Ordinary Differential Equations

Modern Techniques in Modelling

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Outline for session 1

- What are ordinary differential equations (ODEs)?
- How do we use ODEs to model an epidemic?
- Using the R package `deSolve`
  - Practical: SI, SIR, SEIR models in R

Ordinary differential equations

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Reminder: Difference equations

In the previous session, we explored difference equations:

$S(t+1)$	$= \quad S(t)$	$- \beta\, S(t)\, I(t)$
$I(t+1)$	$= \quad I(t)$	$+ \beta\, S(t)\, I(t) - \gamma\, I(t)$
$R(t+1)$	$= \quad R(t)$	$+ \gamma\, I(t)$
next value	= current value	+ change in value

These changes are the interesting part –
they are what define the behaviour of the system.

Ordinary differential equations

Difference equations

$$
\begin{aligned}
S(t+1) &= S(t) - \beta\, S(t)\, I(t) \\
I(t+1) &= I(t) + \beta\, S(t)\, I(t) - \gamma\, I(t) \\
R(t+1) &= R(t) + \gamma\, I(t)
\end{aligned}
$$

Ordinary differential equations have a similar structure, but only the rate of change is given:

$$
\begin{aligned}
dS(t)/dt &= -\beta\, S(t)\, I(t) \\
dI(t)/dt &= \beta\, S(t)\, I(t) - \gamma\, I(t) \\
dR(t)/dt &= \gamma\, I(t)
\end{aligned}
$$

The explicit dependence on time is often omitted
(e.g. *S* is written instead of *S*(*t*))

Ordinary differential equations

Difference equations

$$
\begin{aligned}
S(t+1) &= S(t) - \beta\, S(t)\, I(t) \\
I(t+1) &= I(t) + \beta\, S(t)\, I(t) - \gamma\, I(t) \\
R(t+1) &= R(t) + \gamma\, I(t)
\end{aligned}
$$

Ordinary differential equations have a similar structure, but only the rate of change is given:

$$
\begin{aligned}
dS/dt &= -\beta\, S\, I \\
dI/dt &= \beta\, S\, I - \gamma\, I \\
dR/dt &= \gamma\, I
\end{aligned}
$$

The explicit dependence on time is often omitted
(e.g. *S* is written instead of *S*(*t*))

Ordinary differential equations

Mathematically, dX/dt represents the derivative of *X* with respect to time (i.e. the rate at which *X* is changing over time).

For example, is *S* is the number of susceptibles, *t* is measured in days, and we have

$$dS/dt = -\beta\, S\, I = -2$$

then this means the number of susceptibles is currently shrinking at a rate of 2 people per day, and in one day's time will have around* 2 people fewer.

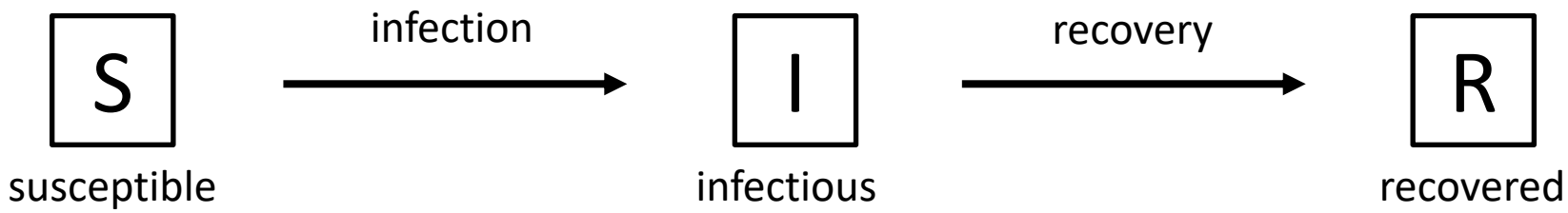
* not exactly 2, because over the course of that day, the value of $-\beta\, S\, I$ will change!

We will look at examples in the next section.

How do we model an infectious disease outbreak using ODEs?

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Turning a model diagram into ODEs



With variables:

S the number of susceptible people

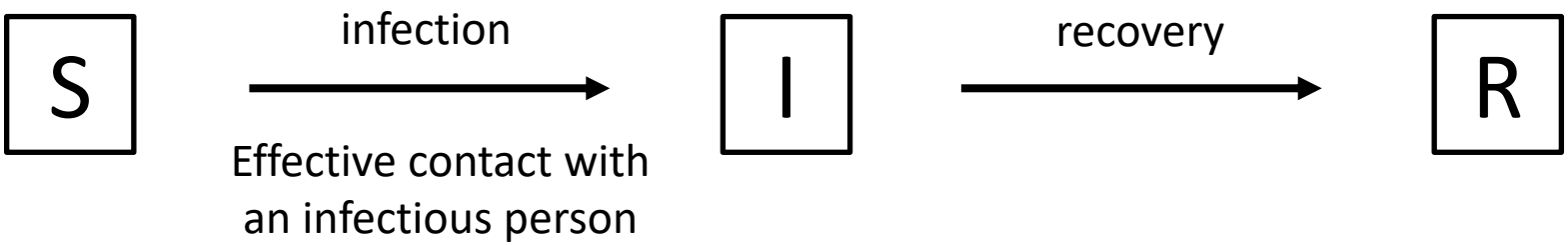
I the number of infectious people

R the number of recovered people

$N = S + I + R$ the total number of people

Let's look at the "infection" and "recovery" transitions in more detail.

Turning a model diagram into ODEs



Rate of effective contact with an infectious person:

A person contacts c people per day…

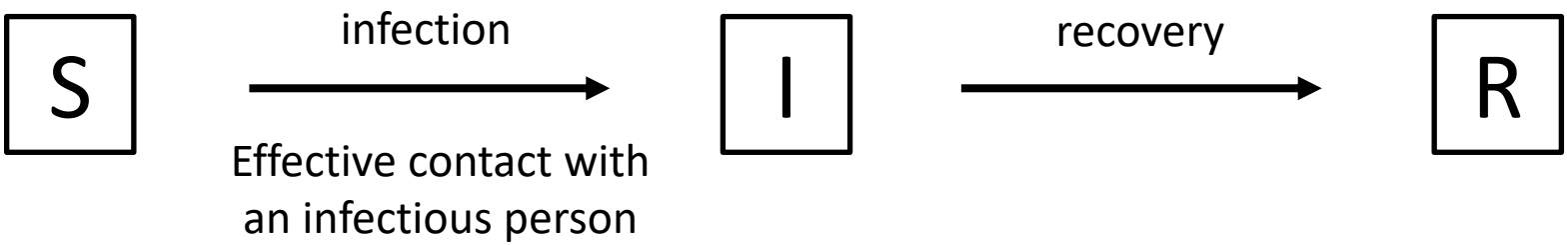
A fraction $I/(S+I+R) = I/N$ of these contacts are infectious…

A fraction p of these contacts with infectious people are effective…

And there are S susceptible people in total at risk of infection.

rate($\mathrm{S} \rightarrow \mathrm{I}$) =

Turning a model diagram into ODEs



Rate of effective contact with an infectious person:

A person contacts c people per day…

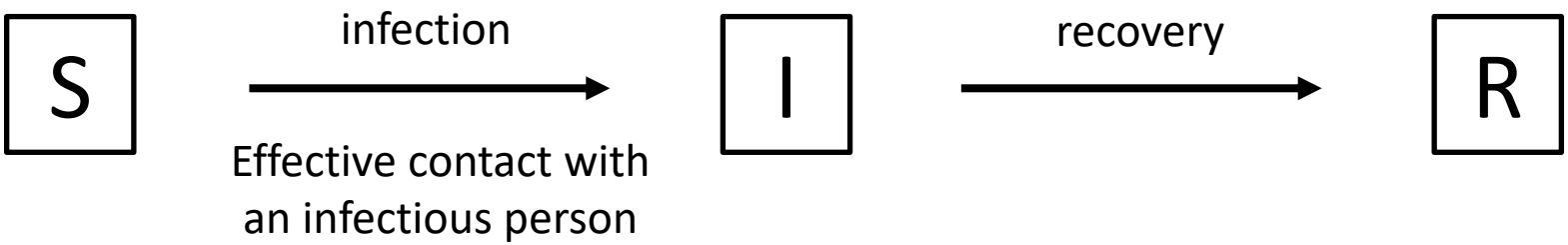
A fraction $I/(S+I+R) = I/N$ of these contacts are infectious…

A fraction p of these contacts with infectious people are effective…

And there are S susceptible people in total at risk of infection.

$$\text{rate}(\mathrm{S} \rightarrow \mathrm{I}) = \boxed{c} \times I/N \times \boxed{p} \times S$$

Turning a model diagram into ODEs



Rate of effective contact with an infectious person:

A person contacts c people per day…

A fraction $I/(S+I+R) = I/N$ of these contacts are infectious…

A fraction p of these contacts with infectious people are effective…

And there are S susceptible people in total at risk of infection.

$$\text{rate}(\mathrm{S} \rightarrow \mathrm{I}) = \beta \times I/N \times S \qquad (\beta = cp)$$

Turning a model diagram into ODEs

$$\boxed{S} \xrightarrow{\beta\, I/N\, S} \boxed{I} \xrightarrow{\text{recovery}} \boxed{R}$$

Rate of effective contact with an infectious person:

A person contacts c people per day…

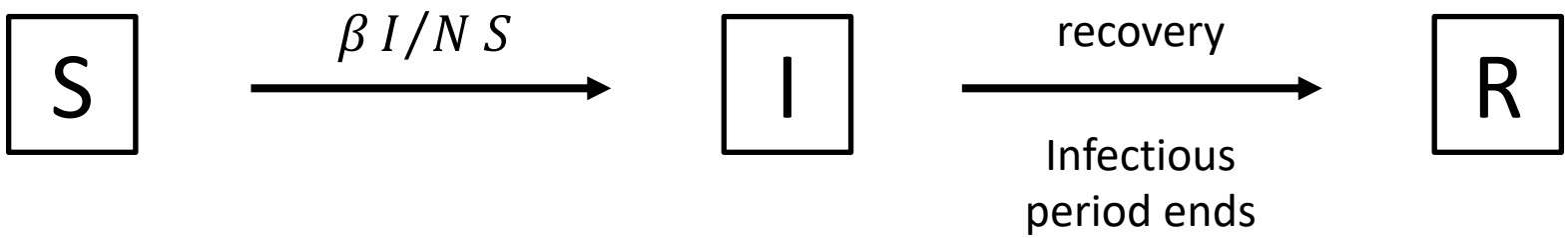
A fraction $I/(S+I+R) = I/N$ of these contacts are infectious…

A fraction p of these contacts with infectious people are effective…

And there are S susceptible people in total at risk of infection.

$$\text{rate}(\mathrm{S} \rightarrow \mathrm{I}) = \beta \times I/N \times S \qquad (\beta = cp)$$

Turning a model diagram into ODEs



Rate of recovery:

Suppose we know the infectious period lasts for d days…

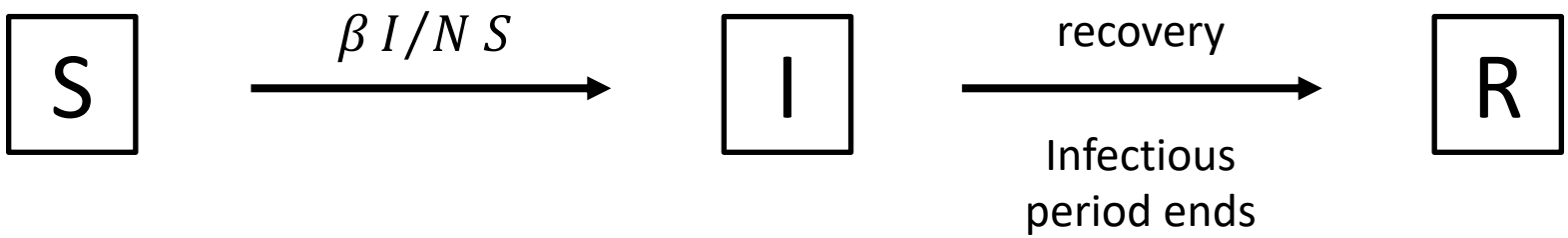
Then the rate of recovery is $1/d$ per day…

(e.g. if something happens 2x per day, on average it happens every 0.5 days)

And there are I infectious people in total at "risk" of recovery.

rate(I $\rightarrow$ R) =

Turning a model diagram into ODEs



Rate of recovery:

Suppose we know the infectious period lasts for d days…

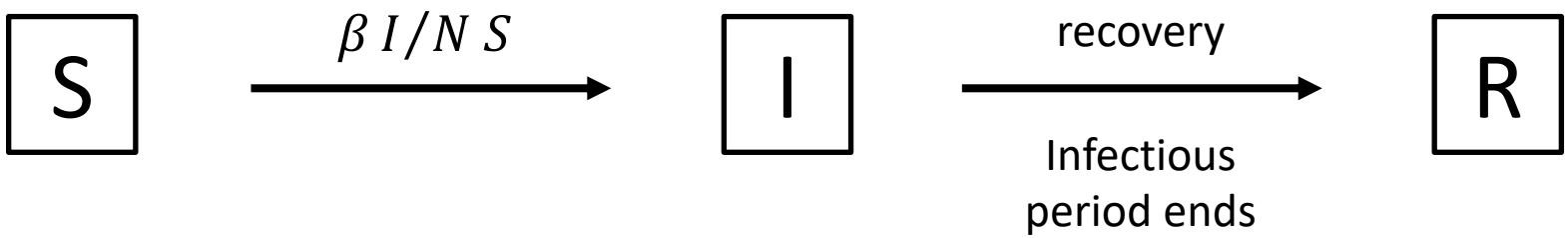
Then the rate of recovery is $1/d$ per day…

(e.g. if something happens 2x per day, on average it happens every 0.5 days)

And there are I infectious people in total at "risk" of recovery.

$$\text{rate}(\mathrm{I} \rightarrow \mathrm{R}) = \boxed{1/d} \times I$$

Turning a model diagram into ODEs



Rate of recovery:

Suppose we know the infectious period lasts for d days…

Then the rate of recovery is $1/d$ per day…

(e.g. if something happens 2x per day, on average it happens every 0.5 days)

And there are I infectious people in total at "risk" of recovery.

$$\text{rate}(\mathrm{I} \rightarrow \mathrm{R}) = \gamma \times I \qquad (\gamma = 1/d)$$

Turning a model diagram into ODEs

$$\boxed{S} \xrightarrow{\beta\, I/N\, S} \boxed{I} \xrightarrow{\gamma\, I} \boxed{R}$$

Rate of recovery:

Suppose we know the infectious period lasts for d days…

Then the rate of recovery is $1/d$ per day…

(e.g. if something happens 2x per day, on average it happens every 0.5 days)

And there are I infectious people in total at "risk" of recovery.

$$\text{rate}(\mathrm{I} \rightarrow \mathrm{R}) = \gamma \times I \qquad (\gamma = 1/d)$$

Turning a model diagram into ODEs

$$\boxed{S} \xrightarrow[\text{infection}]{\beta\, I/N\, S} \boxed{I} \xrightarrow[\text{recovery}]{\gamma\, I} \boxed{R}$$

Note that above, both transitions are specified as:

"rate per person per day" times "number of people at risk"

infection: $\beta\, I/N$ times S

recovery: γ times I

Often in model diagrams, the "number of people at risk" term is omitted, and implied by where the arrow is coming from.

Turning a model diagram into ODEs

$$\boxed{S} \xrightarrow[\text{infection}]{\beta\, I/N} \boxed{I} \xrightarrow[\text{recovery}]{\gamma} \boxed{R}$$

Note that above, both transitions are specified as:

"rate per person per day" times "number of people at risk"

infection: $\beta\, I/N$ times S

recovery: γ times I

Often in model diagrams, the "number of people at risk" term is omitted, and implied by where the arrow is coming from.

Turning a model diagram into ODEs

$$\boxed{S} \xrightarrow[\text{infection}]{\beta\, I/N} \boxed{I} \xrightarrow[\text{recovery}]{\gamma} \boxed{R}$$

To turn this into ODEs, we include each rate twice:

once negative for the "leaving" (subtracting from) compartment,

and once positive for the "entering" (adding to) compartment.

$$\begin{aligned} dS/dt &= -(\beta I/N)S \\ dI/dt &= (\beta I/N)S - \gamma I \\ dR/dt &= \gamma I \end{aligned}$$

Turning a model diagram into ODEs

$$\boxed{S} \xrightarrow[\text{infection}]{\beta\, I/N} \boxed{I} \xrightarrow[\text{recovery}]{\gamma} \boxed{R}$$

A full ODE model specification has the following elements:

System of ordinary differential equations

$$\begin{aligned} dS/dt &= -(\beta I/N)S \\ dI/dt &= (\beta I/N)S - \gamma I \\ dR/dt &= \gamma I \end{aligned}$$

$$N = S + I + R$$

Initial conditions

$$S(0) = 9{,}999$$
$$I(0) = 1$$
$$R(0) = 0$$

Parameters

$$\beta = 0.8$$
$$\gamma = 0.4$$

Times to solve system for

$$t \in \{0, 1, 2, \ldots, 60\}$$

Turning a model diagram into ODEs

$$S \xrightarrow[\text{infection}]{\beta\, I/N} I \xrightarrow[\text{recovery}]{\gamma} R$$

A full ODE

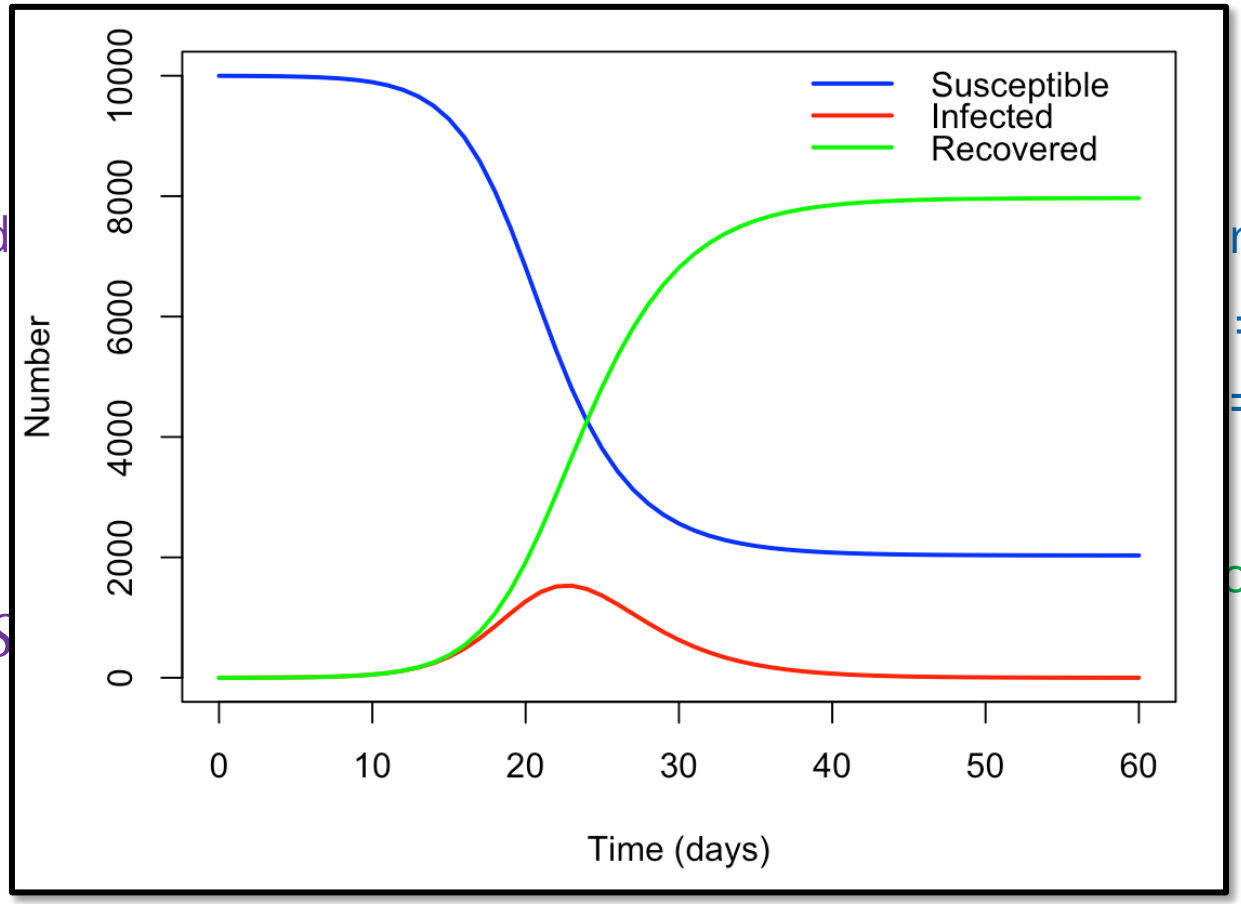
System of ord

$dS/dt =$

$dI/dt =$

$dR/dt =$

$N = S$



rameters

$= 0.8$

$= 0.4$

or

Solving ODE models in R with the `deSolve` package

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Using the R package `deSolve`

- R package which can numerically solve ODEs
- Provides the function `ode()` to solve your model
- You provide to `ode()`:
  - `y`, initial conditions
  - `times`, time points to solve the system for
  - `parms`, parameters
  - `func`, the system of ODEs as an R function
  - (optionally, others we will discuss later…)
- `ode()` returns a matrix with numerical solutions to the ODEs and the times

Susceptible Infected (SI) model

Individuals are either susceptible or infected:

$$\boxed{S} \xrightarrow{\beta\, I/N} \boxed{I}$$

Susceptible individuals become infected via transmission rate β.

$$\begin{aligned} dS/dt &= -(\beta I/N)S \\ dI/dt &= (\beta I/N)S \end{aligned}$$

Solving SI model using deSolve

- Provide to `ode()`:
  - `y`, initial conditions

Assume we have population of $N = 100$, with 1 infected individual:

```
N <- 100
I_0 <- 1
S_0 <- N - I_0

y <- c(S = S_0, I = I_0)
```

Solving SI model using deSolve

- Provide to `ode()`:
  - `times`, time points to solve the system for

Let's solve the equation over a period of 50 days, which we will write inside a vector as follows:

```
times <- seq(from = 0, to = 50, by = 1)
# or times <- 0:50
```

Solving SI model using deSolve

- Provide to `ode()`:
  - `parms`, parameters

We have just one parameter, the transmission rate:

```
parms <- c(beta = 0.4)
```

Solving SI model using deSolve

– Provide to `ode()`:

- `func`, the system of ODEs as an R function

```
SI_model <- function(times, state, parms)
{
  # Get variables
  S <- state["S"]
  I <- state["I"]
  N <- S + I
  # Get parameters
  beta <- parms["beta"]
  # Define differential equations
  dS <- -(beta * I / N) * S
  dI <- (beta * I / N) * S
  res <- list(c(dS, dI))
  return (res)
}
```

Solving SI model using deSolve

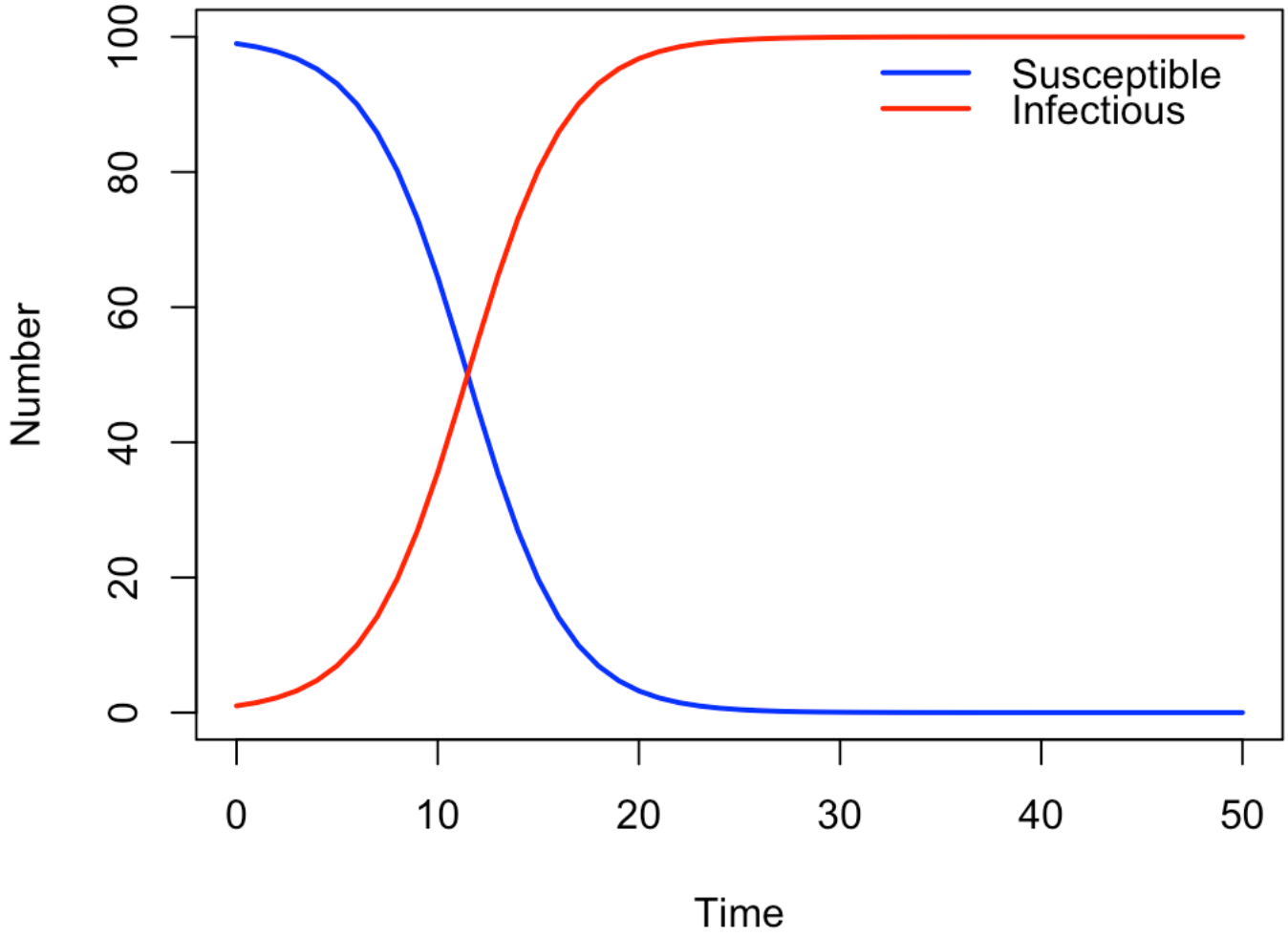
```
# Solve equations
output_raw <- ode(y = y, times = times,
                  func = SI_model, parms = parms)

# Convert matrix to data frame for easier manipulation
output <- as.data.frame(output_raw)

head(output)
```

```
##   time        S        I
## 1    0 99.00000 1.000000
## 2    1 98.60400 1.396000
## 3    2 98.05340 1.946605
## 4    3 97.28991 2.710090
## 5    4 96.23525 3.764747
## 6    5 94.78605 5.213953
```

Solving SI model using deSolve



Practical 1
Solving ODEs using `deSolve`

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Practical 1

- Objective: Solve SI, SIR, SEIR models using `deSolve`
- Answer questions 1, 2 and 3
- Question 4, adding vaccination, is optional.

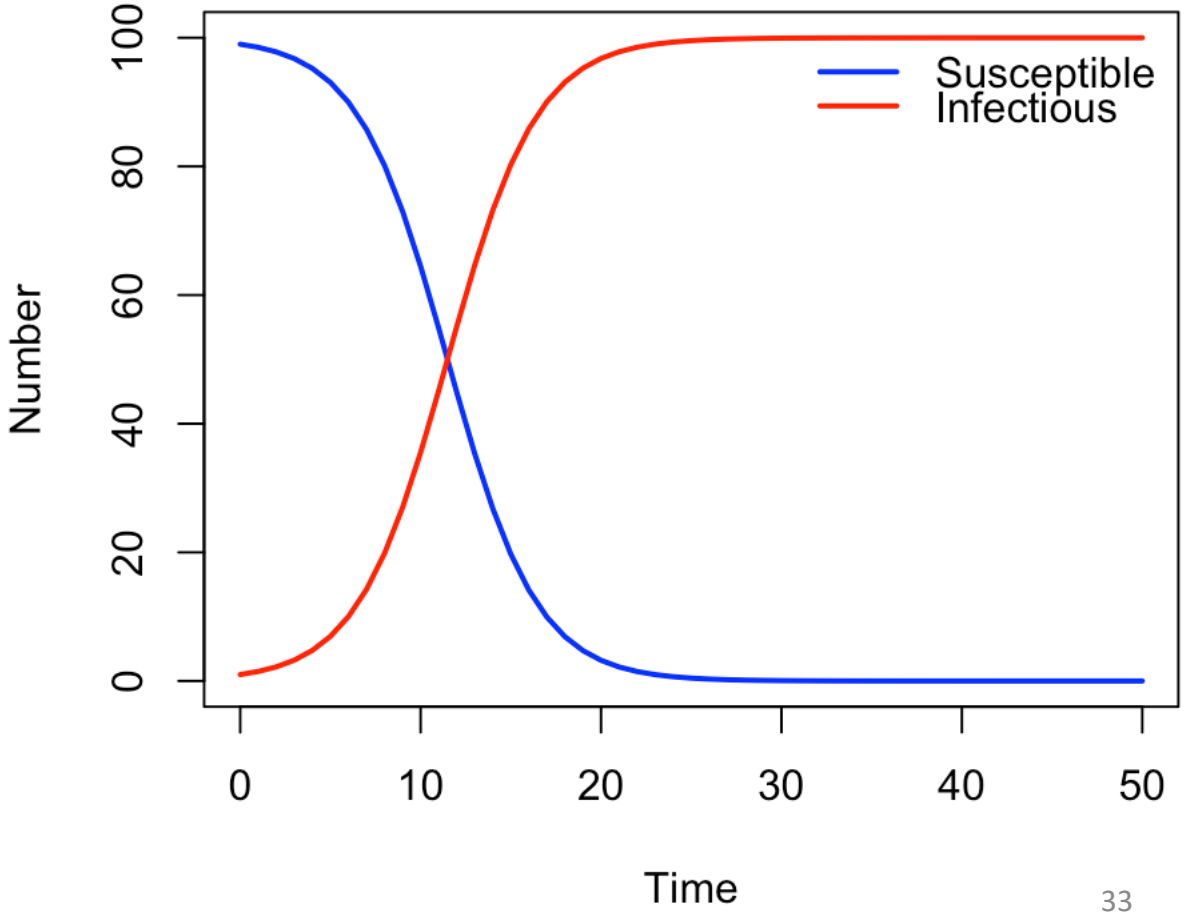
Note: If you are stuck with a grid of plots in R, use

```
par(mfrow = c(1,1))
```

to go back to single-plot mode.

Practical 1: SI model

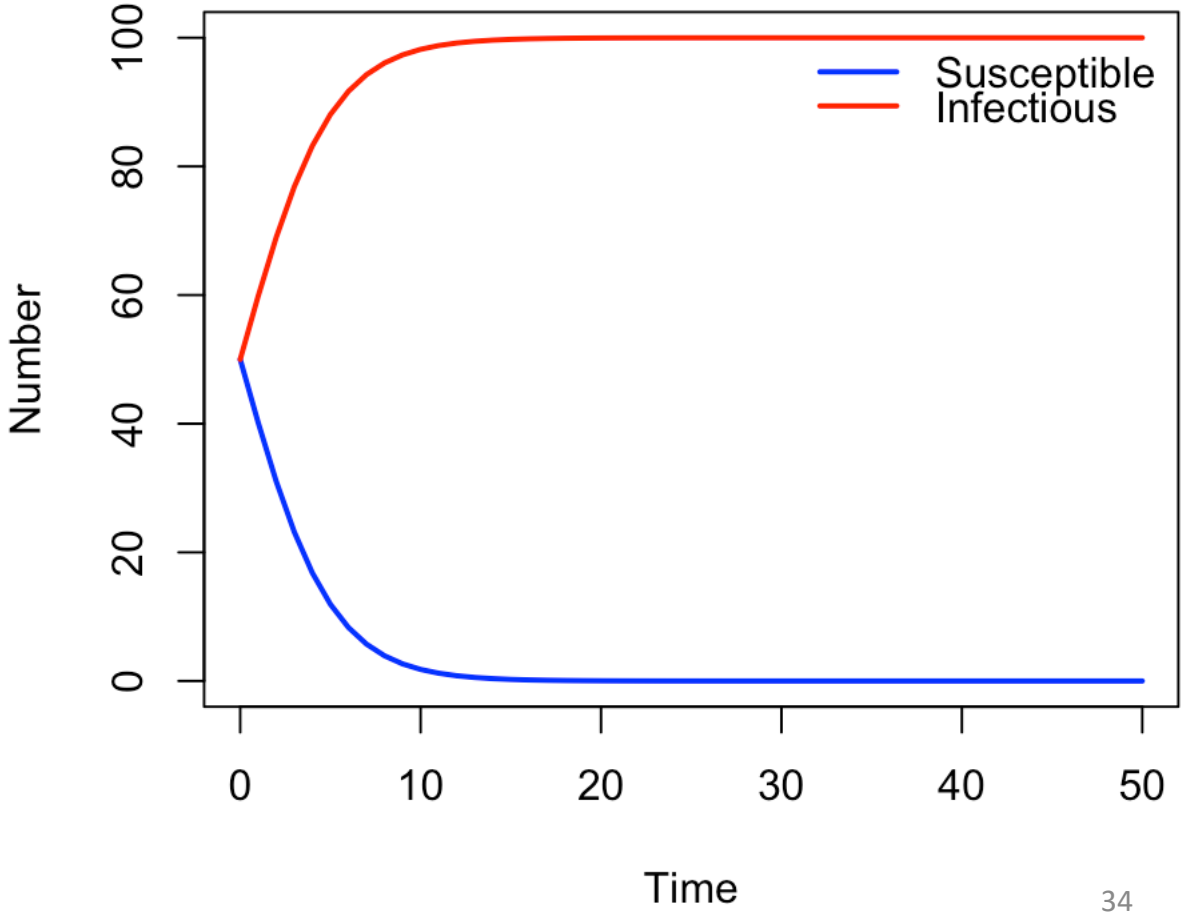
1a. Increase the initial number of infectious individuals. What happens to the output?



Practical 1: SI model

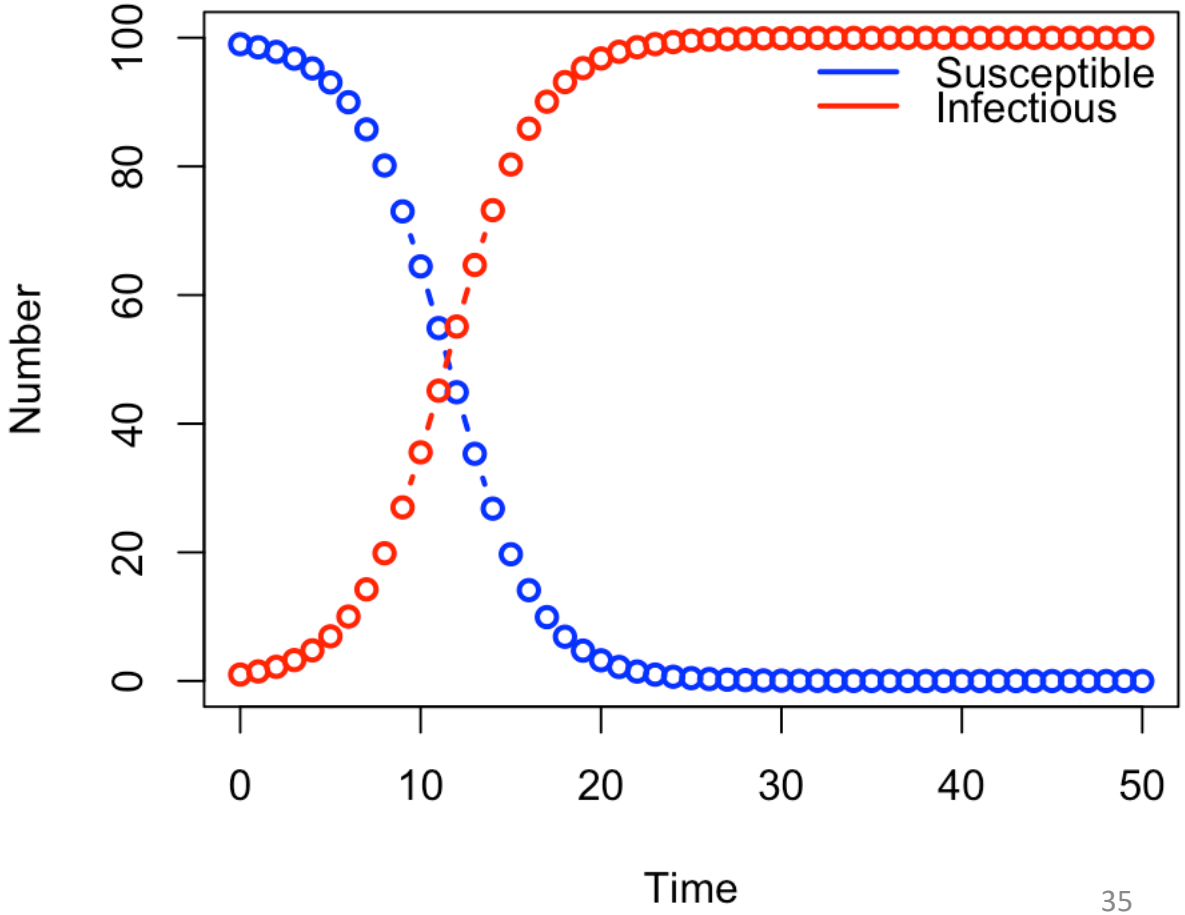
1a. Increase the initial number of infectious individuals. What happens to the output?

The number of infectious has a higher starting point, but the same growth rate from that level, and the same endpoint.



Practical 1: SI model

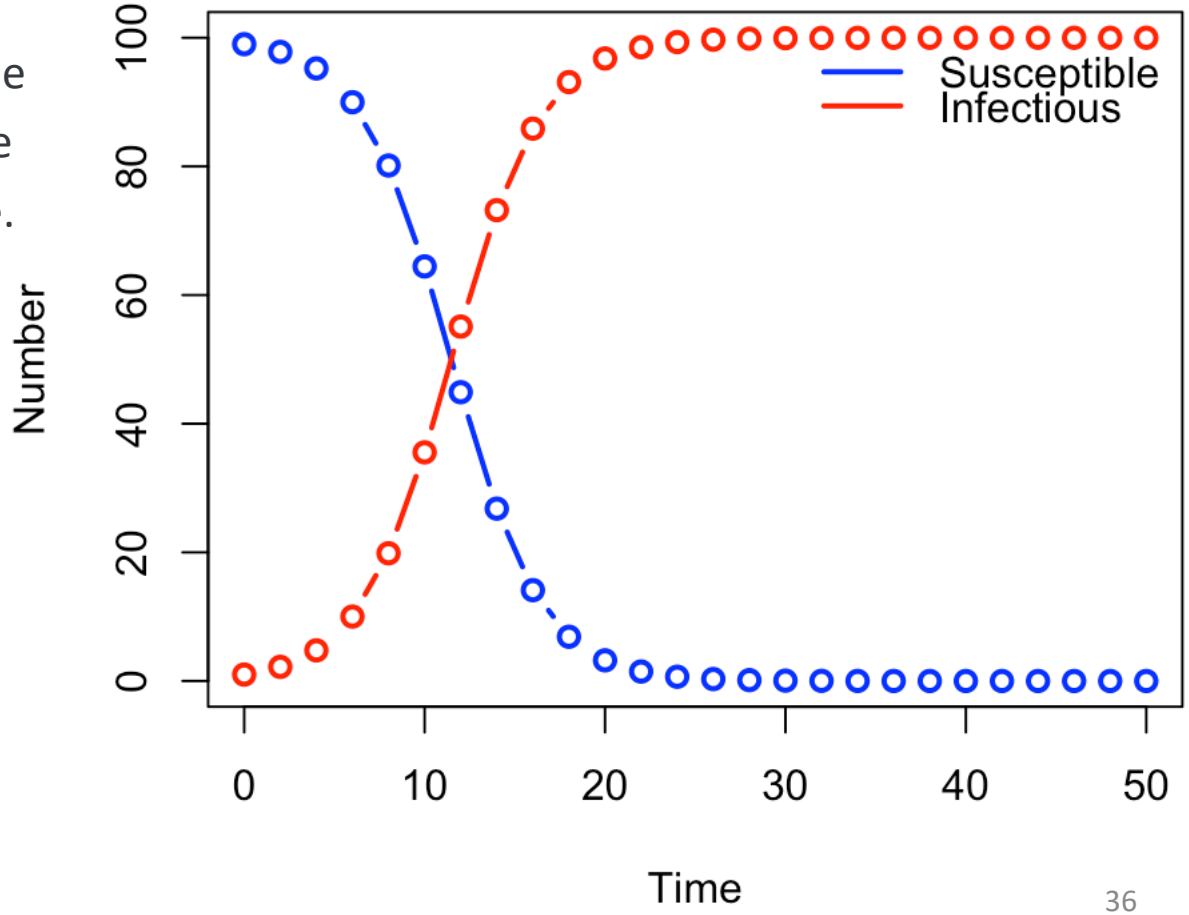
1c. Increase the value of the `by` argument (in the `times` vector).
What happens to the output?



Practical 1: SI model

1c. Increase the value of the `by` argument (in the `times` vector).
What happens to the output?

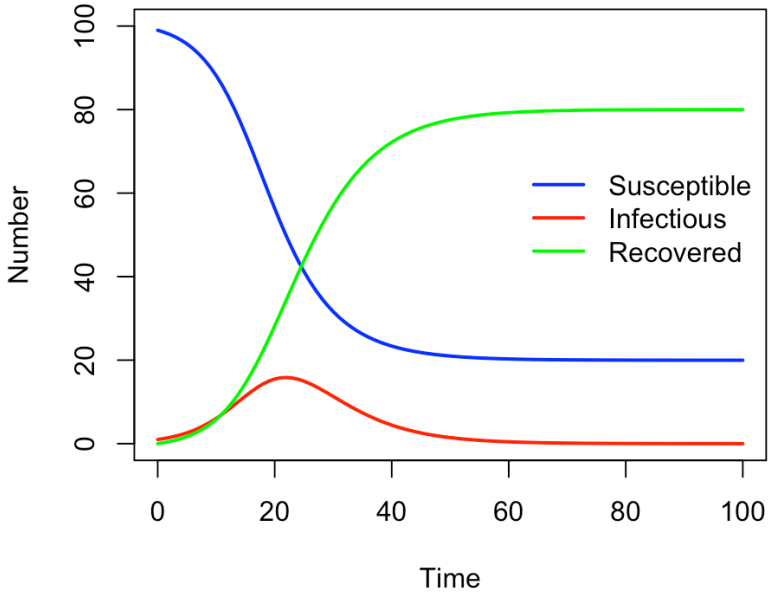
The solution points become more spaced out, but trace the same underlying curve.



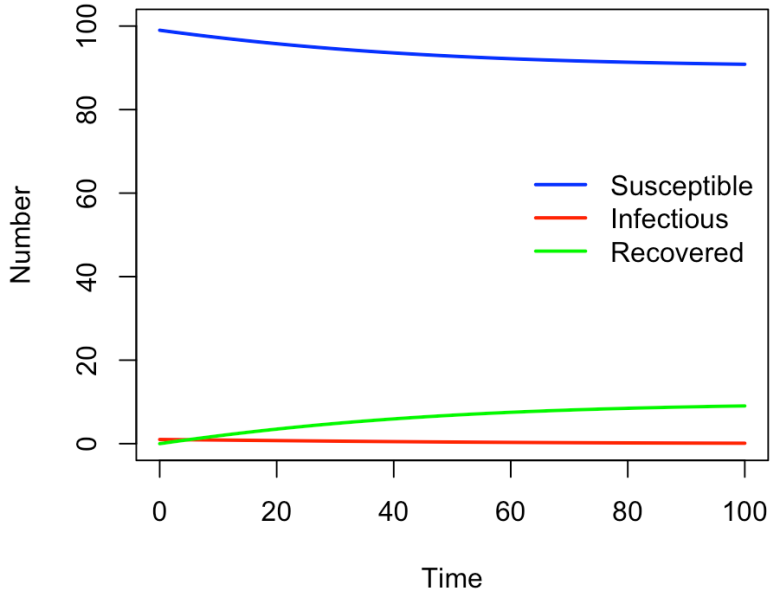
Practical 1: SIR model

2b. Change the value of the transmission rate so that the basic reproduction number is less than one, i.e. $R_0 < 1$. What happens to the output?

$\beta = 0.4, \gamma = 0.2$



$\beta = 0.19, \gamma = 0.2$

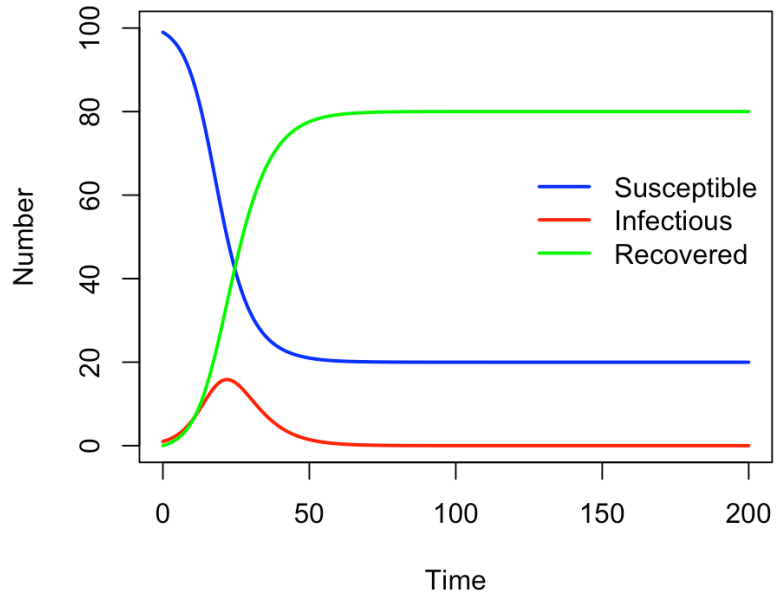


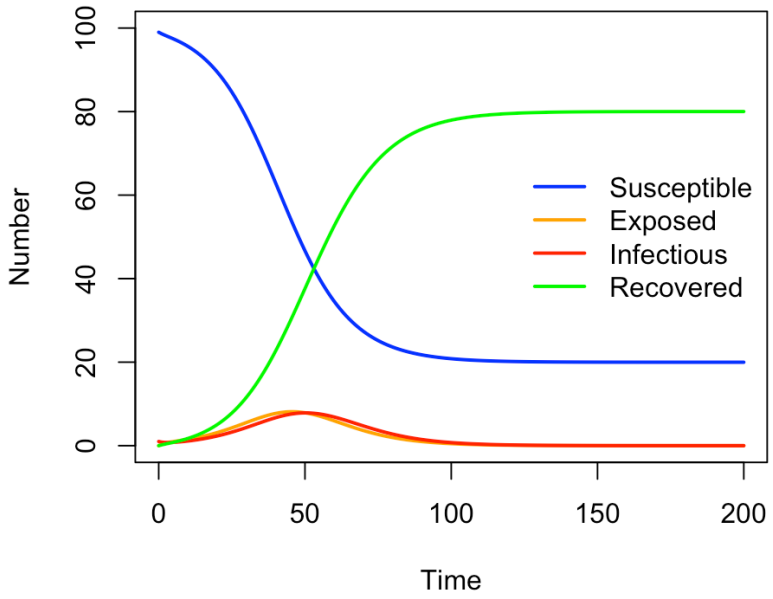
Recall that for an SIR model, the basic reproduction number $R_0 = \beta/\gamma$.

When $R_0 < 1$, the epidemic does not take off.

Practical 1: SEIR model

3b. How does the model output differ from the SIR model you coded previously?





Approximately the same number of people get infected, but the epidemic takes approximately twice as long; generation interval is twice as long.

See Wallinga and Lipsitch 2007, especially section 3a, for discussion of the generation interval, the growth rate and the reproduction number in epidemic models.

ODEs session 1 summary

- ODE models are specified in terms of state variables and their rates of change
- We have seen how to construct ODE systems starting from a flowchart-style model diagram
- To solve an ODE model, we need to provide initial conditions for the state variables, parameter values, and times over which to solve the model
- We have learned how to use `deSolve` to solve ODEs in R
- Next session: Advanced use of `deSolve`.

Ordinary Differential Equations, session 2

Modern Techniques in Modelling

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Outline for session 2

- Recap on ODEs
- How does numerical integration work?
- Advanced use of `deSolve`
  - Practical: time-varying parameters, events, and Rcpp

Ordinary differential equations: Recap

$$\boxed{S} \xrightarrow[\text{infection}]{\beta\, I/N} \boxed{I} \xrightarrow[\text{recovery}]{\gamma} \boxed{R}$$

A full ODE model specification has the following elements:

System of ordinary differential equations

$$\begin{aligned} dS/dt &= -(\beta I/N)S \\ dI/dt &= (\beta I/N)S - \gamma I \\ dR/dt &= \gamma I \end{aligned}$$

$$N = S + I + R$$

Initial conditions

$$S(0) = 9{,}999$$
$$I(0) = 1$$
$$R(0) = 0$$

Parameters

$$\beta = 0.8$$
$$\gamma = 0.4$$

Times to solve system for

$$t \in \{0, 1, 2, \ldots, 60\}$$

Starting from the initial conditions, we use **numerical integration** (e.g. with `deSolve`) to evaluate the variables at times *t*.

How does numerical integration of ODEs work?

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Numerical integration

Systems of ODEs define curves which usually don't have analytical solutions.

We use **numerical integration** to approximate these curves.

Usually done using **piecewise polynomials**.

Recall – examples of polynomials

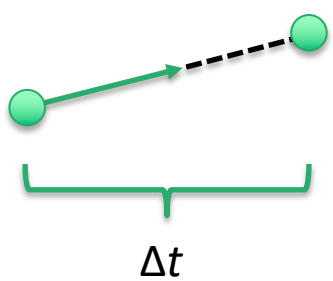
linear	$y = ax + b$
quadratic	$y = ax^2 + bx + c$
cubic	$y = ax^3 + bx^2 + cx + d$

Simplest example: **piecewise linear approximation** (Euler's method)

Euler's method

Choose a time step, Δt.

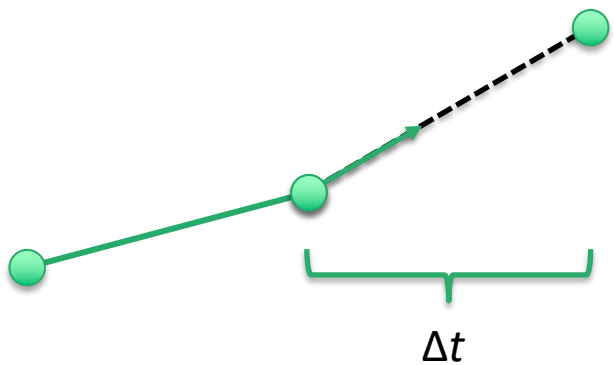
1. Start at initial point **y**(0), i.e. t = 0.
2. Use ODEs to get "slope" of function at this point (d**y** / dt).
3. Move forward to t' = t + Δt along a straight line with this "slope".
4. Repeat steps 2 - 3.



Euler's method

Choose a time step, Δt.

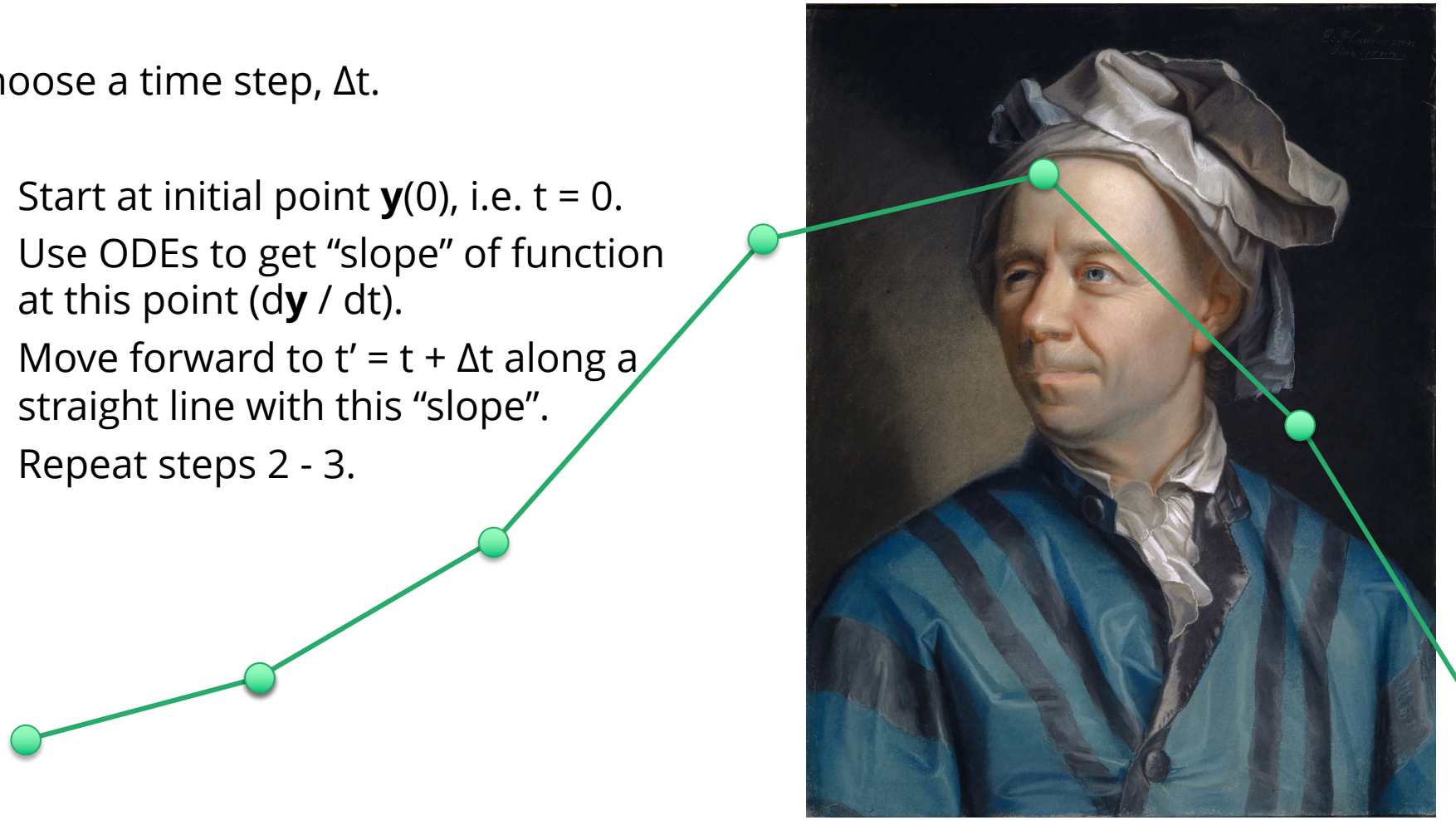
1. Start at initial point **y**(0), i.e. t = 0.
2. Use ODEs to get "slope" of function at this point (d**y** / dt).
3. Move forward to t' = t + Δt along a straight line with this "slope".
4. Repeat steps 2 - 3.



Euler's method

Choose a time step, Δt.

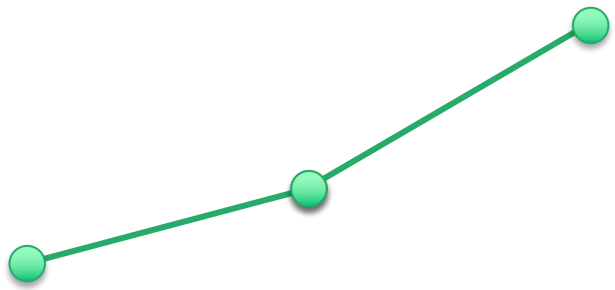
1. Start at initial point **y**(0), i.e. t = 0.
2. Use ODEs to get "slope" of function at this point (d**y** / dt).
3. Move forward to t' = t + Δt along a straight line with this "slope".
4. Repeat steps 2 - 3.



Note: This is very much like pretending your ODEs are difference equations!

Quadratic method

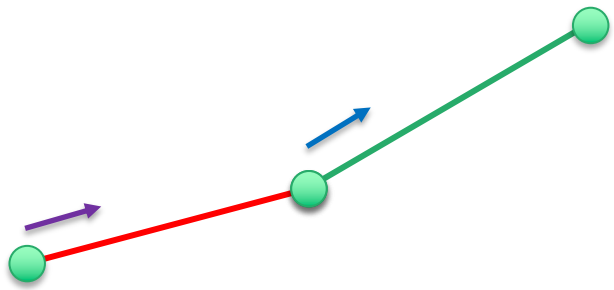
Instead of a piecewise linear function, we can use a piecewise quadratic function.



Quadratic method

Instead of a piecewise linear function, we can use a piecewise quadratic function.

Note that for piece 1, we have a slope measurement on the left side and a slope measurement on the right side.

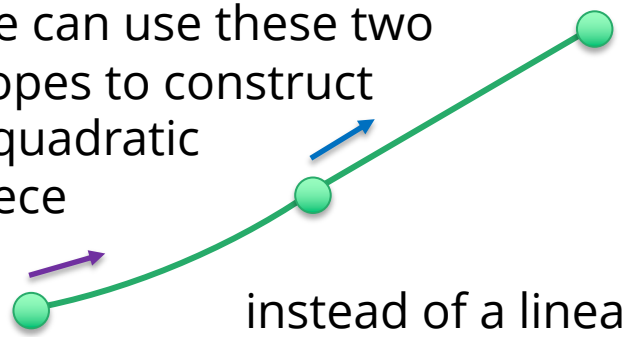


Quadratic method

Instead of a piecewise linear function, we can use a piecewise quadratic function.

Note that for piece 1, we have a slope measurement on the left side and a slope measurement on the right side.

We can use these two slopes to construct a quadratic piece



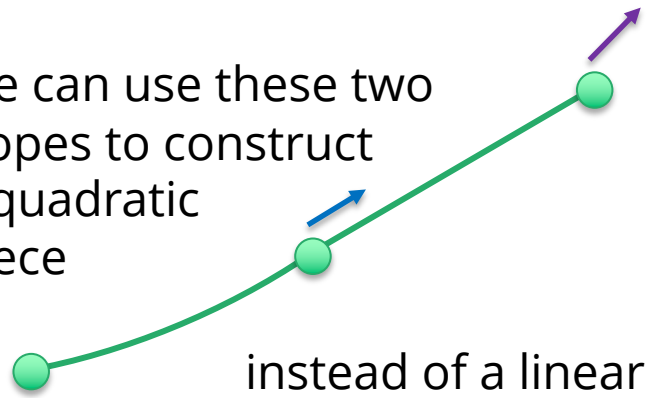
instead of a linear piece.

Quadratic method

Instead of a piecewise linear function, we can use a piecewise quadratic function.

Note that for piece 1, we have a slope measurement on the left side and a slope measurement on the right side.

We can use these two slopes to construct a quadratic piece



instead of a linear piece.

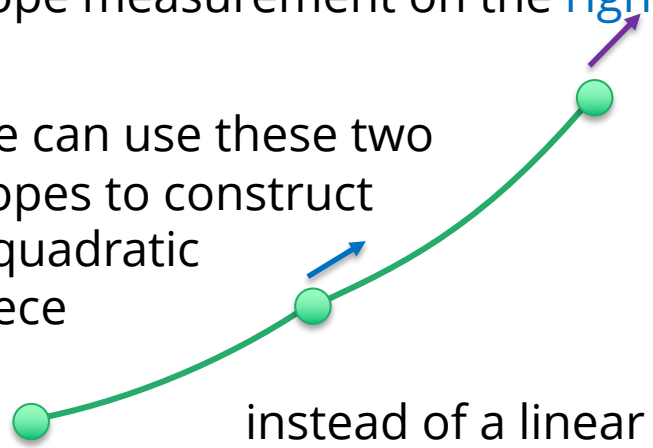
And so on...

Quadratic method

Instead of a piecewise linear function, we can use a piecewise quadratic function.

Note that for piece 1, we have a slope measurement on the left side and a slope measurement on the right side.

We can use these two slopes to construct a quadratic piece



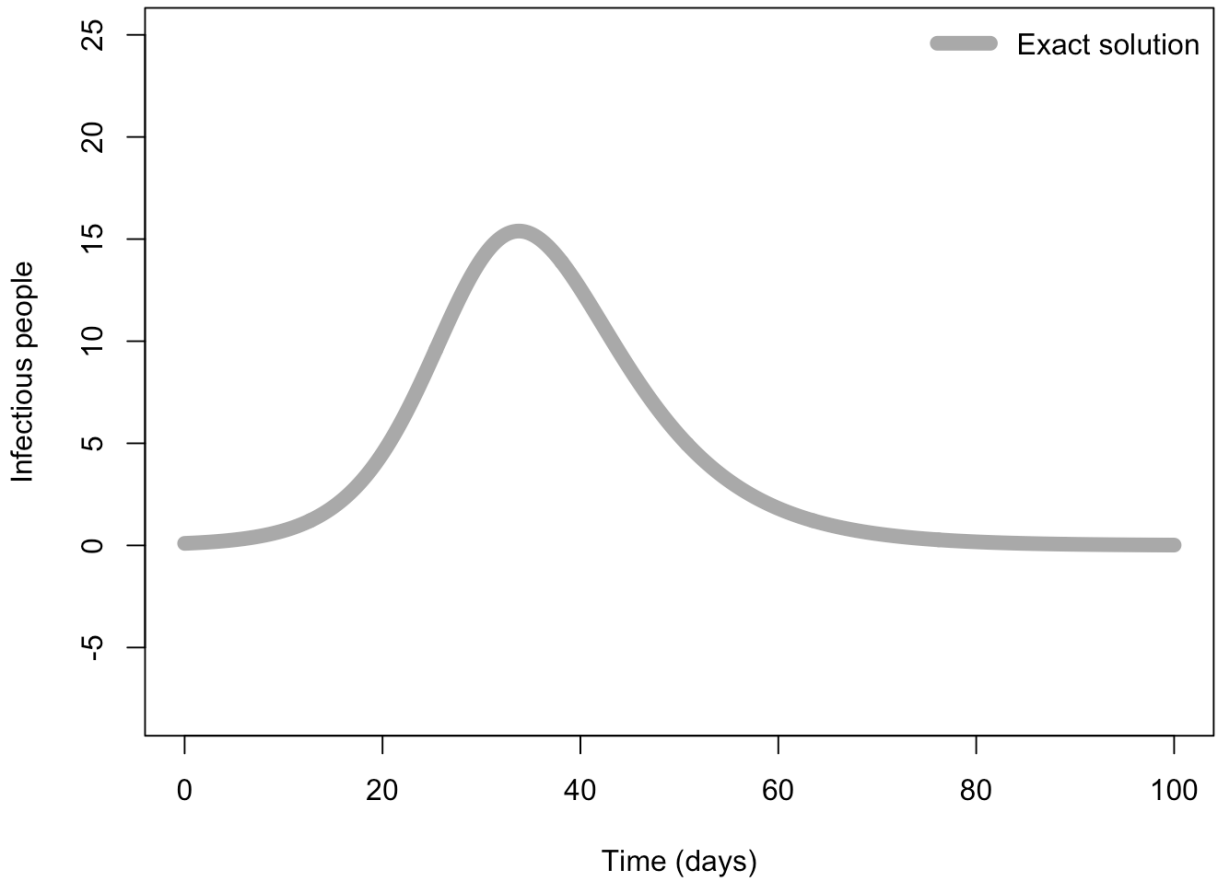
instead of a linear piece.

And so on...

Approximation methods compared

In general, the higher degree polynomials we use, the better our approximation, at a cost of increased computation.

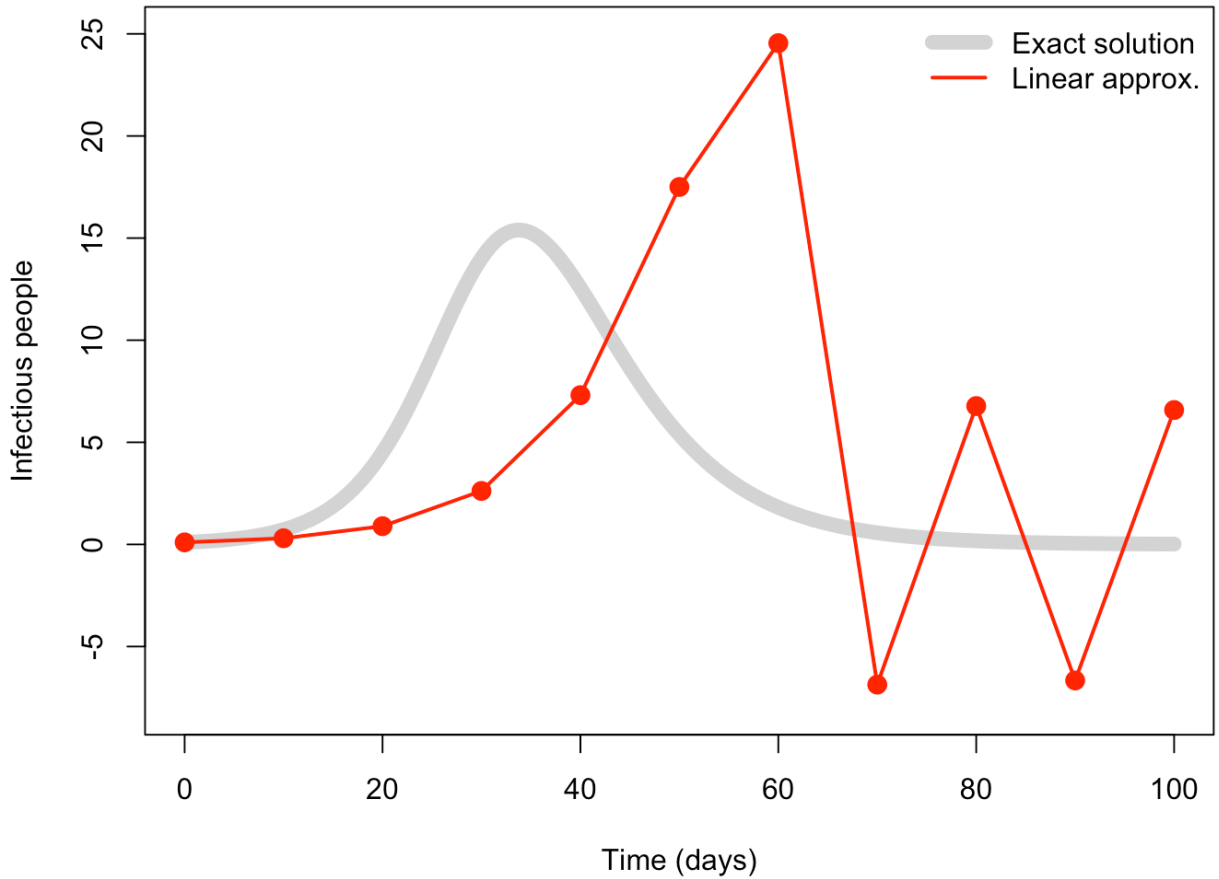
Example: SIR model, I compartment, Δt = 10



Approximation methods compared

In general, the higher degree polynomials we use, the better our approximation, at a cost of increased computation.

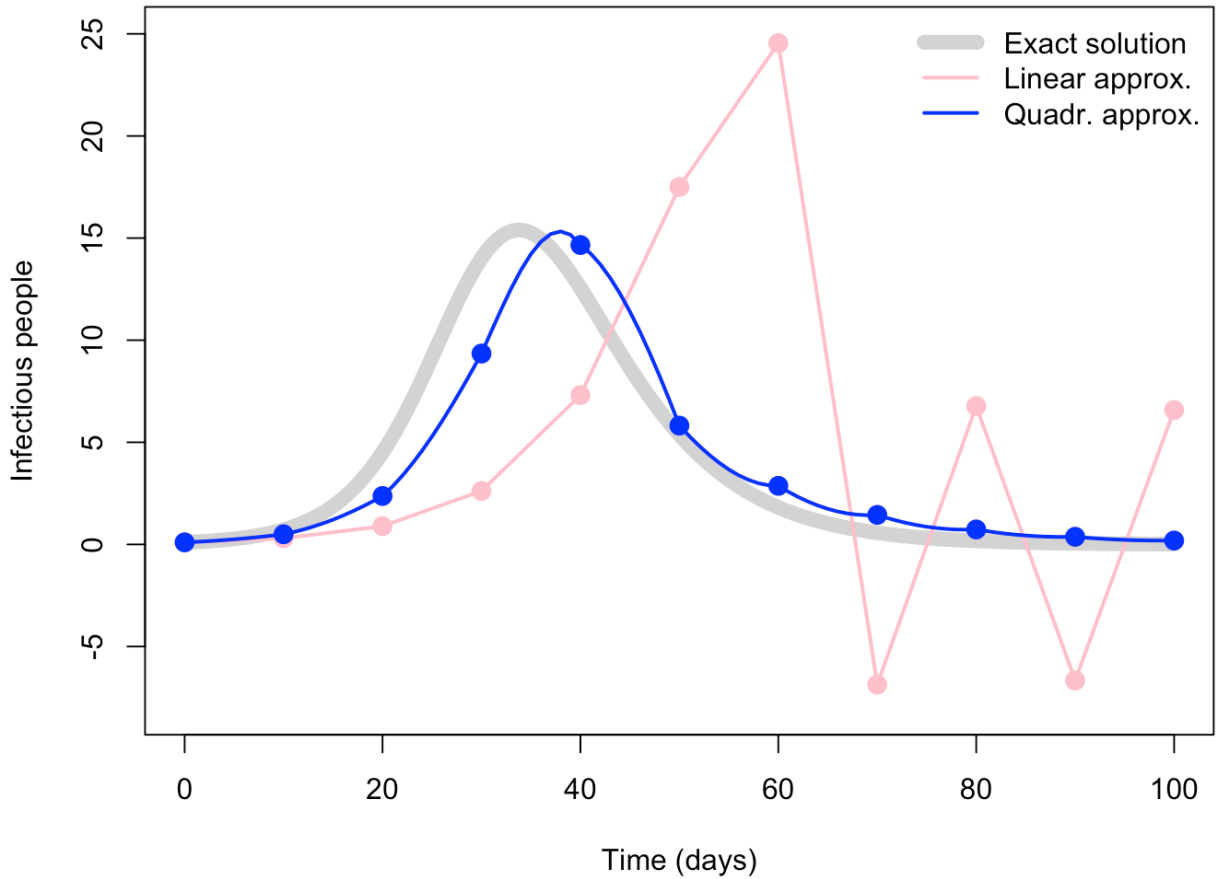
Example: SIR model, I compartment, Δt = 10



Approximation methods compared

In general, the higher degree polynomials we use, the better our approximation, at a cost of increased computation.

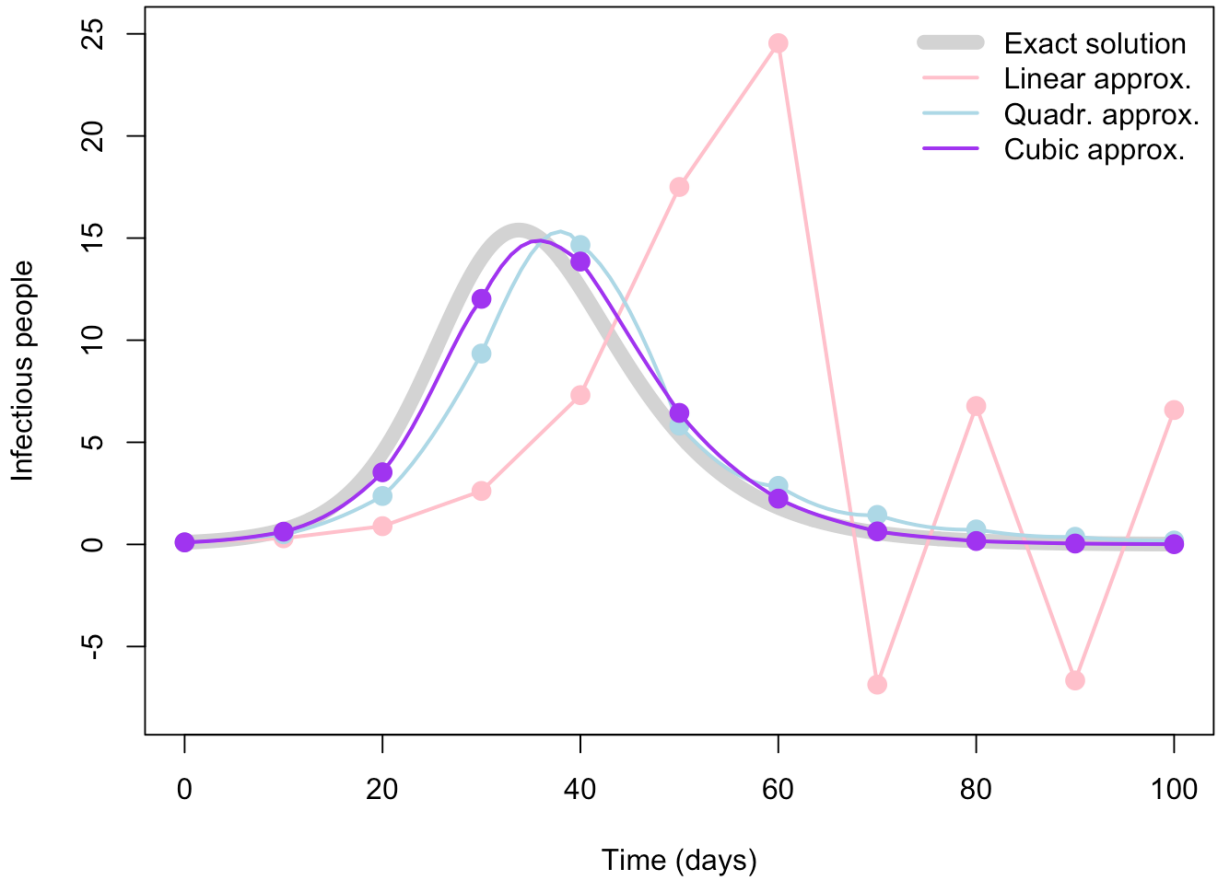
Example: SIR model, I compartment, Δt = 10



Approximation methods compared

In general, the higher degree polynomials we use, the better our approximation, at a cost of increased computation.

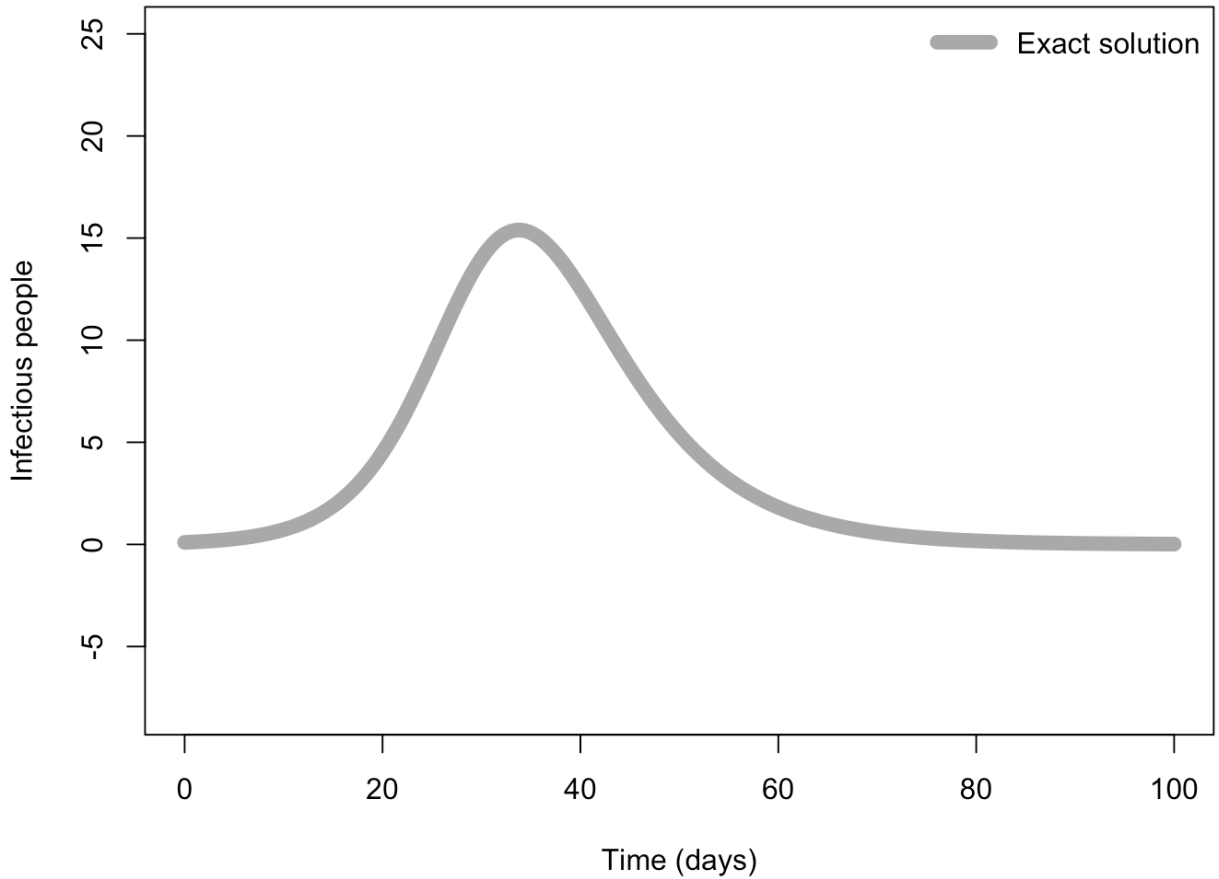
Example: SIR model, I compartment, Δt = 10



Approximation methods compared

In general, the higher degree polynomials we use, the better our approximation, at a cost of increased computation.

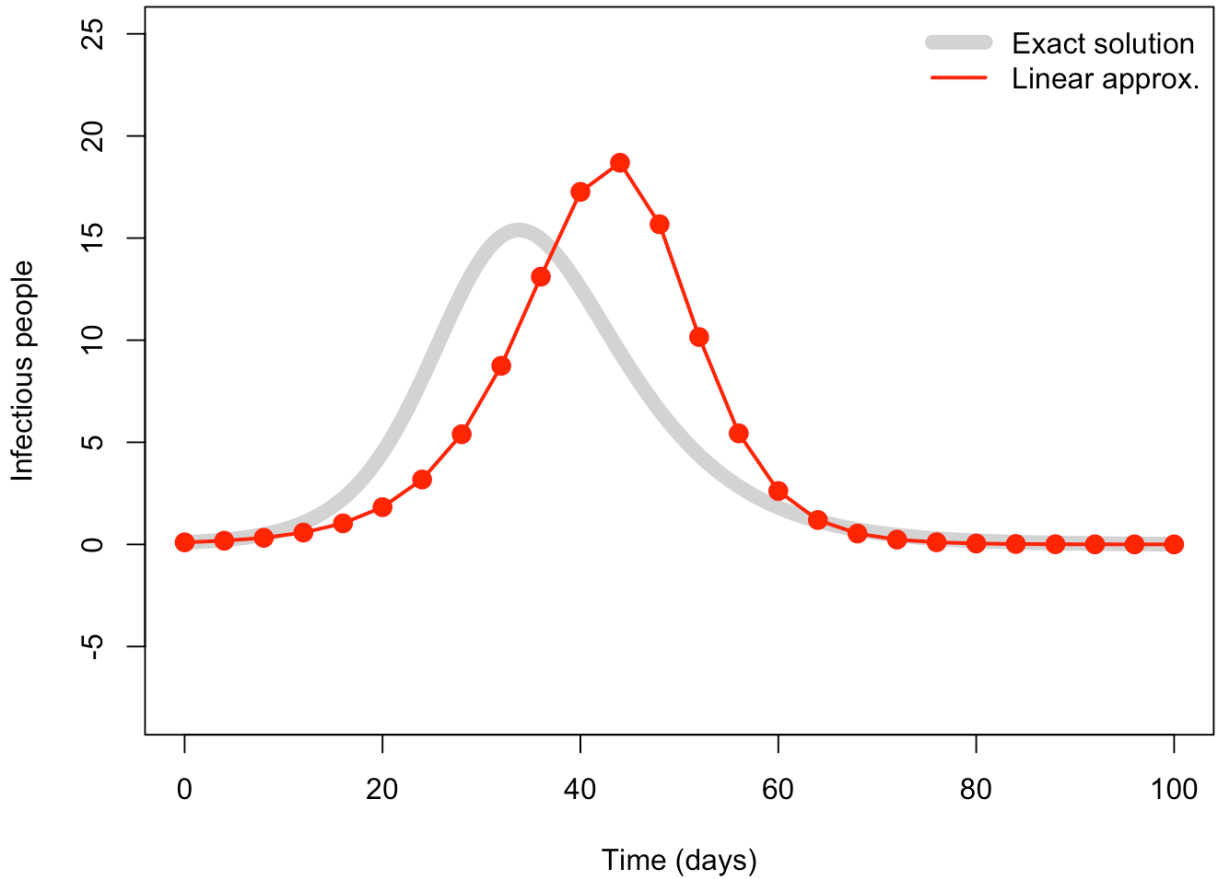
Example: SIR model, I compartment, Δt = 4



Approximation methods compared

In general, the higher degree polynomials we use, the better our approximation, at a cost of increased computation.

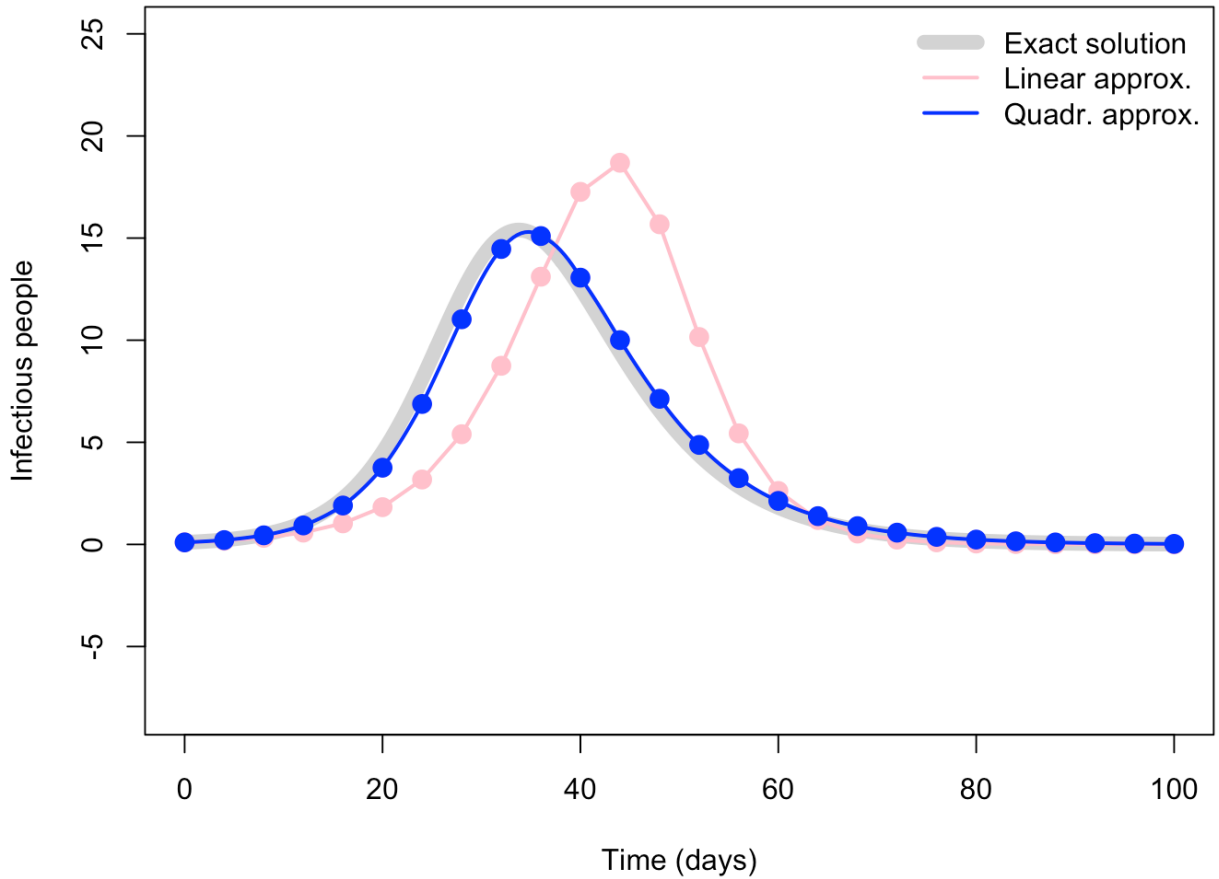
Example: SIR model, I compartment, Δt = 4



Approximation methods compared

In general, the higher degree polynomials we use, the better our approximation, at a cost of increased computation.

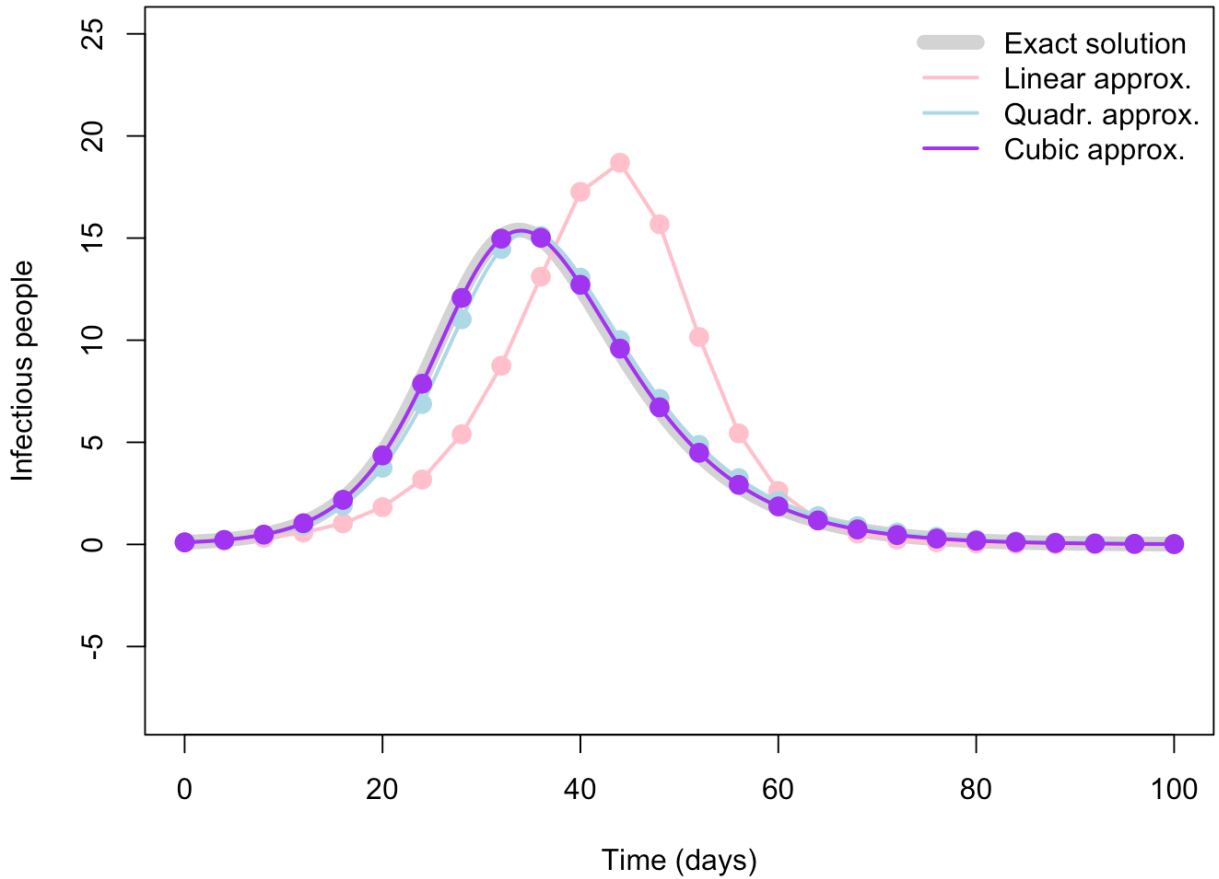
Example: SIR model, I compartment, Δt = 4



Approximation methods compared

In general, the higher degree polynomials we use, the better our approximation, at a cost of increased computation.

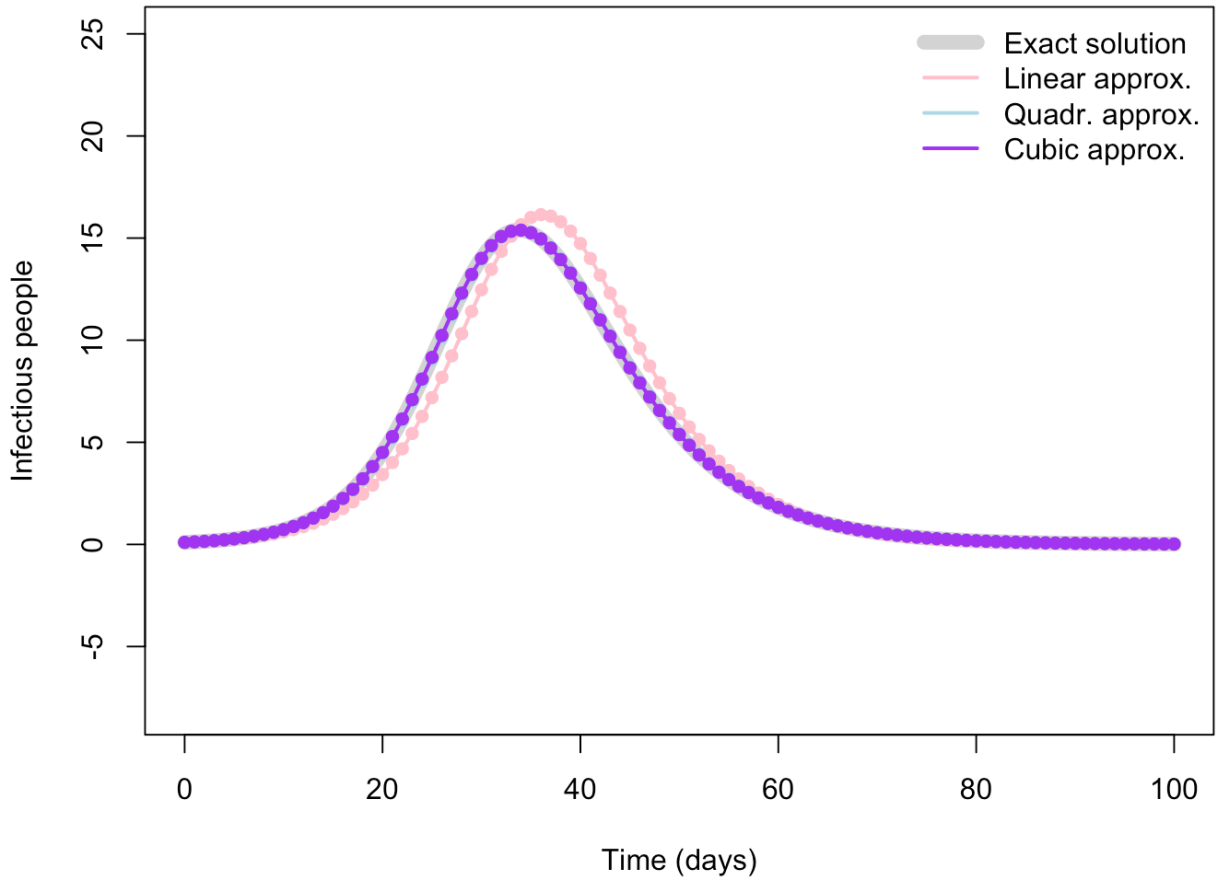
Example: SIR model, I compartment, Δt = 4



Approximation methods compared

In general, the higher degree polynomials we use, the better our approximation, at a cost of increased computation.

Example: SIR model, I compartment, Δt = 1



Usage in deSolve

In `deSolve`, we can specify what approximation method we want to use with the `method` argument to `ode()`:

```
# Solve equations
output_raw <- ode(y = y, times = times,
                  func = SI_model, parms = parms)
```

Usage in deSolve

In `deSolve`, we can specify what approximation method we want to use with the `method` argument to `ode()`:

```
# Solve equations
output_raw <- ode(y = y, times = times,
                  func = SI_model, parms = parms,
                  method = "euler")
```

Usage in deSolve

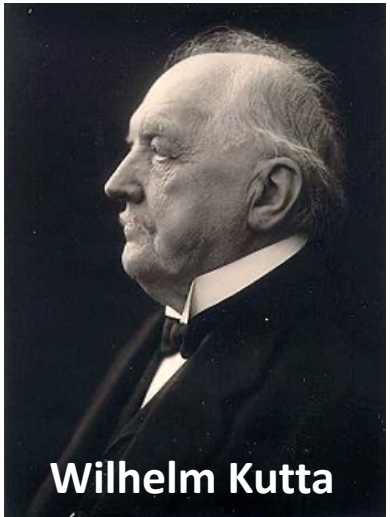
```
# Solve equations
output_raw <- ode(y = y, times = times,
                  func = SI_model, parms = parms,
                  method = "euler")
```

Some common methods:

`"euler"`	Euler's method	Don't use this!
`"rk4"`	4th-order Runge-Kutta	Commonly used, quite good
`"lsoda"`	Petzold & Hindmarsh	Robust (and the default); automatic step size


Leonhard Euler


Carl Runge


Wilhelm Kutta


Linda Petzold


Alan Hindmarsh

Advanced use of `deSolve`

Advanced use of deSolve package

Time-dependent changes to parameters

Events, with or without a “trigger”

Speeding up your model with Rcpp

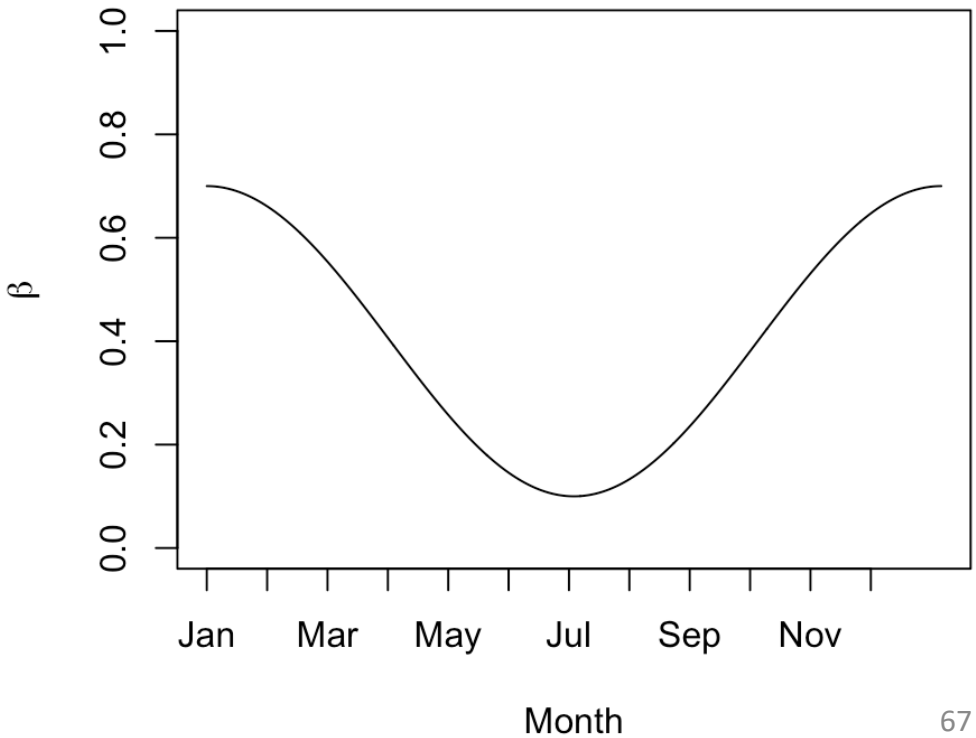
Time dependent parameters

Parameters, like the transmission rate, are “inputs” into the model.

We have been treating these as constants, but they can also vary with time.

What are some reasons the transmission rate might vary over time?

Seasonality…



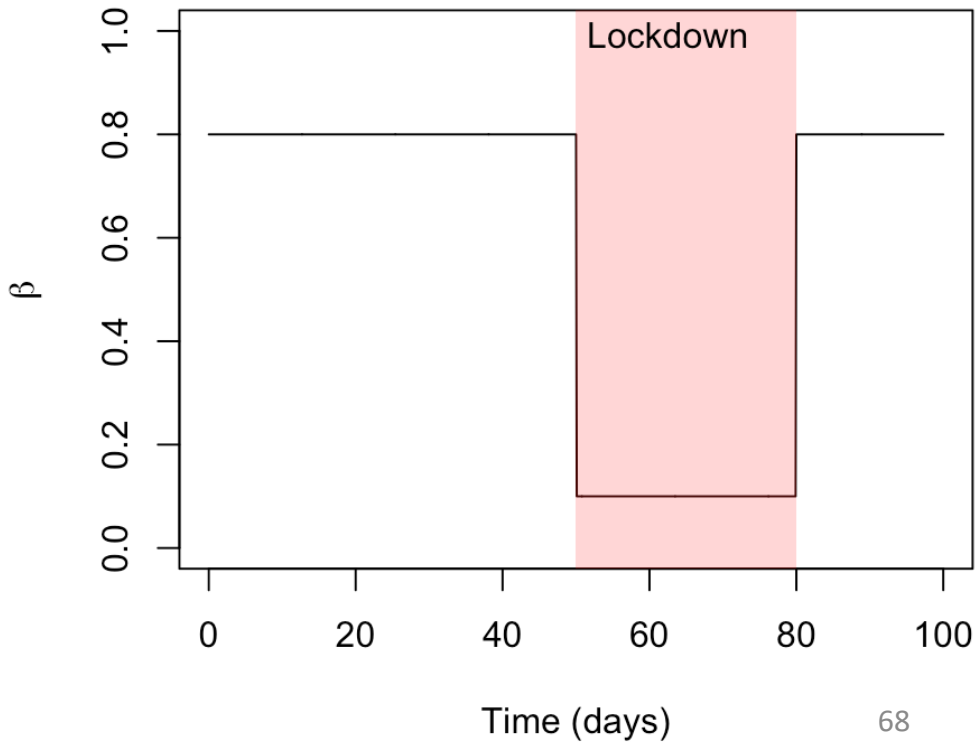
Time dependent parameters

Parameters, like the transmission rate, are “inputs” into the model.

We have been treating these as constants, but they can also vary with time.

What are some reasons the transmission rate might vary over time?

Control measures…



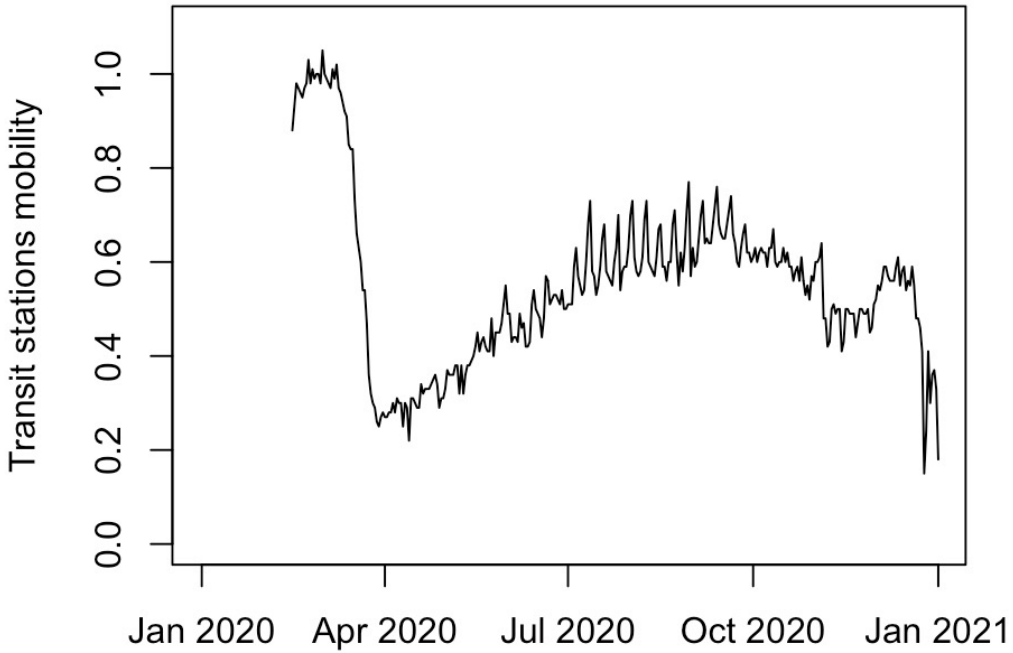
Time dependent parameters

Parameters, like the transmission rate, are “inputs” into the model.

We have been treating these as constants, but they can also vary with time.

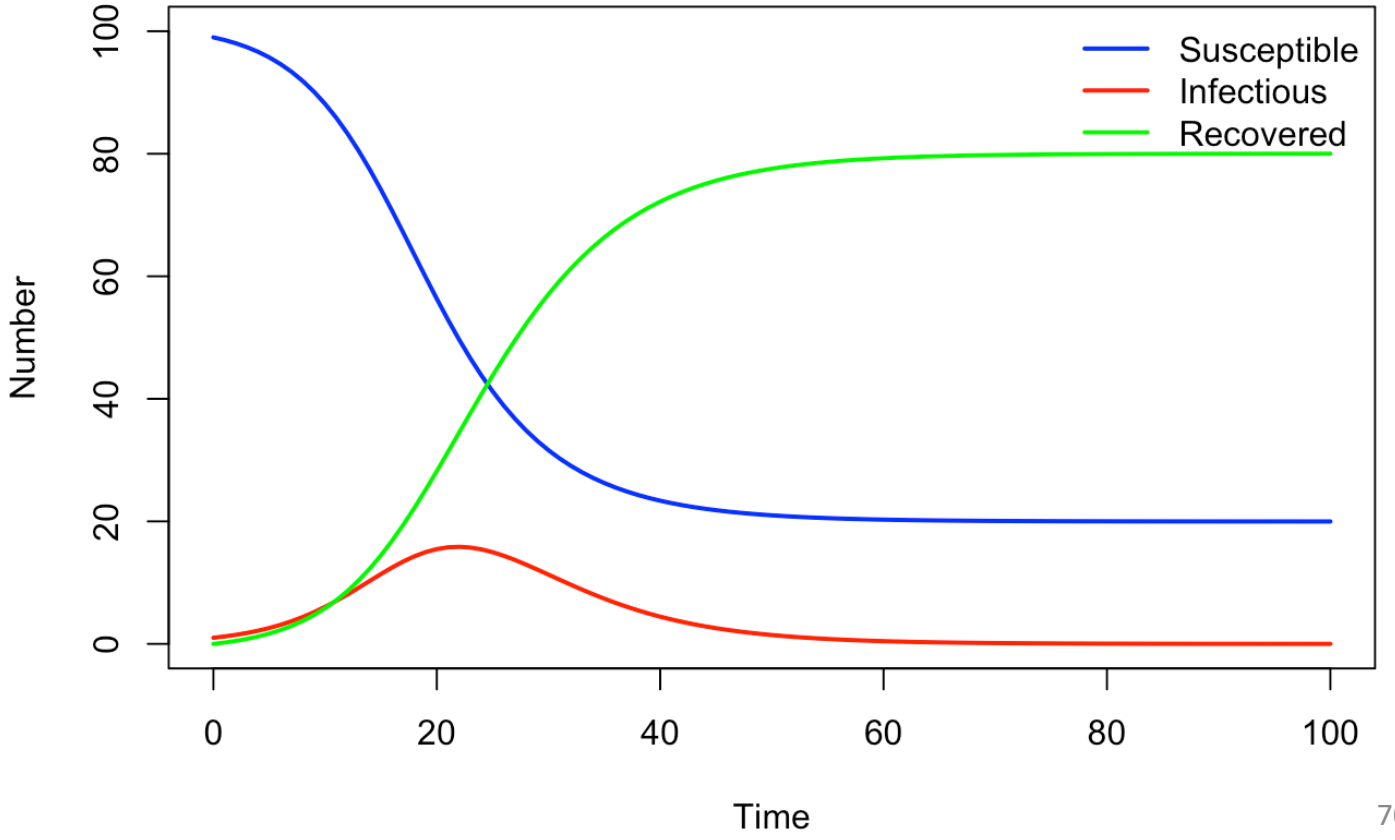
What are some reasons the transmission rate might vary over time?

Behaviour change…



Events

With ODEs, changes that happen to state variables are fundamentally “smooth” – there are no sudden jumps.

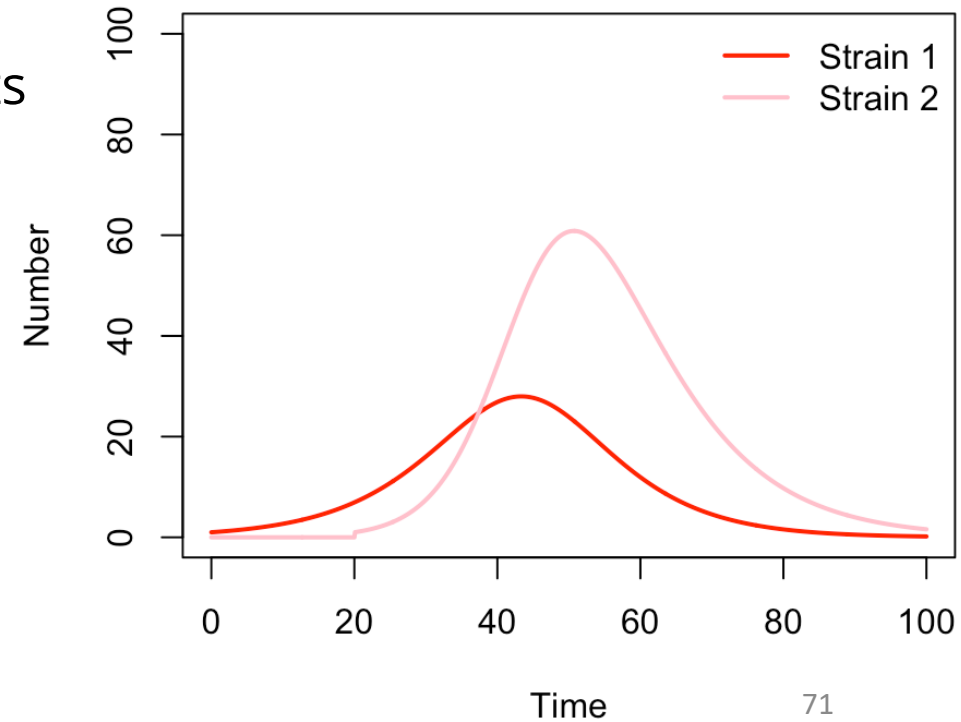


Events

With ODEs, changes that happen to state variables are fundamentally "smooth" – there are no sudden jumps.

What if we need to change the state variables at an instant?

e.g. a new strain of the virus gets introduced on November 20…



Time dependent parameters

Time-dependent parameters can be brought directly into the ODE function.

```
SI_model <- function(t, state, parms) {
  # Get variables
  S <- state["S"]
  I <- state["I"]
  N <- S + I
  # Get parameters
  beta <- parms["beta"]
  # Define differential equations
  dS <- -(beta * S * I) / N
  dI <- (beta * S * I) / N
  res <- list(c(dS, dI))
  return(res)
}
```

Time dependent parameters

Time-dependent parameters can be brought directly into the ODE function.

```
SI_seasonal_model <- function(t, state, parms) {
  # Get variables
  S <- state["S"]
  I <- state["I"]
  N <- S + I
  # Get parameters
  beta_max <- parms["beta_max"]
  period <- parms["period"]
  beta <- beta_max / 2 * (1 + sin(2*pi*t / period))
  # Define differential equations
  dS <- -(beta * S * I) / N
  dI <- (beta * S * I) / N
  res <- list(c(dS, dI))
  return(res)
}
```

Using ‘events’ in deSolve

- deSolve has the capability to include ‘events’
- This can be used when you want to change the value of a state variable based on some condition
- Events can be specified as a data.frame, or in a function.
- Events can also be triggered by a root function.
  - use a data.frame to specify times at which events occur
  - use root function to trigger an event based on some condition

Using ‘events’ in deSolve

- Let’s look at an example of using a root function
- We want to predict infection in a livestock population
  - managed births, i.e. birth rate is a function of some target farm size K
  - assume that death occurs at longer time scale than infection, so we don’t include it

$$\frac{\mathrm{d}S}{\mathrm{d}t} = bN(K-N)/K - \beta SI/N$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \beta SI/N$$

where $N = S + I$.

Using ‘events’ in deSolve

We have our model function,

```
SI_open_model <- function(times, state, parms){
  ## Define variables
  S <- state["S"]
  I <- state["I"]
  N <- S + I
  # Extract parameters
  beta <- parms["beta"]
  K <- parms["K"]
  b <- parms["b"]
  # Define differential equations
  dS <- b * N * (K - N) / K - ( beta * S * I) / N
  dI <- (beta * S * I) / N
  res <- list(c(dS, dI))
  return(res)
}
```

Using ‘events’ in deSolve

– Our event is going to be a herd cull, removing a fraction τ.

– Firstly, we need to write a function which changes the appropriate state variables

```
event_I_cull <- function(times, state, parms) {
  ## Define variables
  I <- state["I"]
  # Extract parameters
  tau <- parms["tau"]

  I <- I * (1 - tau) # cull the infected
population

  state["I"] <- I

  return(state)
}
```

Using ‘events’ in deSolve

– Secondly, we need to write a function which triggers the event

```
root <- function(times, state, parms){
  ## Define variables
  S <- state["S"]
  I <- state["I"]
  N <- S + I
  # Extract parameters
  K <- parms["K"]

  # Our condition is if more than half of the
target herd size becomes infected
  condition <- !(I > K * 0.5) # This is a logical
condition (TRUE/FALSE)
  return(as.numeric(condition)) # Make this
numeric, event occurs if root==0
}
```

Using ‘events’ in deSolve

```
output_raw <- ode(y = state, times = times, func =
SI_open_model, parms = parameters, method = "lsoda",
events = list(func = event_I_cull, root = TRUE),
      rootfun = root)
```

What does the output look like?

Using Rcpp

- Rcpp is an R package that provides an interface between R and C++
- The `func` input in `ode` can be written in C++
- Overcomes some of R's speed issues

Using Rcpp

```
#include <Rcpp.h>
using namespace Rcpp;

// [[Rcpp::export]]
List SIR_cpp_model(NumericVector t, NumericVector state,
                   NumericVector parms)
{
    // Get variables
    double S = state["S"];
    double I = state["I"];
    double R = state["R"];
    double N = S + I + R;

    // Get parameters
    double beta = parms["beta"];
    double gamma = parms["gamma"];

    // Define differential equations
    double dS = -(beta * S * I) / N;
    double dI = (beta * S * I) / N - gamma * I;
    double dR = gamma * I;

    NumericVector res_vec = NumericVector::create(dS, dI, dR );

    List res = List::create(res_vec);

    return(res);
}
```

Practical part 2

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Practical part 2

- Objective: implement SIR with time dependent transmission and use the events function in deSolve
- Answer parts I, II
- Part III is optional