Package 'multiplex'

June 30, 2022

Type Package

Version 2.9.9

Depends R (>= 3.5.0)

Imports methods

Suggests multigraph, Rgraphviz, knitr, rmarkdown

Title Algebraic Tools for the Analysis of Multiple Social Networks

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

Date 2022-06-30

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

Encoding latin1

License GPL-3

VignetteBuilder knitr

NeedsCompilation no

Date/Publication 2022-06-30 18:20:01 UTC

R topics documented:

multiplex-package . . . 3
as.semigroup . . . 5
as.signed . . . 6
as.strings . . . 7
bundle.census . . . 8
bundles . . . 9
cngr . . . 11
comps . . . 12
cph . . . 13
decomp . . . 15
diagram . . . 16
diagram.levels . . . 18
dichot . . . 19
edgeT . . . 20
expos . . . 21
fact . . . 22
fltr . . . 24
galois . . . 25
hierar . . . 26
incubs . . . 28
mlvl . . . 29
mnplx . . . 30
neighb . . . 31
pacnet . . . 33
partial.order . . . 34
perm . . . 35
pfvn . . . 36
pi.rels . . . 37
prev . . . 38
rbox . . . 40
read.dl . . . 41
read.gml . . . 42
read.srt . . . 43
reduc . . . 44
rel.sys . . . 45
rm.isol . . . 47
semigroup . . . 48
semiring . . . 50
signed . . . 52
strings . . . 53
summaryBundles . . . 54
transf . . . 56
wordT . . . 57
write.dat . . . 58
write.dl . . . 59
write.gml . . . 60

write.srt 61
zbind 62

Index **64**

`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** relies on functions to model the local role algebras of the network based on simple and compound relations existing in the system. It also provides a procedure for the construction and analysis of signed networks through the semiring structure, and it is possible to obtain the different relational patterns at the dyadic level in the system, which can serve for further analysis with diverse types of structural theories.

In conjunction with the **multigraph** package, it is possible to visualize multiplex, multimodal, and multilevel structures as graphs or valued graphs.

Details

Package:	multiplex
Type:	Package
Version:	2.9.9
Date:	30 June 2022
License:	GPL-3
LazyLoad:	yes

To work with **multiplex**, we typically start with a specific algebraic structure. A *semigroup* is a closed system made of a set of elements and an associative operation on it. The `semigroup` function constructs this algebraic structure, and it takes an array of (usually but not necessarily) multiple binary relations, which are the generator relations. The Word Table and the Edge Table serve to describe the semigroup completely, and they are constructed with the functions `wordT` and `edgeT`, respectively. The `strings` function gives unique relations of the complete semigroup and the `partial.order` function specifies the ordering of the string elements in the semigroup. For the visualization of the partial order structure, the function `diagram` produces the lattice of inclusions of a structure having ordered relations.

Different forms of decomposition that allow reducing semigroups such as factorization or finding congruence classes by substitution and the decomposition is based on congruence with the function `cngr` or π-relations of the unique strings given by `fact` or imported from *Pacnet*. In these two cases, `pi.rels`, and `decomp` will do this job for reducing either for an abstract or a partially ordered semigroup structure.

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. In this

case, for example, the network exposure of the actors is in the context of multiple networks, or else the resulted algebraic structures can embed the actor attributes.

In addition, it is possible to analyze structural balance in signed networks, which are built by function `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 of performing both balance and cluster semiring either with cycles or with semicycles.

Other capabilities of **multiplex** are not strictly algebraic. For instance, the `dichot` function 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 the function `zbind`.

The **multiplex** package creates a Relation-Box with the `rbox` function to implement the Partial Structural or Compositional Equivalence expressed in the cumulated person hierarchy of the system calculated via the `cph` function. It is from this structure that the partition of multiple networks is possible by counting the multiplicity of their ties.

Relational bundles are identified through the `bundles` function, which provides lists of pair relations. The `transf` function serves to transform such data into a matrix form. The enumeration of the different bundle classes is given by `bundle.census`. 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 two-mode networks.

Finally, multivariate network data can be created through the (s)end (r)eceive (t)ies format that can be loaded and transformed via the `read.srt` function. Other formats for multiple network data like *Ucinet* `dl` or *Visone* `gml` can be imported and exported as well with the **multiplex** package.

Author(s)

J. Antonio Rivero Ostoic

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

References

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

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

Lorrain, François and Harrison C. White, 'Structural Equivalence of Individuals in Social Networks.' *Journal of Mathematical Sociology*, 1, 49-80. 1971.

Boorman, Scott A. and Harrison 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 Multiple Social Networks with **multiplex**.' *Journal of Statistical Software*, 91(11), 1-41. <doi:10.18637/jss.v092.i11>

See Also

`multigraph`

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 (elay...)
semigroup(arr)

## and look at the strings
strings(arr)
```

`as.semigroup` *Coerce to 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, logical, and experimental) is 'x' an edge table?

Details

Since many of the functions 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

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)
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
as.signed(signed(incubA$IM)$s[1:2,1:2])
```

---

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) the labels of the 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	the word tables
ord	the 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.

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 ) )

## Calculate 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) the pair 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 nodes' location 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 can also be considered in the output. Long string labels can be 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
recp	reciprocal
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 an abstract semigroup object
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.5, 1 ) )
#
S <- semigroup(arr)

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

---

comps *Find components in multiple networks*

Description

Function to find different components in the multiple network including isolates

Usage

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

Arguments

x	array representing the network
bonds	the type of bonds to be used in the creation of the relational system for the different components

Details

The network's different components are obtained through the transitive closure of the bundle ties. By default, the "entire" system is chosen, but the option bonds allow discriminating different types of relational bundles for the components.

Value

A list with two possible "components"

com	a component
isol	the 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 components ans 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
rb <- rbox(incubA$IM)

## Calculate the cumulated person hierarchy
cph(rb)
```

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	whether the reduction is based on a a congruence class (option "cc") or rather on a π-relation ("pi"), atoms (option "at"), or a meet-complement of atoms (option "mca") in the 'Pi.rels' class
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 performs a reduction 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 the 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 a 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

---

diagram *Plot a Hasse Diagram of a set of ordered relations*

Description

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

Usage

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

ecol	(optional) the color of the edges
lty	(optional) the shape of the edges
lbs	(optional) labels of the elements in the partially ordered set
ffamily	(optional) the font family of the vertex labels
fstyle	(optional) the font style of the vertex labels with options: 'bold', 'italic', 'bolditalic'
fsize	(optional) the font size of the vertex labels
col.main	(optional) the color of the diagram's title
sep	(optional, only for 'lbs') string separator for equations
...	(optional) additional graphical items

Details

An example of ordered relations is found in the partial order table of relations, which is a product of the 'strings' option in the partial.order function. Another set of ordered relations comes from the table produced on Galois derivations in the mentioned function. In either case, this function plot either the partial order or a linear order diagram, depending on the results as Hasse diagrams.

When the partial order structure is from a decomposition process, string equations can be placed as labels with an equality separator.

Value

A Hasse diagram of the partial order relation.

Warning

This function requires that the *Rgraphviz* package is available.

Note

Roman numerals are used if the elements of the partial order are not labelled.

Author(s)

Antonio Rivero Ostoic

See Also

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

Examples

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

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

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

`diagram.levels` *Levels in the Lattice Diagram*

Description

This function 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 data frame with the elements of the partial order structure with the column names indicating the element class. If the permutation is specified, then 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 available. Besides, since the `pictex` function from `grDevices` is inside this routine, it implies counting with administrator privileges for running.

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 <- partial.order(as.strings(incubA$IM))

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

`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 can be any real number.

Value

Binary values of the input data.

Note

The labels are preserved after the dichotomization.

Author(s)

Antonio Rivero Ostoic

See Also

replace, prev, semigroup.

Examples

```
## Create the data: 2 binary relations among 3 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 2
dichot(arr, c = 2)
```

---

edgeT *Edge Table Generator*

Description

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: 2 binary relations among 3 elements
arr <- round( replace( array(runif(18), c(3,3,2)), array(runif(18),
       c(3,3,2))>.5, 1 ) )

## get 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 calculated 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 ) )

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

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

---

fact *Factorization 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	a semigroup object, S
P	a partial order structure associated to S
uniq	(logical) whether the factorization include the unique induced inclusions
fac	(integer) the 'factor' to be factorized (see details)
atoms	(logical) whether or not include the atoms in the output
mca	(logical) whether or not include the meet-complements of atoms in the output
atmc	(logical) whether or not include the atoms' meet-complements in the output
patm	(logical) whether or not include the potential atoms in the output
k	the length of the induced inclusion (only relevant if patm is activated)

Details

The factorization is part of the decomposition process for partially ordered semigroups, which means that there are two objects in the input. The induced inclusions are additions to the partial order and, depending on the needs; it is possible to customize the rest of the output. Atoms and meet-complement of these are useful for the decomposition through the decomp that produce a number of reduced structures or 'factors'. Argument fac allows choosing a factor for a progressive factorization.

Value

An object of "Ind.incl" class having:

po	the partial order table
iin	list of induced inclusions pairwise listed
niin	length of the induced inclusion
patm	(optional) a vector with the potential atoms
atm	(optional) a vector with the atoms
atmc	(optional) array with meet-complements of atoms
mca	(optional) meet-complements of atoms
note	(optional) induced inclusions without the substitution property

Author(s)

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

References

Pattison, P. and Bartlett, W., 'A factorization procedure for finite algebras,' *Journal of Mathematical Psychology* 25, 51-81. 1982.

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
S <- semigroup(arr)
# string relations
St <- strings(arr)
# partial order
P <- partial.order(St)

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

---

fltr *Principal filters*

Description

A function to find principal filters in a partial order

Usage

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

Arguments

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

Details

This function helps to find principal filters or principal 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 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 filter or 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 <- partial.order(galois(dfr),"galois")

## 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 with full or reduced labeling
sep	(optional) the pair separator 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)
```

---

hierar *Person and Relation Hierarchy*

Description

A 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) the actor of reference, either by its location in the adjacency matrix or by the label.
type	whether the hierarchy is for the 'persons' or for the 'relations' in the network with respect to 'x'

Details

The person hierarchy refers to the inclusion relations among the actors, whereas the relation hierarchy refers to the inclusion relations among the ties. 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: 2 binary relations among 3 elements
arr <- round( replace( array( runif(18), c(3 ,3, 2) ), array( runif(18),
c(3, 3, 2) ) > .5, 3 ) )

## The relation box
rarr <- rbox(arr, k=1)

## Calculated the person hierarchy of a random actor
hierar(rarr, ceiling(runif(1, min=0, max=dim(arr)[2])))
```

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 = 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 2 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 a multiple network as monoplex structure*

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 into a matrix representation with monoplex relations.

Value

A matrix of monoplex relations

Author(s)

Antonio Rivero Ostoic

See Also

zbind, dichot, reduc

Examples

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

## Make it monoplex
mnplx(arr)
```

---

neighb *Neighborhood of an 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, or else just the relational bundles having a mutual or an asymmetric character. 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 ) )

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

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

pacnet *Read Output from Pacnet*

Description

A function to read output files from the Pacnet program with the 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) the pair separator 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 string relations, Galois derivations, or 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, and obtain the partial order
data("incubA")

## the strings in the structure
st <- strings(incubA$IM)

## Get the partial order
partial.order(st)
```

`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) permut the array by sorting dimnames?

Details

This function serves to permutate an array representing relations according to a vector for the clustering membership. By activating the `sort` argument to `TRUE`, all other arguments will be ignored.

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))
perm(s, sort = TRUE)
```

---

pfvn *Pathfinder valued network 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*, John Wiley & Sons. 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`

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

## preview it
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")

## The relation box of the image matrices
## 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, read.srt, read.gml

---

read.gml *Read* gml *Files*

Description

A function to read files with the gml format.

Usage

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

Arguments

file	character vector containing a file name or path
as	should the data be given as a srt or with an array format?
directed	(logical) whether the graph is directed or undirected.
coords	(logical) whether the coordinates 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 to both 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 on the option chosen, the output is either a data frame or an array representing the multigraph. If the coordinates are chosen then these are part of the object structure, but they are not visible.

Note

If the coordinates are chosen, node attributes can also be retrieved.

Author(s)

Antonio Rivero Ostoic

References

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

See Also

write.gml, read.srt, read.dl

---

read.srt *Read edgelist (srt) files*

Description

A function to read edgelist files with a *s*end, *r*eceive, and *t*ies format to make a three-dimensional array.

Usage

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

Arguments

file	path to the file
header	(logical) does the file has a header?
sep	the separator among the columns (default is horizontal tab)
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

Details

srt stands for send, receive, and ties, and it is a data frame with at least 2 columns for the *s*end, *r*eceive, and the *t*ies for a multivariate network, with one column for each type of relation. However, the attr option corresponds to an actor and self-ties data frame file with the option to transform it into a diagonal matrix. When option 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 via e.g. the subset function with the toarray option.

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 given.

Note

The function supports valued networks as well.

Author(s)

Antonio Rivero Ostoic

See Also

write.srt, read.gml, read.dl, galois

---

reduc *Reduce a matrix or array*

Description

Function to reduce a matrix or 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
reduc(s, 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"), sel = NULL,
        loops = FALSE, 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 (array of) matrices into lists of pairwise relations or vice versa
bonds	the type of bonds to be used in the creation of the relational system (default the 'entire' network)
sel	(optional) the set of actors to be selected. For "toarray" att and noatt also supported (see details)
loops	(logical) whether or not the loops should be considered in the relational system
att	the arrays in x corresponding to attributes
sep	(optional) the pair separator 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 for the two options att or noatt.

Value

An object of 'Rel.System' class for the type = "tolist" (default) option. The items are:

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

For type = "toarray" the output is a dichotomous 2D or 3D 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 ) )

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

## the 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 isolated 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) if arrays, should the diagonals be included in the computation?
diag.incl	(optional and logical) if arrays, should the diagonals be included in the output?

Details

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

Value

Matrix or array that represents a multiplex network without the isolated actors.

Author(s)

Antonio Rivero Ostoic

See Also

read.srt, 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*

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	whether the semigroup should be returned with a numerical (default) or in a symbolic form?
cmps	(optional and logical) a logical to indicate whether the composite matrices should be also given in the output.
smpl	(logical and logical) whether to simplify or not the strings of relations.
valued	(logical) whether the semigroup should be with a valued format

Details

A multiplex network 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 be stacked into an object array 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 dichotomize the input data automatically; values higher or equal to a unit are converted to one; otherwise, they are set to zero. You can also use to `dichot` function to specify your 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`	an array with the generator relations
`cmps`	an array with the unique compound relations
`ord`	a number with the dimension of the semigroup
`st`	the strings, i.e. a vector of the unique relations
`S`	the semigroup of relations (see below)

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

Note

For medium-size or bigger sets (e.g. with more than 4 relation types), the semigroup construction can take a long time, and 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

`prev`, `strings`, `edgeT`, `wordT`, `dichot`, `cngr`.

Examples

```
## Create the data: 2 binary relations among 3 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
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 or 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*. New York: John Wiley & Sons. 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
sg <- signed(arr)

## Establish the semiring structure
semiring(sg)
```

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	the valences in the signed matrix
s	the 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; that is 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: 2 binary relations among 3 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) the path where the output file is to be placed
latex	(logical) whether the output should be in latex format or not
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: 2 binary relations among 3 elements
arr <- round( replace( array( runif(18), c(3 ,3, 2) ), array( runif(18),
c(3, 3, 2) ) > .8, 3 ) )

## Establish the different bundles
bd <- bundles(arr)

## Print the different relational bundles
summaryBundles(bd)
```

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

Description

Function to transform data from/to matrix/list formats 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	whether the transformation is from a list of pair relations to an array format ("toarray"), from a matrix to a list of pair relations ("tolist"), from a list of pair relations to a square array ("toarray2"), or else from a matrix or array into an edge list ("toedgel").
lbs	(optional) the labels in the transformation (disabled for "toarray")
lb2lb	(optional and logical) whether the transformation is label-to-label. Default TRUE for "toarray" and FALSE for "tolist"
sep	(optional) the pair separator for the pairwise relations
ord	(optional) the order of the resulted structure ("toarray" option, otherwise ignored)
sort	(optional and logical) sort the arrays in the output?
sym	(optional and logical) symmetrize the arrays? ("toarray" option, otherwise ignored)
add	(optional) add elements in the array's "domain"
adc	(optional) add elements in the array's "codomain"
na.rm	(optional and logical) remove NAs?

Details

The option "tolist" is to transform a matrix or an array into a list of pair elements. In case 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 now supported.

Data frames are also accepted for the "tolist" option, and in case such information is given as a list of pair relations, the output will be a square matrix. By deactivating the option na.rm, it is possible to retain missing data in the data frame for the transformation even if both sender and receiver labels are not available.

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

`read.srt`, `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
transf(s, lb2lb = TRUE, lbs = c('n','m','ñ'))
```

---

wordT *The Word Table of Relations*

Description

The Word Table of multiple 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.

Examples

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

## get the 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 the multiple network structure
path	the path file for the output

Details

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

Value

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

Note

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

Author(s)

Antonio Rivero Ostoic

References

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

http://www.gmw.rug.nl/~stocnet/

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	whether to write the data as a nodelist or as a 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.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

---

write.srt *Write* srt *Files*

Description

A function to write srt files

Usage

```
write.srt(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

srt stands for *s*end, *r*eceive, and *t*ies, and it is a data frame with at least 3 columns for the sender, receiver, and the ties, one column for each type of relation.

Value

A file with the data with a .srt format

Author(s)

Antonio Rivero Ostoic

See Also

read.srt, , write.dl

`zbind` *Combine multidimensional arrays.*

Description

A function to combine arrays with equal or unequal dimensions.

Usage

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

Arguments

`...`	arrays to bind
`sort`	(optional and logical) sort dimnames in output array?
`force`	(optional and logical) force bind for unequal dimensions arrays?

Details

This function is for stacking two- or three-dimensional arrays into a single object to represent a multivariate system structure. Both square and rectangular arrays are supported provided that the dimensions in the input are equal, but data frames need to be transformed into arrays. The `dimnames` in the output correspond to the first array in the input, and a `Warning` message is given when these labels are `NULL`.

Activate `force` when two or more arrays have different dimensions to generate a three-dimensional array that includes all elements. This is the default option that is needed for working with temporal or dynamic networks with changing size and order.

Value

Usually a three-dimensional array representing a multiplex network.

Note

This routine is an extension of both `cbind` and `rbind` functions.

Author(s)

Antonio Rivero Ostoic

See Also

`mnplx`, `dichot`, `strings`, `cbind`

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

∗ **IO**
multiplex-package, 3
pacnet, 33
read.dl, 41
read.gml, 42
read.srt, 43
summaryBundles, 54
write.dat, 58
write.dl, 59
write.gml, 60
write.srt, 61
∗ **algebra**
as.semigroup, 5
as.strings, 7
cngr, 11
cph, 13
decomp, 15
diagram.levels, 18
edgeT, 20
fact, 22
fltr, 24
galois, 25
hierar, 26
partial.order, 34
perm, 35
pi.rels, 37
prev, 38
rbox, 40
semigroup, 48
semiring, 50
strings, 53
wordT, 57
∗ **array**
as.signed, 6
as.strings, 7
cph, 13
decomp, 15
mnplx, 30
perm, 35
rbox, 40
reduc, 44
rm.isol, 47
signed, 52
strings, 53
transf, 56
zbind, 62
∗ **attribute**
bundles, 9
expos, 21
multiplex-package, 3
rel.sys, 45
∗ **cluster**
cngr, 11
comps, 12
decomp, 15
perm, 35
reduc, 44
∗ **datasets**
incubs, 28
∗ **data**
bundle.census, 8
bundles, 9
mlvl, 29
multiplex-package, 3
pacnet, 33
read.dl, 41
read.gml, 42
read.srt, 43
write.dat, 58
write.dl, 59
write.gml, 60
write.srt, 61
∗ **file**
multiplex-package, 3
pacnet, 33
read.dl, 41
read.gml, 42
write.dat, 58

write.dl, 59
write.gml, 60
write.srt, 61
∗ **graphs**
diagram, 16
∗ **list**
bundles, 9
∗ **manip**
diagram.levels, 18
dichot, 19
mnplx, 30
multiplex-package, 3
neighb, 31
perm, 35
pfvn, 36
read.srt, 43
reduc, 44
rm.isol, 47
transf, 56
zbind, 62
∗ **math**
as.semigroup, 5
bundles, 9
cngr, 11
cph, 13
decomp, 15
fact, 22
fltr, 24
galois, 25
hierar, 26
partial.order, 34
perm, 35
prev, 38
semigroup, 48
semiring, 50
∗ **models**
as.signed, 6
comps, 12
expos, 21
mlvl, 29
multiplex-package, 3
neighb, 31
pfvn, 36
rel.sys, 45
semiring, 50
signed, 52
∗ **print**
summaryBundles, 54

as.semigroup, 5
as.signed, 6, *51, 52*
as.strings, 7, *17*, *35*

bundle.census, 8, *10, 11*, *55*
bundles, *9*, 9, *13*, *22*, *32*, *47*, *55*, *57*

cbind, *62*
ccgraph, *57*
cngr, 11, *16*, *24*, *34*, *45*, *49*
comps, 12
cph, 13, *27, 28*, *36*, *41*

decomp, *12*, 15, *23, 24*, *34*, *38*, *45*
diagram, *14*, 16, *18*, *25–27*, *35*
diagram.levels, *17*, 18
dichot, 19, *31*, *49*, *62*

edgeT, 20, *39*, *49*, *58*
expos, 21, *32*, *47*

fact, *12*, *15, 16*, 22
fltr, 24, *26*, *35*

galois, *17*, *25*, 25, *34, 35*, *44*

hierar, 26, *41*

incA *(*incubs*)*, 28
incB *(*incubs*)*, 28
incC *(*incubs*)*, 28
incD *(*incubs*)*, 28
incubA *(*incubs*)*, 28
incubB *(*incubs*)*, 28
incubC *(*incubs*)*, 28
incubD *(*incubs*)*, 28
incubs, 28

mlgraph, *30*
mlvl, 29
mnplx, 30, *62*
multigraph, *4*, *28*, *30*, *37*
multiplex-package, 3

neighb, *22*, 31, *47*

pacnet, *12*, *24*, 33, *38*, *59*
partial.order, *8*, *16–18*, *25–27*, 34, *36*, *54*
perm, *18*, *35*, 35
pfvn, 36
pi.rels, *16*, *34*, 37

prev, *20*, 38, *49*

rbox, *14*, *27*, 40, *45*
read.dl, 41, *43, 44*, *60*
read.gml, *42*, 42, *44*, *61*
read.srt, *42, 43*, 43, *48*, *57*, *61*
reduc, *16*, *31*, 44, *57*
rel.sys, *13*, *22*, *32*, 45, *57*
replace, *19, 20*
rm.isol, 47

semigroup, *5, 6*, *14*, *16*, *20, 21*, *34*, *39*, *41*, 48, *54*, *58*
semiring, *7*, 50, *52*
signed, *7*, *51*, 52
strings, *8*, *17*, *34, 35*, *49*, 53, *57, 58*, *62*
subset, *44*
summaryBundles, *9*, *11*, 54

transf, *11*, 56

wordT, *21*, *49*, 57
write.dat, *34*, 58, *60, 61*
write.dl, *42*, *59*, 59, *61*
write.gml, *43*, *59, 60*, 60
write.srt, *44*, *60*, 61

zbind, *8*, *31*, *48*, 62