> ≡ **Email address**
>
> cameron.smith@biology.ox.ac.uk

Session 2: Mathematical assumptions and analysis

2.1: Our example

Throughout this second workshop, we will work with a particular example, which we will begin by describing biologically. The set up is relatively simple, but should provide enough to demonstrate the key steps, principles and methods required to generate a toy model.
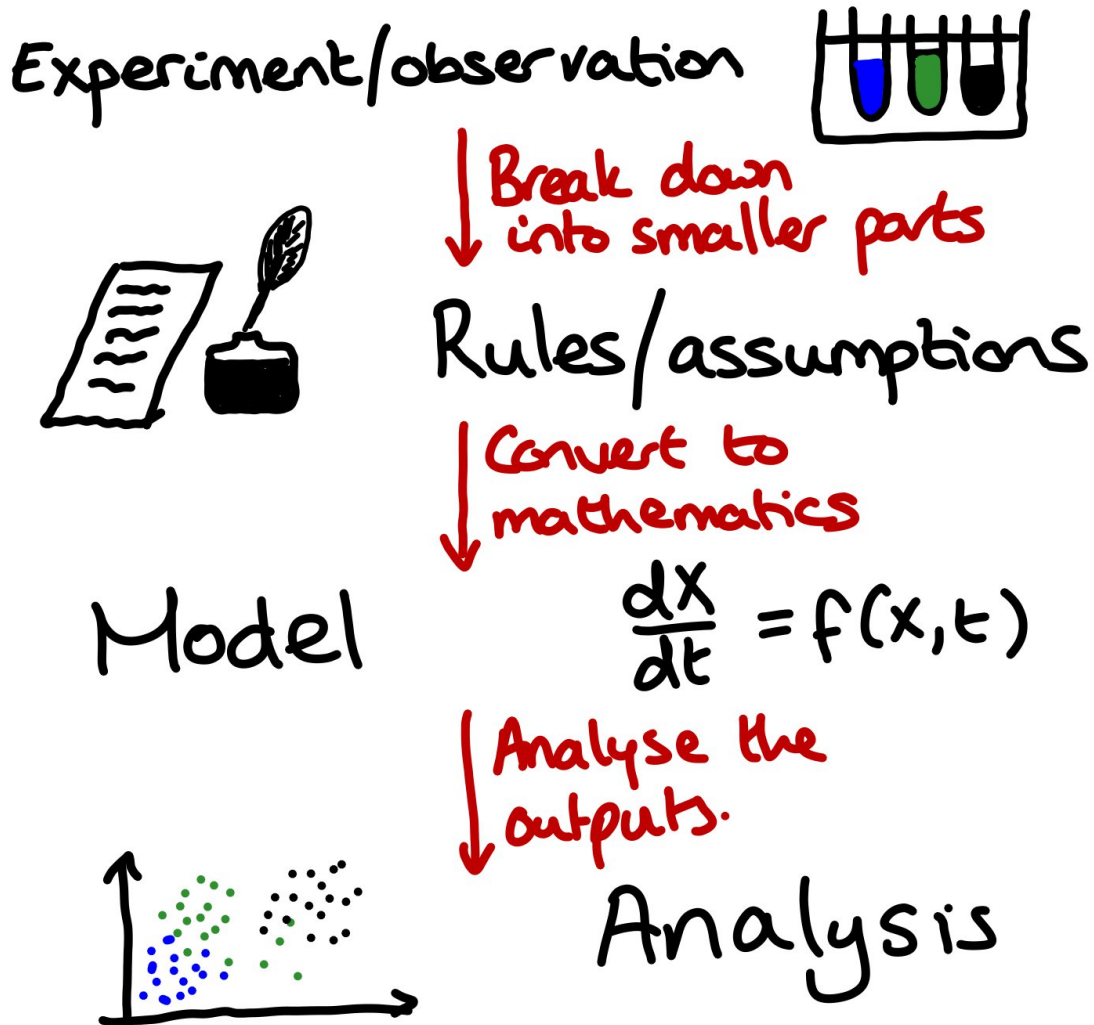
Consider a host population that can be infected by a pathogen, which is only able to be transmitted through contact with infected individuals. The pathogen can be recovered from and can also cause death. These are all of the observations we will work with for now, however, there may be others, such as infection causing a loss in fecundity, different transmission methods to name a few. We will briefly describe some of these in section 2.5.

2.2: Assumptions and mathematical formulation

We will now formulate the above into mathematical model. This requires several steps (see below). We will take a look at each of these in turn using the example described in section 2.1.

THE MODELLING PROCESS

Experiment/observation

Break down into smaller parts

Rules/assumptions

Convert to mathematics

Model

$$\frac{dX}{dt} = f(x, t)$$

Analyse the outputs.

Analysis

2.2.1: From problem to assumption

THe first thing we need to do is to take our observations and knowledge of the biological system, and break it down into smaller chunks, with each of these chunks about one or two aspects of the biology. As with a lot of things, some of these may only seem intuitive after we've talked about them, or maybe won't at all. It will take some practice before getting used to the entire process.

1. Firstly, we note from our description that the pathogen is "only able to be transmitted through contact with infected individuals". This sentence gives us two assumptions. We will talk about the second later, as it is a mathematical assumption, but what it does mean is that there is no vertical transmission. Consequently, we can assume that **all individuals are born susceptible to the disease**.

2. The next assumption isn't explicitly described above, but comes hand-in-hand with birth terms, which we have described in the first assumption. For this we will assume **all**

individuals in the population suffer a background or natural mortality. These deaths will encompass anything that isn't to do with the disease. We include this to ensure a turnover in population, particularly if recovery isn't possible or if we have unbounded growth (see Section 1.4 of the previous workshop), even if it is not observed in experiments.

3. Next, we have observed that the "pathogen can be recovered from". Therefore we will assume that individuals can recover from the pathogen, however, we also need to describe what happens to recovered individuals. There are many choices that can be made here. We can assume that individuals become immune to future infections, they could gain a temporary immunity which wanes over time, or they could become immediately susceptible again. We will assume the latter here, so that **all infected individuals can recover, but become immediately susceptible on recovery**.

4. The final observation which we will convert to an assumption comes from "[the pathogen] can also cause death". This one is very self explanitory, so we assume that **the pathogen can cause excess mortality**.

It is worth noting that assumptions 1 and 2 are not always necessary. For example, if the time period you are interested in modelling over is much shorter than the lifetime of the host, you can ignore those two terms, however if in doubt, it is worth including them.

2.2.2: From assumption to mathematics

We now need to move into including some mathematical assumptions, as well as converting the assumptions above into mathematical expressions. Firstly, looking at what we already have, we need to think about our state variables. These will be the types of hosts that we can have, which here will be susceptible and infected. If, for example we assumed that on recovery you become immune, you would require a third category of host, typically called the recovered class. We will denote the density (think of this as a number of hosts rather than a density if that helps!) of susceptible individuals at a time t as $S(t)$ and the density of infected individuals at time t as $I(t)$ (with total density of $N(t)$). From now on, we will always assume that S and I are functions of time, but will not explicitly write this dependence. We now look at the four assumptions above and formulate them into mathematical expressions:

1. For the birth term, we have assumed that all individuals are born susceptible, which means that we increase the density of susceptible individuals with a rate of $B(N)N$ hosts per unit time, where $B(N)$ is a general birth rate to be defined.

2. For the background mortality, we will define the background mortality rate to be d, or if you prefer, the average lifespan of a host is $1/d$. Therefore, we would lose a density of dS or dI per unit time from the susceptible and infected individuals respectively.

3. For the recovery term, we assume that the average length of time it takes to recover from the pathogen is $1/\gamma$, so that the rate of recovery is γ. Therefore we will lose γI from the infected class and then move them into the susceptible class.

4. For the excess mortality, we will assume that the average length of time that it takes to die from the pathogen is $1/\alpha$, and so occurs at a rate α. Thus, we lose a density of αI from the infected class per unit time.

We now need to add in a few more mathematical assumptions from the description in section 2.1. These are less to do with what to include, but *how* to include them.

5. The term $B(N)$ that we included in assumption 1 requires us to define its form. We will assume that **births are density dependent**, meaning that we will apply a similar rule to section 1.4.2, where we defined

$$B(N) = b(1 - N/K),$$

where b is the maximal growth rate of the population and K is an environmental capacity.

6. The final assumption brings us back to the following statement on transmission: "[the pathogen] is only able to be transmitted through contact with infected individuals". For this, we return back to the law of mass action, from section 1.2.2. Briefly, we assume that the denisty of susceptible individuals infected in a unit of time is proportional to the product of S and I. Thus we lose βSI from the susceptible class and add them to the infected class per unit time.

All of this information can be summarised in a transfer diagram, which we show below.
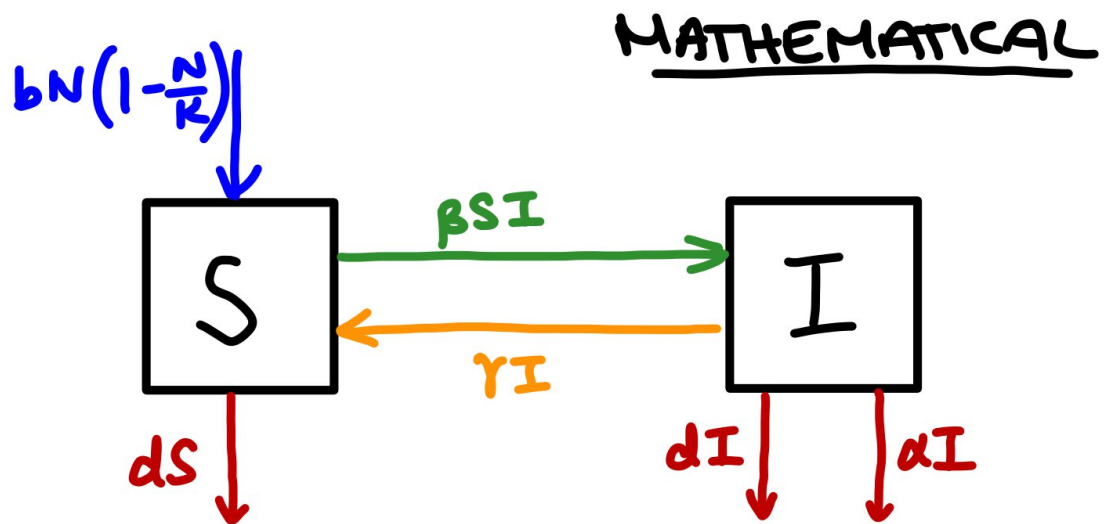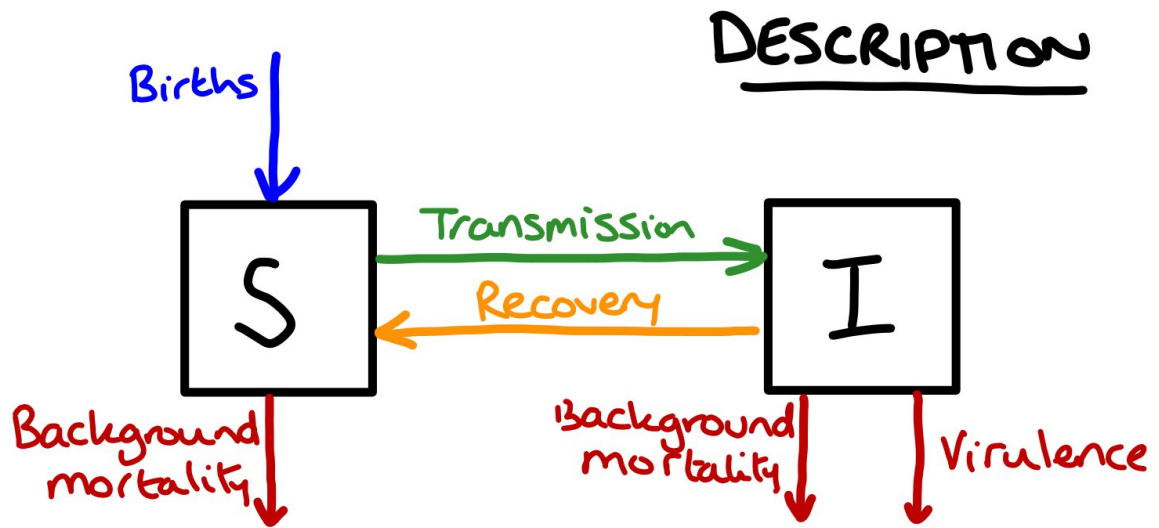
2.2.3: Model formulation

The final step in formulating the model is to convert the mathematical expressions above, together with whether they are losses or gains, into equations. Alternatively we can use the transfer diagram, and we will describe both processes here. Firstly, losses and gains describe where we are going, or whether the population will increase or decrease in size over time. As such, we use the derivatives of the state variables, namely $\mathrm{d}S/\mathrm{d}t$ and $\mathrm{d}I/\mathrm{d}t$.

Let's look at assumption 1. This says that "we increase the density of susceptible individuals with a rate of $B(N)N$ hosts per unit time", which coupled with assumption 5 which describes the form of $B(N)$, yields that:

$$\frac{\mathrm{d}S}{\mathrm{d}t} = bN\left(1 - \frac{N}{K}\right),$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = 0.$$

Since the birth term is an increase in susceptible population, it is included with a plus sign, while the birth term does not directly affect the number of infecitons, yielding a zero on the right-hand side (RHS) of the infection equation. We will next look at assumption 2 which is the background mortality, which sees a reduction in both the susceptible and infectious equations:

$$\frac{\mathrm{d}S}{\mathrm{d}t} = bN\left(1 - \frac{N}{K}\right) - dS,$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = -dI. \tag{2.1}$$

We now have expressions for the birth-death process, which will occur in the absence of the pathogen, which we will now include via assumptions 3, 4 and 6 following the same process as the previous two.

$$\frac{\mathrm{d}S}{\mathrm{d}t} = bN\left(1 - \frac{N}{K}\right) - dS + \gamma I - \beta SI,$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = -dI - \gamma I - \alpha I + \beta SI.$$

We can achieve the same set of ODEs from the transfer diagram. Any arrows which enter a state variable box are included as positive terms in the approprite equation, while any arrows leaving are included with a minus sign.

Finally, I am going to tidy these ODEs up a little so that they appear in a more coherent mathematical order. This is by no means necessary - the equations as they are are sufficient:

$$\frac{\mathrm{d}S}{\mathrm{d}t} = bN\left(1 - \frac{N}{K}\right) - [d + \beta I]S + \gamma I,$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \beta SI - [d + \gamma + \alpha]I. \tag{2.2}$$

We will work from equations (2.1) and (2.2) for the remainder of the workshop.

2.3: Analysis

Now that we have the mathematical, we need to analyse. Unfortunately for us (or fortunately if you don't like maths!), there is not an awful lot more we can do with many models we create. This is because they are likely to be too complicated to even find equilibrium points, let alone solve. However, we can do something in this case, particularly with the disease-free equations (2.1). For these equations, we are able to do some calculations to find the possible equilibrium points that exist. For ODEs that do not depend on time explicitly (see note on explicit time ODEs below), we

can solve $dS/dt = 0 = dI/dt$ simultaneously* for the state variables S and I. In other words, this means finding all pairs (S^*, I^*) (we use * for equilibria) such that:

$$bN^* \left(1 - \frac{N^*}{K}\right) - dS^* = 0,$$

$$-dI^* = 0,$$

where $N^* = S^* + I^*$. Firstly, assuming that $d > 0$ (so that the lifespan of hosts, $1/d$ is finite), we must have $I^* = 0$. This tells us that in the long-term, the number of infectious individuals is zero, which makes sense as there is no disease in the population. Using this information, the first equation becomes:

$$bS^* \left(1 - \frac{S^*}{K}\right) - dS^* = 0.$$

We could have $S^* = 0$, which would yield a population which is extinct in the long term. If $S^* = 0$, then we can divide through by it to yield:

$$b\left(1 - \frac{S^*}{K}\right) - d = 0 \implies S^* = \frac{K(b-d)}{b},$$

which is the equilibrium point where the population reaches it's capacity given the environmental constraints it has.

> ≡ **Time explicit ODEs**
>
> The ODEs (2.2) do not contain time explicitly on the right-hand side (i.e. there is no t which appears on the right-hand side). We call this an *autonomous ODE*, and means that we can (among other things) calculate equilibria as above. However, if time does appear (for example, consider a birth function which is seasonal, so $B(N)$ may look like $B(N, t) = b\sin(\omega t)$), we cannot use the above techniques, and the best approach to use would be something numerical (see section 2.4).

For the ODEs (2.2), finding equilibria is trickier. It is possible to do, and we won't do it here. Instead, we will use a numerical approach to analyse the full model.

2.4: Numerical analysis

Numerical analysis is a large branch of mathematics devoted to approximating the solution of various mathematical constructs. In particular, we are interested in the numerical solutions of ODEs. As it is such a large branch of maths, on its own it could fill several semesters worth of material. But today we are interested in practical advice, which will involve looking at the `deSolve` package in R, together with its function `ode`, in order to solve the ODE (2.2). Before we can start with this though, we should talk a little bit about good coding practices.

2.4.1: Coding principles

We have all been there, done some work several months ago, come to look back it and have no idea what's going on. Okay, maybe it's just me, but with coding, this is surprisingly common. Therefore, we should adopt some good practices whenever we code.

Header comments

At the top of any script, we should write a brief description of what the file does. For the file in the GitHub folder that we will use for this section (named `Host-pathogen.r`), I have added the following description, together with an author name and date created. These last two parts are mostly required if you plan on publishing your code:

```
# Code to run the host-pathogen system from workshop 2.
# This code will use the deSolve package to solve the host-pathogen ODE
# system, and ggplot2 to plot the resultant solution.
#
# Author: Cameron Smith
# Date created: 13/05/24
```

Common-sense naming of variables

When creating variables, it's always good practice to name them something sensible. For example, if I wanted a parameter to represent the background mortality, a choice of `d`, `backMort` or `back_mort` would make a lot of sense and provides some context as to what it is. Don't call it `variable39` or `steve` as these aren't helpful!

Comments

Comments should be enough that they provide context or additional information to the code, but shouldn't state the obvious. As an example, let's take the following line from the R file:

```
N <- S + I
```

We could write a comment that says `# Add together S and I`, but while this is true, it doesn't add anything more to what the code already says. Instead, a comment such as `# Sum the states to get total population` is far more useful; firstly it tells us what `S` and `I` are (state variables) and also tells us what the resulting `N` is (total population). Where you place this comment is up to you, but one place that I prefer is above the line, such as:

```
# Sum the states to get total population
N <- S + I
```

but you could also place it on the same line:

```
N <- S + I  # Sum the states to get total population
```

2.4.2: Differential equations in R

So we have a few guiding principles, and we have our ODES (2.2), so lets now create a script that will allow us to simulate the solution of the ODEs. We begin by (installing and) initialising the required libraries:

```
library(deSolve)
library(ggplot2)
library(reshape2)
```

We require the deSolve package for it's ode function, which will do all of the numerical analysis without us needing to code it explicitly, ggplot2 is required for plotting (it's very basic as it stands!) and reshape2 is needed for the melt function, also for plotting purposes.

The first step to using deSolve is to write the ODEs into a function. This function will have three inputs:

```
hostPathogenFunc <- function(t, state, parameters){
    ...
}
```

The first is time, t. Even if we have an autonomous ODE (no explicit time), the function still requires this as an input. The second and third parameters are named lists. The name would be the label you would give to either the state variable or parameter, and the value would be, well, it's value. For example we may have

```
state <- c(S = 100, I = 1)
parameters <- c(b = 0.1, K = 200, d = 0.01, transmission = 0.05,
                gam = 0.1, alpha = 0.5)
```

Note here that there are some protected names, such as common function names. For example, gamma and beta are mathematical functions which have pre-defined definitions in R, so if you do use them as parameter names, ensure you don't need their function equivalents.

Now we need to fill in the function. In order to access the names of state and parameters, we need to wrap our ODEs inside a secondary loop. This is a with loop which in this case is unwrapping the two named lists:

```
hostPathogenFunc <- function(t, state, parameters){
    with(as.list(c(state, parameters))){
        ...
    }
}
```

Now we are able to write, for example `N = S + I`, and provided `S` and `I` are named in one of either `state` or `parameters`, it will add their two values together. Finally, we need to evaluate the two ODEs and then return the answers in a list:

```
hostPathogenFunc <- function(t, state, parameters){
    with(as.list(c(state, parameters))){

        # Sum the states to get total population
        N <- S + I

        # The susceptible equation
        dSdt <- b*N*(1 - N/K) - (d + transmission*I)*S + gam*I

        # The infceted equation
        dIdt <- transmission*S*I - (d + gam + alpha)*I

        # Output a list. This part is important
        list(c(dSdt, dIdt))
    }
}
```

Note that the output being a list is important. The `ode` function will not work without this. It is also important that the output order of the ODEs is the same as the `state` order. For example, if `state` defines `S` then `I`, the output of this function must be `dSdt` and then `dIdt`.

The next step is to solve the ODE. to do this, we use the `ode` function. There are many arguments that this can have, but we will talk about the bare minimum you require. For more information, you can look up its documentation:

- `y`: This is the initial state variable. In our case, this is a named list such as `state0 <- c(S = 100, I = 1)`. This says that initially, we have a density (or number if you prefer) of 100 of susceptibles, and of 1 for the infected hosts.
- `times`: A list of times to simulate the ODEs at. In general, you will want to use the `seq` function in order to achieve this. For example `times <- seq(0, 100, by = 0.01)` would evaluate the ODE solution at times 0, 0.01, 0.02, ..., 99.9, 100.
- `func`: This is the RHS function. For us this is `hostPathogenFunc`.
- `parms`: This is where we specify the parameters for our ODEs, which we could define as `parameters <- c(b = 0.1, K = 200, d = 0.01, transmission = 0.05, gam = 0.1, alpha = 0.5)`.

Putting all of this together, we would obtain the output as:

```
out <- ode(y = state0, times = times,
           func = hostPathogenFunc,
           parms = parameters)
```

By default, this is of type "double". In order to use ggplot, we convert to a data frame by replacing the line above with:

```
out <- data.frame(ode(y = state0, times = times,
                      func = hostPathogenFunc,
                      parms = parameters))
```

The outut looks like the following:

```
head(out)

  time         S        I
1 0.00 100.00000 1.000000
2 0.01  99.98991 1.044876
3 0.02  99.97757 1.091760
4 0.03  99.96290 1.140738
5 0.04  99.94578 1.191905
6 0.05  99.92612 1.245356
```

Notice that there are three columns. In general, you will have $n + 1$ columns, where n is the number of state variables you have. In this case we have two (S and I), plus a column for the times that we asked for the solution at. We plot this using ggplot as follows:

```
# Plotting. Using ggplot to plot S and I on the same window.
# This first line generates a new dataframe from out, where the time
# remains (id.vars) and then the others are converted to two columns,
# with the first being the column name from out, and the second being its
# value.
df <- melt(out,  id.vars = 'time', variable.name = 'state')
head(df)

  time series     value
1 0.00      S 100.00000
2 0.01      S  99.98991
3 0.02      S  99.97757
4 0.03      S  99.96290
5 0.04      S  99.94578
6 0.05      S  99.92612

plotPP <- ggplot(df, aes(time, value)) + geom_line(aes(colour = state))
plot(plotPP)
```

The resulting figure is the following:
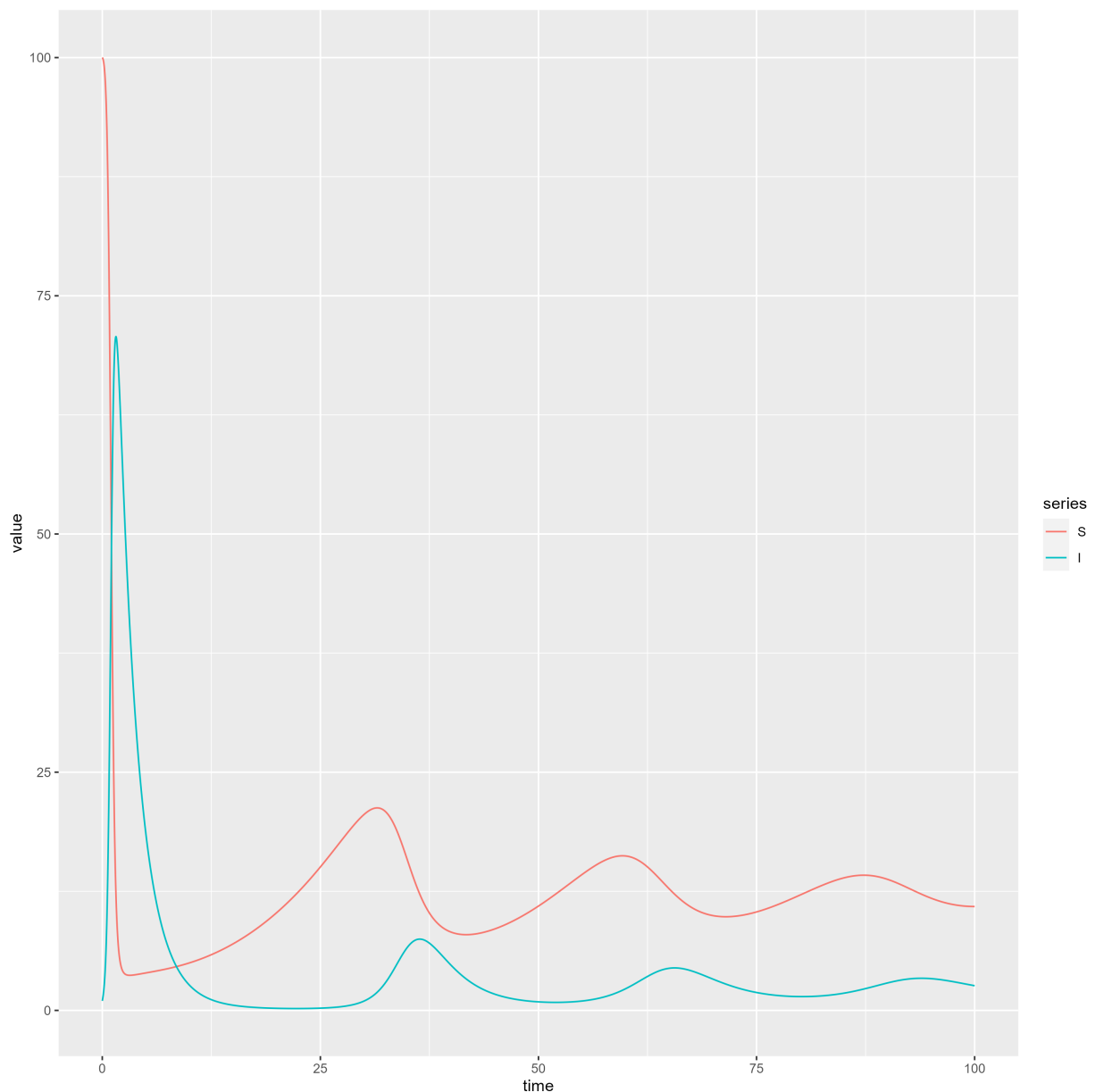
So what are we looking. Let's suppose that the time is measured in days. Then We see an initial spike in infections (the blue line), which peaks after a day and a half, which then subsequently relaxes until there are very few cases after around 15 days. In this period, the number of susceptible individuals increases due to the birth function, which then causes a secondary peak in infections once a sufficient number of individuals are susceptible. This pattern continues, and will eventually settle down to an equilibrium point that contains a density of ~3 infected individuals and ~12 of susceptible individuals. If we do calculate the steady states analytically (not shown here), we can find that $S^* = 12.2$ and $I^* \approx 2.42$. The attached contains a function which outputs all of the possible steady states, called `steadyStates`.

2.4.3: Common issues

We now look at some common troubleshooting issues that arise from numerical analysis. This is by no means an exhaustive list, and also doesn't include methods such as double checking you have typed everything correctly. If you are getting errors, try stack exchange if none of the below work.

Biological realism

The first thing to check is if your parameters and initial conditions are biologically realistic. By this, I mean that if you have a parameter that is a rate, then it is necessarily positive. If you are modelling a population size then that must also be positive, but if you are modelling a proportion of a population, then this must be between 0 and 1.

Choosing the time vector

The time vector that is used in the ode function needs to be chosen with a bit of care, and may require a little trial and error. If you choose the time step to be too big, the estimate of the solution outputted will have a large error, and may even become infinite when it shouldn't (this is called numerical instability). If you choose it too small, the code will take a long time to run.

2.5: Advanced techniques

Below is a list of more advanced topics. They will either include a small introduction to the topic, or will just contain references to take a look at.

2.5.1: Dimensional analysis

While a slightly more advanced technique, this can aid in troubleshooting any models that you have created, and also gives an idea as to the dimension or units of your parameters. From now on, we will use square brackets to denote the dimension or unit of a quantity. For example, if we write $[S]$ we mean the dimension of the density of susceptible individuals. Before we can do anything, we need to talk about the laws of dimensional analysis.

1. If two things are equal, they must have the same dimension.

2. If two terms are summed (or subtracted), they must have the same dimension. For example, if I have two quantities X and Y and I add them together, then it is necessary to have $[X] = [Y]$. Think of it this way. I can't add 4 oranges and 3 bananas together!

3. To multiply (or divide) terms, their dimensions are also multiplied (or divided). Again, if we have $Z = XY$, then $[Z] = [X][Y]$.

4. The dimension for a derivative works a little like division. If we want to find the dimension of dX/dt, then we would obtain $[dX/dt] = [X]/[t] = [X][t]^{-1}$.

5. Finally, if we find that a parameter is dimensionless (or unit-less), meaning that it has no unit (examples of these are numbers, probabilities or proportions), then we typically denote this as $[1]$.

With this in mind, we can now obtain the dimensions of all of our parameters in our model (2.2). I will just run through the first equation, but the second works in the same way. Firstly, let's "take the dimension" of both sides, and apply the rules above. We will also set $[S] = [N]$ and $[t]$ to be fixed.

In theory, we should know what dimensions these are as they are the thing we measure and the time-frame over which we measure them:

$$[\frac{\mathrm{d}S}{\mathrm{d}t}] = [bN\,(1 - \frac{N}{K}) - (d + \beta I)S + \gamma I]$$

$$\implies [N][t]^{-1} = [bN\,(1 - \frac{N}{K}) - (d + \beta I)S + \gamma I] \quad \text{by rule 4 on LHS}$$

Now we employ rules 2 and then 1. Firstly, each term (things being added and subtracted) on the RHS must have the same dimension by rule 2, and by rule 1, this dimension must be $[N][t]^{-1}$:

$$[N][t]^{-1} = [bN\,(1 - \frac{N}{K})]$$

$$= [b][N]\,[1 - \frac{N}{K}] \quad \text{by rule 3 on RHS}$$

$$= [b][N][1] \qquad \text{by rules 2 and 3 on RHS with } [K] = [N]$$

To find $[b]$ we can do some algebra to find that $[b] = [t]^{-1}$, meaning that b is a rate (has units one over time). In the process of this, we also found that the units of K are the same as the units for the state variables (i.e. a density).

$$[N][t]^{-1} = [(d + \beta I)S]$$

$$= [d + \beta I][N] \quad \text{by rule 3 on RHS}$$

Therefore, by rule 2, after rearranging, we get $[d] = [t]^{-1}$ (so is a rate), and $[t]^{-1} = [\beta][N] \implies [\beta] = [N]^{-1}[t]^{-1}$. This says that the transmission is a rate per (infected) host. The final term follows in the same way, with $[\gamma] = [t]^{-1}$.

2.5.2: Adaptive dynamics

Adaptive dynamics is a technique to conduct an evolutionary analysis on a particular trait. As an example, we may want to investigate how the pathogen evolves to become more or less virulent over time. This is an involved technique, so here I will list some references which introduce the topic:

> ### ≡ Adaptive dynamics references
>
> If you want to find out more about adaptive dynamics and how evolution can be used to answer biological questions,then here are some references that should be accessible after completing the problem sheet.
>
> **A practical guide to adaptive dynamics:**
>
> Brännström Å, Johansson J, Von Festenberg N. The Hitchhiker's Guide to Adaptive Dynamics. Games. 2013; 4(3):304-328. https://doi.org/10.3390/g4030304.
>
> **Earlier paper describing evolutionary singular strategies:**
>
> Geritz, S., Kisdi, E., Mesze´NA, G. et al. Evolutionarily singular strategies and the adaptive growth and branching of theevolutionary tree. Evolutionary Ecology 12, 35–57 (1998). https://doi.org/10.1023/A:1006554906681 .

2.5.3: Partial differential equations

Partial differential equations (PDEs) occur when you have multiple independent variables. Consider for example a population which changes in time but is also dependent on location. In this case, our state variable mighy be $N(x,t)$. PDEs are far harder to use at every stage, from formulation to analysis to coding. Often times, it is easier to assume we only have one independent variable as the mathematics become quite involved for PDEs.

2.5.4: