Chapter 2
SIR

2.1 Introduction

The following 10 chapters are devoted to the study of patterns of infection over time and age. The current chapter introduces the basics of compartmental modeling of transmission dynamics. This is followed by a chapter with in-depth discussion of the reproduction number, R_0, which is the most important quantity for understanding epidemics of infectious agents. The subsequent chapters detail the importance of age structure and seasonality in shaping epidemics and pandemics as well as several important time series methods for characterizing and understanding temporal recurrence patterns of infection. The last two chapters explore how ideas from dynamical systems theory can help explain several very curious aspects of the waxing and waning of infection through time.

2.2 The SIR Model

In 1927, Kermack and McKendrick (1927) published a set of general equations (Breda et al., 2012) to better understand the dynamics of an infectious disease spreading through a susceptible population. Their motivation was

> One of the most striking features in the study of epidemics is the difficulty of finding a causal factor which appears to be adequate to account for the magnitude of the frequent epidemics of disease which visit almost every population [...] The problem may be summarized as follows: One (or more) infected person is introduced into a community of individuals, more

This chapter uses the following R packages: `deSolve`, `phaseR`, and `shiny`.
A conceptual understanding of *reproduction numbers* and the *simple epidemic* is useful. Five minute epidemics MOOC introductions are:
Reproduction number https://www.youtube.com/watch?v=ju26rvzfFg4.
Closed epidemic https://www.youtube.com/watch?v=sSLfrSSmJZM.
The `sir.app` shinyApp provides an interactive interface as part of the `epimdr2` package.

O. N. Bjørnstad, *Epidemics*, Use R!, https://doi.org/10.1007/978-3-031-12056-5_2

or less susceptible to the disease in question. The disease spreads from the affected to the unaffected by contact infection. Each infected person runs through the course of his sickness, and finally is removed from the number of those who are sick, by recovery or by death. The chances of recovery or death vary from day to day during the course of his illness. The chances that the affected may convey infection to the unaffected are likewise dependent upon the stage of the sickness. As the epidemic spreads, the number of unaffected members of the community becomes reduced [...] In the course of time the epidemic may come to an end. One of the most important problems in epidemiology is to ascertain whether this termination occurs only when no susceptible individuals are left, or whether the interplay of the various factors of infectivity, recovery and mortality, may result in termination, whilst many susceptible individuals are still present in the unaffected population.

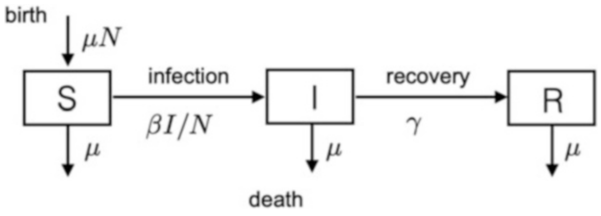

Fig. 2.1: The SIR flow diagram of transitions among Susceptibles (*S*), Infected and Infectious (*I*), and Recovered/Removed (*R*) compartments. Rates are per capita rates among compartments

Following a general mathematical exposé, they suggested a set of pragmatic assumptions that lead to the standard SIR model of ordinary differential equations (ODEs) for the flow of hosts between **S**usceptible, **I**nfectious, and **R**ecovered compartments. In modern notation, the simplest set of equations is (Fig. 2.1)

$$\frac{dS}{dt} = \underbrace{\mu N}_{\text{birth}} - \underbrace{\beta I \frac{S}{N}}_{\text{infection}} - \underbrace{\mu S}_{\text{death}} \tag{2.1}$$

$$\frac{dI}{dt} = \underbrace{\beta I \frac{S}{N}}_{\text{infection}} - \underbrace{\gamma I}_{\text{recovery}} - \underbrace{\mu I}_{\text{death}} \tag{2.2}$$

$$\frac{dR}{dt} = \underbrace{\gamma I}_{\text{recovery}} - \underbrace{\mu R}_{\text{death}} \tag{2.3}$$

The assumptions of Eqs. (2.1)–(2.3) are:

- The infection circulates in a population of size N, with a per capita baseline death rate, μ, which is balanced by a birth rate μN. From the sum of Eqs. (2.1)–(2.3), $dN/dt = 0$ and $N = S + I + R$ is thus constant. N is assumed to be large, so epidemics will unfold according to the predictable clockwork of the coupled deterministic differential equations. We will consider how to accommodate deviations from this assumption throughout the ensuing text, notably in Sects. 3.4, 8.2, and 9.2.

- The infection causes acute morbidity (not mortality); that is, in this version of the SIR model, we assume we can ignore disease-induced mortality. This is reasonable for certain infections like chickenpox, but certainly not for others like rabies, SARS, or Ebola (Sects. 3.7, 3.9, and 10.6 introduce models that relax on this assumption).
- Individuals are recruited directly into the susceptible class at birth (so ignore perinatal maternal immunity).
- Transmission of infection from infectious to susceptible individuals is controlled by a bilinear contact term $\beta I \frac{S}{N}$. This stems from the assumption that the I infectious individuals are independently and randomly mixing with all other individuals, so a fraction S/N of the encounters is with susceptible individuals; β is the contact rate times the probability of transmission given a contact between a susceptible and an infectious individual.
- Chances of recovery or death are assumed not to change during the course of infection.
- Infectiousness is assumed not to change during the course of infection.
- Infected individuals are assumed to move directly into the infectious class (as opposed to the SEIR model introduced in Sect. 3.7) and remain there for an average infectious period of $1/\gamma$ (assuming $\mu << \gamma$).[1]
- The model finally assumes that recovered individuals are immune from reinfection for life.[2]

The basic reproduction number (R_0), interchangeably also termed the basic reproductive ratio, is defined as the expected number of secondary infections from a single index case in a completely susceptible population. This is a pivotal quantity in the theory of infectious disease dynamics. Chapter 3 is entirely devoted to this quantity. For this particular model (Eqs. (2.1)–(2.3)), $R_0 = \frac{\beta}{\gamma+\mu}$, and thus $\beta = R_0(\gamma + \mu)$. The later relationship is useful because while β is one of the key rate parameters in the model, it is often more intuitive to think in terms of R_0 as it can be estimated from a variety of data using a variety of methods (Chap. 3).

[1] The implicit assumptions that stem from the use of deterministic, ordinary differential equation (ODE) are that the infectious periods (and resident times in all compartments) are exponentially distributed. This is a tractable approximation for exploring overall dynamics, but observed duration of infection periods is often much less variable—the *Eimeria*-gut parasite (a relative of the *Plasmodium* parasites that cause malaria) undergoes a fixed number of replication cycles before all parasites leave the host (Smith et al., 2002b) or much more variable. Section 2.9 discusses a practical approach to modeling disease dynamics when the exponential assumption is deemed too simplistic.

[2] Sections 10.5 and 11.4 visits on dynamics under transient immunity via the SIRS and SIRWS models.

2.3 Numerical Integration of the SIR Model

If there are no (or negligible) births and deaths during the duration of an epidemic ($\mu \simeq 0$), the dynamics are commonly referred to as a closed epidemic. While it is occasionally possible to derive analytical solutions to systems of ODEs like Eqs. (2.1)–(2.3), we generally have to resort to numerical integration to predict the numbers over time. The `deSolve` R package provides functions to numerically integrate such equations. Throughout this text numerical integration of a variety of different ODE models will be required. While the models differ, the basic recipe is generally the same: (1) define an R function for the general system of equations, (2) specify the time points at which we want the integrator to save the state of the system, (3) provide values for the parameters, (4) give initial values for all state variables, and finally (5) invoke the `ode` function.

```
require(deSolve)
```

STEP 1: Define the function (often called the gradient function) for the equation systems. The `deSolve` package requires the function to take the following parameters: time, `t`,[3] a vector with the values for the state variables (in this case S, I, and R), `y`, and parameter values (for β, μ, γ, and N), `parameters`:

```
sirmod = function(t, y, parameters) {
    # Pull state variables from y vector
    S = y[1]
    I = y[2]
    R = y[3]
    # Pull parameter values from the input vector
    beta = parameters["beta"]
    mu = parameters["mu"]
    gamma = parameters["gamma"]
    N = parameters["N"]
    # Define equations
    dS = mu * (N - S) - beta * S * I/N
    dI = beta * S * I/N - (mu + gamma) * I
    dR = gamma * I - mu * R
    res = c(dS, dI, dR)
    # Return list of gradients
    list(res)
}
```

STEPS 2–4: Specify the time points at which we want `ode` to record the states of the system (here we use a half year with weekly time increments as specified in the vector `times`), the parameter values (in this case as specified in the vector `paras`), and starting conditions (specified in `start`). If we model the fraction of individuals in each class, we set $N = 1$ (though we could do percentages with $N = 100$ or some other population size of relevance). Let us consider a disease with

[3] Though, in the case of the simple SIR model, there is no time dependence in any of the parameters, so this parameter is not called within the gradient function; this will change when we consider seasonality (Chap. 6).

an infectious period of 2 weeks ($\gamma = 365/14$ per year) for the closed epidemic (no births or deaths so $\mu = 0$). A reproduction number of 4 implies a transmission rate β of 2. For starting conditions, assume that 0.1% of the initial population is infected and the remaining fraction is susceptible.

```
times = seq(0, 0.5, by = 1/365)
paras = c(mu = 0, N = 1, R0 = 4, gamma = 365/14)
paras["beta"] = paras["R0"] * (paras["gamma"] + paras["mu"])
start = c(S = 0.999, I = 0.001, R = 0) * paras["N"]
```

STEP 5: Feed `start` values, `times`, the gradient function `sirmod`, and parameter vector `paras` to the `ode` function as suggested by `args(ode)`.[4] For convenience, we convert the output to a data frame (`ode` returns a `list`). The `head` function shows the first 5 rows of `out` and `round(,3)` rounds the number to three decimals.

```
out = ode(y = start, times = times, func = sirmod, parms = paras)
out = as.data.frame(out)
head(round(out, 3))

   ##    time     S     I     R
   ## 1 0.000 0.999 0.001 0.000
   ## 2 0.003 0.999 0.001 0.000
   ## 3 0.005 0.998 0.002 0.000
   ## 4 0.008 0.998 0.002 0.000
   ## 5 0.011 0.997 0.002 0.000
   ## 6 0.014 0.996 0.003 0.001
```

Figure 2.2 shows how the model predicts an initial exponential growth of the epidemic that decelerates as susceptibles are depleted and finally fade out as susceptible numbers are too low to sustain a chain of transmission.

```
plot(x = out$time, y = out$S, ylab = "Fraction", xlab = "Time",
    type = "l")
lines(x = out$time, y = out$I, col = "red")
lines(x = out$time, y = out$R, col = "green")
```

R allows for a lot of customization of graphics—Rseek.org is a useful resource to find solutions to all things R... Fig. 2.2 has some added features such as a right-hand axis for the effective reproduction number (R_E)—the expected number of new cases per infected individuals in a *not* completely susceptible population—and a legend so as to confirm that the turnover of the epidemic happens exactly when $R_E = R_0 s = 1$, where s is the fraction of remaining susceptibles. The threshold $R_0 s = 1 \Rightarrow s = 1/R_0$ results in the powerful rule of thumb for vaccine-induced elimination and herd immunity: if, through vaccination, the susceptible population is kept below a critical fraction, $p_c = 1 - 1/R_0$, then pathogen spread will dissipate and the pathogen will not be able to reinvade the host population (e.g., Anderson & May, 1982; Roberts & Heesterbeek, 1993; Ferguson et al., 2003). This rule of thumb appeared to work well for smallpox, the only vaccine-eradicated human disease; its R_0

[4] For further details on usage, do `?function` on the R command line, i.e., `?ode` in this instance.

was commonly around 5, and most countries saw elimination once vaccine cover exceeded 80% (Anderson & May, 1982). The actual code for Fig. 2.2 is:

```
R0 = paras["R0"]
# Adjust margins to accommodate a second right axis
par(mar = c(5, 5, 2, 5))
# Plot state variables
plot(x = out$time, y = out$S, ylab = "Fraction", xlab = "Time",
    type = "l")
lines(x = out$time, y = out$I, col = "red")
lines(x = out$time, y = out$R, col = "green")

# Add vertical line at turnover point
xx = out$time[which.max(out$I)]
lines(c(xx, xx), c(1/R0, max(out$I)), lty = 3)

# prepare to superimpose 2nd plot
par(new = TRUE)
# plot effective reproduction number  (w/o axes)
plot(x = out$time, y = R0 * out$S, type = "l", lty = 2,
    lwd = 2, col = "black", axes = FALSE, xlab = NA, ylab = NA,
    ylim = c(-0.5, 4.5))
lines(c(xx, 26), c(1, 1), lty = 3)
# Add right-hand axis for RE
axis(side = 4)
mtext(side = 4, line = 4, expression(R[E]))
# Add legend
legend("right", legend = c("S", "I", "R", expression(R[E])),
    lty = c(1, 1, 1, 2), col = c("black", "red", "green",
        "black"))
```

2.4 Final Epidemic Size

The closed epidemic model has two equilibria: the disease free equilibrium, $\{S = 1, I = 0, R = 0\}$, which is unstable when $R_0 > 1$ and the $\{S^*, I^*, R^*\}$ equilibrium which reflects the final epidemic size, R^*, for which $I^* = 0$ as the epidemic eventually self-extinguish in the absence of susceptible recruitment; S^* is the fraction of susceptibles that escape infection altogether. For the closed epidemic, there is an exact mathematical solution to the final epidemic size (below). It is nevertheless useful to consider computational ways of finding steady states in the absence of exact solutions.

The easiest approach is to use the `ode` function to integrate the system until it settles on a steady state (if it exists).[5]

[5] By varying initial conditions, we should be able to find multiple stable equilibria if there are more than one of them. This approach will not find unstable equilibria, for these we need to use other strategies. Section 10.3 considers in more depth how to find all equilibria whether stable or not.

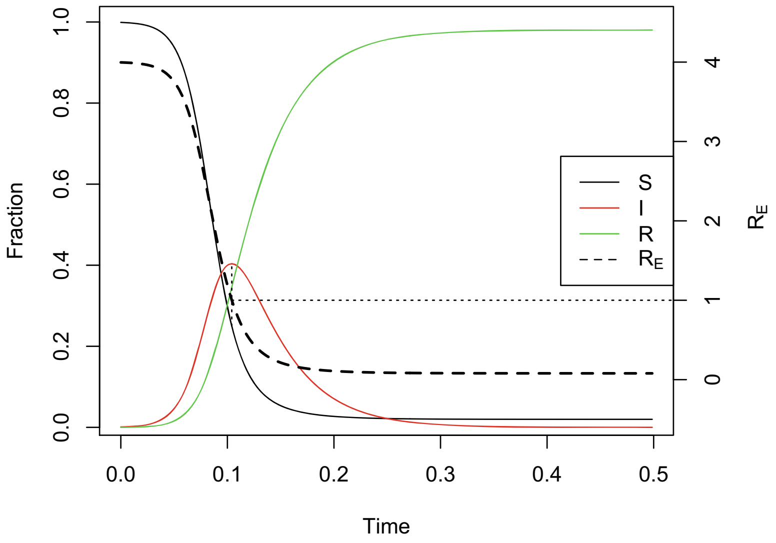

Fig. 2.2: The closed SIR epidemic with left and right axes and effective reproduction number, R_E. The epidemic turns over at $R_E = 1$

```
require(rootSolve)
equil = runsteady(y = c(S = 1 - 1e-05, I = 1e-05, R = 0),
    times = c(0, 1e+05), func = sirmod, parms = paras)
round(equil$y, 3)

  ##    S    I    R
  ## 0.02 0.00 0.98
```

So for these parameters, 2% of susceptibles are expected to escape infection altogether and 98%—the final epidemic size—are expected to be infected during the course of the epidemic.

The final epidemic size depends completely on R_0. For this specific SIR variant, $\beta = R_0(\gamma + \mu)$ and for the closed epidemic $\mu = 0$. Continuing to assume an infectious period of 2 weeks (i.e., $\gamma = 1/2$), we may vary R_0 from 0.1 to 5. For moderate to large R_0, this fraction has been shown to be approximately $1 - \exp(-R_0)$ (e.g., Anderson & May, 1982). We can check how well this approximation holds (Fig. 2.3).[6]

```
# Candidate values for R0 and beta
R0 = seq(0.1, 5, length = 50)
```

[6] The `for` loop here calculates the final epidemic size for a range of values of R_0; a loop works by repeating calculations (in this case 50 times), and after each repeat, the value of the looping variable (in this case `i`) is changed to the next value in the looping vector. So in this example `i` will be 1 first, then 2, and then ... until the loop ends after `i`= 50.

```r
betas = R0 * (paras["gamma"] + paras["mu"])
# Vector of NAs to be filled with numbers
fs = rep(NA, 50)
# Loop over l from 1, 2, ..., 50
for (i in seq(from = 1, to = 50, by = 1)) {
    equil = runsteady(y = c(S = 1 - 1e-05, I = 1e-05, R = 0),
        times = c(0, 1e+05), func = sirmod, parms = c(mu = 0,
            N = 1, beta = betas[i], gamma = 365/14))
    fs[i] = equil$y["R"]
}
plot(R0, fs, type = "l", xlab = expression(R[0]))
curve(1 - exp(-x), from = 1, to = 5, add = TRUE, col = "red")
```

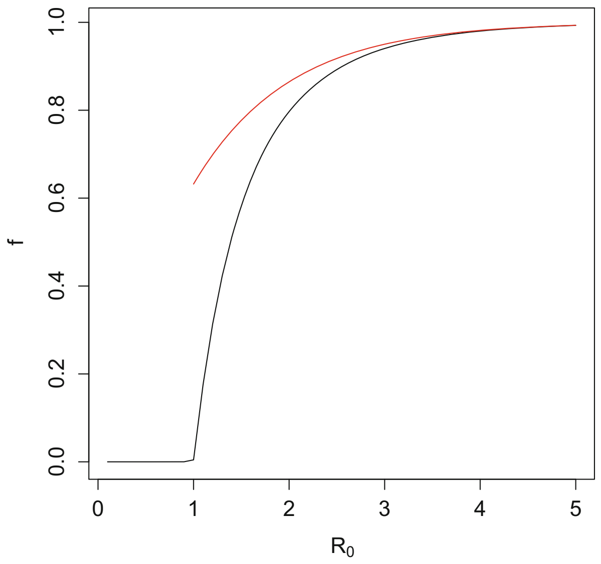

Fig. 2.3: The final epidemic size as a function of R_0. The black line is the solution based on numerically integrating the closed epidemic, the red line is the approximation $f \simeq 1 - \exp(-R_0)$

The approximation is good for $R_0 > 2.5$ but overestimates the final epidemic size for smaller R_0 (and is terrible for subcritical $R_0 < 1$).

For the closed epidemic SIR model, there is an exact mathematical solution to the fraction of susceptibles that escapes infection $(1 - f)$ given by the implicit equation $f = \exp(-R_0(1 - f))$ or equivalently $\exp(-R_0(1 - f)) - f = 0$ (Swinton, 1998). So we can also find the true expected final size by using the `uniroot` function to the equation. The `uniroot` function finds numerical solutions to equations with one unknown variable (which has to be named `x`).

```
# Define function
fn = function(x, R0) {
    exp(-(R0 * (1 - x))) - x
}
1 - uniroot(fn, lower = 0, upper = 1 - 1e-09, tol = 1e-09,
    R0 = 2)$root

  ## [1] 0.7968121

# check accuracy of approximation
exp(-2) - uniroot(fn, lower = 0, upper = 1 - 1e-09, tol = 1e-09,
    R0 = 2)$root

  ## [1] -0.06785259
```

So, for $R_0 = 2$, the final epidemic size is 79.6% and the approximation is off by around 6.7%-points. We will visit on stochastic aspects of the final epidemic size distribution in detail in Sect. 14.6.

2.5 The Open Epidemic

An open epidemic has recruitment of new susceptibles (i.e., $\mu > 0$). As long as $R_0 > 1$, the open epidemic has an endemic equilibrium were the pathogen and host coexist. If we use the SIR equations to model fractions (i.e., set $N = 1$), Eq. (2.2) of the SIR model implies that $S^* = (\gamma + \mu)/\beta = 1/R_0$ is the endemic S equilibrium, which when substituted into Eq. (2.1) gives $I^* = \mu(R_0 - 1)/\beta$, and finally, $R^* = N - I^* - S^*$ as the I and R endemic equilibrium values. We can study the predicted dynamics of the open epidemic using the `sirmod` function. In a stable host population with a life expectancy of 50 years, the per capita weekly birth/death rate is $\mu = 1/(50 * 52)$. For illustration, assume that 19.99% of the initial population is susceptible and 0.01% is infected, and numerically integrate the model for 50 years (Fig. 2.4).

```
times  = seq(0, 50, by=1/365)
paras  = c(mu = 1/50, N = 1, R0=4, gamma = 365/14)
paras["beta"]=paras["R0"]*(paras["gamma"]+paras["mu"])
start = c(S=0.1999, I=0.0001, R = 0.8)*paras["N"]
out = as.data.frame(ode(y=start, times=times,
     func=sirmod, parms=paras))
par(mfrow=c(1,2)) #Make room for side-by-side plots
#Prevalence in time
plot(times, out$I, ylab="Fraction", xlab="Time",
     type="l")
#S-I phase-plane
plot(out$S, out$I, type="l", xlab="Susceptible",
     ylab="Infected")
```

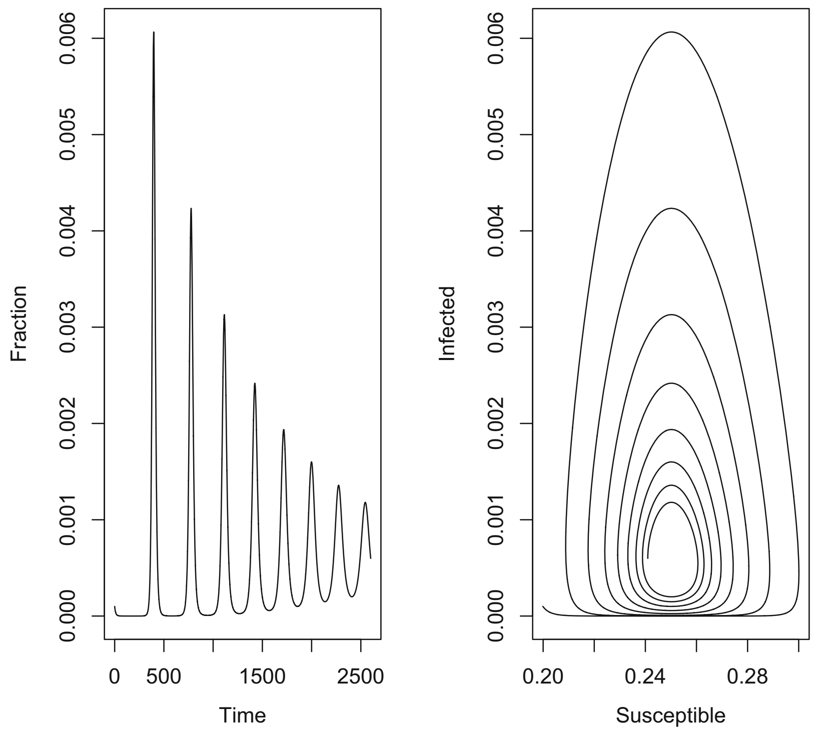

Fig. 2.4: The open SIR epidemic. (**a**) The fraction infected over time. (**b**) The joint time series of infecteds and susceptibles in the S–I phase plane. The trajectory forms a counterclockwise inward spiral in the S–I plane (note that the 50-year simulation is not long enough for the system to reach the steady-state endemic equilibrium at the center of the spiral)

2.6 Phase Analysis

When working with dynamical systems, one is often interested in studying the dynamics in the phase plane and deriving the isoclines that divide this plane into regions of increase and decrease of the various state variables. The `phaseR` package is a wrapper around `ode` that makes it easy to visualize 1- and 2-dimensional differential equation flows.[7] The R state in the SIR model does not influence the dynamics, so we can rewrite the SIR model as a 2D system.

[7] The `phaseR` package requires the gradient function to take the arguments `t`, `y`, and `parameters`.

```
simod = function(t, y, parameters) {
    S = y[1]
    I = y[2]

    beta = parameters["beta"]
    mu = parameters["mu"]
    gamma = parameters["gamma"]
    N = parameters["N"]

    dS = mu * (N - S) - beta * S * I/N
    dI = beta * S * I/N - (mu + gamma) * I
    res = c(dS, dI)
    list(res)
}
```

The isoclines (sometimes called the null-clines) in this system are given by the solution to the equations $dS/dt = 0$ and $dI/dt = 0$ and partition the phase plane into regions where S and I are increasing and decreasing. For $N = 1$, the I-isocline is $S = (\gamma + \mu)/\beta = 1/R_0$ and the S-isocline is $I = \mu(1/S - 1)/\beta$. We can draw these in the phase plane and add a simulated trajectory to the plot (Fig. 2.5). The trajectory cycles in a counterclockwise dampened fashion toward the endemic equilibrium (Fig. 2.5). To visualize the expected change to the system at arbitrary points in the phase plane, we can further use the function `flowField` in the `phaseR` package to superimpose predicted arrows of change.

```
require(phaseR)
#Plot vector field
fld = flowField(simod, xlim = c(0.2,0.3), ylim = c(0,.007),
     parameters = paras, system = "two.dim",
     add = FALSE, ylab = "I", xlab = "S")
#Add trajectory
out = as.data.frame(ode(y = c(S = 0.1999, I = 0.0001),
     times =  seq(0, 52*100, by = 1/365), func = simod,
     parms = paras))
 lines(out$S, out$I, col = "red")
#Add S-isocline
curve(paras["mu"]*(1/x-1)/paras["beta"], 0.15, 0.35,
     xlab = "S", ylab = "I", add = TRUE)
#Add I-isocline
icline = (paras["gamma"] + paras["mu"])/paras["beta"]
lines(rep(icline, 2), c(0,0.01))
legend("topright", legend = c("Transient", "Isoclines"),
     lty = c(1, 1), col = c("red", "black"))
```

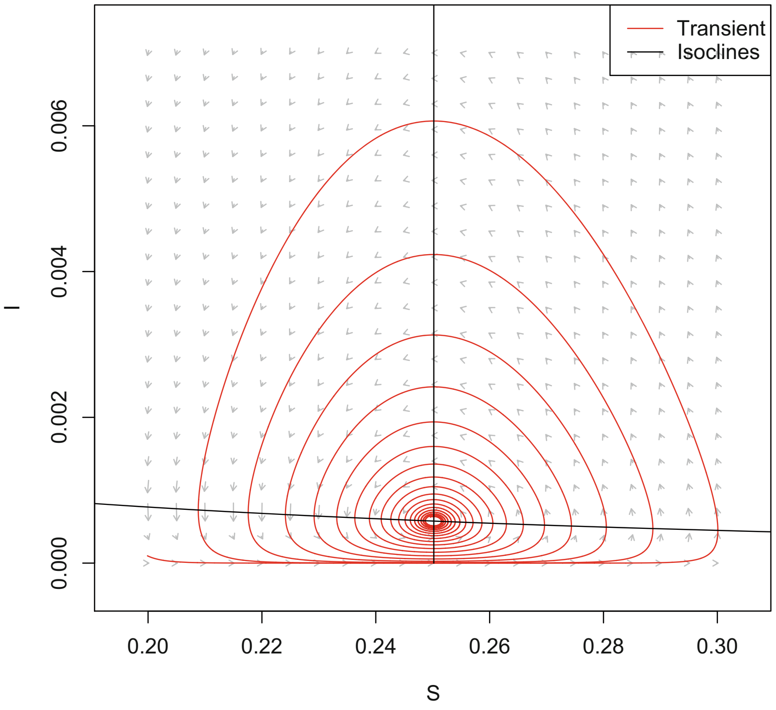

Fig. 2.5: The S–I phase plane with isoclines and the predicted counterclockwise trajectory toward the endemic equilibrium

2.7 Stability and Periodicity

As a preview of more detailed discussions in Chap. 10, this section is just a teaser. For continuous-time ODE models like the SIR, equilibria are locally stable if (and only if) all the real parts of the eigenvalues of the Jacobian matrix when evaluated at the equilibrium are smaller than zero. An equilibrium is (i) a node (i.e., all trajectories moves monotonically toward/away from the equilibrium) if the largest eigenvalue has only a real part and (ii) a focus (i.e., trajectories spiral toward or away from the equilibrium) if the largest eigenvalues are a conjugate pair of complex numbers $(a \pm b\imath)$.[8] For a focus, the imaginary part determines the dampening period of the cycle according to $2\pi/b$. We can thus use the Jacobian matrix to study the SIR model's equilibria. If we set $F = dS/dt = \mu(N - S) - \beta SI/N$ and $G = dI/dt = \beta SI/N - (\mu + \gamma)I$, the Jacobian of the SIR system is

[8] And (iii) a center, as is the case for the Lotka–Volterra predator–prey model, if the conjugate pair only has imaginary parts.

$$\mathbf{J} = \begin{pmatrix} \frac{\partial F}{\partial S} & \frac{\partial F}{\partial I} \\ \frac{\partial G}{\partial S} & \frac{\partial G}{\partial I} \end{pmatrix}, \tag{2.4}$$

and the two equilibria are the disease free equilibrium and the endemic equilibrium as defined above.

R can help with all of this. The endemic equilibrium is:

```
# Pull values from paras vector
gamma = paras["gamma"]
beta = paras["beta"]
mu = paras["mu"]
N = paras["N"]
# Endemic equilibrium
Sstar = (gamma + mu)/beta
Istar = mu * (beta/(gamma + mu) - 1)/beta
eq1 = list(S = Sstar, I = Istar)
```

The elements of the Jacobian using R's differentiation `D` function are

```
# Define equations
dS = quote(mu * (N - S) - beta * S * I/N)
dI = quote(beta * S * I/N - (mu + gamma) * I)
# Differentiate w.r.t. S and I
j11 = D(dS, "S")
j12 = D(dS, "I")
j21 = D(dI, "S")
j22 = D(dI, "I")
```

Pass the values for S^* and I^* in the `eq1` list to the Jacobian,[9] and use the `eigen` function to calculate the eigenvalues:

```
# Evaluate  Jacobian  at equilibrium
JJ = with(data = eq1, expr = matrix(c(eval(j11), eval(j12),
    eval(j21), eval(j22)), nrow = 2, byrow = TRUE))
# Calculate eigenvalues
eigen(JJ)$values

  ## [1] -0.04+1.250554i -0.04-1.250554i
```

For the endemic equilibrium, the eigenvalues are a pair of complex conjugates which real parts are negative, so it is a stable focus. The period of the inward spiral is:

[9] In previous coding of the `sirmod` function, parameter values were pulled from the input arguments inside the function to make the code as transparent as possible; while it makes the code easy to read, it makes for extra coding and can clutter up the workspace with variables that are defined in multiple locations. The `with` function allows the evaluation of an expression using variables defined in a data list.

```
2 * pi/(Im(eigen(JJ)$values[1]))

   ## [1] 5.024321
```

So with these parameters, the dampening period is predicted to be just over 5 years. Thus, during disease invasion, we expect this system to exhibit initial outbreaks every 5 years. A further significance of this number is that if the system is stochastically perturbed by environmental variability affecting transmission, the system will exhibit low-amplitude "phase-forgetting" cycles (Nisbet & Gurney, 1982) with approximately this period in the long run. We can make more accurate calculations of the stochastic system using transfer functions (Priestley, 1981; Nisbet & Gurney, 1982). We will visit on this more advanced topic in Sect. 10.8.

The same protocol can be used for the disease free equilibrium $\{S^* = 1, I^* = 0\}$.

```
eq2 = list(S = 1, I = 0)
JJ = with(eq2, matrix(c(eval(j11), eval(j12), eval(j21),
    eval(j22)), nrow = 2, byrow = TRUE))
eigen(JJ)$values

   ## [1] 78.27429 -0.02000
```

The eigenvalues are strictly real and the largest value is greater than zero, so it is an unstable node (a "saddle"); the epidemic trajectory is predicted to move monotonically away from this disease free equilibrium if infection is introduced into the system. This makes sense because with the parameter values used, $R_0 = 4$, which is greater than the invasion threshold value of 1.

Because we will require Jacobian matrices for a large number of different calculations regarding infectious disease dynamics, Sect. 6.8 will introduce a general-purpose `jacobian` function that is part of the `epimdr2` package.

2.8 Heterogeneities

The bare-bones SIR model makes many simplifying assumption. A lot of the theory in the subsequent chapters contends with making more realistic models by incorporating various heterogeneities. Important complications are age-dependence in susceptibility, infectiousness, contact rates and disease symptomology (Chaps. 4 and 5), a greater number of functionally distinct classes such as nosocomical (hospital associated) transmission being different from that in the community (Sect. 3.10), waning/boosting of immunity (Sect. 11.4), infections having multiple distinct outcomes (Sect. 10.6), seasonal changes in dynamics (Chap. 6), and spatial/social heterogeneities (Chaps. 12 and 14). The need to consider more elaborate models typically depends on the biology/ecology of the host and pathogen and the scientific problem in question.

2.9 Advanced: More Realistic Infectious Periods

As an initial illustrative example of added realism, we can consider how infectivity and removal rates are usually not constant during the course of infection. For acute pathogens, recently infected individuals are usually likely to be infected for a while longer, whereas individuals infected some time ago are likely to have a higher rate of removal either because the immunity is ramping up or increased risk of death or quarantining if disease severity increases over time. We can baby step toward solving the Kermack & McKendrick (1927) general equations of such time dependence by modifying the basic SIR model to consider more realistic infectious periods.

The S(E)IR-type differential equation models assume that the rate of exit from the infectious classes is constant, and the implicit assumption is thus that the infectious period is exponentially distributed among infected individuals. The average infectious period predicted from Eq. (2.2) is $1/(\gamma+\mu)$, but an exponential fraction is infectious much shorter/longer than this. The chain-binomial model, which will be discussed in Sect. 3.4, in contrast, assumes that everybody is infectious for a fixed period and then instantaneously recovers (or dies). These assumptions are mathematically convenient, but in reality neither are particularly realistic. Hope-Simpson (1952) traced the chains of transmission of measles in multi-sibling households. The timing of secondary and tertiary cases was analyzed in detail by Bailey (1956) and Bailey and Alff-Steinberger (1970). The average latent and infectious periods were calculated to be 8.58 and 6.57 days, respectively. While the distribution around each of these averages was not estimated separately (the latent period was assumed to be distributed and the infectious period assumed fixed), the variance around the roughly fortnight period of infection was estimated to be 3.13. The mean duration of infection is thus 15.15 days with a standard deviation of 1.77 (Fig. 2.6). So neither a fixed nor an exponential distribution is very accurate (Keeling & Grenfell, 1997; Lloyd, 2001).

Kermack and McKendrick's (1927) original model allows for arbitrary infectious period distributions. We can write Kermack and McKendrick's original equations as renewal equations (Breda et al., 2012), introducing the additional notation of $k(t)$ being the (instantaneous) incidence at time t (i.e., flux into the I class at time t).

$$\frac{dS}{dt} = \underbrace{\mu N}_{\text{birth}} - \underbrace{\mu S}_{\text{death}} - \underbrace{k(t)}_{\text{outflux}} \tag{2.5}$$

$$k(t) = \beta I(t)\frac{S(t)}{N} \tag{2.6}$$

$$\frac{dI}{dt} = \underbrace{k(t)}_{\text{influx}} - \underbrace{\mu I}_{\text{death}} - \underbrace{\int_0^\infty \frac{h(\tau)}{1-H(\tau)}k(t-\tau)d\tau}_{\text{distributed recovery}} \tag{2.7}$$

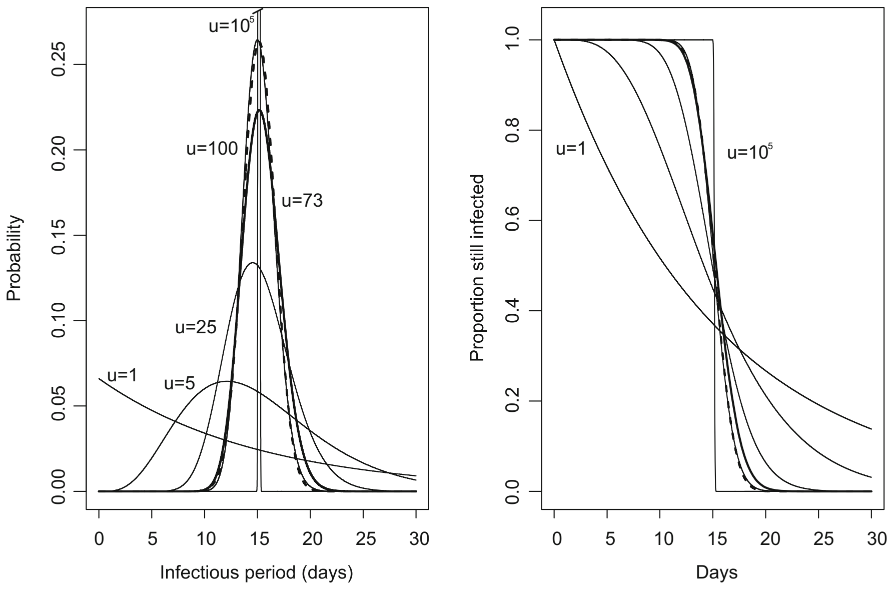

Fig. 2.6: Gamma distributed infectious periods. (**a**) The predicted infectious period distribution based on a gamma distribution with shape $u = 1$, 5, 25, 100, and 100,000; $u = 1$ corresponds to the exponential distribution implicit in the standard SIR model. The bold line ($u = 73$) is the one corresponding to the variance observed in Hope-Simpson's (1952) study of measles. The dotted line (virtually indistinguishable from the $u = 100$) is a Gaussian distribution intended to show that when u is large the gamma distribution converges on the normal distribution. (**b**) The probability of still being infectious as a function of time for the different distributions. As u becomes large, the distribution converges on a fixed infectious period. Note that the empirical distribution (bold) is quite different from the exponential ($u = 1$)

$$\frac{dR}{dt} = \underbrace{\int_0^\infty \frac{h(\tau)}{1-H(\tau)} k(t-\tau)d\tau}_{\text{distributed recovery}} - \underbrace{\mu R}_{\text{death}}, \tag{2.8}$$

where $k(t-\tau)$ is the number of individuals that were infected τ time units ago, $h(\tau)$ is the probability of recovering on infection day τ, and $H(\tau)$ is the cumulative probability of having recovered by infection day τ; $k(t-\tau)/(1-H(\tau))$ is thus the fraction of individuals infected at time $t-\tau$ that still remains in the infected class on day t and the integral is over all previous infections so as to quantify the total flux into the removed class at time t. Though intuitive, these general integro-differential equations (Eqs. (2.5)–(2.8)) are not easy to work with in general. For a restricted set of distributions for the $h()$ function however—the Erlang distribution (the gamma distribution with an integer shape parameter)—the model can be numerically integrated using a gamma-chain model (referred to as “linear chain trickery” by Metz & Diekmann, 1991) of coupled ordinary differential equations (e.g., Blythe et al., 1984; Lloyd, 2001; Bjørnstad et al., 2016). The trick is to separate any distributed-delay compartment into u sub-compartments through which individuals pass at a rate of $x * u$. The resultant infectious period will have a mean duration of $1/x$ and a coefficient of variation of $1/\sqrt{u}$.

A chain SIR model to simulate $S \rightarrow I \rightarrow R$ flows with more realistic infectious period distributions is:[10]

```r
sirChainmod = function(t, logx, parameters) {
    x = exp(logx)
    u = parameters["u"]
    S = x[1]
    I = x[2:(u + 1)]
    R = x[u + 2]
    with(as.list(parameters), {
        dS = mu * (N - S) - sum(beta * S * I)/N
        dI = rep(0, u)
        dI[1] = sum(beta * S * I)/N - (mu + u * gamma) *
            I[1]
        if (u > 1) {
            for (i in 2:u) {
                dI[i] = u * gamma * I[i - 1] - (mu + u *
                  gamma) * I[i]
            }
        }
        dR = u * gamma * I[u] - mu * R
        res = c(dS/S, dI/I, dR/R)
        list(res)
    })
}
```

We can compare the predicted dynamics of the simple SIR model with the $u = 2$ chain model, the $u = 500$ chain model (which is effectively the fixed-delay differential model), and the “measles-realistic” $u = 73$ model.

```r
times = seq(0, 10, by = 1/52)
paras2 = c(mu = 1/75, N = 1, R0 = 18, gamma = 365/14, u = 1)
paras2["beta"] = paras2["R0"] * (paras2["gamma"] + paras2["mu"])
xstart2 = log(c(S = 0.06, I = c(0.001, rep(1e-04, paras2["u"] -
    1)), R = 1e-04))
out = as.data.frame(ode(xstart2, times, sirChainmod, paras2))
plot(times, exp(out[, 3]), ylab = "Infected", xlab = "Time",
    ylim = c(0, 0.003), type = "l")
```

[10] With a high number of compartments, this system of equations can become “stiff” with the computer potentially making rounding errors leading to erroneous negative numbers. We use a “log-trick” available for systems where all state variables are strictly positive: we solve the system in log-coordinates to smooth abrupt changes and force all states to be greater than zero. To employ this technique, log-transform all initial values in `start`, change the first line in the function to `x = exp(logx)` and the last line to return `dS/S`, etc. in place of `dS` which comes from the chain-rule of differentiation and the fact that $D(\log x) = 1/x$.

```
paras2["u"] = 2
xstart2 = log(c(S = 0.06, I = c(0.001, rep(1e-04/paras2["u"],
    paras2["u"] - 1)), R = 1e-04))
out2 = as.data.frame(ode(xstart2, times, sirChainmod, paras2))
lines(times, apply(exp(out2[, -c(1:2, length(out2))]), 1,
    sum), col = "blue")

paras2["u"] = 73
xstart2 = log(c(S = 0.06, I = c(0.001, rep(1e-04/paras2["u"],
    paras2["u"] - 1)), R = 1e-04))
out3 = as.data.frame(ode(xstart2, times, sirChainmod, paras2))
lines(times, apply(exp(out3[, -c(1:2, length(out3))]), 1,
    sum), col = "red", lwd = 2, lty = 2)

paras2["u"] = 500
xstart2 = log(c(S = 0.06, I = c(0.001, rep(1e-04/paras2["u"],
    paras2["u"] - 1)), R = 1e-04))
out4 = as.data.frame(ode(xstart2, times, sirChainmod, paras2))
lines(times, apply(exp(out4[, -c(1:2, length(out4))]), 1,
    sum, na.rm = TRUE), col = "green")

legend("topright", legend = c("SIR", "u=2", "u=500",
    "u=73 (H-S)"), lty = c(1, 1, 1, 2), lwd = c(1, 1, 1, 2),
    col = c("black", "blue", "green", "red"))
```

The more narrow the infectious period distribution, the more punctuated the predicted epidemics. However, infectious period narrowing alone cannot sustain recurrent epidemics. In the absence of stochastic or seasonal forcing, epidemics will dampen to the endemic equilibrium (though the damping period is slightly accelerated and the convergence on the equilibrium is slightly slower with narrowing infectious period distributions) (Fig. 2.7).

In the above we considered non-exponential infectious period distributions. However, the general gamma-chain method can be used for any compartment. Lavine et al. (2011), for example, used it to model non-exponential waning of natural and vaccine-induced immunity to whooping cough.

2.10 An SIR shinyApp

The following code will launch a shinyApp of the SIR model in a local browser. This App can also be launched by calling `runApp(sir.app)` from the `epimdr2` package. Several of the subsequent chapters also have associated shinyApps. Those will be accessible from the `epimdr2` package or the `epimdr2` GitHub site, but not scripted in the text because the code is long and a bit tedious. The `sir.app` is presented here in full, so the interested readers can get a sense of shinyApp coding. Bjørnstad et al. (2020a) provide a more elaborate online accessible shinyApp to study the SIR model at https://shiny.bcgsc.ca/posepi1/.

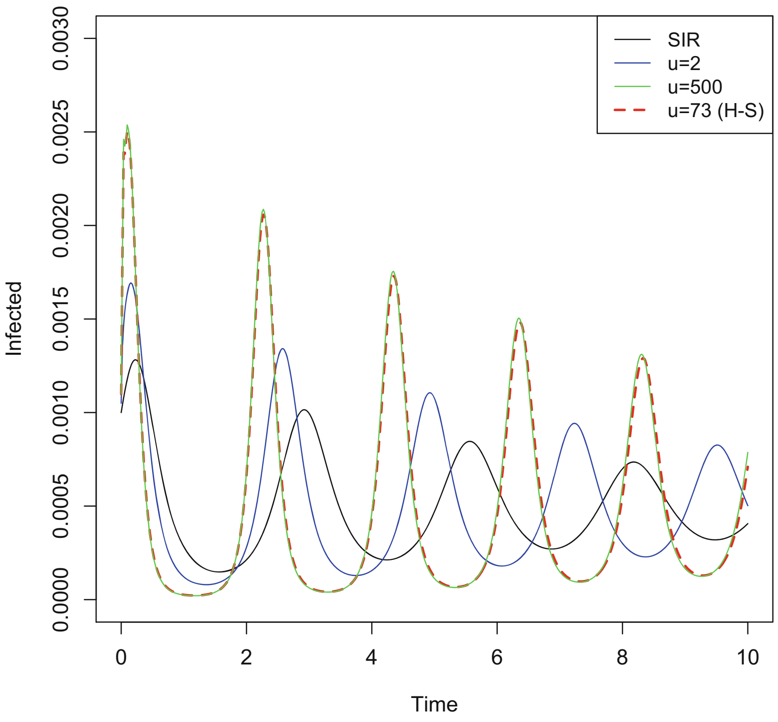

Fig. 2.7: Chain SIR models with different infectious period distributions

```r
require(shiny)
require(deSolve)
require(phaseR)

# This creates the User Interface (UI)
ui <- pageWithSidebar(
headerPanel("The SIR model"),
#The sidebar for parameter input
sidebarPanel(
#Sliders
sliderInput("R0", "R0:", 2,
              min = 0.5, max = 20),
sliderInput("infper", "Infectious period (days)", 5,
              min = 1, max = 100),
sliderInput("mu", "birth rate (yr^-1):", 5,
              min = 0, max = 100),
sliderInput("T", "Time range:",
                  min = 0, max = 1, value = c(0,1))
),
#Main panel for figures and equations
mainPanel(
  #Multiple tabs in main panel
  tabsetPanel(
    #Tab 1: Time plot (plot1 from server)
    tabPanel("Time", plotOutput("plot1")),
    #Tab 2: Phase plot (plot2 from server)
    tabPanel("Phase plane", plotOutput("plot2",
       height = 500)),
    #Tab 3: MathJax typeset equations
    tabPanel("Equations",
      withMathJax(
        helpText("Susceptible $$\\frac{dS}{dt} =
           \\mu (N - S) - \\frac{\\beta I S}{N}$$"),
        helpText("Infecitous $$\\frac{dI}{dt} =
           \\frac{\\beta I S}{N} - (\\mu+\\sigma) I$$"),
        helpText("Removed $$\\frac{dR}{dt} =
           \\gamma I - \\mu R$$"),
        helpText("reproduction number $$R_0 =
           \\frac{1}{\\gamma+\\mu} \\frac{\\beta N}{N}$$")
        ))
  ))) #End of ui()


# This creates the 'behind the scenes' code (Server)
server <- function(input, output) {
  #Gradient function for SIR model
  sirmod=function(t, x, parameters){
    S=x[1]
    I=x[2]
    R=x[3]
    R0=parameters["R0"]
    mu=parameters["mu"]
    gamma=parameters["gamma"]
    N=parameters["N"]
    beta=R0*(gamma+mu)
    dS = mu * (N  - S)  - beta * S * I / N
    dI = beta * S * I / N - (mu + gamma) * I
    dR = gamma * I - mu * R
    res=c(dS, dI, dR)
    list(res)
  }

 #Plot1: renderPlot to be passed to UI tab 1
  output$plot1 = renderPlot({
  #input\$xx's are pulled from UI
  times  = seq(0, input$T[2], by=1/1000)
  paras  = c(mu = input$mu, N = 1, R0 =  input$R0,
    gamma = 365/input$infper)
  start = c(S=0.999, I=0.001, R = 0)
  paras["beta"] = with(as.list(paras), R0*(gamma+mu))
  #Resonant period
  AA=with(as.list(paras), 1/(mu*(R0-1)))
  GG=with(as.list(paras), 1/(mu+gamma))
  rp=round(2*pi*sqrt(AA*GG),2)
  #Integrate ode with parameters pulled from UI
  out=ode(start,  times, sirmod, paras)
  out=as.data.frame(out)

  #Plot1
  sel=out$time>input$T[1]&out$time<input$T[2]
  plot(x=out$time[sel], y=out$S[sel], ylab="fraction",
  xlab="time", type="l", ylim=range(out[sel,-c(1,4)]))
  title(paste("R0=", paras["R0"], "Period=", rp))
  lines(x=out$time[sel], y=out$I[sel], col="red")
  lines(x=out$time[sel], y=out$R[sel], col="green")
  legend("right",
        legend=c("S", "I", "R"),
        lty=c(1,1,1),
         col=c("black", "red", "green"))
   })

 #Plot2: renderPlot to be passed to UI tab 2
  output$plot2 <- renderPlot({
  times  = seq(0, input$T[2], by=1/1000)
  paras  = c(mu = input$mu, N = 1, R0 =  input$R0,
    gamma = 365/input$infper)
  paras["beta"] = with(as.list(paras), R0*(gamma+mu))

  start = c(S=0.999, I=0.001, R = 0)

  #Gradient function used for phaseR phase-plot
  simod=function(t, y, parameters){
   S=y[1]
   I=y[2]
   beta=parameters["beta"]
   mu=parameters["mu"]
   gamma=parameters["gamma"]
   N=parameters["N"]
   dS = mu * (N  - S)  - beta * S * I / N
   dI = beta * S * I / N - (mu + gamma) * I
   res=c(dS, dI)
   list(res)
  }

  #Integrate simod
  out=ode(start[-3], times, simod, paras)
  out=as.data.frame(out)

  AA=with(as.list(paras), 1/(mu*(R0-1)))
  GG=with(as.list(paras), 1/(mu+gamma))
  rp=round(2*pi*sqrt(AA*GG),2)

  plot(x=out$S, y=out$I, xlab="Fraction suceptible",
     ylab="Fraction infected", type="l")
  title(paste("R0=", paras["R0"], "Period=", rp))
  #Add vector field
  fld=flowField(simod, xlim=range(out$S),
  ylim=range(out$I), parameters=paras,
  system="two.dim", add=TRUE,ylab="I", xlab="S")
  #Add isoclines
  abline(v=1/paras["R0"], col="green")
  curve(paras["mu"]*(1-x)/(paras["beta"]*x), min(out$S),
    max(out$S), add=TRUE, col="red")
  legend("topright",
        legend=c("S-isocline", "I-isocline"),
        lty=c(1,1),
       col=c("red", "green"))
   })
  } #End of server()

shinyApp(ui, server)
```