Mathematics 4MB3/6MB3 Mathematical Biology

ELECTRONIC SUPPLEMENTARY MATERIAL
Spatial epidemics dynamics: Synchronization

Model Students
Nicole Dumont, Melody Fong, Carolina Weishaar

Contents

1 Introduction

This is a supplemenary document for the report "Spatial epidemics dynamics: Synchronization" created for reproducibility. This supplement was created using `knitr`, an R package that merges output languages such as LaTeXwith coding languages, allowing for dynamic reproducible report generation [1]. Other packages used are `deSolve`, `titzDevice`, and `adaptivetau`.

```r
library("deSolve")
library("tikzDevice")
library("adaptivetau")
```

Output is currently suppressed during compile in the interest of speed - some code chunks take hours to run - but code can still be run using the uncompiled document `SupplementaryMaterial.Rnw`.

2 Bifurcation Diagram

The bifurcation diagram was created using `XPPAUT` following the script described in the supplementary by Krylova and Earn [2]. In the script, equations for the single patch model with sinusoidal forcing were passed to the argument. The program was then instructed to solve the ODE by evaluating the equations with given initial conditions and parameters ranging over the parameter $\mathcal{R}_0$ from 0-30 using Poincaré maps with period one year.

```
## Equations:
f(x,y)=mu-R0*(gamma+mu)*(1 + a*cos(2*pi*t))*x*y-mu*x
g(x,y)=R0*(gamma+mu)*(1 + a*cos(2*pi*t))*x*y-(gamma+mu)*y
x(t+1)=f(x,y)
y(t+1)=g(x,y)
R0(t+1)=R0

## Auxiliary Output Variables
aux R0

## Initial Conditions
init x=0.058824,y=0.00067043,R0=0
par a=0.1,mu=0.02,gamma=28.077
## auxiliary parameters to corresponding with
## deterministic/stochastic models

## Plotting in XPPAUT
@ xp=R0, yp=y
@ xlo=0,xhi=40,yhi=0.008,ylo=-0.008

## Range over R0
@ rangeover=R0,rangestep=5000
@ rangelow=0,rangehigh=30,rangereset=no
@ rangeoldic=yes,lt=0

## Storing variables
@ total=650,trans=0
@ maxstor=10000000
@ meth=discrete
@ output=bifurcation.dat
done
```

bifurcation.ode

When run using `XPPAUT`, this code outputs a file containing the values of time, susceptibles, infecteds, and $\mathcal{R}_0$ for all specified iterations of $\mathcal{R}_0$.

This file can be generated and read by R via the following console commands:

```r
system("xppaut bifurcation.ode -silent")

bfd <- read.table("bifurcation.dat", col.names=c("time","S","I","R0"))
```

The bifurcation diagram can be generated using the following command:

```r
plot(I$bfd,R0$bfd)
```

3 Deterministic Solution

The SIR meta-patch vector field function, `SIRmeta.vector.field`, returns the values of $\frac{dS_i}{dt}$, $\frac{dI_i}{dt}$, and $\frac{dR_i}{dt}$ for all patches (i.e. $1 \leq i \leq n$) at a given time, given the values of S_i, I_i, and R_i for all patches and the values of parameters $\mathcal{R}_0$ (the basic reproductive number), μ (the death/birth rate), γ (the recovery rate), α (the strength of seasonal forcing), m (the mixing parameter) and EC (a switch: if `TRUE` use equal coupling, otherwise use nearest neighbour coupling).

It creates the beta matrix:

$$\beta(t) = \langle\beta\rangle\,(1 + \alpha\cos(2\pi t))M \tag{1}$$

where M is the mixing matrix. If equal coupling is used then

$$M = \begin{bmatrix} 1-m & \frac{m}{n-1} & \frac{m}{n-1} & \frac{m}{n-1} \\ \frac{m}{n-1} & 1 & \frac{m}{n-1} & \frac{m}{n-1} \\ \frac{m}{n-1} & \frac{m}{n-1} & 1 & \\ & & & \\ \frac{m}{n-1} & & & \ddots \end{bmatrix}$$

Otherwise nearest neighbour coupling is used.

$$M = \begin{bmatrix} 1-m & \frac{m}{2} & 0 & 0 & \dots & \frac{m}{2} \\ \frac{m}{2} & 1-m & \frac{m}{2} & 0 & & \vdots \\ 0 & \frac{m}{2} & 1-m & & & \\ 0 & 0 & & \ddots & & \\ \vdots & & & & \ddots & \frac{m}{2} \\ \frac{m}{2} & & & \dots & \frac{m}{2} & 1-m \end{bmatrix}$$

The proportional dynamics of a single patch in the meta-patch SIR model are given by

$$\begin{aligned} \frac{dS_i}{dt} &= \mu - S_i \sum_{j=1}^{n} \beta_{ij}(t) I_j - \mu S_i \\ \frac{dI_i}{dt} &= S_i \sum_{j=1}^{n} \beta_{ij}(t) I_j - \gamma I_i - \mu I_i \\ \frac{dR_i}{dt} &= \gamma I_i - \mu R_i \end{aligned} \tag{2}$$

```r
SIRmeta.vector.field <- function(t,vars,parms=NULL) {
  ##parms should be of form (R_0,gamma,mu,alpha,m,EC)
  ##vars should be of form (S,I,R)
  with(as.list(c(parms,vars)), {  #So that it can call variables stored in parms
    ##vars contains the values of S, I, and R at time t
    n <- length(vars)/3
    S <- vars[1:n]
    I <- vars[(n+1):(2*n)]
    R <- vars[(2*n+1):(3*n)]

    #Equal Coupling
    if(EC==TRUE){
      ##mixing matrix
      M <- matrix(m/(n-1),n,n)+(1-m*(1+1/(n-1)))*diag(n)
       ##beta matrix
      betam <- R_0*(gamma+mu)*(1+alpha*cos(2*pi*t))*M
    }else{
      #Nearest Neighbors
      ##mixing matrix
      M <- (1-m)*diag(n)
      M[row(M)%%n==(col(M)+1)%%n ] <- m/2
      M[row(M)%%n==(col(M)-1)%%n] <- m/2
      ##beta matrix
      betam <- R_0*(gamma+mu)*(1+alpha*cos(2*pi*t))*M
    }
    dS <- NULL
    dI <- NULL
    dR <- NULL
    for (i in 1:n){
      dS[i] <- mu*(1-S[i]) - S[i]*sum(betam[i,]*I) ##dS_i/dt
      dI[i] <- S[i]*sum(betam[i,]*I) - (mu+gamma)*I[i]  ## dI_i/dt
      dR[i] <- gamma*I[i] -mu*R[i]
    }
    vec.fld <- c(dS=dS,dI=dI,dR=dR)
    return(list(vec.fld))
  })
}
```

The **run.soln** function returns the values of S_i, I_i, and R_i for all patches at times given in **times**, given the initial conditions, the vector field function, and parameter to pass to the vector field function. It also returns the value of "coherence" at all times. A state is said to be coherent at time step t when $I_1(t) = I_2(t) = ... = I_n(t)$ for all n patches [3]. The measure used here of "how coherent" a state is is the coefficient of variation, also called the relative standard deviation, of $\{I_i(t)\}_{i=1}^n$. It is the standard deviation of these values divided by the mean. The smaller it is the closer all states are to the mean and the more coherent the state is.

$$\begin{aligned} c_v(t) &= \frac{SD(I_i(t))}{\overline{I_i(t)}} \\ &= \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(I_i(t) - \overline{I_i(t)}\right)}}{\overline{I_i(t)}} \end{aligned} \tag{3}$$

This is calculated using R's native sd and mean functions.

```
## Get Solutions and coherence measure
run.soln <- function(n,ic=c(S=rep(0,n),I=rep(0,n),R=rep(1,n)),tmax=1,
                     times=seq(0,tmax,
                               by=tmax/1000),
                     func,parms) {
  soln <- ode(ic,times,func,parms)

   #Measure of coherence: the coefficient of variation/relative standard deviation of
  #all the patches at each time step
  coherence <- rep(0,length(times))
  for(tims in 1:length(times)){
    coherence[tims]<- sd(soln[tims,paste0('I',1:n)])/mean(soln[tims,paste0('I',1:n)])
    if(is.nan(coherence[tims])){
      coherence[tims]<- 0
    }
  }
  return(list("times"=times, "soln" = soln, "coherence"= coherence))
}
```

Function **plot.SIRmeta** uses **run.soln** to get the solution of the deteminitic model with the given initial conditions and parameters, and plots the results. It should be noted that when ouputting the images, they were run through the **tikzDevice** package for R to enable the use of LaTeXwithin figures. The images were also saved to PDF through **tikzDevice**.

```r
#Given ics and paramters, plot the det soln of the SIR meta model to pdf
plot.SIRmeta <- function(n,ic=c(S=rep(0,n),I=rep(0,n),R=rep(1,n)),tmax=1,
                         parms,...){
  ##parms should be of form (R_0,gamma,mu,alpha,m,EC)
  with(as.list(c(parms)), {  #So that it can call variables stored in parms
  ## get solutions
  run <- run.soln(n=n,ic=ic,tmax=tmax,
             func=SIRmeta.vector.field,
             parms=parms)
  times<-run$times
  soln<-run$soln
  coherence<-run$coherence

  #Print to pdf using tikz
  filename <- paste0("images/det",if(EC) "EC" else "NN", "R0", R_0, "m", m, ".tex")
  if(file.exists(filename)){ #If the filename is already used
    filename <- paste0("images/det",if(EC) "EC" else "NN", "R0", R_0, "m", m, "(1).tex")
  }
  if(file.exists(filename)){ #If the filename(1) is already used
    filename <- paste0("images/det",if(EC) "EC" else "NN", "R0", R_0, "m", m, "(2).tex")
  }

  tikz(filename,standAlone=TRUE,width=6,height=6)

  #leave room on right margin for coherence label
  par(mar = c(5,5,2,5))
  #Calculates I* to estimate how high the y-axis needs to be
  IEE <- (1 - 1/R_0)*mu/gamma
  ## draw box for plot
  plot(0,0,xlim=c(0,tmax),ylim=c(0,IEE*5),type="n",xlab="Time (years)",
     ylab="Prevalence (I)",las=1)

  #Plot I for all patches with different colours and lty
  for (i in 1:n){
    lines(times,soln[,paste0('I',i)],col=i,lty=i)
  }

  #Plot coherence over top
  par(new = T)
  plot(times,coherence,type='l',lwd=2,axes=F,xlab=NA, ylab=NA,ylim=c(0,1))
  axis(side=4)
  #Label left axis
  mtext(side = 4, line = 3, 'Coherence')

  #Make legend
  labels <- c(1:n)
  for(i in 1:n){
    labels[i]<-paste('$I_{',i,'}$',collapse=NULL)
  }
  legend("topright",legend=labels,col=1:n,lty=1,bty='n')
  dev.off()
  tools::texi2dvi(filename,pdf=T)
  })
}
```

The following chunk of code uses `plot.SIRmeta` to plot solutions to the meta-patch model for different parameters and with initial conditions within $\pm 30\%$ of the Endemic Equilibrium (EE) to preserve asymptotic local stability of the system[4]. This is so that the solutions converge quickly to a stable solution. The endemic equilibruim is

$$\begin{aligned} S^* &= \frac{1}{\mathcal{R}_0} \\ I^* &= \frac{\mu}{\gamma}\left(1 - \frac{1}{\mathcal{R}_0}\right) \\ R^* &= 1 - S^* - I^* \end{aligned} \tag{4}$$

```
#Parameters
R_0 <- 17   ##basic reproduction rate (if the system had no forcing)
gamma <- 365/13  ##inverse of mean infectious period
mu <- 1/50  ##death and birth rate
alpha <- 0.1  ##strength of seasonal forcing
n <- 10  ## Number of patches
m <- 0.2  ##Connectivity matrix parameter
EC <- FALSE ##if true use equal coupling, if false use nearest neighbors

##Initial conditions
#Want ICs near EE for stability. Find EE values:
SEE <-1/R_0
IEE <-(1 - 1/R_0)*mu/gamma
REE <-1-SEE-IEE
#Use random initials conditions within +/-30% of EE values
set.seed(34) #For reproducbility
percent <- 0.3
S0 <- runif(SEE*(1-percent), SEE*(1+percent), n)
I0 <- runif(IEE*(1-percent), IEE*(1+percent), n)
R0 <- 1-S0-I0

tmax<-15

#### Plotting lots of solutions to pdf ####
#Making detNNR017m0.2
plot.SIRmeta(n=n,ic=c(S=S0,I=I0,R=R0),tmax=tmax,
                        parms=c(R_0=R_0,gamma=gamma,mu=mu,alpha=alpha,m=m,EC=EC))
#Making detNNR010m0.2
R_0<-10
plot.SIRmeta(n=n,ic=c(S=S0,I=I0,R=R0),tmax=tmax,
                        parms=c(R_0=R_0,gamma=gamma,mu=mu,alpha=alpha,m=m,EC=EC))
#Making detNNR025m0.2
R_0<-25
plot.SIRmeta(n=n,ic=c(S=S0,I=I0,R=R0),tmax=tmax,
                        parms=c(R_0=R_0,gamma=gamma,mu=mu,alpha=alpha,m=m,EC=EC))
#Making detECR017m0.2s
R_0<-17
tmax<-5
EC<-TRUE
plot.SIRmeta(n=n,ic=c(S=S0,I=I0,R=R0),tmax=tmax,
                      parms=c(R_0=R_0,gamma=gamma,mu=mu,alpha=alpha,m=m,EC=EC))
#Making detNNR017m0.2s
EC<-FALSE
plot.SIRmeta(n=n,ic=c(S=S0,I=I0,R=R0),tmax=tmax,
                      parms=c(R_0=R_0,gamma=gamma,mu=mu,alpha=alpha,m=m,EC=EC))
#Making detNNR017m0.01
tmax<-15
m<-0.01
plot.SIRmeta(n=n,ic=c(S=S0,I=I0,R=R0),tmax=tmax,
                      parms=c(R_0=R_0,gamma=gamma,mu=mu,alpha=alpha,m=m,EC=EC))
#Making detNNR017m0.01wider
percent <- 0.4
S0 <- runif(SEE*(1-percent), SEE*(1+percent), n)
I0 <- runif(IEE*(1-percent), IEE*(1+percent), n)
R0 <- 1-S0-I0
plot.SIRmeta(n=n,ic=c(S=S0,I=I0,R=R0),tmax=tmax,
                      parms=c(R_0=R_0,gamma=gamma,mu=mu,alpha=alpha,m=m,EC=EC))
```

4 Stochastic Simulation

Stochastic simulations of the model were also used.

4.1 Gillespie Algorthim

The Gillespie algorthim is a method of exactly simulating a stochastic process. Consider a process involving transitions between different states. The rate for a transition of type i is a_i. The total transistion rate is the sum of the rates of all possible transitions:

$$a_0 = \sum a_i \tag{5}$$

If we assume that the time spent in a given state is exponentially distributed, then by sampling from an exponential distribution with rate a_0 (using R's `rexp` function) we can simulate how much time was spent in a given state before the next transition.

The probability that transition i occured is $\frac{a_i}{a_0}$. By sampling from a uniform distribution over $[0, a_0]$ we can simulate which transition occured in the following way: if the sampled value is in the i^{th} interval from the list

$$[0, a_1), [a_1, a_1 + a_2), \dots [a_1 + a_2 + \cdots + a_{i-1}, a_1 + a_2 + \cdots + a_i), \dots \tag{6}$$

than say transition i occured.

For this problem, the states are the values of $\{(S_i, I_i, R_i)\}_{i=1}^{n}$. The state space is all possible set of n triplets of non-negative integers. Possible transitions are:

- A birth ($S_i \to S_i + 1$)
- A transmission ($S_i \to S_i - 1$ and $I_i \to I_i + 1$)
- A recovery ($I_i \to I_i - 1$ and $R_i \to R_i + 1$)
- A death of a susceptible ($S_i \to S_i - 1$)
- A death of an infected ($I_i \to I_i - 1$)
- A death of a recovered ($R_i \to R_i - 1$)

for $1 \leq i \leq n$. The transition rates are:

- The birth rate $\mu(S_i + I_i + R_i)$

- The transmission rate $\frac{S_i}{S_i+I_i+R_i}\sum_{j=1}^{n}\beta_{ij}(t)I_j$
- The recovery rate γI_i
- The susceptible death rate μS_i
- The infected death rate μI_i
- The recovered death rate μR_i

for $1 \leq i \leq n$.
The function `SIRmeta.Gillespie` implements a stochastic simulation of the meta-patch SIR model using the Gillespie algorthim. It returns a list of times (starting at zero and ending near the input variable `tmax` with non-uniform step sizes) and the values of S_i, I_i, and R_i for all patches at these times given initial conditions and parameter values.

```
SIRmeta.Gillespie <- function(tmax,ic=c(S0,I0,R0),
                              parms=c(R_0=2,gamma=0.25,mu=4e-5,
                                      alpha=0.1,m=0.2,EC=TRUE)){
  times <- 0

  n <- length(ic)/3
  #S, I, R matrices: column index is patch index and rows (added later in while loop)
  #are values at different time steps
  S <- matrix(0,1,n)
  I <- matrix(0,1,n)
  R <- matrix(0,1,n)
  #Initial Coniditions
  S[1,] <- ic[1:n]
  I[1,] <- ic[(n+1):(2*n)]
  R[1,] <- ic[(2*n+1):(3*n)]

  index <- 1
  while(times[index] < tmax){
    #A matrix of all rates
    rates <- matrix(0,n,6)
    #the first column is all birth rates
    rates[,1] <- mu*(S[index,]+I[index,]+R[index,])
    #thrid column are recovery rates
    rates[,3] <- gamma*I[index,]
    #last three columns are death rates from S and I
    rates[,4] <- mu*S[index,]
    rates[,5] <- mu*I[index,]
    rates[,6] <- mu*R[index,]
    #Second column are transmission rates
    #Equal Coupling
    if(EC){
      M <- matrix(m/(n-1),n,n)+(1-m*(1+1/(n-1)))*diag(n)   ##connectivity matrix
      betam <- R_0*(gamma+mu)*(1+alpha*cos(2*pi*times[index]))*M  ##beta matrix
    }else{
      #Nearest Neighbors
      M <- (1-m)*diag(n)
      M[row(M)%%n==(col(M)+1)%%n ] <- m/2
      M[row(M)%%n==(col(M)-1)%%n] <- m/2
      betam <- R_0*(gamma+mu)*(1+alpha*cos(2*pi*times[index]))*M  ##beta matrix
    }
    rates[,2] <- S[index,]*colSums(t(betam[1:n,])*I[index,])/(S[index,]+I[index,]+R[index,])
    totalrate <- sum(rates)

    #Getting a timestep using R's exp distribution
    timestep <- rexp(n=1, rate= totalrate)
    times <- rbind(times, times[index]+timestep)

    #Adding new row to S, I, R
    S <- rbind(S, S[index,])
    I <- rbind(I, I[index,])
    R <- rbind(R, R[index,])

    #Getting which compartment transistion occurred
    #TO DO: Do a bisection search (or some other faster method) instead
    #Get a random number from a uniform distribution
    randomvar <- runif(n=1,min=0,max=totalrate)
    interval <- 1
    #Make the rates matrix into a vector for conveince
    rateslisted <- c(0,as.vector(t(rates)))
    #Find which interval randomval is in from list:
    #[0,a_1), [a_1, a_1+a_2), etc
    #If in interval [a_1 + .. + a_(i-1), a_1 + ... + a_i),
    #transition given by rate a_i occurred
    while(interval < (6*n)){
      if(randomvar >= sum(rateslisted[1:interval]) &
         randomvar < sum(rateslisted[1:(interval+1)])){
        break
      }else{
        interval<- interval+1
      }
    }
    #Get row, col indices of a_i in rates matrix
    #row index says in which patch a transition occurred
    #col index says what type of transition occurred
    colInd <- interval%%6
    rowInd <- ceiling(interval/6)
    if(colInd ==1){  ##A birth occurred
      S[index+1,rowInd] <- S[index,rowInd]+1
    }else if(colInd ==2){  ##An infection occurred
      S[index+1,rowInd] <- S[index,rowInd]-1
      I[index+1,rowInd] <- I[index,rowInd]+1
    }else if(colInd ==3){  ##A recovery occurred
      I[index+1,rowInd] <- I[index,rowInd]-1
      R[index+1,rowInd] <- R[index,rowInd]+1
    }else if(colInd ==4){  ##A death in S occurred
      S[index+1,rowInd] <- S[index,rowInd]-1
    }else if(colInd ==5){  ##A death in I occurred
      I[index+1,rowInd] <- I[index,rowInd]-1
    }else if(colInd ==0){  ##A death in R occurred
      R[index+1,rowInd] <-  R[index,rowInd]-1
    }
```

```
    index <- index+1
  }
  return(cbind(times, S, I, R))
}
```

The following code uses the `SIRmeta.Gillespie` function to plot a stochastic simulation of the model.

```
#For reprodubility set the seed
set.seed(899)

tmax <- 10
n <- 10

#Parameters
R_0 <- 17    ##basic reproduction rate (if the system had no forcing)
gamma <- 365/13  ##inverse of mean infectious period
mu <- 1/50  ##death and birth rate
alpha <- 0.1  ##strength of seasonal forcing
n <- 10  ## Number of patches
m <- 0.5  ##Connectivity matrix parameter
EC <- TRUE ##if true use equal coupling, if false use nearest neighbors

##Initial conditions
#Want ICs near EE for stability. Find EE values:
SEE <-1/R_0
IEE <-(1 - 1/R_0)*mu/gamma
REE <-1-SEE-IEE
#Sample from initials conditions within +/-10% of EE values
#convert from proportions to numbers and convert to integers
percent <- 0.1
population <- 3000
possibleS0 <- seq(as.integer(population*SEE*(1-percent)),
                  as.integer(population*SEE*(1+percent)),1)
possibleI0 <- seq(as.integer(population*IEE*(1-percent)),
                  as.integer(population*IEE*(1+percent)),1)
S0 <- sample(possibleS0, n,replace=TRUE)
I0 <- sample(possibleI0, n,replace=TRUE)
R0 <- population-S0-I0

 #Get simulation results
results <- SIRmeta.Gillespie(tmax,ic=c(S0,I0,R0),
                             parms=c(R_0,gamma,mu,alpha,m,EC))
times<-results[,1]
I <- results[,(n+2):(2*n+2)]

#Save data for later use and plot to pdf and png
write.table(results,file="GillECR017m0.2.pdf")
pdf(file="images/GillECR017m0.2.pdf")
copy <- dev.cur()
png(file="images/GillECR017m0.2.png")
dev.control("enable")

## draw box for plot
  plot(0,0,xlim=c(0,tmax),ylim=c(0,population*IEE*2),xlab="Time (years)",
       ylab="Prevalence (I)")
  #Plot all I_i using a differnt line colour and lty for each patch
for (i in 1:n) {
    lines(times, I[,i],col=i,lwd=0.2)
}
  #Make legend
  legend("topright",legend=1:n,fill="white",title="Patch",col=1:n,lty=1,cex=0.5)


dev.copy(which=copy)
dev.off
dev.off
```

This code runs very slowly for large populations. This is because the average waiting time between each transition is the inverse of the total rate - which is proportional to the population. Thus, the time steps become very small. For a 250,000 person per patch population it took 3 hours. This method for simulating the stochastic realization of the model is thus not practial.

4.2 Adaptive Tau-Leaping Algorthim

To simulate a stochastic process efficiently, an approximate method can be used. In the adaptive tau-leaping algorithm, instead of having a time step for every single transition, periods of time where transition rates are approximately constant are found by the algorithm and "leaped" over. For such a time period, τ, the number of transitions that occured is given by Possion distributions. The number of i transitions is given by a Possion random variable with parameter τa_i.

The `runadaptau` function contains a list of all possible transitions and `rateFct`, a function that returns the values of all transition rates given the time and current state. `runadaptau` passes these, along with the input initial conditions and parameters, to the `ssa.adaptivetau` function. It returns a list of times (starting at zero and ending near input `tmax` with non-uniform step size) and the values of the state at all these times.

```
  run.adaptau <-function(tmax,ic=c(S0,I0,R0),
                              params=c(R_0,gamma,mu,alpha,m,EC)){
  #Get the # of patches from the ics
  n <- length(ic)/3

  #TO DO: This should be changed so that it works with different n values
  #and is less ugly
  #A list of all possible transitions
  transitions <- list(c(S1 = +1), c(S1 = -1, I1= +1), c(I1 = -1, R1= +1), c(S1 = -1),
                       c(I1 = -1), c(R1 = -1),
                     c(S2 = +1), c(S2 = -1, I2= +1), c(I2 = -1, R2= +1), c(S2 = -1),
                     c(I2 = -1), c(R2 = -1),
                     c(S3 = +1), c(S3 = -1, I3= +1), c(I3 = -1, R3= +1), c(S3 = -1),
                     c(I3 = -1), c(R3 = -1),
                     c(S4 = +1), c(S4 = -1, I4= +1), c(I4 = -1, R4= +1), c(S4 = -1),
                     c(I4 = -1), c(R4 = -1),
                     c(S5 = +1), c(S5 = -1, I5= +1), c(I5 = -1, R5= +1), c(S5 = -1),
                     c(I5 = -1), c(R5 = -1),
                     c(S6 = +1), c(S6 = -1, I6= +1), c(I6 = -1, R6= +1), c(S6 = -1),
                     c(I6 = -1), c(R6 = -1),
                     c(S7 = +1), c(S7 = -1, I7= +1), c(I7 = -1, R7= +1), c(S7 = -1),
                     c(I7 = -1), c(R7 = -1),
                     c(S8 = +1), c(S8 = -1, I8= +1), c(I8 = -1, R8= +1), c(S8 = -1),
                   c(I8 = -1), c(R8 = -1),
                   c(S9 = +1), c(S9 = -1, I9= +1), c(I9 = -1, R9= +1), c(S9 = -1),
                   c(I9 = -1), c(R9 = -1),
                   c(S10 = +1), c(S10 = -1, I10= +1), c(I10 = -1, R10= +1), c(S10 = -1),
                   c(I10 = -1), c(R10 = -1))

#A function that outputs the transition rates given parameters, S,I,R, and t
rateFct <- function(x,params,time){
  #x should be of form: x=c(S,I,R)
  #parms should be of form: paarms=c(R_0,gamma,mu,alpha,m,EC)

  with(
     as.list(c(params)),
     {
  n <- length(x)/3
  S <- as.vector(x[1:n])
  I <- as.vector(x[(n+1):(2*n)])
  R <- as.vector(x[(2*n+1):(3*n)])

  #A matrix of all rates
  rates <- matrix(0,n,6)
  #the first column is all birth rates
  rates[,1] <- mu*(S+I+R)
  #thrid column are recovery rates
  rates[,3] <- gamma*I
  #last three columns are death rates from S and I
  rates[,4] <- mu*S
  rates[,5] <- mu*I
  rates[,6] <- mu*R
  #Second column are transmission rates
  #Equal Coupling
  if(EC==TRUE){
    M <- matrix(m/(n-1),n,n)+(1-m*(1+1/(n-1)))*diag(n)  ##connectivity matrix
    betam <- R_0*(gamma+mu)*(1+alpha*cos(2*pi*time))*M  ##beta matrix
  }else{
    #Nearest Neighbors
    M <- (1-m)*diag(n)
    M[row(M)%%n==(col(M)+1)%%n ] <- m/2
    M[row(M)%%n==(col(M)-1)%%n] <- m/2
    betam <- R_0*(gamma+mu)*(1+alpha*cos(2*pi*time))*M  ##beta matrix
  }
  rates[,2] <- S*colSums(t(betam[1:n,])*I)/(S+I+R)

  return(as.vector(t(rates)))
     }
  )
}

init.values <- c(S=S0,I=I0,R=R0)
#Get approximate simulation
simResults <- ssa.adaptivetau(init.values,transitions, rateFct, params, tf=tmax)
I <- simResults[,(12:21)]
```

```
    #Measure of coherence: the coefficient of variation/relative
  #standard deviation of all the patches at each time step
  coherence <- rep(0,length(simResults[,1]))
  for(tims in 1:length(simResults[,1])){
    coherence[tims]<- sd(I[tims,])/mean(I[tims,])
    #In case the mean is zero (and thus the coefficient of variation becomes nan or inf)
    if(is.nan(coherence[tims]) | is.infinite(coherence[tims])){
      coherence[tims]<- 0
    }
  }

  #Is the disease extinct (I pop below 20% of IEE for awhile) in any of the patches?
  #Store 1s for yes, 0 for nos
  localextinction<-c(1:n)
  for(i in 1:ncol(I)){
    if(mean(I[(nrow(I)-50):nrow(I),i])<= IEE*population*0.2){
      localextinction[i]<-1
    }else{
      localextinction[i]<-0
    }
  }
  localext<-mean(localextinction)

  #If the disease is extinct in all patches there is global extinction
  if(all(localextinction==1)){
      globalextinction<-1
    }else{
      globalextinction<-0
    }

  return(list("sim" = simResults, "coherence"= coherence,"localext"=localext,
              "globalext"=globalextinction))
  }
```

The following function,`plot.adaptau`, uses `runadaptau` to approximately simulate the meta-patch SIR model and plot the results.

```
plot.adaptau <- function(tmax,init.values,params,IEE,population){
  with(as.list(c(parms)), {
  n<-10
  run <- run.adaptau(tmax,init.values,params)

  simResults <- run$sim
  coherence <- run$coherence

  ##Saving the data and the print plot to pdf
  filename <- paste0("adaptau",if(EC==TRUE) "EC" else "NN", "R0", R_0, "m", m)
  if(file.exists(filename)){ #If the filename is already used
    filename <- paste0("adaptau",if(EC==TRUE) "EC" else "NN", "R0", R_0, "m", m, "(1)")
  }
  if(file.exists(filename)){ #If the filename(1) is already used
    filename <- paste0("images/adaptau",if(EC==TRUE) "EC" else "NN", "R0", R_0, "m", m,
                      "(2)")
  }
  write.table(run,file=paste0(filename,".csv"))
  pdf(file=paste0("images/",filename,".pdf"))

  ## draw box for plot
  #leave room on right margin for Coherence label
  par(mar = c(5,5,2,5))
  #Solutions should ocsillate around IEE so ylim should go to around twice
  #that so solns are in the middle of plot
  plot(0,0,xlim=c(0,tmax),ylim=c(0,population*IEE*4),xlab="Time (years)",
       ylab="Prevalence (I)")

  ##plot I_i for each patch
  j<-1:n
  for (i in 1:n) {
   if(i%%8==1) j[i] <- i+3 #indices so that black and solid lines are
   #skipped; saved for legend
   lines(simResults[,1], simResults[,11+i],col=j[i],lty=j[i],lwd=0.5)
   # use a different line colour and line type for each patch
  }

  #Currently unused code for plotting the det soln over this sim
   # draw.single.soln(n,ic=c(S=SEE,I=IEE,R=(1-SEE-IEE)),tmax=tmax,
   #          func=SIR.vector.field,
   #         parms=c(R_0,gamma,mu,alpha,population))

  #Plot coherence onto of pre-existing plot
  par(new = T)
  plot(simResults[,1],coherence,type='l',axes=F,xlab=NA, ylab=NA, ylim=c(0,1))
  #y axis and coherence label on right axis
  axis(side=4)
  mtext(side = 4, line = 3, 'Coherence')

  #Make legends
  legend("topright",legend=1:n,title="Patch",bg='white',col=j,lty=j,lwd=0.5,cex=0.8)
  legend("bottomright", legend="Relative Standard Deviation",bg='white',
       col=1,lty=1, cex=0.8)

  dev.off()
  })
}
```

Using `plot.adaptau` to simulate the model with different parameter values:

```
#Parameters
R_0 <- 17   ##basic reproduction rate (if the system had no forcing)
gamma <- 365/13  ##inverse of mean infectious period
mu <- 1/50  ##death and birth rate
alpha <- 0.1  ##strength of seasonal forcing
n <- 10  ## Number of patches
m <- 0.2  ##Connectivity matrix parameter
EC <- TRUE ##if true use equal coupling, if false use nearest neighbors
params<-c(R_0,gamma,mu,alpha,m,EC)

##Initial conditions
```

```
#Want ICs near EE for stability. Find EE values:
 SEE <-1/R_0
 IEE <-(1 - 1/R_0)*mu/gamma
 REE <-1-SEE-IEE
 #Sample ics from points +/-10% from the EE values
 percent <- 0.1
 population <- 5000000
 possibleS0 <- seq(as.integer(population*SEE*(1-percent)),
                  as.integer(population*SEE*(1+percent)),1)
possibleI0 <- seq(as.integer(population*IEE*(1-percent)),
                  as.integer(population*IEE*(1+percent)),1)
 S0 <- sample(possibleS0, n,replace=TRUE)
 I0 <- sample(possibleI0, n,replace=TRUE)
 R0 <- population-S0-I0
init.values <- c(S=S0,I=I0,R=R0)

tmax <- 15

plot.adaptau(tmax,init.values,params,IEE,population)
m<-0.01
plot.adaptau(tmax,init.values,params,IEE,population)
```

5 Coherence Dependence on Parameters

The dependence of a meta-patch solution becoming "coherent" on the parameters used in the model was explored. A system is considered coherent when the mean value of $c_v(t)$ from $t = 9$ years to $t = 10$ years is below a threshold value. We will refer to this value as the coherence threshold.

A list of $\mathcal{R}_0$ (the basic reproductive number) values from 1.1 to 30 was made, along with a list of m (the mixing parameter) values. For each set of $\mathcal{R}_0$ and m values, a solution or simulation of the model was run ten times each with different initial conditions (all within 10% of the Endemic Equilibrium values). It was stored whether or not each solution or simulation became coherent; a one was saved if it did and a zero was saved otherwise. By taking the mean of these values for all runs, the probability of becoming coherent for a given $\mathcal{R}_0$ and m value was found.

If parameter DetOrAdap is `TRUE` this is done using deterministic solutions to the model, created by the `run.soln` function. Otherwise, this is done using adaptive tau simulations in the `runadaptau` function.

```
coherenceGrid <- function(params=c(gamma,mu,alpha,EC),
                          R_0list,mlist,nsample,DetOrAdap,population=1){
  with(
      as.list(c(params)),
      {
  n <- 10  ## Number of patches
  ##Param for Initial conditions
   percent <- 0.1
  tmax <- 10
  gridsizem<-length(mlist)
  gridsizeR_0 <- length(R_0list)
  avgCoherence<- matrix(0, gridsizem,gridsizeR_0)
  avgLE<- matrix(0, gridsizem,gridsizeR_0)
  avgGE<- matrix(0, gridsizem,gridsizeR_0)

  index2 <- 1
```

```
for (R_0 in R_0list){
  index1 <- 1
   #Want ICs near EE for stability. Find EE values:
   #Note: They depend on R_0
   SEE <-1/R_0
   IEE <-(1 - 1/R_0)*mu/gamma

   if(!DetOrAdap){ #Only need these for adaptau sim
      possibleS0 <- seq(as.integer(population*SEE*(1-percent)),
                        as.integer(population*SEE*(1+percent)),1)
      possibleI0 <- seq(as.integer(population*IEE*(1-percent)),
                        as.integer(population*IEE*(1+percent)),1)
   }

    for (m in mlist){
      coherenceProb <- rep(0, nsample)
      coherenceProb <- rep(0, nsample)
      localextProb <- rep(0, nsample)
      globalextProb <- rep(0, nsample)
      for (sampler in 1:nsample){
       #Get det soln nsample times with different ICs
       #Take mean of coherence near end of soln
       #(since coherence can be somewhat oscilatory)

        if(DetOrAdap){
           S0 <- runif(n,SEE*(1-percent),SEE*(1+percent))
           I0 <- runif(n,IEE*(1-percent),IEE*(1+percent))
           R0 <- 1-S0-I0
           coherenceRun <- run.soln(n,ic=c(S=S0,I=I0,R=R0),tmax=tmax,
                                func=SIRmeta.vector.field,
                              parms=c(R_0,gamma,mu,alpha,m,EC))$coherence
        }else{
          S0 <- sample(possibleS0, n,replace=TRUE)
          I0 <- sample(possibleI0, n,replace=TRUE)
          R0 <- population-S0-I0
          run <- run.adaptau(tmax,init.values=c(S=S0,I=I0,R=R0),
                                    params=c(R_0,gamma,mu,alpha,m,EC))
          coherenceRun<-run$coherence
          localextProb[sampler] <- run$localext
          globalextProb[sampler] <- run$globalext
       }

      mean <- mean(coherenceRun[length(coherenceRun)-(100:0)])
      if(mean<threshold){
          coherenceProb[sampler] <- 1
      }else{
          coherenceProb[sampler] <- 0
       }
     }
   #Take mean of all coherence for all runs with these parameters
   avgCoherence[index1,index2]<- mean(coherenceProb)
   avgLE[index1,index2]<- mean(localextProb)
   avgGE[index1,index2]<- mean(globalextProb)
```

```
     index1<-index1+1
   }
   index2<-index2+1
  }
  #Save data
  filename<-paste0("grid",if(DetOrAdap) "det" else "adap",if(EC) "EC" else "NN",".csv")
  if(file.exists(filename)){ #If the filename is already used
    filename <-paste0("grid",if(DetOrAdap) "det" else "adap",if(EC) "EC" else "NN",
                      "(1).csv")
  }
  cohdata <- cbind.data.frame(avgCoherence,avgLE,avgGE)
  write.table(cohdata,file=filename)
  return(list("coherence"=avgCoherence, "LE"=avgLE,"GE"=avgGE))
       })
}
```

5.1 3D Plot with Deterministic Solution

A 3D plot showing the probability of coherence vs $\mathcal{R}_0$ and m is created in the function `coherence3DPlot`.

```
coherence3DPlot <- function(avgCoherence,params=c(gamma,mu,alpha,EC),R_0list,
                            mlist,nsample,DetOrAdap,population){
  with(
       as.list(c(params)),
       {

  #3D plot:
  filename<-paste0("images/",if(DetOrAdap) "det" else "adap","3D",if(EC) "EC" else "NN",
                   ".pdf")
   pdf(file=filename)
  #Making a colour gradient for the plot
  nbcol <- 100
  #You can change start and end to get different parts of the rainbow
  color <- rev(rainbow(nbcol, start = 3/6, end = 3.5/6))
  #The colour gradient will correspond to the 'height' of the plot
  zcol  <- cut(t(avgCoherence), nbcol)
 persp( R_0list,mlist, t(avgCoherence), theta = 30, phi = 30, expand = 0.5,
        col = color[zcol],ltheta = 120, shade = 0.75, ticktype = "detailed",
      xlab = "Basic Reproduction Number", ylab = "Mixing Parameter",
      zlab = "Probability of Coherence")
  dev.off()
       })
}
```

The following code calls `coherence3DPlot` to make a plot.

```
#Use the det soln if TRUE or adaptau sim if FALSE
DetOrAdap <- TRUE

 #The value at which things are considered coherent
 threshold <- 0.05
 #threshold <- 0.15   #for adaptau higher tol needed
 #Parameter values for grid
 gridsizeR_0 <- 100
 gridsizem <- 100
 nsample <- 10 #Number of runs per grid pt
 mlist <- seq(0.0001, 0.15, (0.15-0)/(gridsizem-1))
 R_0list <- seq(1.1,30,(30-1.1)/(gridsizeR_0-1))

 #Parameters
gamma <- 365/13  ##inverse of mean infectious period
mu <- 1/50  ##death and birth rate
alpha <- 0.1  ##strength of seasonal forcing
EC <- FALSE ##if true use equal coupling, if false use nearest neighbors

#Get data
data <- coherenceGrid(params=c(gamma=gamma,mu=mu,alpha=alpha,EC=EC),
                      R_0list,mlist,nsample,DetOrAdap,population)
avgCoherence <- data$avgCoherence
#Plot
coherence3DPlot(avgCoherence,params=c(gamma=gamma,mu=mu,alpha=alpha,EC=EC),
                R_0list,mlist,nsample,DetOrAdap,population=1)

data <- coherenceGrid(params=c(gamma=gamma,mu=mu,alpha=alpha,EC=TRUE),
                      R_0list,mlist,nsample,DetOrAdap,population)
avgCoherence <- data$avgCoherence
coherence3DPlot(avgCoherence,params=c(gamma=gamma,mu=mu,alpha=alpha,EC=TRUE),
                R_0list,mlist,nsample,DetOrAdap,population=1)
```

5.2 *Four Panel Plot with Adaptive Tau*

The function `fourPanPlot` makes a four panel plot showing the probability of coherence, the probability of local and global extinction, overtop of the bifurcation diagram of the single patch forced SIR model.

```
 fourPanPlot <- function(avgCoherence,avgLE,avgGE,
                         params=c(gamma,mu,alpha,EC),R_0list,mlist,
                         nsample,DetOrAdap,population){
   with(
       as.list(c(params)),
       {
   if(length(mlist)!=4){
     print("Warning: mlist should have length 4 for 4 panel plot
       \\ The first 4 elements of mlist will be used")
   }

#Save plot to pdf
filename<-paste0("images/",if(DetOrAdap) "det" else "adap","4pan",
                 if(EC) "EC" else "NN",".pdf")
   pdf(file=filename)

#Get bifur data for backgrounds
bfd <- read.table("bifurcation.dat",
                  col.names=c("time","S","I","R0","log10S","log10I"))
```

```r
#Get max probs for y lim
maxExt <- max(avgLE,avgGE)
maxCoh <- max(avgCoherence)

#To fit bifur data to probability plot
bfdI <- maxExt*(1.1/max(bfd$I))*bfd$I

#leave room on outside margins for labels on rigth, left and bottom
#par(mar = c(5,5,2,5))
par(oma=c(4,4,2,4))
#A four panel plot
par(mfrow=c(2,2))
par(mar = c(2, 3, 1, 2))
par(tcl = -0.25)
for(ind in 1:4){
   #Make an empty plot
   plot(0,0,xlim=c(1,30),
               ylim=c(0,maxExt*1.1),xlab=NA, ylab=NA,type='n')

   #Plot the bifur background
   points(bfd$R0,bfdI, pch=20, col="grey90", xpd=FALSE)
   #Plot LE, GE probs
   lines(R_0list, avgLE[ind,],col=4)
   lines(R_0list, avgGE[ind,],col=2)
   #Plot coherence prob overtop of current panel
   par(mfg = c(ceiling(ind/2),(ind-1)%%2 +1))
   plot(R_0list, avgCoherence[ind,],type='l',col=3,axes=F,xlab=NA,
        ylab=NA,ylim=c(0,maxCoh*1.1))
   axis(side=4)
   #Label m values on each plot
   mtext(paste0("m=",mlist[ind]), side = 3, line = -2, adj = 0.1,
         cex = 0.8, col = 1)
}

#Axes labels:
mtext("Basic Reproduction Number", side = 1, outer = TRUE, line = 1)

#Left axis label has multiple colors
mtext(expression("Probability of "*phantom("Local/ Global Extinction")),
      side = 2, outer = TRUE, line = 1, col = 1)
mtext(expression(phantom("Probability of ")*"Local"*phantom("/ Global Extinction")),
      side = 2, outer = TRUE, line = 1, col = 4)
mtext(expression(phantom("Probability of Local")*"/"*phantom(" Global Extinction")),
      side = 2, outer = TRUE, line = 1, col = 1)
mtext(expression(phantom("Probability of Local/")*" Global"*phantom(" Extinction")),
      side = 2, outer = TRUE, line = 1, col = 2)
mtext(expression(phantom("Probability of Local/ Global")*" Extinction"),
      side = 2, outer = TRUE, line = 1, col = 1)

mtext("Probability of Coherence", side = 4, outer = TRUE, line = 1, col = 3)

 dev.off()
       })
 }
```

Using `fourPanPlot`:

```
 #The value at which things are considered coherent
 threshold <- 0.15
 #Parameter values for grid
 gridsizeR_0 <- 200
 gridsizem <- 4
 nsample <- 20
 mlist <- c(0.2,0.45,0.7,0.9)
 R_0list <- seq(1.1,30,(30-1.1)/(gridsizeR_0-1))
 DetOrAdap <- FALSE

 #Parameters
gamma <- 365/13  ##inverse of mean infectious period
mu <- 1/50  ##death and birth rate
alpha <- 0.1  ##strength of seasonal forcing
n <- 10  ## Number of patches
EC <- FALSE ##if true use equal coupling, if false use nearest neighbors

population <- 250000

 #Get data
data <- coherenceGrid(params=c(gamma=gamma,mu=mu,alpha=alpha,EC=EC),
                      R_0list,mlist,nsample,DetOrAdap,population)
avgCoherence <- data$avgCoherence
avgLE <- data$avgLE
avgGE <- data$avgGE

   #Plot
fourPanPlot(avgCoherence,avgLE,avgGE,
            params=c(gamma=gamma,mu=mu,alpha=alpha,EC=EC),
                        R_0list,mlist,nsample,DetOrAdap,population)
```

References

[1] Yihui Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*, 2016. R package version 1.15.1.

[2] Olga Krylova and David J D Earn. Effects of the infectious period distribution on predicted transitions in childhood disease dynamics. *Journal of the Royal Society, Interface / the Royal Society*, 10(84):20130098, 2013.

[3] C. Connell McCluskey and David J D Earn. Attractivity of coherent manifolds in metapopulation models. *Journal of Mathematical Biology*, 62(4):509–541, 2011.

[4] N. Model Students: Dumont, M. Fong, C. LeDrew, and C. Weishaar. Mathematics 4mb3/6mb3 mathematical biology 2017 assignment 3, March 2017. An assignment previously submitted for credit for the course 4MB3/6MB3 Mathematical Biology offered at McMaster University.