Package 'multiplex'

July 30, 2026

Type Package

Title Algebraic Tools for the Analysis of Multiple Social Networks

Version 4.0

Depends R (>= 4.6.0)

Imports methods

Suggests multigraph, Rgraphviz, knitr

Description Algebraic procedures for analyses of multiple social networks are provided with this package as described in Ostoic (2020) <DOI:10.18637/jss.v092.i11>. 'multiplex' supports the creation and analysis of multiplex, multimode, and multilevel network data in various formats. It combines methods based on partially ordered semigroups, decomposition procedures, semiring structures, and relational bundles for multivariate network analysis. It also provides an algebraic approach to affiliation networks using Galois derivations, with visualization options.

Date 2026-07-29

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 2026-07-30 09:30:02 UTC

Contents

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

Description

Algebraic procedures for analyses of multiple social networks are provided with this package as described in Ostoic (2020) <DOI:10.18637/jss.v092.i11>. 'multiplex' supports the creation and analysis of multiplex, multimode, and multilevel network data in various formats. It combines methods based on partially ordered semigroups, decomposition procedures, semiring structures, and relational bundles for multivariate network analysis. It also provides an algebraic approach to affiliation networks using Galois derivations, with visualization options.

Details

Package:	multiplex
Type:	Package
Version:	4.0
Date:	29 July 2026
License:	GPL-3

One aim of the **multiplex** package is to provide an analytical tool specifically designed for social networks with relationships at multiple levels. It includes functions for modelling local role algebras based on the simple and compound relations present in the network. **multiplex** also supports the construction and analysis of signed networks through semiring structures. In addition, it can identify dyadic relational patterns that may support further analysis using a range of structural theories. The package can also incorporate actor attributes into the analysis of multiple networks in different ways. For example, it can assess the network exposure of actors in a multiplex context or embed attribute information directly into the resulting algebraic structures.

A typical workflow starts with function `semigroup`, which constructs a semigroup from an array of multiple binary relations. The resulting structure can be described with functions `wordT` and `edgeT`. Unique relations and associated equations are returned by `strings` function together with the set of equations with strings of length k. Ordering among string elements is defined by function `partial.order`, and the corresponding lattice can be displayed with functions `hasse` or `diagram`.

Further semigroup analysis is available through function `green.rel` for equivalence classes and `diagram` function again for "egg-box" visualization. Structure reduction of semigroup objects is supported through decomposition based on congruence relations or π-relations using functions `pi.rels`, `cngr`, and `decomp`.

Signed networks can be created with function `signed` and analysed with function `semiring` for balance or cluster semiring structures.

General network analysis utilities in **multiplex** include function `dichot` for dichotomization of matrices and arrays, `rm.isol` for removing isolates, `perm` for permutation operation, and function `zbind` for building three-dimensional arrays.

The package **multiplex** also provides function `rbox` for Relation-Box construction and function `cph` for Cumulated Person Hierarchy in the network. Bundle analysis is supported by functions

bundles, transf, bundle.census, and summaryBundles. Network exposure and relational-system analysis are available through functions rel.sys and expos, while galois supports algebraic analysis of two-mode networks. Multivariate network data can be imported from send-receive-ties edge lists with function edgel and from formats such as **UCINET** dl and **Visone** gml. Multiple network structures can be visualized further with the related package **multigraph**.

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 a three-dimensional array
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.5, 3 ) )

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

# construct the semigroup structure and look at its Green relations
semigroup(arr) |>
  green.rel()

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

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

Description

A generic function for coercing a relational system into a "`Semigroup`" class object.

Usage

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

Arguments

`x`	An array representing the semigroup.
`gens`	Aarray 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 the input in '`x`' an edge table?

Details

Use this function to convert an array representing a semigroup structure into an R object of class "`Semigroup`" as some routines in **multiplex** require.

Value

A list object of class "`Semigroup`" that is either '`symbolic`' or '`numerical`'.

`ord`	an integer with the dimension of the semigroup
`st`	a vector of the semigroup unique relations or the string-relations
`gens`	a vector with the generators of the semigroup
`S`	multiplication table with the semigroup structure.

Class attribute, and a `warning` message when no generators are provided.

See Also

`semigroup`, `green.rel`

Examples

```
# create a 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 object
attr(s, "dimnames") <- list(1:3, 1:3)

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

`as.signed` *Coerce into a Signed Object*

Description

A generic function for coercing a matrix into a "`Signed`" class object.

Usage

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

Arguments

`x`	A matrix representing the signed network.
`lbs`	(optional) Labels for the signed matrix.

Details

Use this function to convert an array representing a signed network into an R object of class "`Signed`", as required by function `semiring`.

The array in `x` itself can be also a "`Semiring`" class object represented as '`Rel.Q`' attribute for the semiring relational structure.

Value

The input array in `x` as a "`Signed`" class.

See Also

`signed`, `semiring`

Examples

```
# load a network dataset
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 Object into a Strings Class*

Description

A generic function for coercing an R object into a "Strings" class.

Usage

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

Arguments

x	An array, usually with three dimensions of stacked matrices where the multiple relations are placed.
lbs	(optional) Labels of strings.

Details

Use this function to prepare an object for establishing the partial order of relation strings by converting it to class "Strings". It returns as well the number of unique relations in the semigroup represented by the dimensions in x.

Value

The input array in x as an object of "Strings" class

wt	word tables
ord	dimension in x

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 of Multiple Networks*

Description

A function to perform the bundle census in multiple networks.

Usage

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

Arguments

`x`	An array representing multiple relations.
`loops`	(logical) Also include loops in the census?

Details

A *bundle* is a particular type of pattern made of relations at different levels that is binding a pair of nodes or actors, and this function calculates the number of occurrences for each bundle class pattern in a given multiple network in `x`. The different types of relations in the network are typically recorded in a three dimensional array of stacked matrices, and edge-list and pairwise-list of relations need to be transformed first into arrays with for example function `transf`.

Depending on the direction and occurrence of each possible tie, then it is possible to count with seven dyadic configuration classes in the census given in the output where fFunctions `bundles` and `summaryBundles` provide bundle class occurrences in the network with a more detailed information.

Value

A table with the number occurrences in the distinct bundle class patterns occuring in `x`:

- BUNDLES Total number of bundles in `x`, excluding the *Null* pattern.
- NULL *Null*
- ASYMM *Asymmetric*
- RECIP *Reciprocal*
- T.ENTR *Tie Entrainment*
- T.EXCH *Tie Exchange*
- MIXED *Mixed* pattern
- FULL *Full*
- LOOP *Loops* (if `TRUE`)

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, transf

Examples

```
# create 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 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 representing multiple relations.
loops	(logical) Also include loops pattern?
smpl	(logical) Simplify the strings of relations?
lb2lb	(logical) Should labels of the nodes be included in the output?
collapse	(logical) Collapse the distinct levels of relations in the network?
sep	(optional) Separator 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.

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.

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)

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 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 ) )

# look at bundle patterns
bundles(arr)
```

---

cngr *Congruence Relations*

Description

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

Usage

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

Arguments

S	A "Semigroup" class object.
PO	(optional) A partial order table.
uniq	(optional and logical) Return unique congruence relations?

Details

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

In case that the partial order is supplied in `PO` the computation of the congruence classes is slightly faster than for an abstract semigroup.

Value

A "`Congruence`" class list object with:

`S`	Semigroup of relations
`PO`	Partial order table (if specified)
`clu`	Congruence classes

References

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

See Also

`decomp`, `fact`, `pacnet`

Examples

```
# create 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 ) )

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

# look at the congruences in S
cngr(S)
```

`comps` *Find Components in Multiple Networks*

Description

Function to find different components and isolated nodes in the multiple networks.

Usage

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

Arguments

x	An array representing a given network.
bonds	Type of bonds for components creation: • "entire": all bundle patterns • "strong": mutual bundles (reciprocal, exchange, and mixed) • "weak": asymmetrical bundles (and entraiment)
sort	(optional and logical, default FALSE) Sort components by size?

Details

Components and isolates in network x are derived from the transitive closure of bundle ties. By default, the computation takes the *entire* system, but the bonds option lets the selection of other types of relational bundles for the resulted components.

The sort argument outputs components length in ascending size, such as dyads, triads, and larger groups when present. Changing the order of the existing components may have consequences in the layout of the graph when plotting it with function multigraph.

Value

A list object of length two with the bonds type in the class attribute:

com	List or vector with network components
isol	Vector with network isolates

See Also

bundles, summaryBundles, bundle.census, rel.sys

Examples

```
# create 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 ) )

# 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) Labels of the relational system.

Details

Uses the cumulated person hierarchy to determine *partial structural equivalence* among actors in a multiple network. Two nodes are considered partially structurally equivalent only if they have identical role sets. Output is determined by the properties of the Relation-Box product of function rbox, and in case that the length of the labels differs from the order of the relational system, then labels will be ignored.

Value

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

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 a dataset
data("incubA")

# construct the Relation Box of the dataset image matrices and
# compute its cumulated person hierarchy
incubA |> getElement("IM") |>
  rbox() |> cph()
```

---

cscl *Conceptual Scaling for Many-valued Formal Contexts*

Description

A function to interpret a given many-valued formal context through a chosen scaling process.

Usage

```
cscl(x, cl, scale = c("nominal", "ordinal", "intord", "biord", "dichot"), sep)
```

Arguments

x	A table or array representing a many-valued formal context.
cl	A vector with the ordered conceptual scaling-levels in x.
scale	Type of *scaling* in the interpretation of x: • nominal. • ordinal. • intord for one-dimensional interordinal. • biord for biordinal. • dichot for dichotomic.
sep	(optional, default dot) Pair separator among attributes and levels.

Details

Conceptual scaling is a special form of formal context in which each attribute of a many-valued context is interpreted through a scaling procedure. More broadly, it converts many-valued data into dichotomous data, producing a derived context according to the selected type of scaling. Conceptual scaling is therefore an interpretive method where both the choice of scale and the transformation of values are form part of the analysis, rather than following from a mathematically determined procedure.

The formal context represented in x must be associated to its conceptual scaling-levels recorded in cl as an ordered vector with all levels, and a type of scaling derives the context into a dichotomous data frame with attribute ordinal values. Once the partial order of the Galois derivation among the objects and attributes of the formal concept has been established, the resulting derived context with dichotomous data can be represented as a concept lattice.

Value

A data frame with a "conceptual.scaling" class representing a derived many-valued formal context with respect to the conceptual scaling-levels in cl. The ordered conceptual levels and the pair separator are given in the class attribute.

References

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

See Also

galois, partial.order, diagram.

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)
```

Arguments

S	A "Semigroup" class object.
pr	Either an object of a "Congruence" class or an object of a "Pi.rels" class.
type	Type of decomposition on S where 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) Should the returned list include the reduced structures?
fac	(optional) Factor subject to decomposition.

Details

The decomp function reduces an algebraic structure, such as a semigroup, to determine which class members in the system are congruent. The resulting object consists of congruent elements that belong to the lattice of congruence classes in the structure. If the input data comes from the Pacnet program, these elements take the form of π-relations or meet-complements of atoms; otherwise, they are equivalent elements that satisfy the substitution property.

A "Semigroup" class object may contain unavailable values in its multiplication table, typically when it is produced as an image by the fact function. In such cases, the semigroup can be reduced with the force option, which adds equations to the string relations to eliminate NA values in the semigroup data.

Use the reduc function to reduce the partial order table.

Value

A list object of "Decomp" class having:

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

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, shape = c("rectangle", "circle", "box", "ellipse"), col, col0,
        fcol, ecol, lty, lbs, ffamily, fstyle, fsize, cex.main, col.main, bg,
        mar, sep, ...)
```

Arguments

x	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" type diagram, the following arguments are ignored:

attrs	(optional) a list of lists with diagram attributes for node, edge, and graph.
main	(optional) Main title of the plot diagram.
incmp	(optional and logical) Should incomparable elements be included in the lattice diagram?
shape	(optional) Shape of diagram vertices: "rectangle", "circle", "box", "ellipse".
col	(optional) Colourr of diagram vertices.
col0	(optional) Colour of contour in diagram vertices.
fcol	(optional) Font colour of text in diagram vertices.
ecol	(optional) Edge colour of diagram edges.
lty	(optional) Integer for the shape of diagram edges.

lbs	(optional) Labels of elements in x partially ordered set.
ffamily	(optional) Font family of vertex labels in diagram.
fstyle	(optional) Font style of vertex labels in diagram with options: bold, italic, bolditalic.
fsize	(optional) Font size of vertex labels in diagram.
cex.main	(optional) Font size of title in plot diagram.
col.main	(optional) Font colour of title in plot diagram.
bg	(optional) Background colour of plot diagram.
mar	(optional) Margins of plot vindue.
sep	(optional only for concept) Pair separator for pairwise relations inside intents and extents.
...	(optional) Additional graphical items.

Details

diagram is a wrapper function for plotting and manipulating "Hasse", "Concept", and "Egg-box" diagrams.

The first two diagram types represent systems of ordered relations and are displayed as partial or linear order diagrams. For example, the partial order table produced by the 'strings' option in the partial.order function can be plotted as a Hasse diagram. Another example comes from Galois derivations in Formal Concept Analysis, where a Concept diagram shows the ordering of formal concepts defined by intents and extents. Roman numerals are rendered for diagram elements when the partial order table in x is not labelled.

On the other hand, the Egg-box "diagram" represents equivalence classes in an abstract semigroup without a partial order structure, and such representation is in the Console, not on the Graphic Device as the other representations.

Value

Depending on the type diagram:

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

Note

This function requires **Rgraphviz** package installed.

See Also

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

Examples

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

# a partial order of strings from dataset image matrix
po <- incubA |> getElement("IM") |>
  as.strings() |>
  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 to represent a textual interpretation of the levels in a given lattice diagram.

Usage

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

Arguments

`x`	A matrix representing partial order relations.
`perm`	optional and logical) Return automorphism of a permuted structure?

Details

This function provides a textual interpretation of the different levels in the lattice diagram of the partial order structure among actors and ties in the network.

When reducing a multiple network, the partial order structure may define different classes of elements based on their inclusion relations. The illustration produced by the `diagram` function typically shows these levels of ordered relations, which this routine then interprets.

Value

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

If `perm` is specified, the function returns a data frame containing the elements of the partial order structure, with column names indicating the element classes. It also returns a vector of levels and a matrix of the permuted structure.

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.

See Also

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

Examples

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

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

# 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`	An array, matrix or other data in a numeric form.
`c`	A cutoff value to perform the dichotomization in `x`.
`diag`	(optional and logical, default `TRUE`) Should diagonals be included in output?

Details

This is a wrapper of the `replace` base function where in this case is aimed to specify a cutoff value for the dichotomization of the data. Values equal or higher to the cutoff are hence converted to `1`, while the others are set to `0`. Notice that the cutoff value in `c` can be any real number, and labels are preserved after the dichotomization..

Value

Binary values of the input data with the same format as x.

See Also

replace, mnplx, semigroup.

Examples

```
# create 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 and Transform Edge-Lists*

Description

A function to read edge lists as objects or files with a 'sender-receiver-tie' format representing multiplex networks and to transform edge-lists into three-dimensional arrays.

Usage

```
edgel(x, toarray = TRUE, header = TRUE, sep = "\t", valued = FALSE, inc = FALSE,
      rownames = TRUE, add = NULL, na.rm)
```

Arguments

x	Object or file representing network data.
toarray	(logical) Transform edge-list in x into an array?
header	(optional and logical) Does edge-list in file x has a header?
sep	(default horizontal tab) Separator between edge-list columns.
valued	(logical, for toarray) Return a transformation of x as a valued network?
inc	(optional and logical) Is x an incidence matrix?
rownames	(optional and logical, for inc) Treat 'rownames' in x as actors/objects names? See details.
add	(optional) Vector with isolate labels to add to the output.
na.rm	(optional and logical) Remove NAs from x in the output?

Details

The edgel function reads edge lists in 'sender-receiver-tie' format as a data frame with at least two columns for sender and receiver, plus one column for each relation type in a multiplex network. For simple networks, two columns for sender and receiver are sufficient. The input in x may also be a file containing tabular data, where the separator and header are specified with sep and header.

The inc option is used for *incidence* matrices with actors or objects and self-ties in data frames, which are relevant for two-mode data and analyses such as Galois transformations between partite sets. Incidence matrices can be converted into diagonal matrices for further analysis. When toarray is set to FALSE, argument rownames can be used to move labels between row names and the first column.

Additional options make it possible to remove missing values with na.rm, add isolated nodes with add, and choose between dichotomizing or preserving values in networks with valued.

Value

By default the outcome is an array that for multiple networks is a three dimensions of stacked matrices where the relation types are placed. However, if toarray is set to FALSE, then the edgelist with a data frame format is returned. warning messages for missing information, with the tranformation type, and when 'one type of relation assumed' in x.

Note

For backwards compatibility, an alias for this fuction is read.srt.

See Also

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

---

edgeT *Edge Table Generator*

Description

Produce the Edge Table of multiple relations.

Usage

```
edgeT(x, semigroupClass)
```

Arguments

x	An array, usually with three dimensions of stacked matrices where the multiple relations are placed.
semigroupClass	(optional and logical) Return a "Semigroup" class object?

Details

Function edgeT is to produce the *Edge Table* of multiple relations in a network, which is a complete right multiplication table of the semigroup having its elements for each of its generators,

Value

A list of "EdgeTable" class object.

gens	generator relations
ET	Edge Table

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 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 its 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 and logical) Should categories of adopters be included in the output?
allClasses	(optional and logical, only for classes) Include empty classes within the categories of adopters?
allNodes	(optional and logical, only for classes) Include all actors in the network regardless they are in the chosen system?

Details

This function generalises the measure of network exposure for multiple networks based on the characteristics of the selected relational system. Such a system may represent the entire network or a configuration of strong or weak ties among the actors. It is also possible to specify different behaviours of the nodes representing social actors, expressed through the relational system. The network exposure measure is calculated on the basis of the immediate neighbours of the reference actor.

Value

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

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 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 ) )

# compute the exposure measure first array in 'arr' is for attributes with adopter categories
arr |>
  rel.sys(att = 1) |>
  expos(classes = TRUE)
```

`fact` *Factorize Partially Ordered Semigroups*

Description

A function to factorize partially ordered semigroup structures by induced inclusions.

Usage

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

Arguments

`S`	A "`Semigroup`" class object.
`P`	A "`Partial.Order`" "`strings`" class object associated to `S`.
`uniq`	(logical) Should the factorization include unique induced inclusions?
`fac`	(optional) The 'factor' to be factorized for partially ordered structures.
`atoms`	(optional and logical) Short for including `"mca"` and `"atmc"` in the output.
`mca`	(optional and logical) Include pairs of meet-complements of pair of atoms in output?
`atmc`	(optional and logical) Include meet-complements of pair of atoms in output?
`patm`	(optional and logical) Include potential atoms in output?
`k`	(optional) Integer with length of induced inclusion.

Details

Factorization is a subdirect-representation-based decomposition technique for a partially ordered semigroup. This procedure decomposes the algebraic structure into simpler components, called *factors*, each of which is a semigroup factor representing a specific homomorphic image of the algebra.

For partially ordered semigroups, these factors are isotone homomorphic images corresponding to maximally independent features of the original semigroup of relations. The quotient semigroups and the reduced partial order tables together form the factorizing set, which is given by the decomposition with `decomp`.

Value

A list of "`Ind.incl`" class object 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

Class attribute, and a warning message when no substitution property found in induced inclusions.

References

Ardu, S. *ASNET – Algebraic and Statistical Network Analysis. User Manual*. University of Melbourne. 1995. (for the implementation algorithm)

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 structure.

Usage

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

Arguments

x	(integer or character) Reference element in the partial order.
PO	An array for the partial order or a "Partial.Order" class object.
ideal	(logical) Is the "filter" an order ideal?
rclos	(optional and logical) Apply reflexive closure to the outcome?

Details

This function identifies the principal order filter or principal order ideal of an element in a partial order. The input is typically either a concept, an object, or an attribute, together with the corresponding partial order of concepts derived from Galois derivations. If the reference element is a concept, specify it as a positive integer representing the concept label. Alternatively, an object or attribute may be given by its character name, provided it appears among the dimension labels of the partially labelled order table. When the reference element is an object or attribute, principal order filters with full labelling are not supported, and in that case use the corresponding concept number instead.

Value

Returns a named list containing the elements in the upset or downset of the principal order filter or order ideal associated with the reference element in the partial order.

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*

Description

Function to compute Galois derivations between partially ordered subsets.

Usage

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

Arguments

x	An incidence matrix with objects and attributes.
labeling	Type of labeling for the Galois derivations: • full • reduced
sep	(optional) Pair separator for the pairwise relations, default "comma-space".

Details

A *Galois derivation*, or Galois connection, between the power sets of G and M is defined for subsets $A \subseteq G$ and $B \subseteq M$ by the set of attributes common to all objects in A, or A', and the set of objects having all attributes in B, or B'.

An inclusion with the "full" labeling in the output implies first objects and then attributes in the Galois connection, whereas with the "reduced" option the output assumes inclusions of attributes first and then the objects.

Notice that for *many-valued* formal contexts the structure is derived from a concept scaling through function cscl where is possible to define a pair separator between attributes and their values. A warning message is produced when there is a conflict between such separators and the pair separator chosen in sep for extents and intents.

Value

A list of "Galois" class object with Galois derivations of objects and attributes as list labels:

sep	pair separator
gc	galois connections of pairs

Notice that the object attribute class only is printed for "full" labeling, which is an intrinsic part of the outcome for a "reduced" labeling.

References

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

See Also

partial.order, cscl, diagram, fltr.

Examples

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

# compute 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`	A "`Semigroup`" class object.

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. Notice that some systems have the D-class equal to S.

Value

A list object of "`Green.Rels`" and "`Semigroup`" class having:

`S`	multiplication matrix of the abstract semigroup
`gens`	generator relations of S
`dim`	dimension of semigroup generators
`ord`	order of S
`st`	unique string relations
`clu`	list of vectors with clustering information of equivalence classes R and L
`R`	R equivalence classes
`L`	L equivalence classes
`D`	D equivalence classes

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 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 Green's relations of its semigroup
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, concept, attrs = NULL, main = NULL, incmp, shape = c("rectangle",
    "circle", "box", "ellipse"), col, col0, fcol, ecol, lty,
    lbs, ffamily, fstyle, fsize, cex.main, col.main, bg, mar,
    sep, ...)
```

Arguments

x	An object of the "Partial.Order" class or a matrix representing ordered relations.
concept	(optional and logical) Plot a concept diagram?
attrs	(optional) Attributes for diagram.
main	(optional) Title of diagram.
incmp	(optional and logical) Include incomparable elements Hasse diagram?
shape	Shape of diagram vertices, otherwise "rectangle": • rectangle • circle • box • ellipse
col	(optional) Colour of diagram vertices.
col0	(optional) Colour of vertices contour.
fcol	(optional) Font colour in vertices.
ecol	(optional) Edges colour in diagram.
lty	(optional) Edges shape in diagram.

lbs	(optional) Assign labels to elements in x.
ffamily	(optional) Font family of vertex labels.
fstyle	(optional) Font style of vertex labels: • bold • italic • bolditalic
fsize	(optional) Font size of vertex labels.
cex.main	(optional) Font size of main title.
col.main	(optional) Font colour of main title.
bg	(optional) Background colour.
mar	(optional) Margins of diagram plot.
sep	(optional, for concept lattice) 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.

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

Value

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

Note

This function requires **Rgraphviz** package installed.

See Also

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

Examples

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

# a partial order of strings from dataset image matrix
po <- incubA |> getElement("IM") |>
  as.strings() |>
  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 the "`Rel.Box`" class.
`x`	(or character) Actor of reference, either by the label or by an integer with its location in array `w` (cf. `rbox`).
`type`	Type of hierarchy with respect to network in "`w`": • `person` for Person Hierarchy. • `relation` for Relation Hierarchy.

Details

The set of row inclusions in the Relation Plane of a Relation-Box `W` defines a partial order: the ego's *Relation Hierarchy*. Similarly, the ego's or person's inclusions define another partial order: the ego's *Person Hierarchy*. In other words, the Person Hierarchy captures inclusion relations among actors, whereas the Relation Hierarchy captures inclusion relations among ties. Both hierarchies are defined from the perspective of a selected reference actor in the network.

Notice that the Relation Plane is an outcome from `rbox`, whereas the cumulative person hierarchy is obtained through the `cph` function.

Value

A list with an array that represents the partial order structure either of the Person Hierarchy in `ph` or of the Relation Hierarchy in `rh`. Both include the reference ego in `pers`.

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 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 ) )

# 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 dataset*

Description

These are four data sets collected in the 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.

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 $4 \times 4 \times 7$ (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 networks are together in 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, y, type = c("bpn", "cn", "cn2"), symCdm, diag, lbs)
```

Arguments

x	Array or matrix with domain data.
y	(optional for "cn") Dataframe or list of dataframes with codomain data associated to x.
type	Type of multilevel system to construct: • bpn for a binomial projection • cn for a common membership network • cn2 for a co-affiliation of network members
symCdm	(optional and logical) Symmetrize codomains in binomial projection?
diag	(optional and logical) Include entries of the diagonal in matrices?
lbs	(optional) Either a list of length two with domain and codomain element labels or a vector with relation names in multilevel system.

Details

The inputs in x and y accept multiplex networks and an affiliation network that is high dimensional.

The default multilevel system is a binomial projection bpn that requires data for the two domains, and where the projection is on the system codomain as a directed structure. Making symmetric the codomain structure with symCdm can be convenient for the visualization of multilevel graphs as with mlgraph, and for similar practical reasons, self-relations can be removed from the multilevel structure by disabling diag.

A co-affiliation of network members with option cn2 requires both inputs as well, and the codomain structure in this case is symmetric. However, a common membership network with option cn does not need input in "x" since it returns only the co-affiliation of network members from the codomain structure in y as in cn2.

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 of length two. Option lbs as a vector can serve instead to label the relation types in the multilevel structure, which may be needed when the codomain in y is just a data frame.

Value

A list object of 'Multilevel' class with the chosen type in attribute.

mlnet	multilevel network with domain (unless cn) and projected codomain
lbs	(list) domain and codomain labels
modes	vector with either 1M for domain or 2M for codomain in mlnet

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, valued, diag, clu)
```

Arguments

net	A three-dimensional array to be transformed into a matrix.
directed	(optional and logical) Make symmetric the matrix?
valued	(optional and logical) Dichotomize the matrix?
diag	(optional and logical) Include diagonals?
clu	(optional) Vector with a cluster for permutation.

Details

This function is to collapse multiple types of tie of a network net recorded in an array into a matrix representation with monoplex relations. Other transformations on net are dichotomizing a valued network, and convert directed networks into undirected systems having or not self-relations depending on the value of diag. 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.

See Also

perm, zbind, dichot, reduc

Examples

```
# create 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 ) )

# make array monoplex
mnplx(arr)
```

---

neighb *Actor or Group Neighborhood in Relational System*

Description

A function to find with a customized distance the neighbourhood of an actor or group of actors in a relational system representing a simple or multiple network.

Usage

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

Arguments

x	A reference actor labeled in rs or a vector with a group of actors.
rs	Relational system of the network.
type	Type of system: • und for *undirected* (default) • inn for *incoming* node's ties to x • out for *outgoing* arcs from x
k	The "distance" of the neighbour nodes to x where $k = 1$ is for adjacent nodes.
inclx	(logical) Include the reference actor in the output?
expand	(optional and logical) Should the output be given by k, for $k > 1$.

Details

The relational system in rs represents either the full multiple network of actors or the subset of relational bundles that are mutual or asymmetric. Accordingly, this function identifies the nodes adjacent to 'x' within the specified relational system, as well as neighbouring nodes at a user-defined path length. Once the longest path or chain has been reached, increasing k will not add further nodes to the graph. The inn and out options apply to directed networks.

Notice that the output does not differentiate in case the chosen reference actors are in different components of the relational system.

Value

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

See Also

expos, rel.sys, bundles

Examples

```
# create 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 Log-Report 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	Path with filename of Log-Report.
toarray	(logical) Transform induced inclusions into arrays?
uniq	(logical) Consider only unique induced inclusions?
transp	(logical) Transpose partial order tables?
sep	(optional) Pair separator in 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`. Also note that only partial order structures of order 36 and less are currently supported.

Value

A 'Pacnet' class object with items:

ii	: Induced inclusions
at	: Atoms
mc	: Meet complements

References

Pattison, P, S Wasserman, G Robins and AM 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` *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, valued)
```

Arguments

`x`	A '`Strings`' or a '`Galois`' class object.
`type`	Type of partial order: • `strings` for string relations • `galois` for Galois derivations • `pi.rels` for π-relations
`lbs`	(optional) Labels of the unique relations.
`sel`	(optional) Selected elements in `x` for the partial order.
`po.incl`	(optional, only for `pi.rels`) Include the partial order in the π-relations?
`valued`	(optional and logical) Treat string relations in `x` as valued?

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 format.

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 a dataset
data("incubA")

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

---

perm *Array Permutation*

Description

Function to permutate a given array of relations.

Usage

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

Arguments

x	Matrix or array to permute.
clu	Vector of cluster for permutation.
rev	(optional and logical) Revert order in clu?
lbs	(optional) Vector of labels after permutation.
sort	(optional and logical) Sort array according to labels?

Details

This function performs a permutation of a given array or matrix representing relations according to a clustering vector of membership.

Value

A permuted matrix or array

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 produce the skeleton of a valued network with the pathfinder algorithm and triangle inequality.

Usage

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

Arguments

x	Network data, typically a valued array.
r	Distance parameter.
q	Parameter with the minimum distance between actors in the proximity matrix.

Details

Pathfinder analysis is based on a symmetric adjacency matrix or array in x representing valued networks where the values reflect the "proximity" between pairs of network members. This function depends heavily on internal function cpath where the aim is to produce the skeleton or the salient structure of x. The pathfinder structure is for undirected networks, whereas for directed network structures the triangle inequality principle is applied.

Parameter q represents the length of all walks computed over the semiring in the produced dissimilarity matrix, while r is a distance measure computed by the *Minkowski operation*.

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

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

galois, cscl, 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'
## Not run:
pfvn(arr)

## End(Not run)
```

---

pi.rels *π-Relations*

Description

A function to produce the π-relations of a partially ordered structure.

Usage

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

Arguments

x	An object of a "Ind.incl" or "Pacnet" class.
po.incl	(optional and logical) Include partial order in outcome?
vc	(optional) Vector with induced inclusions to compute.
po	(optional) Matrix with partial order associated to induced inclusions.

Details

The totality of π-relations are represented with an array format, and the arrangement of meet-complements in this array are congruences that serve for the decomposition of a partially ordered semigroup structure.

Induced inclusions to the partial order in x as a "Ind.incl" class object are produced by the fact function, while induced inclusions as a "Pacnet" class object comes from the outcome of a Pacnet log-report. With either case, π-relations are derived from the induced inclusions in a factorization process of a partially ordered semigroup.

A visualization of the π-relations partial order arrangement with function diagram is feasible provided that the resulted "Pi.rels" class object computes a "Pi.rels" class with the type option "pi.rels" in function partial.order. In such case, including the partial order with po.incl and a addition in po is a good idea since the partial order is the minimal element of the lattice of congruence relations.

Producing the partial order structure of a set of π-relations can be resource consuming and for exploring and other purposes, with `vc` option one can limit the amount of induced inclusions to compute.

Value

An object of the "`Pi.rels`" class with:

`pi`	π-relations, eventually with partial order
`at`	atoms
`mca`	meet-complements of atoms
`po`	partial order (if included)

References

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

See Also

`fact`, `decomp`, `partial.order`, `diagram`, `semigroup`, `pacnet`

`rbox` *Construct the Relation-Box of Multiple Network*

Description

Function to construct the Relation-Box of a multiple network.

Usage

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

Arguments

`w`	Array with three dimensions of stacked matrices with generating relations.
`transp`	(optional and logical) Include matrix transpose in w?
`smpl`	(optional and logical) Simplify strings of relations?
`k`	Length of Relation-Box (in z).
`tlbs`	(optional) Vector with labels for 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.

Values of `k` until 9 are supported.

Value

An object of the 'Rel.Box' class.

w	Primitive relations in Relation-Box
W	Structure of Relation-Box
lbs	Labels in relational system
Note	(optional) Notes indicating the particularities in the input
Orels	Original labels of relations
Srels	(optional) Simplified labels of relations
Trels	(optional) Transposed relation labels
k	Maximal length of the word
z	Length of the Relation-Box in the z dimension

Warning

May take a long time of computation with many types of relations and when the order of the multiplex network is high, and turning k to more than three.

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:
incubA |> getElement("IM") |>
  rbox()

## 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

Ucinet `dl` format is to represent multiple and two-mode network structures. It is used in **NetDraw**, which is a component of the **Ucinet** program. Besides multiple networks, this function can read two-mode structures as well but the '`EDGELIST`' option in `DL` is not yet supported for reading.

Value

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

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 of the data representing the network.
`as`	Data format for output: • `edgel` Edge list with send/receive/ties format • `array` Two- or three-dimensional array
`directed`	(logical) Is the graph directed?
`coords`	(optional and logical) Include coordenates in `gml` file?

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.

Node attributes can be retrieved as well when coordinates in `coords` are chosen.

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.

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, slbs, valued, row, col)
```

Arguments

x	Matrix or a three-dimensional array to be reduced.
clu	Vector with class membership in x.
lbs	(optional) Vector with labels for the reduced structure.
slbs	(optional) Vector with string labels in the reduced structure.
valued	(optional and logical) Should the reduction preserve valued data?
row	(optional and logical) Reduction by rows.
col	(optional and logical) 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 or affiliation networks with either row or col.

Notice that the "reduction" of a semigroup of x is made through a decomposition process that is computed with function decomp.

Value

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

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 of Network*

Description

Function to create a Relational System of a multiple 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	Array with three dimensions of stacked matrices with generating relations.
type	Is the transformation about: • tolist (Array of) matrices into lists of pairwise relations. • toarray Lists of pairwise relations into (array of) matrices.
bonds	the type of bonds to be used in the creation of the relational system • entire The "entire" network (default, same as full) • strong Strong bonds • weak Weak bonds • asym Asymmetric ties • recp Reciprocal ties • txch Tie exchange bundles • tent Tie entrainment bundles • mixd Mixed bundles • full Whole network (same as entire)
loops	(optional and logical) Include loops in relational system.
sel	(optional) Vector with set of actors to select.
att	(optional) Arrays in x corresponding to attributes.
sep	(optional) Pair separator for pairwise relations.

Details

A relational system of network x is a configuration with a particular pattern like the bond type among members. For instance, when the type of bonds chosen is entire then the nodes with ties are considered in the relational system without isolated nodes. Besides, strong bonds are relational bundles with a mutual character, whereas weak bonds are those patterns exclusively without mutual character.

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. This is possible when choosing from a list with actor attributes.

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.

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 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 system of strong bonds
arr |> rel.sys(bonds = "strong")

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

# select first node
arr |> rel.sys(sel = 1)
```

`rm.isol` *Remove Isolates from Network*

Description

Function to remove isolate nodes in simple and multiple networks.

Usage

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

Arguments

`x`	Matrix or array representing a network.
`diag`	(optional and logical, for arrays) Include diagonals in transformation?
`diag.incl`	(optional and logical, for arrays) Include diagonals in 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

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

See Also

`edgel`, `zbind`

Examples

```
# create 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` *Semigroup of Relations for 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`	Array with three dimensions of stacked matrices with generating relations.
`type`	Semigroup multiplication table to return: • `numerical` format (default) • `symbolic` format
`cmps`	(optional and logical) Include composite matrices in output?
`smpl`	(optional and logical) Simplify strings of relations?
`valued`	(optional and logical) Is the semigroup of 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. For other values 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, and depend on function `cscl`.

Value

A '`Semigroup`' class object with items:

`gens`	Array with generator relations
`cmps`	Array with the unique compound relations
`ord`	Dimension of the semigroup
`st`	Vector of unique string relations

S Multiplication matrix with semigroup of relations (numerical or symbolic)

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

Computing for the semigroup construction can take very long time for medium size or bigger sets (e.g. having more than 4 relation types).

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, strings, edgeT, wordT, cngr, cscl

Examples

```
# create 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 ) )

# optionally 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	A 'Signed' class object.
type	Semiring type: balance semiring (default) • cluster semiring
symclos	(logical) Apply symmetric closure?
transclos	(logical) Apply transitive closure?
k	Length of cycle or semicycle.
lbs	(optional) Labels for semiring to return.

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 the assessment by looking at semicycles in the system; otherwise, the evaluation is through closed paths.

Value

A 'Semiring' class object with items:

val	Valences in the semiring
s	Original semiring structure
Q	Resulted semiring structure
k	Number of cycles or semicycles

Warning

Disabling transitive closure should be made with good substantial reasons.

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 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 ) )

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

`signed` *Signed Network*

Description

Construct the signed network of a system made of contrasting relations.

Usage

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

Arguments

`P`	Array with positive ties and eventually with negative ties (see 'details').
`N`	(optional) Array with the negative ties.
`lbs`	(optional) Labels for returned 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

A '`Signed`' class object with items:

`val`	Valences in `s`
`s`	Signed matrix

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

References

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

See Also

semiring, as.signed

Examples

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

# make signed matrix of image matrix in dataset
incubA |> getElement("IM") |> signed()
```

---

strings *Strings of Relations*

Description

Function to obtain labels of unique relations, and generators and compound relations of a complete semigroup structure.

Usage

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

Arguments

x	Array with three dimensions of stacked matrices with generating relations.
equat	(logical) Include equations in output?
k	Length of strings in equations.
smpl	(optional and logical) Simplify string relations for output?
valued	(optional and logical) Valued format in strings?

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.

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

Value

An object of 'Strings' class.

wt	Generators and compound relations
ord	Order of structure
st	Labels of unique relations
equat	Equations among strings relations

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 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 ) )

# get string relations
strings(arr)
```

---

summaryBundles *Summary of Bundle Classes*

Description

Function for pretty printing of bundle class patterns results.

Usage

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

Arguments

x	A 'Rel.Bundles' class object.
file	(optional) Path where to place output file.
latex	(optional and logical) Latex format for output?
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.

Warning

Files with identical path and name to `file` will be overwritten.

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 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 ) )

# 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, valued,
       sep, ord, sort, sym, add, adc, rm.isol, 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) Labels in the transformation.
lb2lb	(optional and logical) Is the transformation "label-to-label"?
valued	(optional and logical) Treat x as valued?
sep	(optional) Pair separator used for the pairwise relations.
ord	(optional, for "toarray") Order of the transformed array 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*.
rm.isol	(optional and logical) Remove isolates in x?
na.rm	(optional and logical) Remove missing data or NAs?

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. Notice that the default of lb2lb is TRUE for "toarray", while is FALSE for "tolist".

Option type "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).

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

Value

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

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	Array with three dimensions of stacked matrices with generating relations.

Details

The Word Table is a consequence of the Edge Table (cf. edgeT) 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.

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).

Value

A 'WordTable' class object with items:

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

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 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 ) )

# obtain word table
wordT(arr)
```

`write.dat` *Write* `dat` *Files*

Description

A function to write `dat` files.

Usage

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

Arguments

`x`	Array representing a multiple network structure.
`path`	Path with filename for 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.

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

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` *Format Files*

Description

A function to write `dl` files representing multiple networks.

Usage

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

Arguments

`x`	Array or object representing a multiple network structure.
`file`	Path with filename for output.
`type`	Write data format type: • `nodelist` for *node-list* format • `fullmat` for *fullmat* format

Details

Files with `dl` format serve to represent multiple networks, and it is one used in **NetDraw**, which is a component of the **Ucinet** program.

Value

A file in path with the data with a `.dl` format.

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.edgel`, `write.dat`

write.edgel *Write edge-list files*

Description

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

Usage

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

Arguments

x	Array or object representing a multiple network structure.
file	(optional) Path with filename for output.
sep	(optional, default tabular) Separator between columns.
header	(logical) Include header in edge-list 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 data frame having an edge-list format in a file with path or an object.

See Also

edgel, , write.dl

write.gml *Write* gml *Files*

Description

A function to write files with a graph modelling language (gml) format.

Usage

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

Arguments

x	Array or object representing a multiple network structure.
file	(optional) Path with filename for output.

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 graph with modelling language format in a file with path or an object.

Warning

Files with identical path and name to file will be overwritten.

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 in output 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.

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

Value

Mostly a three dimensional array.

Warning

Arrays without dimnames are not fully supported.

See Also

mnplx, dichot, strings

Examples

```
# create two sets with two binary relations among three elements
arr1 <- round( replace( array( runif(18), c(3 ,3, 2) ), array( runif(18),
        c(3, 3, 2) ) > .9, 3 ) )

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

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

Index