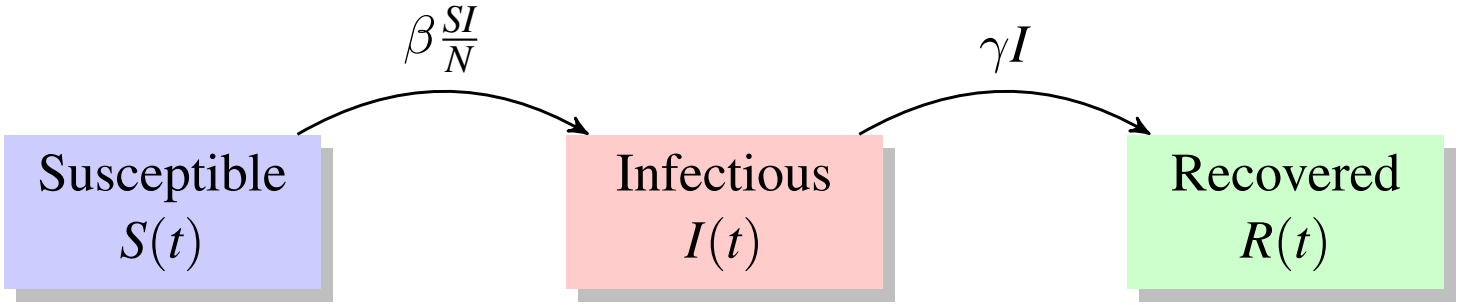


- Individuals in S can become, rate β, infected after positive contact with an I individual. The number of contacts is SI/N.
- They develop immunity to the disease, at a γ cure rate, so they leave I compartment.

---

scientific-python.readthedocs.io/en/latest/notebooks_rst/3_Ordinary_Differential_Equations/02_Examples/Epidemic_model_SIR.html

SIR Model Equations



- $\beta > 0$, the rate of contraction of the disease (transmission parameter).
- $\gamma > 0$, mean recovery rate

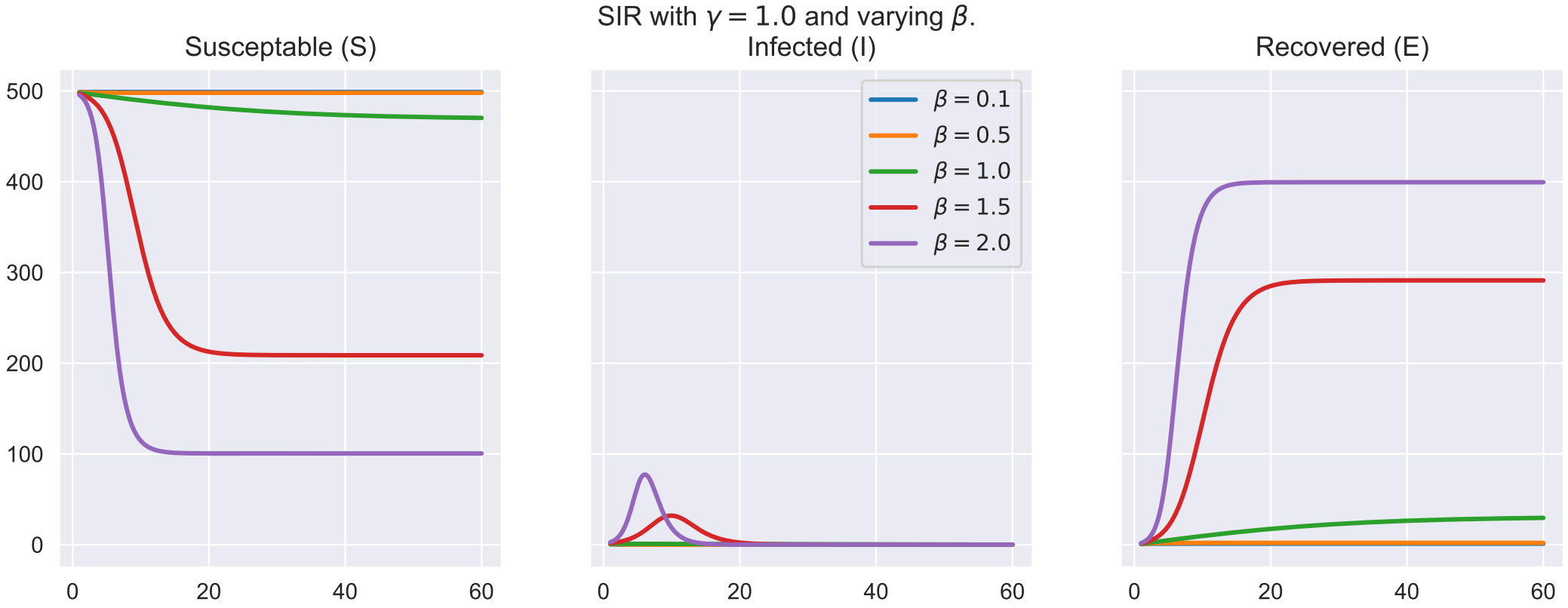
$$\left.\begin{aligned} \frac{dS}{dt} &= -\beta SI/N \\ \frac{dI}{dt} &= \beta SI/N - \gamma I \\ \frac{dR}{dt} &= \gamma I \end{aligned}\right\} \implies \begin{aligned} N &= S + I + R \\ \frac{dN}{dt} &= 0 \end{aligned}$$

Assume disease spread is fast time scale so can ignore natural births/deaths/etc.

Transmission rate, β

$\beta > 0$: the rate of contraction of the disease (transmission parameter).

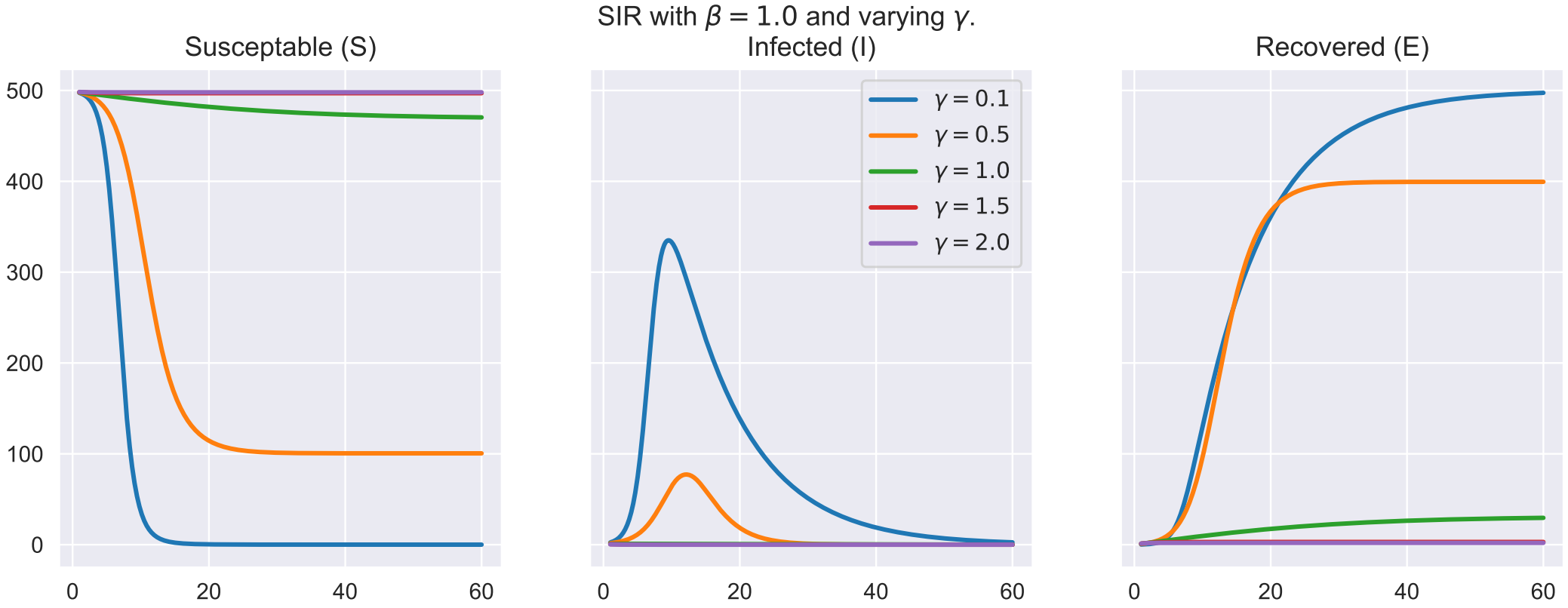
- Greater β reflects a more contagious infection $\implies$ (in the limit) lower S and higher R.



Recovery rate, γ

$\gamma > 0$: mean recovery rate

- Greater γ reflects shorter infectious period $\implies$ (in the limit) higher S and lower R.



Basic Reproduction Number, R_0 I

R_0 is a critical concept in epidemiology that helps to quantify how contagious or transmissible an infectious disease is within a susceptible population. In the context of the SIR model,

$$R_0 = \frac{\beta}{\gamma}$$

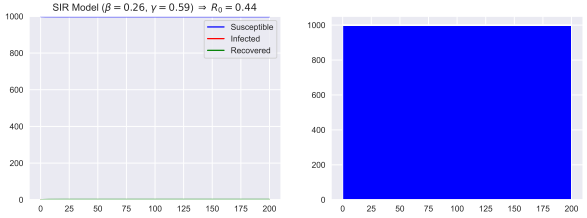
is defined as the average number of secondary infections generated by a single infectious individual in a *fully susceptible population*.

The value of R_0 indicates how the disease will spread in the early stages of an outbreak:

$$R_0 \begin{cases} < 1 & \text{Each infected individual infects fewer than one person on average, meaning the disease will eventually die out.} \\ = 1 & \text{The disease will be endemic, maintaining a steady state without growing or dying out.} \\ > 1 & \text{Each infected person, on average, transmits the infection to more than one other person, leading to an epidemic (the infection will spread).} \end{cases}$$

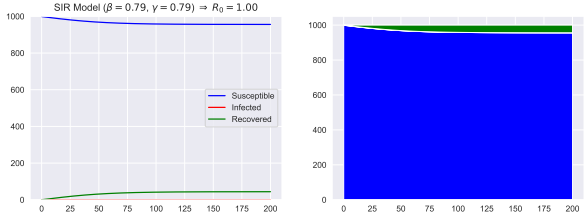
Basic Reproduction Number, R_0 II

$R_0 < 1$



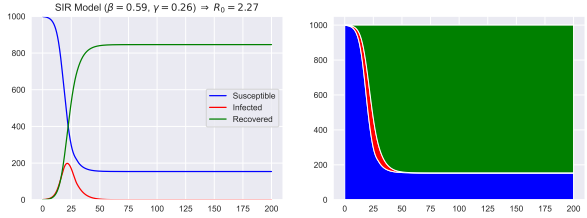
infects fewer than one person on average, meaning the disease will eventually die out.

$R_0 = 1$



disease will be endemic, maintaining a steady state without growing or dying out.

$R_0 > 1$



infects more than one other person, leading to an epidemic (the infection will spread).

Basic Reproduction Number, R_0 III

Factors Influencing, R_0

- **Infectious period:**
  Longer infectious periods (i.e., lower γ) increase R_0.
- **Transmission rate:**
  Higher contact rates or more effective transmission (i.e., higher β) increase R_0.

Limitations of R_0

- It assumes a fully susceptible population, which is rarely the case after an outbreak has begun.
- R_0 can vary across different environments or populations. (Think urban vs. rural.)
- Interventions like quarantine or social distancing are not reflected in the R_0 value.

`en.wikipedia.org/wiki/Basic_reproduction_number`

Herd Immunity Threshold, H

The basic reproduction number also helps determine the **herd immunity threshold**:

$$H = 1 - \frac{1}{R_0}$$

This is the proportion of the population that must be immune (either through recovery or vaccination) to prevent the disease from spreading further.

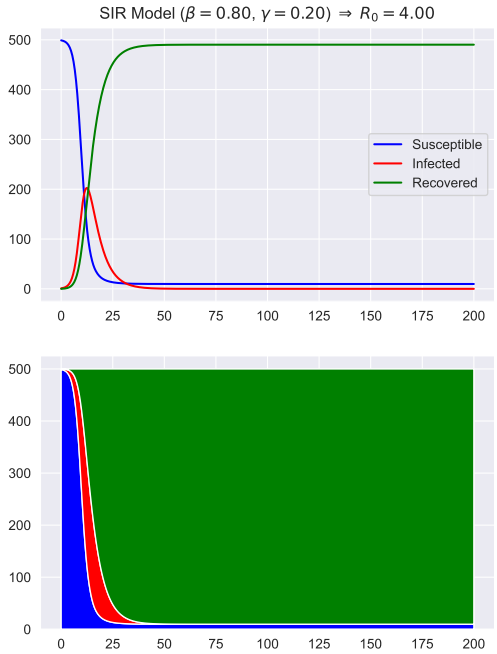
Higher R_0 values require a larger immune proportion in the population to achieve herd immunity.

	R_0	H	
Measles	12–18	92%–94%	(a very high threshold, which is why high vaccination coverage is critical)
Mumps	10–12	90%–92%	
COVID-19	2–3	50%–67%	(initial estimate, current is 2.9–9.5)
Flu	≈1.3	≈23%	
Ebola	1.3–2.0	23%–50%	

Example $\beta = 0.8, \gamma = 0.2 \implies R_0 = 4, H = 0.75$ II

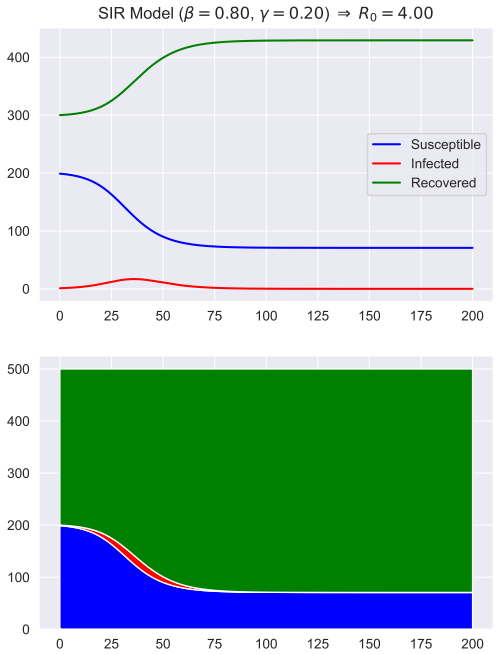
$H = 0.75\%$ so in a population of 500, this corresponds to 375 immune.

0% immune



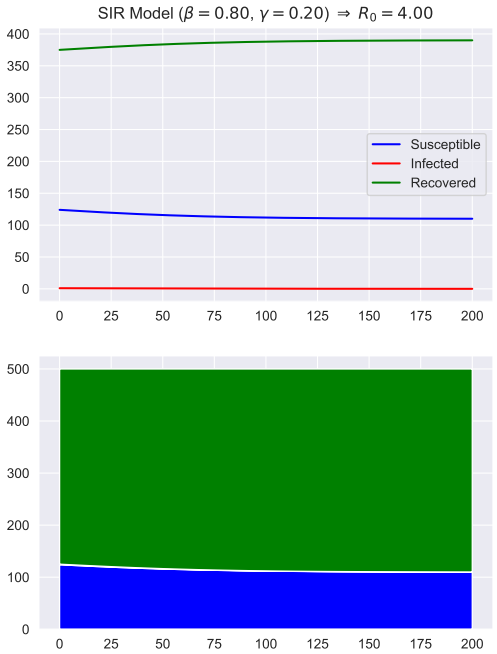
At outbreak there is no immune individuals and $R_0 > 1$, so disease spreads unchecked.

60% immune



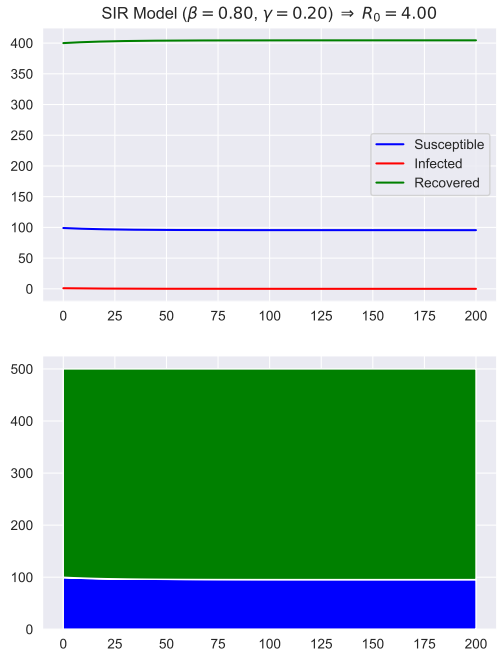
At outbreak, population has (300 individuals) 60% immune, just below threshold. Disease spread is more limited

75% immune



At outbreak, population has (375 individuals) 75% immune, at threshold. Disease outbreak does not occur (even with $R_0 > 1$)

80% immune



At outbreak, population has (400 individuals) 80% immune, above threshold. Disease outbreak does not occur (even with $R_0 > 1$)

Achieving Herd Immunity Through Vaccination

Vaccination is the safest and most effective way to achieve herd immunity, especially for diseases with high R_0. The proportion of the population that needs to be vaccinated to achieve herd immunity is:

$$V = \frac{H}{E} = \frac{1 - \frac{1}{R_0}}{E}$$

where

- V is the vaccination coverage needed for herd immunity,
- E is the **vaccine efficacy** (the percentage of vaccinated people who actually develop immunity).

Example

For example, if a disease has $R_0 = 3$ and a vaccine efficacy of 90%, then:

$$V = \frac{1 - \frac{1}{3}}{0.9} \approx 74\%$$

So approximately 74% of the population would need to be vaccinated to stop the disease from spreading.

Python Implementation — Setup

- Load our standard scientific python modules `numpy`, `matplotlib`, and `seaborn`.
- From `scipy` import `integrate` for solving differential equations and `optimise` for curve fitting/parameter estimation.
- Load `pandas` to read CSV data files.
- Local `ipywidgets` to control generation of results based on varying parameter values.

Setup

```
1
1  import numpy as np
2  import matplotlib.pyplot as plt
3  import seaborn as sns
4  sns.set_style('darkgrid')
5
6  from scipy import integrate, optimize
7
8  import pandas as pd              # to deal with CSV files
9  import ipywidgets as ipw         # interactive plots
```

Python Implementation — Define ODE System

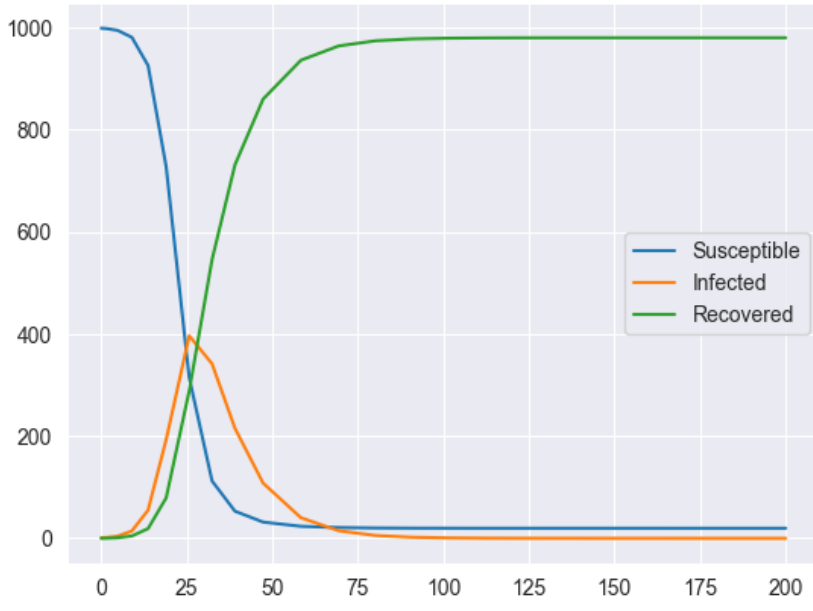
RHS

```
2
1  def rhs_sir(t, state, beta, gamma):
2      S, I, R = state
3
4      N = S + I + R
5      dS = - beta * S * I / N
6      dI = + beta * S * I / N - gamma * I
7      dR = gamma * I
8
9      return np.array([dS,dI,dR])
10
11
12 beta = 0.4
13 gamma = 0.1
14
15 rhs_sir(0, [1_000,1,0], beta, gamma)
```

$$\begin{aligned} \frac{dS}{dt} &= -\beta SI/N \\ \frac{dI}{dt} &= \beta SI/N - \gamma I \\ \frac{dR}{dt} &= \gamma I \end{aligned}$$

Python Implementation — Integrate

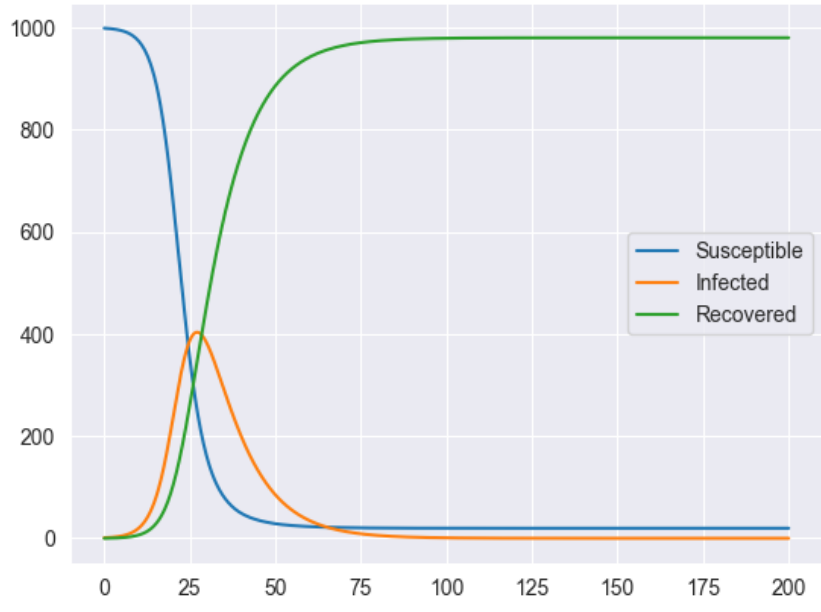
- We will use `solve_ivp` instead of the older `odeint` to integrate an ODE.
- Unlike `odeint`, we only specify the independent variable span (min, max), and `solve_ivp` picks enough points to get solution to sufficient accuracy.
- But visually we might want more points.




```
3
1 T_MAX = 200
2 N = 1_000
3 I0 = 1
4 ic = [N-I0, I0, 0]
5
6
7 sol = integrate.solve_ivp(rhs_sir, (0,T_MAX), ic, args=(beta, gamma))
8 t = sol.t
9 S, I, R = sol.y
```

Python Implementation — Integrate with Interpolation

- To get smoother plots, we use optional term `t_eval` to specify at which points to generate the state.
- This parameter will also be useful when comparing solutions from multiple calls to `solve_ivp`.
- Use `np.linspace`




```
4
1  T_MAX = 200
2  N = 1_000
3  I0 = 1
4  ic = [N-I0, I0, 0]
5
6  t_eval = np.linspace(0, T_MAX, 1000)
7  sol = integrate.solve_ivp(rhs_sir, (0,T_MAX), ic, args=(beta, gamma), t_eval=t_eval)
8  t = sol.t
9  S, I, R = sol.y
```

Python Implementation — Visualisation

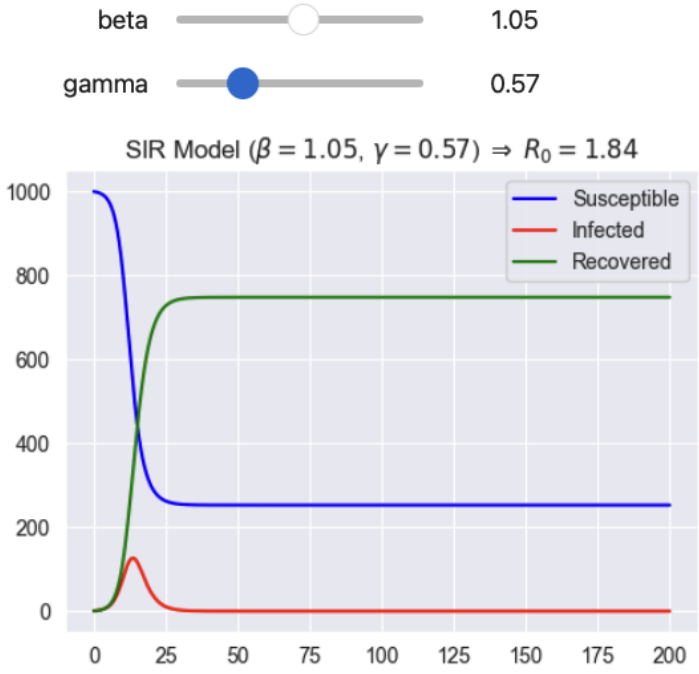
```
5
1  def update_sir(beta=0.2, gamma=0.1):
2
3      sol = integrate.solve_ivp(rhs_sir, (0,T_MAX), ic, args=(beta, gamma), t_eval=t_eval)
4      t, (S, I, R) = sol.t, sol.y
5      N = sum(ic)
6
7      fig = plt.figure(2, figsize=(12,4))
8      axs = fig.subplots(1, 2, sharex=0)
9
10     colors=['blue', 'red', 'green']
11     axs[0].plot(t, S, colors[0], label = "Susceptible")
12     axs[0].plot(t, I, colors[1], label = 'Infected')
13     axs[0].plot(t, R, colors[2], label = 'Recovered')
14     axs[0].legend()
15
16     axs[1].stackplot(t, S, I, R, colors=colors)
17     R_0 = np.nan if gamma==0 else beta/gamma
18     axs[0].set_title(r"SIR Model ($\beta=%.2f$, $\gamma=%.2f$) $\Rightarrow$ $R_0=%.2f$"
19                      % (beta,gamma,R_0))
20     plt.show()
```

Implement a function to draw the required plots

Python Implementation — Interactive

We can control values of parameters β and γ using sliders ...

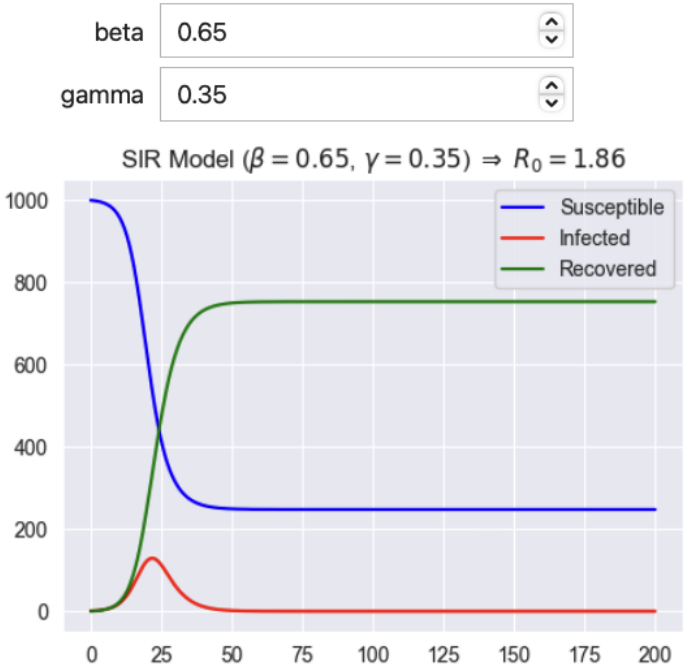
```
6
1  N = 1_000
2  I0 = 1
3  ic = [N-I0, I0, 0]
4  t_eval = np.linspace(0, T_MAX, 1000)
5
6  interactive_plot = ipw.interactive(update_sir,
7      beta=ipw.FloatSlider(0.0, min=0.0, max=2.0, step=0.01),
8      gamma=ipw.FloatSlider(0.0, min=0.0, max=2.0, step=0.01),
9  )
10 output = interactive_plot.children[-1]
11 output.layout.height = '350px'
12 interactive_plot
```



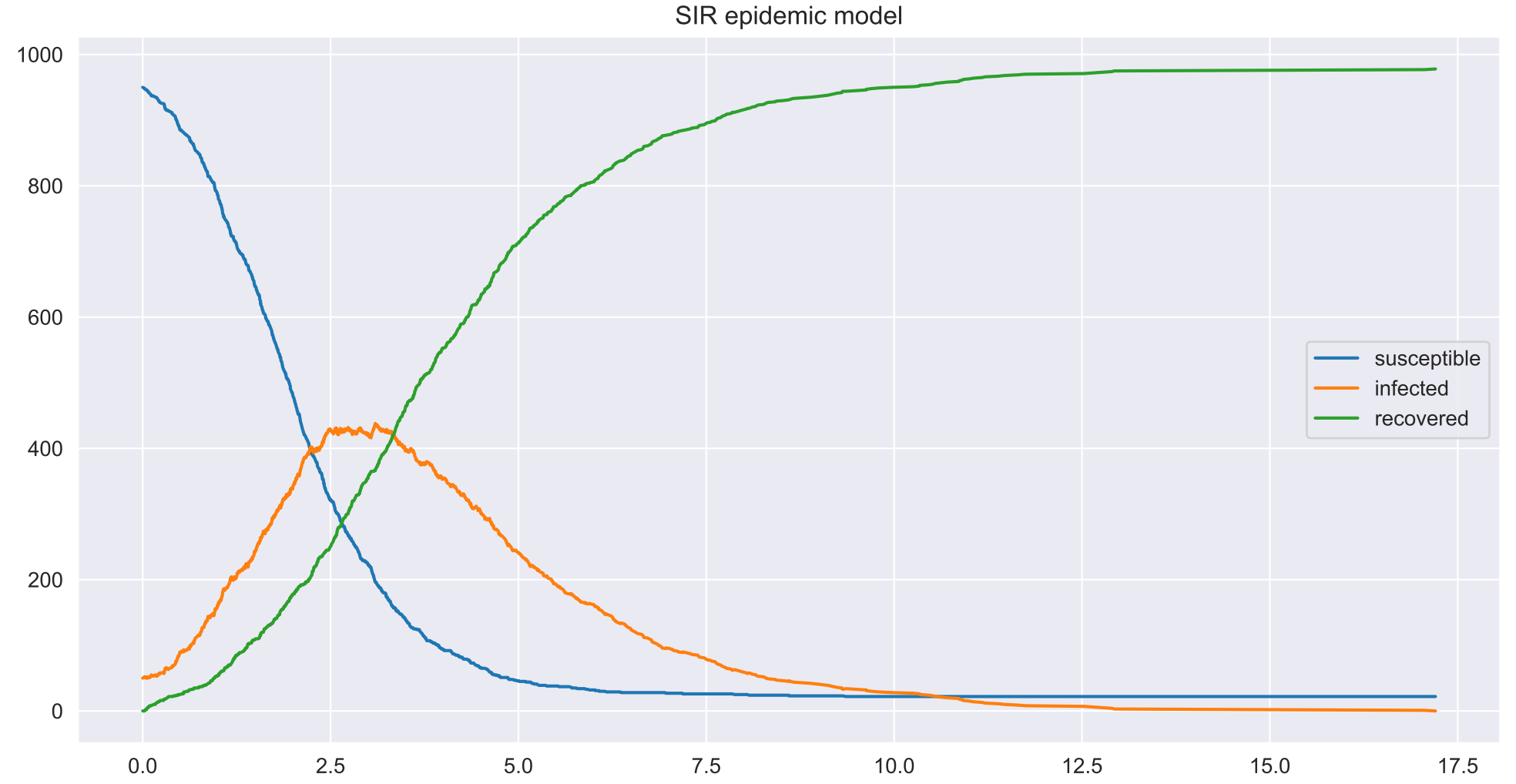
Python Implementation — Interactive II

Colab can be too slow so switch to `FloatText` widgets ...

```
7
1  N = 1_000
2  I0 = 1
3  ic = [N-I0, I0, 0]
4  t_eval = np.linspace(0, T_MAX, 1000)
5
6  interactive_plot = ipw.interactive(update_sir,
7      beta=ipw.FloatText(value=0.0,step=0.05),
8      gamma=ipw.FloatText(value=0.0,step=0.05),
9  )
10 output = interactive_plot.children[-1]
11 output.layout.height = '350px'
12 interactive_plot
```



Python Implementation — Parameter Fitting



Python Implementation — Parameter Fitting II

Load Data . . .

```
8
1 df = pd.read_csv("data.csv")
2 df.head()
```

	t	S	I	R
0	0.000000	950	50	0
1	0.018729	949	51	0
2	0.024190	948	52	0
3	0.030419	948	51	1
4	0.031453	947	52	1

Extract t and y values. . .

```
9
1 df = pd.read_csv("data.csv")
2
3 t_obs = df.t.values
4
5 y_obs = df[ ['S', 'I', 'R'] ].values
6 y_obs
```

```
array([[950, 50,  0],
       [949, 51,  0],
       [948, 52,  0],
       ...,
       [ 22,  2, 976],
       [ 22,  1, 977],
       [ 22,  0, 978]])
```

Python Implementation — Parameter Fitting III

- Use `optimize.minimize` instead of `optimise.curve_fit` as have more control over objective function.
- We want to minimise, the sum of squares of differences

$$\sum_k \left[y_{\text{model},k} - y_{\text{obs},k} \right]^2$$

```
ic = y_obs[0]
T_MAX = t_obs[-1]

def f(x):

    beta, gamma = x

    sol = integrate.solve_ivp(rhs_sir, (0,T_MAX), ic, args=(beta, gamma), t_eval=t_obs)
    return np.linalg.norm(sol.y - y_obs.T)

sol = optimize.minimize(f, [0.5, 0.5], bounds=((0,3),(0,3)) )
sol
```

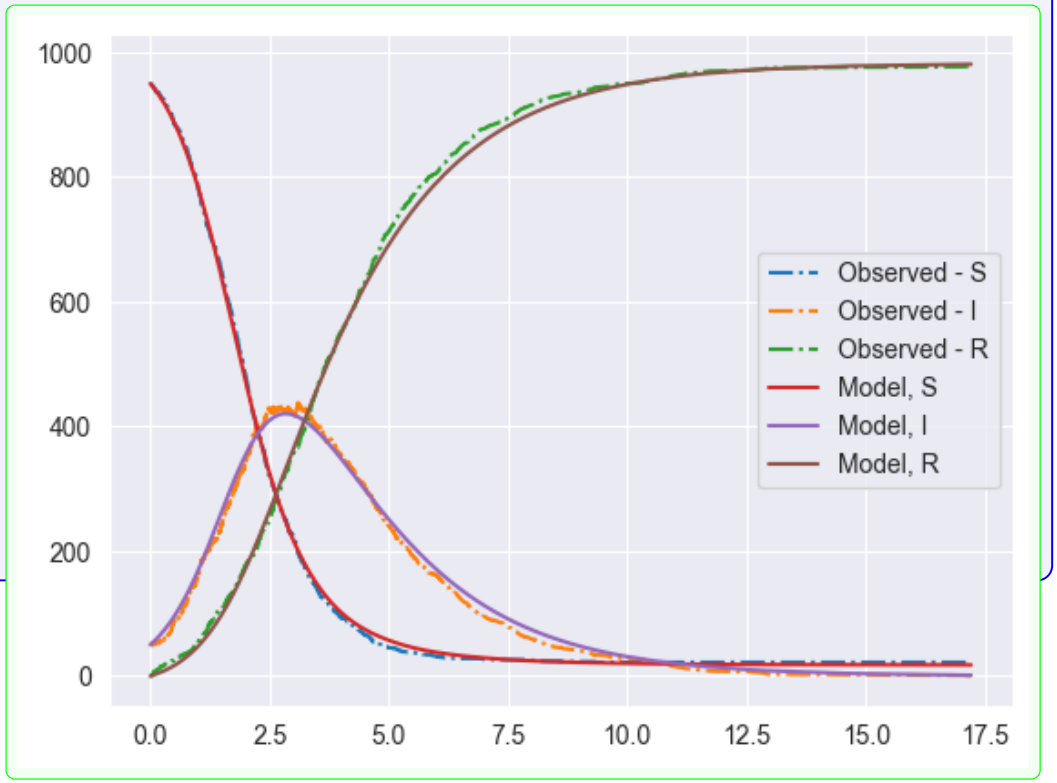
```
 message: CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
 success: True
  status: 0
     fun: 754.0829736360038
       x: [ 1.923e+00 4.758e-01]
     nit: 9
     jac: [-2.246e-01 -1.756e-01]
    nfev: 45
    njev: 15
hess_inv: <2x2 LbfgsInvHessProduct with dtype=float64>
```

Python Implementation — Parameter Fitting IV

```
11 sol_ivp = integrate.solve_ivp(rhs_sir, (0,t_obs[-1]), ic, args=sol.x, t_eval=t_obs)
   t = sol_ivp.t

   plt.plot(t, df.S, '-.', label='Observed - S')
   plt.plot(t, df.I, '-.', label='Observed - I')
   plt.plot(t, df.R, '-.', label='Observed - R')

   plt.plot(t, sol_ivp.y[0], label='Model, S')
   plt.plot(t, sol_ivp.y[1], label='Model, I')
   plt.plot(t, sol_ivp.y[2], label='Model, R')
   plt.legend()
   plt.show()
```



Outline

1. Introduction 2

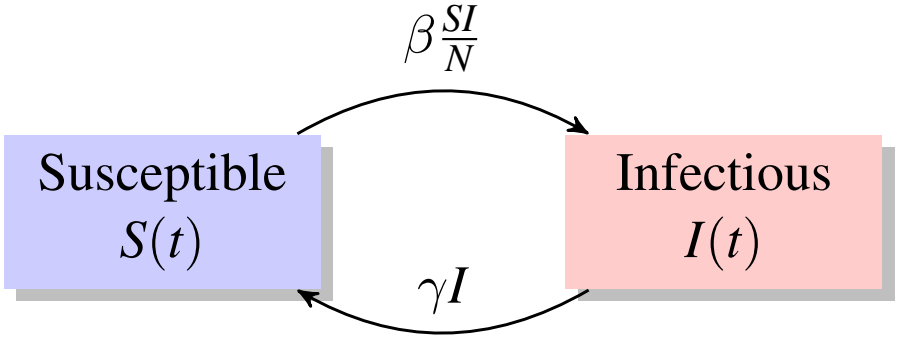
2. SIR 4
2.1. Parameter Dependence 7
2.2. Python Implementation 15

3. SIS 26
3.1. Parameter Dependence 29

4. SIRD 31
4.1. Parameter Dependence 34

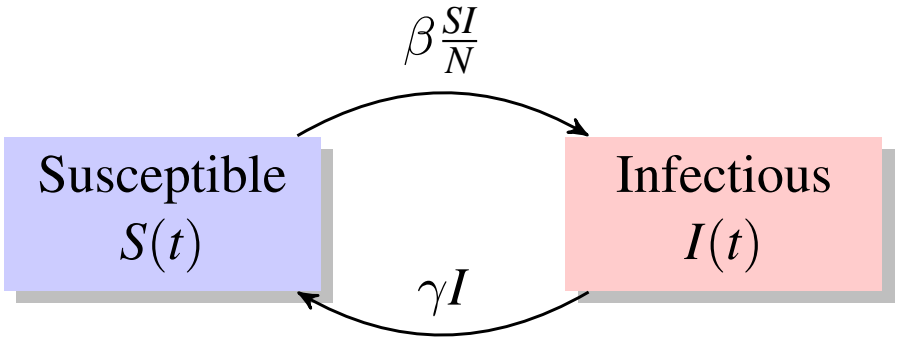
SIS Model

The SIS model is a compartmental model in epidemiology similar to the SIR model, but it's designed to describe diseases where individuals do not acquire long-term immunity after infection, such as the common cold and the flu.
In the SIS model, individuals move between the Susceptible (S) and Infected (I) states, without transitioning to a recovered state. After recovery, individuals return to the susceptible pool, capable of being reinfected.



- Susceptible individuals can become, rate β, infected after positive contact with an I individual. The number of contacts is SI/N.
- Infected individuals recover, at rate γ, and can be reinfected.

SIS Model Equations



- $\beta > 0$, the rate of contraction of the disease (transmission parameter).
- $\gamma > 0$, mean recovery rate

$$\left.\begin{aligned} \frac{dS}{dt} &= -\beta SI/N + \gamma I \\ \frac{dI}{dt} &= \beta SI/N - \gamma I \end{aligned}\right\} \quad \Longrightarrow \quad \begin{aligned} N &= S + I \\ \frac{dN}{dt} &= 0 \end{aligned}$$

Assume disease spread is fast time scale so can ignore natural births/deaths/etc.

Basic Reproduction Number, R_0

The basic reproduction number, R_0, in the SIS model is calculated similarly to the SIR model:

$$R_0 = \frac{\beta}{\gamma}$$

The value of R_0 determines how the disease will spread:

$R_0 \leq 1$: The infection dies out over time, and eventually, there will be no infectious individuals in the population.

$$\lim_{t \to \infty} I(t) = 0$$

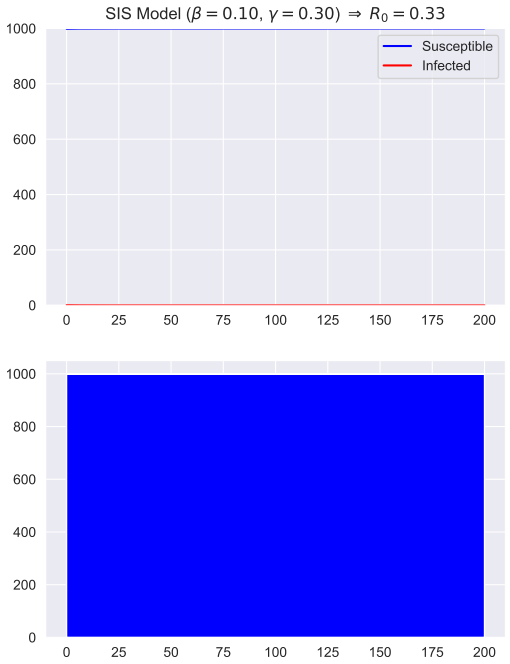
$R_0 > 1$: The infection will spread and reach an endemic equilibrium, where a constant proportion of the population remains infected.

$$\lim_{t \to \infty} I(t) = \left(1 - \frac{1}{R_0}\right) N$$

Basic Reproduction Number, R_0 II

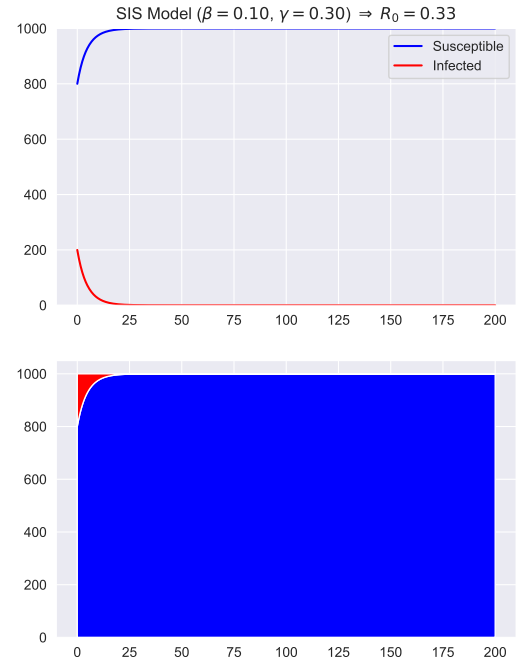
$\longleftarrow R_0 \leq 1 \longrightarrow$

$I_0 = 1$



infects fewer than one person on average, meaning the disease will eventually die out.

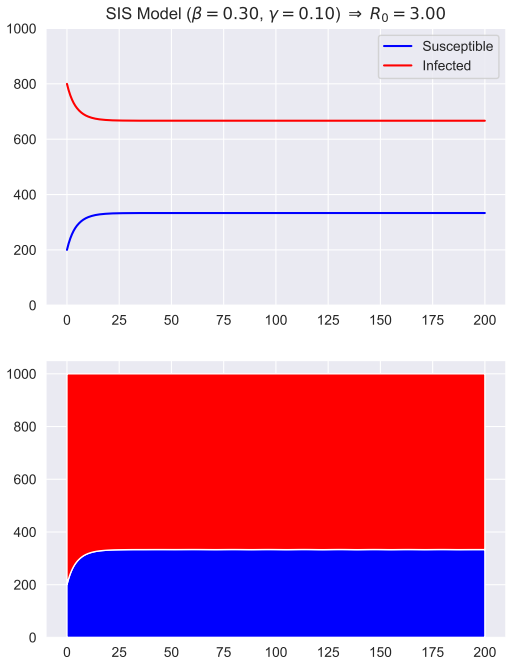
$I_0 = 200$



the disease will eventually die out, even if initial pocket of infected

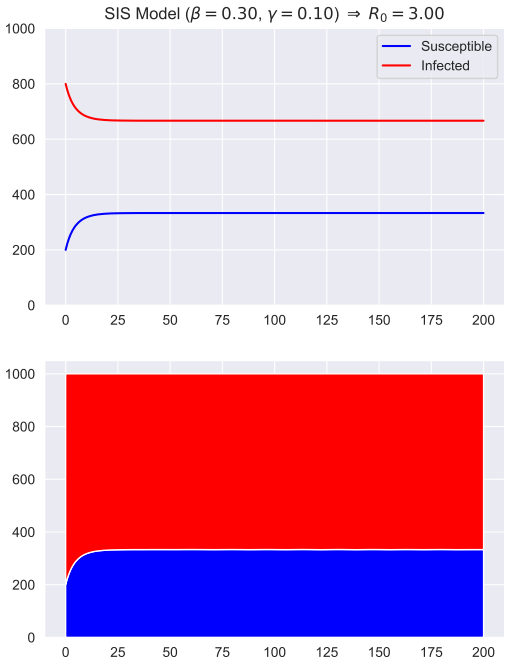
$\longleftarrow R_0 > 1 \longrightarrow$

$I_0 = 1$



Number of infected trends towards limit of $\left(1 - 1/R_0\right) N$.

$I_0 = 800$



Number of infected trends towards limit of $\left(1 - 1/R_0\right) N$.

Outline

1. Introduction 2

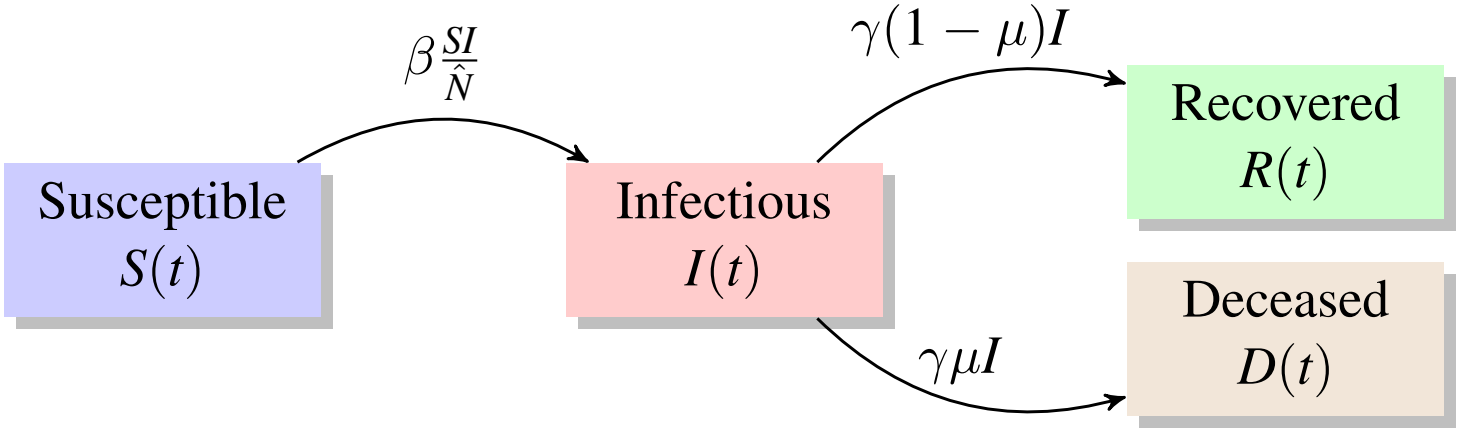
2. SIR 4
2.1. Parameter Dependence 7
2.2. Python Implementation 15

3. SIS 26
3.1. Parameter Dependence 29

4. SIRD 31
4.1. Parameter Dependence 34

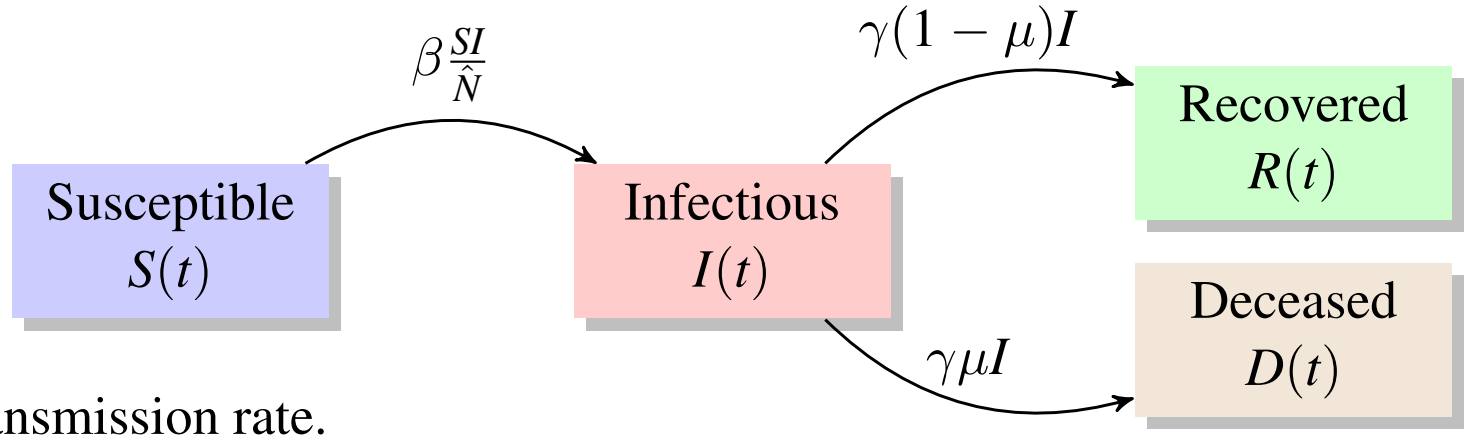
SIRD Model

The SIRD model is an extension of the basic SIR model that incorporates an additional Death (D) compartment to represent individuals who die from the disease. This modification makes the SIRD model more realistic for diseases with significant mortality rates, such as COVID-19, Ebola, and other severe infections.



- Susceptible individuals can become, rate β, infected after positive contact with an I individual. The number of contacts is $SI/\hat{N}$, where $\hat{N} = S + I + R$, i.e. no necrophilism.
- Infected individuals can recover and develop immunity to the disease, at rate γ and move to Recovered, or die at a rate μ and move to Deceased.

SIRD Model Equations



- $\beta > 0$, transmission rate.
- $\gamma(1-\mu) > 0$, mean recovery rate
- $\gamma\mu > 0$, mean mortality rate

$$\left.\begin{aligned} \frac{dS}{dt} &= -\beta SI/\hat{N} \\ \frac{dI}{dt} &= \beta SI/\hat{N} - \gamma I \\ \frac{dR}{dt} &= \gamma(1-\mu)I \\ \frac{dD}{dt} &= \gamma\mu I \end{aligned}\right\} \Longrightarrow \begin{aligned} N &= S + I + R + D \\ \frac{dN}{dt} &= 0 \\ \hat{N} &= S + I + R \end{aligned}$$

Assume disease spread is fast time scale so can ignore natural births/deaths/etc.

Basic Reproduction Number, R_0

The basic reproduction number, R_0, in the SIRD model is:

$$R_0 = \frac{\beta}{\gamma}$$

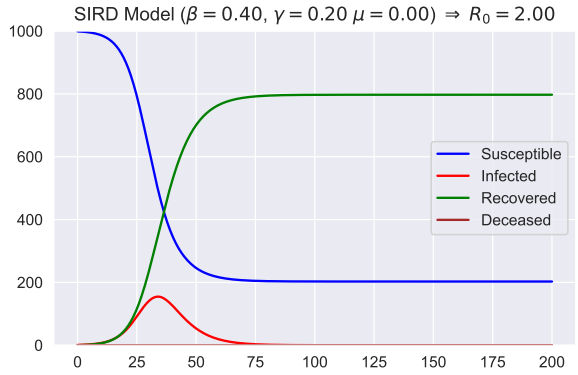
The value of R_0 indicates how the disease will spread in the early stages of an outbreak:

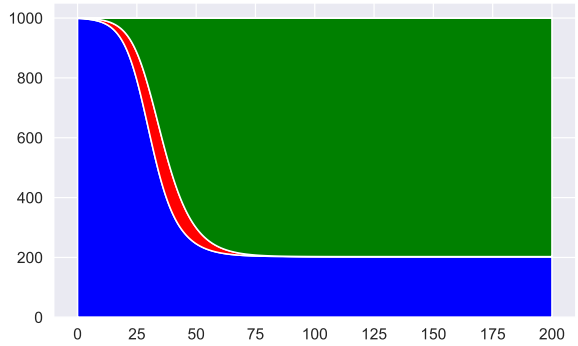
$$R_0 \begin{cases} < 1 & \text{Each infected individual infects fewer than one person on average, meaning the disease will eventually die out.} \\ \geq 1 & \text{Each infected person, on average, transmits the infection to more than one other person, leading to an epidemic (the infection will spread).} \end{cases}$$

Effect of varying μ.

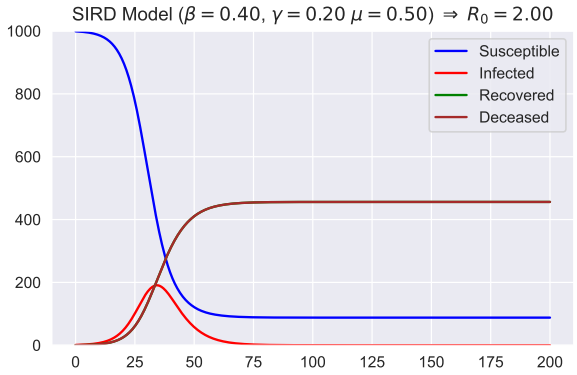
$$\longleftarrow \quad \beta = 0.4, \gamma = 0.2 \implies R_0 = 2 \quad \longrightarrow$$

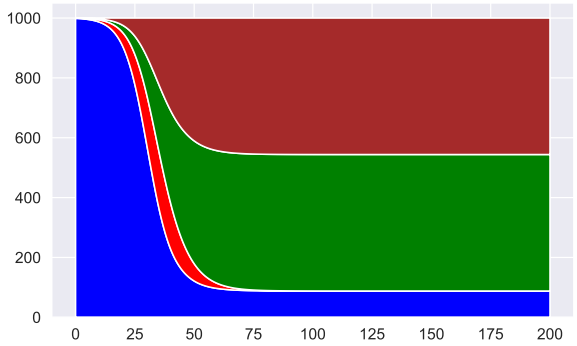
$\mu = 0.0$ $\mu = 0.5$ $\mu = 1.0$



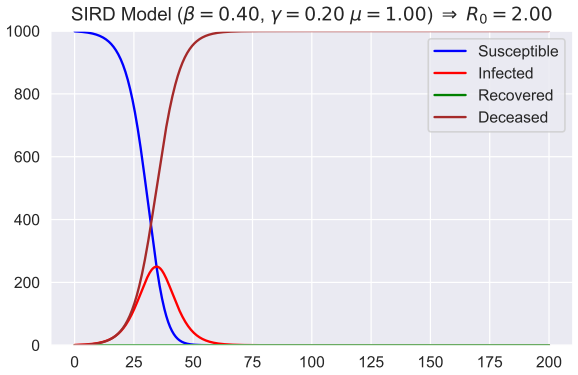


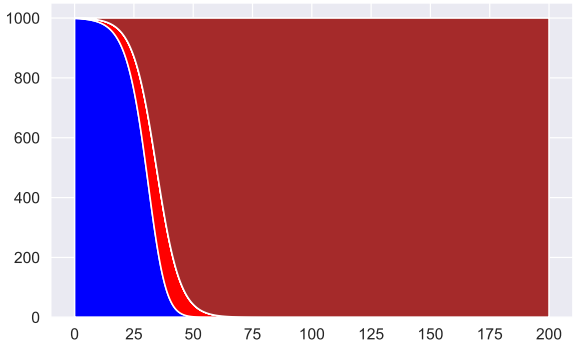
Everybody infected recovers, reduces to SIR model.





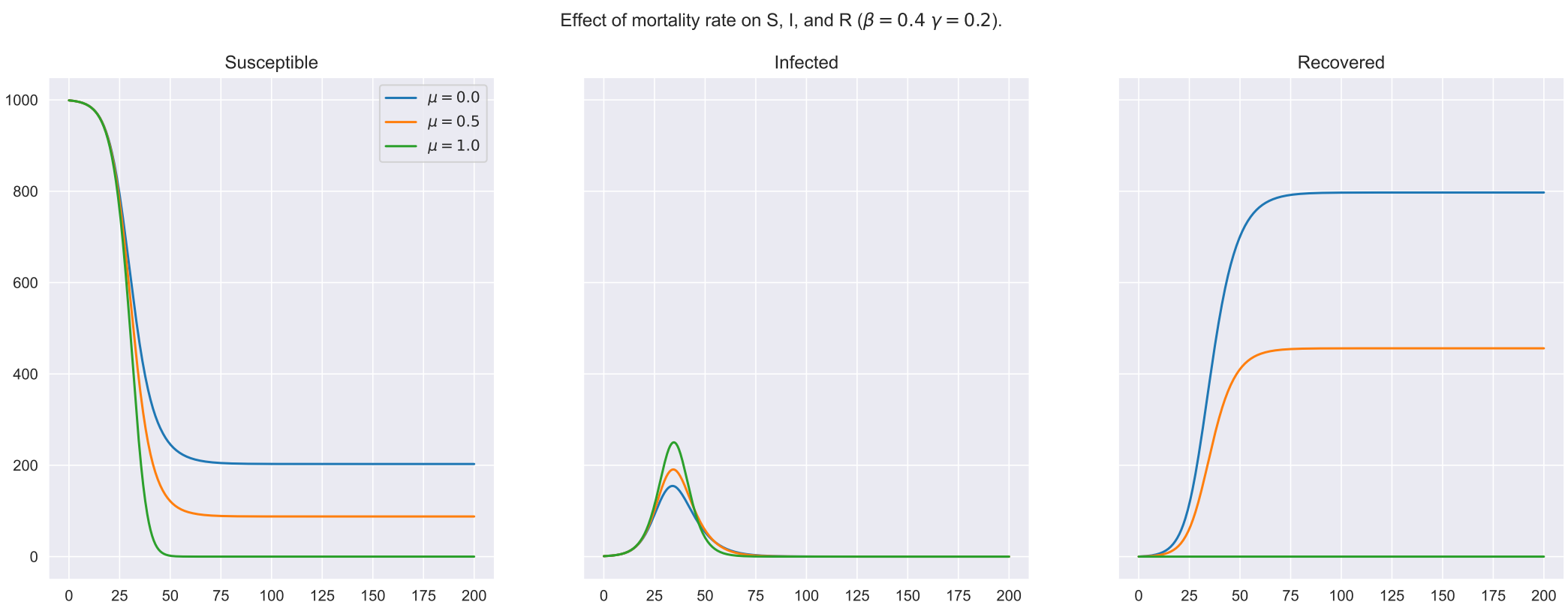
Half of those infected recover and half die.





Everybody dies. Think zombies and the Walking Dead

Effect of varying μ.



S and R behave as expected under varying μ, but why does the infection spike higher and then decay faster as μ increases?

1971 Influenza Outbreaks in Tristan da Cunha