Package 'multiplex'

July 23, 2025

Type Package

Version 3.9

Depends R (>= 4.2.0)

Imports methods

Suggests multigraph, Rgraphviz, knitr

Title Algebraic Tools for the Analysis of Multiple Social Networks

Description Algebraic procedures for analyses of multiple social networks are delivered with this package as described in Ostoic (2020) <DOI:10.18637/jss.v092.i11>. 'multiplex' makes possible, among other things, to create and manipulate multiplex, multimode, and multilevel network data with different formats. Effective ways are available to treat multiple networks with routines that combine algebraic systems like the partially ordered semigroup with decomposition procedures or semiring structures with the relational bundles occurring in different types of multivariate networks. 'multiplex' provides also an algebraic approach for affiliation networks through Galois derivations between families of the pairs of subsets in the two domains of the network with visualization options.

Date 2025-01-29

Author Antonio Rivero Ostoic [aut, cre]

Maintainer Antonio Rivero Ostoic <multiplex@post.com>

URL https://github.com/mplex/multiplex/

BugReports https://github.com/mplex/multiplex/issues/

Repository CRAN

License GPL-3

VignetteBuilder knitr

NeedsCompilation no

Date/Publication 2025-01-29 12:50:02 UTC

Contents

`multiplex-package` *Algebraic Tools for the Analysis of Multiple Social Networks*

Description

One of the aims of the **multiplex** package is to meet the necessity to count with an analytic tool specially designed for social networks with relations at different levels. In this sense, **multiplex** counts with functions that models the local role algebras of the network based on the simple and compound relations existing in the system. **multiplex** has also a procedure for the construction and analysis of signed networks through the semiring structure. With **multiplex**, the different relational patterns at the dyadic level in the network can be obtained as well, which can serve for a further analysis with different types of structural theories.

It is also possible to take the attributes of the actors in the analysis of multiple networks with different forms to incorporate this kind of information to the existing relational structures. For instance, the network exposure of the actors can be taken in the context of multiple networks in this case, or else the attributes can be embedded in the resulted algebraic structures.

Details

Package:	multiplex
Type:	Package
Version:	3.9
Date:	29 January 2025
License:	GPL-3
LazyLoad:	yes

One way to work with this package is typically by starting with a specific algebraic structure like a semigroup that is a closed system made of a set of elements and an associative operation on it. This algebraic structure is constructed by the `semigroup` function, and it takes an array of (usually but not necessarily) multiple binary relations, which constitute the generator relations. The Word Table and the Edge Table serve to describe completely the semigroup, and they are constructed with the functions `wordT` and `edgeT` respectively. Unique relations of the complete semigroup are given by the `strings` function together with the set of equations with strings of length k. The `partial.order` function specifies the ordering of the string elements in the semigroup, and the function `hasse` (or function `diagram` with this type) produces the lattice of inclusions of a structure having ordered relations.

Semigroups can be analysed further by the `green.rel` function, and their found equivalence classes can be visualized as "`egg-box`" type with the `diagram` function. Semigroups can be reduced as well with a decomposition process, which can be based on congruence or π-relations of the unique strings. In this case `pi.rels`, `cngr`, and `decomp` will make this job for you either for an abstract or a partially ordered structure.

In addition, it is possible to analyse structural balance in signed networks, which are built by `signed`, through the algebraic structure of the semiring. A semiring is an algebraic structure that

combines an abstract semigroup with identity under multiplication and a commutative monoid under addition. The `semiring` function is capable to perform both balance and cluster semiring either with cycles or semicycles.

There are other capabilities in the package that are not strictly algebraic. For instance, the `dichot` serves to dichotomize the input data with a specified cut-off value, `rm.isol` removes isolated nodes, and the `perm` function performs an automorphism of the elements in the representative array. All these functions are built for multiple networks represented by high dimensional structures that can be constructed by function `zbind` to produce three-dimensional arrays.

Furthermore, **multiplex** creates a Relation-Box with the `rbox` function, and it implements the Compositional Equivalence expressed in the cumulated person hierarchy of the network computed with the `cph` function.

Relational bundles are identified through the `bundles` function, which provides lists of pair relations. The `transf` function serves to transform pairwise list data into a matrix form and viceversa. The enumeration of the different bundle classes is given by `bundle.census`, while `summaryBundles` prints the bundle class patterns results. An advantage of counting with the bundle patterns is that the different types of bundles serve to establish a system inside the network, in which it is possible to measure the network exposure in multivariate relational systems. Such features can be realized via the `rel.sys` and `expos` functions, respectively. Several attributes can be derived by `galois`, which provides an algebraic approach for the analysis of two-mode networks.

Finally, multivariate network data can be created using a *s*end *r*eceive *t*ies edge list format that can be loaded and transformed to arrays through the `edgel` function. Other formats for multiple network data like **UCINET** `dl` or **Visone** `gml` can be imported and exported as well with the **multiplex** package. Visualization of multiple network structures is possible with the **multigraph** package that depends on **multiplex**.

Author(s)

J. Antonio Rivero Ostoic

Maintainer: Antonio Rivero Ostoic <multiplex@post.com>

References

Pattison, P.E. *Algebraic Models for Social Networks*. Structural Analysis in the Social Sciences. Cambridge University Press. 1993.

Boyd, J.P. *Social Semigroups. A unified theory of scaling and blockmodelling as applied to social networks*. George Mason University Press. 1991.

Lorrain, F. and H.C. White, "Structural Equivalence of Individuals in Social Networks." *Journal of Mathematical Sociology*, 1, 49-80. 1971.

Boorman, S.A. and H.C. White, "Social Structure from Multiple Networks. II. Role Structures." *American Journal of Sociology*, 81 (6), 1384-1446. 1976.

Ostoic, J.A.R. *Algebraic Analysis of Social Networks*. Wiley Series in Computational and Quantitative Social Sciences. Wiley. 2021.

See Also

`multigraph`, `bmgraph`, `ccgraph`.

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.5, 3 ) )

# dichotomize it with customized cutoff value
dichot(arr, c = 3)

# preview
prev(arr)

# create the semigroup and look at Green's relations
semigroup(arr) |>
  green.rel()

# look at string relations
strings(arr)
```

`as.semigroup` *Coerce Relational System into a Semigroup Object*

Description

A generic function for coercing an **R** object to a `semigroup` class.

Usage

```
as.semigroup(x, gens = NA, lbs, numerical, edgeT)
```

Arguments

`x`	an array representing the semigroup
`gens`	array or vector representing the semigroup generators
`lbs`	(optional) label strings for the semigroup
`numerical`	(optional and logical) should the semigroup have numerical format?
`edgeT`	(optional and logical) is '`x`' an edge table?

Details

Because some of the routines in the `multiplex` package require an object of the '`Semigroup`' class, this function produces this class object from an array representing the semigroup structure.

Value

An object of the 'Semigroup' class

ord	a number with the dimension of the semigroup
st	the strings, i.e. a vector of the unique relations
gens	the semigroup generators
S	the multiplication table of the semigroup

Author(s)

Antonio Rivero Ostoic

See Also

semigroup, green.rel

Examples

```
# create labeled multiplication table data
s <- matrix(data=c(1, 1, 1, 3, 3, 3, 3, 3, 3), nrow=3, ncol=3, byrow=TRUE)

# set attributes to 's'
attr(s, "dimnames") <- list(1:3, 1:3)

# make a semigroup object
as.semigroup(s)
```

---

as.signed *Coerce to a Signed Object*

Description

A generic function for coercing an object to a Signed class.

Usage

```
as.signed(x, lbs)
```

Arguments

x	a matrix representing the signed network
lbs	(optional) labels for the signed matrix

Details

Since the semiring function requires an object with a 'Signed' class, this function produces this class object from an array representing the signed network

Value

The array as a Signed class

See Also

signed, semiring

Examples

```
# load the data
data("incubA")

# coerce parts of the signed matrix with two types of relations
signed(incubA$IM)$s[1:2,1:2] |>
  as.signed()
```

---

as.strings *Coerce an Object to a Strings Class*

Description

A generic function for coercing an R object to a Rel.Strings class.

Usage

```
as.strings(x, lbs = NULL)
```

Arguments

x	an array; usually with three dimensions of stacked matrices where the multiple relations are placed.
lbs	(optional) labels of strings

Details

This function is useful to proceed with the establishment of the partial order in the strings of relations where the object should be of a 'Strings' class.

Value

An object of 'Strings' class

wt	word tables
ord	number of unique relations in the semigroup

Author(s)

Antonio Rivero Ostoic

See Also

strings, partial.order, zbind

Examples

```
# create the data: two sets with a pair of binary relations among
# three elements
arr1 <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
        c(3,3,2))>.5, 3 ) )

arr2 <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
        c(3,3,2))>.5, 3 ) )

# bind the data sets
arrs <- zbind(arr1, arr2)

# make the data a strings object
as.strings(arrs)
```

---

bundle.census *Bundle Census*

Description

A function to perform the bundle census in multiple networks.

Usage

```
bundle.census(x, loops = FALSE)
```

Arguments

x	an array; usually with three dimensions of stacked matrices where the multiple relations are placed
loops	(logical) whether or not the loops should be considered

Details

This function calculates the number of occurrences for each bundle class pattern in multiple networks. A bundle is a particular type of pattern made of relations at different levels that is binding a pair of nodes or actors. Depending on the direction and occurrence of each possible tie, then it is possible to count with seven dyadic configuration classes in the census.

Value

A table with the occurrences in the distinct bundle class patterns. The first column in the output gives the number of bundles in the network, excluding the null pattern, and then the totals for each bundle class pattern are specified in the following columns. The last column of the table hosts loops in case these are activated in the input.

Note

Functions bundles and summaryBundles provide bundle class occurrences in the network with a more detailed information.

Author(s)

Antonio Rivero Ostoic

References

Ostoic, J. A. R. "Dyadic Patterns in Multiple Networks," *Advances in Social Networks Analysis and Mining, International Conference on*, 475-481. 2011.

See Also

bundles, summaryBundles

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.8, 3 ) )

# compute the bundle census
bundle.census(arr)
```

---

bundles *Bundle Class Patterns*

Description

Classify the Bundle class patterns in a system of multiple relations

Usage

```
bundles(x, loops = FALSE, smpl = FALSE, lb2lb = TRUE, collapse = FALSE,
        sep)
```

Arguments

x	an array; usually with three dimensions of stacked matrices where the multiple relations are placed
loops	(logical) whether or not the loops should be considered as a particular bundle
smpl	(logical) simplify the strings of relations? (default no)
lb2lb	(logical) should the labels of the nodes be included in the output? (default yes)
collapse	(logical) collapse the distinct levels of relations in the network? (default no)
sep	(optional) pair separator used for the pairwise relations

Details

A bundle is a particular type of pattern made of relations at different levels that is binding a pair of nodes or actors in a network of relationships. A bundle class is a dyadic configuration resulting from the mixture of the direction and the types of ties between the nodes or actors. There are in total seven dyadic configuration classes, which are *null*, *asymmetric*, *reciprocal*, *tie entrainment*, *tie exchange*, *mixed*, and the *full* bundle pattern. This function provides detailed information about the bundle class patterns in multiple networks as lists of pair relations among the nodes or actors, except for the "*null*" pattern.

In case that the nodes are not labeled, then an identification number will be assigned according to the location of the nodes in the array representation and as well when the `lb2lb` option is set to `FALSE`. This function assumes that the network is directed, and self ties are also considered in the output. Long string labels are simplified with `smpl`, whereas the `collapse` option blurs the levels in the strings.

Value

An object of '`Rel.Bundles`' class with the distinct bundle class patterns.

`asym`	asymmetric ties
`recp`	reciprocal ties
`tent`	tie entrainment
`txch`	tie exchange
`mixed`	mixed
`full`	full
`loops`	loops (if chosen)

Note

The input array for this function is always dichotomized, and it is possible to obtain the total number of occurrences in each bundle class pattern with the `bundle.census` function.

Author(s)

Antonio Rivero Ostoic

References

Ostoic, J. A. R. "Dyadic Patterns in Multiple Networks," *Advances in Social Networks Analysis and Mining, International Conference on*, 475-481. 2011.

See Also

`bundle.census`, `summaryBundles`, `transf`.

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.8, 3 ) )

# establish the different bundles
bundles(arr)
```

---

cngr *Congruence Relations*

Description

Find the congruence relations of a given abstract or a partially ordered semigroup.

Usage

```
cngr(S, PO = NULL, uniq)
```

Arguments

S	an object from the 'Semigroup' class.
PO	(optional) the partial order table
uniq	(optional and logical) whether or not return the unique congruence relations

Details

Congruencies are equivalence relations that preserve the operation between the correspondent classes in the algebraic structure. In this case, the different congruence classes are based on the substitution property of the semigroup object.

Value

An object of 'Congruence' class. The items included are:

S	semigroup of relations
PO	partial order table (if specified)
clu	congruence classes

Note

If the partial order is supplied in the input, then the computation of the congruence classes is slightly faster than for an abstract semigroup.

Author(s)

Antonio Rivero Ostoic

References

Hartmanis, J. and R.E. Stearns *Algebraic Structure Theory of Sequential Machines*. Prentice-Hall. 1966.

See Also

decomp, fact, pacnet

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.5, 1 ) )

# record the abstract semigroup of 'arr'
S <- arr |> semigroup()

# look at the congruences in S
cngr(S, PO = NULL)
```

---

comps *Find Components and Isolates in Multiple Networks*

Description

Function to find different components in the multiple network including isolated nodes.

Usage

```
comps(x, bonds = c("entire", "strong", "weak"), sort)
```

Arguments

x	array representing a given network
bonds	type of bonds to be used in the creation of the relational system for the different components
sort	(optional and logical) sort components by size? Default FALSE

Details

The different components in the network are obtained through the transitive closure of the bundle ties. By default, the *entire* system is chosen, but the option bonds allows choosing different types of relational bundles for the components. Argument sort is for output of components having increasing size; that is (if they exist) dyads, triads, and so on.

Value

A list with two possible system "components"

com	network components
isol	network isolates

Author(s)

Antonio Rivero Ostoic

See Also

bundles, rel.sys

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array( runif(18), c(3 ,3, 2) ), array( runif(18),
       c(3, 3, 2) ) > .9, 3 ) )

# find the components and isolates
comps(arr)
```

---

cph *Cumulated Person Hierarchy*

Description

A function to calculate the Cumulated Person Hierarchy in networks of multiple relations

Usage

```
cph(W, lbs)
```

Arguments

W	an object of the 'Rel.Box' class.
lbs	(optional) the labels of the relational system

Details

The cumulated person hierarchy is used to determine the partial structural equivalence among the actors in a multiple network. Two nodes are considered as *partial structural equivalent* iff they have identical role sets.

The outcome of this function depends on the characteristics of the Relation-Box.

Value

An object of 'Partial.Order' class with an array representing the cumulated person hierarchy.

Note

If the length of the labels differs from the order of the relational system, then labels will be ignored.

Author(s)

Antonio Rivero Ostoic

References

Breiger, R.L. and P.E. Pattison, "Cumulated social roles: The duality of persons and their algebras," *Social Networks*, 8, 215-256. 1986.

Mandel, M.J. "Roles and networks: A local approach." B.A. Honours thesis, Harvard University. 1978.

See Also

rbox, semigroup, diagram

Examples

```
# load the data
data("incubA")

# make the Relation Box of the image matrices and
# compute its cumulated person hierarchy
rbox(incubA$IM) |>
  cph()
```

---

decomp *Decomposition of a Semigroup Structure*

Description

A function to perform the decomposition of a semigroup structure

Usage

```
decomp(S, pr, type = c("mca", "pi", "at", "cc"), reduc, fac, force)
```

Arguments

S	an object of a 'Semigroup' class
pr	either an object of a 'Congruence' class or an object of a 'Pi.rels' class
type	type of decomposition; ie. the reduction is based on • mca meet-complements of atoms in the 'Pi.rels' class • pi π-relations in the 'Pi.rels' class • at atoms • cc congruence classes
reduc	(optional and logical) does the return object should include the reduced structures?
fac	(optional) the factor that should be decomposed
force	(optional and logical) force further reduction of the semigroup when S has NAs? (see details)

Details

The decomp function is a reduction form of an algebraic structure like the semigroup that verifies which of the class members in the system are congruent to each other. The decomposed object then is made of congruent elements, which form part of the lattice of congruence classes in the algebraic structure. In case that the input data comes from the Pacnet program, then such elements are in form of π-relations or the meet-complements of the atoms; otherwise these are simply equivalent elements satisfying the substitution property.

Sometimes a 'Semigroup' class object contains not available data in the multiplication table, typically when it is an image from the fact function. In such case, it is possible to perform a reduction of the semigroup structure with the force option, which performs additional equations to the string relations in order to get rid of NAs in the semigroup data.

Value

An object of 'Decomp' class having:

clu	vector with the class membership
eq	the equations in the decomposition
IM	(optional) the image matrices
PO	(optional) the partial order table
ord	(optional) a vector with the order of the image matrices

Note

Reduction of the partial order table should be made by the reduc function.

Author(s)

Antonio Rivero Ostoic

References

Pattison, Philippa E. *Algebraic Models for Social Networks*. Cambridge University Press. 1993.

Hartmanis, J. and R.E. Stearns *Algebraic Structure Theory of Sequential Machines*. Prentice-Hall. 1966.

See Also

fact, cngr, reduc, pi.rels, semigroup, partial.order, green.rel.

---

diagram *Plot Diagrams of Ordered or Linked Relations*

Description

A function to plot and manipulate Hasse and Concept diagrams of ordered relations, or the Egg-box of a semigroup structure.

Usage

```
diagram(x, type = c("hasse", "concept", "egg-box"), attrs = NULL, main = NULL,
       incmp, cex.main, bg, mar, shape, col, col0, fcol, ecol, lty, lbs,
       ffamily, fstyle, fsize, col.main, sep, ...)
```

Arguments

x	a matrix representing ordered relations
type	type of diagram: • hasse Hasse diagram of partially ordered relations • concept Concept lattice of a formal context • egg-box Egg-box diagram of an abstract semigroup

For egg-box, the following arguments are ignored.

attrs	(optional) attributes of the diagram
main	(optional) title of the diagram
incmp	(logical) whether or not incomparable elements should be included in the lattice diagram
cex.main	(optional) font size of diagram's title
bg	(optional) background color of diagram
mar	(optional) margins of plot
shape	(optional) shape of vertices
col	(optional) color of vertices
col0	(optional) color of vertices' contour
fcol	(optional) color of text's vertices

`ecol`	(optional) color of edges
`lty`	(optional) shape of edges
`lbs`	(optional) labels of elements in partially ordered set
`ffamily`	(optional) font family of vertex labels
`fstyle`	(optional) font style of vertex labels with options: • `bold` • `italic` • `bolditalic`
`fsize`	(optional) font size of vertex labels
`col.main`	(optional) font color of main title
`sep`	(optional) pair separator for pairwise relations inside intents and extents
`...`	(optional) additional graphical items

Details

`diagram` is a wrapper function to plot and manipulate "Hasse", "Concept", and "Egg-box" types of diagrams.

The first two diagrams are for systems of ordered relations, and the plotted deviced is either a partial order or a linear order diagram. An example of ordered relations is found in the partial order table of relations product of the '`strings`' option in the `partial.order` function, and which is plotted as a Hasse diagram. Another set of ordered relations comes from the table produced on Galois derivations within Formal Concept analysis where a Concept diagram represents the ordering relations among formal concepts made of intents and extents.

The Egg-box diagram is for equivalence classes in an abstract semigroup not associated to a partial order structure.

Value

Depending on the `type`:

`hasse`	a Hasse diagram of partially ordered relations
`concept`	a Concept diagram of formal concepts in a formal context
`egg-box`	an Egg-box of an abstract semigroup

Warning

Requires *Rgraphviz* package installed.

Note

Roman numerals are given for elements when the partial order table is not labelled.

Author(s)

Antonio Rivero Ostoic

See Also

hasse, partial.order, strings, galois, green.rel, diagram.levels, as.strings, ccgraph.

Examples

```
# load a dataset
data("incubA")

# given e.g. a partial order table in the object 'po'
po <- as.strings(incubA$IM) |>
  partial.order(type="strings")

# plot the order relation as a Hasse diagram
## Not run: if(require(Rgraphviz)) {
plot(diagram(po, type="hasse"))
}
## End(Not run)
```

---

diagram.levels *Levels in Lattice Diagram*

Description

A function that reads the different levels in the lattice diagram of the partial order structure among actors and ties in the network

Usage

```
diagram.levels(x, perm = FALSE)
```

Arguments

x	A matrix representing the partial order
perm	(optional) whether or not to return the permuted structure

Details

When it comes to reduce the structure of a multiple network, many times the partial order structure provides different classes of elements depending in the inclusions these elements have. In this sense, the illustration given by the diagram function provides us typically with different levels of the ordered relations, which are read by this routine.

Value

A named list with components of the "levels" in the concept diagram produced by diagram.

If the permutation is specified in perm, a data frame with the elements of the partial order structure with the column names indicating the element class plus a vector with the levels and a matrix with the permuted structure are given as well.

Note

This function requires that the *Rgraphviz* package is installed. Since function `grDevices::pictex()` inside this routine is for historical interest only since **R** 4.4.0, the warning message has been suppressed before its future replacement.

Author(s)

Antonio Rivero Ostoic

See Also

`partial.order`, `diagram`, `perm`

Examples

```
# load the data
data("incubA")

# given e.g. a partial order table in the object 'po'
po <- as.strings(incubA$IM) |>
  partial.order()

# find the levels in the lattice diagram
## Not run:
diagram.levels(po)

## End(Not run)
```

---

dichot *Dichotomize Data with a Cutoff Value*

Description

Function to dichotomize the input data for the semigroup construction with a cutoff value.

Usage

```
dichot(x, c = 1, diag)
```

Arguments

`x`	some data in a numeric form (usually arrays)
`c`	the cutoff value to perform the dichotomization (default `1`)
`diag`	(optional and logical) whether or not the diagonals should be included (default `TRUE`)

Details

This is a convenient function (or wrapper if you like) of the replace function. In this case, the function is aimed to specify a cutoff value for the dichotomization of the data where the values equal or higher to the cutoff are converted to one, while the others are set to zero. The cutoff value in c can be any real number.

Value

Binary values of the input data.

Note

Labels are preserved after the dichotomization.

Author(s)

Antonio Rivero Ostoic

See Also

replace, prev, semigroup.

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.5, 3 ) )

# dichotomize it with a cutoff value of two
dichot(arr, c = 2)
```

---

edgel *Read Edge List Files*

Description

A function to read edge list files with *s*end, *r*eceive, and *t*ies format for a multivariate network with the possibility to transform it into an three dimenasional array.

Usage

```
edgel(file, header = TRUE, sep = "\t", toarray = TRUE, dichot = FALSE,
      attr = FALSE, rownames = FALSE, add = NULL, na.rm)
```

Arguments

file	path to the file
header	(logical) does the file has a header?
sep	the separator among the columns (default is horizontal tab escaped as "\t")
toarray	(logical) should the data frame be transformed to arrays?
dichot	(logical) should the data be dichotomized?
attr	(logical) whether or not the file corresponds to attribute-based data
rownames	(logical) are rownames the labels of the nodes?
add	(optional) isolates to be added to the network
na.rm	(optional and logical) remove NAs in file?

Details

edgel is a function to read edge list files with *s*end, *r*eceive, and *t*ies format, which is a data frame with at least 2 columns for the sender, receiver and for multiplex networks also the ties, one column for each type of relation. However, the attr option correspond to a actor and self-ties data frame file with the option to transform it into a diagonal matrix. When toarray is set to FALSE, options attr and rownames allow placing the first column of the data frame as the name of the table, which is the format of two-mode data, and compute for instance Galois transformations among the partite sets. If more than one isolate is added, then the data must be included as a vector.

It is also possible to treat the input data as data frame object and manipulate it through e.g. the subset function with the toarray option. Valued networks are now supported as well.

Value

By default an array; usually with three dimensions of stacked matrices where the multiple relations are placed. If toarray = FALSE, then the data frame is returned.

Note

For backwards compatibility, an alias for edgel is read.srt.

Author(s)

Antonio Rivero Ostoic

See Also

write.edgel, read.gml, read.dl, galois

edgeT *Edge Table Generator*

Description

Function to produce the *Edge Table* generator of multiple relations.

Usage

```
edgeT(x)
```

Arguments

x	an array; usually with three dimensions of stacked matrices where the multiple relations are placed.

Details

The Edge Table is the complete right multiplication table of the semigroup having its elements for each of its generators.

Value

An object of the 'EdgeTable' class

gens	the generator relations
ET	the Edge Table

Author(s)

Antonio Rivero Ostoic

References

Cannon, J.J. "Computing the ideal structure of finite semigroup," *Numerische Mathematik*, 18, 254-266. 1971.

Pattison, P.E. *Algebraic Models for Social Networks*. Cambridge University Press. 1993.

See Also

wordT, semigroup.

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.5, 1 ) )

# obtain the edge table
edgeT(arr)
```

---

expos *Network Exposure for Multiple Networks*

Description

Function to measure the network exposure of the nodes according to a chosen relational system representing the multiple network.

Usage

```
expos(rs, classes = FALSE, allClasses = FALSE, allNodes = TRUE)
```

Arguments

rs	an object of 'Rel.System', typically with node attributes.
classes	(optional) whether or not should be included in the output the categories of adopters
allClasses	(optional) whether or not to include empty classes within the categories of adopters. Ignored if classes is FALSE
allNodes	(optional) whether or not to include all actors in the network regardless they are in the chosen system. Ignored if classes is FALSE

Details

This is a generalization of the network exposure measure for multiple networks with the characteristics chosen for the representative relational system. Such a system can be the entire network or configuration with strong or weak bonds among the actors. It is possible to specify different behaviors of the nodes representing social actors, which are indicated in the form of a relational system. The network exposure measure is computed according to the immediate neighbours to the reference actor.

Value

Classes	if option classes is set to TRUE, the adoption membership for the type of relational system chosen, including isolated actors in the system.
Bonds	the type of bonds of the relational system (cf. rel.sys)
Exposure	the exposure to the attribute(s) for acquisition through immediate neighbour relations

Author(s)

Antonio Rivero Ostoic

References

Ostoic, J.A.R. "Creating context for social influence processes in multiplex networks." *Network Science*, 5(1), 1-29.

Valente, T. W. *Social networks and health*. Oxford University Press. 2010.

Rogers, E. *The Diffusion of Innovations*. 5th ed. (1st ed. 1964) The Free Press. 2003.

See Also

rel.sys, neighb, bundles

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array( runif(18), c(3 ,3, 2) ), array( runif(18),
              c(3, 3, 2) ) > .9, 3 ) )

# first array in 'arr' is for attributes
rs <- rel.sys(arr, att = 1)

# compute the exposure measure for an attribute type with adopter categories
expos(rs, classes = TRUE)
```

---

fact *Factorisation of Semigroup Structures*

Description

A function to decompose partially ordered semigroups

Usage

```
fact(S, P, uniq = TRUE, fac, atoms, mca, atmc, patm, k)
```

Arguments

S	semigroup object
P	partial order structure associated to S
uniq	(logical) whether factorisation should include unique induced inclusions
fac	'factor' to be factorised, in case that input factorised partially ordered structures
atoms	(logical) whether or not include in output atoms
mca	(logical) whether or not include in output meet-complements of atoms
atmc	(logical) whether or not include in output atoms' meet-complements
patm	(logical) whether or not include in output potential atoms
k	(for patm) length of induced inclusion

Details

The factorisation is part of decomposition for partially ordered semigroups, and function `fact` allows to obtain elements generated in this process.

Value

An object of '`Ind.incl`' class having:

`po`	partial order table
`iin`	list of induced inclusions pairwise listed
`niin`	length of induced inclusions
`patm`	(for `patm`) a vector with potential atoms
`atm`	vector with atoms
`atmc`	(for `atmc`) array with meet-complements of atoms
`mc`	array of meet-complements of atoms
`note`	(if needed) induced inclusions without substitution property

Note

Data objects imported with `pacnet` are compatible for further factorisation.

Author(s)

Antonio Rivero Ostoic (based on the algorithm described in Ardu, 1995)

References

Ardu, S. *ASNET – Algebraic and Statistical Network Analysis. User Manual*. University of Melbourne. 1995.

See Also

`decomp`, `cngr`, `pacnet`

Examples

```
# create a partially ordered semigroup
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.5, 1 ) )

# semigroup of relations
S <- semigroup(arr)

# string relations and partial order
P <- strings(arr) |>
  partial.order()

# perform the factorisation of PO S
fact(S, P)
```

`fltr` *Principal Order Filters*

Description

A function to find principal order filters and order ideals in a partial order

Usage

```
fltr(x, PO, ideal = FALSE, rclos)
```

Arguments

`x`	a reference element in the partial order (integer or character)
`PO`	the partial order
`ideal`	(logical) whether or not the "filter" is an order ideal
`rclos`	(optional and logical) apply reflexive closure?

Details

This function helps to find principal order filters or principal order ideals for an element in a partial order structure. Such inputs are normally a concept or an object or attribute in the concept together with the associated partial ordering structure of the concepts, which results from Galois derivations. Typically, if the reference element refers to a concept, then it is given as a positive integer indicating the concept label. Another option is to refer to an object or an attribute by a character name, which should be part of the labels of the dimensions of the partial order table with reduced labelling. Principal order filters with full labelling are not allowed if the reference element is an object or an attribute. Use an integer for the concept instead.

Value

A named list with the elements in the upset or downset of the principal order filter or order ideal corresponding to the reference element in the partial order.

Author(s)

Antonio Rivero Ostoic

References

Ganter, B. and R. Wille *Formal Concept Analysis – Mathematical Foundations*. Springer. 1996.

See Also

`galois`, `partial.order`, `diagram`.

Examples

```
# create a data frame
dfr <- data.frame(x=1:3, y=5:7)

# partial ordering of concepts
PO <- dfr |>
  galois() |>
  partial.order(type="galois")

# order filter for the first element
fltr(1, PO, rclos=TRUE)
```

`galois` *Galois Derivations Between Subsets*

Description

Function to perform Galois derivations between partially ordered subsets

Usage

```
galois(x, labeling = c("full", "reduced"), sep, valued, scl,
    sep2)
```

Arguments

`x`	a data frame with objects and attributes
`labeling`	whether the derivations should be • `full` for full labeling • `reduced` for reduced labeling
`sep`	(optional) pair separator used for the pairwise relations
`valued`	(logical) whether the galois derivation is on a many-valued formal context
`scl`	(optional, only for valued) the scale to be used in the galois derivation
`sep2`	(optional, only for valued) the separator in the formal concept

Details

Galois derivations (or connections) are mappings between families of partially ordered subsets of elements. Such derivations are useful to analyze the structure of both subsets, which in a social network are typically the actors and their corresponding affiliations or events. That is, two-mode networks, but also a group of objects with a list of different attributes as used in formal concept analysis.

Value

A labelled list with Galois derivations of objects and attributes

Note

Full labeling implies first objects and then attributes, whereas the reduced option is given the other way around.

Author(s)

Antonio Rivero Ostoic

References

Ganter, B. and R. Wille *Formal Concept Analysis – Mathematical Foundations*. Springer. 1996.

See Also

`partial.order`, `diagram`, `fltr`.

Examples

```
# create a data frame
dfr <- data.frame(x=1:3, y=5:7)

# find galois derivations
galois(dfr)
```

`green.rel` *Green's Relations of Abstract Semigroups*

Description

A function to produce the Green's relations of a semigroup object.

Usage

```
green.rel(x)
```

Arguments

`x`	an object of a '`Semigroup`' class

Details

Function `green.rel` produces the *egg-box diagram* (Green, 1951) of an abstract semigroup S, which is the union of the left compatible R equivalence and the right compatible L equivalence classes that makes the D-class on S.

Value

A list with the abstract semigroup, clustering of equivalence classes, and egg-box diagram that are separated by "|" and "-".

S	multiplication matrix of the input semigroup
ord	dimension of the semigroup
st	vector of the unique string relations
clu	list of vectors with clustering information for R and L classes
R	R equivalence classes
L	L equivalence classes
D	D equivalence classes

Note

Some systems have the D-class equal to S.

Author(s)

Antonio Rivero Ostoic

References

Green, J. "On the structure of semigroups," *Annals of Mathematics* 54(1), 163–172, 1951.

Ostoic, JAR "Relational systems of transport network and provinces in ancient Rome," in *Mathematics for social sciences and arts – algebraic modeling*. Springer Nature. 2023.

See Also

semigroup, diagram, as.semigroup, edgeT, wordT, fact, cngr, decomp

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.5, 1 ) )
# optional string labels
dimnames(arr)[[3]] <- list("n", "m")

# look at the semigroup and its Green's relations
semigroup(arr) |>
  green.rel()
```

hasse *Hasse Diagram of Set of Ordered Relations*

Description

A function to plot the Hasse Diagram of partially ordered relations.

Usage

```
hasse(x, attrs = NULL, main = NULL, incmp, cex.main, bg, mar, shape, col,
      col0, fcol, ecol, lty, lbs, ffamily, fstyle, fsize, col.main, sep, ...)
```

Arguments

x	a matrix representing ordered relations
attrs	(optional) attributes of the diagram
main	(optional) title of the diagram
incmp	(logical) whether or not incomparable elements should be included in the lattice diagram
cex.main	(optional) font size of diagram's title
bg	(optional) background color of diagram
mar	(optional) margins of plot
shape	(optional) shape of vertices
col	(optional) color of vertices
col0	(optional) color of vertices' contour
fcol	(optional) color of text's vertices
ecol	(optional) color of edges
lty	(optional) shape of edges
lbs	(optional) labels of elements in partially ordered set
ffamily	(optional) font family of vertex labels
fstyle	(optional) font style of vertex labels with options: • bold • italic • bolditalic
fsize	(optional) font size of vertex labels
col.main	(optional) font color of main title
sep	(optional) pair separator for pairwise relations inside intents and extents
...	(optional) additional graphical items

Details

A Hasse diagram is a pictorical device to represent systems of partially ordered relations where the `hasse` function provides arguments for visual manipulation of the diagram. An example of a partially ordered system is the partial order table that is the outcome of the '`strings`' option in the `partial.order` function.

Value

A plot of a Hasse diagram with specified settings for a partial or a linear order of relations.

Warning

Requires *Rgraphviz* package installed.

Note

Roman numerals are given for elements when the partial order table have `NULL` dimnames.

Author(s)

Antonio Rivero Ostoic

See Also

`diagram`, `partial.order`, `strings`, `galois`, `green.rel`, `diagram.levels`, `as.strings`.

Examples

```
# load a dataset
data("incubA")

# given e.g. a partial order table in the object 'po'
po <- as.strings(incubA$IM) |>
  partial.order(type="strings")

# plot the order relation as a Hasse diagram
## Not run: if(require(Rgraphviz)) {
plot(hasse(po))
}
## End(Not run)
```

`hierar` *Person and Relation Hierarchy*

Description

Function to establish either the Person or the Relation Hierarchy in a multiple network

Usage

```
hierar(W, x, type = c("person", "relation"))
```

Arguments

`W`	an object of '`Rel.Box`'
`x`	(integer or character) actor of reference, either by its location in the adjacency matrix or by the label.
`type`	whether the hierarchy with respect to network "`x`" is • `person` for persons hierarchy • `relation` for relations hierarchy

Details

The `person` hierarchy refers to the inclusion relations among the actors, whereas the `relation` hierarchy refers to the inclusion relations among the ties, and both are from the perspective of a chosen actor of reference in the given network.

Value

An array that represents the partial order structure of the respective hierarchy.

Note

The cumulative person hierarchy is obtained through the `cph` function.

Author(s)

Antonio Rivero Ostoic

References

Breiger, R.L. and P.E. Pattison, "Cumulated social roles: The duality of persons and their algebras," *Social Networks*, 8, 215-256. 1986.

See Also

`rbox`, `cph`, `partial.order`, `diagram`

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array( runif(18), c(3 ,3, 2) ), array( runif(18),
c(3, 3, 2) ) > .5, 3 ) )

# compute person hierarchy of a random actor from relation box
arr |> rbox(k=1) |>
  hierar(ceiling(runif(1, min=0, max=dim(arr)[2])), type="person")
```

---

incubs *Incubator Networks Datasets*

Description

Four data sets collected in year 2010 (see 'source' for the details) of multiple relations between entrepreneurial firms working in business incubators in Denmark.

Each data set contains the adjacency matrices of the three social relations, coded as `C`, `F`, and `K` for working collaboration, informal friendship, and perceived competition among the firms. There are also a pair of actor attributes corresponding to the adoption of two Web innovations in the year 2010 by the firms where `A` stands for Linkedin and `B` for Facebook websites.

In addition, there is a blockmodel attached to each data set that is a product of Compositional Equivalence (cf. cph) with transposes for each type of social tie labelled with the following letter in the Latin alphabet; i.e. `D` for collaboration, `G` for friendship, and `L` for perceived competition.

Usage

```
data("incubs")
data("incubA")
data("incubB")
data("incubC")
data("incubD")
data("incA")
data("incB")
data("incC")
data("incD")
```

Format

Each data set is a list with a pair of three-dimensional arrays.

For `incubA`, the dimensions of `net` are $26 \times 26 \times 5$, and of `IM` are $4 \times 4 \times 7$. In this case, the two attributes led to the identity matrix.

For `incubB`, the dimensions of `net` are $18 \times 18 \times 5$, and of `IM` are $4 \times 4 \times 8$.

For `incubC`, the dimensions of `net` are $22 \times 22 \times 5$, and of `IM` are $3 \times 3 \times 8$.

For `incubD`, the dimensions of `net` are $15 \times 15 \times 5$, and of `IM` are $4 \times 4 \times 6$.

All four network datasets are gather together in object `incubs`.

To plot automatically actor attributes in the graph with function multigraph, another version of these data sets are given in incA, incB, incC, and incD, which are "Data.Set" objects class having:

- net for the network data
- atnet a vector that indicates whether or not the arrays in 'net' is attribute data
- IM for the Image Matrices of the reduced network data
- atIM a vector that indicates whether or not the array in 'IM' is attribute data
- cite relational content of the ties

Source

Ostoic, J.A.R. "Algebraic methods for the analysis of multiple social networks and actors attributes." PhD Thesis. University of Southern Denmark. 2013.

---

mlvl *Construct Multilevel Networks*

Description

Function to construct multilevel networks from multimodal structures.

Usage

```
mlvl(x = NULL, y = NULL, type = c("bpn", "cn", "cn2", "list"), symCdm,
     diag, lbs)
```

Arguments

x	domain data
y	codomain data
type	type of multilevel system: • bpn for binomial projection • cn for common membership network • cn2 for co-affiliation of network members • list for the multimodal structures as a list
symCdm	(optional and logical, only for bpn) whether or not symmetrize the codomain structure
diag	(optional and logical) whether or not include the entries in the diagonal matrices
lbs	(optional, only for cn2) tie labels

Details

The default multilevel system is a binomial projection bpn that requires data for the two domains, as with cn2 as well.

Option cn does not need the domain in "x" since returns the co-affiliation of network members from the codomain structure.

Since these are different components in the multilevel system for co-affiliation of network members, it is possible to specify the domain and codomain labels in lbs as a list object.

Making symmetric the codomain structure with symCdm is many times convenient for visualization purposes.

Value

An object of 'Multilevel' class of chosen type.

mlnet	the multilevel network
lbs	(list) domain and codomain labels
modes	a vector indicating the domain of the data in mlnet where 1M is for domain and 2M is for the codomain.

Author(s)

Antonio Rivero Ostoic

See Also

mlgraph, multigraph

Examples

```
# array for the domain
arr1 <- round( replace( array(runif(18), c(3,3,2)), array(runif(18), c(3,3,2))>.9, 3 ) )

# rectangle array for the co-domain
arr2 <- round( replace( array(runif(12), c(3,2,2)), array(runif(12), c(3,2,2))>.9, 3 ) )

# multilevel system with default type
mlvl(arr1, arr2)
```

---

mnplx *Make Multiple Networks as Monoplex Structures*

Description

A function to transform multiple networks into a monoplex structure

Usage

```
mnplx(net, directed = TRUE, dichot, diag, clu)
```

Arguments

net	a three-dimensional array to be transformed into a matrix
directed	(optional) whether to make the matrix symmetric or not
dichot	(optional) should the output be dichotomized?
diag	(optional) should the diagonals be included?
clu	(optional) a vector with the cluster for the permutation

Details

With this function, it is possible to collapse multiple types of tie of a given network into a matrix representation with monoplex relations. Besides dichotomizing, directed networks can be converted into undirected systems as well with or without self-relations. Moreover, the resulted matrix can be permuted with a clustering information in a vector in `clu` as with `perm`.

Value

A matrix of monoplex relations

Author(s)

Antonio Rivero Ostoic

See Also

`zbind`, `dichot`, `reduc`

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.5, 1 ) )

# make array monoplex
mnplx(arr)
```

neighb *Neighborhood of Actor or Group of Actors*

Description

A function to find the neighbourhood of an actor or group of actors with a customized distance.

Usage

```
neighb(x, rs, type = c("und", "inn", "out"), k = 1, inclx = FALSE, expand)
```

Arguments

x	the reference actor labeled in rs or a vector of several actors
rs	the relational system of the network
type	whether the system is • und for *undirected* (default) • inn for *incoming* node's ties to the reference actor • out for *outgoing* arcs from the reference actor
k	the "distance" of the neighbour nodes to the reference actor (where k=1 gives the adjacent nodes)
inclx	(logical) should the reference actor be included in the output?
expand	(optional and logical) should the output be given by k (it only makes sense when k>1)

Details

The relational system serves to represent either the entire multiple network made of actors, or else just the relational bundles having a mutual or an asymmetric character in the system. In this sense, this function detects the adjacent nodes to 'x' according to the specified relational system, but as well the neighbours of the adjacent nodes with a customized length. Eventually, when the longest path or chain is reached, adding more value to k obviously will not produce more nodes in the graph system. Type options inn and out are for directed networks.

Value

Depending on expand, the output is either a vector or a list with the neighbour nodes to the reference actor(s).

Note

The output does not differentiate in case the reference actors are in different components of the network.

Author(s)

Antonio Rivero Ostoic

See Also

expos, rel.sys, bundles

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array( runif(18), c(3 ,3, 2) ), array( runif(18),
       c(3, 3, 2) ) > .9, 3 ) )

# establish the system of strong bonds
rs <- rel.sys(arr, bonds = "strong")

# obtain inmediate neighbourhood of the first node
neighb(1, rs)
```

---

pacnet *Import Output from Pacnet*

Description

A function to read output files and import from the Pacnet program with full factorization option.

Usage

```
pacnet(file, toarray = FALSE, uniq = FALSE, transp = FALSE, sep)
```

Arguments

file	character vector containing a file name or path
toarray	(logical) should the induced inclusions be transformed into arrays?
uniq	(logical) should only be considered the induced inclusions that are unique?
transp	(logical) should the partially ordered structures be transposed?
sep	(optional) pair separator used for the pairwise relations

Details

This function is used to read the output file from the Pacnet program, which typically has the .out extension. By default the result is given in a list format, but it is possible to transform the pair lists into arrays. Note that the options in the Pacnet program should include the full factorization in the output; otherwise the object will be NULL.

Value

An object of the 'Pacnet' class with items:

ii	induced inclusions
at	atoms
mc	meet complements

Note

Currently only partial order structures of order 36 and less are supported.

Author(s)

Antonio Rivero Ostoic

References

Pattison, P., S. Wasserman, G. Robins and A.M. Kanfer "Statistical Evaluation of Algebraic Constraints for Social Networks," *Journal of Mathematical Psychology*, 44, 536-568. 2000

See Also

pi.rels, cngr, decomp, write.dat

---

partial.order *The Partial Order of String Relations or of Galois Derivations*

Description

Construct the partial order table of unique relations of the semigroup, or else of the concepts produced by Galois derivations.

Usage

```
partial.order(x, type = c("strings", "galois", "pi.rels"), lbs, sel,
              po.incl, dichot)
```

Arguments

x	an object of a 'Strings' or a 'Galois' class
type	whether the object corresponds to • strings for string relations • galois for Galois derivations • pi.rels for π-relations
lbs	(optional) the labels of the unique relations
sel	(optional) selected elements in x for the partial order

`po.incl`	(optional, works only with type `pi.rels`) should the partial order in the π-relations be included
`dichot`	(optional) should the string relations in `x` be dichotomized?

Details

To get the partial order of an entire semigroup, both generators and compound relations must be considered. This information and the labels of the unique relations are given by the `strings` function. cf. `semigroup` to see how the `x` should be specified properly.

Galois derivations are now possible to be partially ordered as well, and this option is based on the output given by the `galois` function.

Value

An object of '`Partial.Order`' class with the partial order table in a matrix form.

Author(s)

Antonio Rivero Ostoic

References

Pattison, P.E. *Algebraic Models for Social Networks*. Cambridge University Press. 1993.

Ganter, B. and R. Wille *Formal Concept Analysis – Mathematical Foundations*. Springer. 1996.

See Also

`as.strings`, `strings`, `galois`, `perm`, `diagram`, `fltr`.

Examples

```
# load the data to obtain its partial order
data("incubA")

# strings in the structure and partial order
strings(incubA$IM) |>
  partial.order()
```

`perm` *Array Permutation*

Description

Function to permutate a given array of relation.

Usage

```
perm(x, clu, rev, lbs, sort)
```

Arguments

`x`	a matrix or an array to be permuted
`clu`	the cluster for the permutation
`rev`	(optional and logical) whether the order in `clu` sholud be reverted.
`lbs`	(optional) the labels after the permutation
`sort`	(optional and logical) sort array according to labels?

Details

This function serves to permutate an array representing relations according to a vector for the clustering membership.

Value

A permuted matrix or array

Author(s)

Antonio Rivero Ostoic

See Also

`cph`, `partial.order`

Examples

```
# scan the multiplication table data
s <- matrix(data=c(1, 1, 1, 3, 3, 3, 3, 3, 3), nrow=3, ncol=3, byrow=TRUE)

# the permutation as an endomorphism
perm(s, clu = c(1,2,3))
```

`pfvn` *Pathfinder Valued Networks and Triangle Inequality*

Description

A function to establish the skeleton of a valued network with the pathfinder algorithm and triangle inequality

Usage

```
pfvn(x, r, q)
```

Arguments

x	network data, typically valued
r	a distance function parameter
q	parameter with the minimum distance between actors in the proximity matrix

Details

The Pathfinder structure is for undirected networks, whereas for directed network structures the triangle inequality principle is applied

Value

max	max value of the network with the Frobenius norm
r	parameter *r*
q	parameter *q*
Q	salient structure of *x*
Note	A note when triangle inequality is used

Author(s)

Antonio Rivero Ostoic

References

Schvaneveldt, R., Durso, F. and Dearholt, D., "Network structures in proximity data," in G. Bower, ed., *The psychology of learning and motivation: Advances in research & theory*, Vol. 24, Academic Press, pp. 249-284. 1989.

Batagelj, V., Doreian, P., Ferligoj, A. and Kejzar, N., *Understanding Large Temporal Networks and Spatial Networks: Exploration, Pattern Searching, Visualization and Network Evolution*, Wiley. 2014.

See Also

multigraph,

Examples

```
# create valued network data
arr <- round( array(runif(18), c(3,3,2)), array(runif(18), c(3,3,2)) ) * 10L

# pathfinder valued network of 'arr'
pfvn(arr)
```

pi.rels *π-Relations*

Description

A function to establish the π-relations of a partially ordered structure comming from a 'Pacnet' class

Usage

```
pi.rels(x, po.incl, vc, po)
```

Arguments

x	an object of a 'Pacnet' class
po.incl	(optional and logical) should the partial order be included in the outcome?
vc	(optional) vector of the induced inclusions to be computed
po	(optional) the partial order structure

Details

This function process the outcome of the **Pacnet** report by adding induced inclusions to partial order, the minimal element of the lattice of congruence relations. Such type of structure serves for the decomposition of a partially ordered semigroup structure.

Value

An object of the 'Pi.rels' class

pi	the π-relations, eventually with the partial order
mca	the meet-complements of atoms

Author(s)

Antonio Rivero Ostoic

References

Pattison, Philippa E. *Algebraic Models for Social Networks*. Cambridge University Press. 1993.

See Also

pacnet, decomp, semigroup

prev *Preview of the Semigroup Construction*

Description

A function to preview the partial right multiplication table of the semigroup to assess the size of the complete semigroup.

Usage

```
prev(x)
```

Arguments

x	an array; usually with three dimensions of stacked matrices where the multiple relations are placed.

Details

When the input data is large, i.e. having a dozen or more elements and/or more than five dimensions, it is recommended to perform this function before the semigroup construction to get the partial right multiplication table.

That is because the amount of undefined data in such a table gives an idea of how much time may take to get the complete semigroup. However, the performance depends mainly on whether the generator matrices are sparse and/or have a relatively large number of elements for a semigroup construction of the course.

Value

'2stpT'	a partial right multiplication table at two-step.
'PcU2stpT'	the proportion of undefined elements at two-step.
ordr	the dimension of the right multiplication table so far.
Note	a conditional warning message.

Note

The warning message is given only if the percentage of undefined elements and the dimension of the input data are relatively high; however, the semigroup construction can still take a long time without the message.

Author(s)

Antonio Rivero Ostoic

See Also

semigroup, edgeT.

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.5, 1 ) )

# preview a "semigroup" structure
prev(arr)
```

---

rbox *Construct the Relation-Box*

Description

Function to construct the Relation-Box of a multiple network

Usage

```
rbox(w, transp = FALSE, smpl = FALSE, k = 3, tlbs)
```

Arguments

w	an array with three dimensions of stacked matrices of generating relations.
transp	(logical) whether or not the transpose of each matrix in w should be included.
smpl	(logical) whether to simplify or not the strings of relations
k	length of the Relation-Box in z
tlbs	(optional) a vector with the labels for the transpose relations.

Details

If transp = TRUE the labels of the transpose are toggle case of the labels of the original matrices, and in such case, it is advised to simplify the strings of relations. To prevent a transposed structure for a certain array of w, use NA in the vector the transpose labels tlbs corresponding to the respective matrix.

Value

An object of the 'Rel.Box' class.

w	the primitive relations in the Relation-Box
W	the structure of the Relation-Box
lbs	the labels in the relational system
Note	(optional) Notes indicating the particularities in the input
Orels	the original labels of the relations
Srels	(optional) the simplified labels of the relations
Trels	(optional) the labels of the transposed relations
k	the maximal length of the word
z	the length of the Relation-Box in the z dimension

Note

Values of k until 9 are supported. With many types of relations, and when the order of the multiplex network is high, turning k to more than three may take a long time of computation.

Author(s)

Antonio Rivero Ostoic

References

Winship, C. and M.J. Mandel "Roles and positions: A critique and extension of the blockmodelling approach," *Sociological Methodology*, 314-344. 1983.

See Also

cph, semigroup, hierar

Examples

```
# load the data
data("incubA")

# relation box of image matrices in dataset
## Not run:
rbox(incubA$IM)

## End(Not run)
```

---

read.dl *Read* dl *Files*

Description

A function to read files with the Ucinet dl format.

Usage

```
read.dl(file)
```

Arguments

file	character vector containing a file name or path of the data representing the network

Details

Files dl serve to represent multiple network structures, and it is one of the formats used in **NetDraw**, which is a component of the **Ucinet** program. Besides multiple networks, the function can read two-mode structures as well.

Value

a data frame for two-mode networks, or an array representing the multiple networks with one set of actors.

Note

The 'EDGELIST' option in DL is not yet supported for reading.

Author(s)

Antonio Rivero Ostoic

References

Borgatti, S.P., **NetDraw** *Software for Network Visualization*. Analytic Technologies. 2002.

Borgatti, S.P., Everett, M.G. and Freeman, L.C. **Ucinet for Windows**: *Software for Social Network Analysis*. Analytic Technologies. 2002.

See Also

write.dl, edgel, read.gml

---

read.gml *Read* gml *Files*

Description

A function to read files with the gml format.

Usage

```
read.gml(file, as = c("edgel", "array"), directed = TRUE, coords = FALSE)
```

Arguments

file	character vector containing a file name or path
as	should the data be given as • edgel for edge list with send/receive/ties format • array for two- or three-dimensional array
directed	(logical) whether the graph is directed or undirected.
coords	(logical) whether the coordenates in the gml file should be included.

Details

The gml format, an acronym for *graph modelling language*, provides capabilities to represent multiple networks and add arguments both to the nodes and the edges for visualization purposes.

For the multiplexity in the ties the gml file distinguishes "graphics" arguments inside "edge". Both "style" and "fill" are supported here and the former has priority over the latter in case the two are given; otherwise when these arguments are absent, the function separates up to a couple of relational levels when several pairwise ties are specified.

Value

Depending the option chosen, the output is either a data frame or an array representing the multigraph. If the coordenates are chosen then these are part of the obejct structure, but they are not visible.

Note

Node attributes can also be retrieved when the coordinates are chosen.

Author(s)

Antonio Rivero Ostoic

References

visone: *Software for the analysis and visualization of social networks*. http://visone.info

See Also

write.gml, edgel, read.dl

---

reduc *Reduce Matrices or Arrays*

Description

Function to reduce a matrix or an array with a given clustering vector.

Usage

```
reduc(x, clu, lbs = NULL, slbs = NULL, valued, row, col)
```

Arguments

x	a matrix or a three-dimensional array to be reduced
clu	a vector with the class membership
lbs	(optional) the labels to be used in the reduction
slbs	(optional) the string labels to be used in the reduction
valued	(logical) whether the reduction should preserve valued data?
row	(optional) the reduction by rows
col	(optional) the reduction by columns

Details

Given a partition, this function serves to reduce either a matrix representing e.g. a partial order structure. However, the reduction is also generalized to three-dimensional arrays representing multiple relations.

Value

The reduced matrix or a reduced three-dimensional array of the input data according to the clustering information.

Note

Use decomp for the reduction of a semigroup object.

Author(s)

Antonio Rivero Ostoic

See Also

cngr, rbox, decomp

Examples

```
# scan the multiplication table data
s <- matrix(data=c(1, 1, 1, 3, 3, 3, 3, 3, 3), nrow=3, ncol=3, byrow=TRUE)

# reduce the multiplication table
s |> reduc(clu=c(1,2,2))
```

`rel.sys` *Relational System*

Description

Create the Relation System of a multiplex network.

Usage

```
rel.sys(x, type = c("tolist", "toarray"), bonds = c("entire", "strong", "weak",
        "asym", "recp", "txch", "tent", "mixd", "full"), loops = FALSE,
        sel = NULL, att = NULL, sep)
```

Arguments

`x`	an array; usually with three dimensions of stacked matrices where the multiple relations are placed.
`type`	if the transformation is from • `tolist` for (array of) matrices into lists of pairwise relations • `toarray` for lists of pairwise relations into (array of) matrices
`bonds`	the type of bonds to be used in the creation of the relational system • `entire` for the "entire" network (default, same as `full`) • `strong` for strong bonds • `weak` for weak bonds • `asym` for asymmetric ties • `recp` for reciprocal ties • `txch` for tie exchange bundles • `tent` for tie entrainment bundles • `mixd` for mixed bundles • `full` for the whole network (same as `entire`)
`loops`	(logical) whether or not the loops should be considered in the relational system
`sel`	(optional) a set of actors to be selected. For `"toarray"` `att` and `noatt` also supported (see details)
`att`	(optional) arrays in `x` corresponding to attributes
`sep`	(optional) pair separator used for the pairwise relations

Details

When the type of bonds chosen is `entire` then the nodes with ties are considered in the relational system without isolated nodes. `strong` bonds are relational bundles with a mutual character, whereas `weak` bonds are those patterns exclusively without mutual character.

When choosing from a list with actor attributes, it is also possible to select the network members having or *not* having the attribute that is specified in the `Attrs` output by using in argument `sel`.

Value

For `type = "tolist"` (default) option, an object of '`Rel.System`' class where items are:

`ord`	order of network relational system
`nodes`	nodes in relational system
`sel`	selected set of actors
`sys.ord`	order of relational system with chosen bond type
`incl`	nodes included relational system with chosen bond type
`excl`	nodes excluded relational system with chosen bond type
`bond.type`	type of bonds used in relational system creation
`size`	number of ties in relational system
`Note`	(if needed) a note
`sep`	pairwise separator of relational system
`Ties`	ties in relational system
`Attrs.ord`	if `att` is not `NULL`, number of nodes with chosen attribute(s)
`Attrs`	if `att` is not `NULL`, actors with chosen attribute(s)

For `type = "toarray"`, the output is a two or three dimensional dichotomous array recording the relations among the actors in the network.

Author(s)

Antonio Rivero Ostoic

References

Ostoic, J.A.R. "Creating context for social influence processes in multiplex networks." *Network Science*, 5(1), 1-29.

See Also

`expos`, `bundles`, `neighb`

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array( runif(18), c(3 ,3, 2) ), array( runif(18),
       c(3, 3, 2) ) > .9, 3 ) )

# establish the system of strong bonds
rel.sys(arr, bonds = "strong")

# first array is for attributes
rel.sys(arr, att = 1)

# select the first node
rel.sys(arr, sel = 1)
```

`rm.isol` *Remove Isolates*

Description

Function to remove isolate nodes in simple and multiple networks.

Usage

```
rm.isol(x, diag, diag.incl)
```

Arguments

`x`	a matrix or array representing a network
`diag`	(optional and logical) for arrays, should diagonals be included in transformation?
`diag.incl`	(optional and logical) for arrays, should diagonals be included in the output?

Details

Isolated nodes do not have any edges in the network, and in a multivariate system, there is no edges adjacent to these kinds of nodes at any level.

Value

The matrix or array representing a multiple network without the isolated actors.

Author(s)

Antonio Rivero Ostoic

See Also

`edgel`, `zbind`

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array( runif(18), c(3 ,3, 2) ), array( runif(18),
         c(3, 3, 2) ) > .5, 3 ) )

# remove isolates (if exist)
rm.isol(arr)
```

`semigroup` *Constructing the Semigroup of Relations of Multiple Networks*

Description

Function to create the complete semigroup of multiple relations, where the multiplication table can be specified with either a numerical or a symbolic form.

Usage

```
semigroup(x, type = c("numerical", "symbolic"), cmps, smpl, valued)
```

Arguments

`x`	an array; usually with three dimensions of stacked matrices where the multiple relations are placed
`type`	semigroup multiplication table to be returned • `numerical` for a numerical format (default) • `symbolic` for a symbolic format
`cmps`	(optional and logical) whether the composite matrices should be also given in the output
`smpl`	(optional and logical) whether to simplify or not the strings of relations
`valued`	(logical) whether the semigroup should be with a valued format

Details

A multiple relation can be defined by square matrices of `0`s and `1`s indicating the presence and absence of ties among a set of actors. If there is more than one relation type, the matrices must preserve the label ordering of its elements and stacked into an object array in order to be effectively applied to this function.

The semigroup, which is an algebraic structure having a set with an associative operation on it, is calculated considering binary matrices only. This means that if the provided matrices are valued, the function will dichotomise the input data automatically. Values higher or equal to a unit are converted to one; otherwise they are set to zero. If not happy, use function `dichot` to specify a cutoff value for the dichotomization.

Semigroup structures for valued relations apply the max min operation in the composition of generators and strings.

Value

An object of '`Semigroup`' class. The items included are:

`gens`	array with generator relations
`cmps`	array with the unique compound relations
`ord`	dimension of the semigroup

st	vector of the unique string relations
S	multiplication matrix with semigroup of relations (see below)

If the specified type is numerical, then a matrix of semigroup values is given, otherwise the values is returned as a data frame with the strings of the semigroup.

Warning

For medium size or bigger sets (having e.g. more the 4 relation types), the semigroup construction could take a long time.

Note

It is recommendable to perform the function prev before attempting to construct the semigroup, unless the input data has few dimensions.

Author(s)

Antonio Rivero Ostoic

References

Boorman, S.A. and H.C. White, "Social Structure from Multiple Networks. II. Role Structures." *American Journal of Sociology*, 81 (6), 1384-1446. 1976.

Boyd, J.P. *Social Semigroups. A unified theory of scaling and blockmodelling as applied to social networks*. George Mason University Press. 1991.

Pattison, P.E. *Algebraic Models for Social Networks*. Cambridge University Press. 1993.

See Also

green.rel, prev, strings, edgeT, wordT, cngr.

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.5, 1 ) )

# optional: put labels
dimnames(arr)[[3]] <- list("n", "m")

# look at the semigroup with numerical format
semigroup(arr)
```

`semiring` *Semiring Structures for Balance Theory*

Description

A function to construct semiring structures for the analysis of Structural Balance theory.

Usage

```
semiring(x, type = c("balance", "cluster"), symclos = TRUE,
         transclos = TRUE, k = 2, lbs)
```

Arguments

`x`	an object of a '`Signed`' class
`type`	• `balance` for a balance semiring (default) • `cluster` for a cluster semiring
`symclos`	(logical) apply symmetric closure?
`transclos`	(logical) apply transitive closure?
`k`	length of the cycle or the semicycle
`lbs`	(optional) labels for the semiring output

Details

Semiring structures are based on signed networks, and this function provides the capabilities to handle either the balance semiring or the cluster semiring within the Structural Balance theory.

A semiring combines two different kinds of operations with a single underlying set, and it can be seen as an abstract semigroup with identity under multiplication and a commutative monoid under addition. Semirings are useful to determine whether a given signed network is balanced or clusterable. The symmetric closure evaluates this by looking at semicycles in the system; otherwise, the evaluation is through closed paths.

Value

An object of '`Semiring`' class. The items included are:

`val`	the valences in the semiring
`s`	the original semiring structure
`Q`	the resulted semiring structure
`k`	the number of cycles or semicycles

Note

Disabling transitive closure should be made with good substantial reasons.

Author(s)

Antonio Rivero Ostoic

References

Harary, F, Z. Norman, and D. Cartwright *Structural Models: An Introduction to the Theory of Directed Graphs*. Wiley. 1965.

Doreian, P., V. Batagelj and A. Ferligoj *Generalized Blockmodeling*. Cambridge University Press. 2004.

Ostoic, J.A.R. "Creating context for social influence processes in multiplex networks." *Network Science*, 5(1), 1-29.

See Also

signed, as.signed

Examples

```
# create the data: two sets with a pair of binary relations
# among three elements
arr <- round( replace( array( runif(18), c(3 ,3, 2) ), array( runif(18),
       c(3, 3, 2) ) > .5, 3 ) )

# make the signed matrix with two types of relations
# and establish the semiring structure
signed(arr) |>
  semiring()
```

`signed` *Signed Network*

Description

Construct the signed network of a system of contrasting relations

Usage

```
signed(P, N = NULL, lbs)
```

Arguments

`P`	array with the positive ties and possible with negative ties (see 'details')
`N`	(optional) array with the negative ties
`lbs`	(optional) labels for the signed matrix

Details

This function coerces an array(s) to become a 'Signed' object. Positive ties are always in the first argument, and in case that this array has three dimensions, the second dimension is considered as the negative ties, provided that N is still NULL. If ambivalent ties are present in the structure then the signed matrix represent positive, negative, ambivalent, and null ties as p, n, a, and o respectively; otherwise, the values are 1, -1, and 0.

Value

An object of 'Signed' class with items:

val	valences in the signed matrix
s	signed matrix

Note

A warning message is shown when the N argument has more than two dimensions.

Author(s)

Antonio Rivero Ostoic

References

Doreian, P., V. Batagelj and A. Ferligoj *Generalized Blockmodeling*. Cambridge University Press. 2004.

See Also

semiring, as.signed

Examples

```
# load the data
data("incubA")

# make the signed matrix with two types of relations
signed(incubA$IM)
```

---

strings *Strings of Relations*

Description

Function to get the labels of the unique relations of the semigroup, or the generators and compound relations that are the elements of the complete semigroup.

Usage

```
strings(x, equat = FALSE, k = 2, smpl, valued)
```

Arguments

x	an array; usually with three dimensions of stacked matrices where the multiple relations are placed.
equat	(logical) should the equations be included in the output?
k	length of the strings in the equations
smpl	(optional and logical) whether to simplify or not the string relations
valued	(logical) whether the strings are with a valued format

Details

The strings are the unique relations, which constitute the elements of the complete semigroup. These are both the generators and the compound relations after applying the Axiom of Quality, which means that even some generators can be disregarded.

This function is especially useful to construct the partial order of relations and to establish the set of equations in the relational structure.

Value

An object of 'Strings' class.

wt	the generators and compound relations
ord	the order of the structure
st	the labels of the unique relations
equat	the equations among strings of relations

Note

The maximum length of the strings in the equations is currently 4.

Author(s)

Antonio Rivero Ostoic

References

Boorman, S.A. and H.C. White, "Social Structure from Multiple Networks. II. Role Structures." *American Journal of Sociology*, 81 (6), 1384-1446. 1976.

See Also

partial.order, semigroup.

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.5, 1 ) )

# get the strings
strings(arr)
```

`summaryBundles` *Summary of Bundle Classes*

Description

Pretty printing of the bundle class patterns results.

Usage

```
summaryBundles(x, file = NULL, latex = FALSE, byties)
```

Arguments

`x`	an object of the 'Rel.Bundles' class
`file`	(optional) a path where the output file is to be placed
`latex`	(logical) whether or not the output should be in latex format
`byties`	(optional and logical) expand tie patterns and collapse tie labels?

Details

This function prints the bundle census patterns existing in the network with an option to export such information in a friendly format. The dyadic bundle patterns are provided by the function `bundles`; however, the outcome of this function provides a list of pair lists for each bundle with the involved types of relations and nodes in the network. This form for presentation, although is convenient for further computation, it is not always easy to read for the human eye. The pair separator used to print the bundle occurrences is taken from the output of the `bundles` function.

If `latex` is set to `TRUE`, then the path file is activated to obtain a `tex` file with the different bundle class patterns. Finally, the optional argument `byties` provide more precise information about the patterned ties disregarding the relational content.

Value

The distinct bundle class patterns with a user friendly format.

Note

If a file with the same name already exists in the pointed directory, then this file will be overwritten.

Author(s)

Antonio Rivero Ostoic

References

Ostoic, J. A. R. "Dyadic Patterns in Multiple Networks," *Advances in Social Networks Analysis and Mining, International Conference on*, 475-481. 2011.

See Also

bundles, bundle.census

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array( runif(18), c(3 ,3, 2) ), array( runif(18),
c(3, 3, 2) ) > .8, 3 ) )

# print different relational bundles in 'arr'
arr |>
  bundles() |>
  summaryBundles()
```

---

transf *Transform Data from/to Matrix/List Formats*

Description

Function to transform data from/to matrix/list formats or edge list representing a network.

Usage

```
transf(x, type = c("toarray", "tolist", "toarray2", "toedgel"), lbs = NULL, lb2lb,
    sep, ord, sort, sym, add, adc, na.rm)
```

Arguments

x	an array or a list of pair relations
type	type of transformation: • toarray from a list of pair relations to an array format • tolist from a matrix to a list of pair relations • toarray2 from a list of pair relations to a square array • toedgel from arrays to edge list
lbs	(optional) the labels in the transformation
lb2lb	(optional and logical) whether the transformation is *label-to-label*. Default TRUE for "toarray" and FALSE for "tolist"

sep	(optional) pair separator used for the pairwise relations
ord	(optional) for "toarray", the order of the resulted structure
sort	(optional and logical) sort the arrays in the output?
sym	(optional and logical) for "toarray", symmetrize the arrays?
add	(optional) added elements in the array's *domain*
adc	(optional) added elements in the array's *codomain*
na.rm	(optional) remove missing data in NA?

Details

Option "tolist" is for transforming a matrix or an array to a list of pair elements. In case that the lb2lb is enabled in this type of transformation, then lbs must be provided, whereas the pair separator is optional. On the other hand, "toarray" will produce a matrix from a list of pair elements, and in this case is advisable to specify the order of the structure. Three dimensional structures are supported in the transformations with all options.

Data frames are also accepted for the "tolist" option; however, in case that this information is given as a list of pair relations the output will be a square matrix.

When the transformation option is "edgel", the output is a data frame with the first two columns for the sending and receiving ties. For simple networks, these two columns are enough and for multiplex networks additional columns are for the types of tie, one for each (cf. function edgel).

Value

Depending on the input data, the result is either a list of pair relations or a matrix of relations.

Note

For high dimensional arrays, the rel.sys function provides additional information other than the list of pair relations of the entire structure.

Author(s)

Antonio Rivero Ostoic

See Also

edgel, bundles, reduc, rel.sys

Examples

```
# scan the multiplication table data
s <- matrix(data=c(1, 1, 1, 3, 3, 3, 3, 3, 3), nrow=3, ncol=3, byrow=TRUE)

# transform the matrix to a list format
s |> transf(lb2lb = TRUE, lbs = c("n","m","u"))
```

wordT *Word Table of Relations*

Description

Function to produce the *Word Table* of multiple relations as representation form of a semigroup of relations.

Usage

```
wordT(x)
```

Arguments

x	an array; usually with three dimensions of stacked matrices where the multiple relations are placed.

Details

The Word Table is a consequence of the Edge Table and the function gives a list of indexed elements in the complete semigroup.

In terms of the Cayley graph of the semigroup (cf. ccgraph, the collection of unique relations (both compound and generators) are represented by nodes. On the other hand, the generators are edges that record the result of post-multiplying the compound relations by the generators.

The labels for the elements can be retrieved by the strings function.

Value

An object of the 'WordTable' class

gens	the generator relations
WT	the Word Table where "n" stands for *node* and "g" stands for *generator*

The generators do not have values in neither the "node" nor the "generator" of the Word table since they are not a product of any other element in the semigroup (cf. 'details' for the rest of the values).

Author(s)

Antonio Rivero Ostoic

References

Cannon, J.J. "Computing the ideal structure of finite semigroup," *Numerische Mathematik*, 18, 254-266. 1971.

Pattison, P.E. *Algebraic Models for Social Networks*. Cambridge University Press. 1993.

See Also

edgeT, semigroup, strings, ccgraph.

Examples

```
# create the data: two binary relations among three elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.5, 1 ) )

# obtain word table
wordT(arr)
```

---

write.dat *Write* dat *Files*

Description

A function to write dat files.

Usage

```
write.dat(x, path)
```

Arguments

x	an object representing a multiple network structure
path	path file for the output

Details

"dat" files are the format used in the **Pacnet** program inside **StOCNET**. In case that the input data represents a multiple network then a separate file will be produced, each file representing a single type of relationship in the system. The name of the output files depends on the object title.

Value

File(s) output with adjacency matrices with a .dat format.

Note

In case that the directory in the path for the output does not exist then a folder will be created automatically.

Author(s)

Antonio Rivero Ostoic

References

StOCNET *An open software system for the advanced statistical analysis of social networks*.

See Also

pacnet, write.gml, write.dl

---

write.dl *Write* dl *Files*

Description

A function to write dl files representing multiple networks.

Usage

```
write.dl(x, file = NULL, type = c("nodelist", "fullmat"))
```

Arguments

x	an object representing the multiple network
file	path to the file
type	write data with format type: • nodelist for *node-list* format • fullmat for *fullmat* format

Details

dl files serve to represent multiple networks, and it is one of the formats used in **NetDraw**, which is a component of the **Ucinet** program.

Value

A file with the data with a .dl format

Author(s)

Antonio Rivero Ostoic

References

Borgatti, S.P., **NetDraw** *Software for Network Visualization*. Analytic Technologies. 2002.

Borgatti, S.P., Everett, M.G. and Freeman, L.C. **Ucinet for Windows**: *Software for Social Network Analysis*. Analytic Technologies. 2002.

See Also

read.dl, write.gml, write.srt, write.dat

`write.edgel` *Write edge list files*

Description

A function to write edge list files having columns for sender, receiver, and the ties for multiplex networks.

Usage

```
write.edgel(x, file = NULL, sep = "\t", header = TRUE)
```

Arguments

`x`	an object representing the multiple network
`file`	path to the file
`sep`	the separator used between the columns
`header`	(logical) whether the header should be included in the file

Details

Write edge list files with a *s*end, *r*eceive, and *t*ies, which is a data frame with at least two columns for the sender and receiver, and the different types of tie for multiplex networks, one column for each type of relation.

Value

A file with the edge list format

Author(s)

Antonio Rivero Ostoic

See Also

`edgel`, , `write.dl`

write.gml *Write* gml *Files*

Description

A function to write files with a gml format.

Usage

```
write.gml(x, file = NULL)
```

Arguments

x	an object representing the multiple network
file	path to the file

Details

The gml format, an acronym for *graph modelling language*, provides capabilities to represent multiple networks and add arguments to both the nodes and the edges for visualization purposes.

Value

A file with the data with a graph modelling language format.

Note

In case that the file already exists in the pointed directory, then the file will be overwritten.

Author(s)

Antonio Rivero Ostoic

References

visone *Software for the analysis and visualization of social networks*. http://visone.info

See Also

read.gml, write.dl, write.dat

zbind *Bind Matrices and Multidimensional Arrays*

Description

A function to combine or bind matrices and multidimensional arrays.

Usage

```
zbind(..., sort, force)
```

Arguments

`...`	One or more arrays having two or three dimensions
`sort`	(optional and logical) sort array according to labels?
`force`	(optional and logical) force binding arrays with different dimensions?

Details

Function `zbind` is for stacking for instance two-dimensional arrays into a single three-dimensional object to represent a multivariate system structure. Both square and rectangular arrays are supported provided that the dimensions in the input are equal, and data frames should be transformed into arrays for the binding. The `dimnames` in the array output correspond to the names of the first array in the input, and a `warning` message is given when the dimnames are `NULL`.

Value

Usually a three dimensional array

Warning

Arrays without dimnames are not fully supported.

Note

Argument `force` must be set to `TRUE` for matrices and arrays with different dimensions, and a `message` is given.

Author(s)

Antonio Rivero Ostoic

See Also

`mnplx`, `dichot`, `strings`

Examples

```
# create the data: two sets with a pair of binary relations
# among three elements
arr1 <- round( replace( array( runif(18), c(3 ,3, 2) ), array( runif(18),
        c(3, 3, 2) ) > .5, 3 ) )

arr2 <- round( replace( array( runif(18), c(3 ,3, 2) ), array( runif(18),
        c(3, 3, 2) ) > .5, 3 ) )

# bind the data sets
zbind(arr1, arr2)
```

Index