Package ‘tergmLite’

February 29, 2020

Version 2.1.7
Date 2020-02-04
Title Fast Simulation of Simple Temporal Exponential Random Graph Models
Description Provides functions for the computationally efficient simulation of dynamic networks estimated with the statistical framework of temporal exponential random graph models, implemented in the 'tergm' package.
Depends R (>= 3.2.0)
License GPL-3
Imports ergm (>= 3.10.4), statnet.common (>= 4.3.0), tergm (>= 3.6.1), network (>= 1.16.0), Rcpp
Suggests testthat, EpiModel (>= 1.7.5)
LinkingTo Rcpp, ergm
LazyData true
RoxygenNote 7.0.2
Encoding UTF-8
NeedsCompilation yes
Author Samuel M. Jenness [aut, cre], Chad Klumb [aut]
Maintainer Samuel M. Jenness <samuel.m.jenness@emory.edu>
Repository CRAN
Date/Publication 2020-02-29 12:20:05 UTC

R topics documented:

  tergmLite-package ........................................... 2
  add_vertices .................................................. 3
  delete_vertices .............................................. 4
  ergm_prep ..................................................... 5
  init_tergmLite ............................................... 6
Description

Package: tergmLite
Type: Package
Version: 2.1.7
Date: 2020-02-04
License: GPL-3
LazyLoad: yes

Details

The statistical framework of temporal exponential random graph models (TERGMs) provides a rigorous, flexible approach to estimating generative models for dynamic networks and simulating from them for the purposes of modeling infectious disease transmission dynamics. TERGMs are used within the EpiModel software package to do just that. While estimation of these models is relatively fast, the resimulation of them using the tools of the tergm package is computationally burdensome, requiring hours to days to iteratively resimulate networks with co-evolving demographic and epidemiological dynamics. The primary reason for the computational burden is the use of the network class of object (designed within the package of the same name); these objects have tremendous flexibility in the types of networks they represent but at the expense of object size. Continually reading and writing larger-than-necessary data objects has the effect of slowing the iterative dynamic simulations.

The tergmLite package reduces that computational burden by representing networks less flexibly, but much more efficiently. For epidemic models, the only types of networks that we typically estimate and simulate from are undirected, binary edge networks with no missing data (as it is simulated). Furthermore, the network history (edges or node attributes) does not need to be stored for research-level applications in which summary epidemiological statistics (e.g., disease prevalence, incidence, and variations on those) at the population-level are the standard output metrics for epidemic models. Therefore, the network may be stored as a cross-sectional edgelist, which is a two-column matrix of current edges between one node (in column one) and another node (in column two). Attributes of the edges that are called within ERGMs may be stored separately in vector
format, as they are in EpiModel. With this approach, the simulation time is sped up by a factor of 25-50 fold, depending on the specific research application.

---

### Description

This function performs a simple operation of updating the edgelist attribute \( n \) that tracks the total network size implicit in an edgelist representation of the network.

### Usage

```r
add_vertices(el, nv)
```

### Arguments

- `el` A two-column matrix of current edges (edgelist) with an attribute variable \( n \) containing the total current network size.
- `nv` A integer equal to the number of nodes to add to the network size at the given time step.

### Details

This function is used in EpiModel modules to add vertices (nodes) to the edgelist object to account for entries into the population (e.g., births and in-migration).

### Value

Returns the updated the attribute containing the population size on the edgelist, \( el \), based on the number of new vertices specified to be added in \( nv \).

### Examples

```r
library("EpiModel")
nw <- network.initialize(n = 100, directed = FALSE)
formation <- ~edges
target.stats <- 50
coeff.diss <- dissolution_coefs(dissolution = ~offset(edges), duration = 20)
x <- netest(nw, formation, target.stats, coeff.diss, verbose = FALSE)

param <- param.net(inf.prob = 0.3)
init <- init.net(i.num = 10)
control <- control.net(type = "SI", nsteps = 100, nsims = 5, depend = TRUE)

# Full network structure after initialization
dat <- initialize.net(x, param, init, control)

# networkLite representation used by tergmLite
```
delete_vertices

Fast Version of network::delete.vertices for Edgelist-formated Network

Description

Given a current two-column matrix of edges and a vector of IDs to delete from the matrix, this function first removes any rows of the edgelist in which the IDs are present and then permutes downward the index of IDs on the edgelist that were numerically larger than the IDs deleted.

Usage

delete_vertices(el, vid)

Arguments

el A two-column matrix of current edges (edgelist) with an attribute variable n containing the total current network size.

vid A vector of IDs to delete from the edgelist.

Details

This function is used in EpiModel modules to remove vertices (nodes) from the edgelist object to account for exits from the population (e.g., deaths and out-migration)

Value

Returns a updated edgelist object, el, with the edges of deleted vertices removed from the edgelist and the ID numbers of the remaining edges permuted downward.

Examples

library("EpiModel")
nw <- network.initialize(n = 100, directed = FALSE)
formation <- ~edges
target.stats <- 50
coef.diss <- dissolution_coef(dissolution = ~offset(edges), duration = 20)
x <- netest(nw, formation, target.stats, coef.diss, verbose = FALSE)

param <- param.net(inf.prob = 0.3)
init <- init.net(i.num = 10)
control <- control.net(type = "SI", nsteps = 100, nsims = 5, depend = TRUE)

# Set seed for reproducibility
set.seed(123456)

# Full network structure after initialization
dat <- initialize.net(x, param, init, control)

# networkLite representation used by tergmLite
dat <- init_tergmLite(dat)

# Current edges include {1, 87}, {2, 33}, {4, 19}, and {5, 99}
head(dat$el[[1]], 4)

# Remove nodes 1 and 2
nodes.to.delete <- 1:2
dat$el[[1]] <- delete_vertices(dat$el[[1]], nodes.to.delete)

# Old nodes 1 and 2 removed, and old nodes 4 and 5 become nodes 2 and 3
# with edges 19 and 99 also shifted down to 17 and 97
head(dat$el[[1]], 4)

---

**ergm_prep**

Prepare Network and ERGM Objects for tergmLite

**Description**

Converts network object, formation and dissolution formulas, formation and dissolution coefficients, and control settings to a thin list format for ERGM resimulation.

**Usage**

```r
ergm_prep(nw,
    formation,
    coef,
    constraints,
    control = ergm::control.simulate.ergm()
)
```

**Arguments**

- `nw`: An object of class network.
- `formation`: Right-hand sided formation formula.
init_terminLite

coeff Vector of coefficients associated with the formation formula
constraints Constraints for the formation model (only bd) constraints currently supported.
control Control settings passed to ergm::control.simulate.ergm.

Details

This is an internal function used within init_terminLite. It is not exported from the package but it is documented here to demonstrate the internal inputs for init_terminLite.

Value

Returns a list class object with two elements:

- model.form: Model coefficients and data elements.
- MHproposal: Model constraint data elements.

init_terminLite  Initializes EpiModel netsim Object for tergmLite Simulation

Description

Initializes EpiModel netsim Object for tergmLite Simulation

Usage

init_terminLite(dat)

Arguments

dat A list object containing a networkDynamic object and other initialization information passed from netsim.

Details

This function is typically used within the initialization modules of EpiModel to establish the necessary networkLite infrastructure needed for tergmLite network resimulation. Specifically, this function converts (and then removes) the network class objects into an edgelist only format and prepares the ERGM structural information for simulation. The example below demonstrates the specific information returned.

Value

Returns the list object dat and adds two elements to the objects: el is an edgelist representation of the network; and p is a list object that contains all the relevant structural information for ERGM/TERGM simulation. The function also removes the network class object on the dat object, stored under nw because it is no longer needed.
Examples

library("EpiModel")
nw <- network.initialize(n = 100, directed = FALSE)
formation <- ~edges
target.stats <- 50
coeff.diss <- dissolution_coefs(dissolution = ~offset(edges), duration = 20)
x <- netest(nw, formation, target.stats, coeff.diss, verbose = FALSE)

param <- param.net(inf.prob = 0.3)
init <- init.net(i.num = 10)
control <- control.net(type = "SI", nsteps = 100, nsims = 5, depend = TRUE)

# Full network structure after initialization
dat <- initialize.net(x, param, init, control)
str(dat, max.level = 1)

# networkLite representation used by tergmLite
dat <- init_tergmLite(dat)
str(dat, max.level = 1)

# Elements removed are nw (network class object)
# Elements added are el (edgelist representation of network)...
dat$el

# ... and p (contains all relevant ERGM structural information for simulation)
str(dat$p, max.level = 3)

networkLite

networkLite Constructor Utility

Description

Constructor function for a networkLite object.

Usage

networkLite(el, attr)

Arguments

el an edgelist-formatted network representation, including network attributes.
attr a list of named vertex attributes for the network represented by el.

Details

This function takes an edge list el with network attributes attached, and a list of vertex attributes attr, and returns a networkLite object, which is a list with named fields el, attr, and gal, with each of the first two corresponding to the argument of the same name, and gal being the list of
network attributes (copied from attributes(el)) for compatibility with some network accessors. Missing attributes directed, bipartite, loops, hyper, and multiple are defaulted to FALSE. The network size attribute n must not be missing. This new data structure is then used within the `updateModelTermInputs` function for updating the structural information on the network used for ERGM simulation.

**Value**

A networkLite object with edge list el, vertex attributes attr, and network attributes gal.

**Examples**

```r
library("EpiModel")
nw <- network.initialize(n = 100, directed = FALSE)
formation <- ~edges
target.stats <- 50
coeff.diss <- dissolution_coefs(dissolution = ~offset(edges), duration = 20)
x <- netest(nw, formation, target.stats, coeff.diss, verbose = FALSE)

param <- param.net(inf.prob = 0.3)
init <- init.net(i.num = 10)
control <- control.net(type = "SI", nsteps = 100, nsims = 5, depend = TRUE)

# Full network structure after initialization
dat <- initialize.net(x, param, init, control)

# networkLite representation used by tergmLite
dat <- init_tergmLite(dat)

# Conversion to networkLite class format
mwl <- networkLite(dat$el[[1]], dat$attr)
mwl
```

---

**networkLitemethods**  
*networkLite Methods*

**Description**

S3 methods for networkLite class, for generics defined in network package.

**Usage**

```r
## S3 method for class 'networkLite'
as.network(nw, ...)

## S3 method for class 'networkLite'
get.vertex.attribute(x, attrname, ...)
```
## simulate_ergm

### S3 method for class 'networkLite'

```r
list.vertex.attributes(x)
```

**Arguments**

- `nw` a networkLite object.
- `...` any additional arguments.
- `x` a networkLite object.
- `attrname` the name of a vertex attribute in `x`.

**Details**

Allows use of networkLite objects in `ergm_model`.

---

### simulate_ergm

**Fast Version of `ergm::simulate.ergm` for Edgelist-formatted Network**

**Description**

Resimulates a networkLite object given thin network data structure, edgelist, and ERGM model coefficients.

**Usage**

```r
simulate_ergm(p, el, coef)
```

**Arguments**

- `p` A list of network-related nodal covariates and related terms that is produced with `ergm_prep`.
- `el` A two-column matrix of current edges (edgelist) with an attribute variable `n` containing the total current network size.
- `coef` Vector of coefficients associated with the formation formula.

**Details**

This function is used within the network resimulation module in EpiModel to update cross-sectional ERGMs based on the model coefficients and current network structure. If network structure (e.g., number of nodes) or nodal attributes has changed since the last simulation, this network resimulation should be run only after `updateModelTermInputs`.

**Value**

Returns an updated network edgelist object, typically stored on the master `dat` list object, based on the model simulation.
Examples

library("EpiModel")

# Set seed for reproducibility
set.seed(12345)

nw <- network.initialize(n = 100, directed = FALSE)
nw <- set.vertex.attribute(nw, "group", rep(0:1, each = 50))
formation <- ~edges + nodefactor("group")
target.stats <- c(15, 10)
coef.diss <- dissolution_coefs(dissolution = ~offset(edges), duration = 1)
x <- netest(nw, formation, target.stats, coef.diss, verbose = FALSE)

param <- param.net(inf.prob = 0.3)
init <- init.net(i.num = 10)
control <- control.net(type = "SI", nsteps = 100, nsims = 5, depend = TRUE)

# Full network structure after initialization
dat <- initialize.net(x, param, init, control)
str(dat, max.level = 1)

# networkLite representation used by tergmLite
dat <- init_tergmLite(dat)

# Current network structure
dat$el[[1]]

# New network structure (all edges are new)
dat$el[[1]] <- simulate_ergm(p = dat$p[[1]],
el = dat$el[[1]],
coef = dat$nwparam[[1]]$coef.form)
dat$el[[1]]

---

**simulate_network**  
Fast Version of tergm::simulate.network for networkLite Object

**Description**

Resimulates a networkLite object given thin network data structure, edgelist, and STERGM model coefficients.

**Usage**

```
simulate_network(p, el, coef.form, coef.diss, save.changes = FALSE)
```
simulate_network

Arguments

- **p**: A list of network-related nodal covariates and related terms that is produced with `stergm_prep`.
- **el**: A two-column matrix of current edges (edgelist) with an attribute variable `n` containing the total current network size.
- **coef.form**: Vector of coefficients associated with the formation formula.
- **coef.diss**: Vector of coefficients associated with the dissolution formula.
- **save.changes**: Logical; if TRUE, saves a matrix of changed edges as an attribute of the output edgelist matrix.

Details

This function is used within the network resimulation module in EpiModel to update temporal ERGMs based on the model coefficients and current network structure. If network structure (e.g., number of nodes) or nodal attributes has changed since the last simulation, this network resimulation should be run only after `updateModelTermInputs`.

Value

Returns an updated network edgelist object, typically stored on the master dat list object, based on the model simulation. If `save.changes` is TRUE, also returns a list of new edges and dissolved edges with the resimulation.

Examples

```r
library("EpiModel")

# Set seed for reproducibility
set.seed(12345)

nw <- network.initialize(n = 100, directed = FALSE)
nw <- set.vertex.attribute(nw, "group", rep(0:1, each = 50))
formation <- ~edges + nodefactor("group")
target.stats <- c(15, 10)
coef.diss <- dissolution_coefs(dissolution = ~offset(edges), duration = 20)
x <- netest(nw, formation, target.stats, coef.diss, verbose = FALSE)

param <- param.net(inf.prob = 0.3)
init <- init.net(i.num = 10)
control <- control.net(type = "SI", nsteps = 100, nsims = 5, depend = TRUE)

# Full network structure after initialization
dat <- initialize.net(x, param, init, control)
str(dat, max.level = 1)

# networkLite representation used by tergmLite
dat <- init_tergmLite(dat)

# Current network structure
```
dat$el[[1]]

# New network structure
dat$el[[1]] <- simulate_network(p = dat$p[[1]],
   el = dat$el[[1]],
   coef.form = dat$nwparam[[1]]$coef.form,
   coef.diss = dat$nwparam[[1]]$coef.diss$coef.adj,
   save.changes = TRUE)

dat$el[[1]]

# Specific changes listed under changes list
# (new edges: to = 1; dissolved edges: to = 0):
attributes(dat$el[[1]])$changes

---

**stergm_prep**

Prepare Network and STERGM Objects for tergmLite

**Description**

Converts network object, formation and dissolution formulas, formation and dissolution coefficients, and control settings to a thin list format for STERGM resimulation.

**Usage**

```r
stergm_prep(
   nw,
   formation,
   dissolution,
   coef.form,
   coef.diss,
   constraints,
   control = control.simulate.network()
)
```

**Arguments**

- **nw**
  - An object of class network.
- **formation**
  - Right-hand sided formation formula.
- **dissolution**
  - Right-hand sided dissolution formula.
- **coef.form**
  - Vector of coefficients associated with the formation formula.
- **coef.diss**
  - Vector of coefficients associated with the dissolution formula.
- **constraints**
  - Constraints for the formation model (only bd) constraints currently supported.
- **control**
  - Control settings passed to `tergm::control.simulate.network`. 
Details

This is an internal function used within init_tergmLite. It is not exported from the package but it is documented here to demonstrate the internal inputs for init_tergmLite.

Value

Returns a list class object with four elements:

- `model.form`: Formation model coefficients and data elements.
- `model.diss`: Dissolution model coefficients and data elements.
- `MHproposal.form`: Formation model constraint data elements.
- `MHproposal.diss`: Dissolution model constraint data elements.

Description

Function to appropriately update model inputs based on ERGM model terms when using network-Lite representation.

Usage

`updateModelTermInputs(dat, network = 1)`

Arguments

- `dat`: EpiModel dat object tracking simulation state
- `network`: Numeric number of network location for multi-network simulations.

Details

Calls `ergm_model` to update model inputs based on potential exogenous changes to network structure (e.g., number of nodes) or nodal attributes used within ERGM model (see example below). This function is typically used within EpiModel module for network resimulation, immediately prior to calling `simulate_network` or `simulate_ergm`.

Implemented terms are:

- edges
- nodematch
- nodefactor
- concurrent (including heterogeneous by attribute)
- degree (including heterogeneous by attribute)
- degrange
updateModelTermInputs

- `absdiff`
- `absdiffby` (in the EpiModel package)
- `nodecov`
- `nodemix`
- `absdiffnodemix` (in the EpiModel package)
- `triangle`
- `gwesp(fixed=TRUE)`

All other ERGM terms will return errors.

Value

Returns an updated `dat` object with the network list structure inputs used by `simulate_network` or `simulate_ergm` with changes to network size or nodal covariates.

Examples

```r
library("EpiModel")

# Set seed for reproducibility
set.seed(12345)

nw <- network.initialize(n = 100, directed = FALSE)
nw <- set.vertex.attribute(nw, "group", rep(0:1, each = 50))
formation <- ~edges + nodefactor("group")
target.stats <- c(15, 10)
coef.diss <- dissolution_coefs(dissolution = ~offset(edges), duration = 1)
x <- netest(nw, formation, target.stats, coef.diss, verbose = FALSE)

param <- param.net(inf.prob = 0.3)
init <- init.net(i.num = 10)
control <- control.net(type = "SI", nsteps = 100, nsims = 5, depend = TRUE)

# Full network structure after initialization
dat <- initialize.net(x, param, init, control)
str(dat, max.level = 1)

# networkLite representation used by tergmLite
dat <- init_tergmLite(dat)

# Examine the network list structure for nodefactor term
dat$p[[1]]$model$form$terms[[2]]

# inputs vector corresponds to group attribute stored here
dat$attr$group

# As example of what could happen in EpiModel: randomly reshuffle group
# attribute values of 100 nodes
dat$attr$group <- sample(dat$attr$group)
dat$attr$group
```
# Update network list structure
dat <- updateModelTermInputs(dat)

# Check that network list structure for nodefactor term has been updated
dat$p[[1]]$model.form$terms[[2]]
Index

Topic package
  tergmLite-package, 2

add_vertices, 3
as.network.networkLite
  (networkLitemethods), 8

delete_vertices, 4

ergm_prep, 5, 9

get.vertex.attribute.networkLite
  (networkLitemethods), 8

init_tergmLite, 6, 6, 13

list.vertex.attributes.networkLite
  (networkLitemethods), 8

networkLite, 7
networkLitemethods, 8

simulate_ergm, 9, 13, 14
simulate_network, 10, 13, 14

stergm_prep, 11, 12

tergmLite (tergmLite-package), 2

updateModelTermInputs, 8, 9, 11, 13