Journal of Statistical Software

June 2021, Volume 98, Issue 10. doi: 10.18637/jss.v098.i10

Continuous Time Individual-Level Models of Infectious Disease: Package EpiILMCT

Waleed Almutiry
Qassim University

Vineetha Warriyar K V
University of Calgary

Rob Deardon
University of Calgary

Abstract

This paper describes the R package **EpiILMCT**, which allows users to study the spread of infectious disease using continuous time individual level models (ILMs). The package provides tools for simulation from continuous time ILMs that are based on either spatial demographic, contact network, or a combination of both of them, and for the graphical summarization of epidemics. Model fitting is carried out within a Bayesian Markov Chain Monte Carlo framework. The continuous time ILMs can be implemented within either susceptible-infected-removed ($\mathcal{SIR}$) or susceptible-infected-notified-removed ($\mathcal{SINR}$) compartmental frameworks. As infectious disease data is often partially observed, data uncertainties in the form of missing infection times – and in some situations missing removal times – are accounted for using data augmentation techniques. The package is illustrated using both simulated and an experimental data set on the spread of the tomato spotted wilt virus disease.

Keywords: **EpiILMCT**, infectious disease, individual level modeling, spatial models, contact networks, R.

1. Introduction

Innovative mathematical and mechanistic approaches to the modeling of infectious diseases are continuing to emerge in the literature. These can be used to understand the spread of disease through a population – whether homogeneous or heterogeneous – and enable researchers to construct predictive models to develop control strategies to disrupt disease transmission. For example, Deardon *et al.* (2010) introduced a class of discrete time individual-level models (ILMs) which incorporate population heterogeneities by modeling the transmission of disease given various individual-level risk factors. The general framework of ILMs have already been successfully applied to a broad range of epidemic data, e.g., the 2001 UK foot-and-mouth outbreak (Deardon *et al.* 2010; Deeth and Deardon 2016; Malik, Deardon, and Kwong

2016), tomato spotted wilt virus (TSWV) disease (Pokharel and Deardon 2014, 2016), the spread of 1-18-4 genotype of the porcine reproductive and respiratory syndrome in Ontario swine herds (Kwong, Poljak, Deardon, and Dewey 2013), and influenza transmission within households in Hong Kong during 2008 to 2009 and 2009 to 2010 (Malik, Deardon, Kwong, and Cowling 2014). Equivalent continuous time ILMs which capture the complex interactions between susceptible and infected individuals through spatial and contact networks can also be considered. The inference and fitting of such models is generally considered within a Bayesian framework using Markov chain Monte Carlo (MCMC).

However, infectious disease epidemiologists have previously found it difficult to apply these individual-level models to real life problems. This is due to a dearth of readily available software products. The applicability of the aforesaid continuous time ILMs is implemented in an R (R Core Team 2021) package, **EpiILMCT** (Almutiry, Deardon, and Warriyar K V 2021) and is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=EpiILMCT`. In this article, we describe the package **EpiILMCT** which allows users to simulate and fit epidemic data using distance- and/or network-based models (Bifolchi, Deardon, and Feng 2013; Deardon *et al.* 2010; Jewell, Kypraios, Neal, and Roberts 2009), and can also incorporate risk factors associated with both susceptible and infectious individuals. **EpiILMCT** also uses data augmentation techniques to carry out inference when the infection and/or removal times are unknown or censored, as is usually the case. To the extent of our knowledge, this feature is not available in any existing R packages that permit epidemic data analysis and modeling. Tools for the graphical summarization of epidemic data sets and outcomes are also provided. The statistical inferences made in **EpiILMCT** are set in a Bayesian framework and are carried out using MCMC. The main aim here is to provide a fast implementation of continuous time ILMs under different epidemic modeling frameworks. Because of the computationally intensive nature of MCMC for such models, we have coded functions, including the MCMC algorithm, in Fortran to speed up computation.

There are several R packages that permit a range of different modeling tools that allow for fitting spatial-temporal epidemic data. For example, the packages **splancs** (Rowlingson and Diggle 2021), and **lgcp** (Taylor, Davies, Rowlingson, and Diggle 2013, 2015) provides methods for analyzing epidemic data as spatial and space-time point patterns. Also, the package **surveillance** (Meyer, Held, and Höhle 2017) implements a spatio-temporal point process model for epidemic data through the function `twinstim`. Other packages fit a range of autocorrelation regression spatio-temporal models, e.g., **CARBayesST** (Lee, Rushworth, and Napier 2018), **spdep** (Bivand, Hauke, and Kossowski 2013; Bivand and Piras 2015), and **spTimer** (Bakar and Sahu 2020, 2015). Further packages are mentioned in the CRAN Task View "Handling and Analyzing Spatio-Temporal Data" (Pebesma 2021). The R Epidemics Consortium (2018) provides further useful resources for disease outbreak analysis related R software packages.

However, in each case, the functionality (e.g., models available) of the packages above is quite different to that of **EpiILMCT**. The models of the **EpiILMCT** package are "mechanistic" in that they attempt to more directly model the mechanisms of transmission between individuals. Specifically, they take into account the spatial interactions between individuals with differing disease status (e.g., susceptible, infected, notified, removed) at continuous time points of the epidemic process. Those spatial interactions between susceptible and infectious individuals are incorporated as distance-based effects on the infectivity rate of individuals through an

infection kernel function (power-law or Cauchy). The infectivity rates can also depend upon various susceptibility and transmissibility covariates at the individual level. Additionally, and of key importance, none of the aforementioned packages account for uncertainty in the event times using the Bayesian data augmentation MCMC method.

There are several R packages that provide for the visualization, simulation and modeling the spread of epidemics through networks. The package **EpiModel** (Jenness, Goodreau, and Morris 2018) allows epidemic simulation from mathematical models of infectious disease through stochastic contact networks based on exponential-family random graph models (ERGMs). Some packages assume observed contact network or networks when fitting the specified model; for example, **ergm** (Handcock, Hunter, Butts, Goodreau, Krivitsky, and Morris 2021; Hunter, Handcock, Butts, Goodreau, and Morris 2008), **Bergm** (Caimo and Friel 2014), and **hergm** (Schweinberger, Handcock, and Luna 2021). Those packages implement Bayesian methods for fitting exponential-family transmission network models to observed contact network data.

A recently developed package, **epinet** (Groendyke and Welch 2018), allows users to infer transmission networks from time-series epidemic data by modeling the contact network using a generalization of the ERGMs. This package makes use of time-series epidemic data as the input assuming unknown contact network in their functionality, and producing parameter estimates of the epidemic model as well as the contact and transmission networks. The transmission model can contain various covariates that captures important features (summary statistics) of the contact network as well as epidemic transmission.

However, once again these packages have different approaches to that implemented in **EpiILMCT**. We focus here on incorporating a contact network as a covariate in the implemented ILMs in **EpiILMCT**. The response in the ILMs is the event (e.g., infection) time, rather than the transmission network (the transmission network can be inferred later via posterior predictive simulation, of course, but we do not address this here). This is different to **epinet**, for example, which models the transmission network directly. The **EpiILMCT** package allows for any user pre-specified contact networks, including various special cases such as spatial or random unweighted (binary) (un)directed contact networks or weighted contact networks.

As both spatio-temporal and contact network-based mechanisms can be key to understanding the dynamics of infectious disease spread, the ILMs in **EpiILMCT** allow for the incorporation of both contact network and distance-based effects jointly in the infectivity rate of individuals. None of the aforementioned packages have this feature in their functionalities.

The use of individual level data in more mechanistic epidemic models has been implemented in only a few other R packages. The most established of these is **surveillance** (Salmon, Schumacher, and Höhle 2016; Meyer *et al.* 2017), a package for temporal and spatio-temporal disease modeling. It provides tools for outbreak detection in routinely collected surveillance data, as well as a range of models for infectious disease data. The most closely related model in **surveillance** to those of **EpiILMCT** is the additive endemic-epidemic multivariate temporal point process model. These models are implemented in the `twinSIR` function for modeling the susceptible-infectious-recovered ($\mathcal{SIR}$) event history of a fixed population in continuous time using individual level data. However, not only is the underlying model framework different to that considered in the **EpiILMCT** package, but the `twinSIR` function does not allow for uncertainty in event times to be taken into account via data augmentation techniques. The function does not allow for only the epidemic terms of the model to be considered, as can be done in **EpiILMCT**; both endemic (e.g., seasonal) and epidemic terms must be included

in the analysis. In addition, the distance kernel used in the epidemic part of the `twinSIR` function is represented by a linear combination of non-negative basis functions and is thus different from the distance kernels used in the **EpiILMCT** package.

The **EpiILM** package (Warriyar K V, Almutiry, and Deardon 2020) that has recently been made available in R, provides similar utility to **EpiILMCT**, but for discrete-time ILMs. The models it contains provide options to include susceptible individual covariate information, as well as a choice to describe population heterogeneity. However, the package is limited to discrete-time distance-based or network-based infection kernels and requires known event histories (i.e., there is no data augmentation feature).

As stated previously, inference for the models of **EpiILMCT** is carried out in a Bayesian MCMC framework. Although there are packages available in R to implement MCMC algorithms such as **MCMCpack** (Martin, Quinn, and Park 2011) and **adaptMCMC** (Scheidegger 2021), all are based on the random walk Metropolis-Hastings (M-H) algorithm. The data augmented MCMC algorithm used in the **EpiILMCT** package to fit various models uses random walk and independence sampler (within Gibbs) steps within a M-H algorithm. The independence sampler algorithm in our package appears to be essential for updating the missing data efficiently (event times and infectious periods), given that the authors have not found it possible to achieve well-mixing MCMC chains if purely random walk M-H algorithms are used (even if tuned adaptedly).

Our main purpose of developing this package is to make the use of continuous time ILMs available to epidemiologists and statisticians, through R, one of the most commonly used statistical software packages. Overall, **EpiILMCT** offers greatly increased flexibility for analyzing complex disease data. The remainder of this paper is laid out as follows. In the next section, we describe the general continuous individual-level model implemented in **EpiILMCT**. We also discuss the different infection kernel functions implemented in the package. Sections 3 and 4 discuss the functions contained within the package and the underlying Bayesian inference, respectively. Section 5 illustrates the application of **EpiILMCT** to simulated and real data, while Section 6 concludes the paper with a short summary of the software package and its implications.

2. Model

The **EpiILMCT** package allows for the implementation of continuous time equivalents, and extensions, of the discrete-time individual-level models (ILMs) of Deardon *et al.* (2010). The compartmental frameworks considered are the susceptible-infectious-removed ($\mathcal{SIR}$) and susceptible-infectious-notified-removed ($\mathcal{SINR}$). In both frameworks, each individual is assumed to be in one of these states at any point in time, $t \in \mathbb{R}^+$. In the $\mathcal{SIR}$ framework, infected individuals transition between states, susceptible to infectious and from infectious to removed. Individuals are assumed to be in the susceptible ($\mathcal{S}$) state until they become infected at which point they become immediately infectious ($\mathcal{I}$), then being able to transmit the disease for the duration of their infectious periods before entering the removed ($\mathcal{R}$) state. In the $\mathcal{SINR}$ framework, infectious individuals are assumed to move from the infectious state ($\mathcal{I}$) to a notified ($\mathcal{N}$) state. The latter represents a state in which individuals have been identified as having the disease, and may be subjected to various restrictions (e.g., government-imposed movement constraints in the 2001 UK FMD outbreak). The $\mathcal{N}$-state infectivity rate is often

assumed to be lower than that of $\mathcal{I}$-state. As infectious individuals enter the $\mathcal{R}$-state, they are removed from the infectious population (e.g., because of recovery and acquired immunity, death or quarantine) and from thereon play no role in transmitting the disease.

A full epidemic history consists of all transition event times for all individuals, and defines the state of all n individuals at each point in time. For example for the $\mathcal{SINR}$ framework, $\mathcal{S}(t), \mathcal{I}(t), \mathcal{N}(t)$ and $\mathcal{R}(t)$ at time t for $t \in [0, t_{obs}]$ are defined by all infection, notification and removal times. Here, t_{obs} is the maximum removal time i.e., the time that the last notified individual enters the removed state. We assume that each susceptible individual j at time t has an infectivity rate[1] with a given infectious individual i:

$$\lambda_{ij}(t) = \begin{cases} \lambda_{ij}^{-}(t) & i \in \mathcal{I}(t), j \in \mathcal{S}(t) \\ \lambda_{ij}^{+}(t) & i \in \mathcal{N}(t), j \in \mathcal{S}(t) \end{cases},$$

where

$$\lambda_{ij}^{-}(t) = \Omega_S(j)\Omega_T(i)\kappa(i,j)$$
$$\lambda_{ij}^{+}(t) = \gamma\Omega_S(j)\Omega_T(i)\kappa(i,j), \quad \gamma > 0,$$

where $\Omega_S(j)$ and $\Omega_T(i)$ are the susceptibility and transmissibility functions, respectively. They are defined as:

$$\Omega_S(j) = \mathbf{S}\mathbf{X}_{\cdot j}^{\phi} \quad \text{and} \quad \Omega_T(i) = \mathbf{T}\mathbf{Z}_{\cdot i}^{\xi}, \quad \phi, \xi > 0,$$

where $\mathbf{S}$ and $\mathbf{T}$ are the (coefficient) parameter vectors of the susceptibility and transmissibility covariates with sizes equal to the number of susceptibility (p_S) and transmissibility (p_T) covariates, respectively; $\mathbf{X}_{\cdot j}^{\phi}$ and $\mathbf{Z}_{\cdot i}^{\xi}$ are the jth and ith columns of the susceptibility and transmissibility risk factor matrices $\mathbf{X}^{\phi} \in \mathbb{R}_{p_S \times n}^{+}$ and $\mathbf{Z}^{\xi} \in \mathbb{R}_{p_T \times n}^{+}$, respectively; and ϕ and ξ are vectors of the power parameters of the susceptibility and transmissibility functions with sizes equal to p_S and p_T, respectively. Note that, $\mathbf{X}^{\phi}$ and $\mathbf{Z}^{\xi}$ are constrained to be positive. These power parameters allow for non-linearity between the susceptibility and transmissibility risk factors and the infection rate (Deardon *et al.* 2010). The notification effect parameter γ is used to measure the risk of infection after notification that can be reduced or increased depending on the disease type. For example, the transmissibility has been observed to increase after symptoms in SARS (Pitzer, Leung, and Lipsitch 2007), whereas, it can be lower for the 2001 UK FMD (Jewell *et al.* 2009). The latter stated this effect parameter in their general model as a control measure parameter that accounts only the reduction in the risk of infection. In the case of $\gamma = 1$, notification has no effect on infectivity.

So, the total rate of infectivity of each susceptible individual j at time t is given by:

$$\lambda_j(t) = \left[\sum_{i \in \mathcal{N}^{-}(t)} \lambda_{ij}^{-}(t) + \sum_{i \in \mathcal{N}^{+}(t)} \lambda_{ij}^{+}(t) \right] + \epsilon(j,t), \tag{1}$$

where $\mathcal{N}^{-}(t)$ is the set of infectious individuals at time t who have been infected but have not reached the notified state; and $\mathcal{N}^{+}(t)$ is the corresponding set for notified individuals (Jewell *et al.* 2009).

[1]Note that, technically the infectivity rates are conditioned upon the past epidemic history, so might be written $\lambda_{ij}(t|H_t)$ where H_t is the epidemic history up to time t. However, for the sake of brevity and simplicity we have dropped the conditioning from the notation.

Model	Kernel type	Kernel function
Distance-based ILMs	Power-law	$\kappa(i,j) = d_{ij}^{-\beta}, \quad \beta > 0$
	Cauchy	$\kappa(i,j) = \frac{\beta}{d_{ij}^2+\beta^2}, \quad \beta > 0$
Network-based ILMs	Unweighted, undirected	$\kappa(i,j) = c_{ij}\ , \quad c_{ij} = 0$ or 1
	Weighted	$\kappa(i,j) = w_{ij}, \quad w_{ij} \in [0,\infty)$
Combined distance and network-based ILMs	Power-law	$\kappa(i,j) = d_{ij}^{-\beta_1} + \beta_2 c_{ij}$ $\kappa(i,j) = d_{ij}^{-\beta_1} + \beta_2 w_{ij}$
	Cauchy	$\kappa(i,j) = \frac{\beta_1}{(d_{ij}^2+\beta_1^2)} + \beta_2 c_{ij}$ $\kappa(i,j) = \frac{\beta_1}{(d_{ij}^2+\beta_1^2)} + \beta_2 w_{ij}, \quad \beta_1, \beta_2 > 0$

Table 1: Types of kernel functions that are applied in the **EpiILMCT** package for fitting continuous time ILMs.

The nomenclature is the same for the $\mathcal{SIR}$ framework, but without the $\mathcal{N}(t)$ state, there is not need to compartmentalize infectious individuals into pre- and post-notification sets. Therefore, the total rate of infectivity of each susceptible individual j at time t is given by:

$$\lambda_j(t) = \left[\sum_{i \in \mathcal{I}(t)} \lambda_{ij}^{-}(t) \right] + \epsilon(j,t), \tag{2}$$

where $\mathcal{I}(t)$ is the set of infectious individuals at time t (i.e., they have been infected, but not yet removed).

The infectivity rate $\lambda_j(t)$ also contains a spark function that is denoted by $\epsilon(j,t)$ which allows for random infections otherwise unexplained by the model. This might represent, for example, the infection of a susceptible individual from a source outside of the observed population. In this model, we fix the spark term $\epsilon(j,t)$ such that $\epsilon(j,t) = \epsilon$; $\epsilon \geq 0$.

The infection kernel $\kappa(i,j)$ represents shared risk factors between pairs of infected and susceptible individuals. In the **EpiILMCT** package we consider three kernel types: distance-based, network-based, and combined distance and network-based. Two sub-types of distance-based kernel are also considered: Cauchy and power-law. The infection kernel functions are given in Table 1. In the distance-based ILMs, the kernel function is based on the distances d_{ij} between individuals generally, but not always, spatial Euclidean distance. In the network-based ILMs, the kernel function is based on the connections between individuals in a contact network that are represented by binary connections $c_{ij} = 0$ or 1, or weighted connections $w_{ij} \in [0,\infty)$. In the combined ILMs the kernel consists of a linear function of both.

2.1. Likelihood function

We label the m infected individuals $i = 1, 2, \ldots, m$ with corresponding infection (I_i) and removal (R_i) times such that $I_1 \leq I_2 \leq \cdots \leq I_m$. The $N - m$ individuals who remain uninfected after t_{obs} are labeled $i = m+1, m+2, \ldots, N$ with $I_i = R_i = \infty$. We then denote infection and removal time vectors for the population as $\mathbf{I} = \{I_1, \ldots, I_m\}$ and $\mathbf{R} = \{R_1, \ldots, R_m\}$, respectively. We assume that infectious periods follow a gamma distribution with a fixed shape δ_a and rate δ_b, $\delta = (\delta_a, \delta_b)$ (Jewell *et al.* 2009). The likelihood function can

be divided into two independent components: the infectious and the removed components. As we assumed earlier that each susceptible individual j has a total infectivity rate $\lambda_j(I_j)$ (their total specific infectious pressure) at the time of being infected (I_j) from infectious individuals $i \in \mathcal{I}(I_j)$, the infectious component under the $\mathcal{SIR}$ continuous time ILMs can be written as:

$$L_1 = \prod_{j=2}^{m} \left(\epsilon + \sum_{i:I_i<I_j\leq R_i} \lambda_{ij}^{-}(I_j) \right) \times \exp\left\{ - \int_{I_1}^{t_{obs}} \left(\sum_{i\in\mathcal{S}(u)} \epsilon + \sum_{i\in\mathcal{I}(u)} \sum_{j\in\mathcal{S}(u)} \lambda_{ij}^{-}(u - I_i) \right) du \right\}$$

where the product term represents the total specific infectious pressure that each infected individual receives from infectious individuals at the time of being infected, and the exponential integral represents the total person-to-person infectious pressure during the course of the epidemic.

The removed component then contains the contribution of the infectious periods to the likelihood function via their densities. As the infectious period of an infected individual i ($\mathcal{D}_i = R_i - I_i$) is independent of others, the removed component is simply:

$$L_2 = \prod_{i=1}^{m} f(\mathcal{D}_i; \delta)$$

The likelihood function of the general $\mathcal{SIR}$ continuous time ILMs can then be formed by combining the infectious and removal parts given as follows:

$$\begin{aligned} L(\boldsymbol{I}, \boldsymbol{R}|\boldsymbol{\theta}) &= L_1 \times L_2 \\ &= \prod_{j=2}^{m} \left(\epsilon + \sum_{i:I_i<I_j\leq R_i} \lambda_{ij}^{-}(I_j) \right) \exp\left\{ - \sum_{i=1}^{m} \left(\sum_{j=1}^{N} ((R_i \wedge I_j) - (I_i \wedge I_j)) \lambda_{ij}^{-}(I_j) \right) \right\} \\ &\times \exp\left(-\epsilon \sum_{i=1}^{N} [(t_{obs} \wedge I_i) - I_1] \right) \prod_{i=1}^{m} f(\mathcal{D}_i; \delta), \quad \delta > 0, \end{aligned}$$

where the wedge symbol $\wedge$ denotes the minimum operator; $\boldsymbol{\theta}$ is the vector of unknown parameters; $f(\bullet; \delta)$ indicates the density of the infectious period distribution; and $\mathcal{D}_i$ is the infectious period of infected individual i defined as $\mathcal{D}_i = R_i - I_i$. The integral in Equation 3, which represents the total person-to-person infectious pressure through the course of the epidemic, can be written as the double sum in the lower equation (Britton and O'Neill 2002; Jewell *et al.* 2009). The integral is transformed by discretizing it into a sum over the successive events of the epidemic and is substituted by the double sum. The likelihood function of the general $\mathcal{SINR}$ continuous time ILMs can be formed in a very similar manner (see Appendix A).

3. Contents of the EpiILMCT package

The **EpiILMCT** package can be used to simulate and graphically summarize epidemics, and, for a given model, carry out Bayesian inference and calculate the log-likelihood. Most of the main package functions are written in Fortran 95 (called from within the R wrapper), since they are computationally intensive tasks. The functions contained in the package are reviewed in Table 2.

Function	Usage
`contactnet`	Generates undirected unweighted (binary) contact network matrices from spatial (`powerlaw`, or `Cauchy`), or `random`, network models.
`plot.contactnet`	Provides plot of a contact network of class '`contactnet`'.
`datagen`	Generates epidemics from distance/network-based individual level models.
`as.epidat`	Generates objects of class '`datagen`' that contain the individual event history of an epidemic along with other individual level information.
`plot.datagen`	Provides different plots summarizing an epidemic of class '`datagen`'.
`epictmcmc`	Runs a Bayesian data augmented MCMC algorithm for fitting specified models ($\mathcal{SIR}$ or $\mathcal{SINR}$).
`print.epictmcmc`	Prints the contents of '`epictmcmc`' object to the console.
`summary.epictmcmc`	Summary method for '`epictmcmc`' objects.
`plot.epictmcmc`	Plots the output of '`epictmcmc`' object.
`loglikelihoodepiILM`	Calculates the log likelihood for a given compartmental framework and kernel type of the continuous time ILMs.

Table 2: Description of functions and their usages in the **EpiILMCT** package.

3.1. Contact network

Various types of contact network can be considered. First, we consider unweighted (binary) contact networks which can be directed or undirected. In an undirected unweighted contact network, each pair of individuals share the same symmetric connection such that $c_{ij} = c_{ji}$ for $i \neq j$; $i, j = 1, \ldots, N$; and each network is defined by $\binom{N}{2}$ elements where $c_{ij} = 1$ if a connection exists between individuals i and j, and 0 otherwise. In a directed unweighted contact network, it is not necessary for individuals to share the same symmetrical relationship so that $c_{ij} \neq c_{ji}$ for $i \neq j$; $i, j = 1, \ldots, N$. This leads to a non-symmetric contact network matrix. Weighted contact networks can also be considered in the **EpiILMCT** package in which the connections between individuals are not described as present or absent but are weighted according to their strength. These too can be directed or undirected.

A function (`contactnet`) is included to generate undirected unweighted contact networks. It can simulate both spatial networks where connections are more likely to occur between individuals closer in space ("spatial contact networks"), as well as random contact networks. The function `contactnet` has three available options (`"powerlaw"`, `"Cauchy"`, and `"random"`) for the network model, where the first two options simulate spatial contact networks in which the probability of connections between individuals are based on required XY coordinate input.

The inclusion of the two options `"powerlaw"` and `"Cauchy"` in the argument `type` is to allow the user to choose between two commonly assumed spatial forms to describe the underlying population. For example, the power-law network model is taken from Bifolchi *et al.* (2013) who use this network to test how well purely spatial power-law ILMs can approximate disease

spread through networks. The Cauchy model was used by Jewell *et al.* (2009) to model the 2001 UK foot-and-mouth outbreak in Cumbria; they found this kernel the most appropriate for predicting transmission of those tested.

We now describe the three model options in detail. First, in the power-law contact network model of Bifolchi *et al.* (2013) the probability of a connection between individual i and j is given by:

$$p(c_{ij} = 1) = 1 - e^{-\nu(d_{ij}^{-\beta})}, \quad \nu, \beta > 0,$$

where d_{ij} is the Euclidean distance between individuals i and j; β is the spatial parameter; and ν is the scale parameter.

Under the Cauchy contact network model, as used in Jewell *et al.* (2009), the probability of a connection between individual i and j is given by:

$$p(c_{ij} = 1) = 1 - e^{-\beta/(d_{ij}^2+\beta^2)}, \quad \beta > 0,$$

where d_{ij} is the Euclidean distance between individuals i and j; and β is the spatial parameter.

Finally, under the random contact network model, the probability of a connection is simply generated from a Bernoulli distribution with probability equal to β.

Let us now consider some examples. To create the above undirected unweighted contact networks, the function requires the network model to be specified (`"powerlaw"`, `"Cauchy"`, or `"random"`) via the `type` argument. If `"powerlaw"` or `"Cauchy"` are selected, the XY coordinates of individuals (`location`) have to be specified through the argument `location`. The function `contactnet` produces a list which includes the contact network matrix in a class, '`contactnet`'.

To obtain a plot of the contact network, we introduce an S3 `plot` method for '`contactnet`' objects, which uses as its input an object of the class '`contactnet`'. The `plot` method for '`contactnet`' objects uses code internal to **EpiILMCT** for the layout when plotting power-law or Cauchy network models, but depends on the package **igraph** (Csardi and Nepusz 2006) when plotting random network model.

The following code generates the three types of contact networks for a population of 50 individuals, with a uniformly distributed spatial layout for the spatial network models.

```
R> library("EpiILMCT")
R> set.seed(12345)
R> loc <- cbind(runif(50, 0, 10), runif(50, 0, 10))
R> net1 <- contactnet(type = "powerlaw", location = loc, beta = 1.5,
+    nu = 0.5)
R> net2 <- contactnet(type = "Cauchy",  location = loc, beta = 0.5)
R> net3 <- contactnet(type = "random", num.id = 50, beta = 0.08)
R> par(mfrow = c(2, 2))
R> plot(net1)
R> plot(net2)
R> plot(net3, xlab = "(random)", vertex.color = "red", vertex.size = 20,
+    edge.color = "black", vertex.label.cex = 0.5,
+    vertex.label.color = "black")
```

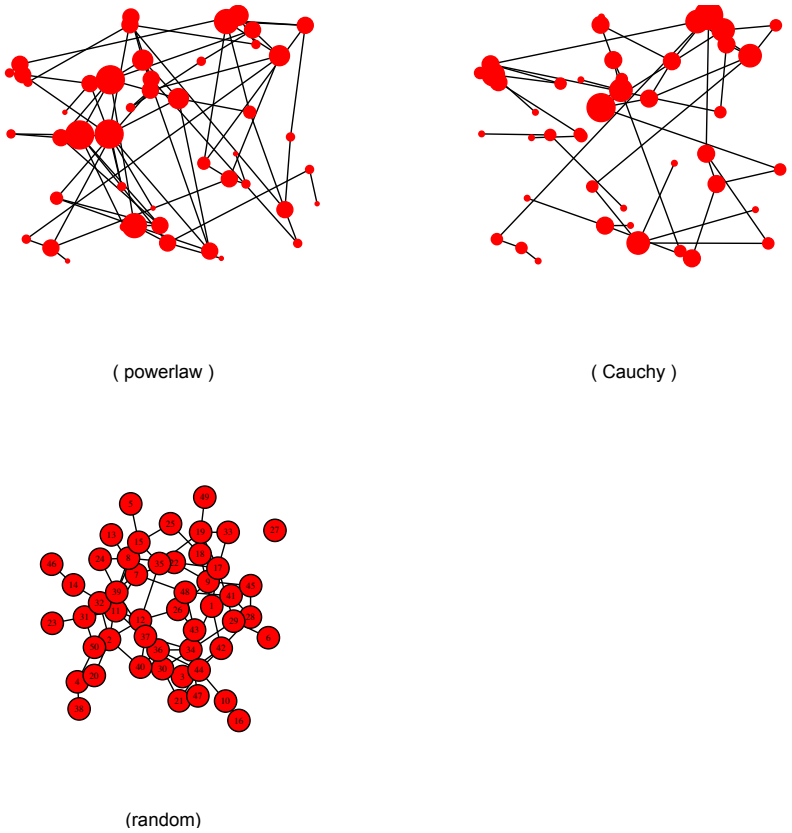

Figure 1: Examples of the three undirected unweighted (binary) contact network models generated for the same population. Red dots represent nodes with size corresponding to their degree (number of edges).

A realization of the three networks for a given population is shown in Figure 1. Note the underlying spatial layout of the nodes is the same for both spatial network models.

3.2. Epidemic simulation

The function `datagen` allows the user to generate epidemics from the continuous time ILMs under the $\mathcal{SIR}$ or $\mathcal{SINR}$ compartmental frameworks. Which framework is to be used is specified through the `type` argument. Each infected individual in a simulated epidemic has an infection life history defined by their time of infection and the length of time spent in the infectious state. We assume the conditional intensity functions stay constant between events, such that the time to the next infection, given that the last infection occurred at time t, follows $W_j \sim \text{Exp}(\lambda_j(t))$. Here, W_j represents the "waiting time" for susceptible individual j becoming infected.

Under the $\mathcal{SIR}$ framework, and using the chosen distribution of the infectious period, an epidemic is simulated starting with a randomly chosen initial infected individual k at time $I_1 = 0$, or with initial infected individual(s) specified via the argument `initialepi`. This argument requires a vector or matrix containing the id number(s), removal time(s), infectious

period(s) and infection time(s) of the infected individual(s). At time I_s, the waiting time until infection for susceptible individual j is then drawn from $W_j \sim \text{Exp}(\lambda_j(I_s))$.

The individual with the minimum W is taken as the next infected individual and assigned an infection time $I_{s+1} = I_s + \min(W)$; an infection period $\mathcal{D}_j$ (generated from $f(\mathcal{D}_j; \delta)$); and a removal time $R_{s+1} = I_{s+1} + \mathcal{D}_j$. The process is repeated until no infectives remain in the population or $I_{s+1} > t_{max}$, where t_{max} is the time at which the epidemic simulation is set to end. t_{max} can be then specified via the argument `tmax`.

Under the $\mathcal{SINR}$ framework, each infected individual is considered to have an incubation period comprising the time from infection to notification, and a delay period comprising the time from notification to removal. Together the incubation and delay periods constitute the infectious period. An epidemic is simulated in the same manner described above for the $\mathcal{SIR}$ framework, except that the infection period is replaced by incubation and delay periods $\mathcal{D}^{(inc)}$ and $\mathcal{D}^{(delay)}$ (generated from $f(\mathcal{D}_j^{(inc)}; \delta^{(inc)})$ and $f(\mathcal{D}_j^{(delay)}; \delta^{(delay)})$, respectively); and notification and removal times are assigned as $N_{s+1} = I_{s+1} + \mathcal{D}_j^{(inc)}$ and $R_{s+1} = N_{s+1} + \mathcal{D}_j^{(delay)}$, respectively.

In this function, the infectious, incubation and delay periods are assumed to follow either exponential or gamma distributions. These distributions can be specified through the `delta` argument. Under the $\mathcal{SIR}$ framework, `delta` is a vector containing the shape and rate parameters of a gamma distribution, whereas under the $\mathcal{SINR}$ framework it is a 2×2 matrix where each row represents the parameters of the incubation and delay period distributions. Note that – as is often done – an exponential distribution can be assigned to any of these distributions by setting the shape parameter equal to one.

The epidemic data structure output of the `datagen` function is used throughout the **EpiILMCT** package. Under an $\mathcal{SIR}$ ILM, it returns a matrix with four columns representing: the id numbers of the individuals, removal times, infectious periods, and infection times. Under an $\mathcal{SINR}$ ILM, it returns a matrix with six columns: the id numbers of the individuals, removal times, delay periods, notification times, incubation periods, and infection times. Uninfected individuals are assigned infinity values (`Inf`) for both their removal and infection times. Epidemic data from other modeling packages can be extracted and modified to be used in **EpiILMCT**. For example, we show how this can be done using the individual level models from the **surveillance** package in Appendix B.

The choice of the kernel function $\kappa(i, j)$ is specified using the `kerneltype` argument. This takes one of three options: `"distance"` for distance-based, `"network"` for network-based, or `"both"` for distance and network-based. The appropriate kernel matrix must also be provided via the `kernelmatrix` argument. If `"distance"` is chosen as the `kerneltype`, the user must choose a spatial kernel (`"powerlaw"` or `"Cauchy"`) through the `distancekernel` argument. The distance matrix can be obtained from XY coordinate data using the `dist` function from the **stats** package (R Core Team 2021). Otherwise the distance matrix can be specified by the user. Other arguments in the `datagen` function require the data and coefficient parameters for the susceptibility and transmissibility risk factors as explained in Section 2.

We define an object of class '`datagen`' to take a list of values needed for the use of other functions, such as, the `plot` method for '`datagen`' objects and `epictmcmc`. This list contains: `type`, `kerneltype`, `epidat` (event times), `location` (XY coordinates of individuals), and `network` (contact network matrix). In the case of setting the `kerneltype` to `"distance"`, a `NULL` value will be assigned to the `network` option. The package has also a separate

function `as.epidat` that generates an object of class '`datagen`' for a given epidemic data set (Appendix B contains a brief example of using this function).

The package also contains an S3 `plot` method for '`datagen`' objects, which illustrates disease spread through the epidemic timeline. This function can be used for either distance-based or network-based ILMs. The first input of this function has to be of class '`datagen`'. If the `plottype` argument is set to `"history"`, the function produces epidemic curves of infection and removal times. Example plots are shown in Figure 3. Conversely, setting this argument to `"propagation"` produces plots of the epidemic propagation over time. With the latter option, exactly which plots are output varies by kernel. With the network kernel, the function plots all the connections between individuals and overlays these with the epidemic pathway direction over time. This path direction consists of directed edges from all infectious individuals connected to a given newly infected individual i with infection time I_i (one per plot). Thus, this produces directed networks showing possible pathways of the disease propagation. With the distance kernel, the function plots the spatial epidemic dispersion over time. It shows the changes in the individual status that related to the chosen compartmental framework. To avoid displaying too many plots, the `time.index` argument allows user to obtain propagation plots at specific infection time points rather than at every infection time.

4. Bayesian inference

Prior distributions of the model parameters are selected from one of three options: gamma, positive half normal or uniform distribution. Then, Metropolis-Hastings MCMC is performed to estimate the joint posterior of the model parameters and latent variables (the latter if various event times are assumed unknown). This is achieved using the function `epictmcmc`. The parameters of the susceptibility and transmissibility functions, infection kernel and spark term (collectively denoted $\boldsymbol{\theta}$) are updated using the random-walk proposals. The user is required to tune the proposal variances to achieve good mixing properties. Thus, the user must provide a vector of initial values, a prior distribution (`"gamma"`, `"uniform"`, or `"halfnormal"`), the prior parameters, and the variance of the normal proposal distribution for each parameter as shown in Figure 2. In case of running multiple MCMC chains, the user should provide a vector of initial values of the model parameters. Note that, setting the variance of the normal proposal distribution to zero fixes a parameter at its initial value. This option allows the user to fix such a parameter in the model while updating others (i.e., conditioning on the parameters).

Using the `datatype` argument, the `epictmcmc` function allows for three scenarios in terms of event time uncertainty: `"known epidemic"` can be used to model a fully observed epidemic with known infection and removal times; `"known removal"` can be used to model a partially observed epidemic where the infection times are unknown; and `"unknown removal"` can be used to model a partially observed epidemic where removal and infection times are unknown. The latter option is only available for the $\mathcal{SINR}$ continuous time ILMs where notification times are assumed correctly known. When the `datatype` argument is set to `"known epidemic"`, the infectious periods are fixed by default.

When infection times are unknown, the rate(s) of the infectious, incubation and/or delay period distributions are assigned gamma prior distributions with shape a and rate b. Thus, the rate parameters have conditional distributions with a standard form following the gamma

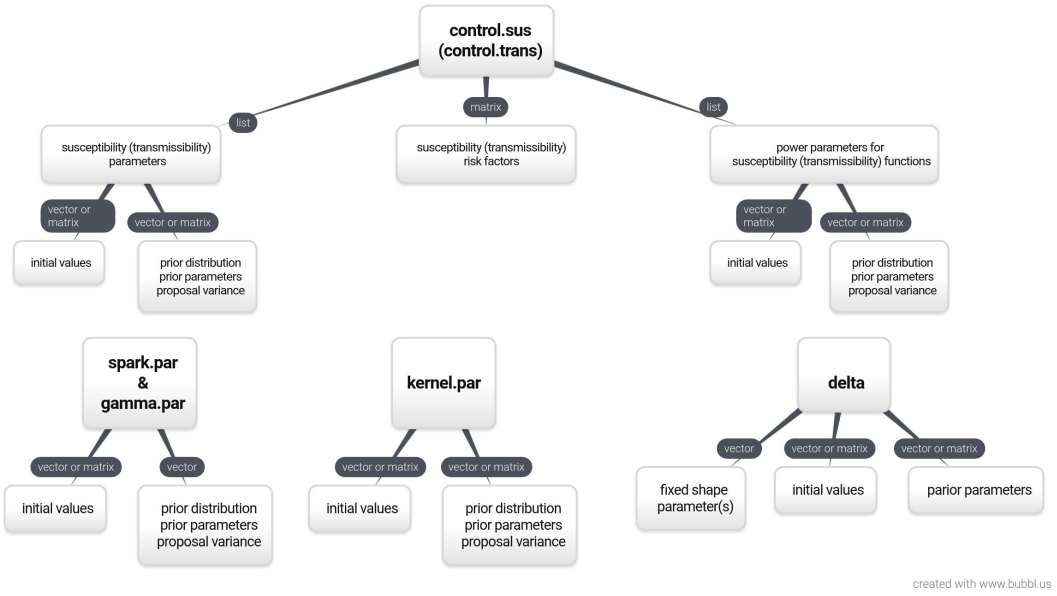

Figure 2: A diagram of the input structure for the arguments `control.sus`, `control.trans`, `kernel.par`, `spark.par`, `gamma.par` and `delta` in the function `epictmcmc`.

distribution. For the $\mathcal{SIR}$ continuous time ILMs, this is as follows:

$$\delta|\boldsymbol{\theta}, \boldsymbol{I}, \boldsymbol{R} \sim \Gamma(m + a_\delta, M + b_\delta),$$

where δ is the rate of the infectious period distribution; $M = \sum_{i=1}^{m} (R_i - I_i)$; and a_δ and b_δ are the prior parameters of the infectious period rate. For the $\mathcal{SINR}$ continuous time ILMs, the distribution of the incubation rate and delay parameters are as follows:

$$\delta^{(inc)}|\boldsymbol{\theta}, \boldsymbol{I}, \boldsymbol{N}, \boldsymbol{R} \sim \Gamma(m + a_{\delta^{(inc)}}, M_{inc} + b_{\delta^{(inc)}}),$$

where $\delta^{(inc)}$ is the rate of the incubation period distribution; $M_{inc} = \sum_{i=1}^{m} (N_i - I_i)$; and $a_{\delta^{(inc)}}$ and $b_{\delta^{(inc)}}$ are the prior parameters of incubation period rate; and

$$\delta^{(delay)}|\boldsymbol{\theta}, \boldsymbol{I}, \boldsymbol{N}, \boldsymbol{R} \sim \Gamma(m + a_{\delta^{(delay)}}, M_{delay} + b_{\delta^{(delay)}}),$$

where $\delta^{(delay)}$ is the rate of the delay period distribution; $M_{delay} = \sum_{i=1}^{m} (R_i - N_i)$; and $a_{\delta^{(delay)}}$ and $b_{\delta^{(delay)}}$ are the prior parameters of delay period rate.

A Gibbs update (i.e., sampling from the conditional posterior distribution) is used for the infectious period rate (for the $\mathcal{SIR}$ continuous time ILMs) or the incubation and/or delay period rates (for the $\mathcal{SINR}$ continuous time ILMs). The required information for each period distribution are entered via the `delta` argument. We assume each period type follows a gamma distribution with fixed shape and unknown rate. Thus, to update the rate parameter of each period we specify `delta`, a list containing a vector of the fixed shape value(s), a vector (matrix) of the initial values of the rate(s), and a vector (matrix) for the parameters of the prior distribution of the rate parameter(s). In the case of incubation and delay periods being estimated, the input of the initial values is a 2×`nchains` matrix, and the prior parameters is a 2×2 matrix where each row contains the required information for each period rate.

An independence sampler is then used to update the infection times/infectious periods (for the $\mathcal{SIR}$ continuous time ILMs), or the infection times/incubation periods and/or the removal times/delay periods (for the $\mathcal{SINR}$ continuous time ILMs). For the $\mathcal{SIR}$ continuous time ILMs, the ith infection time I_i is updated by generating an infectious period $\mathcal{D}_i^*$ from a gamma proposal distribution such that $\mathcal{D}_i^* \sim \Gamma(a, b)$. Then, the new infection time is the difference between the observed removal time and the new infectious period of the ith individual. The same procedure is used for updating the missing event times, infectious periods and corresponding parameters for the $\mathcal{SINR}$ continuous time ILMs. The parameter values of the gamma proposal distribution could be provided through the `periodproposal` argument. If they are not provided, the parameters of the gamma proposal distribution are then based on the fixed shape and updated rate values from the argument `delta`. Computationally, it may be more efficient to apply a block update for the periods and event times. This can be implemented using the `blockupdate` argument, which requires that the user specifies m (assuming removal and infection times are known for the first m individuals), and the size of each block.

The `epictmcmc` function allows for sampling from multiple MCMC chains. This is done by providing the number of chains to be run via the option `nchains`. Additionally, multiple chains can be run in parallel by setting `parallel = TRUE`. This implies the use of the `parLapply` function from the **parallel** package (R Core Team 2021). The number of cores to be used is set to the minimum of the number of chains and the available cores on the user's computer. Note that, if `parallel` is set to `FALSE` and `nchains` is greater than one, multiple MCMC chains are run sequentially. When `parallel` is set to `TRUE`, the `clusterSetRNGStream` function from the **parallel** package (R Core Team 2021) is used to distribute the setting seed value by the `set.seed` function to each core to reproduce the same results, otherwise each core sets its seed value from the current seed of the master process.

The output of this function is an object of class '`epictmcmc`'. There are S3 methods: `print`, `summary` and `plot` that depend on the **coda** package (Plummer, Best, Cowles, and Vines 2006). The latter function has a `plottype` argument to specify which samples need to be plotted. This argument has three options: `"parameter"` to produce trace plots of the posterior distributions of the model parameters, and `"inf.times"` (`"rem.times"`) to produce plots of the average posterior and 95% CI of the unobserved infection (removal) times when `datatype` set to `"known removal"` (`"unknown removal"`). The S3 `plot` method for '`epictmcmc`' objects has the same options as the method for '`mcmc`' objects in the **coda** package, for example, `start`, `thin`, and `density`.

The class '`epictmcmc`' contains the MCMC samples of the model parameters and the missing information (if `datatype` is not set to `"known epidemic"`) as an `mcmc` matrix, and other useful information to be used in other functions, such as the above S3 methods. So standard summary methods from **coda**, such as `summary` and `plot` methods for '`mcmc`' objects, can be employed using these MCMC samples as inputs.

Posterior predictive checks of the fitted model can be performed using the `datagen` function. This requires that the user supplies the model parameter values with a combined sample of the MCMC model parameter outputs. If desired, the simulation can be constrained to the first m infected individuals and their event times. This can be achieved by appending this information to the `initialepi` option.

5. Examples

5.1. Simulated network-based epidemic

In this section, we illustrate the **EpiILMCT** package by fitting a simple $\mathcal{SIR}$ network-based continuous time ILM to a simulated epidemic. We consider an isolated population of 50 individuals distributed uniformly in an area of 10×10 units. We also consider a binary susceptibility covariate z which can be thought as being, say, an individual's treatment or vaccination status. Thus, the infectivity rate given in Equation 2 becomes:

$$\lambda_j(t) = (\alpha_0 + \alpha_1 z_j) \sum_{i \in \mathcal{I}(t)} c_{ij}, \qquad \alpha_0, \alpha_1 > 0,$$

where the susceptibility function $\Omega_S(j) = \alpha_0 + \alpha_1 z_j$; there are no transmissibility covariates $\Omega_T(i) = 1$; and $\epsilon = 0$. First, let us simulate the XY coordinates of individuals and the binary covariate z as follows:

```
R> set.seed(91938)
R> loc <- cbind(runif(50, 0, 10), runif(50, 0, 10))
R> cov <- cbind(rep(1, 50), rbinom(50, 1, 0.5))
```

To simulate the epidemic, we generate a contact network using the `contactnet` function. Here, we use the power-law contact network model with $\beta = 1.8$ and $\nu = 1$, as illustrated in the following code:

```
R> net <- contactnet(type = "powerlaw", location = loc, beta = 1.8, nu = 1)
```

Figure 4 shows the contact network (grey lines). The epidemic is then generated using the `datagen` function. Here, the epidemic is initialized with a randomly chosen infectious individual; then generated by providing the function with the contact network matrix, the susceptibility covariate and the following parameter values: $\alpha_0 = 0.08$, $\alpha_1 = 0.5$, and $\mathcal{D}_i \sim \Gamma(4, \delta = 2)$. This is coded as follows:

```
R> epi <- datagen(type = "SIR", kerneltype = "network",
+    kernelmatrix = net, suspar = c(0.08, 0.5), delta = c(4, 2),
+    suscov = cov)
```

The object `epi` is stored in the data file `NetworkData` as a class '`datagen`', along with the susceptibility covariate (`cov`), available in the **EpiILMCT** package.

```
R> data("NetworkData", package = "EpiILMCT")
R> class(NetworkData[[1]])

[1] "datagen"

R> names(NetworkData[[1]])

[1] "type"       "kerneltype" "epidat"     "location"   "network"
```

```
R> head(NetworkData[[1]]$epidat)

     id.individual rem.time inf.period  inf.time
[1,]            50 1.526078  1.5260782 0.0000000
[2,]            16 2.612491  1.9933013 0.6191893
[3,]             5 2.394094  1.6567882 0.7373061
[4,]            45 3.169602  2.2370141 0.9325876
[5,]            44 1.805656  0.5661341 1.2395222
[6,]            19 1.737867  0.4576725 1.2801945
```

To illustrate the propagation of the epidemic, we set the argument `plottype` to `"propagation"`. To limit the number of plots, we assign the `time.index` option to be a vector containing time points for plots to be generated as shown in the following code:

```
R> plot(NetworkData[[1]], plottype = "propagation",
+    time.index = seq_len(6))
```

We can also produce density plots of the infection and removal times, and a plot of the infectious periods, by specifying the argument `plottype` to `"history"` as shown in the following code:

```
R> plot(NetworkData[[1]], plottype = "history")
```

Figure 3 shows the densities of the infection and removal times, and the infectious periods; while Figure 4 shows the epidemic propagation plot.

To illustrate fitting continuous time ILMs to data, we analyze the epidemic using the function `epictmcmc`. This is done under two observation scenarios: `"known epidemic"` and `"known removal"`. For the former analysis, we assign $\Gamma(1, 0.1)$ gamma prior distributions to the model parameters α_0 and α_1 and use normal MCMC proposals with variances equal to 0.5 and 1, respectively. As we have two susceptibility parameters, the argument `control.sus` is then a list that contains: 1) a list of a vector of initial values of α_0 and α_1, and a 2×4 matrix in which each row represents the required information for updating each parameter; and 2) a 50×2 matrix of the covariates representing the unity intercept and the binary covariate z. Now, we run the MCMC using the `epictmcmc` function for sampling a single chain of 150,000 iterations using the following code:

```
R> set.seed(91938)
R> suscov <- list(NULL)
R> suscov[[1]] <- list(c(0.01, 0.1), matrix(c("gamma", "gamma",
+    1, 1, 0.1, 0.1, 0.5, 1), ncol = 4, nrow = 2))
R> suscov[[2]] <- NetworkData[[2]]
R> mcmc1 <- epictmcmc(object = NetworkData[[1]],
+    datatype = "known epidemic", nsim = 150000, control.sus = suscov)
```

The estimates of the model parameters can be obtained through the S3 `summary` method for '`epictmcmc`' objects. The posterior means and 95% credible intervals of these parameters can be obtained via the following command:

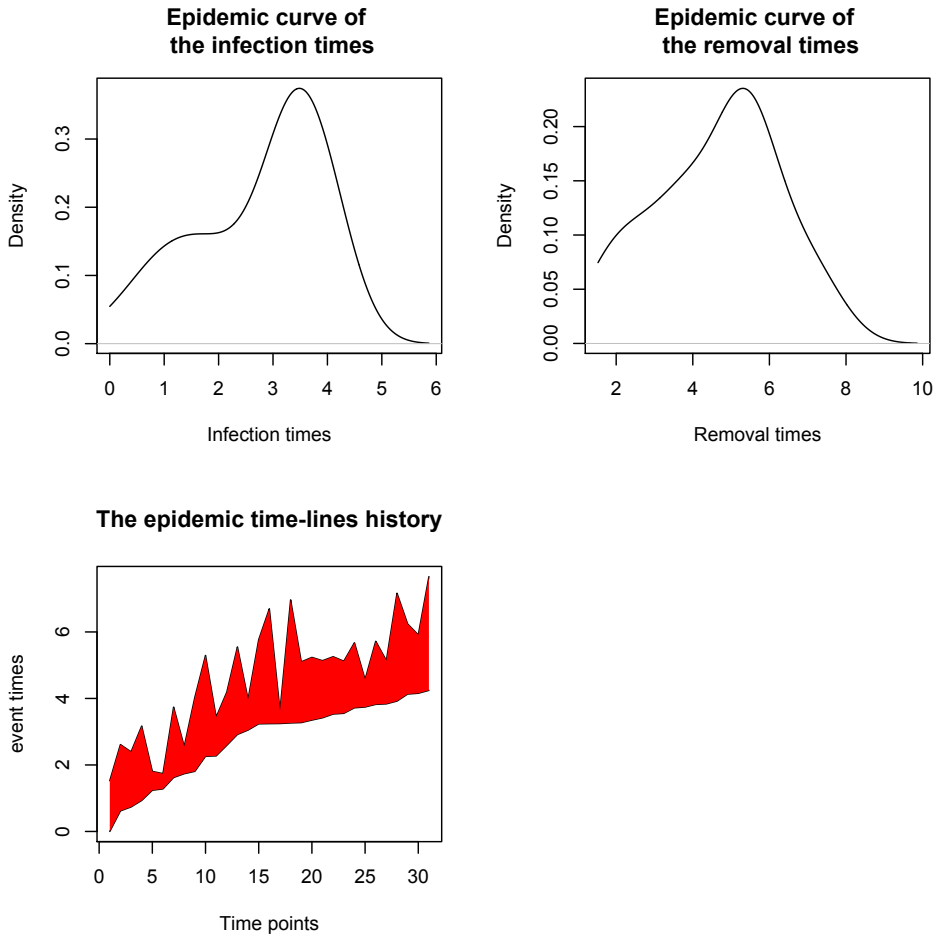

Figure 3: The epidemic curves of the infection and removal times for the epidemic that was generated using the simple network-based continuous time ILM. The red shaded area in the third plot represents the infectious periods.

```
R> summary(mcmc1, start = 10000, thin = 10)

**********************************************************
Model: SIR network-based continuous-time ILM
Method: Markov chain Monte Carlo (MCMC)
Data assumption: fully observed epidemic
number.chains : 1 chains
number.iteration : 140000 iterations
number.parameter : 2 parameters

**********************************************************
 1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
                Mean        SD    Naive SE Time-series SE
Alpha_s[1] 0.0850579 0.0268504 0.000226919    0.000298624
Alpha_s[2] 0.5082012 0.1290994 0.001091050    0.001179665
```

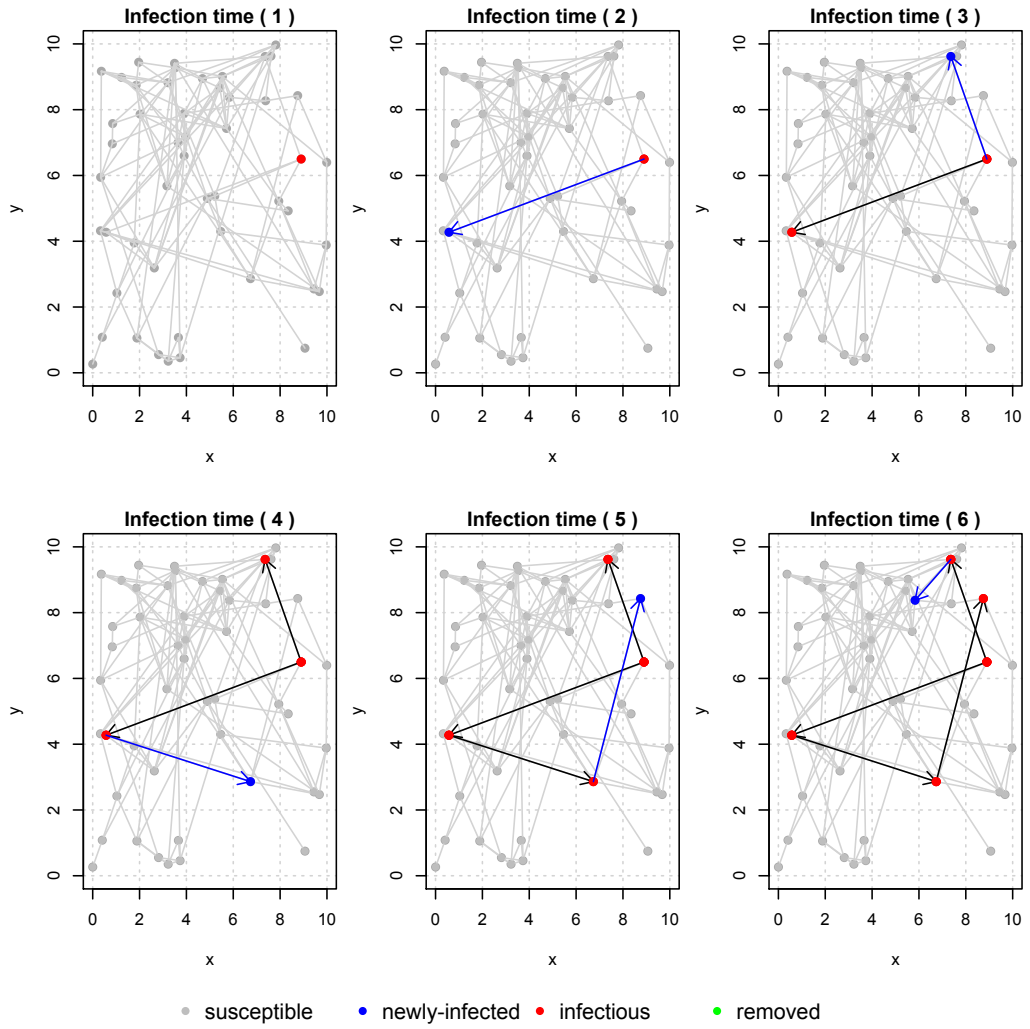

Figure 4: The directed pathway network of the generated epidemic over its time-line using the simple network-based ILMs.

```
 2. Quantiles for each variable:
                2.5%       25%       50%       75%     97.5%
Alpha_s[1] 0.0417253 0.0655198 0.0824374 0.101868 0.143758
Alpha_s[2] 0.2856068 0.4163682 0.4982712 0.587971 0.789077
 3. Empirical mean, standard deviation, and quantiles for the log likelihood,
          Mean             SD       Naive SE Time-series SE
   -55.8040938      1.0188095      0.0086102      0.0104071

    2.5%      25%      50%      75%    97.5%
-58.5949 -56.1864 -55.4943 -55.0810 -54.8118
 4. acceptance.rate :
Alpha_s[1] Alpha_s[2]
  0.112361   0.222748
```

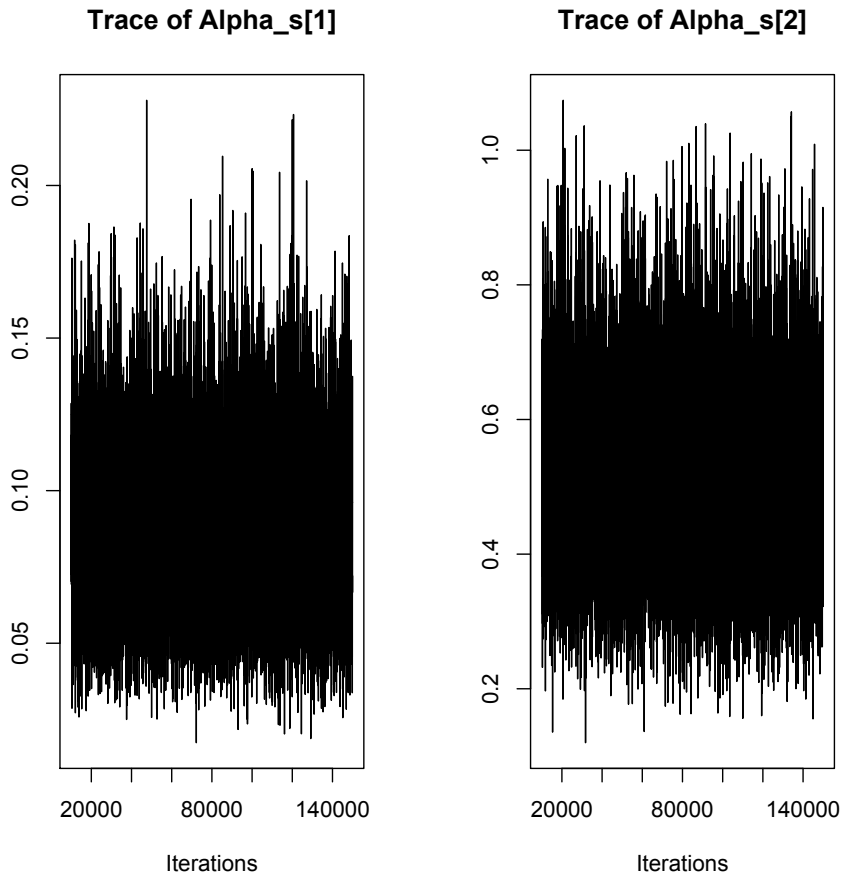

Figure 5: The MCMC chains of the posterior distributions of the model parameters fitting the simulated epidemic using the network-based continuous time ILM assuming fully observed epidemic.

The MCMC trace plots for the model parameters can be produced using the S3 `plot` method for '`epictmcmc`' objects.

```
R> plot(mcmc1, plottype = "parameter", start = 10000, thin = 10,
+    density = FALSE)
```

Figure 5 shows the MCMC trace plots for the model parameters α_0 and α_1. We observe a posterior mean of $\hat{\alpha}_1 = 0.508$ with 95% credible (percentile) interval (0.286, 0.789) and a posterior mean of $\hat{\alpha}_0 = 0.085$ with 95% credible interval (0.042, 0.144). We also observed well-mixed MCMC chains for both model parameters. The computation time for running the above MCMC code was 16 seconds on an Apple MacBook Pro with i5-core Intel 2.4 GHz processors with 8 GB of RAM.

For the known removal times analysis, **EpiILMCT** uses data augmented MCMC to infer infection times and the infectious period rate. Here, we assume that the infectious period follows a gamma distribution with shape 4 and unknown rate parameter δ; so $\mathcal{D}_i \sim \Gamma(4, \delta)$. Here, we also include a spark term ϵ. This is not strictly necessary but tends to improve MCMC mixing. We assigned gamma prior distribution $\Gamma(4, 2)$ for δ and an exponential prior distribution with rate 0.01 for ϵ.

We can then run the MCMC using the `epictmcmc` function for sampling a single chain of 150,000 iterations using the following code:

```
R> set.seed(91938)
R> suscov <- list(NULL)
R> suscov[[1]] <- list(c(0.01, 0.1), matrix(c("gamma", "gamma", 1, 1, 0.1,
+    0.1, 0.2, 0.8), ncol = 4, nrow = 2))
R> suscov[[2]] <- NetworkData[[2]]
R> spark <- list(0.01, c("gamma", 1, 0.01, 0.1))
R> mcmc11 <- epictmcmc(object = NetworkData[[1]],
+    datatype = "known removal", nsim = 150000, control.sus = suscov,
+    spark.par = spark, delta = list(4, 2, c(4, 2)))
```

The computation time for the above code on the aforementioned machine was 201 seconds. Figure 6 shows typical MCMC trace plots for the model parameters α_0, α_1, ϵ, and δ. Well-mixed MCMC chains are observed for all the model parameters.

As the posterior samples of the model parameters are stored in the '`epictmcmc`' object as an '`mcmc`' object of the type used in the **coda** package, the standard `summary` methods from **coda** can be employed, inserting `mcmc1$parameter.samples` as the input of this function. This is illustrated in the following command:

```
R> summary(window(mcmc11$parameter.samples, start = 10000, thin = 10))

Iterations = 10000:150000
Thinning interval = 10
Number of chains = 1
Sample size per chain = 14001
1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
                           Mean      SD  Naive SE Time-series SE
Alpha_s[1]              0.05717 0.03497 0.0002955      0.0004005
Alpha_s[2]              0.42976 0.14320 0.0012102      0.0015706
Spark                   0.03742 0.02001 0.0001691      0.0002553
Infectious period rate 2.64673 0.49444 0.0041786      0.0074440
2. Quantiles for each variable:
                            2.5%     25%     50%     75%   97.5%
Alpha_s[1]              0.004402 0.03059 0.05322 0.07817 0.13770
Alpha_s[2]              0.189793 0.32899 0.41698 0.51633 0.75099
Spark                   0.004977 0.02246 0.03548 0.04999 0.08208
Infectious period rate 1.821692 2.29301 2.59927 2.94168 3.73862
```

Thus, the posterior means and 95% credible intervals of the model parameters are as follows: $\hat{\alpha}_0 = 0.057$ (0.004, 0.138), $\hat{\alpha}_1 = 0.430$ (0.190, 0.751), $\hat{\epsilon} = 0.037$ (0.005, 0.082), and $\hat{\delta} = 2.647$ (1.822,3.739). The infection times are also well-approximated (see Figure 7). Figures 6 and 7 are produced using the following code:

```
R> plot(mcmc11, plottype = "parameter", start = 10000, thin = 10,
+    density = FALSE)
```

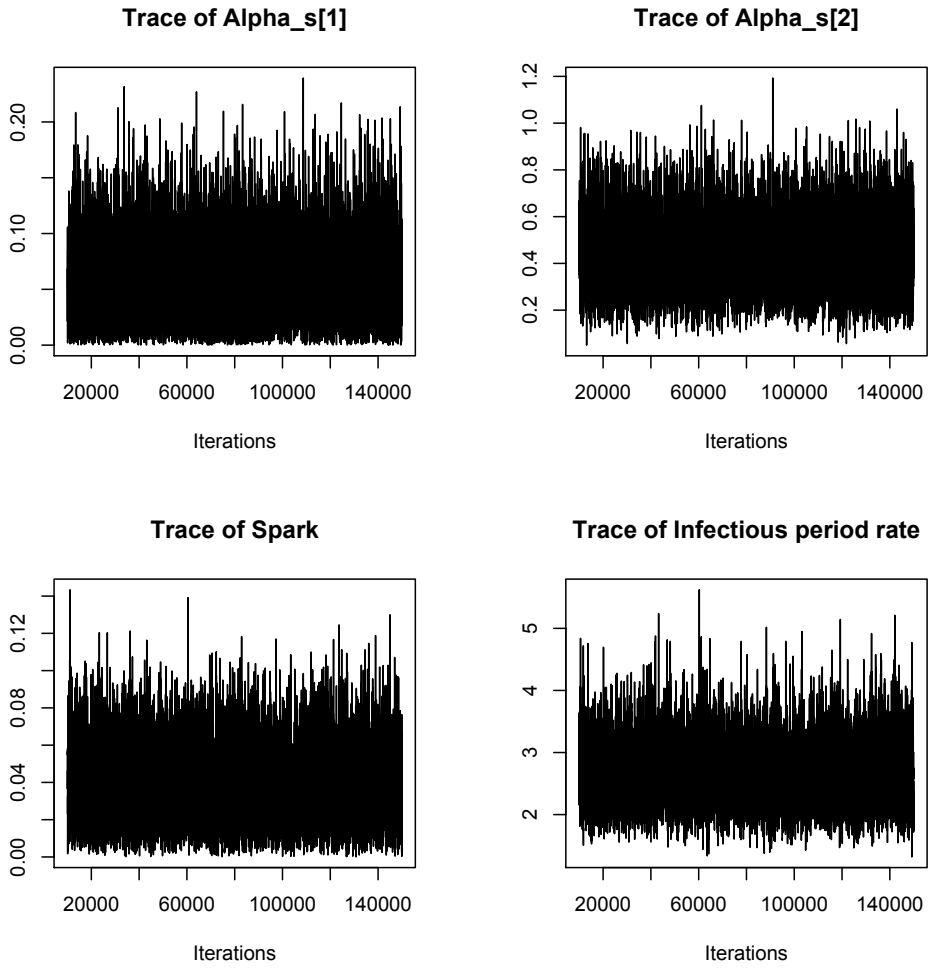

Figure 6: The MCMC chains of the posterior distributions of the model parameters for fitting the simulated epidemic using the network-based continuous time ILM assuming partially observed epidemic (unknown infection times).

```
R> plot(mcmc11, epi = NetworkData[[1]], plottype = "inf.times",
+    start = 10000, thin = 10)
R> lines(NetworkData[[1]]$epidat[,4], type = "l", col = "blue")
```

To check the fit of the model, we consider the posterior predictive distribution of four statistics. Specifically, we consider: T_1, the total number of infected individuals; T_2, the average removal time; T_3, the variance of the removal times; and T_4, the length of the epidemic. Here, we simulate 10,000 epidemics based on random draws of the model parameters from the MCMC output (excluding burnin) of the known removal times analysis (i.e., unknown infection times). We condition our simulation on the first ten infected individuals, then calculated the four statistics for each simulation. This simulation procedure is implemented in parallel using the `future_lapply` function from the **future.apply** package (Bengtsson 2021) as follows:

```
R> set.seed(524837)
R> mb <- sample(seq(10000, 150000), 10000)
```

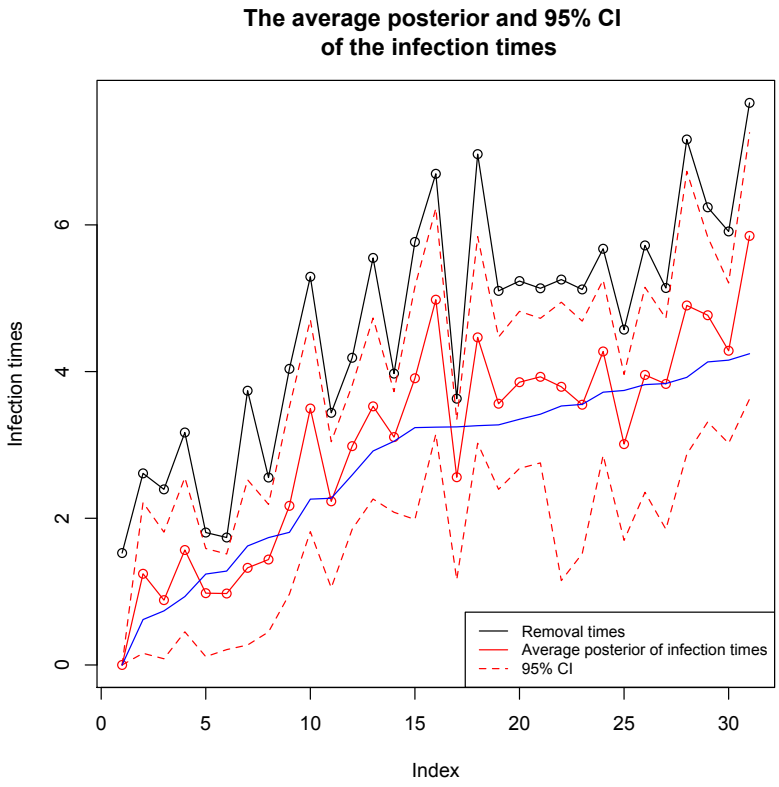

Figure 7: The posterior means and 95% credible intervals of the infection times for fitting the simulated epidemic using the network-based continuous time ILM assuming partially observed epidemic (unknown infection times). The black line represent the observed removal times, solid red line represent the posterior means, dotted red lines represent the 95% credible interval, and the blue line represents the observed infection times.

```
R> posterior.pred <- function(x) {
+    epi <- datagen(type = "SIR", kerneltype = "network",
+      kernelmatrix = NetworkData[[1]]$network, initialepi =
+      matrix(NetworkData[[1]]$epidat[1:10, ], ncol = 4, nrow = 10),
+      suspar = c(mcmc11$parameter.samples[x, 1],
+        mcmc11$parameter.samples[x, 2]),
+      spark = mcmc11$parameter.samples[x, 3],
+      delta = c(4, mcmc11$parameter.samples[x, 4]),
+      suscov = NetworkData[[2]])$epidat
+    numinf <- sum(epi[, 2] != Inf)
+    muremtime <- mean(epi[1:numinf, 2])
+    varremtime <- var(epi[1:numinf, 2])
+    lengthepi <- max(epi[1:numinf, 2])
+    result <- c(numinf, muremtime, varremtime, lengthepi)
+    return(result)
+  }
R> library("future.apply")
R> plan(multiprocess, workers = 4)
```

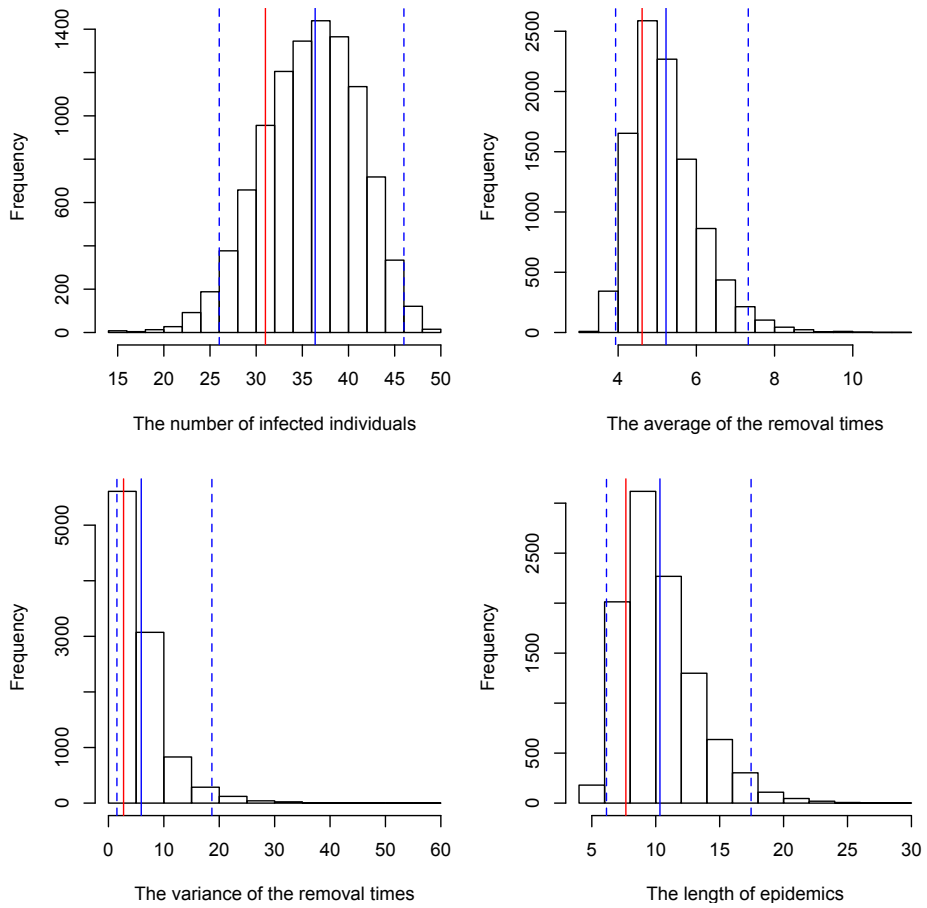

Figure 8: The posterior predictive distributions of four statistics: The number of infected individuals, the average removal times, the variance of removal times, and the length of epidemic for fitting partially observed epidemic (unknown infection times) using network-based continuous time ILM. The red vertical lines represent the observed statistic values and the solid and dotted blue vertical lines represents the posterior predictive means and 95% credible intervals of the four statistics.

```
R> datmcmc <- future_lapply(mb, FUN = posterior.pred, future.seed = TRUE)
R> summary.results <- sapply(datmcmc, unlist, simplify = TRUE)
```

The posterior predictive distributions of the four statistics are shown in Figure 8. We can see that each distribution captures the observed statistics well.

5.2. Case study: Tomato spotted wilt virus (TSWV) data

We further illustrate the **EpiILMCT** package by analyzing the TSWV data as described in Hughes, McRoberts, Madden, and Nelson (1997) and analyzed with spatial ILMs by Pokharel and Deardon (2014, 2016). These data represent the results of an experiment designed to study the spread of the disease amongst 520 pepper plants raised in a greenhouse. Plants were evenly distributed across a 10×26 meter area as shown in Figure 9. The experiment began on May 26, 1993 and finished on August 16, 1993. Plants were checked for the disease

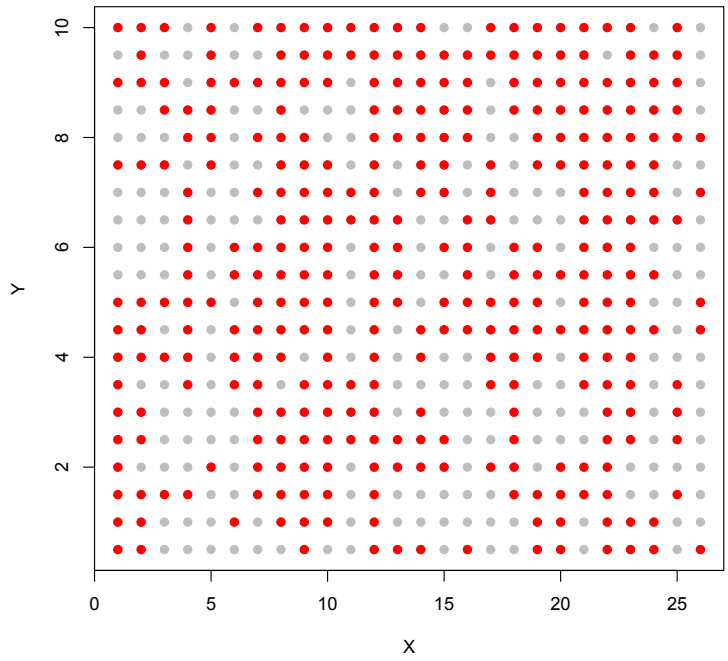

Figure 9: A grid plot of the XY coordinates of plants of TSWV data. The red points represent infected plants at the end of the disease.

every 14 days, and ultimately 327 were infected. Following Pokharel and Deardon (2014, 2016) these observation points are recorded to $t = 1, 2, \ldots, 7$. We set the initial infection time to $t = 2$ in line with the original data set.

We here analyze the epidemic under two data availability scenarios. First, we assume that the event times of the TSWV disease are fully observed. Here, the infectious period was fixed at three time points (42 days) following Pokharel and Deardon (2014, 2016). Additionally, the last observed time point was at $t = 7$. Second, we assume the epidemic is partially observed. Specifically, we assume that the infection and removal times are unknown, and treat the reported infection times as the notified time points. This entails considerable uncertainty and makes the MCMC analysis much more time consuming (more than 13 times longer than the computation time of the first analysis), because it is necessary to estimate both incubation and delay periods along with the infection and removal times.

The data is stored in the data file `tswv`, available in the **EpiILMCT** package. It contains a list of the TSWV epidemic data set for the two compartmental frameworks ($\mathcal{SIR}$ and $\mathcal{SINR}$) structured as a '`datagen`' class.

The following code shows how the TSWV data set can be extracted and the associated Euclidean distance matrix built.

```
R> data("tswv", package = "EpiILMCT")
R> names(tswv)

[1] "tswvsir"   "tswvsinr"

R> plot(tswv$tswvsir$location, col = "gray", pch = 19)
```

```
R> k1 <- sum(tswv$tswvsir$epidat[,2] != Inf)
R> points(tswv$tswvsir$location[tswv$tswvsir$epidat[1:k1, 1], ],
+    col = "red", pch = 19)
```

Following Pokharel and Deardon (2014, 2016), we implement the distance-based continuous time ILM with power-law kernel and without susceptibility and transmissibility covariates. For the first analysis, an $\mathcal{SIR}$ distance-based continuous time ILM is used where the infectivity rate given in Equation 2 becomes:

$$\lambda_j(t) = \left(\alpha \sum_{i \in \mathcal{I}(t)} d_{ij}^{-\beta} \right), \qquad \alpha, \beta > 0.$$

To perform the MCMC, the `epictmcmc` function should be used with `datatype` set to `"known epidemic"`. Here, we assume exponential prior distributions with rate 0.01 for the model parameters α and β; and we request 150,000 MCMC samples. The code to achieve this is as follows:

```
R> covsus <- list(NULL)
R> covsus[[1]] <- list(0.02, c("gamma", 1, 0.01, 0.01))
R> covsus[[2]] <- rep(1, length(tswv$tswvsir$epidat[,1]))
R> kernel1 <- list(2, c("gamma", 1, 0.01, 0.1))
R> set.seed(524837)
R> tswv.full.observed <- epictmcmc(object = tswv$tswvsir,
+    distancekernel = "powerlaw", datatype = "known epidemic", nsim = 150000,
+    control.sus = covsus, kernel.par = kernel1)
R> plot(tswv.full.observed, plottype = "parameter", start = 10000, thin = 10,
+    density = FALSE)
```

Figure 10 shows the resulting MCMC chains for the model parameters with a burn-in of 10,000 iterations and thinning interval of 10. The posterior mean of α and β were $\hat{\alpha} = 0.012$ and $\hat{\beta} = 1.306$, with 95% credible intervals of (0.007, 0.017) and (0.973,1.592), respectively. The estimates of $\hat{\alpha}$ and $\hat{\beta}$ are consistent with those of Pokharel and Deardon (2014, 2016). The above `epictmcmc` function had a run time of one hour on an Apple MacBook Pro with i5-core Intel 2.4 GHz processors with 8 GB of RAM.

In the second analysis (i.e., where infection and removal times are treated as unknown), we assume notified times were observed for all infected individuals. Consequently, an $\mathcal{SINR}$ distance-based continuous time ILM is used where the infectivity rate given in Equation 1 becomes:

$$\lambda_j(t) = \left(\alpha \sum_{i \in \mathcal{N}^-(t)} d_{ij}^{-\beta} \right) + \gamma \left(\alpha \sum_{i \in \mathcal{N}^+(t)} d_{ij}^{-\beta} \right).$$

We here assume the risk of infection does not reduce after notification, and set the notification effect parameter to $\gamma = 1$. The infectious period here is divided into the incubation and delay periods. We assume the total infectious period to be within three time points (42 days) following Pokharel and Deardon (2014, 2016); Brown *et al.* (2005). Thus, we assumed the incubation periods to follow an exponential distribution such that $D_i^{(inc)} \sim \text{Exp}(\delta^{(inc)})$ with initial value of $\delta^{(inc)} = 1$, whereas the delay periods are assumed to follow a gamma

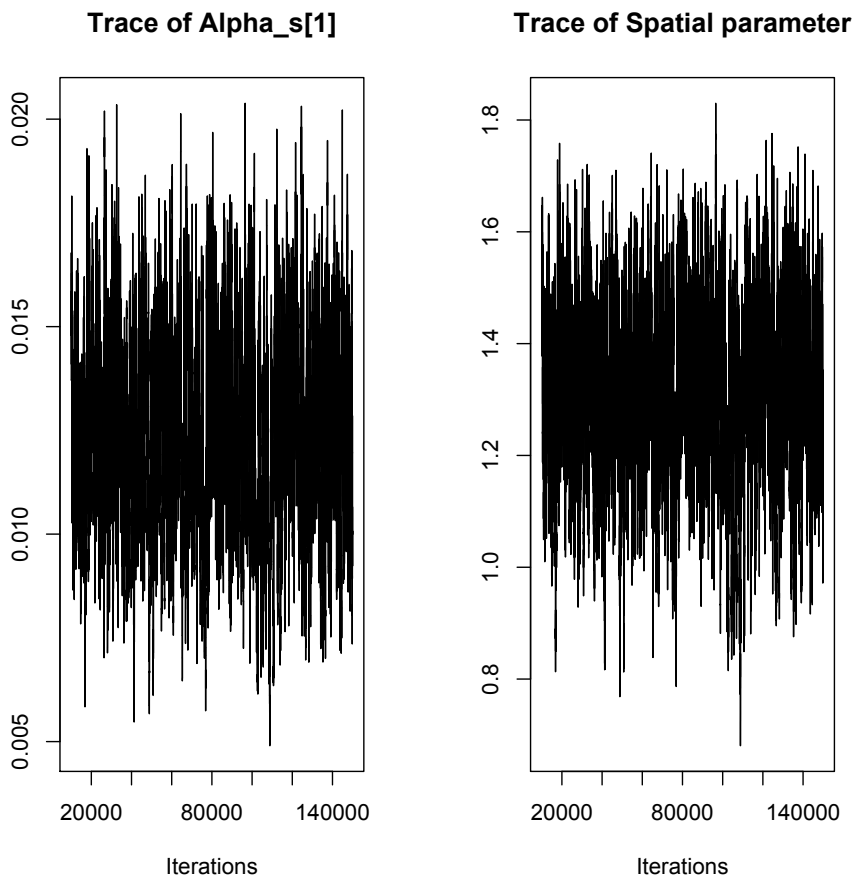

Figure 10: The MCMC chains of the posterior distributions of the model parameter α and β for fitting the fully observed TSWV data using the SIR distance-based continuous time ILM.

distribution such that $D_i^{(delay)} \sim \Gamma(10, \delta^{(delay)})$ with initial value of $\delta^{(delay)} = 5$. We assign gamma prior distributions for both rates such that $\delta^{(inc)} \sim \Gamma(10, 10)$ and $\delta^{(delay)} \sim \Gamma(60, 12)$. These choices are to cover the support of our assumptions about the infectious periods. For simplicity, we assume the infection time of the first infected plant is known. We set its incubation period to one time point.

We assign exponential prior distributions with rate 0.01 to the model parameters α and β. To perform the MCMC, we use the `epictmcmc` function with `type` and `datatype` set to `"SINR"` and `"unknown removal"`, respectively. At each iteration, the infection and removal times are updated in blocks of 10 randomly selected individuals (`blockupdate`). For faster implementation, we run the `epictmcmc` function in parallel to obtain 50,000 samples from four MCMC chains with four different sets of initial values of the model parameters and seed values. To do so, we set the argument `nchains = 4` and set `parallel = TRUE`. The number of cores to be used depends on the minimum number of the available cores and the number of chains (`nchains`). The following code was run using the four available cores of an Apple iMac with i5-core Intel 2.4 GHz processors and 8 GB of RAM.

```
R> covsus <- list(NULL)
R> covsus[[1]] <- list(NULL)
```

	α	β	$\delta^{(inc)}$	$\delta^{(delay)}$
Mean	0.043	2.037	2.992	9.139
95% CI	(0.034, 0.051)	(1.780, 2.275)	(2.264, 3.874)	(8.046, 10.292)

Table 3: The posterior means and 95% credible intervals (CIs) of the model parameters, with a burn-in of 5,000 iterations and thinning interval of 10 for each of the four MCMC chains, for fitting the TSWV using the SINR distance-based continuous time ILM under the assumption of unknown removal and infection times.

```
R> covsus[[1]][[1]] <- c(0.02, 0.1, 1.5, 3)
R> covsus[[1]][[2]] <- c("gamma", 1, 0.01, 0.01)
R> covsus[[2]] <- rep(1, length(tswv$tswvsir$epidat[,1]))
R> kernel1 <- list(c(0.1, 5, 1, 10), c("gamma", 1, 0.01, 0.1))
R> delta1 <- list(NULL)
R> delta1[[1]] <- c(1,10)
R> delta1[[2]] <- matrix(c(10, 5, 1, 0.5, 15, 2, 1, 15), ncol = 4, nrow = 2)
R> delta1[[3]] <- matrix(c(10, 60, 10, 12), ncol = 2, nrow = 2)
R> set.seed(524837)
R> tswv.unknown.remov.infect1 <- epictmcmc(object = tswv$tswvsinr,
+    distancekernel = "powerlaw", datatype = "unknown removal",
+    blockupdate = c(1, 10), nsim = 50000, nchains = 4, parallel = TRUE,
+    control.sus = covsus, kernel.par = kernel1, delta = delta1)
```

Figure 11 shows the MCMC trace plots and Gelman-Rubin convergence diagnostic plots for the model parameters α, β, $\delta^{(inc)}$ and $\delta^{(delay)}$ with a burn-in of 5,000 iterations removed and a thinning interval of 10 for the four MCMC chains. Figure 11 can be produced using the following code:

```
R> layout(matrix(c(5, 1, 6, 2, 7, 3, 8, 4), ncol = 2, byrow = TRUE))
R> m1 <- window(tswv.unknown.remov.infect1$parameter.samples[[1]],
+    start = 5000)
R> m2 <- window(tswv.unknown.remov.infect1$parameter.samples[[2]],
+    start = 5000)
R> m3 <- window(tswv.unknown.remov.infect1$parameter.samples[[3]],
+    start = 5000)
R> m4 <- window(tswv.unknown.remov.infect1$parameter.samples[[4]],
+    start = 5000)
R> gelman.plot(mcmc.list(m1, m2, m3, m4), auto.layout = FALSE)
R> plot(tswv.unknown.remov.infect1, plottype = "parameter", start = 5000,
+    thin = 10, density = FALSE, smooth = FALSE, auto.layout = FALSE)
```

The posterior means and 95% credible intervals of these parameters are given in Table 3. The MCMC chains show good mixing with both trace and Gelman-Rubin plots suggesting convergence.

Figures 12 and 13 show the posterior means and 95% credible intervals of the infection and removal times. These figures can be produced using the following commands:

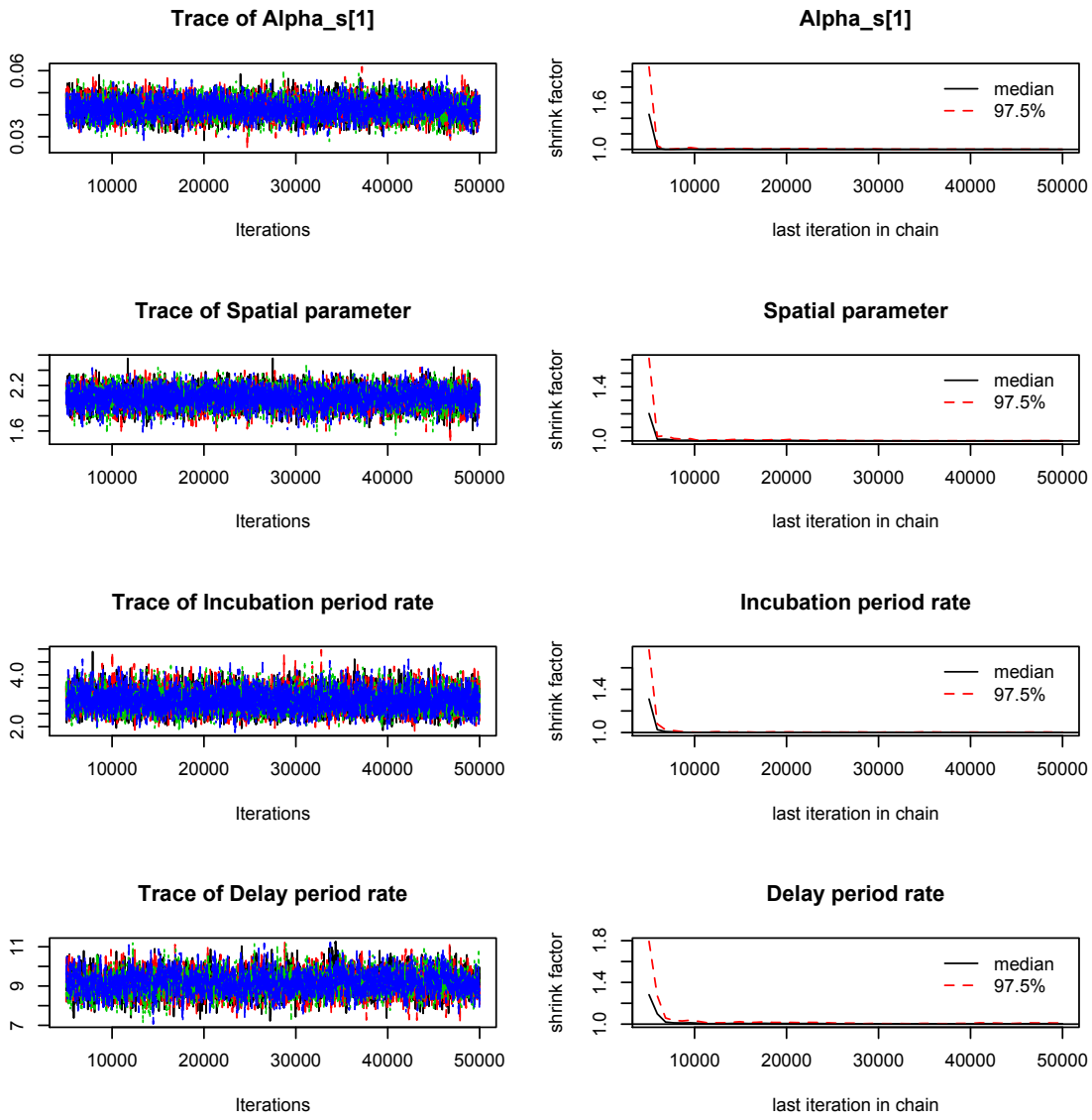

Figure 11: The four MCMC chains (left) and Gelman-Rubin convergence diagnostic (right) plots of the posterior distributions of the model parameters α, β, $\delta^{(inc)}$ and $\delta^{(delay)}$ for fitting the partially observed TSWV data (unknown infection and removal times) using the SINR distance-based continuous time ILM.

```
R> plot(tswv.unknown.remov.infect1, epi = tswv$tswvsinr,
+    plottype = "inf.times", start = 5000, thin = 10)
R> plot(tswv.unknown.remov.infect1, epi = tswv$tswvsinr,
+    plottype = "rem.times", start = 5000, thin = 10)
```

Using the `summary` function of the object `tswv.unknown.remov.infect1`, the posterior means (95% CIs) of the incubation and delay periods were found to be 0.320 (0.242, 0.414) and 1.082 (0.957, 1.224) observation time points, respectively, indicating an average infectious period of 19.628 days (1.402 time points).

Note that, infection and removal times are updated here in blocks of 10 (via the `blockupdate`

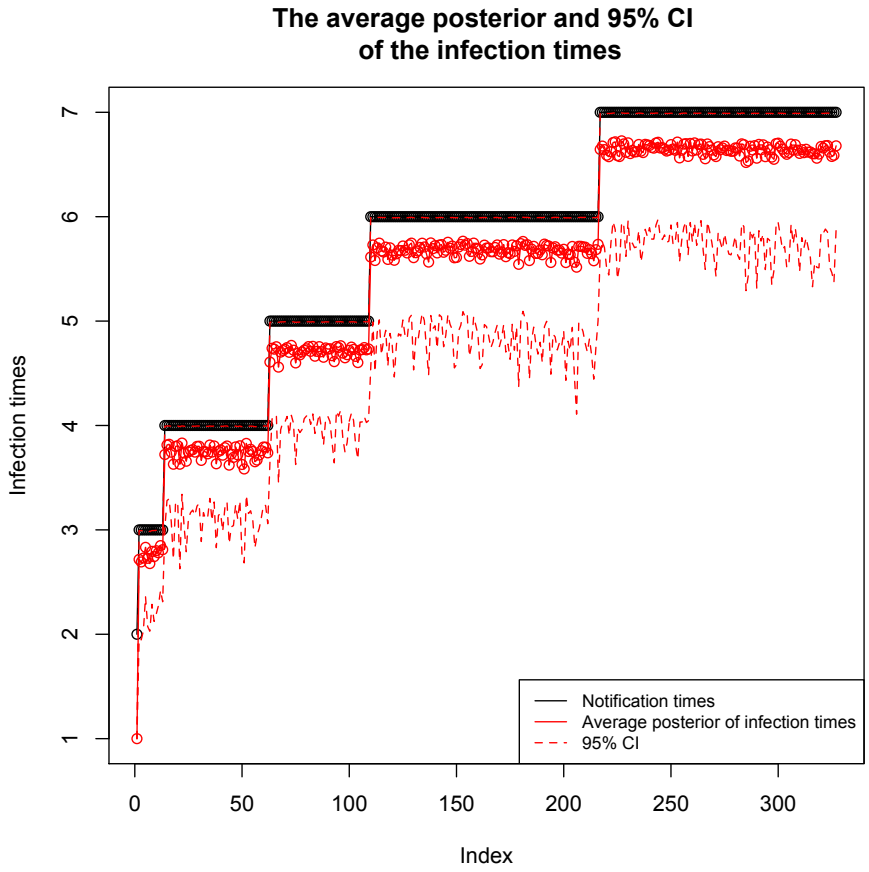

Figure 12: The posterior means (solid red line) and 95% credible intervals (dotted red lines) of the infection times for fitting the partially observed TSWV data (unknown infection and removal times) using the SINR distance-based continuous time ILM. The black line represents the observed notification times.

argument) for reasons of computational efficiency. The `epictmcmc` function had a run time of 9.51 hours using the parallel method with 4 cores, but this was computationally much more efficient than if single updates were used ($\approx$ 124 hours, calculated based on ten MCMC iterations).

6. Conclusion

This paper introduces the R software package **EpiILMCT**, which facilitates the use of a broad range of continuous time ILMs under two compartmental frameworks ($\mathcal{SIR}$ and $\mathcal{SINR}$). It also allows for the analysis of partially observed infectious diseases data, achieved using data augmented MCMC within a Bayesian framework. We illustrated the package by fitting continuous time ILMs on simulated and real epidemic data. The paper did not cover all functionality of the package. For instance, we did not illustrate incorporating both distance and network in the kernel function, or allowing for nonlinearity between the susceptibility and transmissibility risk factors and the infection rate. However, implementation of such

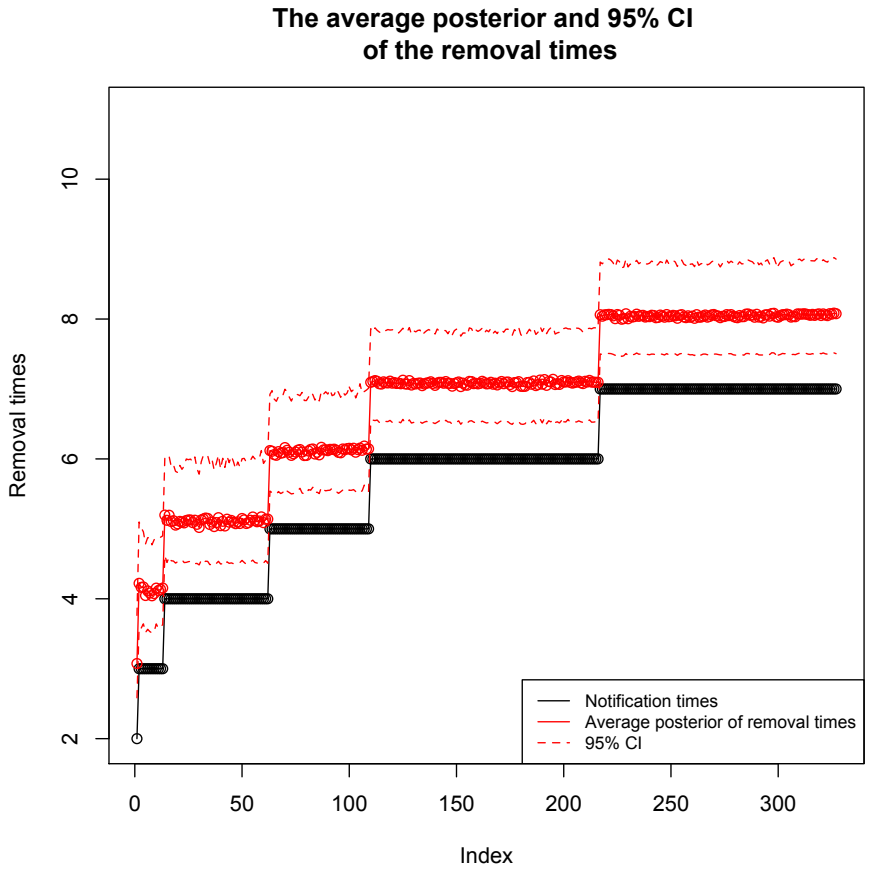

Figure 13: The posterior means (solid red line) and 95% credible intervals (dotted red lines) of the removal times for fitting the partially observed TSWV data (unknown infection and removal times) using the SINR distance-based continuous time ILM. The black line represents the observed notification times.

facets is simple. Additional functionality that was not covered in Sections 5 can be found via `help(package = "EpiILMCT")`.

Also, it is possible to use **EpiILMCT** to test the efficacy of disease control strategies (e.g., vaccination or culling) via simulation study. This can be done by simulating epidemics in small time steps and then manipulating infection and/or removal times according to a given control policy, before simulating the next step of the epidemic simulation conditional upon the manipulated epidemic history just determined. We illustrate this via a simple ring-culling strategy in Appendix C.

In terms of future developments, the authors intend to expand the modeling framework to allow for latent periods i.e., susceptible-exposed-infectious-removed ($\mathcal{SEIR}$) and susceptible-exposed-infectious-notified-removed ($\mathcal{SEINR}$). This would be useful for many disease systems in which the time between infection (exposure) and infectiousness cannot be reasonably ignored. Additionally, expanding the compartmental frameworks to allow for reinfection would also be useful for diseases such as influenza. That is, we could allow for frameworks: susceptible-infectious-susceptible ($\mathcal{SIS}$), susceptible-exposed-infectious-susceptible ($\mathcal{SEIS}$), etc.

Incorporating more data uncertainty into the analyses, especially under the network-based model, is an option for future development of this package **EpiILMCT**. For example, networks are often only partially observed. However, the data augmentation could easily make the computation time for data analyses prohibitive. Various strategies for mitigating this might be available. For example, approximate forms of inference such as Gaussian process emulation (Pokharel and Deardon 2016), approximate Bayesian computation (Beaumont, Cornuet, Marin, and Robert 2009), machine learning based model classification (Pokharel and Deardon 2014), data-sampled likelihood approximation (Malik *et al.* 2016), or data-aggregation (Deeth and Deardon 2016) could all prove useful for overcoming these computational issues. Finally, it would be possible to extend our modeling framework to allow for multiple, interacting disease strains or pathogens (Romanescu and Deardon 2016).

Acknowledgments

This work was funded by the Ontario Ministry of Agriculture, Food and Rural Affairs (OMAFRA), and the Natural Sciences and Engineering Research Council of Canada (NSERC). Almutiry was also funded by Qassim University through the Saudi Arabian Cultural Bureau in Canada. Warriyar was funded by the University of Calgary Eyes High Post Doctoral Scholarship scheme. We thank the editor and referees for their valuable suggestions and comments, which greatly improved both the software package and this manuscript.

References

Almutiry W, Deardon R, Warriyar K V V (2021). ***EpiILMCT**: Continuous Time Distance-Based and Network-Based Individual Level Models for Epidemics*. R package version 1.1.7, URL https://CRAN.R-project.org/package=EpiILMCT.

Bakar KS, Sahu SK (2015). "**spTimer**: Spatio-Temporal Bayesian Modeling Using R." *Journal of Statistical Software*, **63**(15), 1–32. doi:10.18637/jss.v063.i15.

Bakar KS, Sahu SK (2020). ***spTimer**: Spatio-Temporal Bayesian Modeling Using R*. R package version 3.3.1, URL https://CRAN.R-project.org/package=spTimer.

Beaumont MA, Cornuet JM, Marin JM, Robert CP (2009). "Adaptive Approximate Bayesian Computation." *Biometrika*, **96**(4), 983–990. doi:10.1093/biomet/asp052.

Bengtsson H (2021). ***future.apply**: Apply Function to Elements in Parallel Using Futures*. R package version 1.7.0, URL https://CRAN.R-project.org/package=future.apply.

Bifolchi N, Deardon R, Feng Z (2013). "Spatial Approximations of Network-Based Individual Level Infectious Disease Models." *Spatial and Spatio-Temporal Epidemiology*, **6**, 59–70. doi:10.1016/j.sste.2013.07.001.

Bivand R, Hauke J, Kossowski T (2013). "Computing the Jacobian in Gaussian Spatial Autoregressive Models: An Illustrated Comparison of Available Methods." *Geographical Analysis*, **45**(2), 150–179. doi:10.1111/gean.12008.

Bivand R, Piras G (2015). "Comparing Implementations of Estimation Methods for Spatial Econometrics." *Journal of Statistical Software*, **63**(18), 1–36. doi:10.18637/jss.v063.i18.

Britton T, O'Neill PD (2002). "Bayesian Inference for Stochastic Epidemics in Populations with Random Social Structure." *Scandinavian Journal of Statistics*, **29**(3), 375–390. doi:10.1111/1467-9469.00296.

Brown S, Csinos A, Díaz-Pérez JC, Gitaitis R, LaHue SS, Lewis J, Martinez N, McPherson R, Mullis S, Nischwitz C, *et al.* (2005). "Tospoviruses in Solanaceae and Other Crops in The Coastal Plain of Georgia." *Research Report 704*, The University of Georgia College of Agriculture and Environmental Sciences.

Caimo A, Friel N (2014). "**Bergm**: Bayesian Exponential Random Graphs in R." *Journal of Statistical Software*, **61**(2), 1–25. doi:10.18637/jss.v061.i02.

Csardi G, Nepusz T (2006). "The **igraph** Software Package for Complex Network Research." *InterJournal*, **Complex Systems**, 1695.

Deardon R, Brooks SP, Grenfell BT, Keeling MJ, Tildesley MJ, Savill NJ, Shaw DJ, Woolhouse MEJ (2010). "Inference for Individual-Level Models of Infectious Diseases in Large Populations." *Statistica Sinica*, **20**(1), 239.

Deeth LE, Deardon R (2016). "Spatial Data Aggregation for Spatio-Temporal Individual-Level Models of Infectious Disease Transmission." *Spatial and Spatio-Temporal Epidemiology*, **17**, 95–104. doi:10.1016/j.sste.2016.04.013.

Groendyke C, Welch D (2018). "**epinet**: An R Package to Analyze Epidemics Spread across Contact Networks." *Journal of Statistical Software*, **83**(11), 1–22. doi:10.18637/jss.v083.i11.

Handcock MS, Hunter DR, Butts CT, Goodreau SM, Krivitsky PN, Morris M (2021). ***ergm**: Fit, Simulate and Diagnose Exponential-Family Models for Networks*. The Statnet Project (https://statnet.org/). R package version 4.0.1, URL https://CRAN.R-project.org/package=ergm.

Höhle M, Meyer S, Paul M (2021). ***surveillance**: Temporal and Spatio-Temporal Modeling and Monitoring of Epidemic Phenomena*. R package version 1.19.1, URL https://CRAN.R-project.org/package=surveillance.

Hughes G, McRoberts N, Madden LV, Nelson SC (1997). "Validating Mathematical Models of Plant-Disease Progress in Space and Time." *Mathematical Medicine and Biology: A Journal of the IMA*, **14**(2), 85–112. doi:10.1093/imammb/14.2.85.

Hunter DR, Handcock MS, Butts CT, Goodreau SM, Morris M (2008). "**ergm**: A Package to Fit, Simulate and Diagnose Exponential-Family Models for Networks." *Journal of Statistical Software*, **24**(3), 1–29. doi:10.18637/jss.v024.i03.

Jenness SM, Goodreau SM, Morris M (2018). "**EpiModel**: An R Package for Mathematical Modeling of Infectious Disease over Networks." *Journal of Statistical Software*, **84**(8), 1–47. doi:10.18637/jss.v084.i08.

Jewell CP, Kypraios T, Neal P, Roberts GO (2009). "Bayesian Analysis for Emerging Infectious Diseases." *Bayesian Analysis*, **4**(3), 465–496. doi:10.1214/09-ba417.

Kwong GPS, Poljak Z, Deardon R, Dewey CE (2013). "Bayesian Analysis of Risk Factors for Infection with A Genotype of Porcine Reproductive and Respiratory Syndrome Virus in Ontario Swine Herds Using Monitoring Data." *Preventive Veterinary Medicine*, **110**(3-4), 405–417. doi:10.1016/j.prevetmed.2013.01.004.

Lee D, Rushworth A, Napier G (2018). "Spatio-Temporal Areal Unit Modeling in R with Conditional Autoregressive Priors Using the **CARBayesST** Package." *Journal of Statistical Software*, **84**(9), 1–39. doi:10.18637/jss.v084.i09.

Malik R, Deardon R, Kwong GPS (2016). "Parameterizing Spatial Models of Infectious Disease Transmission That Incorporate Infection Time Uncertainty Using Sampling-Based Likelihood Approximations." *PLOS One*, **11**(1), e0146253. doi:10.1371/journal.pone.0146253.

Malik R, Deardon R, Kwong GPS, Cowling BJ (2014). "Individual-Level Modeling of the Spread of Influenza within Households." *Journal of Applied Statistics*, **41**(7), 1578–1592. doi:10.1080/02664763.2014.881787.

Martin AD, Quinn KM, Park JH (2011). "**MCMCpack**: Markov Chain Monte Carlo in R." *Journal of Statistical Software*, **42**(9), 22. doi:10.18637/jss.v042.i09.

Meyer S, Held L, Höhle M (2017). "Spatio-Temporal Analysis of Epidemic Phenomena Using the R Package **surveillance**." *Journal of Statistical Software*, **77**(11), 1–55. doi:10.18637/jss.v077.i11.

Pebesma E (2021). *CRAN Task View: Handling and Analyzing Spatio-Temporal Data*. Version 2021-05-26, URL https://CRAN.R-project.org/view=SpatioTemporal.

Pitzer VE, Leung GM, Lipsitch M (2007). "Estimating Variability in the Transmission of Severe Acute Respiratory Syndrome to Household Contacts in Hong Kong, China." *American Journal of Epidemiology*, **166**(3), 355–363. doi:10.1093/aje/kwm082.

Plummer M, Best N, Cowles K, Vines K (2006). "**coda**: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL https://CRAN.R-project.org/doc/Rnews/.

Pokharel G, Deardon R (2014). "Supervised Learning and Prediction of Spatial Epidemics." *Spatial and Spatio-Temporal Epidemiology*, **11**, 59–77. doi:10.1016/j.sste.2014.08.003.

Pokharel G, Deardon R (2016). "Gaussian Process Emulators for Spatial Individual-Level Models of Infectious Disease." *Canadian Journal of Statistics*, **44**(4), 480–501. doi:10.1002/cjs.11304.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

R Epidemics Consortium (2018). "Released Projects and Packages." Retrieved 2018-03-28 from http://www.repidemicsconsortium.org/projects/.

Romanescu R, Deardon R (2016). "Modeling Two Strains of Disease via Aggregate-Level Infectivity Curves." *Journal of Mathematical Biology*, **72**(5), 1195–1224. doi:10.1007/s00285-015-0910-3.

Rowlingson B, Diggle P (2021). ***splancs**: Spatial and Space-Time Point Pattern Analysis*. R package version 2.01-42, URL https://CRAN.R-project.org/package=splancs.

Salmon M, Schumacher D, Höhle M (2016). "Monitoring Count Time Series in R: Aberration Detection in Public Health Surveillance." *Journal of Statistical Software*, **70**(10), 1–35. doi:10.18637/jss.v070.i10.

Scheidegger A (2021). ***adpatMCMC**: Implementation of a Generic Adaptive Monte Carlo Markov Chain Sampler*. R package version 1.4, URL https://CRAN.R-project.org/package=adaptMCMC.

Schweinberger M, Handcock M, Luna P (2021). ***hergm**: Hierarchical Exponential-Family Random Graph Models with Local Dependence*. R package version 4.1-7, URL https://CRAN.R-project.org/package=hergm.

Taylor BM, Davies TM, Rowlingson BS, Diggle PJ (2013). "**lgcp**: An R Package for Inference with Spatial and Spatio-Temporal Log-Gaussian Cox Processes." *Journal of Statistical Software*, **52**(4), 1–40. doi:10.18637/jss.v052.i04.

Taylor BM, Davies TM, Rowlingson BS, Diggle PJ (2015). "Bayesian Inference and Data Augmentation Schemes for Spatial, Spatiotemporal and Multivariate Log-Gaussian Cox Processes in R." *Journal of Statistical Software*, **63**(7), 1–48. doi:10.18637/jss.v063.i07.

Warriyar K V V, Almutiry W, Deardon R (2020). ***EpiILM**: Spatial and Network Based Individual Level Models for Epidemics*. R package version 1.5.2, URL https://CRAN.R-project.org/package=EpiILM.

A. The likelihood of the general $\mathcal{SINR}$ continuous time ILMs

$$
\begin{aligned}
L(\boldsymbol{I},\boldsymbol{N},\boldsymbol{R}|\boldsymbol{\theta}) &= \prod_{j=2}^{m}\left(\epsilon + \sum_{i:I_i<I_j\leq N_i}\lambda_{ij}^{-}(I_j) + \sum_{i:N_i<I_j\leq R_i}\lambda_{ij}^{+}(I_j)\right) \\
&\times \exp\left\{-\int_{I_1}^{t_{obs}}\left(\sum_{i\in\mathcal{S}(u)}\epsilon + \sum_{i\in\mathcal{I}(u)}\sum_{j\in\mathcal{S}(u)}\lambda_{ij}^{-}(u-I_i) + \sum_{i\in\mathcal{N}(u)}\sum_{j\in\mathcal{S}(u)}\lambda_{ij}^{+}(u-I_i)\right)du\right\} \\
&\times \prod_{i=1}^{m} f(\mathcal{D}_i^{(inc)};\delta^{(inc)})\prod_{i=1}^{m} f(\mathcal{D}_i^{(delay)};\delta^{(delay)}) \\
&= \prod_{j=2}^{m}\left(\epsilon + \sum_{i:I_i<I_j\leq N_i}\lambda_{ij}^{-}(I_j) + \sum_{i:N_i<I_j\leq R_i}\lambda_{ij}^{+}(I_j)\right) \\
&\times \exp\left\{-\sum_{i=1}^{m}\left(\sum_{j=1}^{N}((t_{obs}\wedge N_i\wedge I_j)-(I_i\wedge I_j))\lambda_{ij}^{-}(I_j)\right)\right\} \\
&\times \exp\left\{-\sum_{i=1}^{m}\left(\sum_{j=1}^{N}((t_{obs}\wedge R_i\wedge I_j)-(I_i\wedge I_j)) - ((t_{obs}\wedge N_i\wedge I_j)-(I_i\wedge I_j))\lambda_{ij}^{+}(I_j)\right)\right\} \\
&\times \exp\left(-\epsilon\sum_{i=1}^{N}[(t_{obs}\wedge I_i)-I_1]\right) \\
&\times \prod_{i=1}^{m} f(\mathcal{D}_i^{(inc)};\delta^{(inc)})\prod_{i=1}^{m} f(\mathcal{D}_i^{(delay)};\delta^{(delay)}), \qquad \delta^{(inc)},\delta^{(delay)}>0, \qquad (3)
\end{aligned}
$$

where the wedge symbol $\wedge$ denotes the minimum operator; and $\mathcal{D}_i^{inc}$ and $\mathcal{D}_i^{delay}$ are the incubation and delay periods such that $\mathcal{D}_i^{inc} = N_i - I_i$ and $\mathcal{D}_i^{delay} = R_i - N_i$, respectively.

B. R code to extract individual level data from surveillance

Here, we illustrate the extraction of individual level data from the **surveillance** package for use in the **EpiILMCT** package. We consider the toy data set representing a population of 100 individuals that is used in the `twinSIR` examples of the **surveillance** package (Höhle, Meyer, and Paul 2021).

```
R> library("surveillance")
R> data("fooepidata", package = "surveillance")
R> names(fooepidata)
```

```
 [1] "BLOCK"   "id"      "start"   "stop"    "atRiskY" "event"   "Revent"
 [8] "x"       "y"       "z1"      "z2"      "B1"      "B2"
```

The `fooepidata` event history consists of 178 time `BLOCK`s of 100 rows, where each row describes the state of individual id during the corresponding time interval (`start`, `stop`).

```
R> head(fooepidata, n = 5)

     BLOCK id start      stop atRiskY event Revent           x          y
1        1  1     0 0.6970682       1     0      0  1.262954285  0.7818592
246      1  2     0 0.6970682       1     0      0 -0.326233361 -0.7767766
369      1  3     0 0.6970682       1     0      0  1.329799263 -0.6159899
612      1  4     0 0.6970682       1     0      0  1.272429321  0.0465803
760      1  5     0 0.6970682       1     0      0  0.414641434 -1.1303858
     z1        z2 B1 B2
1     0 0.0000000  0  0
246   1 0.6931472  0  0
369   0 1.0986123  0  0
612   1 1.3862944  0  0
760   1 1.6094379  0  0
[....]
```

The `start` and `stop` variables represent the start and end of interval time points (in continuous time) that indicate the waiting time between consequence event times (infection and removal times). The binary variables `event` and `Revent` are used to indicate the occurrence of newly infected or removed individuals at the stop time of each time interval (`BLOCK`), respectively. Thus, the `stop` time is taken to be the infection or removal times of the infected or removed individuals in each time interval. The coordinates of individuals is represented in columns `x` and `y`. The `fooepidata` data set contains also endemic and epidemic covariates. Endemic covariates are represented by the columns named `z1` and `z2` (the exact interpretation of these covariates is not given). Epidemic covariates are represented by the columns named `B1` and `B2`, and they indicate the count of currently infective individuals for each individual within, and greater than one unit distance, respectively. See (`help(epidata, package= "surveillance")`) for more details about the data structure. From this data set, we extract only the event times and XY coordinates of each individual, ignoring the purely spatial epidemic covariates which are directly modelled by the distance kernel in **EpiILMCT**.

```
R> epi <- summary(fooepidata)$byID
R> loc <- summary(fooepidata)$coordinates
R> epi[is.na(epi)] <- Inf
R> epi <- transform(epi, period = ifelse(is.infinite(time.I), 0,
+    time.R - time.I))
R> epi$id <- as.integer(as.character(epi$id))
R> epidat <- as.matrix(epi[c("id", "time.R", "period", "time.I")])
R> library("EpiILMCT")
R> epi <- as.epidat(type = "SIR", kerneltype = "distance",
+    inf.time = epidat[, 4], rem.time = epidat[, 2],
+    id.individual = epidat[, 1], location = loc)
```

The object `epi` of class '`datagen`' can be now used in the **EpiILMCT** package using the model given in Equation 2 without covariates through the following code:

```
R> set.seed(101)
R> sus.par <- list(NULL)
```

```
R> sus.par[[1]] <- list(0.1, c("gamma", 1, 0.001, 0.005))
R> sus.par[[2]] <- matrix(rep(1, length(epi$epidat[, 1])), ncol = 1)
R> kernel <- list(0.1, c("gamma", 1, 0.001, 0.1))
R> spark <- list(0.1, c("gamma", 1, 0.001, 0.05))
R> mcmc1 <- epictmcmc(object = epi, distancekernel = "powerlaw",
+    datatype = "known epidemic", nsim = 50000, control.sus = sus.par,
+    kernel.par = kernel, spark.par = spark)
```

We include the spark term here to best model the endemic component used in the `twinSIR` model. The inclusion of the spark term also allows for the fact that there are no infectious individuals during times intervals (`BLOCK`) of the epidemic. The infection of individuals in these periods is captured by the endemic part in `twinSIR` function.

Without incorporating the spark term in the `epictmcmc` function, a zero likelihood will result, preventing the successful fitting of the model to the data. To get the output estimates of the model parameters, we used the S3 `summary` method for '`epictmcmc`' objects as follows:

```
R> summary(mcmc1, start = 1000)

**********************************************************
Model: SIR distance-based continuous-time ILM
Method: Markov chain Monte Carlo (MCMC)
Data assumption: fully observed epidemic
number.chains : 1 chains
number.iteration : 49000 iterations
number.parameter : 3 parameters
**********************************************************
 1. Empirical mean and standard deviation for each variable,
plus standard error of the mean:
                       Mean         SD    Naive SE Time-series SE
Alpha_s[1]        0.00889042 0.00110553 4.99425e-06    1.54972e-05
Spark             0.00778819 0.00436839 1.97342e-05    6.85098e-05
Spatial parameter 0.94175614 0.18258926 8.24846e-04    3.82173e-03
 2. Quantiles for each variable:
                       2.5%        25%        50%        75%     97.5%
Alpha_s[1]        0.00686125 0.00811098 0.00884201 0.00962789 0.0111386
Spark             0.00131269 0.00452088 0.00718377 0.01034424 0.0180864
Spatial parameter 0.54032375 0.82833931 0.95593444 1.07000331 1.2615483
 3. Empirical mean, standard deviation, and quantiles for the log likelihood,
         Mean             SD       Naive SE Time-series SE
  -2.30176e+02    1.23456e+00    5.57714e-03    2.00104e-02
    2.5%      25%      50%      75%    97.5%
-233.367 -230.757 -229.854 -229.263 -228.752
 4. acceptance.rate :
      Alpha_s[1]             Spark Spatial parameter
        0.253945          0.169543          0.810156
```

We also demonstrate the modeling of these data using `twinSIR` function with no endemic covariates. However, a baseline term (baseline hazard rate) will be included in this case

in the endemic component to represent the background rate of infection in the population, as explained in the note Section in `help("twinSIR", package = "surveillance")`. The following code illustrates the use of `twinSIR` in analyzing this data set.

```
R> fit1 <- twinSIR( ~ B1 + B2, data = fooepidata)
R> summary(fit1)


Call:
twinSIR(formula = ~B1 + B2, data = fooepidata)
Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
B1                0.023960   0.004208   5.693 1.25e-08 ***
B2                0.003395   0.001119   3.034  0.00241 **
cox(logbaseline) -6.010580   0.659257  -9.117  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Total number of infections:  88
One-sided AIC: 474.05
Log-likelihood: -235.2
Number of log-likelihood evaluations: 26
```

The posterior means of the ILM parameters (α, β) are 0.009 and 0.945, respectively. Figure 14 shows the ILM power-law distance kernel function under the posterior mean, along with the distance function suggested by the MLEs of the model parameters from the `twinSIR` analysis. We can see broad agreement, although the step function assumption of the `twinSIR` seems less reasonable than the continuous decay of the ILM kernel for short distances (less than one distance unit).

C. R code to implement ring-based control strategy

Here, we illustrate the use of the **EpiILMCT** package in testing the efficacy of a ring-based control strategy for mitigating the spread of disease. We consider an example in which an infectious disease is transmitted between 625 individuals located in a square area of 50×50 units. These individuals could be thought to represent farms or trees, say. We implement control measures upon all individuals within a circle of r radius of newly infected individuals. This control strategy essentially places these individuals in the removed set. These measures could be thought to represent vaccination or quarantine, but here we assume it is a culling strategy.

To illustrate we first simulate the XY coordinates of individuals from a uniform distribution. This is done as follows:

```
R> library("EpiILMCT")
R> set.seed(101)
R> n <- 625
R> loc <- cbind(runif(n, 0, 50), runif(n, 0, 50))
```

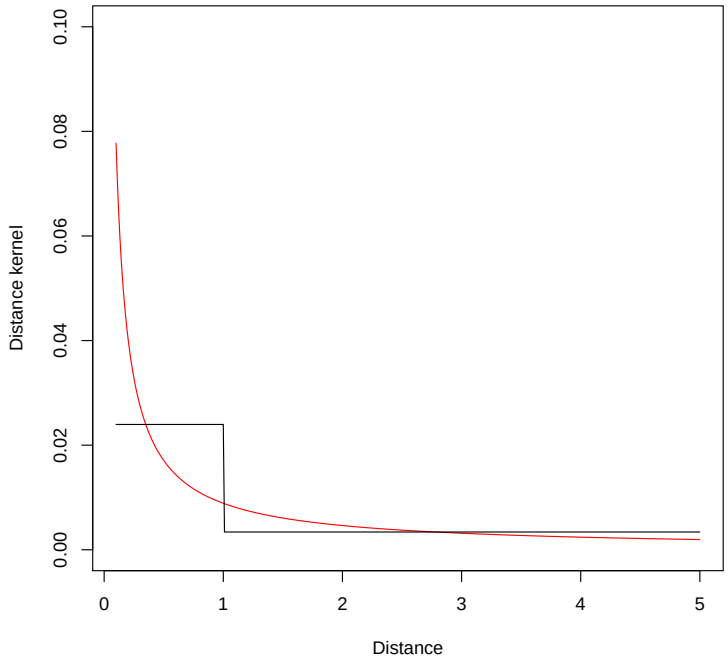

Figure 14: The marginal posterior distribution of the distance kernels. Black line represents the spatial terms of the **surveillance** package, and red line represents the distance kernel function of the **EpiILMCT** package.

We assume that the epidemic starts with an initial infected individual $k = 386$, who has an infection time $I_1 = 0$ and an infectious period of 3 days. We then implement the culling policy within an epidemic simulation study using the `datagen` command to simulate epidemics in a specified small time steps (e.g., a day at a time). This is done by setting the argument `tmax`, and starting each new simulation step with initially infected and removed individuals set according to the epidemic history, and the culling policy implemented at the current time step. This is done using the `initialepi` option. We build a `control.strategy` function to implement the above culling policy using an $\mathcal{SIR}$ distance-based continuous time ILM with power-law kernel and no covariates, in which the infectivity rate given in Equation 2 becomes:

$$\lambda_j(t) = \left(\alpha \sum_{i \in \mathcal{I}(t)} d_{ij}^{-\beta} \right), \qquad \alpha, \beta > 0,$$

with infectious periods assumed to follow a gamma distribution such that $\gamma_i \sim \Gamma(6, \delta)$.

```
R> control.strategy <- function(init.epi, location, inf.time, par.sus,
+    par.ker, delt, cov.sus = NULL, radius) {
+    n <- length(location[, 1])
+    tss <- init.epi
+    cov1 <- cov.sus
+    dis <- as.matrix(dist(location))
+    for (i in 2:length(inf.time)) {
+      mn <- sum(tss[, 4] <= inf.time[i-1])
```

```
+      initial1 <- matrix(tss[1:mn,], ncol = 4, nrow = mn)
+      tss1 <- datagen(type = "SIR", kerneltype = "distance",
+        kernelmatrix = location, distancekernel = "powerlaw",
+        initialepi = initial1, tmax = inf.time[i], suspar = par.sus,
+        transpar = NULL, kernel.par = par.ker, delta = delt,
+        transcov = NULL, suscov = cov1)
+      tss <- tss1$epidat
+      newlyinfected <- tss[which(tss[, 4] > inf.time[i-1] &
+      tss[, 4] <= inf.time[i]), 1]
+      num.infected  <- sum(tss[, 2] != Inf)
+      uninfected <- tss[(num.infected+1):n, 1]
+      for (j in 1:length(newlyinfected)) {
+        mk <- as.integer(which(dis[newlyinfected[j], uninfected] <
+          radius))
+        if (length(mk) > 0) {
+          cov1[uninfected[mk], ] = 0
+        }
+      }
+    }
+    list(tss1, cov1)
+  }
```

Let us assume we have estimates of the model parameters as $\hat{\alpha} = 1.5$, $\hat{\beta} = 4$, and $\hat{\delta} = 2$. Using these estimates, we test the above function for eight values of the radius of the culling policy, and obtain 32 replicated epidemics for each radius setting. The code to achieve this is as follows:

```
R> id.init <- 386
R> inf.period.init <- 3
R> kl <- which(seq_len(625) != id.init)
R> init.epi <- rbind(c(386, inf.period.init, inf.period.init, 0),
+    cbind(kl, rep(Inf, 624), rep(0, 624), rep(Inf, 624)))
R> rr <- seq_len(8)
R> inf.time <- seq(0, 30, by = 1)
R> par.sus <- 1.5
R> par.ker <- 4.0
R> delt <- c(6, 2)
R> sus.cov <- matrix(rep(1, 625), ncol = 1)
R> ninfected <- matrix(0, ncol = 32, nrow = length(rr))
R> numb.culled <- matrix(0, ncol = 32, nrow = length(rr))
R> len.infection <- matrix(0, ncol = 32, nrow = length(rr))
R> for (i in 1:length(rr)) {
+    for (j in 1:32) {
+      epi.cont <- control.strategy(init.epi, location = loc, inf.time,
+        par.sus, par.ker, delt, cov.sus = sus.cov, radius = rr[i])
+      ninfected[i, j] <- sum(epi.cont[[1]]$epidat[, 2] != Inf)
+      numb.culled[i, j] <- n - apply(epi.cont[[2]], 2, sum)
```

```
+      len.infection[i, j] <- max(epi.cont[[1]]$epidat[1:ninfected[i, j],
+        2]) - min(epi.cont[[1]]$epidat[1:ninfected[i, j], 4])
+    }
+  }
```

The output of the above loops is an 8×32 matrices of the number of infected and culled individuals and the length of epidemics for the radius set. We then use the function `apply` from the **base** package (R Core Team 2021) to get the average of each summary at each radius, and plot them versus radius using the following code:

```
R> plot(rr, apply(ninfected, 1, mean), type = "o", ylab = "Number of
+    individuals", xlab = "radius", ylim = c(0, n), pch = 19)
R> lines(rr, apply(numb.culled, 1, mean), type = "o", pch = 19, col = "red")
R> legend("topright", c("Average number of infected individuals", "Average
+    number of culled individuals"), col = c("black", "red"), lty = c(1, 1),
+    pch = c(19, 19))
R> plot(rr, apply(len.infection, 1, mean), type = "o", ylab = "Length of
+    epidemic", xlab = "radius", pch = 19)
```

Figure 15 shows the average number of infected and culled individuals at each radius. We can see that the number of infected individuals tends to decrease dramatically as the radius of the ring increases, levelling off once we have to get around $r = 5$ units. However, the number of culled individuals also increases quite dramatically with increasing the radius of the ring, also levelling off around $r = 7$ units. We can also see from Figure 16 increasing the radius r tends to decrease the length of the epidemic.

Of course, the `control.strategy` function can be easily modified to impose other control strategies. For example, instead of culling within a time step as in the case here, we could allow for (stochastic) delays between infection and culling for surrounding individuals, or allow for only a probability of failure regarding each cull or vaccination event.

D. Comparing computation times to run different models

Here, we compare the effect of population size and the number of infected individuals on the computation time for running the `epictmcmc` function. We considered five population sizes (50, 250, 450, 650 and 850 individuals), and generated three different epidemics using $\mathcal{SIR}$ distance-based continuous time ILMs, via the `datagen` function, resulting in different numbers of infected individuals. These epidemics are categorized into three levels as: small, medium, large defined as epidemics in which the number of infected individuals are less than 25%, between 25% and 50%, or greater than 50% of the population, respectively. Then, we run the `epictmcmc` three times assuming `datatype` = `"known epidemic"`, `"known removal"` with single chain, and `"known removal"` with three chains, updating the infection times in blocks of size five.

Figure 17 shows the computation times in hours for running the `epictmcmc` function on the above epidemics to obtain 150,000 MCMC samples. The computation times were approximated on the basis of running ten iterations, as our concern here is just to see to estimate the effect of population size and number of infected individuals upon computation time. We

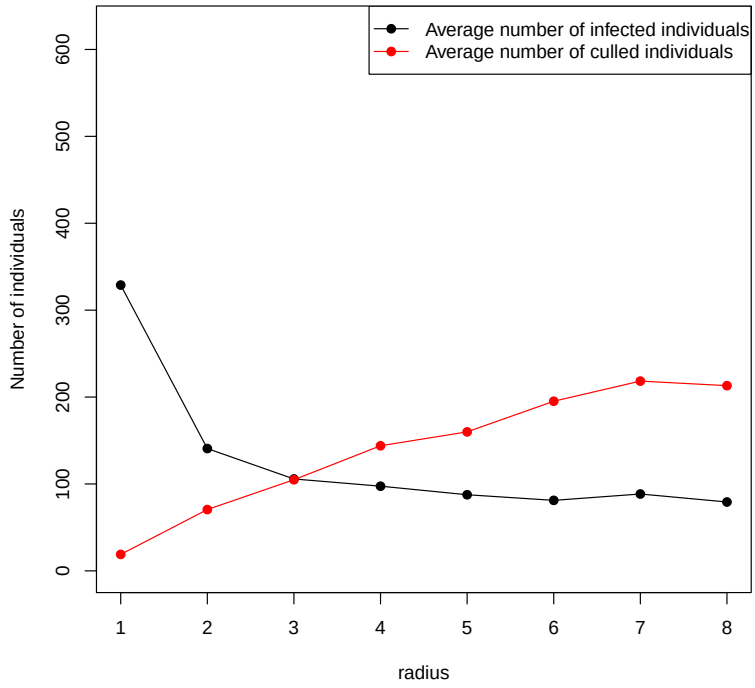

Figure 15: The average number of infected (black) and culled (red) individuals at each radius.

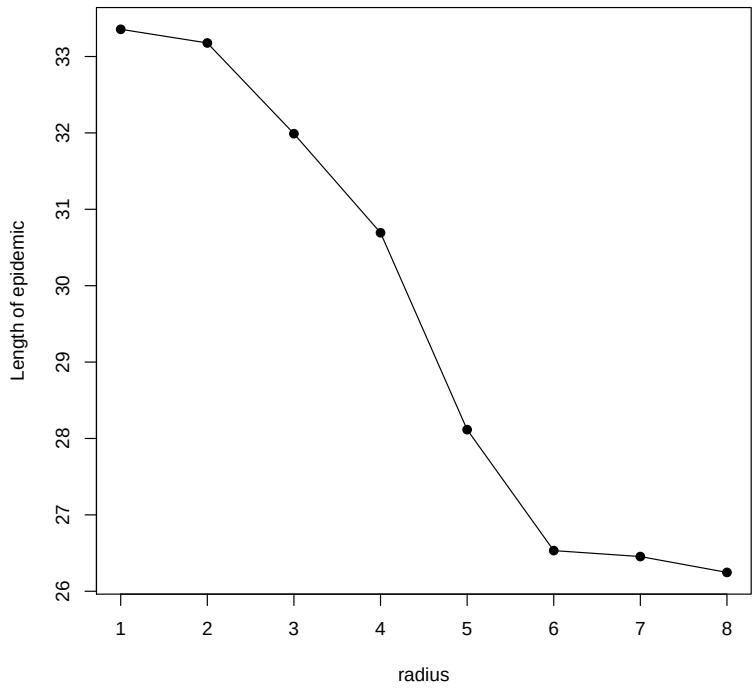

Figure 16: The average length of epidemics at each radius.

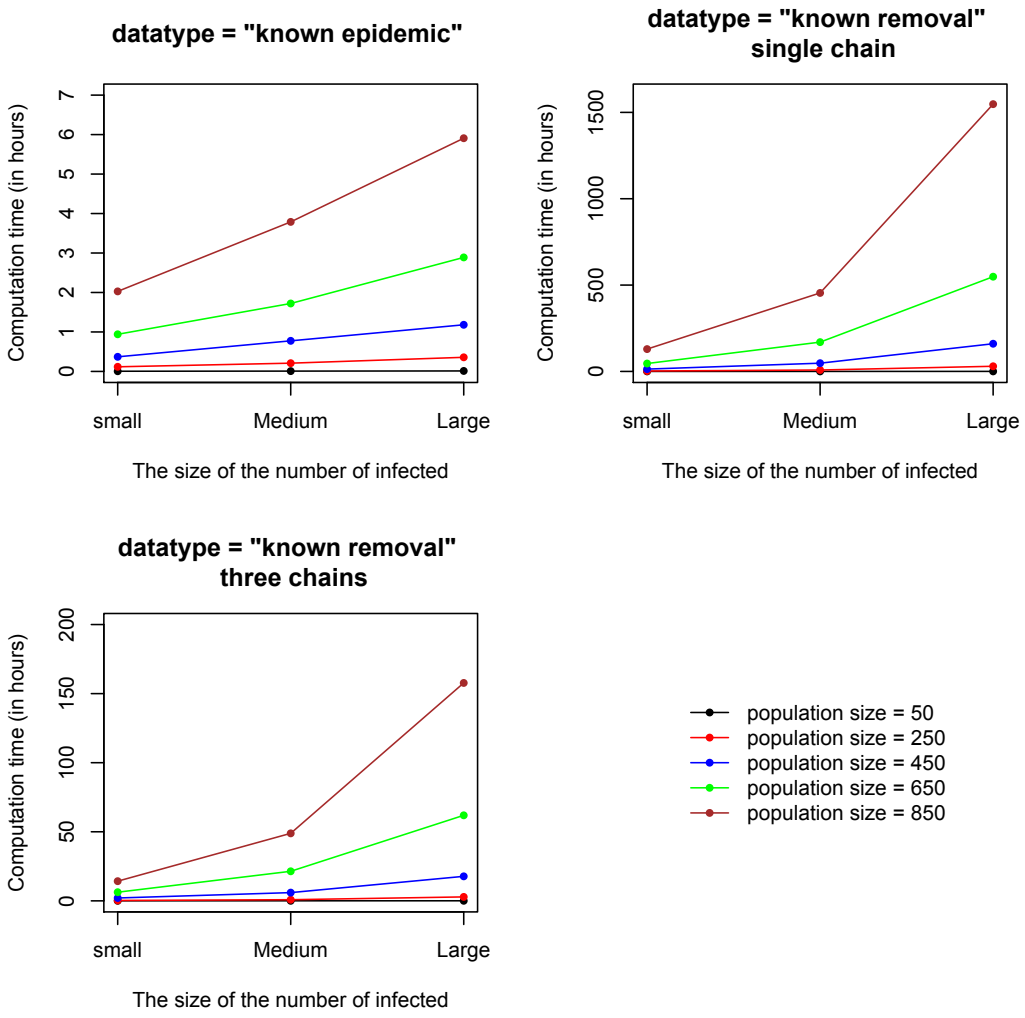

Figure 17: Approximate computation times of running the `epictmcmc` function for fitting different epidemic data sets, with different population sizes and number of infected individuals, using $\mathcal{SIR}$ distance-based continuous time ILMs under three scenarios: fully observed epidemics, partially observed epidemics with a single MCMC chain, and three MCMC chains.

observed strong correlation between the population sizes and number of infected individuals in all of the considered analysis scenarios.

We see that under the fully observed epidemic assumption (`datatype = "known epidemic"`), the function `epictmcmc` can be performed in reasonable time for all scenarios. However, computation time becomes an issue for partially observed epidemics (`datatype= "known removal"`) when updating the infection times in turn in a single chain. Larger epidemics with larger population sizes were estimated to take more than four weeks to obtain 150,000 MCMC samples. Computation time is greatly reduced by running `epictmcmc` over multiple chains and updating infection times in blocks of size five. For example, with an epidemic in a population of size was 850 individuals, and almost all of individuals infected, the computation time was reduced from approximately 1548 hours (≈65 days) to 157 hours (≈7 days).

Affiliation:

Waleed Almutiry
Department of Mathematics
College of Science and Arts in Ar Rass
Qassim University
Qassim, Saudi Arabia
E-mail: wkmtierie@qu.edu.sa

Vineetha Warriyar K V
Sport Injury Prevention Research Centre
Faculty of Kinesiology
University of Calgary
Calgary, AB Canada
E-mail: vineethawarriyar.kod@ucalgary.ca

Rob Deardon
Faculty of Veterinary Medicine
Department of Mathematics and Statistics
University of Calgary
Calgary, AB Canada
E-mail: robert.deardon@ucalgary.ca
URL: http://people.ucalgary.ca/~robert.deardon/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
June 2021, Volume 98, Issue 10 *Submitted:* 2018-03-30
doi:10.18637/jss.v098.i10 *Accepted:* 2020-01-25