# Social Distancing: Solutions

John M. Drake & Pejman Rohani

***Exercise 1.** Using the approach outlined in this lab, show that the final epidemic size over a one-year period looks as below. Assume that initially everyone is susceptible and 1 out of 58 million are infectious. The social distancing strategy is in place for 12 weeks and $R_0 = 1.8$ with a mean infectious period of 2.6 days. You can ignore host births/deaths.

First, we load the `deSolve` package, specify parameters and initial conditions, and create a function to solve the social distancing model. For this exercise we use the "closed" $SIR$ model (no demography).

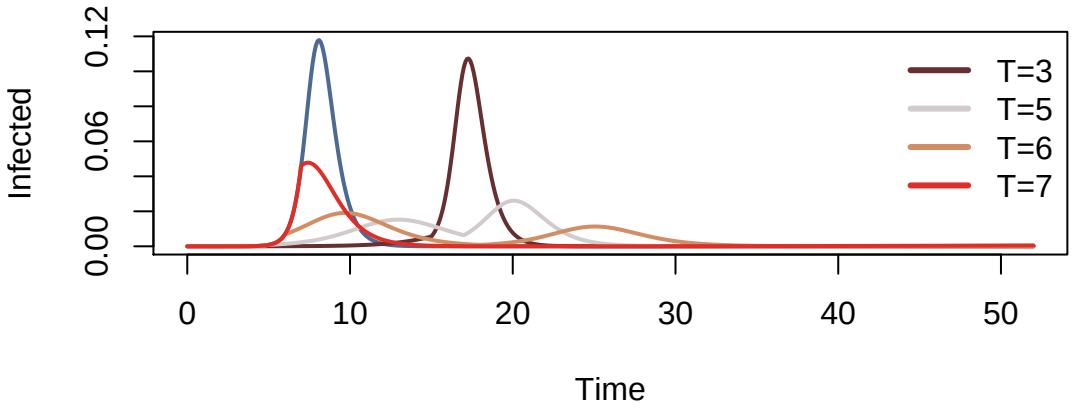
```
> require(deSolve)
> R0 <- 1.8
> N <-  1                                          #population size
> gamma <- 7/2.6                                  #recovery rate (in weeks)
> beta <- R0*(gamma)/N                  #transmission rate
> phi <- 0.3333
> D <- 12
> xstart <- c(S=0, I=0, R=0)          #initial conditions, must sum to one
> xstart[2] <- 1/58000000
> xstart[1] <- 1-xstart[2]
> xstart[3] <- 1-sum(xstart)
> Tmax <- 52                     #integrate for 200 years after transients
> params <- c(beta=beta, gamma=gamma)   #parameter vector
> tau <- 0.1                     #size of time step
> times <- seq(0, Tmax, by=tau) #function seq returns a sequence
> sir.model.closed <- function (t, x, params) {    #here we begin a function with three arguments
+   S <- x[1]                                 #create local variable S, the first element of x
+   I <- x[2]                                 #create local variable I
+   R <- x[3]                                 #create local variable R
+   with(                                     #we can simplify code using "with"
+        as.list(params),                     #this argument to "with" lets us use the variable names
+        {                                    #the system of rate equations
+          dS <- -beta*S*I
+          dI <- beta*S*I-gamma*I
+          dR <- gamma*I
+          dx <- c(dS,dI,dR)                  #combine results into a single vector dx
+          list(dx)                           #return result as a list
+        }
+        )
+ }
```

Evaluating the model at $\phi = 0.333$ and initialization times 3, 5, 6 and 7 generates the solution shown in the handout.

```
> out <- as.data.frame(ode(xstart,times,sir.model.closed,params, method='ode45', rtol=1e-7))
```

```
> # color palette
> pulp <- c(rgb(101,48,47, maxColorValue=255),    #brown
+           rgb(210,202,203, maxColorValue=255),  #gray
+           rgb(211,141,101, maxColorValue=255),  #peach
+           rgb(223,45,39, maxColorValue=255),    #red
+           rgb(250,208,10, maxColorValue=255),   #yellow
+           rgb(16,16,18, maxColorValue=255),     #black
+           rgb(76,106,147, maxColorValue=255)    #blue
+           )
> # plot baseline again
> plot(out$time,
+      out$I,
+      type='l',
+      lwd=2,
+      xlab='Time',
+      ylab='Infected',
+      col=pulp[7])  #plot the I variable against time
> # create a list to store information for the legend
> legend.info <- data.frame(T=numeric(0),col=character(0),stringsAsFactors=FALSE)
> Tvec <- c(3, 5, 6, 7)
> for(i in 1:length(Tvec)){
+   T <- Tvec[i]
+
+   out1 <- ode(xstart,                              # start at initial condition
+               seq(0,T,tau),                        # solve from 0 to T
+               sir.model.closed,params,
+               method='ode45',
+               rtol=1e-7)
+
+   out2 <- ode(tail(out1,1)[2:4],                   # start at end of last solution
+               seq(T,T+D,tau),                      # solve from T to T+D
+               sir.model.closed,
+               c(beta=beta*(1-phi), gamma=gamma),   # change beta
+               method='ode45',
+               rtol=1e-7)
+
+   out3 <- ode(tail(out2,1)[2:4],                   # start at end of last solution
+               seq(T+D, Tmax, tau),                 # solve from T+D to Tmax
+               sir.model.closed,
+               params,                              # reset parameters
+               method='ode45',
+               rtol=1e-7)
+
+   data <- as.data.frame(rbind(out1, out2, out3))
+
+   lines(data$time,
+         data$I,
+         col=pulp[i],
+         lwd=2)
+
+   legend.info[i,] <- c(T=T, col=as.character(pulp[i]))
+ }
```

```
> legend('topright',
+        legend=paste('T=',legend.info$T, sep=''),
+        lty=1,
+        lwd=3,
+        col=legend.info$col,
+        bty='n')
```




Solving the problem in the exercise requires evaluating the model over over values of $\phi$ and $T$ and storing the terminal value for the $R$ class, which contains every individual ever infected. Note: we make use of a convenient function, `outer`, in the following code to solve the problem more compactly than by looping. However, `outer` requires a function for which the arguments may be vectors (a "vectorized" function), which we create using the function `Vectorize`. We use the function `image.plot` from the `fields` package to generate the heat map.s

```
> total.epidemic <- function(T,phi){
+   require(deSolve)
+
+   R0 <- 1.8
+   N <-  1                                  #population size
+   gamma <- 7/2.6                           #recovery rate (in years)
+   beta <- R0*(gamma)/N                  #transmission rate
+   D <- 12
+   xstart <- c(S=0, I=0, R=0)           #initial conditions, must sum to one
+   xstart[2] <- 1/58000000
+   xstart[1] <- 1-xstart[2]
+   xstart[3] <- 1-sum(xstart)
+   Tmax <- 52                      #integrate for 200 years after transients
+   params <- c(beta=beta, gamma=gamma)   #parameter vector
+
+   out1 <- ode(xstart,                                 # start at initial condition
+               c(0,T),                              # solve from 0 to T
+               sir.model.closed,
```

```
+              params,
+              method='ode45',
+              rtol=1e-7)
+
+   out2 <- ode(tail(out1,1)[2:4],                    # start at end of last solution
+              c(T,T+D),                          # solve from T to T+D
+              sir.model.closed,
+              c(beta=beta*(1-phi),
+              gamma=gamma),     # change beta
+              method='ode45',
+              rtol=1e-7)
+
+   out3 <- ode(tail(out2,1)[2:4],                    # start at end of last solution
+              c(T+D, Tmax),                    # solve from T+D to Tmax
+              sir.model.closed,
+              params,                                # reset parameters
+              method='ode45',
+              rtol=1e-7)
+
+   total.epidemic <- out3[8]  #the last value of R
+ }
> Tvec <- seq(0.25, 20, length=100)
> phivec <- seq(0,0.5, length=100)
> data <- outer(Tvec,phivec, Vectorize(total.epidemic))
> require(fields)
> image.plot(Tvec, phivec, data, xlab='T', ylab=expression(phi))
```