Ordinary Differential Equations

Modern Techniques in Modelling

LONDON SCHOOL *of* HYGIENE & TROPICAL MEDICINE

Outline

- Ordinary differential equations (ODEs)
- How do we ‘solve’ ODE’s
  - integration
  - numerical integration
  - using the deSolve package
- Using R package deSolve
  - SI, SIR, SEIR models in R
  - time dependent parameter changes
  - events based on numbers of infection

Ordinary differential equations

LONDON SCHOOL of HYGIENE & TROPICAL MEDICINE

Reminder: Difference equations

In the previous session, we explored difference equations:

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

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!

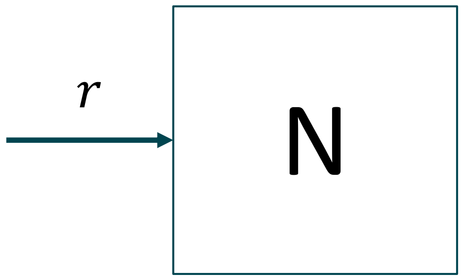
We will look at examples in the next section.

How do we ‘solve’ ODE’s

LONDON SCHOOL *of* HYGIENE &TROPICAL MEDICINE

Population growth

- Some ODE models are simple enough to solve by hand.
- Consider a population of size N, which grows at a per capita rate r with initial population size N_0.
- The per capita growth rate r means that each individual generates "offspring" at rate r.
- We can formulate this model as a flow diagram:



Population growth

- We are interested in the rate of change of N with respect to time t, so we can write the ODE as follows:

$$\frac{\mathrm{d}N}{\mathrm{d}t} = rN,$$

- With initial coniditions (values of the state variables at time $t = 0$)

$$N(0) = N_0$$

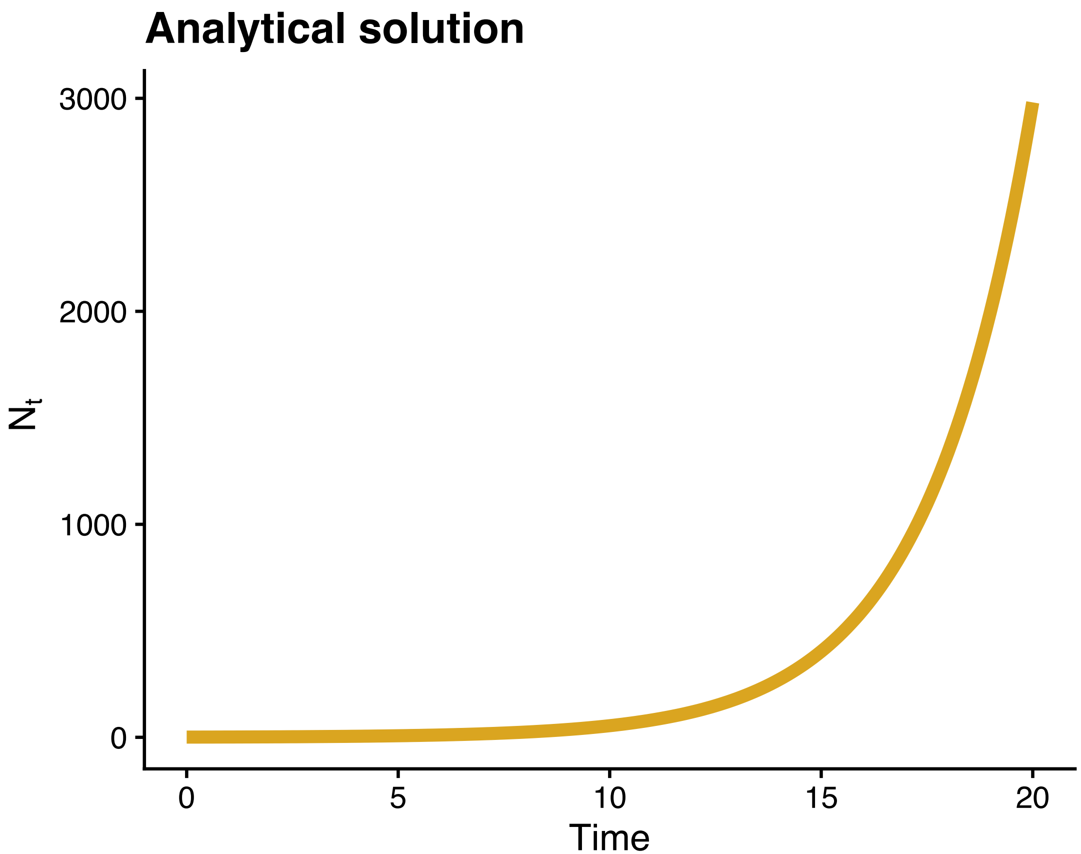
Population growth

– We can rearrange $\frac{\mathrm{d}N}{\mathrm{d}t} = rN$ by hand to get the solution as follows:

$$
\begin{aligned}
\frac{1}{N}\,\mathrm{d}N &= \quad r\,\mathrm{d}t \\
\int \frac{1}{N}\,\mathrm{d}N &= \quad \int r\,\mathrm{d}t \\
\log(N) &= \quad rt + c \\
N(t) &= \quad e^{rt+c} \\
N(t) &= \quad N_0 e^{rt}
\end{aligned}
$$

Population growth

Assume there is one animal at time 0, $N(0) = 1$ with $r = 0.4$, then for $t = 0, \ldots, 20$ the predicted growth is:



Numerical solutions

LONDON
SCHOOL of
HYGIENE
&TROPICAL
MEDICINE

Euler’s method

- When you can’t find the solution by hand, numerical methods are required
- Euler’s method finds numerical solutions using difference equations
- This method approximates continuous time using discrete time steps

Euler's method

– Consider the formal definition of the derivative,

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \lim_{\Delta t \to 0} \frac{y(t + \Delta t) - y(t)}{\Delta t}$$

– If we set Δt to be some small, fixed value we obtain the following approximation

$$\begin{aligned} \frac{\mathrm{d}y}{\mathrm{d}t} &= f(y(t), t) \\ \frac{y(t + \Delta t) - y(t)}{\Delta t} &\approx f(y(t), t) \\ y(t + \Delta t) &= y(t) + \Delta t f(y(t), t) \end{aligned}$$

Euler's method

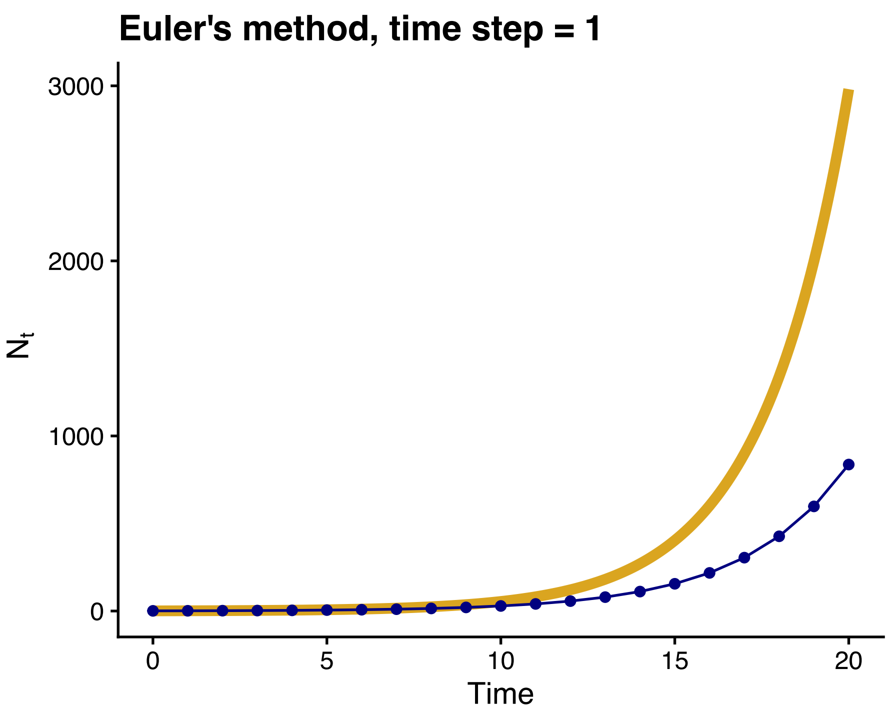
We can rewrite our population growth ODE as follows,

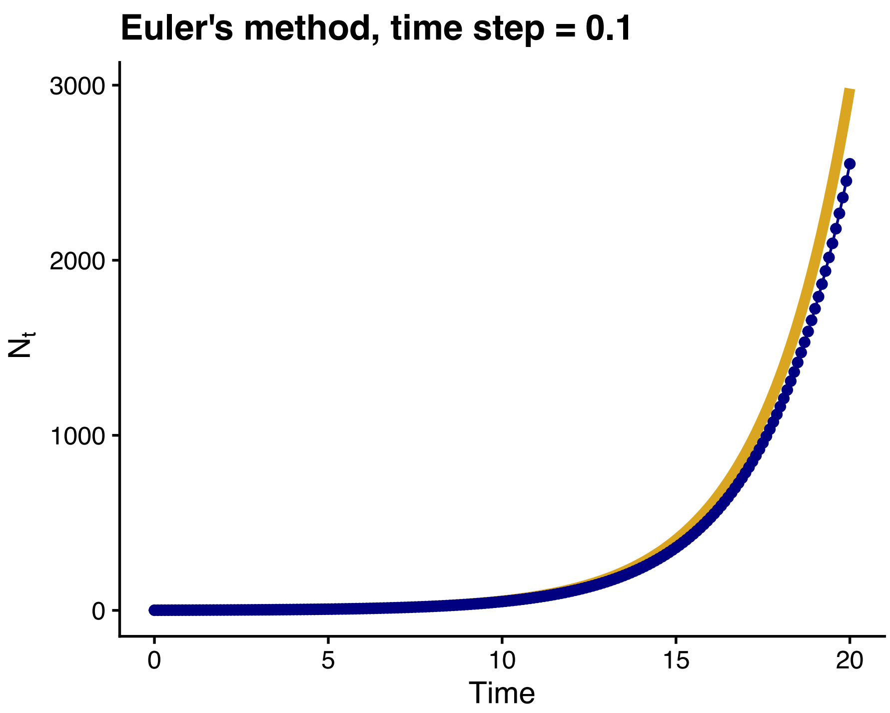
$$
\begin{aligned}
N(t + \Delta t) &= N(t) + \frac{\mathrm{d}N}{\mathrm{d}t}\Delta t \\
&= N(t) + rN(t)\Delta t
\end{aligned}
$$

Euler's method

Using $\Delta t = 1$ leads to a poor approximation.

Using $\Delta t = 0.1$ is better, but does not approximate the growth well towards the end of the 20 days.





What alternatives are there?

- Runge-Kutta method
- Commonly referred to as RK4
- So called because it calculates 4 different increments in its approximation
- We will implement this using the package deSolve

Using R package deSolve

- R package which can numerically solve systems of differential equations using different integrators
- Function `ode()` stands for ordinary differential equation
- Inputs:
  - `y`, the initial conditions
  - `times` the time points to solve the ODE(s)
  - `parms`, a vector parameter values
  - `func`, a function describing the ODE(s)
- Outputs:
  - a matrix consisting of the numerical solution to the equations and the times

Solving the population growth model using deSolve

- Inputs:
  - `y`, the initial conditions

We have one animal at time 0, so $N(0) = 1$, which we will write inside a vector as follows:

```
state <- c(N = 1)
```

Solving the population growth model using deSolve

– Inputs:
  - `times` the time points to solve the ODE(s)

Let's solve the equation over a period of 20 days, which we will write inside a vector as follows:

```
times <- seq(from = 0, to = 20, by = 1)
```

Solving the population growth model using deSolve

- Inputs:
  - `parms`, a vector parameter values

We have just one parameter, the growth rate:

```
parameters <- c(r = 0.4)
```

Solving the population growth model using deSolve

– Inputs:
  - `func`, a function describing the ODE(s)

```
pop_model <- function(times, state, parms){
  ## Define variables
  N <- state["N"]
  # Extract parameters
  r <- parms["r"]
  # Define differential equations
  dN <- r * N
  res <- list(c(dN))
  return(res)
}
```

Solving the population growth model using deSolve

```r
# Solve equations
output_raw <- ode(y = state, times = times,
                  func = pop_model, parms = parameters,
                  method = "euler")
# Convert to data frame for easy    raction of columns
output <- as.data.frame(output_ra

head(output)
##   time       N
## 1    0 1.00000
## 2    1 1.40000
## 3    2 1.96000
## 4    3 2.74400
## 5    4 3.84160
## 6    5 5.37824
```

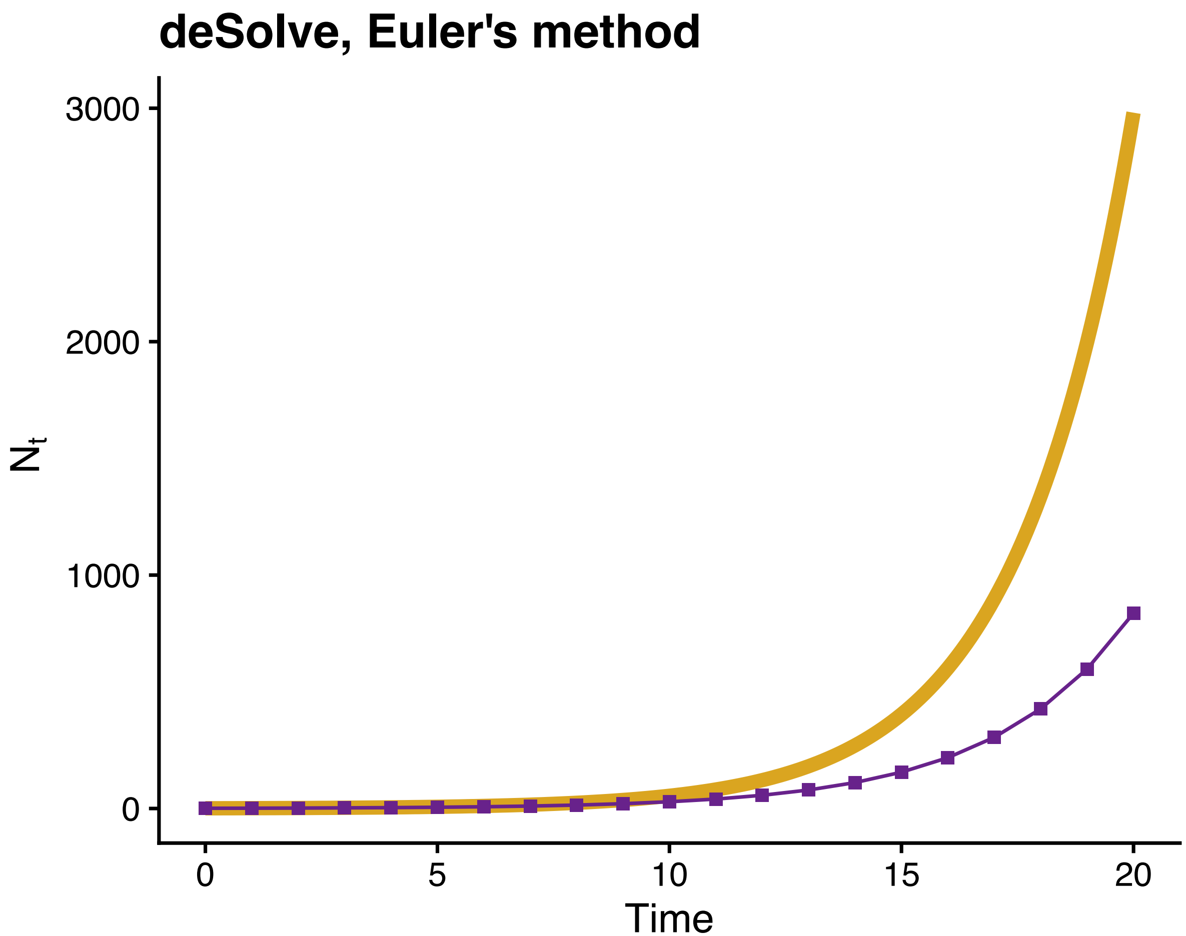
Solving the population growth model using deSolve

The output from `ode` is the same as our difference equations with $\Delta t = 1$.

```
N_0 <- state["N"]
N_t <- N_0 * exp(parameters["r"] * times)

plot(times, N_t, pch = 17, col = "orange")
points(output$time, output$N, type='b', lwd
= 2, pch = 19, col = "navy")
```

Solving the population growth model using deSolve



Solving the population growth model using deSolve

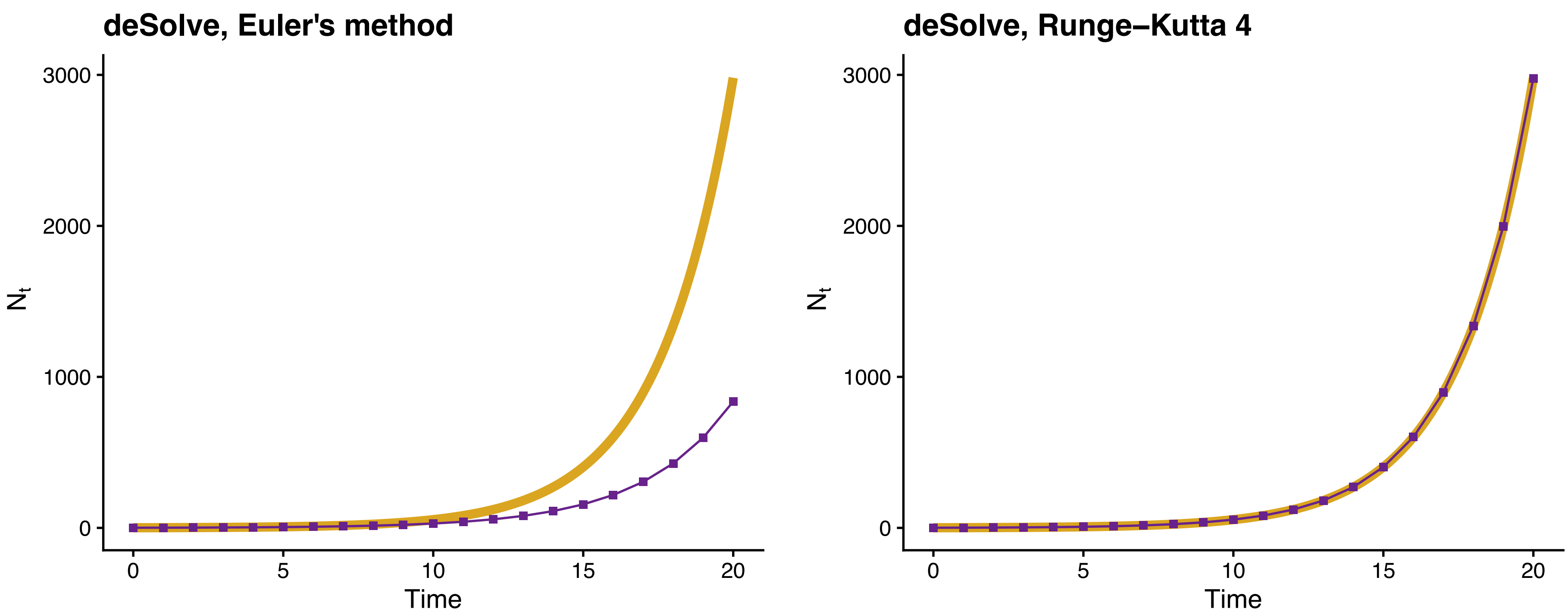
```
output_raw_rk4 <- ode(y = state, times = times,
                      func = pop_model,
                      parms = parameters,
                      method = "rk4")

# Convert to data frame
# for easy extraction of columns
output_rk4 <- as.data.frame(output_raw_rk4)

output_raw_euler <- ode(y = state, times = times,
                        func = pop_model,
                        parms = parameters,
                        method = "euler")

output_euler <- as.data.frame(output_raw_euler)
```

Solving the population growth model using deSolve



In the practical we will use RK4, BUT it will be up to you in your research to make sure that you know which method you are using and you why you are using it.

Susceptible Infected model

Susceptible Infected (SI) model

– Individuals are either susceptible or infected

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

– Susceptible individuals become infected via transmission rate β.

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

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

Solving SI model using deSolve

- Inputs:
  - `y`, the initial conditions

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

```
N <- 100
I_0 <- 1
S_0 <- N - I_0
state <- c(S = S_0, I = I_0)
```

Solving SI model using deSolve

- Inputs:
  - `times` the time points to solve the ODE(s)

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)
```

Solving SI model using deSolve

- Inputs:
  - `parms`, a vector parameter values

We have just one parameter, the transmission rate:

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

Solving SI model using deSolve

– Inputs:
  - `func`, a function describing the ODE(s)

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

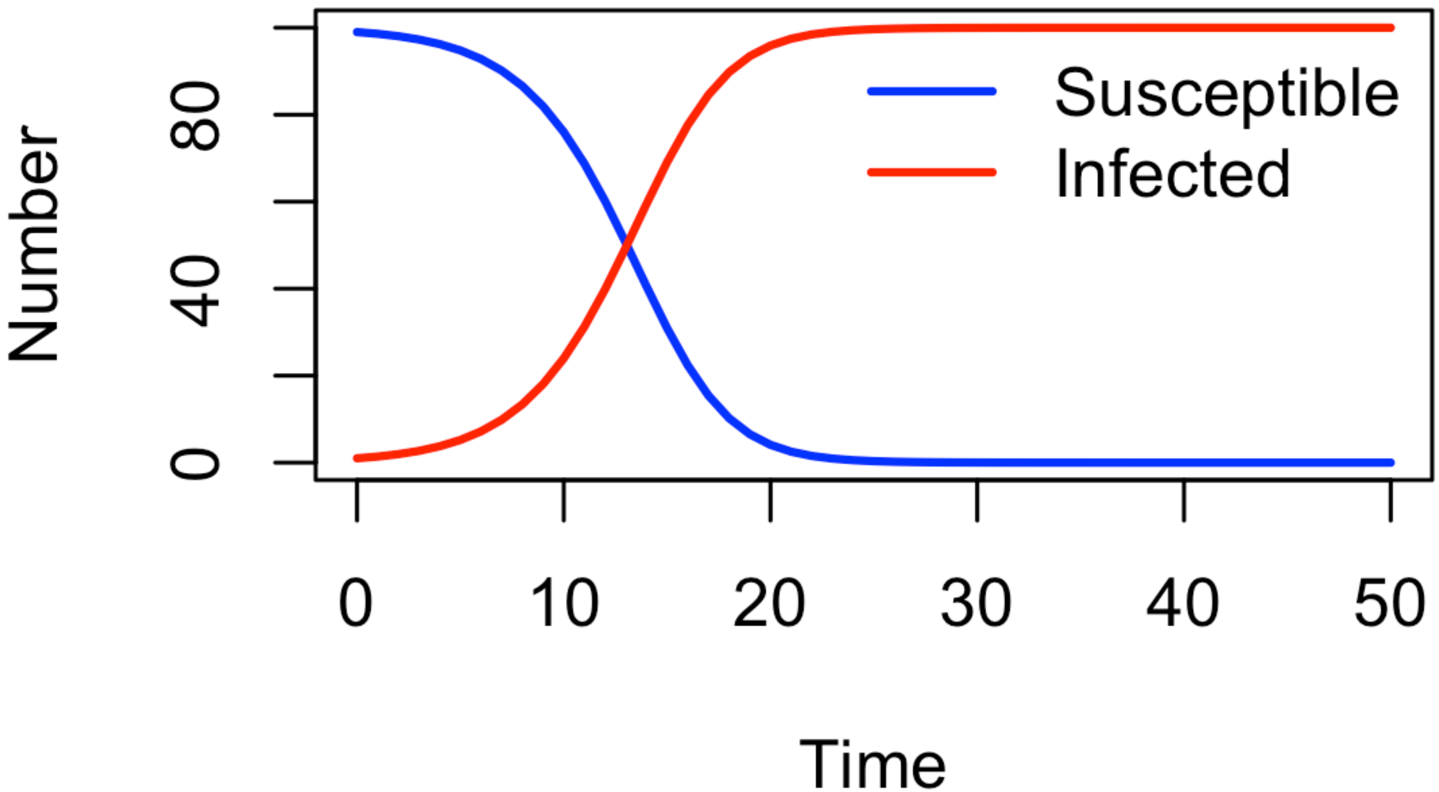
Solving SI model using deSolve

```
# Solve equations
output_raw <- ode(y = state, times = times,
                  func = SI_model, parms = parameters,
                  method = "rk4")
# Convert to data frame for easy extraction of columns
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 part 1

Practical part 1

- Open the file ‘04_ODEs/01_ODE_SIR.R’ and add your code to the existing script
- Objective: Solve SI, SIR, SEIR models using deSolve
- Answer questions 1, 2 and 3
- Question 4 is optional

Practical part 1 - 04_ODEs/01_ODE_SIR.R

Around line 215…

```
# plot results
par(mfrow = c(1, 1))
plot(output$time, output$S, type = "l", col = "blue", lwd = 2, ylim = c(0, N),
     xlab = "Time", ylab = "Number")
lines(output$time, output$I, lwd = 2, col = "red")
lines(output$time, output$R, lwd = 2, col = "green")
lines(output$time, output$E, lwd = 2, col = "cyan")
legend("topright",
       legend =  c("Susceptible", "Exposed", "Infected", "Recovered"),
       lty = 1, col = c("blue", "red", "green", "cyan"), lwd = 2, bty = "n")
```

The four plot/lines commands should be:

```
plot(output$time, output$S, type = "l", col = "blue", lwd = 2, ylim = c(0, N),
     xlab = "Time", ylab = "Number")
lines(output$time, output$E, lwd = 2, col = "red")
lines(output$time, output$I, lwd = 2, col = "green")
lines(output$time, output$R, lwd = 2, col = "cyan")
```

Part 1 summary

- There is a defined list of outputs that you need to tailor to your model
  - Inputs:
    - `y`, the initial conditions
    - `times` the time points to solve the ODE(s)
    - `parms`, a vector parameter values
    - `func`, a function describing the ODE(s)
- Be aware of what 'method' is being used to numerically solve your model

Advanced use of deSolve package

LONDON SCHOOL of HYGIENE &TROPICAL MEDICINE

Advanced use of deSolve package

Speeding up code,

- Using Rcpp

Different types of models,

- Time dependent parameters
- Using ‘events’ in deSolve
  - `method ="lsoda"`

Using Rcpp

- Rcpp is a CRAN package that provides a interface between R and C++
- The `func` input in `ode` can be written in Rcpp
- Why? speed

Using Rcpp

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

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

    // Extract 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);
}
```

Time dependent parameters

Recall the SI model is written as follows:

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

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

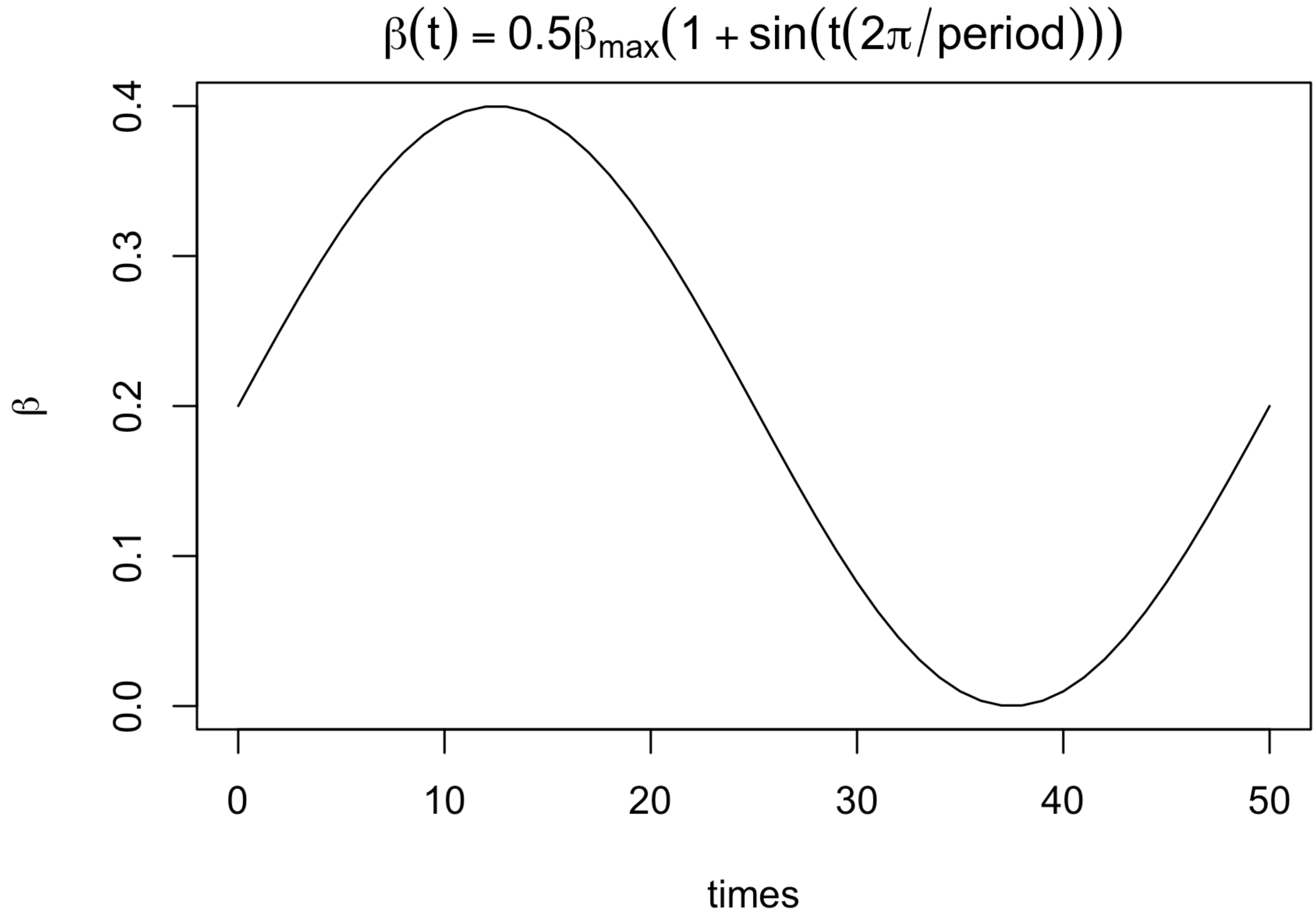
Time dependent parameters

Note that S and I are also functions of time t,

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

What if a model parameter, i.e. the transmission rate was a function of time?

Time dependent parameters



Time dependent parameters

```
SI_seasonal_model <- function(times, state, parms){
  ## Define variables
  S <- state["S"]
  I <- state["I"]
  N <- S + I
  # Extract parameters
  beta_max <- parms["beta_max"]
  period <- parms["period"]
  # Calculate time dependent transmission rate
  beta <- beta_max / 2 * (1 + sin(times * (2 * pi /
period) ) )

  # Define differential equations
  dS <- - (beta * S * I) / N
  dI <- (beta * S * I) / N
  res <- list(c(dS, dI))
  return(res)
}
```

`method = "lsoda"`

- deSolve has a useful method option called `lsoda`
- switches automatically between stiff and non-stiff methods
- what is a stiff problem?
  - broadly, rapid changes in state variables
- why is it a problem?
  - some numerical methods will lead to poor approximation
- We will use this in `events`

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 at rate 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?

Practical part 2

Practical part 2

- Download the file ’04_ODEs/02_ODE_Extras.R’ and add your code to the existing script
- Objective : implement SIR with time dependent transmission and use the events function in deSolve
- Answer questions 1,2
- Question 3 is optional

Summary

- Baseline structure of `deSolve`
- Be aware of what `method' is being used to solve your ODEs
- `deSolve` has many different types of extensions!
- Soetaert, K., Petzoldt, T. & Setzer, R.W. (2010) Solving differential equations in R: Package deSolve. Journal of Statistical Software, 33, 1–25.