Structured models for host heterogeneities*

John M. Drake & Pejman Rohani

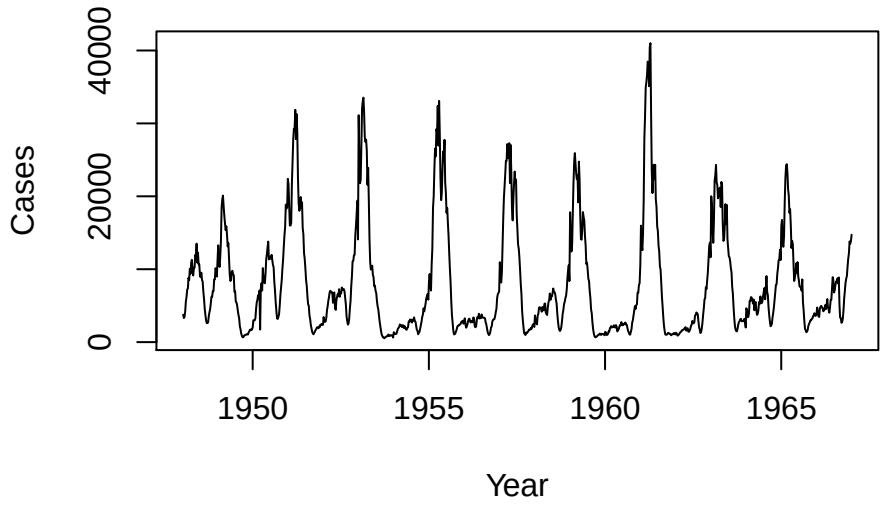
1 Introduction

In this session, we study a model for understanding how variation in host characteristics can affect epidemiological dynamics. As before, we will be solving differential equations and require the R package `deSolve`.

```
> require(deSolve)
```

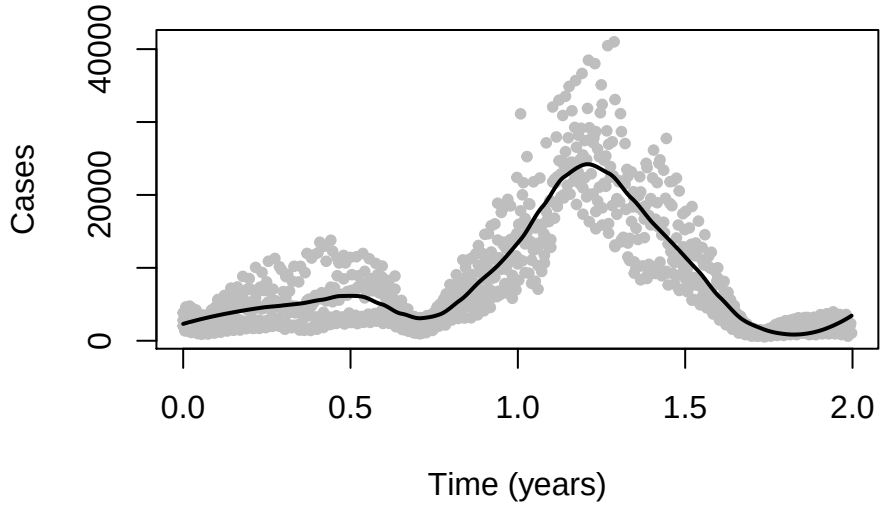
As a continuing example in this section, we will study the endemic dynamics of measles in England and Wales prior to vaccination, which exhibit periodic cycles. Our goal is to understand how age-structure and mixing among age-classes has given rise to these patterns. First, we look at the raw time series.

```
> load('data.RData')    #load the data
> plot(measles$Time,measles$Cases,type='l', xlab='Year',ylab='Cases')  #plot cases over time
```



Now, we overplot each two year period to look at the predictability of this cycle. Notice that the local linear regression smoother `loess`, which we use to draw a curve through the scatterplot, takes the same symbolic formula as the linear regression we studied in the first session.

```
> #plot measles cases in subsequent two year intervals using the constructed variable "TwoYEAR"
> plot(measles$TwoYear,measles$Cases,type='p',pch=20,col='grey',xlab='Time (years)',ylab='Cases')
> #fit a smooth line using loess -- notice data must be ordered for loess to fit properly
> smooth.cases<-loess(measles$Cases[order(measles$TwoYear)]~measles$TwoYear[order(measles$TwoYear)],
+                     span=0.3)
> lines(smooth.cases$x,smooth.cases$fitted,lwd=2)  #add smooth fit
```



One approach to modeling age structure is to allow individuals to age continuously. Such a model would consist of coupled *partial differential equations*. Another approach is to have individuals age all at once, once per year (perhaps at the start the school year, since we are concerned with the mixing of different aged children). This approach would consist of simultaneousy solving a large number (several hundreds) of ordinary differential equations. Both of these approaches are cumbersome programming and prone to errors of different kinds. Therefore, we will adopt a model that is a bit more of an approximation, but nevertheless might adequately represent the key dynamical processes. In what follows, we divide the population into four age classes, reflecting our best understanding of the main categories of transmission potential: pre-school (0-5 years), primary school (5-11 years), secondary school (11-19 years), and adults (20+). As described in the lecture portion for the model with two age classes, we allow the transmission among all age classes with pairwise rates (requiring 16 different transmission parameters–though we will assume that some of these 16 are the same. We will then have a system of 12 ODEs representing the rate of change of each class (S, E, and I) in each age group. As before, the equations will be solved with `lsoda`. Births and deaths will be represented in the usual way: births enter the youngest age class at a rate proportional to size of the population aged 20 and over, mortality occurs at rate 4.98×10^-5 per day for the oldest age class and zero for other age classes, giving an average life span of 75 years. Aging is accomplished once per year by moving $\frac{1}{6}$ of the first age class to the second age class, $\frac{1}{4}$ of the second age class to the third age class, and $\frac{1}{10}$ of the third age class to the fourth age class.

The following code sets up the model. Notice the R convention to represent matrix multiplication by %*%.

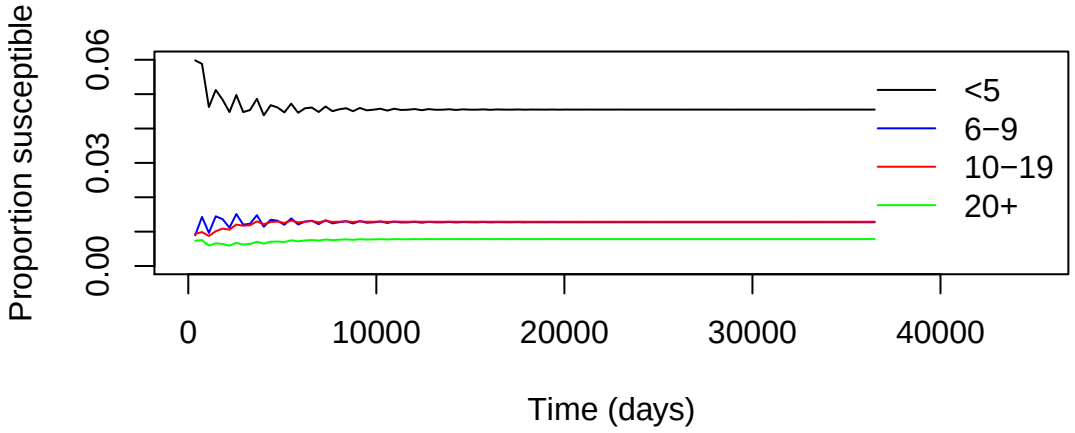
```
> age.model<-function(t,x,parms){  #a function to return derivatives of age structured model
+  S<-x[1:4]     #S are the first four elements of x
+  E<-x[5:8]     #E are the next four elements of x
+  I<-x[9:12]    #I are the last four elements ofx
+  dx<-vector(length=12)   #a vector to store the derivatives
+   for(a in 1:4){   #loop over age classes
+    tmp <- (parms$beta[a,]%*%I)*S[a]   #temporary variable with infection rate
+    dx[a] <- parms$nu[a]*55/75 - tmp - parms$mu[a]*S[a]                 #dS
+    dx[a+4] <- tmp - parms$sigma*E[a] - parms$mu[a]*E[a]                #dE
+    dx[a+8] <- parms$sigma*E[a] - parms$gamma*I[a] - parms$mu[a]*I[a]   #dI
+  }
+ return(list(dx))  #return the result
+ }
```

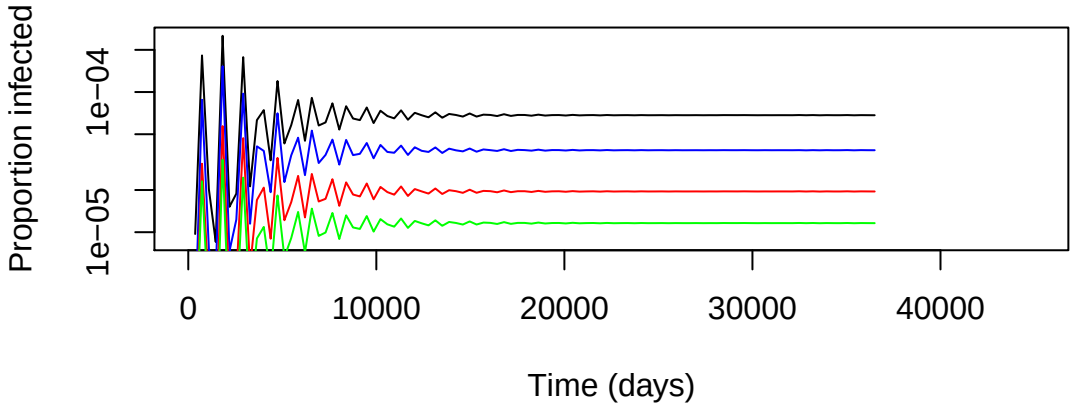
Now we set some initial conditions and parameters. By inspection of `age.model` we see that the first four elements are S for the four age classes, elements 5 through 8 are E for the four age classes, and so forth. We use the R function `matrix` to indicate that β is a matrix object and therefore an appropriate object for multiplication with the vector I in the first line of the `for` loop in the function `age.model`.

```
> y0<-c(0.05, 0.01, 0.01, 0.008, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001, 0.0001)
> #initialize state variables
>
> #a list of model parameters
> parms<-list(beta=matrix(c(2.089, 2.089, 2.086, 2.037, 2.089, 9.336, 2.086, 2.037, 2.086, 2.086,
+             2.086, 2.037, 2.037, 2.037, 2.037,2.037),nrow=4,byrow=TRUE),
+             sigma=1/8, gamma=1/5, nu=c(1/(55*365),0,0,0),  mu=c(0,0,0,1/(55*365)))
> parms

$beta
      [,1]  [,2]  [,3]  [,4]
[1,] 2.089 2.089 2.086 2.037
[2,] 2.089 9.336 2.086 2.037
[3,] 2.086 2.086 2.086 2.037
[4,] 2.037 2.037 2.037 2.037

$sigma
[1] 0.125

$gamma
[1] 0.2

$nu
[1] 4.98132e-05 0.00000e+00 0.00000e+00 0.00000e+00

$mu
[1] 0.00000e+00 0.00000e+00 0.00000e+00 4.98132e-05
```

Finally, we solve the equations one year at a time, moving age classes up between years, and plot.

```
> n=c(6,4,10,55)/75   #number of years in each age class
> maxTime <- 100*365  #number of days in 100 years
> T0=0                #initial time
> S=c()               #initialize S
> E=c()               #initialize E
> I=c()               #initialize I
> T=c()               #initialize T, a vector to hold times
> while(T0<maxTime){  #loop over times
+   y=lsoda(y0,c(T0, T0+365),age.model,parms) #solve diff'l equation for each time
+   T=rbind(T, y[2,1])      #store results
+   S=rbind(S, y[2,2:5])
+   E=rbind(E, y[2,6:9])
+   I=rbind(I, y[2,10:13])
+   #Now do the yearly movements
+   #Note use of "tail" to pull off the last value in a vector
+   y0[1]=tail(y,1)[2]-tail(y,1)[2]/6
+   y0[2]=tail(y,1)[3]+tail(y,1)[2]/6 - tail(y,1)[3]/4
+   y0[3]=tail(y,1)[4]+tail(y,1)[3]/4 - tail(y,1)[4]/10
+   y0[4]=tail(y,1)[5]+tail(y,1)[4]/10
+   y0[5]=tail(y,1)[6]-tail(y,1)[6]/6
+   y0[6]=tail(y,1)[7]+tail(y,1)[6]/6 - tail(y,1)[7]/4
+   y0[7]=tail(y,1)[8]+tail(y,1)[7]/4 - tail(y,1)[8]/10
+   y0[8]=tail(y,1)[9]+tail(y,1)[8]/10
+   y0[9]=tail(y,1)[10]-tail(y,1)[10]/6
+   y0[10]=tail(y,1)[11]+tail(y,1)[10]/6 - tail(y,1)[11]/4
+   y0[11]=tail(y,1)[12]+tail(y,1)[11]/4 - tail(y,1)[12]/10
+   y0[12]=tail(y,1)[13]+tail(y,1)[12]/10
+   T0=tail(T,1)
+ }


> #plot
> par(mfrow=c(2,1))    #set up plotting region
> plot(T,S[,1],type='l',xlim=c(0,45000),ylim=c(0,0.06),xlab='Time (days)',
+      ylab='Proportion susceptible') #plot susceptibles in youngest age class
> lines(T,S[,2],col='blue')           #susceptibles in second age class
> lines(T,S[,3],col='red')            #susceptibles in third age class
> lines(T,S[,4],col='green')          #susceptibles in oldest age class
> legend(x='topright',legend=c('<5','6-9','10-19','20+'),  #add legent
+      col=c('black','blue','red','green'),lty=1,bty='n')
> plot(T,I[,1],type='l',log='y',xlim=c(0,45000),xlab='Time (days)', #plot infected
+      ylab='Proportion infected')
> lines(T,I[,2],col='blue')
> lines(T,I[,3],col='red')
> lines(T,I[,4],col='green')
```





Evidently, the age-structured SIR model shows damped oscillations just like the non-structured version studied in the previous session.

Exercise 1. Modify the WAIFW matrix to reflect greater assortativity in mixing. What effect does this have on the epidemiological dynamics?