EpiModel

Mathematical Modeling of Infectious Disease

Version 0.95

Samuel M. Jenness
Department of Epidemiology
University of Washington

Steven M. Goodreau
Department of Anthropology
University of Washington

Li Wang
Department of Statistics
University of Washington

Martina Morris
Departments of Statistics & Sociology
University of Washington

January 22, 2014

Contents

1 Introduction

The EpiModel package provides tools for building, solving, and plotting mathematical models of infectious disease in R. The goals of the package are twofold. The first goal is to provide tools for comparative epidemic modeling in multiple frameworks for pedagogical purposes. The key here is *comparative* – the package has been designed to facilitate the understanding and exploration of different modeling approaches: stochastic versus deterministic, compartmental versus individual-based versus network-based.

The second goal is to support the development and extension these tools using the package's core modeling and utility functions for research. EpiModel is built on top of powerful software tools that provide access to a much wider range of functionality than in reviewed in this tutorial. Due to the open source platform of R, researchers can address scientific questions not currently supported by extending the EpiModel code to use these more advanced tools.

EpiModel currently supports modeling for three different model classes (deterministic compartmental models, individual contact models, and stochastic network models), each with three types of disease state trajectories (SI, SIR, and SIS).

This vignette provides a general tutorial on the core functionality in EpiModel. There are several other sources of guidance within the package. Smaller, HTML-based vignettes cover specific topics for specialized or utility functions. Help files for nearly all functions provide concrete examples of how the functions are used. A good place to start is the listing of all help files for EpiModel and the primary help file for the package.

```
help(package = "EpiModel")
?EpiModel
browseVignettes(package = "EpiModel")
```

Additionally, EpiModel depends heavily on the work of the deSolve package for solving deterministic models and the Statnet suite of R software for network models. Reviewing the documentation of those packages is strongly recommended for users who are planning to use EpiModel for research.

Version 0.95 Caveat The current version of EpiModel is v0.95, which indicates that some fundamental design elements to the user interface have not been settled. Therefore, we must note that **any code developed using this version of EpiModel is not guaranteed to work with the v1.0 and following versions.** With the v1.0 and following, the major user-interface elements (e.g., function argument structures) will be designed to be stable over time; some minor elements may change, but with the goal of backwards compatibility.

2 Deterministic Compartmental Models

Deterministic compartmental models solve differential equations representing an analytic epidemic system in continuous time. The models are *deterministic* because their solutions

are fixed mathematical functions of the input parameters and initial conditions, with no stochastic variability in the disease and demographic transition processes. The models are *compartmental* because they divide the population into groups representing discrete disease states (susceptible versus infected), and further on demographic, biological, and behavioral traits that influence disease transmission. In contrast to the stochastic models presented below, individuals within the population are not discretely represented.

2.1 A Basic SI Model

Starting with the basic Susceptible-Infected (SI) disease model in which there is random mixing within the population, the size of each compartment over time is represented by the equations:

$$\begin{aligned} \frac{dS}{dt} &= -\lambda S \\ \frac{dI}{dt} &= \lambda S \end{aligned} \tag{1}$$

where λ is the force of infection and represents $\frac{\beta c I}{N}$. β is the probability of transmission per contact, c is the rate of contact per person per unit time, I is the number infected at time t and and N is the population size at time t (we drop the t subscript from the differential equations as it is implicit throughout). Therefore, the force of infection is the hazard of per person per unit time of becoming infected.

Because "contact" may be ambiguously defined in the modeling literature, we use the word act to represent the activity (e.g., face-to-face talking or sexual intercourse) by which disease may be transmitted. The force of infection is multiplied by the current state sizes of the S to solve the differential equation yielding the rate of change for the compartment.

To construct a deterministic model in `EpiModel`, we use the `epiDCM` function. This currently provides functionality for several model types with homogeneous and heterogeneous mixing in the population. Many of the arguments for `epiDCM` are set by default, so we start at the most basic with a population of 501, with just one person infected at t_0. The `trans.rate` argument sets the transmission probability per act, and `act.rate` sets the acts per person per unit time.

```
mod <- epiDCM(type = "SI", s.num = 500, i.num = 1,
              trans.rate = 0.2, act.rate = 0.25, nsteps = 500)
```

We have stored all the model output in our object `mod`. The options for analyzing the results are aligned across all the model classes and types, with minimal variation. Printing the model object (simply typing the object into the R console) provides basic information on the model, including model parameters and output available for plotting and statistical analysis.

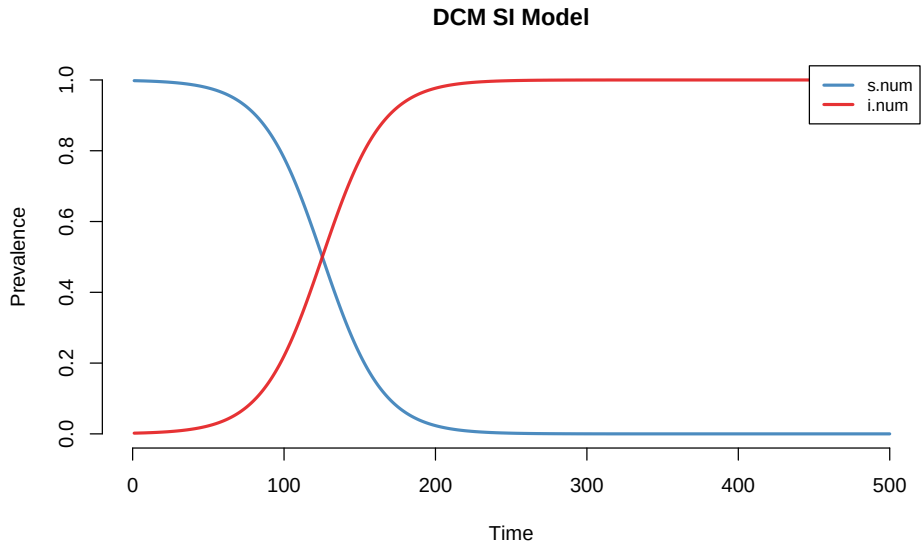
```
mod

EpiModel Object
=======================
Model class: epiDCM

Simulation Summary
-----------------------
Model type: SI
No. runs: 1
No. time steps: 500
No. groups: 1

Model Parameters
-----------------------
trans.rate = 0.2
act.rate = 0.25

Model Output
-----------------------
Compartments: s.num i.num
Flows: si.flow
```

The output shows that two compartments and one flow are available. In `EpiModel`, regardless of the model class, compartments are those discrete disease states and flows are the transitions between states and, when demographic processes are introduced, transitions in and out of the population. The `i.num` compartment is the size of the I compartment at each of the solved time steps. The endogenous disease flow names represent the starting and ending state. For example, the `si.flow` is the number of people moving from S to I at each time step. In epidemiological terms, `i.num` and `si.flow` are disease prevalence and incidence. One may extract these values directly by using the dollar operator, although later we introduce the `as.data.frame` method for easier extraction.

```
head(mod$i.num)

[1] 1.000 1.051 1.105 1.161 1.221 1.283

head(mod$si.flow)

[1] 0.04990 0.05245 0.05513 0.05794 0.06089 0.06400
```

Next, to plot the results of the model we use the generic `plot` function, which by default plots all the model compartment state prevalences (state size / population size) over time. There are many plotting options that we will explore in detail later.

```
plot(mod)
```



After examining the plot, we would like to know the size of each compartment at a specific time. That is available with the generic `summary` function, with the user inputting the time of interest. At t_{150}, 22.5% of the population have become infected, and the incidence at that time is 4.37 new infections.

```
summary(mod, time = 150)

EpiModel Summary
========================
Model class: epiDCM

Simulation Summary
------------------------
Model type: SI
No. runs: 1
No. time steps: 500
No. groups: 1

Model Statistics
--------------------------------
Time: 150  Run: 1
--------------------------------
                  n      pct
Suscept.    112.845    0.225
Infect.     388.155    0.775
```

```
Total       501.000    1.000
S -> I        4.371       NA
---------------------------------
```

2.2 SIR Model with Demography

In a Susceptible-Infectious-Recovered (SIR) model, infected individuals recover from disease into a life-long recovered state; they are never again susceptible to disease. In this section, we model an SIR disease by adding to our basic SI model a recovery process. We also introduce demographic processes so that persons may enter and exit the population through births and deaths. The model is represented by the following system of differential equations:

$$\begin{aligned}
\frac{dS}{dt} &= -\lambda S + fN - \mu_s S \\
\frac{dI}{dt} &= \lambda S - \nu I - \mu_i I \\
\frac{dR}{dt} &= \nu I - \mu_r R
\end{aligned} \tag{2}$$

where f is the birth rate, μ are the mortality rates specific for each compartment, and ν is the recovery rate. In an SIR model, the recovery rate is the reciprocal of the average duration of disease infectiousness, and likewise the reciprocal of the death rates are the average life expectancy for persons in those compartments.

Solving In `EpiModel`, introducing these new transition processes into the model is straightforward. Most importantly, it is necessary to set `type="SIR"`. Next one sets the recovery rate, birth rate, and state-specific death rates as below. The model is solved through 500 time steps, but here we also specify `dt=0.5` to obtain model results in fractional time units (e.g., results are available for t_1, $t_{1.5}$, t_2, ..., $t_{499.5}$, t_{500}).

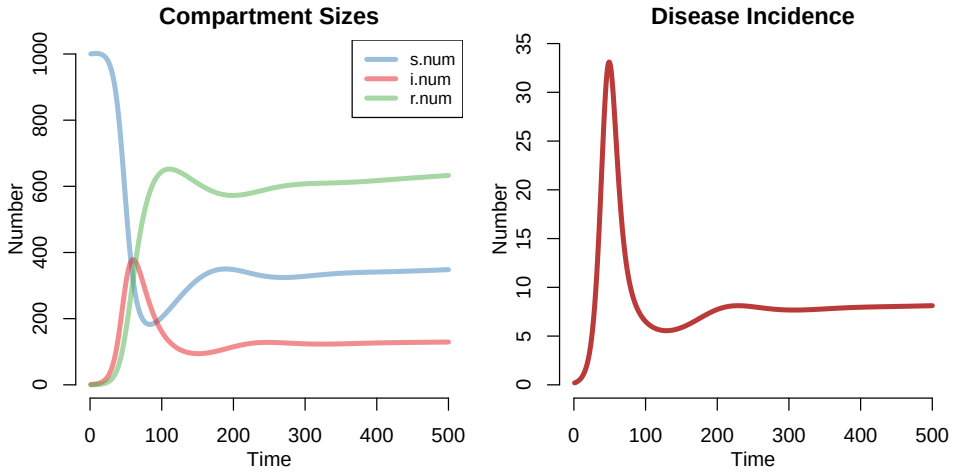
```
mod <- epiDCM(type = "SIR", s.num = 1000, i.num = 1, r.num = 0,
              trans.rate = 0.2, act.rate = 1, rec.rate = 1/20,
              b.rate = 1/95, ds.rate = 1/100, di.rate = 1/80,
              dr.rate = 1/100, nsteps = 500, dt = 0.5)
```

In words, our model parameters imply that the birth rate is slightly higher than the underlying death rates, and that there is disease-induced mortality because the `di.rate` is larger than the other two death rates.

Plotting Next we plot the results of the model with several plot arguments set to non-default values. The `par` options set general plotting options, and here two side-by-side plots are set. In the left plot, the `popfrac=FALSE` argument plots the compartment size (rather than prevalence) and `alpha` increases the transparency of the lines for better visualization.

By default, the `plot` function will plot the prevalences for all compartments in the model, but in the right plot we override that using the `y` argument to specify that disease incidence (the `si.flow` element of the model object) should be plotted. Also by default in any plots with a "flow" output, the absolute number will be plotted (i.e., `popfrac=FALSE` by default).

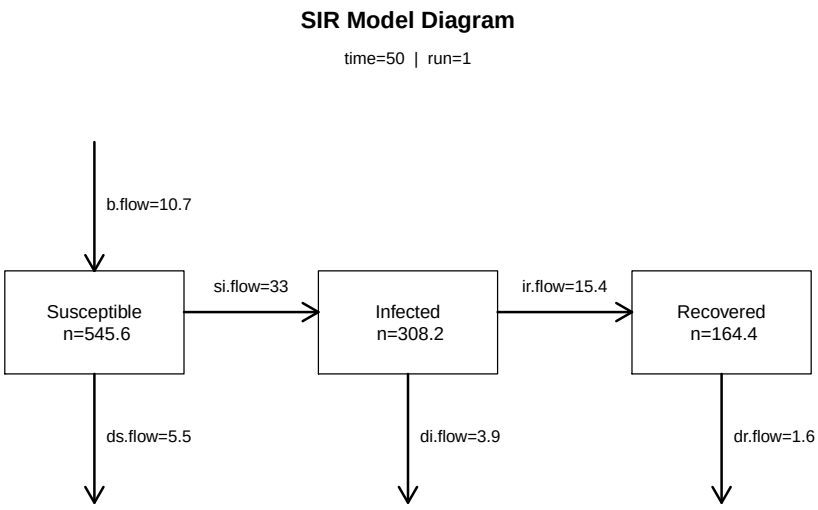
```
par(mar = c(3.2,3,2,1), mgp = c(2,1,0), mfrow = c(1,2))
plot(mod, popfrac = FALSE, alpha = 0.5,
     lwd = 4, main = "Compartment Sizes")
plot(mod, y = "si.flow", lwd = 4, col = "firebrick",
     main = "Disease Incidence", leg = "n")
```



Finally, it is also possible to specify a single line color, a vector of colors, or a color palette (more on that next) using the `col` argument, and the legend options are set using the `leg` argument.

Summaries In Section 2.1, we obtained time-specific model values using the `summary` function. It is also possible to obtain that information graphically with the `comp.plot` function. This plot provides a standard state-flow diagram that is typically presented in the epidemiological literature.

```
par(mfrow = c(1,1))
comp.plot(mod, time = 50, digits = 1)
```



The plot shows the three state sizes and flows at t_{50}. This plot is also built into the `summary` function through the `comp.plot` argument to that function (see the help pages for the class-specific summary functions). Currently these plots are limited to one-group models only, but this functionality will be expanded in future releases.

2.3 SIS Model with Sensitivity Analyses

It is often of scientific interest to know how model outputs (e.g., disease prevalence and incidence) vary with starting parameter values. A key design feature of `epiDCM` class models is to run these type of sensitivity analyses with minimal programming to facilitate the exploration of model counterfactuals.

In this Section, we illustrate an example for a Susceptible-Infected-Susceptible (SIS) disease, an example of which is a curable bacterial sexually transmitted infection like gonorrhea. For ease of presentation, we model this in a closed population (no demography), but those processes may be added to this base model in the same way as the SIR model in Section 2.2.

The SIS model is represented by the following system of differential equations:

$$\begin{aligned} \frac{dS}{dt} &= -\lambda S + \nu I \\ \frac{dI}{dt} &= \lambda S - \nu I \end{aligned} \tag{3}$$

where the ν parameter now represents "recovery" back into the susceptible state (no one achieves life-long immunity from disease). The force of infection and recovery equations are mirrors of one another, since individuals flow back and forth between states over time.

In `EpiModel`, running an SIS model requires specifying `type="SIS"` and supplying a `rec.rate` parameter, just as with an SIR model. To conduct a sensitivity analysis, we enter the parameter to be varied as a vector of values rather than a scalar (a single value). Here,

we vary the `act.rate` parameter from 0.25 to 0.50 acts acts per person per unit time in increments of 0.05 acts. Therefore, this sensitivity analysis will run 6 models to investigate epidemic trajectories given a changing `act.rate` but holding all other parameters constant.

```
mod <- epiDCM(type = "SIS", s.num = 500, i.num = 1,
              trans.rate = 0.2, act.rate = seq(0.25, 0.5, 0.05),
              rec.rate = 1/50, nsteps = 350)
```

When we print the model output, it is clear that a sensitivity model has been run because the output indicates 6 runs, with the appropriate range of values for the `act.rate` parameter.

```
mod

EpiModel Object
=======================
Model class: epiDCM

Simulation Summary
----------------------
Model type: SIS
No. runs: 6
No. time steps: 350
No. groups: 1

Model Parameters
----------------------
trans.rate = 0.2
act.rate = 0.25 0.3 0.35 0.4 0.45 0.5
rec.rate = 0.02

Model Output
----------------------
Compartments: s.num i.num
Flows: si.flow is.flow
```

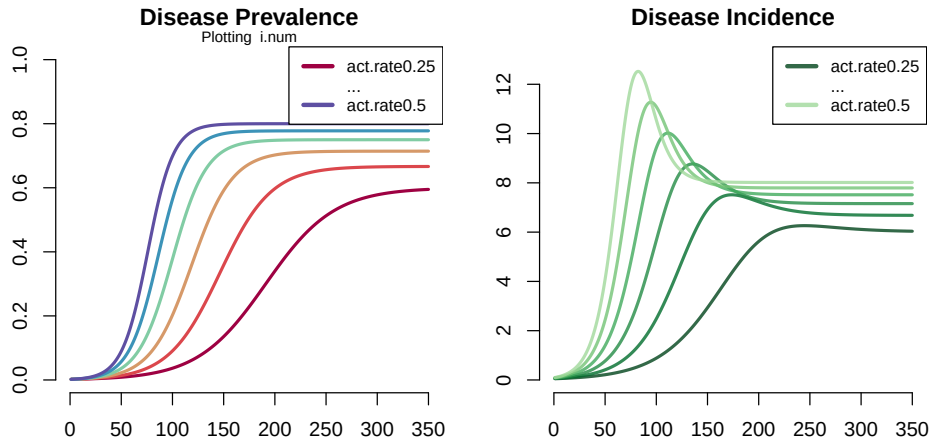
Extracting Output In Section 2.1, we demonstrated how to extract model output using the dollar sign syntax. In a sensitivity analysis mode, the output of each compartment or flow (e.g., `mod$s.num`) is not a vector of values but a data frame with columns for each model run and rows for each time step. To extract all the compartment or flow values across runs, we still use the dollar sign syntax. But to extract all the model output *for all compartments and flows from a specific run*, we can use the generic `as.data.frame` function.

Below we extract primary model output from the fifth model run for the first five time steps. The run-specific data frame may also be saved out to its own object for further analysis.

```
head(as.data.frame(mod, run = 5))

  time s.num i.num si.flow is.flow num
1    1 500.0 1.000 0.08982 0.02000 501
2    2 499.9 1.072 0.09630 0.02145 501
3    3 499.9 1.150 0.10325 0.02300 501
4    4 499.8 1.233 0.11069 0.02466 501
5    5 499.7 1.322 0.11867 0.02644 501
6    6 499.6 1.418 0.12722 0.02835 501
```

Plotting For plotting a sensitivity analysis, `EpiModel` features specialized tools to highlight the range of values across model runs. Below is a two-panel plot of disease prevalence and incidence across all 6 models. By default, plotting a sensitivity model will show the disease prevalence over time: the proportion of persons infected in each model run at each time step.

```
par(mfrow = c(1,2), mar = c(3.2,3,2.5,1))
plot(mod, alpha = 1, main = "Disease Prevalence")
plot(mod, y = "si.flow", col = "Greens", alpha = 0.8,
     main = "Disease Incidence")
```



`EpiModel` uses a robust color palette system to set the range of colors across the models in order to easily differentiate lines. The underlying framework is the `RColorBrewer` package, which provides access to visually distinct color palettes used in geographic mapping. The default color palette is featured for the prevalence plot, but any palette in `display.brewer.all()` may be set using the `col` argument, as with the incidence plot.

The color argument also accepts vectors of colors as well. For example, the following are also acceptable color specifications for a sensitivity analysis plot. It is also possible to set run-specific line types in a similar fashion using the `lty` argument.

```
plot(mod, col = "black")
plot(mod, col = 1:6)
plot(mod, col = c("black", "red", "blue", "green", "purple", "pink"))
plot(mod, col = rainbow(6))
```

Varying Multiple Parameters It is also possible to vary multiple parameters simultaneously. The only limitation is that the number of model runs implied must be equal across all varying parameters. For example, if one specifies `act.rate` as a vector of length six, and one is also interested in simultaneously varying the transmission probability, the length of that `trans.rate` vector must also be six. Below is an example showing those two parameters simultaneously varied, although the results are not shown.

In total, six models are requested in this sensitivity analysis. There are three different act rates, and for each act rate, there are two different transmission probabilities to model. The varying act and transmission parameters will then be evaluated in that order, with the first model having `act.rate=0.2` and `trans.rate=0.1`, and the last model having `act.rate=0.6` and `trans.rate=0.2`.

```
act.rates <- c(0.2, 0.2, 0.4, 0.4, 0.6, 0.6)
trans.rates <- c(0.1, 0.2, 0.1, 0.2, 0.1, 0.2)
mod <- epiDCM(type = "SIS", s.num = 500, i.num = 1,
              trans.rate = trans.rates, act.rate = act.rates,
              rec.rate=1/50, nsteps = 350)
```

2.4 Two-Group SI Model with Vital Dynamics

`EpiModel` also includes heterogeneous two-group models, which break the random mixing patterns implied by one-group models. These two-group models are available for all three model classes, but for network models we call them two-*mode* (or bipartite models) in accordance with network science terminology (see Section 4.2).

In the two-group models as they are currently structured, mixing between groups is *purely heterogeneous*: one group only has contacts with the other group; there are no within-group contacts. This framework would be appropriate for epidemic modeling on a purely heterosexual partnership structure, with the simplifying assumption of no same-sex contacts. More flexible assortative mixing, in which mixing occurs imperfectly across groups, is currently possible for network models (see Section 4.1) but not yet for deterministic and individual contact models (that functionality will be added in future versions).

Mathematical Structure We build upon the basic SI model featured in Section 2.1. The model now includes four compartments, two disease states $\times$ two groups, requiring the following set of four differential equations. The equation variables are subscripted by group number; in `epiDCM` one must specify group-specific parameters for the force of infection, birth rate, and death rates.

$$\begin{aligned}
\frac{dS_{g1}}{dt} &= -\lambda_{g1}S_{g1} + f_{g1}N_{g1} - \mu_{s,g1}S_{g1} \\
\frac{dI_{g1}}{dt} &= \lambda_{g1}S_{g1} - \mu_{i,g1}I_{g1} \\
\frac{dS_{g2}}{dt} &= -\lambda_{g2}S_{g2} + f_{g2}N_{g2} - \mu_{s,g2}S_{g2} \\
\frac{dI_{g2}}{dt} &= \lambda_{g2}S_{g2} - \mu_{i,g2}I_{g2}
\end{aligned} \tag{4}$$

The critical heterogeneous mixing component is actually embedded within the group-specific λ transmission rates. The formula for the two lambdas that specifies mixing is:

$$\begin{aligned}
\lambda_{g1} &= \tau_{g1} \times \alpha_{g1} \times \frac{I_{g2}}{N_{g2}} \\
\lambda_{g2} &= \tau_{g2} \times \alpha_{g2} \times \frac{I_{g1}}{N_{g1}}
\end{aligned} \tag{5}$$

In words, the force of infection for group one is the product of the group one transmission probability per act, the group one act rate per unit time, and the probability that an infected is selected among the possible group 2 contacts. The difference with the one-group model is that group-specific τ and α parameters are allowed.

The probability of coming into contact with an infected person is not based on the prevalence of *all* infected persons, but only those of the opposite group. The group-specific τ parameter is the *probability of a transmission occurring to that group member* (e.g., τ_{g1} is the probability that a member of group one is infected given contact with a member of group two).

Balancing Act Another important point concerns the act rate parameter, α. In the formulas for the group-specific λ above, it was implied that each group may have its own act rate. However, in practice with these purely heterogeneous mixing models, the act rate of one group has a specific mathematical relationship with the act rate of the other group:

$$\alpha_{g1}N_{g1} = \alpha_{g2}N_{g2} \tag{6}$$

The number of acts per unit time for each group is the act rate times the group size. The total number of acts in one group in this purely heterogeneous mixing model must equal the total number of acts in the other. To accomplish we use act rate "balancing". This is particularly important in an open population model since the group population sizes change over time.

There are a variety of ways to accomplish this balancing, but we implement a basic method currently in `EpiModel`. One specifies an α parameter for *either* group one or group two, and also that this is the group the other group's α should be balanced on. For example, if one specifies α_{g1}, then the act rate for group one is fixed at that value and the rate for group two is derived over time by rearranging the equation above as follows:

$$\alpha_{g2} = \frac{\alpha_{g1} N_{g1}}{N_{g2}} \tag{7}$$

It is mathematically possible to implement more flexible balancing that averages two act rates or deals with changes in other ways, but the balancing is currently limited in `EpiModel` to the above formulation.

Parameterizing Groups With the `epiDCM` function, it is simple to add another group to the model. The first step is to specify that `groups=2`, and then set the group-specific initial state sizes and model parameters. The names of the parameters for the first group has not changed: the sizes and parameters specific to group one are those *without* the `g2` suffix.

An important use of two-group models is to allow for some biological or behavioral distinctions between groups that differentially impacts disease transmission. In the model below, we specify that the transmission probability for group one is four times as high as that of group two. This could represent a four-fold higher risk of infection for females (group 1) than for males (group 2).

We enforce the act rate balancing as noted above, by specifying an act rate for group one only and also using the `balance = "g1"` argument. Needing to specify which group's act rate trumps in this argument is redundant, since it is implied by the group that has an act rate, but future versions of `EpiModel` will use this argument to allow the user to specify how balancing should occur.

```
mod <- epiDCM(type = "SI", groups = 2,
              s.num = 500, i.num = 1,
              s.num.g2 = 500, i.num.g2 = 0,
              trans.rate = 0.4,  trans.rate.g2 = 0.1,
              act.rate = 0.25, balance = "g1",
              b.rate = 1/100, b.rate.g2 = NA,
              ds.rate = 1/100, ds.rate.g2 = 1/100,
              di.rate = 1/50, di.rate.g2 = 1/50,
              nsteps = 500)
```

Note also how the group-specific birth rates are input. Because these purely heterogeneous mixing models are ideal for heterosexual disease transmission in which groups are sexes, we can represent group one as females and group two as males. In this case, the birth rate into a population is not a function of the total population size but that of the female group. Therefore, one may specify a birth rate for females only, and set the group two birth

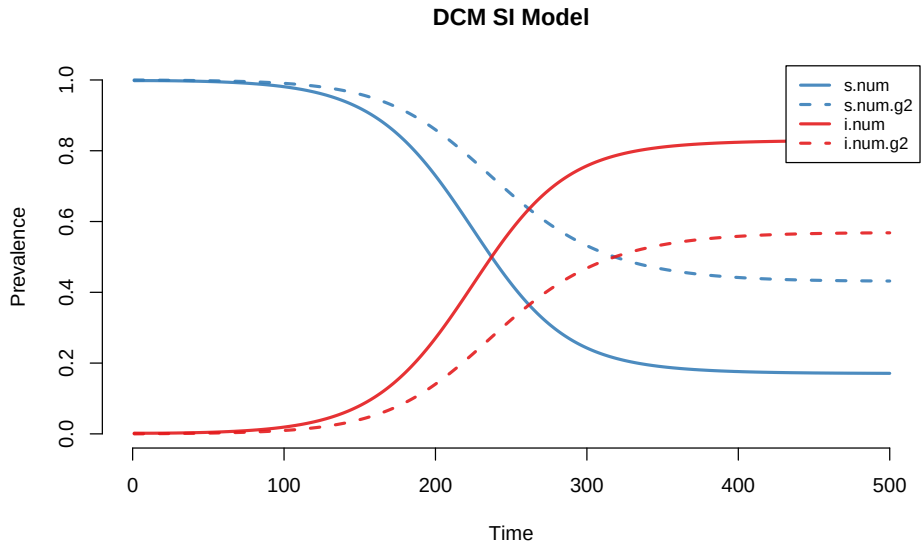
rate to `NA`. If so this is done, new births are evenly allocated between the two groups based on the size of group one and the rate specified by `b.rate`.

Solving the model, we print to show its contents. The number of compartments and flows available to plot and analyze has now doubled. Here again, the group one outcomes are those without the g2 suffix.

```
mod

EpiModel Object
=======================
Model class: epiDCM

Simulation Summary
-----------------------
Model type: SI
No. runs: 1
No. time steps: 500
No. groups: 2

Model Parameters
-----------------------
trans.rate = 0.4
act.rate = 0.25
b.rate = 0.01
ds.rate = 0.01
di.rate = 0.02
trans.rate.g2 = 0.1
b.rate.g2 = NA
ds.rate.g2 = 0.01
di.rate.g2 = 0.02

Model Output
-----------------------
Compartments: s.num s.num.g2 i.num i.num.g2
Flows: si.flow si.flow.g2 b.flow b.flow.g2 ds.flow
ds.flow.g2 di.flow di.flow.g2
```

The default plot shows the prevalence of all four compartments in the model. Note that two-group models with sensitivity analyses begin to visually overwhelm the plotting of compartments. One should freely deviate from the default plotting options to specify which specific compartments or flows and from which model runs should be represented.

```
plot(mod)
```



2.5 Graphical Interface for epiDCM

To help introduce DCM models primarily for teaching purposes, we have included in `EpiModel` a web browser-based graphical user interface (GUI). The GUI is based on the `shiny` package that enables the creation of interactive data analysis tools from within `R`. To run the GUI, simply type the following into the console. This will open a browser window. Parameter values and initial state sizes may be changed. Output includes plots, data summaries, and raw data to extract.

```
gui.epiDCM()
```

The GUI is currently limited to one-group homogeneous mixing models only but may be expanded to include more complex mixing structures in a forthcoming release.

3 Individual Contact Models

In this Section, we outline the framework for the class of models called individual contact models. These models are intended to be stochastic analogs of the deterministic compartmental models featured in Section 2 and solved with the `epiDCM` function. The main differences between the two model classes are as follows.

1. **Rates are stochastic:** these are stochastic models in which nearly all of the rates governing transitions between states are now random draws from distributions summarized by those rates. This will be explained in detail in the examples below.

2. **Time is discrete:** The `epiICM` class models are in discrete time, in contrast to the continuous time of the `epiDCM`. In the former, everything within a time step happens in a series of processes, since there is no instantaneous occurrence of events independent of others, as is possible in deterministic models. This has the potential to introduce bias or artificiality into the disease process if the unit for the time step is large (e.g., a month) since the transitions that might occur within that time step cannot necessarily be considered independent. In general, *caution is needed* when simulating any discrete-time model with long unit time steps or large rate parameters, given the potential for competing risks within each step.
3. **Units are individuals:** `epiICM` models simulate disease spread over a simulated population of individually identifiable elements; in contrast, `epiDCM` treats the population as an aggregate whole which is infinitely divisible, with individual elements in this population neither identifiable nor accessible for modeling purposes. The implications of this are relatively minor if the stochastic simulations occur on a sufficiently large population, but there are some critical modeling considerations of individual-based simulations that will be reviewed in the examples below.

A major goal of `EpiModel` is to facilitate comparisons across different model classes, so the interface and functionality of the stochastic models in `epiICM` is very similar to `epiDCM`. The syntax and argument names for all of the shared parameters and initial state sizes are the same. One difference between the two functions is the way *sensitivity analyses* are handled: `epiICM` will not run multiple models across varying parameters because each model with a given parameter set is typically run many times to quantify the stochasticity in the model. Therefore sensitivity analyses would need to be run from separate `epiICM` function calls.

3.1 SI Model Stochasticity

We introduce `epiICM` by using the same model parameterization as the deterministic SI model in Section 2.1, with one person infected, an average activity or contact rate of 0.25 per time step, and a transmission probability of 0.2 per act. We simulate this model 10 times over 300 time steps.

```
mod <- epiICM(type = "SI", s.num = 500, i.num = 1,
              trans.rate = 0.2, act.rate = 0.25,
              nsims = 10, nsteps = 300)
```

By default, the function prints the model progress (although that is suppressed here): stochastic simulation models generally take longer to run than the deterministic models, especially for larger populations and longer time ranges, because transition processes must be simulated for each individual at each time step.

The model results may be printed, summarized, and plotted in very similar a fashion to the `epiDCM` models. Printing the ICM model object shows the number of simulations for which there are results.

```
mod

EpiModel Object
=======================
Model class: epiICM

Simulation Summary
-----------------------
Model type: SI
No. simulations: 10
No. time steps: 300
No. groups: 1

Model Parameters
-----------------------
trans.rate = 0.2
act.rate = 0.25

Model Output
-----------------------
Compartments: s.num i.num
Flows: si.flow
```

In contrast to `epiDCM`, the summary function now *does not* take a `run` argument (used only for sensitivity analyses anyway). Instead, the output summarizes the mean and standard deviation of model results at the requested time step across all simulations. Here, we request those summaries at time step 125; the output shows the compartment and flow averages across those 10 simulations.

```
summary(mod, time = 125)


EpiModel Summary
=======================
Model class: epiICM

Simulation Details
-----------------------
Model type: SI
No. simulations: 10
No. time steps: 300
No. groups: 1
```

```
Model Statistics
--------------------------------
Time: 125
--------------------------------
             mean         sd     perc
Suscept.    286.3    119.647    0.571
Infect.     214.7    119.647    0.429
Total       501.0      0.000    1.000
S -> I        5.2      2.573       NA
--------------------------------
```

Summary statistic values like means and standard deviations may be of primary interest for analysis and plotting, so the generic `as.data.frame` function for `epiICM` objects allows for this. As described in the function help page (see `?as.data.frame.epiICM`), the output choices are for the time-specific means, standard deviations, and individual simulation values.

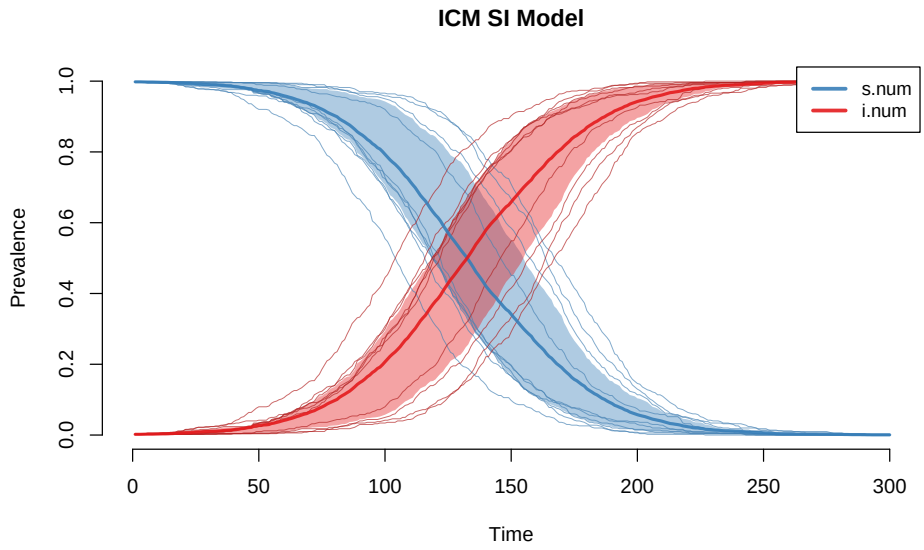
```
head(as.data.frame(mod, out='mean'))

  time s.num i.num si.flow num
1    1 500.0   1.0     0.0 501
2    2 500.0   1.0     0.0 501
3    3 500.0   1.0     0.0 501
4    4 499.9   1.1     0.1 501
5    5 499.8   1.2     0.1 501
6    6 499.7   1.3     0.1 501
```

Plotting Plotting stochastic model results requires thinking through what summary measures best represent the model. In some models, it may be sufficient to plot the simulation means, but in others visualizing the individual simulations is necessary.

The generic plotting function for `epiICM` objects generates these visual outputs simply but robustly. Standard plotting output includes individual simulation lines, means of those simulations, and simulation quantiles. Plotting the model object with no options will show the mean and inter-quartile range for all compartment sizes in the model for one-group models, and the mean lines for two-group models.

Each of the elements may be toggled on or off using the plotting arguments listed in `?plot.epiICM`. Here, we add the individual simulation lines to the default plot using `sim.lines` and then color those lines with `sim.col`.

```
plot(mod, sim.lines = TRUE, sim.col = c("steelblue", "firebrick"))
```



Note that we use the `RColorBrewer` color palette system to generate the default colors but these plots, but the colors are closely approximated by the built-in R standard colors *steelblue* for blue, *firebrick* for red, and *seagreen* for green.

3.2 SIR Stochastic-Deterministic Comparison

One methodological question for comparative mathematical modeling is how model results vary with model structure while fixing parameters. `EpiModel` allows for easy comparison between model classes using the same parameters. In this Section, we show how to compare a deterministic with a stochastic SIR model in an open population (i.e., with demography). First, the deterministic model is run with the following parameters.

```
det <- epiDCM(type = "SIR", s.num = 1000, i.num = 100,
              trans.rate = 0.2, act.rate = 0.8, rec.rate = 1/50,
              b.rate = 1/100, ds.rate = 1/100, di.rate = 1/90,
              dr.rate = 1/100, nsteps = 300)
```

Note that we have a large number of infected at t_0 because with just one infected it usually takes discrete-time, individual-based models much longer to grow the epidemic. This itself is an interesting property of individual-based, discrete-time stochastic models that may better represent reality; but here, our goal is very close approximation by minimizing these differences.

The `epiICM` model is simulated 10 times, with the same parameters as the deterministic model. In this model, the contact, transmission with contact, recovery, birth, and death processes are all governed by random draws from Poisson or binomial distributions with parameters set by the rates specified here.

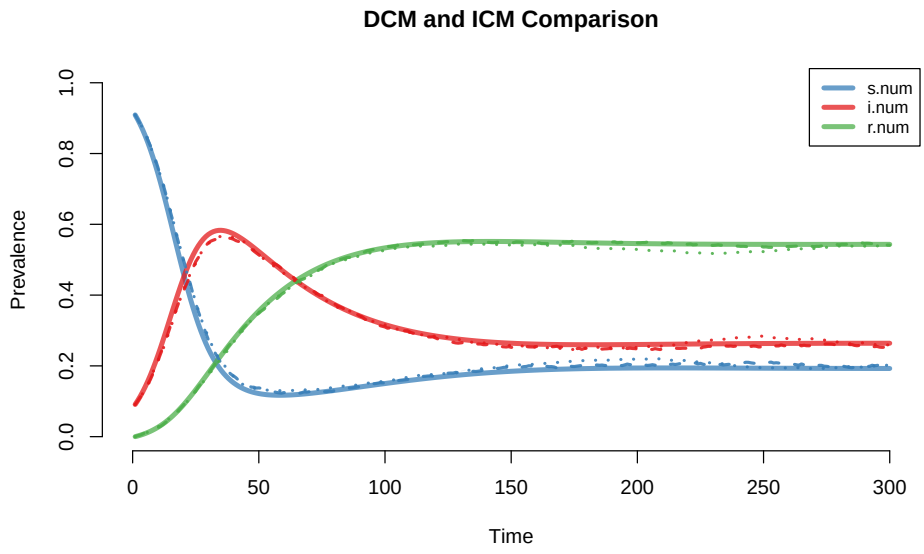
```
sim <- epiICM(type = "SIR", s.num = 1000, i.num = 100,
              trans.rate = 0.2, act.rate = 0.8, rec.rate = 1/50,
              b.rate = 1/100, ds.rate = 1/100, di.rate = 1/90,
              dr.rate = 1/100, nsteps = 300, nsims = 10)
```

As a third model, we change our ICM model to toggle off the stochastic elements of the birth death processes. Instead of setting the number of new births and deaths in each compartment at each time step as random draws from a Poisson distribution with rate parameters set by the arguments, these transition flows are calculated by rounding the product of the rate and the state size.

```
sim2 <- epiICM(type = "SIR", s.num = 1000, i.num = 100,
               trans.rate = 0.2, act.rate = 0.8, rec.rate = 1/50,
               b.rate = 1/100, ds.rate = 1/100, di.rate = 1/90,
               dr.rate = 1/100, nsteps = 300, nsims = 10,
               b.rand = FALSE, d.rand = FALSE)
```

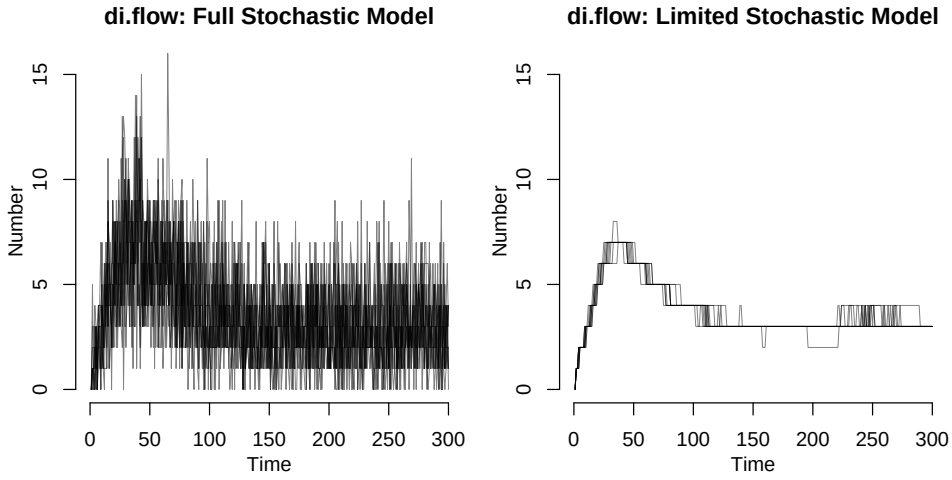
Comparing Means In our plot, the deterministic results are shown in the solid line, the first stochastic results in the dashed line, and second stochastic results in the dotted line. In this example, the three lines are generally consistent, which is as expected since we are only visualizing the means, we have a sufficiently large population, and the demographic transition rates are relatively low.

```
plot(det, alpha = 0.75, lwd = 4, main = "DCM and ICM Comparison")
plot(sim, qnts = FALSE, add = TRUE, mean.lty = c(2,2,2), leg = FALSE)
plot(sim2, qnts = FALSE, add = TRUE, mean.lty = c(3,3,3), leg = FALSE)
```



Comparing Variance Note, however, that the variation in flows across simulations is vastly different as a component of the stochastic process of mortality in the full stochastic model. In the following side-by-side plots, we show the individual simulation lines for the out-transition (mortality) from the infected state. In the full stochastic model, there is a relatively wide range of numbers of infected deaths over time, whereas there is little variability in the limited stochastic model.

```
par(mfrow = c(1,2), mar = c(3,3,2,1), mgp = c(2,1,0))
plot(sim, y = "di.flow", mean.line = FALSE,
     sim.lines = TRUE, sim.alpha = 0.5,
     ylim = c(0, max(sim$di.flow)),
     main = "di.flow: Full Stochastic Model")
plot(sim2, y = "di.flow", mean.line = FALSE,
     sim.lines = TRUE, sim.alpha = 0.5,
     ylim = c(0, max(sim$di.flow)),
     main = "di.flow: Limited Stochastic Model")
```



For a time-specific analysis of variance, we can compare the standard deviations of the two model results at time step 50 (about when the infected death incidence peaks) using the data extraction through `as.data.frame` generic function:

```
icm.compare <- rbind(round(as.data.frame(sim, out = "sd")[50,], 2),
                     round(as.data.frame(sim2, out = "sd")[50,], 2))
row.names(icm.compare) <- c("full", "lim")
icm.compare

     time s.num i.num r.num si.flow ir.flow b.flow ds.flow di.flow
full   50 15.32 23.64 24.16    3.47    3.27    3.5    0.67    1.58
lim    50  9.20 19.68 19.39    4.69    3.65    0.0    0.32    0.32
     dr.flow   num
full     3.2 38.89
```

```
lim      0.0  2.22
```

The minor variations in the birth and death flows at some time points because the compartment size at each time point is still a function of the random infection transition across time.

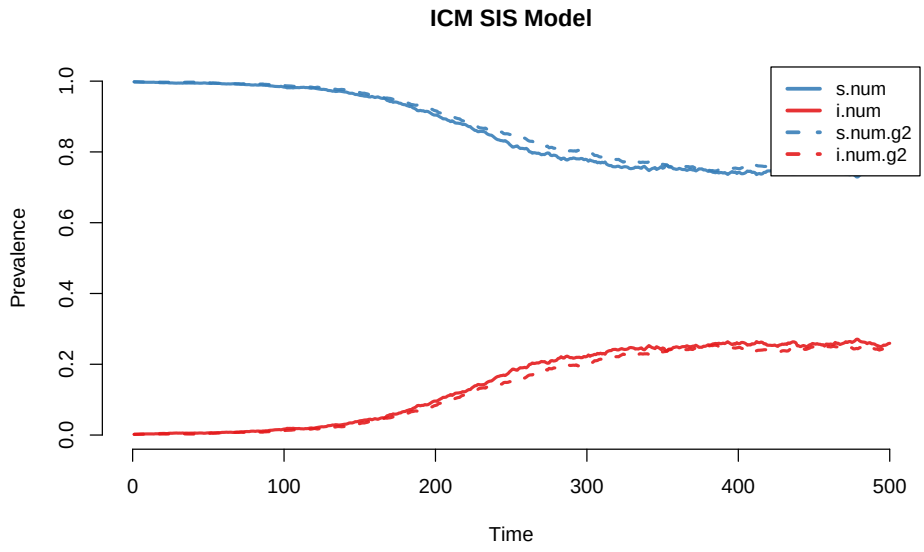
3.3 Two-Group SIS Model with Demography

The last model type we feature for `epiICM` class models is a two-group Susceptible-Infected-Susceptible (SIS) model in which there are two groups that mix purely heterogeneously. The model is parameterized in much the same way as the deterministic two-group SIR model featured in Section 2.4. There are the same act rate balancing considerations. One must also specify group-specific recovery rates. In this model, we simulate a disease in which the first group has twice the probability of infection, but recovers back into the susceptible state at twice the rate as the second group. This might occur, for example, if first group differentially had access to curative treatment for disease.

```
set.seed(12345)
sim <- epiICM(type = "SIS", groups = 2,
              s.num = 500, i.num = 1,
              s.num.g2 = 500, i.num.g2 = 1,
              trans.rate = 0.2, trans.rate.g2 = 0.1,
              act.rate = 0.5, balance = "g1",
              rec.rate = 1/25, rec.rate.g2 = 1/50,
              b.rate = 1/100, b.rate.g2 = NA,
              ds.rate = 1/100, ds.rate.g2 = 1/100,
              di.rate = 1/90, di.rate.g2 = 1/90,
              nsteps = 500, nsims = 10)
```

The plot shows a similar disease burden for both groups, with a disease prevalence of around 16% in both groups at time step 400. But we see similar epidemic trajectories for both groups because their transmission probabilities and recovery rates have balanced out: both groups have an $R_0 = 2.5$.

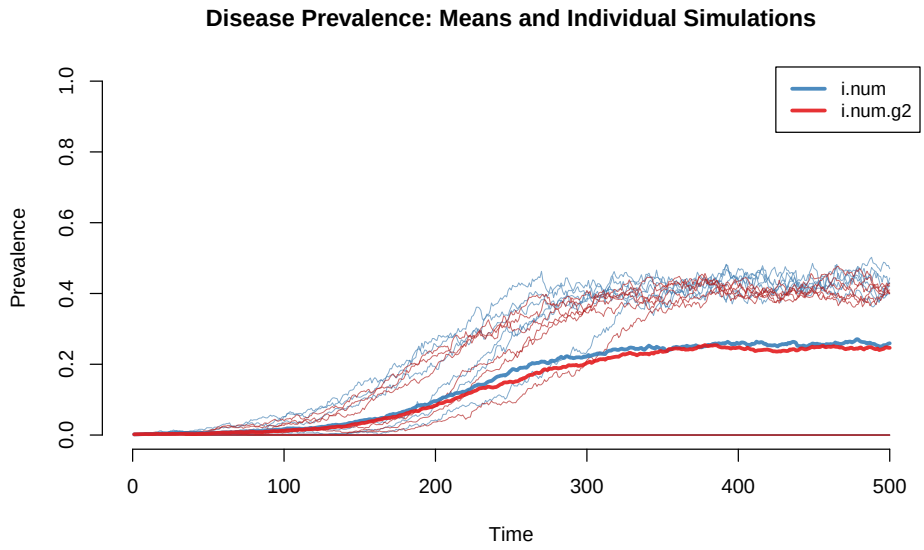
```
par(mfrow = c(1,1))
plot(sim)
```



The default plotting options for two-group models will only show the simulation means, so the additional summary information (quantiles and individual simulation lines) must be toggled on as needed (see `?plot.epiICM` for help).

This second plot shows that we must be careful to only look at the simulation means. In this case, the mean lines are an average of normally occurring epidemics in 6 simulations and extinct epidemics in 4 simulations. Model extinctions occur in this case because the recovery rate in the first group is relatively short and there is only one person initially infected.

```
plot(sim, y = c("i.num", "i.num.g2"), mean.lwd=3,
     sim.lines = TRUE, sim.col = c('steelblue', 'firebrick'),
     main = "Disease Prevalence: Means and Individual Simulations",
     leg = TRUE)
```



Another useful diagnostic for this behavior is found in the `summary` function, where we would find that at time step 400, the mean number infected is 114.5, and the standard deviation around that mean is 99.4.

If this phenomenon of stochastic model extinction represents the underlying epidemic process of interest in which there is one initial infected, these `epiICM` models have utility per se. But if the number of initial infected is arbitrary (or unknown), the model extinctions may be an artificiality to be removed: in that case, one may increase the total population size (specifically the initial number infected) or reduce the size of the time step (and also, adjust the denominators of the parameters with units of time in them).

4 Stochastic Network Models

Network models move beyond individual contact models by explicitly modeling phenomena within and across partnership dyads (pairs of individuals who remain in contact) over time. This enables partnerships to have duration in time, allowing for repeated acts with the same person, specification of partnership formation and dissolution rates, control over the temporal sequencing of multiple partnerships, and specification of network-level features. As one dyad may now be connected to other dyads, this forms a partnership network.

Model Framework `EpiModel` uses exponential-family random graph models (ERGMs) to estimate and simulate complete partnership networks based on patterns of density, degree, assortivity, and other network features. Since epidemic models are dynamic, evolving systems, we use temporal ERGMs, in which partnership formation and dissolution are modeled and simulated over time.

Dynamic network models may be estimated from several different types of empirical data, including panel data on a complete network over time. For a description of these op-

tions, please consult the help documentation and vignettes for the `tergm` and `networkDynamic` packages. `EpiModel` currently has functionality only run network estimations with target statistics for formation that may be estimated using so-called "egocentric network sampling:" a random sample of the population is drawn, and those individuals are asked about a complete or limited set of their partnerships within an interval. Network models may thus be parameterized by inputting summary target statistics for the distribution of partnership number at one point in time, assortivity in partner traits, and other dyadic and network-level features. On top of this, an average of partnership duration is estimated from the data, and used to govern partnership dissolution rates.

Model Processes `EpiModel` has the capacity to simulate disease epidemics over these partnership networks by integrating this statistical framework for networks with stochastic transmission processes similar to those featured in the `epiICM` class disease models. Similar to `epiICM`, these network models require specification of epidemic parameters that govern the transmission, recovery, and other other transition processes of individual persons in discrete time. The three model types currently supported are the same: SI, SIR, and SIS disease types. These types will be expanded in future `EpiModel` releases.

In contrast to `epiDCM` and `epiICM` models, which solve or simulate the entire epidemic system with one function, network models require multiple steps: 1) the partnership network is parameterized, fit, and diagnosed; 2) a complete network is simulated based on that model fit; and 3) the disease processes are simulated on top of the dynamic simulated network.

Independent versus Dependent Models A key distinction for network models is whether two dynamic processes, the partnership network dynamics and the disease transmission dynamics, are treated as independent or dependent. **Independent** network models assume no influence of the disease simulation on the structure of the temporal network, although the structure of the temporal network most certainly impacts the disease. **Dependent** network models allow for the disease process to influence the network. Examples include cases like serosorting – where disease status influences partner selection – and demographic processes (births, deaths, and migration) – where the partner selection process must adapt to changing population size and composition.

Model Functions Network models in `EpiModel` thus involve two or three main functions given the choice of independent or dependent model structure:

1. `epiNet.est` estimates the generative model for the dynamic partnership networks. This function is a wrapper around the `ergm` and `stergm` functions in those similarly named packages, with additional diagnostic tables and plots useful for dynamic epidemic models.
2. `epiNet.simNet` simulates networks given a model fit with `epiNet.est`. These network simulations are used for network epidemic models in which there is no dependence

between the network structure and the disease process (thus, the network structure may be fully simulated ahead of the disease simulation).

3. `epiNet.simTrans` then runs the stochastic epidemic models, with a given network model fit from `epiNet.est` or set of network simulations from `epiNet.simNet`, respectively. For models involving dependence between the network structure and the disease trajectories (e.g., disease causes death, which dissolves partnerships), the direct model fit is used and the network re-simulated at each time step.

We explain these processes in further detail with two network modeling tutorials, one for an independent SIS model with one mode and one for a dependent SI model with two modes.

4.1 Independent One-Mode SIS Model

In this section, we simulate a recoverable disease (e.g., a bacterial sexually transmitted infection) over a one-mode network in which we assume the disease status does not influence partnership structure. In the language of DCM models in Section 2 and ICM models in Section 3, this is a *one-group* network in a closed population. In network terminology applied to epidemic modeling, modes are essentially groups or categories of nodes that govern the rules of mixing. One-mode networks allow for partnership mixing throughout the network, whereas two-mode (or bipartite) networks restrict mixing to across modes (no within-group partnerships are allowed).

4.1.1 Network Estimation and Diagnostics

The first step in any network model is to specify a network structure, including features like size and compositional traits. Here, we construct an empty network of 100 nodes with two races of equal node size.[1] The `network.initialize` function creates an object of class `network` with 100 nodes and no edges (partnerships) between them. The next line creates a vertex (node) attribute called "race", which for simplicity has categories 0 and 1: there are 50 nodes of race 0 and 50 nodes of race 1.

```
nw <- network.initialize(n = 100, directed = FALSE)
nw %v% "race" <- rep(0:1, each = 50)
```

Model Parameters Next, we specify the partnership formation and dissolution model formulas for the STERGM fit. The `formation` object is a right-hand side ERGM formula for edge formation. The `target.stats` are the target statistics that are passed to the ERGM estimation based on calculations from egocentric sampling of partnerships: there are an average of 45 total partnerships at any point in time, 83.1% of those partnerships were of the same

[1] We have used very small networks that are simulated over a short number of time steps for computational efficiency in building this tutorial vignette; in running your own models, it is suggested that you increase both as necessary depending on your data and scientific questions.

race, 36 nodes have no partners (a degree of zero), and 18 nodes have two or more ongoing partnerships. The dissolution object specifies that the partnership dissolution is a fixed (offset) parameter set below resulting in a homogeneous exponential dissolution model.

```
formation <- ~ edges + nodematch("race") + degree(0) + concurrent
target.stats <- c(45, 37.4, 36, 18)
dissolution <- ~ offset(edges)
```

The fixed dissolution coefficient is calculated based on the average duration of partnerships. Here, the average is indicated as 20 months. The `dissolution.coefs` function performs the necessary log transformation of the duration vector given the dissolution model specified above. See the additional `EpiModel` vignette for Network Utility Functions for more information. In brief, the function provides the transformed coefficient here; a coefficient adjusted for exogeneous partnership dissolutions due to death is also possible but not used in this closed-population model (therefore the crude and adjusted coefficients are the same).

```
duration <- 20
coef.diss <- dissolution.coefs(dissolution, duration)
coef.diss

Dissolution Coefficients
========================
Dissolution Model: ~offset(edges)
Edge Duration: 20
Adjusted Coefficient: 2.944
Crude Coefficient: 2.944
```

Model Fit The `epiNet.est` function performs both the STERGM estimation and basic diagnostics for assessing the model fit. For this function, it is necessary to specify the base network object, the formation and dissolution formulas, the target statistics, and dissolution coefficient. By default, the temporal model is estimated using an approximation of a static ERGM to a STERGM fit using a correction to the formation coefficients that are within the dissolution model. This so-called "Edges Dissolution" correction is often more computationally efficient than the full STERGM fit, which may be requested by specifying `edapprox=FALSE`.

```
est1 <- epiNet.est(nw,
                   formation,
                   dissolution,
                   target.stats,
                   coef.diss)
```

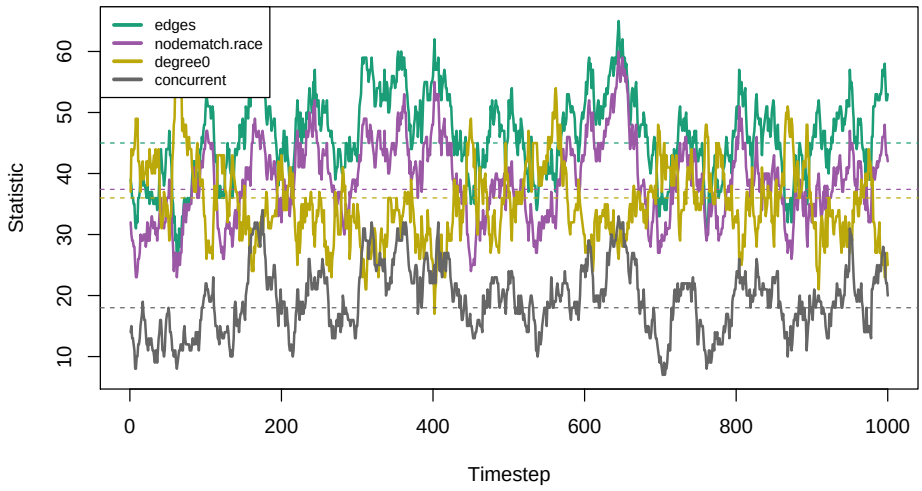
By default, the diagnostic statistics will be run, with the diagnostic network statistics set as the formation formula. To request alternative network statistics, see the help file for the `stats.formula` argument for the function. It will take any right-hand sided ERGM formulation for summary statistics.

Model Diagnostics Printing the object will show the basic diagnostic summaries of average fit to the target statistics and dissolution coefficients. By default, the function will simulate the complete network from the model fit one time for a series of 1000 time steps (the time steps may be changed). Then, mean and standard deviations are calculated for the time series simulation.

```
est1

EpiModel Object
=======================
Model class: epiNet.est
Esimation Method: ERGM with Edges Approximation

ERGM Model Form
-----------------------
Formation: ~edges + nodematch("race") + degree(0) + concurrent
Dissolution: ~offset(edges)
Constraints: ~.

Formation Diagnostics
-----------------------
               targets stats.means stats.sd
edges             45.0       46.11    6.852
nodematch.race    37.4       38.70    6.570
degree0           36.0       35.20    6.302
concurrent        18.0       19.26    5.493

Duration Diagnostics
-----------------------
  target.dur sim.mean.dur   sim.sd.dur
       20.00        19.47        18.75
```

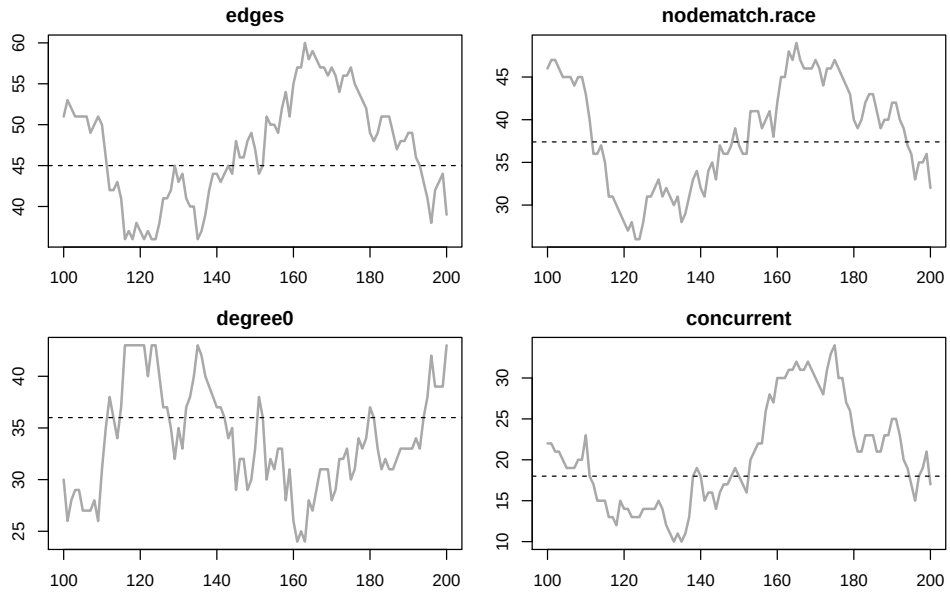
Plotting the object will show the time-varying target statistics for the formation fit over time. On the x-axis is the 1000 time steps that the dynamic model has been run, and on the y-axis are the target values for the formation summary statistics. For those model diagnostics with a matching target statistic (it is certainly possible to request diagnostics for a network statistic that is not part of the formation model), a single dotted line shows the values (compare these to the `target.stats` set above).

```
plot(est1)
```



By default, the plotting function tries to be smart about when to plot everything together and when to separate into distinct plots, but these can also be controlled manually using the `plots.joined` argument. Here we use separate plots for each of the four statistics, and only request 100 time steps of summary statistics. The model may appear to show poor fit here, but note that the time range is shorter and the y-axis scale is much narrower for each plot compared to the joined plot above.

```
plot(est1, plots.joined = FALSE, dx.start = 100, dx.end = 200)
```



4.1.2 Network Simulation

In an independent network epidemic model, the dynamic networks can be simulated first, before running the disease transmission simulation. The `epiNet.simNet` function does just that, with the input of the `epiNet.est` fitted model as above, the desired number of time steps, and number of network simulations. Here, we run 10 network simulations based on the fit above for 50 time steps.

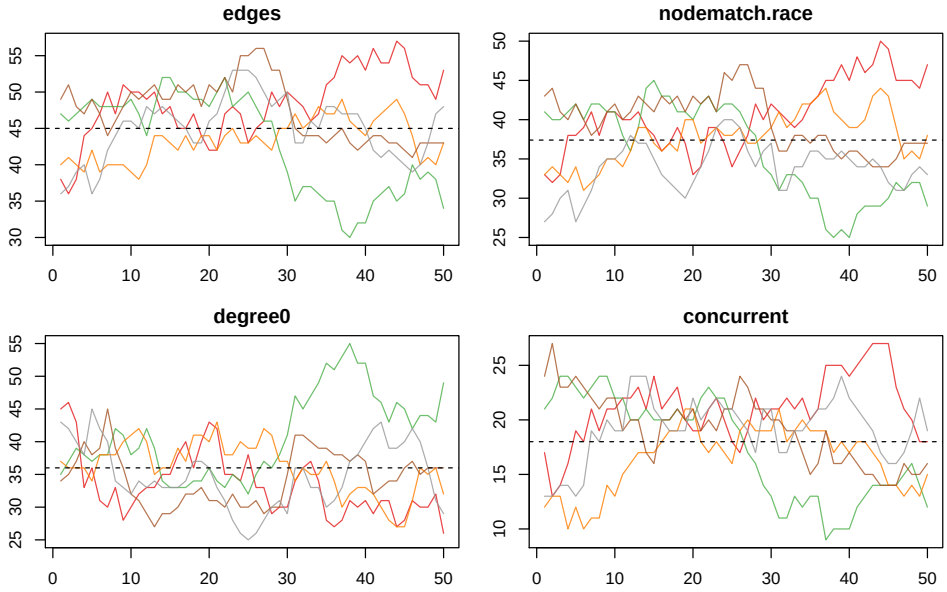
```
nwsims <- epiNet.simNet(est1, nsteps = 50, nsims = 5)
```

Printing the object will show the structure of the network simulation output object, including the number of simulations and time steps per simulation. Also included is information on the base network object, which is the static network from which the dynamic networks are simulated. This is helpful to see, for example, the overall network structure and names of the vertex attributes.

```
nwsims

EpiModel Object
=======================
Model class: epiNet.simNet

NW Simulation Summary
-----------------------
No. simulations: 5
No. time steps: 50

Base Network Object
----------------------
 Network attributes:
  vertices = 100
  directed = FALSE
  hyper = FALSE
  loops = FALSE
  multiple = FALSE
  bipartite = FALSE
  total edges= 45
    missing edges= 0
    non-missing edges= 45

 Vertex attribute names:
    race vertex.names

No edge attributes
```

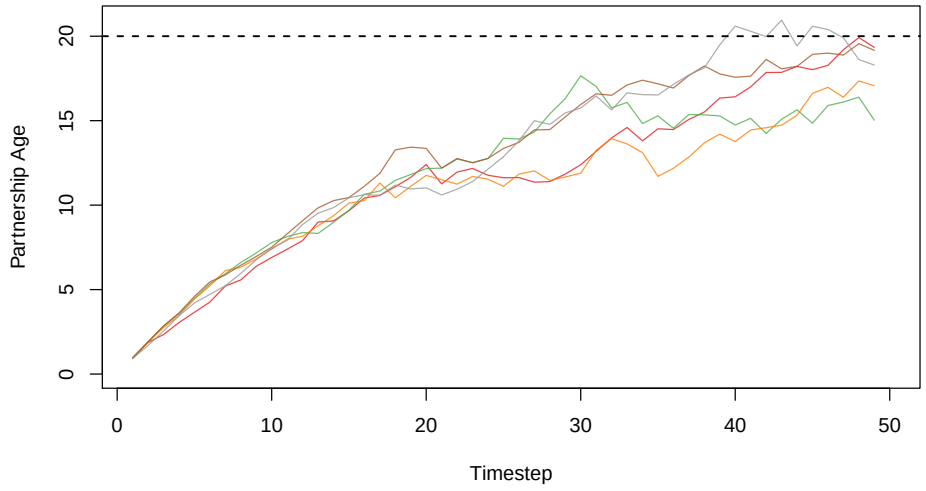
Plotting the network simulation now will show the results from each of the simulations over time, with the same sort of formation statistic output as in the diagnostic plots of the `epiNet.est` object. Note that simulating network multiple times and plotting the network simulation output like this provides additional diagnostic tools on top of those in `epiNet.est`.

```
plot(nwsims, plots.joined = FALSE)
```



An additional plot is available for `epiNet.simNet` objects by specifying a duration plot. Here, the average age of all partnerships up until time t is shown for all times from 1 to the number of time steps specified in the network simulation. In the model fit, we specified that the average duration should be 20 time steps (shown in the dotted line); since all partnerships start at time 1 (actually, they are left-censored at this step), the average age of partnerships at that time is 0. Over time, the average age should approximate the target duration (it does if we were to run our simulation out longer).

```
plot(nwsims, type = "duration")
```



4.1.3 Disease Simulation on the Network

The final step is for the disease transmission process to be simulated over the dynamic simulated networks. The disease simulation function is `epiNet.simTrans`.

For this tutorial, we model disease as a basic SIS epidemic process in which there is a constant transmission probability given contact and a recovery rate given infection. The `sims.per.nw` argument governs how many disease simulations per simulated partnership network should be run. By default, one disease simulation per simulated network is conducted. Since we run the simulation only for 50 time steps, we set the transmission probability to be quite high.

```
sim1 <- epiNet.simTrans(nwsims,
                        type = "SIS",
                        sims.per.nw = 1,
                        trans.rate = 0.5,
                        act.rate = 3,
                        rec.rate = 0.1,
                        i.num = 10)
```

In contrast to the `act.rate` parameter of `epiDCM` and `epiICM`, here the parameter means the **average number of acts within a partnership per unit time**. It does not govern how frequently new partnerships are formed, which is set by the dissolution coefficients in the network model estimation stage. Therefore, the final transmission probability per partnership per unit time is based on a simple function of the transmission probability per act and number acts: $1 - ((1 - \tau)^{\alpha})$ where τ is the `trans.rate` and α is the `act.rate`.

Also in contrast to the other model classes, we set the initial state sizes only for the infected compartment, with `i.num`. The total population size thus the number of susceptibles

at baseline is calculated as a function of the network size that was set in the first step. The initial infected state may be specified either as a deterministic or randomly generate number or prevalence, or by a deterministic vector of specific nodes that are infected.

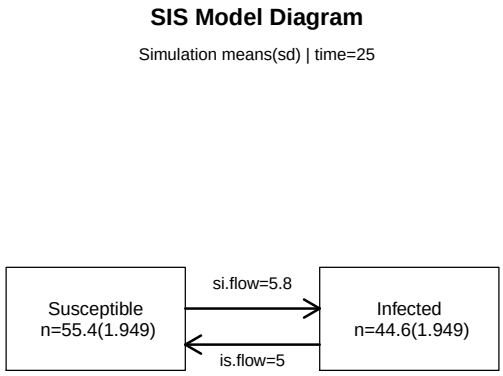
Summaries Similar to `epiDCM` and `epiICM` models, printing the object shows the basic structure and output from the disease simulation model. Note that this shows all the available compartments and flows. Also available in network models are the networks and transmission data frames, which we explain below.

```
sim1

EpiModel Object
=======================
Model class: epiNet.simTrans

Simulation Summary
-----------------------
Model type: SIS
No. simulations: 5
No. time steps: 50
No. NW modes: 1

Model Output
-----------------------
Compartments: s.num i.num
Flows: si.flow is.flow
Networks: sim1 ... sim5
Transmissions: sim1 ... sim5
```

Similar to `epiICM` class models, epidemic statistics may be obtained using the summary function. This summary shows the mean and standard deviation of simulations at time step 25, and also generates a compartment plot showing the same information at this time step visually.

```
summary(sim1, time = 25, comp.plot = TRUE)


EpiModel Summary
=======================
Model class: epiNet.simTrans

Simulation Details
-----------------------
```

```
Model type: SIS
No. simulations: 5
No. time steps: 50
No. NW modes: 1

Model Statistics
--------------------------------
Time: 25
--------------------------------
             mean       sd     perc
Suscept.     55.4    1.949    0.554
Infect.      44.6    1.949    0.446
Total       100.0    0.000    1.000
S -> I        5.8    3.899       NA
I -> S        5.0    3.162       NA
--------------------------------
```



Extraction Similar to the `epiICM` class models, means, standard deviations, and individual simulation run data is easily extracted using the `as.data.frame` function. The default as before is to output the means, but here we show how to extract the model values from the second simulation only.

```
head(as.data.frame(sim1, out = 'vals', sim = 2))

  time s.num i.num si.flow is.flow num
1    1    85    15       0       0 100
```

```
2    2    79    21       7       1 100
3    3    73    27      10       4 100
4    4    67    33       8       2 100
5    5    64    36       6       3 100
6    6    62    38       5       3 100
```

The simulated `networkDynamic` objects with type-specific partnership and disease infection status information are stored under the `network` list in the main model object. They may be extracted and stored to an external object for further analysis.[2]

```
( nw1 <- sim1$network$sim1 )

NetworkDynamic properties:
  distinct change times: 52
  maximal time range: -Inf to Inf

Includes optional net.obs.period attribute:
 Network observation period info:
  Number of observation spells: 2
  Maximal range of observations: 1 to 51
  Temporal mode: discrete
  Time unit: step
  Suggested time increment: 1

 Network attributes:
  vertices = 100
  directed = FALSE
  hyper = FALSE
  loops = FALSE
  multiple = FALSE
  bipartite = FALSE
  net.obs.period: (not shown)
  total edges= 170
    missing edges= 0
    non-missing edges= 170

 Vertex attribute names:
    active race vertex.names
```

[2]Note that there are some intended redundancies in data built into network models to facilitate analysis and visualization purposes. In running these models for research purposes, one should consider which information it is necessary to store in the output object, as this may increase the computational efficacy for the simulations. See the help file for `epiNet.simTrans` to see how to toggle these storage options.

```
 Edge attribute names:
    active
```

A listing of the disease transmissions over time is available through the transmissions data frame. The columns show the infection time, infecting partner, susceptible partner, edge ID number, the time at which the infecting partner was him/herself infected, the associated `trans.rate` parameter, and final transmission probability that is a function of the `trans.rate` and `act.rate`.

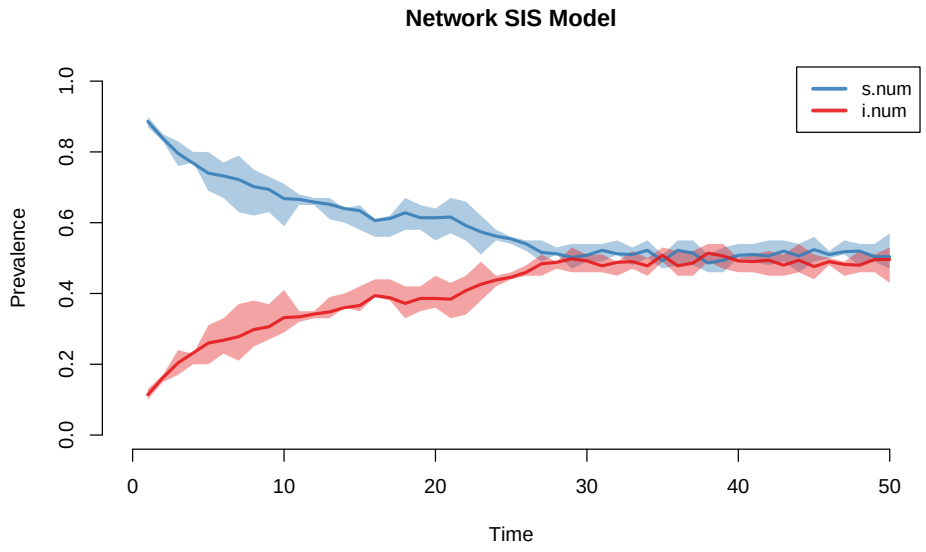
```
trans1 <- sim1$trans$sim1
head(trans1, 10)
```

	time	inf	sus	infdeg	susdeg	inft	trans.rates	act.rates	tprob
1	2	57	72	1	2	3	0.5	3	0.875
2	2	26	20	1	3	38	0.5	3	0.875
3	2	79	77	2	3	20	0.5	3	0.875
4	2	1	38	1	2	19	0.5	3	0.875
5	2	59	54	1	1	45	0.5	3	0.875
6	3	20	11	3	1	1	0.5	3	0.875
7	3	77	88	3	1	1	0.5	3	0.875
8	3	20	34	3	2	1	0.5	3	0.875
9	3	77	73	3	1	1	0.5	3	0.875
10	3	1	38	1	2	20	0.5	3	0.875

Plotting There are two main ways to plot the results of this network disease model: the standard line plots showing the epidemic trajectories similar to the individual contact model plots (see Section 3), and static network plots showing a snapshot of the partnership network with information on disease status.

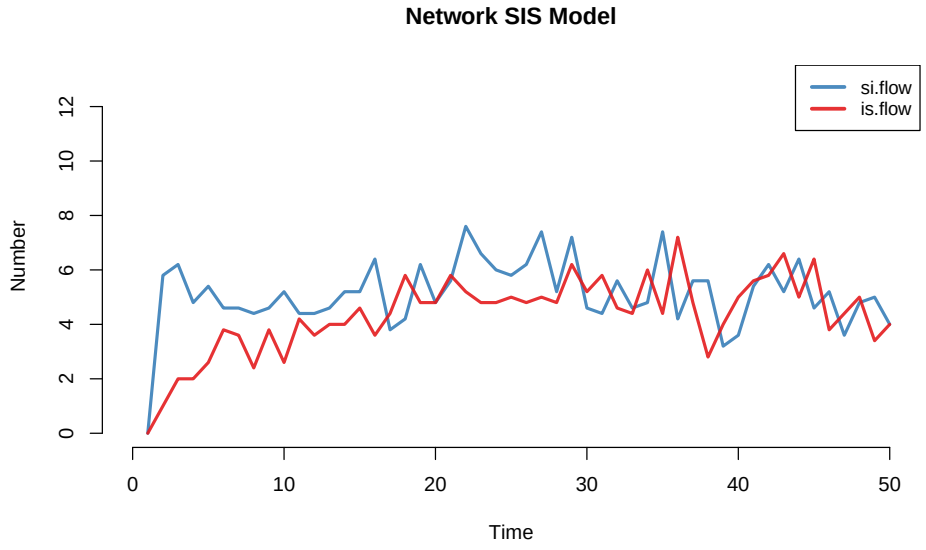
By default, the plot shows disease trajectories (compartment sizes) over time. The plotting function uses the same defaults for the stochastic model results as in `epiICM` class models, and therefore, the same arguments options apply. In a one-mode network model, the default is to plot the mean and inter-quartile range of all compartments in the model for the full length of the model was run.

```
plot(sim1)
```



To plot the means of disease *incidence* and new recoveries over time, it is necessary to specify the outcomes of interest using the `y` argument, and also to set the population fraction denominator to `FALSE`. The `si.flow` is the number of transitions from susceptible to infected at that time, and the `is.flow` is the number of transitions back from infected to susceptible.

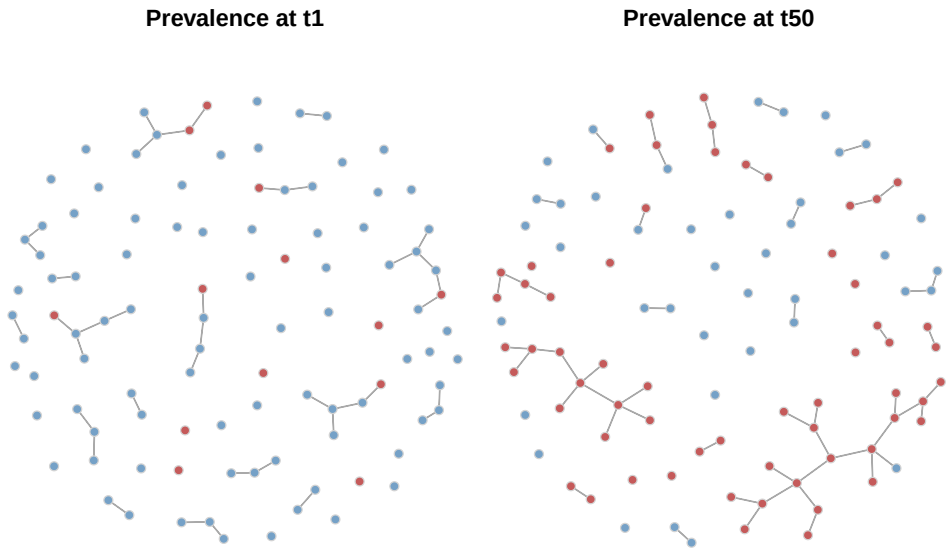
```
plot(sim1, y = c("si.flow", "is.flow"), leg = TRUE)
```



Plotting the static network at different time steps and over different simulations can show the patterns of partnership formation and disease spread over those partnerships. By default, the plot type is the compartment size line plots (`type="sim"`), but we set the network plot by specifying `type="network"`.

Below, we plot two time points from the same simulation, at time steps 1 and 50. The `col.inf` argument takes care of the color coding for easy plotting of the infected (in red) versus negative (in blue). Note that the `zeromarg` argument sets the network plot margins to zero by default for better visualization of dense networks, but *must* be toggled `FALSE` in plots when a non-zero margin is needed to allow for plot titles.

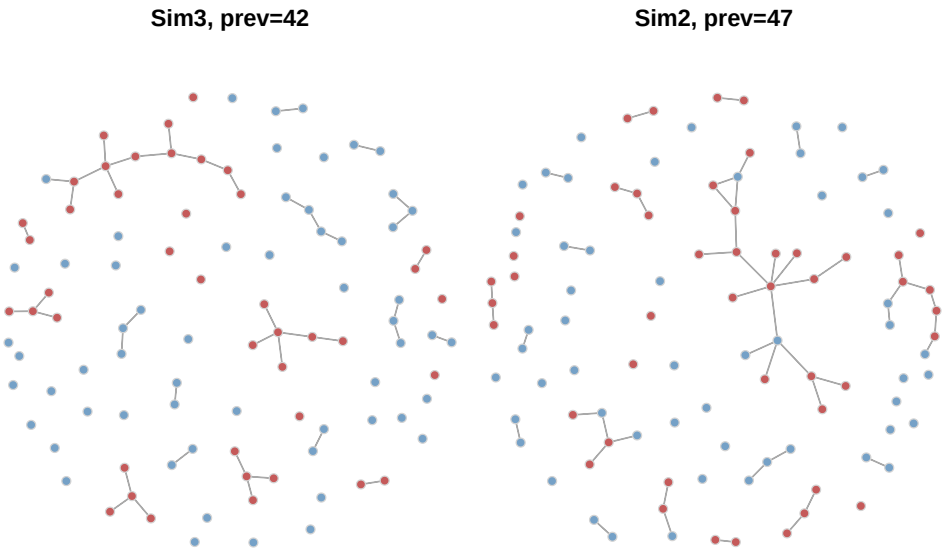
```
par(mfrow = c(1,2), mar = c(0,0,2,0))
plot(sim1, type = "network", at = 1, col.inf = TRUE,
     zeromarg = FALSE, main = "Prevalence at t1")
plot(sim1, type = "network", at = 50, col.inf = TRUE,
     zeromarg = FALSE, main = "Prevalence at t50")
```



Given the stochasticity of the model, another interesting network plot would show the variability in simulations at a specific time step. Here, we plot the two simulations with the lowest and highest prevalence at time step 25. First, we have to find the simulation number with the lowest and highest values of infecteds at that time, then those extract those values.

```
time <- 25
lsim <- which.min(sim1$i.num[time, ])
hsim <- which.max(sim1$i.num[time, ])
lprev <- sim1$i.num[time, lsim]
hprev <- sim1$i.num[time, hsim]
par(mfrow = c(1,2), mar = c(0,0,2,0))
plot(sim1, type = "network", at = 25, sim = lsim,
     col.inf = TRUE, zeromarg = FALSE,
     main = paste("Sim", lsim, ", prev=", lprev, sep=""))
```

```
plot(sim1, type = "network", at = 25, sim = hsim,
     col.inf = TRUE, zeromarg = FALSE,
     main = paste("Sim", hsim, ", prev=", hprev, sep=""))
```



As this example shows, it is possible to extract, analyze, and plot many objects embedded within the larger `epiNet.simTrans` object to investigate the epidemic trajectory over time and at specific time points.

4.2 Dependent Bipartite SI Model

In the second network model example, we introduce dependence between the disease simulation and the network structure. The main forms of dependence currently implemented in `EpiModel` are demographic transition processes for births and deaths. This is integrated into network models similarly to the stochastic transition processes in `epiICM` class models, in which the number of new births and deaths at each time step is determined as a random draw from a Poisson distribution with the rate parameter set by the parameter arguments in the model. These processes are handled in a specialized way in network models as nodes and their associated edges have to be "activated" and "deactivated" over time to allow for correct simulation of the partnership network. Other forms of network/disease dependence, including the dependence between disease infection and probability of partnership formation (i.e., serosorting), will be added in future `EpiModel` releases.

Additionally in this example, we simulate an epidemic model here within a bipartite network. Such a network may be used to represent purely heterogeneous mixing as in heterosexual-only models of disease transmission. This is *not the only way* to represent this form of mixing, but it does facilitate the modeling of mode-specific network terms (e.g., different degree distributions) and disease features as one might need in sex-differentiated epidemics.

4.2.1 Network Estimation and Diagnostics

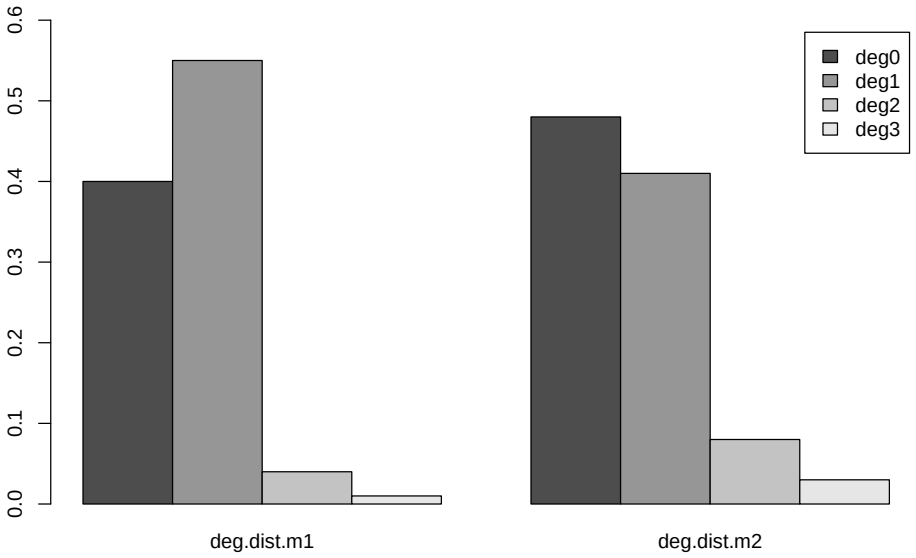
As with the independent network model in Section 4.1, the first step is to specify a network structure, including features like size and nodal attributes. Here, we construct an empty network of 100 nodes, with 50 in each mode. One might conceive of the first mode as females and the second mode as males.

```
num.m1 <- 50
num.m2 <- 50
nw <- network.initialize(num.m1 + num.m2,
                         bipartite = num.m1, directed = FALSE)
```

Degree Balancing For our target statistics, we will use sex-specific degree distributions. Similar to the contact balancing requirements in acts described in Section 2.4, it is necessary to balance the number of partnerships implied by a degree distribution in one mode to that of another. This is particularly an issue when one sex tends to report higher average numbers of partners in surveys used to calculate model parameters.

In `EpiModel`, we can check that the implied number of partnerships match given different degree distributions specified in vectors of fractional values using the `bip.degdist.check` function. Consider two degree distributions in which the proportion of females currently having 0, 1, 2, or 3 partners is 40%, 55%, 4%, and 1%, respectively. The distribution for men may be different within each degree but the total number of partnerships must match. Below we plot the mode-specific degree distributions.

```
deg.dist.m1 <- c(0.40, 0.55, 0.04, 0.01)
deg.dist.m2 <- c(0.48, 0.41, 0.08, 0.03)
par(mar=c(3,3,2,1))
barplot(cbind(deg.dist.m1, deg.dist.m2), beside = TRUE,
        legend.text = paste("deg", 0:3, sep=""), ylim = c(0,0.6))
```



Given these fractional degree distributions and the mode sizes set in when initializing the network, the `bip.degdist.check` function checks for balance. The output is table with the number of nodes with each degree, as well as the total number of edges in that mode. The latter must match across modes to ensure degree balance. See the additional `EpiModel` vignette on Network Utility Functions for more information.

```
bip.degdist.check(num.m1, num.m2,
                  deg.dist.m1, deg.dist.m2)

Bipartite Degree Distribution Check
===========================================
        m1.dist   m1.cnt   m2.dist   m2.cnt
Deg0       0.40     20.0      0.48     24.0
Deg1       0.55     27.5      0.41     20.5
Deg2       0.04      2.0      0.08      4.0
Deg3       0.01      0.5      0.03      1.5
TOTAL      1.00     33.0      1.00     33.0
===========================================
** distributions balanced
```

Model Fit As with the independent network model, we next specify the partnership formation and dissolution model formulas for the STERGM fit (see Section 4.1 for more information). The target statistics for partnership formation are the overall number of partnerships at one point in time, the number of nodes in the first mode with no partners or only one partner, and the similar degree terms for the second mode nodes. The dissolution model is a homogeneous exponential decay with mean partnership duration of 25 time units. Note

that we need to specify a background death rate using the `d.rate` parameter, to account for the fact that death presents an exogenous competing risk to partnership dissolution on top of the estimated, endogenous average duration. See the HTML vignette, *EpiModel Network Utility Functions*, for more details.

```
formation <- ~ edges + b1degree(0:1) + b2degree(0:1)
target.stats <- c(33, 20, 27.5, 24, 20.5)
dissolution <- ~ offset(edges)
coef.diss <- dissolution.coefs(dissolution, duration=25, d.rate=0.01)
coef.diss

Dissolution Coefficients
========================
Dissolution Model: ~offset(edges)
Edge Duration: 25
Adjusted Coefficient: 3.866
Crude Coefficient: 3.178
```

The network estimation process for this model uses the `epiNet.est` function in the exact same way as our independent model in Section 4.1. For this example, we also show how to obtain diagnostic formation statistics that are not included in the main formation model: although we only target degree of zero and one for the model, we would like to monitor the average number of nodes in each mode with two, three, four, and five partners. To do that, we specify a right-hand sided formation formula called `dx.stats`, which we enter into the `epiNet.est` in the `stats.formula` argument. Note also that we set the randomization seed here; in testing the model fitting process, it takes quite some time for the Markov-chain monte carlo (MCMC) estimation process to converge (see the help documentation for the `tergm` package for details on these processes).

```
dx.stats <- ~ edges + b1degree(0:5) + b2degree(0:5)
set.seed(12345)
est2 <- epiNet.est(nw,
                   formation,
                   dissolution,
                   target.stats,
                   coef.diss,
                   stats.formula = dx.stats)
```

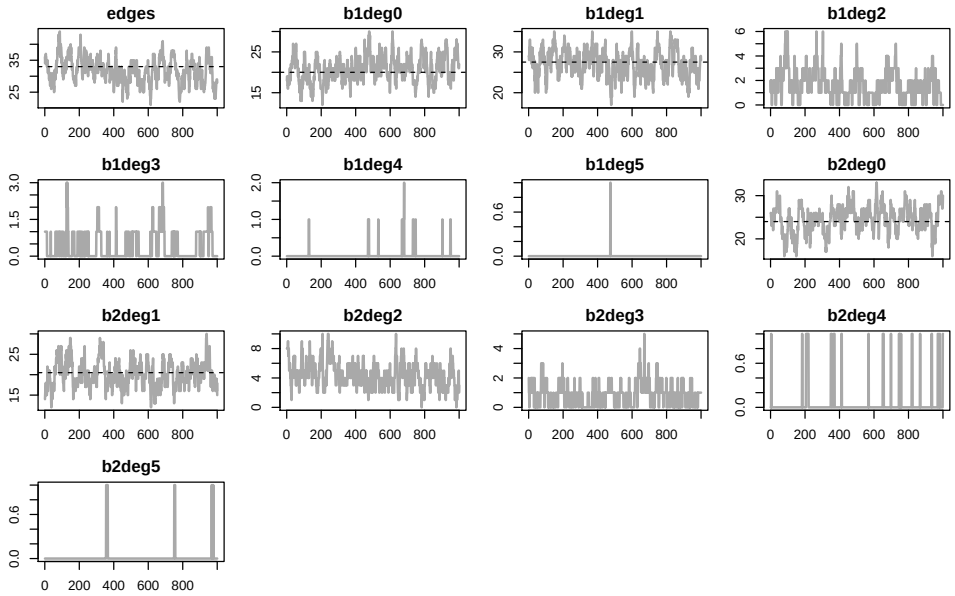
Model Diagnostics Printing the object will show the basic diagnostic summaries of average fit to the target statistics. Note that the diagnostic statistics for the terms that were monitored but not entered into the model (degree terms above 2 for both modes) show as `NA` in the target statistics column but summary means are provided. This also works in converse

(it is possible to diagnostically monitor a set of network statistics that excludes terms from the formation formula, although this is not recommended).

```
est2

EpiModel Object
=======================
Model class: epiNet.est
Esimation Method: ERGM with Edges Approximation

ERGM Model Form
-----------------------
Formation: ~edges + b1degree(0:1) + b2degree(0:1)
Dissolution: ~offset(edges)
Constraints: ~.

Formation Diagnostics
-----------------------
       targets stats.means stats.sd
edges     33.0      31.787    3.912
b1deg0    20.0      20.780    3.229
b1deg1    27.5      27.118    3.307
b1deg2      NA       1.667    1.225
b1deg3      NA       0.407    0.608
b1deg4      NA       0.026    0.165
b1deg5      NA       0.002    0.045
b2deg0    24.0      24.544    3.010
b2deg1    20.5      20.153    3.127
b2deg2      NA       4.387    1.768
b2deg3      NA       0.828    0.777
b2deg4      NA       0.066    0.248
b2deg5      NA       0.020    0.140

Duration Diagnostics
-----------------------
  target.dur sim.mean.dur   sim.sd.dur
       25.00        23.40        23.75
```

Plotting the object will show varying target statistics fit over time. By default, the plots are split because of the large number of statistics. Each plot has a different y-axis scale. There are very few nodes of either mode with three or more partnerships at any one time.

```
plot(est2)
```



4.2.2 Disease Simulation on the Network

In a dependent network epidemic model (disease status influences partnership formation and dissolution, and vice versa), the dynamic networks are all simulated concomitantly with the main disease transmission simulation function, `epiNet.simTrans`. Therefore, no simulation of a series of partnership networks is needed as with independent models (see Section 4.1.2). In this model, we incorporate births and deaths into the simulation to allow for an open population.

The main model parameters are as follows. We set the disease initial prevalence to 10%; the exact number of nodes infected will be based on random draws from a binomial distribution summarized by that prevalence. Similar to `epiDCM` and `epiICM`, different transmission probabilities by mode are allowed to incorporate a higher susceptibility for disease for one mode.

Additionally, we set the birth rate to be based on the death rate for susceptibles to achieve a relatively stable equilibrium prevalence (which we will not see here because of the short time steps). Since the birth rate is calculated based on the size of the mode-one population in bipartite simulations (again, with females implied as this mode), we multiply it by two. Finally, the death rate for the infecteds is slightly higher than for the susceptibles.

```
i.num <- 10
trans.rate <- 0.5
trans.rate.m2 <- 0.1
b.rate <- 2/100
ds.rate <- 1/100
```

```
di.rate <- 1/90
```

Note that for the disease simulation function, we pass the network estimation object directly from `epiNet.est` and specify `vital=TRUE` to include these demographic processes. Also, the `sims.per.nw` has a different meaning than in independent simulations: here it is simply the number of independent disease simulations requested, since each simulation uses a newly simulated network because of the resimulation at each time step.

```
sim2 <- epiNet.simTrans(est2,
                        type = "SI",
                        vital = TRUE,
                        i.num = i.num,
                        trans.rate = trans.rate,
                        trans.rate.m2 = trans.rate.m2,
                        b.rate = b.rate,
                        ds.rate = ds.rate,
                        di.rate = di.rate,
                        sims.per.nw = 3,
                        nsteps = 50)
```

Printing the object shows that we have an `EpiModel` object that is an SI network model with three simulations over 50 time steps with a two-mode network. The model output now includes mode-specific output for compartments and flows, including the birth and death transitions. The model output may be summarized using the `summary` function and extracted using the `as.data.frame` function, similar to the independent model example in Section 4.1.

```
sim2

EpiModel Object
=======================
Model class: epiNet.simTrans

Simulation Summary
-----------------------
Model type: SI
No. simulations: 3
No. time steps: 50
No. NW modes: 2

Model Output
-----------------------
Compartments: s.num i.num s.num.m2 i.num.m2
```

```
Flows: si.flow b.flow ds.flow di.flow si.flow.m2 b.flow.m2
ds.flow.m2 di.flow.m2
Networks: sim1 ... sim3
Transmissions: sim1 ... sim3
```

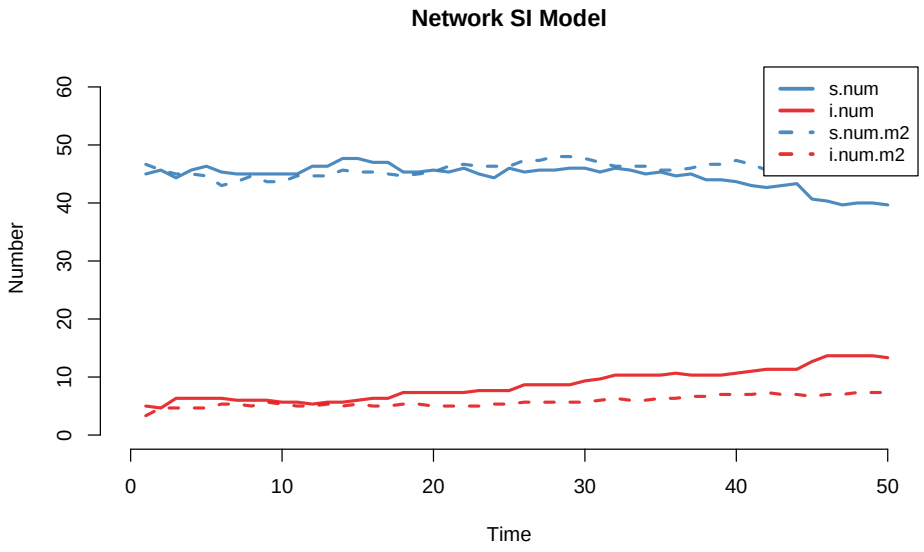
As with the independent network model, a listing of the transmissions by time is available through the transmissions data frames stored on the object. The columns show the infection time, infecting partner, susceptible partner, edge ID number, infection time of the infecting partner, the `trans.rate` parameter tied to that mode, and final transmission probability that is a function of the `trans.rate` and `act.rate`.

```
trans1 <- sim2$trans$sim1
head(trans1)

  time inf sus infdeg susdeg inft trans.rates act.rates tprob
1    2  F8  M1      3      1  172         0.1         1   0.1
2    2 F45 M12      1      2  184         0.1         1   0.1
3    2 F20 M28      1      2  172         0.1         1   0.1
4    3 M12  F1      2      2    1         0.5         1   0.5
5    3 M38 F47      1      1   13         0.5         1   0.5
6    3 M33 F40      1      1   89         0.5         1   0.5
```

Note that the transmission rates are higher when the mode 1 nodes are the susceptible partner, since the definition of the rates is the probability of infection to that mode given contact with an infected of the other mode. Also, note that the ID numbers for the nodes begin the a F or M prefix: because of the way that the ID system works for dynamic bipartite networks, it is necessary to use this specialized ID that persists over time (see the help file for the `init.pids` function).

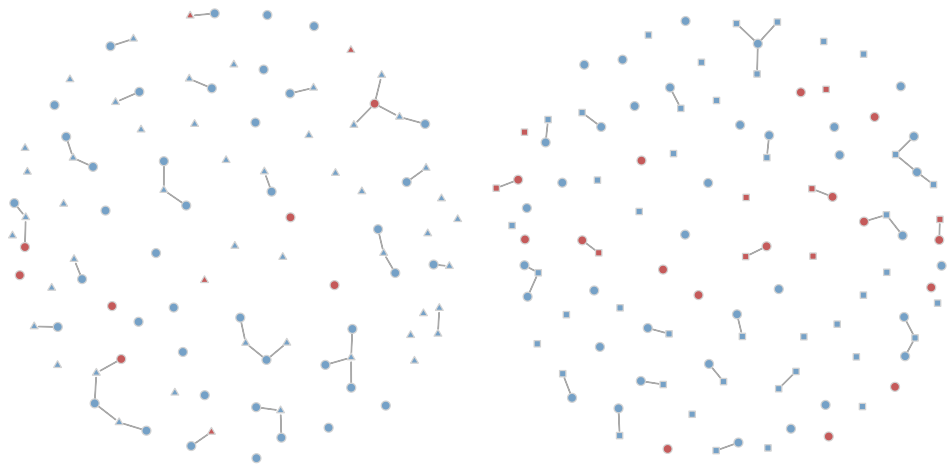
Plotting The plotting for dependent disease simulations is the same as for independent simulations as described in Section 4.1.3. Since this is a bipartite network, the default plot shows the means only of the state sizes over time for each of the modes. Here we plot the absolute numbers from the simulations, which is suggested at least for diagnostics after an open population simulation is run because the compartment prevalences may use unexpectedly small denominators if the death rate is misspecified.

```
plot(sim2, popfrac=FALSE)
```



Similar to the static network plots for the independent network model in Section 4.1.3, plotting the network of partnership structure and disease status is made easy with arguments to the generic network plotting functions. In addition to the `col.inf` argument that automatically colors the infected red, bipartite simulations also allow for differential shapes for the modes. Here, one may set the second-mode shapes to either triangle or square using the `shp.bip` argument. Note how one mode only has formed partnerships with nodes of the opposite mode, as signified by the different shapes.

```
par(mfrow=c(1,2))
plot(sim2, type = "network", at = 1, col.inf = TRUE, shp.bip = "triangle")
plot(sim2, type = "network", at = 50, col.inf = TRUE, shp.bip = "square")
```



4.3 Dynamic Visualization with `ndtv`

A dynamic network animation that shows partnership change along with disease transitions is available using the `ndtv` package. More detailed examples are provided in the `ndtv` package vignette.

```
library(ndtv)
```

First, we extract the `networkDynamic` object from the `epiNet.simTrans` object from above. Here we are just pulling the fifth network. The `colorTEA` function that sets up some color attributes based on infection status for the animation (see the help file for this function).

```
nw <- sim2$network$sim1
nw <- colorTEA(nw)
```

The next step is to compute coordinates for all the nodes over time. Here, we set the animation for time steps 1 to 25, although the simulation runs twice as long. Additional options are explained in the `ndtv` vignette.

```
slice.par <- list(start = 1, end = 50, interval = 1,
                  aggregate.dur = 1, rule = "any")
compute.animation(nw, slice.par = slice.par,
                  animation.mode = "MDSJ")
```

The `render.animation` function takes the dynamic coordinates and creates an animation object for the network. The function takes all the standard graphical parameters for `plot.network`.

```
render.par=list(tween.frames = 10,
                show.time = FALSE)
plot.par=list(mar = c(0,0,0,0))
render.animation(nw,
                 render.par = render.par,
                 plot.par = plot.par,
                 vertex.cex = 0.9,
                 vertex.col = "ndtvcol",
                 edge.col = "darkgrey",
                 vertex.border = "lightgrey",
                 displaylabels = FALSE)
```

Finally, the animation may be saved out to a variety of formats, but here is one option, an animated GIF file, that looks good on webpages.

```
saveGIF(ani.replay(),
        ani.width = 600,
        ani.height = 600,
        outdir = getwd())
```

Here's how to save out out the animation to a `mp4` video file.

```
saveVideo(ani.replay(),
          video.name = "EpiModelndtv.mp4",
          other.opts = "-b 5000k",
          clean = TRUE,
          ani.width = 1200,
          ani.height = 1200)
```

5 Future Work

In future releases of `EpiModel`, we will extend the functionality of the software in a number of ways. These include:

- **Adding disease types:** Currently, `EpiModel` model allows for easy modeling of SI, SIR, and SIS disease types with relatively few governing parameters. These model types may be insufficient for more complicated disease structures, such as an HIV model with multiple stages of disease, or an SIRS model in which there is waning immunity after infection. Our plan is to continue to implement basic structures of these models across all three model classes (`epiDCM`, `epiICM`, and `epiNet`) for pedagogical and basic research purposes.
- **Increasing the flexibility of models:** Currently, our models are limited to basic partnership mixing and disease features that we have used in our teaching on modeling. As we build in additional functionality for new model types, we will also increase the flexibility of existing model types by, for example, allowing more than purely dissassortative mixing.
- **Implementing research-level tools:** Mathematical modeling of epidemics is difficult to provide in general software because models often require very specific and complex processes that dictate the partnership/contact process, the disease transitions given contact, and underlying demographic features. Our goal with `EpiModel` is not to provide "out-of-the-box" solutions for all possible model types, but instead to provide documented modules for use in these models.
  - With some general knowledge of programming in R, it should be possible to extend these tools, specifically the network-based modeling tools, to accommodate a variety of research needs.

  - We have modularized the specific processes of simulating stochastic network models in the `epiNet.simTrans` function, and one starting place to start unpacking that main function is the help file for those modules:

```
help('epiNetModules')
```

- **Optimization:** The first goal of this new software was to get it right, and only then can we start working on making it fast. Network models in particular may take substantial computational resources, especially for large networks over long time steps. The *Statnet* team is continually working on optimizing the core packages along with `EpiModel` to reduce computational burdens.