Package: networkLite (via r-universe)

November 8, 2026

Version 1.1.0

Date 2025-01-08

Title An Simplified Implementation of the 'network' Package Functionality

Description An implementation of some of the core 'network' package functionality based on a simplified data structure that is faster in many research applications. This package is designed for back-end use in the 'statnet' family of packages, including 'EpiModel'. Support is provided for binary and weighted, directed and undirected, bipartite and unipartite networks; no current support for multigraphs, hypergraphs, or loops.

License GPL-3

URL https://github.com/EpiModel/networkLite/

BugReports https://github.com/EpiModel/networkLite/issues

Depends R (>= 3.5), network (>= 1.19.0)

Imports statnet.common (>= 4.11.0), tibble, dplyr

Suggests testthat

RoxygenNote 7.3.2

Encoding UTF-8

Roxygen list(markdown = TRUE)

Repository https://epimodel.r-universe.dev

Date/Publication 2025-03-07 05:01:45 UTC

RemoteUrl https://github.com/epimodel/networklite

RemoteRef HEAD

RemoteSha 29b70b0e500a8fbcaa1ae869bfeefe94cd9576ff

Contents

networkLite-package . . . 2
$.networkLite . . . 3
+.networkLite . . . 4
add.edges.networkLite . . . 4
add.vertices.networkLite . . . 5
as.edgelist.networkLite . . . 6
as.networkLite . . . 7
atomize . . . 8
delete.edges.networkLite . . . 9
delete.vertices.networkLite . . . 9
get.inducedSubgraph.networkLite . . . 10
get.vertex.attribute.networkLite . . . 10
is.na.networkLite . . . 12
is.networkLite . . . 13
mixingmatrix.networkLite . . . 13
network.edgecount.networkLite . . . 14
networkLite . . . 14
permute.vertexIDs.networkLite . . . 16
print.networkLite . . . 17
to_network_networkLite . . . 17
valid.eids.networkLite . . . 18

Index **19**

`networkLite-package` *networkLite Package*

Description

Package:	networkLite
Type:	Package
Version:	1.1.0
Date:	2025-01-08
License:	GPL-3
LazyLoad:	yes

Details

The networkLite package provides an alternative implementation of some of the functionality in the network package, based on a different data structure that is faster for certain applications. It is intended for use as a backend data structure in EpiModel and statnet packages, and its implementation is subject to change.

The networkLite data structure is a named list with three components:

- el, a tibble edgelist, including edge attributes
- attr, a tibble of vertex attributes
- gal, a named list of network attributes

These components should not be referred to directly by the user in their own code. Instead, the various access, coercion, etc. methods provided by this package should be used. See networkLite for information on how to construct a networkLite.

Certain names in el, attr, and gal have special significance. These are:

- For el: ".tail" and ".head", of class integer, which are the tails and heads of edges, and must be preserved as atomic integer vectors with no NAs; "na", which is a logical attribute indicating if the edge is missing or not, and should take TRUE/FALSE values only (behavior for other values is undefined, and NAs are not allowed); "na" may be structured as either an atomic logical vector or a list.
- For attr: "na", which is a logical attribute indicating if the vertex is missing or not, and "vertex.names", which provides names for the vertices in the network; the attribute "na" should take values TRUE or FALSE only (behavior for other values is undefined).
- For gal: "n" (the network size), "directed" (a logical indicating if the network is directed), "bipartite" (either FALSE to indicate the network is not bipartite, or the size of the first bipartition if the network is bipartite), "hyper" (a logical indicating if the network is a hypergraph), "multiple" (a logical indicating if the network is a multigraph), and "loops" (a logical indicating if the network is allowed to have loops).

For networkLites, the three network attributes "hyper", "multiple", and "loops" must all be FALSE. Even with these restrictions, networkLites do not provide all the functionality that networks do, but attempt to offer what is necessary for backend use in ergm, tergm, and EpiModel.

`$.networkLite` *Last-resort utilities*

Description

Since the purpose of networkLite is to be a drop-in replacement for network, these utilities emulate the internal structure of network. They are likely to perform poorly and should be avoided.

Usage

```
## S3 method for class 'networkLite'
x$name

## S3 method for class 'networkLite'
x[...]
```

Arguments

x	a networkLite object
name	passed to [[, but some special cases ("oel" nd "iel" are handled)

Note

Please do not rely on these unless you absolutely have to; since they slow down access to networkLite objects, they may be removed if they ever stop being necessary.

+.networkLite *Add and Subtract* networkLite*s*

Description

Add and Subtract networkLites

Usage

```
## S3 method for class 'networkLite'
e1 + e2

## S3 method for class 'networkLite'
e1 - e2
```

Arguments

e1, e2	networkLite objects

Value

For the + method, a networkLite whose edges are those in either e1 or e2. For the - method, a networkLite whose edges are those in e1 and not in e2.

---

add.edges.networkLite *Methods to Add or Modify Edges in a* networkLite.

Description

Methods to Add or Modify Edges in a networkLite.

Usage

```
## S3 method for class 'networkLite'
add.edges(x, tail, head, names.eval = NULL, vals.eval = NULL, ...)

## S3 replacement method for class 'networkLite'
x[i, j, names.eval = NULL, add.edges = FALSE] <- value
```

Arguments

x	A networkLite.
tail	Vector of tails of edges to add to the networkLite.
head	Vector of heads of edges to add to the networkLite.
names.eval	Names of edge attributes, or NULL to indicate that attributes are not being specified. For add.edges, this argument should be structured as a list of length equal to length(tail), each element of which is a character vector of attribute names for the corresponding edge. For the replacement method [<-.networkLite, this should argument should be a single attribute name, which is applied to all edges.
vals.eval	Value(s) of edge attributes, or NULL to indicate that attributes are not being specified. This argument should be structured as a list of length equal to length(tail), each element of which is a list of attribute values, in the same order as the corresponding attribute names in names.eval.
...	additional arguments
i, j	Nodal indices (must be missing for networkLite method).
add.edges	logical; should edges being assigned to be added if they are not already present?
value	Edge values to assign (coerced to a matrix).

Value

A networkLite object with edges added (if calling add.edges) or set to specified values (if calling [<-.networkLite).

---

add.vertices.networkLite

Add Vertices to a networkLite.

Description

Add Vertices to a networkLite.

Usage

```
## S3 method for class 'networkLite'
add.vertices(x, nv, vattr = NULL, last.mode = TRUE, ...)
```

Arguments

x	A networkLite object.
nv	Number of vertices to add to the networkLite.
vattr	A list (of length nv) of named lists of vertex attribute values for added vertices, or NULL to indicate vertex attribute values are not being passed.
last.mode	logical; if x is bipartite, should the new vertices be added to the second mode?
...	additional arguments

Value

A networkLite object with vertices added.

---

as.edgelist.networkLite *Convert a* networkLite *to a Matrix or* tibble.

Description

Convert a networkLite to a Matrix or tibble.

Usage

```
## S3 method for class 'networkLite'
as.edgelist(
  x,
  attrname = NULL,
  output = c("matrix", "tibble"),
  na.rm = TRUE,
  ...
)

## S3 method for class 'networkLite'
as_tibble(
  x,
  attrnames = (match.arg(unit) == "vertices"),
  na.rm = TRUE,
  ...,
  unit = c("edges", "vertices")
)

## S3 method for class 'networkLite'
as.matrix(
  x,
  matrix.type = c("adjacency", "incidence", "edgelist"),
  attrname = NULL,
  ...
)
```

Arguments

x	A networkLite.
attrname	Name of an edge attribute in x.
output	Type of edgelist to output.
na.rm	should missing edges be dropped from edgelist?
...	additional arguments

attrnames	Vector specifying edge attributes to include in the tibble; may be logical, integer, or character vector, the former two being used to select attribute names from list.edge.attributes(x), and the latter being used as the attribute names themselves
unit	whether to return attributes for edges or for vertices
matrix.type	type of matrix to return from as.matrix.networkLite

Value

A matrix or tibble (possibly of class edgelist) constructed from the networkLite.

---

as.networkLite *Convert to* networkLite *Representation.*

Description

Convert to networkLite Representation.

Usage

```
as.networkLite(x, ...)

## Default S3 method:
as.networkLite(x, ...)

## S3 method for class 'network'
as.networkLite(x, ..., atomize = TRUE)

## S3 method for class 'networkLite'
as.networkLite(x, ...)
```

Arguments

x	A network or networkLite object.
...	additional arguments
atomize	Logical; should we call atomize on the networkLite before returning it?

Details

as.networkLite.network converts a network object to a networkLite object. as.networkLite.networkLite returns the networkLite object unchanged.

The last resort is to call as.network(x, ...) and then convert the result to networkLite.

Currently the network attributes hyper, multiple, and loops must be FALSE for networkLites; attempting to convert a network to a networkLite when this is not the case will result in an error.

The ... are passed to atomize and can be used to set the upcast argument controlling attribute conversion.

Value

A corresponding networkLite object.

See Also

to_network_networkLite

---

atomize *Convert Lists to Atomic Vectors Where Possible*

Description

Convert Lists to Atomic Vectors Where Possible

Usage

```
atomize(x, ...)

## S3 method for class 'networkLite'
atomize(x, ..., upcast = FALSE)

## S3 method for class 'tbl_df'
atomize(x, ..., upcast = FALSE)
```

Arguments

x	A networkLite or tibble object.
...	additional arguments
upcast	logical; are we allowed to upcast atomic types when converting lists to atomic vectors?

Details

The tibble method examines each column of the tibble and replaces the column with the result of calling unlist on the column if all of the following are true: the column is.list of length greater than zero, each element of which is.atomic of length one, and either upcast is TRUE or there is only one unique class among all elements of the column.

The networkLite method applies the tibble method to the edgelist and vertex attribute tibbles in the networkLite.

Value

The networkLite or tibble with list columns replaced by atomic vector columns where possible.

delete.edges.networkLite

Delete edges from a networkLite.

Description

Delete edges from a networkLite.

Usage

```
## S3 method for class 'networkLite'
delete.edges(x, eid, ...)
```

Arguments

x	A networkLite object.
eid	Edge ids (between 1 and network.edgecount(x, na.omit = FALSE)) to delete in x. Note that the edge id of an edge in x is simply its row index in x$el.
...	additional arguments.

Value

A networkLite object with the specified edges deleted.

delete.vertices.networkLite

Delete vertices from a networkLite.

Description

Delete vertices from a networkLite.

Usage

```
## S3 method for class 'networkLite'
delete.vertices(x, vid, ...)
```

Arguments

x	A networkLite object.
vid	Vertex ids (between 1 and network.size(x)) to delete from x. Note that edges involving deleted vertices will also be deleted.
...	additional arguments.

Value

A networkLite object with the specified vertices deleted.

`get.inducedSubgraph.networkLite` *Return an induced subgraph*

Description

Return an induced subgraph

Usage

```
## S3 method for class 'networkLite'
get.inducedSubgraph(x, v, alters = NULL, ...)
```

Arguments

`x`, `v`, `alters`, `...` see `network::get.inducedSubgraph()`

`get.vertex.attribute.networkLite` `networkLite` *Attribute Methods*

Description

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

Usage

```
## S3 method for class 'networkLite'
get.vertex.attribute(x, attrname, ..., null.na = TRUE, unlist = TRUE)

## S3 method for class 'networkLite'
set.vertex.attribute(
  x,
  attrname,
  value,
  v = seq_len(network.size(x)),
  ...,
  upcast = FALSE
)

## S3 method for class 'networkLite'
list.vertex.attributes(x, ...)

## S3 method for class 'networkLite'
get.network.attribute(x, attrname, ..., unlist = FALSE)
```

```
## S3 method for class 'networkLite'
set.network.attribute(x, attrname, value, ...)

## S3 method for class 'networkLite'
list.network.attributes(x, ...)

## S3 method for class 'networkLite'
get.edge.attribute(x, attrname, ..., null.na = FALSE, unlist = TRUE)

## S3 method for class 'networkLite'
get.edge.value(x, attrname, ..., null.na = FALSE, unlist = TRUE)

## S3 method for class 'networkLite'
set.edge.attribute(
  x,
  attrname,
  value,
  e = seq_len(network.edgecount(x, na.omit = FALSE)),
  ...,
  upcast = FALSE
)

## S3 method for class 'networkLite'
set.edge.value(
  x,
  attrname,
  value,
  e = seq_len(network.edgecount(x, na.omit = FALSE)),
  ...,
  upcast = FALSE
)

## S3 method for class 'networkLite'
list.edge.attributes(x, ...)

## S3 method for class 'networkLite'
delete.vertex.attribute(x, attrname, ...)

## S3 method for class 'networkLite'
delete.edge.attribute(x, attrname, ...)

## S3 method for class 'networkLite'
delete.network.attribute(x, attrname, ...)
```

Arguments

x	A networkLite object.

attrname	The name of an attribute in x; must be a length one character vector.
...	additional arguments
null.na	Logical. If TRUE, replace NULL attribute values with NA in get.vertex.attribute and get.edge.attribute. Applied before the unlist argument. Note that the behavior of null.na in network is somewhat different.
unlist	Logical. In get.vertex.attribute and get.edge.attribute, if unlist is TRUE, we call unlist on the attribute value before returning it, and if unlist is FALSE, we call as.list on the attribute value before returning it. In get.network.attribute, if unlist is TRUE, we call unlist on the attribute value before returning it, and if unlist is FALSE, we return the attribute value without any modification.
value	The attribute value to set in vertex, edge, and network attribute setters. For set.vertex.attribute and set.edge.attribute, value should be either an atomic vector or a list, of length equal to that of v or e. For set.edge.value, it should be an n by n matrix where n is the network size of x.
v	Indices at which to set vertex attribute values.
upcast	Logical. Are we allowed to upcast atomic types when setting vertex or edge attribute values on the networkLite? Setting upcast = FALSE prevents upcasting, while setting upcast = TRUE allows but does not guarantee upcasting.
e	Indices at which to set edge attribute values.

Details

Allows basic attribute manipulation for networkLites. Note that an edge or vertex attribute not present in the networkLite is treated as a list of NULLs of length equal to the number of edges or vertices (respectively) before applying the null.na and unlist arguments.

Value

Behavior and return values are analogous to those of the corresponding network methods, with network data structured in the networkLite format.

---

is.na.networkLite *Extract* networkLite *with Missing Edges Only*

Description

Extract networkLite with Missing Edges Only

Usage

```
## S3 method for class 'networkLite'
is.na(x)
```

Arguments

x	A networkLite.

Value

A networkLite with the same network size, directedness, and bipartiteness as x, whose edges are precisely those edges in x that are missing in x. Edges in the returned networkLite are marked as not missing.

---

is.networkLite *Test whether the object is a* networkLite

Description

Test whether the object is a networkLite

Usage

```
is.networkLite(x)
```

Arguments

x	an object

---

mixingmatrix.networkLite *Extract Mixing Matrix from* networkLite

---

Description

Extract Mixing Matrix from networkLite

Usage

```
## S3 method for class 'networkLite'
mixingmatrix(object, attr, ...)
```

Arguments

object	A networkLite object.
attr	The name of a vertex attribute in object.
...	additional arguments

Value

The mixing matrix (of class table) for object and attr.

network.edgecount.networkLite *Count Edges in a* networkLite

Description

Count Edges in a networkLite

Usage

```
## S3 method for class 'networkLite'
network.edgecount(x, na.omit = TRUE, ...)

## S3 method for class 'networkLite'
network.naedgecount(x, ...)
```

Arguments

x	A networkLite object.
na.omit	logical; omit missing edges from edge count?
...	additional arguments

Details

The network.edgecount method provides a count of the number of edges in the networkLite, including missing edges if na.omit = FALSE and omitting them if na.omit = TRUE. The network.naedgecount method provides a count of the number of missing edges in the networkLite.

Value

The number of edges (of the appropriate type) in x.

networkLite *networkLite Constructor Utilities*

Description

Constructor methods for networkLite objects.

Usage

```
networkLite(x, ...)

## S3 method for class 'edgelist'
networkLite(
  x,
 attr = list(vertex.names = seq_len(net_attr[["n"]]), na = logical(net_attr[["n"]])),
 net_attr = attributes(x)[setdiff(names(attributes(x)), c("class", "dim", "dimnames",
    "vnames", "row.names", "names", "mnext"))],
  ...,
  atomize = FALSE
)

## S3 method for class 'matrix'
networkLite(
  x,
 attr = list(vertex.names = seq_len(net_attr[["n"]]), na = logical(net_attr[["n"]])),
 net_attr = attributes(x)[setdiff(names(attributes(x)), c("class", "dim", "dimnames",
    "vnames", "row.names", "names", "mnext"))],
  ...,
  atomize = FALSE
)

## S3 method for class 'numeric'
networkLite(x, directed = FALSE, bipartite = FALSE, ...)

networkLite_initialize(x, directed = FALSE, bipartite = FALSE, ...)
```

Arguments

x	Either an edgelist class network representation, or a number specifying the network size. The edgelist may be either a tibble or a matrix. If a tibble is passed, it should have integer columns named ".tail" and ".head" for the tails and heads of edges, and may include edge attributes as additional columns. If a matrix is passed, it should have two columns, the first being the tails of edges and the second being the heads of edges; edge attributes are not supported for matrix arguments. Edges should be sorted, first on tails then on heads. See network::as.edgelist for information on producing such edgelist objects from network objects.
...	additional arguments
attr	A named list of vertex attributes, coerced to tibble. Each element of attr should be an atomic vector or list of length equal to the number of nodes in the network.
net_attr	A named list of network attributes. Must include the network size attribute named "n". Defaults to a subset of the attr-style attributes of x for backwards compatibility; it is recommended that new code specify net_attr explicitly rather than relying on this default.

`atomize` Logical; should we call `atomize` on the `networkLite` before returning it? Note that unlike `as.networkLite`, the default value here is `FALSE`.

`directed, bipartite` Common network attributes that may be set via arguments to the `networkLite.numeric` method.

Details

Currently there are several distinct `networkLite` constructor methods available.

The `edgelist` method takes an `edgelist` class object `x`, a named list of vertex attributes `attr`, and a named list of network attributes `net_attr`, and returns a `networkLite` object, which is a named list with fields `el`, `attr`, and `gal`, corresponding to the arguments `x`, `attr`, and `net_attr`. Missing network attributes `directed` and `bipartite` are defaulted to `FALSE`; the network size attribute `n` must not be missing.

The `numeric` method takes a number `x` as well as the network attributes `directed` and `bipartite` (defaulting to `FALSE`), and returns an empty `networkLite` with these network attributes and number of nodes `x`.

The constructor `networkLite_initialize` is also available for creating an empty `networkLite`, and its `x` argument should be a number indicating the size of the `networkLite` to create.

Within `EpiModel`, the `networkLite` data structure is used in the calls to `ergm` and `tergm` `simulate` and `summary` functions.

Value

A `networkLite` object constructed according to the inputs.

Examples

```
edgelist <- cbind(c(1, 2, 3), c(2, 4, 7))
attr(edgelist, "n") <- 10 # network size
vertex_attributes <- list(a = 1:10, b = runif(10))
nwL <- networkLite(edgelist, vertex_attributes)
nwL
```

---

permute.vertexIDs.networkLite

Permute vertices

Description

Permute vertices

Usage

```
## S3 method for class 'networkLite'
permute.vertexIDs(x, vids, ...)
```

Arguments

x, vids, ...	see network::permute.vertexIDs()

---

print.networkLite *Print Basic Summary of a* networkLite

Description

Print Basic Summary of a networkLite

Usage

```
## S3 method for class 'networkLite'
print(x, ...)
```

Arguments

x	A networkLite object.
...	additional arguments

Details

This method prints a basic summary of a networkLite object, including network size, edge count, and attribute names.

Value

The networkLite is returned invisibly.

---

to_network_networkLite *Convert a* networkLite *object to a* network *object*

Description

Convert a networkLite object to a network object

Usage

```
to_network_networkLite(x, ...)

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

Arguments

x	A networkLite object.
...	additional arguments.

Details

The to_network_networkLite function takes a networkLite and returns a corresponding network.

The as.network.networkLite method returns the networkLite unchanged, for compatibility with ergm.

Value

For to_network_networkLite, a network object corresponding to x is returned. For as.network.networkLite, the networkLite x is returned unchanged.

See Also

as.networkLite

---

valid.eids.networkLite *valid.eids*

Description

valid.eids

Usage

```
## S3 method for class 'networkLite'
valid.eids(x, ...)
```

Arguments

x	A networkLite object.
...	additional arguments.

Details

Returns seq_len(network.edgecount(x, na.omit = FALSE)), to support the edge attribute assignment operator \%e\%<-. Note that the edge id of an edge in x is simply its row index within x$el.

Value

The sequence seq_len(network.edgecount(x, na.omit = FALSE)).

Index

+.networkLite, 4
-.networkLite *(+.networkLite)*, 4
[.networkLite *($.networkLite)*, 3
[<-.networkLite
(add.edges.networkLite), 4
$.networkLite, 3

add.edges.networkLite, 4
add.vertices.networkLite, 5
as.edgelist.networkLite, 6
as.matrix.networkLite
(as.edgelist.networkLite), 6
as.network.networkLite
(to_network_networkLite), 17
as.networkLite, 7, *16*, *18*
as_tibble.networkLite
(as.edgelist.networkLite), 6
atomize, *7*, 8, *16*

delete.edge.attribute.networkLite
(get.vertex.attribute.networkLite),
10
delete.edges.networkLite, 9
delete.network.attribute.networkLite
(get.vertex.attribute.networkLite),
10
delete.vertex.attribute.networkLite
(get.vertex.attribute.networkLite),
10
delete.vertices.networkLite, 9

get.edge.attribute.networkLite
(get.vertex.attribute.networkLite),
10
get.edge.value.networkLite
(get.vertex.attribute.networkLite),
10
get.inducedSubgraph.networkLite, 10
get.network.attribute.networkLite
(get.vertex.attribute.networkLite),
10
get.vertex.attribute.networkLite, 10

is.na.networkLite, 12
is.networkLite, 13

list.edge.attributes.networkLite
(get.vertex.attribute.networkLite),
10
list.network.attributes.networkLite
(get.vertex.attribute.networkLite),
10
list.vertex.attributes.networkLite
(get.vertex.attribute.networkLite),
10

mixingmatrix.networkLite, 13

network, *3*
network.edgecount.networkLite, 14
network.naedgecount.networkLite
(network.edgecount.networkLite),
14
network::as.edgelist, *15*
network::get.inducedSubgraph(), *10*
network::permute.vertexIDs(), *17*
networkLite, *3*, *13*, 14
networkLite-package, 2
networkLite.edgelist *(networkLite)*, 14
networkLite.matrix *(networkLite)*, 14
networkLite.numeric *(networkLite)*, 14
networkLite_initialize *(networkLite)*, 14

permute.vertexIDs.networkLite, 16
print.networkLite, 17

set.edge.attribute.networkLite
(get.vertex.attribute.networkLite),
10
set.edge.value.networkLite
(get.vertex.attribute.networkLite),
10

set.network.attribute.networkLite
(get.vertex.attribute.networkLite), 10
set.vertex.attribute.networkLite
(get.vertex.attribute.networkLite), 10

to_network_networkLite, *8*, 17

valid.eids.networkLite, 18