Basic Network Models with EpiModel

Introduction

This tutorial provides a mathematical and theoretical background for stochastic network models, with instructions on how to run the built-in models designed for learning in EpiModel. For information on how to extend these models with new mathematical frameworks, see the related tutorial New Network Models with EpiModel (NewNet.html). If you are new to epidemic modeling, we suggest that you start with our tutorials Basic DCMs with EpiModel (BasicDCMs.html) and Basic ICMs with EpiModel (BasicICMs.html) to get a background in deterministic and stochastic modeling. The material below assumes familiarity with that material.

Network models explicitly represent contact phenomena within and across 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, a network is formed.

Model Framework

EpiModel uses separable-temporal exponential-family random graph models (STERGMs) to estimate and simulate complete networks based on individual-level, dyad-level, and network-level patterns of density, degree, assortivity, and other features influencing edge formation and dissolution. Building and simulating a network-based epidemic models in EpiModel is a multi-step process, starting with estimation of a temporal ERGM and continuing with simulation of a dynamic network and epidemic processes on top of that network.

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 options, please consult the help documentation and vignettes for the **tergm** and **networkDynamic** packages. EpiModel currently supports estimation of network models with summary target statistics for formation that may be estimated using *egocentric network samples:* 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 be parameterized with egocentric network data by inputting summary target statistics for network structures derived from this data, including the distribution of partner numbers at one point in

time, assortivity in partner traits, and other dyadic and network-level features. On top of this, mean statistics summarizing the average duration of partnerships is estimated from empirical data and used to govern partnership dissolution rates.

Model Processes

EpiModel can simulate disease epidemics over these partnership networks by integrating this statistical framework for networks with stochastic transmission processes similar to those featured in `icm` class disease models. Similar to ICMs, 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.

In contrast to DCMs and ICMs, which simulate the epidemic system with one function, network models require multiple steps:

- An empty network structure is initialized;
- A network model is fit and diagnosed; and
- The epidemic processes are simulated on top of a dynamic simulated network consistent with the network structure and model fit.

Independent versus Dependent Models

A key distinction for network models concerns whether two dynamic processes, the 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 certainly impacts disease. *Dependent* network models allow for the epidemiological and demographic processes to influence the network structure. Examples include serosorting – where disease status influences partner selection – and demographic processes (births, deaths, and migration) – where the contact network process must adapt to changing population size and composition.

Model Functions

Simulating network models in EpiModel involves three main functions for both independent and dependent models:

1. `netest` estimates the generative model for the dynamic partnership networks. This function is a wrapper around the `ergm` and `stergm` functions in the **ergm** and **tergm** packages.
2. `netdx` simulates time-series of the dynamic network consistent with the model fit from `netest` to check for model fit to the egocentric target statistics.
3. `netsim` then runs the stochastic epidemic models with a network model fit from `netest` . For independent models, the full dynamic network is simulated at the start of each epidemic simulation. For dependent models, the network is re-simulated at each time step as a function of varying population size and changing nodal attributes.

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

Independent One-Mode SIS Model

In this section, we work through a model of a Susceptible-Infected-Susceptible (SIS) epidemic in a closed population. An example of an SIS disease would be a bacterial sexually transmitted infection such as Gonorrhea, in which persons may acquire infection from sexual contact with an infected partner, and then recover from infection either through natural clearance or through antibiotic treatment. We will use a simplifying assumption of a closed population, in which there are no entries or exits from the network; this may be justified by the short time span over which the epidemic will be simulated.

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 1000 nodes with two races of equal node size. We use small networks that are simulated over a short number of time steps for computational efficiency in this tutorial; research-level models may require larger-scale simulations. The `network.initialize` function creates an empty network object with 1000 nodes but no edges. The next line creates a vertex attribute called `race`, which will have categories 0 and 1: there are 500 nodes of Race 0 and 500 nodes of Race 1.

```
nw <- network.initialize(n = 1000, directed = FALSE)
nw <- set.vertex.attribute(nw, "race", rep(0:1, each = 500))
```

Model Parameters

Next, we specify the partnership formation formula for the network model estimation. The `formation` object is a right-hand side formula, and the `target.stats` are the summary statistics based on egocentric network sampling that summarize the mean values in the formula. The `edges` term is the expected number of edges each time step. The `nodefactor` term specifies the number of nodes of Race 1 that in an edge; using a transformation below, we can use it to express the race-specific mean degree. The term is specified for one less than the number of groups in the attribute because twice the sum of terms for all groups is equal to the number of edges. The `nodematch` is the number of edges between nodes of the same race. The `concurrent` term is the number of nodes that have two or more edges at each time step (i.e., momentary degree of at least 2).

```
formation <- ~edges + nodefactor("race") + nodematch("race") + concurrent
```

To calculate the target statistics, we need to translate from empirical egocentric data to the statistical forms required by ERGMs. Nodes will have a mean degree of 0.5, equivalent to 250 expected edges given the network size (n/2 times mean degree). The mean degree will vary by race: the mean degree of Race 1 will be 0.75, whereas it will be 0.25 for Race 0. The `nodefactor` term requires only the statistic for Race 1; it is the product of the size of Race 1 and their mean degree (500 * 0.75 = 375). For racial mixing, 90% of edges are same-race. The expected number of same-race edges is the product of the number of edges and the probability of a same-race edge (250 * 0.90 = 225). The number of nodes with a concurrent degree is 100, equivalent to 10% of total nodes.

```
target.stats <- c(250, 375, 225, 100)
```

The dissolution model is parameterized from an average edge duration estimated from cross-sectional egocentric data. The dissolution model is parameterized as an offset because the dissolution coefficient is not estimated; it is instead fixed at the value implied by the average edge duration. The dissolution models may be as arbitrarily complex as any ERGM formation models, but given the analytic calculation of coefficients and other complexities, only a limited set of dissolution models are supported

in EpiModel. Our dissolution model will be an “edges” model in which the probability of edge dissolution is homogeneous conditional on edge existence. The average partnership duration is 25 time steps (e.g., months or some other arbitrary unit). The `dissolution_coefs` function takes as input the dissolution formula and average duration and transforms this into a logit coefficient for use in estimation. See the help page of this function to see other supported dissolution models.

```
coef.diss <- dissolution_coefs(dissolution = ~offset(edges), duration = 25)
coef.diss
```

```
Dissolution Coefficients
========================
Dissolution Model: ~offset(edges)
Target Statistics: 25
Crude Coefficient: 3.178054
Mortality/Exit Rate: 0
Adjusted Coefficient: 3.178054
```

The output from this function indicates both an adjusted and crude coefficient, which are equivalent in this case. In the next example, they will differ.

Model Fit

In EpiModel, network model estimation is performed with the `netest` function, which is a wrapper around the estimation functions in the `ergm` and `tergm` packages. The function arguments are as follows:

```
function (nw, formation, target.stats, coef.diss, constraints,
    coef.form = NULL, edapprox = TRUE, output = "fit", set.control.ergm,
    set.control.stergm, verbose = FALSE)
NULL
```

The four arguments that must be specified with each function call are:

- `nw` : an initialized empty network.
- `formation` : a RHS formation formula.
- `target.stats` : target statistics for the formation model.
- `coef.diss` : output object from `dissolution_coefs` , containing the dissolution coefficients.

Other arguments that may be helpful to understand when getting started are:

- `constraints` : sets the model constraints, passed to `ergm` and `stergm` (see `help("ergm")`).
- `coef.form` : sets the coefficient values of any offset terms in the formation model.
- `edapprox` : if `TRUE` , uses a static ERGM with coefficient adjustment (details below) rather than a full STERGM fit for the sake of computational efficiency.
- `nonconv.error` : if `TRUE` , will error if the model has not converged after the specified number of MCMC iterations (the default of the `ergm` function is to allow non-converged output to be returned).

In the `netest` function, the network model may be estimated using one of two methods:

1. **Direct method**: uses the functionality of the `tergm` package to estimate the separable formation and dissolution models for the network.

2. **Approximation method**: uses `ergm` estimation for a cross-sectional network (the prevalence of edges) with a manual adjustment of the edges coefficient to account for dissolution (i.e., transformation from prevalence to incidence). This approximation method may introduce bias into estimation in certain cases (high density and short durations) but these are typically not a concern for the low density cases in epidemiologically relevant networks. As a general rule, the approximation bias is minimal when the edge duration greater than 20 time units and the mean degree is less than 1. The benefit of using the approximation is computational efficiency, since direct STERGMs on low density/short duration networks are difficult and time-consuming to fit.

Within `netest`, the direct method is specified by `edapprox=FALSE` and the indirect by `edapprox=TRUE` (the default). Because we have a dyadic dependent model, MCMC will be used to estimate the coefficients of the model given the target statistics.

```
est1 <- netest(nw, formation, target.stats, coef.diss, edapprox = TRUE)
```

Model Diagnostics

There are two forms of model diagnostics for a dynamic ERGM fit with `netest`: static and dynamic diagnostics. When the approximation method has been used, static diagnostics check the fit of the cross-sectional model to target statistics. Dynamic diagnostics check the fit of the model adjusted to account for edge dissolution. When running a dynamic network simulation as we do with EpiModel, it is good to start with the dynamic diagnostics, and if there are fit problems, work back to the static diagnostics to determine if the problem is due to the cross-sectional fit itself or with the dynamic adjustment (i.e., the approximation method). A proper fitting ERGM using the approximation method does not guarantee well-performing dynamic simulations.

For this tutorial, we will examine dynamic diagnostics only. These are run with the `netdx` function, which simulates from the model fit object returned by `netest`. One must specify the number of simulations from the dynamic model and the number of time steps per simulation. Here, we simulate the model 5 times over 500 time steps. Choice of both simulation parameters depends on the stochasticity in the model, which is a function of network size, model complexity, and other factors.

By default, the network statistics to be diagnosed are those in the formation formula of the network model, although the `nwstats.formula` may be used to monitor any set of statistics. In this example, we set a formula for diagnostic statistics that varies slightly from the formation formula. The difference is the specification of `base=0` in the `nodefactor` term. If we were to specify this in the formation formula itself, there would be collinearity with the edges term. But for diagnostics, this is a useful tool to quickly see the race-specific mean degrees.

```
dx <- netdx(est1, nsims = 5, nsteps = 500,
            nwstats.formula = ~edges + nodefactor("race", base = 0) +
                               nodematch("race") + concurrent)
```

Printing an object output from `netdx` will show summary tables of the simulated network statistics against the target statistics. The mean and sd values for the formation diagnostics are the mean and standard deviations of those statistics across all simulations and time steps. The simulated edges is slightly higher than targeted, but within a standard deviation of that target. There is no target statistic for Race 0 in the `nodefactor` term because it was not in the formation formula.

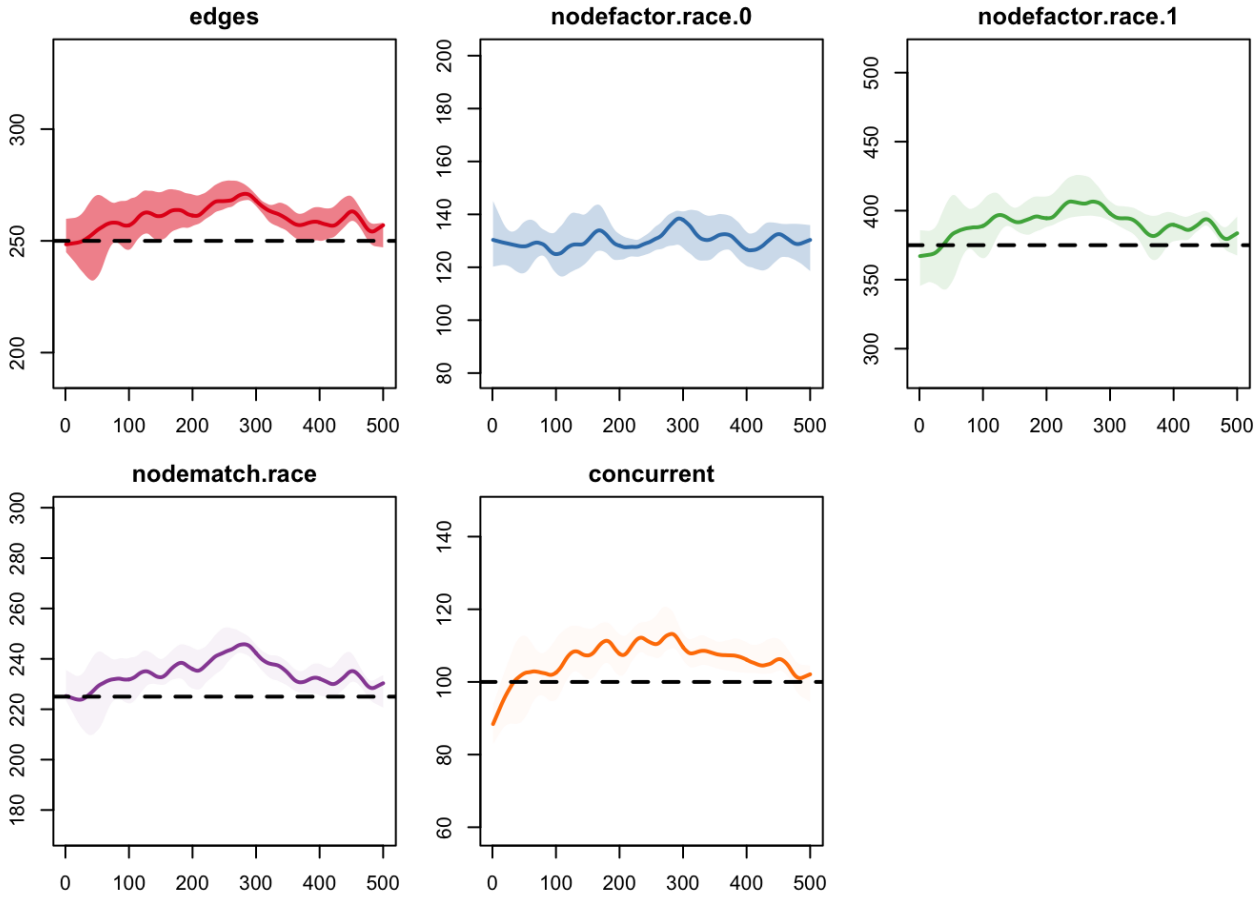
```
dx
```

```
EpiModel Network Diagnostics
============================
Diagnostic Method: Dynamic
Simulations: 5
Time Steps per Sim: 500

Formation Diagnostics
---------------------
                  Target Sim Mean Pct Diff Sim SD
edges                250  260.275    0.041 15.352
nodefactor.race.0     NA  130.062       NA 13.955
nodefactor.race.1    375  390.487    0.041 26.116
nodematch.race       225  234.185    0.041 14.015
concurrent           100  106.132    0.061 11.435

Dissolution Diagnostics
-----------------------
               Target Sim Mean Pct Diff Sim SD
Edge Duration   25.00   23.738   -0.050 22.831
Pct Edges Diss   0.04    0.040   -0.002  0.012
```

There are two forms of dissolution diagnostics. The edge duration row shows the mean duration of partnerships across the simulations; it tends to be lower than the target unless the diagnostic simulation interval is very long since its average includes a burn-in period where all edges start at a duration of zero (illustrated below in the plot). The next row shows the percent of current edges dissolving at each time step, and is not subject to bias related to burn-in. The percentage of edges dissolution is the inverse of the expected duration: if the duration is 25 time steps, then we expect that 1/25 or 4% dissolve each day.

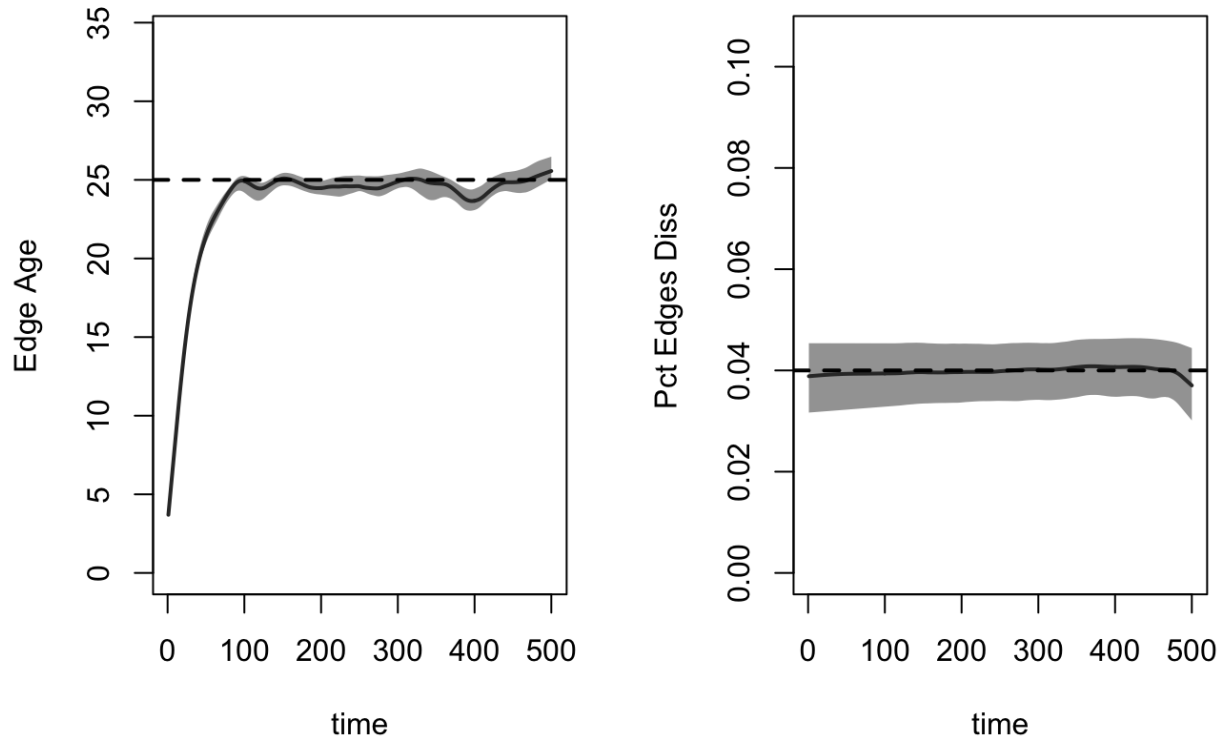
Plotting the diagnostics object will show the time series of the target statistics against any targets. The other options used here specify to smooth the mean lines, give them a thicker line width, and plot each statistic in a separate panel. The black dashed lines show the value of the target statistics for any terms in the model. Similar to the numeric summaries, the plots show a good fit over the time series.

```
par(mar = c(3,3,1,1), mgp = c(2,1,0))
plot(dx)
```



The dissolution model fit may also be assessed with plots by specifying either the `duration` or `dissolution` type, as defined above. The duration diagnostic is based on the average age of edges at each time step, up to that time step; there is a burn-in period because edges in existence at t_1 are right-censored. Both metrics show a good fit of the dissolution model to the target duration of 50 time steps.

```
par(mfrow = c(1, 2))
plot(dx, type = "duration")
plot(dx, type = "dissolution")
```



There are many options available better visualize the results. Please consult the help files for the `plot.netdx` for examples of function arguments.

If the model diagnostics had suggested poor fit, then additional diagnostics and fitting would be necessary. If using the approximation method, one should first start by running the cross-sectional diagnostics (see the `dynamic` argument in `netdx`). If the cross-sectional model fits well but the dynamic model does not, then a full STERGM estimation may be necessary. If the cross-sectional model does not fit well, different control parameters for the ERGM estimation may be necessary (see the help file for `netdx` for instructions).

Epidemic Simulation

EpiModel simulates disease epidemics over dynamic networks by integrating dynamic model simulations with the simulation of other epidemiological processes such as disease transmission and recovery. Like the network model simulations, these processes are also simulated stochastically so that the range of potential outcomes under the model specifications is estimated. The specification of epidemiological processes to model may be arbitrarily complex, but EpiModel includes a number of "built-in" model types within the software. Additional components must be programmed and plugged into the simulation API. Here, we will start simple with an SIS epidemic using this built-in functionality.

Our SIS model will rely on three parameters. The *act rate* is the number of sexual acts between that occur within a partnership each time unit. The overall frequency of acts per person per unit time is a function of the incidence rate of partnerships and this act parameter. The *infection probability* is the risk of transmission given contact with an infected person. The *recovery rate* for an SIS epidemic is the speed at which infected persons become susceptible again. For a bacterial STI like gonorrhea, this may be a function of biological attributes like gender or use of curative agents like antibiotics.

EpiModel uses three helper functions to input epidemic parameters, initial conditions, and other control settings for the epidemic model. First, we use the `param.net` function to input the per-act transmission probability in `inf.prob` and the number of acts per partnership per unit time in `act.rate`. The recovery rate implies that the average duration of disease is 50 time steps.

```
param <- param.net(inf.prob = 0.1, act.rate = 5, rec.rate = 0.02)
```

For initial conditions, one can use the `i.num` to set the initial number infected at the start, or as the example here shows, pass in a vector with a disease status for each of the nodes in the network. EpiModel stores the individual-level disease status as a vector of lower-case letters: “s” for susceptible, “i” for infected, and “r” for recovered. Here, we specify that Race 0 has a baseline prevalence of 10% (randomly assigned), whereas there are no nodes in Race 1 infected.

```
status.vector <- c(rbinom(500, 1, 0.1), rep(0, 500))
status.vector <- ifelse(status.vector == 1, "i", "s")
init <- init.net(status.vector = status.vector)
```

The control settings contain the structural features of the model. The `epi.by` argument allows us to pass in a nodal attribute that the epidemic output should be split by: here we request output by the subgroups of race.

```
control <- control.net(type = "SIS", nsteps = 500, nsims = 10, epi.by = "race")
```

Once the model has been parameterized, simulating the model is straightforward. One must pass the fitted network model object from `netest` along with the parameters, initial conditions, and control settings to the `netsim` function. With an independent model like this (i.e., there are no vital dynamic parameters), the full dynamic network time series is simulated at the start of each epidemic simulation, and then the epidemiological processes are simulated over that structure.

```
sim1 <- netsim(est1, param, init, control)
```

Summaries

Printing the model output lists the inputs and outputs of the model. The output includes the sizes of the compartments (`s.num` is the number susceptible and `i.num` is the number infected) and flows (`si.flows` is the number of infections and `is.flow` is the number of recoveries). Methods for extracting this output is discussed below.

```
sim1
```

```
EpiModel Simulation
=======================
Model class: netsim

Simulation Summary
-----------------------
Model type: SIS
No. simulations: 10
No. time steps: 500
No. NW modes: 1

Model Parameters
-----------------------
inf.prob = 0.1
act.rate = 5
rec.rate = 0.02

Model Output
-----------------------
Variables: s.num s.num.race0 s.num.race1 i.num i.num.race0
i.num.race1 num num.race0 num.race1 is.flow si.flow
Networks: sim1 ... sim10
Transmissions: sim1 ... sim10
```

Similar to ICMs, epidemic statistics may be obtained using the summary function. This summary shows the mean and standard deviation of simulations at time step 500.

```
summary(sim1, at = 500)
```

```


EpiModel Summary
=======================
Model class: netsim

Simulation Details
-----------------------
Model type: SIS
No. simulations: 10
No. time steps: 500
No. NW modes: 1

Model Statistics
------------------------------
Time: 500
------------------------------
            mean      sd    pct
Suscept.   501.6  22.780  0.502
Infect.    498.4  22.780  0.498
Total     1000.0   0.000  1.000
S -> I       9.2   3.225     NA
I -> S       9.8   4.211     NA
------------------------------
```

Extraction

Similar to the ICMs, means, standard deviations, and individual simulation run data is easily extracted using the `as.data.frame` function.

```
head(as.data.frame(sim1))
```

```
  time s.num s.num.race0 s.num.race1 i.num i.num.race0 i.num.race1  num
1    1 950.0       450.0       500.0  50.0        50.0         0.0 1000
2    2 945.9       447.0       498.9  54.1        53.0         1.1 1000
3    3 942.6       444.7       497.9  57.4        55.3         2.1 1000
4    4 941.7       444.5       497.2  58.3        55.5         2.8 1000
5    5 939.4       443.3       496.1  60.6        56.7         3.9 1000
6    6 938.3       443.2       495.1  61.7        56.8         4.9 1000
  num.race0 num.race1 is.flow si.flow
1       500       500     0.0     0.0
2       500       500     0.6     4.7
3       500       500     1.4     4.7
4       500       500     1.1     2.0
5       500       500     0.9     3.2
6       500       500     1.9     3.0
```

The default as before is to output the means, but here we show how to extract the model values from the second simulation.

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

```
  time s.num s.num.race0 s.num.race1 i.num i.num.race0 i.num.race1  num
1    1   950         450         500    50          50           0 1000
2    2   947         447         500    53          53           0 1000
3    3   940         442         498    60          58           2 1000
4    4   941         443         498    59          57           2 1000
5    5   936         442         494    64          58           6 1000
6    6   935         442         493    65          58           7 1000
  num.race0 num.race1 is.flow si.flow
1       500       500       0       0
2       500       500       0       3
3       500       500       1       8
4       500       500       2       1
5       500       500       1       6
6       500       500       3       4
```

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. This is accomplished with the `get_network` function, specifying the simulation number.

```
nw <- get_network(sim1, sim = 1)
nw
```

```
NetworkDynamic properties:
  distinct change times: 502
  maximal time range: -Inf until  Inf

 Dynamic (TEA) attributes:
  Vertex TEAs:    testatus.active

Includes optional net.obs.period attribute:
 Network observation period info:
  Number of observation spells: 2
  Maximal time range observed: 1 until 501
  Temporal mode: discrete
  Time unit: step
  Suggested time increment: 1

 Network attributes:
  vertices = 1000
  directed = FALSE
  hyper = FALSE
  loops = FALSE
  multiple = FALSE
  bipartite = FALSE
  net.obs.period: (not shown)
  vertex.pid = vertex.pid
  edge.pid = edge.pid
  total edges= 5407
    missing edges= 0
    non-missing edges= 5407

 Vertex attribute names:
    active race testatus.active vertex.names vertex.pid

 Edge attribute names not shown
```

A matrix is stored that records some key details about each transmission event that occurred. Shown below are the first 10 transmission events for simulation number 1. The `sus` column shows the unique ID of the previously susceptible, newly infected node in the event. The `inf` column shows the ID of the transmitting node. The other columns show the duration of the transmitting node's infection at the time of transmission, the per-act transmission probability, act rate during the transmission, and final per-partnership transmission rate (which is the per-act probability raised to the number of acts).

```
head(get_transmat(sim1, sim = 1), 10)
```

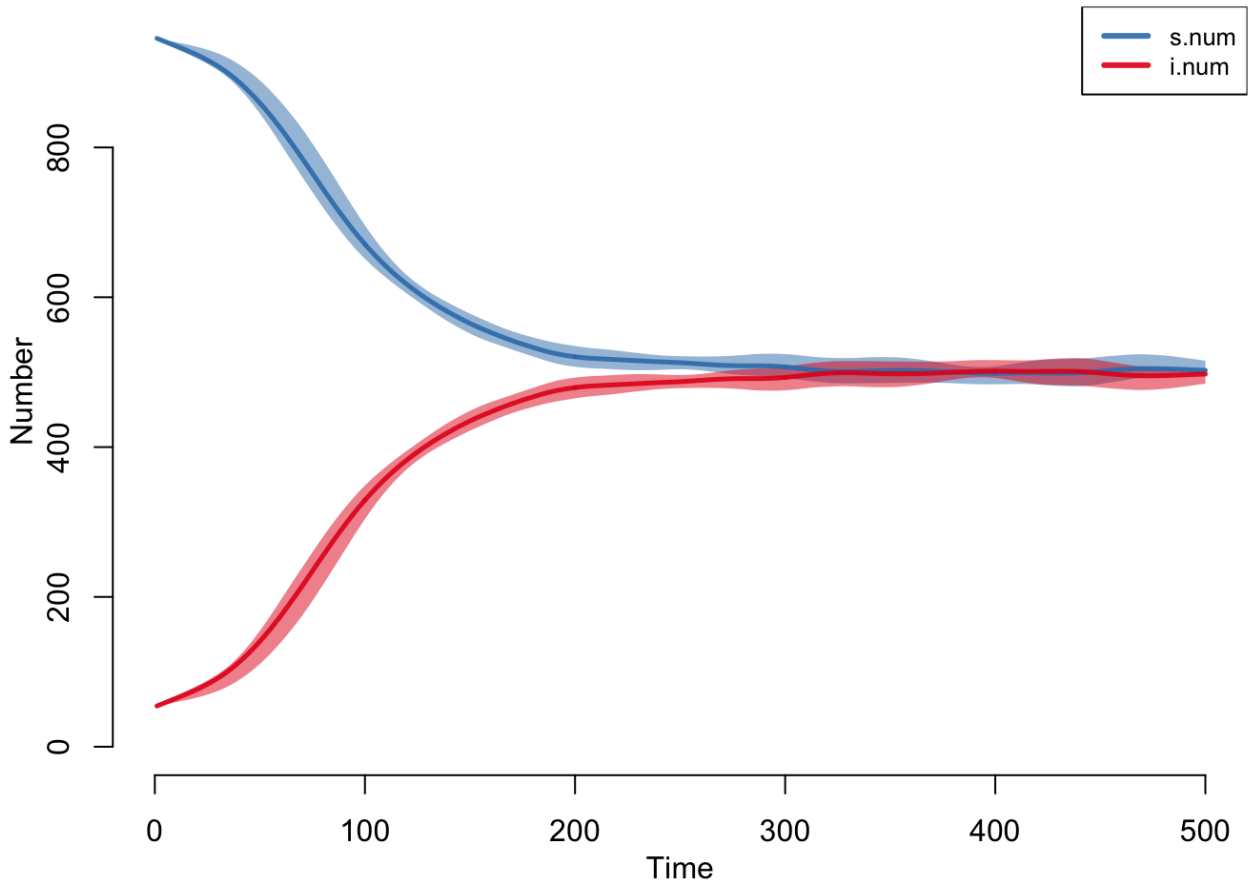
```
   at         sus         inf infDur transProb actRate finalProb
1   2 5ff4952cb01 5ff373df9b6     27       0.1       5   0.40951
2   2 5ff1dfbdb3c 5ff5278d95b     28       0.1       5   0.40951
3   2 5ff5831c407 5ff7f48d4b9      7       0.1       5   0.40951
4   2 5ff5f80368a 5ff3153849a     24       0.1       5   0.40951
5   2 5ff6fc977ef 5ff3153849a     24       0.1       5   0.40951
6   2  5ffcda6ab1 5ff4a407a09     40       0.1       5   0.40951
7   3 5ff5831c407 5ff7f48d4b9      8       0.1       5   0.40951
8   3 5ff6cf91e6c   5ff681a44     50       0.1       5   0.40951
9   4 5ff137eb1db 5ff4952cb01      2       0.1       5   0.40951
10  4 5ff12fe42b4 5ff110cca80     10       0.1       5   0.40951
```

Plotting

Plotting the output from the epidemic model will display the prevalence of the compartments in the model across simulations. The individual simulations are represented by the thin lines, the means across simulations at each time step are plotted with thicker lines, and the polygon band shows the inter-quartile range across simulations

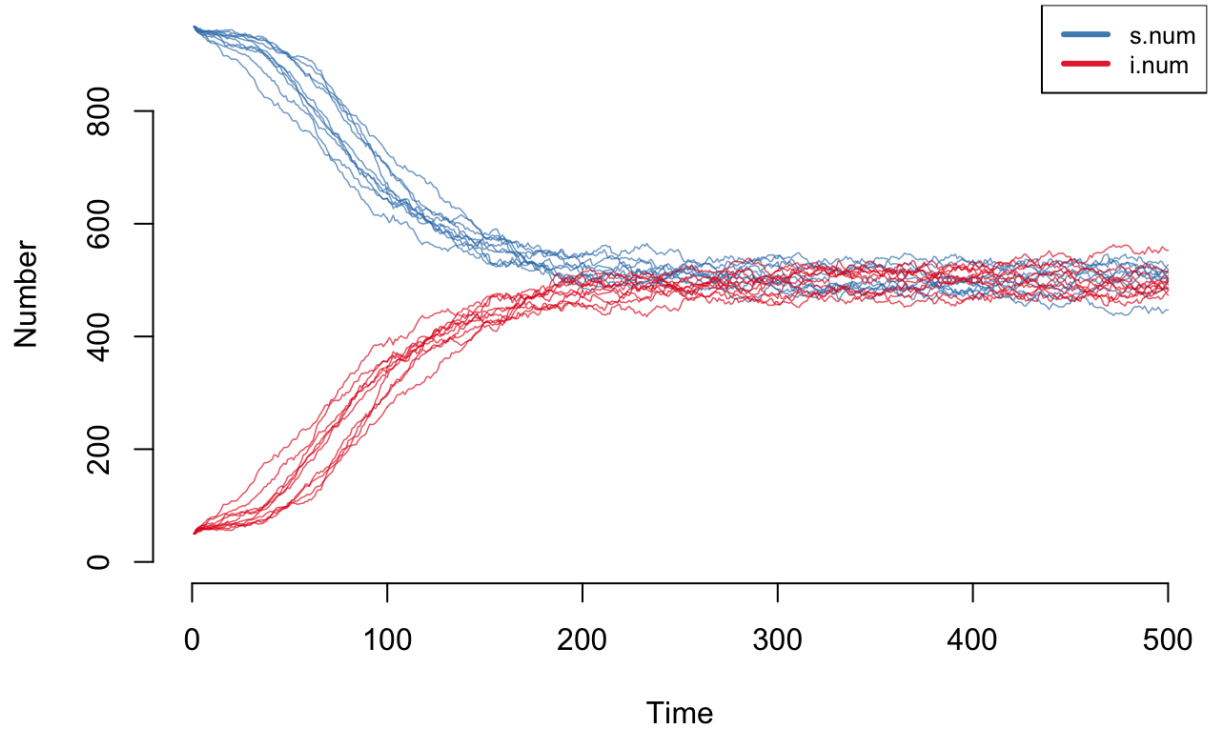
By default, the plot shows disease trajectories (compartment sizes) over time. The plotting function uses the same defaults for the stochastic model results as in ICMs, and therefore, the same arguments options apply. In a one-mode network model, the default is to plot the mean and inter-quartile range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

```
par(mfrow = c(1,1), mar = c(3,3,1,1), mgp = c(2,1,0))
plot(sim1)
```



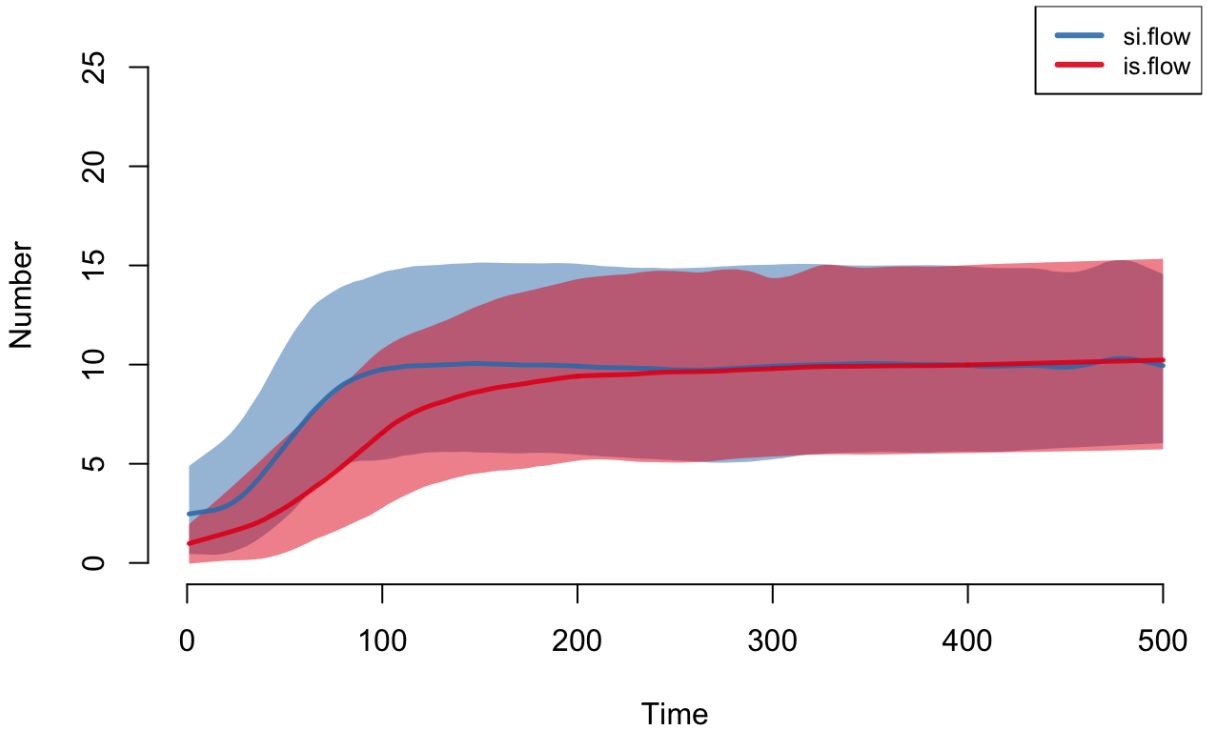
Different elements of the plot can be toggled on and off. Here is the same plot, but only showing the individual simulation lines.

```
plot(sim1, mean.line = FALSE, qnts = FALSE, sim.lines = TRUE)
```



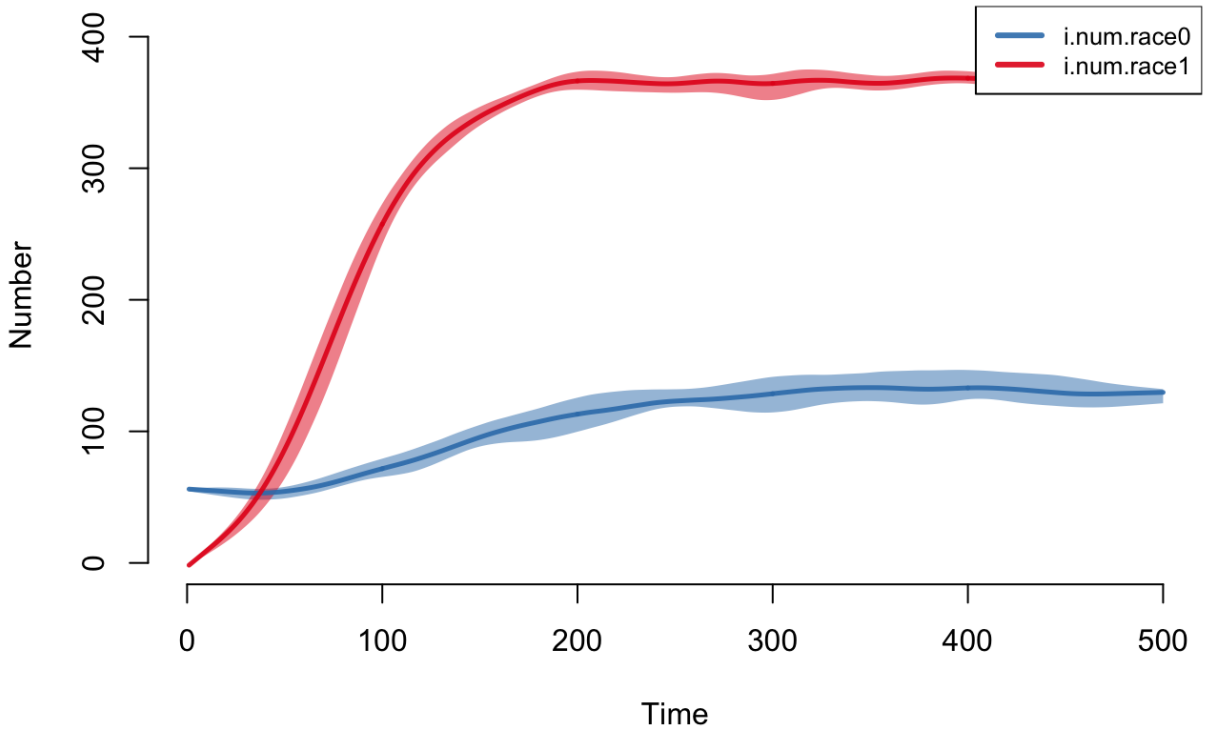
To plot the disease incidence and recoveries, it is necessary to specify these outcome variables using the `y` argument. The `si.flow` is the number of transitions from susceptible to infected each time step, and the `is.flow` is the number of transitions back from infected to susceptible. The `qnts` argument may range from 0 to 1 to plot an inter-quantile range of data; a quantile of 1 means to plot the entire range of the data.

```
plot(sim1, y = c("si.flow", "is.flow"), qnts = 1, legend = TRUE)
```



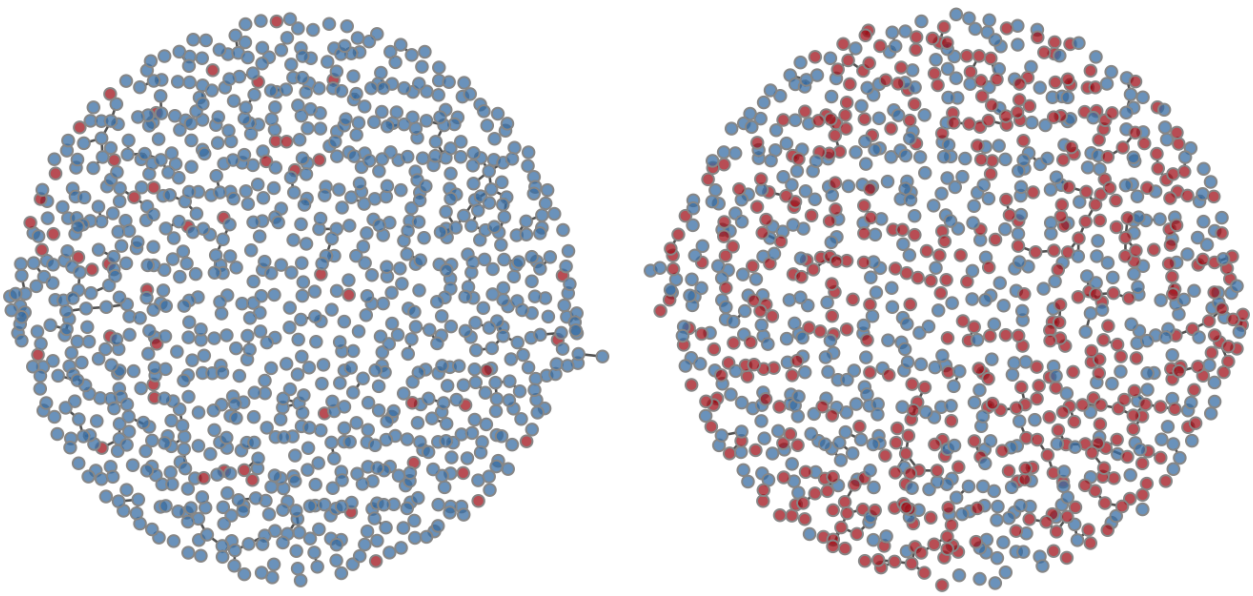
To compare the results by race, the race-specific prevalence numbers are input in `y`. Here, we see that the initial prevalence in the Race 0 group is 10%, as specified in `init.net`. However, the epidemic reaches a much higher prevalence for Race 1 because of their higher mean degree, along with the high levels of within-group mixing.

```
plot(sim1, y = c("i.num.race0", "i.num.race1"), legend = 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 epidemic time series (`type="epi"`), 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 500. The `col.status` argument handles the color coding for easy plotting of the infected (in red) versus susceptible (in blue).

```
par(mfrow = c(1,2), mar = c(0,0,1,0))
plot(sim1, type = "network", at = 1, col.status = TRUE,
     main = "Prevalence at t1")
plot(sim1, type = "network", at = 500, col.status = TRUE,
     main = "Prevalence at t500")
```



Dependent Bipartite SI Model

In this second network model example, we introduce dependence between the disease simulation and the network structure. There are two main forms of dependence implemented in the built-in stochastic network models in EpiModel:

1. **Vital dynamics:** these demographic transition processes for births and deaths are integrated into the network model simulations similarly to the stochastic transition processes in ICMs, in which the number of new births and deaths at each time step is determined as random draws from a binomial distributions. Vital dynamics create dependency because the deaths of nodes removes their associated edges, and births create new isolate nodes (no partnerships) onto which edges may be formed.
2. **Serosorting:** this is a process by which the probability of partnership formation is influenced by disease status. One could model the process by which disease-negative persons preferentially partner with other disease-negative persons; this requires a continual resimulation of the network as the status attribute changes over time.

In this example, we demonstrate the vital dynamics dependency. Additionally in this example, we simulate an epidemic model here on 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.

Network Estimation and Diagnostics

As with the independent network model, the first step is to specify a network structure, including features like size and nodal attributes. Here, we construct an empty network of 1000 nodes, with 500 in each mode. One might conceive of the first mode as females and the second mode as males.

```
num.m1 <- 500
num.m2 <- 500
nw <- network.initialize(num.m1 + num.m2, bipartite = num.m1, directed = FALSE)
```

Edge Balancing

For our target statistics, we will use sex-specific degree distributions. Similar to the contact balancing requirements in acts in the Basic DCMs Tutorial (BasicDCMs.html), we must balance the number of partnerships implied by a degree distribution in one mode to that of another. This is particularly important in cases where one sex reports higher numbers of partners.

In EpiModel, we check that the implied number of edges match given different degree distributions specified in vectors of fractional values using the `check_bip_degdist` 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 values for males is represented by a second vector. Men report more concurrency than women, but also more men have no partners.

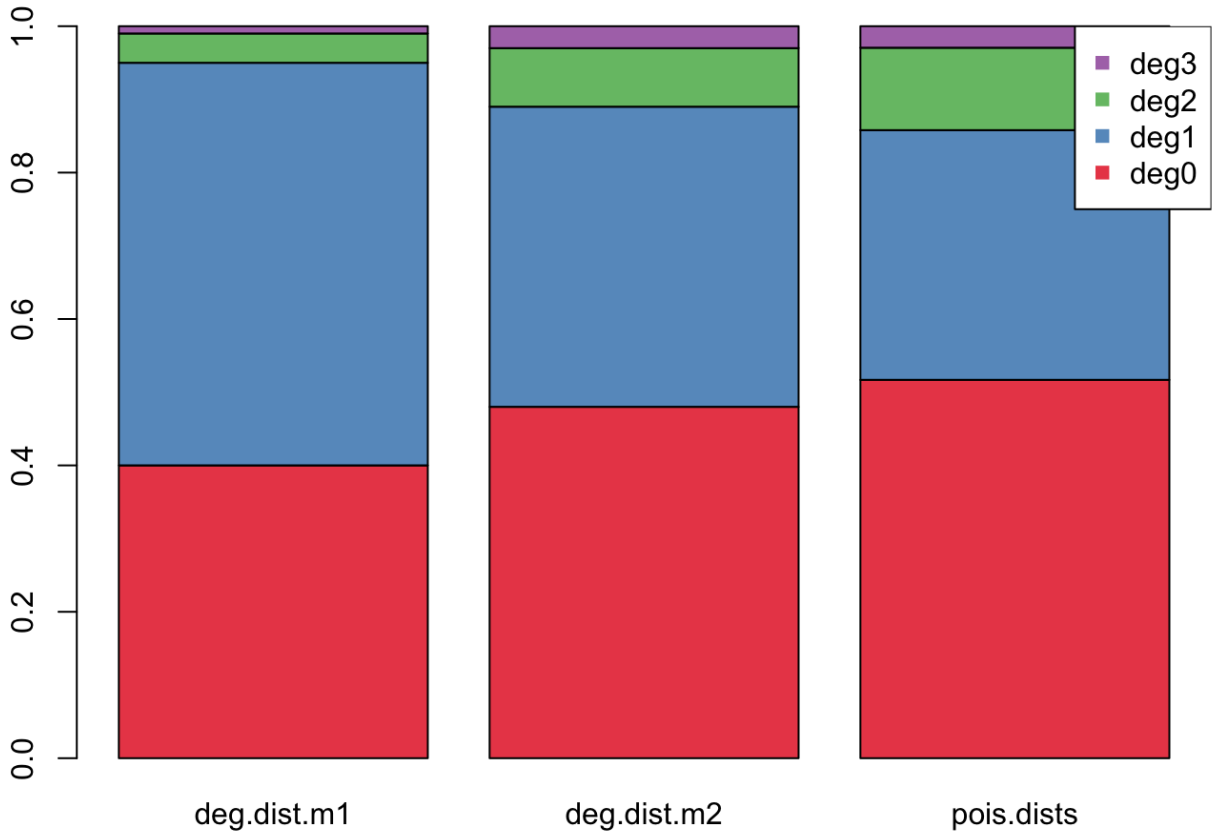
```
deg.dist.m1 <- c(0.40, 0.55, 0.04, 0.01)
deg.dist.m2 <- c(0.48, 0.41, 0.08, 0.03)
```

In our model, we will specify a mean degree of 0.66 that applies to both men and women. How do our degree distributions compare to the distribution expected under a Poisson distribution with a rate equal to this the mean degree? The `dpois` function returns the probability mass for degree 0 through 2 and `ppois` sums the cumulative mass for degree 3+.

```
pois.dists <- c(dpois(0:2, lambda = 0.66), ppois(2, lambda = 0.66, lower = FALS
E))
```

The bar plot compares the two observed degree distribution against the expected distribution, each adding to 100%. The degree distribution for men more closely the expectation, but both empirical distributions have more mass on single partnerships and less mass on concurrent partnerships than expected.

```
par(mar = c(3, 3, 2, 1), mfrow = c(1, 1))
cols <- transco(RColorBrewer::brewer.pal(4, "Set1"), 0.8)
barplot(cbind(deg.dist.m1, deg.dist.m2, pois.dists),
        beside = FALSE, ylim = c(0, 1), col = cols)
legend("topright", legend = paste0("deg", 3:0),
       pch = 15, col = rev(cols), bg = "white")
```



Given these fractional degree distributions and the mode sizes set when initializing the network, the `check_bip_degdist` function checks for balance. The table shows the fractional distribution and number of nodes with each degree, as well as the total number of edges in that mode. The number of edges for each mode should approximately match for balancing to occur.

```
check_bip_degdist(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      200      0.48      240
Deg1       0.55      275      0.41      205
Deg2       0.04       20      0.08       40
Deg3       0.01        5      0.03       15
Edges      1.00      330      1.00      330
============================================
** Edges balanced **
```

Model Fit

The network formation model will target the overall mean degree in the network, as well as elements of the sex-specific distribution since the Poisson expectations are not close to observed. 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. Notice that these can be extracted directly from the table above.

```
formation <- ~edges + b1degree(0:1) + b2degree(0:1)
target.stats <- c(330, 200, 275, 240, 205)
```

The dissolution model is a homogeneous exponential decay with mean partnership duration of 25 time units. Since there will be births and deaths in the model, 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 edge dissolution on top of the estimated, endogenous average duration. Since the coefficient for the dissolution model actually represents the probability of edge persistence, the value of the adjusted coefficient is higher than the crude coefficient to increase the duration of edges by a small factor to account for mortality.

```
coef.diss <- dissolution_coefs(dissolution = ~offset(edges), duration = 25,
                               d.rate = 0.005)
coef.diss
```

```
Dissolution Coefficients
========================
Dissolution Model: ~offset(edges)
Target Statistics: 25
Crude Coefficient: 3.178054
Mortality/Exit Rate: 0.005
Adjusted Coefficient: 3.464903
```

The network estimation process uses the `netest` function in the same way as our independent model above. The edges dissolution approximation method is again used to fit a static ERGM and manually adjust the coefficients to replicate a full STERGM.

```
est2 <- netest(nw, formation, target.stats, coef.diss)
```

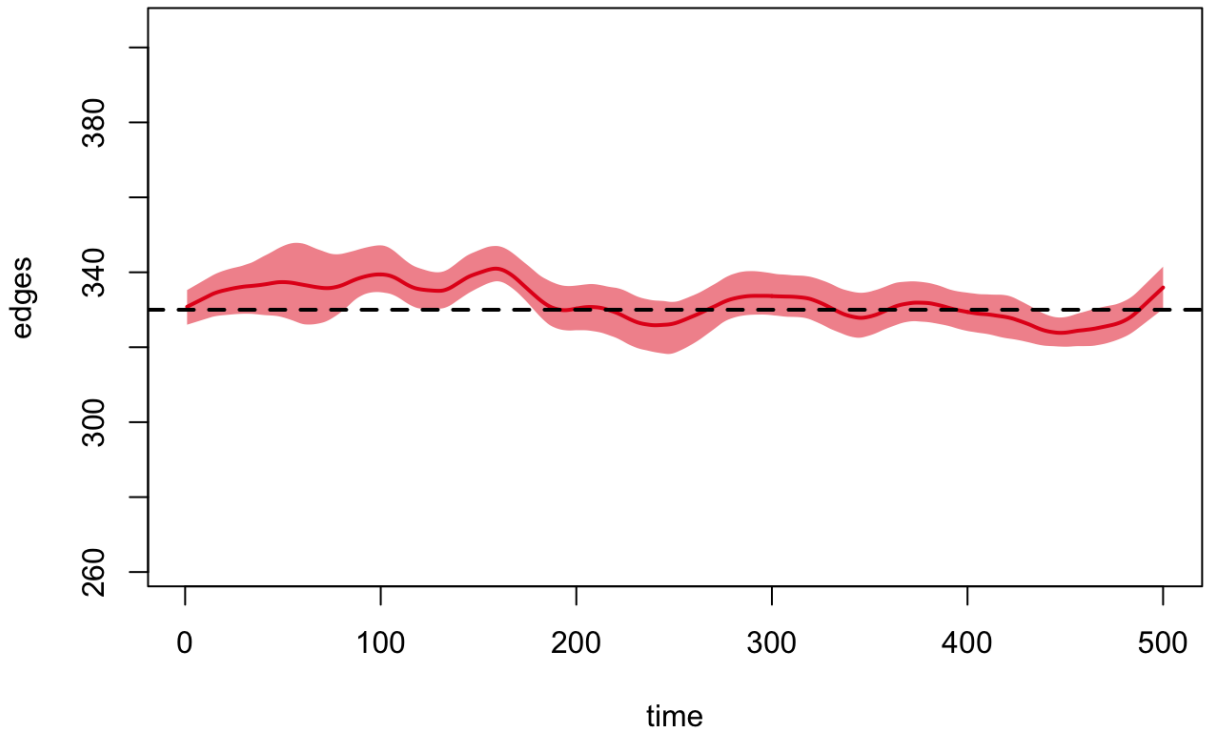
Model Diagnostics

Model diagnostics certainly should be run before moving to epidemic simulation, but one important caveat is that the simulated models here do not include the vital dynamics processes that could influencing the structure of the network. So it will be necessary to also run post-simulation diagnostics. For these diagnostics, we simulate the dynamic network 5 times over 500 time steps. The main printing and plotting methods detailed for the independent model may be used. Since we have not specified any network statistics to monitor in `netdx` , the function uses the formation formula terms. Both the formation and dissolution summary statistics are right on target.

```
dx <- netdx(est2, nsims = 5, nsteps = 500)
dx
```

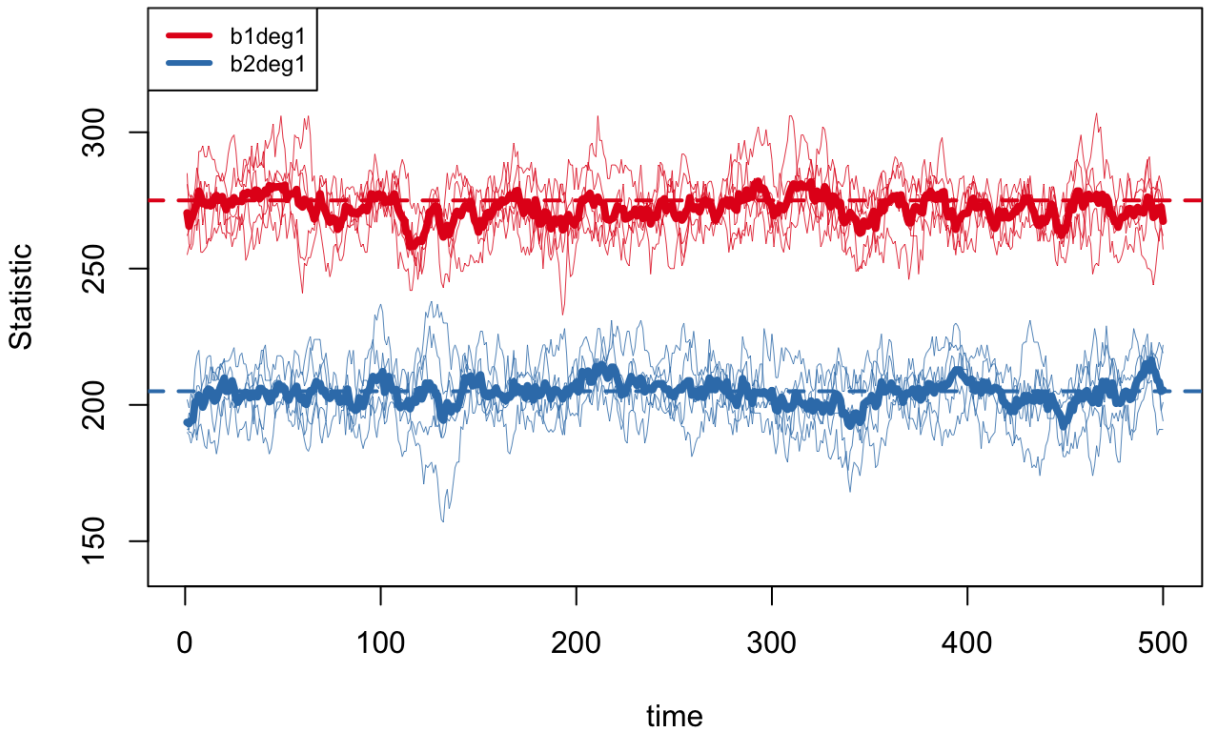
The `stats` argument of the `plot.netdx` function allows for sub-setting the statistics for better visualization.

```
plot(dx, stats = "edges")
```



Similar to epidemic plots, different elements of the plots may be toggled to highlight different components of the diagnostics.

```
plot(dx, stats = c("b1deg1", "b2deg1"), sim.lines = TRUE, sim.lwd = 0.4,
     qnts = FALSE, mean.lwd = 4, mean.smooth = FALSE)
```



Epidemic Simulation

In a dependent model, the dynamic networks are simulated at each time step along within the main simulation function, `netsim` . The main model parameters are as follows. Different transmission probabilities by mode are allowed to incorporate a higher susceptibility for disease for one mode compared to the other. The birth rates are parameterized with the `b.rate.m2` parameter set as `NA` to signify that the number of births into the second mode should be based on the size of the first mode (e.g., females). For bipartite simulations, it is necessary to specify mode-specific parameters. There is nothing in the parameterization that explicitly tells `netsim` to run a dependent model: the dependency is implied by the input of the vital dynamics parameters above.

```
param <- param.net(inf.prob = 0.3, inf.prob.m2 = 0.1,
                   b.rate = 0.005, b.rate.m2 = NA,
                   ds.rate = 0.005, ds.rate.m2 = 0.005,
                   di.rate = 0.005, di.rate.m2 = 0.005)
```

The initial number infected will be 50 for both the first and second modes. For the control settings, this simulation uses the `delete.nodes` parameter to remove inactive (e.g., dead) nodes from the network object at each time step. Computationally, this results in a faster simulation since the `networkDynamic` object with the complete edge history of all nodes ever in the network is discarded. The drawback is that this prevents analysis of edge or nodal history, the latter of which is used in network plots with node coloring by disease status. For simplicity, we will use the `nwstats.formula` to monitor only the number of edges in the simulations and the mean degree (a function of edges and the population size).

```
init <- init.net(i.num = 50, i.num.m2 = 50)
control <- control.net(type = "SI", nsims = 5, nsteps = 500,
                       nwstats.formula = ~edges + meandeg, delete.nodes = TRUE)
sim2 <- netsim(est2, param, init, control)
```

Printing the object shows that we have an EpiModel object that is an SI network model with 5 simulations over 500 time steps with a bipartite 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.

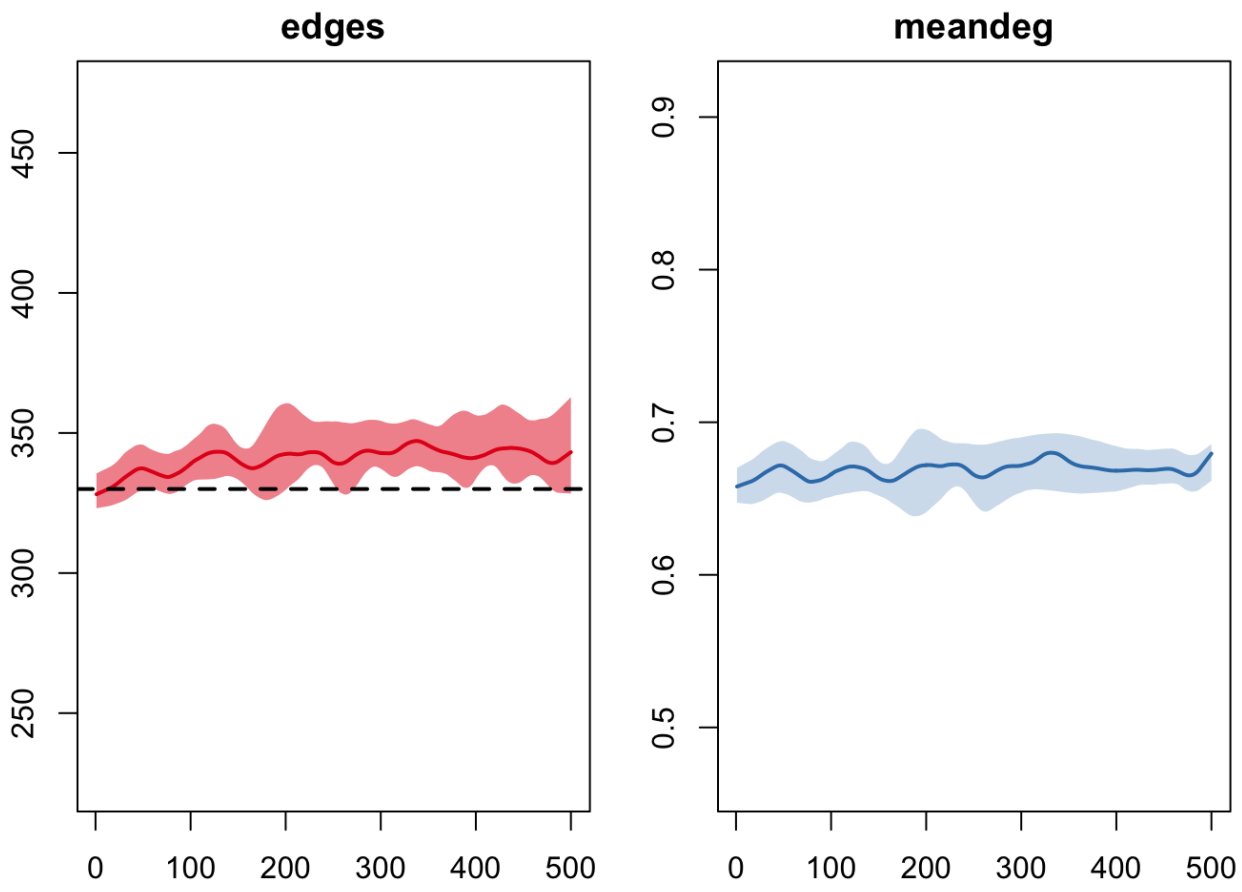
```
sim2
```

```
EpiModel Simulation
=======================
Model class: netsim

Simulation Summary
------------------------
Model type: SI
No. simulations: 5
No. time steps: 500
No. NW modes: 2

Model Parameters
------------------------
inf.prob = 0.3
b.rate = 0.005
ds.rate = 0.005
di.rate = 0.005
inf.prob.m2 = 0.1
b.rate.m2 = NA
ds.rate.m2 = 0.005
di.rate.m2 = 0.005
act.rate = 1

Model Output
------------------------
Variables: s.num i.num num s.num.m2 i.num.m2 num.m2
ds.flow di.flow ds.flow.m2 di.flow.m2 b.flow b.flow.m2
si.flow si.flow.m2
Networks: sim1 ... sim5
Transmissions: sim1 ... sim5
```

Diagnostics

In dependent simulations, it is wise to examine the network structure of the simulations. Summary network statistics are saved at each time step. One important check is to see that the vital dynamics processes did not lead to any systematic biases in the expected edges or mean degree. Here we plot values of those statistics across the five simulations against their expected targets. Plot arguments for these post-simulation diagnostics are found in the `plot.netsim` help page, but match all the arguments for the `plot.netdx` function.

```
plot(sim2, type = "formation", plots.joined = FALSE)
```

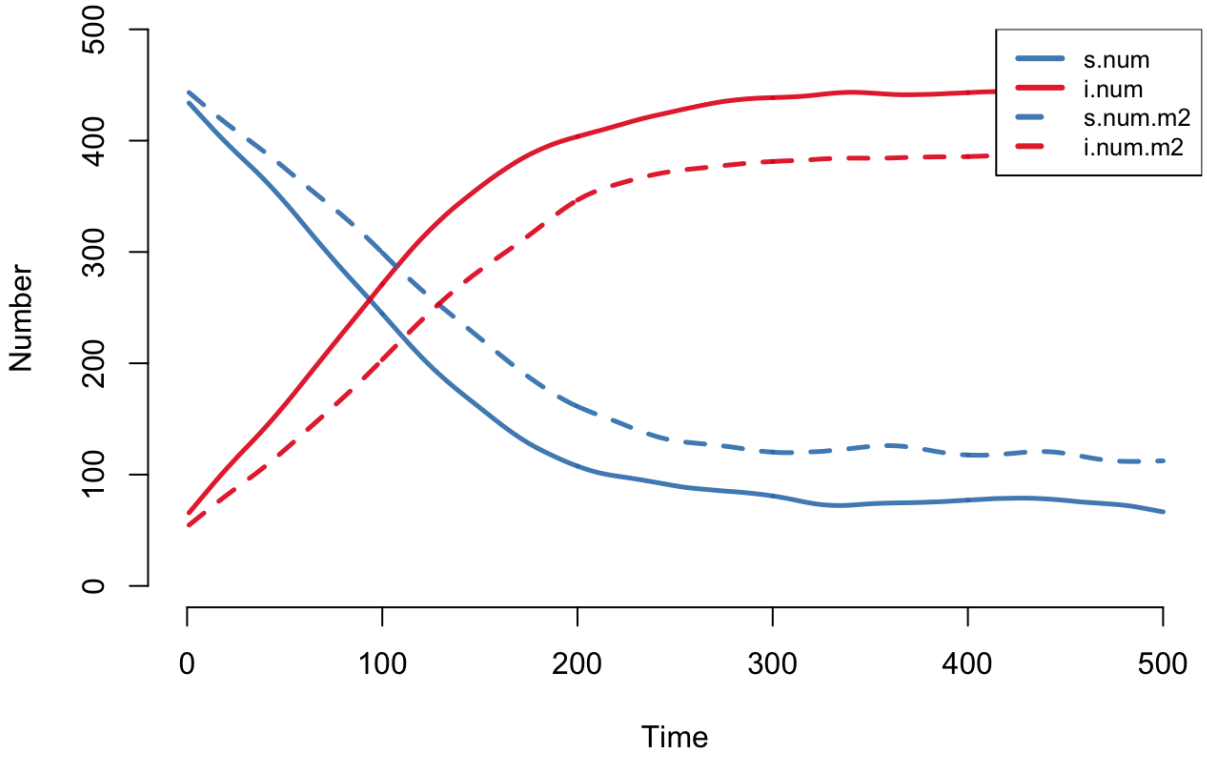


In this case, the stochastically vary around the targets in both cases, so there is no need for additional model diagnostics.

Plotting

Plotting dependent disease simulations is the same as for independent simulations. Since this is a bipartite network, the default plot shows the means of the compartment sizes over time for each of the modes. This plot shows the absolute numbers from the compartments, 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)
```



The plot shows the disease prevalence is higher in mode 2 than in mode 1, both as a result of the higher levels of concurrency in mode 2 (a transmission risk for mode 1) and the higher disease susceptibility of mode 1 persons.

Next Steps

This tutorial has provided the basics to get started with exploring stochastic network models with EpiModel. If you want to learn how to build your network models that remove some of the assumptions and specifications of these basic network models, the New Network Models with EpiModel (NewNet.html) tutorial is appropriate. Also, please consult the materials for our week-long Network Modeling for Epidemics (http://statnet.github.io/nme/) course.

Last updated: 2017-06-01 with EpiModel v1.5.0

Back to Top (BasicNet.html) | Back to epimodel.org (http://www.epimodel.org/)