SPARSEMODr: Rapid simulations of spatially explicit and stochastic models of infectious disease

Joseph R Mihaljevic[1*]

Seth Borkovec[1]
Saikanth Ratnavale[1]
Toby D Hocking[1]
Kelsey E Banister[1]
Joseph E Eppinger[1]
Crystal Hepp[1,2]
Eck Doerry[1]

1. School of Informatics, Computing, and Cyber Systems, Northern Arizona University, Flagstaff, AZ 86011;
2. Pathogen and Microbiome Institute, Northern Arizona University, Flagstaff, AZ 86011;

* Corresponding author; e-mail: joseph.mihaljevic@nau.edu.

Manuscript type: Application
Keywords: epidemiological model, spatial disease models, R, C++, disease ecology, host-pathogen interactions

Prepared using LaTeX.

Summary

1. Simulating the dynamics of realistically complex models of infectious disease is conceptually challenging and computationally expensive. This results in a heavy reliance on customized software and, correspondingly, lower reproducibility across disease modeling studies.
2. SPARSEMOD stands for **SPA**tial **R**esolution-**SE**nsitive **M**odels of **O**utbreak **D**ynamics. The goal of our project, encapsulated by the `SPARSEMODr` package for R package, is to offer a framework for rapidly simulating the dynamics of stochastic and spatially-explicit models of infectious disease for use in pedagogical and applied contexts.
3. We outline the universal functions of our package that allow for user-customization while demonstrating the common work flow.
4. `SPARSEMODr` offers an extendable framework that should allow the open-source community of disease modelers to add new model types and functionalities in future releases.

1 Introduction

The recent emergence of SARS-CoV-2 has reinforced the strong role that mathematical models of disease spread play in understanding pathogen transmission and in designing effective public health interventions (Ferguson *et al.*, 2020; Tian *et al.*, 2020; Saad-Roy *et al.*, 2020). Models of infectious disease transmission vary widely in their structural form and complexity (Keeling & Rohani, 2008). Classical models take the form of ordinary differential equations describing "compartments" of the population (e.g., susceptible versus infectious), but even these models can become quite complex, containing numerous equations that might, for instance, account for heterogeneities in the host population or for the progression of pathogen-induced disease through various stages. Some of the most complex - and perhaps realistic - disease models explicitly account for spatial dynamics, referred to as meta-population models.

In meta-population models of disease, distinct host populations are delineated and are explicitly situated geographically, such that movement of host individuals between the populations affects transmission (Rohani *et al.*, 1999; Lachiany & Stone, 2012; Ferrari *et al.*, 2010; Pei *et al.*, 2020). These spatial models are therefore used to understand how host or pathogen movement influences within-population transmission dynamics and the regional patterns of epidemics that emerge by aggregating the distinct populations. For example, spatial models can help us understand how quickly a pathogen is expected to travel across a landscape once it emerges; they can be used to understand the synchrony of outbreaks in the face of spatial heterogeneity; and they can be used to simulate spatial intervention strategies (e.g., targeted vaccination). In this way, spatial models are often necessary to explain large-scale patterns of disease transmission that cannot be captured by simpler, non-spatial models (Eggo *et al.*, 2021). Yet the use of spatial disease models in practice can be hindered by their level of complexity (Riley *et al.*, 2015; Willem *et al.*, 2017), meaning that spatial models may be underutilized in educational settings and in applied situations.

Spatial disease models may become computationally burdensome due to the necessity of tracking the values of multiple state variables (i.e., host compartments) within each distinct population over time, while also simulating explicit movement dynamics between host populations over time (Riley, 2007; Riley *et al.*, 2015). We believe that this leads to two problems. First, spatial models must be custom-coded, which may reduce the reproducibility of results between studies, especially if model code is not open-source. We are unaware of any open-source frameworks that allow users to simulate spatial disease models in a flexible way without having to code their own model. One could argue that spatial disease models are complex enough that they should only be constructed and analyzed by highly qualified experts; however, this seems philosophical and debatable. There are distinct advantages of imagining a software environment in which many spatial models of various structural form could be simulated with some universal, but user-controlled constraints. In such a scenario, researchers could cite the universal parameters, which would increase reproducibility and facilitate peer-review.

A second problem is that because spatial models require customized software and the availability of computational resources, this may limit the opportunity for hands-on learning in the classroom. From experience, having students code their own spatial model or attempt to read and understand example code from a spatial model can be extremely time-intensive. This may lead some instructors to rely on lecture-style learning to convey the importance of spatial disease dynamics, depriving students of critical experiences in deriving their own understanding through manipulating the models. Therefore, it would be very useful to have a software environment that allows students to simulate spatial models "out-of-the-box" while still deriving an understanding of fundamental concepts in epidemiology.

Here we introduce the `SPARSEMODr` package, which allows users to rapidly simulate complex models of infectious disease in a spatially-explicit and stochastic environment. The first release of this package contains two model structures, each based on coupled systems of ordinary differential equations. First, we supply a classic Susceptible-Exposed-Infectious-Removed (SEIR) model with host demography (births and deaths), which can be used in the classroom to understand the cycling of disease outbreaks and the

spatial synchrony of these cycles in a spatially-explicit context. This model is not fully described herein; however, our package vignettes provide a detailed model description and coded examples. Second, and described in detail below, we supply a more complex and specific model of SARS-CoV-2 transmission and disease progression through the hospital system (Gel *et al.*, 2020). Our group has been using this model in an applied context to make projections of hospitalization on a county-specific basis in Arizona, but the model could also be used in a pedagogical context. Importantly, the `SPARSEMODr` package has a set of conventions (i.e., universal constraints) that dictate how hosts move between populations, that allow for multiple sources of stochasticity, that define the effects of host density on transmission (i.e., frequency- versus density-dependent transmission), and that allow users to specify time-varying parameters (e.g., time-varying transmission rates). In this way the package is extendable: in future releases, more model structures can be added that follow the same core conventions. We hope that the package will increase reproducibility in the field of spatially-explicit disease modeling, while also providing a straightforward way for students and instructors to learn about these perhaps otherwise inaccessible model types.

2 Universal features of `SPARSEMODr` disease models

All of our models are coded in C++ and use the Rcpp package to conveniently wrap the functions into the R computing environment (Eddelbuettel, 2013). We therefore take advantage of the speed offered by C++ and the user-friendliness offered by the higher-level language, R. The output of each model is a data frame that includes the value of each state variable per time step per population, as well as the number of new "events" per time step per population (e.g., new exposure events or recovery events). We supply detailed vignettes that describe different use-cases of the `SPARSEMODr` package on our website: `https://sparsemod.nau.edu/rpkg/`. Here we describe the key universal features of these spatially-explicit and stochastic disease models.

2.1 Stochastic dynamics

Demographic stochasticity All of our models implement a form of demographic stochasticity, which represents the effects of probabilistic events that befall individuals in a population and that can affect epidemic trajectories. These random processes are especially important early in outbreaks and in small host populations (Keeling & Rohani, 2008). To implement demographic stochasticity, we simulate the differential equations forward one day at a time using a Gillespie-style algorithm known as the tau-leaping algorithm (Gillespie, 2001). In addition to accounting for important random processes, this approach has several practical advantages. First, this simulation approach is more computationally efficient compared to some numerical integration techniques (Ganyani *et al.*, 2021). Second, this approach allows us provide integer outputs that represent the number of new "events" that occurs on each day of the simulation. Whereas the number of new events would have to be estimated in numerical integration of continuous-time differential equations, because the output values would be real numbers. One downside to the tau-leaping approach is that it forces the user to work with integers, and simulating the equations can be more unstable when the integers are very small. However, this is simply an issue of scaling; the state variables and model parameter values can easily be re-scaled to improve the stability of the simulations.

Stochastic transmission process We implement daily stochastic variation in the transmission rate that scales with the number of infectious individuals in the focal population. In other words, as the number of infectious individuals increases, the variation in transmission rate decreases, emphasizing that stochasticity has larger effects in smaller populations (i.e., larger effects when there are few infectious individuals) (Keeling & Rohani, 2008). This type of stochasticity can account for super-spreader events, which have disproportionate effects early in the epidemic. To implement this stochasticity, we draw a random variate from a normal distribution with a mean of zero and a standard deviation of `stoch_sd`, and we call this random variate `noise`. We calculate the total number of infectious individuals as `infect_sum`. The functional form

of stochasticity is then:

$$\beta_{\text{realized}} = \left| \beta_t * \left(1 + \frac{\text{noise}}{\sqrt{\text{infect\_sum}}} \right) \right|. \tag{1}$$

2.2 Spatial dynamics

Our models allow migration between populations in the meta-population to affect local and regional transmission dynamics. In general, susceptible individuals in a focal population can become exposed to the pathogen by infectious "visitors" from other populations or by infectious visitors from outside of the meta-population ("immigrants"). Similarly, susceptible individuals can visit a different population within the meta-community and these susceptible travelers may become exposed by infectious individuals in those other populations.

The user can control the per-capita movement rate m of susceptible and infectious hosts. The probability of moving to any specific population is controlled by a simple, distance-based dispersal kernel:

$$p_{i,j} = \frac{1}{\exp(d_{i,j}/\text{dist\_param})}. \tag{2}$$

Here $p_{i,j}$ is the probability of moving from population j to population i, and $d_{i,j}$ is the euclidean distance between the two populations. Larger values of the `dist_param` constant make it more likely for hosts to travel farther distances. In future releases we could include the option of using more complicated kernels, like gravity kernels, which assume movement probability is also dependent on the local size of population i.

To determine which individuals will move to which outside population, we draw from a multinomial probability distribution using the $p_{i,j}$ values. Once individuals are assigned to their new, temporary populations, transmission can occur dependent upon the local composition of infectious individuals. Thus, transmission occurs within a population and then additional transmission can occur after hosts commute.

The model also allows immigrants, who do not usually reside in the meta-population, to commute to the meta-population each day. As an analogy, imagine modeling the meta-population within a state, but wanting to model the effects of out-of-state persons that visit the State of interest. In this case, the user can define the parameter `imm_frac`, which is the proportion of the focal population that may constitute visitors on any given day. For example if for a given focal population, the population size is 1000 hosts, and `imm_frac` $= 0.05$, an average of 50 temporary immigrants may arrive on a given day. The exact number of these immigrants arriving on a given day is determined by drawing from a Poisson distribution. Then, the number of infectious visitors from this pool of immigrants is assumed to be proportional to the number of infectious residents in the focal population. In other words, we assume that the pathogen is present in "non-resident" populations at similar prevalence as the focal population. The exact number of infectious immigrants is again determined by a Poisson draw. After immigrants arrive at the focal population, transmission between susceptible residents and infectious immigrants is determined, and the immigrants then leave the population. In other words, immigrants are commuters and only visit for one day at a time.

2.3 Time-varying parameters

The `SPARSEMODr` models allow users to change transmission dynamics over time by specifying time-varying parameters. Time-varying parameters can take on unique values per day, and we provide two ways for users to specify these changes. First, users can specify "time windows" over which the parameter values change linearly from a starting value to an ending value. Second, users can supply a vector of values for a given parameter such that it takes on a user-specified value per day in the simulation (see sec 3.1 for more details on time windows).

All models allow the transmission rate to vary over time; however, because the value of the transmission rate can be hard to interpret or estimate over time, we instead allow the user to input a time-varying reproductive number. Note that this not the same as the instantaneous reproductive number, so we call this

value the time-varying R_0. In each model we use the value of R_0 at each time point to back-calculate a time-specific value of the transmission rate ($\beta_{(t)}$), but we do not adjust for the current proportions of susceptible or recovered hosts. Hence, it is different from the instantaneous reproductive number, R_t (Gostic *et al.*, 2020). In this way, we are assuming that time-varying R_0 is effectively changing only due to changes in the $\beta_{(t)}$ term, which encapsulates the effective contact rate among individuals and the probability of transmission, given contact between a susceptible and infectious individual. For simple models, the calculation of $\beta_{(t)}$ from a time-varying R_0 value is straightforward algebra; e.g., for a classic SIR model with host demography: $\beta_{(t)} = R_{0(t)}(\gamma+\mu)$, where γ is the recovery rate and μ is the natural mortality rate. For more complex models with many state variables, this becomes more complicated. Therefore, to back-calculate the $\beta_{(t)}$ parameter from the time-varying R_0 value, we derive the equation for time-varying R_0 for each `SPARSEMODr` model. We implement a root-finding algorithm with the Brent-Dekker method using the GNU Scientific Library to calculate $\beta_{(t)}$ in our underlying C++ code, which is fast, robust, and does not require an initial guess (Brent, 2013). Thus, the user inputs the time-varying R_0 value, and we calculate the value of $\beta_{(t)}$ on the back-end.

In the model of SARS-CoV-2 transmission, we also allow a number of other relevant parameters to fluctuate through time, which we describe below. Therefore, each model that is developed in `SPARSEMODr` could include any number of time-varying parameters following the universal conventions.

2.4 Frequency- and density-dependent transmission

In `SPARSEMODr` models, the transmission process can be described as frequency-dependent, where contact rates are invariable to population density, or density-dependent, where contact rates depend on population density (Hopkins *et al.*, 2020).

To model frequency-dependent transmission, we scale the time-specific transmission rate by the local population size $\beta_{scaled} = \beta_{(t)}/N_i$, where N_i is the size of a given focal population. The day-specific transmission rate $\beta_{(t)}$ is back-calculated from the user-provided value of time-varying R_0 (above).

We also allow the user to specify a relationship between the realized transmission rate in a population and the population's density. For instance, for some pathogens, we may expect the effective contact rate to increase with increasing population density (i.e., density-dependent transmission). We therefore allow the user to specify a non-linear Monod equation (Monod, 1949) to describe the relationship between host population density and the transmission rate $\beta_{(t)}$. The Monod equation is:

$$\beta_{\text{realized}} = \beta_{(t)} \frac{\mathrm{D}}{K+\mathrm{D}}, \tag{3}$$

where $\beta_{(t)}$ becomes the maximum possible transmission rate for that day. `D` is the density of the focal host population (N_i/census_area), and K is a constant that controls the effect of density, which is the user-specified value of `dd_trans_monod_k`. We again scale the realized value of the transmission rate by the population size: $\beta_{scaled} = \beta_{\text{realized}}/N_i$.

3 Work flow

Here we demonstrate the work flow of the `SPARSEMODr` package using our version of the COVID-19 model with frequency-dependent transmission, described by the following equations (see also Gel *et al.* (2020)):

$$
\begin{aligned}
\frac{dS}{dt} &= -\beta_t \lambda_t S \\
\frac{dE}{dt} &= \beta_t \lambda_t S - \delta_1 E \\
\frac{dI_a}{dt} &= \delta_1 \rho_1 E - \gamma_a I_a \\
\frac{dI_p}{dt} &= \delta_1 (1-\rho_1) E - \delta_2 I_p \\
\frac{dI_s}{dt} &= \delta_2 I_p - \delta_3 I_s \\
\frac{dI_b}{dt} &= \delta_3 (1-\rho_2-\rho_3) I_s - \gamma_b I_b \\
\frac{dI_h}{dt} &= \delta_3 \rho_2 I_s - \delta_4 I_h \\
\frac{dI_{c1}}{dt} &= \delta_3 \rho_3 I_s + \delta_4 \rho_4 I_h - \delta_5 I_{c1} \\
\frac{dI_{c2}}{dt} &= \delta_5 (1-\rho_5) I_{c1} - \gamma_c I_{c2} \\
\frac{dD}{dt} &= \delta_5 \rho_5 I_{c1} \\
\frac{dR}{dt} &= \gamma_a I_a + \gamma_b I_b + \gamma_c I_{c2} + \delta_4 (1-\rho_4) I_h
\end{aligned}
\tag{4}
$$

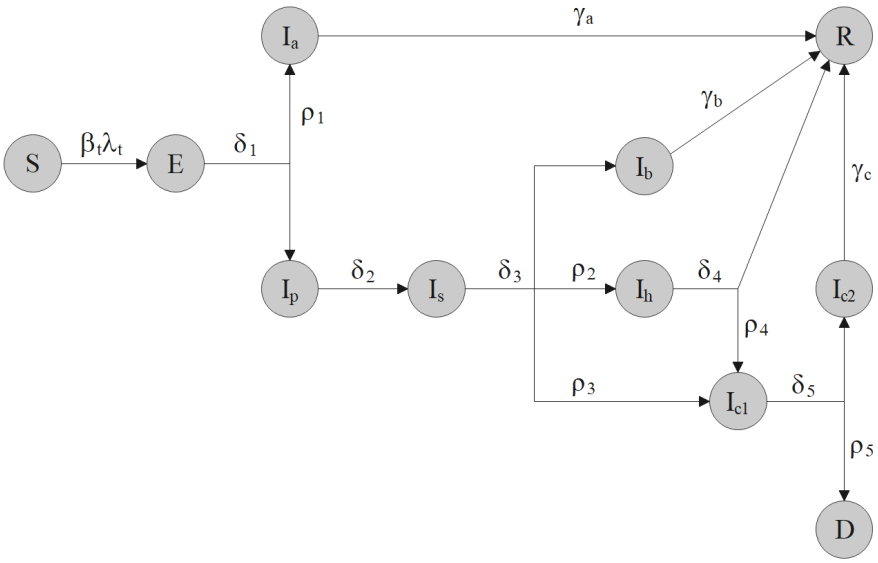

Figure 1: COVID-19 model schematic.

Here, λ_t represents a component of the force of infection, given by:

$$\lambda_t = \frac{\omega_1 I_a + I_p + I_s + I_b + \omega_2 (I_h + I_{c1} + I_{c2})}{N - D}. \tag{5}$$

Definitions of state variables and model parameters are displayed in the tables below, which also show the corresponding model inputs to the `SPARSEMODr` package. We provide default parameter values as a guide in the package documentation, but all of them can be user-specified. The package also conducts parameter validation steps via warnings and errors to ensure specified parameters are within feasible limits. Note that we also allow individuals in the S, I_a, I_p, and I_s compartments to move between populations, controlled the per-capita movement rate m. However, these dynamics are not explicitly represented in the model equations.

Table 1: COVID-19 model state variables

State Variable	Description	Corresponding model input
S	Number of susceptible individuals	`input_S_pops`
E	Number of exposed individuals	`input_E_pops`
I_a	Number of asymptomatic individuals	`input_I_asym_pops`
I_p	Number of pre-symptomatic individuals	`input_I_presym_pops`
I_s	Number of mildly symptomatic individuals	`input_I_sym_pops`
I_b	Number of mildly symptomatic individuals on bed rest at home	`input_I_home_pops`
I_h	Number of hospitalized individuals	`input_I_hosp_pops`
I_{c1}	Number of individuals in the ICU	`input_I_icu1_pops`
I_{c2}	Number of individuals in the recovery (step-down) ICU	`input_I_icu2_pops`
D	Number of deceased individuals	`input_D_pops`
R	Number of recovered individuals	`input_R_pops`

Table 2: COVID-19 model parameter descriptions

Parameter	Description	Corresponding model input
β_t	Time-varying transmission rate	Internally calculated
ω_1	Proportion reduction in transmission for asymptomatic folks	`frac_beta_asym`
ω_2	Proportion reduction in transmission for hospitalized folks	`frac_beta_hosp`
N	Total number of individuals in population	`input_N_pops`
δ_1	Transition rate: exposed to pre-symptomatic	`delta`
δ_2	Transition rate: pre-symptomatic to symptomatic	`recov_p`
δ_3	Transition rate: symptomatic to home or regular hospital bed or ICU	`recov_s`
δ_4	Transition rate: regular hospital bed to home or ICU	`recov_hosp`
δ_5	Transition rate: ICU to step-down ICU or decease	`recov_icu1`
γ_a	Recovery rate: asymptomatic	`recov_a`
γ_b	Recovery rate: home bed	`recov_home`
γ_c	Recovery rate: step-down ICU	`recov_icu2`
ρ_1	Fraction of exposed that transition to asymptomatic	`asym_rate`
ρ_2	Fraction of symptomatic that transition to hospital bed	`hosp_rate`
ρ_3	Fraction of symptomatic that transition directly to ICU bed	`sym_to_icu_rate`
ρ_4	Fraction of hospitalized that transition to ICU	`icu_rate`
ρ_5	Fraction of patients in ICU that die of disease	`death_rate`

3.1 Process overview

Our software package runs the spatial, stochastic modeling framework across a user-specified grid of populations. Figure 2 shows a simulated case-study using the COVID-19 model described above. In this case, we

simulated populations that are scattered across a spatial lattice; one could imagine these are local communities situated within counties, within a state, such that the "Regions" represent counties (Fig. 2(a)). Across the whole meta-population (i.e., state), we impose a time-varying R_0, such that each local population experiences the same pattern of time-varying R_0 (Fig. 2(b)). In our spatial modeling framework, we simulate the set of ordinary differential equations (above) within each local population, allowing movement (as described above) during each time step of one day. This means that we can track the spread of the disease in each population individually ((Fig. 2(c)), and we can aggregate patterns in local populations to higher spatial scales (e.g., "Regions"; Fig. 2(d)) to look at higher-level, emergent patterns. Below we describe the necessary setup and declarations to run the spatial simulations in `SPARSEMODr`.

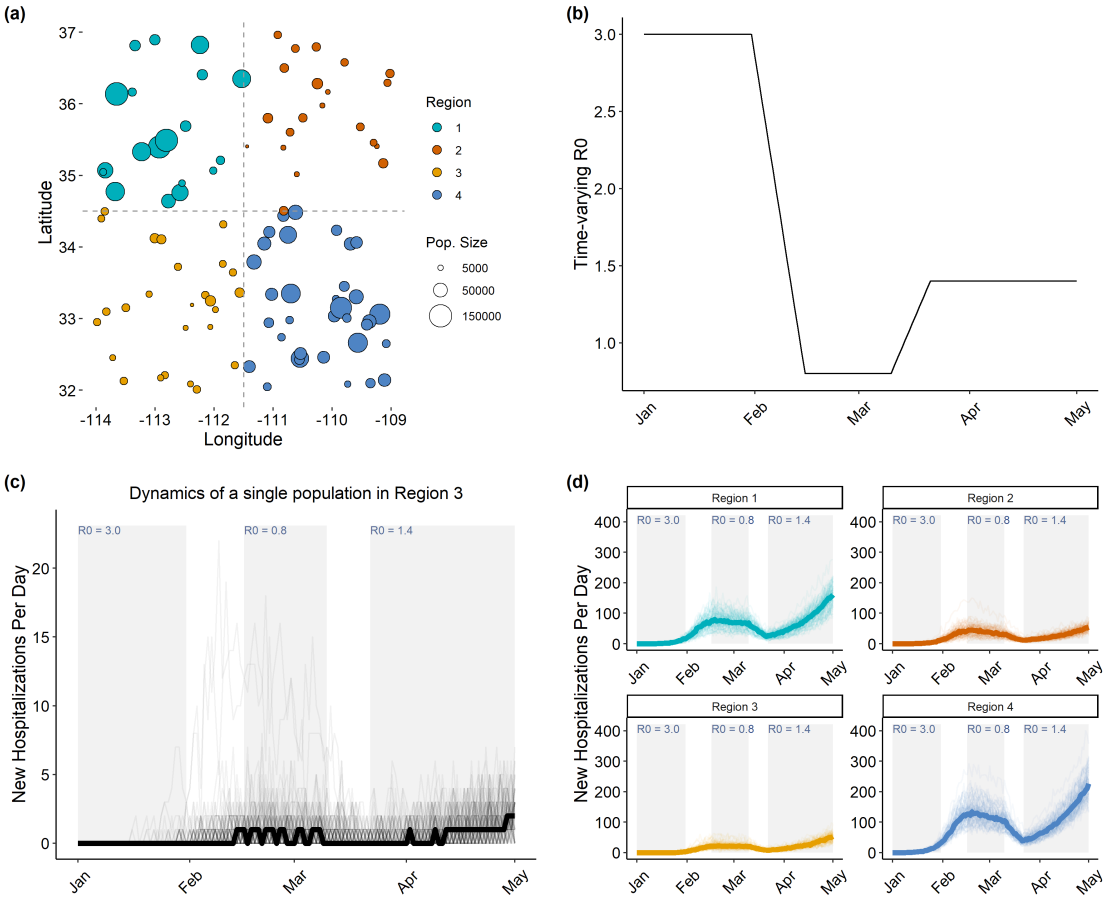

Figure 2: Overview of stochastic, spatial modeling scheme using simulated data and the COVID-19 `SPARSEMODr` model. (a) Simulated host populations of varying abundance on a spatially-explicit grid. (b) Imposing a pattern of time-varying R_0 across all of the local populations. Note that hospitalization rate may also be an underlying change over time. (c) Pattern of new hospitalizations over time from a particular population. The lighter lines are individual realizations of the stochastic model, while the dark line is the median trajectory of these realizations. (d) Aggregating local patterns to regional scales to explore differences in epidemic trajectories across regions. Light and dark lines as in panel (c).

3.2 Declaring initial conditions and constant parameters

Users must specify the initial conditions of the state variables. To initiate the epidemic, at least one of the following vectors must be supplied with a value of greater than one for at least one population: `input_E_pops`, `input_I_asym_pops`, `input_I_presym_pops`, `input_I_sym_pops`, `input_I_home_pops`,

`input_I_hosp_pops`, `input_I_icu1_pops`, or `input_I_icu2_pops`. Any of these population parameters not supplied will be assumed to be a vector of zeroes. Moreover, either a vector of `input_N_pops` or a vector of `input_S_pops` must be supplied. Let's take the following data set for as inputs for the our examples:

```
N_pops <- rep(5000, 10)           # 10 populations each with 5000 individuals.
E_pops <- c(0,1,0,3,2,0,14,3,0,0) # Number of initially exposed in each pop.
S_pops <- N_pops - E_pops
```

The user must also declare some inputs that are universal to all `SPARSEMODr` models: `input_dist_mat` is a matrix that specifies the distance between populations; `input_realz_seeds` is a vector of seeds for the random realizations of the model; `stoch_sd` is the standard deviation of the stochastic transmission rate, as described above; `trans_type` specifies the type of transmission type (frequency- or density-dependent, see above); `input_census_area` is the spatial area of each population (only required when transmission is density-dependent); `dd_trans_monod_k` the parameter controlling density's effect on transmission, as described above (only required when transmission is density-dependent); and `input_tw` a time window object, which we will describe in the next section. More information on these parameters is found in our package documentation.

The next step is to populate the model-specific "control" object, which is a special R class defined in our package. Each model in the package must have its own control class that includes the model-specific parameters and initial conditions of the state variables. This also lets the system know which model from the library of `SPARSEMODr` models will be run.

A simple example, given the initial conditions above is as follows. In this case, all of the constant model parameters would use default values, except the two specified.

```
my_covid19_control <- SPARSEMODr::covid19_control(
  input_S_pops = S_pops, # susceptible population counts
  input_E_pops = E_pops, # exposed population counts
  asym_rate    = 0.4,    # fraction of exposed that become asymptomatic
  recov_icu1   = 0.125)  # average ICU recovery rate, i.e., 8 days (1/8)
```

Also, in this case, because `input_N_pops` was not provided, the package internally assumes `input_N_pops = input_S_pops + input_E_pops`. `covid19_control()` returns a named list of vectors that must be supplied when running the model. Error and warning messages help the user with understanding which parameters are necessary and which are suggested.

3.3 Declaring time-varying parameters with the `time-windows` object

A `time_windows` object is required to specify the time-varying parameters (or whether these will be constant in the simulation). Importantly, there are two ways to specify these "time windows": (1) daily, or (2) start and end dates of the window. For the "daily" method, the parameter vectors must be of size equal to the number of days in the simulation. For the start/end method, values for each parameter are assigned at the beginning and end of the time window. The model then implements a linear interpolation to assign daily values; in other words, the parameter values change linearly from the starting value to the ending value over the number of days within the time window.

Here is an example that was used to generate the time-varying R_0 pattern of Figure 2(b). In this case, we specify the start and end dates of our time windows, that each vary in the number of days included. Then we specify the values of each potentially time-varying parameter at the start of each time window. The system internally populates vectors of length equal to the number of days represented by the whole simulation, where parameters are assumed to change linearly within the time windows. Below, we show examples and dates are formatted using the `lubridate` package (Grolemund & Wickham, 2011).

```
# Function to specify all of the required parameters in a single time-window
# In this example, only r0 is variable between the time periods
one.window <- function(
    start_dates,    #start of time window (date class)
    end_dates,      #end of time window (date class)
    r0,             #value of time-varying R0
    dist_param=150, #controls dispersal kernel (see Key Features vignette)
    m=0.1,          #per-capita movement rate, i.e., move every 10 days on avg.
    imm_frac=0){    #zero outside immigration
    data.frame(r0, start_dates, end_dates, dist_param, m, imm_frac)
}

library(lubridate) # for mdy()
time.window.args <- rbind(# Specify the components of 5 time windows
  one.window(mdy("1-1-20"),  mdy("1-31-20"), r0=3.0),
  one.window(mdy("2-1-20"),  mdy("2-15-20"), r0=0.8),
  one.window(mdy("2-16-20"), mdy("3-10-20"), r0=0.8),
  one.window(mdy("3-11-20"), mdy("3-21-20"), r0=1.4),
  one.window(mdy("3-22-20"), mdy("5-1-20"),  r0=1.4)
)

# Populate the required object of class time_windows
my_tw <- do.call(SPARSEMODr::time_windows, time.window.args)
```

The package documentation gives additional examples of how to flexibly structure these time window objects to allow parameters to vary over time.

3.4 Running stochastic realizations in parallel

We suggest running stochastic realizations of the `SPAREMODr` spatial models in parallel for efficiency. We have therefore created a function, `model_parallel()`, that relies upon the `future` package in R to use R-level parallelization and to speed up run times (Bengtsson, 2020). The `model_parallel()` function also compiles the output of the individual model runs into a user-friendly data frame.

As an example:

```
# Specify the number of realizations to run:
n_realz <- 75

# Specify unique seeds to run the realizations.
## Note, realizations with the same seeds will produce
## equivalent output
my_realz_seeds <- 1:n_realz

# Run the model in parallel and store the output:
model_output <- SPARSEMODr::model_parallel(
                                    input_dist_mat = dist_mat,
                                    input_census_area = census_area,
                                    input_tw = my_tw,
                                    input_realz_seeds = my_realz_seeds,
```

```
                                    control = my_covid19_control,
                                    ...universal_model_params...)
```

Here "...universal_model_params..." represents the universal model parameters, such as trans_type or stoch_sd, as described above an in our package documentation. Note that dist_mat is a pairwise distance matrix, and census_area is a vector of population areas (e.g., in km^2). See the package vignettes for examples of creating these two inputs. Since model_parallel() produces a data frame, the output can easily be subset and summarized at the user's discretion for plotting. Our package vignettes provide coding examples for subsetting and plotting output (e.g., Figure 2) using tidyverse packages.

4 Conclusion

The SPARSEMODr package therefore allows for advanced simulations of disease models that are both stochastic and spatially explicit. The package could be used in pedagogical contexts to demonstrate fundamental concepts in epidemiology, such as spreading waves of transmission or spatial (a)synchrony of epidemics. In applied contexts, parameterized models could be simulated in realistic spatial lattices to project transmission dynamics under various scenarios of population movement. We envision multiple extensions to our package in the future. First, more model structures can be added, including vector-borne disease models. For package contributors, a model would have to be written in C++ and follow the conventions herein: tau-leaping algorithm, commuter-model of host (or vector) movement between sub-populations, outputs as data frames, etc. Our team would provide guidance and approve all outside contributed model structures via pull and merge requests on Github. Second, we plan to introduce a graphical user-interface that allows users to dynamically simulate toy models under various assumptions. Such an interface could aid in understanding the effects of targeted interventions on disease control. Given the open-source nature of this R package, we hope that the community of epidemiological modelers will help us to refine the package, make it more accessible to broad audiences, add vignettes and case-studies, and add functionality that makes the package more meaningful in applied contexts, such as wildlife conservation or human public health.

5 Author Contributions

JRM, CH, and ED designed the project. JRM, TDH, SB, SR, KEB, and JEE contributed code and methods. JRM, TDH, and SB designed the R package structure. JRM, SB, and SR wrote the first draft of this manuscript, and all authors contributed to manuscript revisions.

6 Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 2028629.

References

Bengtsson, H. (2020) *future: Unified Parallel and Distributed Processing in R for Everyone*. R package version 1.18.0.

Brent, R.P. (2013) *Algorithms for minimization without derivatives*. Courier Corporation.

Eddelbuettel, D. (2013) *Seamless R and C++ Integration with Rcpp*. Springer, New York. ISBN 978-1-4614-6867-7.

Eggo, R.M., Dawa, J., Kucharski, A.J. & Cucunuba, Z.M. (2021) The importance of local context in covid-19 models. *Nature Computational Science*, **1**, 6–8.

Ferguson, N., Laydon, D., Nedjati Gilani, G., Imai, N., Ainslie, K., Baguelin, M., Bhatia, S., Boonyasiri, A., Cucunuba Perez, Z., Cuomo-Dannenburg, G. *et al.* (2020) Report 9: Impact of non-pharmaceutical interventions (npis) to reduce covid19 mortality and healthcare demand.

Ferrari, M.J., Djibo, A., Grais, R.F., Bharti, N., Grenfell, B.T. & Bjornstad, O.N. (2010) Rural-urban gradient in seasonal forcing of measles transmission in Niger. *Proceedings of the Royal Society B: Biological Sciences*, **277**, 2775–2782.

Ganyani, T., Faes, C. & Hens, N. (2021) Simulation and analysis methods for stochastic compartmental epidemic models. *Annual Review of Statistics and Its Application*, **8**, 69–88.

Gel, E.S., Jehn, M., Lant, T., Muldoon, A.R., Nelson, T. & Ross, H.M. (2020) Covid-19 healthcare demand projections: Arizona. *PloS one*, **15**, e0242588.

Gillespie, D.T. (2001) Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, **115**, 1716–1733.

Gostic, K.M., Mcgough, L., Baskerville, E., Abbott, S., Joshi, K., Tedijanto, C., Kahn, R., Niehus, R., Hay, J., Salazar, P.D., Meakin, S., Munday, J., Bosse, N.I., Sherrat, K., Robin, N., White, L.F., Huisman, J.S., Stadler, T., Wallinga, J., Funk, S., Lipsitch, M. & Cobey, S. (2020) Practical considerations for measuring the effective reproductive number , R t.

Grolemund, G. & Wickham, H. (2011) Dates and times made easy with lubridate. *Journal of Statistical Software*, **40**, 1–25.

Hopkins, S.R., Fleming-Davies, A.E., Belden, L.K. & Wojdak, J.M. (2020) Systematic review of modelling assumptions and empirical evidence: Does parasite transmission increase nonlinearly with host density? *Methods in Ecology and Evolution*, **11**, 476–486.

Keeling, M. & Rohani, P. (2008) *Modeling infectious diseases in humans and animals*. Princeton University Press.

Lachiany, M. & Stone, L. (2012) A Vaccination Model for a Multi-City System. *Bulletin of Mathematical Biology*, **74**, 2474–2487.

Monod, J. (1949) The growth of bacterial cultures. *Annual review of microbiology*, **3**, 371–394.

Pei, S., Kandula, S. & Shaman, J. (2020) Differential effects of intervention timing on COVID-19 spread in the United States. *Science Advances*, **6**, eabd6370.

Riley, S. (2007) Large-Scale Models of Infectious Disease. *Science*, **316**, 1298–1301.

Riley, S., Eames, K., Isham, V., Mollison, D. & Trapman, P. (2015) Five challenges for spatial epidemic models. *Epidemics*, **10**, 68–71.

Rohani, P., Earn, D. & Grenfell, B.T. (1999) Opposite Patterns of Synchrony in Sympatric Disease Metapopulations. *Science*, **286**, 968–971.

Saad-Roy, C.M., Wagner, C.E., Baker, R.E., Morris, S.E., Farrar, J., Graham, A.L., Levin, S.A., Mina, M.J., Metcalf, C.J.E. & Grenfell, B.T. (2020) Immune life history, vaccination, and the dynamics of sars-cov-2 over the next 5 years. *Science*, **370**, 811–818.

Tian, H., Liu, Y., Li, Y., Wu, C.H., Chen, B., Kraemer, M.U., Li, B., Cai, J., Xu, B., Yang, Q. *et al.* (2020) An investigation of transmission control measures during the first 50 days of the covid-19 epidemic in china. *Science*, **368**, 638–642.

Willem, L., Verelst, F., Bilcke, J., Hens, N. & Beutels, P. (2017) Lessons from a decade of individual-based models for infectious disease transmission: A systematic review (2006-2015). *BMC Infectious Diseases*, **17**, 1–16.