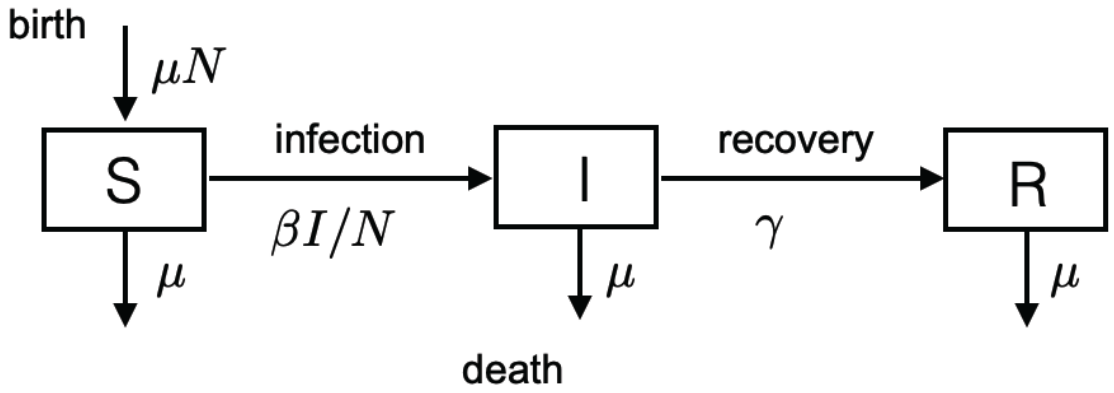
Session 4

Ottar N. Bjørnstad



From Session 2-3:

The basic equations for the flow of hosts between **S**usceptible, **I**nfectious and **R**ecovered compartments are:

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

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

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

```
require(epimdr2)
```

```
## Loading required package: epimdr2
```

```
## Loading required package: shiny
```

```
## Loading required package: deSolve
```

```
## Loading required package: plotly
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout

## Loading required package: polspline

require(deSolve)
```

STEP 1: Define the function (often called the gradient functions) for the equation systems. The deSolve package requires the function to take the following parameters: time t 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)
}
```

STEP 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 $N = 1$. Let's consider a disease with 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 which implies a transmission rate $$ of 2. For our 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). *For further details on usage, do* ?function on the R command line, ?ode* in this instance. 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
```

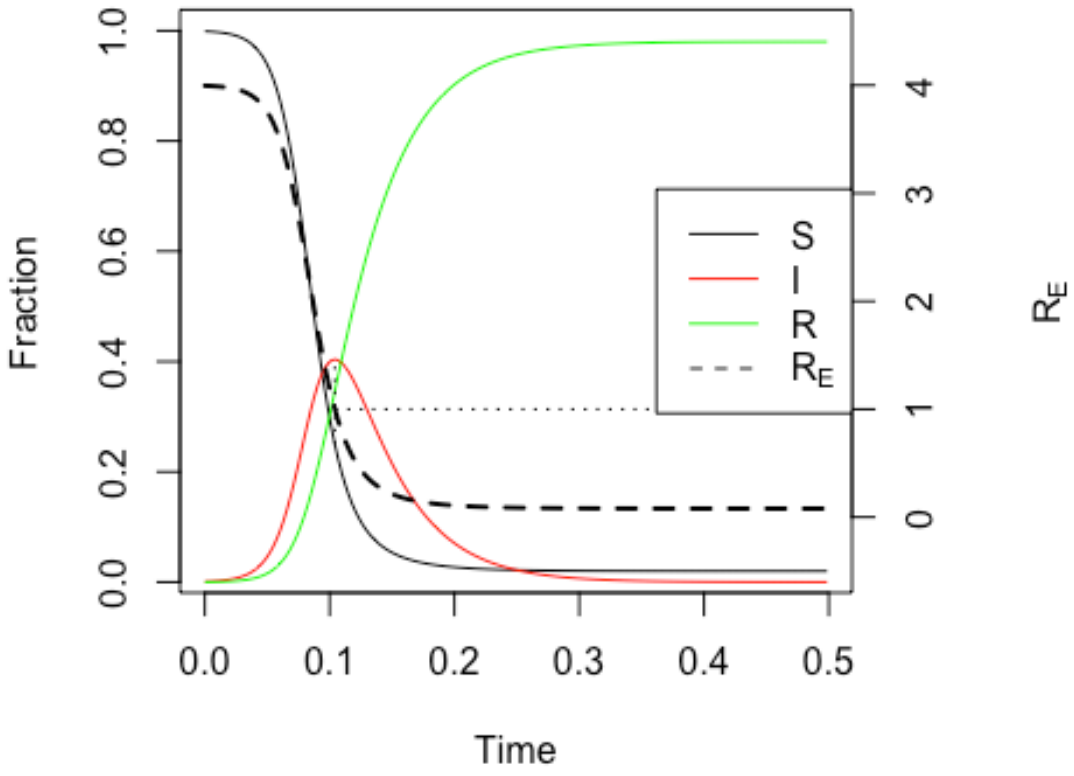
Plot:

```
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(-.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"))
```

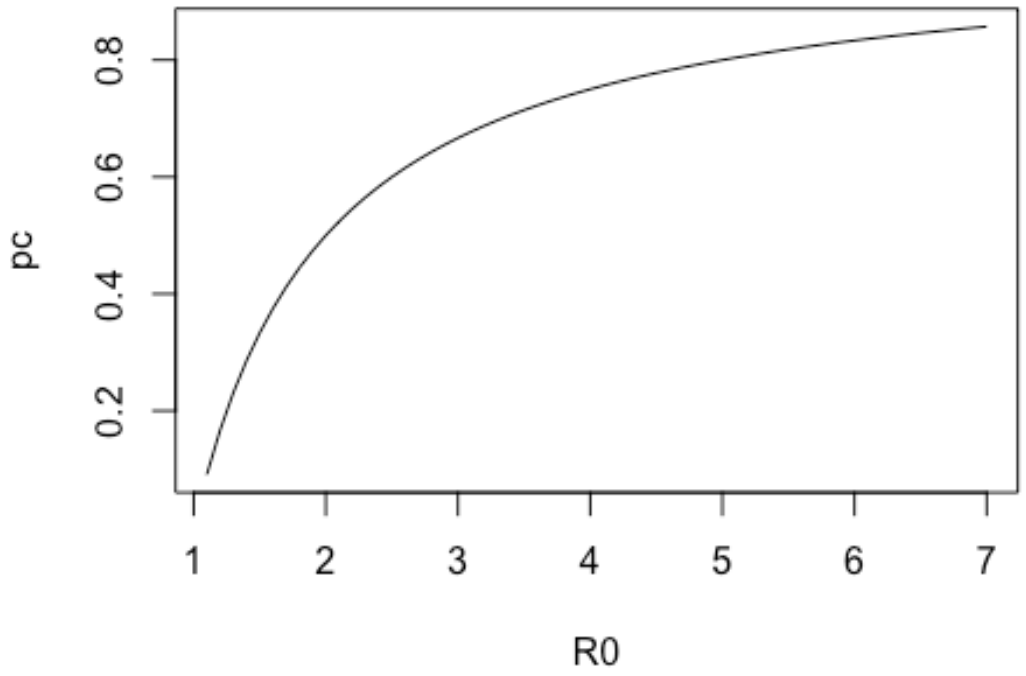


Reproduction numbers and control

The *basic reproction number* R_0 is the expected number of secondary cases from an index case in a completely susceptible population. The *effective* reproduction number (R_E) is the expected number of new cases per infected individuals in a **not** completely susceptible population. The plot confirms 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. This rule of thumb appeared to work well for smallpox, the only vaccine-eradicated human disease; Its R_0 was commonly around 5, and most countries saw elimination once vaccine cover exceeded 80%.

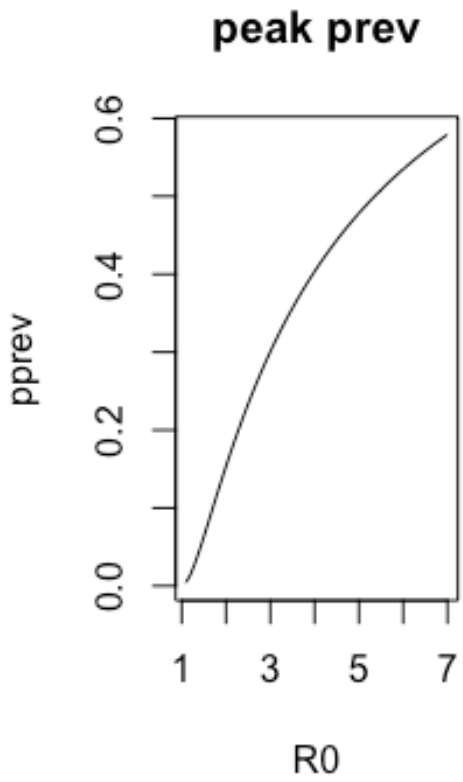
We can explore p_c as a function of R_0:

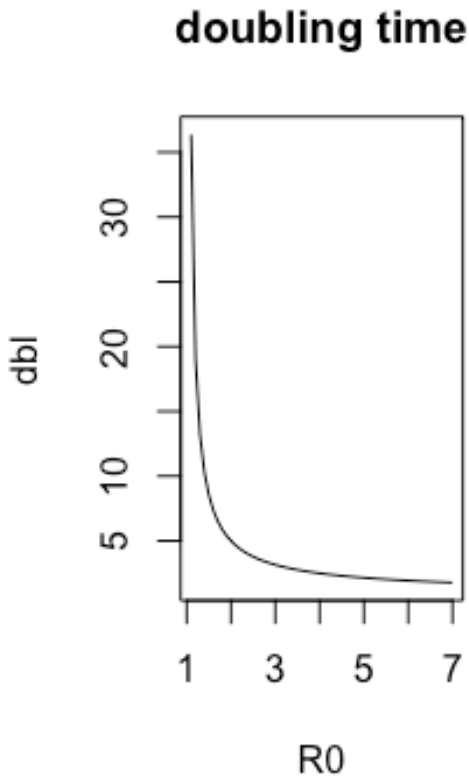
```
R0=seq(from=1.1, to=7, by=0.1)
pc=1-1/R0
plot(x=R0, y=pc, type="l")
```



In a closed epidemic the *peak prevalence* is $1-(1+\log R_0)/R_0$ and the early doubling time is $\log(2)V/\log R_0$, where V is the serial interval (the average infection-to-onwards-transmission time). We can consider the peak prevalence which is important for flattening the curve and doubling time. For Covid V was 5ish days:

```
R0=seq(from=1.1, to=7, by=0.1)
pprev=1-(1+log(R0))/R0
V=5
dbl=log(2) * V /log(R0)
par(mfrow=c(1,2)) #side by side plots
plot(x=R0, y=pprev, type="l")
title("peak prev")
plot(x=R0, y=dbl, type="l")
title("doubling time")
```


peak prev
pprev
0.0
0.2
0.4
0.6
1
3
5
7
R0


doubling time
dbl
5
10
20
30
1
3
5
7
R0