Modelling spatial spread II - Supplementary

Andrea Parisi

Main contact for this notebook: Andrea Parisi (andrea.parisi@warwick.ac.uk)

Course context

Purpose and scope of the course

This material has been developed as part of the *GeMVi* project: *NIHR Global Health Research Group on the Application of Genomics and Modelling to the Control of Virus Pathogens* in East Africa and the University of Warwick.

In these workshops you will be introduced to some common techniques used in infectious disease modelling. The topics covered will include the implementation of deterministic and stochastic compartmental models, the use of maximum likelihood estimation to analyse super-spreading behaviour in novel disease outbreaks, modelling of contact patterns, and optimisation techniques for fitting epidemic models to real data.

Contributors

The contributors are all part of the GeMVi project:

- Prof. James Nokes (JNokes@kemri-wellcome.org)
- Prof. Matt Keeling (m.j.keeling@warwick.ac.uk)
- Dr. Joe Hilton (j.hilton@warwick.ac.uk)
- Dr. Rabia Aziza (rabia.aziza@warwick.ac.uk)
- Dr. Samuel Brand (s.brand@warwick.ac.uk)
- Dr. Andrea Parisi (andrea.parisi@warwick.ac.uk)

Helpful references for the course

- Anderson, R. M., & May, R. M. (1992). Infectious Diseases of Humans: Dynamics and Control.
- Bjørnstad, O. N. (2018). Epidemics, models and data using R. https://doi.org/10.1007/978-3-319-97487-3
- Diekmann, O., & Heesterbeek, J. A. P. (2000). Mathematical epidemiology of infectious diseases: model building, analysis and interpretation. 104: John Wiley and Sons.
- Keeling, Matt J., & Pejman Rohani. Modeling Infectious Diseases in Humans and Animals. Princeton University Press, 2008.
- Martcheva, M., 2015. An Introduction to Mathematical Epidemiology, Texts in Applied Mathematics. Springer US, Boston, MA. https://doi.org/10.1007/978-1-4899-7612-3
- Vynnycky, Emilia, & White, Richard G. An Introduction to Infectious Disease Modelling

Requirements

The following packages are required for this tutorial. Please have them installed before the start.

1. pracma - *Practical Numerical Math Routines* implements a number of mathematical functions, a couple of which are used in this tutorial.
2. deSolve - *Differential Equation solver* implements methods to integrate initial value differential equations
3. viridis - *Viridis colour palette* implements the viridis colour palettes, a set of colour palettes that provide perceptually uniform colours, are appropriate for color blindness and render properly in black and white. Particularly useful for visualising maps.
4. sf - *Simple Features* is a package useful to handle GIS (Geographic Information Systems) datasets.
5. lwgeom [optional] - This is an optional package and is not required for this tutorial. However, it may be needed to handle some vector maps.

The packages may be installed using (uncomment as needed and copy-paste into the console):

```
#install.packages("pracma", "deSolve", "viridis", "sf")
#install.packages("lwgeom")
```

Integration of model with tracking of movement

Here we want to implement a simple code to integrate the set of differential equations described below:

$$\frac{dN_{ii}(t)}{dt} = -\sum_{j\neq i}\bar{\rho}_{ji}N_{ii}(t) + \sum_{j\neq i}\tau N_{ji}(t)$$

$$\frac{dN_{ij}(t)}{dt} = \bar{\rho}_{ij}N_{jj}(t) - \tau N_{ij}(t)$$

$$\frac{dS_{ii}(t)}{dt} = \mu(N_{ii}(t) - S_{ii}(t)) - \beta_i S_{ii}(t)\frac{\sum_j I_{ij}(t)}{\sum_j N_{ij}(t)} - \sum_{j\neq i}\bar{\rho}_{ji}S_{ii}(t) + \sum_{j\neq i}\tau S_{ji}(t)$$

$$\frac{dS_{ij}(t)}{dt} = \mu(N_{ij}(t) - S_{ij}(t)) - \beta_i S_{ij}(t)\frac{\sum_j I_{ij}(t)}{\sum_j N_{ij}(t)} + \bar{\rho}_{ij}S_{jj}(t) - \tau S_{ij}(t)$$

$$\frac{dI_{ii}(t)}{dt} = \beta_i S_{ii}(t)\frac{\sum_j I_{ij}(t)}{\sum_j N_{ij}(t)} - (\gamma+\mu)I_{ii}(t) - \sum_{j\neq i}\bar{\rho}_{ji}I_{ii}(t) + \sum_{j\neq i}\tau I_{ji}(t)$$

$$\frac{dI_{ij}(t)}{dt} = \beta_i S_{ij}(t)\frac{\sum_j I_{ij}(t)}{\sum_j N_{ij}(t)} - (\gamma+\mu)I_{ij}(t) + \bar{\rho}_{ij}I_{jj}(t) - \tau I_{ij}(t)$$

In this set of equations, the parameter β_i is the person-to-person transmission rate within subpopulation i, whereas the *spatial contact matrix* $\bar{\rho}_{ij}$ represents the daily fraction of individuals from subpopulation j visiting subpopulation i, constrained to $\sum_i \bar{\rho}_{ij} = 1$. The rate τ is the rate of return of individuals to their home location.

To integrate the set of equation we notice that the last two elements of each equation represent a transport operator that has the same form for all three sets of unknowns $N_{ij}(t)$, $S_{ij}(t)$ and $I_{ij}(t)$. Identifying with X_{ij} one of the three sets, the operator has the form:

$$\Omega(\{X\})_{ij}(t) \begin{cases} -\sum_{j \neq i} \bar{\rho}_{ji} X_{ii}(t) + \sum_{j \neq i} \tau X_{ji}(t) & \text{for } i = j \\ \bar{\rho}_{ij} X_{jj}(t) - \tau X_{ij}(t) & \forall\ i \neq j \end{cases}$$

The fact that the form of this operator is the same for all sets of unknowns is due to the assumption that individuals move between counties in a way that is independent of their epidemiological status. If we were to consider restriction to infectives, for instance, the form of the transport operator would be dependent on that. Substituting the transport operator in the equations, the set of equation simplifies to:

$$\begin{aligned} \frac{dN_{ij}(t)}{dt} &= \Omega(\{N\})_{ij}(t) \\ \frac{dS_{ij}(t)}{dt} &= \mu(N_{ij}(t) - S_{ij}(t)) - \beta_i S_{ij}(t) \frac{\sum_j I_{ij}(t)}{\sum_j N_{ij}(t)} + \Omega(\{S\})_{ij}(t) \\ \frac{dI_{ij}(t)}{dt} &= \beta_i S_{ij}(t) \frac{\sum_j I_{ij}(t)}{\sum_j N_{ij}(t)} - (\gamma + \mu) I_{ij}(t) + \Omega(\{I\})_{ij}(t) \end{aligned}$$

where we do not need to distinguish the diagonal elements, because such distinction is now internal to the $\Omega(\{X\})$ operator. These sets of equations thus operate of variables of the form X_{ij} that may be interpreted as matrices, where the sum over the first index returns the number of Xs residing location j, and the sum over the second index returns the number of individual at time t in location i (both resident and visitors). We can now turn the equations into code.

We first load the set of required libraries:

```
library(pracma)
library(deSolve)
```

```
##
## Attaching package: 'deSolve'
```

```
## The following object is masked from 'package:pracma':
##
##     rk4
```

```
library(viridis)
```

```
## Loading required package: viridisLite
```

As usual, we use the ode function from the deSolve library to integrate the metapopulation model. The function takes 5 arguments: 1) an initial condition, 2) a set of times on which we want the integrated model to be evaluated, 3) the ode model to be integrated, 4) a set of parameters that may be used in the evaluation of (3), and finally 5) an integration method.

```
res <- deSolve::ode( xstart, times, sir.model, params, method=deSolve::rk4 )
```

The first step is to write the transport operator for a generic set of unknowns x. We assume x to be a matrix. We also note that $\sum_{j\neq i}\bar{\rho}_{ji}$ is a sum over column i of the transpose of $\bar{\rho}_{ij}$ which can be read equivalently as a sum over row i of $\bar{\rho}_{ij}$. Similarly, $\sum_{j\neq i}N_{ji}(t)$ can be read as the sum over row i of $N_{ij}(t)$. Thus, by avoding to calculate the transpose of those two matrices, the transport operator takes the form:

```
transportOp <- function( x, rho, tau, nn )  {
    mm <- matrix(0, nrow=nn, ncol=nn)
    for (ii in 1:nn)  {
        for (jj in 1:nn)  {
            if (ii == jj)  {
                mm[ii,jj] = -(sum(rho[ii,])-rho[ii,ii])*x[ii,ii] + tau*(sum(x[ii,])-x[ii
            } else {
                mm[ii,jj] =  rho[ii,jj] * x[jj,jj] - tau*x[ii,jj]
            }
        }
    }
    return(mm)
}
```

The simulation model, thus has a similar structure to the one we introduced previously. Here we unpack matrices rather than vectors, and population sizes are part of the unknowns. The calculation could be done for the full matrix, but by going through by rows, we can calculate the sum appearing in the transmission term easily. We thus have:

```
sir.model <- function( t, x, params )  {
    patches <- params[[1]]   # This is the number of sub-populations
    mu <- params[[2]]        # Birth/death rate
    gamma <- params[[3]]     # Recovery rate
    beta <- params[[4]]      # Common transmission rate
    rho <- params[[5]]       # Contact matrix
    tau <- params[[6]]       # Rate of return

    # Extract the susceptibles and infectives
    NN      <- matrix(x[1:(patches*patches)], nrow=patches, ncol=patches)
    pop.sus <- matrix(x[(patches*patches+1):(2*patches*patches)], nrow=patches, ncol=pat
```

```
    pop.inf <- matrix(x[(2*patches*patches+1):(3*patches*patches)], nrow=patches, ncol=p
    dn <- matrix( 0, nrow = patches, ncol = patches )
    ds <- matrix( 0, nrow = patches, ncol = patches )
    di <- matrix( 0, nrow = patches, ncol = patches )

    # Find rhs of (1) using vector calculus
    omegaN = transportOp( NN,      rho, tau, patches )
    omegaS = transportOp( pop.sus, rho, tau, patches )
    omegaI = transportOp( pop.inf, rho, tau, patches )
    for (ii in 1:patches)  {
        dn[ii,] <- omegaN[ii,]
        cc <- sum(pop.inf[ii,])/sum(NN[ii,])
        ds[ii,] <- mu*(NN[ii,]-pop.sus[ii,])-beta[ii]*pop.sus[ii,]*cc + omegaS[ii,]
        di[ii,] <- beta[ii]*pop.sus[ii,]*cc - (gamma+mu)*pop.inf[ii,] + omegaI[ii,]
    }

    res <- c(dn,ds,di)
    return(list(res))
}
```

Now we can build an instance of the metapopulation model: we initalize the model by placing individuals in their home locations: in terms of the unknowns, we are initializing individuals on the diagonal terms of the N_{ij} matrices, and we will do similarly for the S_{ij} and I_{ij}. We assume again 16 patches of 10,000 individuals each. Thus:

```
patches <- 16
NN <- 10000*diag(patches)
```

Let us use a set of parameters typical of flu. For the rate of return, we identify it with the inverse of the time spent at destination: assuming for instance to be dealing with daily commuting data, this would be the inverse of the average time spent at work:

```
mu <- 1.0/(70*365)
gamma <- 1./2.2
beta  <- rep(1.66, patches)
tau   <- (7*24)/40
```

For this model we use once again a random matrix, but it is easy to use real commuting data. For a change, let us assume a smaller interaction between counties

```
rho <- matrix(2*runif(patches*patches), nrow=patches, ncol=patches) / patches
# Diagonal elements represent individuals not moving
# and are set by applying the constraint on rho
for (jj in 1:nrow(rho))  {
    rho[jj,jj] <- 0
}
```

```
# Make sure that sum_i rho_ij = 1
for (jj in 1:ncol(rho))  {
    if (sum(rho[,jj]) > 1)  {
        rho[,jj] <- rho[,jj] / sum(rho[,jj])
    }
    rho[jj,jj] <- 1-sum(rho[,jj])
}
```

We can now group all parameters in the list params:

```
params <- list(patches, mu, gamma, beta, rho, tau)
```

We are almost done: we need a sequence of times and an initial condition (ten infectives in patch 1).

```
times <- seq( from = 0, to = 40, by = 0.05)
start.I0 <- matrix(0, nrow=patches, ncol=patches)
start.I0[1,1] <- 10
start.S0 <- NN-start.I0
xstart <- c(NN, start.S0, start.I0 )
```

We are ready now to integrate the model

```
res <- deSolve::ode( xstart, times, sir.model, params, method=deSolve::rk4 )
```

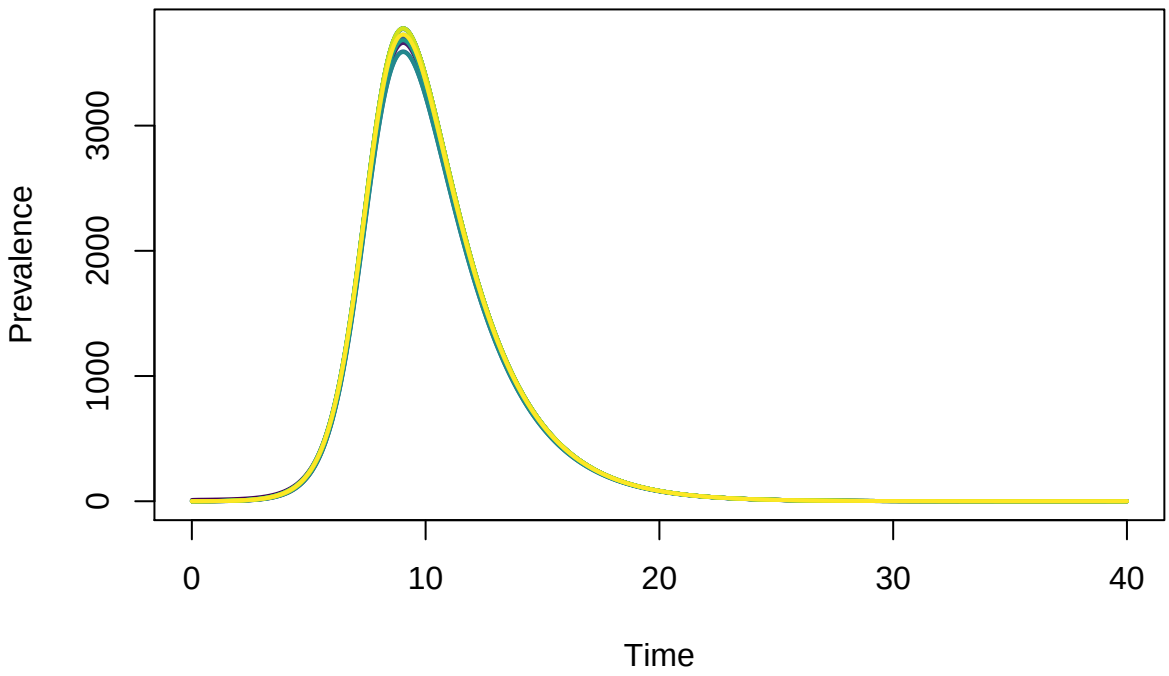
Visualization of results: here we focus on infectives in each patch. We introduce the function showOutput2 that plots the number of infectives *pertaining to location* j as a function of time in all patches. We will use it here and later on.

```
output <- data.frame( res )

showOutput2 <- function(out, nregions)  {
    x.min <- min(out$time)
    x.max <- max(out$time)
    yy <- out[,(2*nregions*nregions+2):ncol(out)]  # Only take infectives

    yout <- apply(yy, 1, function(x) {apply(matrix(x, nregions, nregions),2,sum)})
    y.min <- 0
    y.max <- max(yout)
    plot( c(x.min, x.max), c(y.min, y.max), type="n", col=3, lwd=2,
          ylim=c(0,y.max), xlab="Time", ylab="Prevalence" )
    for (ii in 1:nregions)  {
        lines( out$time, yout[ii,], lwd=2, col=viridis(nregions)[ii] )
    }
}

showOutput2(output, patches)
```


Prevalence
0
1000
2000
3000
0
10
20
30
40
Time