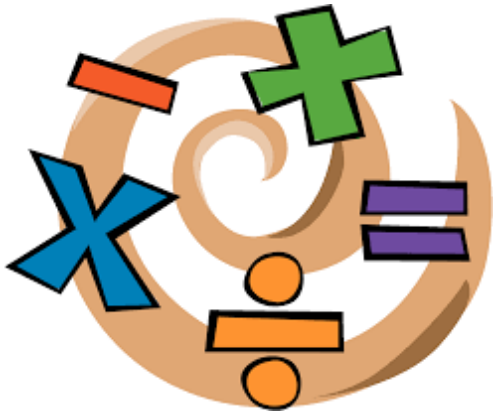
Introduction to Modeling Infectious Disease: Building and Analyzing Models

Lauren M. Childs

Associate Professor, Mathematics

Virginia Tech

How can mathematics inform our understanding of infectious diseases?



Divide population by disease status

- Susceptible
  - Susceptible to infection

S

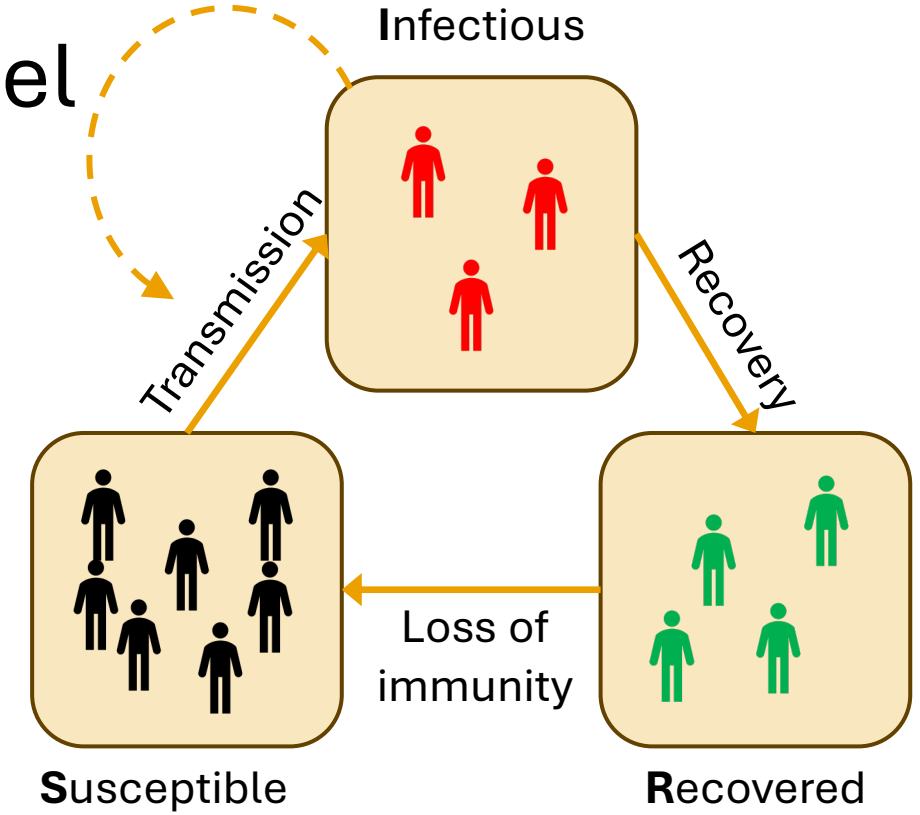
- Infectious
  - Able to transmit the infection

I

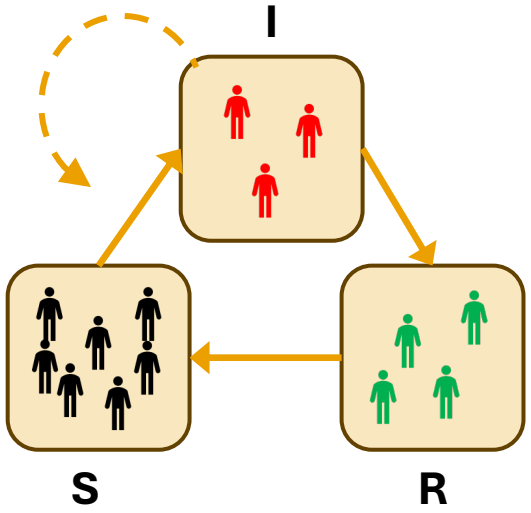
- Recovered/Removed
  - Unable to transmit the infection

R

Basic SIR model



Underlying assumptions

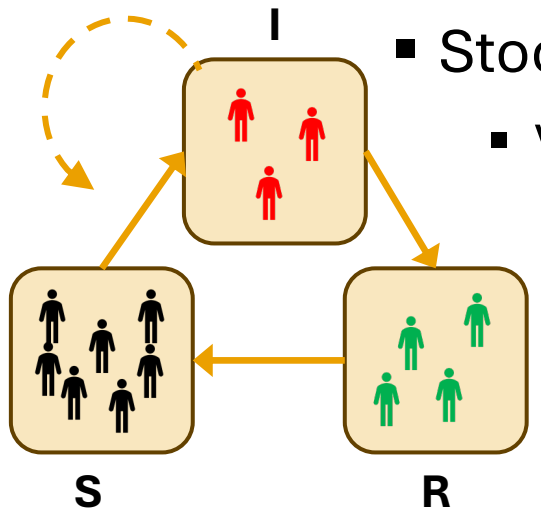


- Biological
  - Individuals can only be in one state at any given time.
  - Infection depends on interaction between S and I.
  - Recovery (and loss of immunity) occur at fixed rate.

- Mathematical
  - All parameters are biologically realistic (e.g., non-negative)
  - Variables represent densities (scaled to be between 0 and 1)
  - Non-linear term: infection depends on interaction between S and I.

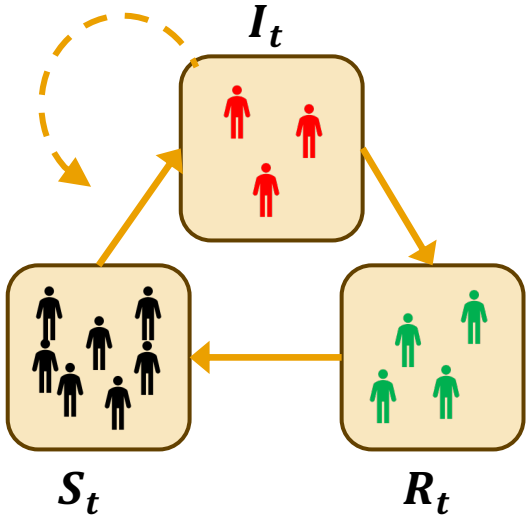
Mathematical Implementation

- Continuous-time
  - Differential equations (rates)
- Discrete-time
  - Difference equations (risk)
- Deterministic
  - Identical outputs with fixed inputs
- Stochastic
  - Varied outputs with fixed inputs



Mathematical Implementation

- Deterministic
  - Identical outputs with fixed inputs
- Discrete-time
  - Difference equations



- Consider the population at time $t + 1$ as a function of the population at time t
  - Movement between classes depends on transition risk (across the time step) and current population size
  - Order can matter!

Discrete-time deterministic system

Susceptible

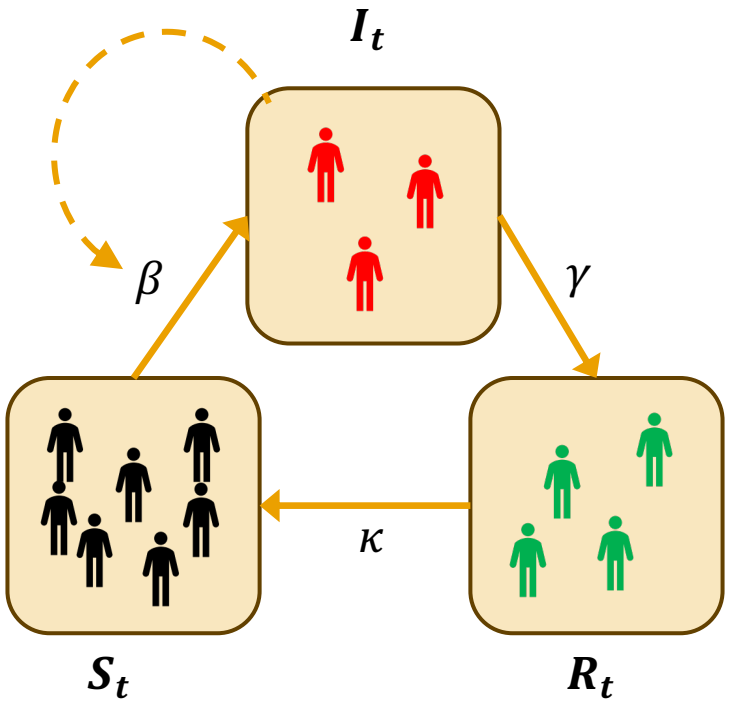
$$S_{t+1} = S_t - \beta S_t I_t + \kappa R_t$$

Infectious

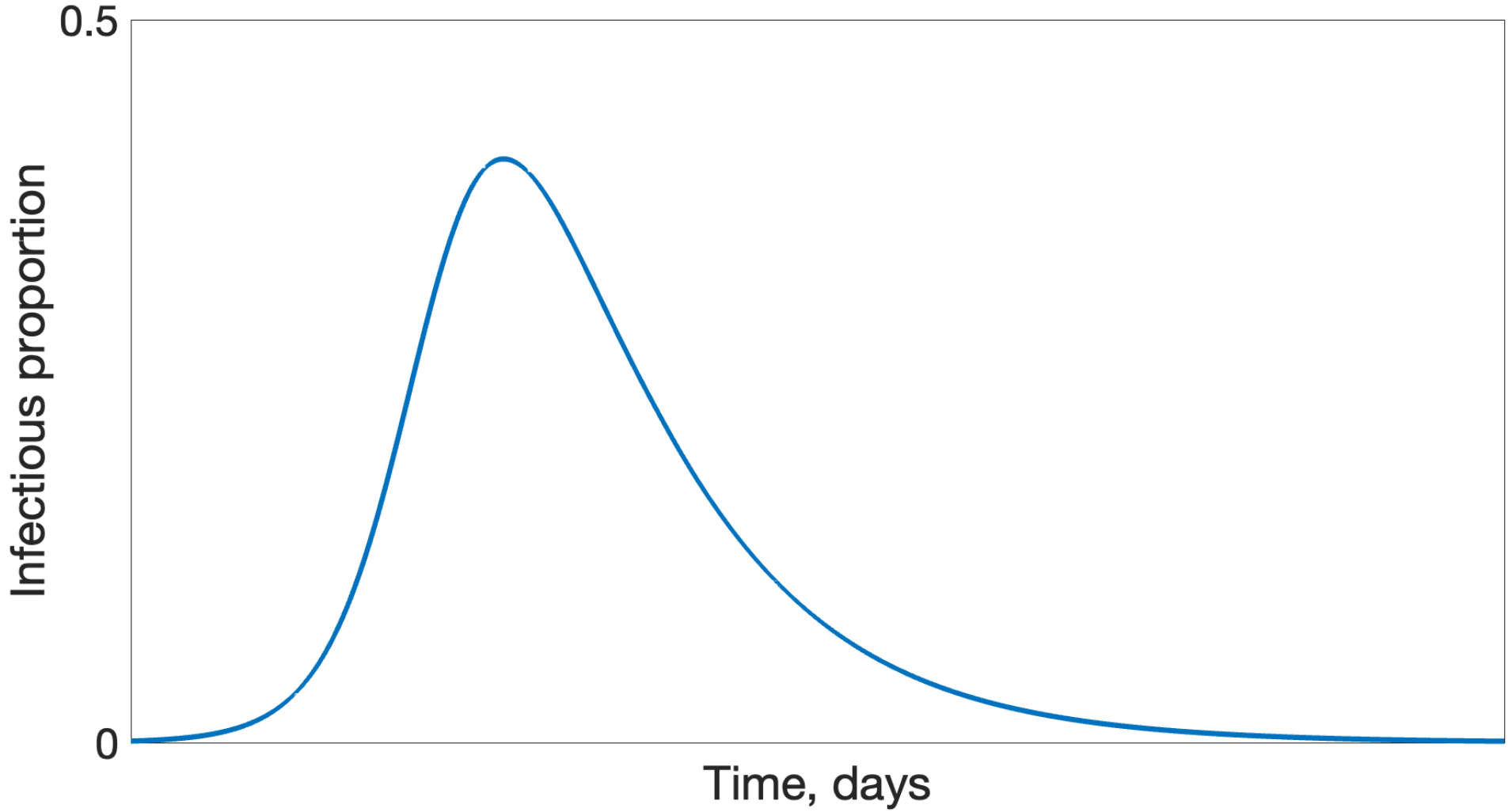
$$I_{t+1} = I_t + \beta S_t I_t - \gamma I_t$$

Recovered/Removed

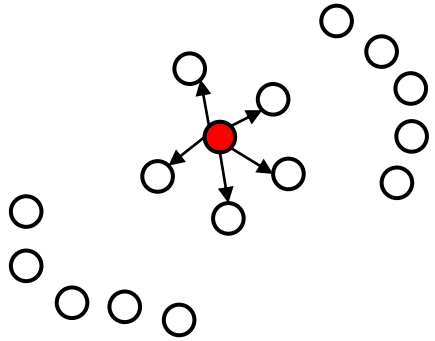
$$R_{t+1} = R_t + \gamma I_t - \kappa R_t$$



Dynamics of Basic SIR model (no loss of immunity)



Phases of an epidemic

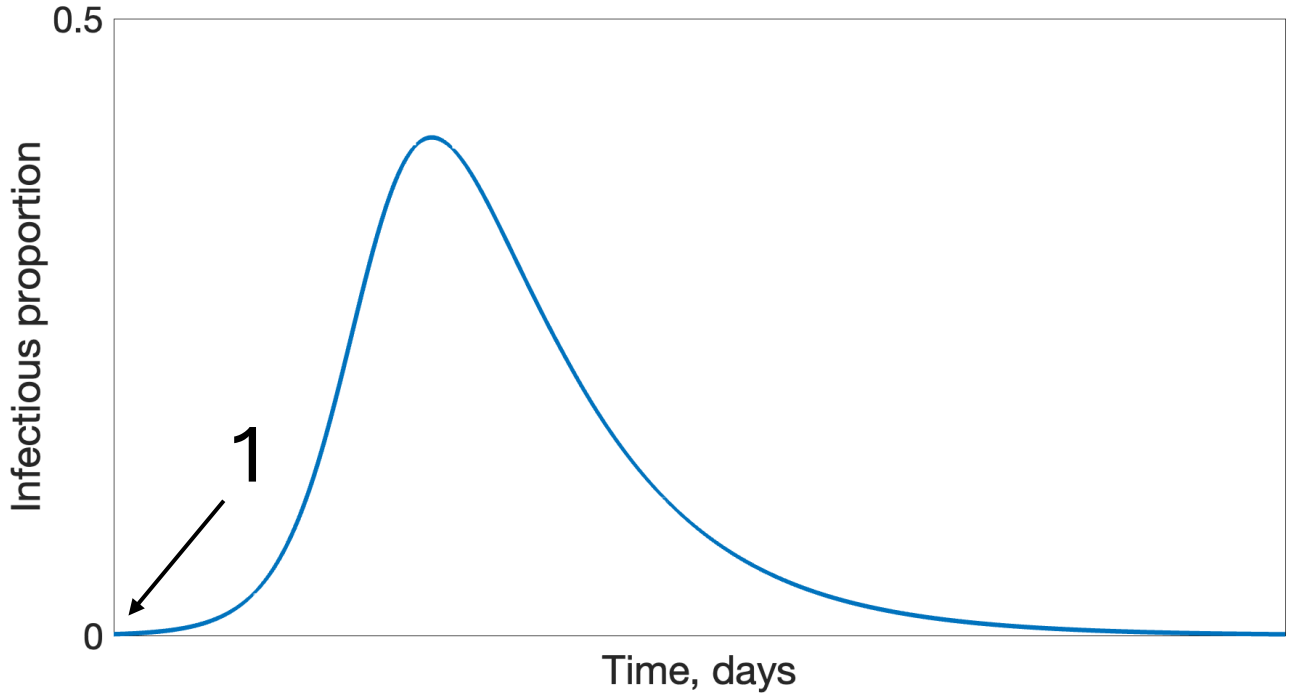


Susceptible

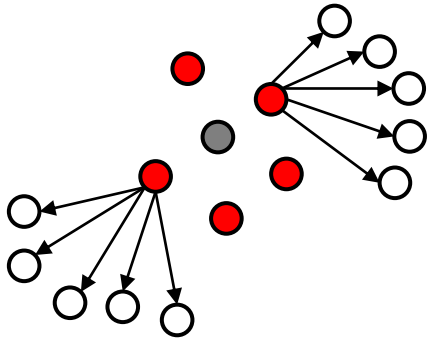
Infectious

Recovered

1. Start



Phases of an epidemic

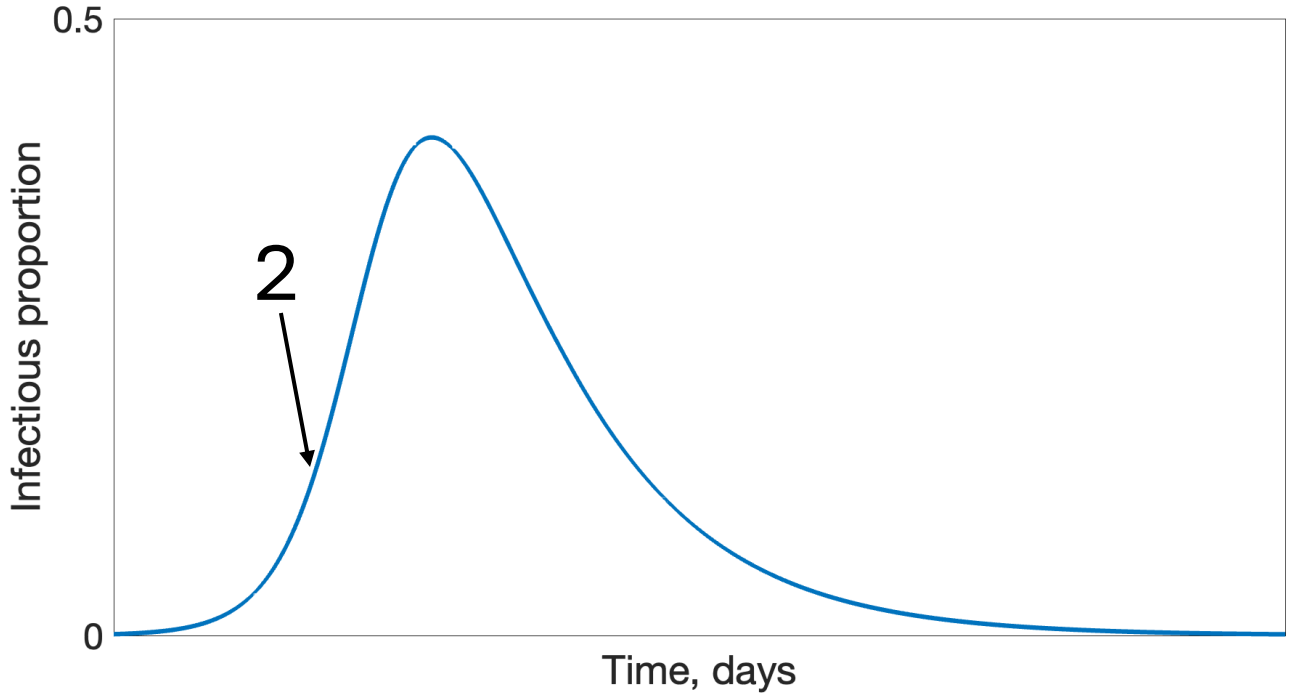


Susceptible

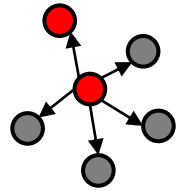
Infectious

Recovered

1. Start
2. Expansion



Phases of an epidemic

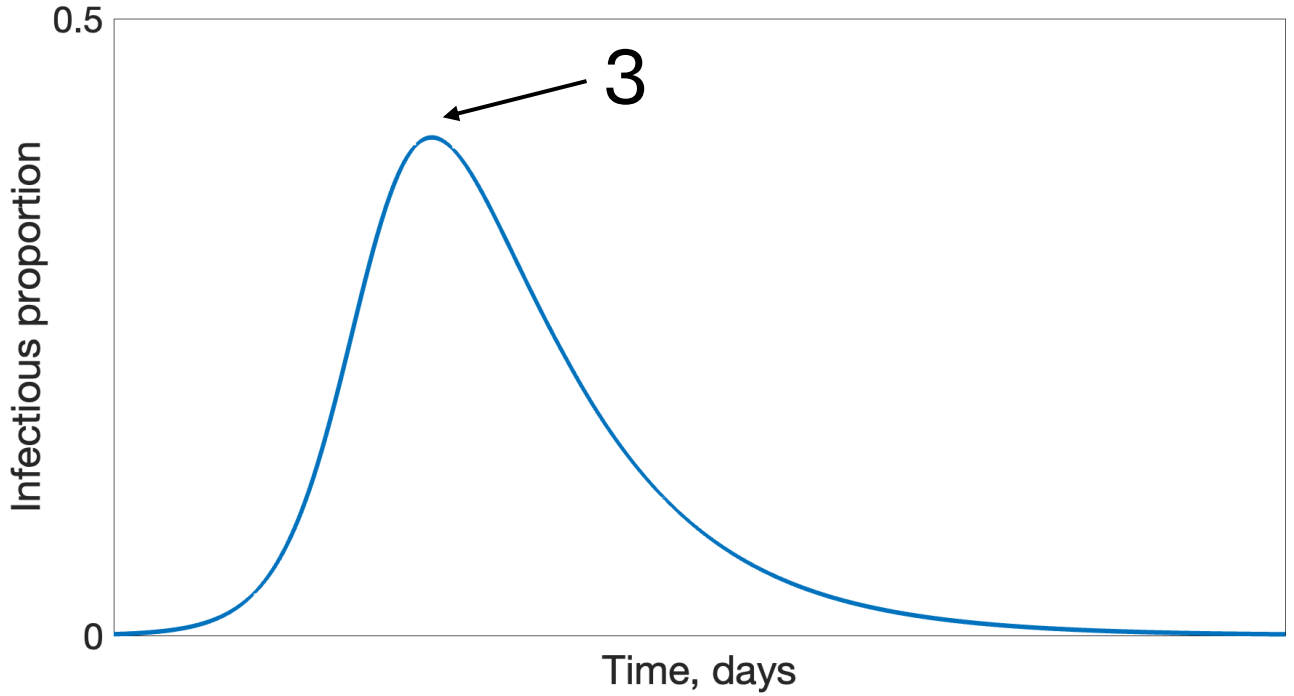


Susceptible

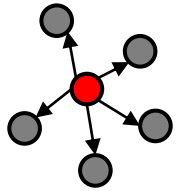
Infectious

Recovered

1. Start
2. Expansion
3. Peak



Phases of an epidemic

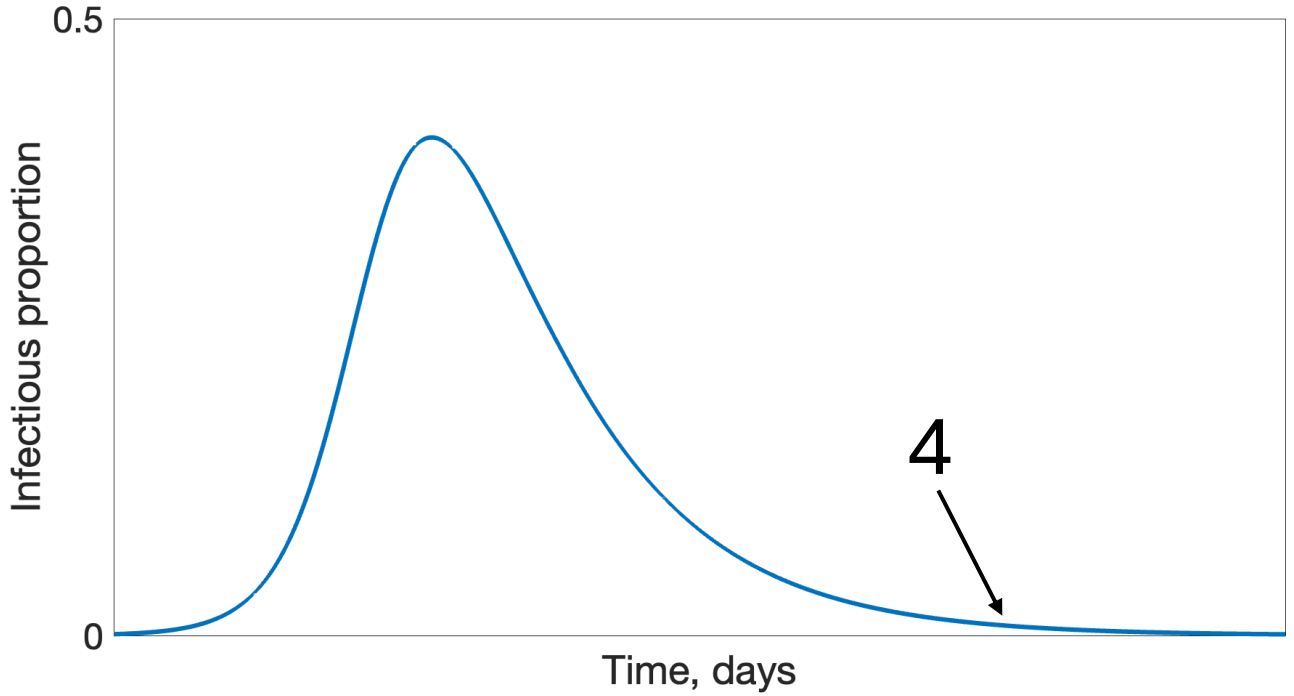


Susceptible

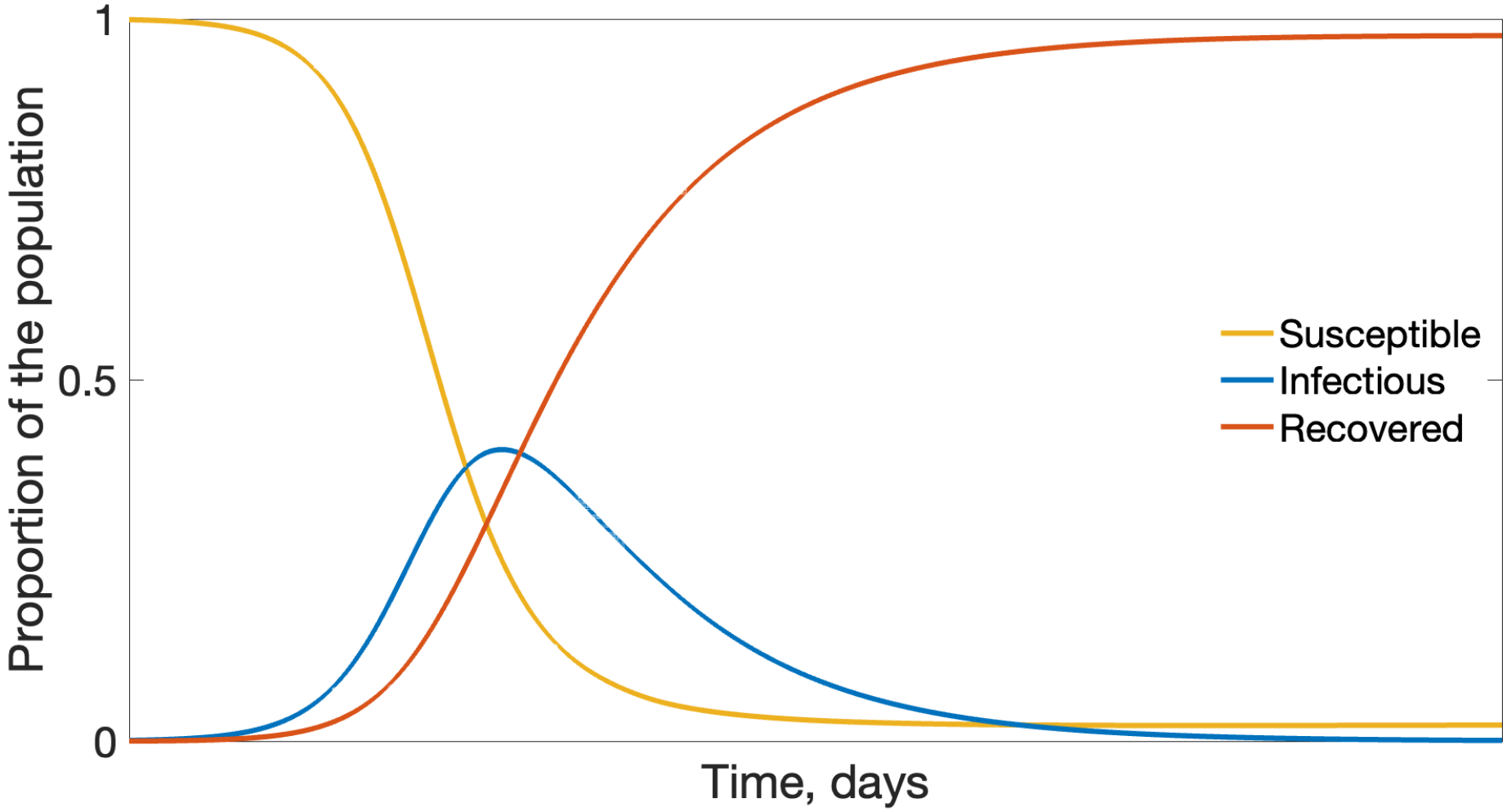
Infectious

Recovered

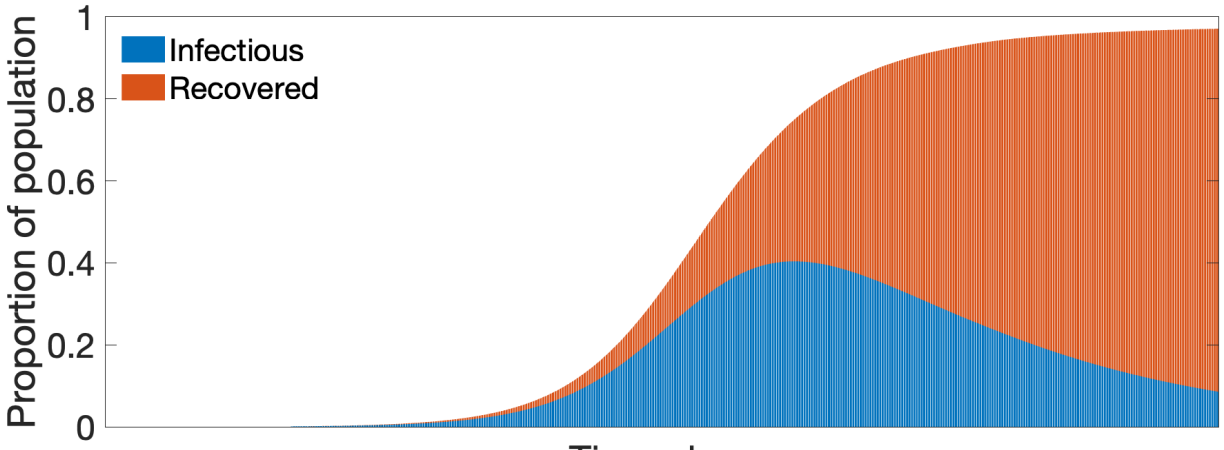
1. Start
2. Expansion
3. Peak
4. Resolution

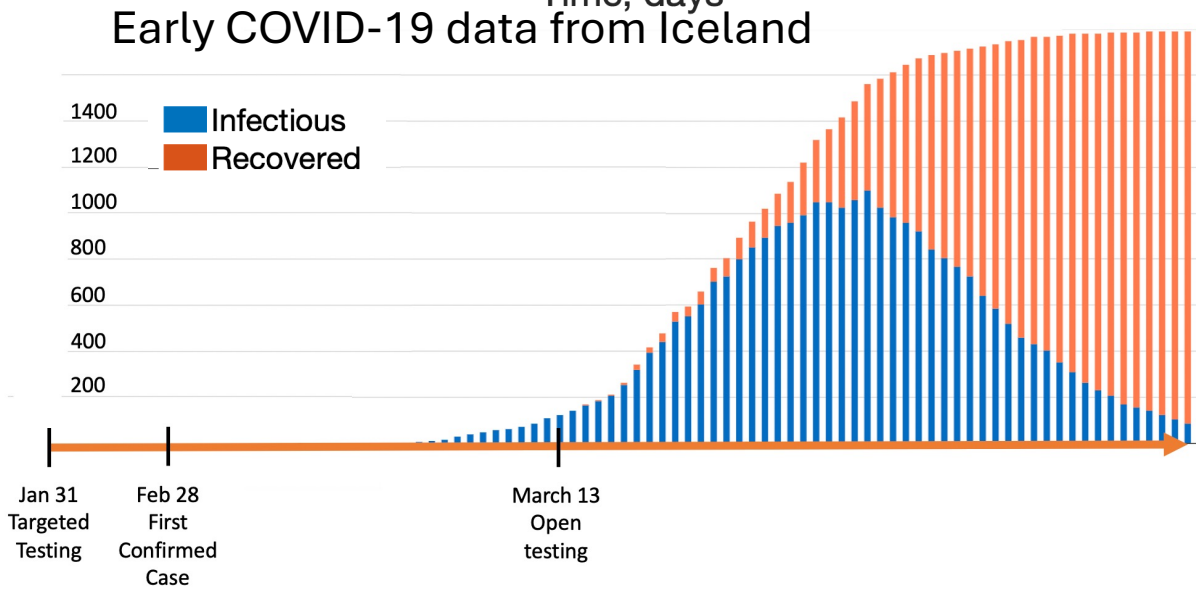


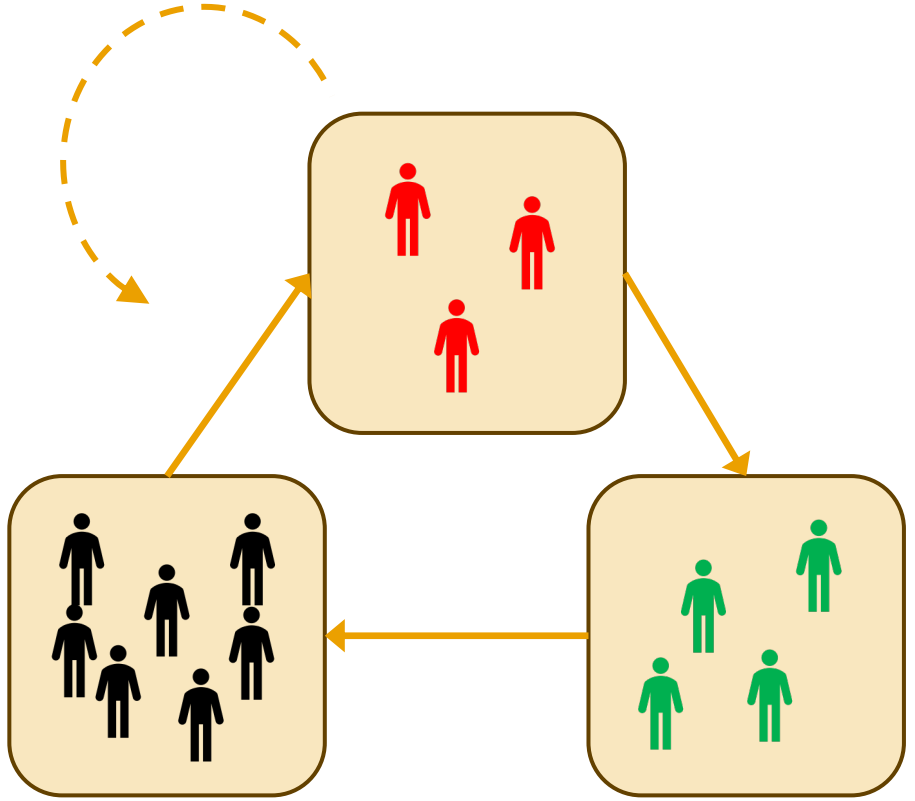
Dynamics of Basic SIR model



Basic model for infectious disease spread







Simulating the discrete-time deterministic system

Susceptible

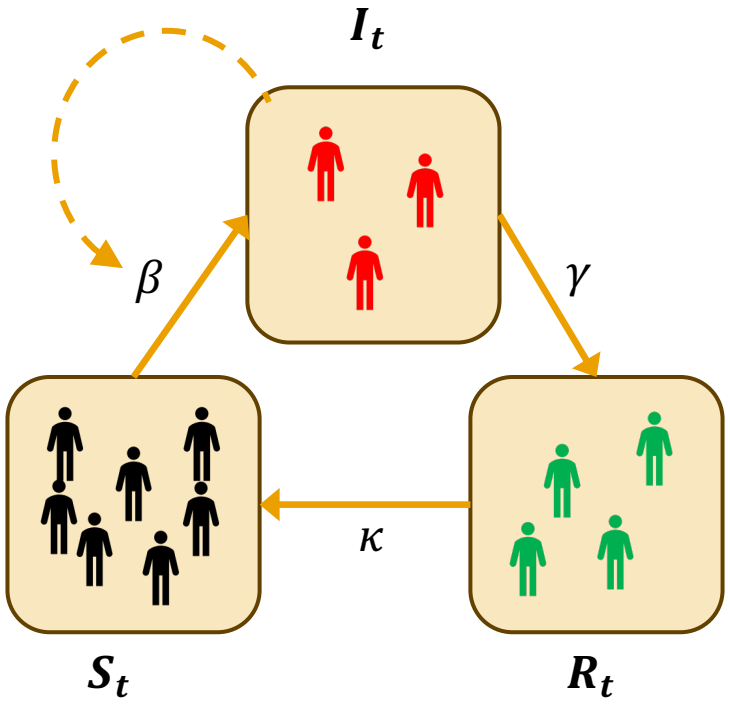
$$S_{t+1} = S_t - \beta S_t I_t + \kappa R_t$$

Infectious

$$I_{t+1} = I_t + \beta S_t I_t - \gamma I_t$$

Recovered/Removed

$$R_{t+1} = R_t + \gamma I_t - \kappa R_t$$



Simulating the discrete-time deterministic system

1. Choose the parameters, initial starting values, and time vector.

$$\{\beta, \kappa, \gamma\} \quad \{S(0), I(0), R(0)\} \quad t = [0,10]$$

Important to pre-allocate memory for the vectors.

2. Iterate forward the simulation from starting to ending time with time step Δt.

Careful with time step and indexing.

$$S(\Delta t) = S(0) - \beta S(0)I(0) + \kappa R(0) \quad S_1 = S_0 - \beta S_0 I_0 + \kappa R_0$$
$$I(\Delta t) = I(0) + \beta S(0)I(0) - \gamma I(0) \quad I_1 = I_0 + \beta S_0 I_0 - \gamma I_0$$
$$R(\Delta t) = R(0) + \gamma I(0) - \kappa R(0) \quad R_1 = R_0 + \gamma I_0 - \kappa R_0$$

Simulation of SIR model in R

Code file: *1_Intro_ID_SIR.R*

Starting your R script

Basic set up

```
# Start from a fresh workspace
rm(list=ls()) # clear workspace
if (!is.null(dev.list())){dev.off()} # clear figures
cat("\014") # clear console

# Required Libraries
if (!require("ggplot2")) install.packages("ggplot2")
library(ggplot2)
```

Setting up your simulation

Parameters, initial conditions, time steps, and pre-allocate memory

```
# choose parameter values
parameters <- c(beta = 2/3, kappa = 1/30, gamma = 1/3)
```

Setting up your simulation

Parameters, initial conditions, time steps, and pre-allocate memory

```
# choose parameter values
parameters <- c(beta = 2/3, kappa = 1/30, gamma = 1/3)

# initial state variables
inits <- c(S = 499, I = 1, R = 0)
```

Setting up your simulation

Parameters, initial conditions, time steps, and pre-allocate memory

```
# choose parameter values
parameters <- c(beta = 2/3, kappa = 1/30, gamma = 1/3)

# initial state variables
inits <- c(S = 499, I = 1, R = 0)

# time to run simulation
time_vector <- seq(0, 300, 1)
```

Setting up your simulation

Parameters, initial conditions, time steps, and pre-allocate memory

```
# choose parameter values
parameters <- c(beta = 2/3, kappa = 1/30, gamma = 1/3)

# initial state variables
inits <- c(S = 499, I = 1, R = 0)

# time to run simulation
time_vector <- seq(0, 300, 1)

# pre-allocate array for variables
S = rep(NA,length(time_vector))
I = rep(NA,length(time_vector))
R = rep(NA,length(time_vector))
```

Setting up your simulation

Parameters, initial conditions, time steps, and pre-allocate memory

```
# choose parameter values
parameters <- c(beta = 2/3, kappa = 1/30, gamma = 1/3)

# initial state variables
inits <- c(S = 499, I = 1, R = 0)

# time to run simulation
time_vector <- seq(0, 300, 1)

# pre-allocate array for variables
S = rep(NA,length(time_vector))
I = rep(NA,length(time_vector))
R = rep(NA,length(time_vector))
```

```
# place initial conditions in first position of array
nTotal = sum(inits)
S[1] = inits["S"]
I[1] = inits["I"]
R[1] = inits["R"]
```

Setting up your simulation

Parameters, initial conditions, and pre-allocate memory

```
# choose parameter values
parameters <- c(beta = 2/3, kappa = 1/30, gamma = 1/3)

# initial state variables
inits <- c(S = 499, I = 1, R = 0)

# time to run simulation
time_vector <- seq(0, 300, 1)

# pre-allocate array for variables
S = rep(NA,length(time_vector))
I = rep(NA,length(time_vector))
R = rep(NA,length(time_vector))
```

```
# place initial conditions in first position of array
nTotal = sum(inits)
S[1] = inits["S"]
I[1] = inits["I"]
R[1] = inits["R"]

# shorter access for parameters
beta = parameters["beta"]
gamma = parameters["gamma"]
kappa = parameters["kappa"]
```

Code for running your simulation

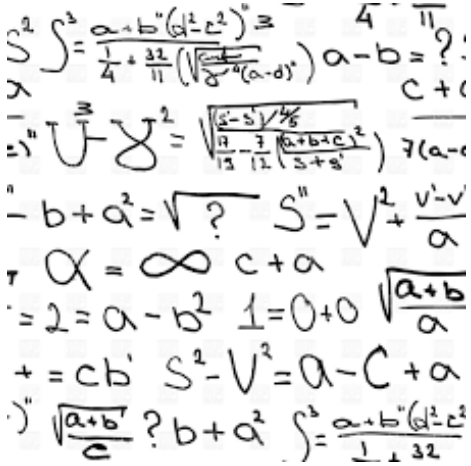
For loop for simulation

```
# loop over time vector
for (t in 1:(length(time_vector)-1)){
  # Epidemic events
  newly_infected = I[t] * S[t] * beta / nTotal
  recovered_today= I[t] * gamma
  losing_immunity_today = R[t] * kappa

  # Updating epidemic equations
  S[t+1] = S[t] - newly_infected                      + losing_immunity_today
  I[t+1] = I[t] + newly_infected - recovered_today
  R[t+1] = R[t]                  + recovered_today - losing_immunity_today

}
```

Plotting

```
# pivot to long data frame
df.long = pivot_longer(df, cols = c("S", "I", "R"), # select the columns you want to make longer
                       names_to = "compartment", # new column name that contains the names of the old columns (S, I, R)
                       values_to = "population") # values from the old columns (S, I, R)


# plot
ggplot(data = df.long, aes(x = time_vector, y = population, color = compartment) ) + # Data and aesthetics
  geom_line() +  # Line plot
  labs(x = "Time, days",                                # x label
       y = "Population size",                           # y label
       title = "SIR model",                             # plot title
       color = "") +                                    # legend title
  theme_minimal()+
  scale_color_discrete(breaks=c('S', 'I', 'R'))         # reorder lines
```

How do we convey results of the mathematical model of infectious disease?

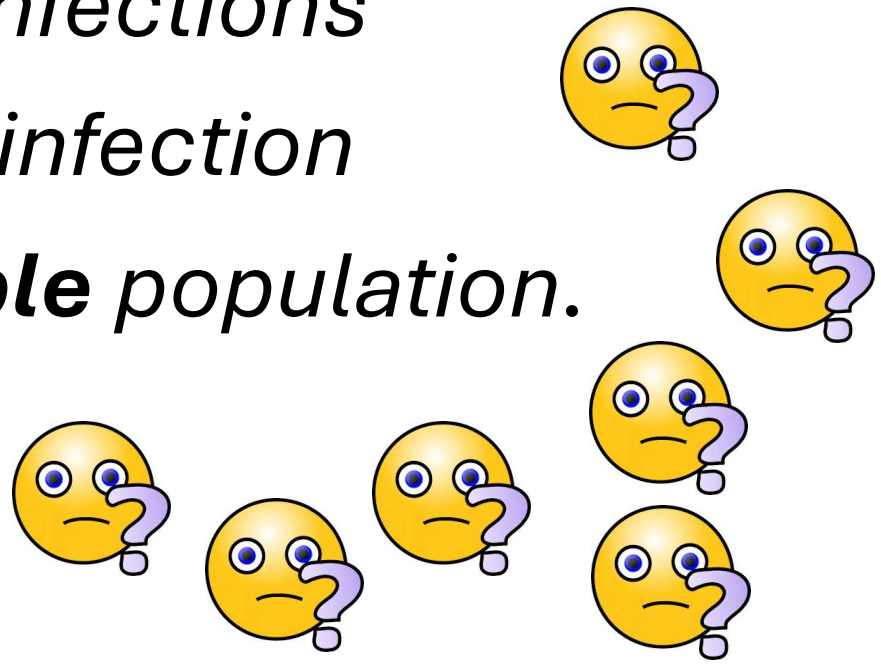


Basic Reproductive Number: R_0

Average *number of infections*

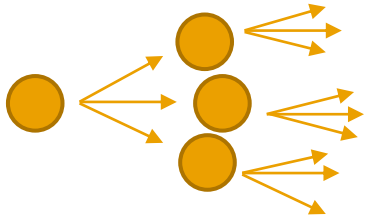
caused by a ***typical*** *infection*

in a completely ***susceptible*** *population.*



R_0 as a threshold

Epidemic: $R_0 > 1$



Endemic: $R_0 = 1$

Elimination: $R_0 < 1$

Determining the basic reproductive number

Consider if infectious compartment is increasing

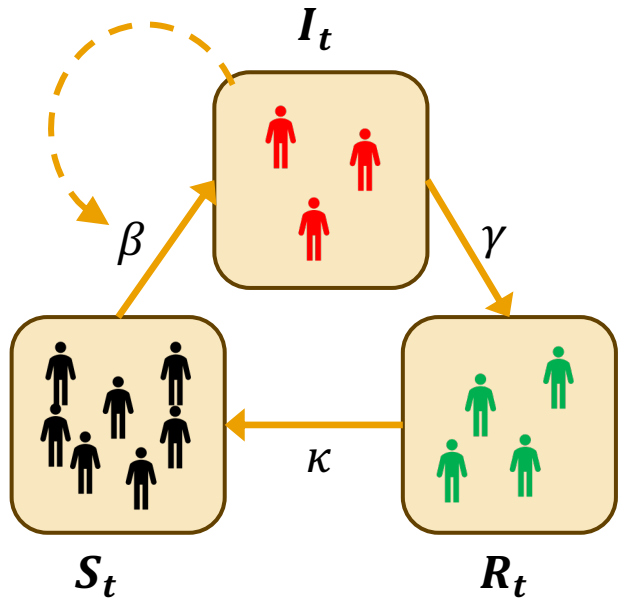
$$I_{t+1} - I_t > 0$$

Using our infectious equation

$$\begin{aligned} I_{t+1} - I_t &= I_t + \beta S_t I_t - \gamma I_t - I_t \\ &= \beta S_t I_t - \gamma I_t \\ &= (\beta S_t - \gamma) I_t \end{aligned}$$

As $I_t > 0$, it only matters if $(\beta S_t - \gamma) > 0$!

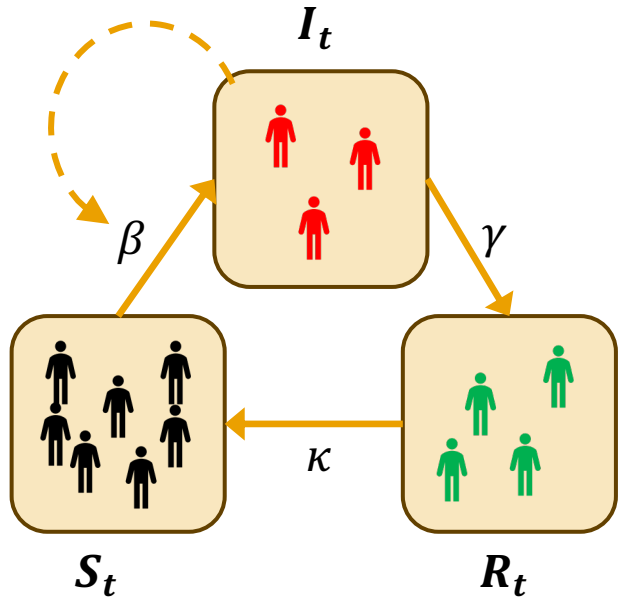
Equivalent to $\frac{\beta S_t}{\gamma} > 1$!



Basic reproductive number: SIR

Define basic reproduction number:

$$R_0 = \frac{\beta S_0}{\gamma} > 1$$



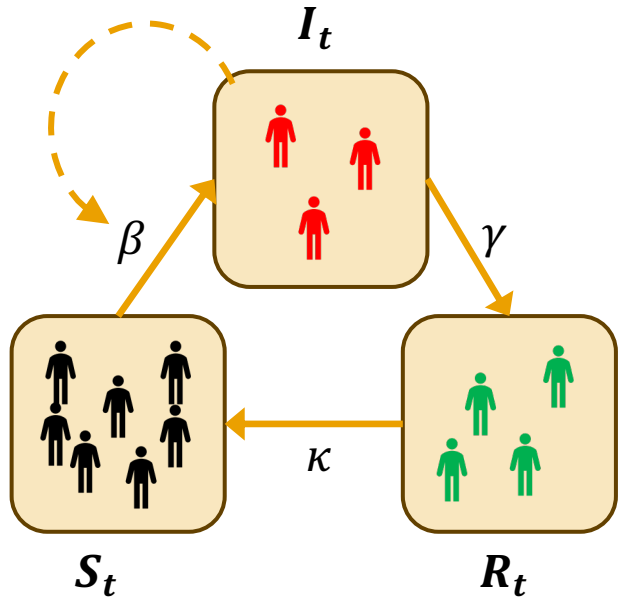
Interpretation:

$$R_0 = \underbrace{\beta S_0}_{\text{New infections per unit time}} * \underbrace{\frac{1}{\gamma}}_{\text{Length of time infectious}}$$

Basic reproductive number: SIR

Define basic reproduction number:

$$R_0 = \frac{\beta}{\gamma} > 1 \qquad (\text{as } S_0 \approx 1)$$



Interpretation:

$$R_0 = \underbrace{\beta}_{\text{New infections per unit time}} * \underbrace{\frac{1}{\gamma}}_{\text{Length of time infectious}} \qquad (\text{as } S_0 \approx 1)$$

Adding basic reproduction calculation for SIR model in R

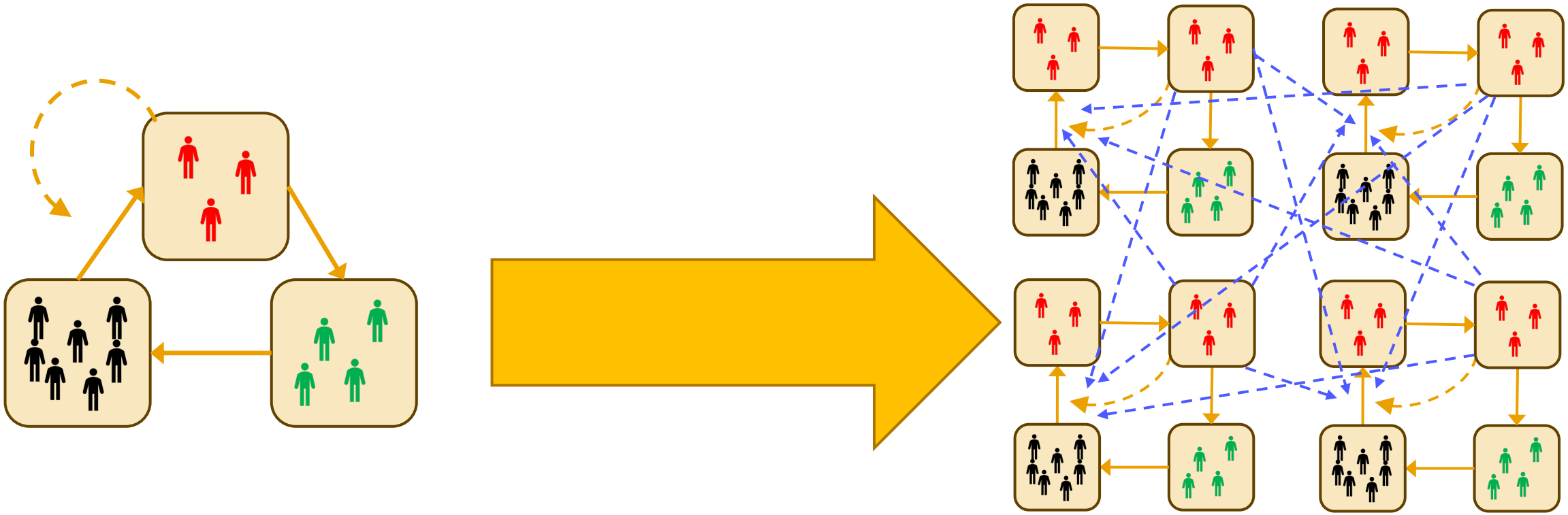
Code file: *1_Intro_ID_SIR.R*

Calculating R_0

```
# calculating R0
R0 <- beta/gamma
print(R0)
```

Note that we assume inits["S"] = nTotal so the fraction of susceptible individuals is nearly 1.

How do we add biological realism to our mathematical model of infectious disease?



Adding a compartment: SEIR

Susceptible

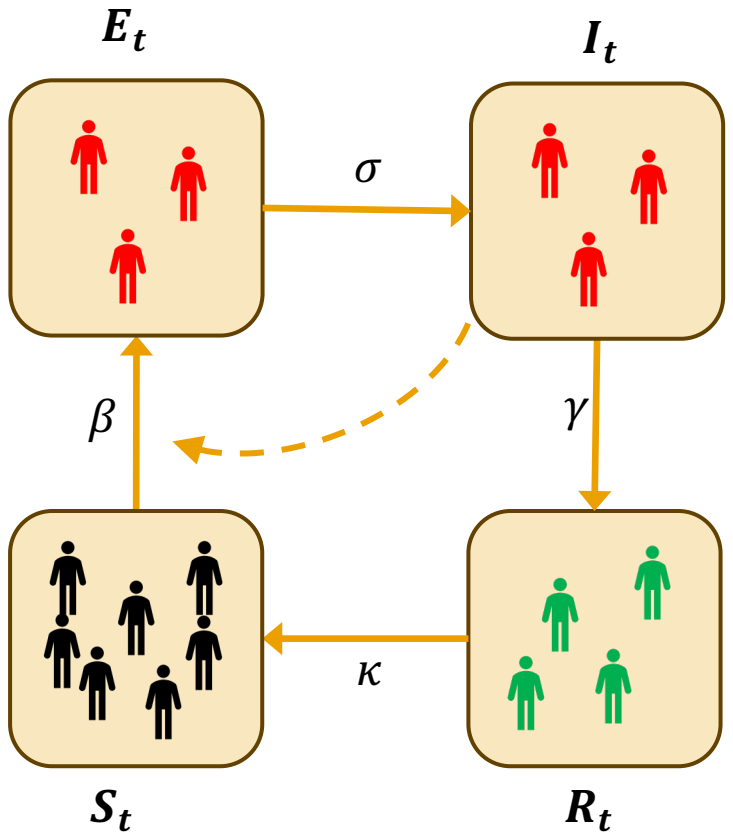
Exposed

Infected but not yet *infectious*

Time prior to infectiousness = latent period

Infectious

Recovered/Removed



Discrete time model of SEIR system

Susceptible

$$S_{t+1} = S_t - \beta S_t I_t + \kappa R_t$$

Exposed

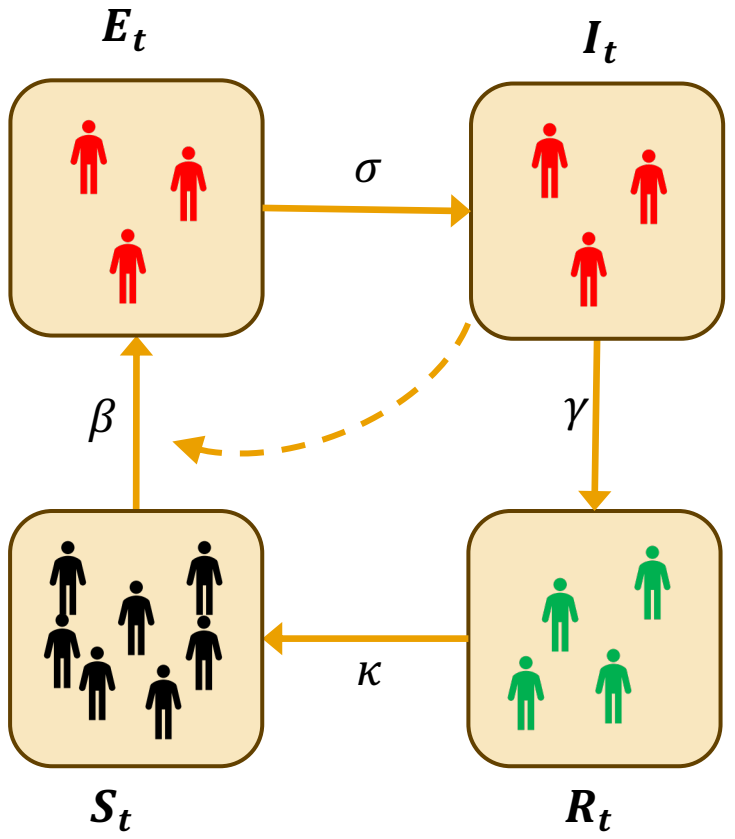
$$E_{t+1} = E_t + \beta S_t I_t - \sigma E_t$$

Infectious

$$I_{t+1} = I_t + \sigma E_t - \gamma I_t$$

Recovered/Removed

$$R_{t+1} = R_t + \gamma I_t - \kappa R_t$$



Simulation of SEIR model in R

Code file: *2_Intro_ID_SEIR.R*

Changes necessary for SIR to SEIR model

- Four parameters
  - beta, kappa, gamma, sigma
- Four initial conditions and pre-allocated state variables
  - S, E, I, R
- New epidemic event
  - Four equations to update

Changes necessary for SIR to SEIR model

- Four parameters
  - beta, kappa, gamma, sigma
- Four initial conditions and pre-allocated state variables
  - S, E, I, R
- New epidemic event
  - Four equations to update

```
# choose parameter values
parameters <- c(beta = 2/3, kappa = 1/30, gamma = 1/3, sigma = 1/10)

# initial state variables
inits <- c(S = 499, E = 0, I = 1, R = 0)

# time to run simulation
time_vector <- seq(0, 300, 1)

# pre-allocate array for variables
S = rep(NA,length(time_vector))
E = rep(NA,length(time_vector))
I = rep(NA,length(time_vector))
R = rep(NA,length(time_vector))
```

Determining the basic reproductive number

Consider if infected compartments are increasing

$$(E_{t+1} + I_{t+1}) - (E_t + I_t) > 0$$

Using our exposed and infectious equations

$$\begin{aligned}(E_{t+1}+I_{t+1}) - (E_t+I_t) &= (E_t+\beta S_t I_t - \sigma E_t + I_t + \sigma E_t - \gamma I_t) - (E_t+I_t) \\ &= \beta S_t I_t - \gamma I_t \\ &= (\beta S_t - \gamma) I_t\end{aligned}$$

As $I_t > 0$, it only matters if $(\beta S_t - \gamma) > 0$!

Equivalent to $\frac{\beta S_t}{\gamma} > 1$!

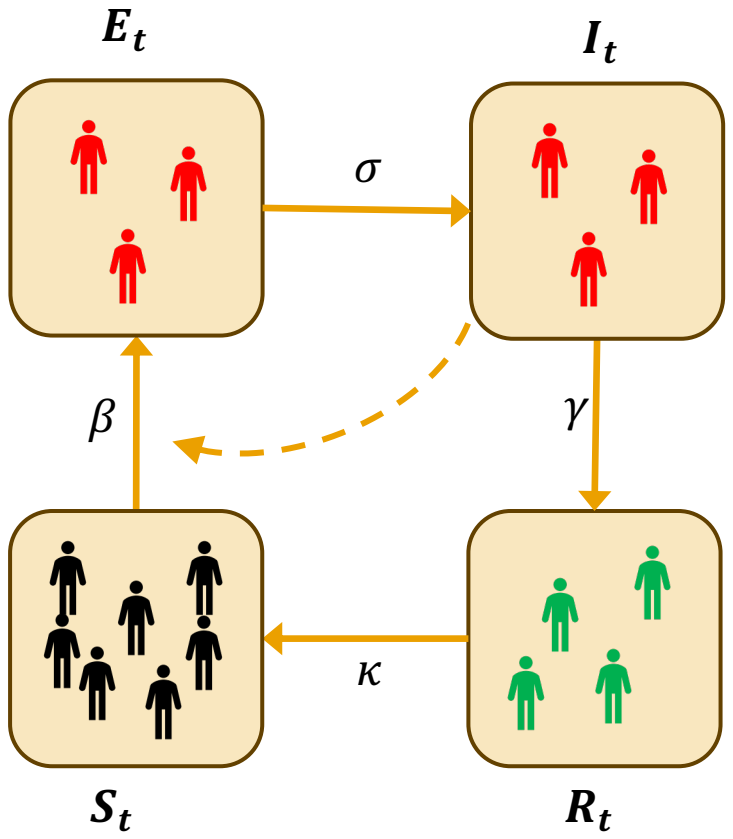
Basic reproductive number: SEIR

Define basic reproduction number:

$$R_0 = \frac{\beta S_0}{\gamma} > 1$$

Interpretation:

$$R_0 = \underbrace{\beta S_0}_{\text{New infections per unit time}} * \underbrace{\frac{1}{\gamma}}_{\text{Length of time infectious}}$$



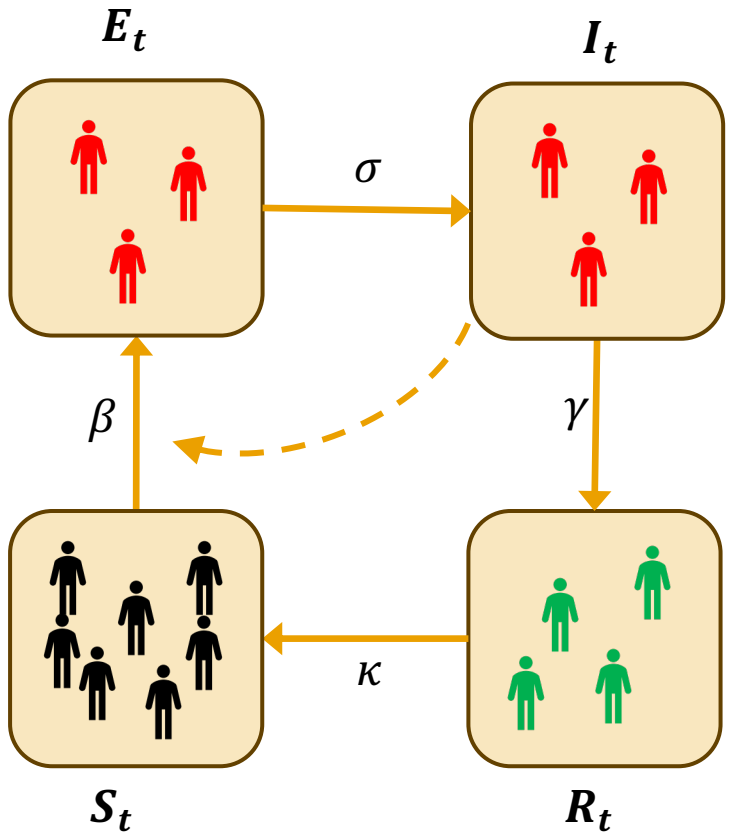
Basic reproductive number: SEIR

Define basic reproduction number:

$$R_0 = \frac{\beta}{\gamma} > 1 \qquad (\text{as } S_0 \approx 1)$$

Interpretation:

$$R_0 = \underbrace{\beta}_{\text{New infections per unit time}} * \underbrace{\frac{1}{\gamma}}_{\text{Length of time infectious}} \qquad (\text{as } S_0 \approx 1)$$



Adding transitions: SEIR with demography

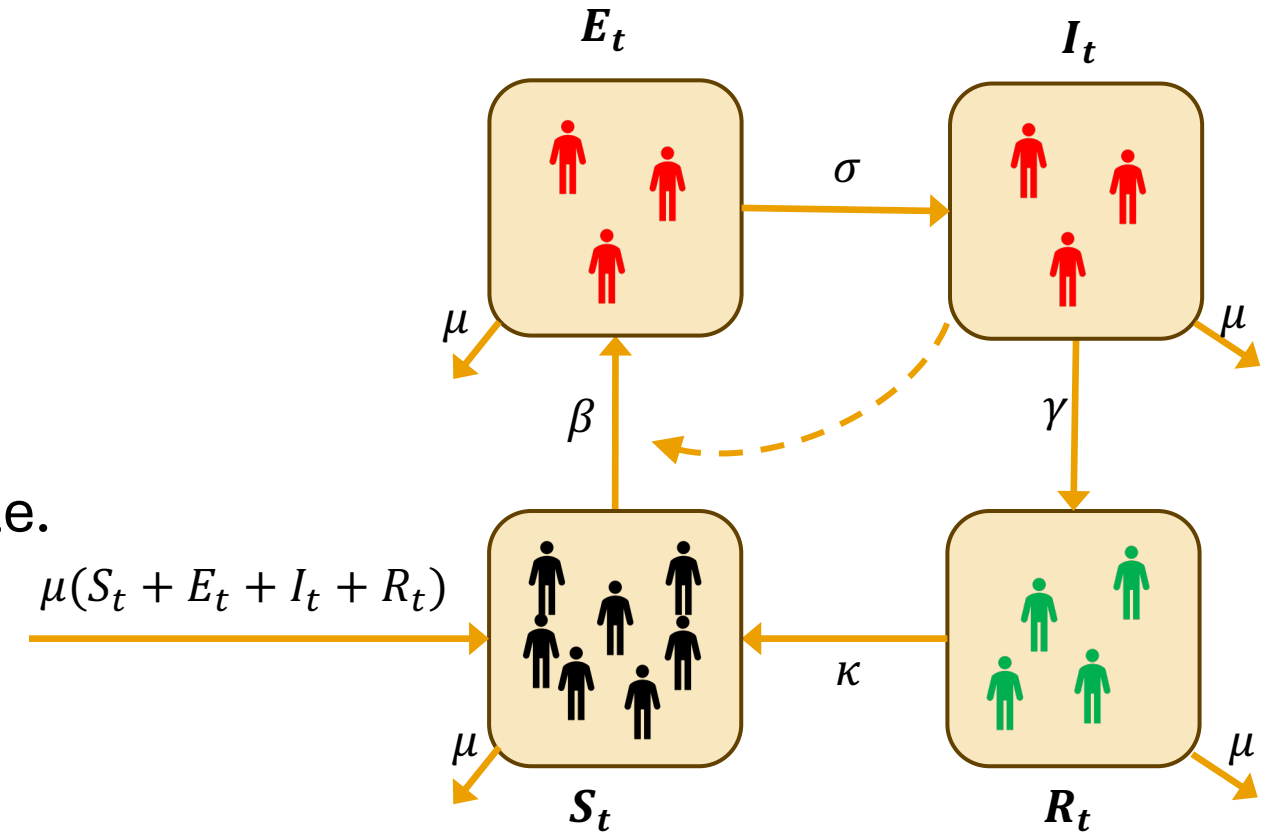
Add natural birth and mortality.

Assumptions:

All individuals are born susceptible.

All individuals contribute to birth.

Birth and death are balanced.



Discrete time model of SEIR system

Susceptible

$$S_{t+1} = \mu + S_t - \beta S_t I_t + \kappa R_t - \mu S_t$$

Exposed

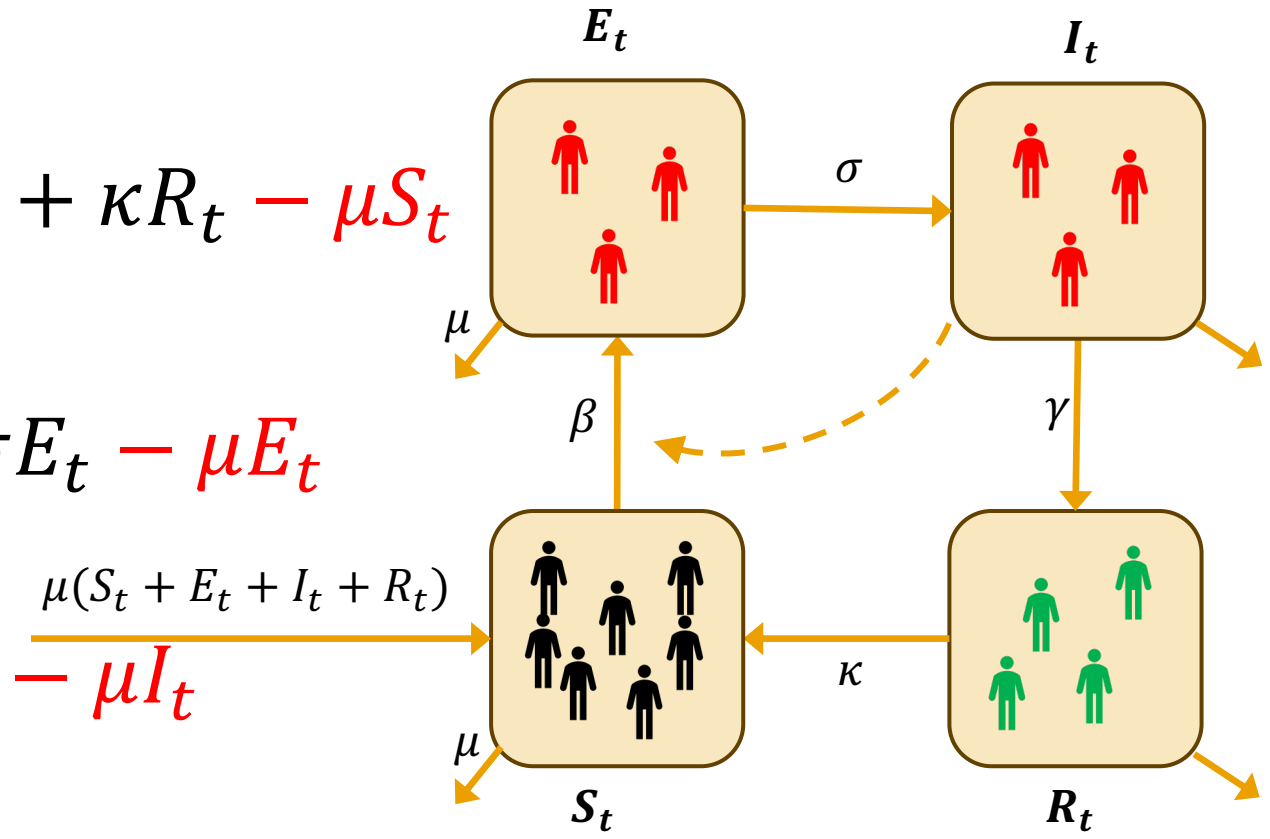
$$E_{t+1} = E_t + \beta S_t I_t - \sigma E_t - \mu E_t$$

Infectious

$$I_{t+1} = I_t + \sigma E_t - \gamma I_t - \mu I_t$$

Recovered/Removed

$$R_{t+1} = R_t + \gamma I_t - \kappa R_t - \mu R_t$$



Determining the basic reproductive number

Consider if infected compartments are increasing

$$(E_{t+1} + I_{t+1}) - (E_t + I_t) > 0$$

Using our exposed and infectious equations

$$\begin{aligned}(E_{t+1}+I_{t+1}) - (E_t+I_t) &= (E_t+\beta S_t I_t - \sigma E_t - \mu E_t \ + I_t + \sigma E_t - \gamma I_t - \mu I_t) - (E_t+I_t) \\ &= \beta S_t I_t - \gamma I_t - \mu E_t \ - \mu I_t \\ &= (\beta S_t - \gamma - \mu) I_t - \mu E_t\end{aligned}$$

Assume E_t is in quasi-equilibrium: $E_t = \frac{\beta S_t I_t}{\sigma+\mu}$ and then some algebra!

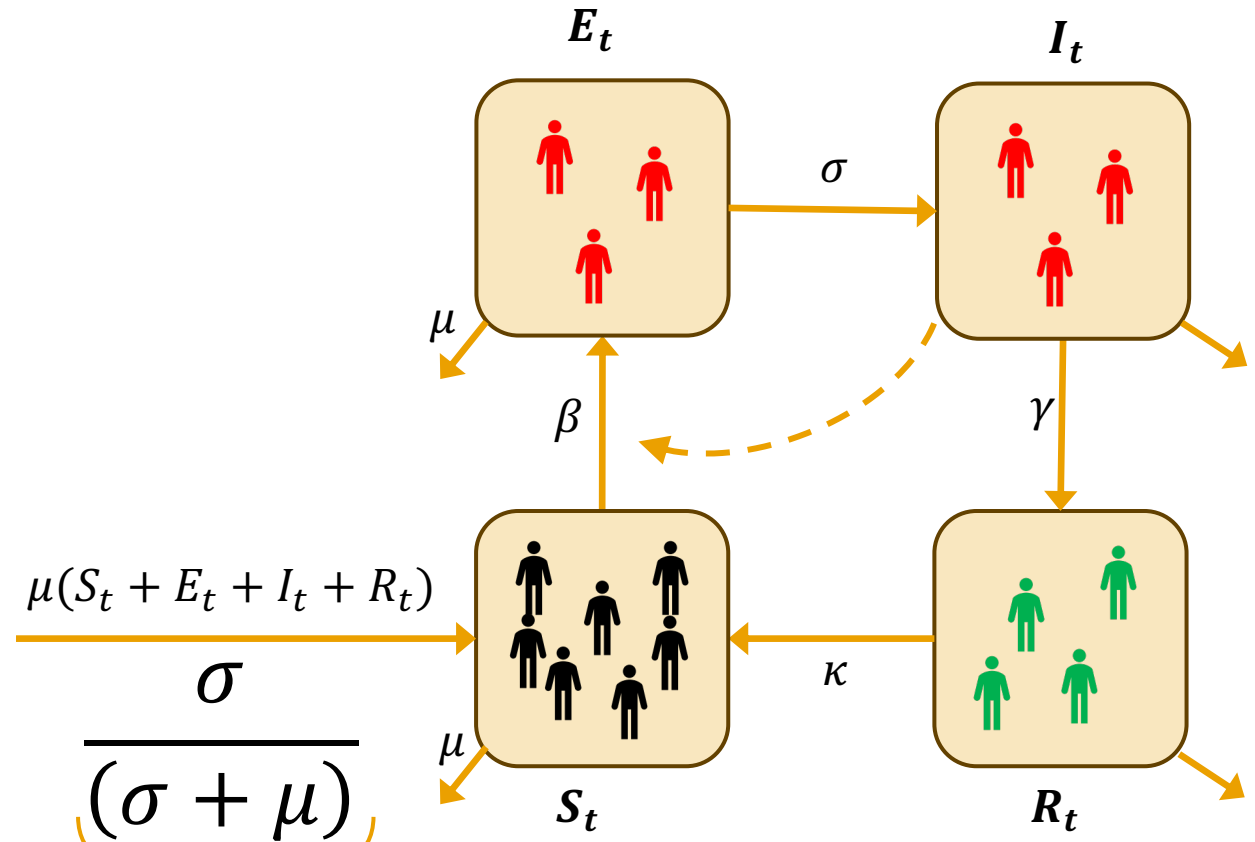
Basic reproductive number: SEIR

Define basic reproduction number:

$$R_0 = \frac{\beta S_0 \sigma}{(\gamma+\mu)(\sigma+\mu)} > 1$$

Interpretation:

$$R_0 = \underbrace{\beta S_0}_{\text{New infections per unit time}} * \underbrace{\frac{1}{(\gamma+\mu)}}_{\text{Length of time infectious}} * \underbrace{\frac{\sigma}{(\sigma+\mu)}}_{\text{Probability of surviving latent period}}$$



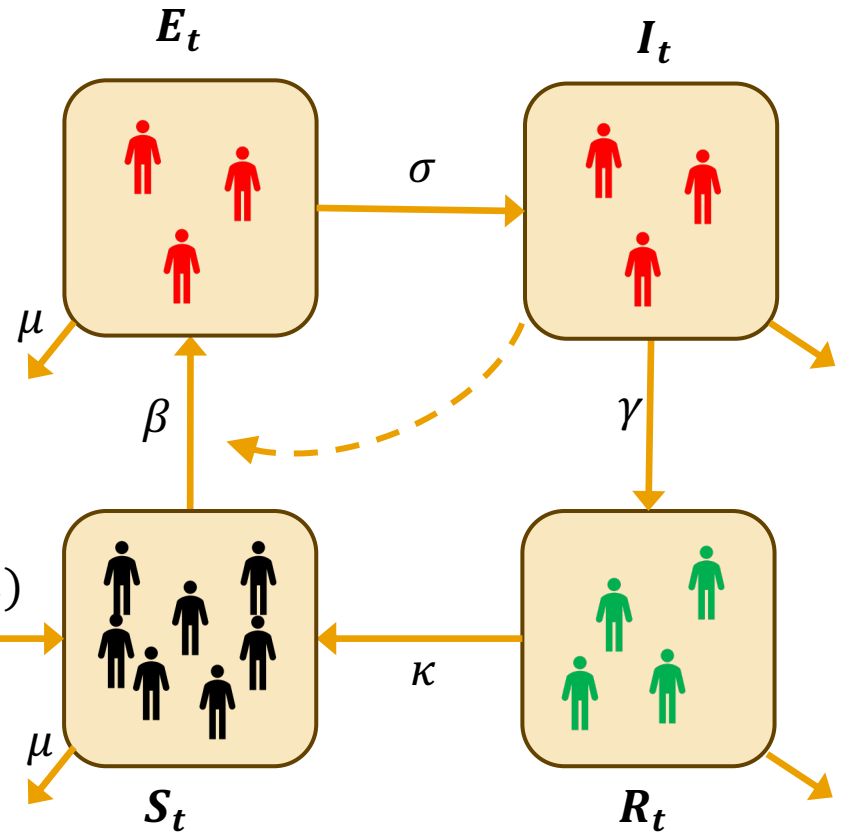
Basic reproductive number: SEIR

Define basic reproduction number:

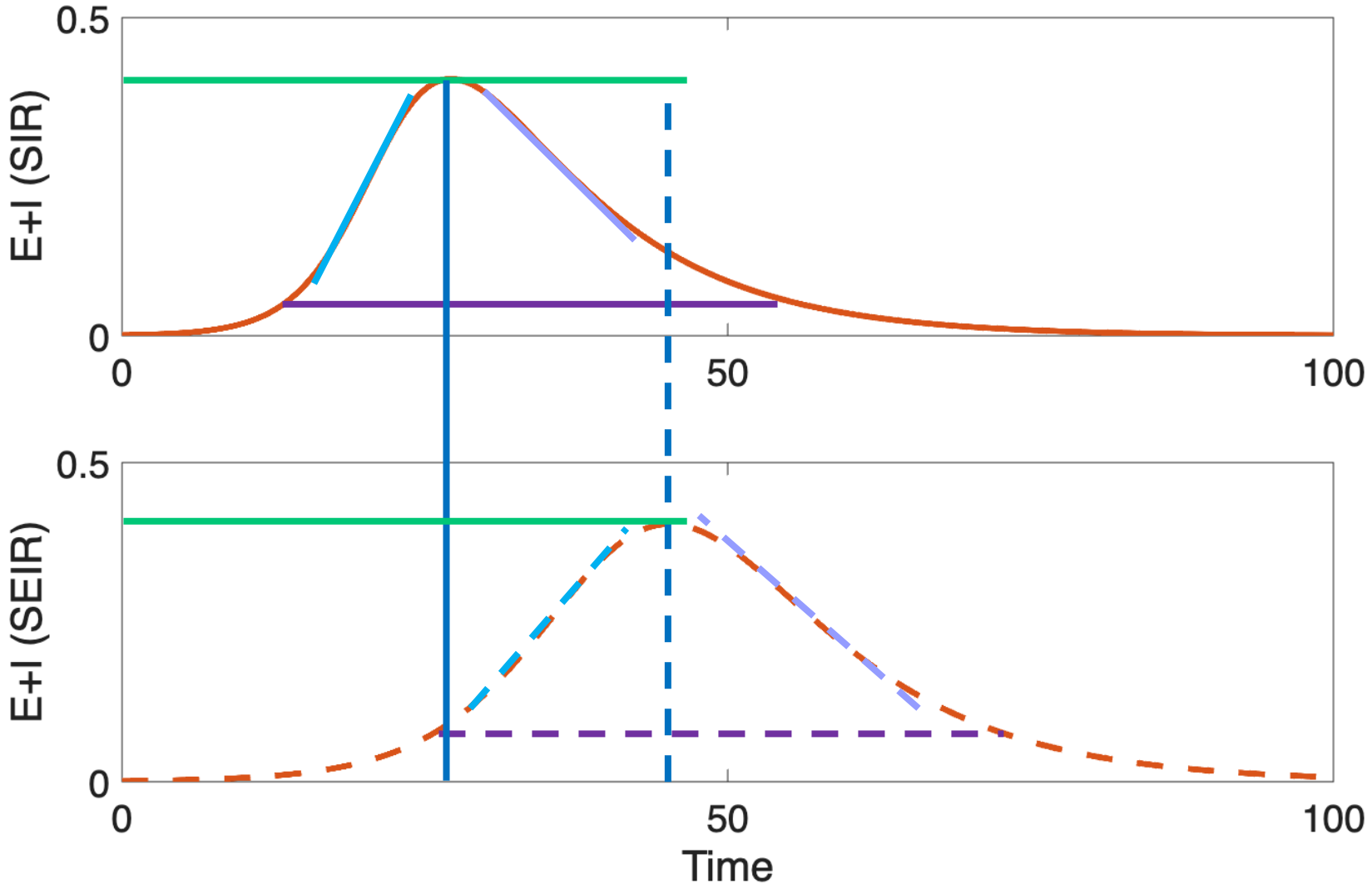
$$R_0 = \frac{\beta\sigma}{(\gamma+\mu)(\sigma+\mu)} > 1 \qquad \text{(as } S_0 \approx 1\text{)}$$

Interpretation:

$$R_0 = \underbrace{\beta}_{\text{New infections per unit time}} * \underbrace{\frac{1}{(\gamma+\mu)}}_{\text{Length of time infectious}} * \underbrace{\frac{\frac{\mu(S_t+E_t+I_t+R_t)}{\sigma}}{(\sigma+\mu)}}_{\text{Probability of surviving latent period}}$$



Differences in SIR and SEIR dynamics



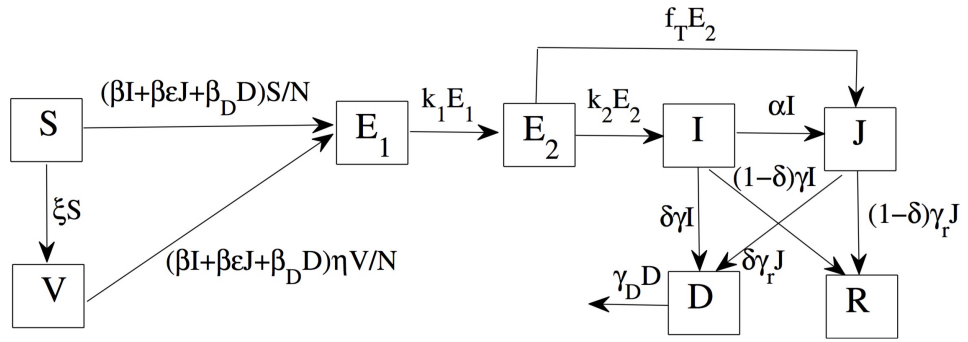
Differences in:

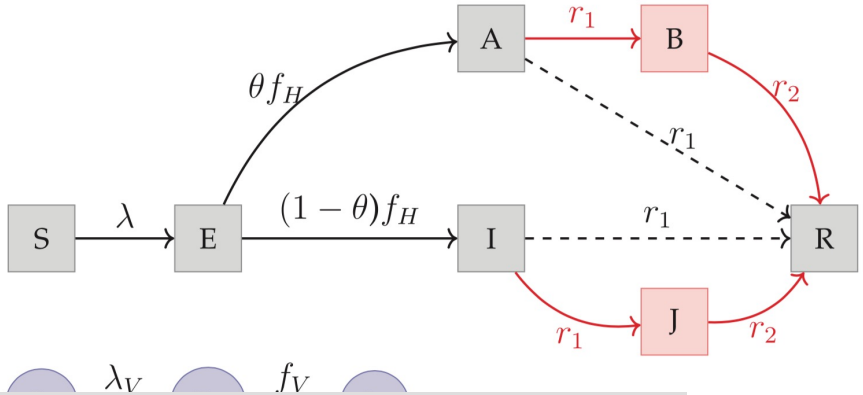
- Slope of spread
- Timing of peak
- Width of peak
- Slope of decline

No difference in:

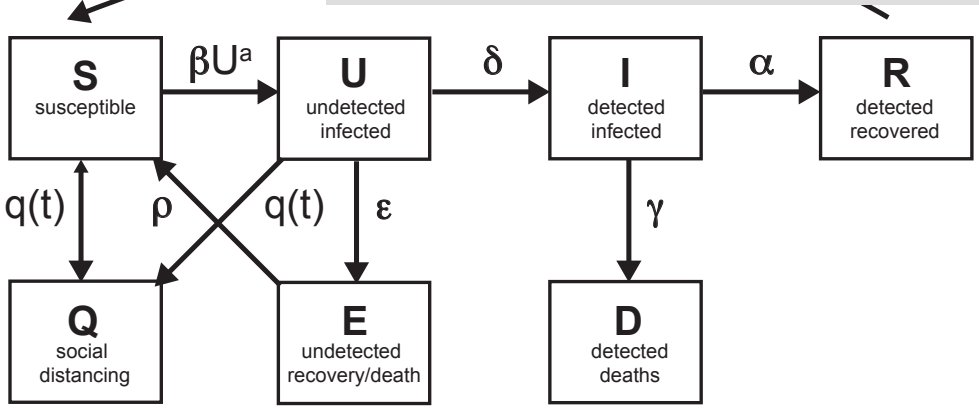
- Height of peak

Easy to add compartments and transitions



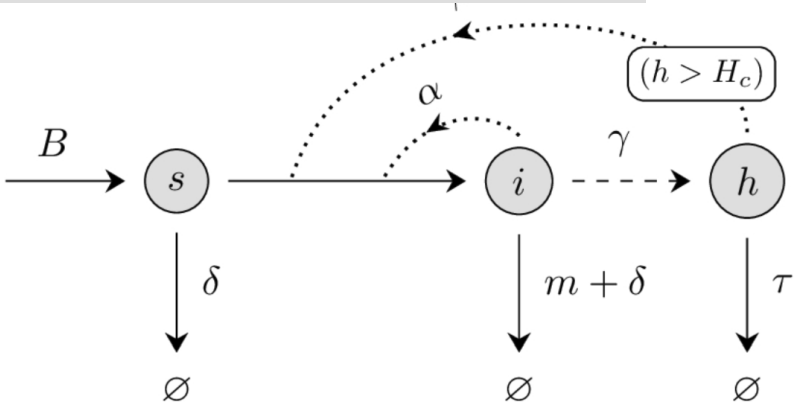


Math becomes more difficult!



Khan, Bussel, & Hussain, *arXiv*, 2020.

mics, 2017.



McClure and Powell, *Bulletin of Mathematical Biology*, 2025.

Simulation of SIR model in R with functions

Code files:

3_R_Intro_functions.R
4_Epi_functions.R
5_Intro_ID_SIR_functions.R

Simulation of SIR model in R with functions

Code file: *3_R_Intro_functions.R*

Introducing functions

Generally, functions take input and give output

```
# Functions can take no arguments
hello_world <- function() {
  return(print("Hello world!"))
}
hello_world()
```

Introducing functions

Generally, functions take input and give output

```r
# Functions can take no arguments
hello_world <- function() {
  return(print("Hello world!"))
}
hello_world()
```

```r
# Functions can take one argument
# Set default arguments for your functions
my_name_function <- function(name = "John Doe") {
  return(paste0("Hello, my name is ", name, "."))
}
my_name_function()
my_name_function("Mickey Mouse")
```

Simulation of SIR model in R with functions

Code file: *4_Epi_functions.R*

Function to initiate multiple populations

```
InitiatePop = function(IDs, initialInf_vec,totalPop_vec,beta_vec,gamma_vec,kappa_vec){
  list(
    ID = IDs,
    I = initialInf_vec,
    S = totalPop_vec - initialInf_vec,
    R = rep(0,length(initialInf_vec)),
    beta = beta_vec,
    gamma = gamma_vec,
    kappa = kappa_vec,
    nTotal = totalPop_vec,
    frac_I = initialInf_vec/totalPop_vec
  )
}
```

Function for epidemic events

```
##### Epidemic functions: infection, recovery, loss of immunity  ####
epidemic_timestep = function(HPop){

  for (i in 1:length(HPop$R)){
    # Epidemic
    newly_infected = HPop$I[i] * HPop$S[i] * HPop$beta[i] / HPop$nTotal[i]
    recovered_today_i= HPop$I[i] * HPop$gamma[i]
    losing_immunity_today_i = HPop$R[i] * HPop$kappa[i]

    # Updating epidemic equations
    HPop$S[i] = HPop$S[i] - newly_infected                      + losing_immunity_today_i
    HPop$I[i] = HPop$I[i] + newly_infected - recovered_today_i
    HPop$R[i] = HPop$R[i]                  + recovered_today_i - losing_immunity_today_i
  }
  HPop
}
```

Master function to run simulation

```
###### Master function without mobility  ####
runSimNoMobility = function(HPop,day_list) {

  epidemic_curve <- data.frame(day=c(),
                               inf=c(),
                               stringsAsFactors=FALSE)


  all_spread = matrix(0,length(day_list),length(HPop$I))
  all_spread_today = matrix(0,length(day_list),length(HPop$I))
  colnames(all_spread) = HPop$ID
  #print(all_dates)
  for (current_day in 1:length(day_list)){

    print(day_list[current_day])

    HPop = epidemic_timestep(HPop)

    epidemic_curve = rbind(epidemic_curve,data.frame(day = day_list[current_day], inf = sum(HPop$I)))
    all_spread[current_day,] = HPop$I

  }
  all_spread_2 = data.frame(day = day_list)
  all_spread_2 = cbind(all_spread_2,all_spread)
  list(HPop = HPop,epidemic_curve = epidemic_curve,all_spread=all_spread_2)
}
```

Simulation of SIR model in R with functions

Code file: *5_Intro_ID_SIR_functions.R*

Load functions from another file

- Files must be in the same folder, unless a path is given.

```
# source file with functions
source("Sat_PM/4_Epi_functions.R")
```

- This loads the following functions:
  - InitiatePop
  - epidemic_timestep
  - runSimNoMobility
  - movement_timestep (Note: we did not discuss!)
  - runSim (Note: we did not discuss!)

Setting up the information for each patch

- These 5 patches will be nearly identical! They differ in total population size.

```
# define
n = 5 # number of patches
# create n identical patches
IDs = seq(1,n) # ID of patch
initialInf_vec = rep(1,n) # initial number of infected individuals
totalPop_vec = sample(10000,n) # total population size
beta_vec = rep(2/3,n) # transmission parameter
kappa_vec = rep(1/30,n) # waning immunity parameter
gamma_vec = rep(1/3,n) # recovery parameter
day_list = seq(0,50,by=1) # days for simulation
```

Running the simulation

```
# initialize populations
HPop1 <- InitiatePop(IDs,initialInf_vec,totalPop_vec,beta_vec,gamma_vec,kappa_vec)
```

Running the simulation

```
# initialize populations
HPop1 <- InitiatePop(IDs,initialInf_vec,totalPop_vec,beta_vec,gamma_vec,kappa_vec)
```

```
# run simulation
HPop1 <- runSimNoMobility(HPop1,day_list)
```

Plotting the output

- Plot the outbreaks from each of the patches

```
# pivot to long data frame
df.long = pivot_longer(HPop1$all_spread, cols = c(colnames(HPop1$all_spread[,1:n+1])), # columns you want to make longer
                       names_to = "compartment", # new column name that contains the names of the old columns (S, I, R)
                       values_to = "population") # values from the old columns (S, I, R)


# plot
ggplot(data = df.long, aes(x = Iday, y = population, color = compartment) ) + # Data and aesthetics
  geom_line() +  # Line plot
  labs(x = "Time, days",                                  # x label
       y = "Population size",                             # y label
       title = "SIR model",                               # plot title
       color = "") +                                      # legend title
  theme_minimal()
```