Package 'intergraph'

August 21, 2023

Type Package

Title Coercion Routines for Network Data Objects

Version 2.0-3

Description Functions implemented in this package allow to coerce (i.e. convert) network data between classes provided by other R packages. Currently supported classes are those defined in packages: network and igraph.

URL https://mbojan.github.io/intergraph/

BugReports https://github.com/mbojan/intergraph/issues

Imports network (>= 1.4-2), igraph (>= 0.6-0), utils

Suggests rmarkdown, knitr, roxygen2, testthat, tibble

VignetteBuilder knitr

License GPL-3

LazyLoad yes

LazyData yes

RoxygenNote 7.2.3

Encoding UTF-8

Depends R (>= 2.10)

Config/testthat/edition 3

NeedsCompilation no

Author Michał Bojanowski [aut, cre] (<https://orcid.org/0000-0001-7503-852X>)

Maintainer Michał Bojanowski <michal2992@gmail.com>

Repository CRAN

Date/Publication 2023-08-20 23:22:33 UTC

R topics documented:

intergraph-package 2
asDF 3
asIgraph 5
asNetwork 6
attrmap 8
dumpAttr 9
exNetwork 10
netcompare 12

Index **14**

`intergraph-package` *Coercion Routines for Network Data Objects*

Description

This package contains methods for coercion between various classes used to represent network data in R.

Details

Functions implemented in this package allow to coerce (i.e. convert) network data between classes provided by other R packages. Currently supported classes are: "network" from package **network**, "igraph" from package **igraph**.

The main functions are:

- `asNetwork` and its methods to create objects of class "network".
- `asIgraph` and its methods to create objects of class "igraph".

See their help pages for more information and examples.

As all the supported packages are written using S3 methods, so are the methods in this package.

If you find this package useful in your work please cite it. Type `citation(package="intergraph")` for the information how to do that.

Author(s)

Written and maintained by Michal Bojanowski <m.bojanowski@icm.edu.pl>.

Examples

```
# example of converting objects between classes 'network' and 'igraph'
# needs packages igraph and network attached
if( require(network) & require(igraph) )
{

  ### convert 'network' -> 'igraph'
```

```
  # example network as object of class "network"
  summary(exNetwork)

  # convert to class "igraph"
  g <- asIgraph(exNetwork)

  # check if 'exNetwork' and 'g' are the same
  # (dropping some aux attributes)
  all.equal( structure(as.matrix(exNetwork, "edgelist"), n=NULL, vnames=NULL),
    igraph::get.edgelist(g) )

  # compare results using 'netcompare'
  netcompare(exNetwork, g)


  ### convert 'igraph' -> 'network'

  # example network as object of class "igraph"
  summary(exIgraph)

  # convert to class "network"
  gg <- asNetwork(exIgraph)

  # check if they are the same
  # (dropping some attributes)
  all.equal( get.edgelist(exIgraph),
             structure(as.matrix(gg, "edgelist"), n=NULL, vnames=NULL))
  netcompare(exIgraph, gg)
}
```

---

asDF *Convert network to data frame(s)*

Description

Convert a network data object to, possibly two, data frames: a data frame with an edge list with edge attributes (if any), and a data frame of vertexes with vertex attributes (if any). This is a generic function, see below for available methods.

Usage

```
asDF(object, ...)

## S3 method for class 'network'
asDF(object, ...)

## S3 method for class 'igraph'
asDF(object, ...)
```

Arguments

`object`	R object representing a network, see below for available methods
`...`	other arguments passed to/from other methods

Details

Currently there are methods for `object` being in one of the following classes: "network", "igraph".

The function first gets the graph edge list using the appropriate function depending on the class of `object` (see below). Edge attributes, if any, are then extracted using `dumpAttr` and added to it.

The vertex data frame is constructed with a vertex id as a sequence of integer numbers. Details are method-specific, see below. Vertex attributes are extracted with `dumpAttr` and added to this data frame.

Method-specific notes:

For objects of class "network". Objects of this class store the vertex ids as integer numbers. There is also an attribute "vertex.names" which is always created when using graph constructors provided in the package **network**. `asDF` adds "vertex.names" to the vertex data frame as a normal attribute and does not use it as a vertex id in the edge list.

The edge list is created using `as.matrix.network` function and contains integer vertex ids.

Objects of class "igraph", as provided by the **igraph** package. Vertex ids in these objects integers starting from 1 (in **igraph** version prior to 0.6-0 vertex ids started from 0). However, it is also possible to provide a vertex attribute "name". It is added to the vertex data frame as a normal vertex attribute and is not used on the edge list data frame.

The edge list is created using `get.edgelist` function with argument `names` set to `FALSE` so that integer vertex ids are used.

Value

List with two components:

`edges` containing an edge list data frame at first two columns and edge attributes on further ones.

`vertexes` with vertex id in the first column, named `id` and any vertex attributes in the other columns.

Examples

```
# using method for 'network' objects
d1 <- asDF(exNetwork)
str(d1)

# using method for 'igraph' objects
d2 <- asDF(exIgraph)
str(d2)
```

asIgraph *Coerce an object to class "igraph"*

Description

Coerce objects to class "igraph".

Usage

```
asIgraph(x, ...)

## S3 method for class 'network'
asIgraph(x, amap = attrmap(), ...)

## S3 method for class 'data.frame'
asIgraph(x, directed = TRUE, vertices = NULL, vnames = NULL, ...)
```

Arguments

x	R object to be converted
...	other arguments from/to other methods
amap	data.frame with attribute copy/rename rules, see attrmap
directed	logical, whether the created network should be directed
vertices	NULL or data frame, optional data frame containing vertex attributes
vnames	character, name of the column in vertices to be used as a name vertex attribute, if NULL no vertex names are created

Details

asIgraph is a generic function with methods written for data frames and objects of class "network".

If x is a data frame, the method used is a wrapper around graph.data.frame in package **igraph**. The vnames argument was added so that the user can specify which vertex attribute from the data frame supplied through vertices argument is used for vertex names (the name attribute in igraph objects) in the returned result. By default the vertex names are not created.

If x is of class "network" (package **network**) the function uses asDF to extract data on edges and vertex with their attributes (if present). Network attributes are extracted as well. Not all vertex/edge/network attributes are worth preserving though. Attributes are copied, dropped or renamed based on rules given in the amap argument, see attrmap for details. The function currently does not support objects that represent neither bipartite networks nor hypergraphs.

Value

Object of class "igraph".

See Also

graph.data.frame

Examples

```
### using 'asIgraph' on objects of class 'network'

g <- asIgraph(exNetwork)

# compare adjacency matrices
netmat <- as.matrix(exNetwork, "adjacency")
imat <- as.matrix(g, "adjacency")
# drop the dimnames in 'netmat'
dimnames(netmat) <- NULL
# compare
identical( netmat, imat )


### using 'asIgraph' on data.frames

# data frame with vertex ids and vertex attributes
v <- 1:4
vd <- data.frame(id = v + 5, label=letters[1:4])
print(vd)

# edge list (first two columns) and edge attributes
e <- c(1,2, 2,3, 3,4, 4,1)
ed <- data.frame(id1 = e[seq(1,8, by=2)]+5, id2=e[seq(2, 8, by=2)]+5, a=letters[1:4])
print(ed)

# build the network
# without vertex attributes
g <- asIgraph(ed, directed=FALSE)
# with vertex attributes
gv <- asIgraph(ed, vertices=vd, directed=FALSE)

# NOTE: Even though vertex ids start at 6 the network has 4 nodes:
range(vd$id) # min and max of node ids
if(require(igraph)) igraph::vcount(gv) # number of nodes in 'gv'
```

---

asNetwork *Convert objects to class "network"*

Description

Convert objects to class "network"

Usage

```
asNetwork(x, ...)

## S3 method for class 'data.frame'
asNetwork(x, directed = TRUE, vertices = NULL, ...)

## S3 method for class 'igraph'
asNetwork(x, amap = attrmap(), ...)
```

Arguments

x	an R object to be coerced, see Details for the description of available methods
...	other arguments from/to other methods
directed	logical, whether the created network should be directed
vertices	NULL or data frame, optional data frame containing vertex attributes
amap	data.frame with attribute copy/rename rules, see attrmap

Details

This is a generic function which dispatches on argument x. It creates objects of class "network" from other R objects.

The method for data frames is inspired by the similar function in package **igraph**: graph.data.frame. It assumes that first two columns of x constitute an edgelist. The remaining columns are interpreted as edge attributes. Optional argument vertices allows for including vertex attributes. The first column is assumed to vertex id, the same that is used in the edge list. The remaining colums are interpreted as vertex attributes.

The method for objects of class "igraph" takes the network of that class and converts it to data frames using asDF. The network is recreated in class "network" using asNetwork.data.frame. The function currently does not support bipartite "igraph" networks.

Value

Object of class "network".

See Also

graph.data.frame

asIgraph for conversion in the other direction.

Examples

```
# require package 'network' as 'asNetwork' generic is defined there
if(require(network, quietly=TRUE))
{
  ### demonstrating method for data frames
  l <- asDF(exNetwork)
  g <- asNetwork( l$edges, vertices=l$vertexes)
```

```
  stopifnot(all.equal(g, exNetwork))

  ### method for igraph objects
  ig <- asNetwork(exIgraph)
  identical( as.numeric(as.matrix(g, "adjacency")),
             as.numeric(as.matrix(ig, "adjacency")))
}
```

---

attrmap *Network attribute copying/renaming table*

Description

Setting and retrieving rules through which network/edge/vertex attributes are copied or renamed when converting network objects.

Usage

```
attrmap(newdf = NULL)
```

Arguments

newdf	data.frame, new set of copy/rename rules, see Details

Details

Different classes for network data use different attribute names to store the same information. Some of the classes define attributes not used by other classes. This function is used to retrieve or set the rules in which these attributes are copied, renamed or dropped when converting network objects.

The rules are stored in a data frame with the following columns (all of mode character):

type type, or level of the attribute, one of "vertex", "edge" or "network"

fromcls name of the class which is being coerced

fromattr name of the "source" attribute

tocls name of the class to which coercion is being done

toattr name of the attribute in the result of coercion, or NA

When converting network `x`, of class `xclass` to network `y` of class `yclass` the coercion method looks up the rows in this table for which `fromcls=xclass` and `tocls=yclass`. If network `x` has any of the attributes listed in the `fromattr` in the resulting subset of the table then, it is renamed to the corresponding value of `toattr`. If `toattr` is `NA` the attribute is dropped from the result.

Value

If `newdf` is NULL, a data frame with the current copy/rename rules. Otherwise `newdf` are set as new rules and the old ones are returned invisibly, much like `par`.

See Also

This is used by asIgraph and asNetwork

Examples

```
# Current values
attrmap()
```

---

dumpAttr *Dump network attributes to a list*

Description

Given a network return a list of all the attributes.

Usage

```
dumpAttr(x, ...)

## S3 method for class 'network'
dumpAttr(x, type = c("all", "network", "vertex", "edge"), ...)

## S3 method for class 'igraph'
dumpAttr(x, type = c("all", "network", "vertex", "edge"), ...)
```

Arguments

x	network object
...	other arguments from/to other methods
type	character, type of attributes to dump

Value

If type is one of "network", "vertex" or "edge" then a list of corresponding attributes.

If type is "all" then lists of lists of attributes.

Examples

```
# using 'igraph' object
l <- dumpAttr( exIgraph )   # all attributes
identical( dumpAttr(exIgraph, "network"), l$network )
identical( dumpAttr(exIgraph, "vertex"), l$vertex )
identical( dumpAttr(exIgraph, "edge"), l$edge )

# using 'network' object
```

```
l <- dumpAttr( exNetwork )   # all attributes
identical( dumpAttr(exNetwork, "network"), l$network )
identical( dumpAttr(exNetwork, "vertex"), l$vertex )
identical( dumpAttr(exNetwork, "edge"), l$edge )
```

---

exNetwork *Sample network structure*

Description

An examples of networks together with network, edge and vertex attributes used primarly for testing. The same networks are stored in objects of class `network` and `igraph`.

Format

exNetwork,exNetwork2 is of class `network`

exIgraph,exIgraph2 is of class `igraph`

Objects `exNetwork` and `exIgraph` store directed version of the network. Objects `exNetwork2` and `exIgraph2` store the undirected version: all direction information from the edges is removed.

The network consists of 15 vertices and 11 edges.

- Vertex 1 is an isolate.
- Vertices 2-6 constitute a star with vertex 2 as a center.
- Vertices 7-8 and 9-10 make two dyads
- Vertcies 11, 12, 13,14 and 15 make a stem-and-leaf network.

Details

Vertices and edges has attribute `label`. For vertices these are simply letters from "a" to "o". For edges these are two-letter sequences corresponding to the neightboring vertices, i.e. the label for the edges linking nodes "b" and "c" will be "bc". The order is irrelevant.

In the `exNetwork` object the `label` attribute is also copied to the `vertex.names` attribute to facilitate plotting.

The `exIgraph` object has additional graph attribute `layout` so that by default Fruchterman-Reingold placement is used for plotting.

Examples

```
if(require(network, quietly=TRUE) ) print(exNetwork)
if( require(igraph, quietly=TRUE) ) print(exIgraph)


# showing-off 'network' versions
if(require(network, quietly=TRUE))
{
```

```
  op <- par(mar=c(1,1,1,1))
  layout( matrix(1:2, 1, 2, byrow=TRUE) )
  # need to change the family to device default because of faulty 'igraph'
  plot(exNetwork, main="Directed, class 'network'", displaylabels=TRUE)
  plot(exNetwork2, main="Undirected, class 'network'", displaylabels=TRUE)
  par(op)
}

# not running because of a bug in 'igraph': plot.igraph wants to set default
# font for vertex labels to 'serif', which is not supported on all devices
if(FALSE) {
# showing-off 'igraph' versions
if(require(igraph, quietly=TRUE))
{
  op <- par(mar=c(1,1,1,1))
  layout( matrix(1:2, 1, 2, byrow=TRUE) )
  plot(exIgraph, main="Directed, class 'igraph'")
  plot(exIgraph2, main="Undirected, class 'igraph'")
  par(op)
}
}

# The data was generated with the following code
if(FALSE) {
# directed igraph
g <- igraph::graph( c(2,1, 3,1, 4,1, 5,1, # star
 6,7, 8,9, # two dyads
 10,11, 11,12, 12,13, 13,14, 14,12), # stem-leaf
 n=14, directed=TRUE)
# add some vertex attributes
vl <- letters[seq(1, vcount(g))]
g <- igraph::set.vertex.attribute(g, "label", value=vl)
# add some edge attributes
m <- igraph::get.edgelist(g)
l <- matrix(vl[m+1], ncol=2)
el <- apply(l, 1, paste, collapse="")
g <- igraph::set.edge.attribute(g, "label", value=el)
g <- igraph::set.graph.attribute(g, "layout", igraph::layout.fruchterman.reingold)
rm(vl, l, m, el)
exIgraph <- g

# undirected igraph
exIgraph2 <- igraph::as.undirected(exIgraph)
exIgraph2 <- igraph::set.edge.attribute(exIgraph2, "label",
value=igraph::get.edge.attribute(exIgraph, "label"))


# copy as a 'network' object through adjacency matrix
m <- igraph::get.adjacency(exIgraph)
g <- network::network(m, vertex.attr=list(label=vattr(exIgraph, "label")),
    vertex.attrnames="label", directed=TRUE)
network::set.vertex.attribute(g, "vertex.names", value=vattr(exIgraph, "label"))
network::set.edge.attribute(g, "label", igraph::get.edge.attribute(exIgraph, "label"))
```

```
exNetwork <- network::network.copy(g)

# copy as a 'network' object through adjacency matrix
m <- igraph::get.adjacency(exIgraph2)
g <- network::network(m, vertex.attr=list(label=vattr(exIgraph2, "label")),
    vertex.attrnames="label", directed=FALSE)
network::set.vertex.attribute(g, "vertex.names", value=vattr(exIgraph2, "label"))
network::set.edge.attribute(g, "label", igraph::get.edge.attribute(exIgraph2, "label"))
exNetwork2 <- network::network.copy(g)
}
```

`netcompare` *Comparing and testing network objects*

Description

Compare or test network objects for (near) equality.

Usage

```
netcompare(target, current, test = FALSE, ...)
```

Arguments

`target, current`	network objects, currently `network` and `igraph` classes are supported
`test`	logical, whether to perform the test or return comparison data, see Details
`...`	other arguments, currently ignored

Details

Arguments `target` and `current` can be network objects of one of the supported classes. They do not have to be of the same class though.

The function does a series of comparisons between `target` and `current`:

1. The network structure comparison is made based on adjacency matrices (mind this when using for huge networks).
2. Network/edge/vertex attributes are checked for presence in both objects.
3. Common network/edge/vertex attribures are checked for equality.

All the results are collected in a list of class `netcompare` with an associated `print` method.

If `test` is TRUE then instead of the detailed test results the function returns TRUE or FALSE depending on some of the checks resulted positively. Currently attribute checks are ignored, i.e. what is taken into account is:

1. Equivalence of adjacency matrices

2. Directed / undirected character of the network
3. Edge set size
4. Vertex set size

Value

Depending on the value of `test` either an object of class `netcompare` containing the results of all the tests (if `test=FALSE`) or (if `test=TRUE`) a logical whether or not the networks are (nearly) the same.

See Also

`all.equal`, `identical`

Examples

```
netcompare( asIgraph(exNetwork), exNetwork)
netcompare( asIgraph(exNetwork), exNetwork, test=TRUE)
```

Index

∗ **datasets**
exNetwork, 10
∗ **package**
intergraph-package, 2

all.equal, *13*
as.matrix.network, *4*
asDF, 3, *5*, *7*
asIgraph, *2*, 5, *7*, *9*
asNetwork, *2*, 6, *9*
attrmap, *5*, *7*, 8

dumpAttr, *4*, 9

exIgraph *(*exNetwork*)*, 10
exIgraph2 *(*exNetwork*)*, 10
exNetwork, 10
exNetwork2 *(*exNetwork*)*, 10

get.edgelist, *4*
graph.data.frame, *5–7*

identical, *13*
intergraph-package, 2

netcompare, 12

par, *8*